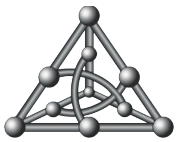

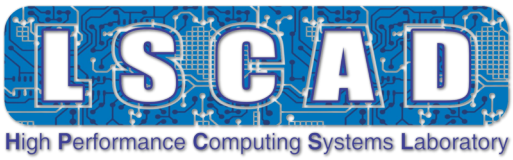

Protocolo Serial Peripheral Interface

Descrição do processo e aplicações para Arduino

Raphael Rocha da Silva [1]
Hugo Rodrigues Anjos [2]
Acadêmicos de Engenharia de Computação pela
Universidade Federal de Mato Grosso do Sul

Laboratório de Sistemas de Computação de Alto Desempenho [3]
Faculdade de Computação [4]
Universidade Federal de Mato Grosso do Sul [5]
Campo Grande, novembro de 2013

1 http://lattes.cnpq.br/0145978111563399 / http://raphael.neocities.org
2 http://lattes.cnpq.br/4469839530602883
3 http://lscad.facom.ufms.br
4 http://facom.sites.ufms.br
5 http://ufms.br

INTRODUÇÃO

O barramento Serial Peripheral Interface (ou barramento SPI) é um protocolo de comunicação de dados. É ideal para transferência de dados entre dois ou mais dispositivos periféricos ao longo de uma curta distância. Eventualmente, pode ser usado para a comunicação entre dois microcontroladores. O protocolo é bastante flexível, e pode sofrer variações para se adequar às necessidades.

Este material será focado à plataforma Arduino™ Mega 2560, principalmente quando forem usados exemplos. Contudo, o processo propriamente dito é bastante genérico, e pode ser utilizado em outros microcontroladores, porquanto as diferenças podem ser verificadas realizando-se consultas às documentações apropriadas.

Desse modo, este material é adequado a estudantes que estejam fazendo pesquisa teórica a respeito do SPI, contanto que já tenham prévio conhecimento sobre a plataforma Arduino; é adequado também para desenvolvedores que queiram utilizar o SPI na integração do Arduino Mega 2560 com outros dispositivos. Desenvolvedores de outras plataformas também podem consultá-lo, desde que sempre se atentem às peculiaridades do microcontrolador usado, e que utilizem, em paralelo, material específico para tal microcontrolador. Além disso, é recomendado ter em mãos as documentações dos demais dispositivos envolvidos.

CARACTERIZAÇÃO DO PROCESSO

O processo de comunicação de dados para o SPI deve apresentar algumas características, dentre as quais:

Serialidade: A comunicação serial de dados é o processo onde a transmissão de dados é feita um bit de cada vez.

Sincronismo: A comunicação é dita “síncrona” porque suas partes estão em sincronia com um sinal de clock, isto é, as mudanças de nível lógico nos componentes do circuito ocorrem todas ao mesmo tempo.

Duplexidade: A comunicação pode ser feita em modo full-duplex, que permite a transmissão bidirecional simultânea. Nesse modo, dois dispositivos conseguem, ao mesmo tempo, enviar e receber dados um do outro .

Modelo master/slave: Os dispositivos envolvidos na comunicação seguem o modelo master/slave, onde um dos dispositivos (o master) tem controle unidirecional sobre outros (slaves). No texto deste documento, o dispositivo master será sempre representado pelo Arduino. Os slaves podem ser dispositivos como sensores, displays, placas de funções específicas, etc.

O barramento SPI define quatro sinais lógicos, abaixo descritos com os pinos de controle referentes ao Arduino Mega 2560.

SCLK (serial clock): Sinal de clock (saída do master) - pino 51

MOSI (master out, slave in): Dados do master para os slaves (saída do master) - pino 50

MISO (master in, slave out): Dados dos slaves para o master (saída do slave) - pino 52

SS (slave select): slave select (saída do master) - pino 53

Nota: na literatura, existem diversos nomes alternativos para esses sinais.

O pino 53, que controla o SS, é útil apenas quando o Arduino é utilizado como slave. Como a biblioteca de SPI nativa oferece suporte apenas ao modo master para o Arduino, o pino 53 deve ser sempre setado como OUTPUT. Caso contrário, pode ocorrer o problema de a interface SPI ser colocada automaticamente no modo slave pelo hardware, o que tornaria a biblioteca inoperante.

Na Figura 1, há um esquema mostrando a relação master/slave entre os dispositivos:

