

### About this document

### Scope and purpose

This application note describes steps to enable a CY8CPROTO-062-4343W PSoC<sup>™</sup> 6 Wi-Fi Bluetooth<sup>®</sup> prototyping kit to work with an external flash memory by modifying the board hardware and software. By replacing the attached flash with the external connection and adding commands into PSoC<sup>™</sup> 6 MCU serial memory interface (SMIF), this board can be turned into a memory evaluation board for any external flash.

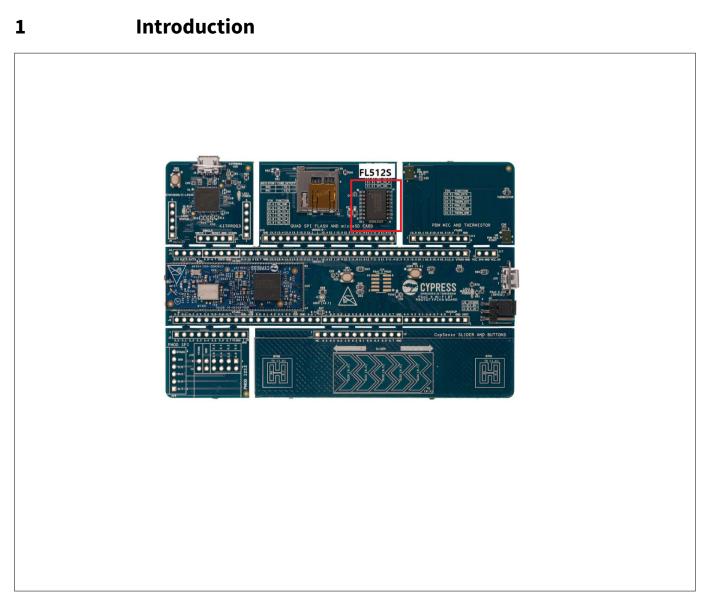
#### **Intended** audience

This application note targets PSoC<sup>™</sup> 6 MCU board users who intend to further investigate operation with external flash memory devices other than the offered S25FL512S flash on the board. This method will free the limitation of the users to evaluate any flash. The flash device in the SOIC-16 package on the board may be replaced freely; the external flash outside the board with any type of package can be tested.

### **Table of contents**

	About this document
	Table of contents    1
1	Introduction
2	Procedures
2.1	Modifying the hardware
2.1.1	Materials required
2.1.2	Customize the board
2.2	Modifying the software
2.2.1	ModusToolbox <sup>™</sup> software
2.2.2	Serial memory interface (SMIF)
2.2.3	Import ModusToolbox <sup>™</sup> QSPI flash code example8
2.2.4	Determine the serial flash operation structure11
2.2.5	Add additional flash operation commands15
3	<b>PSoC<sup>™</sup> 6 MCU board operation</b> 21
4	<b>Use cases</b>
5	<b>Conclusion</b>
	<b>References</b>
	Revision history
	<b>Disclaimer</b>

### 1 Introduction



### Figure 1

### CY8CPROTO-062-4343W PSoC<sup>™</sup> 6 Wi-Fi Bluetooth<sup>®</sup> prototyping board

The PSoC<sup>™</sup> 6 Wi-Fi Bluetooth<sup>®</sup> prototyping kit (CY8CPROTO-062-4343W) is provided with the S25FL512S 512-Mb SPI NOR flash using SMIF. SMIF is an SPI-based communication interface for interfacing external devices to a PSoC<sup>™</sup> MCU device. However, as this PSoC<sup>™</sup> 6 MCU board comes with external flash attached to the board. Because there are several commands to evaluate all flash operations, there are limitations on evaluating flash memories fully. This application note describes a method to modify this PSoC<sup>™</sup> 6 MCU board hardware and software for customizing the board into a general evaluation tool for flash memory devices.

Infineon

2 Procedures

#### **Procedures** 2

Both hardware and software modifications are required for this application.

#### Modifying the hardware 2.1

#### 2.1.1 **Materials required**

SMT socket - wide SOIC-16, pin header Figure 2

- CY8CPROTO-062-4343W PSoC<sup>™</sup> 6 W-Fi Bluetooth<sup>®</sup> prototyping kit board •
- SMT socket-wide SOIC-16
- 8-pin headers
- Cutter
- Soldering machine

#### 2.1.2 **Customize the board**

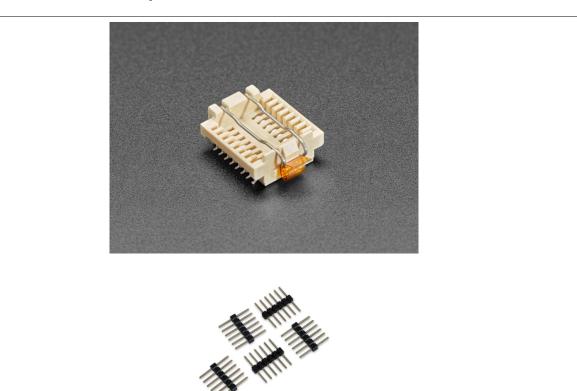
Prepare the board and SMT socket-wide SOIC-16, and 8-pin headers. You may also need a cutter to remove the existing flash device and a soldering machine to solder the socket and pin headers.

- Remove the attached flash with the cutter. Gently cut off the flash lead frames and then remove the 1. leftovers with solders. Be careful not to strip off the PCB pattern
- Fit the SMT socket-wide SOIC-16 into the PCB footprint. The existing flash on the board is in SOIC 16 pin 2. package, so it is compatible with the new socket. Solder the socket along the footprint

3

- Solder the pin headers to GND, VDDIO, pins 11.6, 11.5, 11.7, 11.2, 11.3, and 11.4 pin holes. See Table 1 3. and Figure 3
- Make sure that all pins are connected 4.

