SCS3101 – Electronics

Lecture 10 – Embedded Communication 1

Hiran Ekanayake

University of Colombo School of Computing

Communication Interface

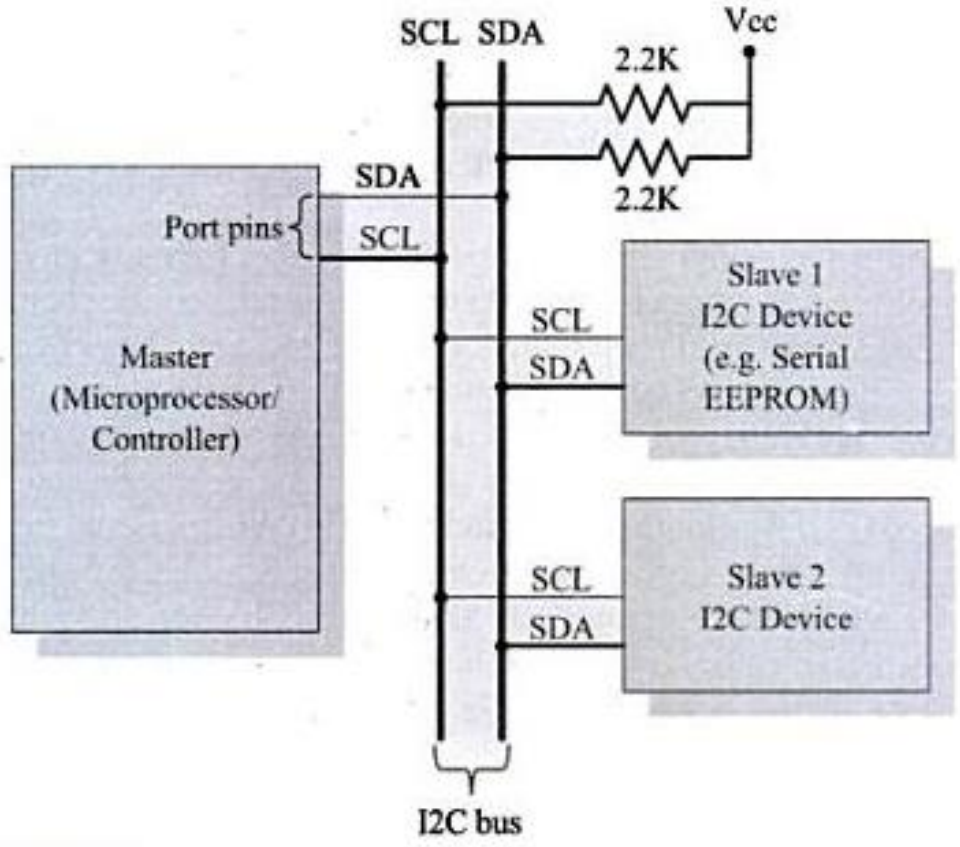
- Communication interface is essential for communicating with various subsystems of the embedded system and with the external world. It can be viewed from two perspectives:
  - Device/board level (onboard) communication interface, i.e., serial interfaces like I2C, SPI, UART, and 1-Wire and parallel bus interfaces
  - Product level (external) communication interface which can be either a wired media or wireless media and it can be a serial or a parallel interface. I.e., Wireless: Infrared (IR), Bluetooth (BT), Wireless LAN (Wi-Fi), Radio Frequency waves (RF), GRPS, etc.; Wired: RS-232C/RS-422/RS-485, USB, Ethernet IEEE 1394 port, Parallel port, CF-II interface, SDIO, PCMCIA, etc.

Inter-Integrated Circuit (I2C, I^2C, IIC) Communication (1)

- I2C is a synchronous bi-directional half duplex (one-directional communication at a given point of time) two wire serial interface bus
- It allows communication between the uC and low speed peripherals, e.g., external EEPROMs, digital sensors, and LCD drivers
- It consists of two bus lines: Serial Clock (SCL) and Serial Data (SDL)
  - SCL is responsible for generating synchronization clock pulses
  - SDL is responsible for transmitting the serial data across devices
  - Note: apart from SCL and SDL, it needs two additional lines for GND and Vcc
- Devices connected to I2C bus can act as either ‘Master’ device or ‘Slave’ device
  - Master: responsible for controlling the communication by initiating/terminating data transfer, sending data and generating necessary synchronization clock pulses
  - Slave: waits for the commands from master and responds upon receiving the commands

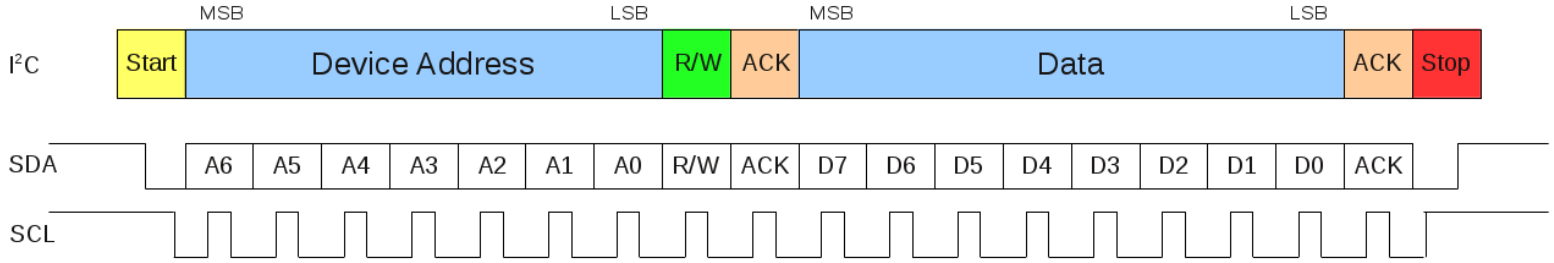
Inter-Integrated Circuit (I2C, I^{2}C, IIC) Communication (2)

