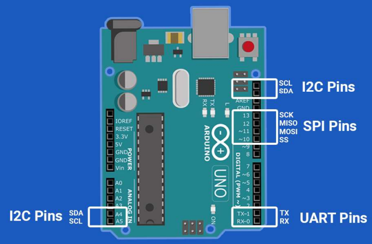
Communication

1

Functions Related to Serial Communication



2

Inter-Processor Communication

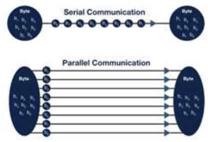
- Need for MCU to communicate with external devices arises often in Mechatronics
- Oftentimes we need MCU to share digital data with another device. Examples:
  - MCU receives data from GPS receiver
  - MCU transmits data to computer
  - MCU communicates with other MCU's

Courtesy of Texas Instruments

3

Serial vs. Parallel Communication

- When two devices communicate binary data, there are two ways to implement it:
  - **Parallel** – multiple bits are sent simultaneously
  - **Serial** – one bit sent at a time



- Parallel communication requires more TX/RX lines (wires)
- 2-way serial can be implemented with just **3 wires**
- Parallel communication used to be popular because it provided higher communication bandwidth

Now that processor speeds are so fast, simplicity of serial has outweighed bandwidth penalty. Now serial is used almost exclusively.

4

Synchronous vs. Asynchronous Communication

- Likewise, interdevice communication can occur **synchronously** or **asynchronously**

Synchronous	Data is sent in **continuous stream** at **constant rate** Clocks in transmitting and receiving devices must run at exactly the same rate (be synchronized)
Asynchronous	**Data sent as needed**, not in continuous stream Clocks can be running at different rates, but devices just agree on data rate **Start and stop bits** used to synchronize

5

Modern MCU Communication

- Digital communication to and from modern microcontrollers is commonly done using **asynchronous serial interfaces**
  - Specifically, Universal Asynchronous Receiver Transmitter (UART) hardware device

TL28L92 Dual UART IC (Courtesy of Texas Instruments)

Most modern microcontrollers have one or more onboard UARTs built in.

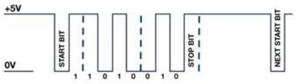
UART Serial Communication

- UART Serial transmission involves sending **one bit at a time**
  - For 1-way communications, only **two wires** are needed: Signal, Ground
  - For 2-way communications, only **three wires** are needed: Signal out, Signal in, Ground
- To send a byte of data, we must send 8 bits individually, plus a start and a stop bit
  - Total of 10 bits for each byte of data we send!

6

TTL Serial Communication

- TTL (Transistor-transistor logic) means 3.3V or 5V is logic high, 0V is logic low
- **1 byte transmitted at a time** (called a "frame")
  - LSB transmitted first, MSB transmitted last
- Example: Transmission of 01001011 (75 decimal) using 5V TTL levels



- **Start bit**: Transmission line is at 5V at idle
  - To start transmission, line goes low (start bit)
- **Data bits**: 8 data bits sent at one time, LSB first
- **Stop bit**: Line goes high after 8th data bit
  - This signifies end of current frame
  - Process is repeated for next frame

7

7

Baud Rate

- Total number of bits transmitted per second is called the baud rate
- The bit time T is 1/(baud rate), signifying how long it takes to transmit a single bit
- Baud rate is set during serial comms configuration
  - Both devices must use the same baud rate
- Baud rate tradeoffs:
  - High baud rates → higher communication speed
  - Low baud rates → higher accuracy

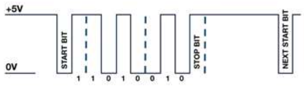
Bytes per sec = (Baud rate) / 10

8

8

Serial Baud Rate

- Recall that the serial **baud rate** is the number of bits transmitted per second with our serial protocol
  - Standard UART serial requires 10 bits to transmit 1 byte: 8 data bits, 1 start bit, 1 stop bit
  - Typical Baud Rates (bits/sec): 9600, 57600, 115200



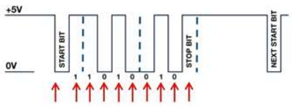
- Recall that there is no shared clock signal between devices communicating with serial UART
  - Device 1 might think that 1 ms is "slightly longer" than Device 2
  - But they both agree on a Baud Rate

12

12

Serial Baud Rate

- Consider the example frame shown below



- Once start bit is detected (line goes low), receiver waits 1.5 x T and samples line 9 times at T sec intervals to read data (T is bit time)
- How long does it take to transmit each bit if the Baud rate is 9600? How many bytes are transmitted each second?

