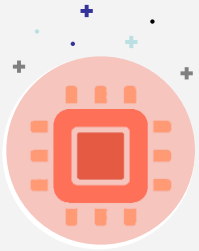
1405/03



Microcontrollers

Lecture 11:

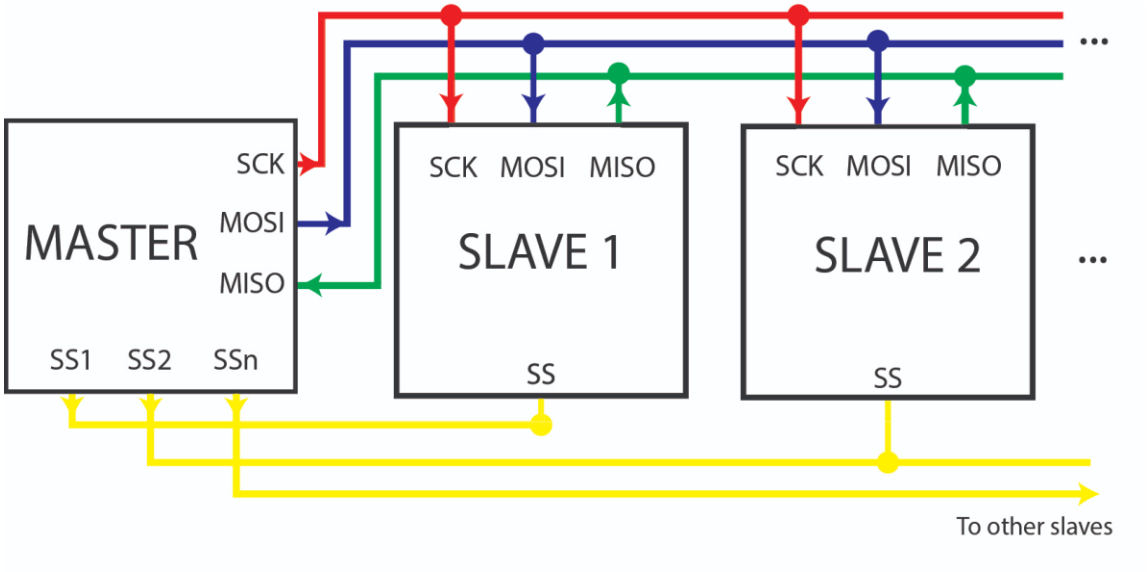
SPI Protocol

By: M.Razeghizadeh

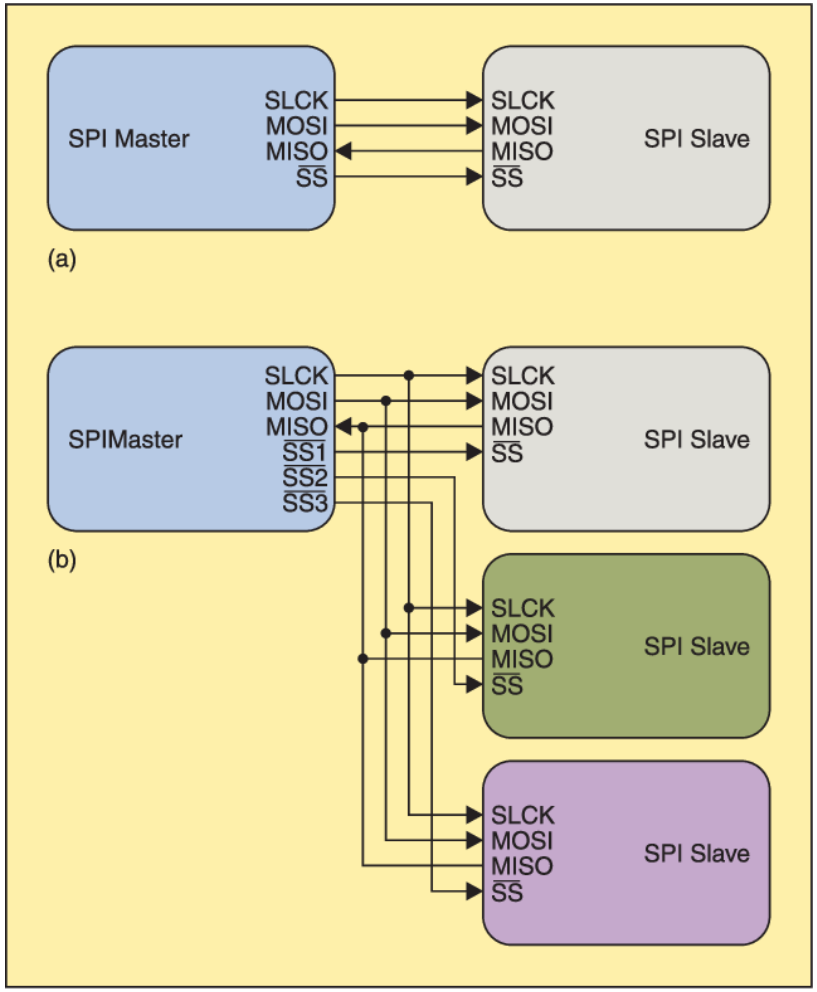
Start now!

مقدمه‌ای بر ارتباط SPI

SPI یک گذرگاه سریال پرسرعت است که برای ارتباط بین یک کنترل‌کننده و چندین تجهیز جانبی به کار می‌رود.



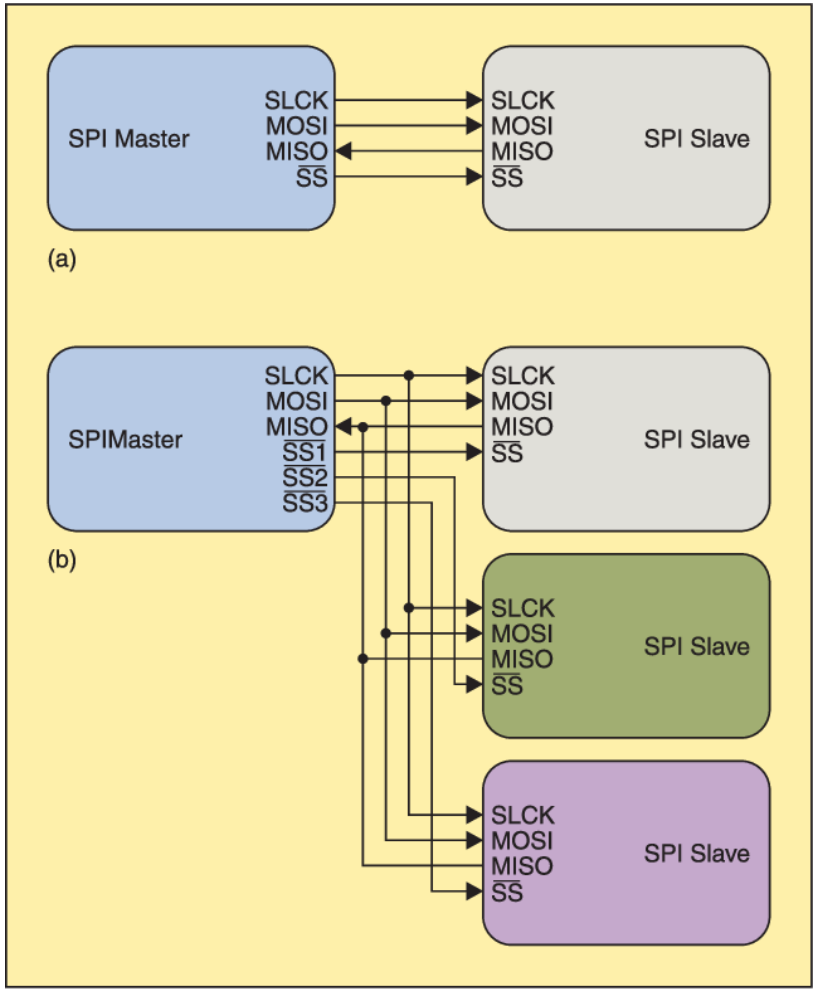
مقدمه‌ای بر ارتباط SPI



- SPI مخفف Serial Peripheral Interface
- یک گذرگاه ارتباطی سریال (انتقال بیت به بیت داده) و سنکرون (انتقال داده وابسته به کلا مشترک) برای ارتباط بین میکروکنترلر و تجهیزات جانبی است.
- دارای سرعت انتقال داده نسبتا بالایی است.
- کاربرد اصلی آن برای ارتباط بین دو قطعه روی برد است و برای فواصل زیاد مناسب نیست. لذا بیشتر کاربرد آن در بردهای الکترونیکی و سیستم‌های تعبیه شده به چشم می‌خورد.
- بین یک Master و یک یا چند Slave کار می‌کند.

