

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

## About this document

### Scope and purpose

This application note describes steps to enable a CY8CPROTO-062-4343W PSoC™ 6 Wi-Fi Bluetooth® 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™ 6 MCU serial memory interface (SMIF), this board can be turned into a memory evaluation board for any external flash.

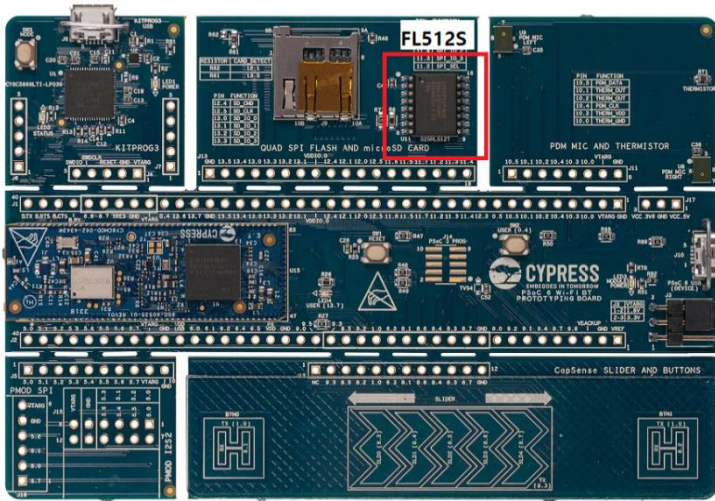
### Intended audience

This application note targets PSoC™ 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

<b>About this document.....</b>	<b>1</b>
<b>Table of contents.....</b>	<b>1</b>
<b>1 Introduction .....</b>	<b>2</b>
<b>2 Procedures .....</b>	<b>3</b>
2.1 Modifying the hardware .....	3
2.1.1 Materials required .....	3
2.1.2 Customize the Board.....	3
2.2 Modifying the software .....	6
2.2.1 ModusToolbox™ software.....	6
2.2.2 Serial memory interface (SMIF) .....	6
2.2.3 Import ModusToolbox™ QSPI flash code example .....	6
2.2.4 Determine the serial flash operation structure.....	10
2.2.5 Add additional flash operation commands .....	14
<b>3 PSoC™ 6 MCU board operation .....</b>	<b>19</b>
<b>4 Use cases.....</b>	<b>21</b>
<b>5 Conclusion .....</b>	<b>23</b>
<b>References.....</b>	<b>24</b>
<b>Revision history.....</b>	<b>25</b>

### 1 Introduction



**Figure 1** CY8CPROTO-062-4343W PSoC™ 6 Wi-Fi Bluetooth® prototyping board

The PSoC™ 6 Wi-Fi Bluetooth® 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™ MCU device. However, as this PSoC™ 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™ 6 MCU board hardware and software for customizing the board into a general evaluation tool for flash memory devices.

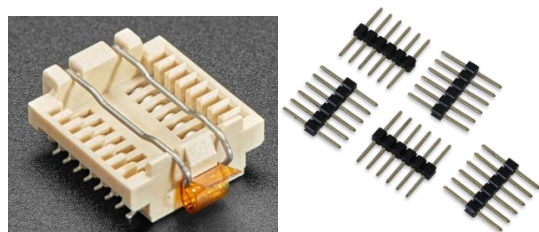
## Procedures

## 2 Procedures

Both hardware and software modifications are required for this application.

### 2.1 Modifying the hardware

#### 2.1.1 Materials required



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

- CY8CPROTO-062-4343W PSoC™ 6 W-Fi Bluetooth® 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.

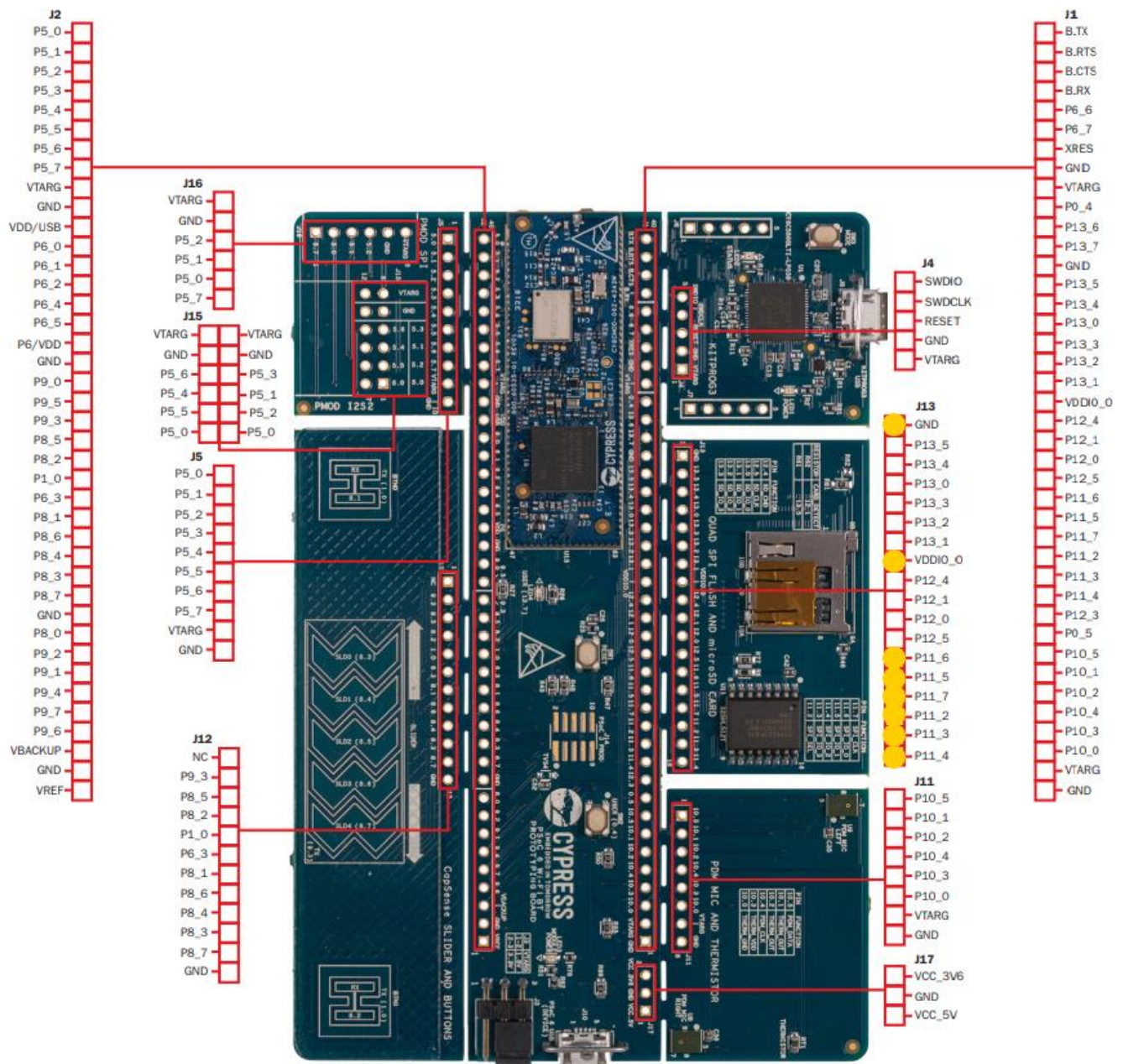
1. Remove the attached flash with the cutter. Gently cut off the flash lead frames and then remove the leftovers with solders. Be careful not to strip off the PCB pattern.
2. Fit the SMT socket-wide SOIC-16 into the PCB footprint. The existing flash on the board is in SOIC 16 pin package, so it is compatible 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](#) and [Figure 3](#).
4. Make sure that all pins are connected.