Answer:

T = 1/9600 = 104.2 µs

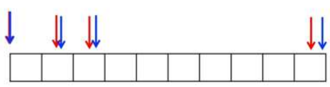
Bytes per sec = 9600/10 = 960

13

13

- If clock on receiving device is off by 4 µs each time you read a bit, will the data be read correctly at:
  - 9600 Baud?
  - 38400 Baud?

9600 Baud: T = 1/9600 = 104.2 µs



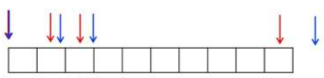
- Blue arrow is time that receiving device samples signal, red is center bit time of sending device
- By last bit, we will be 9 x 4 = **36 µs** off, which is still well within half of our bit time (T = 104.2 µs)
- **So data will be read correctly**

14

14

- If clock on receiving device is off by 4 µs each time you read a bit, will the data be read correctly at:
  - 9600 Baud?
  - **38400 Baud?**

38600 Baud: T = 1/38600 = 26.0 µs



- Blue arrow is time that receiving device samples signal, red is center bit time of sending device
- By last bit, we will be 9 x 4 = **36 µs** off, which is way longer than half of our bit time (T = 26 µs)
- **So data will NOT be read correctly**

Clock rate mismatches between sending and receiving devices are the main factor that limits our Baud rate

Modern microcontrollers use fairly low-cost clocks (digitally-controlled oscillators) that limit Baud rates around 100,000 bits/sec

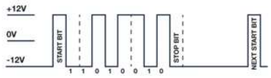
15

15

RS-232

- RS-232 is a serial communication protocol used by computers (not by microcontrollers)
- Same basic protocol except with different voltage levels:
  - +12 V is considered "low"
  - -12 V is considered "high"
  - 0 V is ground reference

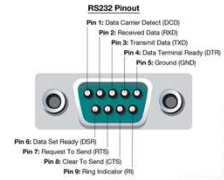
  Typical values used in PC's, but can vary between 5-15V
- Transmission of decimal 75 using RS-232:



16

RS-232

- Older computers had RS-232 port which used DB9 connectors (before USB)
- 9 total wires, but for simple RS-232 communication only 3 wires are needed
  - Pin 2 (receive)
  - Pin 3 (transmit)
  - Pin 5 (common ground)



17

Serial Conversion to RS-232 and USB

- Because TTL serial uses a different protocol and voltage levels than RS-232 and USB, we often need to **convert TTL serial to RS-232 or USB** when connecting a microcontroller to a computer
  - Special conversion boards (or cables) used for this purpose

- ***Pololu 23201a Serial Adapter Fully Assembled***
- *Converts TTL serial to RS-232 and vice versa*
- *Pins for TTL serial TX/RX on one side, DB9 connector on other*

Courtesy of Pololu

- ***Pololu USB to serial adapter***
- *Converts TTL serial to USB and vice versa*
- *Pins for TTL serial TX/RX on one side, mini-USB connector on other*
- *Use when sending data from MCU (UART) to computer via USB*

18

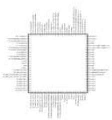
UART Operation

20

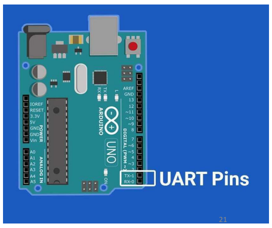
Universal Asynchronous Receiver-Transmitter

- Recall that UART is a **physical hardware device** that implements TTL serial protocol
  - UART integrated circuits are available, but they are already integrated into most microcontrollers
  - MSP432 has four UARTs. From datasheet we find:

UART0 RX Pin = P1.2
UART0 TX Pin = P1.3
UART1 RX Pin = P2.2
UART1 TX Pin = P2.3
UART2 RX Pin = P3.2
UART2 TX Pin = P3.3
UART3 RX Pin = P9.6
UART3 TX Pin = P9.7



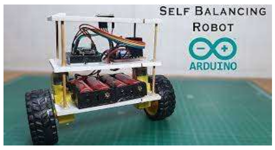
Courtesy of Texas Instruments



21

I2C Communications Protocol

IMO sensor : accelerometer and gyro

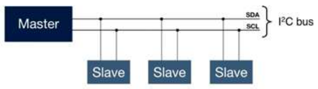


https://youtu.be/nPYRuT2dx7o

34

I²C Communication

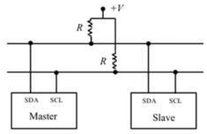
- **Inter-Integrated Circuit** (I²C) is a common serial protocol used to communicate between microcontrollers, sensors, and actuators
  - Developed by Phillips Semiconductors in 1980's
- Primary advantage over UART is that **multiple devices can communicate over same 2-wire bus**
- **SDA:** Data wire. Data bits flow along this line.
- **SCL:** Clock wire. This transmits clock signal which is used to synchronize data transfer.
- **Master:** Device (usually MCU) which controls clock (SCL) line. Usually only 1 per bus.
- **Slaves:** Devices (MCUs, sensors, actuators, etc) that monitors SCL and communicates on SDA.



36

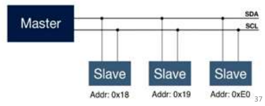
I²C Wiring

- When implementing I²C bus, pull-up resistor is usually needed on each line to set the logic level being used (either 3.3V or 5V).
- On MSP432 Launchpad, SCL and SDA lines already tied to 3.3V lines via internal resistor, so no need to use external one



I²C Slave Addressing

- Only I²C Master can initiate data transfer
- I²C Slaves respond to Master request
- Data can be sent in either direction
- Each slave has a **unique address** assigned to it
  - Usually 7 bits but sometimes 10 bits
  - You can set address values of each slave device



37

I²C Protocol

Message

Start	7 or 10 Bits	Read/ Write Bit	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	Stop
Start	Address Frame			Data Frame 1		Data Frame 2		Stop

- In I2C communication, data is sent in **messages**
- Each message is broken up into **frames**
- **Address Frame** specifies which slave should be communicating (holds slave address)
- **Data Frames** hold binary data that is transmitted (can be arbitrary number in single message)

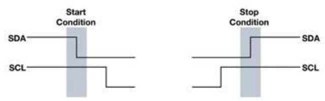
38

I²C Protocol

Message

Start	7 or 10 Bits	Read/ Write Bit	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	Stop
Start	Address Frame			Data Frame 1		Data Frame 2		Stop

- Every message starts with **Start Condition**, ends with **End Condition**



- **Start:** SDA line changes from high to low while SCL (clock line) held high
- **End:** SDA line changes from low to high while SCL (clock line) held high
- Only time SDA is allowed to change while SCL remains constant

39

I²C Protocol

Message

Start	7 or 10 Bits	Read/ Write Bit	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	8 Bits	ACK/ NACK Bit	Stop
Start	Address Frame			Data Frame 1		Data Frame 2		Stop

- Every frame ends with **ACK or NACK bit**
- If frame (address or data) was successfully received by receiving device, **ACK bit sent** (0). If frame was not successfully received or understood, receiving device sends **NACK bit** (1).

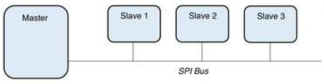
40

Serial Peripheral Interface
SPI protocol

49

Serial Peripheral Interface

- **Serial peripheral interface** (SPI) is a common serial communication protocol used for embedded devices
- Like I2C, SPI uses the notion of "master" and "slave" devices
  - Master device is usually microcontroller
  - Slave devices can be sensors, microcontrollers, actuators, etc.



