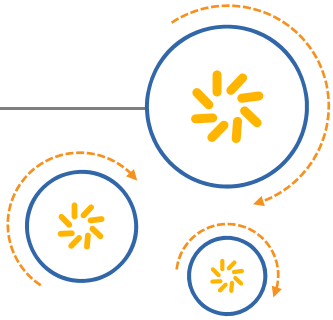
Qualcomm Technologies International, Ltd.



Confidential and Proprietary – Qualcomm Technologies International, Ltd.

(formerly known as Cambridge Silicon Radio Ltd.)

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies International, Ltd.or its affiliated companies without the express approval of Qualcomm Configuration Management.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies International, Ltd.

Any software provided with this notice is governed by the Qualcomm Technologies International, Ltd. Terms of Supply or the applicable license agreement at https://www.csrsupport.com/CSRTermsandConditions.

Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. All Qualcomm Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies International, Ltd.
Churchill House
Cambridge Business Park
Cambridge, CB4 0WZ
United Kingdom

Push every boundary.®

CSR µEnergy®

Interfacing Large Serial Flash and EEPROM

Application Note

Issue 3

Document History

Revision	Date	History
1	14 JAN 15	Original publication of this document
2	15 JAN 15	Correct SPI Flash interface tables
3	03 AUG 15	Minor clarification

Contacts

General information	www.csr.com
Information on this product	sales@csr.com
Customer support for this product	www.csrsupport.com
More detail on compliance and standards	product.compliance@csr.com
Help with this document	comments@csr.com

Trademarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc and/or its affiliates.

Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR.

Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc or its affiliates.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Use of this document is permissible only in accordance with the applicable CSR licence agreement.

Safety-critical Applications

CSR's products are not designed for use in safety critical devices or systems such as those relating to: (i) life support; (ii) nuclear power; and/or (iii) civil aviation applications, or other applications where injury or loss of life could be reasonably foreseeable as a result of the failure of a product. The customer agrees not to use CSR's products (or supply CSR's products for use) in such devices or systems.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.

Contents

Tables and Figures

1. Introduction

CSR µEnergy CSR101x devices use a single SPI Flash memory or I²C EEPROM device for storing application code and data. Up to version 2.4.3 of the CSR µEnergy Software Development Kit (SDK), the maximum SPI Flash memory and I²C EEPROM device size supported was 512-kbit.

This application note describes the methods used to support higher memory sizes in SDK version 2.4.4 and later.

Note:

The term CSR101x is used in this document to refer to all of CSR1010, CSR1011, CSR1012 and CSRB31010 devices.

1.1. Supported Large Memory Parts

CSR101x devices are compatible and have been tested with the following memory parts for sizes above 512-kbit, see Table 1.1:

Manufacturer	Part	Size	Type
Atmel	AT24CM01	1-Mbit	I²C EEPROM
Adesto	AT25DF011	1-Mbit	SPI Flash memory
Macronix	MX25L4006E	4-Mbit	SPI Flash memory

Table 1.1: Supported Large Memory Parts

2. Hardware

2.1. I²C EEPROM Configuration

Table 2.1 lists the CSR101x I²C EEPROM interface pins:

CSR101x Chip Pinout	Pin details
PIO[2]	Programmable Input / Output line used for driving the VCC of EEPROM
I2C_SCL/SF_CLK	I²C Clock (internal pullup)
I2C_SDA/SF_DOUT	I²C Data Input / Output (internal pullup)

Table 2.1: I²C EEPROM Interface

Figure 2.1 shows a CSR101x device connected through I²C to an external EEPROM:

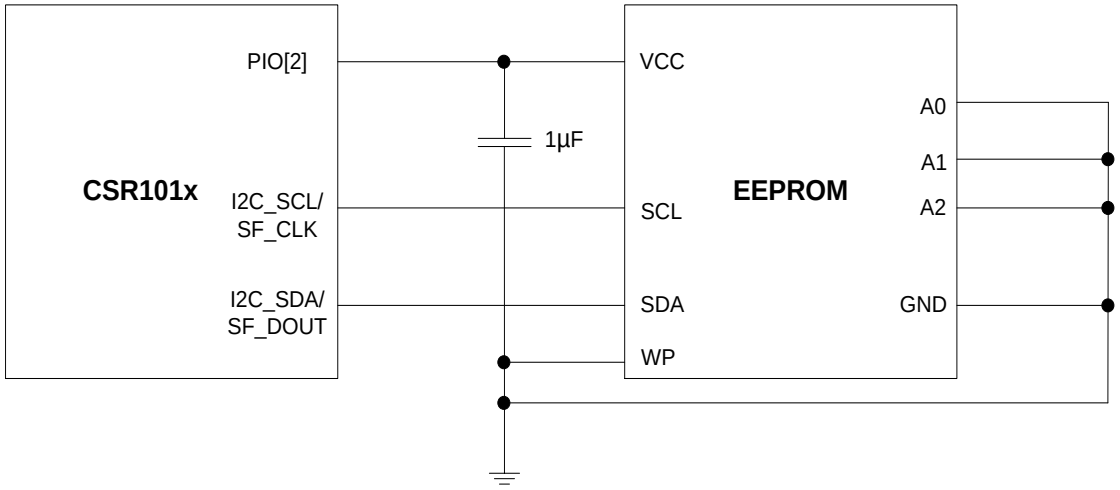


Figure 2.1: I²C EEPROM Interface

For example consider the AT24CM01, a 1-Mbit EEPROM which has the I²C Device Address as shown in Table 2.2:

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
1	0	1	0	A2	A1	P0	R/W

Table 2.2: I²C Device Address (AT24CM01)

In this device the memory is organised as two pages of 512-kbit each. To access the memory in the first page, P0 is set to 0, and two 8-bit data words are used to address a particular 8-bit data word within the page. To access the memory in the second page, P0 is set to 1, i.e. P0 acts as an A16 address line.

The Firmware Library API distributed in the SDK provides a method to specify the I²C Device Address which can be used to address separate 512-kbit pages in the I²C EEPROM device. See the I²C Serial Interface module in the *SDK Reference Documentation* for more information.

2.2. SPI Flash Memory Configuration

Table 2.3 lists the CSR101x SPI Flash memory interface pins:

CSR101x Chip Pinout	Pin Details
PIO[2]	Programmable I/O line used for driving the VCC of SPI Flash memory
PIO[3]/SF_DIN	SPI Master Output Slave Input
PIO[4]/SF_CS#	SPI Chip Enable Active Low
I2C_SCL/SF_CLK	SPI Clock
I2C_SDA/SF_DOUT	SPI Master Input Slave Output

Table 2.3: SPI Flash Memory Interface

Figure 2.2 shows a reference SPI interface connected to a 512-kbit SPI Flash memory:

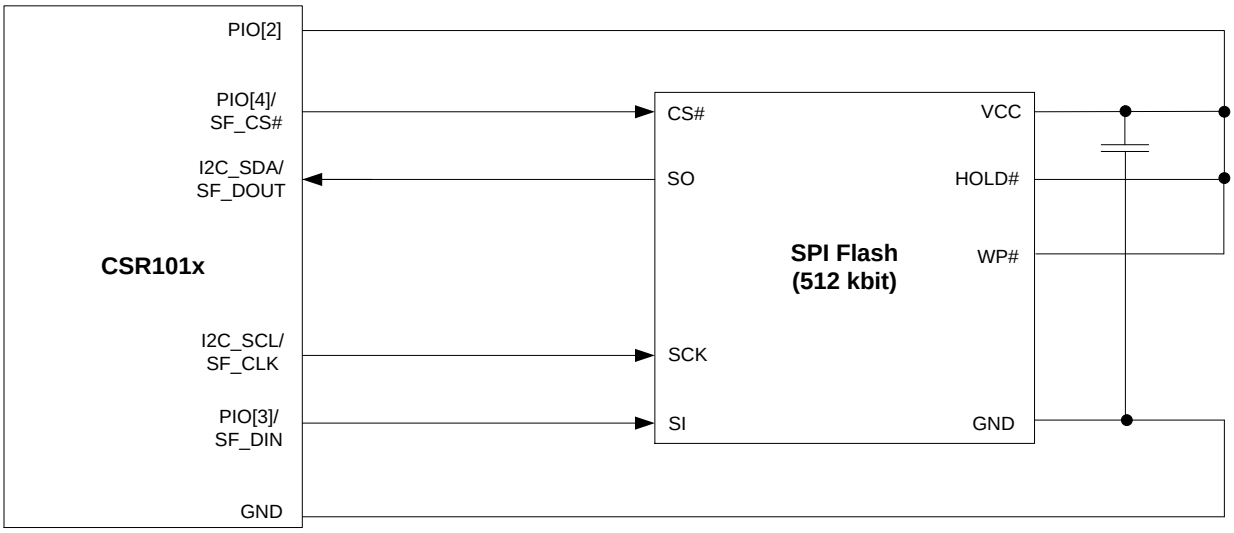


Figure 2.2: SPI Flash Memory Interface

Table 2.4 lists the CSR101x extended SPI Flash memory interface pins:

Pin details	Description
PIO[2]	Programmable I/O line used for driving the VCC of SPI Flash memory
PIO[3]/SF_DIN	SPI Master Output Slave Input
PIO[4]/SF_CS#	SPI Chip Enable Active Low
I2C_SCL/SF_CLK	SPI Clock
I2C_SDA/SF_DOUT	SPI Master Input Slave Output
PIO[X]	Any unused Programmable I/O from PIO[0] to PIO[15]
PIO[Y]	Any unused Programmable I/O from PIO[0] to PIO[15]

Table 2.4: Extended SPI Flash Memory Interface

Figure 2.3 shows the modifications required in the SPI interface connected to a Large SPI Flash memory:

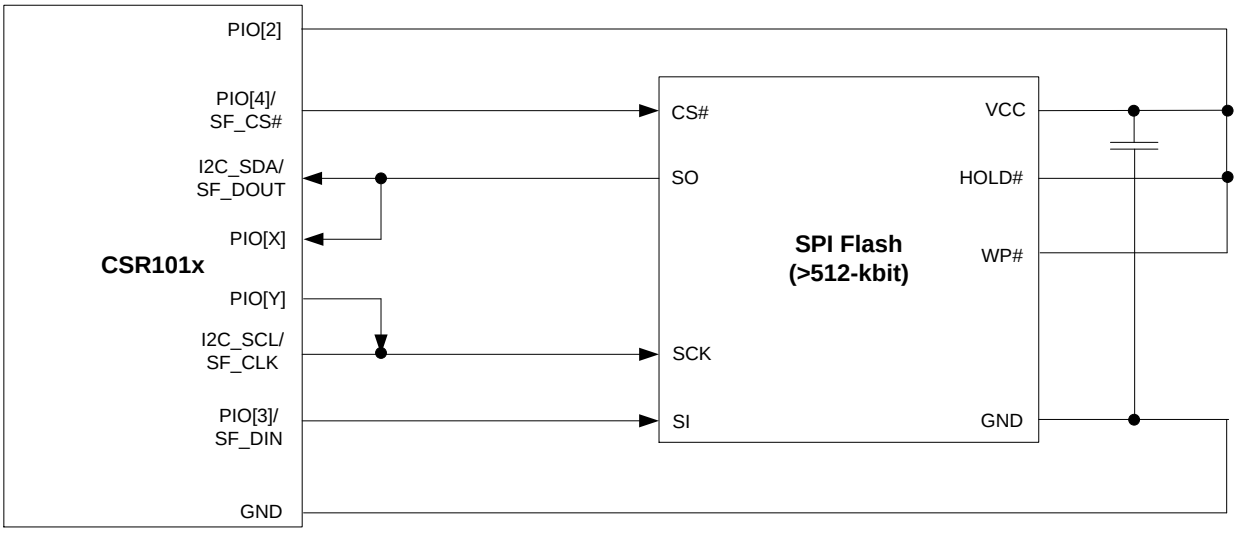


Figure 2.3: Extended SPI Flash Memory Interface

2.2.1. Additional PIOs for SPI Flash Memory Interface

The SDK tool chain and Firmware Library SPI API use the hardware memory interface of the CSR101x to access the external SPI Flash memory device. The hardware can only address 16-bit memory addresses, limiting the area of access to 512-kbit. A software SPI Flash memory driver is required to access memory above the 16-bit address space using four PIOs.

