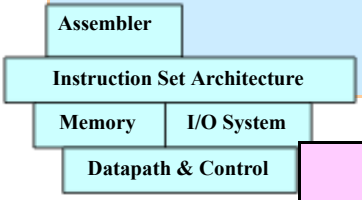
ELEC 3300
Introduction to Embedded Systems

Topic 9

Serial Communication

Prof. Vinod Prasad

Course Overview



Introduction to Embedded Systems

More about Embedded Systems

Basic Computer Structure

MCU Main Board

Digital and Analog Interfacing

USART, IEEE1394, USB, I2C and SPI

Microcontroller Structure

A/D Port

Serial Port

CPU

External Memory Port

External Interrupt Port

Interrupt Organization

Timer and Counter

External Timer Port

Simple I/O Port

Buffering and Direct Memory Access (DMA)

Memory, Interfacing to Memory, Memory Timing and applications

Interfacing LCD

Motor Interfacing

In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered | In progress | Done

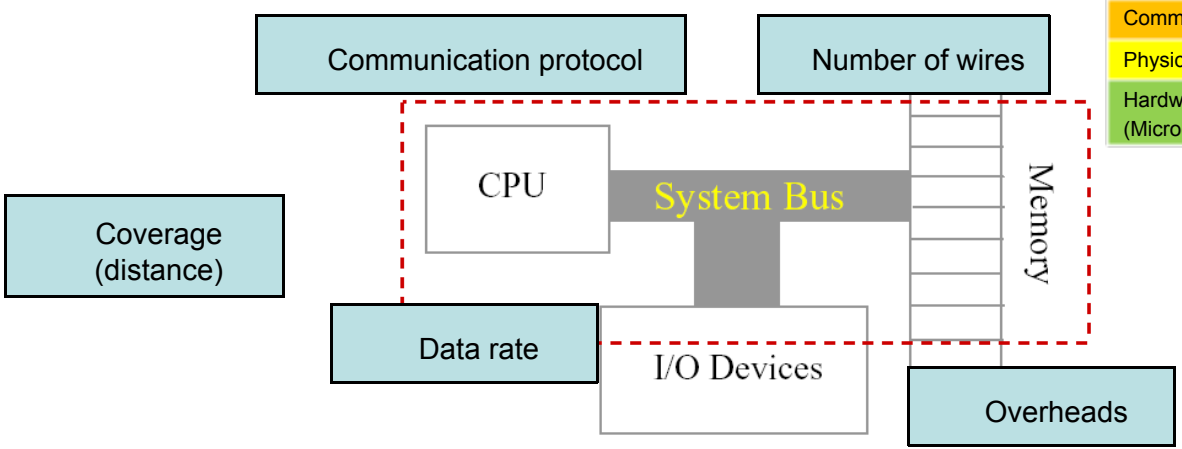
Expected Outcomes

- On successful completion of this topic, you will be able to
  - Summarize the main points of parallel and serial communications
  - Build both hardware and software configurations of the serial communication
  - Describe different serial communication protocols.

Data Communication

- CPU-device / CPU-memory interface usually consists of
  - Unidirectional address bus
  - Bidirectional data bus
  - Read/write/Ready/Acknowledge control lines

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)



Data Communication

Communication protocol: Rules that govern data transmission and reception.

The protocol defines the rules, syntax, semantics and synchronization of communication and also possible error recovery methods.

Syntax defines how data is structured.

Semantics is the meaning of the information/data

Data rate: Communication speed (bits per second)
Example: GSM - 9.6 kbits/sec, WCDMA - 2 Mbps, 4G – 100 Mbps, 5G – 10 Gbps

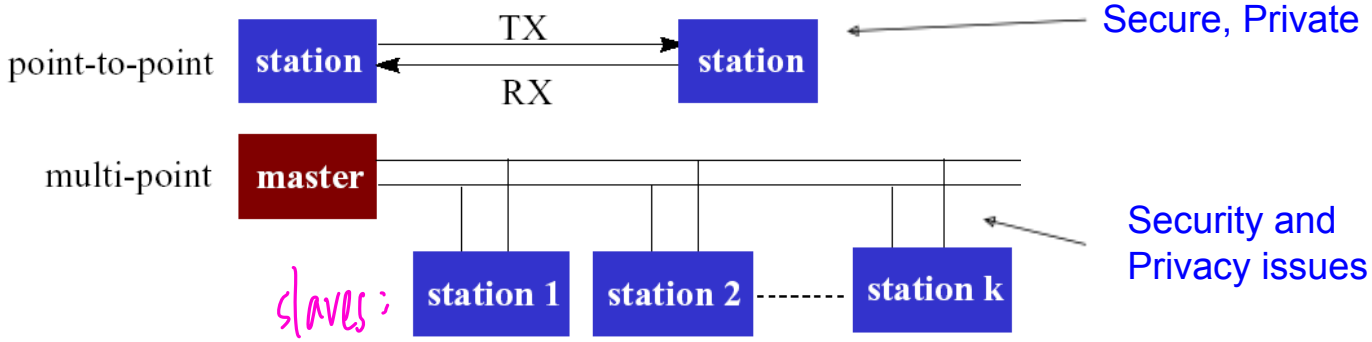
Coverage: Bluetooth – 10 mtrs, Zigbee: 10-100 mtrs, WiFi: 50-100 mtrs (can be even less in the presence of obstacles or strong EMI)

Overhead: start bit, stop bit, parity bit (odd/even), packet size, error correction codes, etc.

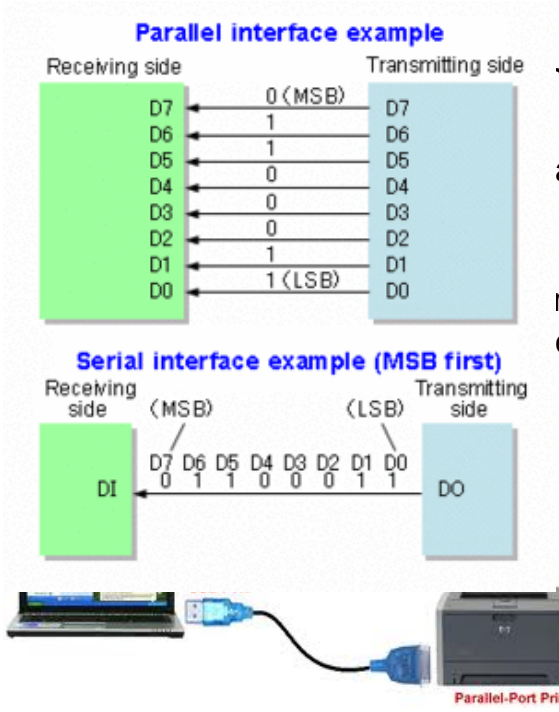
Basic information: Types of Communication Links

- Two common communication topologies (hardware configurations)
  - **Point-to-point**
    - Two end stations communicate as peers through a dedicated link.
    - Example: Telephone
  - **Multi-point**
    - One device is designated as master (primary) and another as slave (secondary) – communicate through shared link.
    - Example: video conference, distance learning (distributed networks).

