Laboratório 2 - Jogo do tempo de resposta

Aluno: Lucas Silva Beneti

Para o Laboratório 2 foi requisitado o estudo e desenvolvimento de um simples jogo de tempo de resposta do jogador após um Led específico ser aceso.

Pergunta a ser respondida:

O exemplo original do blinky usa um loop para gerar a temporização da piscada do LED. Porque a variável ui32Loop é declarada como volatile dentro de main() ?

Tem relação com a otimização que o compilador realiza, pois sem o *volatile,* ele considera que, como não ocorre nada nos loops que utilizam a ui32Loop, pode basicamente ignorar toda essa operação. Com o *volatile*, você basicamente fala pro compilador que não é pra utilizar de otimização ao alocar a variável, o que nesse caso faz com que faça os *for loops* funcionar como é esperado nesse método de *pooling*.

Planejamento do jogo de tempo de resposta

1. Definição do problema

O princípio do jogo é bem simples, quando o LED 1 acender, o jogador deverá responder o mais rápido possível. Ao apertar o botão, será mostrado para o jogador o número de ciclos de clock que demorou para a resposta dele.

2. Levantamento de requisitos

O projeto tem os seguintes requisitos:

Funcionais:

RF1 - O jogo deve ligar o LED D1 para informar o jogador do início da contagem de tempo.

RF1.1 - o LED deve ser aceso até 1 segundo após o início da operação da placa.

RF2 - O jogo usa o botão SW1 para a entrada de dados pelo usuário.

RF3 - O jogo deve apresentar a contagem de tempo no Terminal

Não funcionais:

RFN 1 - O limite superior de tempo será de 3s (passado esse tempo, o jogador perde automaticamente).

3. Plataforma de Hardware

A plataforma determinada:

- Placa Tiva EK-TM4C1294XL
- Utiliza o processador TM4C1294NCPDT

Na placa, deverão ser usados no mínimo o LED 1 (PN0) e Switch 1 (PJ0). Não há restrições em relação ao clock que deverá ser utilizado.

4. Plataforma de Software

Para a interface com a placa, será utilizada a biblioteca TivaWare, junto com algumas configurações iniciais necessárias para a placa Tiva. Para a configuração da placa no IAR, deverá ser usado o arquivo startup_ewarm.c que contém a configuração do vetor de interrupções.

5. Estudo da plataforma de Software

Estudando a biblioteca TivaWare, foi possível notar que existem funções de acesso direto ao SysTick do processador e para configuração do mesmo , como possibilidade de utilização de um período arbitrário.

Foram encontradas também as funções para lidar com os GPIO, que englobam tanto o LED quanto o *switch* que deverá ser utilizado pelo jogador. Essas funções podem ser utilizadas para *setup* desses periféricos como entrada ou saída de dados e registrar *handlers* para interrupções nesses GPIOs se for preciso.

6. Design da Solução

- Configurar o clock da placa para 120 MHz
- Configurar pino GPIO PJ0 e o LED PN1
- Configurar SysTick com período adequado para a marcação de milisegundos
- Habilitar interrupções para o SysTick
- Habilitar interrupções para o botão PJ0
- Programar interrupção de contador para o Systick
- Programar interrupção para o botão PJ0
- Programar o LED PN1 para acender após 1 segundo
- Inicializar a contagem do Systick
- Caso ocorra interrupção feita pelo PJ0 (SW1), Systick deve ser desabilitado e o número de ciclos contabilizados deve ser obtido
- Mostrar no terminal uma mensagem de conclusão da tentativa e o número de ciclos que o jogador demorou para responder
- Caso o jogador demore mais que 3 segundos, mostrar mensagem no terminal falando que ele perdeu e caso queira, recomeçar o jogo
- Para recomeçar o jogo, deve-se resetar a placa

Conclusões

Durante o desenvolvimento desse jogo, por mais simples que seja, foi interessante ver que o quanto a biblioteca TivaWare ajuda em relação a acessar os periféricos e até o SysTick que foi um temporizador muito importante para esse projeto.