- Both SDA and SCL lines are open drain drivers, i.e., the I2C driver can only pull the lines low, therefore, external pull-up resistors (e.g. 2.2K) are required
- I2C can communicate data at rates 100 Kbps (standard mode), 400 Kbps (fast mode), and 3.4 Mbps (high-speed mode)
- I2C used 7-bit (or 10-bit) addressing and each data is 8-bits long



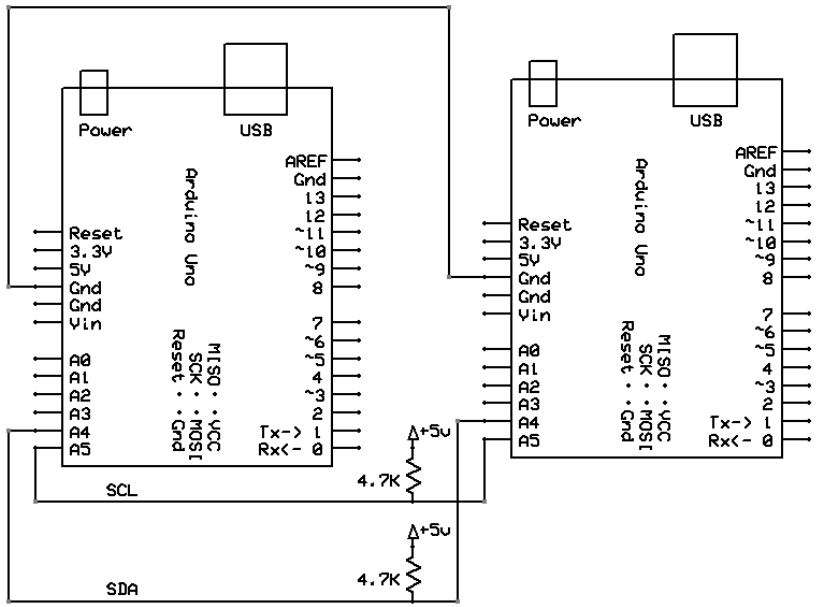
Inter-Integrated Circuit (I2C, I²C, IIC) Communication (3)

- The sequence of operations for communicating with an I2C slave device:
  - The master device pulls the SCL to ‘High’ and then the SDA to ‘LOW’ (indicating the START condition)
  - The master device sends the address of the slave device to which it wants to communicate over the SDA. Clock pulses are generated at the SCL line for synchronizing
  - The master device sends the Read (bit=1) or Write (bit=0) bit according to the requirement
  - The master device waits for the acknowledgment bit from the slave device (on SDA line)
  - Upon receiving the acknowledge bit, the master device sends the 8-bit data to the slave device over SDA line, if the requested operation is ‘Write to Device’. If the requested operation is ‘Read from Device’, the slave device sends data to the master
  - The master or slave sends an acknowledge bit depending on the Read/Write operation
  - The master device terminates the transfer by pulling the SDA line ‘High’ when the SCL line is ‘High’ (indicating STOP condition)



Arduino to Arduino Communication Using I2C

```
// i2c master
#include <Wire.h>

void setup() {
  Serial.begin(9600);
  Wire.begin();
}

void loop() {
  while (Serial.available()) {
    char c = Serial.read();

    // address of slave is 5
    // it can be 1..127
    if (c=='H') {
      Wire.beginTransmission(5);
      Wire.write('H');
      Wire.endTransmission();
    } else if (c=='L') {
      Wire.beginTransmission(5);
      Wire.write('L');
      Wire.endTransmission();
    }
  }
}
```



Arduino:

SDA = A4

SCL = A5

```
// i2c slave
#include <Wire.h>

void setup() {
  // address is 5
  Wire.begin(5);
  Wire.onReceive(react);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
}

void loop() {
}

void react(int n) {
  while (Wire.available()) {
    char c = Wire.read();

    if (c=='H') {
      digitalWrite(13,HIGH);
    } else if (c=='L') {
      digitalWrite(13,LOW);
    }
  }
}
```

Arduino to Arduino Communication Using I2C – 2 (Master Requesting Data From Slave)

```
// i2c master
#include <Wire.h>

void setup() {
  Serial.begin(9600);
  Wire.begin();
  // address, #bytes requesting
  Wire.requestFrom(5,5);
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);
  }
}

void loop() {
}
```

