

G+1 0

mais

Próximo blog»

welbertdeoliveira@gmail.com

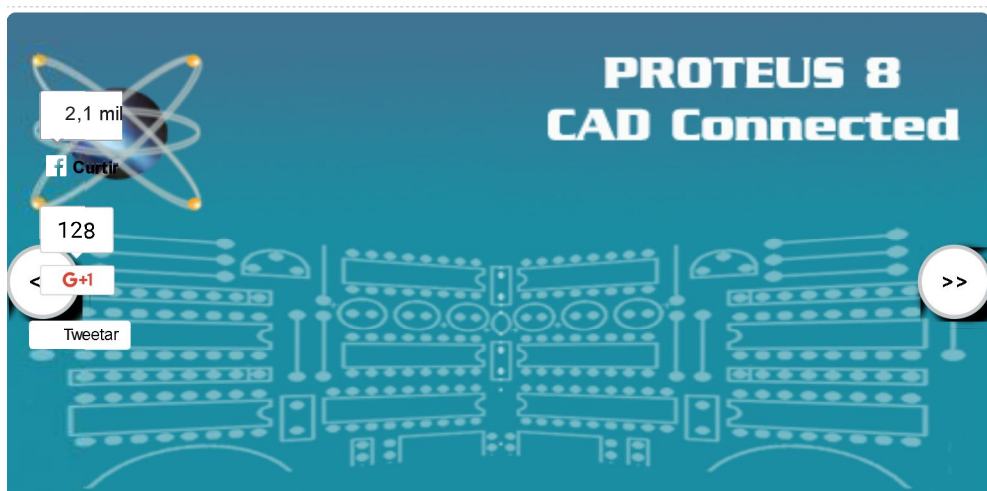
Painel

Sair

Projetos e tutoriais sobre programação de PIC, AVR e Arduino.

Início

Fórum



Proteus 8.1

[1](#) [2](#) [3](#) [4](#) [5](#)

Microsoft

Microsoft Azure
App Service

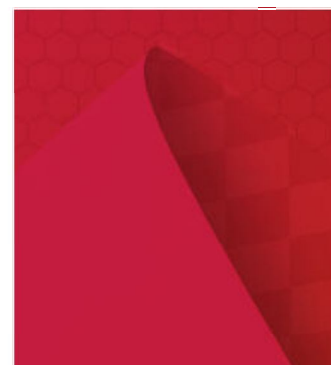
Área restrita gratuita.
Sem complicações.

Crie um
aplicativo Web.
E explore novas
possibilidades.

5/03/13

Comunicação SPI

Comunicação SPI (Serial Peripheral Interface
)



PESQUISAR NESTE BLOG

NOVIDADES DO BLOG - 05/06

PIC: MCP3201

PIC: DS1631

AVR: Display 7 Segmentos

AVR: Pull-ups

AVR: Pisca-Pisca com ATmega328

Curso: Aula 03 - Variáveis, vetores e fu

PIC: 25XX320

PIC: RDA5807

PIC: SSD1306

Curso: Ementa Atualizada

Curso: Aula 02 - Portas I/O

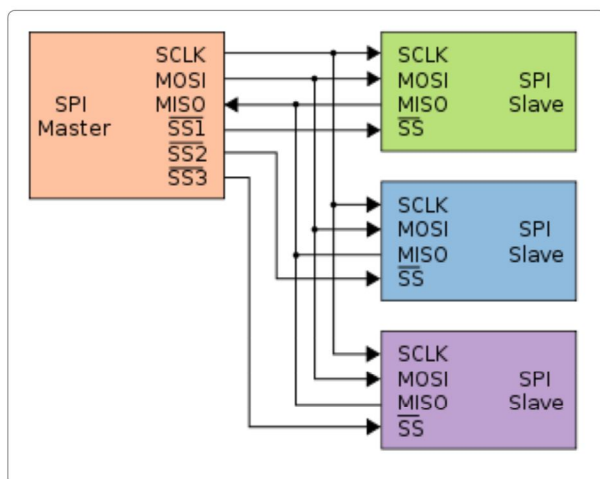
PIC: PCF8583

Curso: Aula 01

Atualização de Código: DHT11

Tutorial: Ellipsis

PIC: MCP4XXX



Serial Peripheral Interface é um protocolo de dados seriais síncronos utilizado em microcontroladores para comunicação entre o microcontrolador e um ou mais periféricos. Também pode ser utilizado entre dois microcontroladores.

Comunicação SPI sempre tem um Master, isto é, sempre um será o Master e o restante será Slave. Por exemplo, o PIC é o Master e os outros periféricos são Slave. Esta comunicação contém 4 conexões:

- **MISO** (*Master IN Slave OUT*) - Dados do Slave para Master;
- **MOSI** (*Master OUT Slave IN*) - Dados do Master para Slave;
- **SCK** (*Serial Clock*) - Clock de sincronização para transmissão de dados entre o Master e Slave;
- **SS** (*Slave Select*) - Seleciona qual escravo receberá os dados.

