PSI3441 – Arquitetura de Sistemas Embarcados

17ª Aula

Introdução: CPUs, Interrupções e Gerenciamento de Memória

PSI3441 – Conteúdo 2ª Prova

Parte 2 – Considerando os Conjuntos de Instruções

- Taxonomia de arquitetura de computadores e de linguagem assembly
- Exemplos de 4 arquiteturas: ARM, PIC16F, TI C55x e TI C64x

Parte 3 - Considerando CPUs:

- Quais os mecanismos de Entrada e Saída (I/O)?
- O que são modo supervisor (supervisório?), exceções e armadilhas (traps)
- Como é feito o mapeamento de memória e codificação (translation) de endereços?
- O que são Caches?
- Como determinar e especificar desempenho e consumo de potência de CPUs?
- Exemplo de projeto: Compressor de Dados

Material adicional referente à última aula

- ARM University program https://www.arm.com/support/university/index.php
- Sugestões de vídeos:

Vídeo que cobre vários tópicos que abordei na última aula

- https://youtu.be/JH4j7fCT_o4

Arm Academy

- https://youtu.be/7LqPJGnBPMM

O responsável pela microarquitetura do primeiro ARM

- https://youtu.be/_VYxIaw1kBU

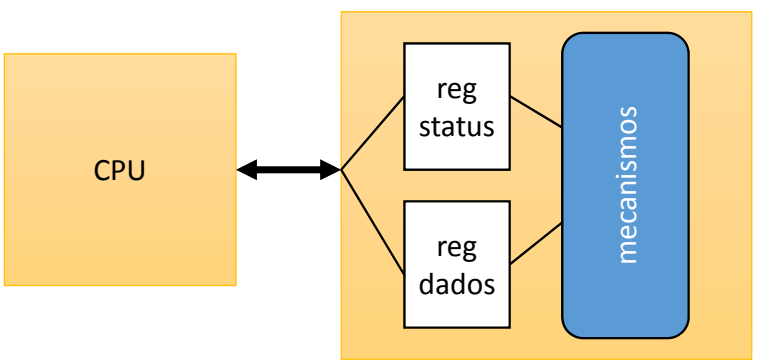
A responsável pela ISA do primeiro ARM

- https://youtu.be/_9mzmvhwMqw

Parte B: Entrada-Saída

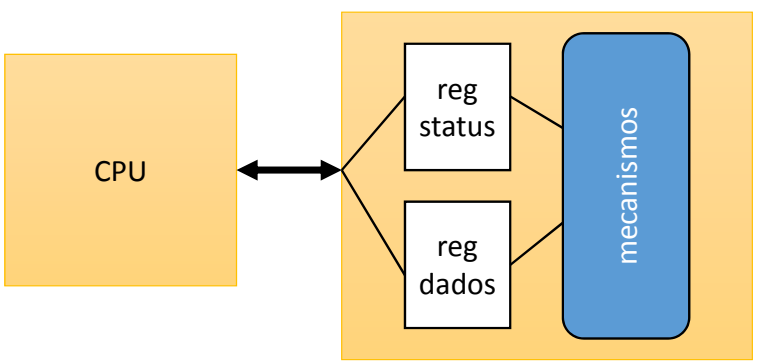
3.2 Dispositivos de I/O (Entrada/Saída)

- Dispositivos de entrada e saída usualmente contam com algum componente analógico ou não-eletrônico:
  - Um disco rígido possui um disco em rotação e eletrînica analógica de escrita/leitura
  - Por outro lado, a lógica digital mais próxima da CPU segue as os mesmos conceitos da lógica em sistema computacional
- Estrutura típica de um dispositivo I/O:
- A interface entre a CPU e o os elementos internos do I/O tratam-se de registradores

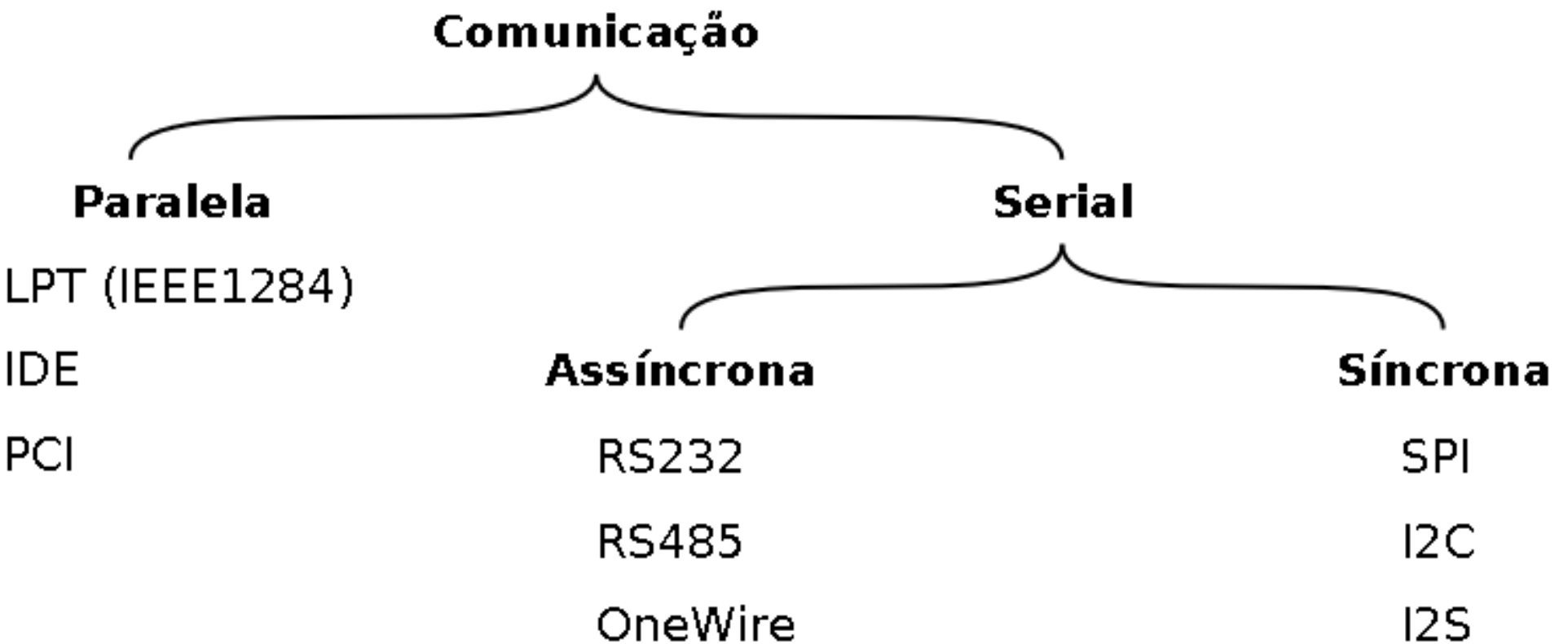


3.2 Dispositivos de I/O (Entrada/Saída)

- A CPU conversa com o dispositivo lendo e escrevendo nos registradores. Dispositivos de I/O possuem tipicamente muitos tipos de registradores:
  - Registradores de Dados armazenam valores que são tratados como dados pelo dispositivos, como dados para escrita/leitura
  - Registradores de Status fornecem informações sobre a operação do dispositivo, p.ex. se a operação atual terminou. Alguns registradores podem ser de apenas leitura, como registradores de status que indicam quando o dispositivo terminou.
- Novamente, normalmente inclui componentes não digitais



