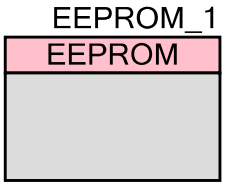
# EEPROM
1.60

## Features




- 512 B to 2 KB EEPROM memory
- 1,000,000 cycles, 20-year retention
- Read 1 byte at a time
- Program 16 bytes at a time

## General Description

The EEPROM component provides an API to write a row (16 bytes) of data to the EEPROM. The term write implies that it will erase and then program in one operation.

The EEPROM component is tightly coupled with various system elements contained within the cy_boot component. These elements are generated upon a successful build. Refer to the *System Reference Guide* for more information about the cy_boot component and its various elements.

### When to Use an EEPROM

You can use an EEPROM component:

- For additional storage of data (freeing up on-chip RAM)
- For read-only (or rarely-changing) program data
- For data that must survive power cycles (for example, calibration tables or device configuration)

## Input/Output Connections

There are no I/O connections for the EEPROM component. It is an API only.

## Component Parameters

The EEPROM has no configurable parameters other than standard Instance Name and Built-in parameters.

# Resources

| Analog Blocks | Digital Blocks | | | | | API Memory (Bytes) | | Pins (per External I/O) |
|---|---|---|---|---|---|---|---|---|
| | Datapaths | Macro cells | Status Registers | Control Registers | Counter7 | Flash | RAM | |
| N/A | N/A | N/A | N/A | N/A | N/A | 724 | 0 | N/A |

# 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.

You do not need APIs to read from the EEPROM. The entire contents of the EEPROM are mapped into memory space and can be read directly. Use the following defines for reading EEPROM:

- CYDEV_EE_BASE – The base pointer of the EEPROM memory
- CYDEV_EE_SIZE – The size of the EEPROM memory space in bytes

EEPROM_1_EEPROM_SIZE is also defined as the size of the EEPROM memory space in bytes (where EEPROM_1 is the instance name of the EEPROM component).

To write to the EEPROM, you must first acquire the die temperature by calling the CySetTemp() and CySetFlashEEBuffer() functions. You only need to acquire the temperature once to use the write functions. If the application will be used in an environment where the die temperature changes 20 °C or more, the temperature should be refreshed to allow the Smart Write Algorithm to work correctly.

By default, PSoC Creator assigns the instance name “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. For readability, the instance name used in the following table is “EEPROM.”

| Function | Description |
|---|---|
| EEPROM_Enable() | Enables EEPROM block operation |
| EEPROM_Start() | Starts EEPROM |
| EEPROM_Stop() | Stops and powers down EEPROM |
| EEPROM_EraseSector() | Erases an EEPROM sector |
| EEPROM_Write() | Blocks while writing a row to EEPROM |

| Function | Description |
|---|---|
| EEPROM_StartWrite() | Starts writing a row of data to EEPROM |
| EEPROM_QueryWrite() | Checks the state of a write to EEPROM |

## void EEPROM_Enable(void)

**Description:** Enables EEPROM block operation. This API is available only for PSoC 3 Production or later.

**Parameters:** None

**Return Value:** None

**Side Effects:** A call to EEPROM_Start() calls EEPROM_Enable(). You can call either EEPROM_Start() or EEPROM_Enable() directly; both have the same effect.

## void EEPROM_Start(void)

**Description:** Starts the EEPROM. This API is available only for PSoC 3 Production or later.

**Parameters:** None

**Return Value:** None

**Side Effects:** A call to EEPROM_Start() calls EEPROM_Enable(). You can call either EEPROM_Start() or EEPROM_Enable() directly; both have the same effect.

## void EEPROM_Stop(void)

**Description:** Stops and powers down the EEPROM. This API is available only for PSoC 3 Production or later.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## cystatus EEPROM_EraseSector(uint8 sectorNumber)

**Description:** Erases a sector (64 rows) of memory. This function blocks until the operation is complete. This API is available only for PSoC 5 and PSoC 3 Production or later.

**Parameters:** uint8 sector. Sector number to erase

**Return Value:** CYRET_SUCCESS if the operation was successful
CYRET_BAD_PARAM if the parameters were invalid
CYRET_LOCKED if the EEPROM control block is busy
CYRET_UNKNOWN if there was an EEPROM control block error

**Side Effects:** None

## cystatus EEPROM_Write(uint8 * rowData, uint8 rowNumber)

**Description:** Writes a row (16 bytes) of data to the EEPROM. This is a blocking call. It will not return until the function succeeds or fails.

