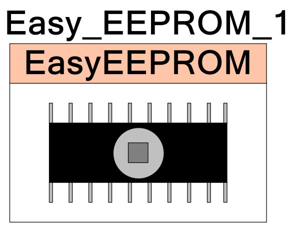
Easy_EEPROM

1.00



Features

- Makes using the EEPROM as simple as possible
- Start, Byte Write and Byte read.

General Description

The Easy EEPROM component simplifies using the EEPROM. Simply start the Easy_EEPROM and use it.

_*note: you must start the Easy_EEPROM component in order to read data from the EEPROM._

Parameters and Setup

None

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "Easy_EEPROM_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

Function	Description
Void Easy_EEPROM_1_Start(void)	This function enables the EEPROM hardware and measures the temperature of the device. You must start the EEPROM in order to read data from it.
Void Easy_EEPROM_1_WriteByte(uint8 value, uint16 address)	Writes the byte value into the EEPROM location specified by address.
Uint8 Easy_EEPROM_1_ReadByte(uint16 address)	Returns a byte from the EEPROM at location specified by address. You must start the EEPROM in order to read data from it.
Void Easy_EEPROM_1_Stop(void)	Disables the EEPROM hardware.

Function	Description
Void Easy_EEPROM_1_UpdateTemp(void)	Call this function before writing to the EEPROM if you believe that the temperature has changed by more than 5 degrees Celsius since the Start() function has been called.

Void Easy_EEPROM_1_Start(void)

Description: This function enables the EEPROM and measures the current device temperature. The EEPROM must be enabled before you can read data from the EEPROM. The temperature of the device is also required before the EEPROM can be written to. If it is possible for the temperature to change by more than 5 degrees Celsius between calling this function and the EEPROM byte writes, you can either call the start function again, or call Easy_EEPROM_1_UpdateTemp().

Parameters: None

Return Value: None

Void Easy_EEPROM_1_WriteByte(uint8 value, uint16 address)

Description: This function writes a single byte into the EEPROM location specified by the address. The address is a zero based index into the EEPROM memory block. The EEPROM has `Easy_EEPROM_1_EEPROM_SIZE` bytes. This is a blocking function call. The function will not return until the write is complete. There are no error codes generated by this function call. You must verify the EEPROM information to ensure the data was written as you intended.

Parameters:

- **Uint8 value:** This is the byte that you want to write into the EEPROM.
- **Uint16 address:** This is the address in the EEPROM you would like to write the byte. The address starts at 0, and goes up to the size of the EEPROM-1. There is a define created with the component, `Easy_EEPROM_1_SIZE` that specifies the size of the EEPROM in bytes. As a simple check, this function will only attempt to write to the EEPROM if the address passed is less than `Easy_EEPROM_1_EEPROM_SIZE`. If you attempt to write to an address beyond the EEPROM, the function will just return and nothing will be written.

Return Value: None

Uint8 Easy_EEPROM_1_ReadByte(uint16 address)

Description: This function reads a single byte from the EEPROM location specified by the address. The address is a zero based index into the EEPROM memory block. The EEPROM has `Easy_EEPROM_1_EEPROM_SIZE` bytes.

Parameters:

- **Uint16 address:** This is the address in the EEPROM you would like to write the byte. The address starts at 0, and goes up to the size of the EEPROM-1. There is a define created with the component, `Easy_EEPROM_1_EEPROM_SIZE` that specifies the size of the EEPROM in bytes. As a simple check, this function will only attempt to write to the EEPROM if the address passed is less than `Easy_EEPROM_1_SIZE`. If you attempt to write to an address beyond the EEPROM, the function will just return and nothing will be written.

Return Value:

- **Uint8 value:** The byte value stored in the EEPROM at the address specified.

Void Easy_EEPROM_1_Stop(void)

Description: This function disables the EEPROM. This will save some power, but you will not be able to read data from the EEPROM while it is powered down.

Parameters: None

Return Value: None

Void Easy_EEPROM_1_UpdateTemp(void)

Description: This function checks the device temperature and updates the global temperature value for the EEPROM writing functions. You only need to call this function if you intend to write to the EEPROM and suspect that the temperature of the device may have changed by more than 5 degrees Celsius.

Parameters: None

Return Value: None

Defines

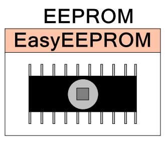
The size in bytes of the EEPROM is provided as:

```
#define Easy_EEPROM_1_EEPROM_SIZE
```

These can be found in the Easy_EEPROM_1.h file

Sample Firmware Source Code

The following example code shows a very simple example of the Easy_EEPROM being used. In this example code, the component instance name is just "EEPROM".



```
#include <device.h>

void main()
{
    uint16 address = 0;
    uint8 temp_write = 0xAA;
    uint8 temp_read;

    EEPROM_Start();

    for(;;)
    {
        EEPROM_WriteByte(temp_write, address);
        temp_read = EEPROM_ReadByte(address);

        if(temp_read == 0xAA)
        {
            // hooray!
        }
    }
}
```

© Cypress Semiconductor Corporation, 2009-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.