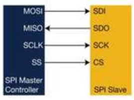
- **Advantages:**
  - Extremely fast protocol often used for streaming continuous data
    - Data rates **up to 10 Mbps** (a lot faster than I2C)
    - Unlike I2C, data can flow continuously without being interrupted by address bytes, start/stop bits, etc.
  - One bus can handle (theoretically) unlimited number of slaves
- **Disadvantages:**
  - Uses four wires to connect only two devices, additional wires for more slaves.

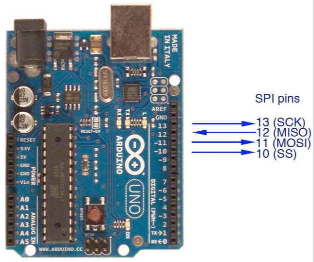
51

51

Serial Peripheral Interface – Wiring

- Four wires used to connect two SPI devices
  - **SCLK**: Clock wire. This transmits clock signal which is used to synchronize data transfer.
  - **MOSI**: Master out slave in wire. Transfers data from Master to slave device.
  - **MISO**: Master in slave out wire. Transfers data from slave to Master device.
  - **SS**: Slave select wire. Activates slave for communication.





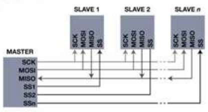
52

52

SPI Master-Slave Configuration

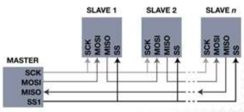
- Multiple slaves can be located on same SPI bus

General Configuration



- Use multiple slave select lines
- Set SSx line to low to read from slave x
- Most commonly-used configuration - this is what we will focus on.
- Each slave requires its own slave select wire.

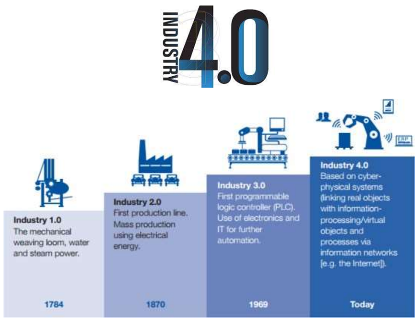
Daisy-Chained Configuration



- Uses single slave select line
- Data passed from Master through slaves sequentially
- Used when sending data out from master exclusively
- MISO line on Master typically not used

53

53



61

61

Industrial Automation

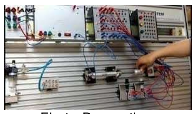
- **Industry 3.0** Industry Revolution
- Wide range of technologies that reduces **manual intervention**
- **Mechanical, Hydraulic, Pneumatic, Electrical and Computers/Controllers**
- Can be **closed loop or open loop system**
- **Discrete Control, PID control or Sequential Control**

prapoational integral
derivative

62

Industrial Automation


Pneumatics


Electro Pneumatics


Sensors


Hydraulics


Controllers

63

Controllers

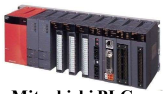
- It's a **Computer Based Device**
- **Monitors and controls** the process
- It is the **Brain of the application**
- Programs can be written depending on the application
- **Receives the data** from inputs and **gives input to the devices.**

64

Types of Controllers


Rockwell PLC


Siemens PLC


Mitsubishi PLC

Industrial Controllers


Arduino UNO


Raspberry Pi


Node MCU

MicroControllers

65

Poll

Arduino can be used in Industrial MPRC system

a. True

b. False

66

MicroControllers Vs PLC

- PLCs are designed with the **ruggedness and ability to work** in industrial environment conditions like shock, vibration, noise, temperature etc.
- Industrial sensors and actuators are designed according to the IEC standard which **works at a range of current/voltage** and can be directly connected to PLC
- Multiple tasks can be performed.
- **Large number of I/Os** can be connected with PLC.
- Can be easily mounted on a panel
- **More efficient, sturdy and reliable** than microcontroller

- Can be used in small applications. **Not an ideal system** for industries as it cannot withstand extreme conditions.
- Additional Hardware is required to connect industrial sensors and actuators which increases the cost.
- Designed to run **one specific program** which is dedicated to one certain task.
- **Limited number of I/Os** are compatible with microcontroller
- Complex Mounting
- **Less efficient and reliable**

67

68