Muitos periféricos são apenas Slave, por exemplo, cartão SD, memória flash e alguns sensores. Normalmente estes periféricos contêm a mesma pinagem que acima ou a pinagem abaixo:

- **SDI** (*Slave Data IN*) - Pino de dados de entrada;
- **SDO** - (*Slave Data OUT*) - Pino de dados de saída;
- **CS** - Seleção de Chip;
- **SCK** - Clock de sincronização.

Desvantagens:

- Comunicação full duplex;
- Flexibilidade protocolo completo para os bits transferidos;
- Não se limita a 8-bit palavras;
- Escolha arbitrária do tamanho da mensagem, conteúdo e finalidade Interface de hardware extremamente simples;
- Normalmente mais baixos requisitos de energia do que I2C devido a menos circuitos;
- Não necessita de resistores de pull-up;
- Escravos usar relógio do mestre, e não precisam de osciladores de precisão;
- Os escravos não precisam de um endereço único;

Vantagens:

- Requer mais pinos do CI;
- Sinais de out-of-band Chip Select são obrigatórios nos barramentos comuns;
- Sem controle de fluxo de hardware pelo escravo (mas o mestre pode atrasar o clock seguinte para diminuir a taxa de transferência);
- Não há reconhecimento do escravo (o mestre poderia estar transmitindo a lugar nenhum e não

Fonte: wikipedia

TRANSLATE

Selecione o idioma

Powered by [Google Tradutor](#)

MARCADORES

[Apostilas e Livros](#) (3)

[Arduino](#) (11)

[AVR](#) (12)

[Bibliotecas](#) (39)

[Circuitos](#) (3)

[Componentes](#) (1)

[Curso](#) (4)

[Downloads](#) (10)

[Games](#) (2)

[Ladder](#) (1)

[PIC](#) (79)

[Projetos](#) (47)

[Sensores](#) (9)

[Tutoriais](#) (47)

[Wireless](#) (2)

SEGUIR-ME POR EMAIL

Email address...

Sub

POSTAGENS DO BLOG

► [2015](#) (7)

► [2014](#) (34)

▼ [2013](#) (60)

► [Dezembro](#) (6)

► [Novembro](#) (3)

► [Outubro](#) (1)

► [Setembro](#) (6)

► [Agosto](#) (2)

► [Julho](#) (7)

► [Junho](#) (9)

► [Maio](#) (5)

► [Abril](#) (4)

▼ [Março](#) (8)

[PIC: Configurando o Oscilador Interno](#)

[Conversor A/D](#)

[Snake Game com PIC16F877A](#)

[PIC: Mouse e Teclado USB](#)

[Supervisão para Arduino](#)

[PIC: DataLogger com PIC18F](#)

[Arduino: Matriz de LEDs](#)

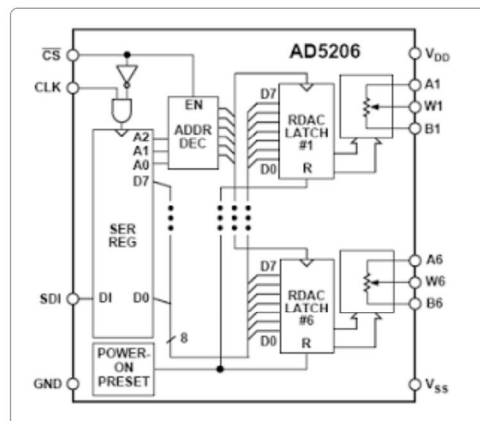
saberíamos);

- Suporta apenas um dispositivo mestre;
- Nenhum protocolo de verificação de erros é definido;
- Geralmente propenso a causar picos de ruído em comunicação defeituoso;
- Sem um padrão formal, validar a conformidade não é possível;
- Lida apenas com curtas distâncias em relação ao RS-232, RS-485, ou CAN-BUS;
- Muitas variações existentes, o que torna difícil encontrar ferramentas de desenvolvimento, como adaptadores de host que suportam essas variações;

Fonte: wikipedia

mplo:

exemplo controla o potenciômetro Digital AD5206. Possui 6 canais, isto é, 6 potenciômetros. Possui os seguintes pinos: A1, W1, B1 ... A6, W6, B6.



ligo: Exemplo com PIC

MikroC PRO PIC

```

01.  sbit CS0 at RC0_bit;
02.  sbit CS0_Direction at TRISC0_bit;
03.
04.
05.  void AD5206_write(short channel, short value){
06.  CS0 = 0; //seleciona o periferico
07.  SPI1_Write(channel); //envia o endereço de um dos potenciômetros(0 - 5)
08.  SPI1_Write(value); //envia o valor de posição do potenciômetro(0 - 255)
09.  CS0 = 1; //deseleciona o periferico
10.  }
11.
12.  void main() {
13.  int i;
14.  CS0_Direction = 0; //define pino cs como saída
15.  CS0 = 1;
16.
17.  SPI1_Init();
18.
19.  while(1){
20.  //este loop faz alterar o valor do potenciômetro do canal 0
21.  for(i=0; i<255; i++){
22.  AD5206_Write(0, i); //incrementa de 0 a 255
23.  delay_ms(100);
24.  }