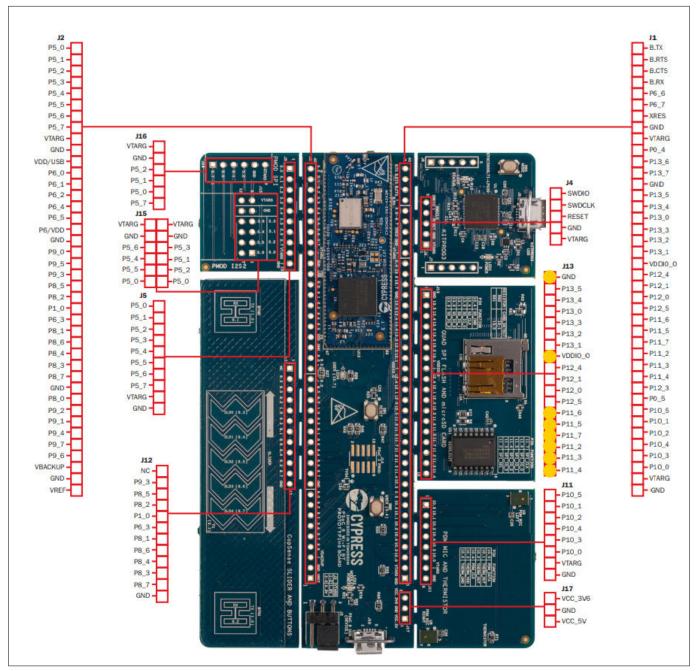






2 Procedures

Table 1	CY8CPROTO-062-4343W pinout				
Pin	Flash	Pin	Flash		
GND	VSS	P11.7	SCK		
VDDIO_O	VCC	P11.2	#CS		
P11.6	SI/IO0	P11.3	HOLD/IO3		
P11.5	SO/IO1	P11.4	WP/IO2		





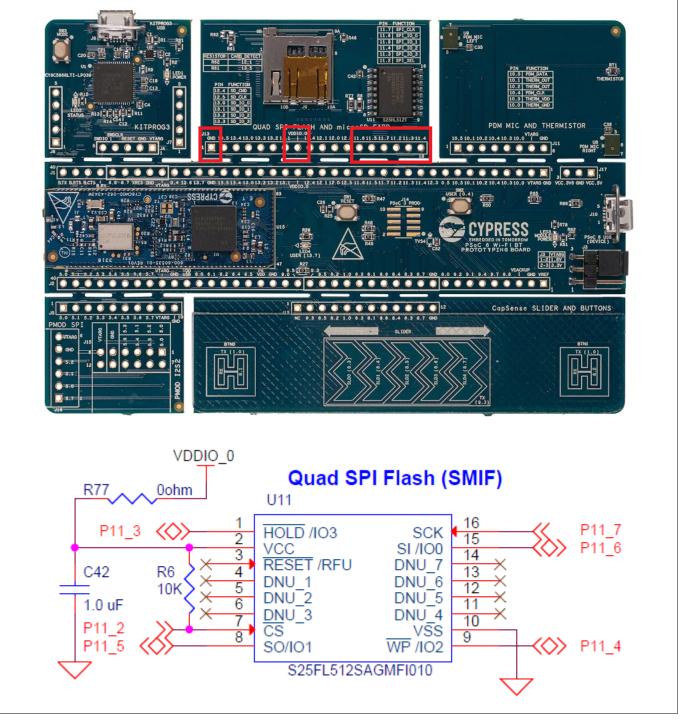
**Board pinout** 

Test the flash devices as follows:



### 2 Procedures

- To test any flash device in SOIC-16 package, replace the flash chip in the socket as shown in Figure 5
- To test flash devices in other package types or to test the flash device that is attached to another board like shown in Figure 6, use the pin header connector. In this case, keep the socket empty because the socket flash signal and external signal through pin header can be overlapped



#### CY8CPROTO-062-4343W external connections

Figure 5 and Figure 6 show the boards with the hardware modifications:

- Figure 5 shows the flash device in the socket being tested
- Figure 6 shows an external flash device on another board being tested. Note that the jumpers are connected to the pin header while the socket is empty

**Figure 4** 

#### Figure 5

**PSoC<sup>™</sup> 6 MCU board with the socket attached** 

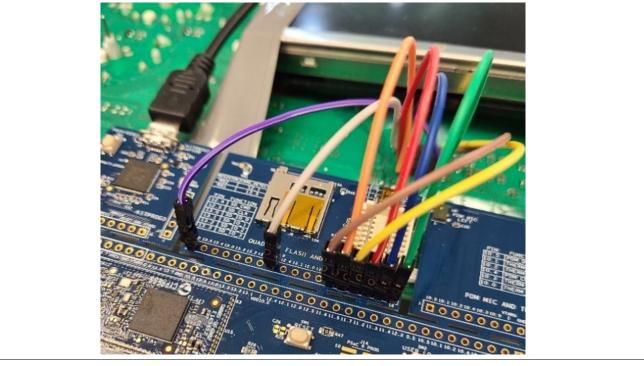


Figure 6

External flash memory connected with a pin header



### 2 Procedures

### 2.2 Modifying the software

### 2.2.1 ModusToolbox<sup>™</sup> software

PSoC<sup>™</sup> 6 MCU boards use Eclipse IDE for ModusToolbox<sup>™</sup> and C language for programming.



### Download the software from https://www.cypress.com/products/modustoolbox#tabs-0-bottom\_side-6.

### 2.2.2 Serial memory interface (SMIF)

odusToolbox

The SMIF Component implements an SPI-based communication hardware block for interfacing external memory devices with the PSoC<sup>™</sup> 6 MCU device. The firmware uses the source code (cy\_smif\_memconfig.c and cy\_smif\_memconfig.h files) generated from the SMIF configurator. This source code defines the data structures that hold the memory configuration.

7

Modern software development

Build, ship and maintain innovative

IoT products in ModusToolbox

with ModusToolbox™





### 2 Procedures

### 2.2.3 Import ModusToolbox<sup>™</sup> QSPI flash code example

- 1. On Eclipse IDE for ModusToolbox<sup>™</sup> software, import example project into your project. Do one of the following:
  - Select File > New > ModusToolbox<sup>™</sup> Application
  - Click New Application on the Start tab on the Quick Panel

	🖌 ModusToolbox - Eclipse IDE for ModusToolbox									
File	Edit	Navigate	Search	Project	Run	Window	Hel	р		
	New					Alt+Shift	+N >		Project	- 5
		n File							Other	Ctrl+N
		Projects f	rom File	System				*	ModusToolbox Application	Ctrl+7
		nt Files					>			
	Close					Ctrl+				
	Close	All				Ctrl+Shift+	W			
	Save					Ctrl	+S			
	Save						_			
	Save					Ctrl+Shift	+S			
	Move						50			
	Rena Refre						F2 F5			
		ert Line De	limiters 1	То			>	Ŀ		
	Print.					Ctr	. D			
	Impo					Cui	TP			
	Expo							Ŀ		
	Prop					Alt+En	tor			
						AICTEN				
		h Workspa	ice				>			
	Resta	irt								
	Exit									
-										
		anel (x)= Va	riables 🤋	Express	ions	• Breakpo	oints			
Mo	ipse I dusT	DE for oolbox®								
- 5	itart									
		Applicatio		1						
		h Online fo		-						
		h Online fo		es and BS	SPS					
-0	Refre	sh Quick P	anel							
* P	Projec	t								Console 🛛

2. Choose the board support package (BSP) for the CY8CPROTO-062-4343W board and then click Next



### 2 Procedures

🔛 Project Creator 1.2 - Choo	ose Board Suppo	ort Package (BSP)	- 🗆 X
Settings Help			
Enter filter text			CY8CPROTO-062-4343W
CY8T-343052-EVAL         CY           CY8T-343052-EVAL         CY           CY8T-413055-EVAL         CY           CY8T-413055-EVAL         CY           CY8T-413055-EVAL         CY           CY8T-413055-EVAL         CY           CY8T-413055-EVAL         CY           CY8T-433054-EVAL         CY           CY8T-230054-EVAL         CY           CY8T-230054-EVAL         CY           CY8T-23017182EVA-02         CY           CY8020717182EVA-03         CY           CY8020717182EVA-03         CY           CY8020717182EVA-03         CY           CY8020717182EVA-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01         CY           CY802071982EV8-01 <t< td=""><td>W20706A2 #7-34055-02 #7-413055-02 #7-413055-02 #7-413055-02 #7-413055-02 #7-423054-02 #7-43054-02 #7-</td><td>Connectivity Device Connectivity Device CW0207364 CW0207364 CW0207364 CW0207364 CW0207364 CW0207362 CW0207362 CW0207362 CW0207362 CW0207362 CW02073162 CW020746 CW02</td><td><ul> <li>The CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit is a low-cost hardware platform that enables design and debug of PSoC 6 MCUs. It comes with a Murata LBEESKLIDX module, based on the CYW4343W combo device, industry-leading CapSense for touch buttons and silder, on-board debugger/programmer with KitProg3, microSD card interface, 512-Mb Quad-SPI NOR flash, PDM-PCM microphone, and a thermistor. This kit is designed with a snap-away form-factor, allowing the user to separate the different components and features that come with this kit and use independently. In addition, support for Digilent's Pmod interface is also provided with this kit.</li> <li>Kit Features:</li> <li>Support of up to 2MB Flash and 1MB SRAM</li> <li>Dedicated SDHC to interface with WICED wireless devices.</li> <li>Delivers dual-cores, with a 150-MHz Arm Cortex-M4 as the primary application processor and a 100-MHz Arm Cortex-M0+ as the secondary processor for low-power operations.</li> <li>Supports Full-Speed USB, capacitive-sensing with CapSense, a PDM-PCM digital microphone interface, a Quad-SPI interface, 13 serial communication blocks, 7 programmable analog blocks, and 56 programmable digital blocks.</li> <li>Kit Contents:</li> <li>PSoC 6 Wi-Fi BT Prototyping Board</li> <li>USB Type-A to Micro-B cable</li> <li>Quick Start Guide</li> </ul></td></t<>	W20706A2 #7-34055-02 #7-413055-02 #7-413055-02 #7-413055-02 #7-413055-02 #7-423054-02 #7-43054-02 #7-	Connectivity Device Connectivity Device CW0207364 CW0207364 CW0207364 CW0207364 CW0207364 CW0207362 CW0207362 CW0207362 CW0207362 CW0207362 CW02073162 CW020746 CW02	<ul> <li>The CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit is a low-cost hardware platform that enables design and debug of PSoC 6 MCUs. It comes with a Murata LBEESKLIDX module, based on the CYW4343W combo device, industry-leading CapSense for touch buttons and silder, on-board debugger/programmer with KitProg3, microSD card interface, 512-Mb Quad-SPI NOR flash, PDM-PCM microphone, and a thermistor. This kit is designed with a snap-away form-factor, allowing the user to separate the different components and features that come with this kit and use independently. In addition, support for Digilent's Pmod interface is also provided with this kit.</li> <li>Kit Features:</li> <li>Support of up to 2MB Flash and 1MB SRAM</li> <li>Dedicated SDHC to interface with WICED wireless devices.</li> <li>Delivers dual-cores, with a 150-MHz Arm Cortex-M4 as the primary application processor and a 100-MHz Arm Cortex-M0+ as the secondary processor for low-power operations.</li> <li>Supports Full-Speed USB, capacitive-sensing with CapSense, a PDM-PCM digital microphone interface, a Quad-SPI interface, 13 serial communication blocks, 7 programmable analog blocks, and 56 programmable digital blocks.</li> <li>Kit Contents:</li> <li>PSoC 6 Wi-Fi BT Prototyping Board</li> <li>USB Type-A to Micro-B cable</li> <li>Quick Start Guide</li> </ul>
Summary:			
BSP: CY8CPROTO-062-4343W			
Press "Next" to select application.			
			Next > Close
			<u>ilieu &gt;</u>
0			

**3.** On the Select Application window, select **QSPI Flash Read Write**. Optionally, change the application name



### 2 Procedures

Roject Creator 1.2 - Select Application		- 🗆 ×
Settings Help		
Application(s) Root Path: C:/ModusToolbox		Browse
Search	🛛 🖉 🌹 🎼 🔠 This example demonstrates interfacing with an external NOR flash memory in Quad-SPI mode using the Seriel	Memory Interface (SMIF) block in PSoC 6 MCU.
Template Application Emulated EEPROM Fault Handling Free-Running Multi-Counter Watchdog Timer GPIO Interrupt	New Application Name     Por more details, see the <u>README on Getwal</u> .	
Hello World IZC Master IZC Master EziZC Slave IZC Slave Using Callbacks IZS Audio		
Litz Jaire Gang Cancards     Litz Jaire Gang Smart IO and SPI     Litz Master using Smart IO and SPI     Littlef Flagstam     Low-power CapSense FreeRTOS     Low-Power Timer     MCUBoor-Based Basic Bootloader		
PDM PCM Audio     PDM to I2S Audio     Protection Units FreeRTOS     PWM Square Wave		
COSP Fish Read Write     SOPI Fish Read Write Using SFDP     QSPI XIP     Ramping LED using Smart IO     RT Basis     SCS SPI Master DMA     SCS UART Transmit and Receive using DMA	OSPI Flash Read Winte	
SRID Target Instruction Backing Joint SRID Target Instruction Backing SRID Target Instruction USS Audio Device FreeNIOS USS Audio Device FreeNIOS USS AUdio Recorder USS AUDIO Recorder USS COC Exho USS HUD Gerenic		
USB MID Genenc USB Mass Storage File System USB Mass Storage Logger USB Suspend and Resume Watchdog Timer	-	
Summary: BSP: CY8CPROTO-062-4343W Template Application(s): CSPI Flash Read Write Application(s) Root Path: C:/ModusToolbox		
Press "Create" to create the selected application(s).		
	< Back	<u>Greate</u> <u>Glose</u>
€		

The following screen appears when the selected example is successfully imported:

ModusToolbox - QSPI_Flash_Read_Write/README.md - Eclipse IDE for ModusToolb	ANK CONTRACTOR OF	- 0	×	
File Edit Navigate Search Project Run Window Help				
□ • □ 0   0 • 5 • □   0 • 0 • 9 •   > = = H > >     5 ≠   3   3   3   3   3   3   3   3   3	★ ■ 10 年 前 = 0 0 = 0 = 10	Quick Access	e 1 🕿	
Se Project Explorer = Debug III Registers & Peripherals	□ ® READMEnd □ =	3 In Outline Ⅲ P B	× = □	
✓	PSoC 6 MCU: QSPI Flash Read and Write			
> ab build > ab deps > ab images > ab lbs	This example demonstrates interfacing with an external NOR flash memory in Quad-SPI mode using the Serial Memory Interface (SMIF) block in PSoC® 6 MCU.	<ul> <li>h2. Hardware Setup</li> <li>h2. Software Setup</li> <li>v h2. Using the Code Example</li> <li>h3. In Eclipse IDE for ModusToolbox</li> </ul>		
<ul> <li>G main.c</li> <li>LICENSE</li> </ul>	Requirements	h3. In Command-line Interface (CLI): h3. In Third-party IDEs:		
A - Madelle     Tradellint     readellint     README md	ModusToolbox <sup>11</sup> software V2.1     Programming Language: C     Associated Parts: All <u>PSoC® 6 MCU</u> parts     Supported Kits	<ul> <li>h2. Operation</li> <li>h2. Debugging</li> <li>h2. Design and Implementation</li> <li>h5. Table 1. Application Resources</li> <li>h2. Related Resources</li> <li>h2. Other Resources</li> <li>h2. Document History</li> </ul>		
N Quel Pres	PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-062-4343W) - Default target     PSoC 6 WiFi-BT Proneer Kit (CY8CKIT-062-WiFi-BT)     PSoC 6 BLE Phoneer Kit (CY8CKIT-062-48LE)     PSoC 623 Wi-Fi BT Proneer Kit (CYW9P625143433EVB-01)     PSoC 623 Wi-Fi BT Proneer Kit (CYW9P625143433EVB-01)     PSoC 623 Wi-Fi BT Proneer Kit (CYW9P625143012EVB-01)     PSoC 623 Wi-Fi BT Proneer Kit (CYW9P625143012EVB-01)     PSoC 623 Wi-Fi BT Proneer Kit (CY8CPROTO-06253-4343W)			
Cuick Panel - Variables - Expressions - Breakpoints Eclipse IDE for				
ModusToolbox*	Hardware Setup			
* Start				
Mew Application	This example uses the board's default configuration. See the kit user guide to ensure that the board is configured correctly.			
Search Online for Code Examples	This example date the board a details comparation, dee the kit date guide to end the board is compared controlay.			
	Note: The PSoC 6 BLE Pioneer Kit and the PSoC 6 WiFi-BT Pioneer Kit ship with KitProg2 installed. ModusToolbox software requires KitProg3. Before			
S Refresh Quick Panel	using this code example, make sure that the board is upgraded to KitProg3. The tool and instructions are available in the Firmware Loader GitHub			
* QSPI Flash Read Write (CY8CPROTO-062-4343W)		Low de la		
Build QSPI Flash Read Write Application	Console = 12 Problems 0 Memory al News	142 H G V G V	• • •	
Clean QSPI Flash Read Write Application	ModuNoDbox Console Log file(s) for this session are stored at: C:\Users\pyostell\AppData\Local\Temp\Logs183569665838581490			
* Launches	Opening project creator from C:/Users/pyostell/ModusToolbox/tools_2.2/project-creator/project-creator			
QSPL Flash Read_Write Debug (ILink)	Importing (GPT_Flash_Read_Write (1/1) Log file(s) for Project Creator are stored at: C:\ModusToolbox\project-creator.log			
QSP1_Flash_Read_Write Debug (KitProg3_MiniProg4)	Done.			
QSPL Flash_Read_Write Program (JLink)				
QSPL Flash_Read_Write Program (kttProg3_MiniProg4)				
Generate Launches for QSPI. Flash. Read. Write				
Tools				
Library Manager 1.2				
Blustooth Confinizator 2.20 (new confinization)     S	A.			
		C/C++ Indexer: (27%)	-	



2 Procedures

### 2.2.4 Determine the serial flash operation structure

Note that the application already includes supports for some flash operation commands for QSPI Flash Read and Write operation. The cy\_smif\_memslot.h file includes code to support the WRR(01h), WRDI(04h), RDSR1(05h), WREN(06h), 4QPP(34h), RDCR(35h), BE(60h), 4SE(DCh), and 4QIOR(ECh) commands. However, these commands may not be sufficient to test all flash operations.



#### 2 Procedures

Code Listing 1: Built-in functions in cycfh\_qspi\_memslot.c

```
typedef struct
{
/*This specifies the number of address bytes used by the memory slave device, valid values 1-4
*/
 uint32_t numOfAddrBytes;
  /*The memory size: For densities of 2 gigabits or less - the size in bytes; For densities
4 gigabits and above - bit-31 is set to 1b to define that this memory is 4 gigabits and
above; and other 30:0 bits define N where the density is computed as 2^N bytes. For example,
0x80000021 corresponds to 2^30 = 1 gigabyte.*/
 uint32_t memSize;
/*This specifies the Read command */
 cy stc smif mem cmd t* readCmd;
/*This specifies the Write Enable command */
 cy_stc_smif_mem_cmd_t* writeEnCmd;
/*This specifies the Write Disable command */
 cy_stc_smif_mem_cmd_t* writeDisCmd;
/*This specifies the Erase command */
 cy_stc_smif_mem_cmd_t* eraseCmd;
/*This specifies the sector size of each Erase */
 uint32 t eraseSize;
/*This specifies the Chip Erase command */
 cy_stc_smif_mem_cmd_t* chipEraseCmd;
/*This specifies the Program command */
 cy_stc_smif_mem_cmd_t* programCmd;
/*This specifies the page size for programming */
 uint32_t programSize;
/*This specifies the command to read the WIP-containing status register */
 cy stc smif mem cmd t* readStsRegWipCmd;
/*This specifies the command to read the QE-containing status register */
 cy_stc_smif_mem_cmd_t* readStsRegQeCmd;
/*This specifies the command to write into the QE-containing status register */
 cy_stc_smif_mem_cmd_t* writeStsRegQeCmd;
/*This specifies the read SFDP command */
 cy stc smif mem cmd t* readSfdpCmd;
/*This specifies the Read ID command */
 cy_stc_smif_mem_cmd_t* readIDCmd;
/* The Busy mask for the status registers */
 uint32_t stsRegBusyMask;
/*The QE mask for the status registers */
 uint32_t stsRegQuadEnableMask;
/*Max time for erase type 1 cycle time in ms*/
 uint32_t eraseTime;
/*Max time for chip erase cycle time in ms */
 uint32 t chipEraseTime;
/*Max time for page program cycle time in us */
 uint32_t programTime;
/*This specifies the number of regions for memory with hybrid sectors */
 uint32_t hybridRegionCount;
/*This specifies data for memory with hybrid sectors */
```



### 2 Procedures

```
cy_stc_smif_hybrid_region_info_t** hybridRegionInfo;
} cy stc smif mem device cfg t;
```

This section provides steps to add additional operation commands other than the commands provided by the example code. The flash commands vary from the products and the interface—see the datasheet for each flash for supported command sets.

For example, a brief list of supported commands for the S25FL512S device is in shown in Table 2. See the datasheet for detailed descriptions for each command.

The example code provided commands WRR(01h), WRDI(04h), RDSR1(05h), WREN(06h), 4QPP(34h), RDCR(35h), BE(60h), 4SE(DCh), 4QIOR(ECh) are shown in **Bold**.

Instruction (in HEX)	Command name	Instruction (in HEX)	Command name
01	WRR	6C	4QOR
02	PP	75	ERSP
03	READ	7A	ERRS
04	WRDI	85	PGSP
05	RDSR1	8A	PGRS
06	WREN	90	READ_ID
07	RDSR2	9F	RDID
0B	FAST_READ	A3	МРМ
0C	4FAST_READ	A6	PLBWR
0D	DDRFR	A7	PLBRD
0E	4DDRFR	AB	RES
12	4PP	B9	BRAC
13	4READ	BB	DIOR
14	ABRD	BC	4DIOR
15	ABWR	BD	DDRDIOR
16	BRRD	BE	4DDRDIOR
17	BRWR	C7	BE
18	ECCRD	D8	SE
20	P4E	DC	4SE
21	4P4E	E0	DYBRD
2B	ASPRD	E1	DYBWR
2F	ASPP	E2	PPBRD
30	CLSR	E3	РРВР
32	QPP	E4	PPBE
34	4QPP	E5	Reserved-E5
35	RDCR	E6	Reserved-E6

#### Table 2Command lists

(table continues...)

. . . .



### 2 Procedures

.

...

.....

- - - -

Table 2 (continued) Command lists			
Instruction (in HEX)	Command name	Instruction (in HEX)	Command name
38	QPP	E7	PASSRD
3B	DOR	E8	PASSP
3C	4DOR	E9	PASSU
41	DLPRD	EB	QIOR
42	OTPP	EC	4QIOR
43	PNVDLR	ED	DDRQIOR
4A	WVDLR	EE	4DDRQIOR
4B	OTPR	F0	RESET
60	BE	FF	MBR
6B	QOR		

Use the top-down approach on the codes to determine the architecture of the serial flash memory operation. For example, consider the cy\_serial\_flash\_qspi\_read operation starting from main.c to investigate the SMIF structure.



#### Figure 7 SM

SMIF structure

By exploring from main.c and all the way down to cy\_smif.c, you can determine that the actual operation command is delivered to Cy\_SMIF\_TransmitCommand in cy\_smif.c with command arguments, command width, address width, mode, mode width, the number of dummy cycles, and data width. The arguments are specified in cycfg\_qspi\_memslot.c for each operation command.



#### 2 Procedures

Code Listing 2: S25FL512S\_4byteaddr\_SlaveSlot\_0\_readCmd in cycfg\_qspi\_memslot.c

```
const cy_stc_smif_mem_cmd_t
  S25FL512S 4byteaddr SlaveSlot 0 readCmd =
    {
       /* The 8-bit command. 1 x I/O read command. */
       .command = 0 \times ECU,
       /* The width of the command transfer. */
       .cmdWidth = CY_SMIF_WIDTH_SINGLE,
       /* The width of the address transfer. */
       .addrWidth = CY_SMIF_WIDTH_QUAD,
       /* The 8-bit mode byte. This value is 0xFFFFFFF when there is no mode present. */
       .mode = 0x01U,
       /* The width of the mode command transfer. */
       .modeWidth = CY_SMIF_WIDTH_QUAD,
       /* The number of dummy cycles. A zero value suggests no dummy cycles. */
       .dummyCycles = 4U,
       /* The width of the data transfer. */
       .dataWidth = CY SMIF WIDTH QUAD
    };
```

See the datasheet for the argument values in the command sequence.

cs#
sckกากการการผู้บากการการการการการการกา
IO1 29 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1 5 1
Phase Instruction Adverss Mode Dummy D1 D2 D3 D4

Figure 8

Quad I/O Read command sequence (4-byte address, ECh or EBh)

### 2.2.5 Add additional flash operation commands

To add an additional command other than the supported commands, reverse the top-down approach. This is demonstrated with the RDID (9Fh) command.

**1.** Check the command sequence for RDID 9Fh

CS#			<u>∭</u>
SCK			
s 📃	7 6 5 4 3 2 1 0		l
<u>so</u> —		7 6 5 4 3 2 1 0	6 5 4 3 2 1 0
Phase	Instruction	Data 1	Data N

### Figure 9 Read Identification (RDID 9Fh) command sequence

2. Specify the arguments for the operation command in cycfg\_qspi\_memslot.c to define the SMIF-driver memory configuration



### 2 Procedures

Code Listing 3: S25FL512S\_4byteaddr\_SlaveSlot\_0\_readIDCmd in cycfg\_qspi\_memslot.c

```
const cy_stc_smif_mem_cmd_t
S25FL512S 4byteaddr SlaveSlot 0 readIDCmd =
{
 /* The 8-bit command. 1 x I/O read command. */
 .command = 0x9FU,
 /* The width of the command transfer. */
 .cmdWidth = CY SMIF WIDTH SINGLE,
 /* The width of the address transfer. */
 .addrWidth = CY_SMIF_WIDTH_NA,
 /* The 8-bit mode byte. This value is 0xFFFFFFF when there is no mode present. */
 .mode = 0xFFFFFFFU,
 /* The width of the mode command transfer. */
 .modeWidth = CY SMIF WIDTH NA,
 /* The number of dummy cycles. A zero value suggests no dummy cycles. */
 .dummyCycles = OU,
 /* The width of the data transfer. */
 .dataWidth = CY_SMIF_WIDTH_SINGLE
};
```

Add the read ID command to the command structure in cycfg\_qspi\_memslot.c
 Code Listing 4: deviceCfg\_S25FL512S\_4byteaddr\_SlaveSlot\_@in cycfg\_qspi\_memslot.c

```
const cy_stc_smif_mem_device_cfg_t
deviceCfg_S25FL512S_4byteaddr_SlaveSlot_0 =
    {
        ...
        .readIDCmd =
  (cy_stc_smif_mem_cmd_t*)&S25FL512S_4byteaddr_SlaveSlot_0_readIDCmd,
        ...
    }
```

**4.** Add the function that actually operates the command in cy\_smif\_memslot.c where the source code for memory-level APIs for the SMIF driver is provided



#### 2 Procedures

**Code Listing 5:** Cy\_SMIF\_MemCmdReadID in cy\_smif\_memslot.c

```
cy_en_smif_status_t Cy_SMIF_MemCmdReadID(SMIF_Type *base,
                             cy_stc_smif_mem_config_t const *memDevice,
                             uint8_t* readBuff,
                             uint32_t size,
                             cy_stc_smif_context_t *context)
{
  cy en smif status t result = CY SMIF BAD PARAM;
  cy_en_smif_slave_select_t slaveSelected;
  cy_stc_smif_mem_device_cfg_t *device = memDevice->deviceCfg;
  cy_stc_smif_mem_cmd_t *cmdReadID = device->readIDCmd;
  if(NULL == cmdReadID)
  {
    result = CY_SMIF_CMD_NOT_FOUND;
  }
  else
  {
    slaveSelected = (OU == memDevice->dualQuadSlots)?memDevice->slaveSelect:
                    (cy_en_smif_slave_select_t)memDevice->dualQuadSlots;
    result = Cy_SMIF_TransmitCommand( base,
             (uint8_t)cmdReadID->command,
             cmdReadID->cmdWidth,
             CY_SMIF_CMD_WITHOUT_PARAM, CY_SMIF_CMD_WITHOUT_PARAM,
                     CY_SMIF_WIDTH_NA,
             slaveSelected, CY_SMIF_TX_NOT_LAST_BYTE, context);
    if(CY_SMIF_SUCCESS == result)
    {
      result = Cy_SMIF_ReceiveData(base, readBuff, size,
                     cmdReadID->dataWidth, NULL, context);
    }
  }
  return(result);
}
```

5. Add the function that returns the data to the rxBuffer in cy\_smif\_memslot.c where the source code for memory-level APIs for the SMIF driver is provided



#### 2 Procedures

**Code Listing 6:** Cy\_SMIF\_MemCmdReadID in cy\_smif\_memslot.c

```
cy_en_smif_status_t Cy_SMIF_MemReadID(SMIF_Type *base,
                cy_stc_smif_mem_config_t const *memConfig,
                uint8_t rxBuffer[],
                uint32_t length,
                cy_stc_smif_context_t *context)
{
  cy_en_smif_status_t status = CY_SMIF_BAD_PARAM;
  uint32_t chunk = OUL;
  CY_ASSERT_L1(NULL != memConfig);
  CY_ASSERT_L1(NULL != rxBuffer);
  if(1)
  {
       /* SMIF can read only up to 65536 bytes in one go. Split the larger
          read into multiple chunks */
    while (length > 0UL)
    {
       /* Get the number of bytes which can be read during one operation */
      chunk = (length > SMIF MAX RX COUNT) ?
                  (SMIF_MAX_RX_COUNT) : length;
       /* Send the command to read data from the external memory to the rxBuffer array */
       status = Cy_SMIF_MemCmdReadID(base, memConfig,
                  (uint8_t *)rxBuffer, chunk, context);
       if(CY_SMIF_SUCCESS == status)
         {
            /* Wait until the SMIF block completes receiving data */
            status = PollTransferStatus(base, CY_SMIF_REC_CMPLT, context);
         }
         if(CY_SMIF_SUCCESS != status)
         {
          break;
         }
         /* Recalculate the next rxBuffer offset */
        length -= chunk;
         rxBuffer = (uint8_t *)rxBuffer + chunk;
    }
  }
  return status;
}
```

6.

