Introduction to I2C & SPI
Issues with Asynch. Communication Protocols

Asynchronous Communications

- Devices must agree ahead of time on a data rate
- The two devices must also have clocks that are close to the same rate
- Excessive differences between clock rates on either end will cause garbled data
- Asynchronous serial ports require hardware overhead
- The UART at either end is relatively complex and difficult to accurately implement in software if necessary
- Most UART devices only support a certain set of fixed baud rates, and the highest of these is usually around 230400 bits per second

https://learn.sparkfun.com/tutorials/i2c?_ga=1.229206105.432923716.1447312656
Asynchronous Transmission

Example of an “A” followed by a “B”

A = 0x41

B = 0x45
Synchronous Transmission

Example of an “A” followed by a “B”

Clock

Data

A = 0x41
B = 0x42
The Inter-integrated Circuit (I²C)

- The Inter-integrated Circuit (I²C) Protocol is a protocol intended to allow multiple “slave” (or secondary) digital integrated circuits (“chips”) to communicate with one or more “master” chips.
- Multi-master system, allowing more than one master (or primary) to communicate with all devices on the bus.
- When multiple primary devices are used, the master devices can’t talk to each other over the bus and must take turns using the bus lines.
- In I²C there are three additional modes specified: fast-mode plus, at 1MHz; high-speed mode, at 3.4MHz; and ultra-fast mode, at 5MHz.

Characteristics
- Serial, byte-oriented
- Multi-master, multi-slave
- Two bidirectional open-drain lines, plus ground
  - Serial Data Line (SDA)
  - Serial Clock Line (SCL)
  - SDA and SCL need to pull up with resistors
• A **START** condition is a high-to-low transition on SDA when SCL is high.
• A **STOP** condition is a low to high transition on SDA when SCL is high.
• The address and the data bytes are sent most significant bit first.
• **Master generates** the clock signal and sends it to the slave during data transfer
Inter-Integrated Circuit (I2C)

- **Master generates** the clock signal

  ![I2C Circuit Diagram](image)

- SDA and SCL have to be **open-drain**
  - Connected to positive if the output is 1
  - In high impedance state if the output is 0
- Each Device has an unique address (7, 10 or 16 bits). Address 0 used for broadcast
- STM32 internal pull-up is too weak (internal 100KΩ)
- External pull-up (4.7 kΩ for low speed, 3 kΩ for standard mode, and 1 kΩ for fast mode).
  - Fast mode refers to fast rise time!
Inter-Integrated Circuit (I2C)

- The I²C bus drivers are OPEN DRAIN meaning that they can pull the corresponding signal line **low, but cannot drive it high**
- There can be no bus contention where one device is trying to drive the line high while another tries to pull it low, eliminating the potential for damage to the drivers or excessive power dissipation in the system
- STM32 internal pull-up is too weak (internal 100KΩ)
- External pull-up (4.7 kΩ for low speed, 3 kΩ for standard mode, and 1 kΩ for fast mode – fast rise time!)

“Wired-AND” bus: A sender can pull the lines to low, even if other senders are trying to drive the lines to high
Multiple Masters

Connected to positive if the output is 1
In high impedance state if the output is 0

- In single master systems, arbitration is not needed.
- **Arbitration** for multiple masters:
  - During data transfer, the master constantly checks whether the SDA voltage level matches what it has sent.
  - When two masters generate a START setting **concurrently**, the first master which detects SDA low while it has actually intended to set SDA high will **lose the arbitration** and let the other master complete the data transfer.
Basic Protocol (7-bit Addressing)

- Setting the address
- Write the data value
- R/W: 0 --> Primary sends data

1= Master requires data
0=Master sends data
Basic Protocol (10-bit Addressing)

- Setting the address
- Write the data value
- R/W: 0 --> Primary sends data

Code: 11110 xx
Repeated Starts

Despite the idle state of the bus, no other master may assert control of the bus during this period.

After the repeated start, a new transfer, complete with address frame(s), must begin.

Last data frame of prior transfer

No stop condition is present

Another start occurs-this is the "repeated start"
Interfacing Serial Digital Thermal Sensor

![Diagram of TC 74 Serial Digital Thermal Sensor Interconnection]
Communicating with TC74 with an address of 0x4D

I²C Master

- Start Bit
- 7-bit Address: 1 0 0 1 1 0 1
- R/W: 0

I²C Slave (TC 74)

- ACK Bit
- 8-bit Command: 0 0 0 0 0 0 0 1
- 8-bit Data: 1 0 0 0 0 0 0 0
- ACK Bit
- Stop Bit

Select which TC74's register is written to:
- 0x00 = Temperature register
- 0x01 = Configuration register

Diagram showing the connection between the I²C Master and the TC74 slave, including the pins and their connections.
I2C Data
Sending Data to I2C Secondary Via Polling

1. Configuring register CR2
   - Set direction as writing to slave
   - Number of bytes to be written
   - Slave address
   - Start bit
     \[
     \text{Hardware sets TXIS flag if register TXDR is empty}
     \]
2. Wait until hardware sets TXIS flag in register ISR
3. Write data to register TXDR
   \[
   \text{Hardware clears TXIS flag if the byte to be sent is written to TXDR}
   \]
4. Wait until hardware sets TXIS flag in register ISR
5. Write data to register TXDR
   \[
   \text{Hardware sets TC flag when transmission completes.}
   \]
6. Wait until hardware sets TC flag in register ISR

\[
\begin{align*}
\text{I}^2\text{C Master} & \quad \text{Send One Start Bit} \\
\text{I}^2\text{C Slave} & \quad \text{Master pulls SDA low} \\
\text{Send Eight Bits (7-bit addr + 0)} & \quad \text{LSB = 0 indicating the Master is the transmitter.} \\
\text{1-bit Acknowledgement} & \quad \text{Send 1\textsuperscript{st} Data Byte} \\
\text{1-bit Acknowledgement} & \quad \text{Send 2\textsuperscript{nd} Data Byte} \\
\text{1-bit Acknowledgement} & \quad \text{Send One Stop Bit} \\
\text{Master pulls SDA high} & \\
\end{align*}
\]
A Brief Introduction to SPI

SPI: Serial Peripheral Interface
Serial Peripheral Interface (SPI)

- Synchronous full-duplex communication
- Can have multiple slave (secondary) devices
- No flow control or acknowledgment
- Slave (secondary) cannot communicate with slave directly.

- SCLK: serial clock
- MOSI: master out slave in
- SS: slave select (active low)
- MISO: master in slave out
Data Exchange

- Master has to provide clock to slave
- **Synchronous exchange**: for each clock pulse, a bit is shifted out and another bit is shifted in at the same time. This process stops when all bits are swapped.
- Only master can start the data transfer
Clock
Read and Write

Diagram showing the communication between a master and a slave in a digital interface. The diagram includes labels for SCK (clock), MOSI (master out, slave in), MISO (master in, slave out), and SS (slave select). The timing is shown for both master to slave and slave to master communications, with examples of binary data transmission and corresponding ASCII characters.
Multiple Chip Selects