Introduction to Embedded Systems

EE 186 – Lecture 6

Zerina Kapetanovic

Announcements

- HW 3 (due Thurs 10/22) and Lab 3 (due next Tues 10/27), Final Project Proposal (due next Tues 10/27)
- Oral discussion for Lab 1-2 This Friday
  - Sign up for individual slot & join Zoom on time
  - Be prepared to share your screen (your code & relevant datasheet)
  - There will be no regular section at Packard 058
- HW 1 and Lab 1 grades released
  - Please make a regrade request within 3 days of grade release
  - For Lab 1, if you get points deducted for missing screenshots/videos, send the missing files to us and our grader (urja@stanford.edu) and request for a regrade
  - Make sure to add photos/videos directly to repo for future labs. Do not embed them in README.

Midterm

- Midterm review is this Thursday
- We will post practice problems today
- Midterm is next Thursday!
  - October 29th during class time
- One sheet of notes
- You can bring a calculator, but you probably don’t need it

Today we are learning about…

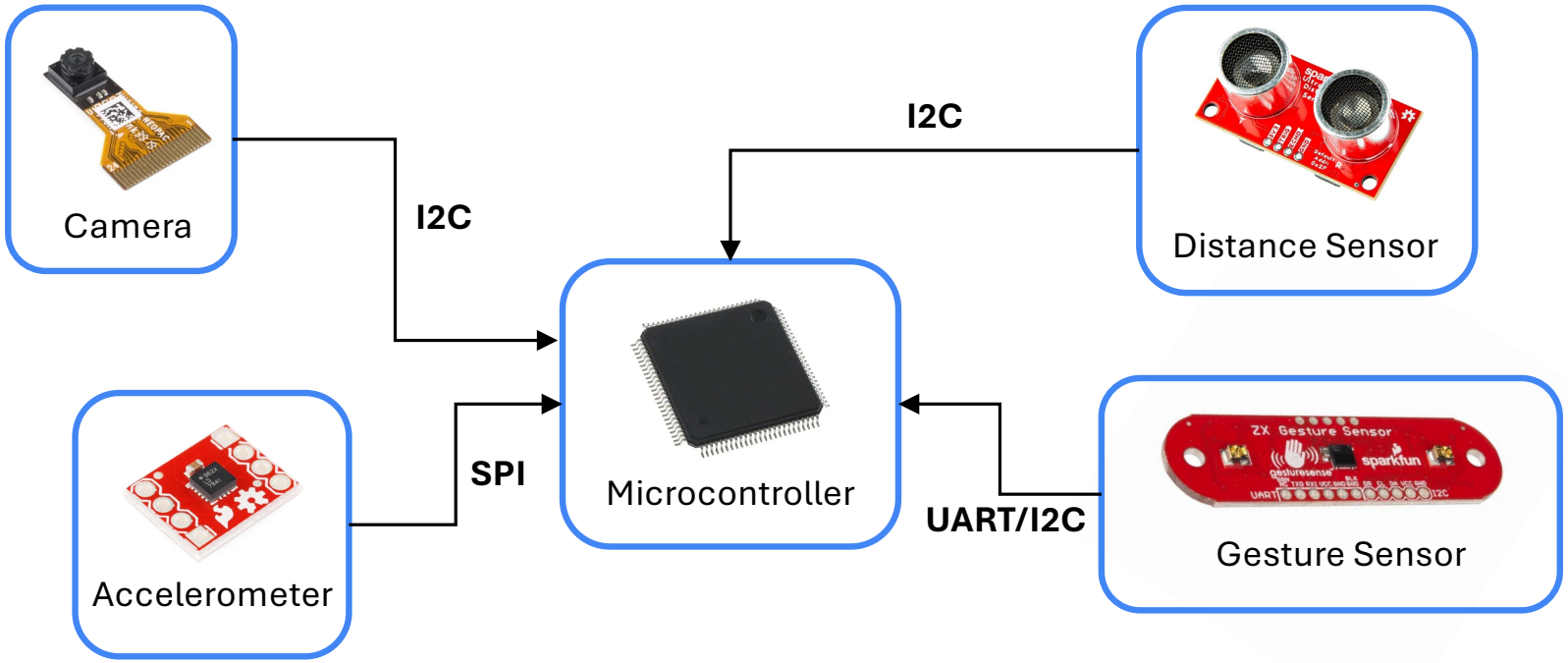
- Communication busses
  - **Serial Bus Architecture Overview**
  - UART – Universal Asynchronous Receiver/Transmitter
  - SPI – Serial Peripheral Interface
  - I2C – Inter-Integrated Circuit

Reading

- **STM RM0432 Reference Manual**
  - Section 50: Universal Synchronous/Asynchronous Receiver Transmitter (USART/UART)
  - Section 51: Low-power Universal Asynchronous Receiver Transmitter (LPUART)
  - Section 49: Inter-integrated Circuit (I2C) Interface
  - Section 52: Serial Peripheral Interface (SPI)
- **Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C (Third Edition)**
  - Chapter 22: Serial Communication Protocols

Communication Buses: Why do we need them?

Allows us to interface with various peripherals and exchange data

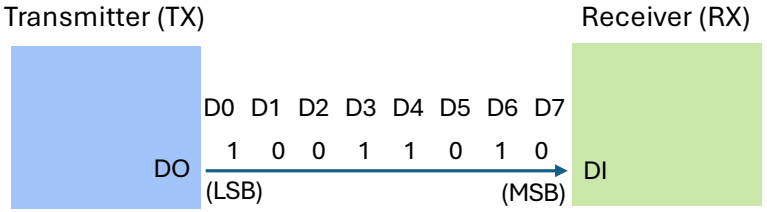


Bus Topologies

Parallel Interface Example



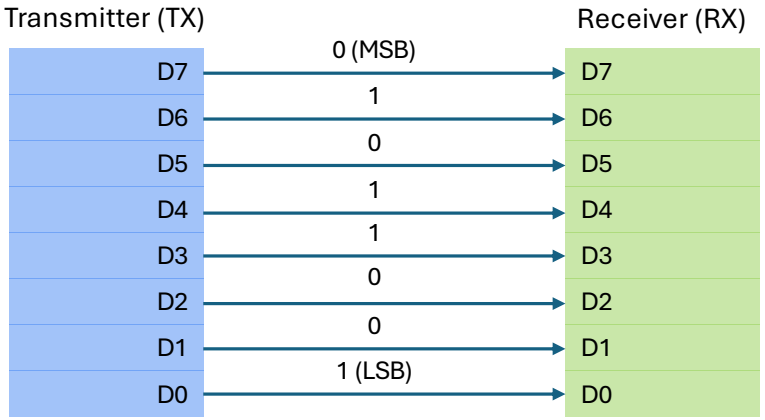
Serial Interface Example

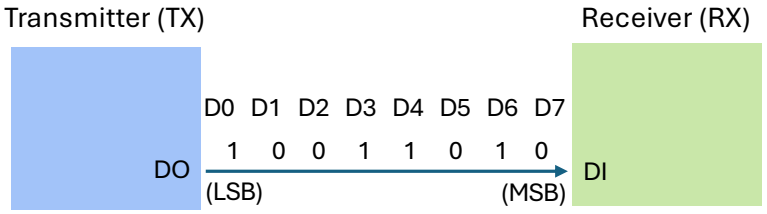


LSB = Least Significant Bit
MSB = Most Significant Bit

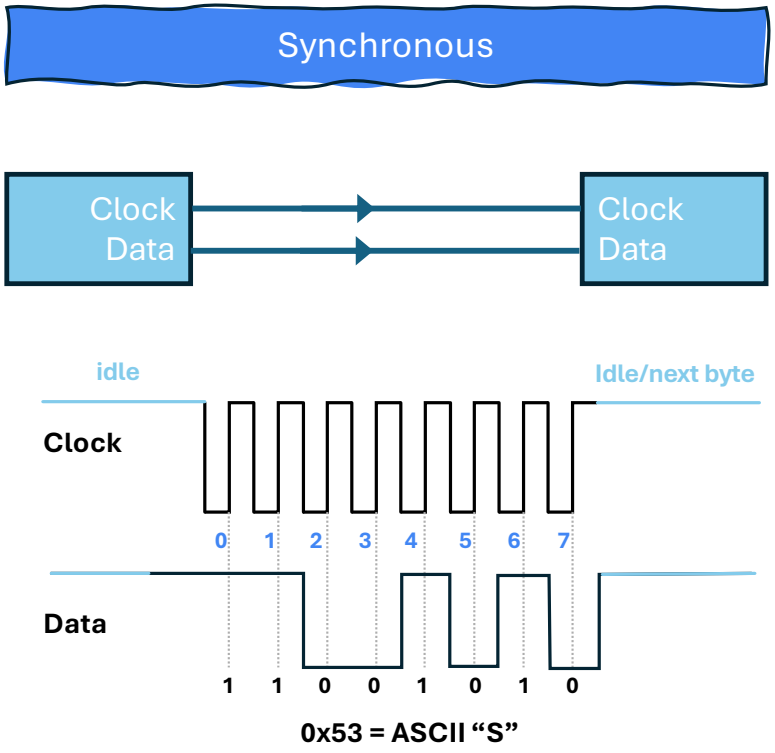
Things to Consider

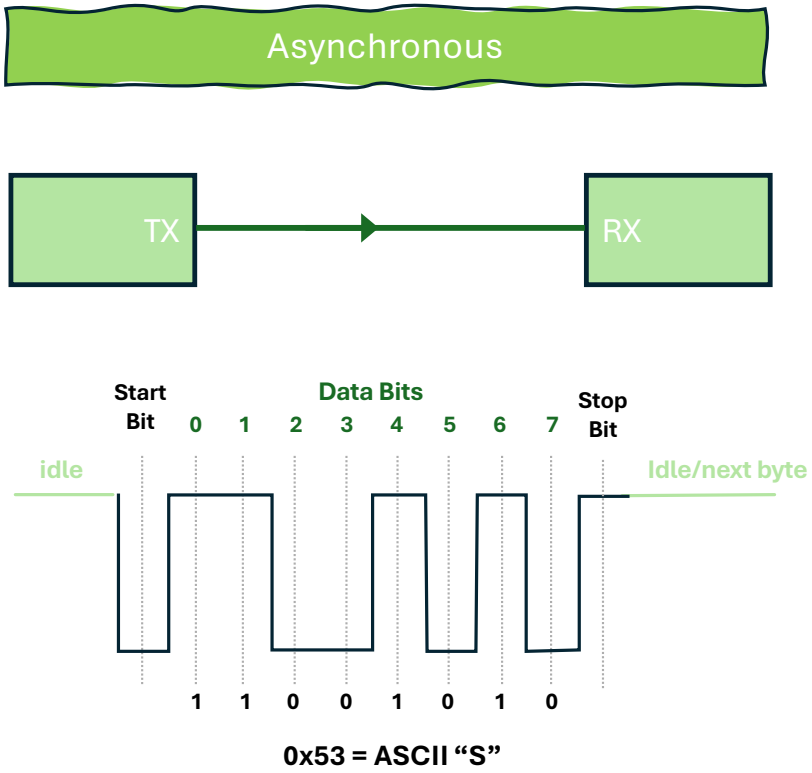
- Encoding
- Data transfer rates
- Cost of pins, wires, modularity
- Error detection/correction





Timing Methodology





Timing Methodology

Synchronous

- Need an extra clock wire
- No synchronization overhead
- Can be high-speed if delays are small and can be controlled
- Common in parallel schemes

Asynchronous

- Less wires (no clock)
- Synchronization Overhead
- Appropriate for loosely-coupled systems
- Common in serial schemes

Arbitration Schemes

Daisy Chaining / Token Passing

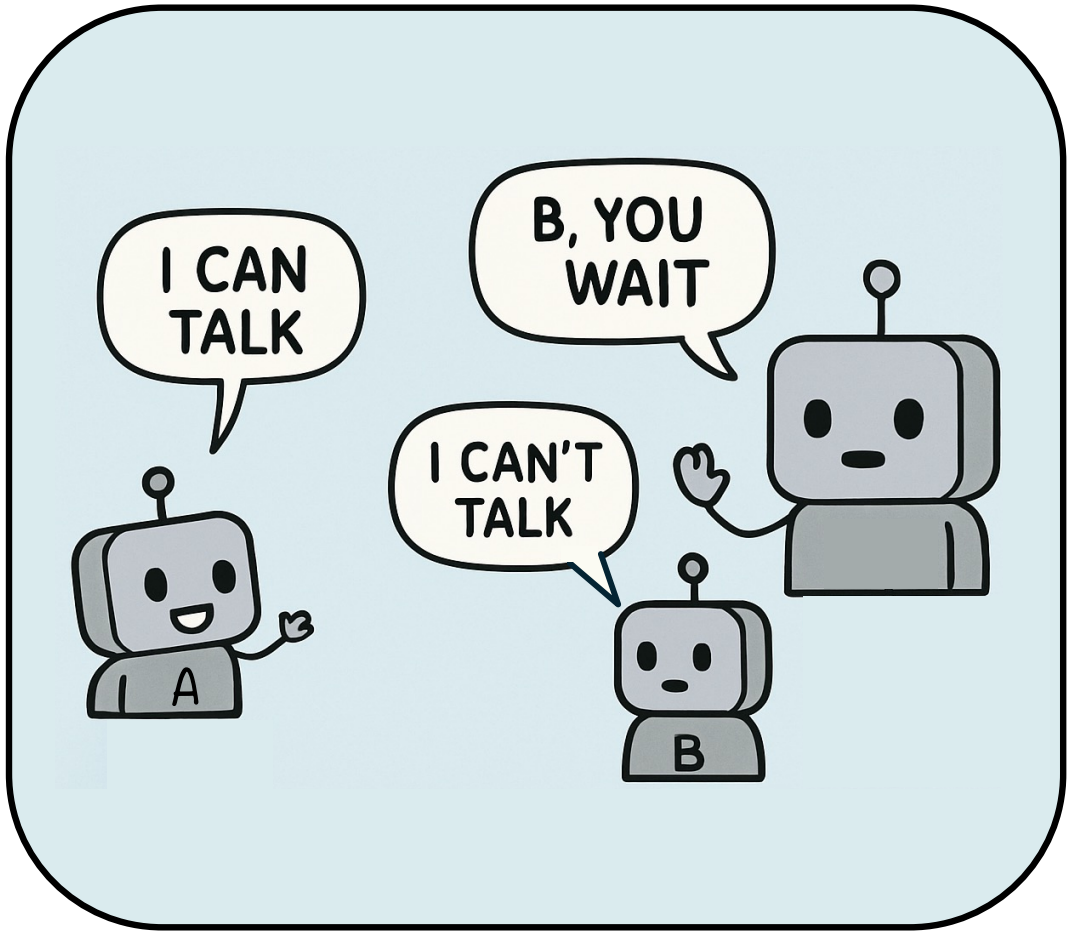
- Devices either act or pass to the next
- Fixed priority order
- As many wires as devices
- Fairness issues

Centralized

- Request to central arbiter
- Center arbiter implements priority scheme
- Wires to/from each device
- Priority/fairness can be changed dynamically

Distributed

- No central arbiter
- Common set of wires
- Fixed priority/fairness



Network Topologies

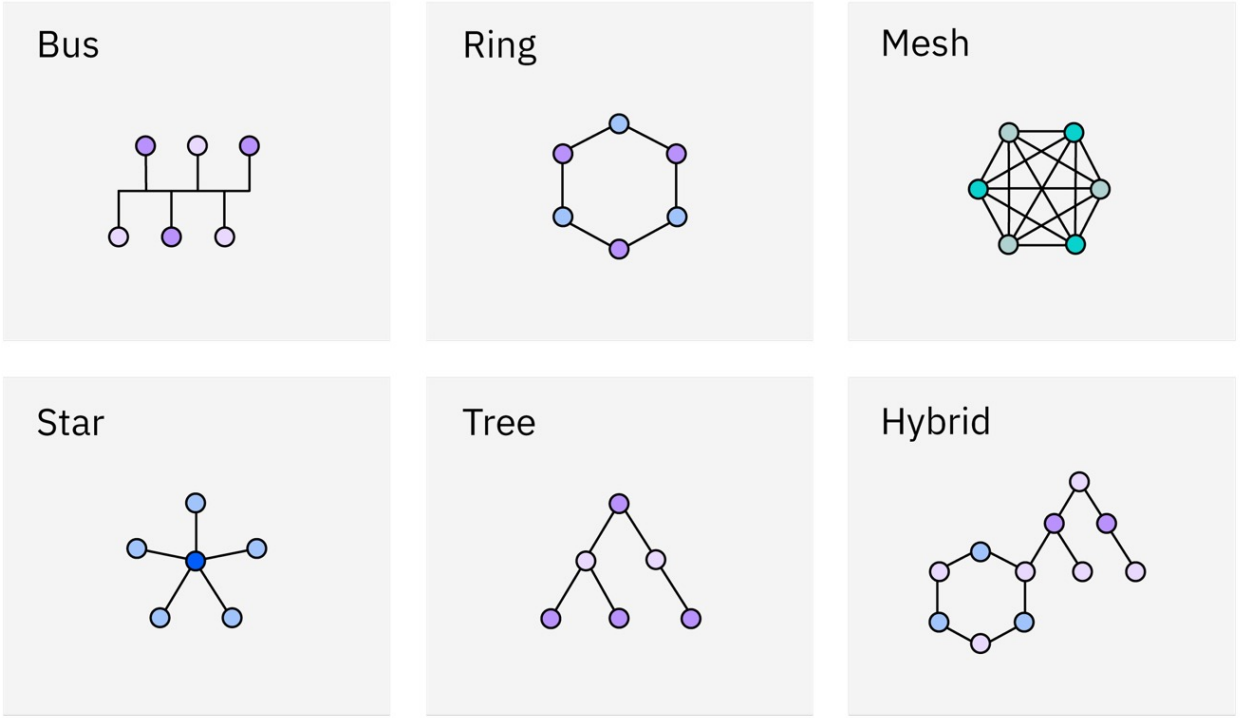


Photo Credit: IBM

Today we are learning about…

- Communication busses
  - Serail Bus Architecture Overview
  - **UART – Universal Asynchronous Receiver/Transmitter**
  - SPI – Serial Peripheral Interface
  - I2C – Inter-Integrated Circuit

Universal Asynchronous Receiver/Transmitter (UART)

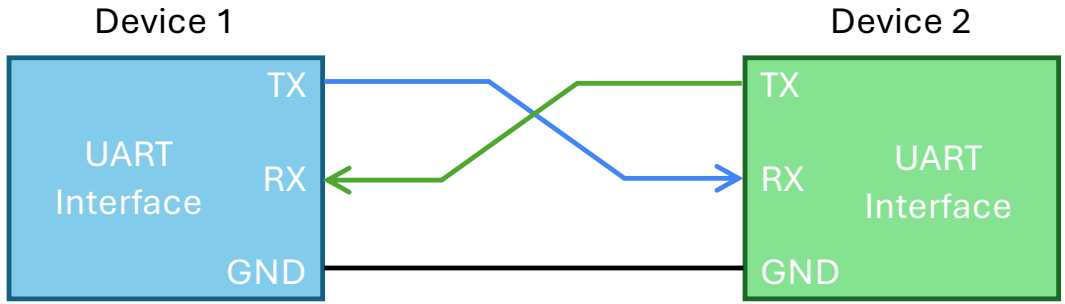
UART

- A type of “asynchronous receiver/transmitter” … computer hardware that **translates data between parallel and serial forms**
- Voltage signals for UART are defined in different standards
  - **RS-232, RS-422, RS-485**
  - RS = Recommended Standard
- Serial transmission, sending one bit at a time, such that data is spread out over time

Used for chip-to-chip communication

RXD signal of a UART is the signal that receives the data (RX = Receive)

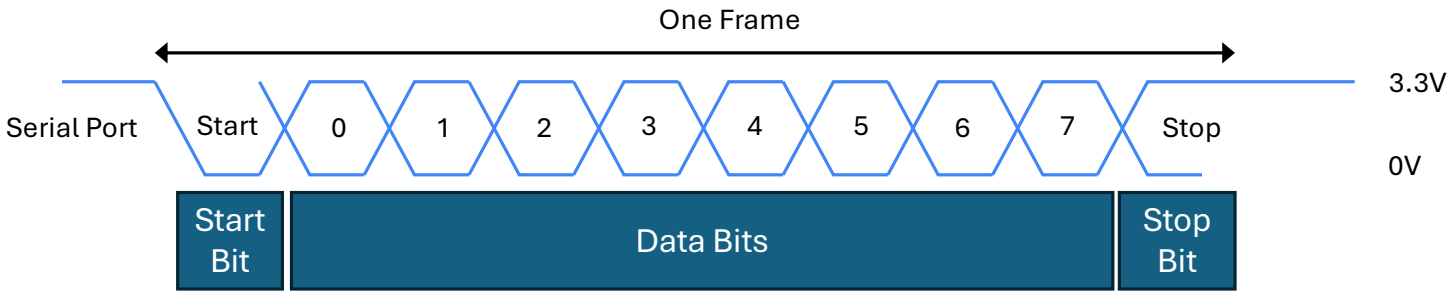
TXD signal of a UART is the signal transmitting the data (TX = Transmit)



Common Mistake !

Connecting TX to TX and RX to RX

Universal Asynchronous Receiver/Transmitter (UART)



The total number of bits transmitted per second is called the **baud rate**.

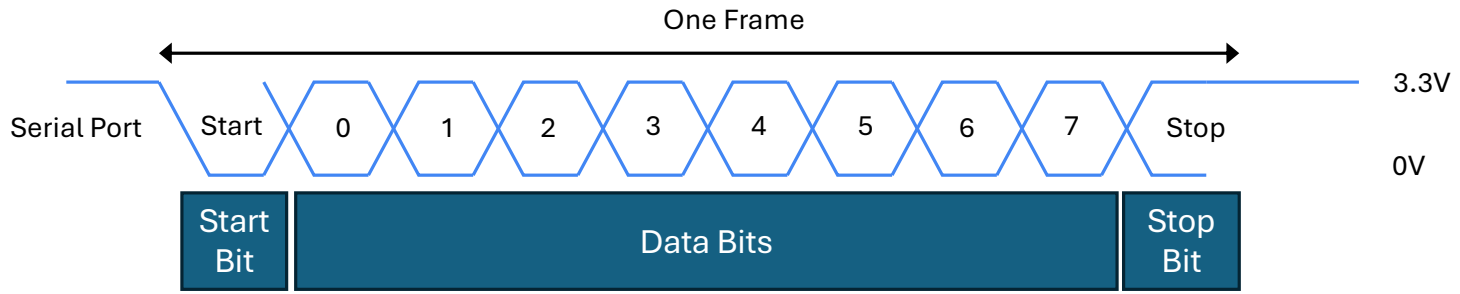
A **frame** is the smallest complete unit of serial transmission.

Start bit (0)
Eight bits of data
Stop bit (1)

The information rate or **bandwidth** is defined as the amount of data or useful information transmitted per second

Data Frame Configuration

- One start bit
- Least significant bit (LSB) or most significant bit (MSB) first
- 5, 6,7, or 8 bits
- Optional parity bit
- One or two stop bits
- Transmitter and receiver must agree on a baud rate



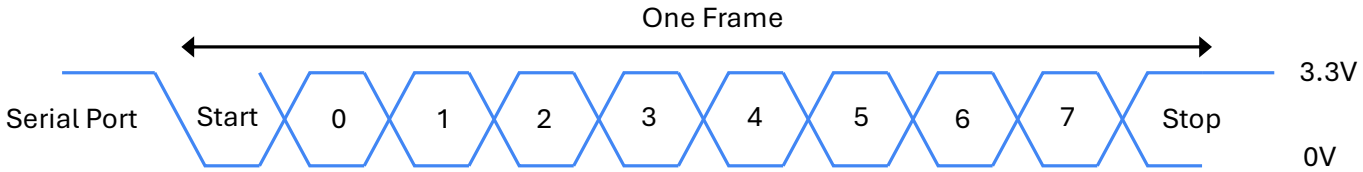
Baud Rate

- The baud rate is the number of bits physically transferred per second
  - Including the actual data content and protocol overhead
  - Results in a bit rate lower than the baud rate

Example: The baud rate is 9600 and the frame consists of a start bit, 8 data bits, and a stop bit

$$\mathbf{9600} \times \left(\frac{\mathbf{8}}{\mathbf{10}}\right) = \mathbf{7860}\ \boldsymbol{bps} = \mathbf{960}\ \boldsymbol{bytes\ per\ second}$$

Group Discussion: When do we latch in the data?



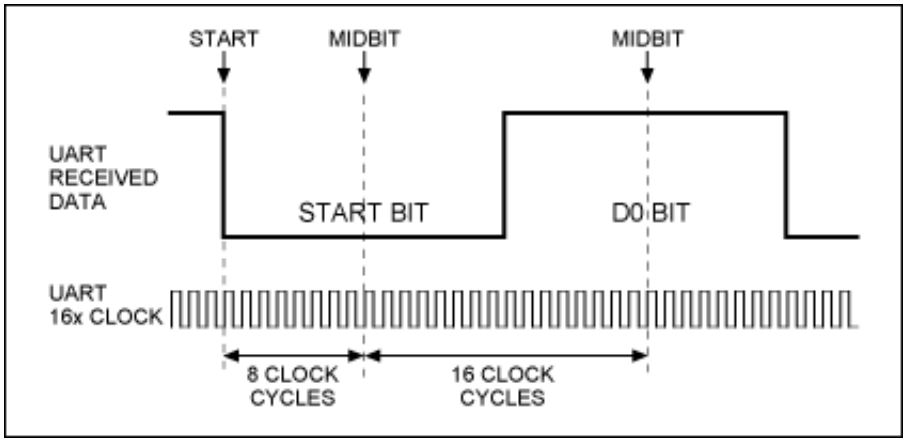
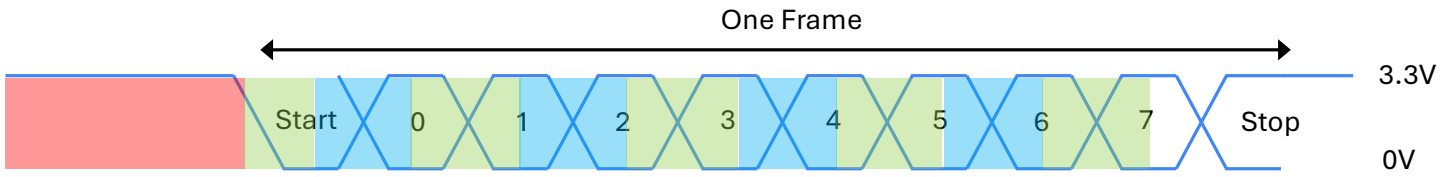


Photo Credit: Analog Devices

There is no clock synchronization!

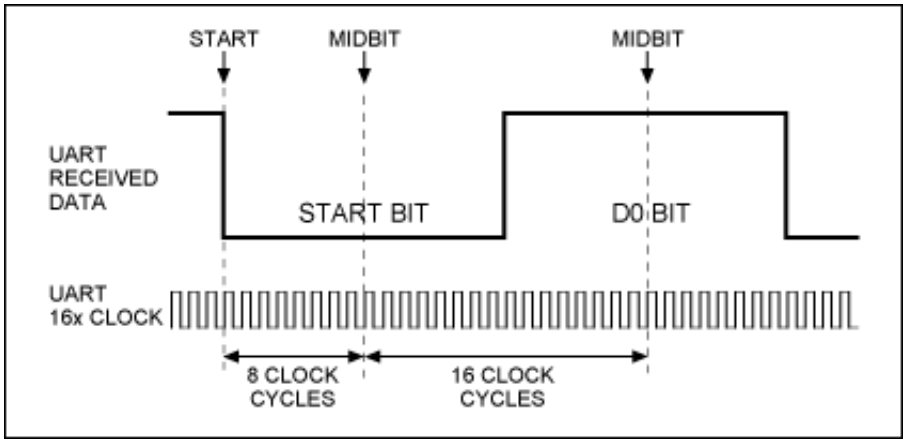


Group Discussion: When do we latch in the data?



Detect the transition from high to low

Start a tick counter and detect the mid bit



Baud Rate Configuration on the STM32L4

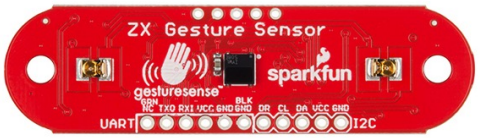
OVER8 = 0 if oversampling by 16
OVER8 = 1 if oversampling by 8

Clock frequency of the processor

$$Baud\ Rate = \frac{(1 + OVER8) \times f_{PCLK}}{USARTDIV}$$

Divider USARTDIV stored in the Baud Rate Register(BRR)

What configuration do I use?



Pin Label	Description
GRN	Not used
TXO	UART transmit out from the ZX Sensor
RXI	UART receive. Not used at this time.
VCC	3.3 - 5 V power supply
GND	Connect to ground
BLK	Not used, but connected to GND
DR	Data Ready. High when there is data to be read via I^2C
CL	I^2C clock
DA	I^2C data

UART Communication Protocol

Physical Interface:

The sensor reports periodically over an asynchronous serial interface (UART) using the following configuration:

Baud-rate =115200 Data bits = 8 Parity = None No Handshaking

Data are reported only when a reflecting object is detected. The data refresh rate is approximately 50 samples per second.

XYZ Interactive Technologies, ZX SparkFun Sensor Datasheet

Introduction

XYZ Interactive Technologies (in conjunction with SparkFun Electronics Inc.) has developed introductory ZX Sensor Models that can be used for rapid proof of concept or for sensor related applications.

These sensors consist of two transmitter(s) and one receiver mounted on one board that constitutes a short-range reflective distance measurement solution. Current design utilizes infrared technology.

The sensor may be preprogrammed to operate as a ranging device or as a gesture interface. The package includes a sensor, a USB to serial board, Mini-B USB cable, and downloadable serial software. The user can demonstrate simple position detection, distance measurement and basic gestures within minutes of opening the package.

This sensor includes a UART for demo applications and I2C as a communication protocol for embedded applications like robotics and toys, or devices that have to communicate directly to a processor.

ZX Sensor

The following describes the ZX sensor. There are two types of the ZX sensor available (as shown). The ZX sensor is available as versions with 1) Through-Hole parts (TH ZX sensor) which allows sensing solutions that have longer range because of lensing on the emitter snd receiver, and 2) SMT parts (SMT ZX sensor) which allows sensing solutions that are more compact and may be hidden under IR transmissible plastic.

The ZX sensor is designed with the following characteristics:

1. UART at 115200 baud (see UART details below)
2. I2C interface (see I2C details below)
3. Continuous data output at a rate of 50 samples/sec
4. Simultaneous UART/I2C data output
5. 3D object detection in depth and in the periphery
6. On-board gesture detection and output (swipes, hand-push, etc.)
7. Optional on-board pull-down resistors for I2C
8. Support for up to two ZX sensors on one I2C bus
9. Optional I2C ID's as 20 (unsoldered plug) and 22 (soldered plug)

Error Detection: Parity Bit

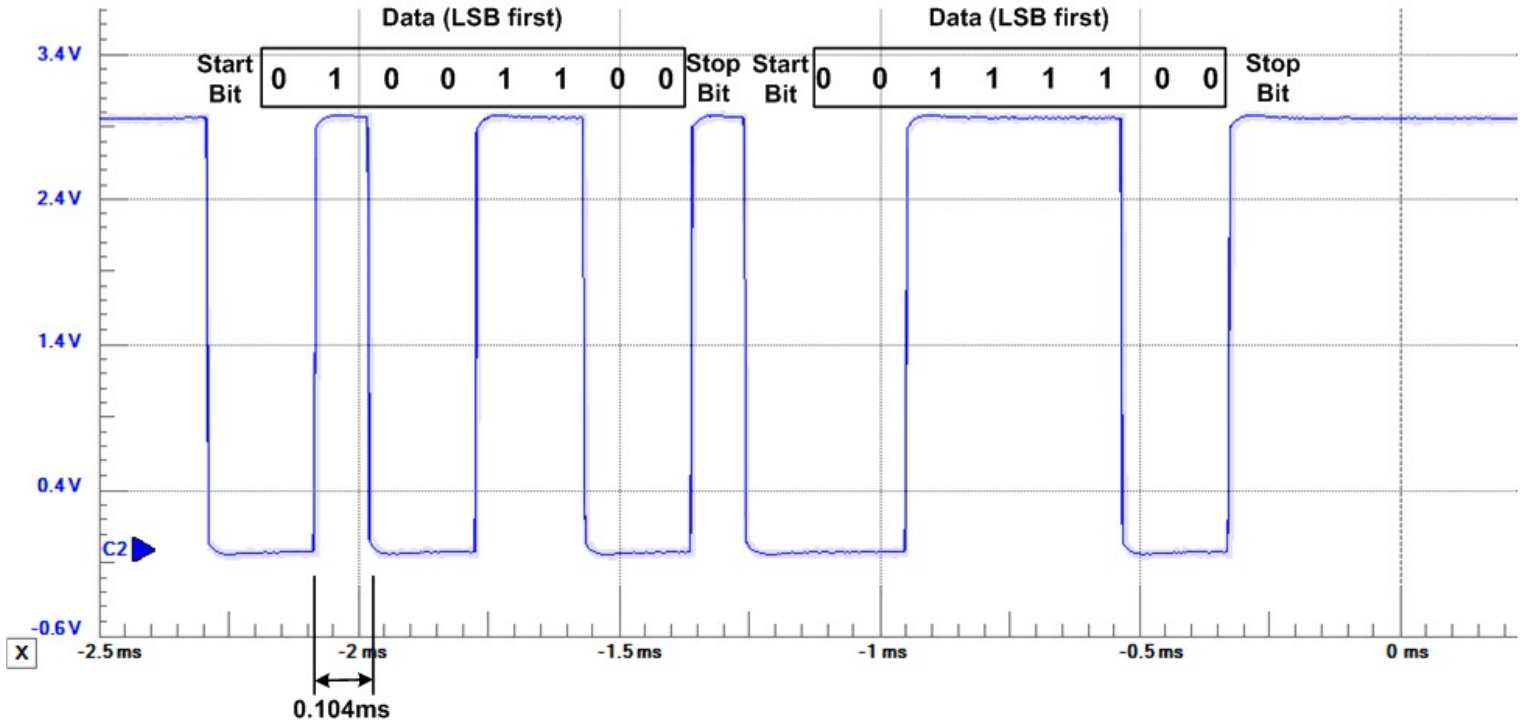
- **Even Parity:** total number of “1” bits is even
- **Odd Parity:** total number of “1 bits is odd
- Allows you to detect single-bit data corruption

If my data = 10110101, then what should my parity bit be?

If using **even parity**, then the **parity bit should be 1** because the total number of 1’s is odd

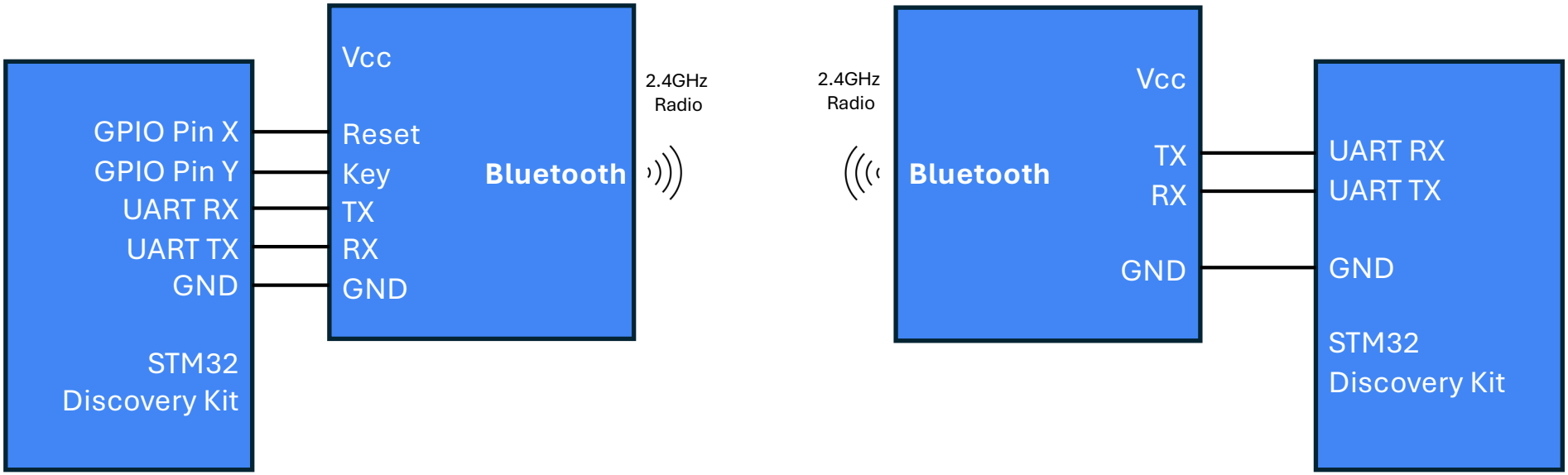
If using **odd parity**, then the **parity bit should be 0** because the total number of 1’s is already odd

Example


Data (LSB first)
Start Bit
0 1 0 0 1 1 0 0
Stop Bit
Data (LSB first)
Start Bit
0 0 1 1 1 1 0 0
Stop Bit
3.4 V
2.4 V
1.4 V
0.4 V
-0.6 V
C2
X
-2.5 ms
-2 ms
-1.5 ms
-1 ms
-0.5 ms
0 ms
0.104ms

When would you use UART?

Example: Interfacing with BLE module



UART on the STM32L4

From the datasheet

Table 12. USART/UART/LPUART features

USART modes/features(1)	USART1	USART2	USART3	UART4	UART5	LPUART1
Hardware flow control for modem	X	X	X	X	X	X
Continuous communication using DMA	X	X	X	X	X	X
Multiprocessor communication	X	X	X	X	X	X
Synchronous mode	X	X	X	-	-	-
Smartcard mode	X	X	X	-	-	-
Single-wire half-duplex communication	X	X	X	X	X	X
IrDA SIR ENDEC block	X	X	X	X	X	-
LIN mode	X	X	X	X	X	-
Dual clock domain	X	X	X	X	X	X
Wakeup from Stop 0 / Stop 1 modes	X	X	X	X	X	X
Wakeup from Stop 2 mode	-	-	-	-	-	X
Receiver timeout interrupt	X	X	X	X	X	-
Modbus communication	X	X	X	X	X	-
Auto baud rate detection	X (4 modes)					-
Driver enable	X	X	X	X	X	X
LPUART/USART data length	7, 8 and 9 bits					

1. X = supported.

UART, USART, LPUART

- **UART:** asynchronous only
- **USART:** can do both asynchronous and synchronous (clocked) data transfers
- **LPUART:** low-power UART

50 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

50.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

51 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

51.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the device is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

On the NUCLEO-L4R5ZI

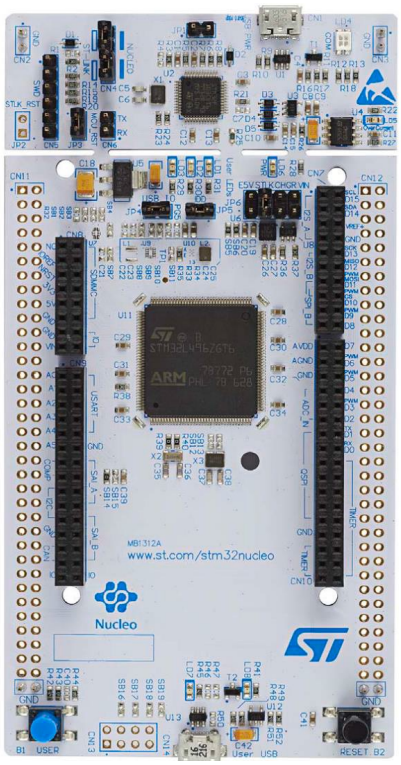
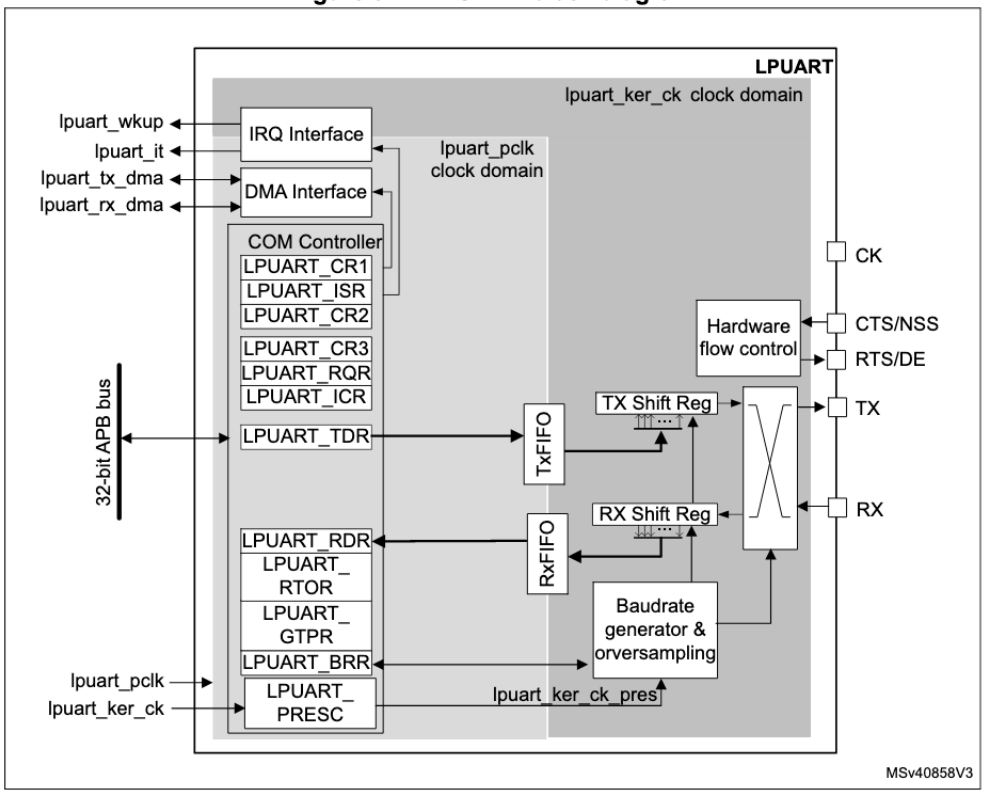
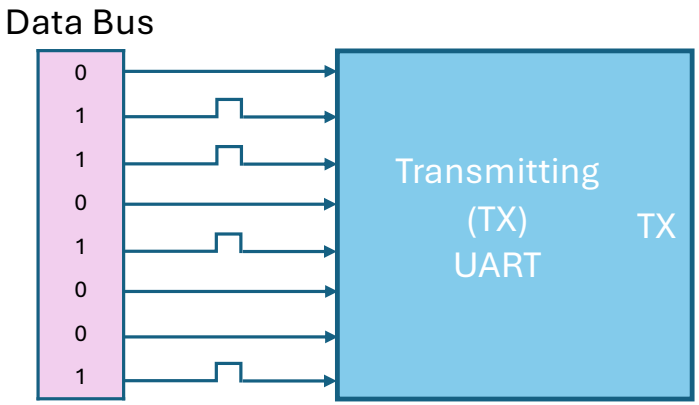


Figure 517. LPUART block diagram

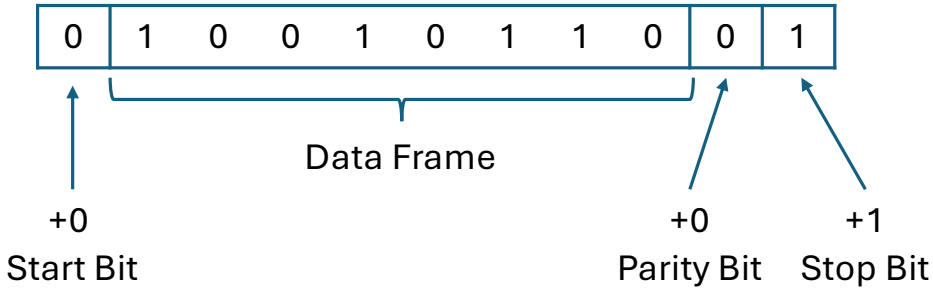


UART Transmission Steps

1. TX UART receives data in parallel from the data bus

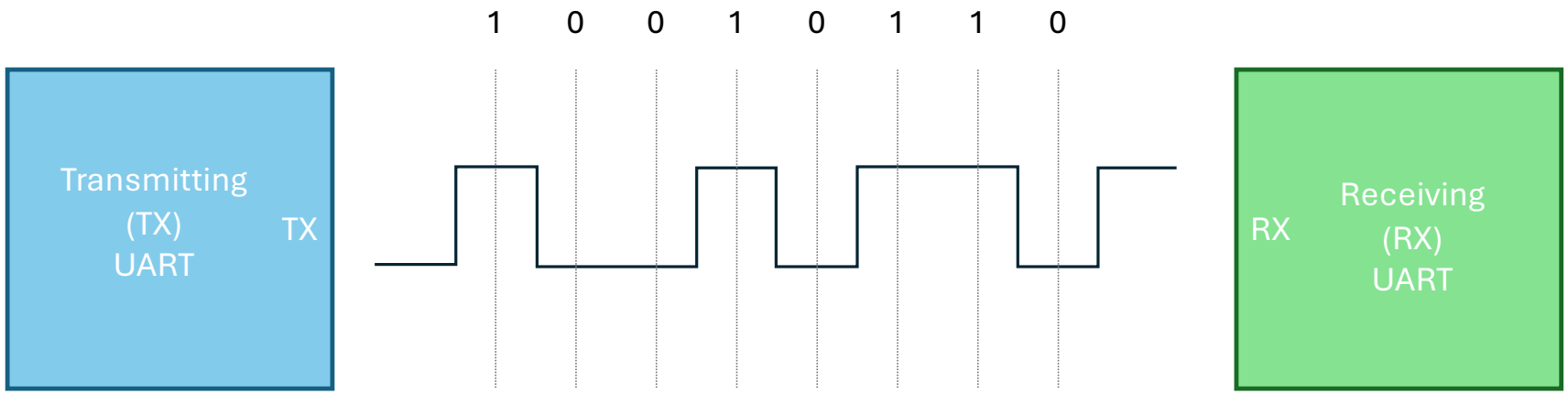


2. TX UART adds start bit, parity bit (optional), and stop bit to the data frame



UART Transmission Steps

3 The entire packet is sent serially from TX to RX UART. The RX UART samples the data line at the agreed upon baud rate



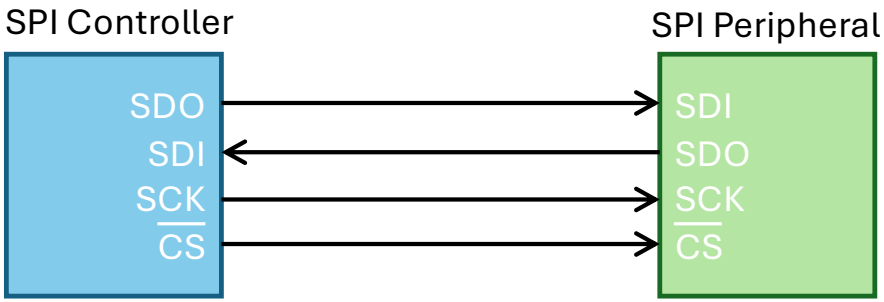
Today we are learning about…

- Communication busses
  - Serail Bus Architecture Overview
  - UART – Universal Asynchronous Receiver/Transmitter
  - **SPI – Serial Peripheral Interface**
  - I2C – Inter-Integrated Circuit

Serial Peripheral Interface (SPI)

SPI Overview

- Common communication protocol
- Single controller with multiple peripherals
- Synchronized
- Full duplex (can transmit and receive data at the same time)
- Multiple Mbps transmission speed



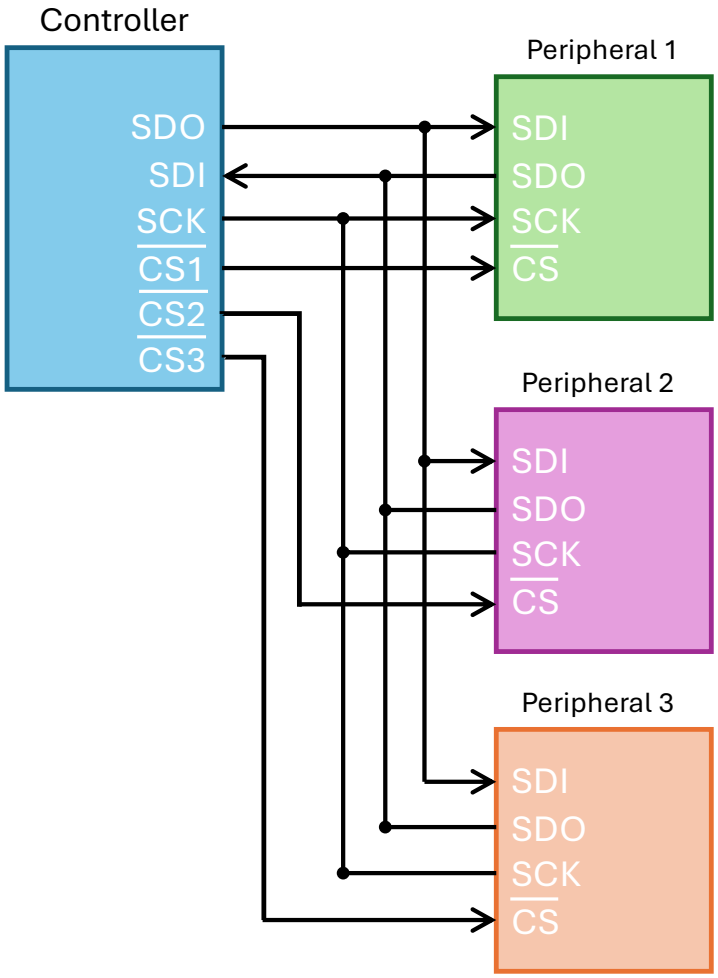
SPI Terminology and Wiring

SDO = Serial Data Out
SDI = Serial Data In
SCK = System Clock
CS = Chip Select

The controller selects a chip with a “low”

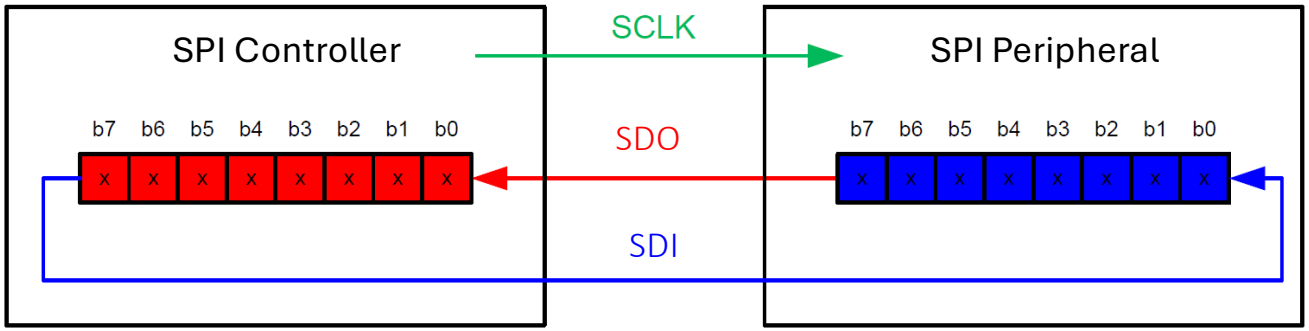
The controller generates a clock

Shift registers, shift data in and out

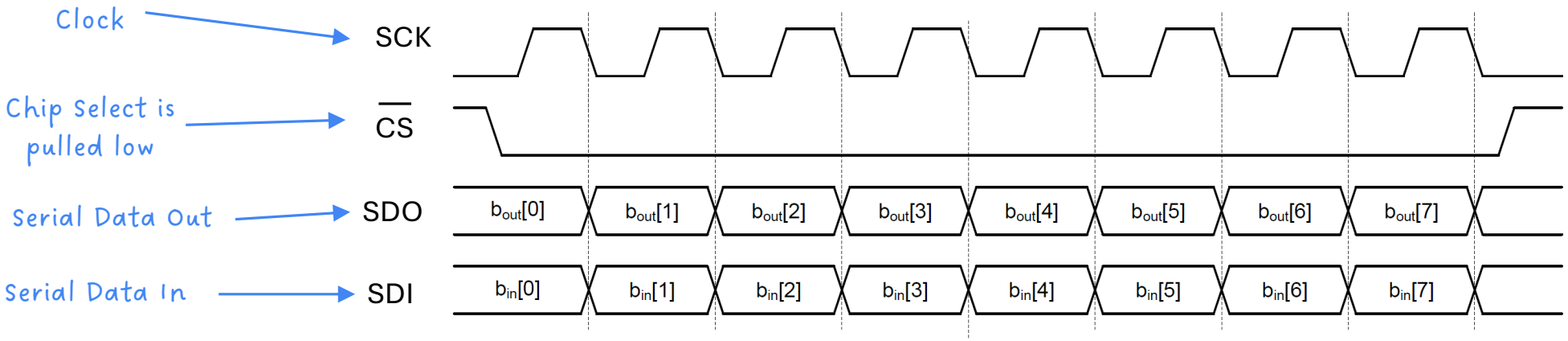


SPI Protocol

- 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
- Packet size 4 to 16 bits, typically 8 bits
- No flow control or acknowledgement
- The peripheral cannot communicate with the controller directly

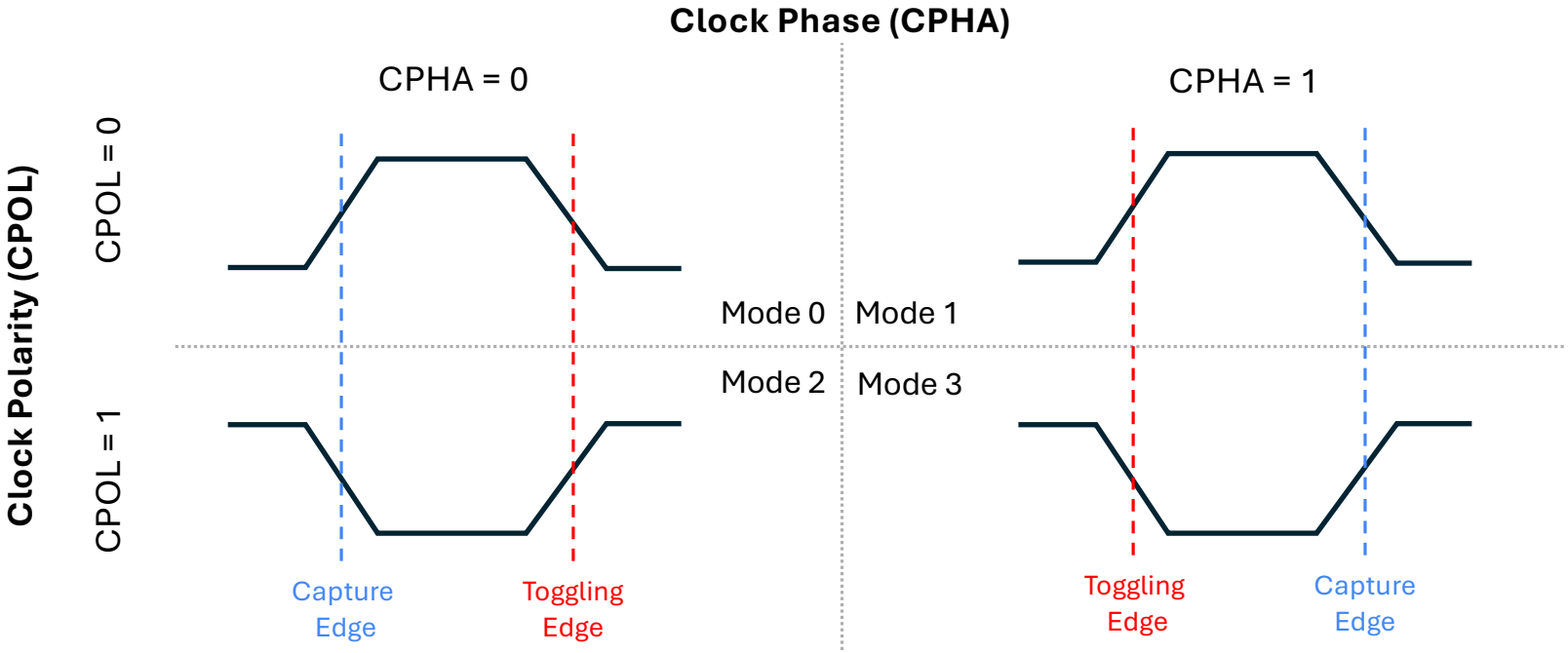


SPI : Timing Diagram



You can sample data at rising or falling clock edge

Clock Phase and Polarity



Clock Polarity (CPOL): The CPOL bit sets the polarity of the clock signal during the idle state.

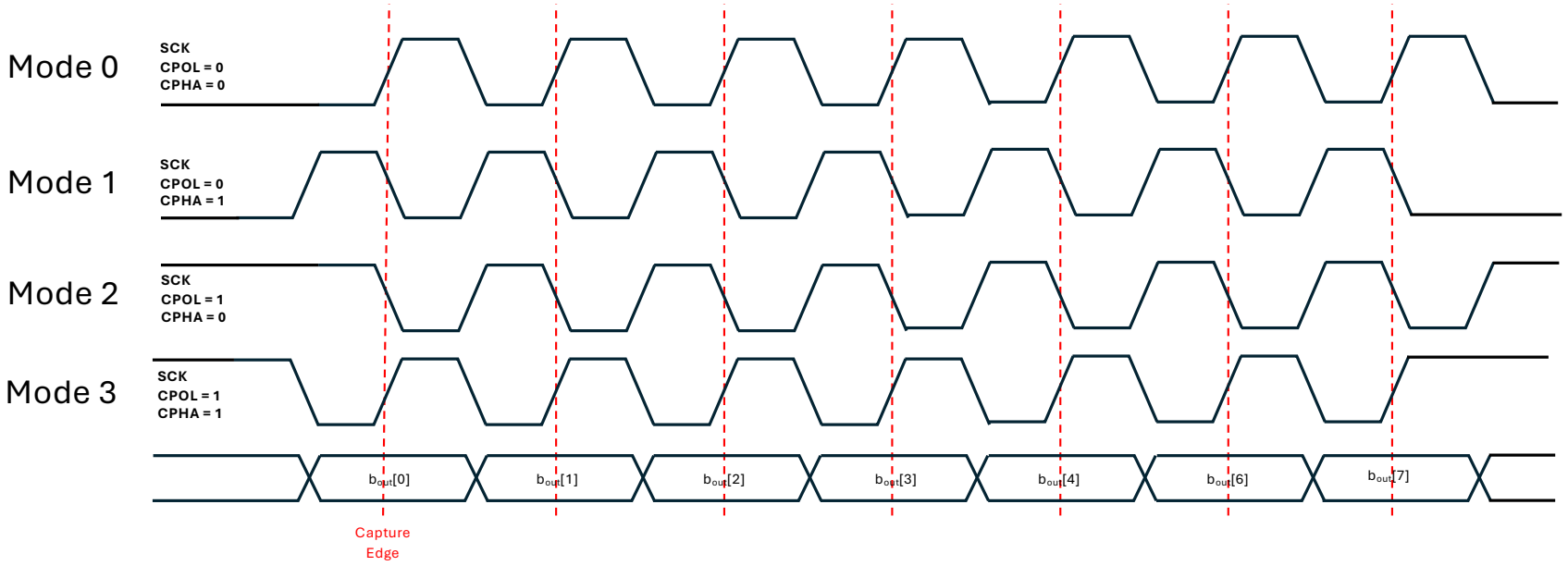
Clock Phase (CPHA): The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data.

Clock Phase and Polarity

Clock Polarity (CPOL)

Clock Phase (CPHA)

Clock (SCK)



Data Rate

- SPI protocol does not define any maximum data rate. Can range from 1 to 20Mbps
- The clock speed determines the data transfer rate
- You must read the datasheet for the peripheral you are interfacing with
- Some “standard” speeds: 700 kbps, 3.4 Mbps,10 Mbps

SPI Pros and Cons

Pros

- Simple wiring
- Faster than asynchronous serial
- No addressing / simple to implement

Cons

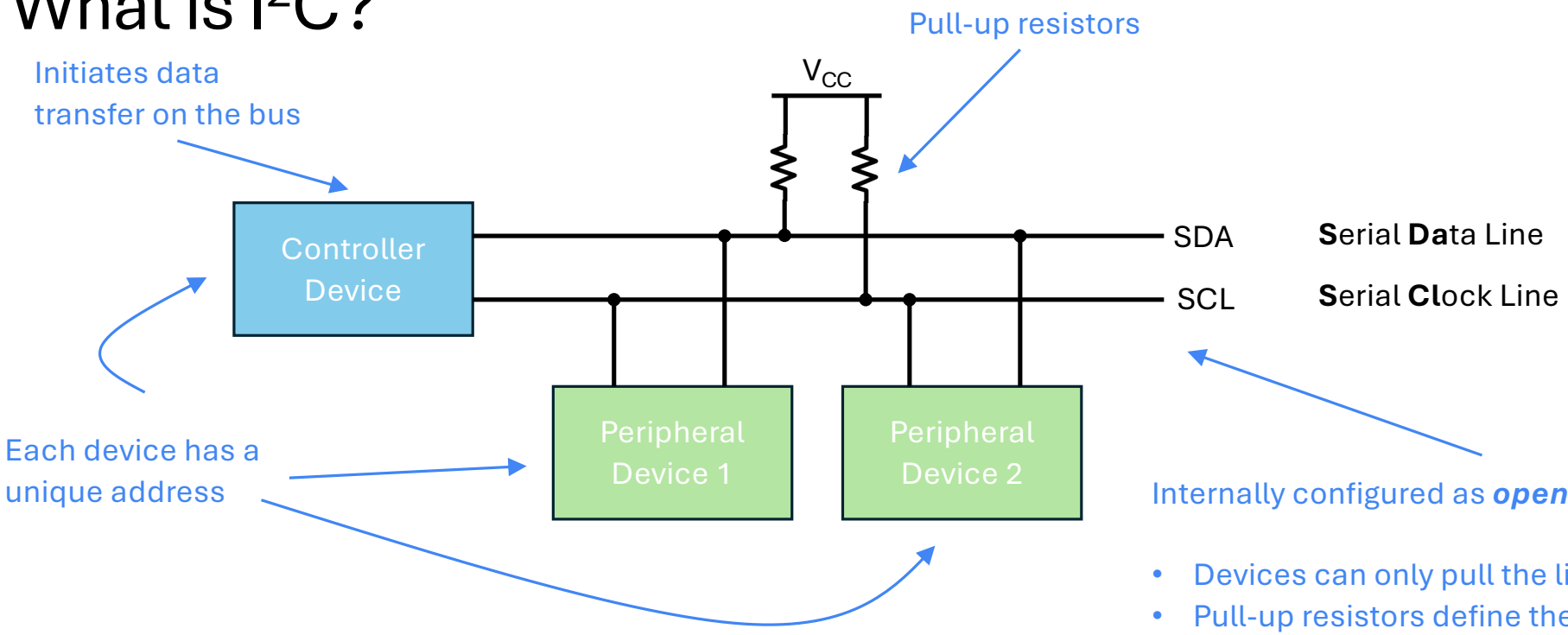
- Select line doesn’t scale
- No acknowledgement
- Control must initiate all communications

Inter-Integrated Circuit (I2C)

What is I^2C?

- I2C = Inter-integrated circuit
  - “eye-squared-see” or ”eye-two-see”
- Originally invented in 1982 by Philips Semiconductor (now NXP Semiconductor)
- **Today:** de-facto standard for 2-wire communication
- Very common in peripherals used for embedded systems

What is I²C?

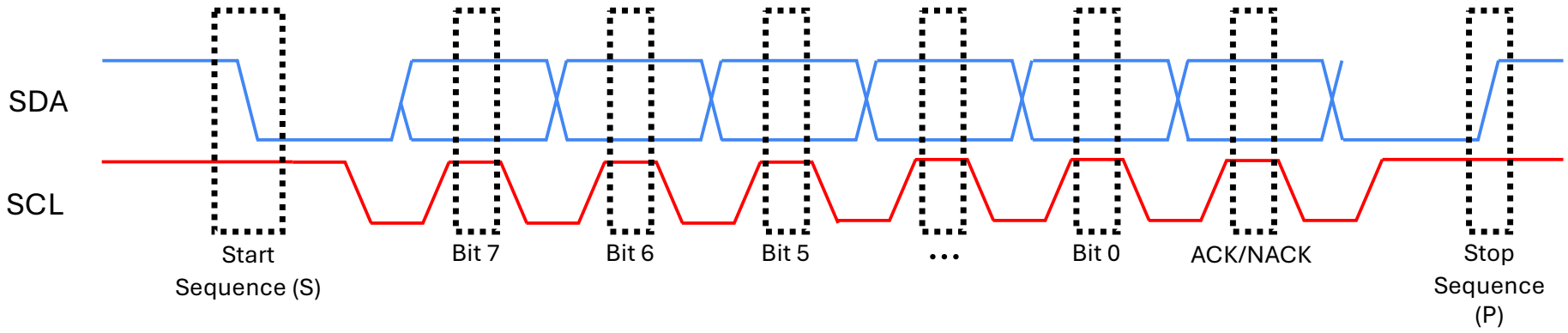


Speed and External Pull-up

- Low Speed Mode: up to 100 kbps, 4.7kΩ
- Standard Mode: up to 400kbps, 3kΩ
- High Speed Mode: up to 3.4Mbps, 1kΩ

Timing Diagram

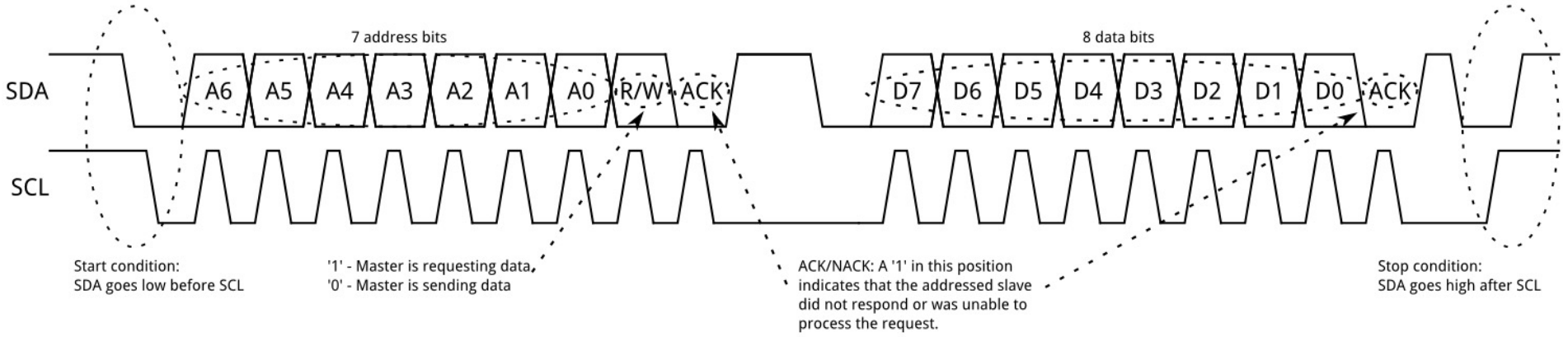
ACK = Acknowledgement
NACK = Not Acknowledgment



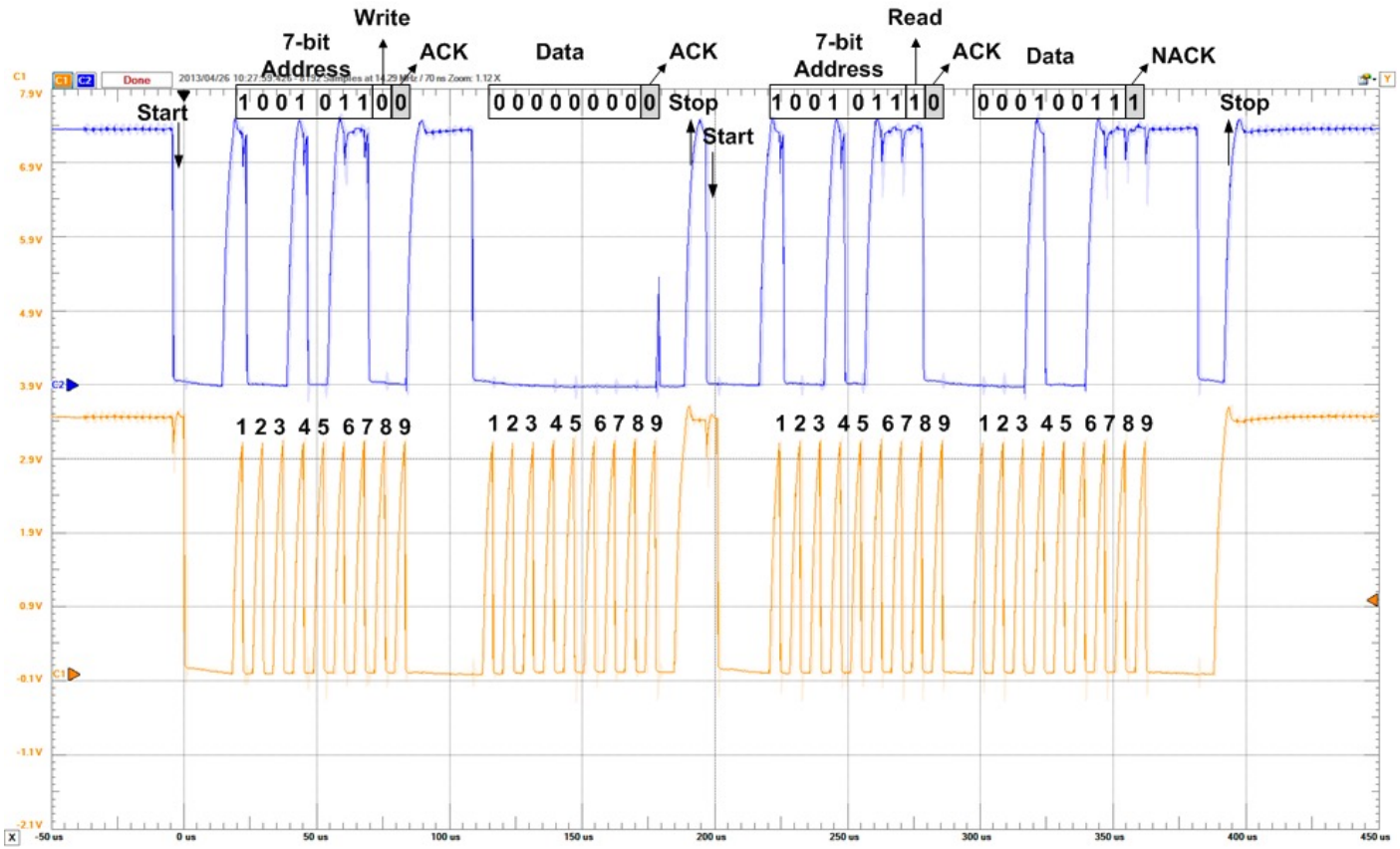
- Communication begins with a start bit (S) and is terminated with a stop bit (P)
- **START:** high-to-low transition of SDA while SCL is high
- **STOP:** low-to-high transition of SDA while SCL is high
- Address and data bytes are sent most significant bit first
- Controller generates the clock signal and sends it to the peripheral during data transfer

Frames and Data

- Arbitration is done by addressing a particular device on the bit
- 7-bit address frame is sent
- Receiver ACKS or NACKS
- Read/Write bit defines if the Data Frame is sent by the controller or the peripheral
- Data frame is 8 bits



I2C Data


Start
7-bit Address
Write
ACK
Data
ACK
Stop
Start
7-bit Address
Read
ACK
Data
NACK
Stop
1 0 0 1 0 1 1 0 0
0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 1 1 0
0 0 0 1 0 0 1 1 1
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

I2C with the STM32L

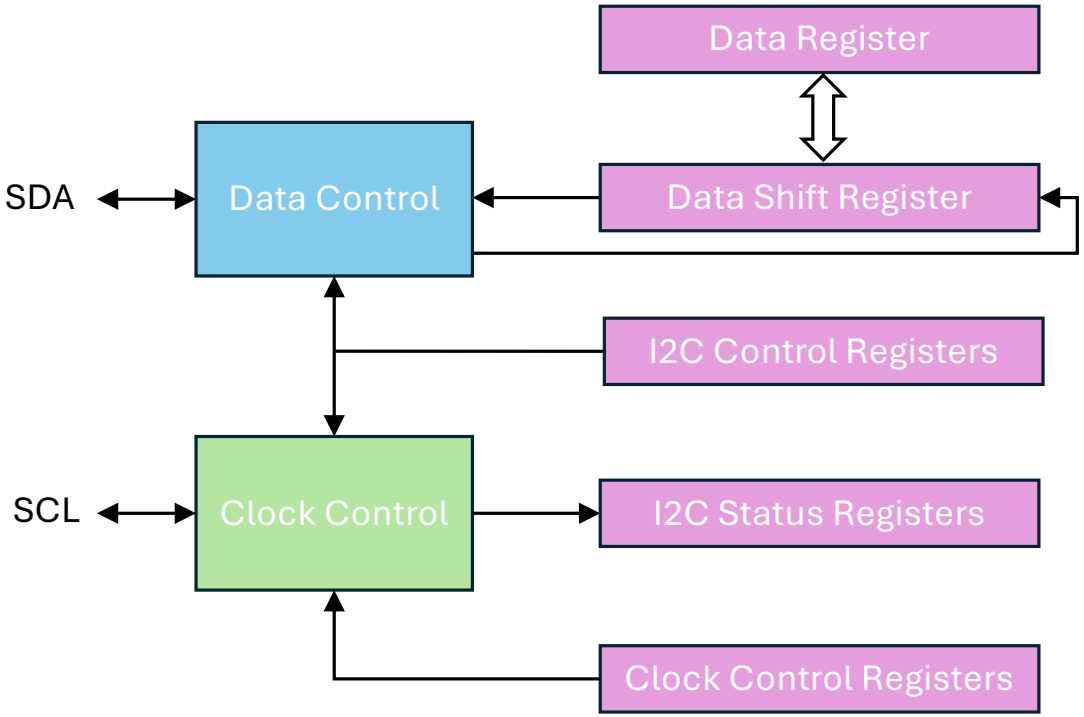
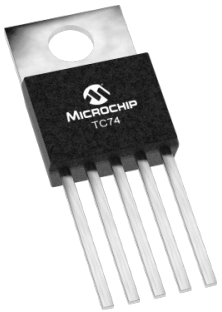


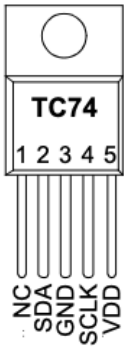
Table 335. I2C implementation

I2C features[1]	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wakeup from Stop 1 mode	X	X	X	X
Wakeup from Stop 2 mode	-	-	X	-
SMBus/PMBus	X	X	X	X

Example: Interface with TC74 Temperature Sensor

- TC74 is a digital thermal sensor from Microchip Technology
- 8-bit temperature measurement





Pin #	Symbol	Type	Description
1	NC	None	No internal connection
2	SDA	Bidirectional	Serial Data Line
3	GND	Ground	System Ground
4	SCLK	Input	Serial Clock Line
5	VDD	Power	Power Supply Input

Example: Interface with TC74 Temperature Sensor

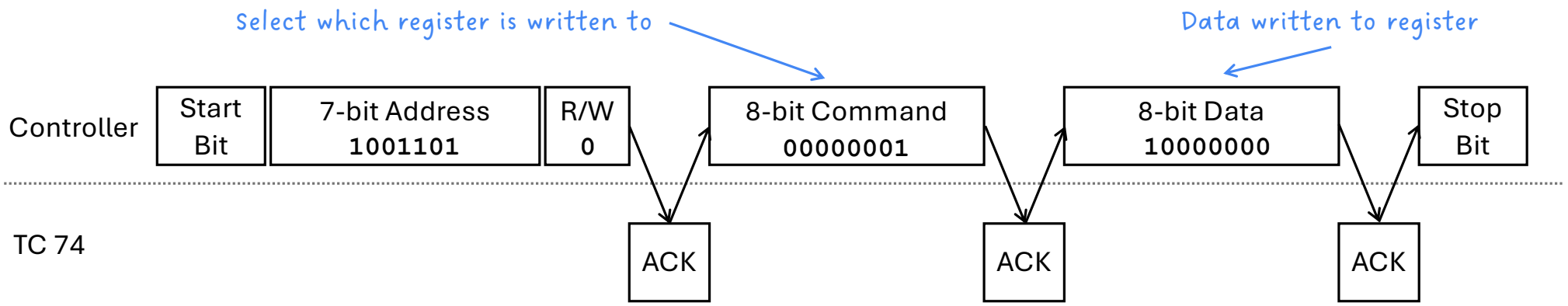
- The TC74 sensor has a programmable 7-bit address
  - Default for the TC74 A5: 1001101

- Two 8-bit registers
  - Temperature register (address 0x00)
  - Configuration register (address 0x01)

S	Address	R/W	Command	ACK	Data	ACK	P
1 bit	7 bits	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

- R/W bit is 0 for writing data to the configuration register
- The command byte selects which register the data is written to
- The data byte specifies the data content to be written to the target register
- The ACK bit is sent by the sensor to the microcontroller to acknowledge the receipt of each byte

Example: Interface with TC74 Temperature Sensor



- R/W bit is 0 for writing data to the configuration register
- The command byte selects which register the data is written to
- The data byte specifies the data content to be written to the target register
- The ACK bit is sent by the sensor to the microcontroller to acknowledge the receipt of each byte

I2C Pros and Cons

Pros

- Easy wire interfacing
- Only two wires (plus GND)
- Widely support

Cons

- Speed/power tradeoff
- Max speed defined by hardware (resistors)
- No hardware arbitration

Acknowledgements

- These lecture slides were adapted from the University of Michigan's EECS 373 course (many thanks to Alanson Sample!)

- These lecture slides also leverage materials from the following books:
  - Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C
    - Yifeng Zhu
  - The Definitive Guide to ARM Cortex – M3 and Cortex – M4 Processors
    - Joseph Yiu
  - Introduction to ARM Cortex – M Microcontrollers, Embedded Systems
    - Jonathan W. Valvano