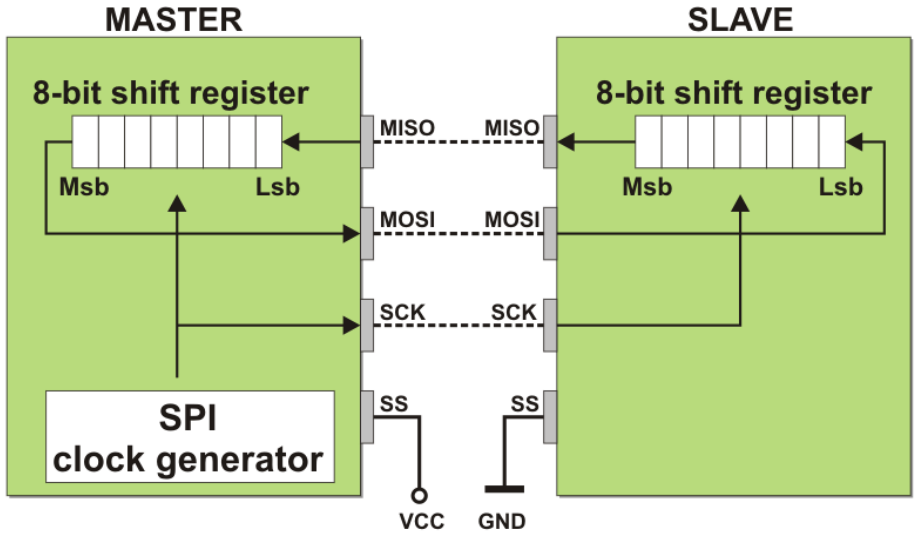
```
// i2c slave
#include <Wire.h>

void setup() {
  // address is 5
  Wire.begin(5);
  Wire.onRequest(react);
}

void loop() {
 delay(500);
}

void react() {
  Wire.write("1234567890");
}
```

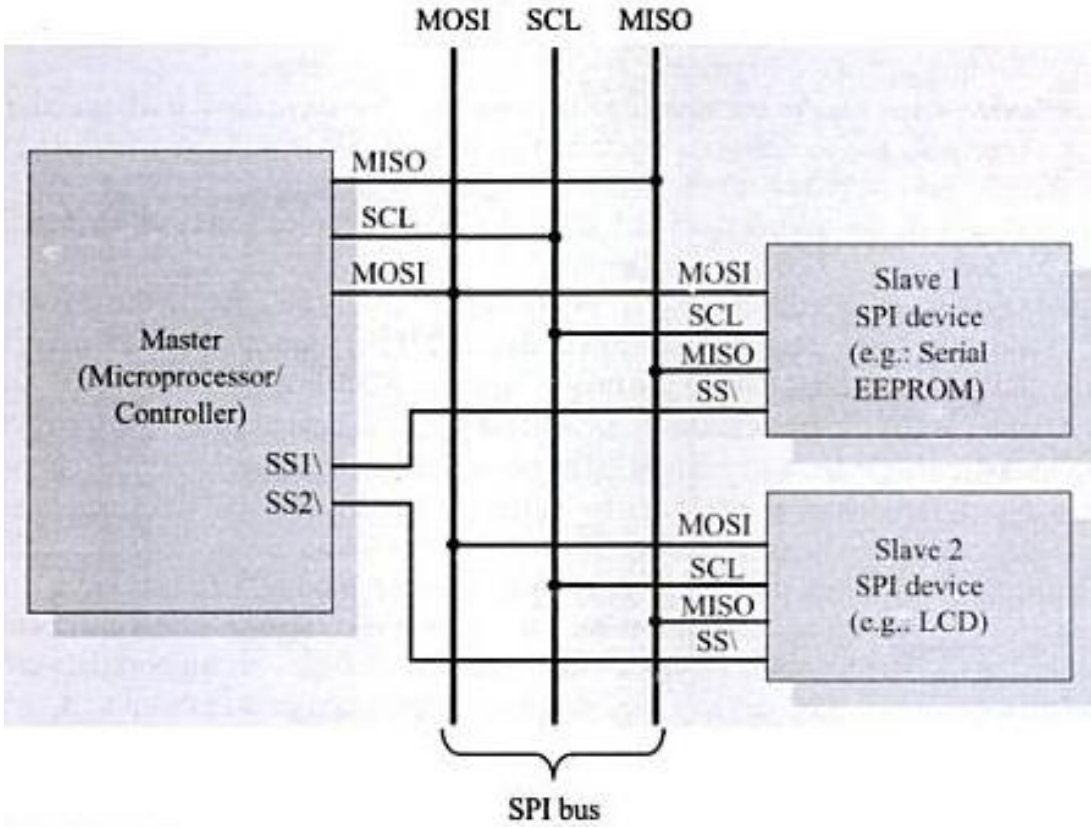
Serial Peripheral Interface (SPI) Bus (1)

- SPI bus is a synchronous bi-directional full-duplex four-wire serial interface bus (originally introduced by Motorola)
- SPI bus is most suitable for applications requiring transfer of data in ‘streams’
- SPI is a single master multi-slave system
  - However, it is possible to have a system where more than one SPI device can be master, provided the condition only one master device is active at any given point of time, is satisfied
- SPI requires four lines for communication:
  - Master Out Slave In (MOSI): Serial line carrying the data from master to slave device. It is also known as Slave Input/Slave Data In (SI/SDI)
  - Master In Slave Out (MISO): Serial line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)
  - Serial Clock (SCLK): Serial line carrying the clock signals
  - Slave Select (SS): Signal line for slave device select. It is an active low signal

Serial Peripheral Interface (SPI) Bus (2)

