

Aula 14

- A interface SPI (*Serial Peripheral Interface*)
- Sinalização
- Sequência de operação
- Arquiteturas de ligação
- Tipos de transferências
- Passos de configuração de um *master* SPI

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

Introdução

- SPI – sigla para "Serial Peripheral Interface"
- Interface definida inicialmente pela Motorola (Microwire da National Semiconductor é um *subset* do protocolo SPI)
- O SPI é utilizado para comunicar com uma grande variedade de dispositivos:
 - Sensores de diverso tipo: temperatura, pressão, etc.
 - Cartões de memória (MMC / SD)
 - Circuitos: memórias, ADCs, DACs, Displays LCD (e.g. telemóveis), comunicação entre corpo de máquinas fotográficas e as lentes, ...
 - Comunicação entre microcontroladores
- **Ligação a curtas distâncias (dezenas de cm)**

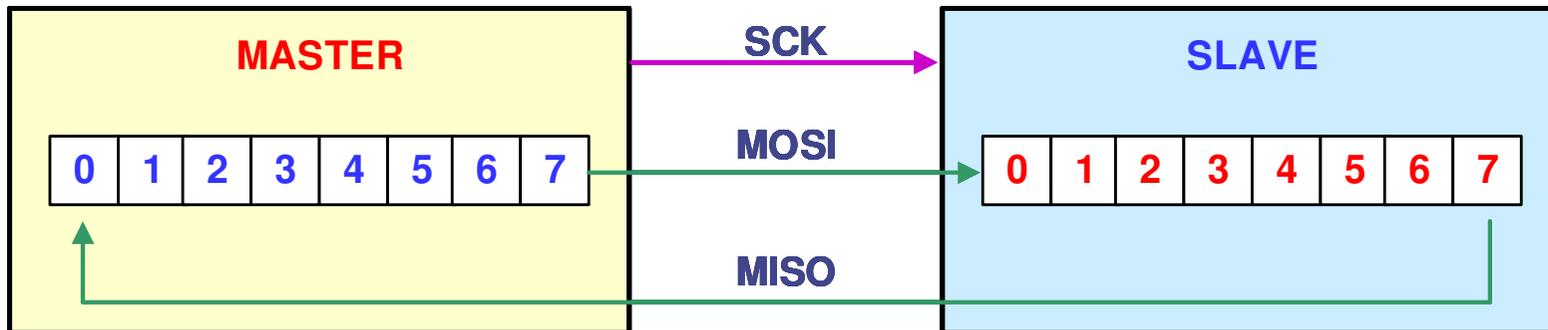
Descrição geral

- Arquitetura "Master-Slave" com ligação ponto a ponto
- Comunicação bidirecional "full-duplex"
- Comunicação síncrona (relógio explícito do *master*)
 - Relógio é gerado pelo *master* que o disponibiliza para todos os *slaves*
 - Não é exigida precisão ao relógio - os bits vão sendo transferidos a cada transição de relógio. Isto permite utilizar um oscilador de baixo custo no *master* (não é necessário um cristal de quartzo)
- Fácil de implementar por hardware ou por software
- Não são necessários "line drivers" (ou "transceivers") - circuitos de adaptação ao meio de transmissão. Os níveis lógicos correspondem aos da diferença de potencial de alimentação dos dispositivos (e.g. 3.3V)

Descrição geral

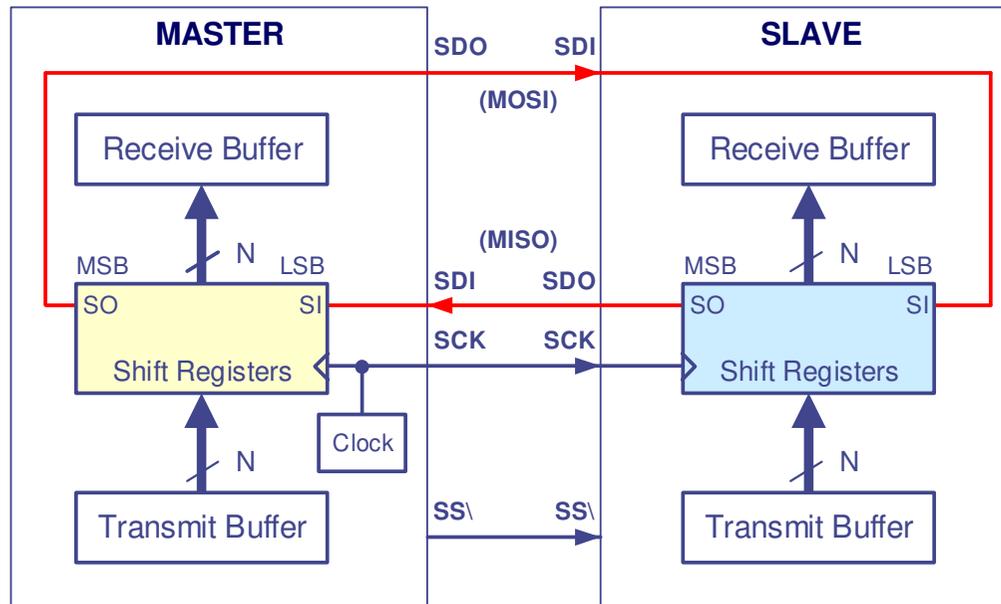
- Arquitetura "Master-Slave"
 - O sistema só pode ter um *master*
 - O *master* é o único dispositivo no sistema que pode controlar o relógio
- Um *master* pode estar ligado a vários *slaves*: para cada comunicação, apenas 1 *slave* é selecionado pelo *master* (daí ligação ponto a ponto)
- O *master* inicia e controla a transferência de dados
- Sinalização:
 - **SCK** – clock
 - Relógio gerado pelo *master* que sincroniza a transmissão/receção de dados
 - **MOSI** – Master Output Slave Input (SDO no *master*)
 - Linha do *master* para envio de dados para o *slave*
 - **MISO** – Master Input Slave Output (SDI no *master*)
 - Linha do *slave* para enviar dados para o *master*
 - **SS** – Slave select
 - Linha do *master* que seleciona o *slave* com quem vai comunicar

