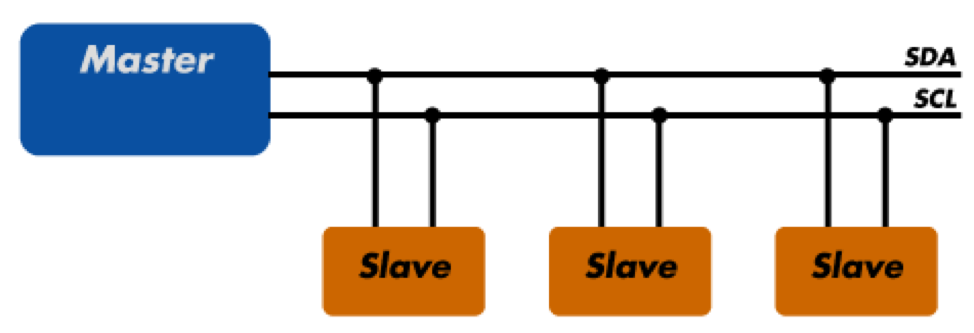
## Understanding I2C and SPI Communication Protocols

### I2C Protocol

[I2C](https://www.totalphase.com/blog/2020/07/understanding-i2c-communication-and-how-to-debug-the-i2c-protocol/), or Inter-Integrated Circuit, is a simple communication protocol often used in embedded systems as a way to transfer data between a master (or multiple masters) and a single slave (or multiple slaves) device. **It is a bidirectional two-wire serial bus that uses serial clock (SCL) and serial data (SDA) wires to send and manage data bit by bit between devices connected to the bus.**



In I2C operations, the master controls the exchange of data between the devices. A master device will signal to a slave in order to send data or request a response. To accomplish this, all slave devices must have a **unique address that is included in the I2C message.**

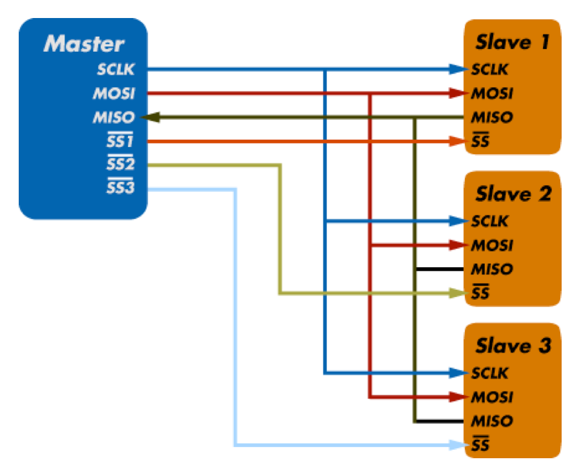
When sending data over the bus, **each I2C message includes an address frame of the slave device and one or more data frames containing the data being transmitted.** The message also includes start and stop conditions, read/write bits from either the master or slave, and ACK/NACK bits sent from the receiver for error checking.

I2C is considered to be **synchronous, meaning it operates using a serial clock.** The clock is driven by the master device which allows the output of bits to be synchronized to the sampling of bits by the clock signal shared between the master and the slave.

The standard data transfer rate of the **I2C protocol is 100 kbps, although data speeds of up to 5 Mbps** are possible with I2C devices configured in "fast mode" or "ultra-fast mode".

### SPI Protocol

[SPI](https://www.totalphase.com/blog/2020/07/what-is-spi-protocol-how-to-debug-spi-communication/), or Serial Peripheral Interface, is a serial communication protocol often used in embedded systems for high-speed data exchanges between devices on the bus. It operates using a **master-slave paradigm that includes at least four signals: a clock (SCLK), a master output/slave input (MOSI), a master input/slave output (MISO), and a slave select (SS) signal.** The SCLK, MOSI, and MISO signals are **shared** by all devices on the bus. The SCLK signal is generated by the master device for synchronization, while the **MOSI and MISO lines used for data exchange.** Additionally, each slave device added to the bus has its own SS line. The **master pulls low on a slave's SS line to select a device for communication.**



SPI communication supports [full-duplex communication](https://www.totalphase.com/blog/2018/12/spi-protocol-speed-embedded-systems-need/), meaning **that both the master and slave can transmit data simultaneously.**

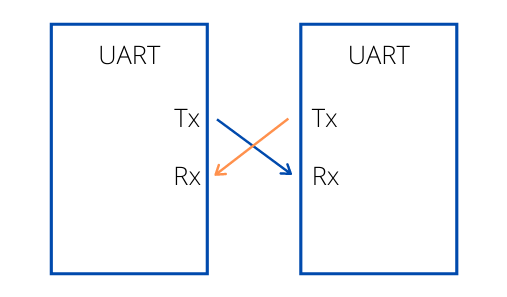
The exchange itself has no pre-defined protocol which makes SPI ideal for data-streaming applications. It also has no maximum speed; data speeds in excess of **100 MHz have been achieved.**

## Understanding UART

UART, or **Universal Asynchronous Receiver/ Transmitter, is a physical circuit** in a microcontroller or single [integrated circuit](https://www.totalphase.com/blog/2020/12/differences-between-integrated-circuit-microprocessor/) (IC) that is used to implement serial communication between devices in an embedded system. Essentially, a UART’s main purpose is to transmit and receive serial data.

**In UART communication, two UARTs communicate directly with each other; the UART on the sender device, or the transmitting UART, receives parallel data from the CPU (**[**microprocessor or microcontroller)**](https://www.totalphase.com/blog/2019/12/microcontroller-vs-microprocessor-what-are-the-differences/)**and converts it to serial data. This serial data is transmitted to the UART on the receiver device, or the receiving UART. The receiving UART converts the received serial data back to parallel data and sends it to the CPU. In order for UART to convert serial-to-parallel and parallel-to-serial data, shift registers on the transmitting and receiving UART are used.**

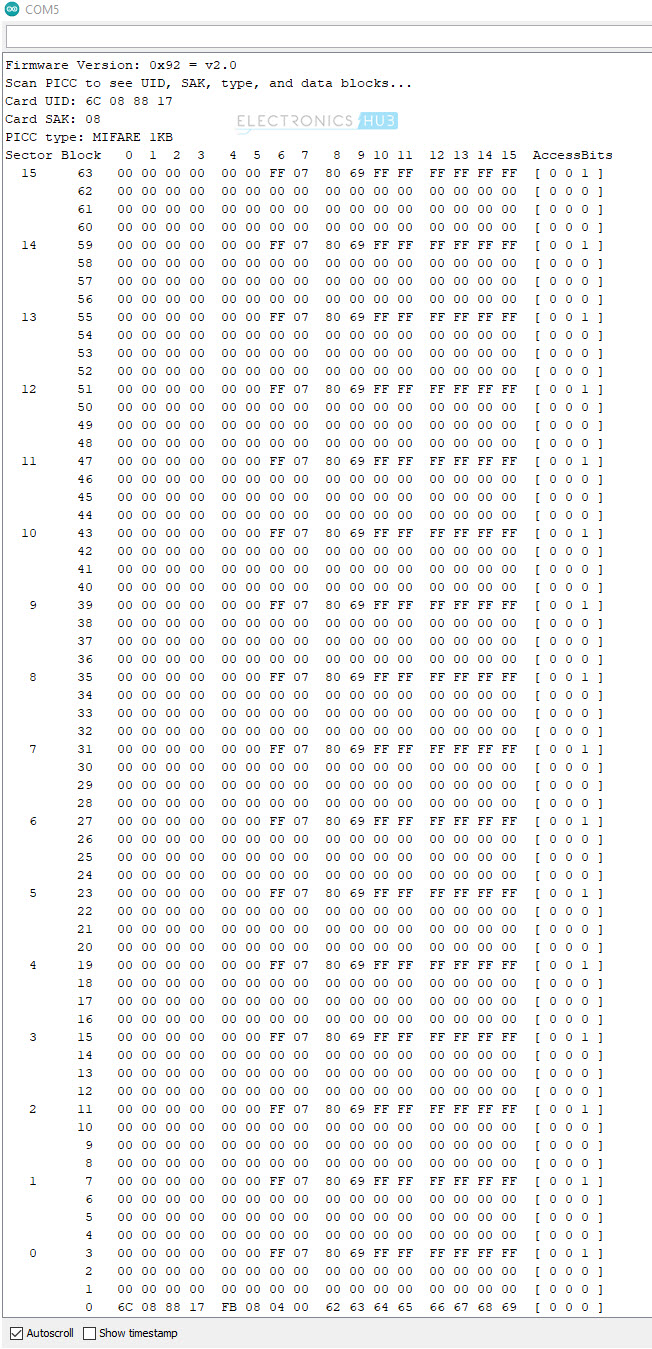
In UART communication, only two wires are required for communication: data flows from the Tx pin of the transmitting UART (Transmitter Tx) to the Rx pin of the receiving UART (Receiver Rx).



**UART data is sent over the bus in the form of a packet. A packet consists of a start bit, data frame, a parity bit, and stop bits. The parity bit is used as an error check mechanism to help ensure data integrity.**

UART is considered to be “universal” because the parameters including transfer speed and data speed are configurable by the developer. UART supports bidirectional data transmission, including half-duplex and full-duplex operations. **It is also asynchronous, meaning it doesn’t use a clock signal to synchronize the output bits from the transmitting UART to the sampling bits on the receiving UART. Without a clock, the receiving and transmitting UART need to be on the same baud rate, or bit rate. This allows the system to know where and when the bits have been clocked.**

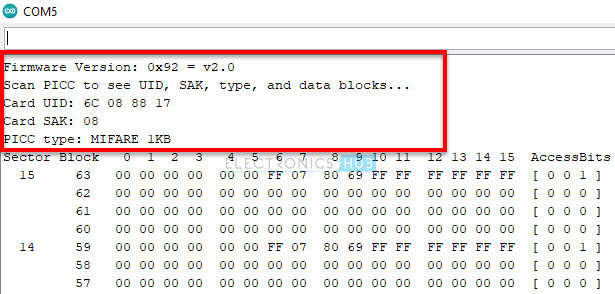
The following image is a screen shot of the serial monitor output of ‘DumpInfo’ example. Let us now analyze this.

[](https://www.electronicshub.org/wp-content/uploads/2021/02/RC522-DumpInfo-Serial-1.jpg)

**Analyzing Serial Monitor Output**

The first line shows the firmware version of the MFRC522 IC. In this case, the result is 0x92. Here, ‘9’ stands for MFRC522 IC and ‘2’ stands for software version 2.0. After scanning the RFID Card, we get the UID, SAK and Type of RFID tag.

In this case, the UID is ‘6C 08 88 17’, SAK is ‘08’ and the type of card is MIFARE 1K.

[](https://www.electronicshub.org/wp-content/uploads/2021/02/RC522-DumpInfo-Serial-2.jpg)

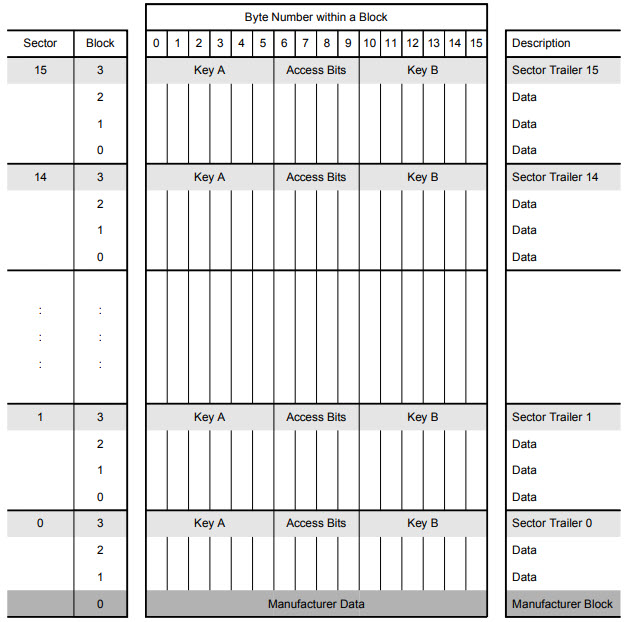
Next, you can see the actual memory dump of the MIFARE 1K Tag. A typical MIFARE 1K RFID tag has 1K Byte of memory organized into 16 Sectors (Sector 0 to Sector 15). Each Sector consists of 4 Blocks.

**Understanding memory Map of MIFARE 1K Tag**

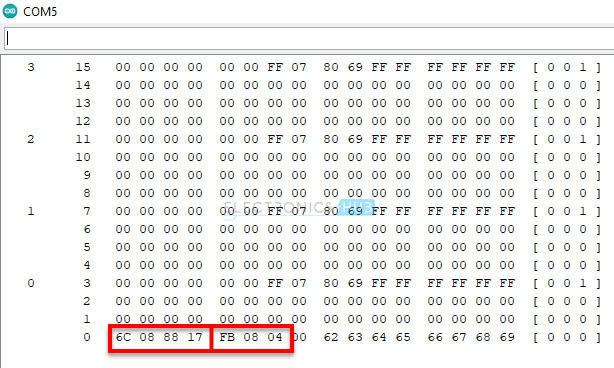
For example, Sector 0 has Blocks 0, 1, 2 and 3. Sector 1 has Blocks 4, 5, 6 and 7 and so on and finally Sector 15 has Blocks 60, 61, 62 and 63. Each Block can store 16 Bytes of data.

**NOTE:** This numbering is just to understand the memory layout.

So, 16 Sectors \* 4 Blocks \* 16 Bytes = 1024 Bytes = 1K

[](https://www.electronicshub.org/wp-content/uploads/2021/02/MIFARE-1K-Memory-Map.jpg)

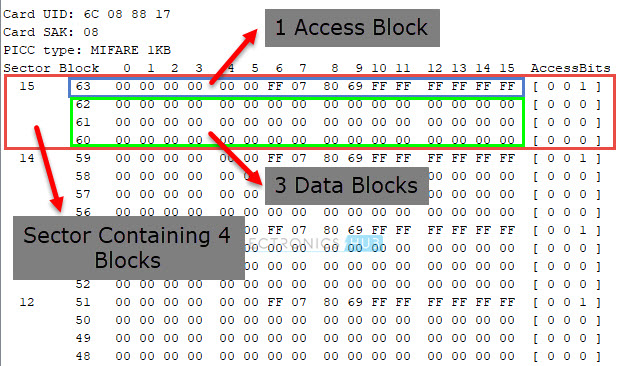
Block 0 of Sector 0 is reserved for storing Manufacturer Data. Usually, this Block contains 4 Byte UID (Unique ID) in case of MIFARE 1K Tags (and also MIFARE 4K, MIFARE Mini tags from NXP). Advanced Tags like MIFARE Plus, MIFARE Ultralight, MIFARE DESFire consists of a 7 Byte UID.

[](https://www.electronicshub.org/wp-content/uploads/2021/02/RC522-DumpInfo-Serial-3.jpg)

Each Sector consists of three Data Blocks, which can be used for storing user data. The last Block of each Sector i.e., Block 3 in case of Sector 0, Block 7 in case of Sector 1 and so on is known as Sector Trailer.

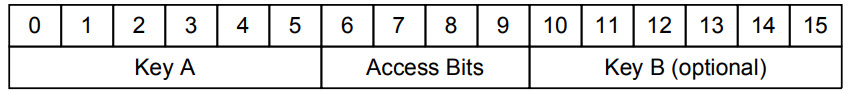
As there are 16 Sectors, there are 16 Sector Trailers. Each Sector Trailer consists of the following information:

* A mandatory 6 Byte Key A.
* 4 Bytes for Access Bits.
* Optional 6 Byte Key B (if not used, data can be stored).

[](https://www.electronicshub.org/wp-content/uploads/2021/02/RC522-DumpInfo-Serial-4.jpg)

**NOTE:** Byte 9 in the ‘Access Bits’ region is available for user data.

**NOTE:** All sectors have three data blocks and one sector trailer except sector 0. It has one block (Block 0) reserved for Manufacturer Data. So, Sector 0 has two data blocks and one sector trailer.

