Projeto de Sistemas RF

Aula 09 - Transmissão de dados através do protocolo SPI

Apresentação

Na aula de hoje, vamos estudar como realizar a transmissão de dados utilizando o protocolo SPI, na prática. Vamos ver o que é o protocolo SPI, como ele funciona e como configurar o microcontrolador para se comunicar com o SPI. Veremos também como fazer um programa que envie e um que receba dados pela SPI e, além disso, vamos ver os circuitos que você vai montar no laboratório do Metrópole Digital, durante a prática da aula. Parece difícil? Não tanto. Vamos só colocar em prática o que vimos nas aulas anteriores, mas aplicado a um novo protocolo de comunicação. Prontos? Então, vamos começar.

Objetivos

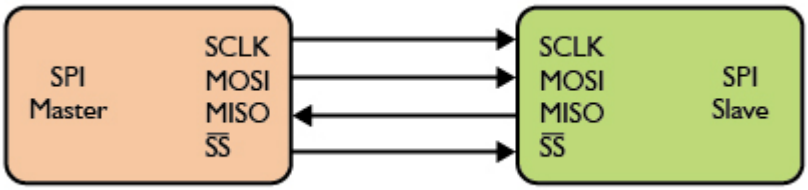
Ao final desta aula, você deverá ser capaz de:

- Desenvolver um código para um PIC que envie um dado pelo protocolo SPI.
- Desenvolver um código para um PIC que receba um dado pelo protocolo SPI.
- Aprender a usar as ferramentas de compilação do código e gravação de PIC.
- Realizar uma experiência prática de montagem de um circuito que realize a comunicação sobre o protocolo SPI.

O que é SPI?

Você deve estar se perguntando o quê significa SPI. Essa sigla quer dizer "***S****erial* ***P****eripheral* ***I****nterface Bus"*, e se pronuncia *"esse-pi-ai"* ou como a palavra *"spy"*, em inglês. Corresponde a um padrão de comunicação serial entre microcontroladores que se comunicam em uma rede. Nos PICs (do inglês *Programmable Intelligent Computer*), ele é implementado como sendo um modo de operação do módulo MSSP (*Master Synchronous Serial Port*). Ou seja, pra realizar uma comunicação pelo SPI, você precisa configurar corretamente esse módulo do seu microcontrolador e fazer as conexões corretas. Mas antes de passar para as configurações, vamos entender, primeiramente, como o SPI funciona. Dê uma olhada na figura abaixo:

Figura 01 - Funcionamento do protocolo SPI



Fonte: Autor

A Figura 1 mostra o funcionamento básico do protocolo SPI: a comunicação entre dois dispositivos. O SPI funciona sobre o conceito de troca de mensagens entre um dispositivo mestre e um dispositivo escravo. A troca de mensagens é síncrona, ou seja, existe um sinal de temporizador (ou *clock*) que sincroniza a transmissão dos dados. O mestre é o dispositivo encarregado de gerar esse sinal de *clock*, e por isso ele tem esse nome. Os escravos somente percebem esse sinal e transmitem seus dados em sincronia com ele. Na figura, o sinal é gerado no pino SCLK do mestre, que está conectado no SCLK do escravo.

Como já foi dito, o SPI funciona com troca de mensagens. Quando o mestre deseja enviar um dado para o escravo, o escravo também vai enviar um dado para o mestre. Esses dados são trocados pelos pinos MOSI (***M****aster* ***O****utput,* ***S****lave* ***I****nput*) e MISO (***M****aster* ***I****nput,* ***S****lave* ***O****utput*). Note o sentido de comunicação nas setas da figura. Ou seja, para o mestre transmitir um dado, é preciso colocar um valor no

buffer do mestre e iniciar uma transmissão. Após ser transmitido, o dado no *buffer* do mestre vai ser o dado que estava no *buffer* do escravo, porque eles trocaram mensagens. Para receber um dado de um escravo, o mestre tem que enviar alguma coisa para o escravo, e assim receber o dado que deseja ser lido. Parece meio estranho, mas é assim mesmo que acontece. Quando se deseja só ler, é preciso "enviar" algum "lixo" para o escravo, fazendo com que o dado que interessa apareça no *buffer* do mestre. Se você deseja só transmitir algo, basta colocar o dado no *buffer* e ignorar o que vai estar lá no fim da transmissão. Parece complicado, mas, de fato, não é. Por sorte, as bibliotecas que trabalham com SPI já tratam desses detalhes para você, então, não há muito com o que se preocupar. Mas é importante saber como as coisas realmente funcionam.

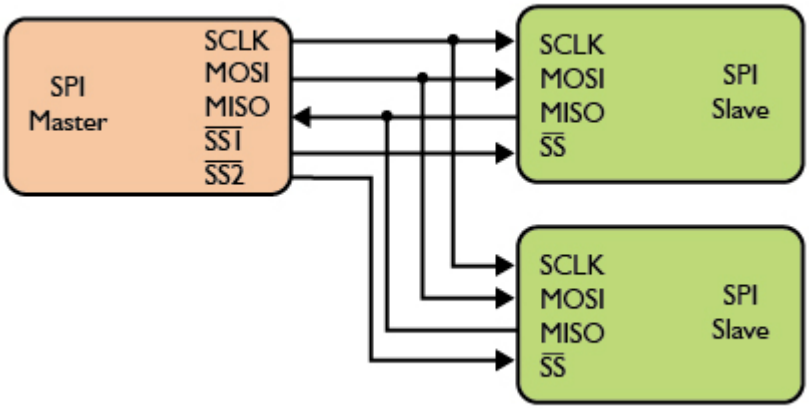
E o que é o sinal "$\overline{SS}$"? Ele é chamado de "Slave Select". Esse sinal serve para selecionar qual o escravo que vai se comunicar com o mestre no momento. Não faz muito sentido conectar o $\overline{SS}$ no caso de existir apenas um escravo (inclusive, o modulo MSSP prevê QUE essa configuração disponibiliza uma configuração que dispensa o uso do $\overline{SS}$), mas mesmo assim ele pode ser utilizado. E no caso de existir mais de um escravo, ele se torna essencial. A barra em cima do $\overline{SS}$ indica que ele é ativado em um valor lógico baixo, ou seja, aplicando 0V.