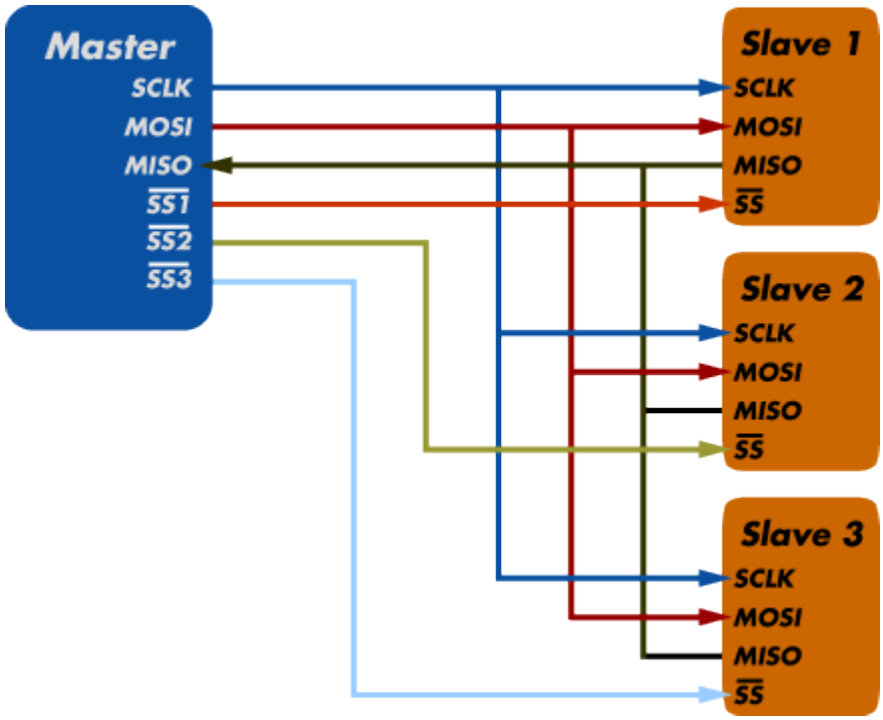

Figura 1: O sinal SCLK é sincronizado pelo master entre todos os slaves. O sinal MOSI envia os dados do master para um dos slaves; o sinal MISO envia os dados de um dos slaves para o master. Os sinais SS1, SS2 e SS3 indicam qual dos slaves está envolvido na transferência dos sinais MOSI e MISO: apenas um dentre os três (SS1, SS2 e SS3) fica em nível baixo, indicando o slave correspondente. Os outros dois slaves recebem sinal SS em nível alto, e portanto ignoram os sinais MOSI durante o momento; o master, por sua vez, ignora os sinais MISO para os quais o sinal SS do dispositivo correspondente foi colocado em nível alto.

É importante compreender que o master só consegue se comunicar com um slave de cada vez; é importante também saber que os sinais SS são setados pelo master, e os slaves respeitam esses sinais, ignorando ou não o sinal de dados MOSI. Assim, é o master que decide com qual dos dispositivos ele pretende trocar informações.

Se um dispositivo A tentar se comunicar com o master enquanto este estiver se comunicando com um outro dispositivo B, o sinal enviado por A será perdido, a menos que ele espere que seu SS seja setado em 0.

Também, um dispositivo não consegue receber informações do master, a menos que ele esteja participando da comunicação; dessa maneira, fica garantido que um slave não vai receber um sinal por engano, isto é, um sinal que não foi enviado para ele (desde que o master esteja configurando os sinais corretamente), mesmo compartilhando o barramento MOSI com outros dispositivos, como visto na Figura 1.

Se existir um único slave, o sinal SS de tal dispositivo pode ser fixado em nível baixo (contudo, alguns slaves podem requerer uma mudança de borda (nível alto para baixo) para serem selecionados). Ao trabalhar-se com múltiplos slaves, é necessário ter um sinal SS independente para cada um deles.

IMPLEMENTAÇÃO DE CÓDIGOS

O Arduino provê uma biblioteca que simplifica o desenvolvimento de aplicações envolvendo a comunicação SPI. Para a configuração do modo de operação, são disponíveis as funções SPI.setBitOrder(), SPI.setClockDivider() e SPI.setDataMode().

Para início e fim de comunicação, utilizam-se as funções SPI.begin() e SPI.end().

Finalmente, a transmissão e recepção (simultâneas) de dados é comandada pela função SPI.transfer().

A documentação oficial do Arduino [1] recomenda a avaliação de algumas propriedades antes de implementar-se um código para SPI:

→ O shift feito no dado é pelo bit mais significativo (MSB) ou menos significativo (LSB) primeiro?

→ O sinal de clock fica inativo em nível alto ou baixo?

→ Os shifts do dado ocorrem na borda de subida ou de descida?

→ A qual velocidade o SPI vai operar?

Os padrões definidos para SPI são bem flexíveis, e as implementações podem ter variações para dispositivos diferentes. Portanto, para escrever o código, deve-se verificar algumas informações sobre o dispositivo no seu respectivo datasheet.

Abaixo, são mostrados exemplos de trechos de códigos que utilizam a biblioteca SPI, com suas funções comentadas (em inglês), retirados de [3].

```
void setup(){
  // initialize the bus for a device on pin 4
  SPI.begin(4);
  // initialize the bus for a device on pin 10
  SPI.begin(10);
}
```

```
void setup(){
  // initialize the bus for the device on pin 4
  SPI.begin(4);
  // Set clock divider on pin 4 to 21
  SPI.setClockDivider(4, 21);
  // initialize the bus for the device on pin 10
  SPI.begin(10);
  // Set clock divider on pin 10 to 84
  SPI.setClockDivider(10, 84);
}
```

```
void loop(){
  byte response = SPI.transfer(4, 0xFF);
}
```

```
void loop(){
//transfer 0x0F to the device on pin 10, keep the chip selected
SPI.transfer(10, 0xF0, SPI_CONTINUE);
//transfer 0x00 to the device on pin 10, keep the chip selected
SPI.transfer(10, 0x00, SPI_CONTINUE);
//transfer 0x00 to the device on pin 10, store byte received in
response1, keep the chip selected
byte response1 = SPI.transfer(10, 0x00, SPI_CONTINUE);
//transfer 0x00 to the device on pin 10, store byte received in
response2, deselect the chip
byte response2 = SPI.transfer(10, 0x00);
}
```

REFERÊNCIAS BIBLIOGRÁFICAS

1. SPI Library. Disponível em <http://arduino.cc/en/Reference/SPI>. Acesso em 10 de nov. de 2013.

2. Tutorial: Comunicação SPI (Serial Peripheral Interface) com Arduino. Disponível em <http://labdegaragem.com/profiles/blogs/tutorial-comunica-o-spi-serial-peripheral-interface-com-arduino>. Acesso em 6 de nov. de 2013.

3. Extended SPI library usage with the Due. Disponível em <http://arduino.cc/en/Reference/DueExtendedSPI>. Acesso em 14 de nov. de 2013.