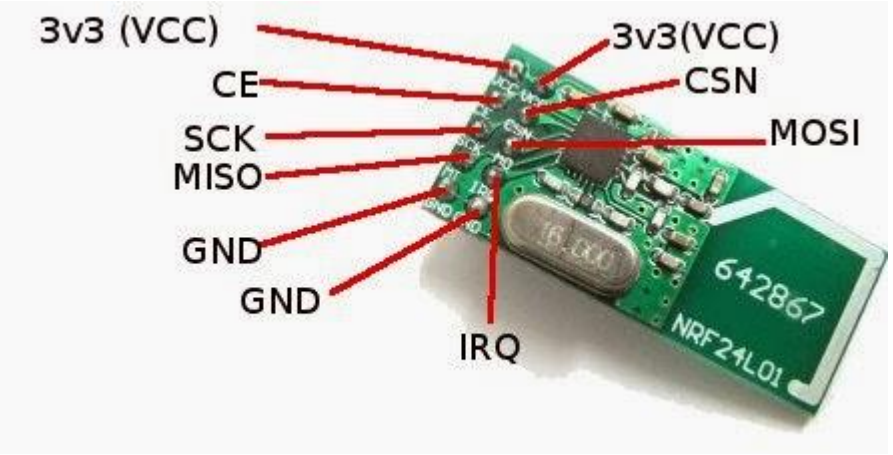
- SPI works on the principle of ‘Shift Register’:
  - During transmission from the master to slave, the data in the master’s shift register is shifted out to the MOSI pin and it enters the shift register of the slave through the MOSI pin of the slave. At the same time the shifted out data bit from the slave device’s shift register enters the shift register of the master device through MISO pin. I.e., master and slave forms a circular buffer

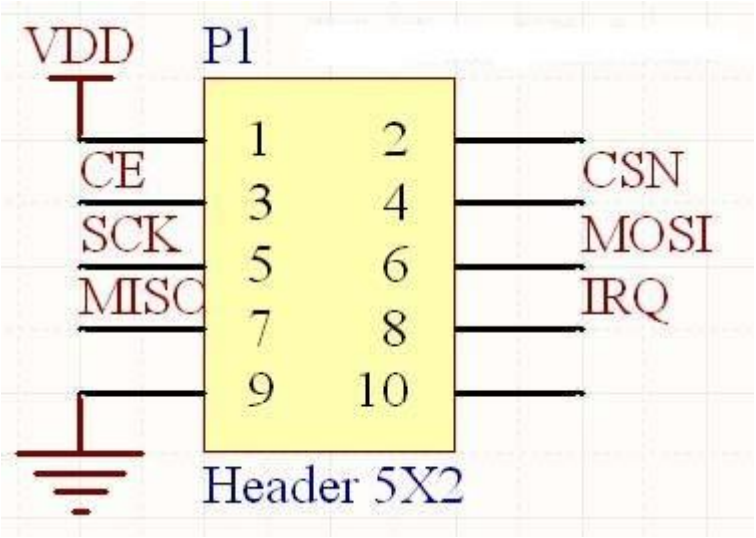




Interfacing the nRF24L01 RF Transceiver

- Inexpensive 2.4 GHz ISM RF transceiver
- 256kbps, 1Mbps, 2Mbps data rates
- Works on 3.3V ONLY





nRF24L01 → Arduino

GND → GND

3v3 → 3.3V

CE → D8

CSN → D7

SCK → D13

MOSI → D11

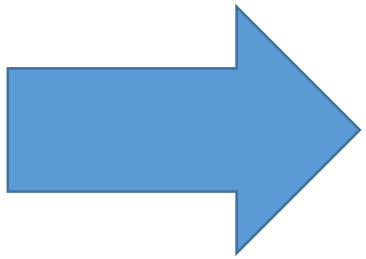
MISO → D12

See http://playground.arduino.cc/InterfacingWithHardware/Nrf24L01

Use the Mirf library

nRF24L01 Transmitter Receiver Example

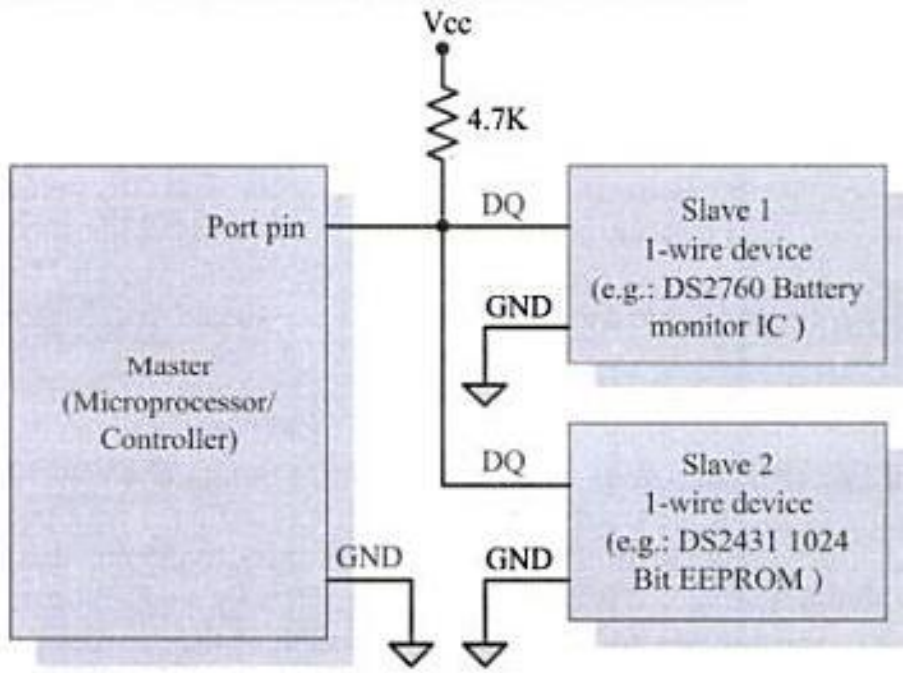
```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

void transmit(const char *string) {
  byte c;

  for( int i=0 ; string[i]!=0x00 ; i++ ) {
    c = string[i];
    Mirf.send(&c);
    while( Mirf.isSending() ) ;
  }
}

void setup(){
  Serial.begin(9600);

  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"serv1");
  Mirf.payload = 1;
  Mirf.config();
}

void loop(){
  transmit("Hello World!~");
  delay(1000);
}
```



```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

String message;

void setup(){
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"serv1");
  Mirf.payload = 1;
  Mirf.config();
}

void loop(){
  byte c;
  if(!Mirf.isSending() && Mirf.dataReady()){
    Mirf.getData(&c);
    char letter = char(c);
    if (letter != '~') {
      message = String(message + letter);
    } else {
      Serial.println(message);
      message = "";
    }
  }
}
```