**Parameters:** uint8 * rowData. Address of the data to write to the EEPROM

uint8 rowNumber. EEPROM row number to program

**Return Value:** CYRET_SUCCESS if the operation was successful
CYRET_BAD_PARAM if the parameters were invalid
CYRET_LOCKED if the EEPROM control block is busy
CYRET_UNKNOWN if there was an EEPROM control block error

**Side Effects:** None

## cystatus EEPROM_StartWrite(uint8 * rowData, uint8 rowNumber)

**Description:** Starts the SPC write function. This function does not block; it returns after the command has begun the SPC write function. After this function is called, the SPC is locked until EEPROM_QueryWrite() does not return CYRET_STARTED. To abandon the write, call CySpcUnlock() to unlock the SPC.

**Parameters:** uint8 * rowData. Address of the data to write to the EEPROM

uint8 rowNumber. EEPROM row number to program

**Return Value:** CYRET_STARTED if the command to write was successfully started
CYRET_BAD_PARAM if the parameters were invalid
CYRET_LOCKED if the EEPROM control block is busy
CYRET_UNKNOWN if there was an EEPROM control block error

**Side Effects:** None

### cystatus EEPROM_QueryWrite(void)

**Description:** Checks the state of a write to EEPROM. This function must be called until the return value is not CYRET_STARTED.

**Parameters:** void

**Return Value:** CYRET_SUCCESS if the operation was successful
CYRET_STARTED if the command to write was successfully started
CYRET_UNKNOWN if there was an EEPROM control block error

**Side Effects:** None

## Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

## References

See the Die Temperature component datasheet and the *System Reference Guide*.

## DC and AC Electrical Characteristics for PSoC 3

The following values indicate expected performance and are based on initial characterization data.

### DC Specifications

| Parameter | Description | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Erase and program voltage | | 1.71 | -- | 5.5 | V |

### AC Specifications

| Parameter | Description | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $T_{WRITE}$ | Single row erase/write cycle time | | -- | 2 | 15 | ms |
| | EEPROM endurance | | 1 M | -- | -- | program/ erase cycles |
| | EEPROM data retention time | Retention period measured from last erase cycle (up to 100 K cycles) | 20 | -- | -- | years |

## DC and AC Electrical Characteristics for PSoC 5

The following values indicate expected performance and are based on initial characterization data.

### DC Specifications

| Parameter | Description | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Erase and program voltage | | 2.7 | -- | 5.5 | V |

### AC Specifications

| Parameter | Description | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $T_{WRITE}$ | Single row erase/write cycle time | | -- | 8.3 | 32 | ms |
| | EEPROM data retention time, retention period measured from last erase cycle | Average ambient temp, $T_A \leq$ 55 °C, 1 M erase/program cycles | 20 | -- | -- | years |

# Component Changes

This section lists the major changes in the component from the previous version.

| Version | Description of Changes | Reason for Changes / Impact |
|---|---|---|
| 1.60 | Minor code changes in the APIs EEPROM_Write(), EEPROM_StartWrite() and EEPROM_QueryWrite(). | For better performance. |
| | Code changes in the EEPROM_ EraseSector() API to support PSoC 5. | PSoC 5 silicon supports the Erase Sector command. |
| | EEPROM_EraseSector() API description changes in the datasheet | |
| | Added PSoC 5 characterization data to datasheet | |
| 1.50.b | Added explanation of how to acquire temperature to datasheet. | Clarity |
| 1.50.a | Added characterization data to datasheet | |
| | Noted in EEPROM_EraseSector() API in datasheet that it is only available for PSoC 3 Production or later. | |
| | Minor datasheet edits and updates | |
| 1.50 | Modified the *EEPROM.c* file to switch the include file from *cydevice.h* file to *cydevice_trm.h*. | The *cydevice.h* file is obsolete. So the generated source and APIs provided with PSoC Creator should use *cydevice_trm.h* instead. |
| | Updated the EEPROM_EraseSector() section of the example code. | The Erase Sector command does not function on EEPROM for PSoC 3 ES1 and ES2 silicon or on PSOC 5 silicon. This API can be used on EEPROM for PSOC 3 Production or later. |
| | Added EEPROM_Enable(), EEPROM_Start(), and EEPROM_Stop() APIs. | To support PSoC 3 Production silicon requirement that the EEPROM is powered off by default.<br>A call to EEPROM_Start() will call EEPROM_Enable(). You can call either EEPROM_Start() or EEPROM_Enable() function directly; both have the same effect. |
| 1.20.a | Moved component into subfolders of the component catalog. | |
| | Added information to the component that advertizes its compatibility with silicon revisions. | The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device. |
| 1.20 | Updated the Configure dialog. | Digital Port was changed to Pins component in the schematic. |

© Cypress Semiconductor Corporation, 2011. 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.