Experimento 7 – Comunicação Serial

OBJETIVO:

Este roteiro tem como objetivo revisar conceitos de comunicação serial e treinar a configuração do módulo dedicado do MSP430. Em especial, iremos estudar o funcionamento dos protocolos UART e I2C.

INTRODUÇÃO:

Comunicação serial é o processo de envio de dados de maneira sequencial, um bit após o outro através de uma linha de dados. Em sistemas embarcados, dispositivos com funcionalidade específica geralmente implementam algum tipo de comunicação serial para enviar ou receber dados e instruções. Dispositivos como acelerômetros, cartões de memória, GPS, sensores de temperatura, umidade e pressão, todos possuem internamente um microcontrolador que trata o dado bruto e o envia por uma linha serial. Diante disso, é de suma importância compreender as diferentes formas de comunicação serial para interfacear o microcontrolador com outros dispositivos.

Duas formas de comunicação serial serão estudados neste roteiro: Um empregando o UART (comunicação assíncrona) e outra usando o protocolo I2C (Síncrona)

A comunicação serial assíncrona com um dispositivo UART (Universal Asynchronous Receiver/Transmitter) é provavelmente a forma serial mais simples de todas. Sua popularidade vem justamente da sua simplicidade e facilidade de implementar usando lógica digital ou microcontroladores. Ele usa apenas 1 fio unidirecional para transmitir dados de uma ponta a outra.

A linha está ociosa quando em nível alto. Para iniciar um envio, é necessário puxar a linha para nível baixo e transmitir bit a bit sequencialmente, começando pelo bit menos significativo. A taxa de transferência é um acordo prévio, ou seja, ambas as partes devem estar configuradas para se comunicar na mesma velocidade.

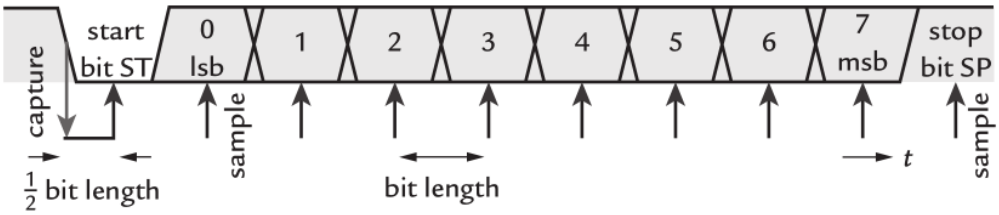

Figura 1: Protocolo UART

A UART não permite comunicação bidirecional através de um único fio. Não é possível conectar mais de dois dispositivos no barramento senão com o uso de fios em paralelo. Uma particularidade desta forma de comunicação é a ausência de sincronia entre o transmissor e receptor. Isto, ao mesmo tempo em que traz simplicidade, pode levar a erros de comunicação caso haja diferença significativa entre os clocks internos de cada dispositivo..

O protocolo I2C (Inter-Integrated Circuit) propõe um mecanismo de comunicação que resolve os problemas mencionados acima. Ele permite que vários dispositivos se conectem a um mesmo barramento e se comuniquem entre si. O barramento utiliza apenas dois fios para realizar uma comunicação bidirecional entre diversos dispositivos. Na terminologia utilizada, o dispositivo que está gerando uma mensagem é chamado de transmissor, enquanto o dispositivo que está recebendo a mensagem é chamado de receptor. O dispositivo que controla o barramento é chamado de mestre (master), e os dispositivos que respondem ao mestre são chamados de escravos (slave). Os dispositivos mestre e escravo assumem o papel de transmissor ou receptor em diferentes etapas da comunicação. O fio do barramento que carrega dados é chamado de SDA (**S**erial **DA**ta). O outro fio que estabelece o ritmo de comunicação é chamado de SCL (**S**erial **CL**ock). Apenas o mestre controla o clock, mesmo quando estiver recebendo uma mensagem (como veremos abaixo).

