# Microprocessors & Microcontrollers
## : Arm Cortex M0+
## (Using RP2040)

# ESM_21

## Introduction to I2C

Mouli Sankaran

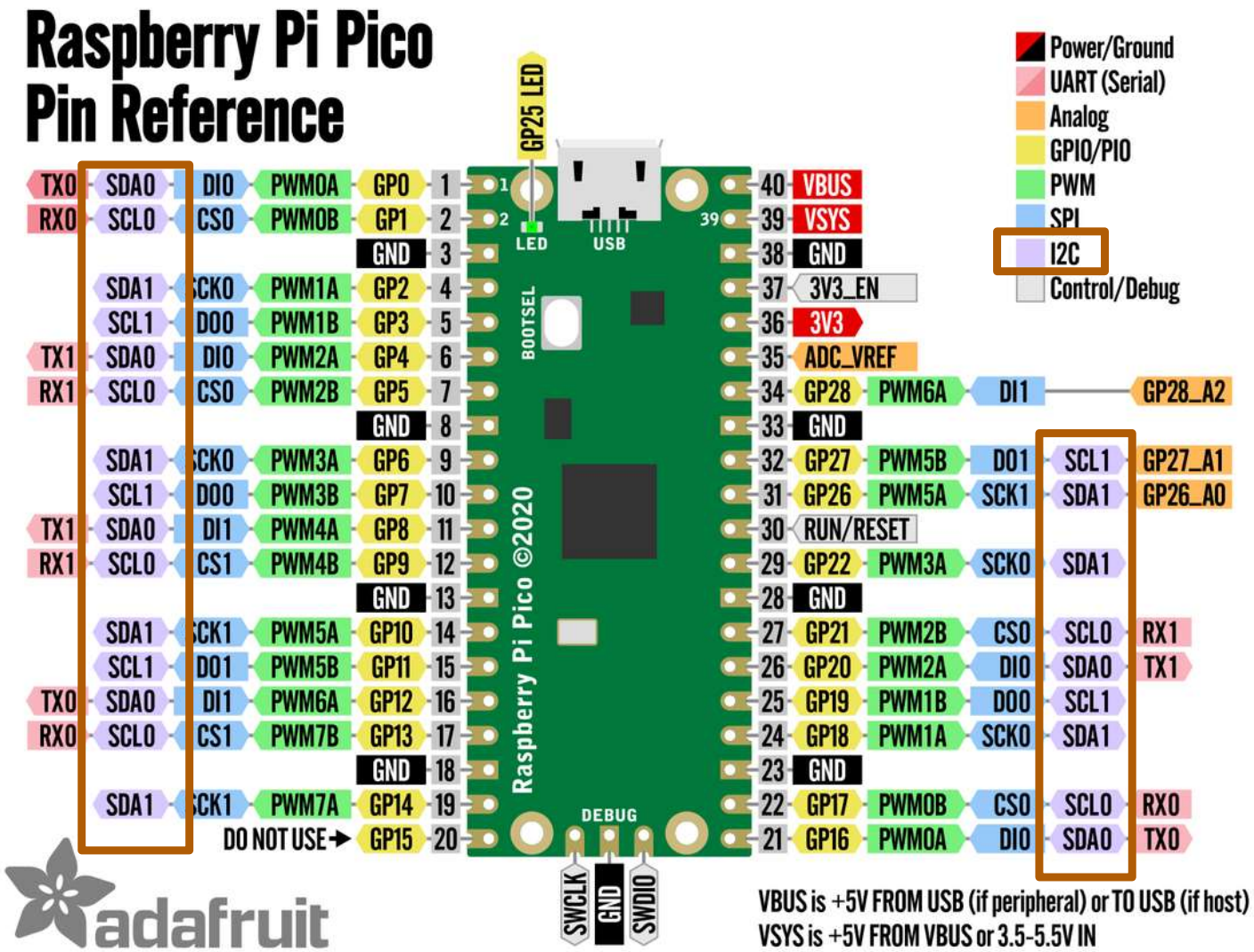
# Lecture 3.4.1 Focus

- I2C Introduction
- I2C Basics
  - Configuration
  - Specification
  - Start and Stop conditions
  - Working Principle
- I2C in RP2040

**$I^2C$: I2C: Inter-Integrated Circuit**

# I2C in RP2040

# RP2040 Pin Out:

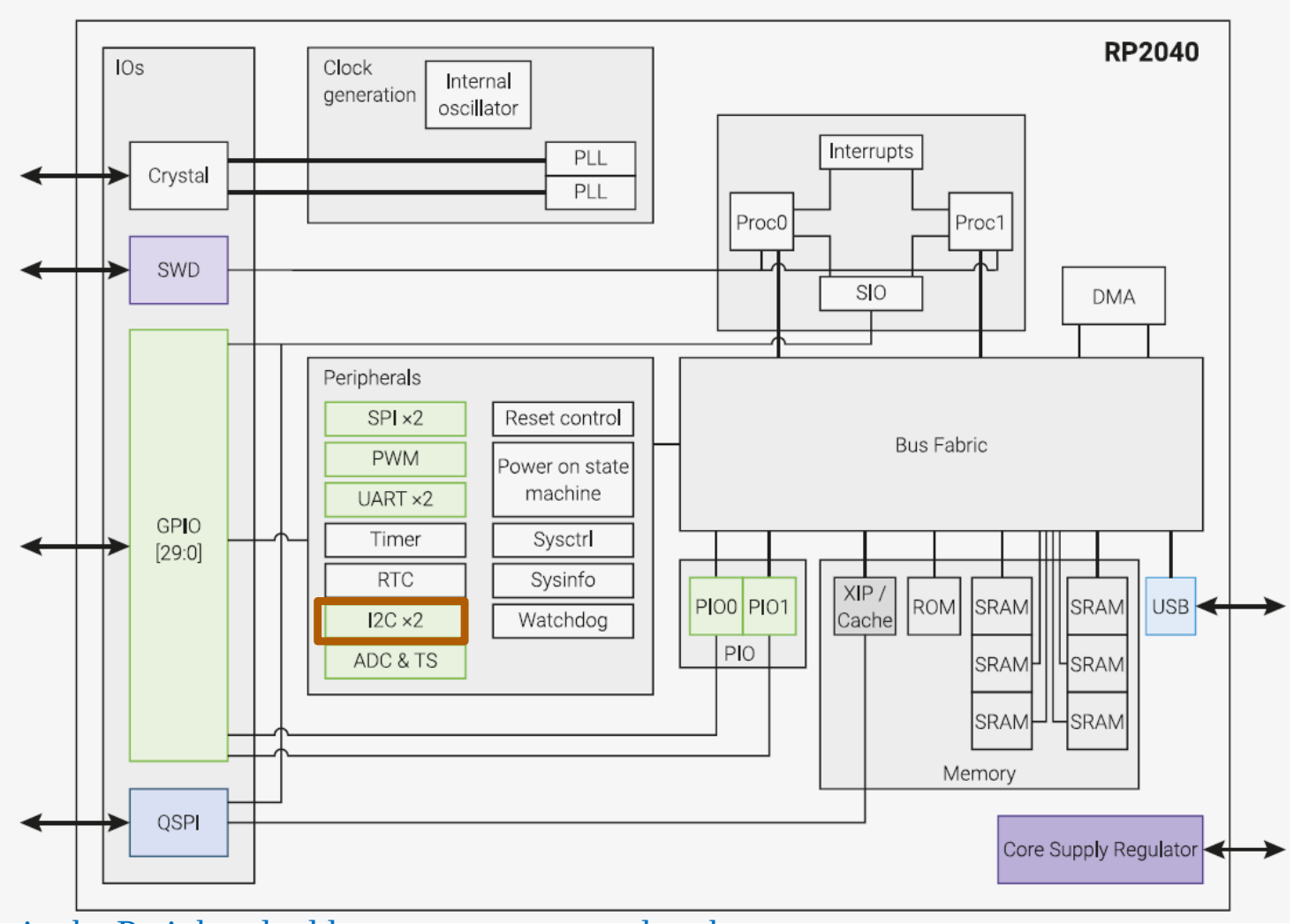



There are two I2C devices here

I2C0 and I2C1

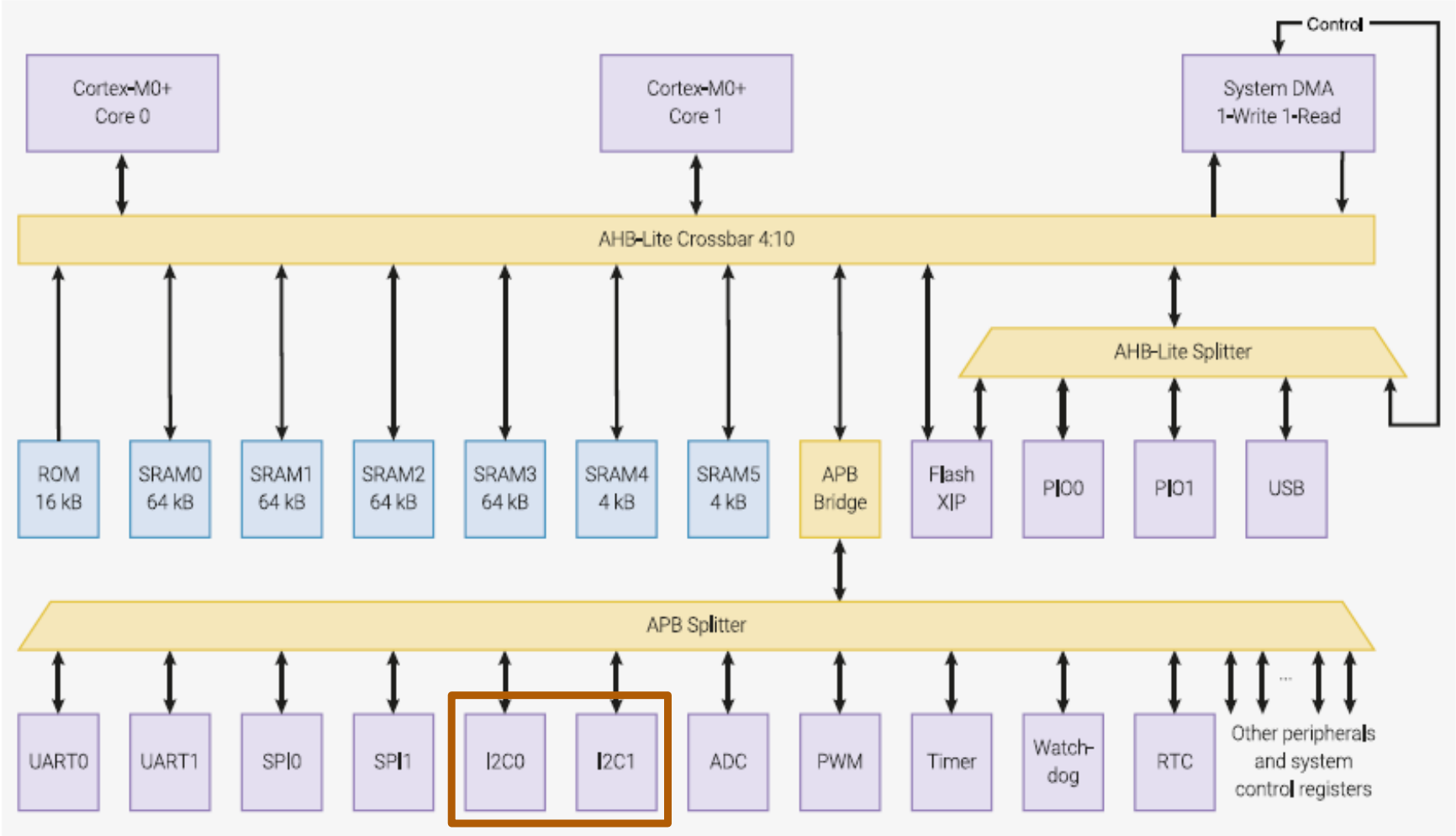
There are many pins that can be mapped to I2C devices to enable flexible interfacing of other ICs in the board.

# I2C in RP2040 System Block




This is in the Peripheral address space connected to the GPIO bank to be able to connect it to the outside world

# I2C Interface with the APB Bus (low speed)




**Ref 5**: RP2040 Data sheet

**Page**: 460/647, Section 4.3

# RP2040: Address Mapping of Peripherals and I2C

| | |
|---|---|
| ROM | 0x00000000 |
| XIP | 0x10000000 |
| SRAM | 0x20000000 |
| APB Peripherals | 0x40000000 |
| AHB-Lite Peripherals | 0x50000000 |
| IOPORT Registers | 0xd0000000 |
| Cortex-M0+ internal registers | 0xe0000000 |

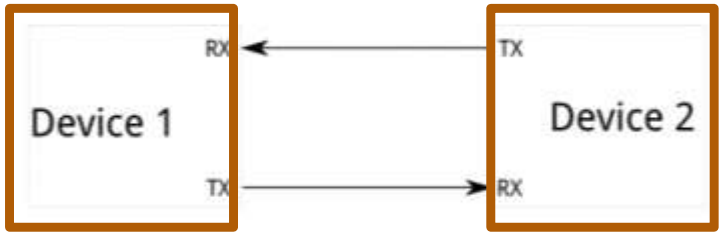
APB Peripherals:

| | |
|---|---|
| I2C0_BASE | 0x40044000 |
| I2C1_BASE | 0x40048000 |

# I2C Introduction

# Problems with Serial Port

- Because serial ports are asynchronous (no clock is transmitted), devices using them must agree ahead of time on a data rate.




- The two devices must also use the same data rate (bauds), if they are different, garbled data reception results.
- Need for an UART chip on both ends of the channel, is an overhead.
- For every 8 bits of data minimum two additional (start and stop) bits are required for every 8 bits data, eating up the data rate of the channel.
- Other major drawback is that only two devices can talk to each other, when there are requirements to connect multiple devices, UART is not suitable.
- Finally, data rate is an issue. While there is no theoretical limit to asynchronous serial communications, most UART devices are built to only support a certain set of fixed baud rates, to take care of inter-operability between devices from different vendors.
  - And the highest of these is usually around **2,30,400 bits per second (bauds).**

**UART**: Universal Asynchronous Transmitter Receiver

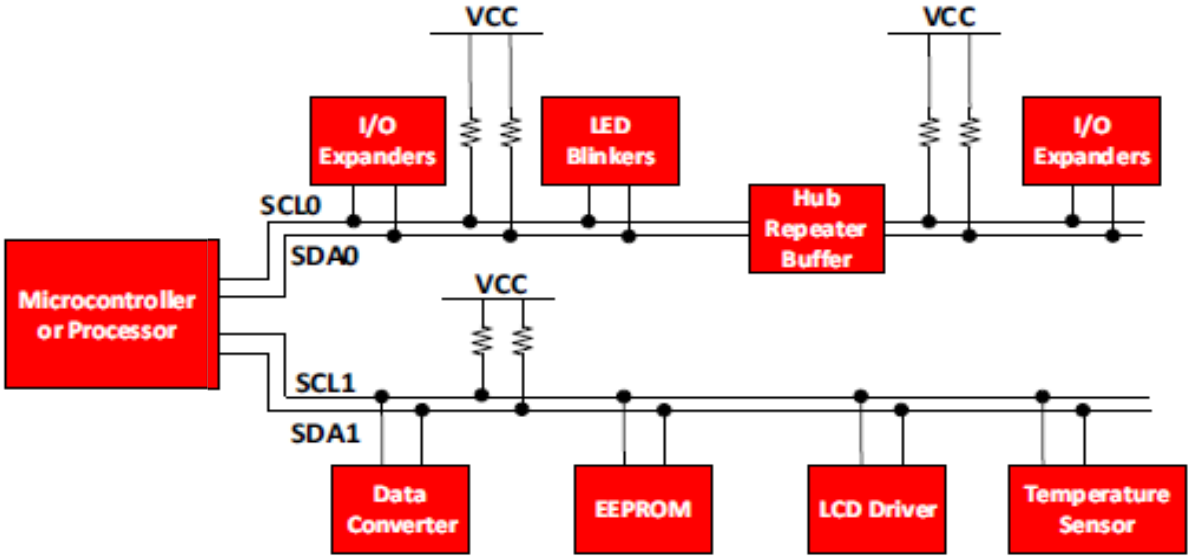
# I2C : Introduction

- $I^2C$ (Inter-Integrated Circuit, eye-squared-C), alternatively known as $I^2C$ or IIC, is a:
  - synchronous,
  - multi-controller/multi-target,
  - packet switched,
  - single-ended,
  - serial communication bus
- It was invented in 1982 by Philips Semiconductors.
- It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication

# I2C Basics

# I2C Bus: How is it used?

(One typical usage scenario)

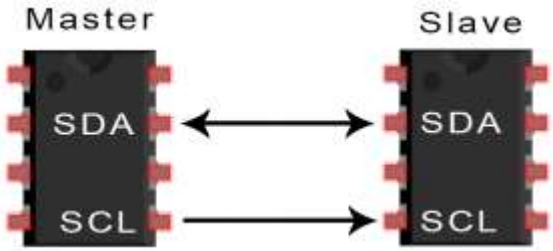



**Note: This is to connect multiple sensors (slave devices) to a control processor (master device)**

Mainly used for communication between Microcontrollers and other devices like, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter, etc.

# I2C : Basics

- I2C combines the best features of SPI (Serial Peripheral Interface) and UARTs.
- With $I^2C$, we can connect multiple slave devices to a single master (like SPI) and we can have multiple masters controlling/exchanging information with single, or multiple slave devices on a bus.



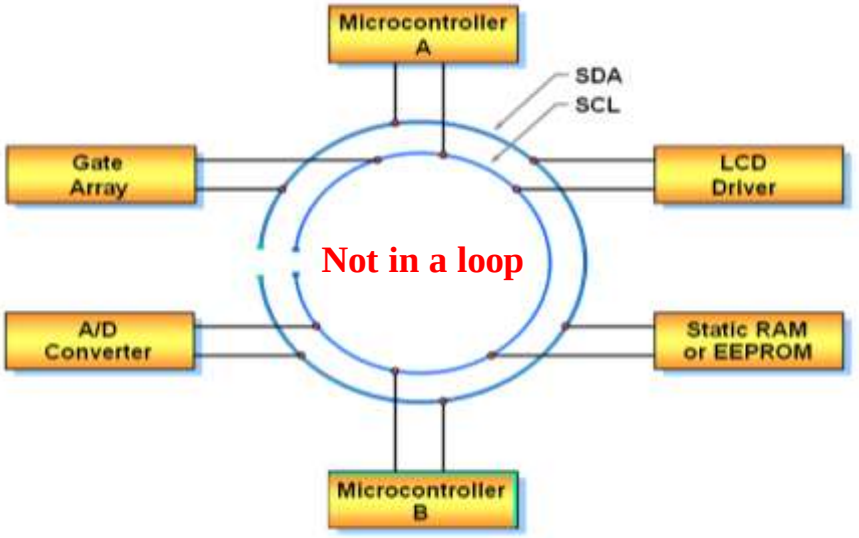

**SDA**: Serial Data
**SCL**: Serial Clock

- This is really useful when we want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.
- I2C uses only two wires to exchange data between devices. No need for GND line.
- **SDA** is for the devices on the bus to exchange data between them.
- **SCL** is line that carries the clock signal to all the connected devices to have a synchronous exchange of data.

**Note**: The $I^2C$ literature uses Master/Slave nomenclature to represent the device which is in control of the transmission on the shared bus, at any point in time.
The SCL (clock) is generated by the device which is configured to be a master.

# I2C Bus Definitions and A Sample Configuration

- **Master:**
  - Initiates a transfer by generating start and stop conditions
  - Generates the clock
  - Transmits the slave address
  - Determines data transfer direction
- **Slave:**
  - Responds only when addressed
  - Timing is controlled by the clock line



# I2C : Specification

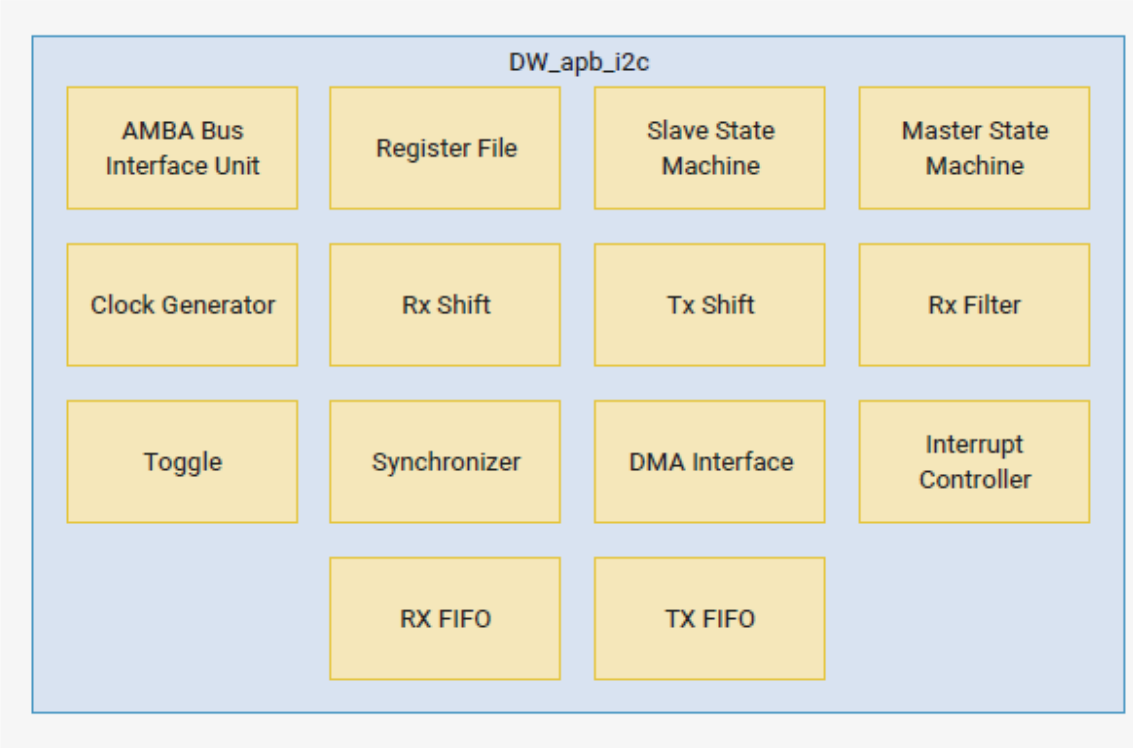
**Ref**: On I2C

| | |
|---|---|
| **Wires Used** | 2 |
| **Maximum Speed** | Standard mode= 100 kbps |
| | Fast mode= 400 kbps |
| | High speed mode= 3.4 Mbps |
| | Ultra fast mode= 5 Mbps |
| **Synchronous or Asynchronous?** | Synchronous |
| **Serial or Parallel?** | Serial |
| **Max # of Masters** | Unlimited |
| **Max # of Slaves** | 1008 |

**Note**: Slave devices address uses a 7-Bit address but the bus specification allows an extension to 10 bits as well.

# I2C in RP2040

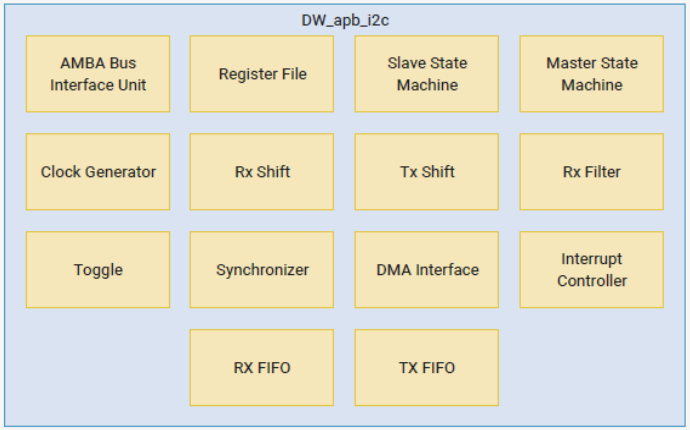
# I2C Block Diagram in RP2040



# I2C Block Diagram in RP2040: Explained

The following define the functions of the blocks in Figure 64:

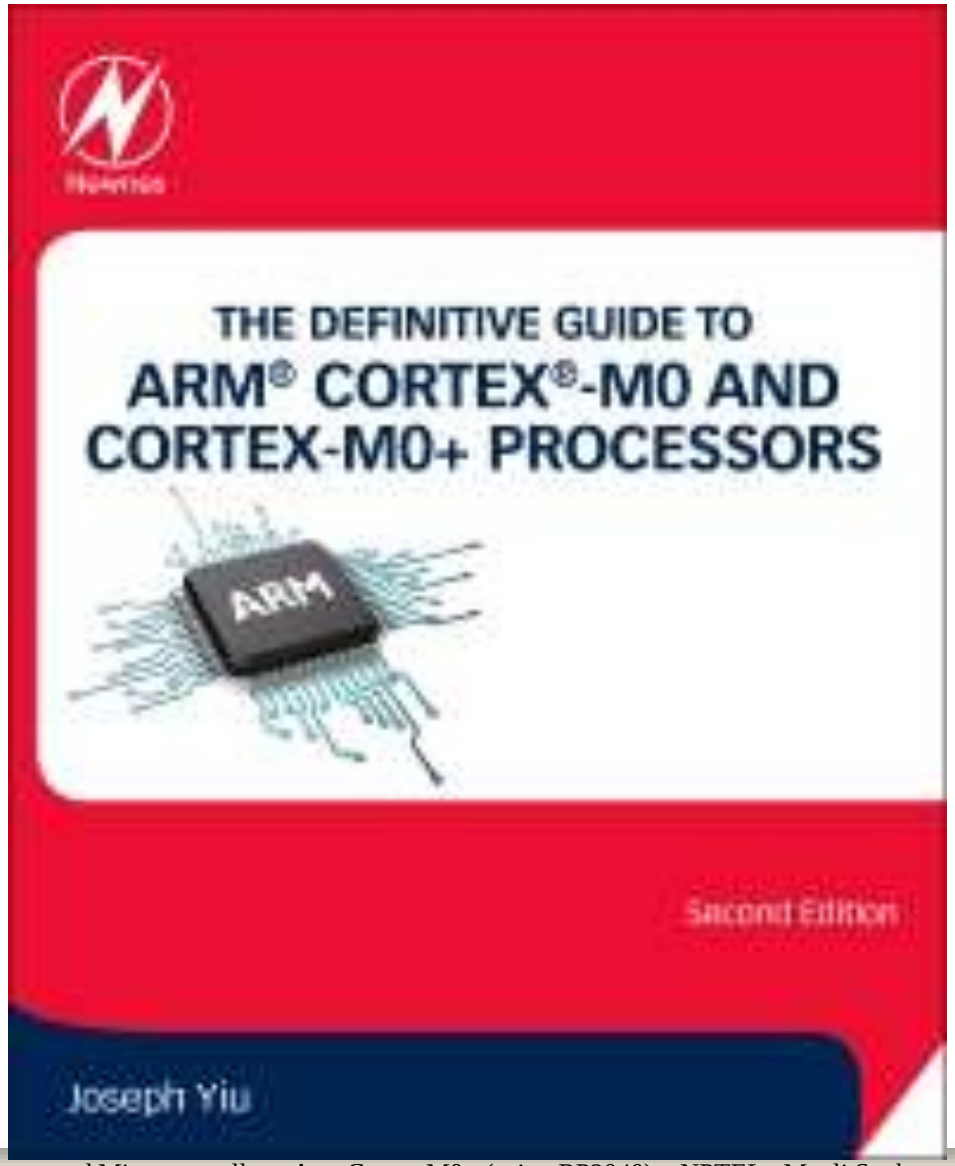
- **AMBA Bus Interface Unit** – Takes the APB interface signals and translates them into a common generic interface that allows the register file to be bus protocol-agnostic.
- **Register File** – Contains configuration registers and is the interface with software.
- **Slave State Machine** – Follows the protocol for a slave and monitors bus for address match.
- **Master State Machine** – Generates the I2C protocol for the master transfers.
- **Clock Generator** – Calculates the required timing to do the following:
  - Generate the SCL clock when configured as a master
  - Check for bus idle
  - Generate a START and a STOP
  - Setup the data and hold the data
- **Rx Shift** – Takes data into the design and extracts it in byte format.
- **Tx Shift** – Presents data supplied by CPU for transfer on the I2C bus.
- **Rx Filter** – Detects the events in the bus; for example, start, stop and arbitration lost.
- **Toggle** – Generates pulses on both sides and toggles to transfer signals across clock domains.
- **Synchronizer** – Transfers signals from one clock domain to another.
- **DMA Interface** – Generates the handshaking signals to the central DMA controller in order to automate the data transfer without CPU intervention.
- **Interrupt Controller** – Generates the raw interrupt and interrupt flags, allowing them to be set and cleared.



# Summary

- I2C Introduction
- I2C Basics
  - Configuration
  - Specification
  - Start and Stop conditions
  - Working Principle
- I2C in RP2040

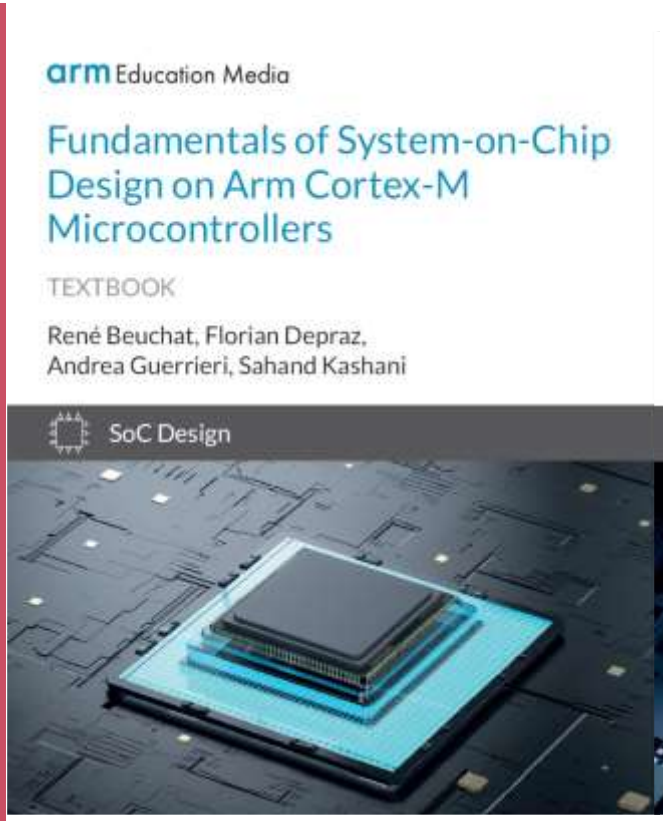
# Additional Reference 1



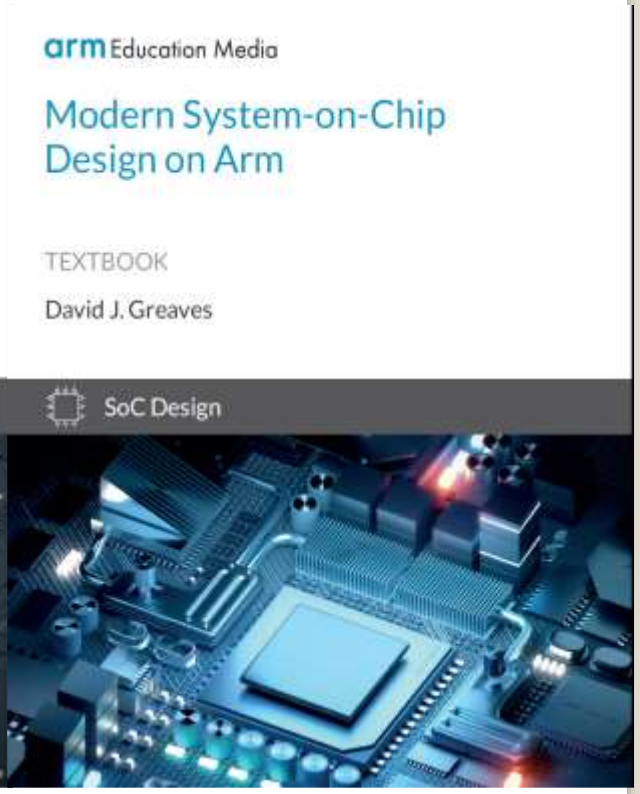
# References - 1

**Ref 0**


**Ref 1**

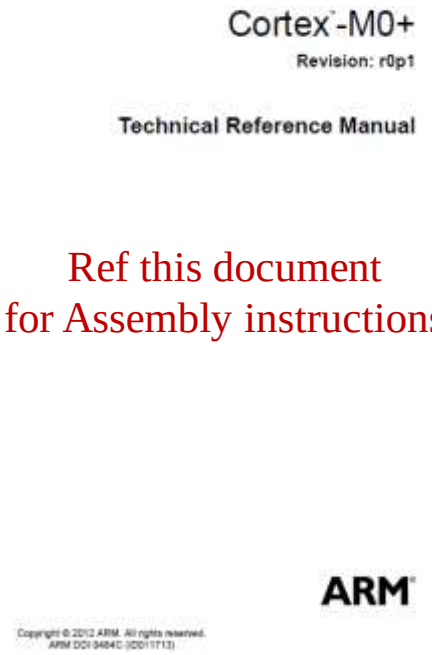



**Ref 2**



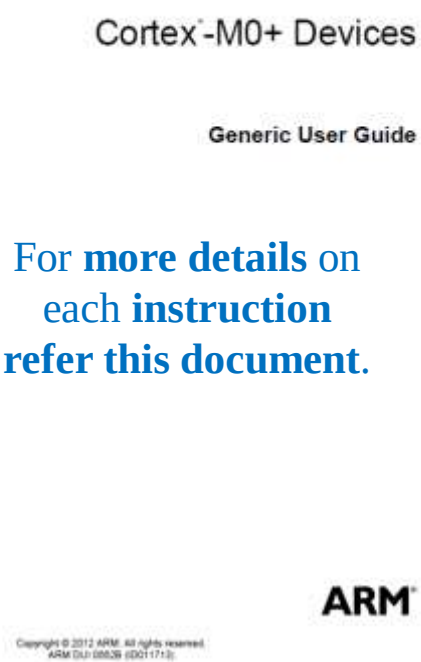
# References - 2

**Ref 3**



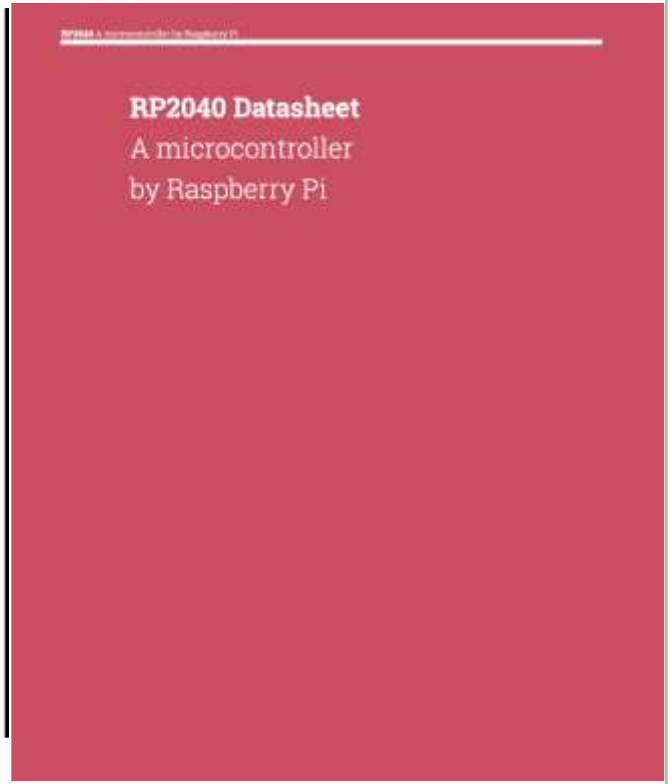

Ref this document
for Assembly instructions

**Ref 4**




For **more details** on
each **instruction**
**refer this document**.

**Ref 5**




**Ref 6**