Descrição geral – esquema de princípio



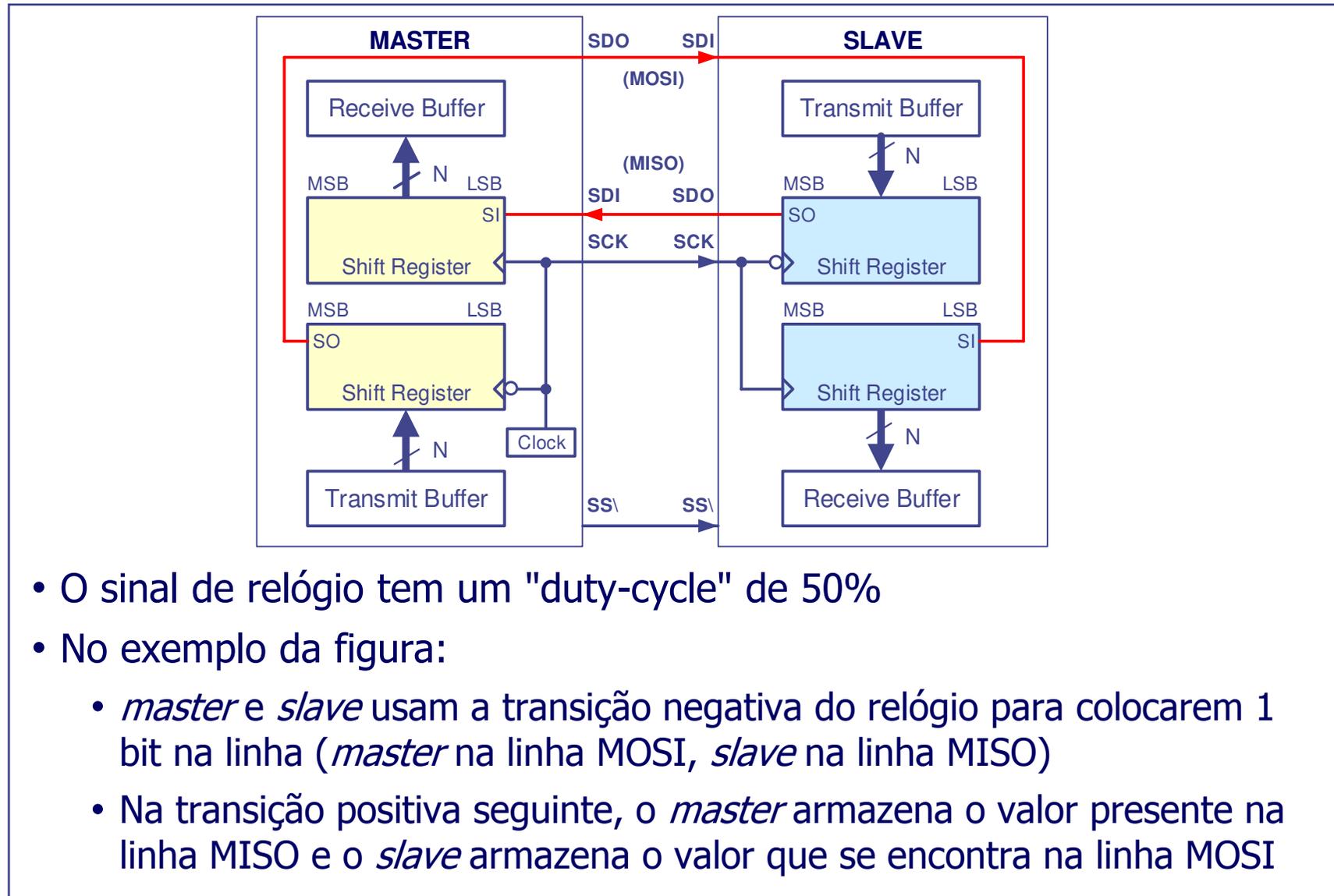
- Transmissão "full-duplex" baseada em dois *shift-registers* (um no *master* e outro no *slave*)
- Em cada ciclo de relógio:
 - O *master* coloca 1 bit na linha MOSI e o *slave* recebe-o
 - O *slave* coloca 1 bit na linha MISO e o *master* recebe-o
- Ao fim de N ciclos de relógio o *master* enviou uma palavra de N bits e recebeu do *slave* uma palavra com a mesma dimensão – "Data Exchange"
- Esta sequência é realizada mesmo quando é pretendida uma comunicação unidirecional

Sinalização

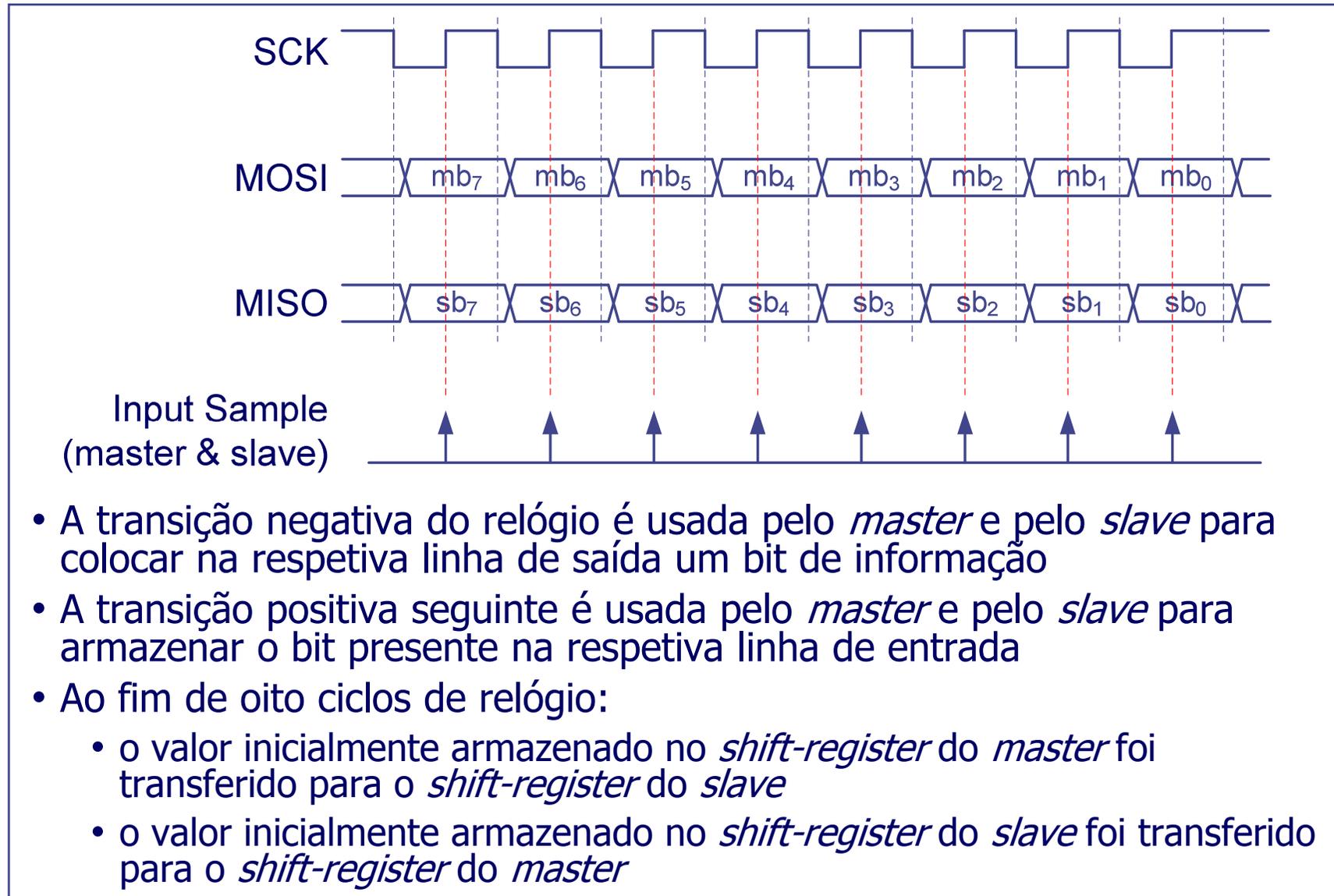


- Dados:
 - MOSI – Master Output Slave Input (SDO – serial data out no *master*)
 - MISO – Master Input Slave Output (SDI – serial data in no *master*)
- Controlo:
 - SS\ – Slave select (sinal ativado pelo *master* para seleccionar o *slave* com quem vai comunicar)
 - SCK – serial clock