مقدمه‌ای بر ارتباط SPI

- برای انتقال داده بین میکروکنترلر (Master) و دستگاه‌های جانبی (Slave) مثل سنسورها، حافظه‌های فلش، ADC، DAC، RTC، LCD، SD و تجهیزات مشابه مورد استفاده قرار می‌گیرد.
- توسعه‌یافته در دهه ۱۹۸۰ توسط Motorola برای کاربردهای سیستم‌های نهفته
- پروتکلی با قابلیت ارسال و دریافت داده به‌صورت هم‌زمان (فول‌داپلکس)
- کلاک در SPI توسط Master تولید شده و تعیین‌کننده نرخ انتقال داده بین قطعه‌ها می‌باشد.
- سرعت بالاتر کلاک منجر به ارسال و دریافت سریع‌تر داده‌ها در این پروتکل می‌شود.



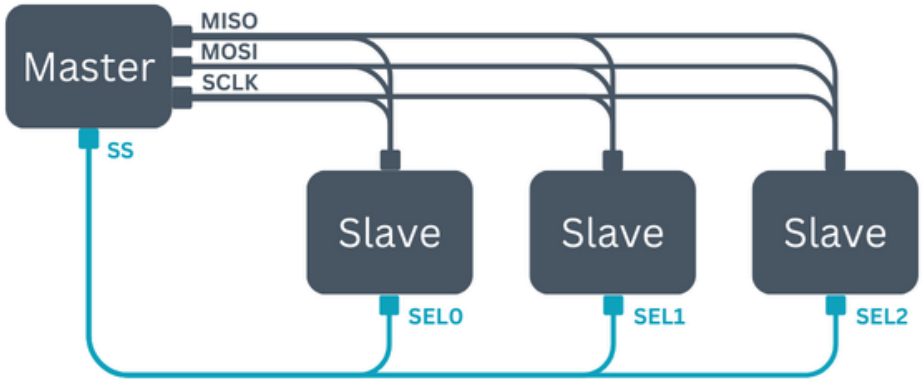
اجزای اصلی ارتباط SPI

1- Master:

- ✓ دستگاه کنترل کننده که فرمان ارسال و دریاقت را مدیریت می‌کند.
- ✓ معمولا یک میکروکنترلر است.
- ✓ معمولا Master کنترل گذرگاه را هم در دست دارد.

2- Slave:

- ✓ دستگاه‌های جانبی که داده‌ها را دریافت یا ارسال می‌کنند.



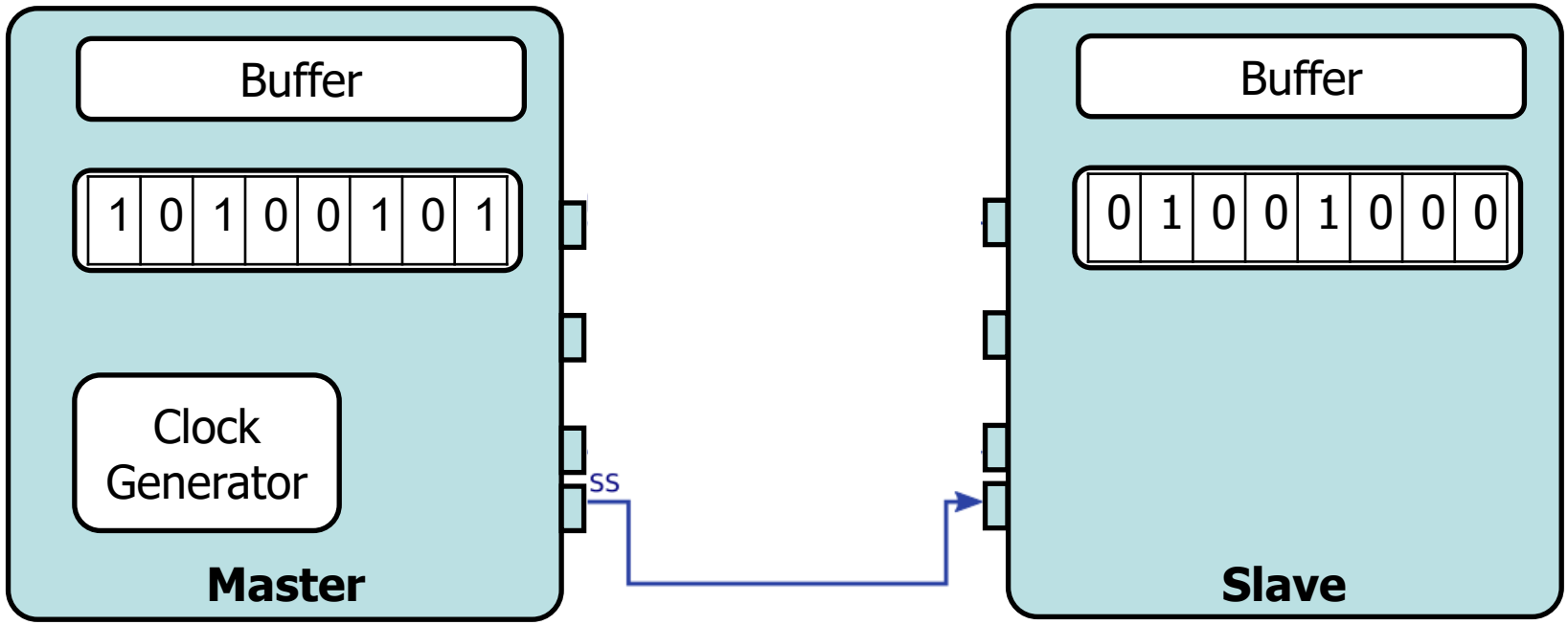
3- خطوط ارتباطی (گذرگاه):

- ✓ MOSI: Master Out Slave In
- ✓ MOSI: Master In Slave Out
- ✓ SCLK (SCK): Serial Clock
- ✓ SS (CS): Slave (Chip) Select