1-Wire Interface (1)

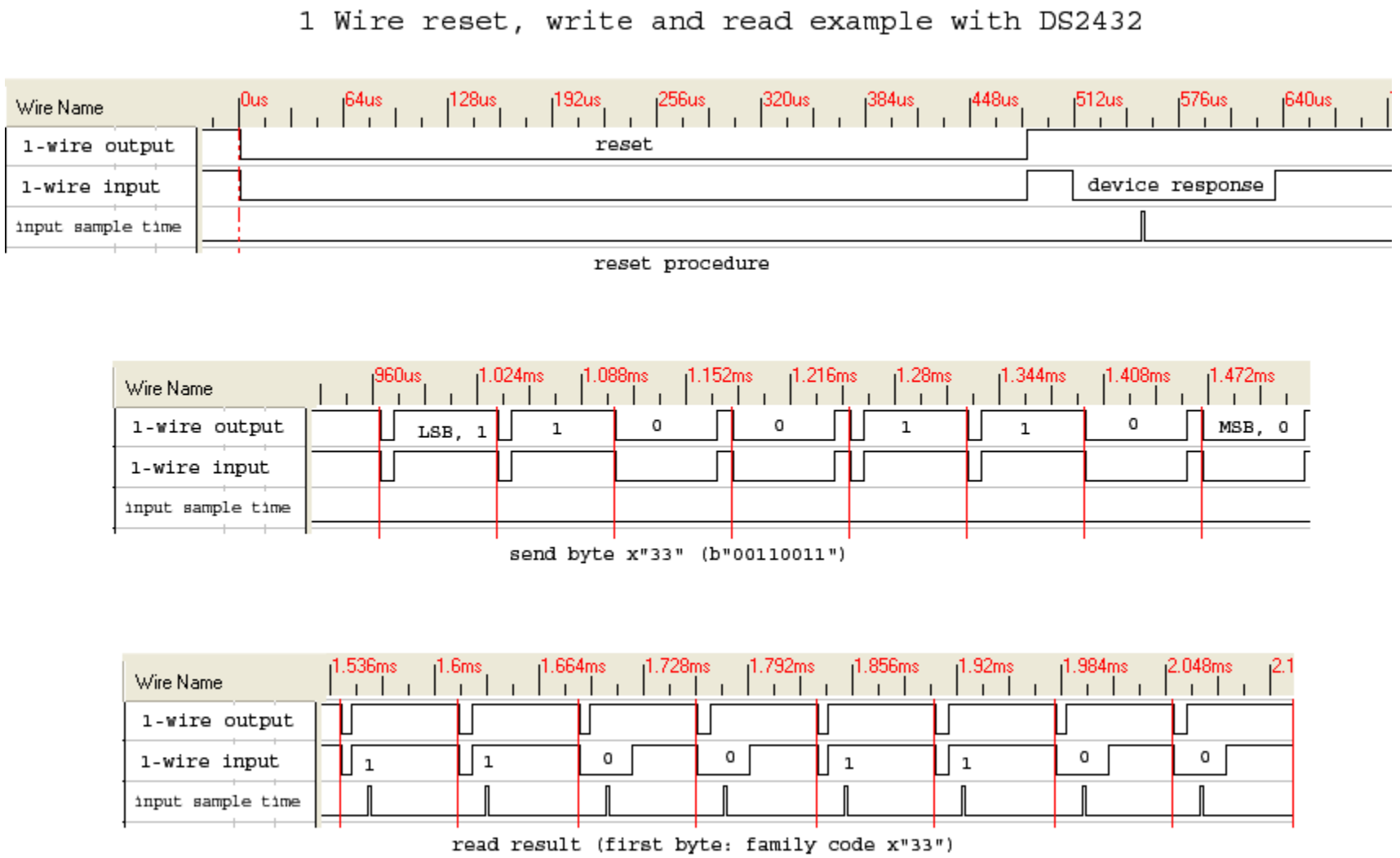
- 1-wire interface is an asynchronous half-duplex communication protocol (originally developed by Maxim Dallas Semiconductor)
- It uses only a single signal line (wire) called DQ for communication and follows the master-slave communication model
- The power is also sent along the signal wire and the slave devices incorporate internal capacitor (order of 800 pF) to power the device from signal line
- Every 1-wire device contains a globally unique 64-bit identification number stored with it which is used for addressing individual devices
  - 64-bit has three parts: 8-bit family code + 48-bit serial number + 8-bit CRC



