**ABOUT THE SPI PROTOCOL AND THE W25Q128 FLASH MEMORY**

SOURCE OF INFORMATION: <https://www.youtube.com/watch?v=OSfu4ST3dlY>

**INTRODUCTION:**

The following image is an extraction from the datasheet, which is shown in this document as a reference of the W23Q Flash Memory that has 16 Megabytes in order to use it as a reference for how to electrically wire it to our MCU:

A diagram of a circuit

Description automatically generated

These W23Q Flash Memories use the either the Dual/Quad SPI Protocol or the Standard SPI Protocol for establishing communication between the desired Flash Memory and whatever external device we are using with it (e.g., an MCU). There are several versions of these W23Q Flash Memories, where they differ on the total amount of Flash Memory Size and where they all work similarly at the base level. However, the most practical aspect of these Flash Memories is that their Memory is distributed the same way in all of them at the lower level.

Note that in this project, the Standard SPI Protocol will be used.

**HOW TO CONNECT A W25Q FLASH MEMORY TO OUR MCU**

The following image illustrates how to connect a W23Q Flash Memory to a MCU:

A circuit board with wires

Description automatically generated

Where this reference image I picked from the internet that conceptually shows how to connect several devices to the same SPI lines. It is important to note that each device can be independently selected by using their corresponding Chip Select Pin (i.e., the CS Pin). However, please have in mind that at least in my case, with the STM32F103C8T6 Blue Pill MCU, which looks alike to the MCU Blue Pill of that image, I was not able to make the SPI work with those lines which are supposedly enabled with the Alternate Pin function of the STMicroelectronics MCU/MPU devices (which is a functionality in the peripheral configurations of those devices where you can use transfer specific peripherals in your MCU/MPU to other pins). At least with that particular STM32F103C8T6 Blue Pill MCU that I used for this project, the SPI lines that actually worked were the ones that are activated as default and whenever I used the ones with the Alternate Pin function, it did not work for me.

As a reference, know that DI stands for Data In and that DO stands for Data Out.

**SOME GENERAL ASPECTS TO KNOW ABOUT THE W25Q128 FLASH MEMORY THAT WILL BE USED IN THIS PROJECT**

* The W25Q128 Flash Memory has a Flash Memory array organized into 65’356 programmable pages of 256 bytes each.
* Up to 256 bytes can be programmed at a time.
* The W25Q128 Flash Memory has a Flash Memory has a total size of 128 Mbits (i.e., 16Mbytes).
* Pages can be erased in groups of 16 pages (i.e., 4KB sector erase), groups of 128 pages (i.e., 32KB block erase), groups of 256 pages (i.e., 64KB block erase) or the entire chip (i.e., chip erase).
* The W25Q128 Flash Memory has 4’096 erasable sectors and 256 erasable blocks respectively.
* The small 4KB sectors allow for greater flexibility in applications that require data and parameter storage.
* SPI clock frequencies of up to 104MHz are supported, where it is also allowed to have equivalent clock rates of 208MHz (i.e., 104MHz x 2) for Dual I/O and 416MHz (i.e., 104MHz x 4) for Quad I/O when using the Fast Read Dual/Quad I/O and QPI instructions respectively.
* The Continuous Read Mode allows for accessing memory with as few as 8 clocks of instruction overhead to read a 24-bit address, where this reading mode can be made for 8, 16, 32 or 64 bytes in a Wrap.

The following image illustrates the Block Diagram of a W25Q128 Flash Memory:

A diagram of a computer

Description automatically generated

**INFORMATION TO KNOW BEFORE READING THE ID OF A W25Q FLASH MEMORY FROM OUR MCU**

A necessary thing to do whenever communicating with a W25Q Flash Memory is to learn its corresponding ID, which is unique to each product series.

In particular for the W25Q128FV Flash Memory in Standard SPI Protocol, you can check its ID under the following instruction section:

A white rectangular box with black text

Description automatically generated

**CONFIGURING YOUR STM32 MCU FOR ESTABLISHING COMMUNICATION WITH A W25Q FLASH MEMORY**

Although the instructor, from the Tutorial from which the information of this document has been based from, has suggested to set a prescaler such that the Baud Rate is around 2.5 Megabits per second, since I am using a different MCU than this person and because I desire to configure it with a Clock Frequency of 2MHz (this is because in many applications where I used the MCU that I used, which is the STM32F103C8T6, I want to set a low Clock Frequency to attempt to lower the energy consumption of the device as much as possible, but that is also something with which most general applications can work with), I will be using 1Mbit/sec Baud Rate instead.

Also, according to the following extraction of information from the W25Q128FV datasheet:

A close-up of a computer screen

Description automatically generated

This type of Flash Memory may work with either the SPI Mode 0 or SPI Mode 3, where basically the only difference between them is the polarity that the Flash Memory will consider with respect to the rising/falling edges of the Clock signal (read the extraction of information shown for more details on this).

Because of this information, it is now known that we can configure the SPI of our MCU to have a Clock Polarity of either Low or High Type. So, in order to follow the same configuration as the instructor from the tutorial from which all the information of this document has been based on, the Low Clock Polarity will be configured in the SPI of our MCU.

Also, our MCU will be configured with a Clock Phase of 1 Edge, which means that the data will be sampled on the first clock edge, which should correspond to the rising edge. In addition, the Clock Polarity configured will be Low. As a result, since Clock Phase has a value of 1 Edge and since the Clock Polarity has a value of Low, then this means that our MCU will be configured to work with Mode 0 Standard SPI.

Another aspect to take into account is that whatever GPIO Output Pin you configure in your STM32 MCU for controlling the Chip Select Pin of the W25 Flash Memory you are programming, is to configure the initial GPIO Output Level of that MCU’s GPIO Output Pin to be High initially so that the W25 Flash Memory is disabled right after our MCU boots. This way, we can enable that Flash Memory whenever we desire by simply changing the State of that GPIO Output Pin to be from High to Low.

One last thing to consider is that the instructor, of the tutorial from which the information of this document was based on, configured the Chip Select GPIO Output Pin, that was just mentioned in the previous paragraph, to work with the highest possible Maximum Output Speed. However, I will try it with the Lowest one since I am more interested in being able to use this Flash Memory in Low Power Applications (the lower the frequency used in the peripherals, the lower the energy consumption in our MCU).

**HOW TO RESET THE W25Q128FV FLASH MEMORY PRIOR TO START USING IT**

The following image is an extraction from the W25Q128FV Datasheet that contains the information that needs to be known for sending the Reset Command to a W25Q128FV Flash Memory, which can be made by sending the Enable Reset and then the Reset Device commands in that order, as explained there:

A diagram of a computer program

Description automatically generated

**HOW TO READ THE MANUFACTURER DEVICE ID OF THE W25Q128FV FLASH MEMORY**

There are several IDs that can be read from the W25Q128FV Flash Memory Device according to its datasheet, like the Manufacturer ID, the Device ID and the JEDEC ID. However, from among these, the JEDEC ID is more particular to a specific type of W25 Flash Memory Device (It will give the same ID for the same type of Flash Memories with the same Memory Size, but with that ID, you could tell the type of Flash Memory and its size as it was just mentioned). Therefore, the one that we will be reading for this project is the JEDED ID:

A diagram of a computer

Description automatically generated

***NOTE:*** *The expected response from a W25Q128FV Device after sending to it the Read JEDEC ID Instruction in Standard SPI Protocol, is to receive the value of 0x40 for the Memory Type and the value of 0x18 for the Memory Capacity, according to the following table extracted from its datasheet:*

A white rectangular box with black text

Description automatically generated

**HOW TO READ DATA FROM THE W25Q128FV FLASH MEMORY**

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Read Data instruction:

A diagram of a computer

Description automatically generated

**HOW TO FAST READ DATA FROM THE W25Q128FV FLASH MEMORY**

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Fast Read Data instruction:

A diagram of a computer

Description automatically generated

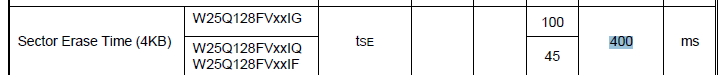
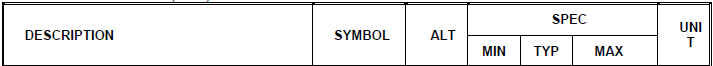
**HOW TO SECTOR ERASE (THE MINIMUM ERASABLE SIZE) IN THE W25Q128FV FLASH MEMORY**

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Sector Erase instruction:

A close-up of a document

Description automatically generated

Where the datasheet states the following value for :



**THE WRITE ENABLE INSTRUCTION AND THE WRITE DISABLE INSTRUCTION OF THE W25Q128FV FLASH MEMORY**

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Write Enable instruction:

A diagram of a computer program

Description automatically generated

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Write Disable instruction:

A diagram of a computer

Description automatically generated

**HOW TO WRITE FROM ONE TO 256 BYTES (I.E., A PAGE) IN THE W25Q128FV FLASH MEMORY**

The following image is an extraction from the datasheet of the W25Q128FV Flash Memory about how to use the Page Program instruction:

A page of a computer program

Description automatically generated

Where:

