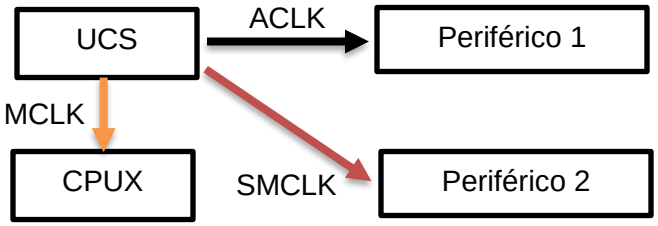
Experimento 6 - Timers

OBJETIVO:
Praticar o uso dos módulos de clock, timers.

PRÉ-RELATÓRIO
Trazer de casa a tabela de linearização da resposta de luminosidade do LED para controle da PWM do programa 17b. Em outras palavras, trazer uma tabela com os N estágios de um sinal PWM que gera uma resposta luminosa exponencial percebida de maneira linear pelo olho humano.

DADOS:

Temporizadores, ou simplesmente timers em inglês, são peças fundamentais em sistemas embarcados. Timers permitem que uma referência de tempo seja usada em aplicações que demandam sincronismo ou precisão temporal. A linguagem de programação C não possui a noção de tempo. Cada linha de código é executada sequencialmente, uma após a outra, e não há nada na linguagem que indique a velocidade ou tempo de execução do programa. A única forma de sabermos o tempo que leva para cada instrução executar é conhecendo o hardware em que o programa está rodando e principalmente a velocidade do clock que bate e dá o ritmo de execução das instruções. O MSP430 possui um módulo específico para a configuração dos clocks que são distribuídos no sistema.



O dispositivo que controla a distribuição dos clocks dentro do MSP430 é chamado de Unified Clock System (UCS). Nele podemos configurar as fontes de clock e até a velocidade de operação de cada clock. Cinco fontes de clock estão disponíveis no sistema:

- Cristal 1 – 32768Hz
- Cristal 2 – 4MHz
- VLO – 10kHz ± 40%
- REFO – 32768kHz ± 3,5%
- DCO – Configurável e gerado a partir dos cristais ou do REFOCLK.

O DCO (Digitally Controlled Oscillator) é um gerador de clock digital que pode gerar frequências maiores que as referências supracitadas e ainda assim garantir a sincronia com a referência. Isso é feito através da FLL (Frequency Locked Loop) seguindo a fórmula abaixo:

$$F_{DCO} = FLLD \times (FLLN + 1) \times F_{ref}$$

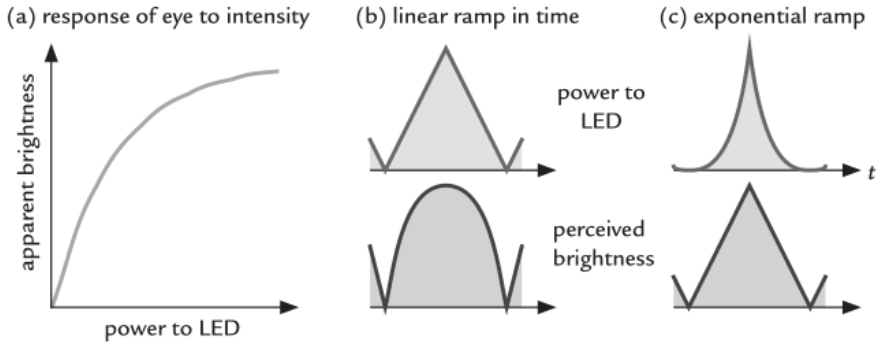
Onde F_{ref} é a frequência da entrada da referência selecionada.

PEDIDOS:

Programa 16:
Escreva um programa em C que configure o clock do processador (MCLK) para rodar em 1MHz e o clock do timer (SMCLK) para rodar em aproximadamente 400kHz. Use essa referência de tempo (400 kHz) para fazer piscar o LED verde (P4.7) em exatamente 1Hz (0,5s apagado e 0,5s aceso). Use a técnica de amostragem da flag de overflow (TAIFG) do timer para saber quando o timer atingiu o valor máximo.

Programa 17:
Use o timer B para gerar um sinal PWM por software com período de 1/50 = 20ms (frequência de 50Hz) que controle a luminosidade do LED verde. Controle a luminosidade deste led através dos botões S1 e S2. Os valores possíveis do ciclo de carga vão desde 0% até 100%, em 20 passos de 5%. Use S1 para diminuir o ciclo de carga (a largura do pulso) PWM em 5% e S2 para aumentar o ciclo de carga (a largura do pulso) em 5%. Opcionalmente, escreva no LCD, em porcentagem, o valor instantâneo do PWM.