چگونگی انتقال داده در پروتکل ارتباطی SPI

1- فعال شدن خط SS (CS) مربوط به Slave

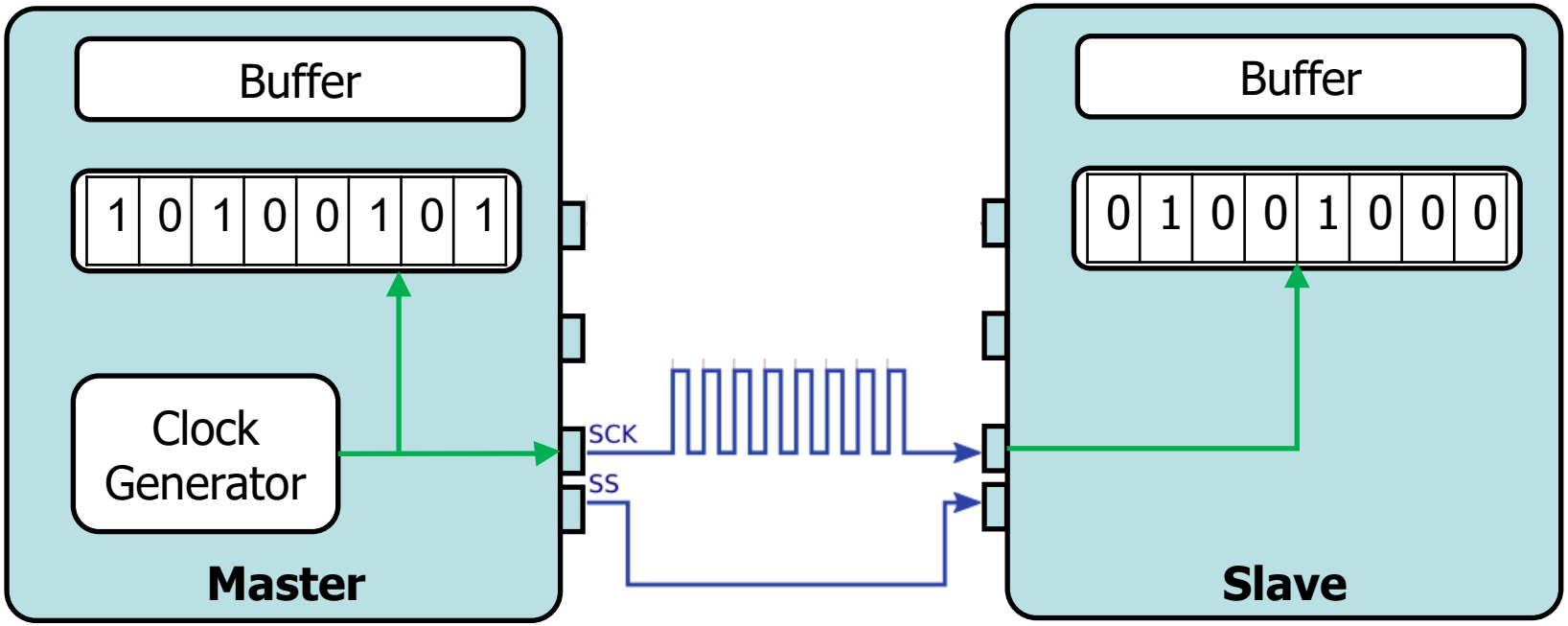
- در شروع ارتباط، Master تصمیم می‌گیرد تا با کدام Slave صحبت کند.
- خط CS مربوط به آن را فعال می‌کند.
- در صورت وجود Slave دیگر، باتوجه به غیرفعال بودن CS آن‌ها، هیچ واکنشی نشان نمی‌دهند.



چگونگی انتقال داده در پروتکل ارتباطی SPI

2- تولید سیگنال SCLK توسط Master

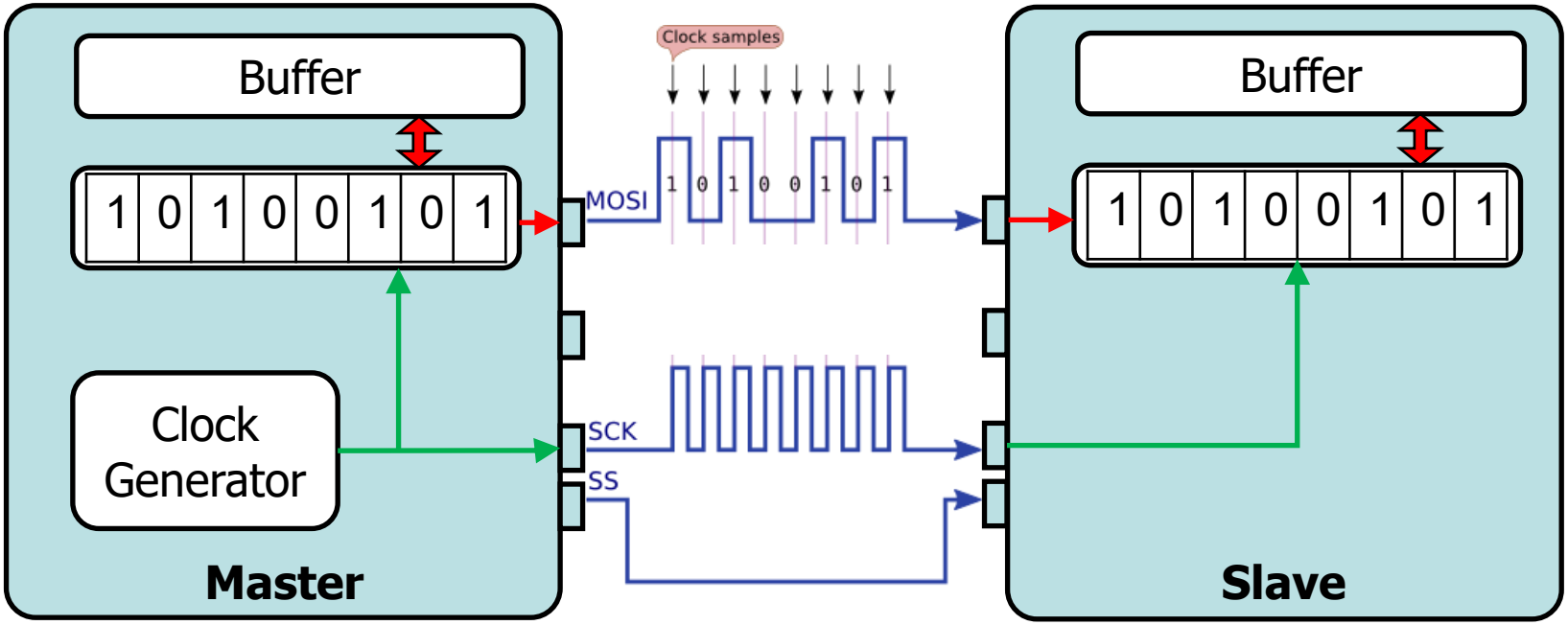
- در این مرحله Master شروع به تولید سیگنال کلاک می‌کند.
- این کلاک‌ها تعیین می‌کند که چه زمانی باید یک بیت داده ارسال یا قرائت شود.
- Slave ها همیشه کلاک را از Master دریافت می‌کنند.