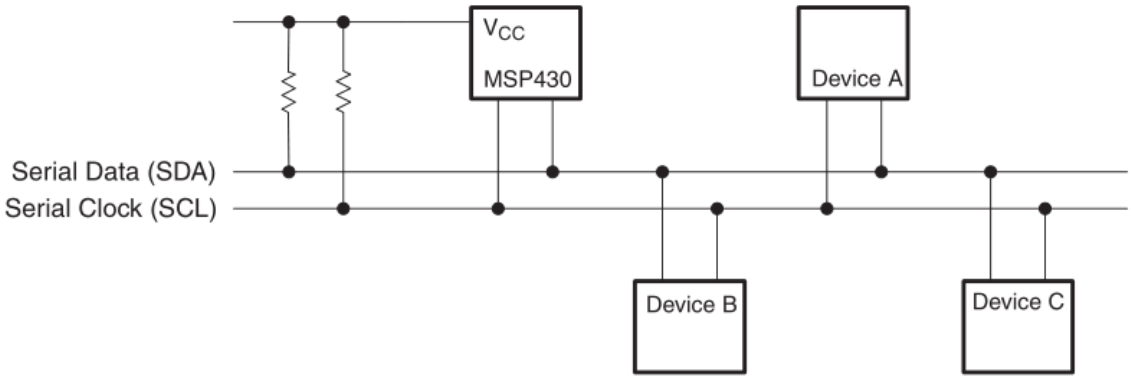

Figura 2 – Barramento I2C

Sinais de controle: Start e Stop

Quando o barramento está ocioso SDA e SCL ficam em nível lógico HIGH. Isso é garantido através de resistores de pull-up como o da figura 2. Para iniciar uma transmissão, o mestre força a descida de SDA (mudança de HIGH para LOW) enquanto o clock está em nível alto. Este sinal é a condição de partida (START), e prepara os dispositivos para a comunicação.

Para finalizar a transmissão, o mestre libera SDA (mudança de LOW para HIGH) quando o clock está em nível alto. Este sinal é a condição de parada (STOP) liberando assim o barramento para outras comunicações. Se for necessário reestabelecer a comunicação, deve-se gerar novamente a condição de partida descrita no parágrafo anterior.

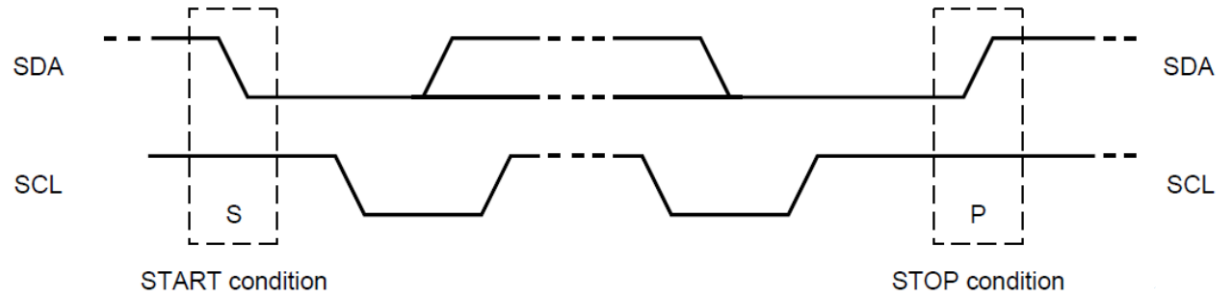

Figura 3 – Comandos de Start e Stop

Transferência de 1 bit

Na transferência de um bit, o dado em SDA deve permanecer estável entre a subida e a descida do clock (SCL). Mudanças em SDA quando SCL está em nível alto são entendidas como sinais de controle (como visto nas condições de START e STOP). Durante a transmissão, o valor de SDA é constante enquanto SCL estiver em nível HIGH e assume o valor do bit a ser transmitido.

