SMART ARM-based Microprocessor

SAMA5D2 Quad SPI (QSPI) Performance

APPLICATION NOTE

Introduction

The Atmel® | SMART SAMA5D2 Series is a high-performance, power-efficient embedded MPU based on the ARM® Cortex®-A5 processor. It features 2 Quad SPI (QSPI) interfaces which enable booting from QSPI and eXecuting In Place (XIP) with QSPI.

This Application Note describes how to run an RTOS from QSPI, measures the real-time performance of the RTOS, and shows the factors which affect performance. The purpose is to show the performance of QSPI and to provide an option to run a DDR-less system on SAMA5D2.

QSPI features:

- Master SPI Interface
  - Programmable Clock Phase and Clock Polarity
  - Programmable Transfer Delays Between Consecutive Transfers, Between Clock and Data, Between Deactivation and Activation of Chip Select
- SPI Mode
  - Interface to Serial Peripherals such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - 8-bit/16-bit/32-bit Programmable Data Length
- Serial Memory Mode
  - Interface to Serial Flash Memories Operating in Single-bit SPI, Dual SPI and Quad SPI
  - Supports “eXecute In Place” (XIP) — Code Execution by the System Directly from a Serial Flash Memory
  - Flexible Instruction Register for Compatibility with All Serial Flash Memories
  - 32-bit Address Mode (default is 24-bit address) to Support Serial Flash Memories Larger than 128 Mbit
  - Continuous Read Mode
  - “On-The-Fly” Scrambling/Unscrambling
- Connection to DMA Channel Capabilities Optimizes Data Transfers
  - One Channel for the Receiver, One Channel for the Transmitter
- Register Write Protection

Table of Contents

1. Hardware and Software Environment

1.1. SAMA5D2 Xplained Ultra Evaluation Kit

The Atmel | SMART SAMA5D2 Xplained Ultra Evaluation Kit (“SAMAD2-XULT” in this document) is ideal for evaluating and prototyping the Atmel ARM Cortex-A5-based microprocessors in the SAMA5D2x series.

In this Application Note, the SAMA5D2-XULT board is used with the Micron QSPI module N25Q128A13ESE40G.

1.2. FreeRTOS Introduction

FreeRTOS is a market-leading, de facto standard and cross-platform Real-Time Operating System (RTOS) offering the following advantages:

- Single and independent solution for many different architectures and development tools
- Minimal ROM, RAM and processing overhead. Typically, an RTOS kernel binary image is in the region of 6 to 12 Kbytes.
- Simplicity - the core of the RTOS kernel is contained in only 3 C files. The .zip file download includes many files, but most relate to the numerous demonstration applications.
- Comes with a porting, platform development, or application development service should it be required
- Contains a preconfigured example for each port. No need to figure out how to setup a project - just download and compile
- Excellent, monitored, and active free support forum
- Sample documentation provided
- Very scalable, simple and easy to use

For more information and a demo of FreeRTOS, visit www.freertos.org.