چگونگی انتقال داده در پروتکل ارتباطی SPI

3- ارسال داده روی خط MOSI توسط Master

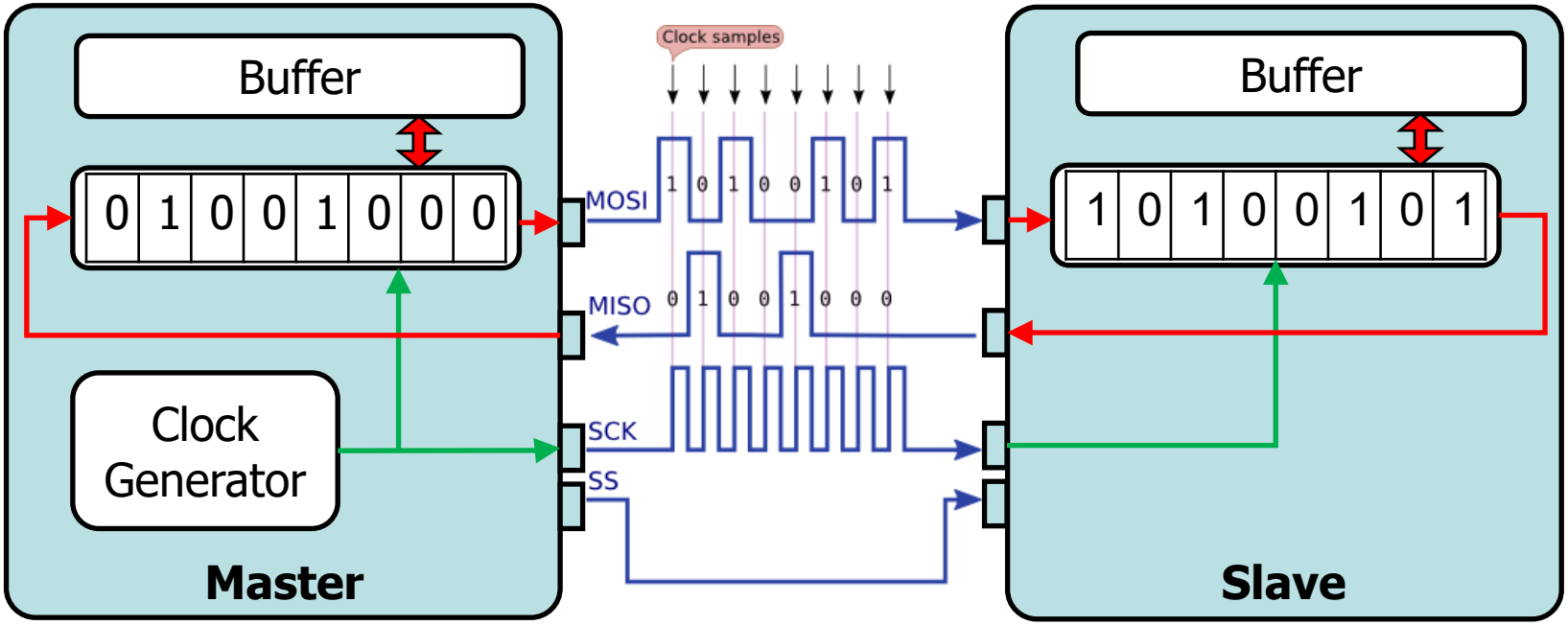
- در این مرحله، Master قصد دارد تا داده‌ای را که از Buffer دریافت کرده، ارسال نماید.
- این داده توسط Master در هر لبه کلاک، بیت به بیت روی خط MOSI قرار می‌گیرد.
- متناسب با همان لبه کلاک بیت ارسالی توسط Slave دریافت می‌شود.



چگونگی انتقال داده در پروتکل ارتباطی SPI

4- ارسال داده روی خط MISO توسط Slave

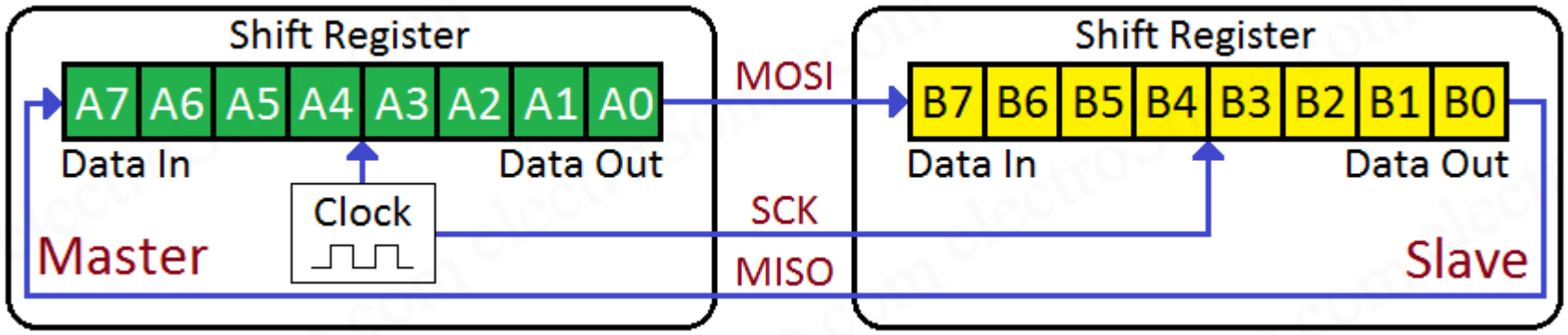
- این مرحله از نظر زمانی دقیقا همزمان با مرحله 3 اتفاق می‌افتد.
- Slave قصد دارد تا داده‌ای را که از Buffer خود دریافت کرده، همزمان با ورود هر بیت از سمت Master متقابلا به سمت Master ارسال نماید. (ارسال و دریافت همزمان – فول داپلکس)
- بدین جهت داده خود را از طریق خط MISO به سمت Master ارسال می کند.



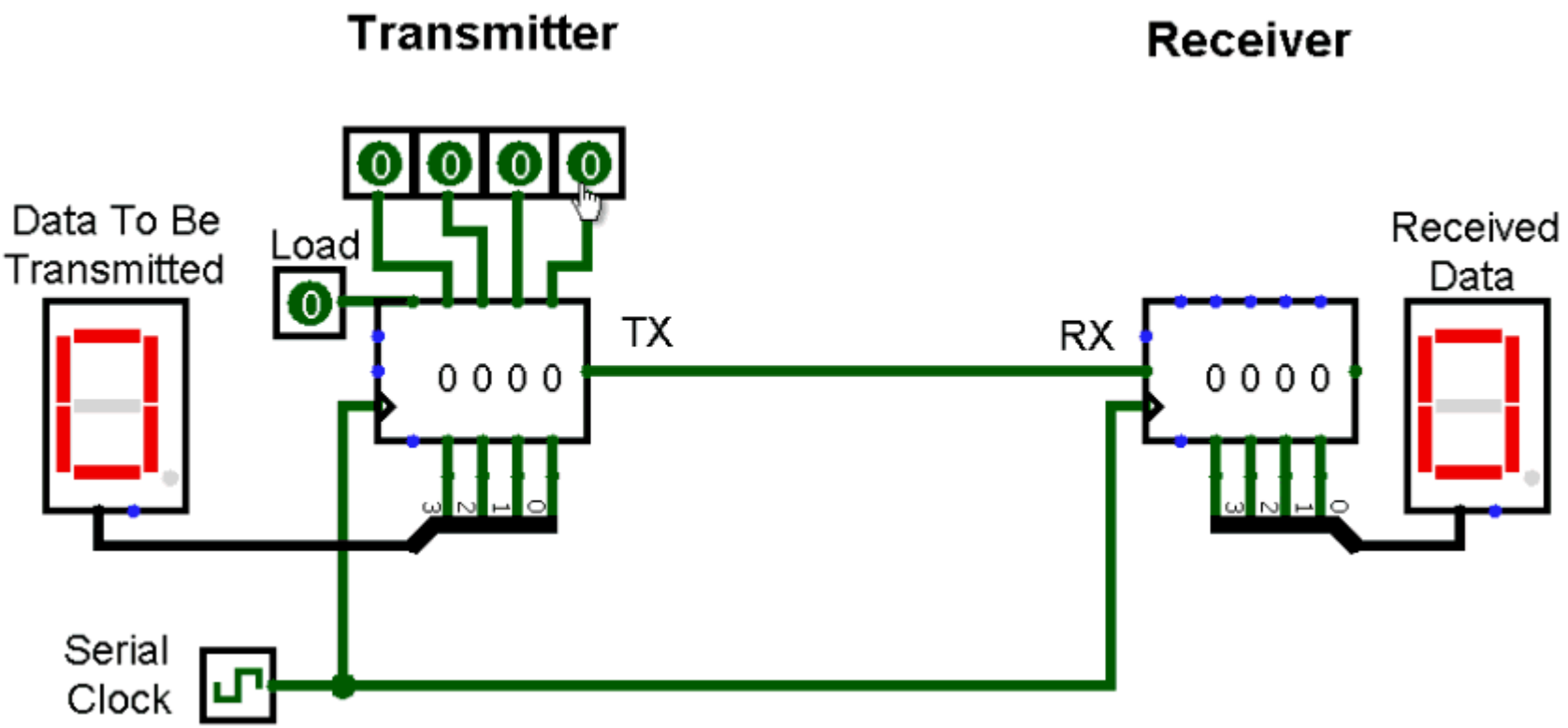
چگونگی انتقال داده در پروتکل ارتباطی SPI

5- شیفت رجیسترها

- هر دو دستگاه Master و Slave در خود یک شیفت رجیستر دارند که معمولا 8 یا 16 بیتی است.
- با هر پالس کلاک یک بیت از شیفت رجیستر Master وارد خط MOSI و همزمان یک بیت از شیفت رجیستر Slave وارد خط MISO می‌شود. بنابراین همزمان هر دو دستگاه یک بیت جدید دریافت می‌کنند.
- وقتی تبادل به طور کامل انجام می‌شود، Master خط CS را غیرفعال می‌کند و Slave دیگر به گذرگاه گوش نمی‌دهد.



