EMBEDDED SYSTEM COMMUNICATION PROTOCOLS: -

* Communication protocols are a set of rules and regulations that govern the communication between 2 or more devices or systems via any physical medium.
* These rules and regulations mainly consist of semantics, code syntax and synchronization parameters that need be followed and maintained in order to successfully establish communication between 2 or more systems.
* Protocols can be implemented either via software or hardware or by using a combination of both (which is typically the case in embedded systems) and there are many different types of communication protocols with each protocol having its own field of application.
* Embedded systems are one of the main applications of various communication protocols because they contain various types of sensors and peripherals that are interfaced with the microcontroller and in order to exchange data with them, communication protocols are needed.
* Example of such embedded system communication protocols are UART, USART, SPI, I2C, CAN, LIN, USB, MODBUS, etc. and each of these protocols have their own characteristic rules and regulations for communication.

Embedded System Communication Protocols

Inter – System Protocols Intra – System Protocols

These protocols govern the communication These protocols govern the communication

between microcontroller and other devices between microcontroller and sensors or any

like PC, laptop, recorders, analyzers, etc. other peripherals interfaced with it.

For Example – UART, USART, USB, etc. For Example – SPI, I2C, CAN, LIN, MODBUS, etc.

System – 2

System – 1

PC's, Laptops, Data Recorders, Oscilloscopes, Signal Generators, Analyzers, Plotters, etc.

In the above diagram, the pink lines represent Inter – System Protocols being used for Inter – System Communication between System – 1 and System – 2.

The purple lines represent Intra – System Protocols being used for Intra – System Communication between the components of System – 1 itself.

Some of the most commonly used terms while learning embedded system communication protocols are: -

1. Master: - It is the device which governs the entire communication process and it always initiates the communication process by generating the clock signal and is the commanding device.
2. Slave: - It is the device which receives commands from the master and performs the necessary action as per those commands and the clock signal is received by it from the master.
3. Full – Duplex Communication Protocol: - This type of protocol provides separate data lines for information exchange between master and slave.
4. Half – Duplex Communication Protocol: - This type of protocol provides only a single data line for information exchange between master and slave.

SPI and I2C protocols are synchronous communication protocols because they require the presence of a clock signal for data exchange between master and slave devices.

One very important criterion for successfully establishing synchronous communication between master and slave is that their clock frequencies must be exactly the same otherwise it can cause erroneous data exchange between the devices leading to misinterpretation of data.

SPI PROTOCOL: -

* SPI is one of the most fundamental embedded system communication protocols and it stands for Serial Peripheral Interface (Developed by Motorola in the early 80's).
* Many sensors and peripherals use this protocol like SD Card Readers, Flash Memories, RFID Readers, Graphical Displays, DAC's, ADC's, SRAM's, Accelerometers, Special Driver IC's, etc.
* The SPI interface between master and slave devices can be done in 2 ways i.e., 3 – Wire Interface and 4 – Wire Interface and typically the 4 – Wire Interface is frequently used and is shown as follows:-

Slave Device

Master Device

Any GPIO Pin

SCK

MOSI / SDO

MISO / SDI

CS / SS

SCK

MOSI / SDI

MISO / SDO

* In the 4 – Wire Interface, there are 4 pins needed for the master and slave each in order to be interfaced.
* The 4 pins are described as follows: -

1). CS / SS = Chip Select / Slave Select Pin. It is used by the master device to select the slave device with which it wants to interact. The master device can control this pin using any one of its normal GPIO pins. This chip / slave select pin is active low enabled i.e., the slave will be selected if the master gives logic 0 to this pin and the slave will be unselected if the master gives logic 1 to this pin. The master's GPIO pin acts as an output pin and the slave's chip / slave select pin acts as an input pin.

2). SCK = Serial Clock Pin. It acts as an output pin for the master and as an input pin for the slave. The master device always generates and provides the clock signal of the desired frequency through this pin in order to interact with the slave device and the slave device receives the clock signal through this pin. The desired clock signal frequency that must be generated by the master, can be found out by referring the technical documents of the slave device. The entire process of data exchange between master and slave is completely synced with the clock signal.

3). MOSI = Master – Out Slave – In Pin. It is the pin using which the master sends data to the slave and the slave receives data from the master through this pin. This pin acts as an input pin to the slave and as an output pin from the master. In the old convention, this pin is known as the SDO (Serial Data Out) pin for the master and it is known as the SDI (Serial Data In) pin for the slave.

4). MISO = Master – In Slave – Out Pin. It is the pin using which the slave sends data to the master and the master receives data from the slave through this pin. This pin acts as an output pin from the slave and as an input pin to the master. In the old convention, this pin is known as the SDO (Serial Data Out) pin for the slave and it is known as the SDI (Serial Data In) pin for the master.

* In the 3 – Wire Interface, connections for the pins SCK, MOSI and MISO remain the same but the CS / SS pin of the slave device will be connected to ground as shown below: -

Master Device

Slave Device

GND

CS / SS

SCK

MOSI / SDI

MISO / SDO

SCK

MOSI / SDO

MISO / SDI

\*\* The 3 – Wire Interface is not usually recommended for communication using "SPI protocol". It can be used when the master is connected to only one slave device and if it is required that the master device must interact with the slave continuously. If multiple slaves are connected to a master device, then using the 3 – Wire Interface will cause all the slave devices to get selected but the master can properly interact with one slave device at a time. Thus, the 4 – Wire Interface is highly recommended for all devices operating on SPI protocol.

\*\* SDI and SDO are the old conventions of representing MISO and MOSI lines of SPI interface but it is mostly not used now-a-days because this is a confusing convention as SDO and SDI each mean different lines from the master and slave device perspective i.e.,

* From the above discussion, it is seen that SPI protocol is a full – duplex communication protocol because there are 2 separate lines for data transfer from master to slave (MOSI) and from slave to master (MISO) and it is synchronous in nature as data exchange between master and slave is completely in synchronization with the clock signal from the master.
* SPI clock frequency has no well – defined, ideal range of values but it's upper limit is always half the operating frequency of the embedded system and its speed of data transfer is usually 10 Mbps but there are other varieties of SPI protocol like dual – SPI, quad – SPI, octo – SPI, etc. where speeds can go up to 80 Mbps.
* Since the transmitting device is capable of driving the logic levels of the data and clock lines either high or low, the time taken for change in logic states of the data and clock wires is minimized, thus, allowing SPI protocol to reach high data transfer rates as compared to other embedded protocols.
* Also, the data transfer speed depends on the number of bits transferred during each SPI clock cycle. For example: -

Standard SPI Interface = 1 bit per SPI clock cycle i.e., 1 byte is transferred in 8 SPI clock cycles.

Dual – SPI Interface = 2 bits per SPI clock cycle i.e., 1 byte is transferred in 4 SPI clock cycles.

Quad – SPI Interface = 4 bits per SPI clock cycle i.e., 1 byte is transferred in 2 SPI clock cycles.

Octo – SPI Interface = 8 bits per SPI clock cycle i.e., 1 byte is transferred in 1 SPI clock cycles.

* SPI protocol supports single master and multiple slave configuration only i.e., there can be only one master device and multiple slave devices can be interfaced with it and this can be achieved using 2 methods: -

a). Independent Slave Method

b). Daisy – Chain Method

INDEPENDENT SLAVE METHOD: -

* In this method, each slave is interfaced with the master independently i.e., each slave has its own chip / slave select line and the connections for SCK, MOSI and MISO pins are common to all the slaves as shown below: -

SPI

Slave 3

SPI

Slave 2

SPI

Slave 1

SPI

Master

MISO

MISO

MISO

MISO

MOSI

MOSI

MOSI

MOSI

SCK

SCK

SCK

SCK

GPIO Pin 1

GPIO Pin 2

GPIO Pin 3

CS1 / SS1

CS2 / SS2

CS3 / SS3

* From the above diagram, we can see that number of chip / slave select lines is equal to the number of slave devices interfaced and the remaining 3 lines i.e., SCK, MOSI and MISO are commonly connected to all the slave devices.
* Here, the data takes approximately the same amount of time to reach all the slave devices since the data lines have been made common to all of them. For example, if standard SPI protocol is used, then all the slave devices will receive their corresponding data or command in 8 SPI clock cycles.

Slave 1 is Selected

CS1 / SS1

SCK

Slave 2 is Selected

CS2 / SS2

Slave 3 is Selected

CS3 / SS3

8 SPI Clock Cycles

8 SPI Clock Cycles

8 SPI Clock Cycles

DATA / CMD 3

DATA / CMD 2

DATA / CMD 1

MOSI

DATA 1

DATA 2

DATA 3

MISO

* Since, each slave device has its own chip / slave select line, the master device can independently interact with each one of them but multiple chip select lines should not be made active because selecting multiple slaves for data exchange will cause data corruption on the MISO line as the master will not be able to know as to which slave device is sending the particular byte of data.
* The major disadvantage of this method is that the number of slave devices that can be interfaced with the master device, becomes limited because the number of chip / slave select lines will increase if more slaves are interfaced and this leads to increase in circuit complexity and exhaustion of GPIO pins of the master device.

DAISY – CHAIN METHOD: -

* In this method, the chip / slave select lines and the serial clock lines of all slaves are joined together but the data transfer lines are connected in a cascaded format as shown in the diagram below: -

MISO3

MISO2

MISO1

MOSI3

MOSI2

MOSI1

MISO

MOSI

GPIO Pin

CS / SS

CS / SS

CS / SS

SCK

SCK

SCK

SCK

SPI

Slave 1

SPI

Master

SPI

Slave 2

SPI

Slave 3

* Here, all the slaves are selected and they receive the clock signal at the same time and the data is sent from master to 1st slave and then 1st slave sends the data to 2nd slave and so on but the last slave sends the data back to the master.
* Since the data propagates from one slave to another, all the slave devices will not receive the data or command in the same number of clock cycles. For example, if standard SPI protocol is used, then 1st slave will receive data after 8 SPI clock cycles, 2nd slave will receive data after 16 SPI clock cycles and 3rd slave will receive data after 24 SPI clock cycles.

All slaves are selected simultaneously

CS / SS

SCK

8 SPI Clock Cycles

8 SPI Clock Cycles

8 SPI Clock Cycles

DATA 3

DATA 2

DATA 1

MOSI 1

DATA 2

DATA 3

----------

MISO 1 / MOSI 2

DATA 3

----------

----------

MISO 2 / MOSI 3

----------

----------

----------

MISO 3

* One advantage of this method is that the circuit complexity will be much less as compared to independent slave method because the chip / slave select line is common to all the slaves.
* However, the number of slaves that can be interfaced using this method is also limited since the data propagation time increases proportionately depending on the position of the slave in the chain which happens to be a disadvantage and also practically, very few devices support this method.

DETAILED WORKING STEPS OF SPI PROTOCOL: -

STEP – 1: Choose which devices are going to act as master and slave.

STEP – 2: Refer technical document of the slave device and find out the required SPI clock phase, clock polarity and clock frequency.

STEP – 3: Configure master device to generate the clock signal as per the above parameters.

STEP – 4: Select the slave device by making the chip / slave select line as logic 0.

STEP – 5: Send the command or data from master device to slave device.

STEP – 6: Receive the data from the slave device and store it for processing.

STEP – 7: De – select the slave device by making chip / slave select line as logic 1.

The data is transferred between master and slave in a serial manner using shift registers as shown in the following figure: -

BIT SHIFTING SEQUENCE

BIT SHIFTING SEQUENCE

MISO

MSB

MSB

LSB

LSB

SLAVE

SHIFT REGISTER

MASTER

SHIFT REGISTER

CS / SS

SCK

SCK

SPI

CLOCK

GENERATOR

MOSI

* Here, the data is shifted out from the master's shift register and shifted into the slave's shift register, one bit at a time per clock pulse of the SPI clock generator.
* As per the direction of the red arrows in the above diagram, the MSB of the data byte will be shifted out 1st and then the subsequent bits (conventional method). If the LSB of the data byte would be shifted out 1st and then the subsequent bits, then the direction of the red arrows will get reversed.
* If there are multiple slaves present, then the overall SPI clock frequency will be decided by the slave device that has the lowest value of SPI clock frequency.
* The datasheet of the slave device contains information about the various commands that master needs to send to the slave for data extraction.
* The datasheet also contains the format in which the slave device will provide the data to the master device (some slaves return 2 or more bytes of data to a single command from the master whereas some slaves return only 1 byte of data to a single command from the master).
* Some slave devices provide the necessary information directly whereas other slave devices provide data that needs further processing (mostly mathematical) and this information is also given in the slave's datasheet.
* When data transfer begins either from master to slave or vice – versa, the master device will decide whether the MSB or the LSB of the data byte will be shifted out 1st from the shift registers and the configurations to be done for the same will be available in the master's datasheet.

SPI MODES AND THE ROLE OF CLOCK PHASE AND POLARITY: -

* It is necessary that the SPI clock frequency be configured properly for successful communication between master and slave devices but clock frequency is not the only deciding factor.
* Clock phase and clock polarity also play very vital roles in the setup of a successful SPI communication between master and slave but it often happens that these 2 parameters are neglected during programming.
* Clock polarity (CPOL) will decide the idle state of the SPI clock and clock phase (CPHA) will decide the edges of the clock pulse (rising or falling), when data will be sampled and shifted out.
* Idle state of the clock refers to the logic on the SCK line when the chip / slave select line is at logic 1 i.e., when the slave is de – selected.
* So, depending on the combinations of CPOL and CPHA values, 4 different modes of SPI protocol are formed and they are listed below: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SPI  Mode | CPOL | CPHA | Idle State of SPI Clock | SPI Clock Edge used to Sample and / or Shift Data |
| 0 | 0 | 0 | Logic 0 / Low | Data sampled on rising edge and shifted out on falling edge of SPI clock |
| 1 | 0 | 1 | Logic 0 / Low | Data sampled on falling edge and shifted out on rising edge of SPI clock |
| 2 | 1 | 0 | Logic 1 / High | Data sampled on rising edge and shifted out on falling edge of SPI clock |
| 3 | 1 | 1 | Logic 1 / High | Data sampled on falling edge and shifted out on rising edge of SPI clock |

* The timing diagrams of all 4 SPI modes are shown below: -

CS / SS

CS / SS

SCK

CS / SS

SCK

SCK

SCK

SPI Mode 2: CPOL = 1 & CPHA = 0

SPI Mode 3: CPOL = 1 & CPHA = 1

SPI Mode 1: CPOL = 0 & CPHA = 1

SPI Mode 0: CPOL = 0 & CPHA = 0

CS / SS

\*\*Pink vertical lines represent data sampling whereas Sky Blue vertical lines represent data shifting.

MERITS / ADVANTAGES OF SPI PROTOCOL: -

* Speed of data transfer is very high (10 Mbps to 100 Mbps or more).
* Supports full – duplex communication i.e., separate data transfer lines from master to slave and vice – versa.
* Simple implementation and low protocol overhead i.e., extra or unused information bits are not present in it.
* Only one master device is allowed, hence, there is no chance for conflict.
* Low power consumption due to lack of additional hardware like pull – up resistors, etc.
* Since chip / slave select line is used to select the slave device, there is no need for individual slave addresses.
* Master clock is configured based on the speed of the slave device.
* Any length of data can be transferred between master and slave devices (this length is limited by the maximum number of bits that can be processed by the master and slave).

DE – MERITS / DISADVANTAGES OF SPI PROTOCOL: -

* Circuit complexity is increased with a greater number of slaves as each slave requires its own chip / slave select line and also not many slaves support daisy – chain connection.
* In case of multiple slave devices, the overall speed of data transfer is decided by the slave device with lowest operating frequency.
* Higher chances of noise or crosstalk due to a greater number of wires in the interface.
* There is no acknowledgement mechanism due to which there is no confirmation that the data is being received correctly by the devices.
* Master has to control all communication and slaves can't talk to each other.
* There is no mechanism for checking errors in data transfer.
* Communication distance is shorter as compared to RS – 232, RS – 485 protocols, etc.
* There are no official standards for this protocol and it is thus, used in application specific implementations.