Execute-in-Place QSPI PSRAM with PSoC62 Microcontroller

Gintaras Drukteinis,
RUTRONIK Electronics Worldwide, Kaunas, Lithuania

***Abstract* — XIP (Execute-in-Place) is a hardware method to access external memory seemingly in the same way as the internal memory of the MCU is accessed. This feature allows having more optimized firmware and shorter development time. The AP Memory PSRAM devices do not require any additional commands for enabling writing and reading data as Flash or FRAM devices would normally require. The PSRAMs are easily integrated and used in combination with PSoC6 MCUs and therefore it is explained how to do it in this application note.**

***Index Terms* — Microcontroller (MCU), Phase Locked Loop (PLL), Frequency Locked Loop (FLL).**

I. Introduction

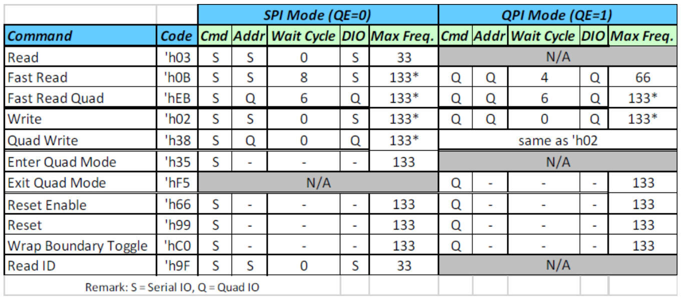
After the MCU is configured to operate with the external PSRAM in XIP mode - all the external memory addresses are mapped into MCU's address space by the SMIF controller and data bytes can be accessed one-by-one or in series. Only two commands are necessary to access the APS6404L-3SQR-ZR PSRAM – "Write" and "Read" and they are issued by the SMIF peripheral automatically each time the memory-mapped location is being accessed by the processor.

II. Hardware

The APS6404L-3SQR-ZR 64Mbit PSRAM has a Quad-SPI interface with a clock rate up to 109MHz[1] meanwhile PSoC62 Quad-SPI supports clock rates only up to 80 MHz and therefore the memory may never reach its maximum speed capability with the PSoC6. Anyway, with only 80MHz in a Single-Data-Rate mode, the PSoC6245 may still reach the data throughput up to 320Mbps. The actual APS6404L-3SQR-ZR memory and PSoC6245 microcontroller combination were tested at full speed on the development platform RutDevKit-PSoC62. Neither the APS6404L-3SQR-ZR nor the PSoC6245 support the Double-Data-Rate data exchange.

The QSPI interface is handled by the hardware block called Serial Memory Interface or SMIF. This versatile peripheral may also have On-the-fly encryption and decryption to protect sensitive external memory content and a 4-KB cache to increase the performance while in XIP mode.

The performance of the data exchange between MCU and PSRAM also slightly varies depending on what "Read/Write" commands are used. The APS6404L-3SQR-ZR has 3 types of commands for the read procedure, 2 different commands for the writing procedure and all these commands may have two modes: "SPI" or "QPI". For "QPI" it is required to go through special command to have the memory configured for this type of communication, otherwise, it works in "SPI" mode from the start-up, by default.



		SPI Mode (QE=0)					QPI Mode (QE=1)				
Command	*Code*	*Cmd*	*Addr*	*Wait Cycle*	*DIO*	*Max Freq.*	*Cmd*	*Addr*	*Wait Cycle*	*DIO*	*Max Freq.*
Read	'h03	S	S	0	S	33	N/A				
Fast Read	'h0B	S	S	8	S	133*	Q	Q	4	Q	66
Fast Read Quad	'hEB	S	Q	6	Q	133*	Q	Q	6	Q	133*
Write	'h02	S	S	0	S	133*	Q	Q	0	Q	133*
Quad Write	'h38	S	Q	0	Q	133*	same as 'h02				
Enter Quad Mode	'h35	S	-	-	-	133	N/A				
Exit Quad Mode	'hF5	N/A					Q	-	-	-	133
Reset Enable	'h66	S	-	-	-	133	Q	-	-	-	133
Reset	'h99	S	-	-	-	133	Q	-	-	-	133
Wrap Boundary Toggle	'hC0	S	-	-	-	133	Q	-	-	-	133
Read ID	'h9F	S	S	0	S	33	N/A				

Remark: S = Serial IO, Q = Quad IO

Fig. 1 The list of all the commands the APS6404L-3SQR-ZR may handle.