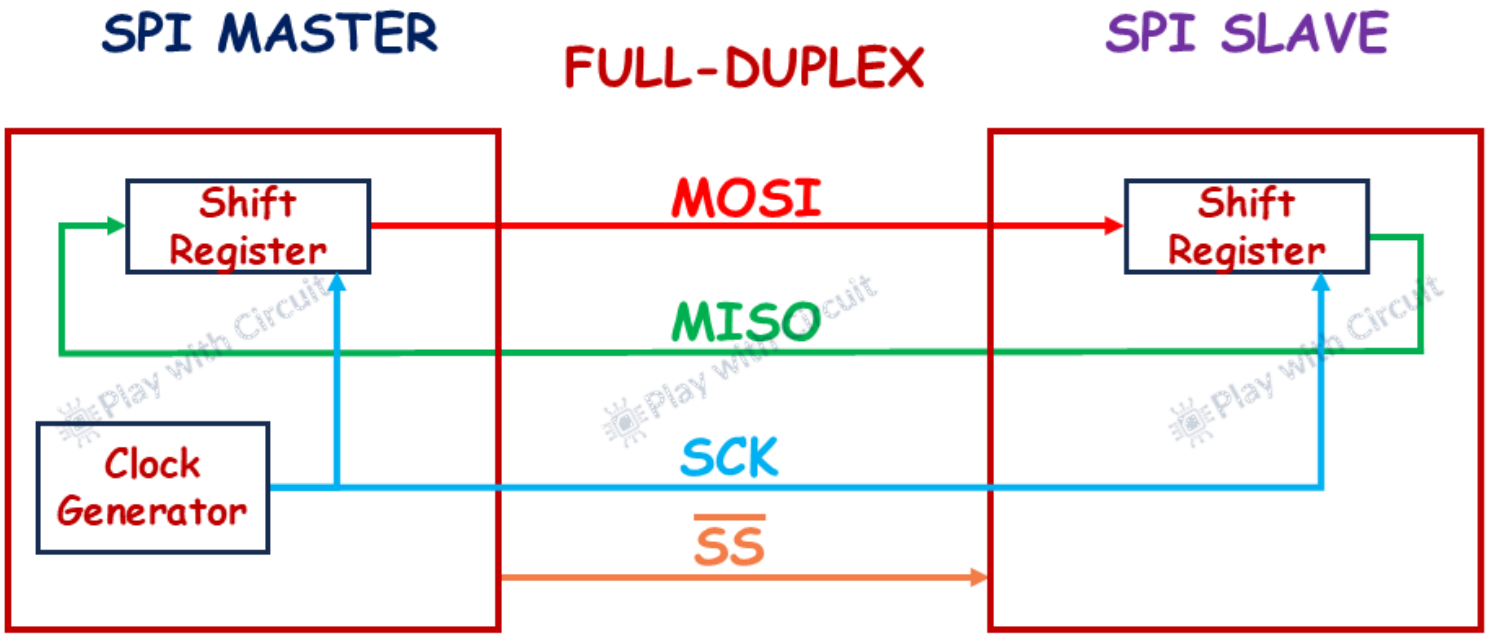
چگونگی انتقال داده از بافر به شیفت رجیسترها



انواع ارتباطات بین Master و Slave در ارتباط SPI

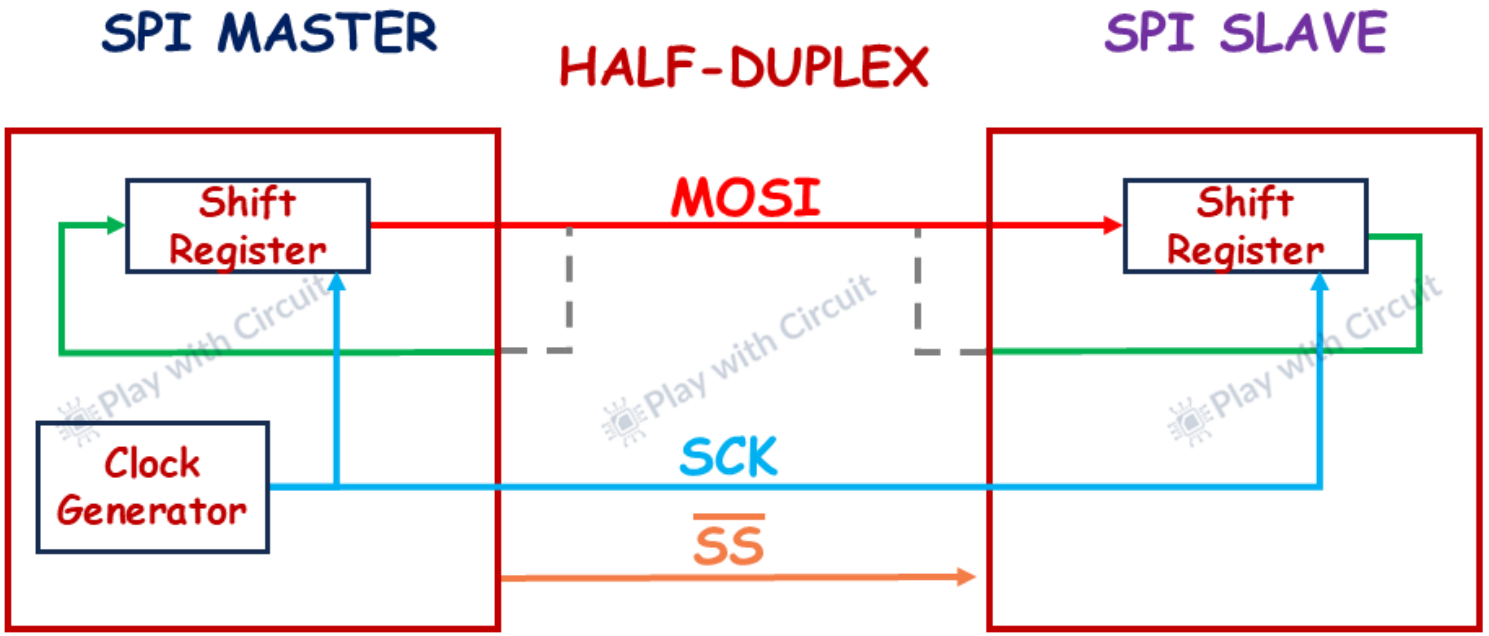
1- Full-Duplex (مثال: SD card or SPI Flash)

در حالت فول‌داپلکس، داده به‌صورت هم‌زمان در هر دو جهت منتقل می‌شود، به‌طوری‌که هم مستر و هم اسلیو می‌توانند به‌طور هم‌زمان داده ارسال و دریافت کنند.