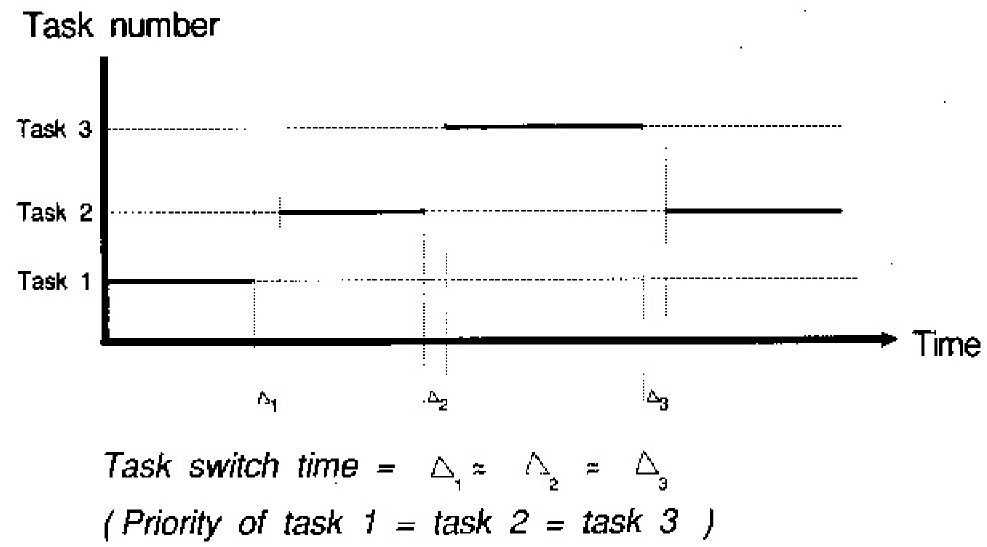
1.3. Real-time Benchmark: Rhealstone

Rhealstone is a real-time benchmark which was proposed by Rabindra P.Kar and Kent Porter in 1989. [Refer to Kar Rabindra P., *Implementing the Rhealstone Real-Time Benchmark*, Dr.Dobb’s Journal, s 46-55, 100-104, April 1990].

Rhealstone is composed of six operations:

- Task Switch Time
  The average time it takes the system to switch between two independent and active tasks with equal priority.

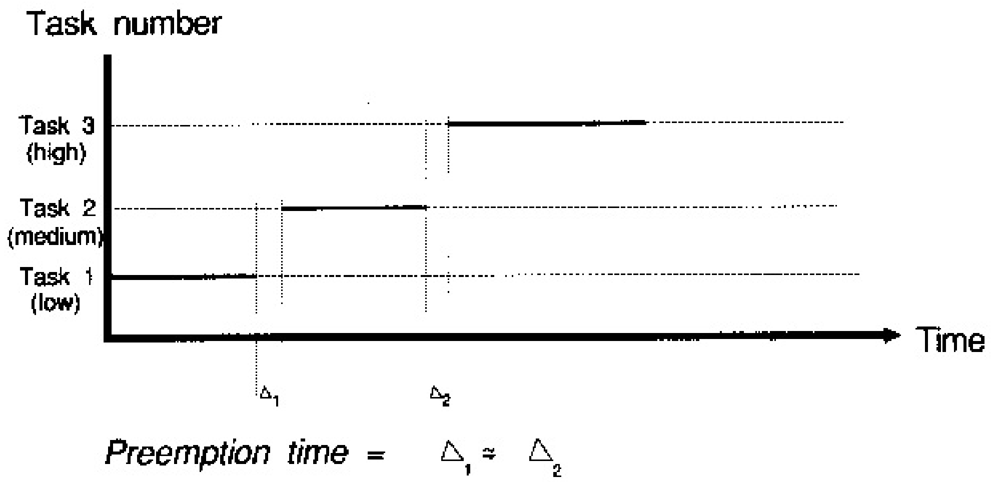
Figure 1-1. Task Switch Time


Task number
Task 3
Task 2
Task 1
Time
Δ1
Δ2
Δ3
Task switch time = Δ1 ≈ Δ2 ≈ Δ3
(Priority of task 1 = task 2 = task 3)

- Task Pre-emption Time

  The average time it takes for a task of higher priority to take control of the system from a running task of lower priority.

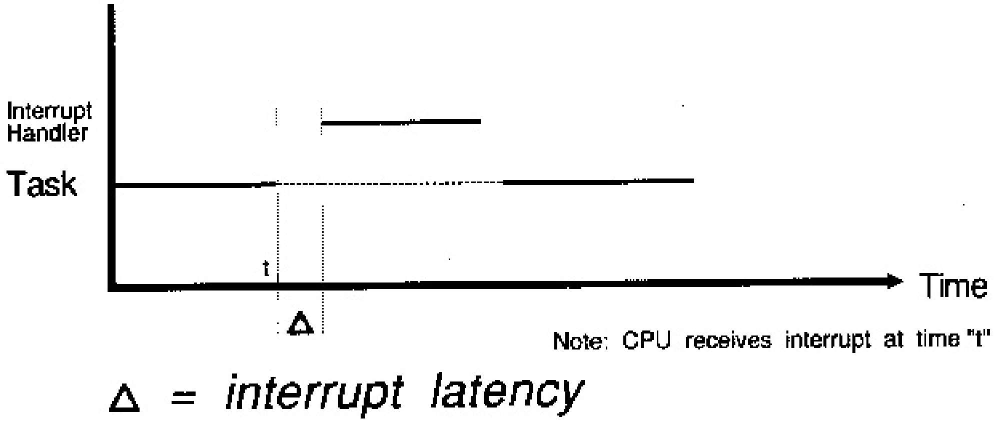
Figure 1-2. Task Pre-emption Time



- Interrupt Latency

  The time lapse between the reception of the interrupt request by the CPU and the execution of the first instruction in the interrupt service routine.