As shown in Figure 2.2, only SF_DIN and SF_CS# are accessible by software, through PIO[3] and PIO[4] respectively. The software cannot access SF_CLK or SF_DOUT as these two lines are not accessible by any of the PIOs. To allow the software access to these lines, two additional PIOs PIO[X] and PIO[Y] are required. These PIOs need to be connected to the SF_DOUT and SF_CLK lines as shown in Figure 2.3.

The PIOs are grouped into two banks of 16 (for CSR101x devices with 16 PIOs or fewer only the first bank is used). The first bank maps PIO[0] through PIO[15] and the second bank maps PIO[16] through PIO[31]. All four SPI PIOs must be in the same bank. Since PIO[3] and PIO[4] are used for the MOSI and chip select signals this means the two additional PIOs must be in the range of PIO[0] to PIO[15].

See the SPI Serial Interface module in the *SDK Reference Documentation* for more information.

3. Application

3.1. Standard Memory Layout

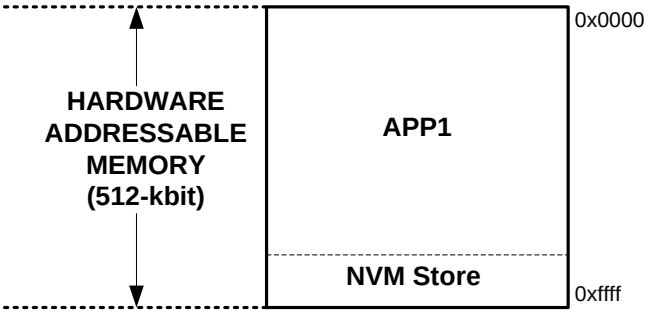


Figure 3.1: Standard Memory Layout

Figure 3.1 shows the Standard Memory Layout for CSR101x devices. Applications smaller than 512-kbit (including NVM Store) can follow this model.

3.1.1. Limitations

- The boot up sequence for CSR101x is controlled by the hardware which loads the application image from external storage into RAM. The hardware is limited to a 16-bit address space in NVM, therefore the application code and data must reside within the first 512-kbit.
- The NVM Store used by applications to store non-volatile data is accessed using the Firmware Library API which is also limited to a 16-bit address space. Therefore the NVM Store must also be within the first 512-kbit.

3.2. Extended Memory Layout

The Standard Memory Layout may be extended to access non-volatile memory addresses above the 512-kbit boundary to accommodate larger applications and multiple images, see Figure 3.2:

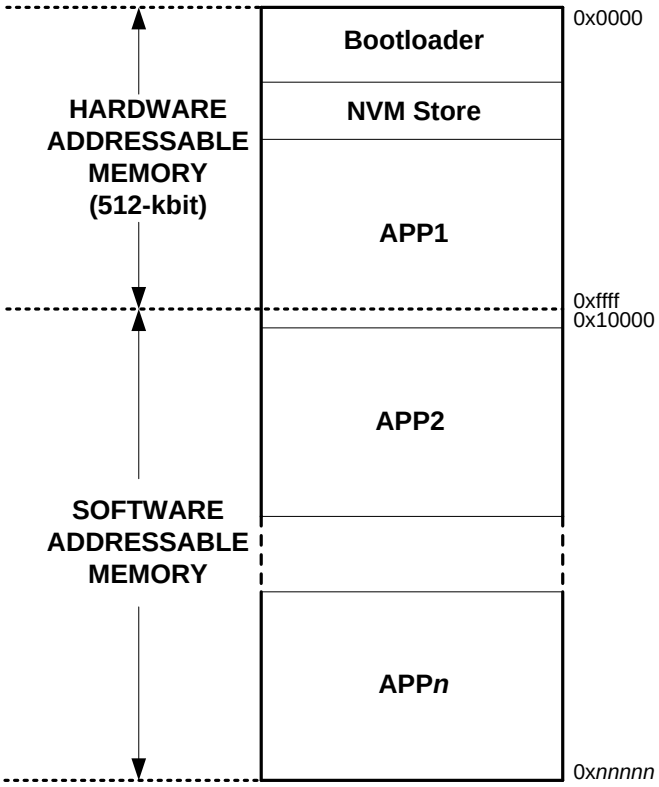


Figure 3.2: Extended Memory Layout

- A Bootloader is required to access applications stored above the 512-kbit boundary. The Bootloader uses a software SPI/I^{2}C driver instead of the Firmware Library API to access NVM above 512-kbit.
- The Bootloader and the NVM Store (shared by all applications) must reside within the first 512-kbit of NVM.
- Application images may be stored above the 512-kbit boundary.