Comunicação Paralela e Serial



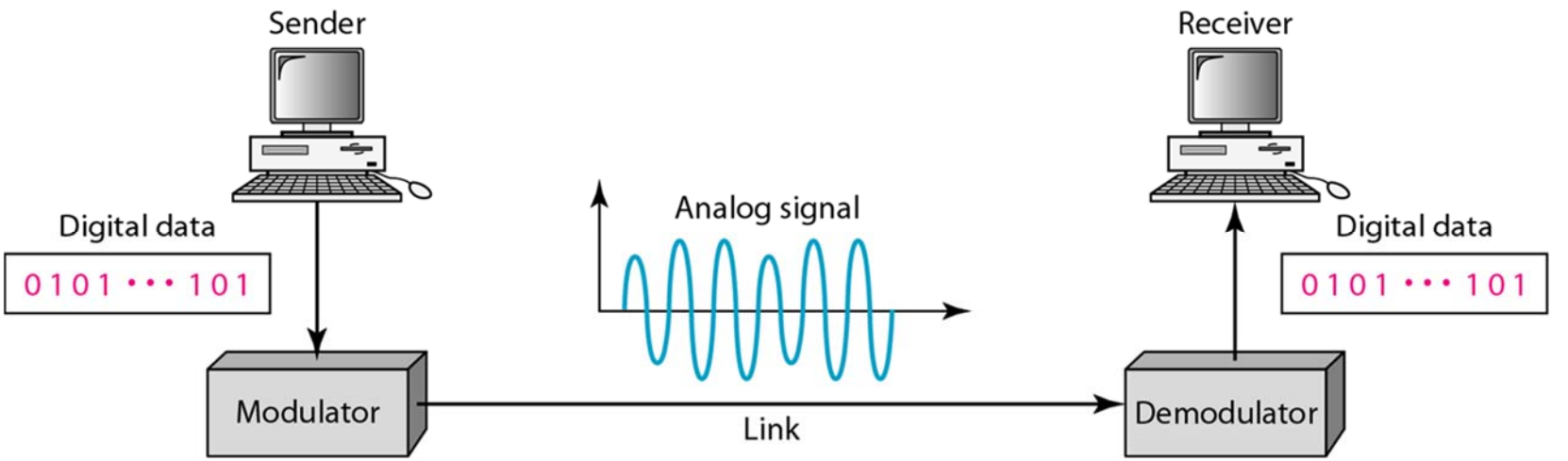
https://www.embarcados.com.br/

Comunicação Paralela e Serial

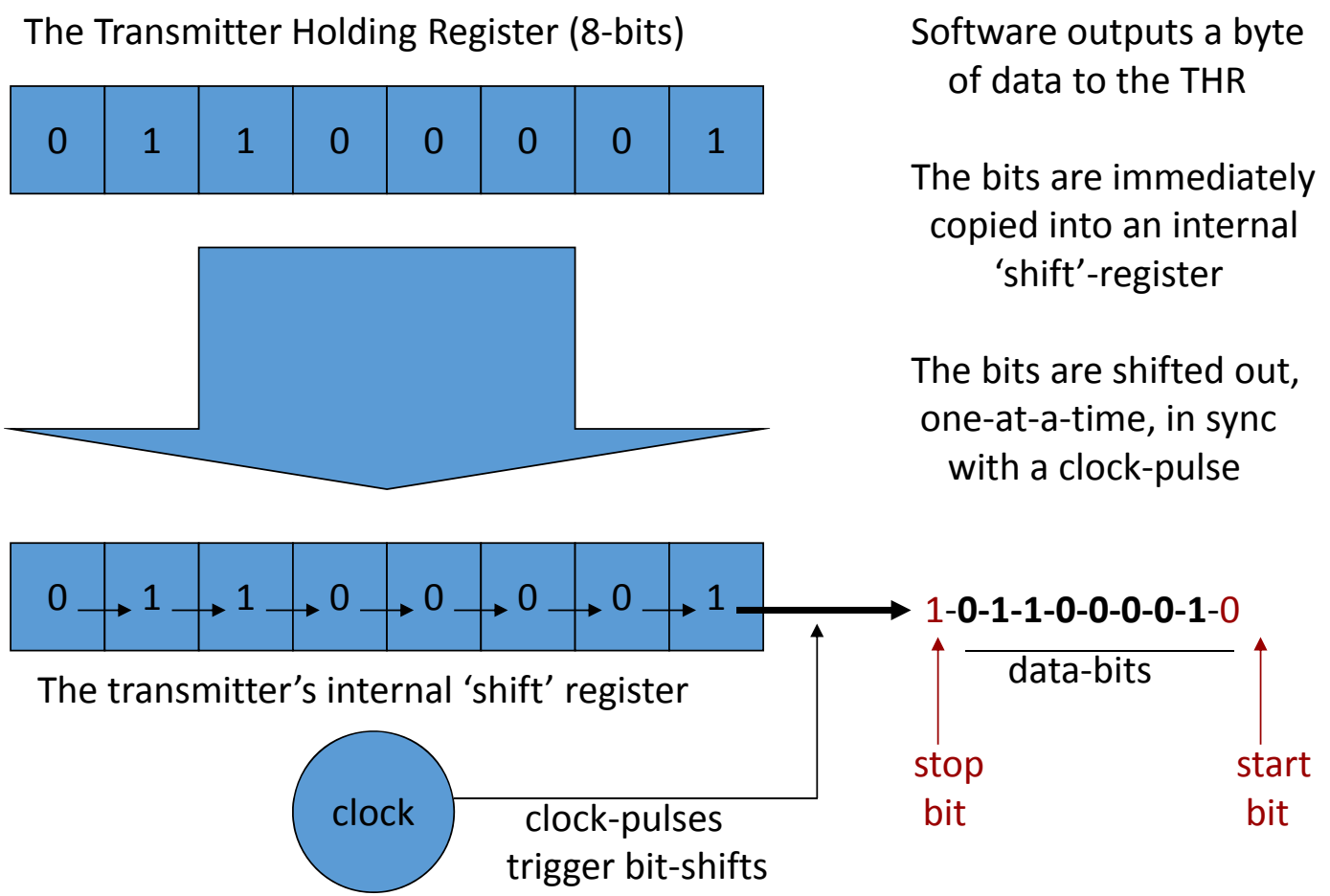
Tecnologia	Barramento de comunicação	Taxa máxima	Fluxo de dados
UART (RS232)	2 (sem controle de fluxo)	115.200 bps	Half ou Full Duplex
SPI	3 + nº de Slaves	2 Mbps	Full Duplex
I2C	2 (até 127 dispositivos)	400 Kbps	Half Duplex