1-Wire Interface (2)

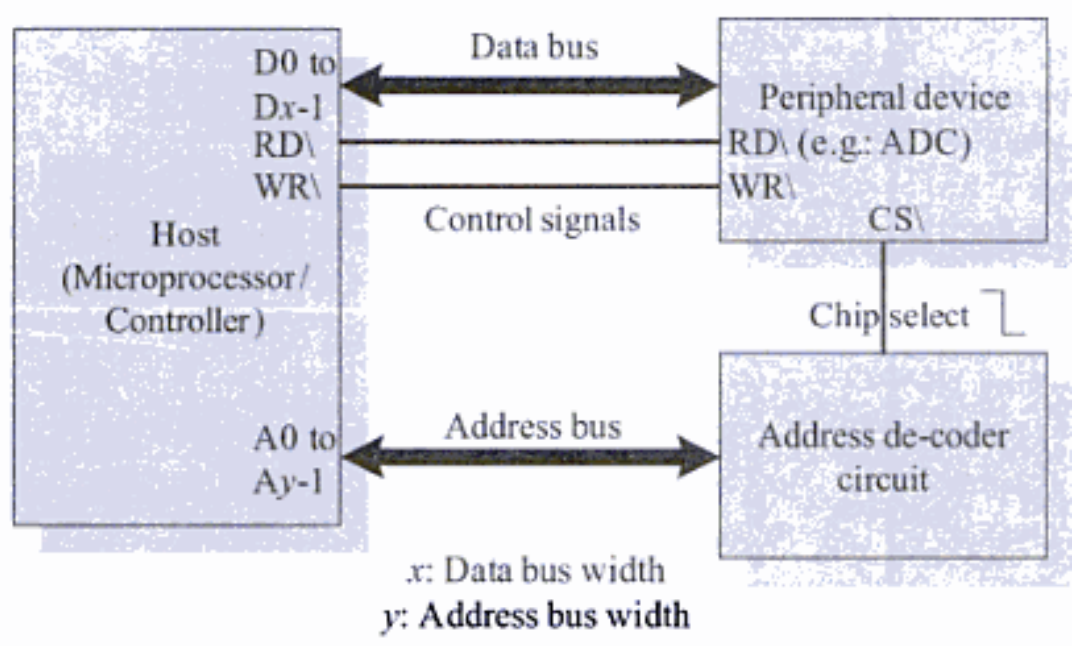
- The communication over the 1-wire bus is divided into timeslots of 60 microseconds
- The sequence of operation in 1-wire is:
  - The master device sends a ‘Reset’ pulse on the 1-wire bus – 8 timeslots (480uS) by pulling the wire ‘LOW’
  - The slave device(s) present on the bus respond with a ‘Presence’ pulse – within 60uS of release of ‘Reset’ pulse by pulling the wire ‘LOW’
  - The master device sends a ROM command. This addresses the slave device(s) to which it wants to initiate a communication
  - The master device sends a read/write function command to read/write the initial memory or register of the slave device
  - The master initiates a Read data/Write data from the device or to the device
    - Write bit=1 to the slave – pulls the wire for 1-15uS and releases for the rest of the period
    - Write bit=0 to the slave – pulls the wire for 60-120uS
    - Read bit=1 from the slave – master pulls for 1-15uS, slave does nothing for the rest of the timeslot
    - Read bit=0 from the slave – master pulls for 1-15uS then slave pulls ‘LOW’ for the rest of the timeslot

1-Wire Interface (3)



On-Board Parallel Interface

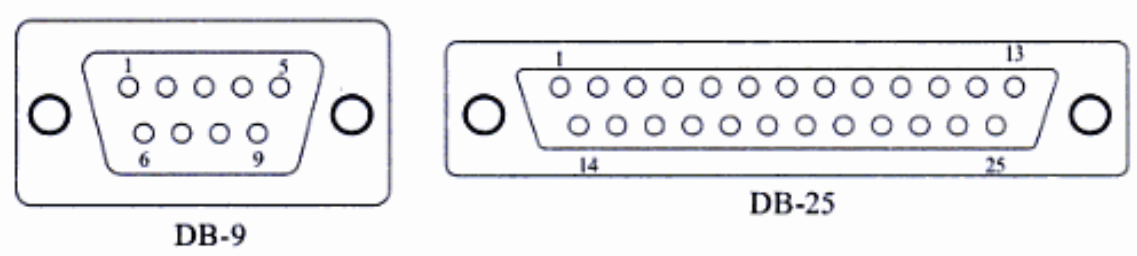
- The parallel bus is used for communicating with peripheral devices which are memory mapped to the host of the system
- The communication through the parallel bus is controlled by the control signal ('Read/Write' and device select signals) between the device and the host
- The communication in parallel bus is host processor initiated. However, if a device wants to initiate communication, it can inform the processor through interrupts
- The width of the parallel bus can be 4-bit, 8-bit, 16-bit, 32-bit, 64-bit, etc. and it offers the highest speed of data transfer with compared to serial communication

