In **Embedded** applications, among the most frequently used devices, there are the **Serial EEPROM**. These devices are available with different memory sizes, different packages and moreover with some, and different, **Communication Protocols**. In this chapter we’ll describe the **I2C BUS Protocol**.

The **EEPROM** are **memory, not Volatile**: this means that they mantains the stored information even without power supply. The devices can be **written** a very high number of times and they can be **read** without limits, as standard **RAM**.

The management operations are defined by accurate rules, that are described below.

The internal structure is reported in the following **diagram** that lists all the logic blocks included in the device.

In order to get a detailed and bright vision of the device, we suggest You firstly to read and study the specific component **Data Sheet** and then to try the developed examples.
I2C BUS SERIAL PROTOCOL

I2C stands for Inter Integrated Circuit (pronounced i-square-c or i-two-c), and it is a Doublewire Serial Communication system used between integrated circuits.

The BUS has been developed by Philips in 1982 and, after the creation of hundreds of components and systems during the ’80s, in 1992 has been released the first version of Protocol. It has been updated in different times and it has generated similar BUSes, too. One of these (named SM BUS) was developed only for commercial reasons and copyrighted by Intel, in 1995.

The I2C BUS is composed by 2 lines named SDA (Serial DAta) and SCL (Serial Clock) and both them are open collector type. The SDA line is bidirectional and it allows data exchanges between the connected I2C device, viceversa SCL line is monodirectional and it is used as a Clock signal to synchronize the communication. This BUS, originally developed by Philips, is used in many different devices types even in commercial products. For example the serial EEPROM of 24Cxx family are widely diffused.

The communication with an I2C component has some features applicable to each device. The begin and the end of I2C communication are always recognized by specific START and STOP conditions, sent on the BUS. These are obtained with defined sequences of logic states assigned to SDA and SCL lines.

START and STOP on BUS

In released conditions the SDA and SCL lines are maintained at logic level 1. The START sequence equal to a falling edge (from 1 to 0) on SDA line when SCL is maintained at level 1. Viceversa the STOP sequence equal to a rising edge (from 0 to 1) on SDA line while SCL is at level 1.

SCL and SDA Signals During START and STOP
Once **Start** sequence has been generated the logic level of **SDA** line is considered valid, and it is read by **I2C** device when **SCL** is at level 1, viceversa when **SCL** is at level 0 each variations of **SDA** don’t care.

**Data Transmission on BUS**

After start sequence, each data to send or receive from **I2C BUS** is composed by 8 bits (that is a byte), plus a nineth bit named **ACK**, dedicated to check if the communication has been succesfully executed. Below there i san example, where 2 bytes are transmitted:

![I2C BUS Signals During Communication](image)

As you can see, the first bit to send is the most significative (MSB).

For any bit it must be generated a pulse on **SCL** line.

The nineth bit defined **ACK** normally has a level 0 and it is generated by the device that has received the Data. The level 1 on ninth bit is defined **NAK** and it signalizes the last communicated byte, an error, the **I2C** device absence, etc.
Device Addressing on BUS

As it happens for all the BUS even for the I2C some different devices can be connected to SCL and SDA lines.

In the communications the first byte sent always coincide with the SLAVE ADDRESS, that is the address of the device involved in communication just started.

Each I2C device has its own single SLAVE ADDRESS, described in Data Sheet, composed by 7 bits plus the eight bit (LSB) that selects the communication direction between Read and Write (R/W).

Blocks Diagram of I2C BUS Serial EEPROM, AT24C08
Many devices have a **SLAVE ADDRESS** partially modificable, by hardware, with the status of **An** signals. In this way on the same **BUS** it can be connected even more devices of the same type as, for example, up to 2 **EEPROM 24C08** or 4 components **24C04**.

The **I2C BUS** communication can be managed by any microcontroller with at least 2 I/O lines: one **Bidirectional** and the other can be only monodirectional output.

Thanks to specific **Instructions** of **BASCOM**, these lines generates the foundamental operations of the communication, that so is managed at high level.

Each couple of **Mini Module** lines can be used as an **I2C BUS** interface. For convention and compatibility reason among the **Pin-Outs** of the various **Mini Modules**, two lines have been selected and standardized as described in the following electric diagrams and inside the **technical sheets**.

The diagrams describe the used **Mini Module** mounted on the test card **GMM TST3** and then connected to external experimentation circuits, through the **CN3** connector.
Example.051. Byte R/W Management of a Serial EEPROM AT24C08

**Added Definitions:**
None

**Added Declarations:**
None

**Added Instructions:**
CONFIG SCL ; CONFIG SDA ; I2CINIT ; I2CSTART ; I2CWBYTE ; I2CRBYTE ; I2CSTOP.

**Added Operators:**
None

Example program 51 of BASCOM AVR course.

Test and management program for I2C BUS EEPROM 24C08A, at byte low level, without BASCOM instructions.
It performs the fundamental operations on the component by using a software 
I2C BUS interface and by interacting with user on a serial console provided of 
monitor and keyboard with a fixed physical protocol at **19.200 Baud, 8 Bit x chr, 1 Stop bit, No parity.**
Connection Between Z2 Socket and CN3 Connector of GMM TST3

This console can be another system capable to support a serial RS 232 communication. In order to simplify the use it can be used a PC provided of one COMx line, that execute a terminal emulation program as HYPERTERMINAL or the homonym modality provided by BASCOM AVR (see IDE Configuration).

The program works only when the GMM AM08 is mounted on Z2 socket of GMM TST3!!
Example 052. R/W Management of a Serial EEPROM 24C08
With Messages Long up to 20 Characters, Displayed on Console

Added Definitions:
None

Added Declarations:
None

Added Instructions:
None

Added Operators:
None

Example program 52 of BASCOM AVR course.

Messages read and write on I2C BUS EEPROM 24C08.

It performs the operations on the component by using a software I2C BUS interface and by interacting with user on a serial console provided of monitor and keyboard with a fixed physical protocol at 19.200 Baud, 8 Bit x chr, 1 Stop bit, No parity.

This console can be another system capable to support a serial RS 232 communication. In order to simplify the use it can be used a PC provided of one COMx line, that execute a terminal emulation program as HYPERTERMINAL or the homonym modality provided by BASCOM AVR (see IDE Configuration).

The program works only when the GMM AM08 is mounted on Z2 socket of GMM TST3!!