Atenção

Importante: o escravo só vai estar habilitado enquanto o sinal estiver em 0V! Se você mudar o SS para 5V durante a transmissão dos dados, irá ocorrer uma falha e nenhum dado será trocado entre o mestre e o escravo.

Observe agora a figura abaixo:

Figura 02 - SPI com 2 dispositivos escravos



Fonte: Autor

Na figura 2, podemos ver como seria uma rede SPI com 1 mestre e 2 escravos. Note que todos os sinais "SCK", "MOSI" e "MISO" estão conectados entre si. A diferença está nos "$\overline{SS}$s". Perceba que, agora, temos um $\overline{SS}$ para cada escravo. O que acontece é que quando desejamos trocar mensagem com o escravo 1, colocamos o sinal $\overline{SS1}$ em baixo. Isso habilita o escravo 1 a trocar mensagens com o mestre. Se desejamos trocar mensagens com o escravo 2, colocamos o sinal $\overline{SS2}$ em nível baixo, mantendo o$\overline{SS1}$ em alto. Enquanto as entradas $\overline{SS}$ dos escravos são pinos dedicados do módulo MSSP do microcontrolador, os sinais $\overline{SS1}$ e $\overline{SS2}$ são saídas digitais padrões do PIC. Ou seja, podemos adicionar mais escravos, desde que existam saídas digitais disponíveis para selecionar tais escravos. É importante que seu programa trate de selecionar somente um escravo por vez para trocar mensagens com o mestre, por meio do sinal $\overline{SS}$ equivalente.

Vamos à Prática!

Agora que você já sabe o básico do funcionamento da comunicação SPI, vamos passar para a prática e ver o protocolo em ação. Como já foi mencionado, utilizamos o SPI no microcontrolador por meio do módulo MSSP. A sua configuração é feita de modo semelhante à realizada no módulo EUSART utilizado para transmissões pelo protocolo USART: você tem que consultar o datasheet do PIC, ver quais os

registradores devem ser modificados e quais os valores que eles devem ter para o módulo funcionar da maneira que você deseja, e configurá-los dessa forma no código que vai ser gravado no dispositivo.

Como já falamos no início, nesta aula você vai fazer uma experiência prática. Porém, ela deve ser realizada no laboratório do Metrópole Digital, pois lá vão ter todos os equipamentos e componentes necessários para a experiência, além de um professor para tirar suas dúvidas e garantir que tudo ocorra em segurança. Vamos descrever o circuito aqui, mas, por enquanto, se preocupe apenas em entendê-lo para montar no laboratório.

O experimento que vamos fazer é bem simples: vamos configurar dois microcontroladores PICs para realizarem uma comunicação SPI. Serão conectados alguns LEDs e um botão a um microcontrolador, e outros LEDs ao outro microcontrolador. O que desejamos é transmitir um dado pelo protocolo SPI de um PIC para o outro quando acionarmos o botão. Os LEDs conectados aos PICs vão servir para mostrar qual o dado que está sendo transmitido e qual o dado que foi recebido. Se tudo estiver funcionando corretamente, os LEDs conectados aos microcontroladores devem mostrar a mesma coisa e devem mudar quando o botão é acionado.

Entenderam bem qual o objetivo? Então, vamos aos materiais que vocês vão precisar para realizar o experimento. Para realizar o experimento iremos utilizar a placa "Demo Board 44-Pin" do PICKIT 3 para os circuitos de transmissão e recepção. A placa tem um microcontrolador PIC18F45K20, 8 LEDs e 1 botão, além de outras interfaces que não iremos utilizar. Abaixo segue o layout da placa.

Agora que você já tem os dois microcontroladores e as interfaces (LEDs e botão) embutidos nas placas de demonstração, você vai ter que conectar os microcontroladores para permitir a comunicação SPI. Para isso você precisa saber quais os são os pinos SCK, MOSI, MISO e SS, conforme visto anteriormente na teoria da comunicação SPI.

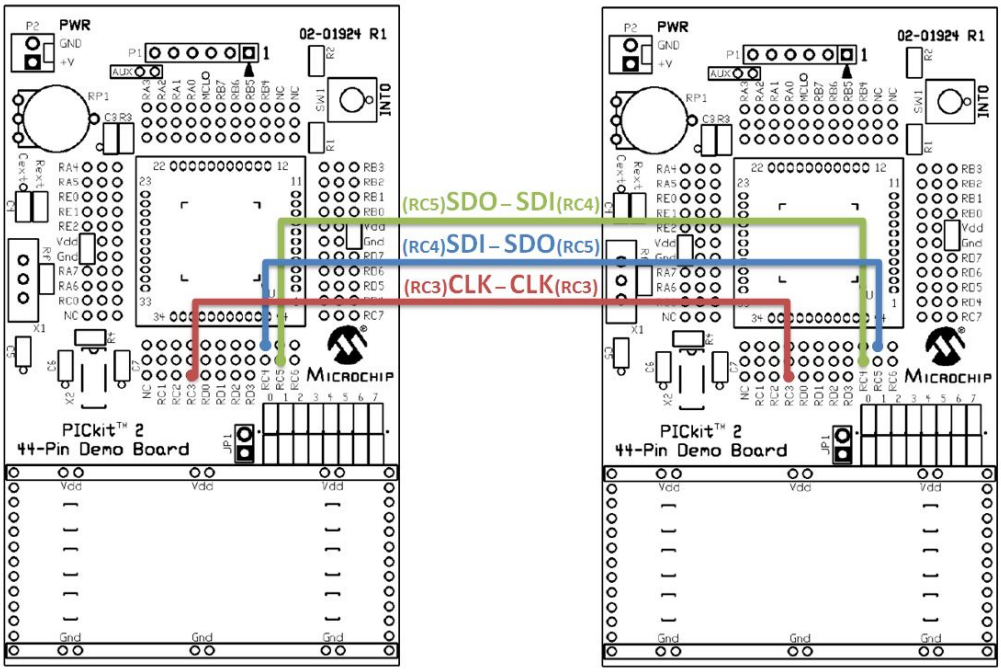
Analisando o *datasheet* do microcontrolador PIC18F45K20 é possível encontrar os pinos que correspondem aos sinais de comunicação SPI. Abaixo segue os pinos e os sinais.