6. Add the function that executes the read ID operation to the external flash in cy\_serial\_flash\_qspi.c where it provides APIs for interacting with an external flash



### 2 Procedures

**Code Listing 7:** cy\_serial\_flash\_qspi\_readID in cy\_serial\_flash\_qspi.c

```
cy_rslt_t cy_serial_flash_qspi_readID(size_t length,
                              uint8 t* buf)
{
  cy_rslt_t result_mutex_rel = CY_RSLT_SUCCESS;
  cy_rslt_t result = _mutex_acquire();
  if (CY RSLT SUCCESS == result)
  {
    /* Cy_SMIF_MemReadID() returns error if (addr + length) > total flash
            size.*/
    result = (cy_rslt_t)Cy_SMIF_MemReadID(qspi_obj.base,
                qspi_block_config.memConfig[MEM_SLOT],
                buf, length,
                &qspi obj.context);
                result_mutex_rel = _mutex_release();
    result_mutex_rel = _mutex_release();
  }
  /* Give priority to the status of SMIF operation when both SMIF
            operation and mutex release fail.*/
            return ((CY_RSLT_SUCCESS == result) ? result_mutex_rel :
                         result);
}
```

7.

In main.c, add the function to operate the read device RDID (9Fh) operation
Code Listing 8: cy\_serial\_flash\_qspi\_readID in main.c

```
int main(void)
{
...
printf("\r\n1. Reading device ID\r\n");
result = cy_serial_flash_qspi_readID(DEVICE_ID, rx_buf);
check_status("Reading Device ID failed", result);
printf("Byte Address: Data\r\n");
for(uint8_t i = 0; i < DEVICE_ID; i++)
printf("0x%02dh : 0x%02Xh \r\n", i, rx_buf[i]);
...
}</pre>
```

Make sure to include the added functions to the header files also.

This example code is in SPI mode. Some flash devices such as S25FS512S supports QPI mode, so if you want to operate RDID (9Fh) in QPI mode, you can modify the command width and data width into Quad.

### 5 1 5 1 5 1 5 1

6

0

## Phase Instruction D1 D2 D3 D4

3

### Figure 10 Read Identification (RDID) QPI mode command

Λ

5

6

Code Listing 9: RDID(9Fh) in QPI mode

4

5

6

Ω

3

2 Procedures

CS#

SCK

100 -

101 -

102 -

103 .

```
const cy_stc_smif_mem_cmd_t
S25FL512S_4byteaddr_SlaveSlot_0_readIDQPICmd =
{
 /* The 8-bit command. 1 x I/O read command. */
 .command = 0x9FU,
 /* The width of the command transfer. */
 .cmdWidth = CY SMIF WIDTH QUAD,
 /* The width of the address transfer. */
 .addrWidth = CY_SMIF_WIDTH_NA,
 /* The 8-bit mode byte. This value is 0xFFFFFFFF when there
            is no mode present. */
  .mode = 0xFFFFFFFF,
 /* The width of the mode command transfer. */
 .modeWidth = CY_SMIF_WIDTH_NA,
 /* The number of dummy cycles. A zero value suggests no dummy cycles. */
  .dummyCycles = OU,
 /* The width of the data transfer. */
 };
```

### PSoC<sup>™</sup> 6 MCU: Modify CY8CPROTO-062-4343W board to work with an external flash memory

n

3

Λ

6

Λ

6

n



Ω

3

D5

Λ

6



3 PSoC<sup>™</sup> 6 MCU board operation

### **3** PSoC<sup>™</sup> 6 MCU board operation

Figure 11 shows the waveforms with the QPI bit set for the FS512S RDID QPI operation signals measured with an oscilloscope. Normally, input signal instructions are latched on the rising edge of the SCK signal. Then, the data output changes after the falling edge of SCK in SDR commands. Because this is a QPI operation, the output data should come after the first 8-bits of the two serial clock instructions on the falling edge. However, for this PSoC<sup>™</sup> 6 MCU board, the signals remain HIGH even after the instruction phase; the data phase comes after A certain hold time. This hold time is driven by the controller.

Even though the flash memory is sending data signals after the instruction phase, the controller's pull-up signal is more dominant, thus ignoring the flash device's first data signal. Therefore, as soon as the controller drops the bus, a dip can be made regardless of the real signal due to the controller's sudden drop from HIGH to LOW. Therefore, in the oscilloscope measurements' point of view, the first data should be read right after the dip but before the second falling edge of the flash device-driven phase. This is because the flash device's output data is already on the signal bus but ignored due to controller's control over the bus.