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)



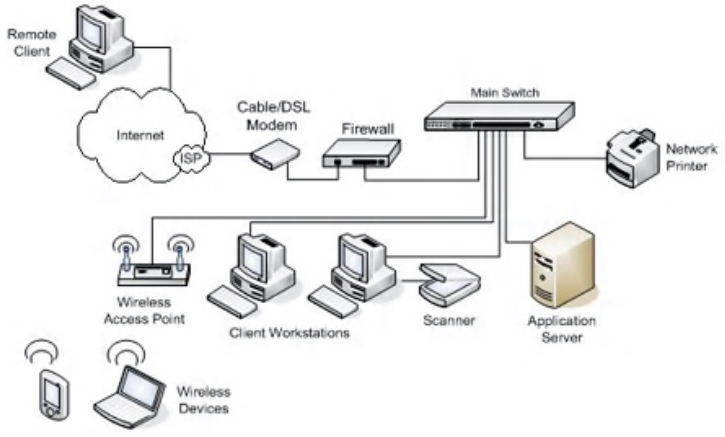
Basic information: Data Communication



Parallel communication

- ı protocols:
- a time over a single wire
- r multiple wires
- e (<20ft) because of parallel wires

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)



Serial communication: Computer Networks

Basic information: Data Communication

- Communications between computer and monitor over serial line
  - data is converted from parallel (bytes) to serial (bits) in the bus interface
  - Bits are sent over wire (TX) to terminal (or back from terminal to computer)
  - Receiving end (RX) translates bit stream back into parallel data (bytes)

Description
Abstract idea of project (Define the functionality of the system)
Data format / representation
Programming Language
Communication Protocol
Physical connection (Pins assignment)
Hardware devices (Microcontroller, Peripherals)

parallel

serial

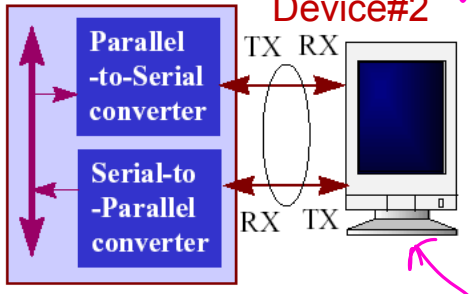
serial

parallel

Device#1 Serial Link Device#2

TX of Device#1 connected to RX of Device#2

RX of Device#1 connected to TX of Device#2



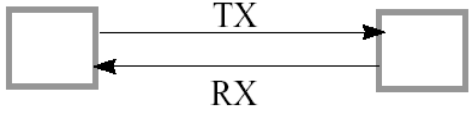
serial-parallel converter

Parallel comm: Motherboard, LCD, etc. Data needs to be converted to serial.

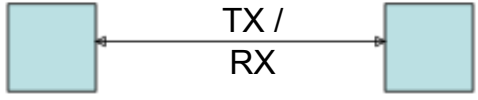
Serial communication is slower than parallel

Basic information: Types of Serial Communication

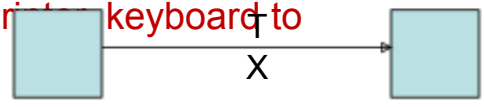
- There are **3 different serial communication protocols.**
  - **Full-duplex communication**
    - One signal wire is for transmitting, the other for receiving
    - Transmit and Receive simultaneously.
    - Example: Telephone



  - **Half-duplex communication**
    - One signal wire for both transmitting and receiving
    - Either transmit or receive at a time.
    - Example: Walkie Talkie



  - **Simplex communication**
    - One signal wire for transmitting only
    - One-way communication
    - Examples: Radio/TV broadcasting, computer to printer, keyboard to computer.



Basic information: Data Transfer Modes

There are **3 different data transfer modes**

- **Synchronous data transfer**
  - Clock is used by both receiver and sender to sample data
- **Asynchronous data transfer**
  - No clock signal in common for data transmissions.
  - The sender starts with extra bits called "start bits" or a "preamble" before sending the data. This allows the receiver to "synchronize with the signal."
- **Isochronous data transfer:** Iso (same) chronous (time)
  - Ensures the data flows at a pre-set rate so that an application can handle it in a timed way.
  - Transmission at regular intervals with a fixed gap between the transmission of successive data items.
  - Necessary for multimedia communication (to ensure that data is delivered as fast as it is displayed, and to ensure that the audio is synchronized with the video).

Basic information: Data Transfer Modes

BIT RATE: Number of bits transmitted per second (bits/sec).

BAUD RATE: Number of symbols per second, i.e., number of times a signal state is changed. Also called Symbol Rate (symbols/sec)

A **Symbol** typically consists of a fixed number of bits based on what the symbol is defined as (example: voltage, frequency, phase).

When will BAUD RATE and BIT RATE are same? When symbol has only one bit.

If N = Number of bits per symbol, then how is Baud rate related to Bit rate?

$$Baud\ rate = \frac{Bit\ rate}{N}$$

Total number of symbols = 2N

$$\text{Symbols/sec} = \frac{\text{bit}}{\text{sec}} \div \frac{\text{bits}}{\text{symbol}}$$

$$= \frac{\text{bit}}{\text{sec}} \times \frac{\text{symbol}}{\text{bit}}$$

Several types of Serial Communication Standards

- In the following, we will discuss
  - Universal synchronous asynchronous receiver transmitter (USART)
  - IEEE 1394
  - Universal Serial Bus (USB)
  - Inter Integrated Circuit (I2C) Bus
  - Serial Peripheral Interface (SPI) Bus

Device driver ← You apply this in the laboratory experiment.

Communication protocol and Hardware ← We will discuss in detail here.

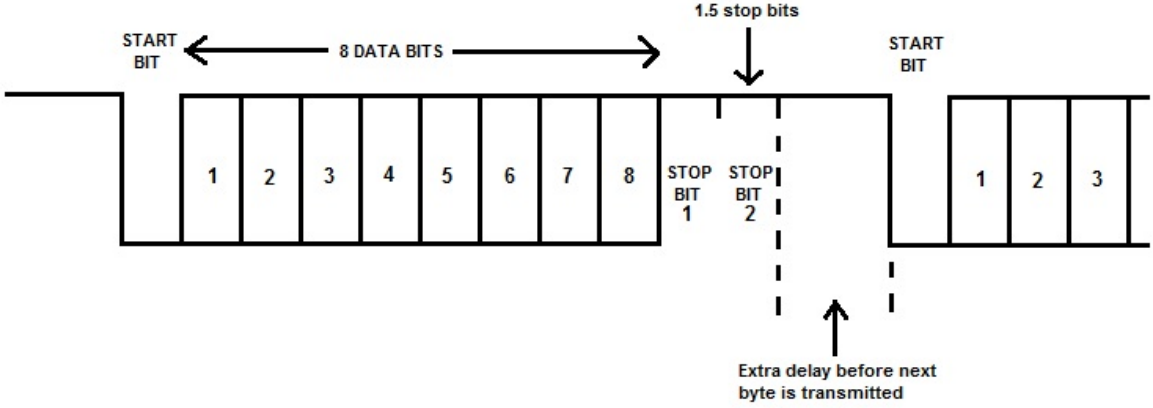
Universal **S**ynchronous/**A**synchronous **R**eceiver/**T**ransmitter (USART)

- Features of USART standard:
  - Supports either Point-to-point or multipoint,
  - Support Half-duplex or Full-duplex communications
  - Support synchronous or asynchronous modes

When to set these features?

During Initialization

1.5 stop bit implies 1.5 times the duration of a bit.



1.5 stop bits: The stop bit is transferred for 150% of the normal time used to transfer one bit.