https://www.embarcados.com.br/

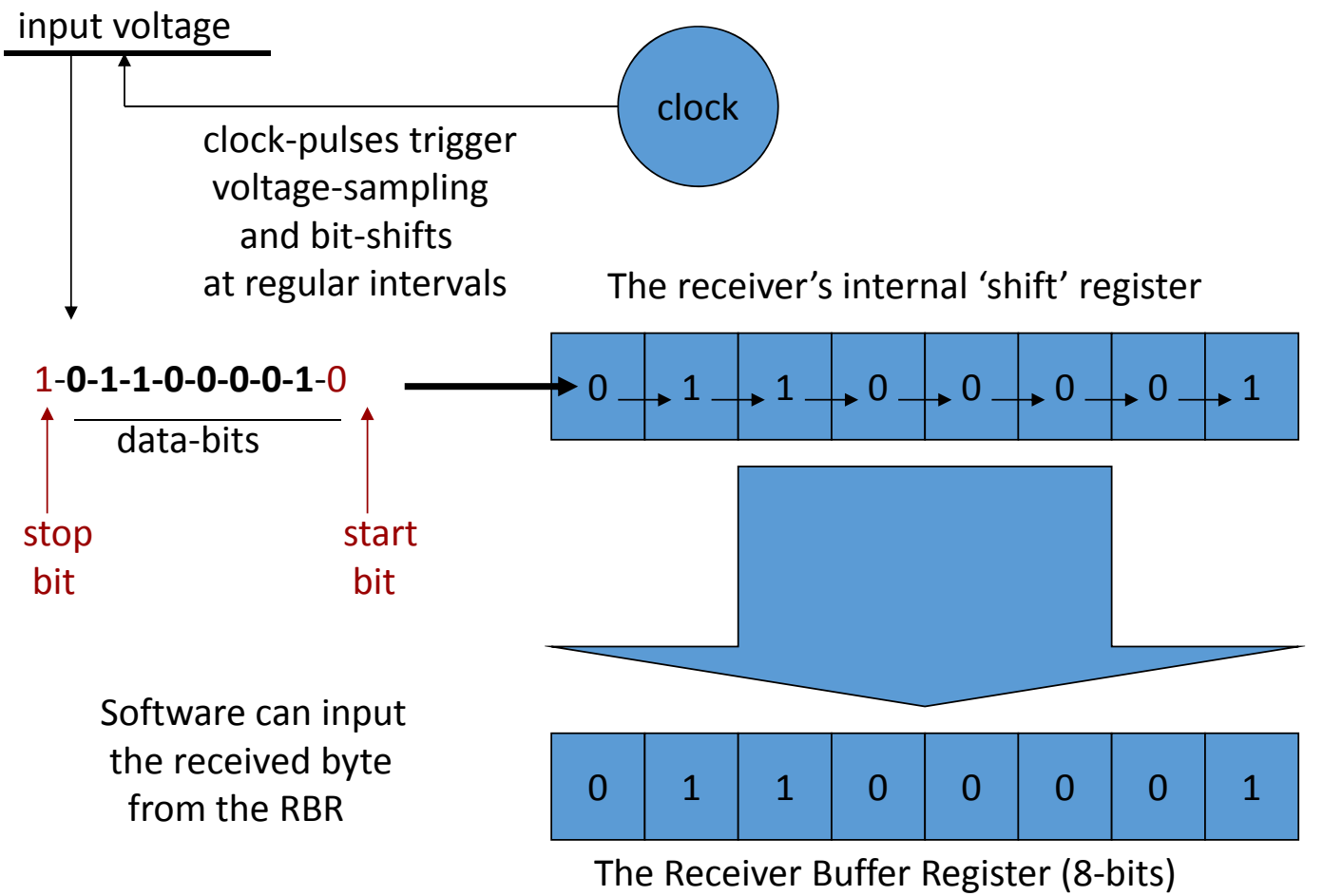
Recapitulação de Comunicação



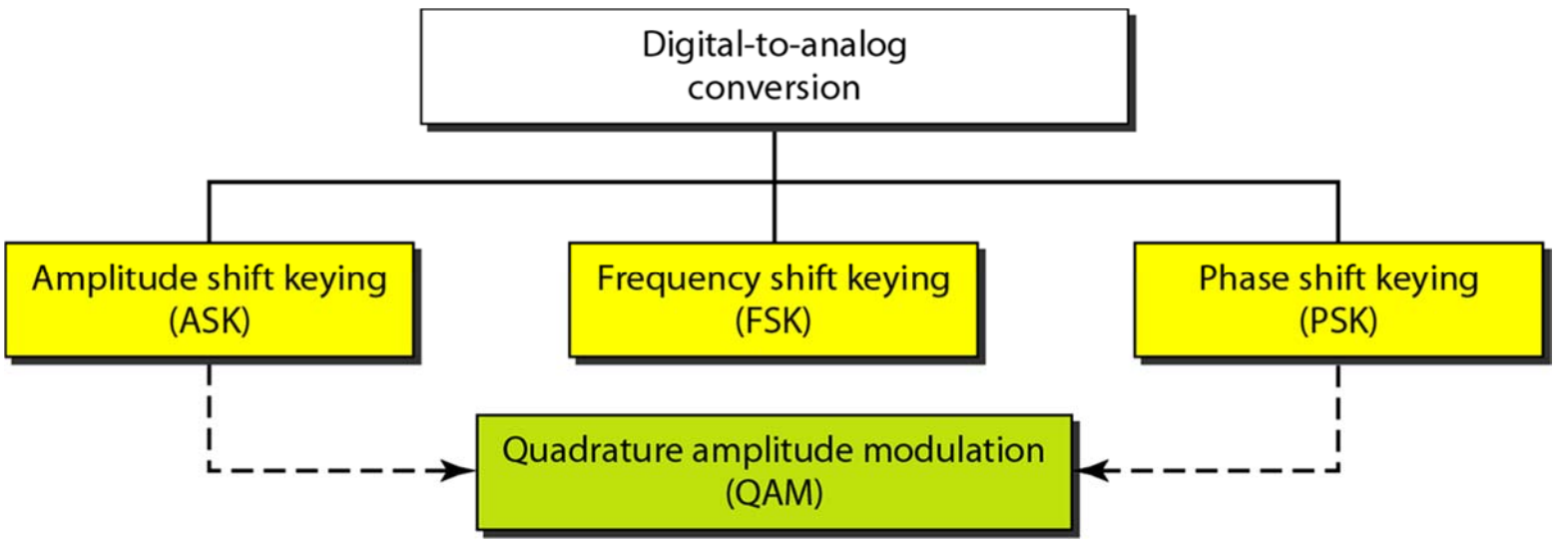
Transmissão Serial



Recepção Serial



Recapitulação de Comunicações



Recapitulação: Comunicação Serial

- Taxa de Símbolos
  - Taxa de modulação em símbolos por segundo
- Baud Rate – Bauds/s ou simplesmente ***bauds***
  - Émile Baudot
- Taxa de bits (Bit Rate) – bit/s
- Bit Rate = (Baud Rate) x (número de bits por baud)