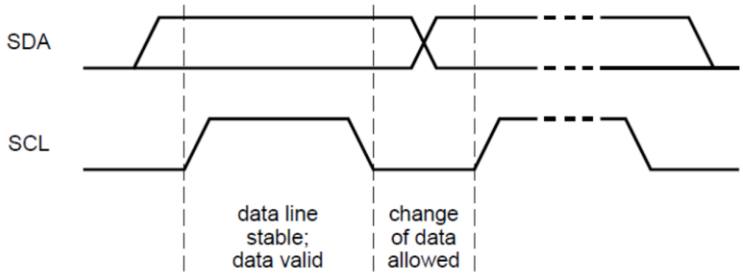

Figura 4 – Transferência de 1 bit

Transferência de uma palavra (8-bits)

A comunicação no barramento é feita em bytes (8 bits). A cada byte transmitido, um sinal de acknowledge deve ser gerado. Portanto, para cada byte transmitido, 9 batidas de clock precisam ser geradas. O número de bytes enviados entre um sinal de START e STOP é ilimitado. A figura 4 mostra a comunicação entre dois dispositivos. O transmissor envia os 8 primeiros bits e libera a linha no nono bit. O receptor então puxa a linha para LOW indicando que entendeu a mensagem (acknowledge). Se a linha ficar em HIGH na batida do nono bit, isso significa um "not-acknowledge" que pode ter diversos significados (pode ser que o escravo não entendeu a palavra enviada; ou que não há nenhum dispositivo na linha escutando; ou que o escravo decidiu enviar um NACK ("not-acknowledge") de propósito para indicar que não tem mais nada para enviar). Na figura, as ações do transmissor e receptor são apresentadas separadas, porém ambos compartilham a mesma linha SDA. Isso deixa claro que é o receptor (e não o transmissor) que envia o acknowledge no momento certo. Note que o clock é controlado pelo mestre, inclusive na batida do acknowledge.

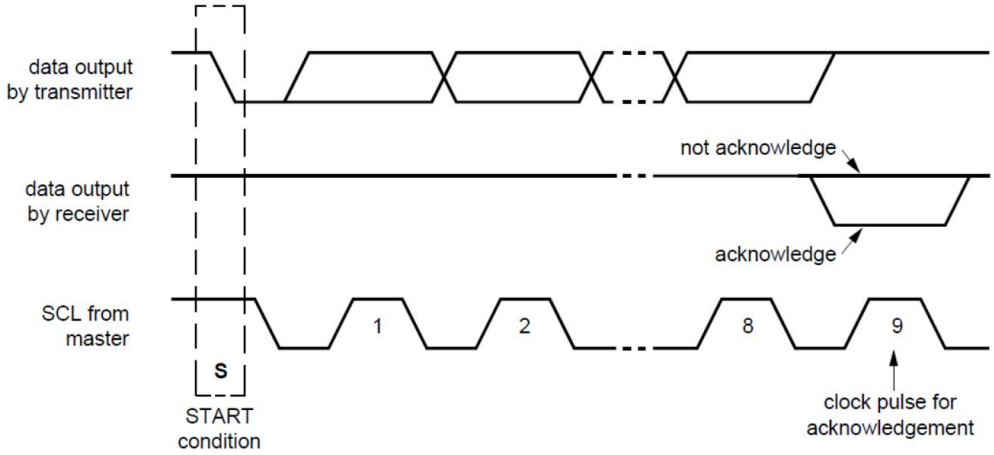

Figura 5 – Transferência de 1 byte

Endereçamento

A primeira palavra enviada no barramento é especial e é interpretada como endereço de destino. Além dos 7-bits de endereço, a primeira palavra também contém um bit de leitura/escrita que indica se o mestre vai ler ou escrever num dispositivo remoto.