Universal synchronous asynchronous receiver transmitter (USART)

- Word length and stop bits

Efficiency = number of information bits / number of bits in a packet

Figure 278. Word length programming

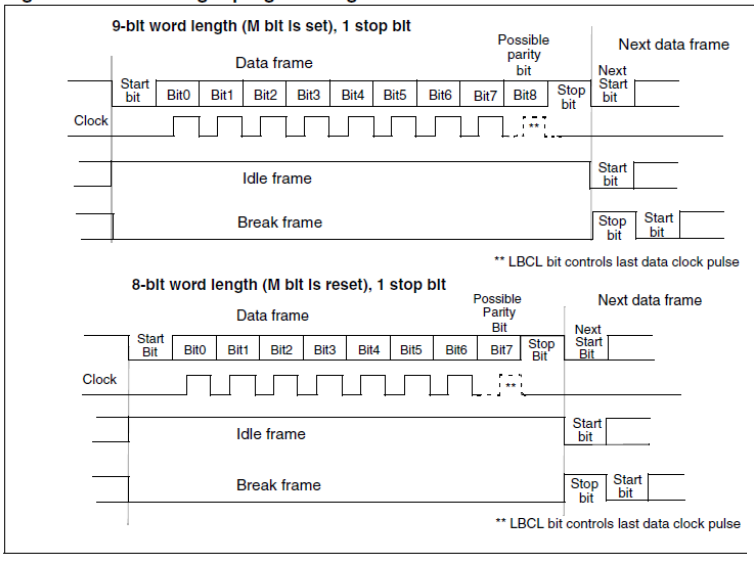
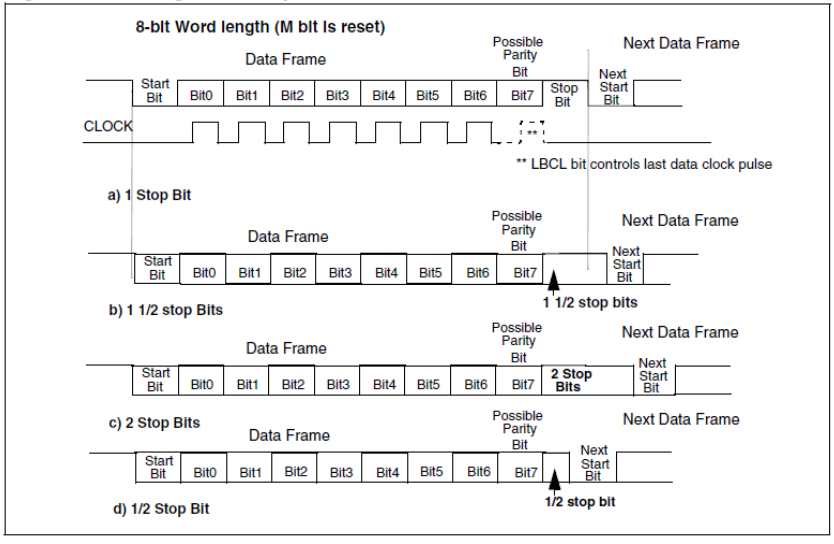


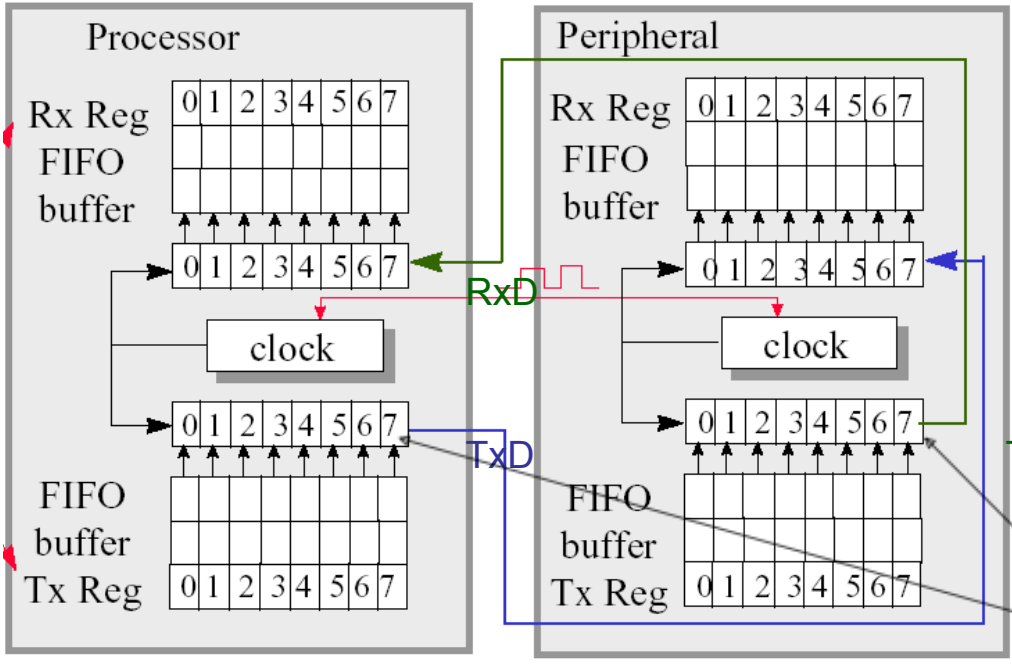
Figure 279. Configurable stop bits

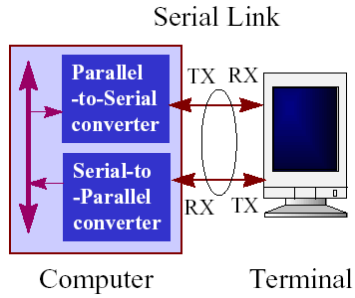


Determine the Efficiency in each case.

Universal synchronous asynchronous receiver transmitter (USART)

Conceptual architectures for serial-to-parallel / parallel-to-serial converter





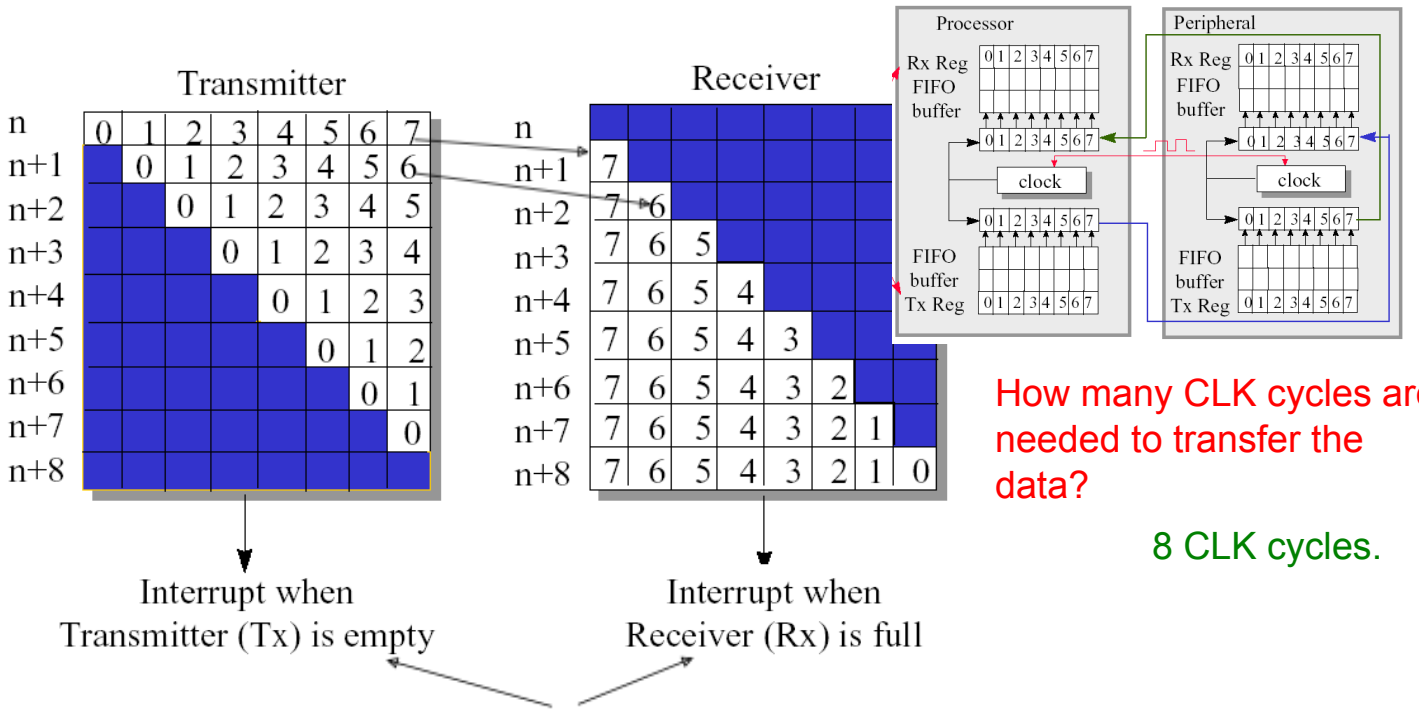
Which digital circuit element can be used to accomplish the bit-by-bit data transfer task?

