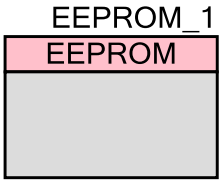
EEPROM
1.50

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(uint16 sectorNumber)

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

Parameters: uint16 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_TIMEOUT if the operation timed out
CYRET_UNKNOWN if there was an EEPROM control block error

Side Effects: None

cystatus EEPROM_Write(uint8 * rowData, uint16 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

uint16 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_TIMEOUT if the operation timed out
CYRET_UNKNOWN if there was an EEPROM CONTROL BLOCK error

Side Effects: None

cystatus EEPROM_StartWrite(uint8 * rowData, uint16 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

uint16 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_TIMEOUT if the operation timed out
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_BAD_PARAM if the parameters were invalid
CYRET_LOCKED if the EEPROM control block is busy
CYRET_STARTED if the command to write was successfully started
CYRET_TIMEOUT if the operation timed out
CYRET_UNKNOWN if there was an EEPROM control block error

Side Effects: None

Sample Firmware Source Code

PSoC Creator provides numerous 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

Refer also to the Die Temperature component datasheet and the *System Reference Guide*.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and 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

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.50.d	Minor datasheet edit.	
1.50.c	Minor datasheet edit.	
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. 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.	

Version	Description of Changes	Reason for Changes / Impact
	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, 2010-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.