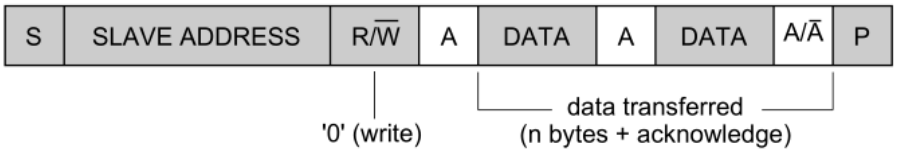

Figura 6 – Endereçamento

Após o envio dessa palavra inicial, as próximas palavras são sempre dados de 8 bits enviados para ou recebidos do escravo. Por fim, termina-se a comunicação com um sinal de parada (P).

PEDIDOS:

Programa 20:
O MSP430 possui um módulo de comunicação serial dedicado. Isso torna o processo de comunicação tão simples quanto escrever num registro e esperar que o byte seja enviado pela linha serial. Isso nem sempre é o caso de microcontroladores mais simples. Nesses casos o protocolo deve ser implementado manualmente através de escritas nas portas do dispositivo. Esse tipo de implementação exige cuidado extra com a temporização para que a linha não perca a sincronia.

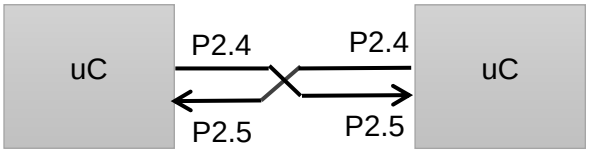

Figura 7 – Comunicação UART

Utilize o pino P2.4 para enviar dados e P2.5 para receber dados. A ligação entre dois microcontroladores deve ser feita conforme a figura 7. Configure essa interface para se comunicar em 9600 de baud rate. A recepção dos dados deve ser feita no centro de cada bit, conforme a figura 1, para evitar erros de recepção caso os clocks dos dois dispositivos estejam ligeiramente desalinhados. Implemente essa comunicação usando apenas escritas e leituras das portas de entrada e saída.

O receptor responder às seguintes palavras

- 0x01 – Acende o LED vermelho
- 0x0A – **Apaga** o LED vermelho
- 0x10 – Acende o LED verde
- 0xA0 – **Apaga** o LED verde

O transmissor deve enviar apenas uma palavra a cada pressionar de botão:

- S1 – Envia **0x01** e **0x0A** alternadamente
- S2 – Envia **0x10** e **0xA0** alternadamente

Exemplo: Ao pressionar uma vez o botão S1 o transmissor deve enviar a palavra **0x01**. Ao pressionar uma segunda vez, o transmissor envia **0x0A**.

Programa 21:
Refaça o programa 20, mas agora usando o módulo dedicado de comunicação serial USCI (Universal Serial Communication Interface). Use as interrupções das portas para enviar dados e as interrupções da USCI para receber dados e controlar os LEDs.

OBS: O periférico de comunicação serial UART USCA0 está roteado nos pinos P3.3 e P3.4. É necessário configurar esses pinos para sua funcionalidade dedicada.

Programa 22:
Desenvolva um scanner de barramento I2C. Seu programa deve testar todos os endereços possíveis de um barramento I2C e salvar num vetor (ou mostrar no LCD) os endereços que responderam com um acknowledge. Conecte o seu scanner no barramento I2C oferecido pelo professor (onde haverá um escravo com um endereço desconhecido) e descubra qual é o endereço desse escravo.

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 (2,5 pontos)
Quais são as limitações da comunicação com um UART? Explique como que o desvio de frequência entre o transmissor e receptor pode afetar a comunicação.

Questão 2 (2,5 pontos)
Explique como funciona a recepção de um sinal serial utilizando voto de maioria.

Questão 3 (2,5 pontos)
Desenhe um barramento I2C com 3 dispositivos. Um mestre e dois escravos.

Questão 4 (2,5 pontos)
Suponha que um escravo esteja configurado no endereço 0x45 do barramento I2C e que ele possui dois valores possíveis de leitura. Um no endereço interno 0xF3 e outro no endereço interno 0xEE. Explique como que um mestre requisitaria a leitura do endereço 0xF3 desse escravo.