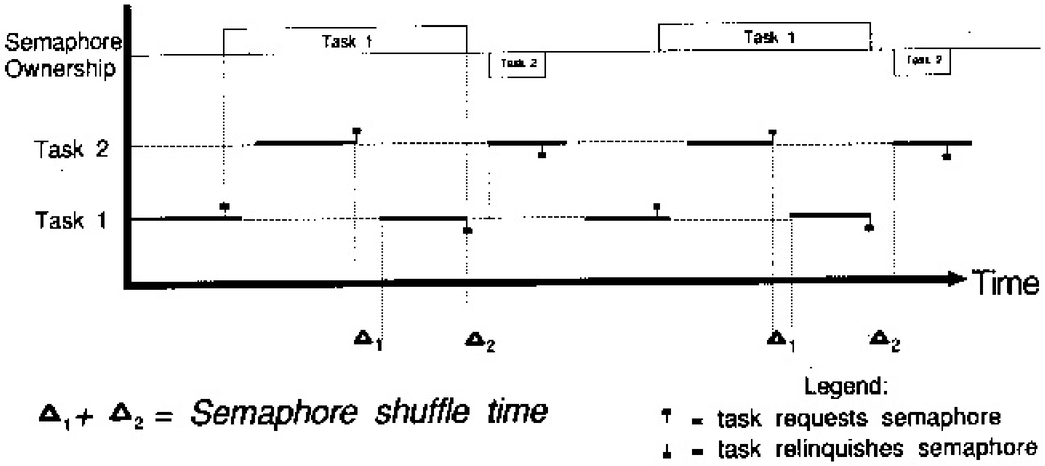
Figure 1-3. Interrupt Latency



- Semaphore Shuffling Time

  The time delay between a task's release of a semaphore until the activation of another task waiting for the semaphore.

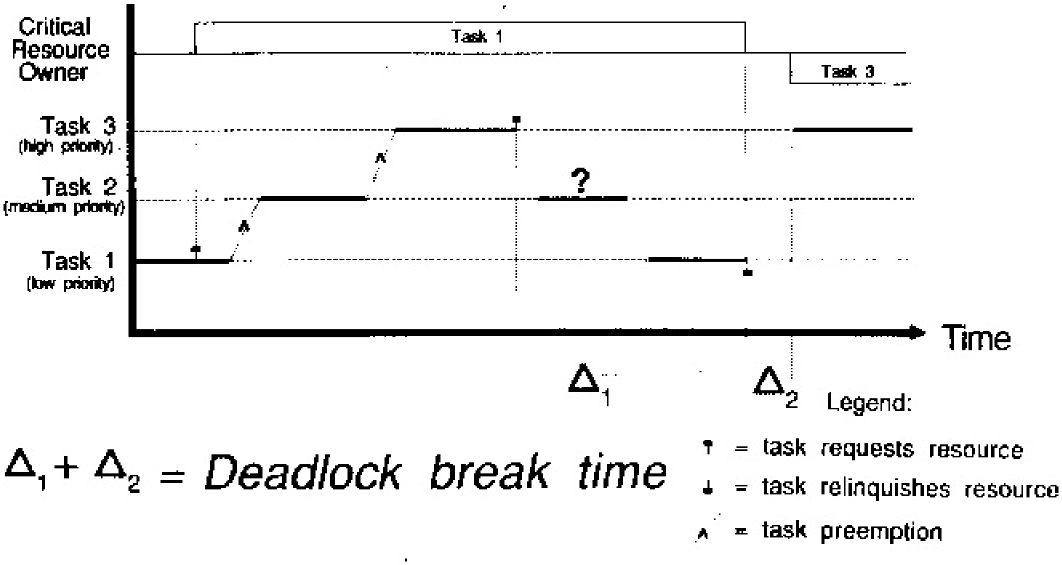
Figure 1-4. Semaphore Shuffling Time


Semaphore Ownership
Task 1
Task 2
Task 1
Task 2
Task 2
Task 1
Time
Δ1
Δ2
Δ1
Δ2
Δ1 + Δ2 = Semaphore shuffle time
Legend:
↑ = task requests semaphore
↓ = task relinquishes semaphore

- Deadlock Breaking Time

  The time required by the system to break a deadlock.