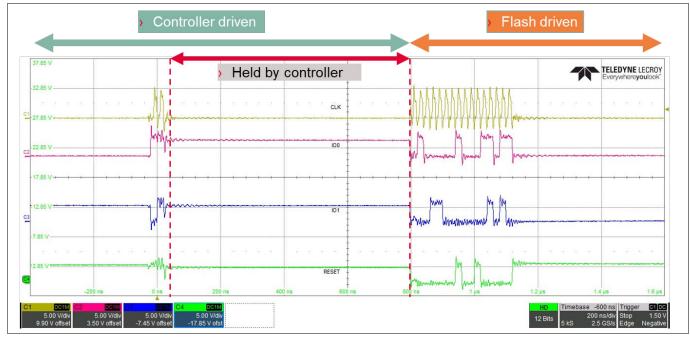


Figure 11

FS512S QPI bit set RDID(9Fh) QPI operation



### 3 PSoC<sup>™</sup> 6 MCU board operation

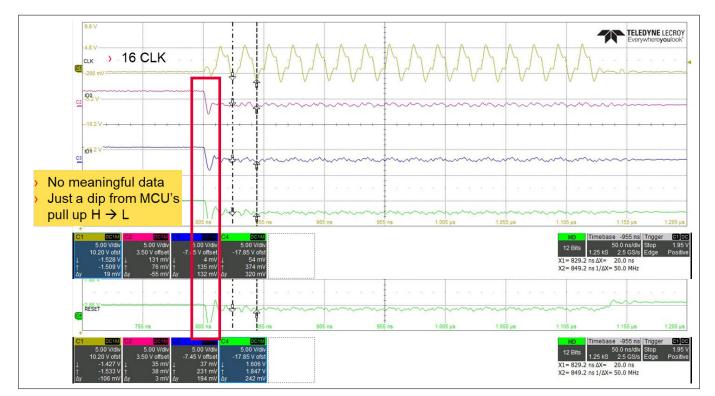


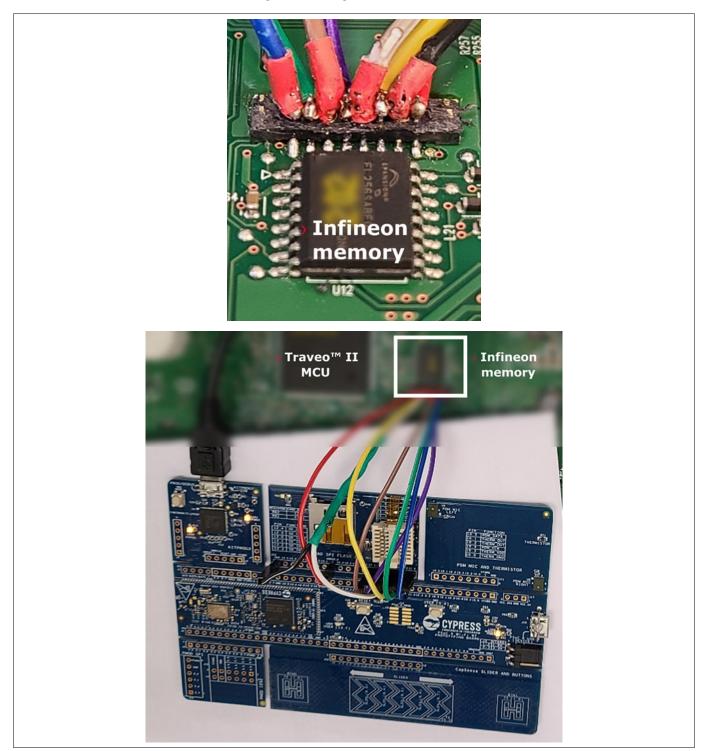
Figure 12 No meaningful dip at the beginning of the data phase

4 Use cases

### 4 Use cases

Even though these changes make the kit a general flash evaluation tool, its usage can be expanded beyond a demo board. Here's an actual use case to a customer's board. When a failure case is submitted, application-level investigation should be done to narrow down the root cause of the failure before failure analysis (FA) is done at the chip level.

Note that the jumpers are connected to the connector pinout on the PCB; it is then connected to the pin header of the PSoC<sup>™</sup> 6 MCU board to avoid damage on the target board.





PSoC<sup>™</sup> 6 MCU board connected externally to a customer's application board





#### 4 Use cases

By reading the device ID using the function added and reading the data on a specific address, identical abnormal operation was reproduced with PSoC<sup>™</sup> 6 MCU board.

**************************************	iv Debugging 20210506*********
*****	*****
*********************	's s25FL256s *********************
0. Reading ID	Customer's board
Reading ID 2 (6 bytes):	
0x01 0x02 0x19 0x40 0x00 0x80 2. Reading External memory data	address: 0x0000
Received Data (64 bytes):	
0x30 0x30 0x30 0x30 0x30 0x30 0x35 0 0x01 0x00 0x02 0x00 0x04 0x00 0	0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00
3. Reading External неногу data	address: 0x20000
Received Data (64 bytes):	
0x30 0x30 0x30 0x30 0x30 0x30 0x35 0 0x01 0x00 0x02 0x00 0x04 0x00 0	)x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30 )x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33 )x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00 )x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00
4. Reading External memory data	address: 0x40000
Received Data (64 bytes):	
0x30 0x30 0x30 0x30 0x30 0x30 0x35 0 0x01 0x00 0x02 0x00 0x04 0x00 0	)x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30 )x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33 )x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00 )x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00
5. Reading External мемогу data	address: 0x60000
Received Data (64 bytes):	
0x30 0x30 0x30 0x30 0x30 0x30 0x35 0 0x01 0x00 0x02 0x00 0x04 0x00 0	0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00

#### Figure 14

**Read ID operation result** 



**5** Conclusion

### 5 Conclusion

Even though CY8CPROTO-062-4343W PSoC<sup>™</sup> 6 prototyping kit is provided with a Quad SPI flash device and example code, its use as a general-purpose flash evaluation tool is limited. By modifying the hardware and software of the board, it can be turned into a general tool to operate any flash device.

By attaching a socket, the flash device can be replaced with any SOIC package flash. By attaching pin headers, it frees the spatial constraints and connects any flash device with any package and even to the flash device on an external board. By adding operation commands, the test range can be extended to any commands.

By using the information in this application note, you will be able to make your own PSoC<sup>™</sup> 6 MCU-based flash evaluation tool.



### References

### References

- PSoC<sup>™</sup> 6 Wi-Fi Bluetooth<sup>®</sup> prototyping kit (CY8CPROTO-062-4343W) https://www.cypress.com/documentation/development-kitsboards/psoc-6-wi-fi-bt-prototyping-kitcy8cproto-062-4343w
- 2. CY8CPROTO-062-4343W PSoC<sup>™</sup> 6 Wi-Fi Bluetooth<sup>®</sup> prototyping kit guide https://www.cypress.com/file/457891/download
- 3. CY8CPROTO-062-4343W schematic https://www.cypress.com/file/457811/download
- **4.** S25FL512S, 512 Mb (64 MB), 3.0 V SPI flash memory datasheet https://www.cypress.com/documentation/datasheets/s25fl512s-512-mb-64-mb-30-v-spi-flash-memory
- 5. ModusToolbox<sup>™</sup> software download link https://www.cypress.com/products/modustoolbox#tabs-0-bottom\_side-6
- 6. ModusToolbox<sup>™</sup> software installation guide https://www.cypress.com/file/520241/download
- ModusToolbox<sup>™</sup> software user guide https://www.cypress.com/file/520251/download



**Revision history** 

### **Revision history**

Document version	Date of release	Description of changes
**	2021-09-17	New application note

#### Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-09-17 Published by Infineon Technologies AG 81726 Munich, Germany

© 2022 Infineon Technologies AG All Rights Reserved.

Do you have a question about any aspect of this document? Email: erratum@infineon.com

Document reference IFX-gip1649233789295

#### Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

#### Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.