Sinalização



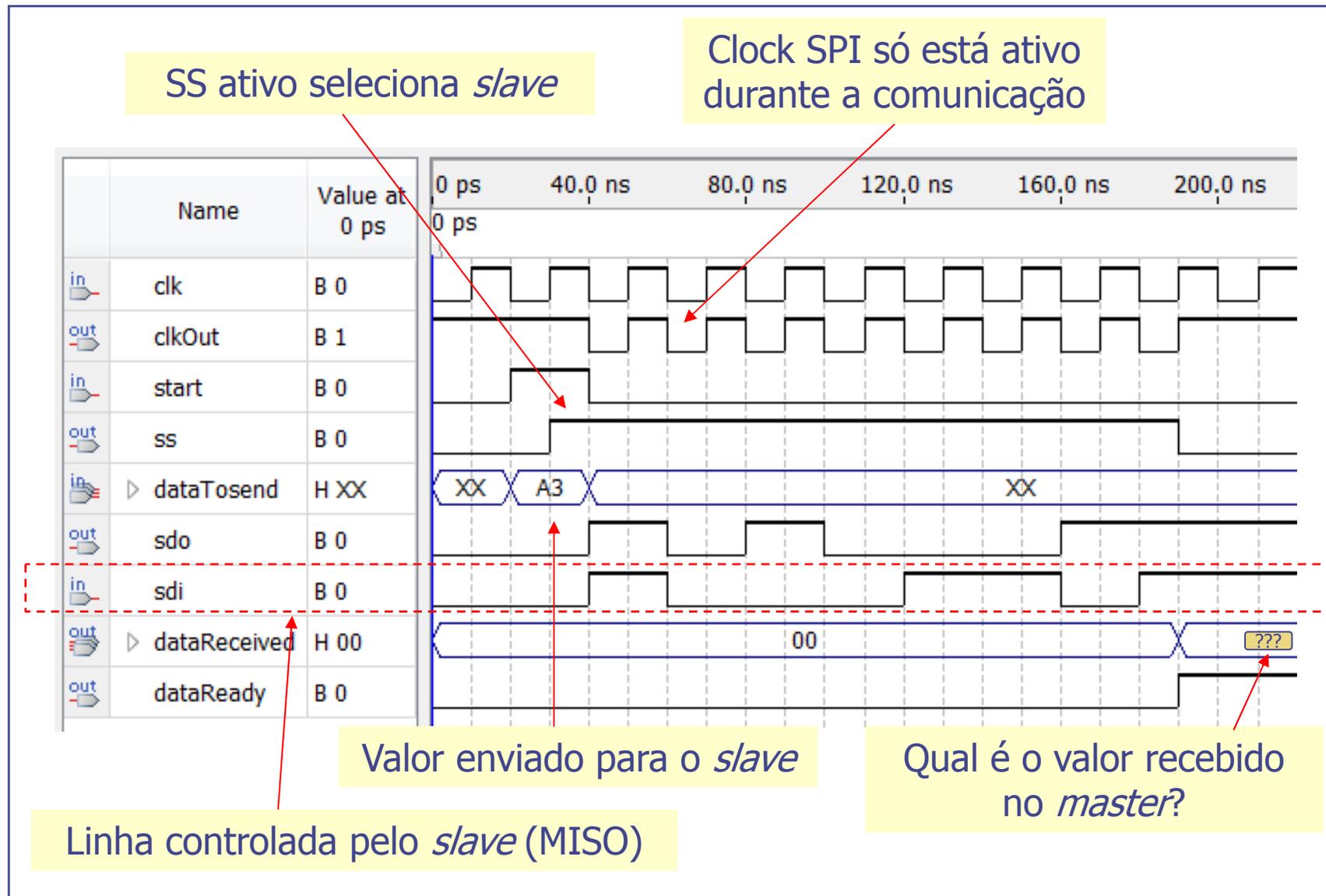
Operação – exemplo



Operação

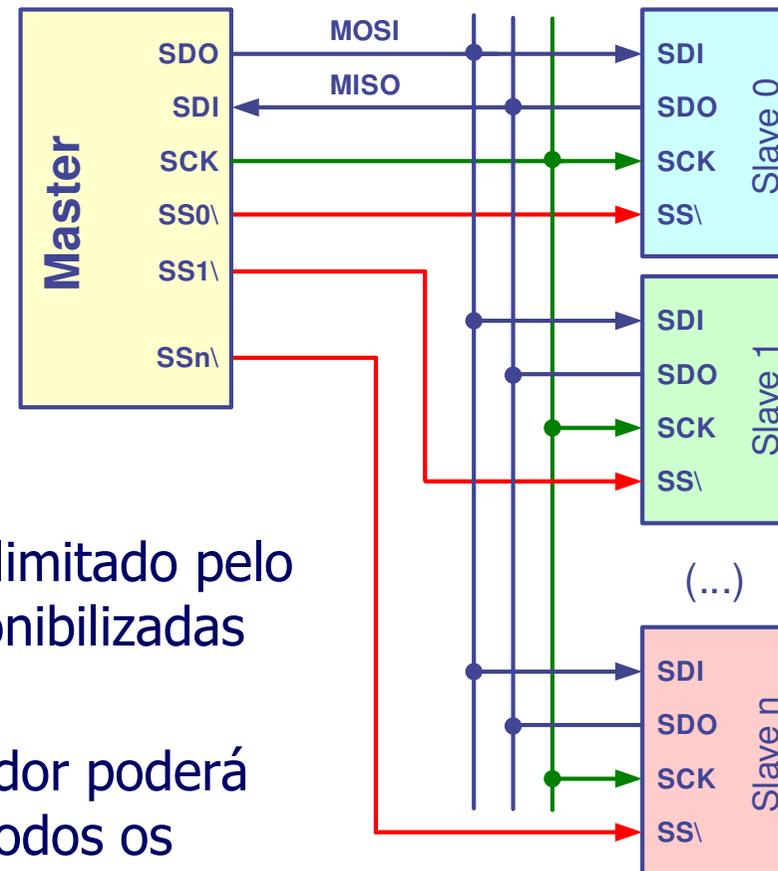
- O *master* ativa a linha SS\ do *slave* com que vai comunicar
- O *master* ativa o relógio que vai ser usado para sincronizar a troca de informação com o *slave* com quem vai comunicar
- Em cada ciclo do relógio, por exemplo na transição positiva
 - O *master* coloca na linha MOSI um bit de informação que é lido pelo *slave* na transição de relógio oposta seguinte
 - O *slave* coloca na linha MISO um bit de informação que é lido pelo *master* na transição de relógio oposta seguinte
- O *master* desativa a linha SS\ e desativa o relógio (que fica estável, por exemplo, no nível lógico 1)
 - Só há relógio durante o tempo em que se processa a transferência
- No final, o *master* e o *slave* trocaram o conteúdo dos seus *shift-registers*

Simulação de um *master* SPI



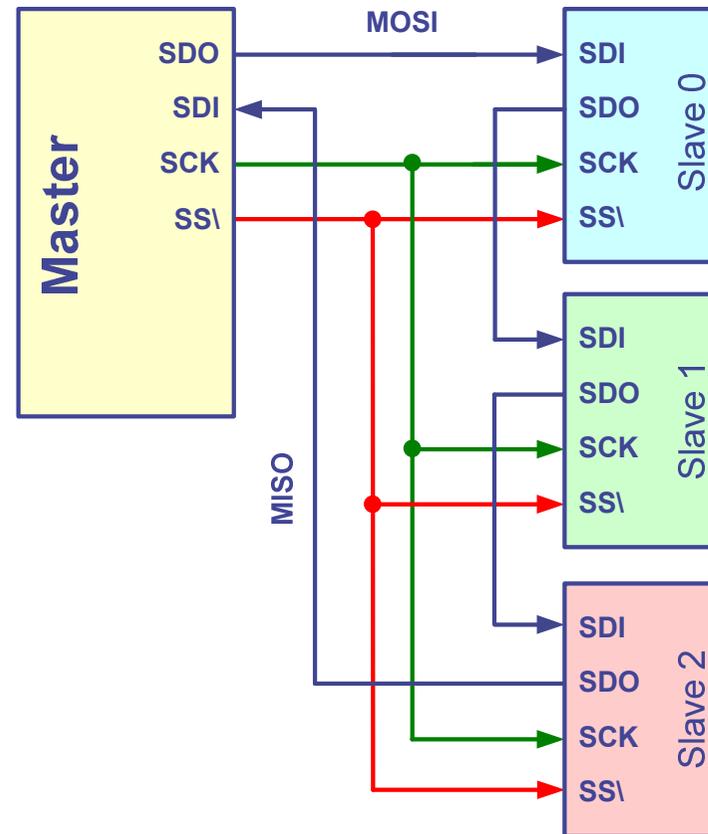
Arquiteturas de ligação – *slaves* independentes

- Sinais de seleção ("slave select") independentes
- Em cada instante apenas um $SSx\backslash$ está ativo, isto é, apenas 1 *slave* está selecionado
- Os sinais SDO dos *slaves* (MISO) não selecionados estão em alta impedância
- O número máximo de *slaves* está limitado pelo número de linhas de seleção disponibilizadas pelo *master*
- Alternativamente, o microcontrolador poderá gerar, através de portos digitais, todos os sinais $SSx\backslash$ necessários para comunicar com os *slaves*, ultrapassando a limitação anterior



Arquiteturas de ligação – Daisy Chain (cascata)

- Sinal "slave select" comum, SDO/SDI ligados em cascata
- Todos os *slaves* recebem o mesmo sinal de relógio gerado pelo *master*
- A saída de dados de cada *slave* liga à entrada de dados do seguinte
- Para que esta arquitetura funcione o *slave* tem de ser capaz de armazenar uma sequência de N bits enviados durante 1 ciclo de comando e enviar para a sua saída a mesma sequência de N bits durante o ciclo de comando seguinte
 - Enquanto o SS estiver ativo o *slave* ignora o comando recebido e envia-o para a saída DO no ciclo de comando seguinte
 - O *slave* apenas executa o comando quando o sinal SS é desativado



Tipos de transferências

- O SPI funciona sempre em modo "data exchange", isto é, o processo de comunicação envolve sempre a troca do conteúdo dos *shift-registers* do *master* e do *slave*
- Cabe aos dispositivos envolvidos na comunicação usar ou descartar a informação recebida
- Podem considerar-se os seguintes cenários de transferência:
 - **Bidirecional**: são transferidos dados válidos em ambos os sentidos (master → slave e slave → master)
 - **Master → slave (operação de escrita)**: *master* transfere dados para o *slave*, e ignora/descarta os dados recebidos
 - **Slave → master (operação de leitura)**: *master* pretende ler dados do *slave*; para isso transfere para o *slave* uma palavra com informação irrelevante (por exemplo 0); o *slave* ignora/descarta os dados recebidos

