Key Interrupt Issues

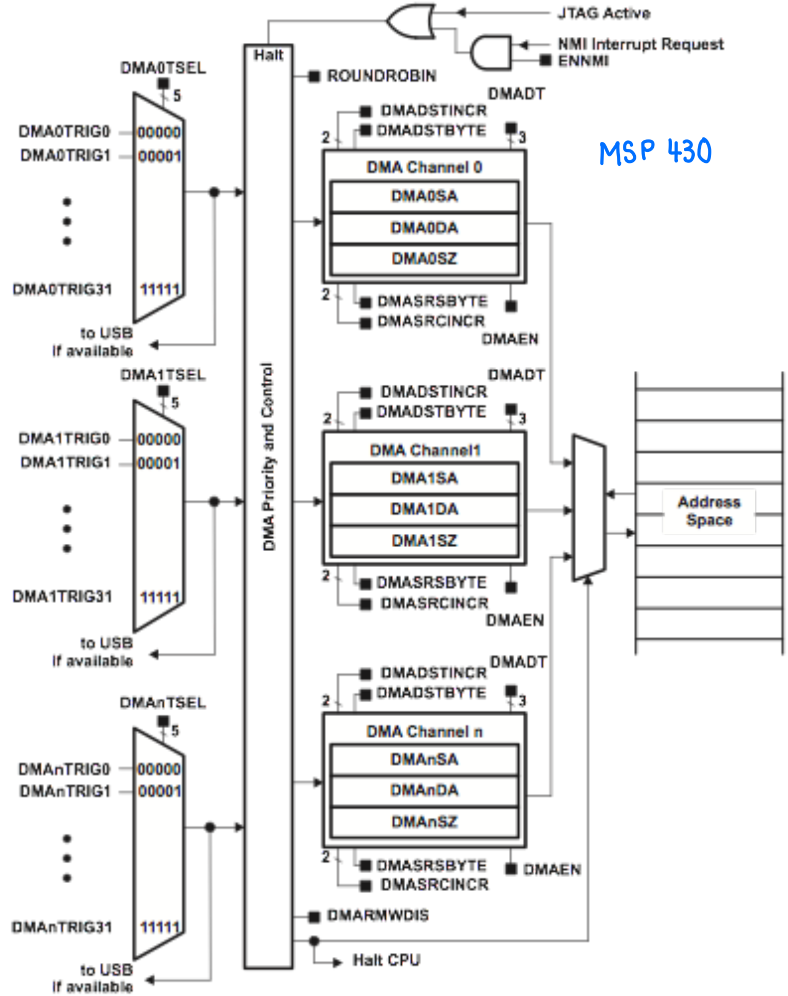
- Vectored Interrupts
- Interrupt Priority
- Maskable vs. Non-Maskable Interrupts
- What information is stored when interrupts occur?
- Nesting of Interrupts (use with care!)

Exception number	Exception type	Priority	Descriptions
1	Reset	-3 (Highest)	Reset
2	NMI	-2	Non-Maskable Interrupt
3	HardFault	-1	Fault handling exception
4–10	Reserved	NA	—
11	SVCall	Programmable	Supervisor call via SVC instruction
12–13	Reserved	NA	—
14	PendSV	Programmable	Pendable request for system service
15	SysTick	Programmable	System Tick Timer
16	Interrupt #0	Programmable	External Interrupt #0
17	Interrupt #1	Programmable	External Interrupt #1
...	...	...	...
47	Interrupt #31	Programmable	External Interrupt #31

Direct Memory Access (DMA)

- Mechanism by which peripherals can directly transfer data to/from memory without the participation of the MCU core (move data without CPU involved)
  from one peripheral-2-another ~ free up CPU
- Used mainly for low power modes
- N independent DMA channels
- Up to T triggers to initiate DMA
  DMA happens when T conditions met
- Various addressing, transfer modes
- DMA channels can be prioritized
  - Transfers are non-preemptive

MSP 430



To move data from one peripheral to another, we don't need to have CPU up and running and using energy

one memory loc to another

DMA Channels - identical & independently operated; each channel can move data from one memory loc to another independently

Trigger Conditions - anything that can trigger a DMA transfer (e.g., interrupt → DMA transfer)

Main purpose of DMA is to transfer data from one peripheral to another when the main MCU will be in low-power mode

Extremely powerful if you know how to use it; if not, you'll mess up

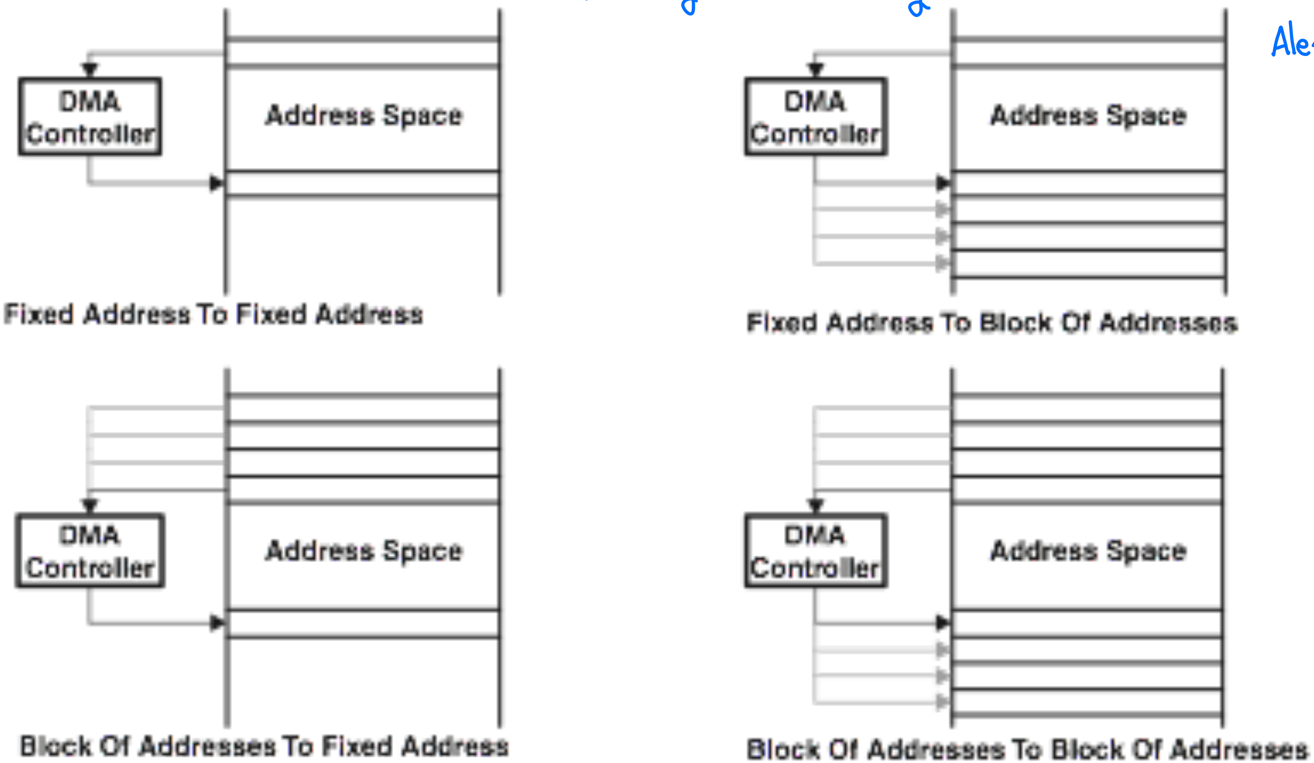
[DMA's are also popular in Embedded Systems Interviews]

DMA Addressing Modes

- Addressing mode for each channel is independently controllable
- Transfers may be byte-byte, word-word, byte to word, or word to byte
  - Only lower byte of word is affected in byte to word and word to byte transfers

DAC : Digital-to- Anolog Converter

Alexa



DMA Transfer Modes

- DMA controller has six transfer modes
- Each channel is individually configurable for its transfer mode
  - Transfer mode is independent of addressing mode
- In mode 000, 001, 100, and 101, the CPU is halted
- In mode 010-011, 110-111, CPU activity is reduced to 20%
  - CPU gets 2 MCLK cycles after every 4 DMA transfers

DMADT	Transfer Mode	Description
000	Single transfer	Each transfer requires a trigger. DMAEN is automatically cleared when DMAxSZ transfers have been made.
001	Block transfer	A complete block is transferred with one trigger. DMAEN is automatically cleared at the end of the block transfer.
010, 011	Burst-block transfer	CPU activity is interleaved with a block transfer. DMAEN is automatically cleared at the end of the burst-block transfer.
100	Repeated single transfer	Each transfer requires a trigger. DMAEN remains enabled.
101	Repeated block transfer	A complete block is transferred with one trigger. DMAEN remains enabled.
110, 111	Repeated burst-block transfer	CPU activity is interleaved with a block transfer. DMAEN remains enabled.

Serial Communication Interfaces

- Communication between electronic devices can either be parallel communication or serial communication

  multiple bits / multiple wires

  1 bit at a time is transferred

- Parallel communication: send several bits together over parallel wires
  - Advantages: Higher throughput, simpler to implement in hardware → no need for serializer/deserializer logic
  - Disadvantages: More expensive, signal integrity issues (clock skew, crosstalk)
  - Examples: on-chip and system buses, PCI, SCSI, DB-25 parallel-port devices

    more wires/area

    set of wires too close to each other

- Serial communication: send one bit at a time, sequentially
  - Advantages: Less expensive (less wires), less signal integrity issues
  - Disadvantages: Lower throughput, require additional SerDes logic
  - Examples: RS-232, SPI, I2C, Ethernet, USB, Firewire, SATA, PCI Express

    CAN-bus → automobiles

- Many devices using parallel comm. are moving to serial comm.
  - A notable exception is in wireless communication → driving factor is higher throughput

    those that don't need higher throughput

Synchronous vs. Asynch. Comm.

- All digital communication requires a clock signal (the transmitter uses the clock to decide when to transmit the next bit and the receiver uses the clock to decide when to read/sample the next bit)

- In synchronous communication standards such as SPI and I2C, the clock signal is sent on a separate wire along with the data.
  - The device that generates the clock is called the Master and the other device is called the Slave

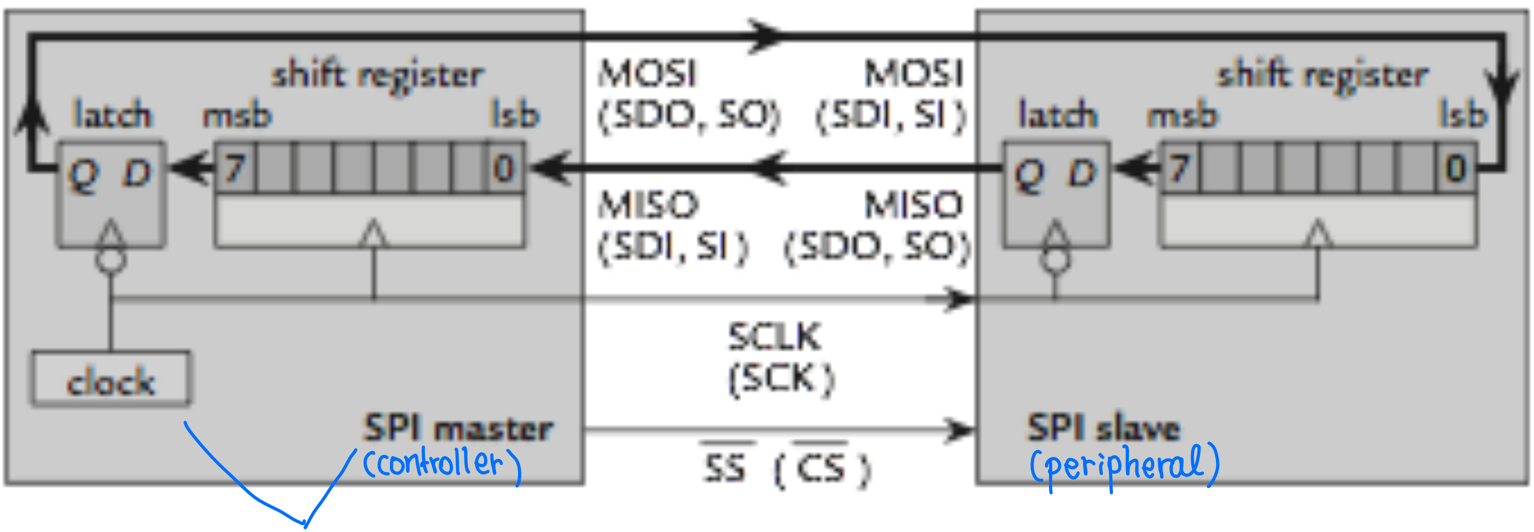
- In asynchronous communication, there is no clock signal sent. The receiver has a separate clock that it usually derives using some additional information in the data line (called clock & data recovery)

- Can use two data wires and have data travel in both directions simultaneously (full-duplex) or use a single wire and have data travel only in one direction at a time (half-duplex)

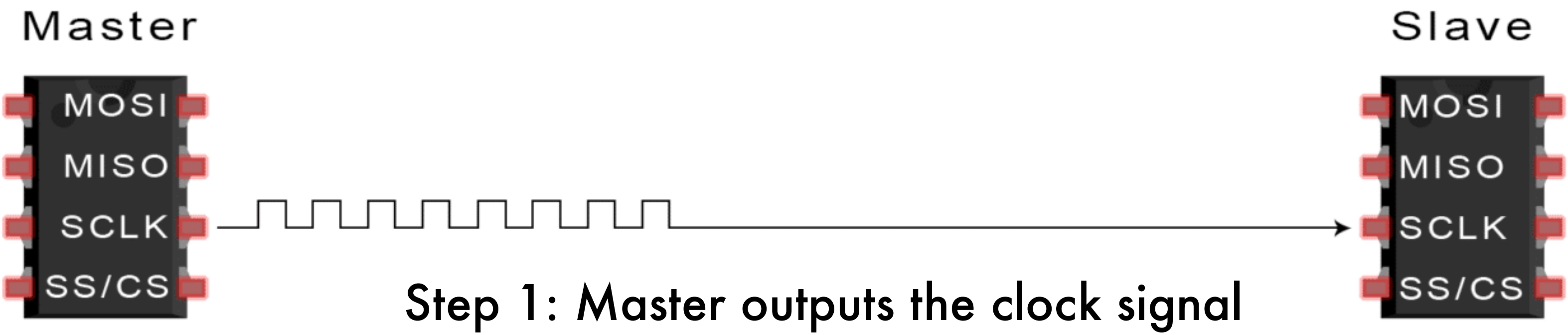
Serial Peripheral Interface (SPI)

- Simple concept - two shift registers connected to each other
- Full version uses 4 wires (MOSI, MISO, SCLK, SS')
  - MOSI: Master Out Slave In (data)
  - MISO: Master In Slave Out (data)
  - SCLK: Clock from master to slave (clock)
  - SS': Slave select to activate the slave, usually active low (control)
- Inherently, full-duplex in nature (MOSI and MISO both carry data)

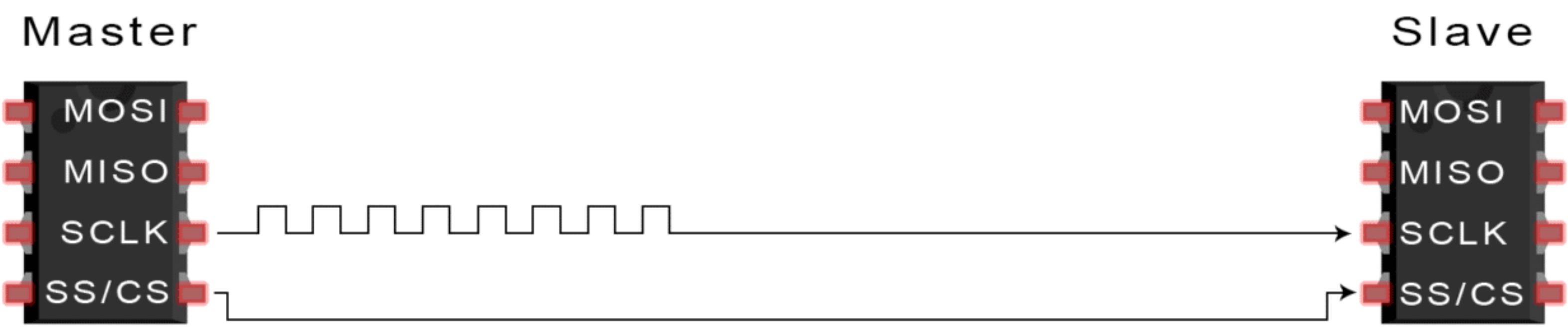
the most commonly used serial communication standard in Embedded Systems



SPI Communication: Sequence of Steps

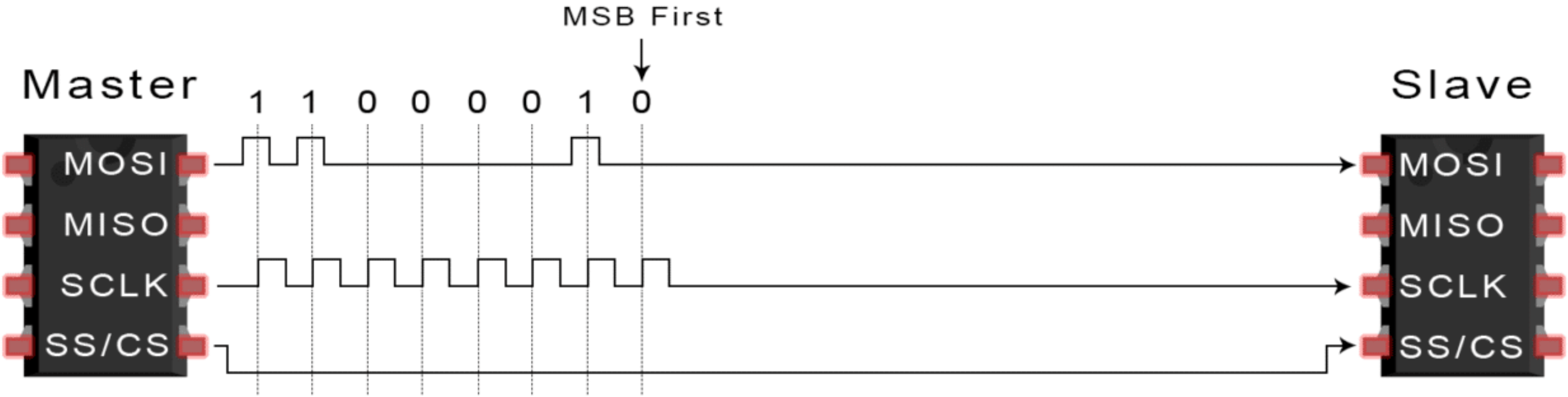


Step 1: Master outputs the clock signal

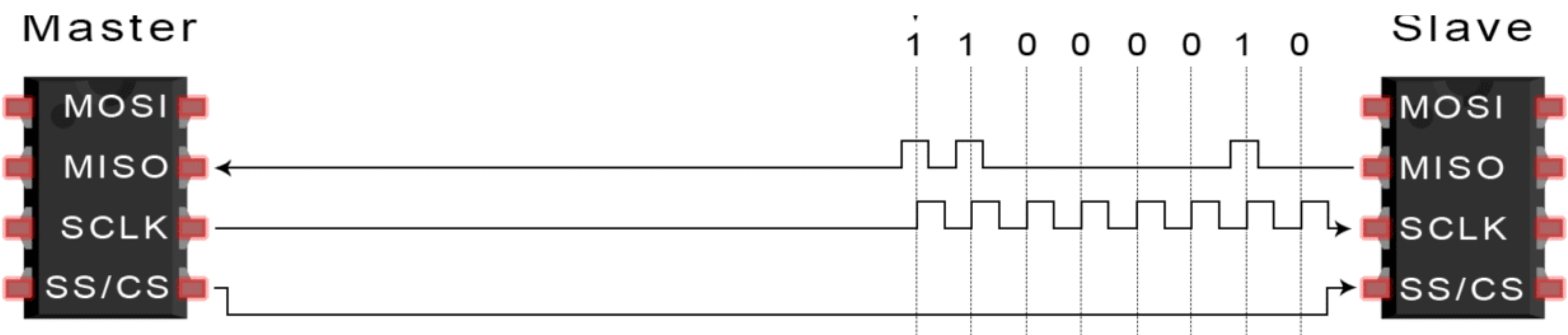


Step 2: Master sets SS low, which activates the slave device

SPI Communication: Sequence of Steps



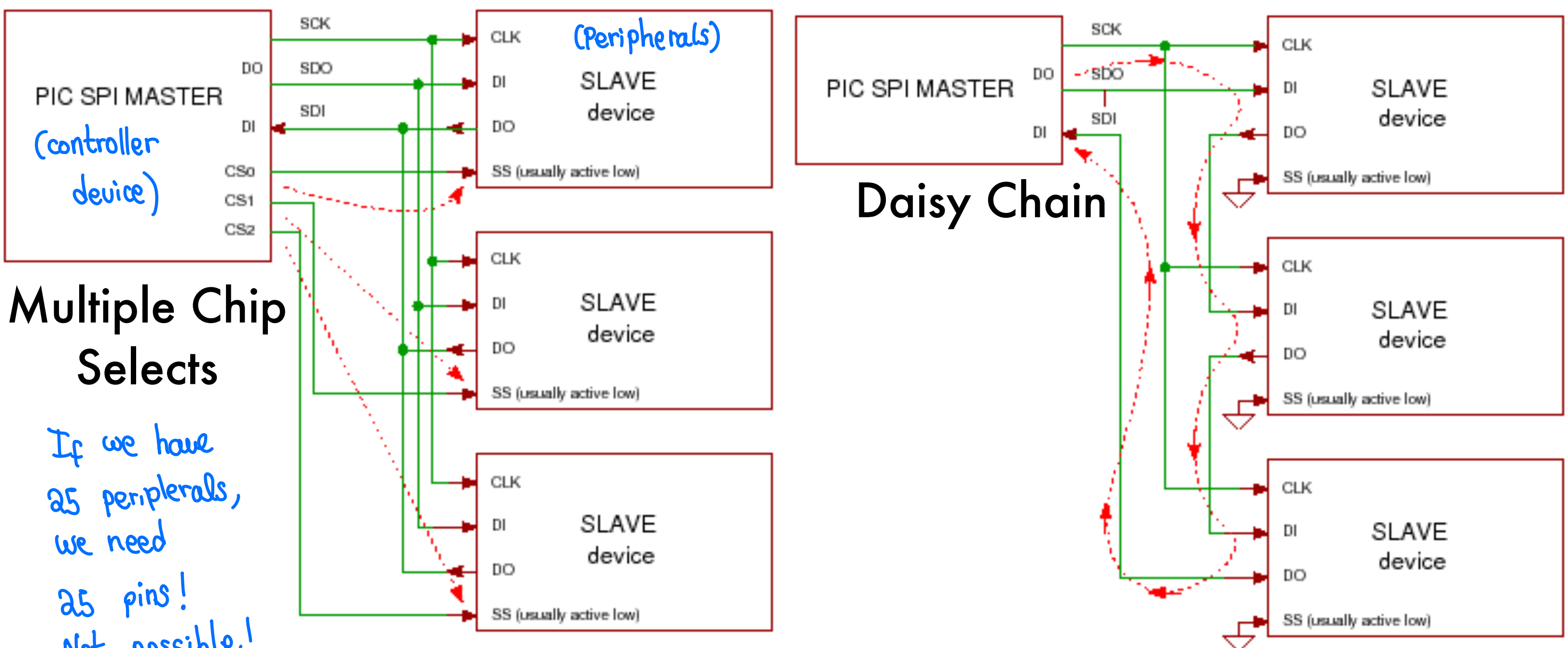
Step 3: Master outputs data one bit at a time to the slave



Step 4: Master reads data one bit at a time from slave (can happen along with step 3) separate wires

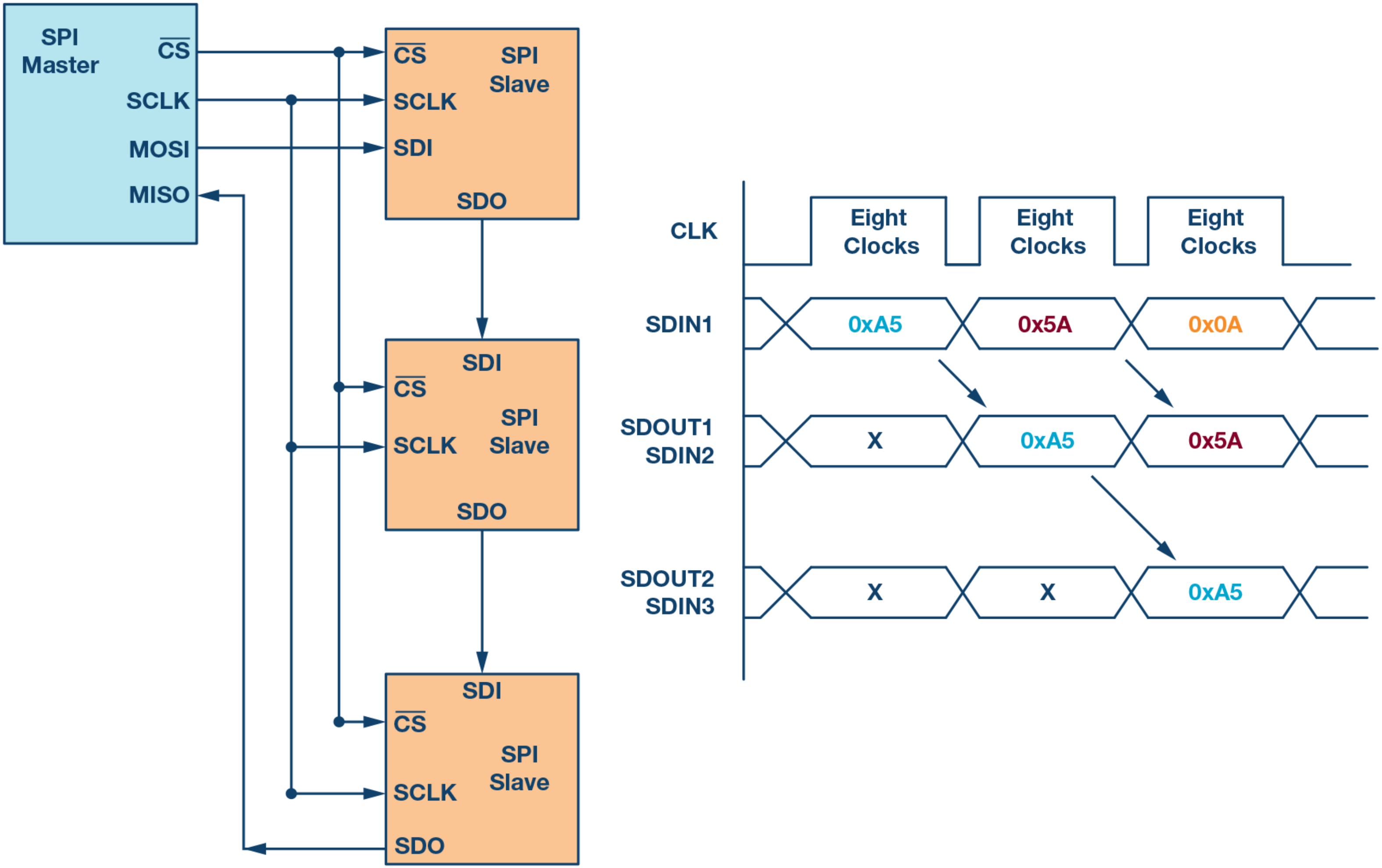
Serial Peripheral Interface (SPI)

- One master can talk to multiple slaves (one at a time) by using a dedicated slave select line for each slave
- What happens if multiple slave select lines are driven low?



If we have 25 periplerals, we need 25 pins! Not possible!