**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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

## Procedures



**Figure 3** Board pinout

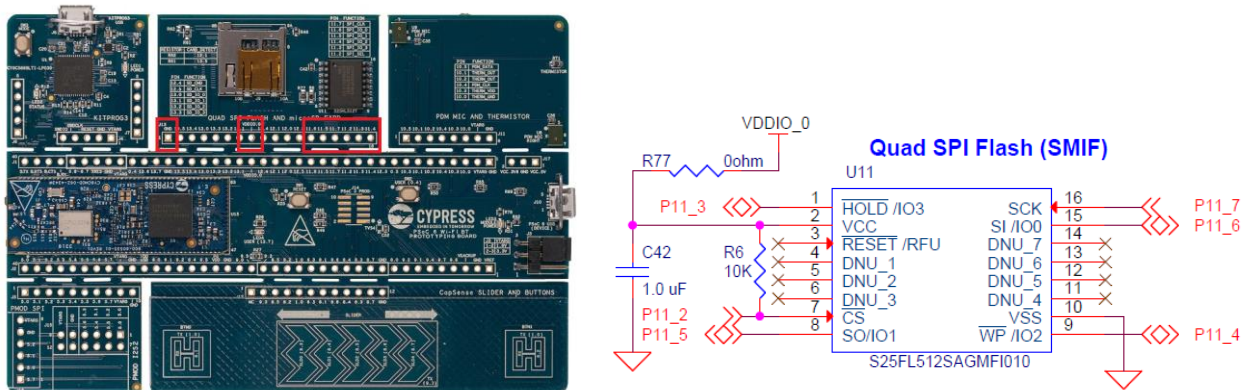
Test the flash devices as follows:

- 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.



# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

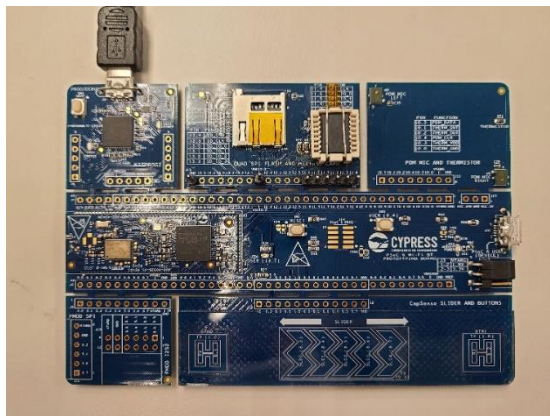
## Procedures



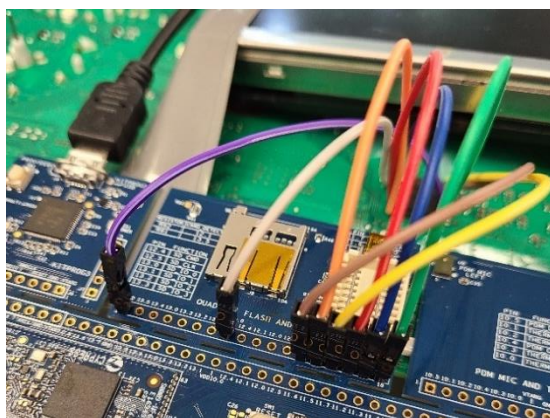
**Figure 4** CY8CPROTO-062-4343W external connections

**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 5** PSoC™ 6 MCU board with the socket attached



**Figure 6** External flash memory connected with a pin header

## Procedures

### 2.2 Modifying the software

#### 2.2.1 ModusToolbox™ software

PSoC™ 6 MCU boards use Eclipse IDE for ModusToolbox™ and C language for programming.



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

#### 2.2.2 Serial memory interface (SMIF)

The SMIF Component implements an SPI-based communication hardware block for interfacing external memory devices with the PSoC™ 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.

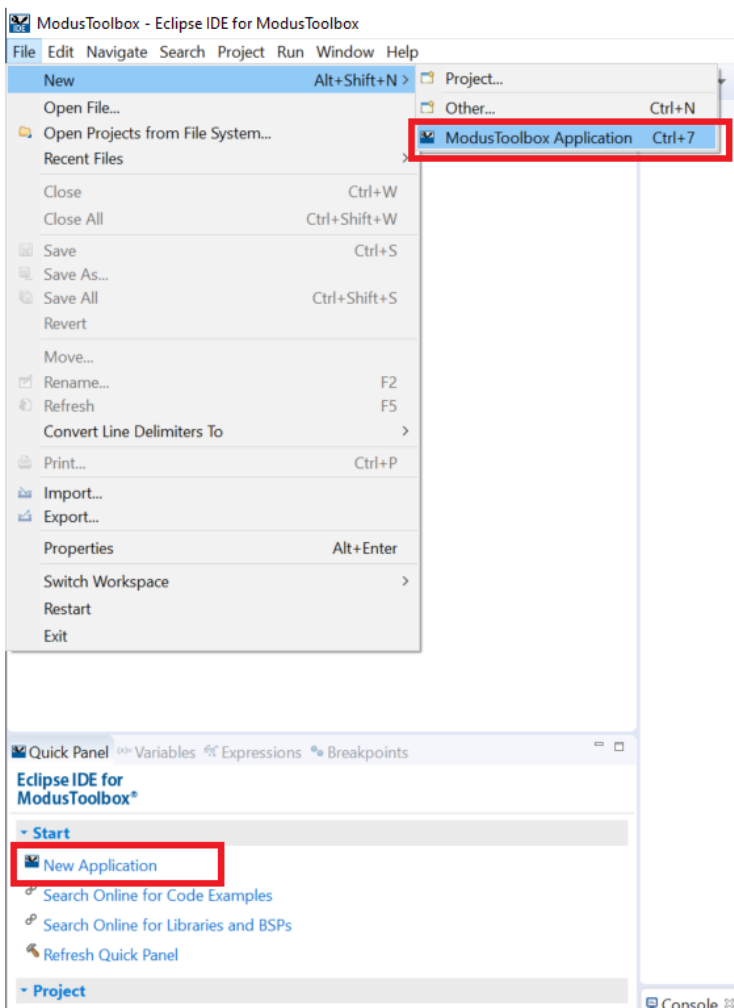
#### 2.2.3 Import ModusToolbox™ QSPI flash code example

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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures



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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

Project Creator 1.2 - Choose Board Support Package (BSP)