Programa 18:
Você irá perceber no programa anterior que a luminosidade do LED varia mais para valores pequenos do ciclo de carga e que quase não varia valores próximos de 100%. Isso acontece porque a percepção de luminosidade do olho humano não é linear, mas segue uma relação logarítmica em relação à potência luminosa produzida pelo LED.



Aproveite o que foi feito no programa 17 e faça o LED pulsar a uma frequência de exatamente 1Hz, ou seja, ir de 0% a 100% em 0,5s e voltar de 100% a 0% em 0,5s. Remova o controle de luminosidade através dos botões e use uma tabela para mapear a largura do pulso à luminosidade correta do LED. A onda PWM gerada deve ter frequência de 50Hz.

Você deve gerar em casa a tabela da resposta do LED em função da luminosidade percebida. Para isso, assuma que a luminosidade segue a seguinte expressão:

$$f(x) = \left\lceil \frac{(k^x - 1)}{k - 1} \cdot MAX \right\rceil$$

Onde MAX é o valor máximo do timer (CCR0), k é uma constante que regula a curvatura da exponencial, x é a entrada da função normalizada (ou seja, assume valores entre 0 e 1) que representa a porcentagem da largura do pulso (de 0% a 100%) e o operador $\lceil . \rceil$ retorna o maior valor inteiro, ou seja, arredonda qualquer valor decimal para cima. Para este experimento é recomendado assumir $k = 2$.

O número de passos da PWM é determinado pelo número de divisões que a entrada x assume. A ideia da expressão acima é transformar um espaçamento linear entre valores de 0 a 1 num conjunto de valores espaçados exponencialmente entre 0 e MAX. Use 25 passos nessa escala para varrer o sinal PWM de 0 a 100%.

DICA: Você pode gerar o vetor de entrada x no matlab/octave com N divisões usando o comando *linspace(0,1,N)*

Para este programa, espera-se que o sinal PWM seja gerada por hardware e não por software como foi o programa 17. Neste caso recomenda-se conectar a saída do canal 1 do timer no pino P4.7. Você pode fazer isso remapeando a funcionalidade do pino da seguinte forma:

```
// Port Mapping
PMAPKEYID = 0x02D52;          // Escritas no modulo de mapeamento são protegidas
                              // por senha. Escrever 2D52 no registro PMAPKEYID
                              // permite reconfiguração de apenas uma porta.
P4MAP7 = PM_TB0CCR1A;         // Conecta o pino P4.7 na saída do canal 1
                              // do Timer B.
```

Programa 19:
Utilizando as rotinas desenvolvidas no experimento passado, desenvolva um cronometro digital. Mostre na primeira linha do LCD, os valores de minutos e segundos que passaram desde que o botão S1 foi pressionado. Ao apertar o mesmo botão, o contador deve parar. Se apertado novamente o contador deve continuar de onde parou.

O botão S2 tem dupla finalidade. Se o cronometro estiver parado, o botão S2 zera o estado atual de contagem. Se o cronometro estiver ativo e o botão S2 for pressionado, o cronometro deve salvar o valor do contador no momento que o usuário pressionar o botão. Essa informação deve aparecer na segunda linha do LCD. Além disso, o contador principal deve recomeçar do zero e continuar contando. Chamaremos essa funcionalidade de Lap.

- ⇨ Resumindo, o botão S1 terá a funcionalidade de Start/Stop e o botão S2 terá a funcionalidade de Reset/Lap.

Use precisão de 0,1 segundo no seu cronometro. Mostre no display o tempo no seguinte formato:

HH:MM:SS,d

RELATÓRIO

O relatório é individual, e deve ser entregue impresso (ou feito à mão). Em hipótese alguma será admitida a entrega do relatório de forma eletrônica.

Questão 1 (x pontos)

Como comprovar que a CPU está realmente rodando com MCLK em 1 MHz?

Questão 2 (x pontos)

Explique a sua escolha das referências de clock e como você fez para gerar duas referências, uma a aproximadamente 400kHz e outra a exatamente 1MHz.

Questão 3 (x pontos)

Qual é a diferença fundamental entre o timer A e o timer B? Como isso ajuda a evitar que a PWM tenha espúrios repentinos (visíveis quando a luminosidade fica cintilando)

Questão 4 (x pontos)

Explique o modo de saída que você escolheu para gerar a PWM. Se fosse necessário gerar uma onda PWM alinhada pelo centro, como o exemplo abaixo, o que seria necessário alterar no programa 18?