lamaPLC Communication: SPI

The *Serial Peripheral Interface* (**SPI**) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

SPI devices communicate in full duplex mode using a master–slave architecture usually with a single master (though some Atmel and Silabs devices support changing roles on the fly depending on an external (SS) pin). The master (controller) device originates the frame for reading and writing. Multiple slave-devices may be supported through selection with individual chip select (CS), sometimes called slave select (SS) lines.

Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface, but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. The SSI protocol employs differential signaling and provides only a single simplex communication channel. For any given transaction SPI is one master and multi slave communication.

Interface

Single master to single slave: basic SPI bus example The SPI bus specifies four logic signals:

- **SCLK:** Serial Clock (output from master)
- **MOSI:** Master Out Slave In (data output from master)
- **MISO:** Master In Slave Out (data output from slave)
- **CS /SS:** Chip/Slave Select (often active low, output from master to indicate that data is being sent)

MOSI on a master connects to MOSI on a slave. MISO on a master connects to MISO on a slave. Slave Select has the same functionality as chip select and is used instead of an addressing concept.

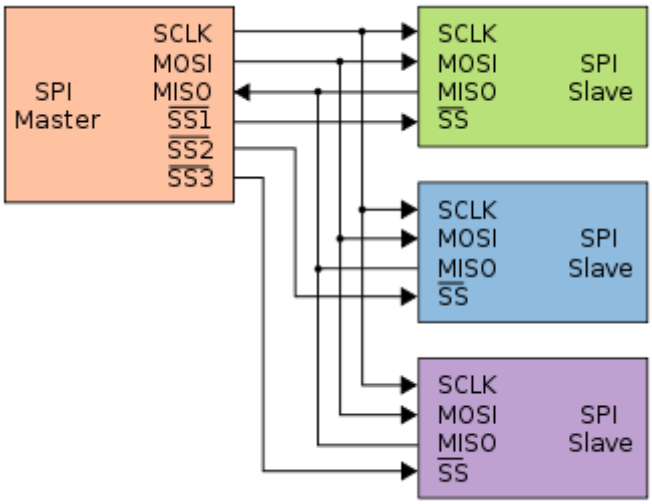
Note: on a slave-only device, MOSI may be labeled as SDI (Serial Data In) and MISO may be labeled as SDO (Serial Data Out)

The signal names above can be used to label both the master and slave device pins as well as the signal lines between them in an unambiguous way, and are the most common in modern products. Pin names are always capitalized (e.g. “Chip Select”, not “chip select”).

SPI network configuration

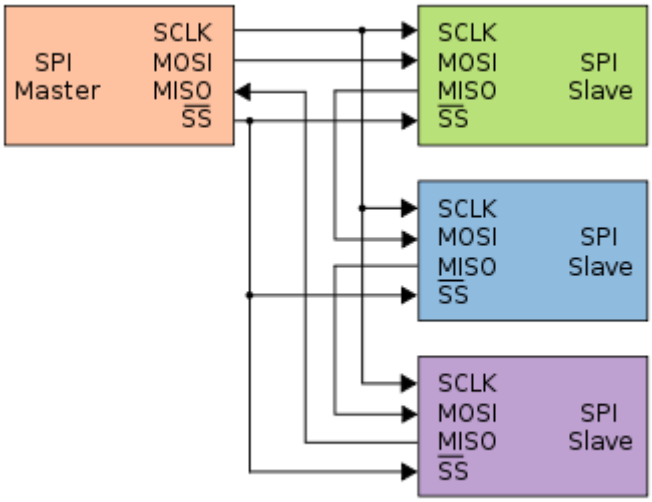
The SPI network can basically be built according to two structures:

Independent slave configuration



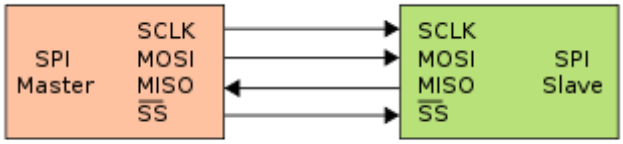
In this structure, the master must have as many selector SS ports as there are slaves in the network. Of these, it always conducts two-way communication only with the selected slave. Since the MISO outputs of the slaves would be connected, they should have three current levels: low, high, high impedance.

Daisy chain configuration



In this case, the MOSI signal sequence issued by the master is first sent to the MOSI of the first slave, and then via the MISO output to the MOSI of the next slave, and so on.

SPI communication



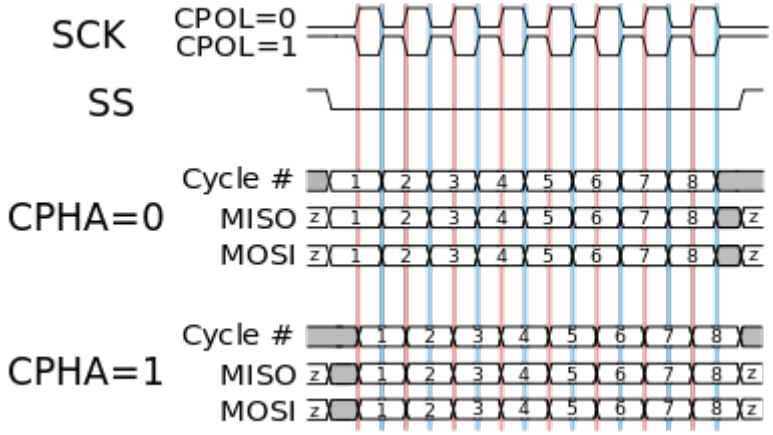
In SPI, the master always initiates the communication by changing the state of the SS signal from high to low towards the selected slave. After that, with the clock signal issued by the master, distribution takes place on both channels simultaneously, the registers used for data transfer are loaded synchronously, usually starting with the highest local values.

After filling the registers, they are stored by both units and the register filling is repeated until the end of the transfer. The registers are most often 8-bit, but there may also be 16/12-bit versions.

When the transfer is complete, the master stops the SLCK clock and sets the SS selector high again.

Clock polarity

The initial polarity of the clock signal (CPOL) can be either rising edge controlled (CPHA=0) or falling edge controlled (CPHA=1), as shown in the image below:



Mode

There are four communication modes available in SPI (MODE 0, 1, 2, 3), which basically determine the edge monitoring of SCLK:

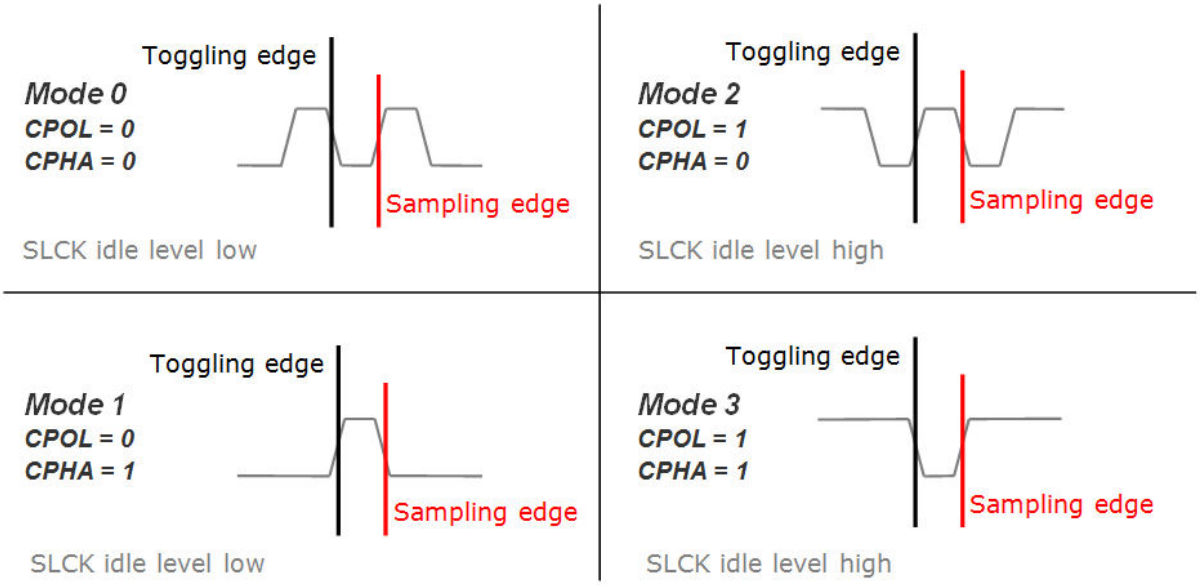


Figure 3 : SPI modes are defined with the parameters 'CPOL' – clock polarity and 'CPHA' – clock phase, which explicitly define 3 parameters: the edges used for data sampling and data toggling and the SCL clock signal idle level – that is the conventional level SCLK is set at when the bus is not in communication.

SPI / Arduino

spi.h

```
#ifndef _SPI_H_
#define _SPI_H_

#include <avr/io.h>


extern void spi_init();
extern void spi_transfer_sync (uint8_t * dataout, uint8_t * datain, uint8_t
len);
extern void spi_transmit_sync (uint8_t * dataout, uint8_t len);
extern uint8_t spi_fast_shift (uint8_t data);
```

spi.c

```
#include "spi.h"

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

#define PORT_SPI    PORTB
```

```
#define DDR_SPI     DDRB
#define DD_MISO     DDB4
#define DD_MOSI     DDB3
#define DD_SS       DDB2
#define DD_SCK      DDB5


void spi_init()
// Initialize pins for spi communication
{
    DDR_SPI &= ~((1<<DD_MOSI)|(1<<DD_MISO)|(1<<DD_SS)|(1<<DD_SCK));
    // Define the following pins as output
    DDR_SPI |= ((1<<DD_MOSI)|(1<<DD_SS)|(1<<DD_SCK));


    SPCR = ((1<<SPE)|               // SPI Enable
            (0<<SPIE)|              // SPI Interrupt Enable
            (0<<DORD)|              // Data Order (0:MSB first / 1:LSB
first)
            (1<<MSTR)|              // Master/Slave select
            (0<<SPR1)|(1<<SPR0)|    // SPI Clock Rate
            (0<<CPOL)|              // Clock Polarity (0:SCK low / 1:SCK hi
when idle)
            (0<<CPHA));             // Clock Phase (0:leading / 1:trailing
edge sampling)

    SPSR = (1<<SPI2X);              // Double Clock Rate

}

void spi_transfer_sync (uint8_t * dataout, uint8_t * datain, uint8_t len)
// Shift full array through target device
{
       uint8_t i;
       for (i = 0; i < len; i++) {
             SPDR = dataout[i];
             while((SPSR & (1<<SPIF))==0);
             datain[i] = SPDR;
       }
}

void spi_transmit_sync (uint8_t * dataout, uint8_t len)
// Shift full array to target device without receiving any byte
{
       uint8_t i;
       for (i = 0; i < len; i++) {
             SPDR = dataout[i];
             while((SPSR & (1<<SPIF))==0);
       }
}
```

```
uint8_t spi_fast_shift (uint8_t data)
// Clocks only one byte to target device and returns the received one
{
    SPDR = data;
    while((SPSR & (1<<SPIF))==0);
    return SPDR;
}
```

Sources

Wikipedia (here)

SPI topics on lamaPLC

Page	Date	Tags
• lamaPLC Communication: SPI	2026/04/23 21:51	bus, communication, spi, basic, arduino, ssi, sdi, miso, sdo
• lamaPLC: AI-Thinker LoRA products	2026/04/23 21:51	ai-thinker, lora manufacturer, communication, lora, modul, ra-01, ra-02, spi, arduino
• lamaPLC: Bi-Directional Logic Level Converter 3.3V ↔ 5V	2026/04/12 00:34	bi-directional, logic level converter, i2c, uart, spi
• LamaPLC: CJMCU-3901/PMW-3901 compact optical flow sensor module/IC by PixArt with SPI communication	2026/04/23 21:52	cjmcu-3901, cjmcu, pmw3901, pmw-3901, optical flow, sensor, pixart, spi, communication, arduino, code, pmw3901mb-txqt
• LamaPLC: CJMCU-6701: Biosensor for measuring Galvanic Skin Response (GSR) with SPI communication	2026/04/23 21:52	cjmcu, cjmcu-6701, acs758, acs-758, galvanic skin response, gsr, electrodermal activity, eda, spi, communication, arduino, code, sensor, healthcare
• lamaPLC: ESP32 / ESP8266	2025/11/22 00:07	esp8266, esp32, esp32-c2, esp32-c3, esp32-c5, esp32-c6, esp32-c61, esp32-h2, esp32-s2, esp32-s3, esp32-p4, espressif systems, communication, ethernet, ip, wi-fi, thread, zigbee, matter, homekit, bluetooth, mqtt, adc, spi, uart, i2c, i2s, rmt, pwm, usb, usb otg, twai
• LamaPLC: GY-9250 MPU-9250/6500 9-axis Attitude Sensor Board	2026/04/23 21:52	ak8963, gy-9250, mpu-9250, 9-axis, motion detection, magnetometer, communication, i c, i2c, spi
• lamaPLC: Max31865 RTD to Digital Converter - PT100/PT1000 Platine	2026/04/23 21:52	max31865, rtd, pt 100, pt 1000, temperature, spi, platinum, arduino, code, sensor, adafruit
• lamaPLC: MCP23017 / MCP23S17 16-Bit I/O Expander with Serial Interface I²C / SPI	2026/04/23 21:52	communication, i2c, mcp23017, mcp23s17, spi, i o expander, serial, cjmcu-2317, cjmcu

• LamaPLC: SC16IS750 / SC16IS752: One or two serial (UART) ports from microcontroller via I²C or SPI communication	2026/04/23 21:52	cjmcu-750, cjmcu-752, cjmcu, nxp, sc16is750, sc16is752, uart, serial, i2c, spi, modul, converter, arduino, code
• LamaPLC: Texas Instruments ADCs: Delta-sigma multi-channel Analog Converters with SPI communication	2026/04/23 21:52	ads111x, ads12xx, delta-sigma, converter, texas instruments, adc, spi, communication, sensor, arduino, code, ads1110, ads1112, ads1113, ads1114, ads1115, ads1118, ads1119, ads1220, ads1232, ads1234, ads1256, ads1261, ads1263, multi channel
• lamaPLC: Waveshare LoRA products	2026/03/07 01:46	waveshare, lora manufacturer, communication, lora, modul, usb-to-lora-xf02, core 1262, 1262, spi, arduino, rp2040-lora, rp2040
• Magnetic angle sensors	2026/03/05 21:19	magnetic angle sensor, magnetic flux, sensor, spi, i2c, pwm, communication, modul, as5047p, as5600, mt6701, mt6816, mt6835, tle5012b, amr, gmr, tmr, anisotropic magnetoresistive

bus, communication, SPI, basic, arduino, SSI, SDI, MISO, SDO