The impedance for all the QSPI traces is kept as close as possible to the 50Ω value and the length difference between all the QSPI traces is equalized down to 8mm to be sure the PCB design issues will not impact the performance of both parts. The RutDevKit-PSoC62 PCB has 4 layers. Every QSPI trace has a thickness of 35um, a with of 0.15mm, and it is routed over the ground signal plate at the distance of 0.12mm. Though 22Ω series resistors were used to avoid ringing in QSPI lines, no serious signal attenuation was noticed during the tests.

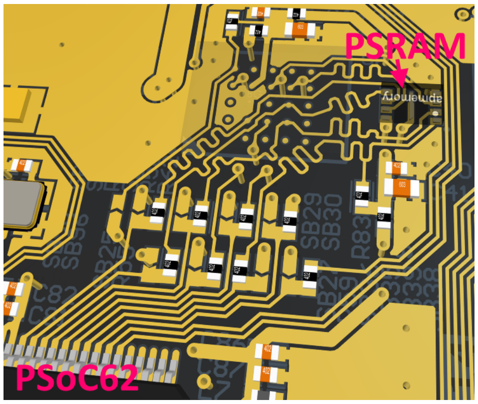
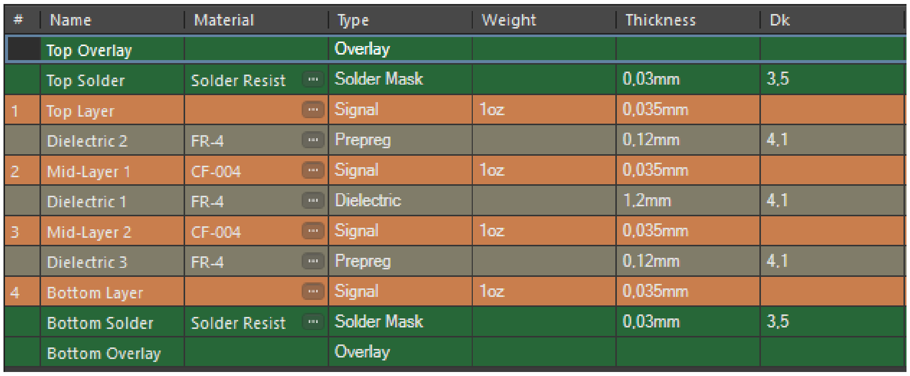


Fig. 2 Traces equalization between PSoC62 PSRAM

[1] The power supply voltage is 3.3V. At the 3.0V – 133MHz maximum frequency is available.

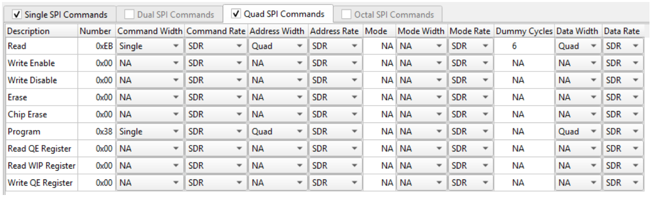


#	Name	Material	Type	Weight	Thickness	Dk
	Top Overlay		Overlay			
	Top Solder	Solder Resist	Solder Mask		0,03mm	3,5
1	Top Layer		Signal	1oz	0,035mm	
	Dielectric 2	FR-4	Prepreg		0,12mm	4,1
2	Mid-Layer 1	CF-004	Signal	1oz	0,035mm	
	Dielectric 1	FR-4	Dielectric		1,2mm	4,1
3	Mid-Layer 2	CF-004	Signal	1oz	0,035mm	
	Dielectric 3	FR-4	Prepreg		0,12mm	4,1
4	Bottom Layer		Signal	1oz	0,035mm	
	Bottom Solder	Solder Resist	Solder Mask		0,03mm	3,5
	Bottom Overlay		Overlay			

Fig. 3 The RutDevKit-PSoC62 PCB Stack-up.

III. Firmware Development Tools

The ModusToolbox™ IDE is a free development environment provided by Infineon[2] and it has the integrated tool "QSPI Configurator" that is used for code generation of required QSPI external memories. The APS6404L-3SQR-ZR configuration is not in the database by default so it has to be created according to the device datasheet and added manually. The firmware example with the APS6404L-3SQR-ZR configuration file is provided for the RutDevKit-PSoC62 board by Rutronik.