- Pino RC3: Sinal de clock (SCK, SCL).

- Pino RC4: Sinal MISO (SDI).
- Pino RC5: Sinal MOSI (SDO).
- Pino RA5: Sinal SS.

Como vamos realizar uma comunicação entre um microcontrolador mestre e um único microcontrolador escravo é opcional o uso do sinal de seleção de escravos (SS). E no nosso caso vamos evitar o uso desse pino (RA5). Abaixo segue o esquema de comunicação SPI utilizando as placas de demonstração ("Demo Board 44-PIN") do PICKIT 3.

Figura 03 - Esquema de comunicação SPI entre os PICs da placa de demonstração.



Observe, também, que não existem as saídas MOSI e MISO. O que existe no PIC são os pinos SDO, de “Serial Data Output”, e SDI, de “Serial Data Input”. Note que o SDO de um PIC está conectado ao SDI do outro, definindo, então, o esquema de troca de mensagens: a saída de um está ligada à entrada do outro, e vice-versa. Enquanto isso, os sinais de *clocks*, que saem nos pinos SCK, estão conectados entre si, o que está de acordo, pois o *clock* será gerado pelo mestre, enquanto o slave vai receber e utilizar esse *clock* para sincronizar a troca de mensagens.

O Programa

Você já deve ter percebido que só o circuito não vai ser suficiente para realizar a experiência. Você também vai precisar fazer um programa, compilar esse programa, e gravá-lo no PIC para que ele possa ser executado. Na verdade, serão dois programas diferentes, pois temos dois PICs no circuito e, apesar dos programas serem parecidos, há pequenas diferenças entre eles para fazer tudo funcionar da forma esperada.

Você não vai precisar fazer todo o código do começo ao fim, apesar de isso ser um ótimo exercício. Nesta aula, você vai ter acesso a um código parcialmente pronto e terá que fazer as partes que estão faltando, que são exatamente a configuração e o acesso à comunicação SPI. Você deve levar esses códigos para a aula prática, para serem gravados nos microcontroladores e testados no circuito durante a parte prática. É interessante também que você analise todo o código para entender bem o que está sendo feito e tire duvidas durante a aula prática, mesmo que não seja, especificamente, sobre SPI (que é o assunto principal desta aula).

Uma coisa muito importante ao programar para um microcontrolador qualquer: tenha sempre em mãos o *datasheet* do microcontrolador sendo utilizado. Sempre o utilize para verificar os nomes dos registradores, os nomes dos bits e o que cada um deles faz.

Primeiro, nós vamos falar de como fazer para visualizar e editar o código que você vai utilizar na experiência. Em seguida, vamos ver o que precisa ser modificado e, por fim, como gravar os PICS e como gravar o circuito.

Acessando os Arquivos do Experimento

Você vai encontrar um arquivo em anexo chamado “aula_pratica_SPI.zip”, nos anexos da aula. Descompacte o arquivo em alguma pasta no seu computador. Você vai, então, encontrar uma pasta chamada “Aula Pratica SPI”, e dentro dela você vai encontrar mais duas pastas: “Mestre” e “Escravo”. Como já deve supor, cada uma dessas pastas vai conter os códigos de um dos PICS a serem gravados.

Cada código foi desenvolvido utilizando o programa MPLAB, da Microchip, e o compilador C18 (caso você não tenha esse programa instalado, consulte os links da referência para baixar e instalar o MPLAB e o compilador C18). Dentro das pastas do Mestre e Escravo você vai encontrar os seguintes arquivos:

- **Mestre.mcp** ou **Escravo.mcp**: arquivos com as informações dos projetos gerenciados pelo MPLAB.
- **Mestre.mcw** ou **Escravo.mcw**: arquivos que contém informações sobre o espaço de trabalho do MPLAB para aquele projeto.
- **main_mestre.c** ou **main_escravo.c**: arquivos que contém o código em C que vão ser gravados nos respectivos PICs mestre e escravo.
- **header_mestre.h** ou **header_escravo.h**: arquivos de cabeçalhos com algumas definições de funções para o programa contido, respectivamente, no main_mestre.c e main_escravo.c.

O arquivo que você vai, efetivamente, editar é o main_mestre.c e o main_escravo.c, mas queremos fazer isso utilizando o ambiente do MPLAB, ou seja, queremos abrir os arquivos **.mcw**. Portanto, vamos começar abrindo o arquivo **Mestre.mcw**.

O Espaço de Trabalho do MPLAB

Ao abrir o arquivo **Mestre.mcw**, a janela do MPLAB deve aparecer para você. No lado esquerdo, você deve ver uma janela mostrando os arquivos do projeto. Dê um clique duplo no arquivo “main_mestre.c” nessa janela para abrir o código que vai ser gravado no PIC mestre. Dê uma olhada no código, prestando bastante atenção nos comentários, para ter uma noção geral do que o código faz. Procure encontrar os espaços no código onde você irá colocar o seu código. É importante você ter uma noção geral do código antes de começar a editá-lo.. Dê uma olhada também no arquivo “header_mestre.h”. Ele não é um arquivo essencialmente obrigatório, mas sempre é interessante separar os protótipos de funções e algumas definições da implementação do código. Os demais arquivos são arquivos do compilador e não devem ser editados pelo usuário: devem apenas ser incluídos nos projeto e utilizados.

Já analisou bem os arquivos? Ótimo! Então, vamos em frente.

Se Liga!

Pode ser que apareçam algumas mensagens de "arquivo não encontrado" ("*file not found*") ao lado do nome de alguns arquivos. Se isso acontecer, significa que a instalação do compilador C18 no seu computador não está no diretório padrão, e você vai ter que localizar esses arquivos. Clique com o botão direito sobre eles e clique em "*Locate missing file*". Então, procure os arquivos no seu computador. O diretório padrão para os arquivos *.h* é o "*C:\Arquivos de Programas\Microchip\mplabc18\v3.40\h*" e o diretório dos arquivos .lkr é "*C:\Arquivos de Programas\Microchip\mplabc18\v3.40\bin\LKR*".