SPI Daisy Chain Mode



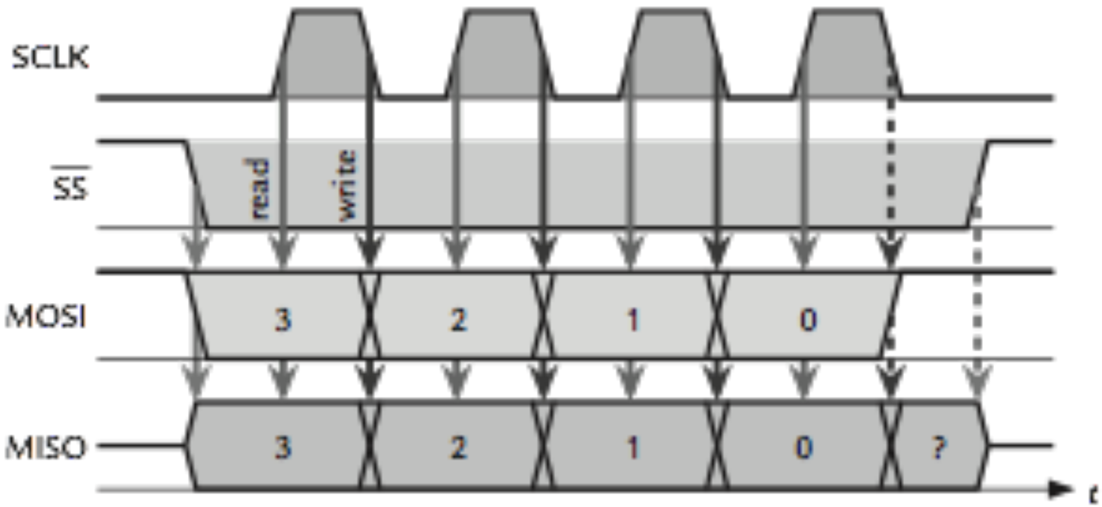
SPI Clock Phase and Polarity

- Four different SPI modes (0 - 3) depending on clock phase (CPHA) and polarity (CPOL) bits
  - If CPOL = 0, clock idles at 0 (low) between transfers
  - If CPOL = 1, clock idles at 1 (high) between transfers
  - If CPHA = 0, data is read on the leading edge of the clock pulse and written on the trailing edge of the clock pulse
  - If CPHA = 1, data is written on the leading edge of the clock pulse and read on the trailing edge of the clock pulse

determines at which edge of the clock is data written

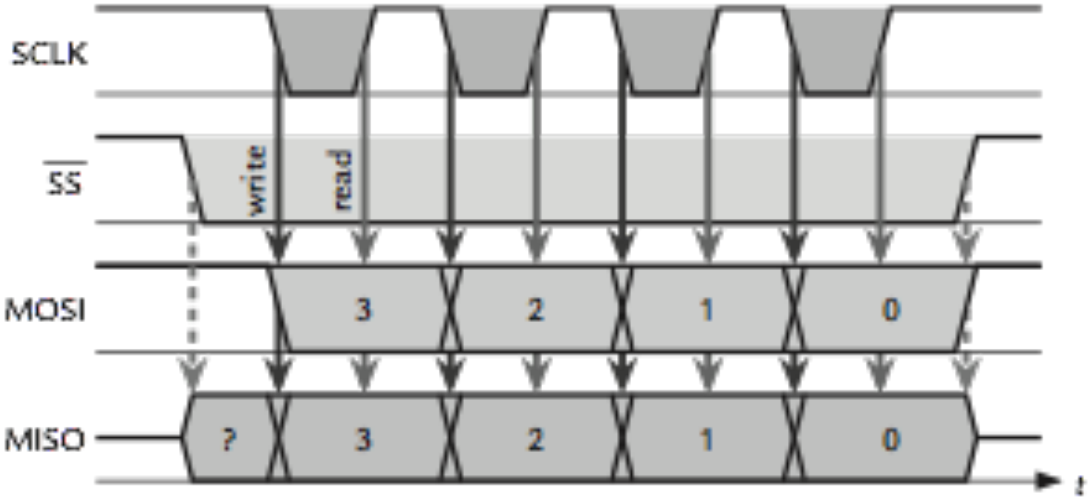
Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Example: SPI Mode 0



- Downward transition on SS' causes the first data bit to be placed on the data lines
- Data is read in at the leading edge (positive edge) of the clock
- Next bit of data is output on the data lines at the trailing edge (negative edge) of the clock
- Upward transition on SS' returns MISO line to floating state

Example: SPI Mode 3



- Downward transition on SS′ starts the transaction
- Each bit of data is output to data lines at the leading edge (negative edge) of the clock
- Data is read at the trailing edge (positive edge) of the clock
- Upward transition on SS′ returns MISO to floating level

SPI - Advanced Modes and Conclusion

- Dual SPI - Send two bits in each clock cycle in half-duplex mode using both MOSI and MISO lines
- Quad SPI - Add two additional data lines to send four bits in each clock cycle

SPI controller is mainly independent of the main code — separate set of transistors from CPU

Advantages

- Simple, no complicated addressing scheme
- Full duplex, so data can be sent in both directions simultaneously
- High-speed (faster than UART and I2C)
- No packets or frames, so data can be sent continuously

Disadvantages

[we do not check whether peripheral received the data]

- Requires 4 lines
- Sending data to multiple slaves either needs multiple SS lines or the nuances of daisy-chaining
- No error checking or acknowledgement that data received
- Only one master