☑ Single SPI Commands | □ Dual SPI Commands | ☑ Quad SPI Commands | □ Octal SPI Commands

Description	Number	Command Width	Command Rate	Address Width	Address Rate	Mode	Mode Width	Mode Rate	Dummy Cycles	Data Width	Data Rate
Read	0xEB	Single	SDR	Quad	SDR	NA	NA	SDR	6	Quad	SDR
Write Enable	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Write Disable	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Erase	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Chip Erase	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Program	0x38	Single	SDR	Quad	SDR	NA	NA	SDR	NA	Quad	SDR
Read QE Register	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Read WIP Register	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR
Write QE Register	0x00	NA	SDR	NA	SDR	NA	NA	SDR	NA	NA	SDR

Fig. 4 The APS6404L-3SQR-ZR QSPI configuration

The QSPI Configurator creates `cycfg_qspi_memslot.c` and `cycfg_qspi_memslot.h` memory configuration files with configuration structures within that can be used with a SMIF driver API. The QSPI Configurator enables to have a QSPI interface, memory-mapped address, and other settings generated for the multiple QSPI slots. Generally, the memory configuration can be reached using the SMIF block configuration structure:

```
const cy_stc_smif_block_config_t smifBlockConfig
```

The microcontroller-related configuration can be done using a "Device Configurator". This tool simplifies the configuration and control of the peripherals. For instance, the PSoC6245 QSPI peripheral needs to be clocked no more than 80 MHz for stable operation. The whole system clock configuration is displayed in a "System" tab of the "Device Configurator" program window. A part of the system clock layout including the QSPI peripheral clock is shown in (Fig. 5). The PLL is configured to generate 150MHz which is connected to the microcontroller cores together with some of the peripherals and the FLL 80MHz is connected to the QSPI peripheral.

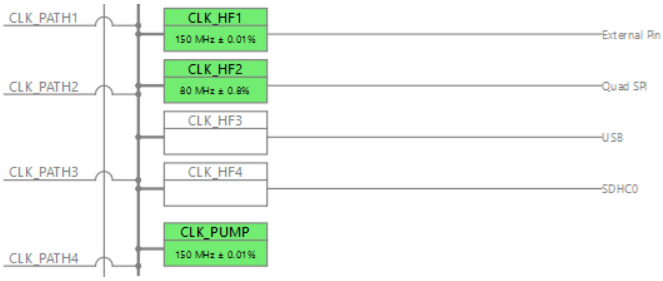


Fig. 5 QSPI clock configuration part.

IV. Firmware Example

The firmware example with all necessary setup is provided by Rutronik. The purpose of this firmware is to test the APS6404L-3SQR-ZR and PSoC6245 compatibility. Additionally, two PSRAM access methods are demonstrated. The first one is indirect access using SMIF driver functions such as:

```
Cy_SMIF_MemWrite();
Cy_SMIF_MemRead();
```

The second access method is a direct XIP mode which is entered after this SMIF driver function is executed:

```
Cy_SMIF_SetMode(base, CY_SMIF_MEMORY);
```

The QSPI peripheral is initialized using the HAL library and the rest of the SMIF peripheral initializations are done using the PDL library.

After the QSPI peripheral and SMIF block initialization is done, the PSRAM is tested using SMIF driver functions by writing and when reading a data package. If the data written to the PSRAM matches what has been read, the program continues with the XIP mode setup. After the XIP mode has been enabled, the changing pattern is written, read, and compared indefinitely.

[2] Previously Cypress.

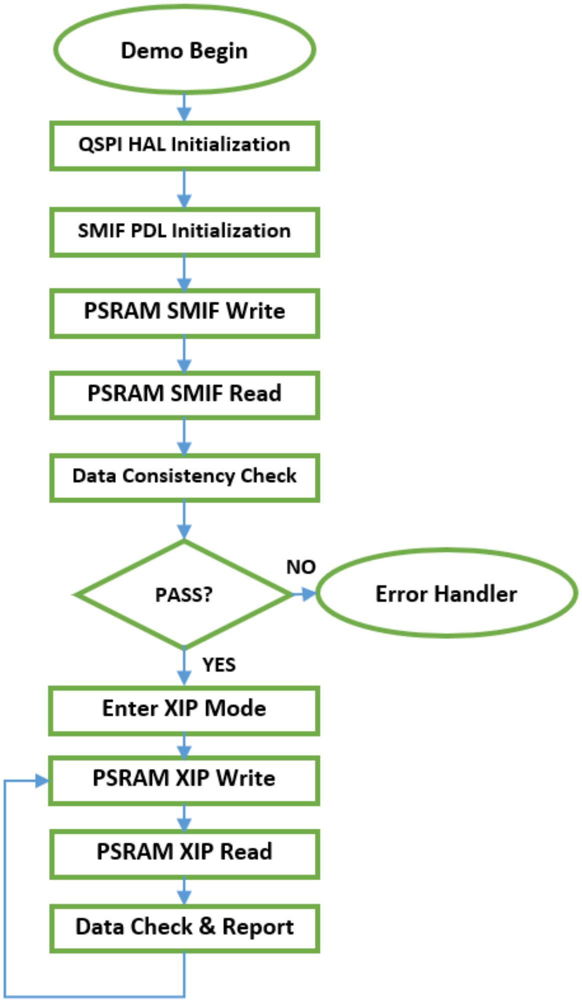
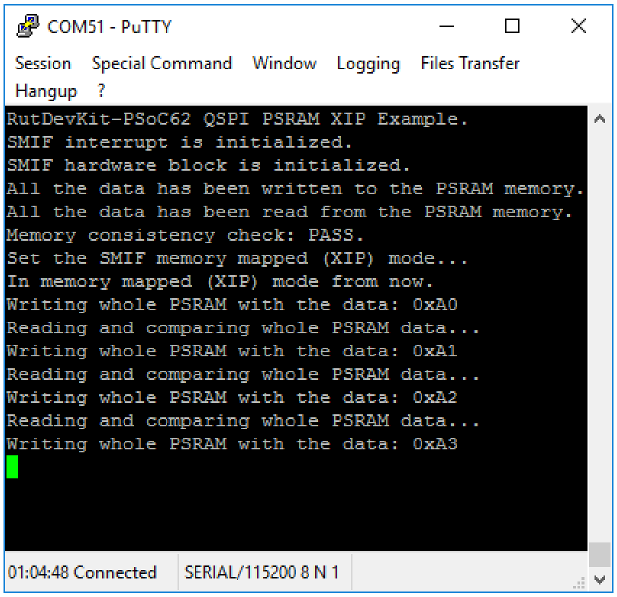


Fig. 6 PSRAM PSoC62 demonstration algorithm.

All the progress can be traced using the terminal since the debugging information is printed out via the KitProg3 UART.



```
RutDevKit-PSoC62 QSPI PSRAM XIP Example.
SMIF interrupt is initialized.
SMIF hardware block is initialized.
All the data has been written to the PSRAM memory.
All the data has been read from the PSRAM memory.
Memory consistency check: PASS.
Set the SMIF memory mapped (XIP) mode...
In memory mapped (XIP) mode from now.
Writing whole PSRAM with the data: 0xA0
Reading and comparing whole PSRAM data...
Writing whole PSRAM with the data: 0xA1
Reading and comparing whole PSRAM data...
Writing whole PSRAM with the data: 0xA2
Reading and comparing whole PSRAM data...
Writing whole PSRAM with the data: 0xA3
```

01:04:48 Connected SERIAL/115200 8 N 1

Fig. 7 Debugging information of the program.

V. Test Results

No error messages have ever been noticed running the firmware with 10 units of RutDevKit-PSoC62 prototypes. Clocking PSRAM with 80MHz, which is a full speed of PSoC6245 QSPI peripheral, the read operation takes approximately 800 ms and the write operation approximately 2200 ms for the whole 64Mbits of memory to complete.

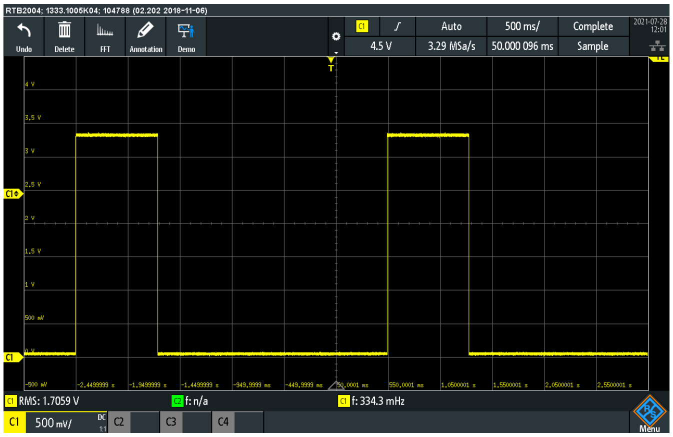


Fig. 8 LED blinking indicates the write operation. Write – Signal Low; Read – Signal High.

VI. Summary

The AP Memory APS6404L-3SQR-ZR together with Infineon microcontroller PSoC6245AZI-S3D72 was implemented and tested on the development platform RutDevKit-PSoC62.

References

[1] “APS6404L-3SQN QSPI PSRAM” Datasheet, by AP Memory (April 2020).

[2] “SQPI PSRAM PCB Layout Guidelines” Guideline, by AP Memory (August 2019).

[3] “ PSoC 6 MCU: CY8C6xx5 Architecture Technical Reference Manual (TRM)” User Manual, by Infineon (July 2020).

[4] “ PSoC 6 MCU: CY8C62X5 Datasheet” Datasheet, by Infineon (November 2020).

Contact:
Gintaras Drukteinis
Technical Support Engineer
RUTRONIK Elektronische Bauelemente GmbH
Jonavos g. 30
44262 Kaunas
Lithuania
gdr@rutronik.com
www.rutronik.com