Settings Help

Enter filter text

Kit Name	MCU	Connectivity Device
CYBT-343026-EVAL	CYW20706A2	CYW20706A2
CYBT-343052-EVAL	CYBT-343052-02	CYW20735B1
CYBT-353027-EVAL	CYW20706A2	CYW20706A2
CYBT-413055-EVAL	CYBT-413055-02	CYW20719B2
CYBT-413061-EVAL	CYBT-413061-02	CYW20721B2
CYBT-423054-EVAL	CYBT-423054-02	CYW20719B2
CYBT-423060-EVAL	CYBT-423060-02	CYW20721B2
CYBT-483056-EVAL	CYBT-483056-02	CYW20719B2
CYBT-483062-EVAL	CYBT-483062-02	CYW20721B2
CYW920706WCDEVAL	CYW20706A2	CYW20706A2
CYW920719B2Q40VB-01	CYW20719B2KJMLG	CYW20719B2
CYW920721B2EVK-02	CYW20721B2KW9G	CYW20721B2
CYW920721M2EVK-01	CYW20721B2KJMLG	CYW20721B2
CYW920721M2EVK-02	CYW20721B2KW9G	CYW20721B2
CYW920735Q60VB-01	CYW20735B1	CYW20735B1
CYW920819VB-02	CYW20819A1KFBG	CYW20819A1
CYW920819REF-KB-01	CYW20819A1KFB1G	CYW20819A1
CYW920820VB-02	CYW20820A1KFBG	CYW20820A1
CYW920835M2VB-01	CYW20835PB1KML1G	CYW20835B1
CYW920835REF-KCU-01	CYW20835B1	CYW20835B1
CYW943012BTEVK-01	CYW43012C0WKWBG	CYW43012C0
CYW989820VB-01	CYW989820BWM1G	CYW20820A1
CYW9M28ASE-43012BT	CYW43012C0	CYW43012C0
PSOC 4 BSPs		
PSOC 6 BSPs		
CY8CKIT-062-BLE	CY8C6347BZ1-BLDS3	<none>
CY8CKIT-062S2-43012	CY8C624A8Z1-S2D44	CYW43012C0WKWBG
CY8CKIT-062S4	CY8C624ALQ1-S4D92	<none>
CY8CKIT-062-WIFI-BT	CY8C6247BZ1-D54	CYW4343WKUBG
CY8CKIT-0648052-4343W	CY80644A8Z1-S2D44	CYW4343WKUBG
<b>CY8CPROTO-062-4343W</b>	<b>CY8C624A8Z1-S2D44</b>	<b>CYW4343WKUBG</b>
CY8CPROTO-062S3-4343W	CY8C6245LQ1-S3D72	CYW4343WKUBG
CY8CPROTO-063-BLE	CY8LE-416045-02	<none>
CY8CPROTO-0648053	CY806445LQ1-S3D42	<none>
CY8LE-416045-EVAL	CY8LE-416045-02	<none>
CY8SYSKIT-01	CY8C624AFN1-S2D43	CYW43012C0KFFBH
CY8SYSKIT-DEV-01	CY8C624AFN1-S2D43	CYW43012C0KFFBH
CYW9P6251-43012VB-01	CY8C6247FQ1-D52	CYW43012C0KUBG
CYW9P6251-43438VB-01	CY8C6247BZ1-D54	CYW43438KUBG
PSOC6-GENERIC	CY8C6347BZ1-BLDS3	<none>

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 LBEE5KL1DX module, based on the CYW4343W combo device, industry-leading CapSense for touch buttons and slider, 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.

Kit Features:

- Support of up to 2MB Flash and 1MB SRAM
- Dedicated SDHC to interface with WICED wireless devices.
- 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.
- 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.

Kit Contents:

- PSoC 6 Wi-Fi BT Prototyping Board
- USB Type-A to Micro-B cable
- Quick Start Guide

Checking if remote manifest is accessible...  
Getting manifests from remote server...  
Processing super-manifest https://github.com/cypresssemiconductors/mib-super-manifest/raw/v2.X/mib-super-manifest-v2.xml... 100%  
Successfully acquired the information.

Summary:  
BSP: CY8CPROTO-062-4343W  
Press "Next" to select application.

Next > Close

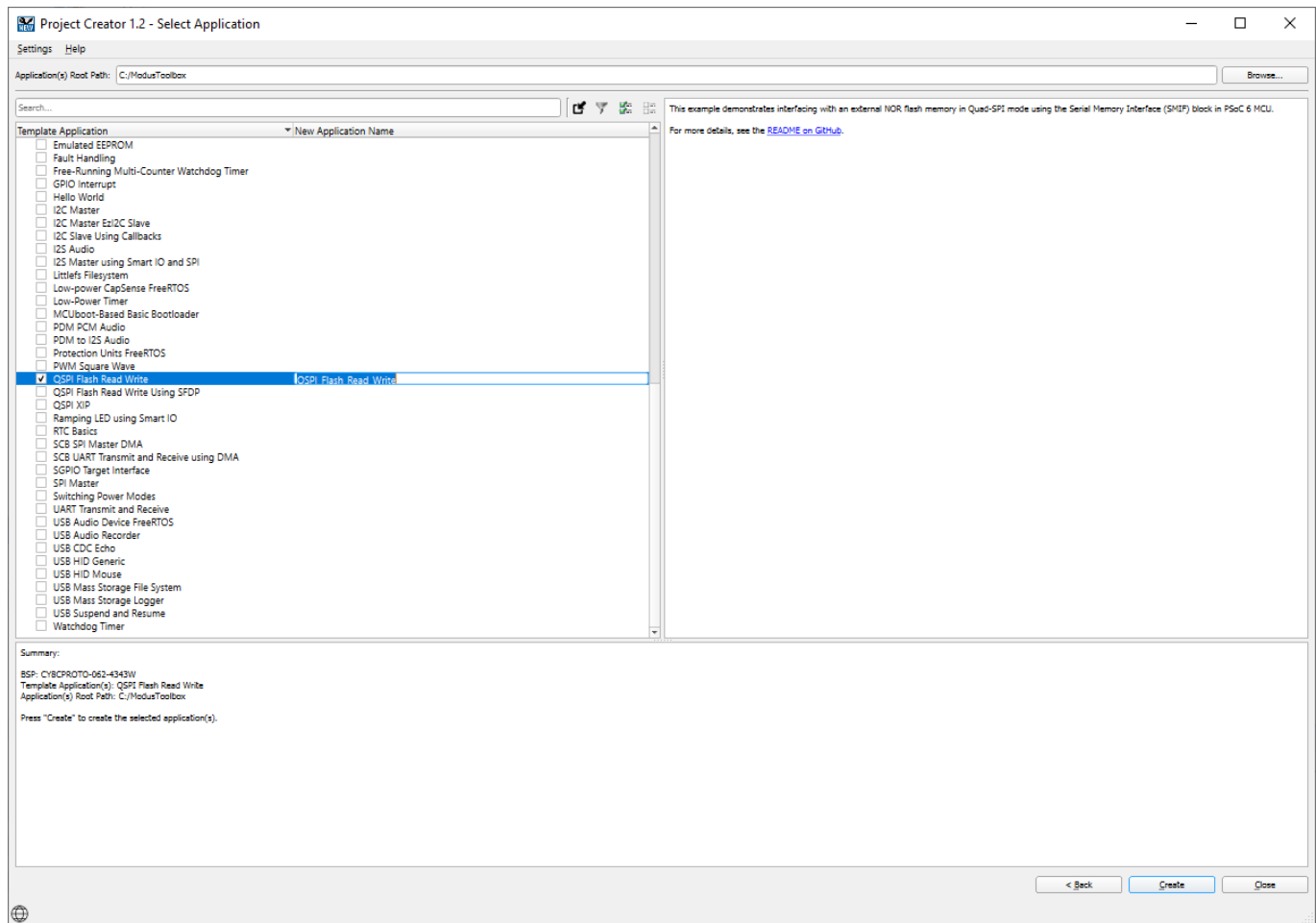
3. On the Select Application window, select **QSPI Flash Read Write**. Optionally, change the application name.



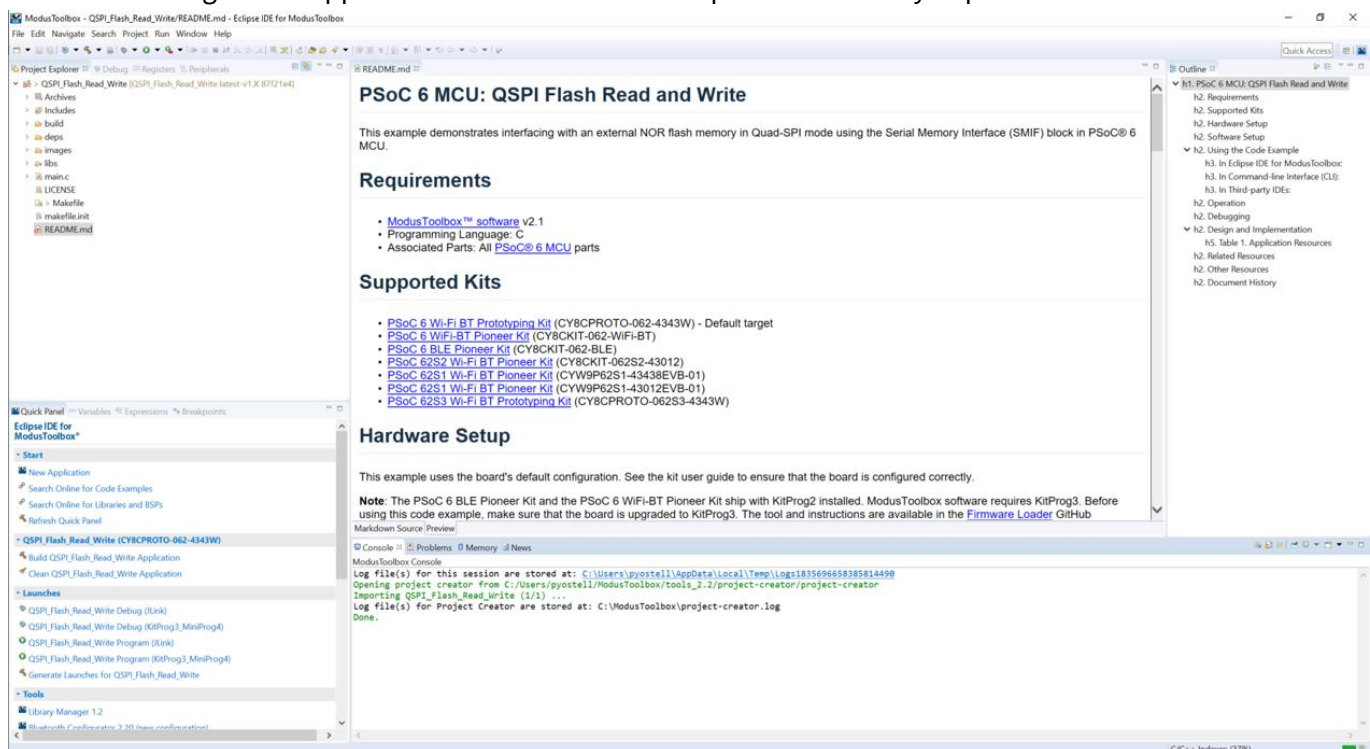
# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures



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



## 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.

#### Code Listing 1 Built-in functions in `cycfh_qspi_memslot.c`

```
001     typedef struct
002     {
003         uint32_t numOfAddrBytes;           /**<
004         This specifies the number of address bytes used by the
005         memory slave device, valid values 1-4 */
006         uint32_t memSize;                 /**<
007         The memory size: For densities of 2 gigabits or less - the size in
008         bytes;
009         densities 4 gigabits and above - bit-31 is set to 1b to define that
010         memory is 4 gigabits and above; and other 30:0 bits define N where
011         density is computed as 2^N bytes.
012         example, 0x80000021 corresponds to 2^30 = 1 gigabyte.
013         cy_stc_smif_mem_cmd_t* readCmd;   /**<
014         This specifies the Read command */
015         cy_stc_smif_mem_cmd_t* writeEnCmd; /**<
016         This specifies the Write Enable command */
017         cy_stc_smif_mem_cmd_t* writeDisCmd; /**<
018         This specifies the Write Disable command */
019         cy_stc_smif_mem_cmd_t* eraseCmd;   /**<
020         This specifies the Erase command */
021         uint32_t eraseSize;                /**<
022         This specifies the sector size of each Erase */
023         cy_stc_smif_mem_cmd_t* chipEraseCmd; /**<
024         This specifies the Chip Erase command */
025         cy_stc_smif_mem_cmd_t* programCmd; /**<
026         This specifies the Program command */
027         uint32_t programSize;              /**<
028         This specifies the page size for programming */
029         cy_stc_smif_mem_cmd_t* readStsRegWipCmd; /**<
030         This specifies the command to read the WIP-containing status register
031         */
032         cy_stc_smif_mem_cmd_t* readStsRegQeCmd; /**<
033         This specifies the command to read the QE-containing status register */
034         cy_stc_smif_mem_cmd_t* writeStsRegQeCmd; /**<
035         This specifies the command to write into the QE-containing status
036         register */
037         cy_stc_smif_mem_cmd_t* readSfdpCmd; /**<
038         This specifies the read SFDP command */
039         cy_stc_smif_mem_cmd_t* readIDCmd;  /**<
040         This specifies the Read ID command */
```

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

```
024      uint32_t stsRegBusyMask;                                /**<
    The Busy mask for the status registers */
025      uint32_t stsRegQuadEnableMask;                          /**<
    The QE mask for the status registers */
026      uint32_t eraseTime;                                      /**<
    Max time for erase type 1 cycle time in ms */
027      uint32_t chipEraseTime;                                  /**<
    Max time for chip erase cycle time in ms */
028      uint32_t programTime;                                    /**<
    Max time for page program cycle time in us */
029      uint32_t hybridRegionCount;                              /**<
    This specifies the number of regions for memory with hybrid sectors */
030      cy_stc_smif_hybrid_region_info_t** hybridRegionInfo;    /**<
    This specifies data for memory with hybrid sectors */
031      } 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 [blue](#) color.

**Table 2** Command lists

Instruction (in HEX)	Command name	Instruction (in HEX)	Command name
<a href="#">01</a>	<a href="#">WRR</a>	6C	4QOR
02	PP	75	ERSP
03	READ	7A	ERRS
<a href="#">04</a>	<a href="#">WRDI</a>	85	PGSP
<a href="#">05</a>	<a href="#">RDSR1</a>	8A	PGRS
<a href="#">06</a>	<a href="#">WREN</a>	90	READ_ID
07	RDSR2	9F	RDID
0B	FAST_READ	A3	MPM
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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

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	MPM
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
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	PPBP
32	QPP	E4	PPBE
34	4QPP	E5	Reserved-E5
35	RDCR	E6	Reserved-E6
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		

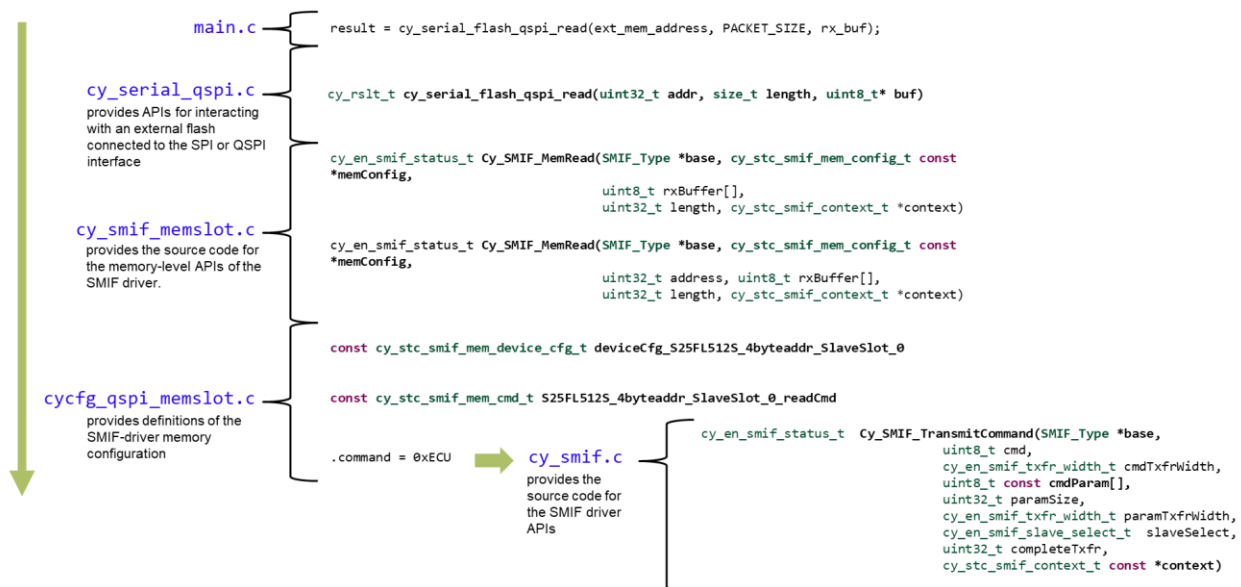
Use the top-down approach on the codes to determine the architecture of the serial flash memory operation.

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

For example, consider the `cy_serial_flash_qspi_read` operation starting from `main.c` to investigate the SMIF structure.



**Figure 7** 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.

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

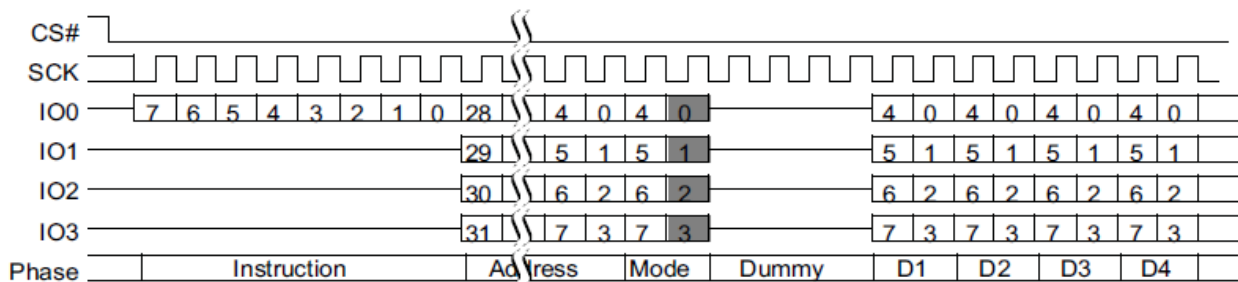
```

001     const cy_stc_smif_mem_cmd_t
002         S25FL512S_4byteaddr_SlaveSlot_0_readCmd =
003     {
004         /* The 8-bit command. 1 x I/O read command. */
005         .command = 0xECU,
006         /* The width of the command transfer. */
007         .cmdWidth = CY_SMIF_WIDTH_SINGLE,
008         /* The width of the address transfer. */
009         .addrWidth = CY_SMIF_WIDTH_QUAD,
010         /* The 8-bit mode byte. This value is 0xFFFFFFFF when there
011         is no mode present. */
012         .mode = 0x01U,
013         /* The width of the mode command transfer. */
014         .modeWidth = CY_SMIF_WIDTH_QUAD,
015         /* The number of dummy cycles. A zero value suggests no
016         dummy cycles. */
017         .dummyCycles = 4U,
018         /* The width of the data transfer. */
019         .dataWidth = CY_SMIF_WIDTH_QUAD
020     };
    
```

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



## Procedures

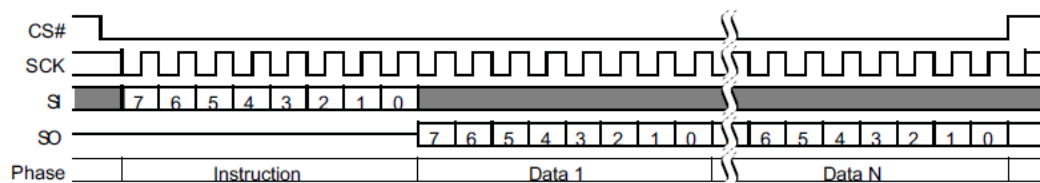


**Figure 8** Quad I/O Read command sequence (4-byte address, ECh or EBh)

## 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.



**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.

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

```

001     const cy_stc_smif_mem_cmd_t
002     S25FL512S_4byteaddr_SlaveSlot_0_readIDCmd =
003     {
004         /* The 8-bit command. 1 x I/O read command. */
005         .command = 0x9FU,
006         /* The width of the command transfer. */
007         .cmdWidth = CY_SMIF_WIDTH_SINGLE,
008         /* The width of the address transfer. */
009         .addrWidth = CY_SMIF_WIDTH_NA,
010         /* The 8-bit mode byte. This value is 0xFFFFFFFF when there
011         is no mode present. */
012         .mode = 0xFFFFFFFFFU,
013         /* The width of the mode command transfer. */
014         .modeWidth = CY_SMIF_WIDTH_NA,
015         /* The number of dummy cycles. A zero value suggests no
016         dummy cycles. */
017         .dummyCycles = 0U,
018         /* The width of the data transfer. */
019         .dataWidth = CY_SMIF_WIDTH_SINGLE
020     };

```

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

3. Add the read ID command to the command structure in *cycfg\_qspi\_memslot.c*.

**Code Listing 4**     **deviceCfg\_S25FL512S\_4byteaddr\_SlaveSlot\_0** in *cycfg\_qspi\_memslot.c*

```
001      const cy_stc_smif_mem_device_cfg_t
deviceCfg_S25FL512S_4byteaddr_SlaveSlot_0 =
002      {
003      ...
004      .readIDCmd =
(cy_stc_smif_mem_cmd_t*)&S25FL512S_4byteaddr_SlaveSlot_0_readIDCmd,
005      ...
006      };
```

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.

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

```
001      cy_en_smif_status_t Cy_SMIF_MemCmdReadID(SMIF_Type *base,
002                                                    cy_stc_smif_mem_config_t const
*memDevice,
003                                                    uint8_t* readBuff,
004                                                    uint32_t size,
005                                                    cy_stc_smif_context_t *context)
006      {
007          cy_en_smif_status_t result = CY_SMIF_BAD_PARAM;
008          cy_en_smif_slave_select_t slaveSelected;
009          cy_stc_smif_mem_device_cfg_t *device = memDevice->deviceCfg;
010          cy_stc_smif_mem_cmd_t *cmdReadID = device->readIDCmd;
011
012          if(NULL == cmdReadID)
013          {
014              result = CY_SMIF_CMD_NOT_FOUND;
015          }
016
017          else
018          {
019              slaveSelected = (0U == memDevice->dualQuadSlots)?
memDevice->slaveSelect :
020              (cy_en_smif_slave_select_t)memDevice->dualQuadSlots;
021
022              result = Cy_SMIF_TransmitCommand( base,
(uint8_t)cmdReadID->command,
023                                                  cmdReadID->cmdWidth,
CY_SMIF_CMD_WITHOUT_PARAM, CY_SMIF_CMD_WITHOUT_PARAM,
024                                                  CY_SMIF_WIDTH_NA,
slaveSelected, CY_SMIF_TX_NOT_LAST_BYTE,
025                                                  context);
026
027              if(CY_SMIF_SUCCESS == result)
028              {
```

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

```
029         result = Cy_SMIF_ReceiveData(base, readBuff, size,
030                                         cmdReadID->dataWidth, NULL, context);
031     }
032 }
033
034     return(result);
035 }
```

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.

### Code Listing 6 Cy\_SMIF\_MemReadID in *cy\_smif\_memslot.c*

```
001     cy_en_smif_status_t Cy_SMIF_MemReadID(SMIF_Type *base,
002     cy_stc_smif_mem_config_t const *memConfig,
003     uint8_t rxBuffer[],
004     cy_stc_smif_context_t *context)
005     {
006         cy_en_smif_status_t status = CY_SMIF_BAD_PARAM;
007         uint32_t chunk = 0UL;
008
009         CY_ASSERT_L1(NULL != memConfig);
010         CY_ASSERT_L1(NULL != rxBuffer);
011
012         if(1)
013         {
014             /* SMIF can read only up to 65536 bytes in one go. Split
015             the larger read into multiple chunks */
016             while (length > 0UL)
017             {
018                 /* Get the number of bytes which can be read during
019                 one operation */
020                 chunk = (length > SMIF_MAX_RX_COUNT) ?
021                 (SMIF_MAX_RX_COUNT) : length;
022
023                 /* Send the command to read data from the external
024                 memory to the rxBuffer array */
025                 status = Cy_SMIF_MemCmdReadID(base, memConfig,
026                 (uint8_t *)rxBuffer,
027                 chunk, context);
028
029                 if(CY_SMIF_SUCCESS == status)
030                 {
031                     /* Wait until the SMIF block completes receiving
032                     data */
033                     status = PollTransferStatus(base,
034                     CY_SMIF_REC_CMPLT, context);
035                 }
036
037                 if(CY_SMIF_SUCCESS != status)
038                 {
039                     break;
040                 }
041             }
042         }
043     }
```

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Procedures

```
033         }
034
035         /* Recalculate the next rxBuffer offset */
036         length -= chunk;
037         rxBuffer = (uint8_t *)rxBuffer + chunk;
038     }
039 }
040
041     return status;
042 }
```

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.

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

```
001     cy_rslt_t cy_serial_flash_qspi_readID(size_t length, uint8_t*
    buf)
002     {
003         cy_rslt_t result_mutex_rel = CY_RSLT_SUCCESS;
004
005         cy_rslt_t result = _mutex_acquire();
006
007         if (CY_RSLT_SUCCESS == result)
008         {
009             // Cy_SMIF_MemReadID() returns error if (addr + length)
    > total flash size.
010             result = (cy_rslt_t)Cy_SMIF_MemReadID(qspi_obj.base,
    qspi_block_config.memConfig[MEM_SLOT],
011                                                     buf, length,
    &qspi_obj.context);
012             result_mutex_rel = _mutex_release();
013         }
014
015         /* Give priority to the status of SMIF operation when both
    SMIF operation
016         * and mutex release fail.
017         */
018         return ((CY_RSLT_SUCCESS == result) ? result_mutex_rel :
    result);
019     }
```

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*

```
001     int main(void)
002     {
003         ...
004         printf("\r\n1. Reading device ID\r\n");
```

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

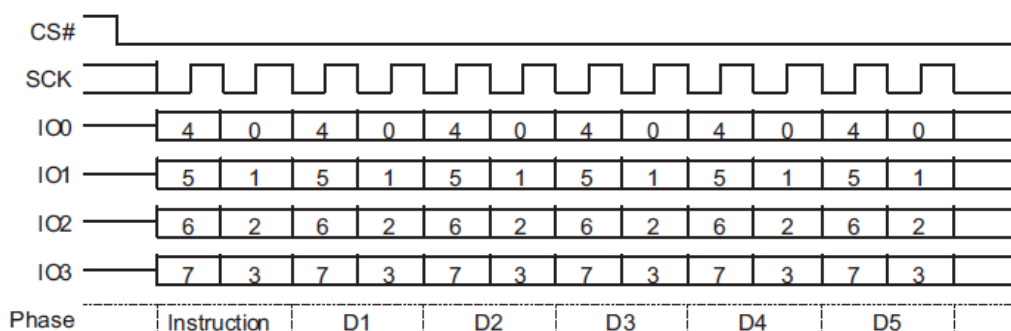


## Procedures

```
005         result = cy_serial_flash_qspi_readID(DEVICE_ID, rx_buf);
006         check_status("Reading Device ID failed", result);
007         printf("Byte Address: Data\r\n");
008         for(uint8_t i = 0; i < DEVICE_ID; i++)
009             printf("0x%02dh : 0x%02Xh \r\n", i, rx_buf[i]);
010     ...
011 }
```

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.



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

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

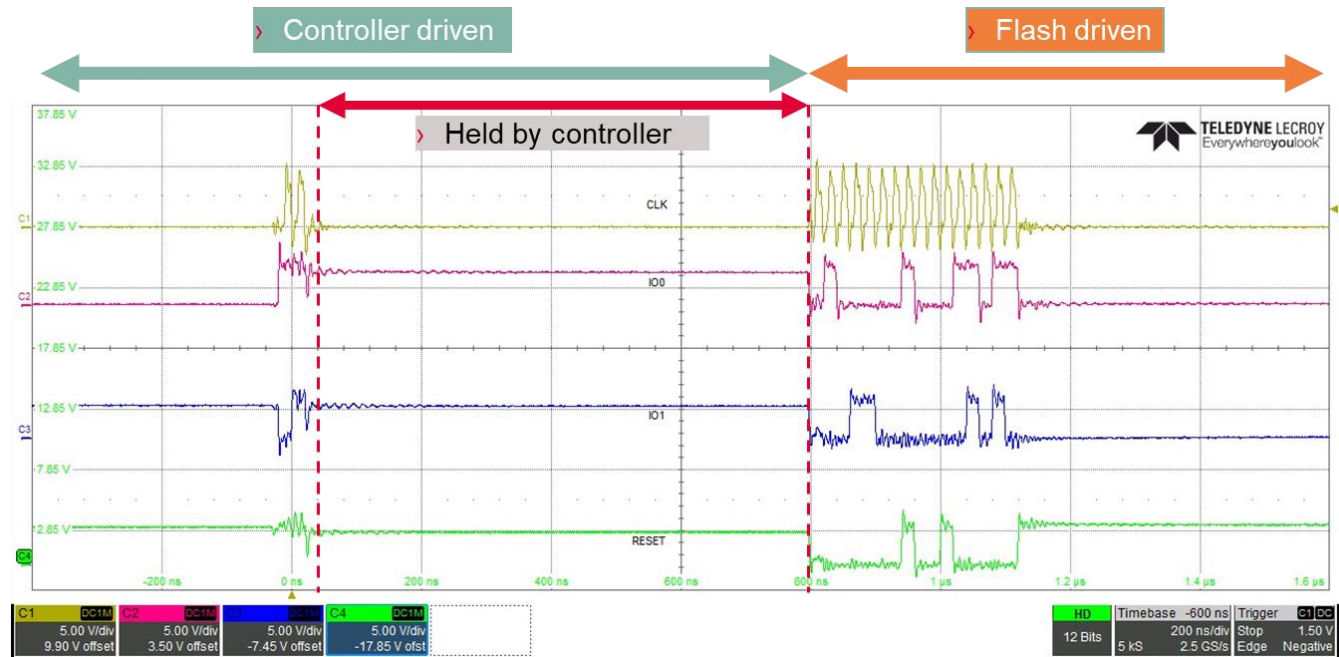
```
001     const cy_stc_smif_mem_cmd_t
002         S25FL512S_4byteaddr_SlaveSlot_0_readIDQPICmd =
003     {
004         /* The 8-bit command. 1 x I/O read command. */
005         .command = 0x9FU,
006         /* The width of the command transfer. */
007         .cmdWidth = CY_SMIF_WIDTH_QUAD,
008         /* The width of the address transfer. */
009         .addrWidth = CY_SMIF_WIDTH_NA,
010         /* The 8-bit mode byte. This value is 0xFFFFFFFF when there
011         is no mode present. */
012         .mode = 0xFFFFFFFFFU,
013         /* The width of the mode command transfer. */
014         .modeWidth = CY_SMIF_WIDTH_NA,
015         /* The number of dummy cycles. A zero value suggests no
016         dummy cycles. */
017         .dummyCycles = 0U,
018         /* The width of the data transfer. */
019         .dataWidth = CY_SMIF_WIDTH_QUAD
020     };
```



### 3 PSoC™ 6 MCU board operation

**Figure 6** 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™ 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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

## PSoC™ 6 MCU board operation

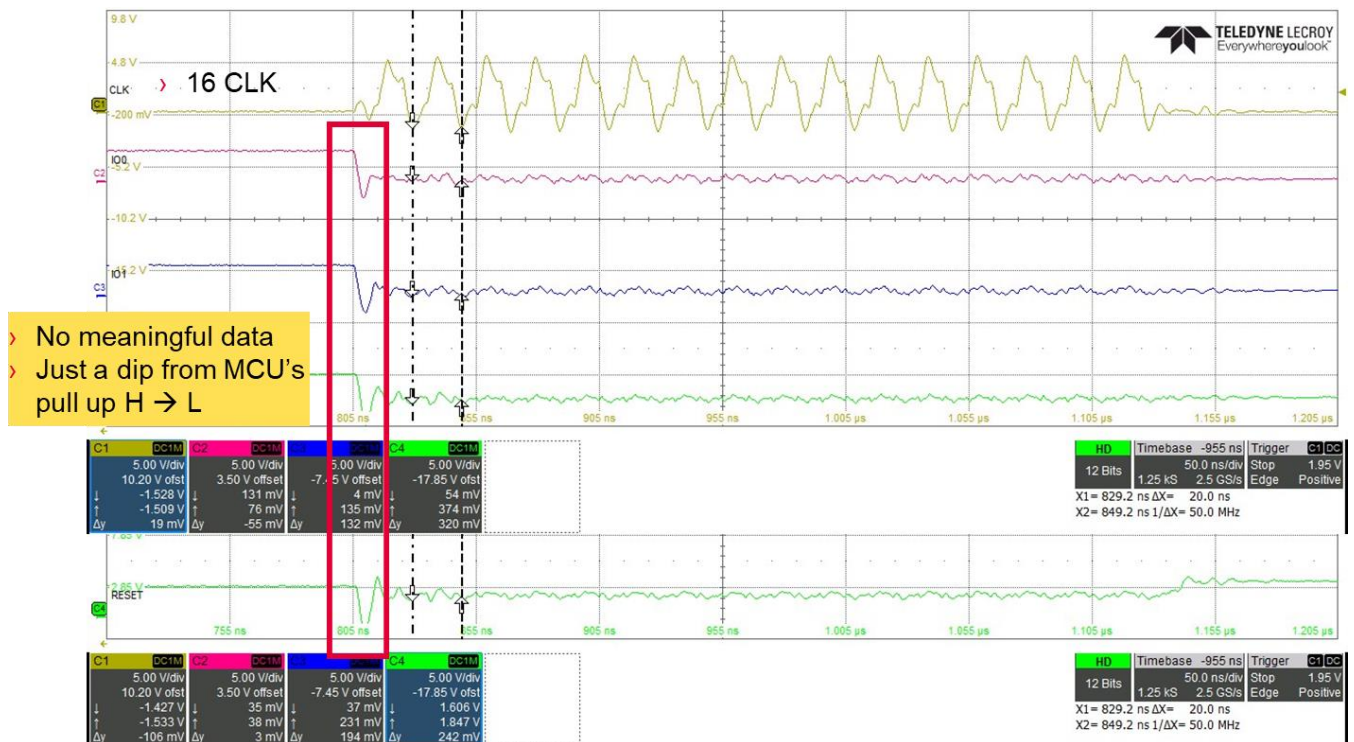


Figure 12 No meaningful dip at the beginning of the data phase

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory

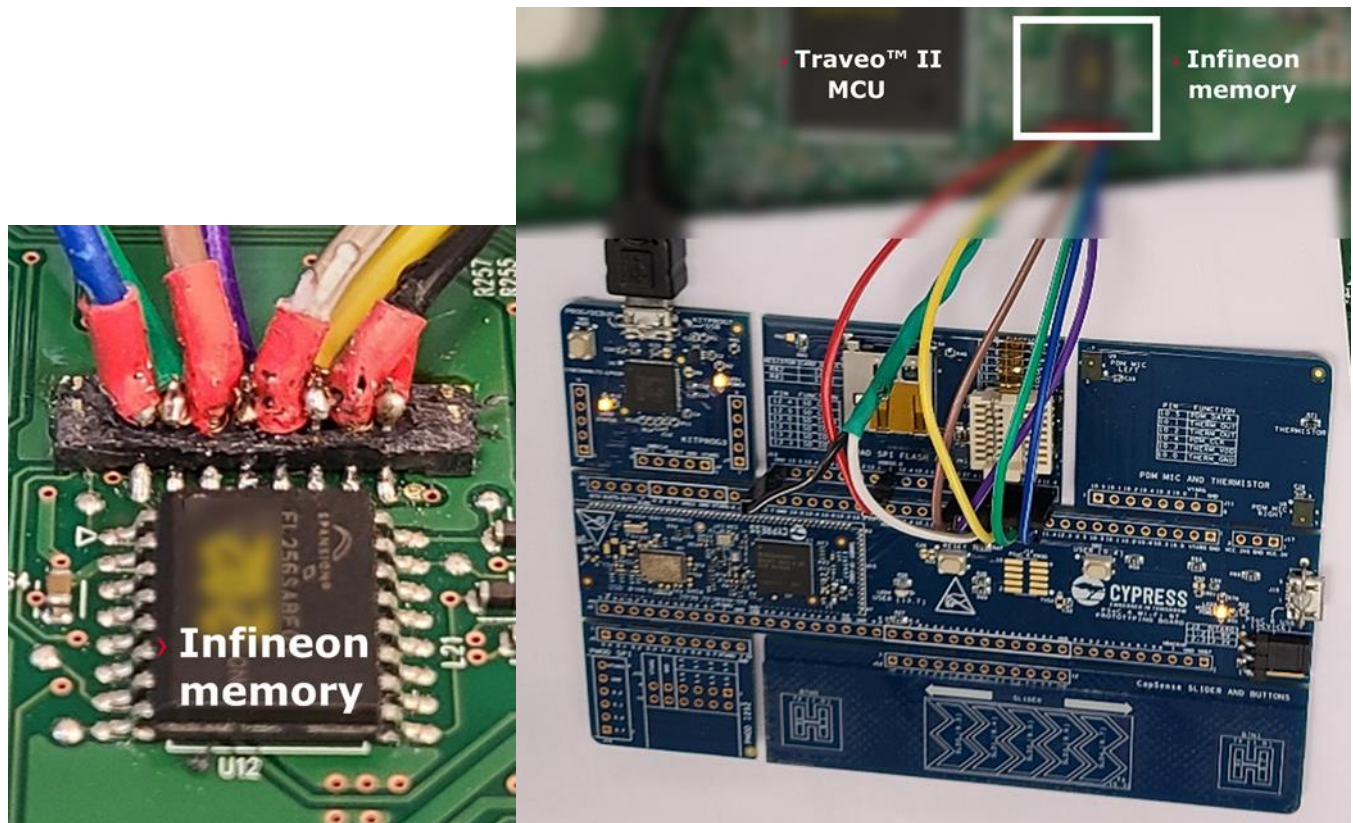


## Use cases

### 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™ 6 MCU board to avoid damage on the target board.



**Figure 13** PSoC™ 6 MCU board connected externally to a customer's application board

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

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## Use cases

```
*****
***** Customer's S25FL256S *****

0. Reading ID
Reading ID
2 (6 bytes):
-----
0x01 0x02 0x19 0x40 0x00 0x80
2. Reading External memory data address: 0x0000
Received Data (64 bytes):
-----
0x54 0x41 0x43 0x53 0x31 0x38 0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30
0x30 0x30 0x30 0x30 0x30 0x35 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33
0x01 0x00 0x02 0x00 0x04 0x00 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00
0x0A 0x00 0x0B 0x00 0x0C 0x00 0x0D 0x00 0x0E 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00

3. Reading External memory data address: 0x20000
Received Data (64 bytes):
-----
0x54 0x41 0x43 0x53 0x31 0x38 0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30
0x30 0x30 0x30 0x30 0x30 0x35 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33
0x01 0x00 0x02 0x00 0x04 0x00 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00
0x0A 0x00 0x0B 0x00 0x0C 0x00 0x0D 0x00 0x0E 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00

4. Reading External memory data address: 0x40000
Received Data (64 bytes):
-----
0x54 0x41 0x43 0x53 0x31 0x38 0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30
0x30 0x30 0x30 0x30 0x30 0x35 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33
0x01 0x00 0x02 0x00 0x04 0x00 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00
0x0A 0x00 0x0B 0x00 0x0C 0x00 0x0D 0x00 0x0E 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00

5. Reading External memory data address: 0x60000
Received Data (64 bytes):
-----
0x54 0x41 0x43 0x53 0x31 0x38 0x30 0x31 0x30 0x32 0x30 0x30 0x34 0x30 0x30 0x30
0x30 0x30 0x30 0x30 0x30 0x35 0x32 0x30 0x30 0x39 0x30 0x32 0x30 0x38 0x38 0x33
0x01 0x00 0x02 0x00 0x04 0x00 0x05 0x00 0x06 0x00 0x07 0x00 0x08 0x00 0x09 0x00
0x0A 0x00 0x0B 0x00 0x0C 0x00 0x0D 0x00 0x0E 0x00 0x0F 0x00 0x10 0x00 0x11 0x00 0x12 0x00 0x13 0x00
[]
```

Customer's board

**Figure 14** Read ID operation result

## Conclusion

### 5 Conclusion

Even though CY8CPROTO-062-4343W PSoC™ 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™ 6 MCU-based flash evaluation tool.



## References

1. PSoC™ 6 Wi-Fi Bluetooth® prototyping kit (CY8CPROTO-062-4343W)  
<https://www.cypress.com/documentation/development-kitsboards/psoc-6-wi-fi-bt-prototyping-kit-cy8cproto-062-4343w>
2. CY8CPROTO-062-4343W PSoC™ 6 Wi-Fi Bluetooth® 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™ software download link  
[https://www.cypress.com/products/modustoolbox#tabs-0-bottom\\_side-6](https://www.cypress.com/products/modustoolbox#tabs-0-bottom_side-6)
6. ModusToolbox™ software installation guide  
<https://www.cypress.com/file/520241/download>
7. ModusToolbox™ software user guide  
<https://www.cypress.com/file/520251/download>

# Modifying CY8CPROTO-062-4343W PSoC™ 6 MCU board to work with an external flash memory



## 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**

**© 2021 Infineon Technologies AG.**

**All Rights Reserved.**

**Do you have a question about this document?**

**Go to [www.cypress.com/support](http://www.cypress.com/support)**

**Document reference**

**002-33648 Rev.\*\***

#### 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.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office ([www.infineon.com](http://www.infineon.com)).

#### 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.