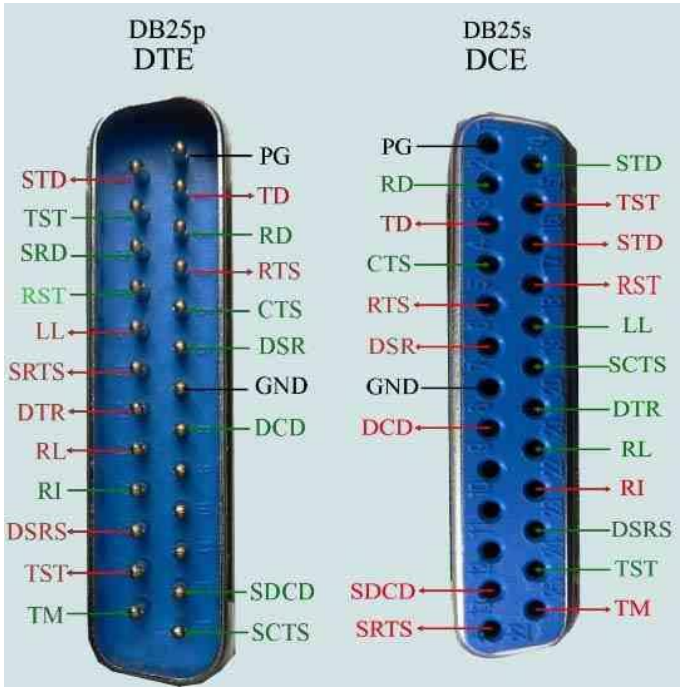


RS-232 C & RS-485 for External Communication (1)

- RS-232 C is a legacy, full duplex, wired, asynchronous serial communication interface (developed by Electronic Industry Association during 1960s). It extends the UART communication signals for external data communication
  - RS-232 follows the EIA standard for bit transmission: logic ‘0’ = +3 to +25V and logic ‘1’ = -3 to -25V
- RS-232 defines various handshaking and control signals for communication apart for transmit and receive signal lines
- RS-232 supports two different types of connectors, namely, DB-9: 9-Pin connector and DB-25: 25-pin connecctor

RS-232 C & RS-485 for External Communication (2)

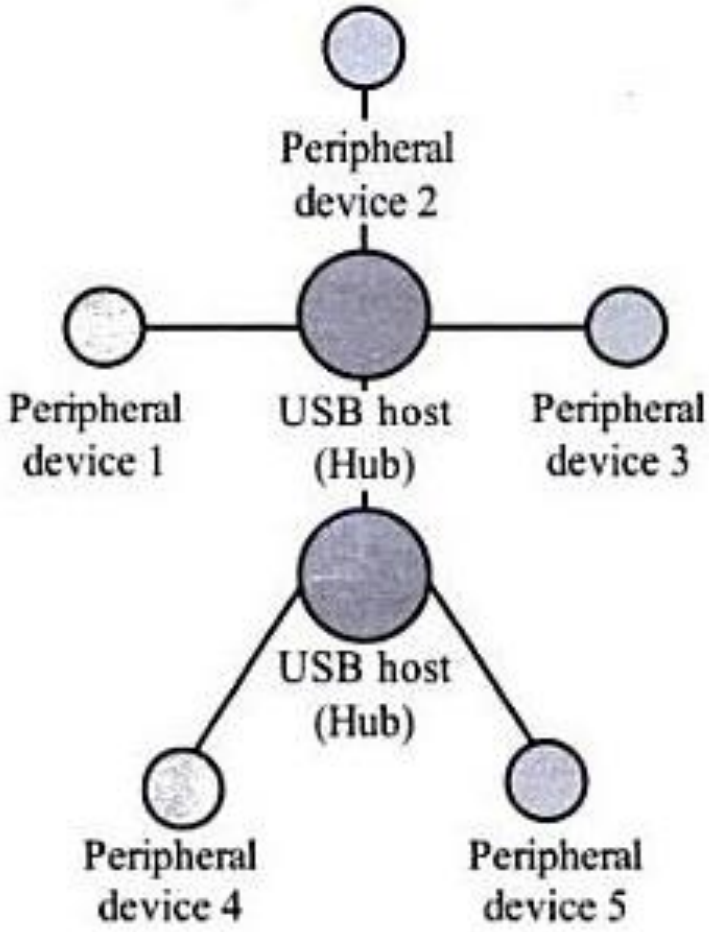




Pin Name	Pin no: (For DB-9 Connector)	Pin no: (For DB-25 Connector)	Description
TXD	3	2	Transmit Pin for Transmitting Serial Data
RXD	2	3	Receive Pin for Receiving Serial Data
RTS	7	4	Request to send.
CTS	8	5	Clear To Send
DSR	6	6	Data Set Ready
GND	5	7	Signal Ground
DCD	1	8	Data Carrier Detect
DTR	4	20	Data Terminal Ready
RI	9	22	Ring Indicator
FG		1	Frame Ground
SDCD		12	Secondary DCD
SCTS		13	Secondary CTS
STXD		14	Secondary TXD
TC		15	Transmission Signal Element Timing
SRXD		16	Secondary RXD
RC		17	Receiver Signal Element Timing
SRTS		19	Secondary RTS
SQ		21	Signal Quality detector
NC		9	No Connection
NC		10	No Connection
NC		11	No Connection
NC		18	No Connection
NC		23	No Connection
NC		24	No Connection
NC		25	No Connection