O Código do Dispositivo Mestre

Bem, chegou a hora de, realmente, escrever algum código para o PIC. Para o mestre funcionar corretamente, você vai ter que desenvolver o código para fazer as seguintes coisas:

1. Configurações do PIC
    - 1.1 Configurar corretamente as portas utilizadas como saídas digitais.
    - 1.2 Configurar corretamente a entrada que será utilizada pelo botão.
    - 1.3 Configurar o módulo MSSP para funcionar como mestre de uma comunicação SPI.
    - 1.4 Configurar o PIC para chamar uma interrupção, quando o botão for pressionado.
2. Implementar a rotina de interrupção que é chamada quando o botão é pressionado

2.1 Atualizar o dado a ser transmitido

2.2 Transmitir o dado pela SPI

3. Implementar a rotina principal do programa

3.1 Colocar o dado na saída digital, de forma a ficar visível nos LEDs do circuito.

Nas seções seguintes, nós vamos explicar, rapidamente, o que você vai precisar implementar no experimento. Os detalhes de implementação estão disponíveis em forma de comentários no arquivo “main_mestre.c”. Você vai precisar consultar o datasheet do microcontrolador PIC para configurar corretamente o funcionamento do Mestre.

Configurações do PIC

As configurações do PIC serão feitas na função “UserInit()”, no arquivo “main_mestre.c”. Essa função é executada uma vez antes do PIC entrar no laço principal (dê uma olhada na função “main()”, no arquivo “main_mestre.c”). Nela, teremos que configurar o valor inicial das variáveis utilizadas, configurar as saídas do PIC conectadas aos LEDs do circuito para funcionarem como saídas digitais, configurar o módulo MSSP para funcionar no protocolo SPI, como Mestre, e configurar o PIC para receber as interrupções ao pressionar o botão no circuito. Verifique no datasheet quais são os registradores relacionados à porta D (que está sendo utilizada como saída digital), à porta B (que vai perceber o botão sendo pressionado), os registradores relacionados ao módulo MSSP, também os relacionados ao mecanismo de interrupção do PIC. Leia, atentamente, os comentários na função “UserInit()” para saber detalhes do que deve ser escrito no programa para configurar o PIC.

Se Liga!

O compilador C18 facilita muito a criação de programas para serem executados em PICs. Ele cria definições para cada um dos registradores do PIC que você está trabalhando, permitindo fácil acesso ao valor do registrador e também aos seus bits. Para tanto, temos que incluir, no início do código, um cabeçalho com

as definições dos registradores do PIC que estamos trabalhando (no nosso caso, o arquivo “p18f45k20.h" ou "xc.h"). Por exemplo, se eu quero colocar o valor 0x05 no registrador PORTB, eu utilizo a seguinte linha de código:

PORTB = 0x05;

e se eu desejo mudar somente o bit RB2 do registrador PORTB para o valor 1, eu utilizo a seguinte linha de código:

PORTBbits.RB2 = 1;

Se familiarize com os nomes dos registradores e bits do PIC lendo o *datasheet* dele.

Atividade 01

Abra o arquivo “main_mestre.c” na pasta “Mestre”, procure a função “UserInit()” e siga as instruções nos comentários para:

1. Definir o valor inicial da variável LED_Display;
2. Configurar a porta D do PIC como saída;
3. Configurar o pino 0 da porta B como entrada;
4. Configurar a comunicação SPI;
5. Configurar a interrupção externa INT0.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Rotina de Interrupção

A interrupção é tratada na função “InterruptServiceHigh”, no arquivo “main_mestre.c”. Com o PIC devidamente configurado, essa função vai ser chamada sempre que o botão do circuito for pressionado. Nela você deve primeiro fazer uma verificação para saber o quê gerou a interrupção. Se a interrupção foi gerada pelo botão, vamos, então, alterar o valor a ser exibido nos LEDS (assim, podemos perceber alguma mudança no funcionamento do PIC, quando pressionarmos o botão), e vamos enviar esse valor pelo protocolo SPI. Detalhes de como realizar essas modificações podem ser vistos nos comentários da função “InterruptServiceHigh”.

Atividade 02

Abra o arquivo “main_mestre.c” na pasta “Mestre”, procure a função “InterruptServiceHigh()” e siga as instruções nos comentários para:

1. Verificar se foi o INT0 que gerou a interrupção
2. Limpar o flag de interrupção
3. Atualizar o valor da variável LED_Display
4. Transmitir o dado pela comunicação SPI.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Rotina Principal do Programa

A rotina principal do programa é o “UserProcess()”. Ela é chamada dentro de um looping infinito dentro da função “main”, no arquivo “main_mestre.c”. Ou seja, depois de configurar tudo na função “UserInit()”, nós vamos ficar executando a função “UserProcess()”, indefinidamente, até o PIC ser reiniciado, seja desligando e religando ou por meio do pino de reset ($\overline{MCLR}$).

Nessa função, queremos somente exibir, nos LEDs, um valor interno atual no PIC. Note que, se o botão não for pressionado, o valor não será alterado, e estaremos mandando os LEDs acenderem sempre com o mesmo valor, fazendo com que não seja notada nenhuma alteração quando observarmos o circuito. Porém, ao apertar o botão, vamos mudar esse valor e, assim que sairmos da rotina de interrupção, o “UserProcess” vai ser executado novamente, e, então, poderemos ver a alteração nos LEDs.

Se você estiver entendendo tudo direitinho até agora, você vai notar que podemos também acender os LEDs dentro da interrupção, já que eles só mudam de valor quando o botão é pressionado, e deixarmos a função “UserProcess” em branco, ou mesmo eliminar essa função do código. Sim, isso é possível, e o código estaria correto. De fato, a divisão entre “UserInit” e “UserProcess” é só uma questão de organização, e, em certas situações, podemos realizar uma mesma operação em diferentes locais de um código, sem que isso altere o resultado desejado.

Recomendo deixar que os LEDs sejam acesos no “UserProcess”, separando, assim, o que realmente só pode ser feito na interrupção com o que pode ser feito fora dela. Se você tentar sempre seguir essa regra, os seus programas ficarão bem mais legíveis e fáceis de alterar ou evoluir no futuro.