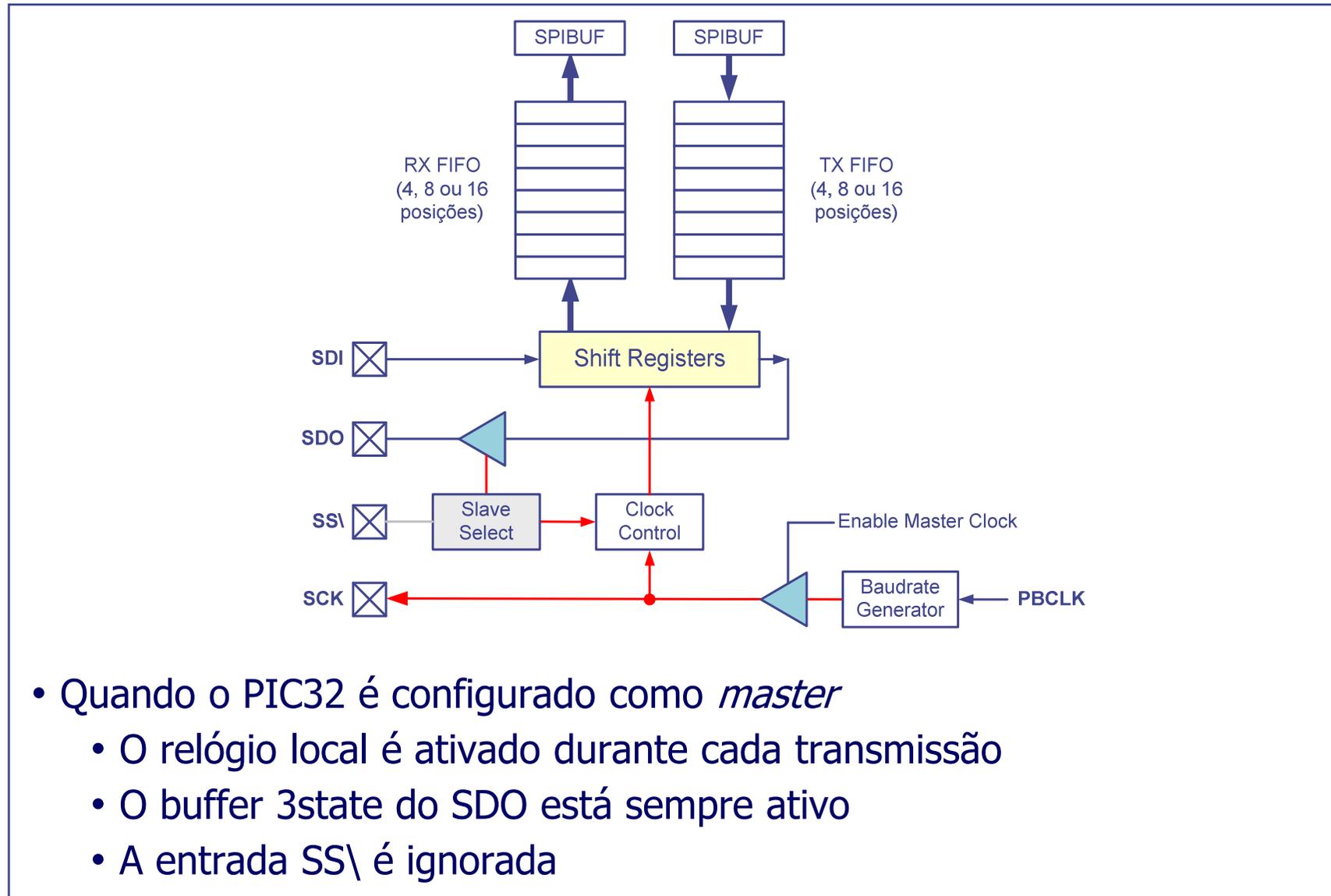
Configuração de um *master* SPI

- Antes de iniciar a transferência há algumas configurações que são efetuadas no *master* (através do seu modelo de programação) para adequar os parâmetros que definem a comunicação às características do *slave* com que vai comunicar:
 1. Configurar a frequência de relógio
 2. Configurar o nível lógico de repouso ("idle") do sinal de relógio
 3. Especificar qual o flanco do relógio usado para a transmissão (a recepção é efetuada no flanco oposto). Esta configuração é feita em função das características do *slave* com o qual o *master* vai comunicar:
 - Transmissão no flanco ascendente (consequentemente, a recepção é feita no flanco descendente)
 - Transmissão no flanco descendente (consequentemente, a recepção é feita no flanco ascendente)

