Software Application development at device level

Introduction to serial communication protocols

Serial Communication Protocols – UART, SPI and I2C

Serial vs. Parallel Communication

Serial Communication	Parallel Communication
Data is transferred bit by bit	Data is transferred in chunks of bits
Requires data formatting	Doesn't require any data formatting
Parallel to serial and Serial to Parallel data converters are required	No Serial – Parallel data converters are required
Usually governed by some higher-layer protocols	No protocols are required
Only one line/pin is required for transmission, so very efficient	Parallel data lines are required so less efficient for implementation
Less speed e.g. UART, SPI, I2C, CAN & USB	High Speed e.g. Memory interfaces with CPU

Synchronous vs. Asynchronous Communication Protocols

Synchronous Communication	Asynchronous Communication
Data is transferred synchronously with clock	Data is not transferred along with clock
Data rate is same as clock rate	Data rate has to be established
Can’t send/receive data without clock	Data can be transferred/received any time
Limited data formatting	Data formatting is must
Used for on board communication	Used for off-board communication
Examples : SPI and I2C	Examples : UART, CAN & USB

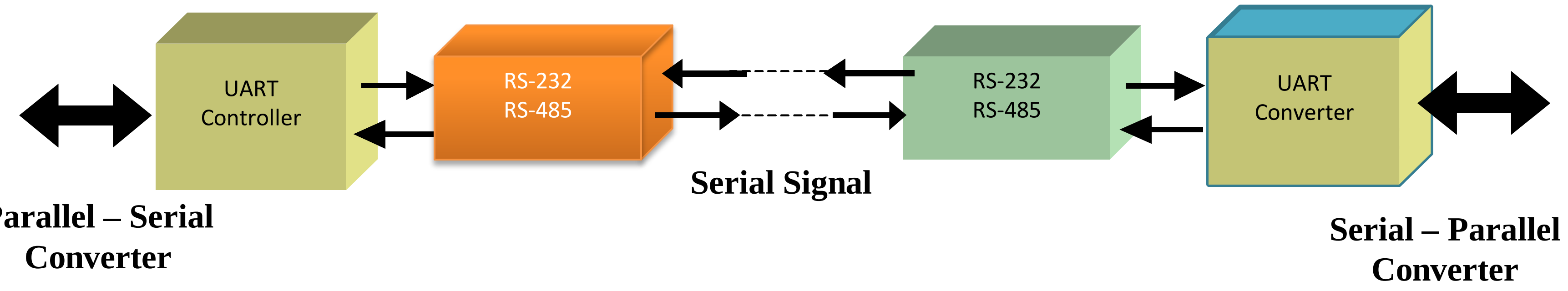
Universal Synchronous/Asynchronous Receiver Transmitter (USART/UART)

What is UART

- Synchronous/Asynchronous communication
- Reduces design complexity – by having a simple bus serial bus design
- UART – Asynchronous variant widely used
- UART controllers have Parallel – Serial & Serial – Parallel data converters
- Transreceivers may used to transmit data as per certain bus standards
- Provides a low-cost design for communication

UART - Design

- Serial to parallel Data conversion
- Data Buffering
- Baud Rate Generation
- Handshaking Signals
- Start & Stop Bits
- Parity Bits



UART Features

- Flexible full duplex communication with Tx & Rx pins
- Industry standard NRZ Asynchronous communication
- USART – Supports synchronous half-duplex communication over single line
- Supports variety of Baud rate – Standard and Programmable

UART Data Frame

- Prior to reception or transmission, Line is held in idle state
- Start bit (0) – Indicates start of frame
- Data bits (8 or 9 bits), lsb first
- 1/1.5/2 Stop bits (1) – Indicating end of frame, usually 1 stop bit is used
- Totally 10 bits are transmitted normally – 1 start bit, 8 data bits and 1 stop bit

UART Flow Control

- RTS :- Request to send, active low asserted signal indicates UART controller is ready to receive data from other device
- CTS :- Clear to send , active low asserted signal indicates UART controller is ready to send data to other device
- Usually RTS of one device is connected to CTS of other device
- CTS/RTS if asserted high indicates to stop sending or receiving data

Serial peripheral Interface (SPI)

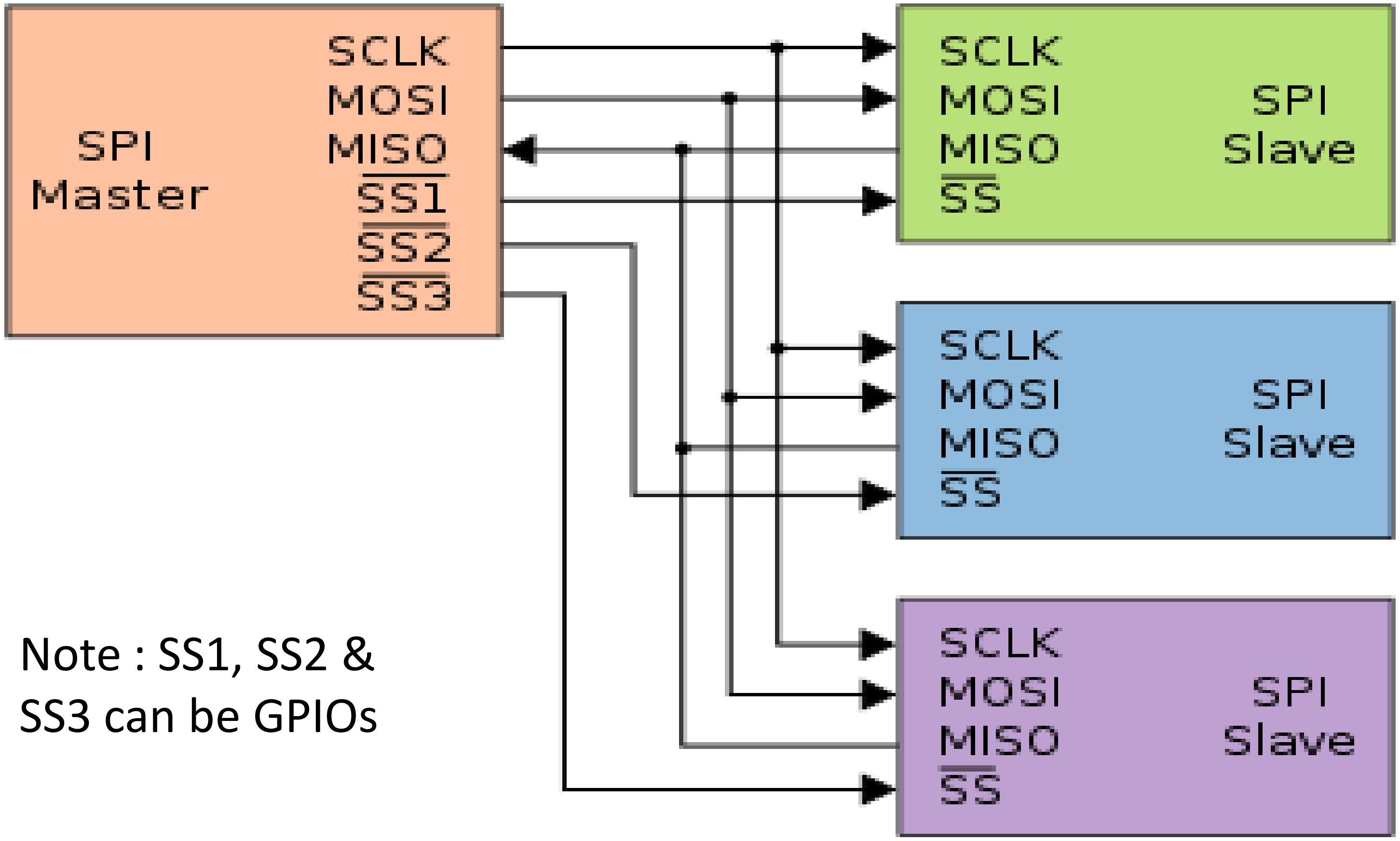
SPI Features

- Originally developed by Motorola
- SPI is synchronous communication protocol used for communication between processor and peripherals
- Supports speeds upto 10Mbs
- Synchronous full/half duplex communication
- Master & Slave functionality
- Single master & multiple slave configuration
- No. of slaves dependents on hardware design and line capacitance

SPI Features Contd.

- SPI has four lines
  - MOSI (Master Output Slave Input)
  - MISO (Master Input Slave Output)
  - SCLK (Serial Clock)
  - SS (Slave select)
- Always Master initiates the communication, generates clock and terminates the communication
- Slave needs to listen to master and participate in the communication when its SS line is asserted by master

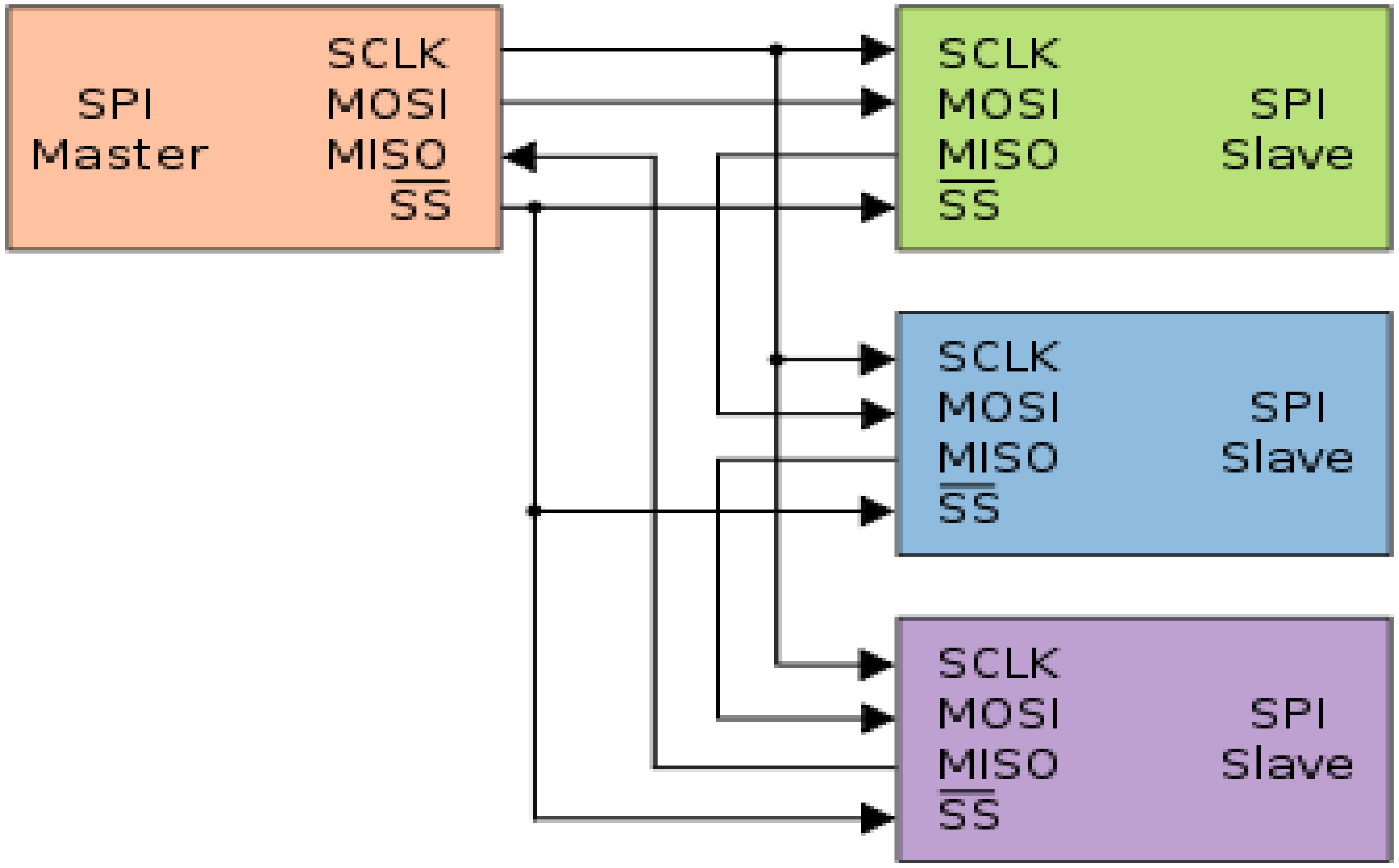
Single Master & Multiple Slave Conf.



Note : SS1, SS2 & SS3 can be GPIOs

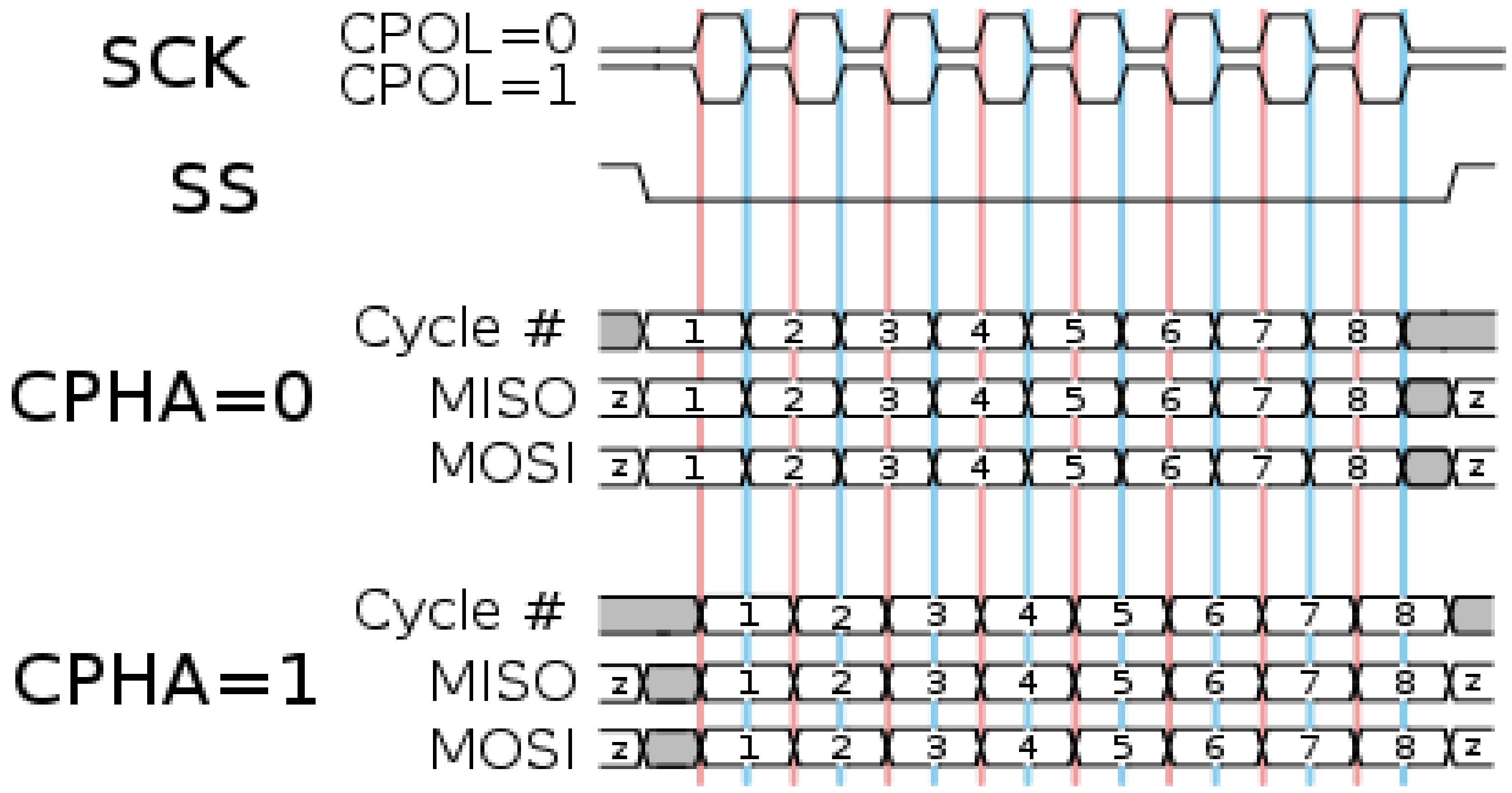
Ref: https://commons.wikimedia.org/wiki/File:SPI_three_slaves.svg

Multiple Slave daisy chain conf.



Ref: https://commons.wikimedia.org/wiki/File:SPI_three_slaves_daisy_chained.svg

SPI Clock and Polarity



Ref: https://commons.wikimedia.org/wiki/File:SPI_timing_diagram2.svg

SPI Clock and Polarity Contd.

- SPI CPOL & CPHA together determines when the data is stable to be read and when a change of data is allowed

- There are four possible modes of operation that can be selected by software

SPI Advantages

- Faster than asynchronous serial communication protocols
- Full- duplex communication
- Receiver hardware could consists of simple shift register
- No data framing
- Simple, efficient and faster on board communication
- Support for multiple slaves

SPI Disadvantages

- Master – Slave communication, need to be always master initiated communication
- Hardware based slave addressing
- Not a multi master bus
- 3 lines involved in communication & 1 line for selecting slave
- Communication data format needs to be well-defined in advance
- No comprehensive error handling mechanisms

Inter-Integerated Circuit (I2C)

I2C Bus Protocol

- Invented by Philips semiconductor (Now NXP semiconductor) in 1982.
- De-facto industry standard protocol followed by most of the semiconductor vendors
- 2 wire Serial Synchronous half-duplex communication protocol
- 2 wires are SDA and SCL with bidirectional communication
- SDA – Serial Data Line & SCL – Serial Clock Line

I2C Bus Protocol Contd.

- I2C bus is a multi-master bus following master-slave communication
- Every device on the I2C bus has a unique 7 bit or 10 bit address
- I2C bus follows – CSMA/CD protocol
- Wired-AND logic based Arbitration
- Clock Synchronization
- Well-defined data frame format

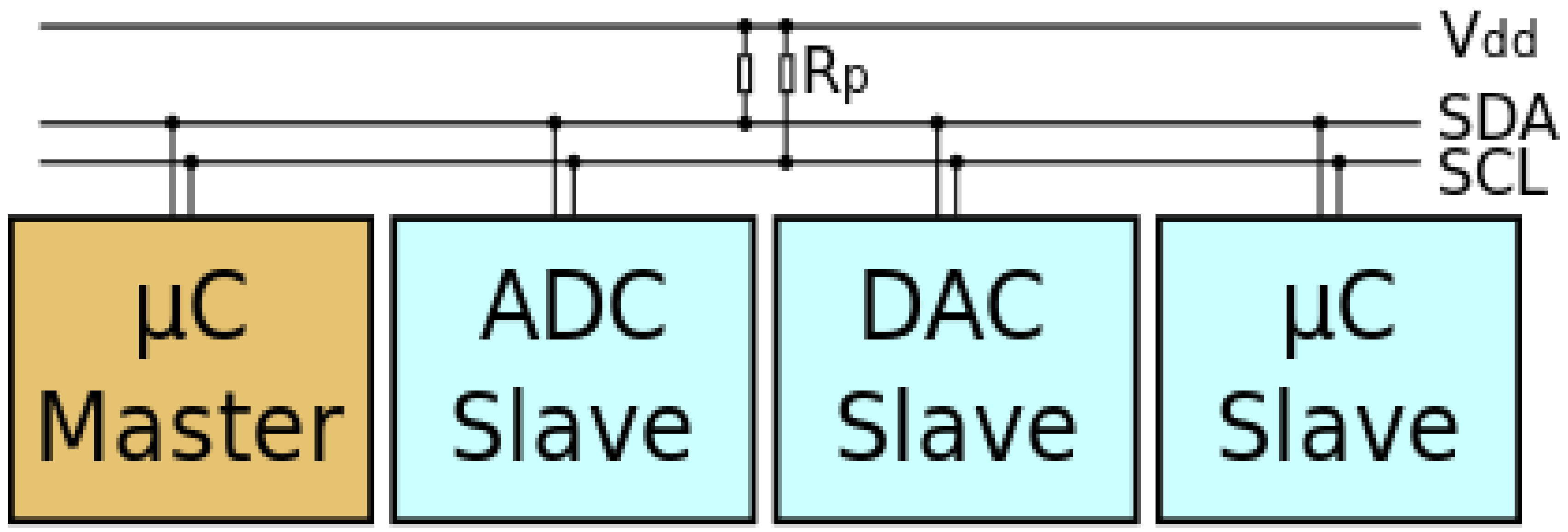
I2C Bus Protocol Applications

- I^{2}C is appropriate for peripherals where simplicity and low manufacturing cost are more important than speed
- Widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication
- microcontroller interfacing with sensors

I2C Protocol Features

- Designed for low-cost, medium data rate applications
- Typical Characteristics :
  - synchronous
  - serial, byte-oriented
  - Bit wise non destructive arbitration
  - moderate speeds:
    - standard mode : 100 Kbits/s
    - fast mode : 400 Kbits/s
    - high speed mode : 3.4 Mbits/s
    - ultra-fast mode : 5 Mbits/s
- Most microcontrollers come with built-in i2c controllers

I2C hardware



- Vdd = 3.3 V or 5 V
- Open collector/drain configuration
- SDA and SCL line pulled high by default when line is free

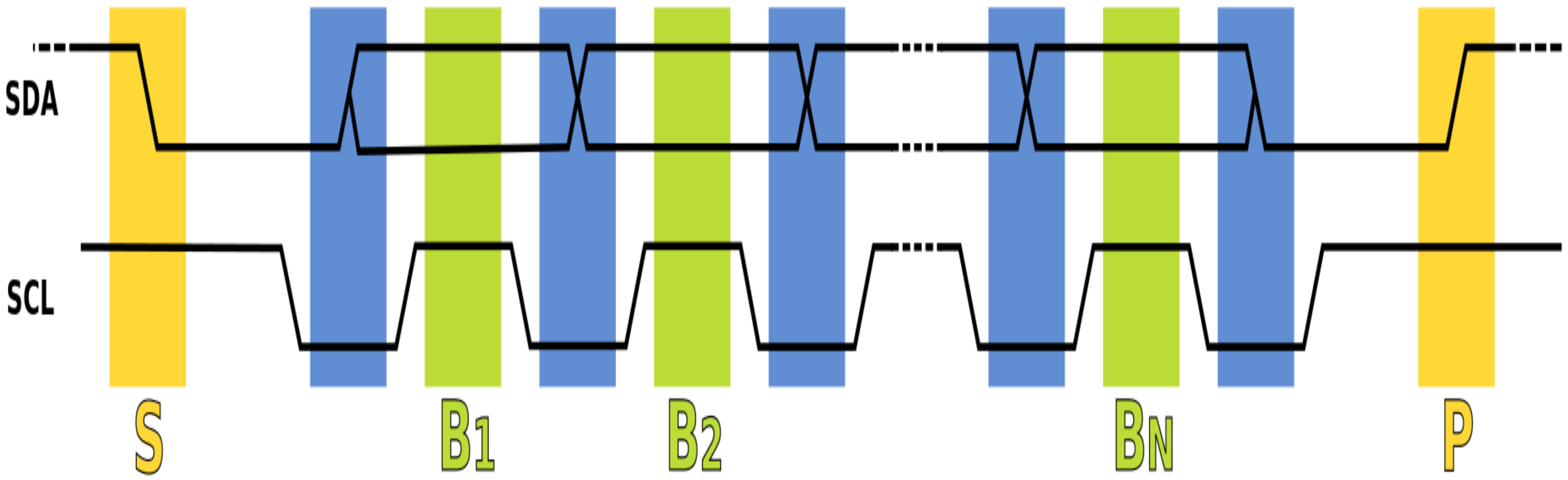
I2C Data Link Layer functions

- Every device on the bus has a unique address
  - Set by either device and or by system designer
  - 7 bit address commonly used (10 bit extension)
  - 7 bit slave address is followed by 8th bit to determine the direction of data flow
- Reserved address (0000000) for broadcast
- Bus transactions are byte oriented after address and r/w bit
- After every byte, receiver needs to Ack

I2C Master & Slave Operations

- Master
  - Always starts or initiates communication
  - Drives the clock line (SCL)
  - Always terminates the transaction
- Slave
  - Need to respond to master
  - Can’t initiate any communication by itself
  - EEPROM, LCDs, RTCs, Sensors

I2C Timing Diagram



S – Start condition, SDA 1→0 when SCL is high

P – Stop condition, SDA 0 →1 when SCL is high

Ref: https://en.wikipedia.org/wiki/I%C2%B2C#/media/File:I2C_data_transfer.svg

I2C Bus Arbitration

- Arbitration is a mechanism to determine who will be the winning master in a multi master scenario
- Based on simple Wired-AND logic, the master whose SDA line is 0 while others is 1 will be the winning master
- The losing master has to immediate go back to slave mode and retry after STOP condition

I2C Clock Synchronization

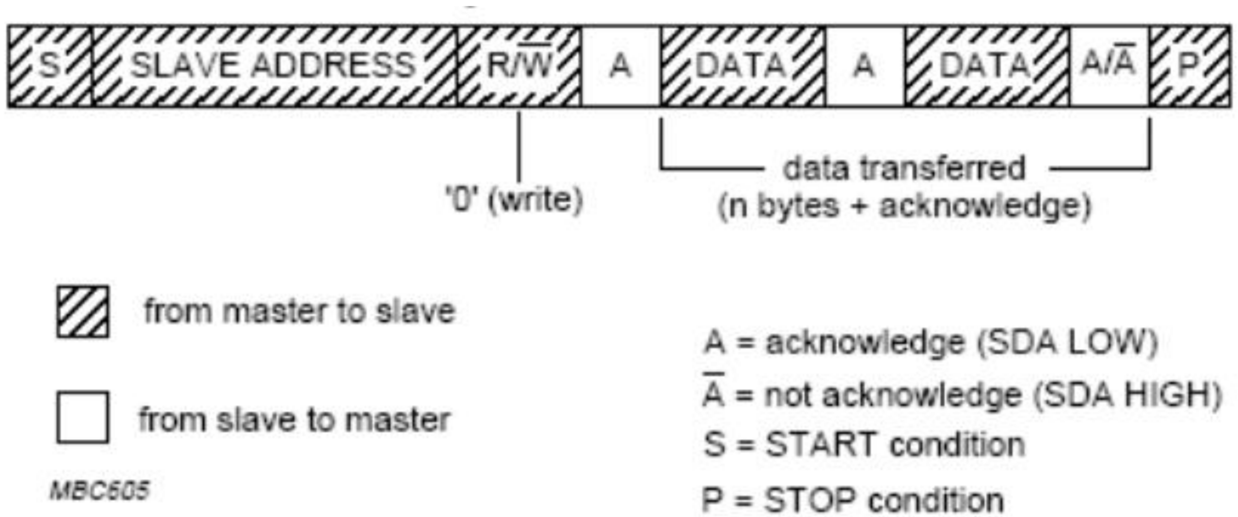
- Clock Synchronization is required when multiple masters are active on the bus
- Each master tries to drive its on clock on SCL line, at that time synchronization is required among the clocks to have a uniform clock signal on SCL
- Clock Synchronization is also based on Wired-AND logic

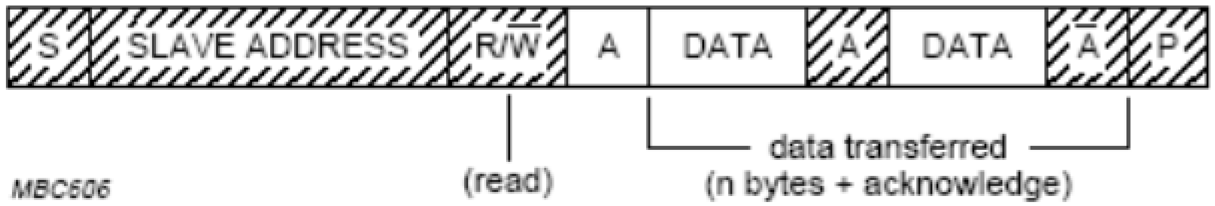
I2C Data transfer sequence

- 8 bits transmission at a time
- After byte of data transmission is followed by ACK bit
- First ACK always comes from the addressed slave device
- MSB first data transmission
- Bus lines are pulled high by hardware, so lines are high when free

I2C Data transmission format

master
To
slave





slave
to
master

Ref : https://www.nxp.com/docs/en/user-guide/UM10204.pdf