Introduction to I2C & SPI

Chapter 22
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 (I\textsuperscript{2}C)

• The Inter-integrated Circuit (I\textsuperscript{2}C) 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 I\textsuperscript{2}C 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)

- 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 100Ω)
- 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

- 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.

SDA: D7 ACK

SCL:

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 interfacing serial digital thermal sensor](image)
Communicating with TC74 with an address of 0x4D

I²C Master
- Start Bit
- 7-bit Address: 1001101
- R/W: 0

I²C Slave (TC 74)
- 8-bit Command: 00000000
- 8-bit Data: 10000000
- ACK Bit
- Stop Bit

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

Diagram of TC74 connections:
- VDD (5V)
- GND
- SDA
- SCL
- DO (Output)
- DI (Input)
- INT (Interrupt)
- 1kΩ pull-up resistors for SDA and SCL
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
     *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
   *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
   *Hardware sets TC flag when transmission completes.*
6. Wait until hardware sets TC flag in register ISR

---

**I²C Slave**

- Send One Start Bit
- Send Eight Bits (7-bit addr + 0)
- 1-bit Acknowledgement
- Send 1st Data Byte
- 1-bit Acknowledgement
- Send 2nd Data Byte
- 1-bit Acknowledgement
- Send One Stop Bit
- Master pulls SDA high

**I²C Master**

- Send One Start Bit
- Write
- Send Eight Bits (7-bit addr + 0)
- 1-bit Acknowledgement
- Send 1st Data Byte
- 1-bit Acknowledgement
- Send 2nd Data Byte
- 1-bit Acknowledgement
- Send One Stop Bit
- Master pulls SDA low

LSB = 0 indicating the Master is the transmitter.
void I2C_Initialization(void) {
    // Enable the clock of I2C
    RCC->APB1ENR |= RCC_APB1ENR_I2C2EN; // I2C 2 clock enable

    // Reset the I2C (set and reset by software)
    RCC->APB1RSTR |= RCC_APB1RSTR_I2C2RST;
    RCC->APB1RSTR &= ~RCC_APB1RSTR_I2C2RST;

    // I2C Control register. ACK is set and cleared by software, and cleared by hardware when PE=0.
    I2C2->CR1 |= I2C_CR1_ACK; // Acknowledge after a byte is received
    I2C2->CR1 &= ~I2C_CR1_SMBUS; // SMBus Mode: 0 = I2C mode; 1 = SMBus mode
    I2C2->CR2 &= ~I2C_CR2_FREQ; // Clear Peripheral CLK frequency FREQ[5:0]
    I2C2->CR2 |= I2C_CR2_FREQ_1; // Set Peripheral clock frequency as 2 MHz
    I2C2->CR2 |= I2C_CR2_ITEVTEN; // Event Interrupt Enable
    I2C2->CR1 &~ I2C_CR1_PE; // Disable I2C to configure TRISE
    I2C2->TRISE &~ I2C_TRISE_TRISE; // Clear Maximum Rise Time
    I2C2->TRISE |= 17; // Set Maximum Rise Time for standard mode

    // I2C own address registers
    // Before the STM32 sends its start sequence it sits listening to the I2C lines waiting for its address
    // This is helpful if STM32 is used as slave
    I2C2->OAR1 %= ~(0x01);
    I2C2->OAR1 = 0x00; // Set up the address byte to select the slave device
    I2C2->OAR2 = 0x00; // Set up the I2C own address2

    // I2C Clock control register
    I2C2->CCR &~ I2C_CCR_FS; // 0 = Standard Mode ((up to 100 kHz));
    // 1 = Fast Mode (up to 400 kHz)
    I2C2->CCR &~ I2C_CCR_CCR;
    I2C2->CCR |= 0x08;
    I2C2->CR1 |= I2C_CR1_PE; // Enable I2C1
}
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 directly with slave

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

**Lines**
- **SCLK**: serial clock line
- **MOSI**: master output, slave input line
- **MISO**: master input, slave output line
- **SS**: slave select line (active low)
• 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

\[ SCLK \]
\[ \overline{SSN_n} \]
\[ \text{MOSI} \]
\[ b_{out}[0] \quad b_{out}[1] \quad b_{out}[2] \quad b_{out}[3] \quad b_{out}[4] \quad b_{out}[5] \quad b_{out}[6] \quad b_{out}[7] \]

\[ \text{MISO} \]
\[ b_{in}[0] \quad b_{in}[1] \quad b_{in}[2] \quad b_{in}[3] \quad b_{in}[4] \quad b_{in}[5] \quad b_{in}[6] \quad b_{in}[7] \]
Read and Write

[Diagram showing Master and Slave connections and timing for data transfer]
Multiple Chip Selects