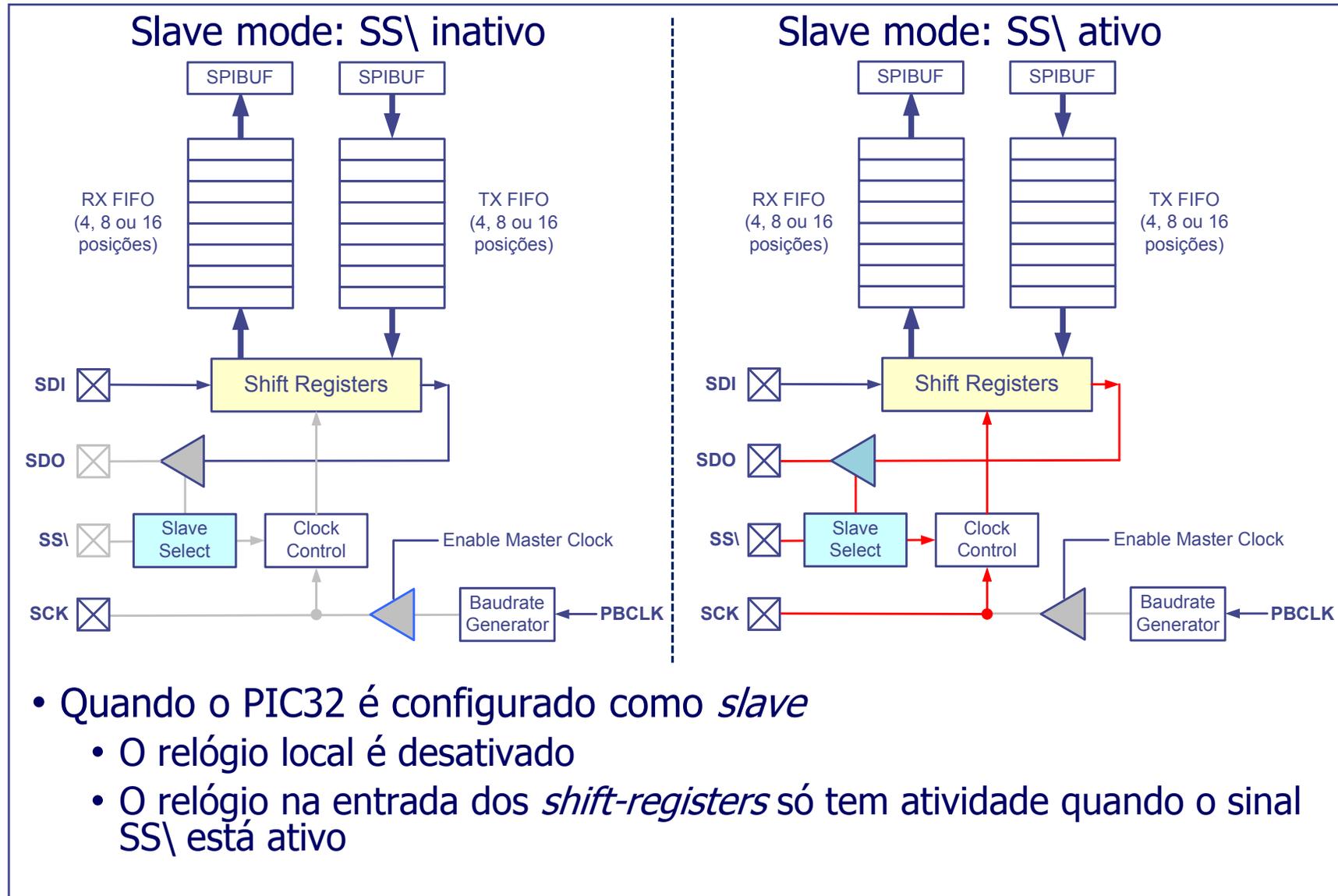
Interface SPI no PIC32

- O PIC32MX795F512H disponibiliza 3 módulos de comunicação SPI
- Cada um dos módulos pode ser configurado para funcionar como *master* ou como *slave*
- Comprimento de palavra configurável: 8, 16 ou 32 bits
- *Shift-registers* separados para receção e transmissão
- Os registos de receção e transmissão são FIFOS:
 - 16 posições se o comprimento de palavra for 8 bits
 - 8 posições se o comprimento de palavra for 16 bits
 - 4 posições se o comprimento de palavra for 32 bits
- Cada um dos módulos pode ser configurado para gerar interrupções em função da ocupação dos FIFOS (e.g. TX FIFO tem, pelo menos, 1 posição livre; RX FIFO tem, pelo menos, 1 palavra disponível para ser lida)

Interface SPI no PIC32



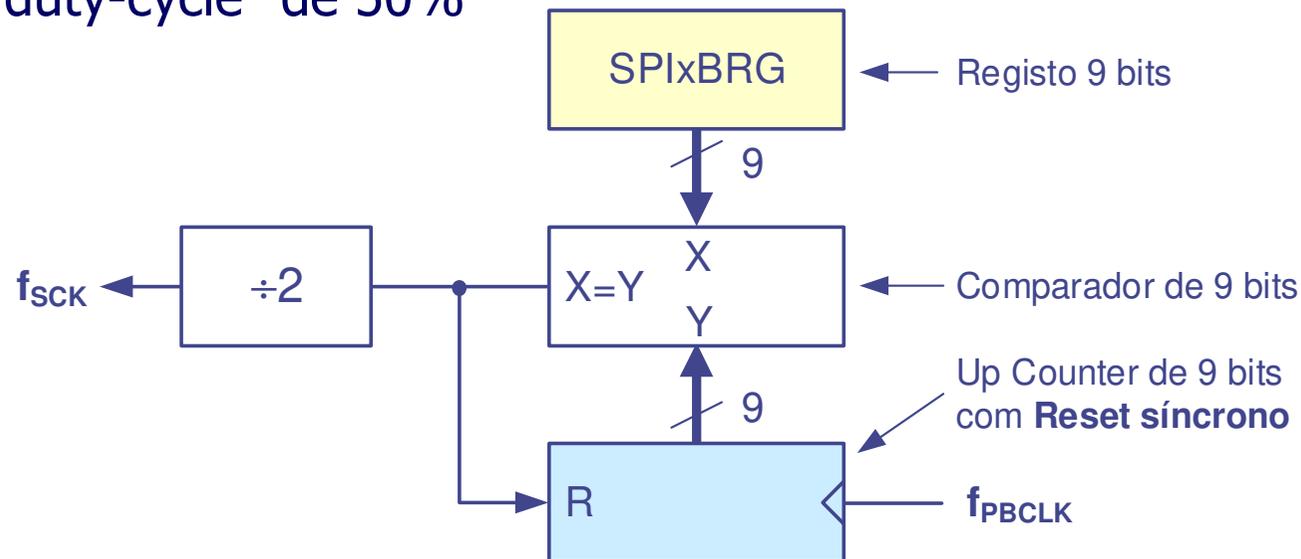
Interface SPI no PIC32



- Quando o PIC32 é configurado como *slave*
 - O relógio local é desativado
 - O relógio na entrada dos *shift-registers* só tem atividade quando o sinal SS\ está ativo

Interface SPI no PIC32 – gerador de relógio

- Utiliza uma arquitetura semelhante à de um timer, em que o sinal de relógio de entrada é o Peripheral Bus Clock (20 MHz na placa DETPIC32).
- Com a divisão por 2 à saída do comparador obtém-se um relógio com "duty-cycle" de 50%



- $f_{SCK} = f_{PBCLK} / (2 * (SPIxBRG + 1))$, em que SPIxBRG representa a constante armazenada no registo com o mesmo nome