Note:

The NVM Store is a non-volatile storage area accessible by applications. It is up to the applications to organise the arrangement of stored data within the NVM Store so that data is not corrupted.

3.3. NVM Store Configuration

The CS keys used to configure NVM Store are the same for both the Standard Memory Layout and the Extended Memory Layout, only the location of the NVM Store changes, see Figure 3.1 and Figure 3.2.

Two CS Keys are used to configure the start address and size of the NVM Store, see Table 3.1:

CS Key	Description
`nvm_start_address`	The address in NVM at which the NVM Store starts, in octets
`nvm_size`	The size of the NVM Store in 16-bit words.

Table 3.1: NVM Store CS Keys

In the Standard Memory Layout, `nvm_start_address + (nvm_size * 2)` must not exceed the size of the NVM device, and `nvm_start_address` must be above the application image.

In the Extended Memory Layout, `nvm_start_address + (nvm_size * 2)` must not exceed the start address of the first application image, and `nvm_start_address` must be above the Bootloader.

In any case the NVM Store area must not cross the 512-kbit boundary, i.e. `nvm_start_address + (nvm_size * 2) <= 0x10000`

Example Case 1

Consider a case where the NVM Store needs to start at `0xff80` and have a storage capacity of `128` octets. For this case, the CS keys in the `.keyr` file would be defined as shown below:

```
&nvm_start_address = ff80     // Byte offset to NVM Store in octets in hex
&nvm_size = 40                // NVM Store size in 16-bit words in hex
```

3.3.1. SPI Flash Memory

When using SPI Flash memory, the firmware uses a more complicated allocation system for NVM Store because unlike I²C EEPROM devices, SPI Flash memory devices can only erase entire blocks at a time. This means an entire block needs to be reserved for the NVM Store, even if the size of the NVM Store required is smaller than the block.

Therefore SPI Flash memory devices require two additional CS keys to configure NVM Store, see Table 3.2:

CS Key	Description
`spi_flash_block_size`	The size of a SPI Flash memory block in octets
`nvm_num_spi_blocks`	The number of SPI Flash memory blocks allocated to NVM Store (maximum of 2)

Table 3.2: SPI Flash Memory CS Keys

In the Standard Memory Layout, `nvm_start_address + (spi_flash_block_size * nvm_num_spi_blocks)` must not exceed the size of the NVM device, and `nvm_start_address` must be above the application image.

In the Extended Memory Layout, `nvm_start_address + (spi_flash_block_size * nvm_num_spi_blocks)` must not exceed the start address of the first application image, and `nvm_start_address` must be above the Bootloader.

In any case the NVM Store area must not cross the 512-kbit boundary, i.e. `nvm_start_address + (spi_flash_block_size * nvm_num_spi_blocks) <= 0x10000`

There are some additional limitations on the values that may be assigned to `nvm_start_address` and `nvm_size` when using SPI Flash memory for NVM Store:

- `nvm_start_address` must align with the start of a SPI Flash memory block. For example, if the SPI Flash memory device has 4KB blocks, then `nvm_start_address` must be a multiple of 4096.
- `spi_flash_block_size` must be divisible by `nvm_size*2`. For example, if `spi_flash_block_size` is `4096` and `nvm_size` is `64`, i.e. 128 octets, this splits the block into exactly 32 pages: 4096/128 = 32
- `nvm_size*2` must be smaller than `spi_flash_block_size`, i.e. each block must contain at least 2 pages. For example, if `spi_flash_block_size` is `4096`, then the largest `nvm_size` is `1024`.

Each SPI Flash memory block is divided into pages, where each page is `nvm_size*2` octets long. Whenever the NVM Store contents are modified, the page containing the NVM Store is copied into a new page with the modifications applied. When all the pages in a SPI Flash memory block have been written the firmware will copy the page to the start of a new SPI Flash memory block and erase the old block.

If only one SPI Flash memory block has been configured then the application has to handle the block erase cycle. In this case, if the application attempts to write to the NVM Store but all the pages have been written, the firmware will return an error. The application should then read the NVM Store into RAM, erase the SPI Flash memory block, then write the modified NVM Store back.

Example Case 2

Consider a case where the number of SPI Flash memory blocks is 2, the SPI Flash memory block size is 4096 octets, and the NVM Store starts at 0xe000. Let each SPI Flash memory block comprise pages of 128 octets each, allowing 32 pages per block (4096/128 = 32).

For this case, the CS keys in the `.keyr` file would be defined as shown below:

```
&nvm_start_address = e000     // Byte offset to NVM Store in octets in hex
&nvm_size = 40                // NVM Store size in 16-bit words in hex
&spi_flash_block_size = 1000 // SPI Flash block size in octets in hex
&nvm_num_spi_blocks = 2       // Two blocks reserved for NVM Store in hex
```

Example Case 3

Consider a case where the number of SPI Flash memory blocks is 1, the SPI Flash memory block size is 4096 octets, and the NVM Store starts at 0xf000. Using a page size of 256 octets, the CS keys in the `.keyr` file would be defined as follows:

```
&nvm_start_address = f000     // Byte offset to NVM Store in octets in hex
&nvm_size = 80                // NVM Store size in 16-bit words in hex
&spi_flash_block_size = 1000 // SPI Flash block size in octets in hex
&nvm_num_spi_blocks = 1       // One block reserved for NVM Store in hex
```

4. NVM Tools

Two command line tools are provided in the SDK that program NVM devices: `e2cmd` for I²C EEPROM, and `nvscmd` for SPI Flash memory.

Prior to SDK 2.4.4 these tools only used the hardware memory interface of the CSR101x, which is limited to a 16-bit address space, i.e. only the first 512-kbit in the NVM may be accessed.

From SDK 2.4.4 these tools use a custom software driver to access addresses beyond 512-kbit on NVM devices.

There is no change in how `e2cmd` is invoked when programming I²C EEPROMs larger than 512-kbit, but `nvscmd` requires an additional command line option, `-pioflash <CLOCK> <MISO>`, to specify which PIOs are used to access the SF_CLK and SF_DOUT lines respectively on SPI Flash memory devices larger than 512-kbit. For example:

```
nvscmd -pioflash 9 10 burn flash_image.xuv
```

where:

`<CLOCK>` corresponds to PIO[9] and is connected to SF_CLK

`<MISO>` corresponds to PIO[10] and is connected to SF_DOUT

`flash_image.xuv` is the name of the image file to download to the device.

There is no change in how `e2cmd` or `nvscmd` are invoked when accessing devices that are 512-kbit or smaller in size.

Passing option `-help` to either tool will display a list of supported options.

4.1. Limitations

Accessing SPI Flash memory / I²C EEPROM addresses above 512-kbit uses a software addressable method. This method is slower compared to the hardware addressable method.

Document References

Document	Reference
CSR1010 QFN Data Sheet	CS-231985
CSR1011 QFN Data Sheet	CS-231986
CSR1012 QFN Data Sheet	CS-238833
SDK Reference Documentation	Supplied with the CSR µEnergy SDK as the Firmware Library Documentation

Terms and Definitions

API	Application Programming Interface
BLE	Bluetooth Low Energy: a Bluetooth technology designed for ultra-low power consumption
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
Bluetooth Smart	A term used to indicate devices supporting BLE
CS	Configuration Store
CSR	Cambridge Silicon Radio
CSR101x	Any of the CSR1010, CSR1011, CSR1012 or CSRB31010 devices
CSR µEnergy®	Group term for CSR's range of Bluetooth Smart wireless technology chips
EEPROM	Electrically Erasable Programmable Read Only Memory
e.g.	*exempli gratia* (for example)
I²C	Inter-Integrated Circuit
i.e.	*id est* (that is)
I/O	Input/Output
KB	Kilobyte: 1,024 bytes, where 1 byte is 1 octet
kbit	Kilobit: 1,024 bits
LSB	Least Significant Bit
Mbit	Megabit: 1,048,576 bits
MSB	Most Significant Bit
NVM	Non-Volatile Memory
PIO	Programmable Input/Output, also known as general purpose I/O
RAM	Random Access Memory
ROM	Read Only Memory
SDK	Software Development Kit
SPI	Serial Peripheral Interface
WP	Write Protect