Shift register

Clock edge triggered data transmission – one bit per clock pulse

Universal synchronous asynchronous receiver transmitter (USART)

The concepts: How is the information being transmitted in the serial communication ?



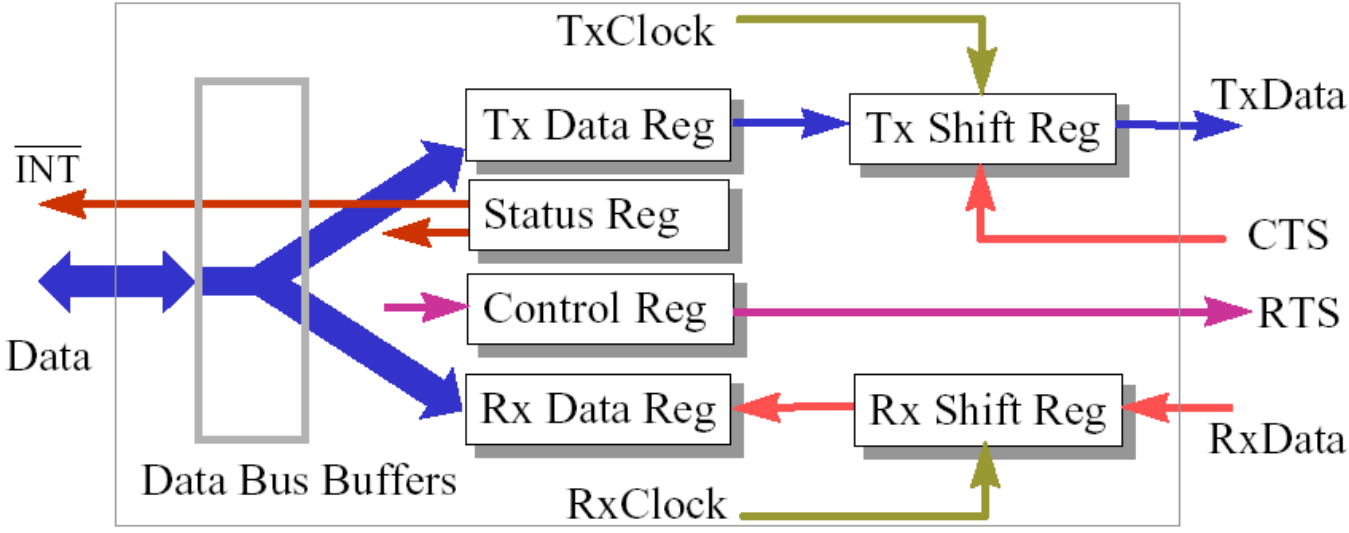
How many CLK cycles are needed to transfer the data?

8 CLK cycles.

Important signals that coordinate the completion of transmission of one byte and the start of transmission of next byte.

Interfacing Serial Data to the Microprocessor

- Processor has parallel buses for data (need to convert serial data to parallel and vice versa).



Control Register: Initiates / ends data communication.

Status Register: Status of Tx (Empty) and Rx (Full).

CTS: Clear To Send

RTS: Request To Send

Data Rate of USART in STM32

$$\text{Tx/Rx Baud} = \frac{f_{CK}}{16 \times \text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

Example:

$$2400 = \frac{36 \times 10^6}{16 \times \text{USARTDIV}}$$

$$\text{USARTDIV} = 937.5$$

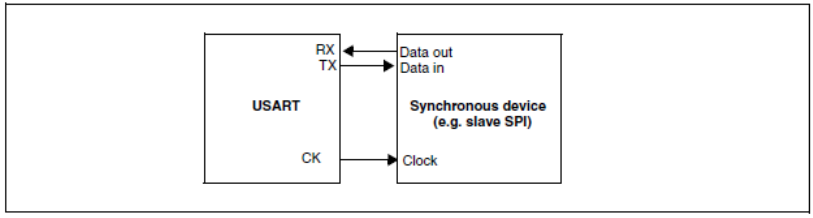
$$\text{Mantissa(USARTDIV)} = 937 = 0x3A9$$

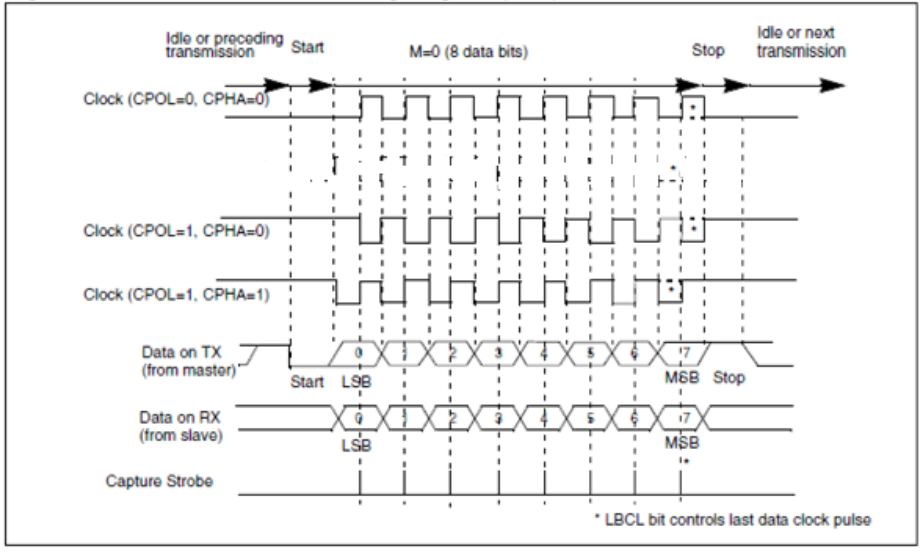
$$\text{Fraction(USARTDIV)} = 16 \times 0.5 = 8 = 0x8$$

$$\text{USARTDIV} = 0x3A98$$

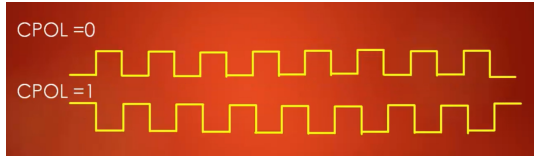
Universal synchronous asynchronous receiver transmitter (USART)

Figure 287. USART example of synchronous transmission





CPOL: Clock Polarity
(idle state level)
CPHA: Clock Phase



CPHA=1 □ data sampled on the trailing edge of the clock.

CPHA=0 □ data sampled on the leading edge of the clock.

Idle state: When no communication happens

: STM32_Reference_Manual.pdf

Examples of USART

- Task:
- Serial Communication with USART

Initialization

```
Void Main{
  Initialization of the USART port
  USART_InitStructure.USART_BaudRate = 230400;
   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
   USART_InitStructure.USART_Parity = USART_Parity_Even;
   USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
   USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART1, &USART_InitStructure);   /* Configure USART1 */
   USART_Cmd(USART1, ENABLE); /* Enable the USART1 */
  USART_Init(USARTz, &USART_InitStructure); /* Configure USART2 */
while(TxCounter < TxBufferSize)
{
    USART_SendData(USARTy, TxBuffer[TxCounter++]); /* Send 1 byte USARTy to USARTz */
   while(USART_GetFlagStatus(USARTy, USART_FLAG_TXE) == RESET) { } ; /* USARTy DR register is empty */
  while(USART_GetFlagStatus(USARTz, USART_FLAG_RXNE) == RESET) { }; /* USARRx DR register in not emptly */
  RxBuffer[RxCounter++] = (USART_ReceiveData(USARTz) & 0x7F); /* Store the received byte in RxBuffer */
 }
}
```

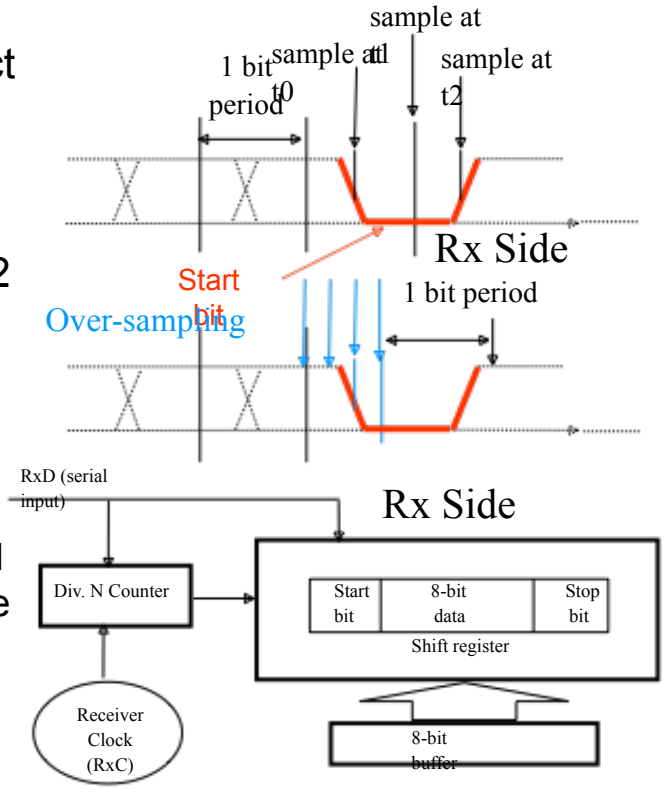
External Switch to control.

Full duplex.

implementation

Receiver of USART: Detection of start bit

- As the transmitter and receiver are not synchronized, how does the receiver detect the start bit?
- For instance, the start bit may not be detected if we take sample at either t0 or t2 .
- The solution to this problem is *over-sampling*
  - Receiver takes samples over the line signal at a frequency higher than transmission rate
  - When a valid start bit is detected, the receiver employs a normal sampling rate
  - Receiver Clock Ratio *N* = Receiver Clock Frequency / Transmission Rate



Asynchronous Receiver Circuitry

Multimedia Bandwidth Requirements

- Consumers Share Video, Audio, Images, and Data
- Faster and easier ways of sharing data is the ultimate goal

Serial comm: speed is 4.5Mbps only.

- High Quality Video (24-bit VGA at 30 frames/ second)
  - Data rate = (30 frames / second) (640 x 480 Pixels) (24-bit color / Pixel)
    ~= 221 Mbps □ factor = 221Mbps / 4.5Mbps ~= 49
- Reduced Quality Video
  - Digital Data = (15 frames / second) (320 x 240 Pixels) (16-bit color / Pixel)

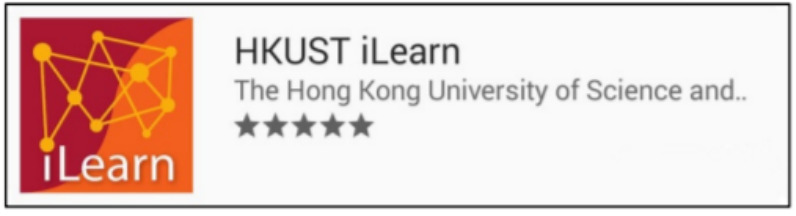
    ~= 18 Mbps □ factor = 18Mbps / 4.5Mbps ~= 4
- High Quality Audio
  - Digital Data = (44,100 audio samples / sec) (16-bit audio samples) (2 audio channels for stereo) = 1.4 Mbps
- Reduced Quality Audio
  - Digital Data = (11,050 audio samples / sec) (8-bit audio samples) (1 audio channel for monaural) = 0.1 Mbps

Too much delay due to large amount of data – Solution?

Data Compression (Lossy/Lossless)

In-class activities

For Android devices, search **HKUST iLearn** at Play Store.

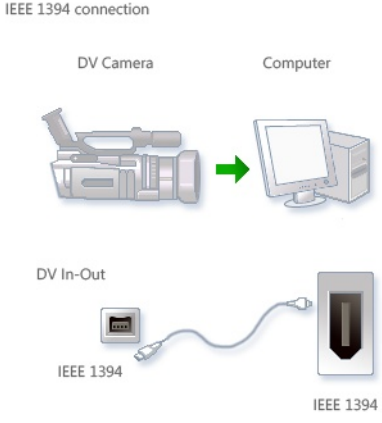


For iOS devices, search **HKUST iLearn** at App Store.

Questions 1 - 3

Introduction to IEEE1394
(developed by Apple, Sony and Panasonic)

- High-speed serial communication bus - Driving the convergence of computers, consumer equipment, and communications
- Convergence will happen when seamless, high-speed communication becomes readily available
  - The IEEE 1394 protocol appears to be a strong contender for the communications channel that will make this happen.



High Performance Serial Bus – IEEE1394 : Two transmission methods

- Asynchronous transport
  - Features:
    - Guaranteed delivery
    - Reliability is more important than timing
    - No interrupt, check CRC then send ACK
    - Allow retry or re-send (if no ACK)
  - Packet size : 1 to 2048 byte(s)

Note: CRC (Cyclic Redundant Check) is an error control coding algorithm.
Provide error detection and correction on data packets.

- Isochronous transport : Iso (same) chronous (time)
  - Features:
    - Guaranteed bandwidth and latency (Necessary for multimedia applications)
    - Late data will be ignored
    - No retry or re-send
    - Check CRC, but **No ACK**
  - Packet size : 1 to 4096 byte(s)

- Isochronous Talker(s) have priority over Asynchronous Talker(s)

IEEE 1394 & Universal Serial Bus (USB)

- USB and 1394 are complementary buses, differing in their application focus

  – USB 2.0/3.0 is the preferred connection for most PC peripherals

  – 1394's primary target is audio/visual consumer electronic devices such as digital camcorders, digital VCRs, DVD players, and digital televisions

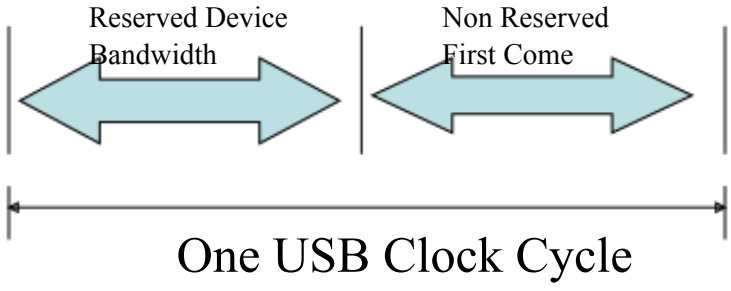
Universal Serial Bus: USB

- The USB was designed by the computer industry to replace the USART serial port technology.
- The USB was designed with the following goals:
  - allow many peripherals to be connected using a single standardized interface socket
  - Data-transfer rates at different versions
    - A **low speed** (USB 1.0) rate of 1.5 Mbit/s, Half duplex
    - The **full speed** (USB 1.1) rate of 12 Mbit/s, Half duplex
    - A **high-speed** (USB 2.0) rate of 480 Mbit/s, Half duplex
    - A **SuperSpeed** (USB 3.0) rate of 5.0 Gbit/s, Full duplex
  - Economical implementation
  - True “plug-n-play” device support
  - Bus-powered devices.