- Exemplo
  - FSK: 1 baud corresponde a 1 bit
  - PSK: 1 baud corresponde a 4 bits

Exercício

- Um sinal analógico tem um ***bit rate*** de 8000 bps e um ***baud rate*** of 1000 baud. Quantos elementos de dado são transportados por element de sinal? Quantos elementos de sinal precisamos?

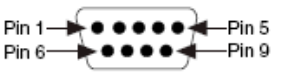
$$S = N \times \frac{1}{r} \quad \rightarrow \quad r = \frac{N}{S} = \frac{8000}{1000} = 8 \text{ bits/baud}$$

$$r = \log_2 L \quad \rightarrow \quad L = 2^r = 2^8 = 256$$

Comunicação Serial

- RS-232
- I2C
- SPI

RS-232

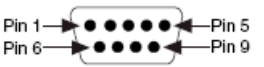


Data	
TXD (pin 3)	Serial Data Output
RXD (pin 2)	Serial Data Input
Handshake	
RTS (pin 7)	Request to Send
CTS (pin 8)	Clear to Send
DSR (pin 6)	Data Set Ready
DCD (pin 1)	Data Carrier Detect
DTR (pin 4)	Data Terminal Ready
Ground	
GND (pin 5)	Ground
Other	
RI (pin 9)	Ring Indicator

Table 1: Pin Functions for RS-232

RS-485 e RS-422

DB-9 Male



Data	
TXD+ (pin 8)	Serial Data Output (differential)
TXD- (pin 9)	Serial Data Output(differential)
RXD+ (pin 4)	Serial Data Input(differential)
RXD- (pin 5)	Serial Data Input(differential)
Handshake	
RTS+ (pin 3)	Request to Send (differential)
RTS- (pin 7)	Request to Send (differential)
CTS+ (pin 2)	Clear to Send (differential)
CTS- (pin 6)	Clear to Send (differential)
DSR (pin 6)	Data Set Ready
Ground	
GND (pin 1)	Ground

Table 2: Pin Functions for RS-485 and RS-422

RS-232, RS-422 e RS-485

Specifications	RS-232	RS-422	RS-485
Mode of Operation	Single-Ended	Differential	Differential
Number of Drivers / Receivers on One Line	1 Driver 1 Receiver	1 Driver 10 Receivers	32 Drivers* 32 Receivers
Maximum Cable Length	50 ft (2500 pF)	4000 ft	4000 ft
Maximum Data Rate (at max cable length)	160 kbits/s (can be up to 1Mbit/s)	10 Mbit/s	10 Mbit/s

Table 3: Specifications of RS-232, RS-422, and RS-485

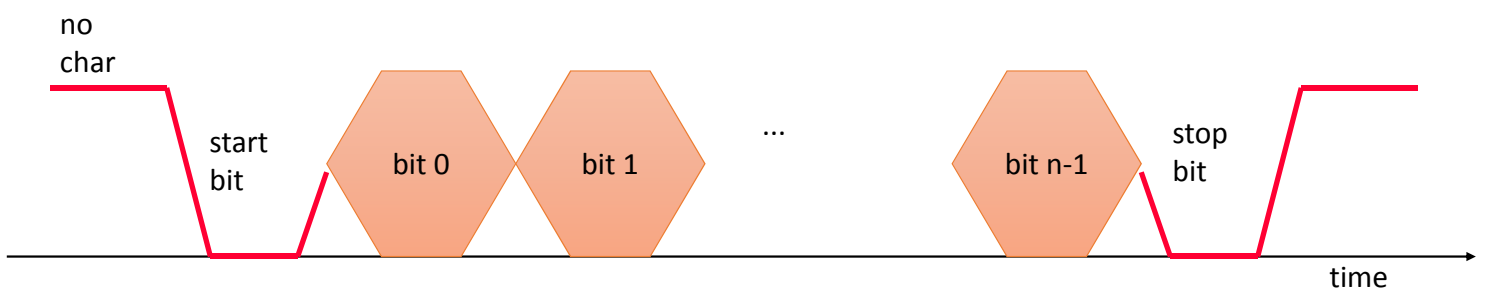
*Only one driver is active at a time

Aplicação: UART 8251

- Universal asynchronous receiver transmitter (UART) : Transmissor/receptor assíncrono universal: possibilita comunicação serial
- As funções do 8251 estão integradas em chips de interface padrão para PCs
- Permite que vários parâmetros de comunicação sejam programados

Comunicação Serial

- A USART 8251 (Universal Asynchronous Receiver/Transmitter) [Int82] é o dispositivo original para uso em comunicação serial, como as portas seriais de um PC. O 8251 foi introduzido como um chip separado para uso com os primeiros processadores, mas hoje suas funções estão embutidas em chips maiores.
- Apesar disso, esses chips mais sofisticados utilizam os padrões básicos definidos pelo 8251
- Neste caso os caracteres são transmitidos separadamente:



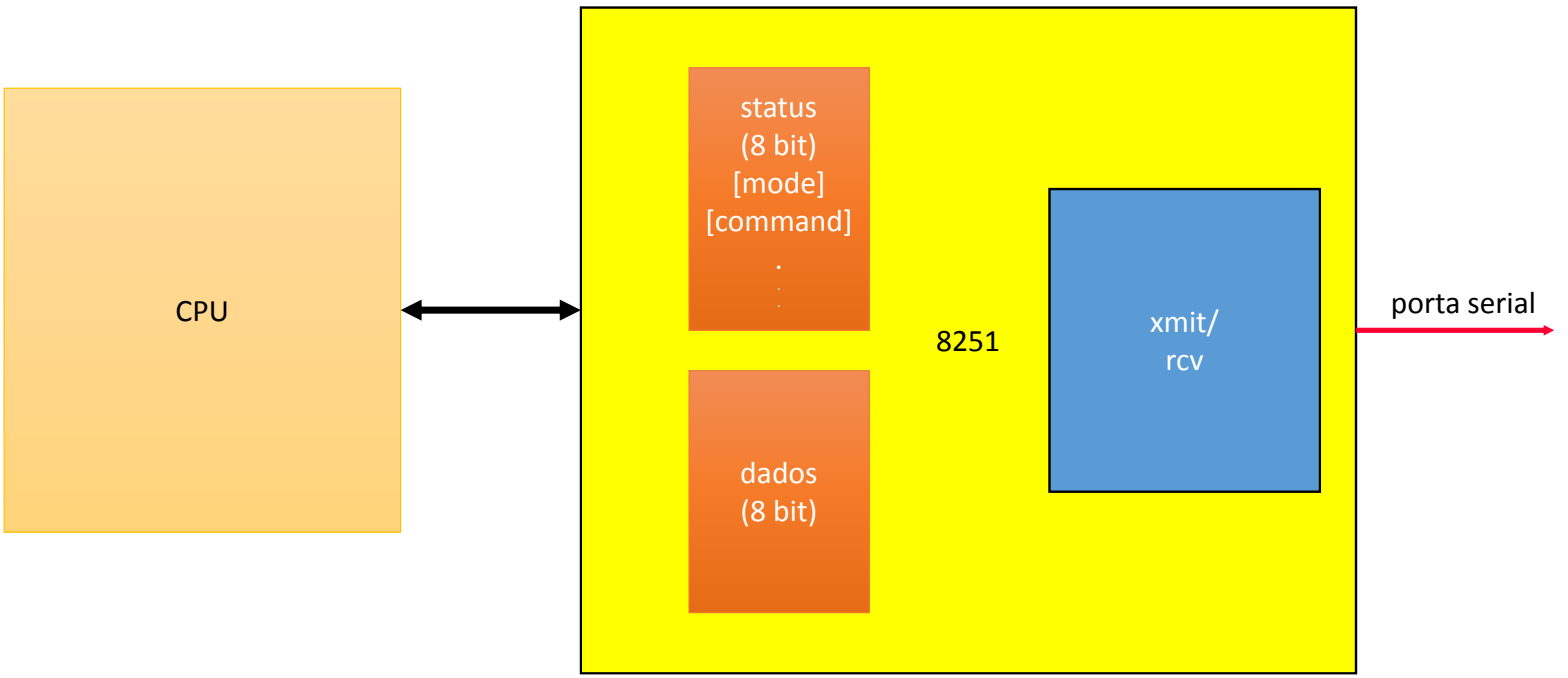
Parâmetros para Comunicação Serial

- Baud (bit) rate
- Número de bits por caracter
- Paridade/não paridade
- Paridade Impar/Par
- Comprimento do stop bit (1, 1,5, 2 bits)

Questões

- Paridade par e Paridade ímpar

Interface do 8251 com a CPU



Bits de modo (mode) da Porta Serial

Antes de enviar/receber dados a CPU precisa configurar o registrador de modo para corresponder aos dados que estão na linha de dados:

- mode[1:0]: baud rate
  - 00: synchronous
  - 01: async, no clock prescaler
  - 10: async, 16x prescaler
  - 11: async, 64x prescaler
- mode[3:2]: bits/char
  - 00: 5 bits
  - 01: 6 bits
  - 10: 7 bits
  - 11: 8 bits.
- mode[5:4]: parity
  - 00, 10: no parity
  - 01: odd parity
  - 11: even parity
- mode[7:6]: stop bit
  - 00: invalid
  - 01: 1 stop bit
  - 10: 1.5 stop bits
  - 11: 2 stop bits

Registrador de comando da Porta Serial

Dizem à UART o que fazer:

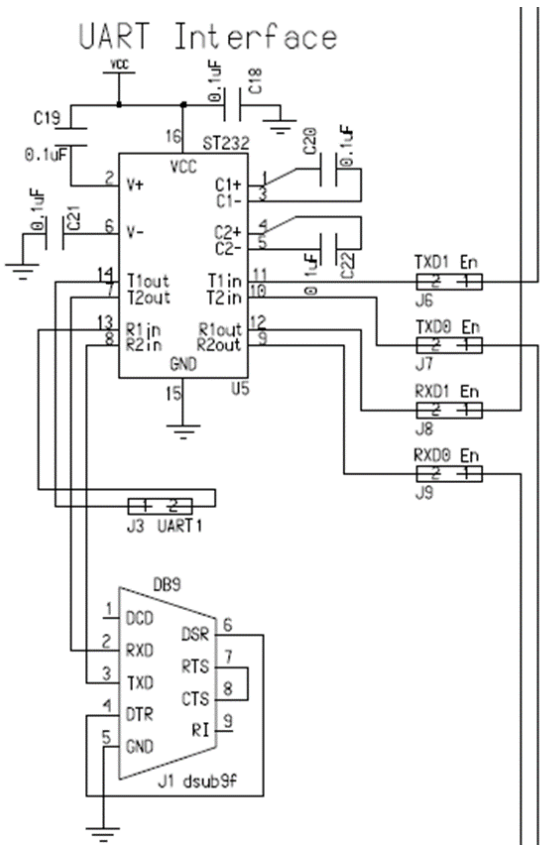
- mode[0]: transmit enable
- mode[1]: set nDTR output
- mode[2]: enable receiver
- mode[3]: send break
- mode[4]: reset error flags
- mode[5]: set nRTS
- mode[6]: internal reset
- mode[7]: hunt mode

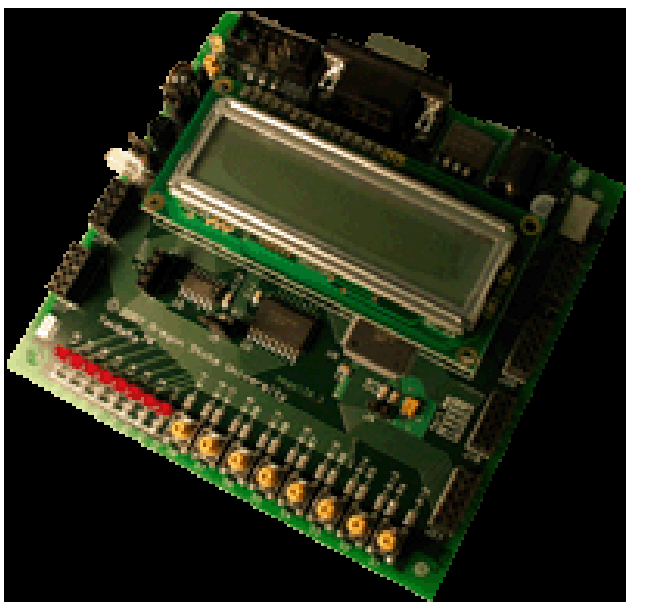
Registrador de status da Porta Serial

Indicam o estado da UART e da Comunicação:

- status[0]: transmitter ready
- status[1]: receiver ready
- status[2]: transmission complete
- status[3]: parity
- status[4]: overrun
- status[5]: frame error
- status[6]: sync char deleted
- status[7]: nDSR value

MAX232 na placa MEGA128





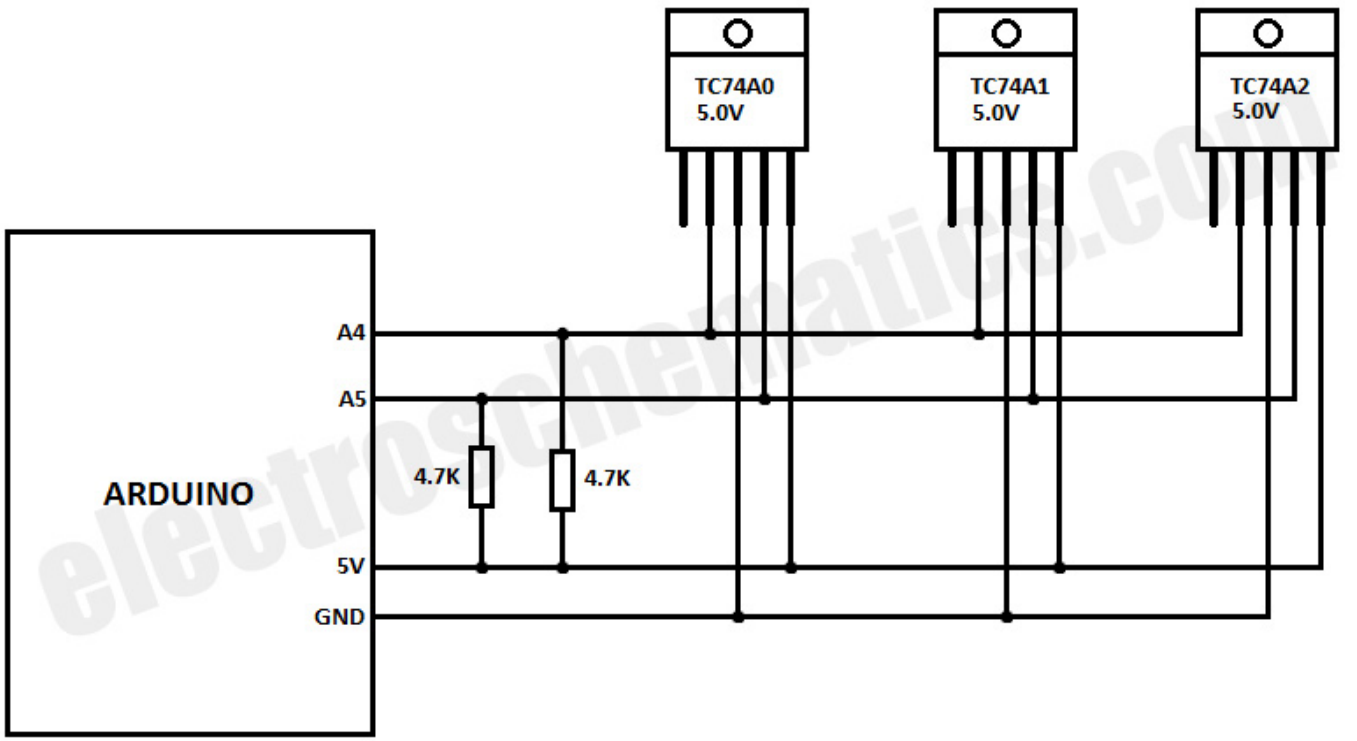
Padrão Inter-Integrated Circuit (I^2C)

- Barramento de Comunicação Serial de 2-Fios (2-Wire Serial Communication Bus)
- Introduzido pela Philips em 1992
- Communicação entre Microcontrolador e Periféricos
  - Real Time Clocks (RTCs)
  - Analog to Digital Converters (ADCs)
  - Various Sensors
  - Many, many, more

Características do I^2C

- Apenas 2 Fios
  - Serial Data (SDA)
  - Serial Clock (SCL)
- Múltiplos Dispositivos conectados no Bus
  - Tipicamente limitado (i.e. 8, 20, 2^7, 2^{10},...?)
- Código similar para todos os periféricos I^2C
- Suporta Várias Taxas de Transferência de Dados

Exemplo



Como Funciona?

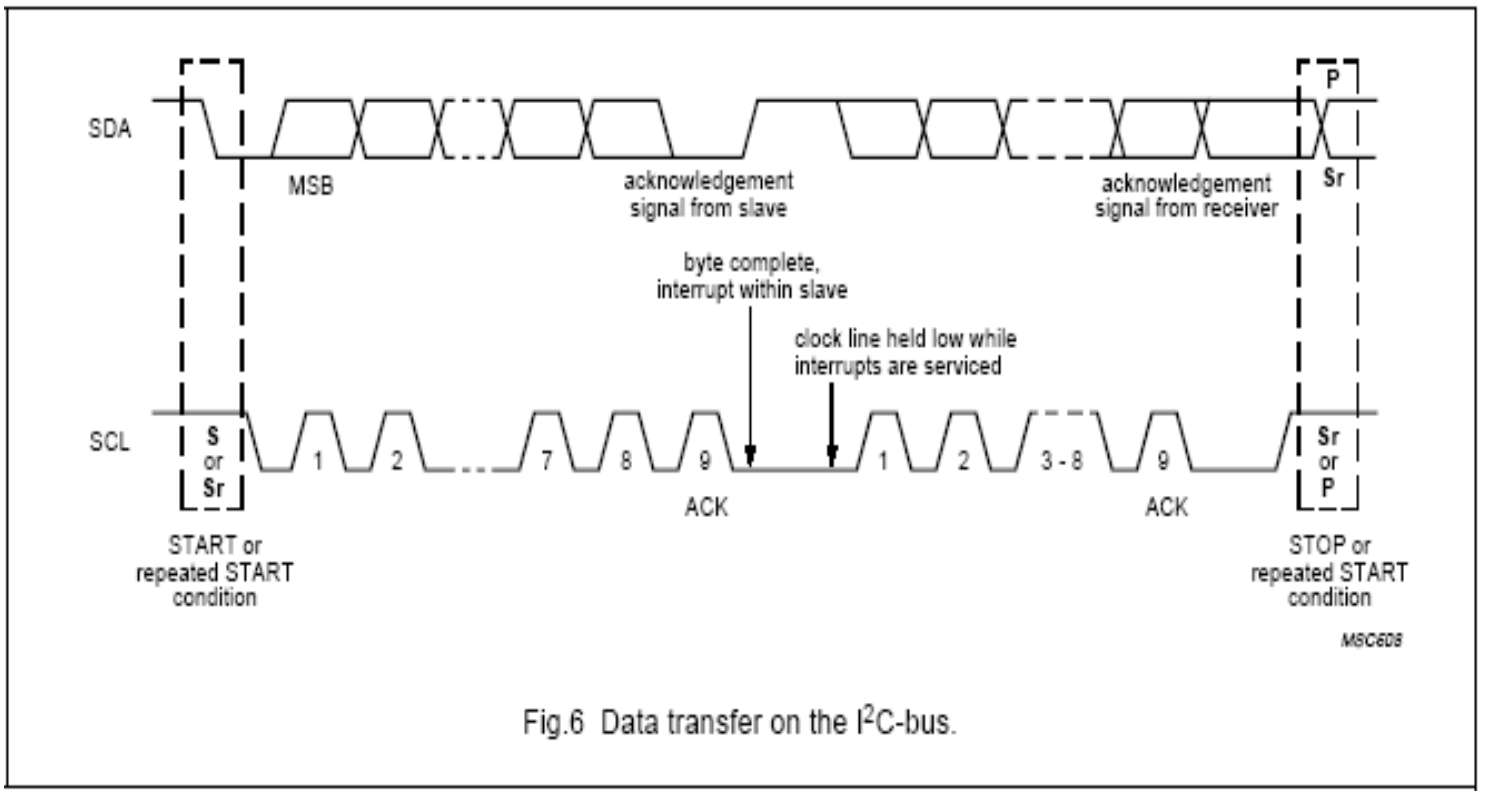
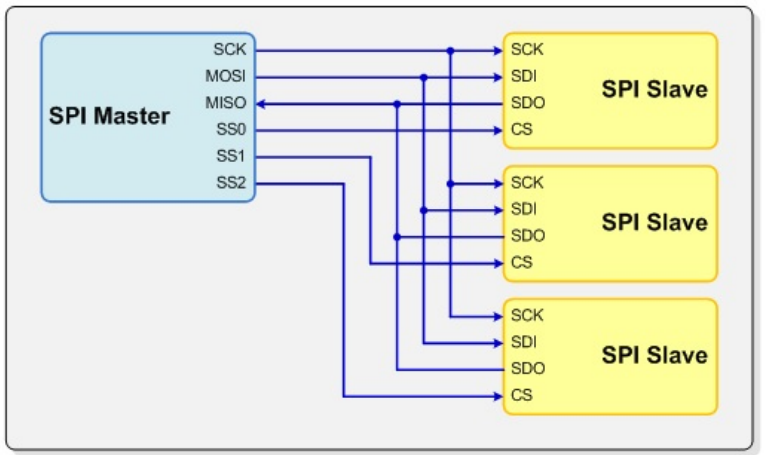


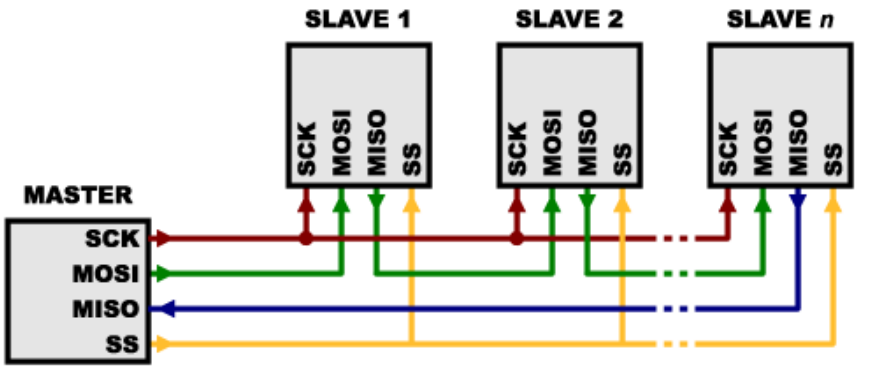
Fig.6 Data transfer on the I^2C-bus.

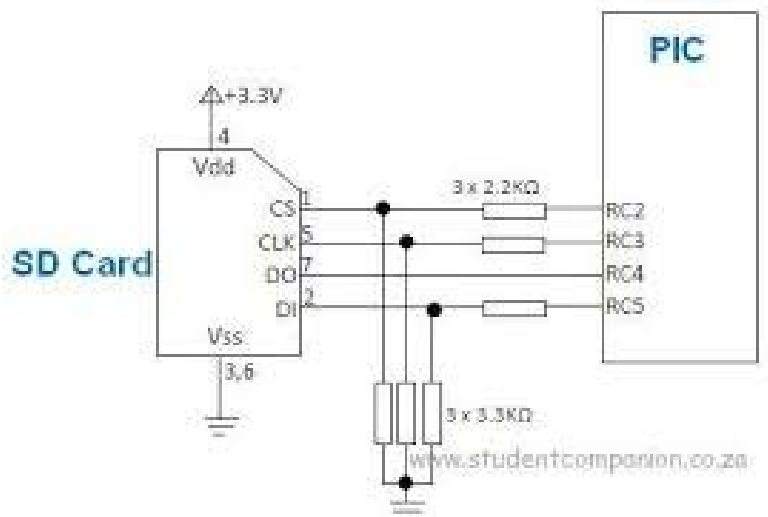
Padrão SPI - Interface de Periférico Serial(Serial Peripheral Interface (SPI)

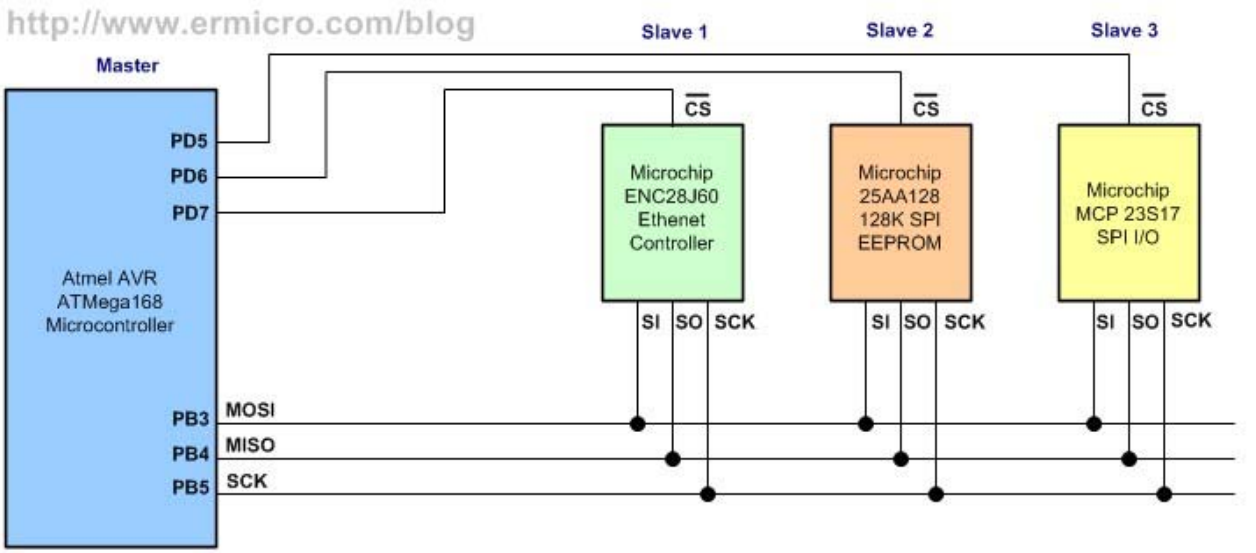
- Protocolo de Comunicação Síncrono
  - 3-Fios (Plus 1 Chip-Select Pin/Device)
  - Hardware no Microcontrolador
- Desenvolvido pela Motorola
- Larga Faixa de Dispositivos Suportados
  - Memoria (i.e. EEPROM, RAM, Etc.)
  - Sensores

Exemplo





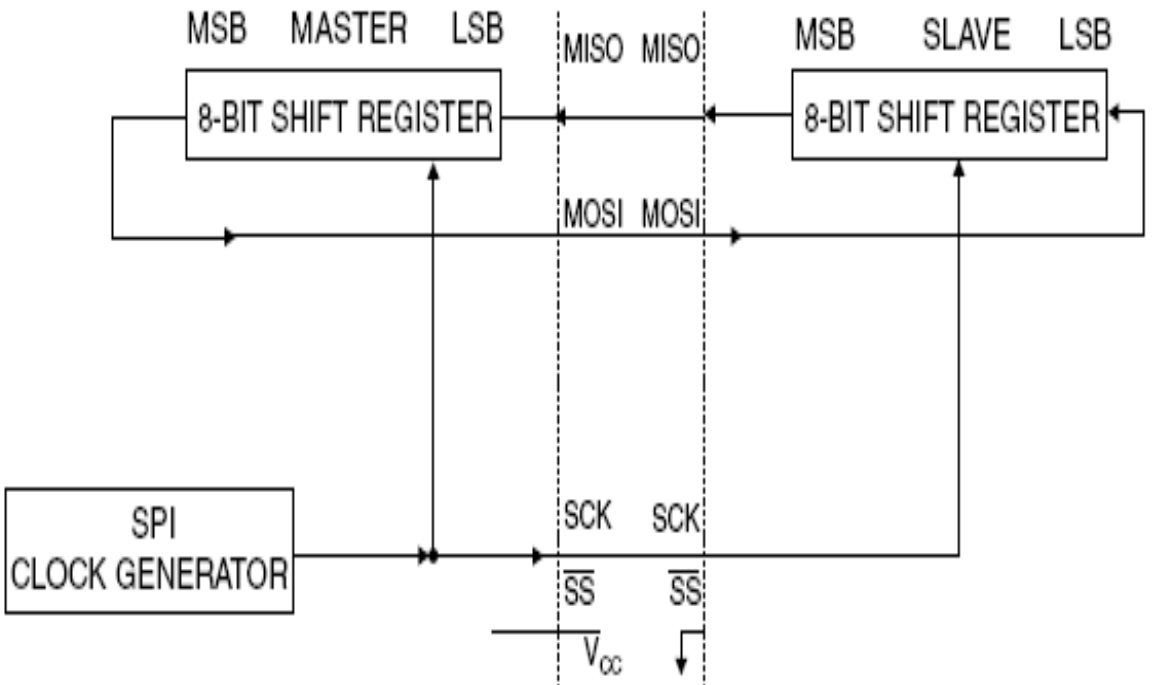




Typical SPI Master with Multiple SPI Slave Device Connection

Como Funciona?

Figure 7. SPI Master-slave Interconnection



Características do SPI

- 3 Fios (Plus Chip Select Pin/Device)
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - Master/Slave Clock Output/Input (SCK)
  - Chip/Slave Select (SS)
- Multiplos Dispositivos Conectados no Barramento
  - Limitedo apenas pelo número de pinos do uC
- Suporta Várias Taxas de Transferência de Dados

Parte C: Programação de E/S

3.2.2 Programação de I/O

- Dois tipos de instruções aceitam I/O:
  - Instruções específicas de I/O;
  - Instruções para mapeamento de memória carrega/armazena (load/store)
- Intel x86 possui instruções `in, out` de propósito definido. A maioria das CPUs utilizam I/O mapeado em memória.
- Instruções específicas de I/O convivem com I/O mapeado em memória.

Questão

- Entrada-Saída mapeada em Memória - Memory mapped I/O
- Entrada-Saída mapeada em porta - Port-mapped I/O (port I/O)

Exemplo 3.1: I/O mapeado em memória para o ARM

- Defina a localização do dispositivo:

```
DEV1 EQU 0x1000
```

- Código de Escrita/Leitura:

```
LDR r1,#DEV1 ; set up device address
LDR r0,[r1] ; read DEV1
LDR r0,#8 ; set up value to write
STR r0,[r1] ; write 8 to device
```

Peek e poke

- Como escrever I/Os diretamente em linguagens de alto nível (HLL) como C, uma vez que o compilador esconde do programador os endereços das variáveis?
- Utilizando ponteiros (pointers) e funções específicas de leitura (peek) e escrita (poke):

```
int peek(char *location) {
  return *location; }
```

- Para ler um registrador

```
#define DEV1 0x1000
….
dev_status = peek (DEV1)
```

Poke

- void poke(char *location, char newval) {

  (*location) = newval; } /* escrever na locação

- Como escrever no registrador de status?

  poke(DEV1,8); /* escrever 8 no registrador */

3.2.3 Saída Ocupada/Aguardando (Busy/wait)

- Maneira mais simples de programar o dispositivo de I/O
- Os dispositivos de I/O são tipicamente mais lentos que a CPU e necessitam de muitos ciclos para completar uma operação:
  - Se a CPU estiver executando múltiplas operações, como escrevendo muitos caracteres em um único dispositivo de saída, a CPU vai precisar esperar uma operação completar antes de iniciar outra.
  - Se tentarmos escrever o segundo caracter antes do dispositivo de saída ter completado a operação de escrita do primeiro, geralmente o dispositivo de saída não vai escrever o primeiro caracter.
  - Perguntar ao dispositivo de I/O se ele terminou a operação simplesmente lendo seu registro de status é comhecido como polling (enquete)

3.2.3 Saída Ocupada/Aguardando (Busy/wait)

- Maneira mais simples de se comunicar com o dispositivo:
  - Utilize instruções para ver quando o dispositivo está disponível, faça *polling*!

```
current_char = mystring;
while (*current_char != '\0') {
  poke(OUT_CHAR,*current_char);
  while (peek(OUT_STATUS) != 0);
  current_char++;
}
```

Próxima aula

- Interrupção, Exceção, Reset
- Introdução a:
  - Hierarquia de Memória
  - Memória Cache
  - Memória Virtual

Obrigado