```

Comunicação SPI

- Fevereiro (3)
- Janeiro (6)
- 2012 (48)

TOP SEMANAL



Proteus 8.1 Professional



Arduino Uno para Proteus



MikroC PRO for PIC, dsPIC, PIC3, 8051, AVR



Livros Sobre Programação



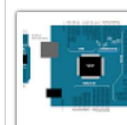
Apostilas e Livros sobre Microcontrolador



Livros sobre Programação com Arduino



Utilizando o TIME do PIC



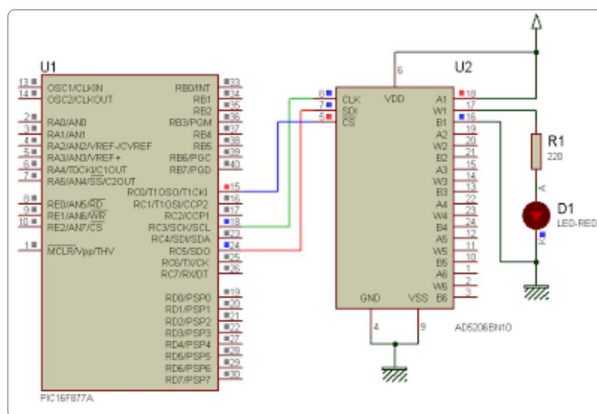
Arduino - Componente para Proteus

SEGUIDORES

```

25.
26. //este loop faz alterar o valor do potenciometro do canal 0
27. for(i=255; i>=0; i--){
28.     AD5206_Write(0, i); //decrementa de 255 a 0
29.     delay_ms(100);
30. }
31. }
32. }

```



igo: Exemplo com Arduino

```

#include <SPI.h>

int slaveSelectPin = 10;

void setup(){
  pinMode(slaveSelectPin, OUTPUT);
  SPI.begin();
}

void loop(){
  for(i=0; i<255; i++){
    digitalPotWrite(0, i);
    delay_ms(100);
  }

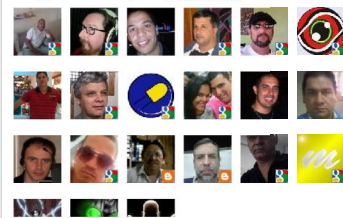
  for(i=255; i>=0; i--){
    digitalPotWrite(0, i);
    delay_ms(100);
  }

  void digitalPotWrite(int address, int value) {
    digitalWrite(slaveSelectPin, LOW);
    SPI.transfer(address);
    SPI.transfer(value);
    digitalWrite(slaveSelectPin, HIGH);
  }
}

```

Arduino IDE

Seguidores (134) Próxima



CONTATO

[microcontrolandos2012@gmail.c](mailto:microcontrolandos2012@gmail.com)

PERFIL

Tiago Melo - Graduando em Ciênc
Computação na Universidade Fedei
Itajubá - UNIFEI e Técnico em Eletr
pelo SENAI.

Poderá também gostar de:




Comunicação I2C	PIC: I2C Slave	TUTORIAL: Comunicação USB no MikroC	Utilizando o TIMER1 do PIC	Conversor A/D
---------------------------------	--------------------------------	---	--	-------------------------------

[Link within](#)

Tweetar

 0



 Recomeinde isto no Google

6 comentários:

 **Vittor** segunda-feira, 6 de janeiro de 2014 14:22:00 BRST

Esse código foi feito em qual programa?

[Responder](#)


Respostas



Tiago Henrique segunda-feira, 6 de janeiro de 2014 14:30:00 BRST

O primeiro foi feito no **MikroC PRO PIC** e o outro foi na IDE do Arduino.

[Responder](#)

 **Carolina** quarta-feira, 19 de março de 2014 16:15:00 BRT

Olá, no código em C, onde estão as rotinas SPI1_Init e SPI1_Write? Obrigada

[Responder](#)

Respostas



Tiago Henrique quarta-feira, 19 de março de 2014 16:22:00 BRT

SPI1_Write e SPI1_Init são da biblioteca SPI, do próprio MikroC PRO PIC.



Carolina domingo, 27 de abril de 2014 19:53:00 BRT

vc tem alguma coisa semelhante pro compilador HITECH? Obrigada pela ajuda

[Responder](#)

Anônimo quarta-feira, 24 de dezembro de 2014 12:50:00 BRST

Olá Tiago será que você pode me ajudar com o CI ISD1730? é um ci para gravação e reprodução de

faixas de áudio, ele trabalha com comunicação SPI, porém não estou conseguindo fazer o microcontrolador acessar a faixa de memória exata que gravei a mensagem e no tempo certo de reprodução. meu e-mail é henrique-sousa1994@hotmail.com

[Responder](#)

Digite seu comentário...

Comentar como:

☐ Notifique-me



nar: [Postar comentários \(Atom\)](#)