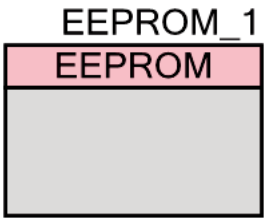
EEPROM
2.10



Features

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

General Description

The EEPROM component provides a set of APIs to erase and write data to nonvolatile EEPROM memory. The term write implies that it will erase and then program in one operation.

An EEPROM memory in PSoC devices is organized in arrays. PSoC 3 and PSoC 5 devices offer an EEPROM array of size 512 bytes, 1 KB or 2 KB depending on the device. This array is divided into rows of size 16 bytes each. The API set of the EEPROM component supports write operations at the byte and row levels and erase operation at the sector level. A sector in EEPROM has 64 rows.

The EEPROM memory is not initialized by the EEPROM component: the initial state of the memory is defined at device datasheet.

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

An EEPROM component can be used for the below purposes:

- 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.

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.

There is no need of APIs to read from the EEPROM. The entire content of the EEPROM is mapped into memory space and can be read directly. EEPROM allows read access at the byte level. The following defines are used 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
- CYDEV_EEPROM_ROW_SIZE – The size of a row of the EEPROM in bytes

Using the base pointer, individual bytes of the EEPROM memory can be read. To navigate to a specific row the base pointer must be incremented by the number of times of the size of the EEPROM row.

SIZEOF_EEPROM_ROW define can be used instead of CYDEV_EEPROM_ROW_SIZE.

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).

It is necessary to acquire the die temperature by calling the CySetTemp() before a series of EEPROM write operations. The CySetTemp() function queries SPC for the die temperature and stores it in a global variable, which is used while performing EEPROM write operations. If the application is used in an environment where the die temperature changes 5°C or more, the temperature should be refreshed to adjust the write times for the optimal performance.

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

Function	Description
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
EEPROM_StartWrite()	Starts writing a row of data to EEPROM
EEPROM_QueryWrite()	Checks the state of a write to EEPROM
EEPROM_ByteWrite()	Writes a byte of data to EEPROM

void EEPROM_Enable(void)

Description: Enables EEPROM block operation. This API is not available for PSoC 5 silicon.

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 has to be called before using write/erase APIs and reading the EEPROM. This API is not available for PSoC 5 silicon.

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 not available for PSoC 5 silicon.

Parameters: None

Return Value: None

Side Effects: None

cystatus EEPROM_EraseSector(uint8 sectorNumber)

Description: Erases a sector (64 rows) of memory by making the bits zero. This function blocks until the operation is complete.

Parameters: uint8 sector. Sector number to erase

Return Value: CYRET_SUCCESS if the operation was successful
CYRET_BAD_PARAM if the parameter out of range
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(const 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: const 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 parameter out of range
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(const uint8 * rowData, uint8 rowNumber)

Description: Starts the SPC write function. This function does not block; it returns once the command has begun the SPC write function. This function must be used in combination with EEPROM_QueryWrite(). Once this function has been called, the SPC will be locked until EEPROM_QueryWrite() returns CYRET_SUCCESS.

Parameters: const 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 parameter out of range
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 write to EEPROM. This function must be called until the return value is not CYRET_STARTED.

Parameters: None

Return Value: CYRET_SUCCESS if the operation was successful
CYRET_STARTED if the command to write is still processing
CYRET_UNKNOWN if there was an EEPROM control block error

Side Effects: None

cystatus EEPROM_ByteWrite(unit8 dataByte, uint8 rowNumber, uint8 byteNumber)

Description: Writes a byte of data to EEPROM. This is a blocking call. It will not return until the function succeeds or fails.

Parameters: uint8 dataByte. Byte of data to write to EEPROM
uint8 rowNumber. EEPROM row number to program
uint8 byteNumber. Byte number within the row to program

Return Value: CYRET_SUCCESS if the operation was successful
CYRET_BAD_PARAM if the parameters out of range
CYRET_LOCKED if the EEPROM CONTROL BLOCK is busy
CYRET_UNKNOWN if there was an EEPROM control block error

Side Effects: None

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The EEPROM component does not have any specific deviations.

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*.

Resources

The EEPROM component uses EEPROM capability of the device.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.
The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5 (GCC)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Default	768	0	552	0	524	0

DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted.
Specifications are valid for 1.71 V to 5.5 V, except where noted.

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	20	ms
	EEPROM data retention time, retention period measured from last erase cycle	Average ambient temp, $T_A \leq 25°C$, 1M erase/program cycles	20	--	--	years
		Average ambient temp, $T_A \leq 55°C$, 100K erase/program cycles	20	--	--	
		Average ambient temp, $T_A \leq 85°C$, 10K erase/program cycles	10	--	--	

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.10	Added MISRA Compliance section.	The component does not have any specific deviations.
2.0.a	Updated AC and DC characteristics section.	Keeping AC and DC characteristics aligned with the device datasheet.
	Added information about EEPROM memory initial state.	Clarify the component operation and refer to the device datasheet for the initial state of the EEPROM memory.
2.0	Added support for PSoC 5LP silicon. Added new API EEPROM_ByteWrite().	To support the byte write capability.
	Codes changes in EEPROM_Write(), EEPROM_StartWrite(), EEPROM_QueryWrite() and EEPROM_EraseSector().	To support the SPC code changes.
	Removed CySetTemp() calls from EEPROM_Write() and EEPROM_StartWrite() APIs.	For better performance.
1.60	Minor code changes in the APIs EEPROM_Write(), EEPROM_StartWrite() and EERPOM_QueryWrite().	For better performance.
	Code changes in the EEPROM_ EaraseSector() API to support PSoC5.	PSoC 5 silicon supports the Erase Sector command.
	EEPROM_EraseSector() API description changes in the 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.

Version	Description of Changes	Reason for Changes / Impact
	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.	
	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, 2010-2012. 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.