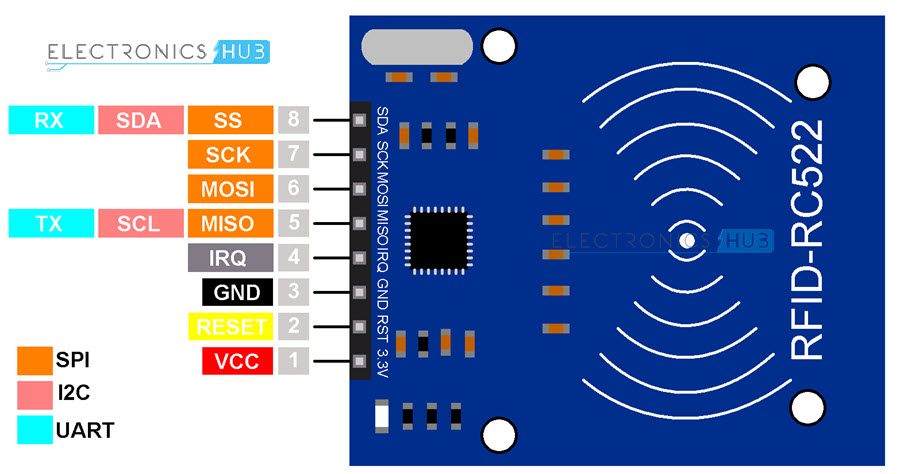
[](https://www.electronicshub.org/wp-content/uploads/2021/03/MIFARE-1K-Sector-Trailer.jpg)

The Access Bits in the Sector Trailer determine the access conditions for all the blocks of a Sector. 3-bits are needed for specifying access conditions for the three data blocks and the sector trailer. The access condition includes Read, Write, Increment, Decrement, Transfer and Restore.

The MFRC522 is a highly integrated reader or writer IC for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO/IEC 14443 A/MIFARE mode.

MFRC522 library:

This library allows you to read/write data from RFID tags easily. Three ways of reading/writing are available: (1) as binary data chunks of any fixed/known size, (2) as binary data chunks identified by labels, with lengths possibly unknown at read-time, or (3) as a dictionary with key/values pairs.

[](https://www.electronicshub.org/wp-content/uploads/2021/02/RC522-RFID-Reader-Pinout.jpg)

**A PICC (Proximity Integrated Circuit Card)** is a standardized model of a card that serves as a reference device for compliance testing of commercial readers and acts as a model for commercial card's ICs (Integrated Circuits) during antenna coil optimization.

RFID reader RC522 has a reading range of approximately 1 meter.

**PCD(Proximity Coupling device):** Also known as RFID readers. They decode the RFID Tags and communicate with them based on ISO14443 standard. PCD can perform read and write operation of data.

**The communication between a PCD (Proximity Coupling Device; aka a reader) and a PICC (= Proximity Integrated Circuit Card) is performed using an electromagnetic field coupling between the reader and the card.**

**The PCD ensures the generation of the magnetic field whereas the antenna of PICC allows receiving the magnetic field.**

MFRC522

|  |  |
| --- | --- |
| **Model** | **MF522-ED** |
| Working frequency | **13.56MHz** |
| Card reading distance | 0-60mm |
| Protocol | SPI |
| Data communication speed | 10Mbit/s Max |

**Kuongshun Electronic**, one of the international well-known manufacturers and suppliers of rc522 mfrc-522 rfid module which is situated in China, now has quality products for sale.

MFRC522 RC522 can read/write RFID cards and tags built using ISO/IEC 14443 protocol. This includes MIFARE-compatible RFID tags like

**MIFARE-mini, MIFARE-1K, MIFARE-4K RFID tags, Key Fob, and NTAG RFID cards.**

An RFID reader is a **radio frequency device that emits a signal through an antenna.** This signal is received by RFID tags that respond to interrogation by the reader. Responses are read by the reader, and through a variety of protocols the reader can communicate with all the RFID tags in its field.

The PICC\_HaltA() function sends a halt command to the RFID card, which stops further communication with the card.

The PCD\_StopCrypto1() function stops the encryption of the data between the RFID card and the reader.