Atividade 03

---

1. Abra o arquivo “main_mestre.c” na pasta “Mestre”, procure a função “UserProcess()” e siga as instruções nos comentários para: Colocar na porta D o valor da variável LED_Display

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

O Código do Dispositivo Escravo

1. Configurações do PIC

    1.1 Configurar corretamente as portas utilizadas como saídas digitais.

    1.2 Configurar o módulo MSSP para funcionar como escravo, sem utilizar o pino SS, em uma comunicação SPI.

    1.3 Configurar o PIC para gerar uma interrupção, quando um dado for recebido pela SPI.

2. Implementar a rotina de interrupção que é chamada quando o botão é pressionado

    2.1 Receber o dado da SPI

3. Implementar a rotina principal do programa

    3.1 Colocar o dado na saída digital, de forma a ficar nos LEDs do circuito.

Configurações do PIC

As configurações do PIC para o escravo são semelhantes às configurações do PIC mestre, com algumas pequenas modificações. As saídas digitais ligadas aos LEDs são as mesmas, logo, a configuração é a mesma. O receptor não vai receber informação do botão, assim, não é preciso configurar nenhuma entrada pela porta B, como é feito no mestre. O módulo MSSP agora vai funcionar no escravo, e como temos apenas um mestre e um escravo, não vamos utilizar a função *"Slave Select"* ($\overline{SS}$).

Não se esqueça de configurar o *clock* do módulo da mesma forma que você configurou no mestre. Também é necessário configurar o PIC para gerar interrupções do módulo SPI (e não de um botão, como no mestre). Queremos que o PIC entre na rotina de interrupção sempre que um novo dado estiver disponível no módulo, recebido do mestre, portanto, é o PIC quem tem que ser configurado adequadamente.

Atividade 04

Abra o arquivo "main_escravo.c" na pasta "Escravo", procure a função "UserProcess()" e siga as instruções nos comentários para:

1. Inicializar a variável LED_Display;
2. Configurar a porta D para funcionar como saída digital
3. Configurar a SPI para funcionar como escravo;
4. Configurar o PIC para receber a interrupção da comunicação SPI.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Rotina de Interrupção

A rotina de interrupção no escravo é bem mais simples do que a mesma rotina no mestre. A rotina vai ser chamada sempre que existir um novo dado disponível, assim que ele for recebido pela SPI. Na rotina de interrupção, vamos precisar, somente, copiar esse valor do buffer do módulo MSSP e colocar na variável que estamos utilizando para acionar os LEDs, além de limpar a flag que gerou a interrupção. Nada mais além disso.

Atividade 05

Abra o arquivo “main_escravo.c” na pasta “Escravo”, procure a função “InterruptServiceHigh()” e siga as instruções nos comentários para:

1. Verificar se foi a SPI que causou a interrupção no PIC;
2. Limpar o flag de interrupção;
3. Colocar o dado recebido na variável LED_Display.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Rotina Principal do Programa

A rotina principal do programa vai ser exatamente ingual à rotina do mestre, ou seja, vamos apenas exibir nos LEDs o valor de uma variável que vai ser atualizada na rotina de interrupção.

Atividade 06

1. Abra o arquivo “main_escravo.c” na pasta “Escravo”, procure a função “UserProcess()” e siga as instruções nos comentários para:
   Colocar na porta D o valor da variável LED_Display.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Compilação e Gravação dos PICs

Para colocar os códigos para funcionar, é preciso compilá-los e gravá-los nos PICs. Para a gravação, vamos utilizar o “PICKit 3”. Ele é um dispositivo que funciona por meio da USB, e tem a capacidade de alimentar um circuito que utilize pouca corrente (como o deste experimento), além de gravar e realizar o debug do PIC em tempo de execução. Porém, neste experimento, vamos apenas utilizá-lo para alimentar e gravar o circuito.

Compilando o Código

Antes de gravar seu PIC, você vai ter que compilar o codigo. Abra o projeto usando o MPLAB e procure os botões, apresentados na Figura 4, na parte superior da janela.

Figura 04 - Botões de compilação

Para compilar corretamente, você deve selecionar “Release”, na caixa de diálogo à esquerda, e depois clicar no botão “Build All” (o que está marcado com um círculo vermelho). O MPLAB vai, desse modo, utilizar o C18 para compilar o código do dispositivo mestre. Uma janela vai abrir, mostrando o resultado da compilação. Se você vir um “**BUILD SUCCEEDED**”, então, o seu código está escrito corretamente. Isso não quer dizer que o seu código está certo: somente que ele está escrito de uma forma correta. Se você vir um “**BUILD FAILED**”, então, seu programa tem algum erro. Veja as mensagens de compilação e procure por erros. Um exemplo de erro é o seguinte:

C:\Escravo\main_escravo.c:241:Error [1105] symbol 'erro_proposital' has not been defined

Se você clicar duas vezes nessa mensagem de erro no MPLAB, você vai ver a linha que gerou o erro. Corrija-o e tente compilar novamente. Repita o processo até você conseguir compilar corretamente.

Atividade 07

---

1. Utilize o procedimento acima para compilar os códigos que você criou para o mestre e para o escravo, e tente corrigir os erros de compilação que aparecerem. Lembre-se que uma compilação bem sucedida não significa que o seu programa está correto. Você só poderá verificar isso durante a experiência prática no laboratório.

Clique aqui para verificar suas respostas.

Respostas

1. Não há resposta. Atividade prática para o aluno.

Gravando o PIC

Agora, para gravar o PIC você vai precisar do programador, que vai estar disponível no laboratório. Conecte o gravador ao computador por meio do cabo USB e depois conecte o gravador do PICKIT 3 à placa de demonstração (Demo Board 44-Pin).

Atenção

Lembra-se do pino que está marcado no esquema do circuito? Observe na imagem a seguir a seta branca, em uma das pontas do conector do PICKit 3 com a placa. Você deve conectar o pino marcado no circuito com furo marcado pela

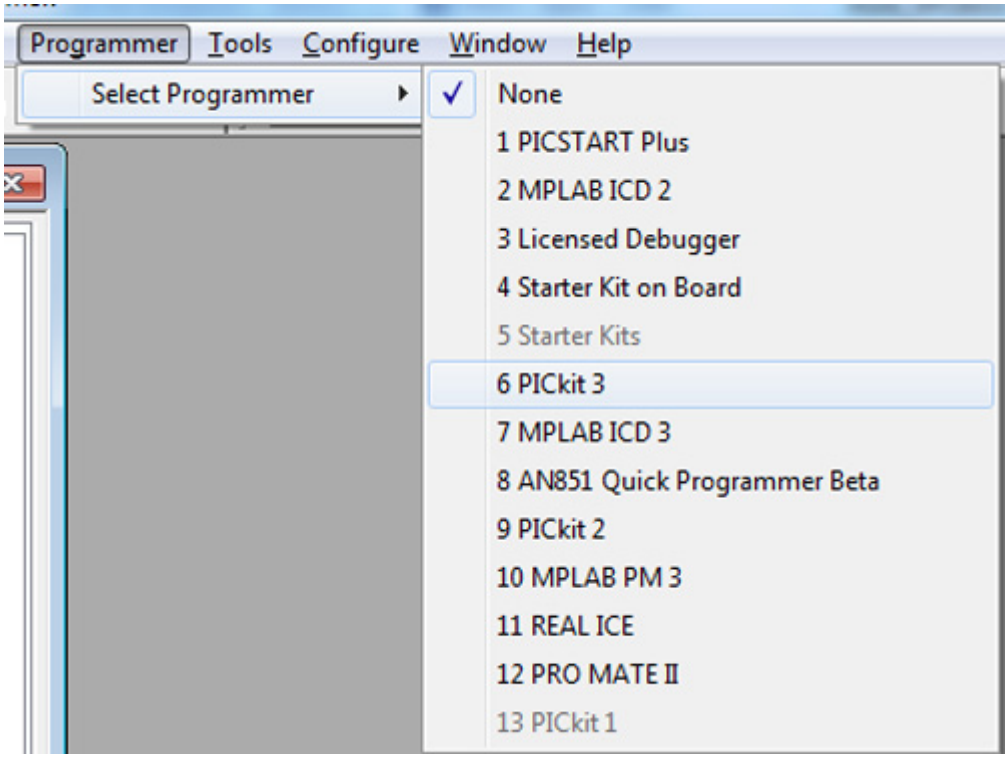
seta branca. É este furo que define a ordem. Caso você inverta, poderá danificar o circuito. Então, tome bastante cuidado.

Figura 05 - O gravador do PICKit 3

Fonte: Manual do PICKit 3

Agora você precisa selecionar qual o gravador que você vai utilizar. Vá em “Programmer > Select Programmer > 6 PICkit 3” para selecionar o gravador que vamos utilizar, como mostra a figura a seguir.

Figura 06 - Menu de seleção do programador PICKit 3



Ao selecionar o PICkit 3, o MPLAB vai tentar conectar ao dispositivo. Como o PICkit 3 tem a capacidade de alimentar o circuito, ele vai perguntar se você tem certeza que o PIC selecionado no MPLAB é o que está sendo utilizado. Caso você esteja utilizando um PIC que funcione com uma voltagem menor, como, por exemplo, o PIC18F45K20 que funciona com 3.3V, e tiver selecionado no MPLAB um microcontrolador que funcione com 5V, como o PIC18F2550, você pode queimar o PIC. Se você abriu o arquivo como indicado, ou seja, abrindo o arquivo "Mestre.mcw", então, o PIC selecionado é o correto. De qualquer forma, você pode ver o nome do PIC na mensagem de aviso, e verificar se é ele mesmo que está no circuito. Estando tudo certo, clique em OK.

Mesmo assim, pode acontecer de o PICkit ainda não conseguir alimentar o circuito. Caso isso aconteça, verifique se a opção de alimentação do circuito está habilitada. Clique em "Programmer > Settings ..." e, depois da aba "Power", marque a opção "Power target circuit from PICkit 3", **NÃO** altere o valor de voltagem de alimentação. Clique em OK e o aviso deve aparecer novamente. Clique em OK, na janela de aviso, e o seu circuito deve estar alimentado.

Você deve ver, agora, alguns novos botões na interface na parte superior do MPLAB. Procure os botões mostrados na figura abaixo.

Figura 07 - Botões de gravação do PIC

Da esquerda para a direita, as funções desses botões são:

- Programar o PIC.
- Ler os dados na memória do PIC.
- Verificar a gravação do PIC
- Apagar a memória do PIC.
- Verificar se o PIC com a memória apagada.
- Tirar o PIC do estado de "*reset* (na imagem, este botão está desabilitado).
- Manter o PIC no estado de "*reset*"
- Alimentar o PIC.

Estamos interessados no primeiro botão, o de gravar o PIC, e, no último, o de alimentar o circuito. Depois de ter compilado e conectado corretamente o gravador, clique no botão de gravação. O programa que foi compilado vai ser gravado no dispositivo. Você deve receber a mensagem “Programming...” enquanto o PICkit estiver programando o dispositivo, e “Programming/Verify complete”, quando a gravação terminar sem erros.

Se, por acaso, você perceber que esses botões estão aparecendo da forma como mostra a Figura 8, isso indica que o PIC não está sendo alimentado e, logo, você não pode gravar o PIC. Clique no último botão para alimentar o circuito. Talvez as mensagens de aviso tornem a aparecer, mas agora você já deve saber como lidar com elas.

Figura 08 - Botões de programação desligados

Pronto. Se você chegou até aqui, então, temos o PIC mestre gravado corretamente e funcionando. Experimente apertar o botão e ver se os LEDs estão mudando da forma esperada.

Feito isso, podemos, então, gravar o dispositivo escravo. Primeiro, feche o MPLAB. Então, mude o PICkit da barra de pinos do mestre para a barra de pinos do escravo. Lembre-se de conectar o gravador no sentido correto. Feito isso, abra o arquivo “Escravo.mcw”. A partir daqui você pode seguir os passos de forma semelhante ao que foi feito para gravar o mestre. Perceba que ao alimentar o escravo, o mestre também será alimentado. Após gravar o programa no PIC escravo, você pode testar o funcionamento do circuito. Experimente apertar o botão e ver o que acontece.

Execução do Programa

Com os microcontroladores PICs gravados e as placas conectadas, vamos executar o programa. Ao alimentar o circuito por meio do PICKit, você vai ver os LEDs do mestre e do escravo mostrando as condições iniciais que você colocou nos respectivos códigos. Experimente pressionar o botão: se tudo estiver correto, os LEDs do mestre vão se alterar de acordo com a lógica que você implementou na interrupção do mestre. Ao mesmo tempo, os LEDs do escravo também serão alterados, pois o mestre enviou o novo valor pela comunicação SPI e o escravo utilizou esse novo valor para alterar a forma que seus LEDs são acesos. Ou seja, os dois grupos de LEDs devem sempre mostrar a mesma coisa (com a exceção da condição inicial, caso você tenha definido condições iniciais diferentes), e devem mudar de estado sempre que o botão for pressionado.

Se você conseguiu ver esse funcionamento no circuito, parabéns! Você entendeu direitinho como realizar uma comunicação pela SPI. Caso algo dê errado, não se preocupe. Tente localizar o problema analisando o comportamento do circuito. Aposto que após algumas tentativas você vai conseguir fazer o circuito funcionar na maneira correta.

Ok, Funcionou! E Agora?

Tudo certo? Terminamos a experiência? Ainda não. Até aqui, configuramos o PIC acessando os seus registradores e ajustando seus bits, ou seja, sempre escrevendo diretamente nos registradores que desejávamos alterar. Pois bem, essa é uma das maneiras de fazer isso. Existe uma maneira mais fácil: utilizando as funções disponíveis por alguma biblioteca, como, por exemplo, a disponível com o compilador C18. Por exemplo, podemos configurar o módulo SPI por meio da função “OpenSPI”, passando os parâmetros desejados a essa função. Podemos escrever algo utilizando a função “WriteSPI”, ou ler de um dado de um escravo utilizando “ReadSPI”. Para poder utilizar essas funções, adicione o arquivo “spi.h” nos cabeçalhos dos seus projetos no MPLAB para o mestre e o escravo, e adicione a linha **#include “spi.h”** nos cabeçalhos, sendo incluídos nos seus arquivos “main_mestre.c” e “main_escravo.c”.

Apesar de essas funções facilitarem a criação e a leitura do código, é importante que você saiba como fazer a mesma coisa sem utilizá-las. Assim, caso algum problema aconteça, você vai saber analisar corretamente o erro e descobrir como corrigi-lo, esteja você utilizando essas funções auxiliares ou não.

Atenção

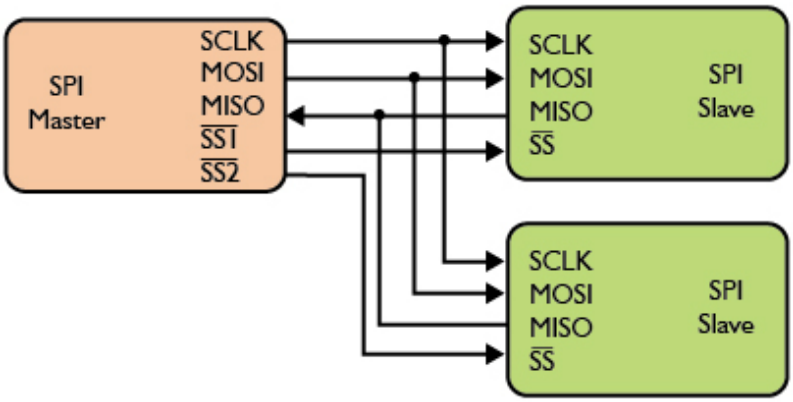
A função “ReadSPI” não deve ser utilizada para fazer a leitura de um dado no escravo. A forma correta de fazer é copiar diretamente o valor do buffer para uma variável. Essa função deve ser utilizada para um dispositivo mestre ler um dado de um escravo. Como o SPI funciona por troca de mensagens, o mestre tem que enviar um dado qualquer para poder receber um dado do escravo, e essa função implementa, exatamente, esse tipo de rotina, evitando que você tenha que implementar tudo isso.

Resumo

Na aula de hoje, vimos como fazer para comunicar um PIC com outro utilizando o protocolo SPI. Estudamos como é feita a comunicação nesse protocolo e como é a conexão dos fios entre os dispositivos. Um deles deve ser programado como mestre e os demais como escravos. Você recebeu códigos parcialmente escritos para um dispositivo mestre e escravo e aprendeu como programar a comunicação SPI em um PIC, tanto como mestre quanto como escravo. Na próxima aula, vamos utilizar esse conhecimento de SPI para transmitir um dado por meio de uma comunicação sem fio, através de um dispositivo que se comunica com o PIC através da SPI.

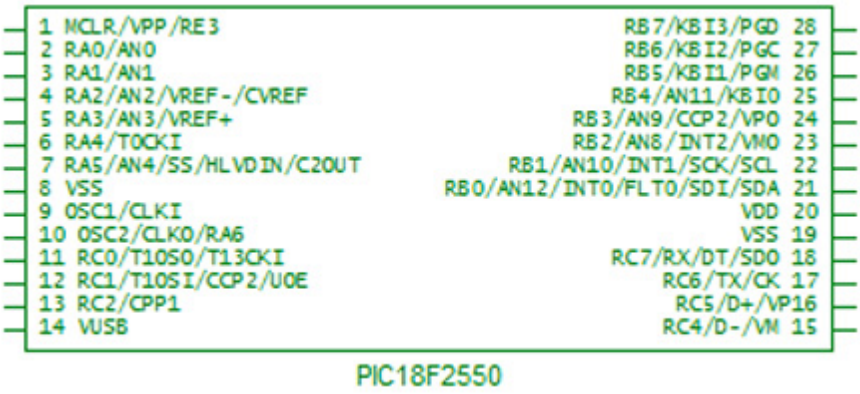
Autoavaliação

1. Baseado na figura a seguir, mostre como seria a conexão em uma rede SPI com um dispositivo mestre e 4 escravos.



Fonte: Autor

2. Agora substitua o bloco que representa o mestre pela figura seguinte e refaça as conexões da rede SPI. Quais saídas digitais você utilizaria para substituir os sinais de “Slave Select” no mestre?



Fonte: Autor

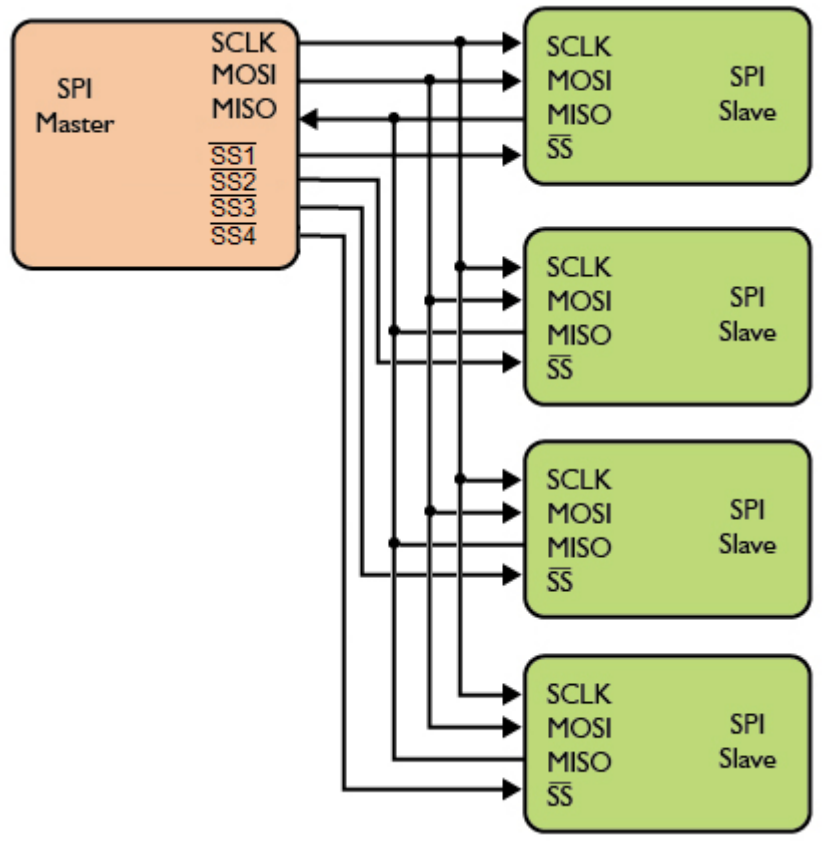
3. Baseado na figura da questão anterior e nos conhecimentos de SPI que você adquiriu na aula, faça um pseudocódigo para o mestre e que pegue um byte do dispositivo escravo número 1 e copie para os dispositivos escravos 2, 3, e 4. Use os pinos que você definiu como slave select na questão 2 para selecionar corretamente os escravos.

4. Agora escreva um pseudocódigo para um mestre que leia um byte dos dispositivos escravos 1 e 4, faça uma comparação entre eles e envie o maior valor para o escravo 2 e o menor para o escravo 3.

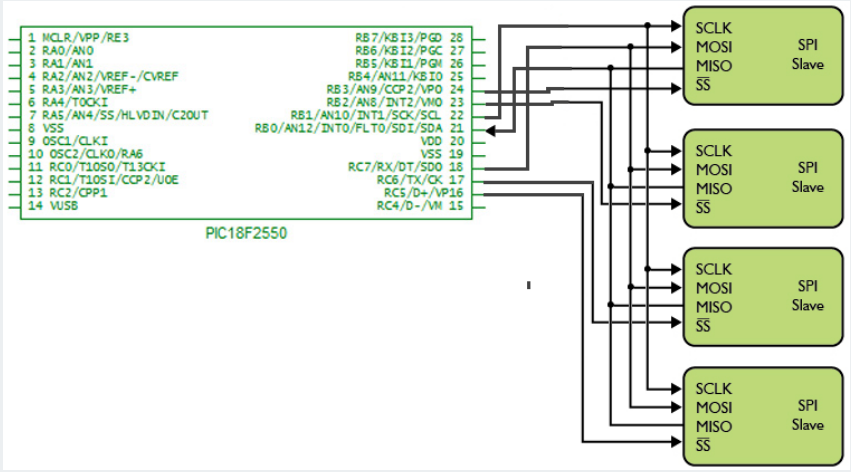
Clique aqui para verificar suas respostas.

Respostas

1. Baseado na figura a seguir, mostre como seria a conexão em uma rede SPI com um dispositivo mestre e 4 escravos.



2. Agora substitua o bloco que representa o mestre pela figura seguinte e refaça as conexões da rede SPI. Quais saídas digitais você utilizaria para substituir os sinais de “Slave Select” no mestre?



3. Baseado na figura da questão anterior e nos conhecimentos de SPI que você adquiriu na aula, faça um pseudocódigo para o mestre e que pegue um byte do dispositivo escravo número 1 e copie para os dispositivos escravos 2, 3, e 4. Use os pinos que você definiu como slave select na questão 2 para selecionar corretamente os escravos.

```
RB3 = 0;
    data = read_SPI();
    RB3 = 1;
    RB2 = 0;
    Write_SPI(data);
    RB2 = 1;
    RC6 = 0;
    Write_SPI(data);
    RC6 = 1;
    RC5 = 0;
    Write_SPI(data);
    RC5 = 1;
```

4. Agora escreva um pseudocódigo para um mestre que leia um byte dos dispositivos escravos 1 e 4, faça uma comparação entre eles e envie o maior valor para o escravo 2 e o menor para o escravo 3.

```
RB3 = 0;
    data1 = read_SPI();
    RB3 = 1;
    RC5 = 0;
    data2 = read_SPI();
    RC5 = 1;
    Se data1>data2{
       RB2 = 0;
       Write_SPI(data1);
       RB2 = 1;
       RC6 = 0;
       Write_SPI(data2);
       RC6 = 1;
    }
    senao{
       RB2 = 0;
       Write_SPI(data2);
       RB2 = 1;
       RC6 = 0;
```

```
        Write_SPI(data1);
        RC6 = 1;
    }
```

Referências

MPLAB IDE USER'S GUIDE: Preliminary

PIC18F2455/2550/4455/4550Data Sheet: 28/40/44-Pin, High-Performance,Enhanced Flash, USB Microcontrollerswith nanoWatt Technology: Preliminary, [2006].

Serial Peripheral Interface Bus Disponível em: <http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus> [Acessado em 9 Ago 2012].