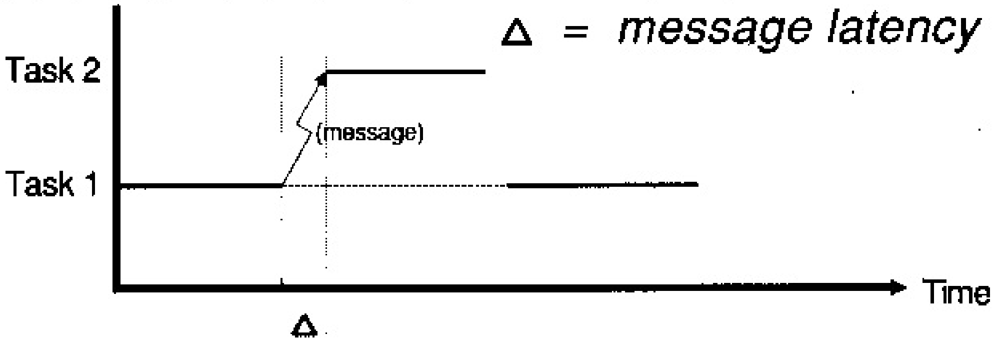
Figure 1-5. Deadlock Breaking Time



- Intertask Messaging Latency

  The time used to send a non zero-length data message from one task to another.

Figure 1-6. Intertask Messaging Latency



Task Switch Time, Task Pre-emption Time and Interrupt Latency are very important indicators of context switching efficiency and real-time performance of a multi-task system, which is mostly affected by the system CPU’s architecture, instruction set, and the memory access speed.

Semaphore Shuffling Time, Deadlock Breaking Time and Intertask Messaging Latency are mostly relative to the message communication mechanism of RTOS which is largely determined by the design of the RTOS.

Since the aim of this application note is to compare the memory performance among QSPI, SRAM and DDR, we choose the first three Rhealstone operations (Task Switch, Task Pre-emption and Interrupt Latency) to benchmark the performance of memories.

2. SAMA5D2 QSPI Introduction

2.1. QSPI – SPI Mode

The QSPI can be set as normal SPI mode to interface to serial peripherals such as ADCs, DACs, LCD controllers, CAN controllers and sensors.

In SPI mode, the QSPI acts as a regular SPI Master. To activate this mode, the SMM bit in Mode Register (QSPI_MR) must be cleared.

Data transfers can be performed by interrupt or polling method. The speed can of course be improved by enabling DMA transfer.

For a more detailed description on how QSPI works in SPI mode, refer to the product datasheet.

2.2. QSPI – Serial Memory Mode

The QSPI can also be set as Serial Memory mode to interface to serial Flash memories. The QSPI can control the serial Flash by specific commands and can also allow the CPU to execute the code from the serial Flash (XIP, or eXecute In Place).

In this mode, provided the TFRTYP field in QSPI_IFR register is correctly set to either TRSFR_READ or TRSFR_WRITE:

- If QSPI_MR.SMRM = 0, data can be read or written directly to the QSPI memory space (0x9000_0000~0x9800_0000 and 0xD000_0000~0xD800_0000)
- If QSPI_MR.SMRM = 1, data can be read or written by accessing QSPI_RDR or QSPI_TDR (refer to "Instruction Frame Transmission" in section Quad Serial Peripheral Interface (QSPI) of the SAMA5D2 datasheet)

2.2.1. Serial Flash Memory Control

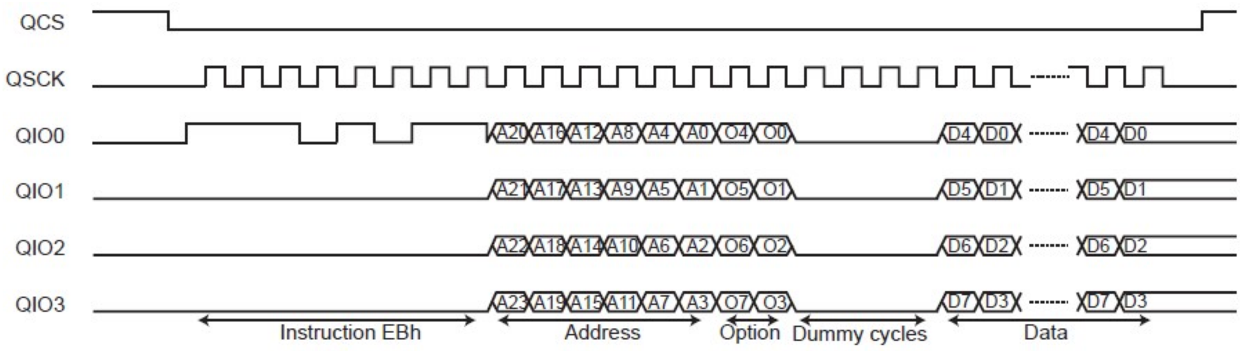
In order to control serial Flash memories, the QSPI is able to send instructions via the SPI bus (ex: READ, PROGRAM, ERASE, LOCK, etc.). Because the instruction set implemented in serial Flash memories is memory vendor-dependent, the QSPI includes a complete Instruction Frame Register (QSPI_IFR), which makes it very flexible and compatible with all serial Flash memories.

An instruction frame includes:

- An instruction code (size: 8 bits). The instruction is optional in some cases (such as Continuous Read mode)
- An address (size: 24 bits or 32 bits). The address is optional but is required by instructions such as READ, PROGRAM, ERASE, LOCK. By default the address is 24 bits long, but it can be 32 bits long to support serial Flash memories larger than 128 Mbits (16 Mbytes).
- An option code (size: 1/2/4/8 bits). The option code is not required, but it is useful to activate the Memory XIP mode or the Continuous Read mode (see Sub-section "Continuous Read Mode" in Section Quad SPI Interface (QSPI) of the SAMA5D2 datasheet) for READ instructions, in some serial Flash memory devices. These modes improve the data read latency.
- Dummy cycles. Dummy cycles are optional but required by some READ instructions.
- Data bytes. Data bytes are optional, but can be used for data transfer instructions such as READ or PROGRAM.

Refer to the following figure.

Figure 2-1. Typical Instruction Frame Transfer Waveform


QCS
QSCK
QIO0
A20 A16 A12 A8 A4 A0 O4 O0 D4 D0 D4 D0
QIO1
A21 A17 A13 A9 A5 A1 O5 O1 D5 D1 D5 D1
QIO2
A22 A18 A14 A10 A6 A2 O6 O2 D6 D2 D6 D2
QIO3
A23 A19 A15 A11 A7 A3 O7 O3 D7 D3 D7 D3
Instruction EBh
Address
Option
Dummy cycles
Data

In Serial Memory mode, QSPI is compatible with three protocols:

- Single-Bit SPI: communicate with external memory with IO0
- Dual SPI: communicate with external memory with IO0 and IO1
- Quad SPI: communicate with external memory with IO0, IO1, IO2 and IO3

For SAMA5D2, when communicating with external memory, the instruction, address and data can be set to use different modes which is configured in the Instruction Frame register (QSPI_IFR). There are in total seven combinations, as follows:

Table 2-1. WIDTH: Width of Instruction Code, Address, Option Code and Data

Value	Name	Description
0	SINGLE_BIT_SPI	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Single-bit SPI
1	DUAL_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Dual SPI
2	QUAD_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Quad SPI
3	DUAL_IO	Instruction: Single-bit SPI / Address-Option: Dual SPI / Data: Dual SPI
4	QUAD_IO	Instruction: Single-bit SPI / Address-Option: Quad SPI / Data: Quad SPI
5	DUAL_CMD	Instruction: Dual SPI / Address-Option: Dual SPI / Data: Dual SPI
6	QUAD_CMD	Instruction: Quad SPI / Address-Option: Quad SPI / Data: Quad SPI

For detailed information on how to transfer instruction, address and data with external memory, refer to the SAMA5D2 Datasheet.