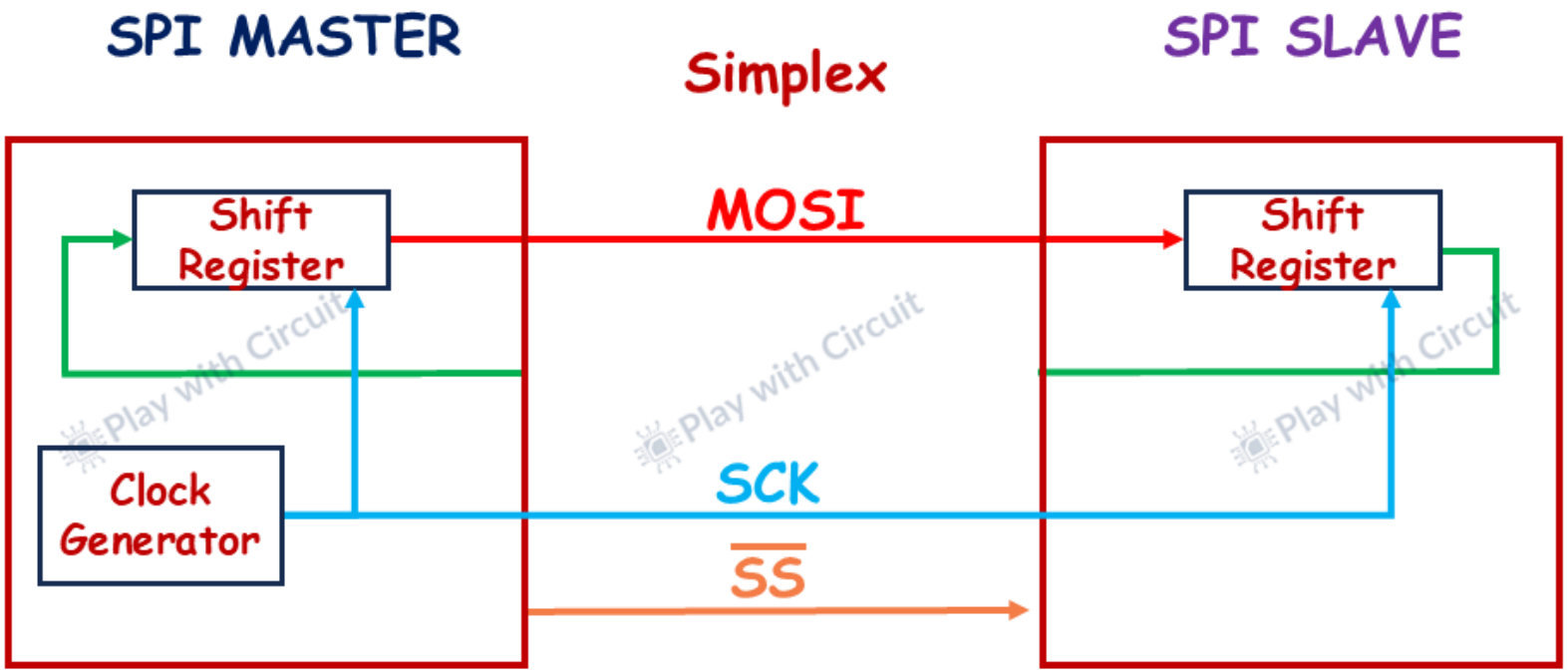
2- Half-Duplex (مثال: Temperature sensor)

در حالت نیمه‌دوطرفه (Half-Duplex)، داده می‌تواند در هر دو جهت منتقل شود، اما در هر لحظه فقط در یک جهت. یا مستر داده را به اسلیو ارسال می‌کند، یا اسلیو داده را به مستر بازمی‌گرداند. این حالت می‌تواند یا با اتصال داخلی خطوط MOSI و MISO از طریق پیکربندی ثبات‌های ویژه (SFR) گذرگاه SPI در میکروکنترلر پیاده‌سازی شود، یا با اتصال خارجی پایه‌ها به کمک قطعات فعال یا غیرفعال اضافی.



3- Simplex (مثال: Display module)

در حالت سیمپلکس (Simplex)، داده فقط در یک جهت منتقل می‌شود؛ یا از مستر به اسلیو، یا از اسلیو به مستر، اما نه هر دو به‌صورت هم‌زمان. در شکل زیر، جهت انتقال داده همواره از مستر به اسلیو خواهد بود.



پیکربندی‌های رابط سریال SPI

قطبیت کلاک (CPOL) و فاز کلاک (CPHA)

کلاک SPI را می‌توان با استفاده از دو ویژگی قطبیت کلاک (CPOL) و فاز کلاک (CPHA) تنظیم کرد. این دو ویژگی با هم تعیین می‌کنند که بیت‌های داده چه زمانی ارسال و دریافت شوند.

قطبیت کلاک (CPOL)

قطبیت کلاک، حالت بیکار (Idle) سیگنال کلاک (SCK) را زمانی که انتقال داده انجام نمی‌شود مشخص می‌کند. این ویژگی اهمیت دارد زیرا مبنایی برای زمان‌بندی و هم‌زمان‌سازی بین مستر و اسلیو ایجاد می‌کند.

CPOL = 0: وقتی داده‌ای منتقل نمی‌شود، سیگنال کلاک LOW است.

CPOL = 1: وقتی داده‌ای منتقل نمی‌شود، سیگنال کلاک HIGH است.

پیکربندی‌های رابط سریال SPI

قطبیت کلاک (CPOL) و فاز کلاک (CPHA)

کلاک SPI را می‌توان با استفاده از دو ویژگی قطبیت کلاک (CPOL) و فاز کلاک (CPHA) تنظیم کرد.

این دو ویژگی با هم تعیین می‌کنند که بیت‌های داده چه زمانی ارسال و دریافت شوند.

فاز کلاک (CPHA)

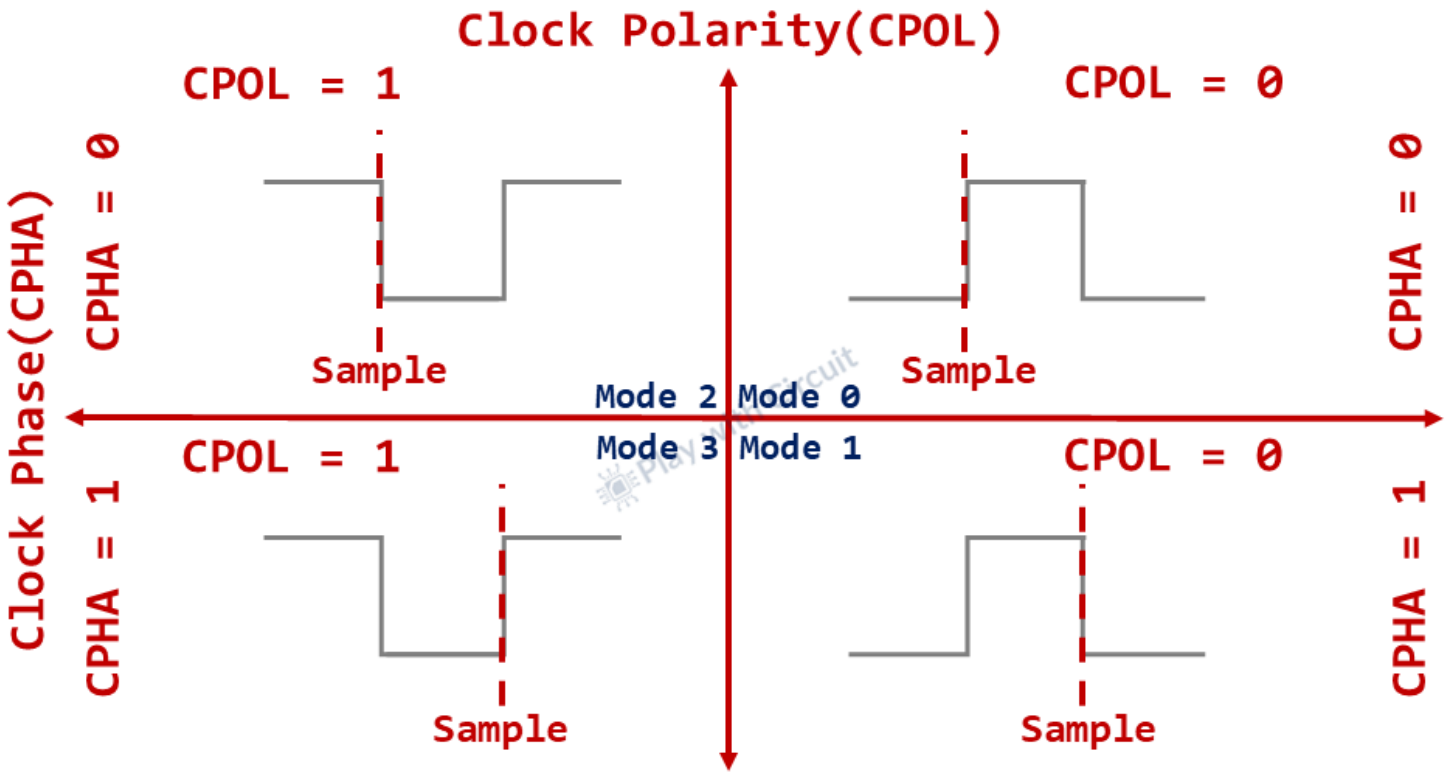
فاز کلاک تعیین می‌کند داده چه زمانی نمونه‌برداری (خوانده) و چه زمانی شیفت داده می‌شود در طول یک سیکل کلاک.

CPHA = 0: داده در اولین لبه کلاک نمونه‌برداری می‌شود و دومین لبه شیفت می‌شود.

CPHA = 1: داده در دومین لبه کلاک نمونه‌برداری می‌شود و در اولین لبه شیفت می‌شود.

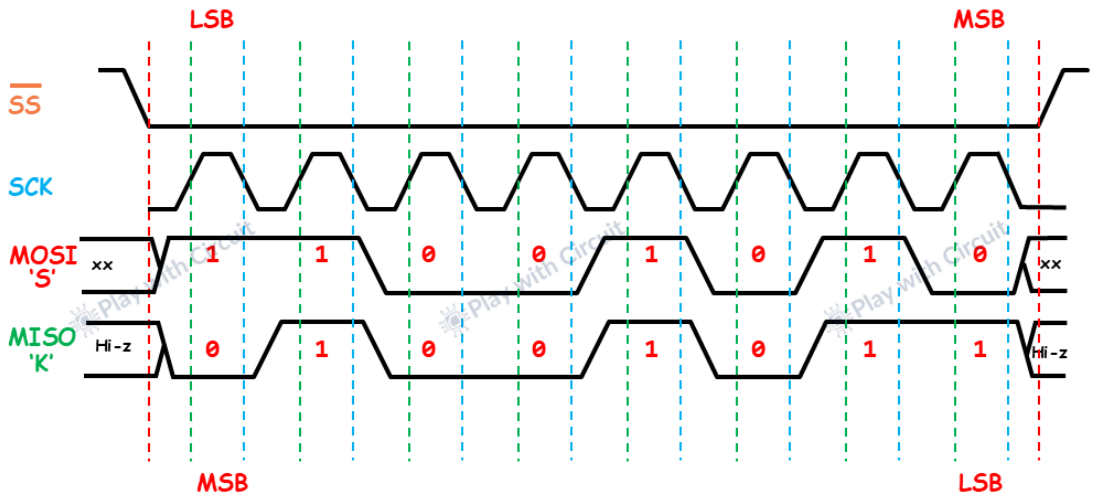
پیکربندی‌های رابط سریال SPI

قطبیت کلاک (CPOL) و فاز کلاک (CPHA)

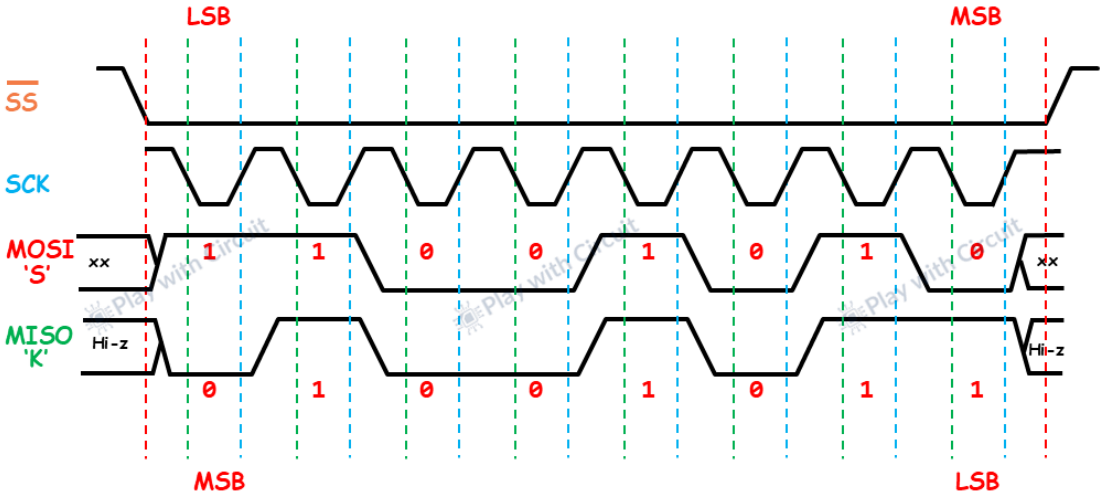


نکته: مستر باید هر دو بیت CPOL و CPHA را مطابق با نیاز اسلیو انتخاب کند تا ارتباط صحیح و هم‌زمان برقرار شود.

پیکربندی‌های رابط سریال SPI

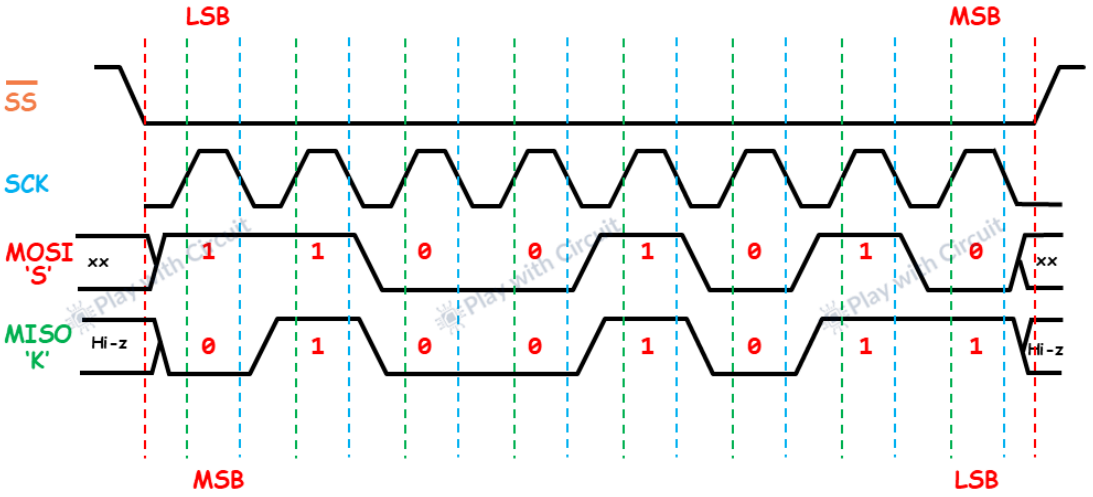


SPI Mode 0, CPOL = 0, CPHA = 0: SCK Idle State is LOW,
Data Sampled on Rising Edge and Shifted on Falling Edge

