**Introduction:**

This module is responsible for save data in the (Flash / Internal EEPROM / External I2C EEPROM / External SPI EEPROM).

**Module APIs:**

|  |
| --- |
| sint\_16 flash\_init(void) |
| sint\_16 flash\_save(enu\_memory\_type mem\_typ , enu\_saved\_data svd\_data , uint\_8 \* data ,uint\_16 data\_size) |
| sint\_16 flash\_load(enu\_memory\_type mem\_typ , enu\_saved\_data svd\_data , uint\_8 \* data ,uint\_16 data\_size) |

1. flash\_init

This Function Responsible For Initializing the module.

1. flash\_save

This Function Responsible for saving Data on the on the EEPROM, if the data exist before it updates it in the same address and if not it saves it in new address

1. flash\_load

This Function Responsible for saving Data on the on the EEPROM, if the data exist it will load it and if not exist it will return no loaded data error.

**Module Design:**

For saving data we should make sure we can loaded easily when the microcontroller reset again.

So we should make frame for the data like the following

|  |
| --- |
| DATA HEADER [D=($data descripted name)S=(data size)] |
| DATA |

**DATA HEADER**: should be represented for making sure there is data here I wrote before, and the size of data is important to know where the next location could save the new data in the EEPROM.

DATA: data the that entered before.

For Example if I had an array of (8) elements for SENSOR\_A

So, Frist we should loge SENSOR\_A in the enum and for example it value in the enum is 9.

The data that will be saved in the memory will be the following

1. DATA HEADER = {‘D’,’=’,0x09,’S’,’=’,0x08}
2. DATA [0:8] byte

And the next location for saving new data will be previous location + DATA HEADER SIZE(6 byte) + previous data size (0x08).

You will find this handling in Function check\_if\_exist..☺