Universal Serial Bus: Isochronous transmission

- Isochrony: A device is guaranteed bandwidth on the bus.
  - If the device, such as a video camera, required a certain amount of "room" on the bus, then it asks the system to reserve an amount of isochronous bandwidth.
  - Case I: Reserved Device Bandwidth
    - If reserved bandwidth is available, then the host guarantees that it is ***always*** available for that device.
  - Case II: Non Reserved, First Come

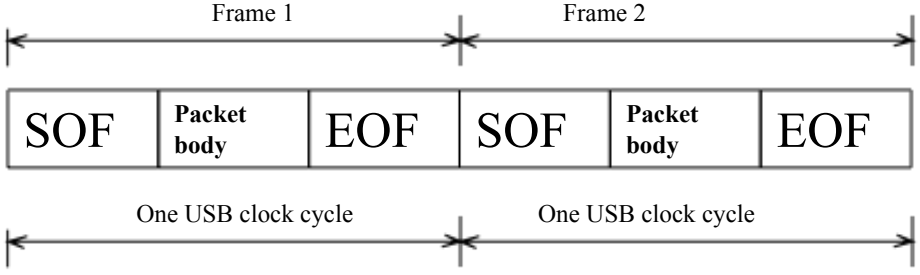
    If the reserved bandwidth is not available, then it is up to the device (or, rather, the software controlling the device) whether to accept the non-guaranteed bandwidth.



Universal Serial Bus: Protocol

- You cannot implement the USB driver without understanding its protocol.
- Communication between devices and hubs on the USB occur as a **serial bit stream.** A serial interface engine (SIE) is a piece of hardware that turns internal data flow into a serial bit stream.

- Frame generation:

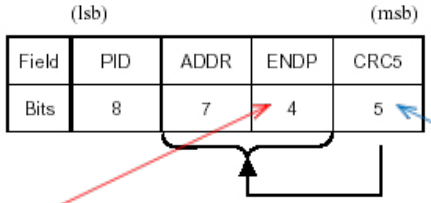


- SOF = Start Of Frame, EOF = End Of Frame

Universal Serial Bus: Types of Packet Body

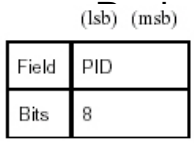
PID: Packet identifier Field
ENDP: End-Point Number

Token Packet



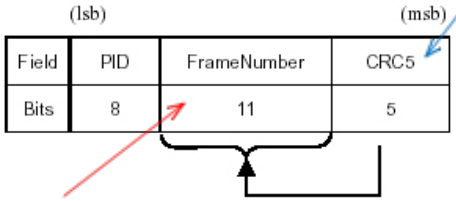
ENDP: Select endpoint hardware source/sink buffer on device. (E.g. PID OUT would be for sending data from host source buffer into the USB device sink buffer.

Handshake



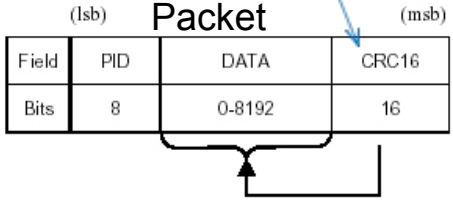
Cyclic Redundancy Check (CRC) Field
Provide error detection and correction on the token and data packets.

Start-Of-Frame



Range: 000 to 7ff (hex)

Data Packet

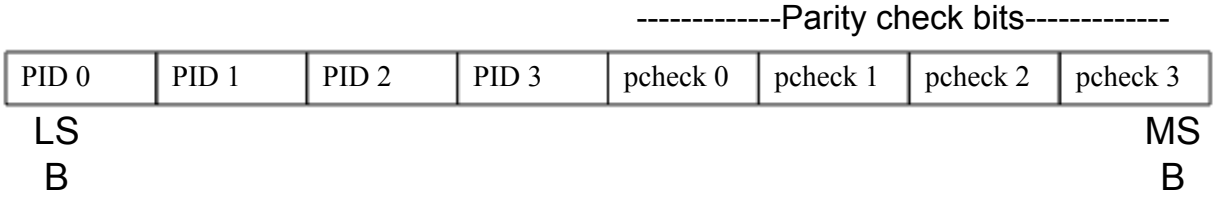


Handshake packets report the status of a data transaction and can return values indicating the reception or rejection of data, halt conditions, or flow control.

Token packet starts all communications on the USB. It is issued by the host and contains the packet identifier (PID), device address (ADDR), and endpoint number (ENDP) to identify a consignee.

Universal Serial Bus: Important Packet Fields in Protocol

- **Packet Identifier Field (PID)**
  - Tells the device what to do with the packet.
  - Format of PID : packet-type field (4 bits) and a check field (4 bits)
    - The check field is a one's complement of the PID field.
    - If a packet is received and the check field isn't correct, rest of the packet is ignored by the device.



Universal Serial Bus: Important Packet Fields in Protocol

- Packet Identifier Field (PID) (Cont'd)

PID Type	PID Name	PID<3:0>*	Description
Token	OUT	0001B	Address + endpoint number in host-to-function transaction
	IN	1001B	Address + endpoint number in function-to-host transaction
	SOF	0101B	Start-of-Frame marker and frame number
	SETUP	1101B	Address + endpoint number in host-to-function transaction for SETUP to a control pipe
Data	DATA0	0011B	Data packet PID even
	DATA1	1011B	Data packet PID odd
	DATA2	0111B	Data packet PID high-speed, high bandwidth isochronous transaction in a microframe (see Section 5.9.2 for more information)
	MDATA	1111B	Data packet PID high-speed for split and high bandwidth isochronous transactions (see Sections 5.9.2, 11.20, and 11.21 for more information)
Handshake	ACK	0010B	Receiver accepts error-free data packet
	NAK	1010B	Receiving device cannot accept data or transmitting device cannot send data
	STALL	1110B	Endpoint is halted or a control pipe request is not supported
	NYET	0110B	No response yet from receiver (see Sections 8.5.1 and 11.17-11.21)
Special	PRE	1100B	(Token) Host-issued preamble. Enables downstream bus traffic to low-speed devices.
	ERR	1100B	(Handshake) Split Transaction Error Handshake (reuses PRE value)
	SPLIT	1000B	(Token) High-speed Split Transaction Token (see Section 8.4.2)
	PING	0100B	(Token) High-speed flow control probe for a bulk/control endpoint (see Section 8.5.1)
	Reserved	0000B	Reserved PID

*Note: PID bits are shown in MSb order. When sent on the USB, the rightmost bit (bit 0) will be sent first.

Universal Serial Bus: Stages of transactions

- Error Detection and Recovery
  - USB provides for three distinct error-detection types:
    - Bit-stuffing: ‘0’ inserted at the end of packets in some cases (all bits are 1s) to detect transitions (USB is asynchronous and hence uses transitions to sync).
    - PID field check bits
    - CRC

Field	**Error**	**Action**
PID	PID check, bit stuff	Ignore packet
Address	Bit-stuff, address CRC	Ignore token
Frame Number	Bit-stuff, frame-number CRC	Ignore frame-number field
Data	Bit-stuff, data CRC	Discard data

I2C: Inter Integrated Circuit Bus

- Technical Specifications

– Support for communication with various slow, on-board peripheral devices that are accessed intermittently.

– It is a simple, low-bandwidth, short-distance protocol.

– Operates at speeds up to 400Kbps, with some venturing up into the low megahertz range.

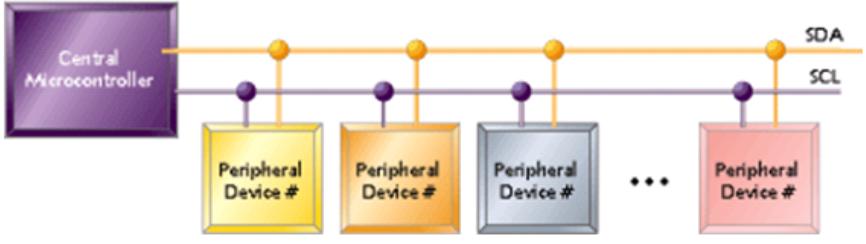
– Use to link multiple devices together since it has a built-in addressing scheme.

– Full duplex.

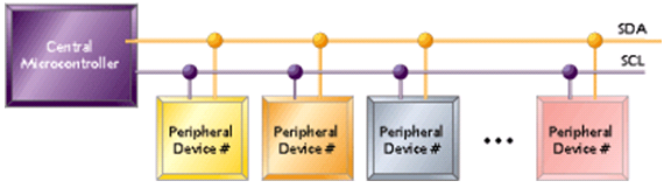
I2C: Inter Integrated Circuit Bus: Hardware configuration

- I2C is a two-wire serial bus including
  - Serial Data (SDA) signal line
  - Serial Clock (SCL) signal line
    - Use to synchronize all data transfers over the I2C bus.
  - Both SCL and SDA lines are "open drain" drivers.

i.e., The chip can drive its output low, but it cannot drive it high. For the line to be able to go high you must provide pull-up resistors to the 5v supply.



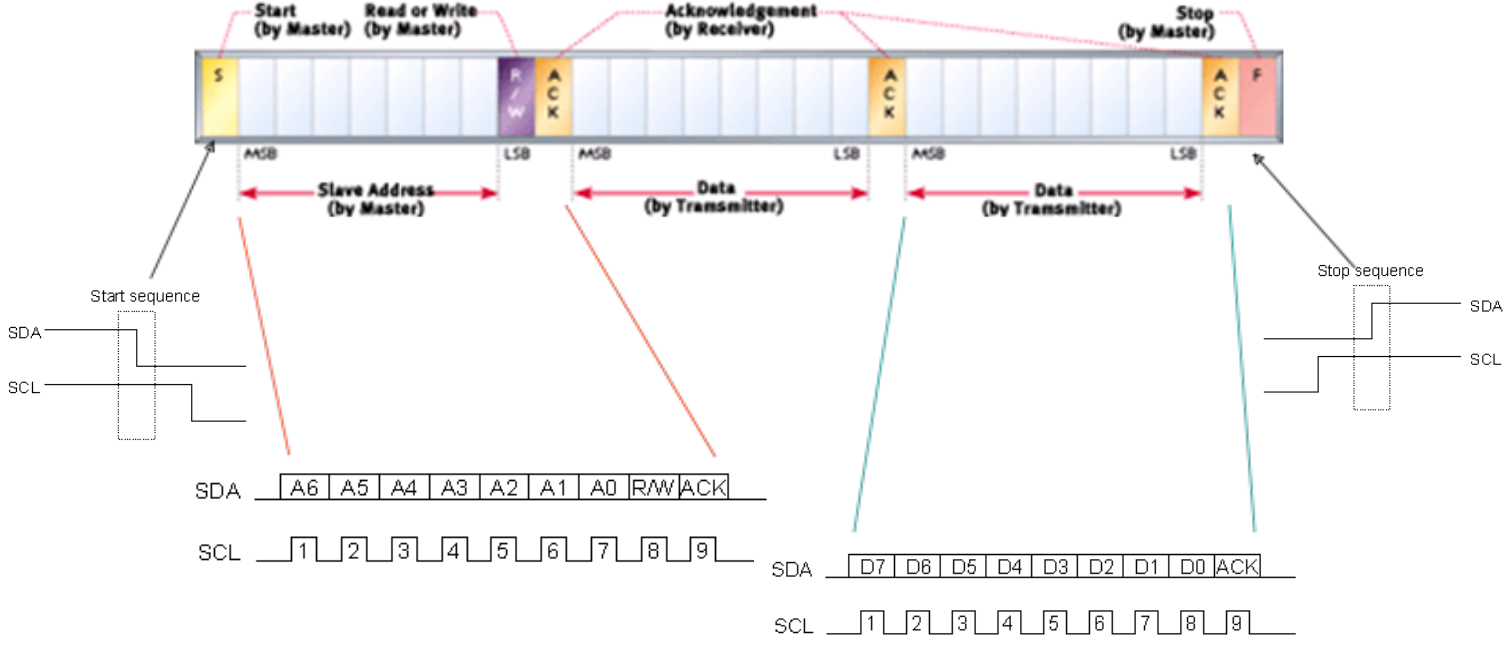
I2C: Inter Integrated Circuit Bus: Hardware configuration



- The architecture includes
  - Master device
    - Initiates a transaction on the I2C b
    - Controls the clock signal
    - Possible to have multiple masters, but most system designs have only one.
  - Slave devices
    - Addressed by the master device
  - Both master and slaves can receive and transmit data bytes.

I2C: Inter Integrated Circuit Bus: Protocol

- Support serial transmission of 8-bit bytes of data-7-bit device addresses plus control bits over the two-wire bus.



SPI : Serial Peripheral Interface Bus

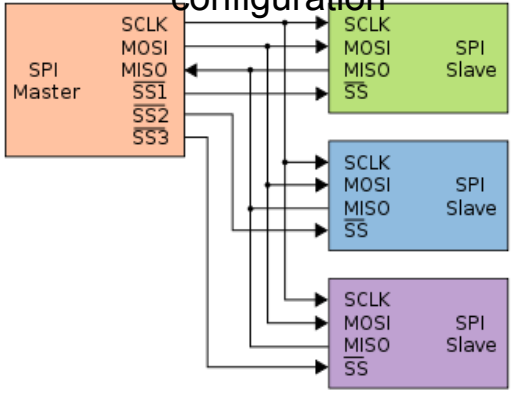
- Technical Specifications
  - a synchronous serial data link operates in full duplex mode.
  - Devices communicate in master/slave mode where the master device initiates the data frame.
  - Multiple slave devices are allowed with individual slave select (chip select) lines.
  - SPI can also achieve significantly higher data rates than I2C. SPI-compatible interfaces often range into the tens of megahertz
  - SPI really gains efficiency in applications that take advantage of its duplex capability.

SPI : Serial Peripheral Interface Bus: Hardware Configuration

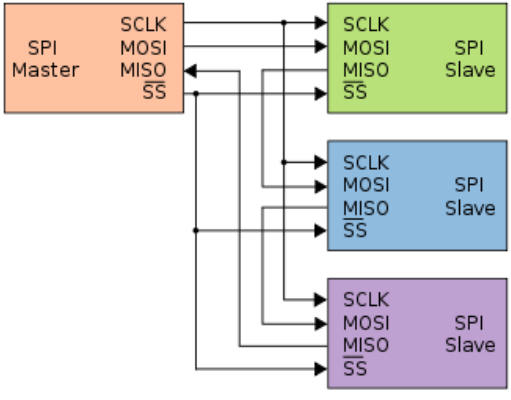
- The SPI bus specifies four logic signals.
  - SCLK : Serial Clock
  - MOSI/SIMO : Master Output, Slave Input
  - MISO/SOMI : Master Input, Slave Output
  - SS : Slave Select (active low)