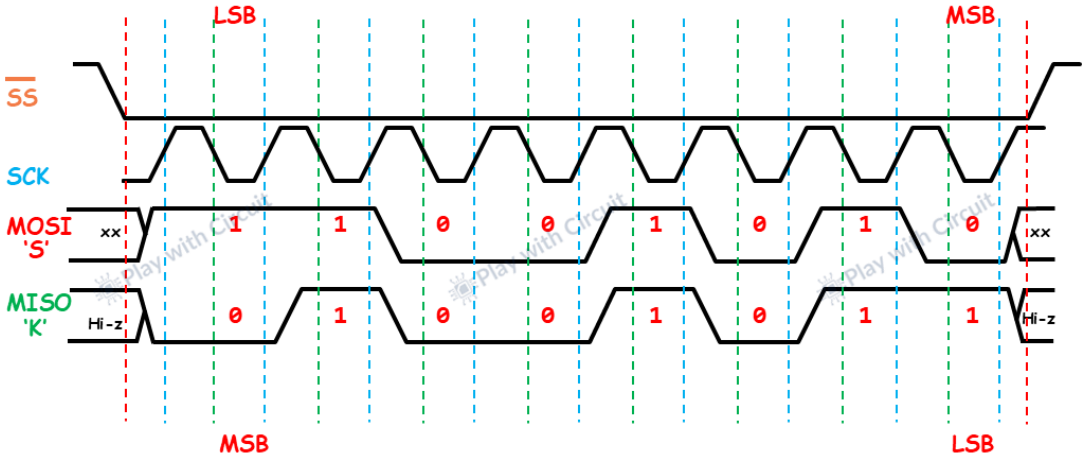


SPI Mode 2, CPOL = 1, CPHA = 0: SCK Idle State is HIGH,
Data Sampled on the Falling Edge and Shifted on the Rising Edge

پیکربندی‌های رابط سریال SPI



SPI Mode 0, CPOL = 0, CPHA = 0: SCK Idle State is LOW,
Data Sampled on Rising Edge and Shifted on Falling Edge

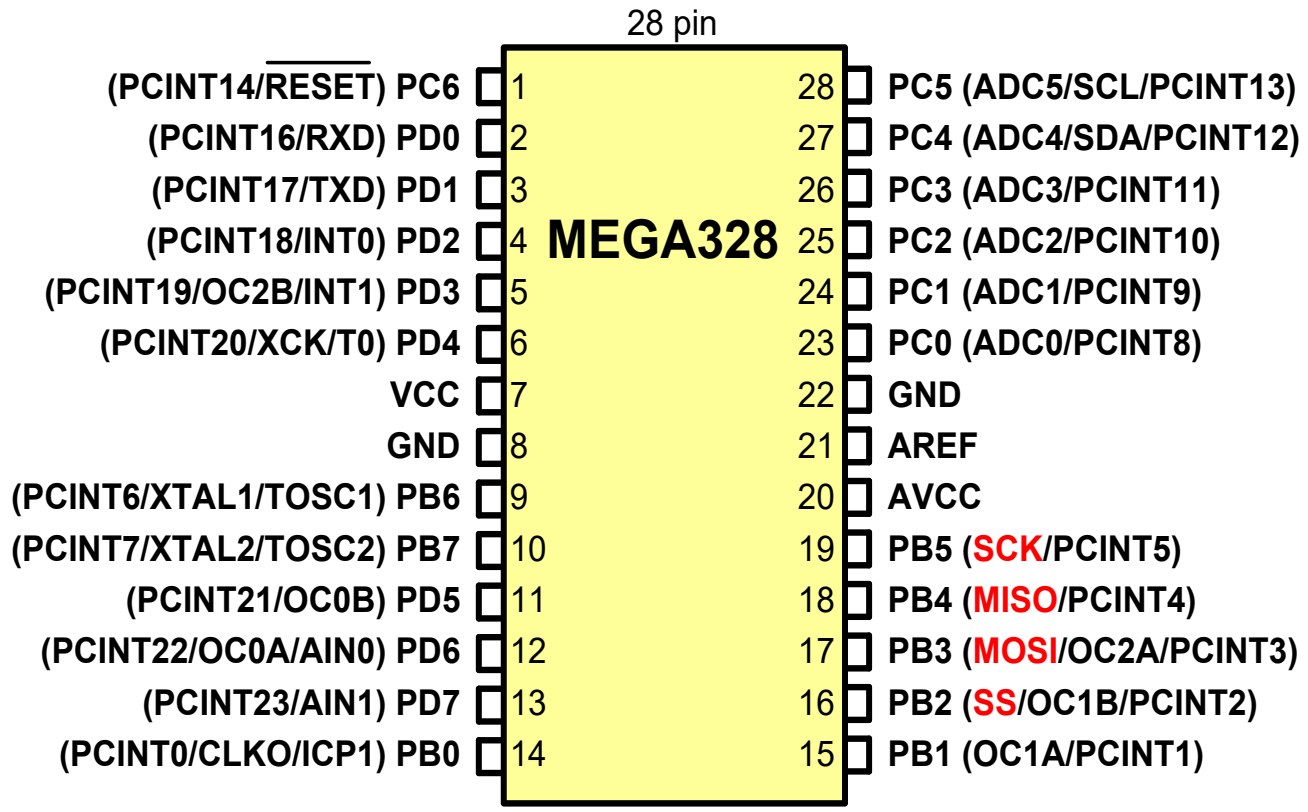


SPI Mode 1, CPOL = 0, CPHA = 1: SCK Idle State is LOW,
Data Sampled on the Falling Edge and Shifted on the Rising Edge

www.playwithcircuit.com

SPI pins in AVR

- MOSI (Master Out Slave In)
- MISO (Master In Slave Out)
- SCK
- SS



SPI Registers in AVR

- Control register:
  - SPCR (SPI Control Register)
- Status Register:
  - SPSR (SPI Status Register)
- Data Register:
  - SPDR (SPI Data Register)

SPSR (SPI Status Register)

- SPIF (SPI Interrupt Flag)
  - A serial transfer is completed.
  - The SS pin is driven low in slave mode
- WCOL (Write Collision)
- SPI2X (Double SPI Speed)

SPCR

SPCR:	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

- SPIE (SPI Interrupt Enable)
- SPE (SPI Enable)
- DORD (Data Order)
- MSTR (Master)
- CPOL (Clock Polarity)
- CPHA (Clock Phase)
- SPR1, SPR0 :SPI Clock Rate

SPI2X	SPR1	SPR0	SCK Freq.
0	0	0	Fosc/4
0	0	1	Fosc/16
0	1	0	Fosc/64
0	1	1	Fosc/128
1	0	0	Fosc/2 (not recommended)
1	0	1	Fosc/8
1	1	0	Fosc/32
1	1	1	Fosc/64

SPCR

SPCR:

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

- SP
- SP
- DO
- MSTR (Master)
- CPOL (Clock Polarity)
- CPHA (Clock Phase)
- SPR1, SPR0 :SPI Clock Rate

CPOL	CPHA	Data Read and Change Time	SPI Mode
0	0	Read on rising edge, changed on a falling edge	0
0	1	Read on falling edge, changed on a rising edge	1
1	0	Read on falling edge, changed on a rising edge	2
1	1	Read on rising edge, changed on a falling edge	3

			eq.
0	1	1	Fosc/128
1	0	0	Fosc/2 (not recommended)
1	0	1	Fosc/8
1	1	0	Fosc/32
1	1	1	Fosc/64

Program 1: Sending ‘G’ through SPI as a master

```
#include <avr/io.h>

#define MOSI 3
#define SCK 5
#define SS 2

int main (void)
{
 DDRB = (1<<MOSI)|(1<<SCK)|(1<<SS); //MOSI and SCK are output
 DDRD = 0xFF; //Port D is output
 SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); //enable SPI as master
 while(1) //do for ever
 {
  PORTB &= ~(1<<SS); //enable slave device
  SPDR = 'G'; //start transmission
  while(!(SPSR & (1<<SPIF))); //wait transfer finish
  PORTD = SPDR; //move received data to PORTD
  PORTB |= (1<<SS); //disable slave device
 }
 return 0;
}
```

Program 2: Sending ‘G’ through SPI as a slave

```
#include <avr/io.h>

#define MISO 4

int main (void)
{
 DDRD = 0xFF; //Port D is output
 DDRB = (1<<MISO); //MISO is output
 SPCR = (1<<SPE); //enable SPI as slave
 while(1)
 {
  SPDR = 'G';
  while(!(SPSR &(1<<SPIF))); //wait for transfer finish
  PORTD = SPDR; //move received data to PORTD
 }
 return 0;
}
```

Thanks!

Do you have any questions?

razeghizade@gmail.com

www.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution