**Embedded System Library Guide**

(i2c\_memory Library)

Last Modified Date:

Wednesday, July 10, 2019

**Code author:** Majid Derhambakhsh

[**GitHub**: Majid-Derhambakhsh](https://github.com/Majid-Derhambakhsh)

[**E-mail**](mailto:majid.do16@gmail.com?subject=Software%20support)

**Contents**

[i2c\_memory Library guide…………………………………………………………………………………….………………… 3](#Guide)

Basic settings

[Functions](#Functions)

[Requirements …………………………………………………………………………………………………………………………. 7](#Required)

[Important tips ……………………………………………………………………………………………………………................. 7](#ImportantPoints)

[Error & Warning's …………………………………………………………………………………………………………………… 8](#ErrorAndWarnings)

[Supported memories ……………………………………………………………………………………………………………… 9](#SuppoertedMemory)

[Supported microcontroller's ………………………………………………………………………………………………… 10](#SupportedMicrocontroller)

[Version history ……………………………………………………………………………………………………………………… 11](#VersionChange)

**Guide**

To use this library, first add the c file to the program and include the following header.

#include "i2c\_memory.h"

**Memory Configuration**

Follow the steps below to set up your library:

1. First open the file **i2c\_memory\_conf.h**:

Edit the following values ​​to adjust the IC code used and its address pins:

#define \_AT24C04

#define \_MEMORY\_A0\_PIN\_STATE 0

#define \_MEMORY\_A1\_PIN\_STATE 0

#define \_MEMORY\_A2\_PIN\_STATE 0

**--** Used IC number ([available in the supported memory section](#SuppoertedMemory)).

-- Logic state of memory address pins (0 or 1).

Edit the following values ​​to config Write Protect pin:

#define \_WP\_DISABLE

#define \_WRITE\_PROTECT\_PORT GPIOx

#define \_WRITE\_PROTECT\_PIN 0

-- For disable Write Protection (comment it for activation).

-- Connected port & pin to Write Protect.

Allowed value for port: GPIOx in ARM Or PORTx in AVR.

Allowed value for pin: 0 ~ Number of pins supported per port.

**I2C Configuration**

Follow the steps below to set up I2C:

1. **I2C in AVR:**

First open the **i2c\_unit\_conf.h** file from the I2C\_UNIT folder and edit the values ​​below.

#define \_F\_SCL 100000UL

#define \_PRESCALER \_PRE1

-- I2C Frequency.

-- I2C Unit prescaler (If not correct operation, edit with the values in Table 1-2):

|  |  |
| --- | --- |
| Value | ID |
| 1 | \_PRE1 |
| 4 | \_PRE4 |
| 16 | \_PRE16 |
| 64 | \_PRE64 |

Table 1-2

**Tip:** Add i2c\_unit.c & gpio\_unit.c in AVR compilers.

1. **I2C in ARM:**

STM32 Series:

First open the **stm32\_i2c \_conf.h** file from the STM32\_I2C folder and edit the values ​​below.

#define STM32F1

#define \_CONNECTED\_I2C hi2c1

#define \_MEM\_DEF\_VAL\_BUFF\_LENGTH 50

-- Microcontroller Series.

-- Structure that contains the configuration information for the specified I2C.

-- Buffer size to clear memory (longer length more speed).

**Tip:** Add stm32\_i2c.c in ARM compilers.

**Functions**

**I2C\_Memory\_Init**

Function name void I2C\_Memory\_Init(void)

Function description This function is used to initialize memory.

Parameters -

Return values -

Example:

* I2C\_Memory\_Init();

**I2C\_MemoryIsReady**

Function name *uint8\_t* I2C\_MemoryIsReady(*uint16\_t* time\_out)

Function description This function is used to check memory availability.

Parameters

* + - **time\_out:** Timeout value in millisecond.

Return values

* + - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* device\_status = I2C\_MemoryIsReady(100);

**I2C\_Memory\_SingleWrite**

Function name *uint8\_t* I2C\_Memory\_SingleWrite(*uint32\_t* address, *uint8\_t* udata, *uint16\_t* time\_out)

Function description This function is used to write a byte of data to memory.

Parameters

* + - **address:** Memory address for write data.
    - **udata:** Data for write in memory.
    - **time\_out:** Timeout value in millisecond.

Return values

* + - \_MEM\_SIZE\_ERROR
    - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* com\_status = I2C\_Memory\_SingleWrite(0, 'A', 50);

**I2C\_Memory\_BurstWrite**

Function name *uint8\_t* I2C\_Memory\_BurstWrite(*uint32\_t* address, *uint8\_t* \*udata, *uint32\_t* size, *uint16\_t* time\_out)

Function description This function is used to write a string of data to memory.

Parameters

* + - **address:** Memory address for write data.
    - **udata:** Data's for write in memory.
    - **size:** Length of data.
    - **time\_out:** Timeout value in millisecond.

Return values

* + - \_MEM\_SIZE\_ERROR
    - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* com\_status = I2C\_Memory\_BurstWrite(0, "Hello", 5, 50);

**I2C\_Memory\_SingleRead**

Function name *uint8\_t* I2C\_Memory\_SingleRead(*uint32\_t* address, *uint8\_t* \*udata, *uint16\_t* time\_out)

Function description This function is used to read a byte of data from memory.

Parameters

* + - **address:** Memory address for reading data.
    - **udata:** Pointer to the variable to store the data received from memory.
    - **time\_out:** Timeout value in millisecond.

Return values

* + - \_MEM\_SIZE\_ERROR
    - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* com\_status = I2C\_Memory\_SingleRead(0, &my\_data, 50);

**I2C\_Memory\_BurstRead**

Function name *uint8\_t* I2C\_Memory\_BurstRead(*uint32\_t* address, *uint8\_t* \*udata, *uint32\_t* size, *uint16\_t* time\_out)

Function description This function is used to read a data string from memory.

Parameters

* + - **address:** Memory address for reading data.
    - **udata:** Pointer to the variable to store the data received from memory.
    - **size:** Length of data.
    - **time\_out:** Timeout value in millisecond.

Return values

* + - \_MEM\_SIZE\_ERROR
    - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* com\_status = I2C\_Memory\_BurstRead(12, received\_data\_array, 10, 50);

**I2C\_Memory\_Erase**

Function name *uint8\_t* I2C\_Memory\_Erase(*uint32\_t* address, *uint32\_t* quantity, *uint16\_t* time\_out)

Function description This function is used to read a data string from memory.

Parameters

* + - **address:** Memory address for reading data.
    - **quantity:** Memory length to clear.
    - **time\_out:** Timeout value in millisecond.

Return values

* + - \_MEM\_SIZE\_ERROR
    - **AVR:** \_STAT\_OK / \_STAT\_ERROR
    - **ARM:** HAL\_OK / HAL\_ERROR

Example:

* com\_status = I2C\_Memory\_Erase(0, 65000, 50);

**Requirement**

* i2c\_unit & gpio\_unit drivers for AVR microcontrollers.
* HAL & stm32\_i2c drivers for ARM microcontrollers STM32 series.

**Important tips**

* All commands and settings begin with \_.
* All functions are written as Camel Case.
* The functions and codes used in all microcontrollers are the same.

**Error & Warning's**

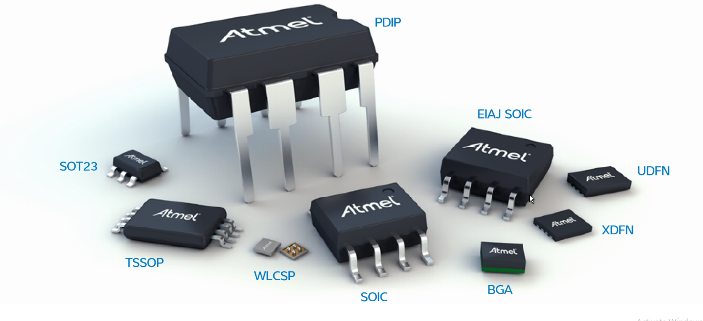
**Error's:**

* **Chip or I2C Library not supported:** This error occurs when the microcontroller or its  
   library not supported.
* **Memory is not selected or not supported:** This error occurs when the memory is not correctly  
   specified or not supported by the library.

**Warning's:**

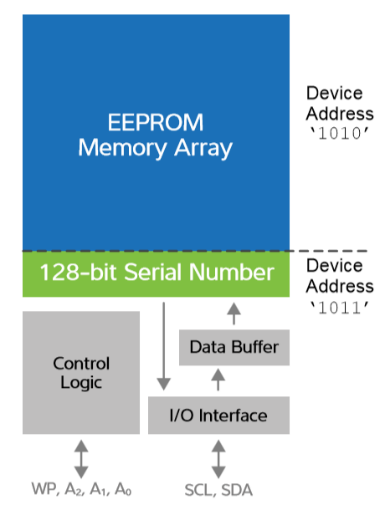
* **Your Ax Pin state in not correct:** This warning appears when the address pins  
   not set correctly.

**Supported memories**



**All 24Cxx family memories:**

**AT24C01 – AT24C02 – AT24C04 – AT24C08 – AT24C16 – AT24C32 – AT24C64 – AT24C128 – AT24C256 – AT24C512 AT24C1024**

****

Memory blocks

**Speed ​​of data writing**

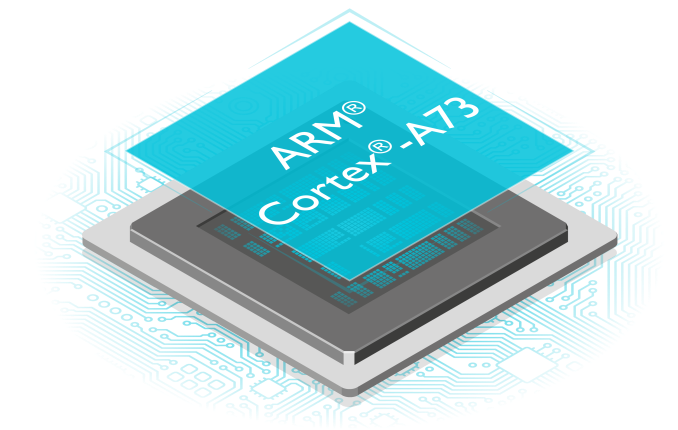
**Supported microcontroller's**



ATmega & ATtiny series of AVR Microcontroller's with i2c\_unit & gpio\_unit driver's

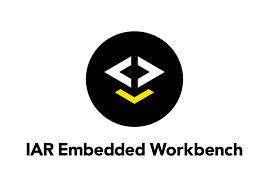
Codevision and GNUC compilers such as AtmelStudio



All STM32 series of ARM Microcontroller's with HAL & stm32\_i2c driver's

All ARM Compiler's



**Version history**

**Version 0.0.1**

Fix: \_MEM\_SIZE\_ERROR in Capacities greater than 256Kb

--------------------------------------------------------------------------------------------------------------------------------------

**Version 0.0.0**

Stable and tested version

--------------------------------------------------------------------------------------------------------------------------------------