Single slave SPI configuration

Independent slave SPI configuration

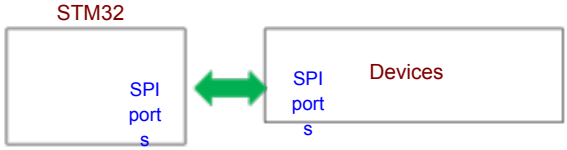


Daisy chain SPI configuration



Examples of SPI

- Task:
- Serial Communication with SPI



Initialization

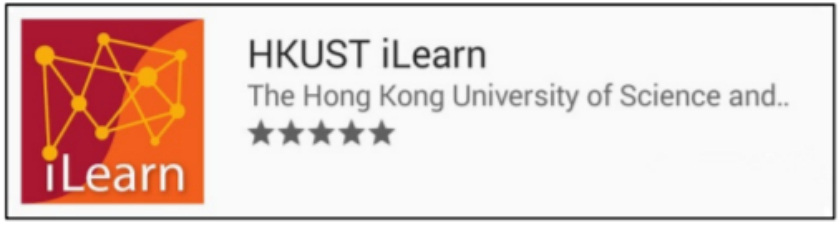
```
Void Main{
  Initialization of the SPI port
  SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
  SPI_InitStructure.SPI_Mode = SPI_Mode_Master; /* Master Mode */
  SPI_InitStructure.SPI_DataSize = SPI_DataSize_16b;
  SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
  SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;
  SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
  SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_8;
  SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
  SPI_InitStructure.SPI_CRCPolynomial = 10;   /*  check with main.c which is generated by STM32CubeMX */
  SPI_Init(SPI1, &SPI_InitStructure);   /* Configure SPI1 */
  SPI_InitStructure.SPI_Mode = SPI_Mode_Slave;  /* Slave Mode */
  SPI_Init(SPI2, &SPI_InitStructure); /* Configure SPI2 */
 SPI_CalculateCRC(SPI1, ENABLE); /* Enable the SPI1 CRC calculation */
 SPI_CalculateCRC(SPI2, ENABLE); /* Enable the SPI2 CRC calculation */
   SPI_Cmd(SPI1, ENABLE); /* Enable SPI1 */
 SPI_Cmd(SPI2, ENABLE); /* Enable SPI2 */

 …… /* steps for SPI data communication  - read-write cycles*/
}
```

implementation

Quiz Questions 4 and 5

For Android devices, search **HKUST iLearn** at Play Store.



For iOS devices, search **HKUST iLearn** at App Store.

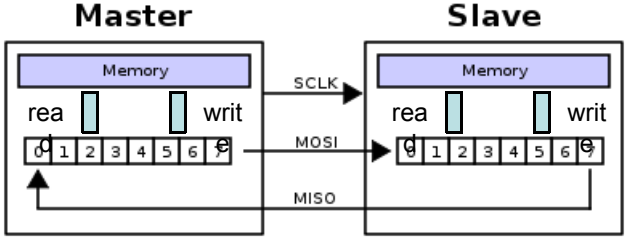
Reflection (Self-evaluation)

- Do you ….
  - State the advantages and disadvantages of parallel and serial communications ?
  - List some applications in using serial communication protocol?
  - List some applications in using parallel communication protocol?
  - Understand basic concepts of following standard
    - UART (universal asynchronous receiver and transmitter) standard? (RS232)
    - IEEE1394 : FireWire
    - USB: Universal Serial Bus
    - I2C: Inter Integrated Circuit Bus
    - SPI: Serial Peripheral Interface Bus
  - Describe the data rate of each serial bus standard ?
  - State the applications of each serial bus standard ?

Additional Notes (Supplementary)

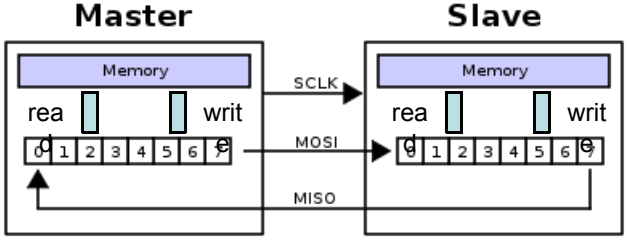
SPI : Serial Peripheral Interface Bus: Protocol

- For data communications:
  - Step 1: The shift registers are loaded with new data
  - Step 2: the master first configures the clock
  - Step 3: The master then pulls the slave select low for the desired chip.
  - Step 4: During each SPI clock cycle, a full duplex data transmission occurs:
    - the master sends a bit on the MOSI line; the slave reads it from that same line
    - the slave sends a bit on the MISO line; the master reads it from that same line



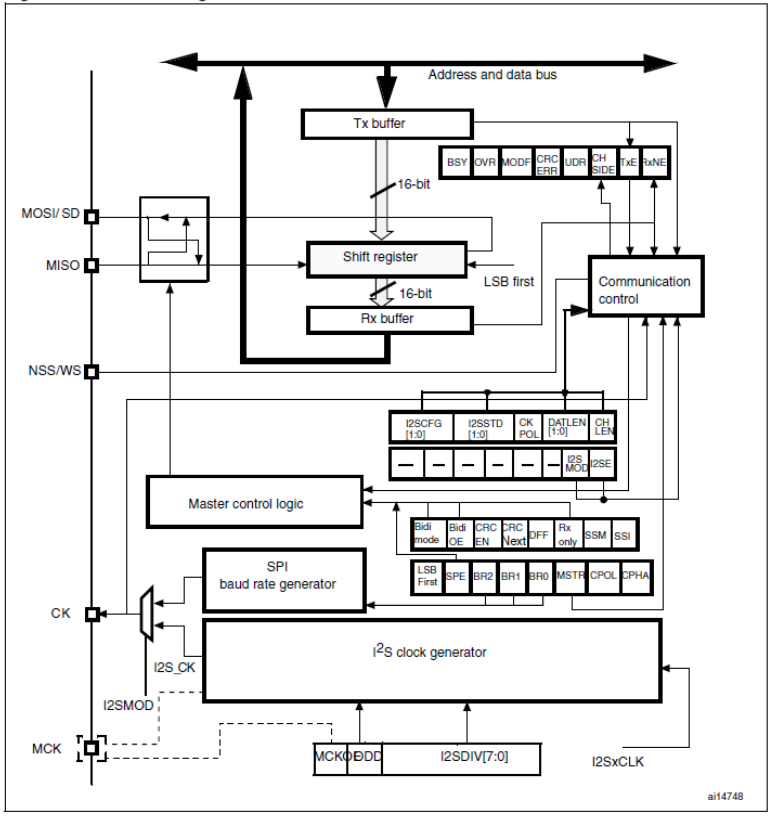
SPI : Serial Peripheral Interface Bus: Protocol

- For data communications (Cont'd):
  - Step 5: each device takes that value and does something with it, such as writing it to memory.
  - Step 6: If there is more data to exchange, the shift registers are loaded with new data and the process repeats. (Go to step 3)
    - Note: Transmissions may involve any number of clock cycles.
  - Step 7: When there is no more data to be transmitted, the master stops toggling its clock. Normally, it then deselects the slave.

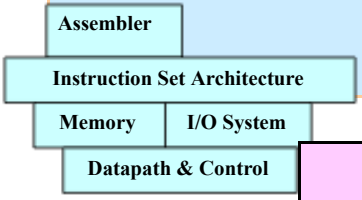


SPI : Serial Peripheral Interface Bus: in STM32

Figure 247. I²S block diagram



Course Overview



Introduction to Embedded Systems

More about Embedded Systems

Basic Computer Structure

MCU Main Board

Digital and Analog Interfacing

USART, IEEE1394, USB, I2C and SPI

Microcontroller Structure

A/D Port

Serial Port

CP U

External Memory Port

External Interrupt Port

Interrupt Organization

Timer and Counter

External Timer Port

Simple I/O Port

Buffering and Direct Memory Access (DMA)

Memory, Interfacing to Memory, Memory Timing and applications

Interfacing LCD

Motor Interfacing

In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered | In progress | Done