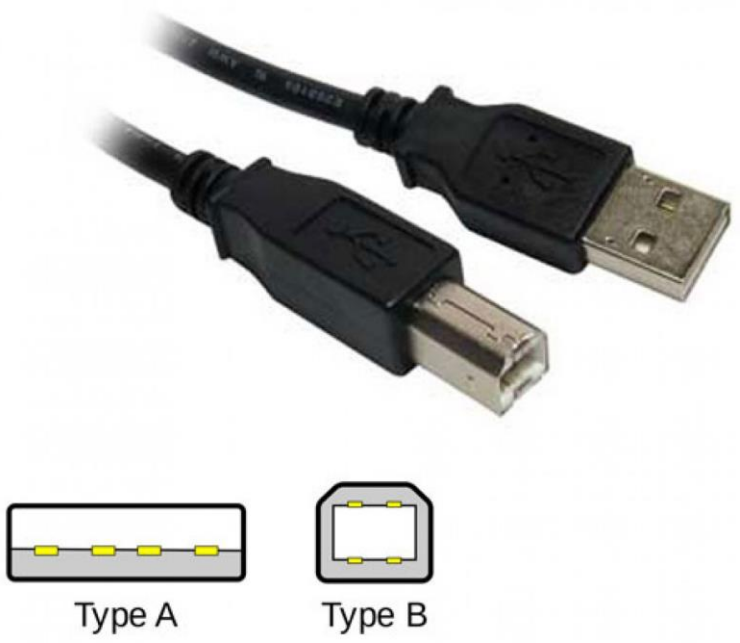
Universal Serial Bus (USB) (1)

- USB is a wired high speed serial bus for data communication
  - Created by a group members consisting of Intel, Microsoft, IBM, Compaq, Digital and Northern Telecom (version 1.0 in 1995)
- It follows a star topology with a USB host at the center and one or more USB peripheral devices/USB hosts connected to it
  - A USB host can support connections up to 127, including slave peripheral devices and other USB hosts
- Presently USB supports four different data rates:
  - Low Speed (USB1.0, 1.5Mbps), Full Speed (USB1.0, 12Mbps), High Speed (USB2.0, 480Mbps), Super Speed (USB3.0, 4.8Gbps)

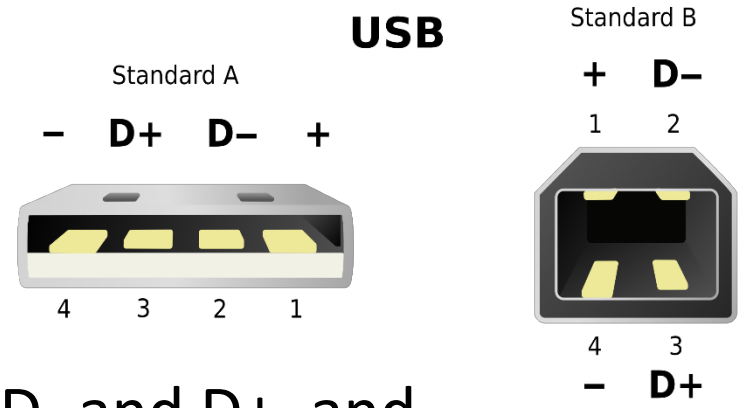


Universal Serial Bus (USB) (2)

- USB transmits data in packet format
- USB communication is host initiated:
  - A host controller is responsible for controlling the data communication, including establishing connectivity with USB slave devices, packetizing and formatting the data
  - There are different standards for implementing the host control interface such as Open Host Control Interface (OHCI) and Universal Host Control Interface (UHCI)
- The physical connection between a USB peripheral device and master device is establish with a USB cable which supports communication distance up to 5 meters
  - ‘Type A’ connector: used for upstream connection (connection with host)
  - ‘Type B’ connector: used for downstream connection (connection with slave device)
  - Additionally, mini and micro USB connectors are available for small form factor devices



Universal Serial Bus (USB) (3)



- USB uses four pins: +5V, Differential data carrier lines D- and D+, and GND
- USB interface has the ability to supply power up to 500 mA at 5V to a connecting device
- Each USB device contains a Product ID (PID) and a Vendor ID (VID) embedded into the USB device manufacturer. PID and VID are essential for loading the drivers corresponding to a USB device
- USB supports four different types of data transfers:
  - Control transfer is used by USB system software to query, configure and issue commands to the USB device
  - Bulk transfer is used for sending block of data to a device, e.g., printer
  - Isochronous data transfer is used for real-time data communication, e.g., audio devices
  - Interrupt transfer is used for transmitting small amount of data, e.g., mouse and keyboard

IEEE 1394 (Firewire)

- IEEE 1394 is a wired, isochronous high speed serial communication bus which is also known as High Performance Serial Bus (HPSB)
  - Implementations: Apple's Firewire, Sony's i.LINK, and Texas Instrument's Lynx
- IEEE 1394 supports peer-to-peer connection and point-to-multipoint communication allowing 63 devices to be connected on the bus in a tree topology
- It supports a data rate of 400 to 3200Mbps and a cable length up to 15 feet
- It supports three different types of connectors: 4-pin, 6-pin (alpha), and 9-pin (beta) connectors

Pin name	Pin no: (4 Pin Connector)	Pin no: (6 Pin Connector)	Pin no: (9 Pin Connector)	Description
Power		1	8	Unregulated DC supply. 24 to 30V
Signal Ground		2	6	Ground connection
TPB–	1	3	1	Differential Signal line for Signal line B
TPB+	2	4	2	Differential Signal line for Signal line B
TPA–	3	5	3	Differential Signal line for Signal line A
TPA+	4	6	4	Differential Signal line for Signal line A
TPA(S)			5	Shield for the differential signal line A. Normally grounded
TPB(S)			9	Shield for the differential signal line B. Normally grounded
NC			7	No connection