# **Embedded Systems Essentials with Arm: Get Practical with Hardware**

## Module 2

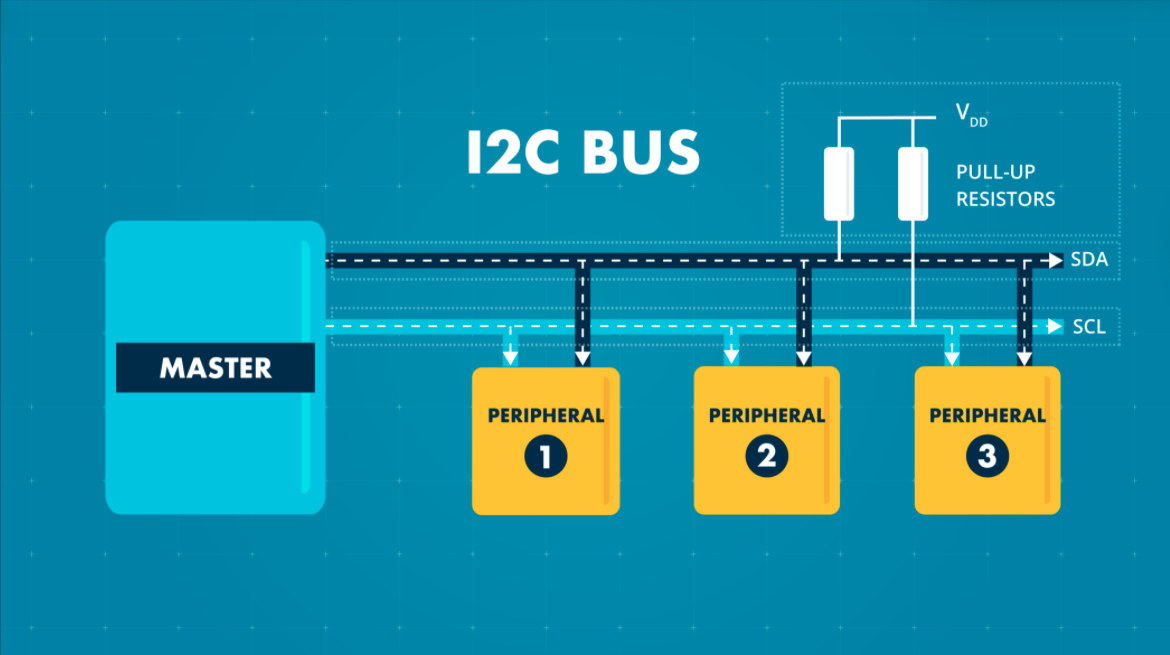
## KV1: Introducing I2C (Inter-Integrated Circuit)

In the previous module we met SPI (serial peripheral interface). We saw that it’s a simple and fast method of data transfer, which was created more than 40 years ago but is still widely used today. We recognized its weaknesses:

* There’s no addressing.
* It becomes inflexible and there’s no acknowledgement when data is transferred.

This becomes a problem if we want to create a complex system because for every slave we add, we have to add an extra chip select line. This increases the complexity of our system and we lose the advantage of serial communication.

In 1982, Philips Electronics proposed a protocol called Inter-Integrated Circuit (I2C). This recognized the strengths of SPI but addressed some of the weaknesses. In those days, electronic products, especially domestic products, were made of a mass of integrated circuits, and data had to be transferred between those integrated circuits. This is where the name Inter-Integrated Circuit comes from.

This is an example of an I2C bus.

You will see that we have just two wires, we don’t add extra wires as we add extra slaves. The same two wires will go to every node on the bus. These two wires are labeled SCL (for serial clock) and SDA (for serial data). Each wire has a pull-up resistor.

This is a master-slave relationship, so at any moment we will have a connection between one master and one or more slaves.

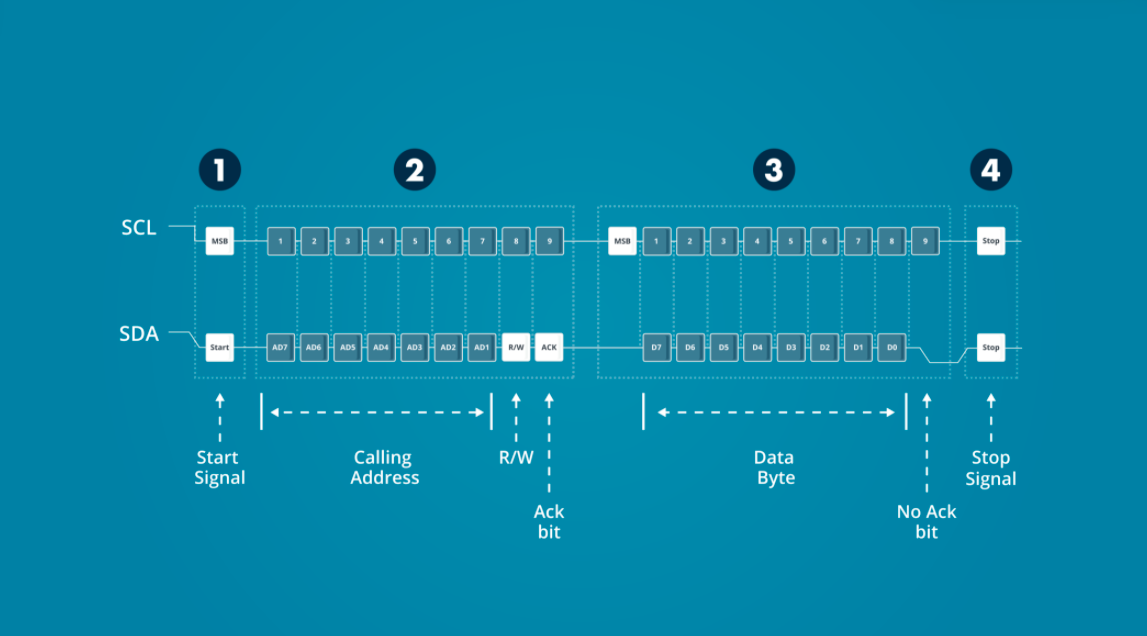
It’s possible to have more than one master, but only one can be active at one time. Unlike in SPI, where we can just send the data, here, we need a start condition and a stop condition. Importantly, within the message there is going to be addressing. There’s no chip select line, but there’s an address contained within the message, and each slave has its own unique address. When the slave receives the data, there will be acknowledgement.

Let’s now have a look at what’s happening inside a single node. The usefulness of this connection is that any node, whether master or slave, can only pull the line low, and when it lets go, the line returns to Logic 1, which is the idle state.

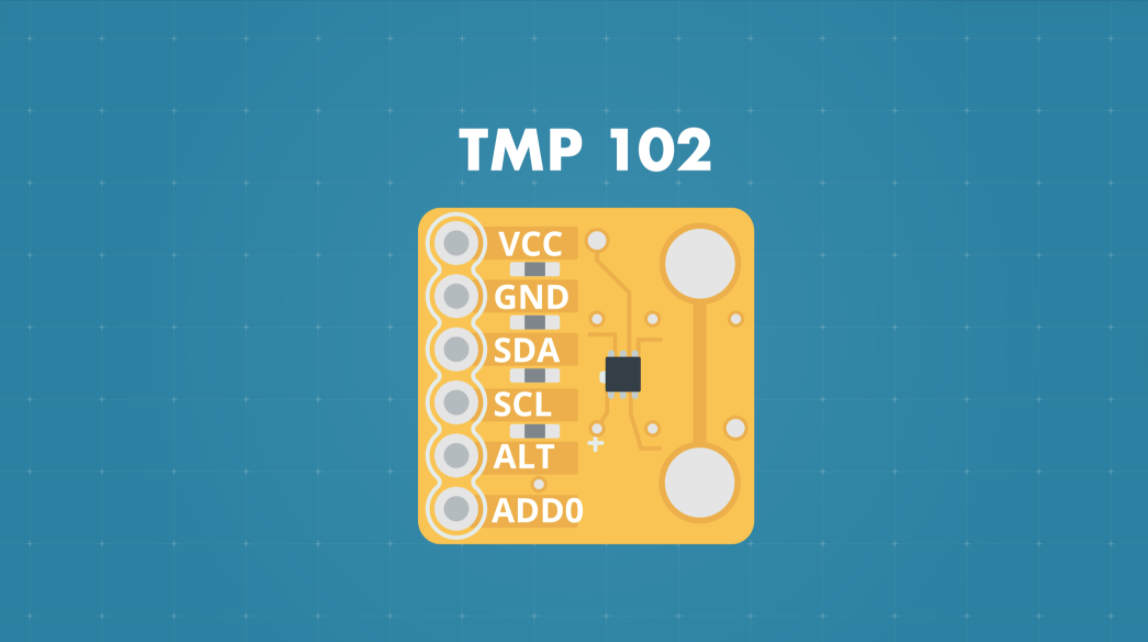
The master always generates the clock, which can start from 400kHz up to 1MHz.

We use a specific format when sending information. We have a start signal and a stop signal and we package our information within that. This is an explicit statement from the master that something is about to happen. Before that the system is idling at Logic 1. We commit the first byte to contain a seven-bit address. Each slave must have a different address, and the slave will be looking at the address of incoming data. If it recognizes its own address it will respond. We have one bit labeled “read-not-write” (R/W), and that is the master communicating with the slave telling it whether it wants to read or write. The rest of the message will be in that modality. The byte will also have an acknowledgement bit. We will then transmit data in the direction indicated.

This diagram shows a single-address byte and a single data byte.



The TMP 102 is an example where the I2C protocol can be applied. Here, we see it on a breakout board, which allows you to experiment with it.



Here is the connection diagram with its recommended connections.

|  |  |
| --- | --- |
| **Signal​** | **Notes​** |
| **VCC (3.3V)​** | ​ |
| **SDA​** | 2.2 kΩ pull-up to 3.3 V​ |
| **SCL​** | 2.2 kΩ pull-up to 3.3 V​ |
| **GND (0V)​** | ​ |
| **ALT (Alert)​** | ​ |
| **ADD0​** | Connect to Slave address​  0V 0x90​  Vcc 0x91​  SDA 0x92​  SCL 0x93​ |

**Visit arm.com for more information.**