2.2.2. SAMA5D2 eXecute In Place (TFRTYP = 1)

By sending an instruction with DATAEN=1 and TFRTYP=1 in QSPI_IFR, the user can access the serial memory data at a random address, allowing the CPU to execute code directly from the serial memory (XIP) without code shadowing to RAM. The serial Flash memory mapping is seen in the system as other memories such as ROM, SRAM, DRAM, embedded Flash memory, etc.

2.2.3. Continuous Read Mode

The QSPI is compatible with the Continuous Read mode which is implemented in some serial Flash memories.

In Continuous Read mode, the instruction overhead is reduced by excluding the instruction code from the instruction frame. When the mode is activated, the instruction code is stored in the memory. For the next instruction frames, the instruction code is not required as the memory used the stored one. The addresses of the system bus read are often non-sequential and this leads to many instruction frames that have the same instruction code. By disabling the sending of the instruction code, the Continuous Read mode reduces the access time of the data. When the SAMA5D2 is set in XIP mode (TFRTYP = 1), working with continuous Read mode can greatly help to improve performance.

3. Running FreeRTOS in QSPI

Follow the instruction provided at the following link to download the FreeRTOS demo:

http://www.freertos.org/a00104.html

FreeRTOS V8.2.3 is used in this Application Note.

The demo can be found in directory: FreeRTOS\Demo\CORTEX_A5_SAMA5D3x_Xplained_IAR to SAMA5D2. The demo project is attached to this Application Note with the User Guide.

3.1. Cache Setting

Cache (L1 and L2) effect to the real-time performance is considered. Performance was measured under two conditions: without Cache&MMU and with Cache&MMU.

Refer to softpack: softpack\target\sama5d2\

3.2. SAMA5D2 QSPI Setting for XIP

To execute directly from the QSPI memory, the ICF files (IAR Linker Configuration File) must be customized as described in the softpack for the IAR and GCC tool chain: softpack\target\sama5d2\toolchain

The QSPI registers must be set as described in the softpack when the SAMA5D2 is set to XIP mode.

3.3. Frequency Setting

Normally, the QSPI clock setting is done by the ROM Code and there is no need for the customer to set the QSPI clock. However, for some application we also researched the effect of the QSPI clock on performance.

The customer can call function *qsip_set_baudrate(Qspi *qspi, uint32_t baudrate)* to set the QSPI clock. This function is defined in file qspi.c, available in the softpack (softpack\drivers\peripherals\).

4. Real-time Performance Result

4.1. Result Overview

This project uses three Rhealstone benchmark items (Task Switch, Task Pre-emption, and Interrupt Latency) on three types of memory:

- QSPI
- SRAM
- DDR3L (MT41K128M16JT)

with two types of setting: with Cache&MMU and without Cache&MMU.

The benchmark results are represented by clock cycles used for each single operation for the benchmark, e.g., clock cycles for task switch once, clock cycles for task pre-emption once, or clock cycles for one interrupt latency. Clock cycles are used to represent the time because it is unique for all system clock settings. The less the number of clock cycles, the better the performance.

Table 4-1. Benchmark Result Overview

	Clock Cycle						Compared Memory Types			
Memory Type	**DDR3L**		**QSPI**		**SRAM**		**QSPI/SRAM**		**QSPI/DDR3L**	
Benchmark Loops	**1**	**50000**	**1**	**50000**	**1**	**50000**	**1**	**50000**	**1**	**50000**
Cache&MMU										
interrupt-latency	1824	1080	7248	1040	1488	1029	4.87	1.01	3.97	0.96
task-preempt	2880	1539	12240	1540	2688	1538	4.55	1.00	4.25	1.00
task-switch	1008	560	3648	560	960	560	3.80	1.00	3.62	1.00
No Cache&MMU										
interrupt-latency	8736	9037	49200	55200	4944	5097	9.95	10.83	5.63	6.11
task-preempt	57504	44264	2E+05	2E+05	31104	23878	7.86	9.78	4.25	5.28
task-switch	17808	13912	71568	62365	8832	7395	8.10	8.43	4.02	4.48

Note:

1. Benchmark Loops:
   - Interrupt Latency: the time between the trigger of an interrupt and the execution of the first code in the interrupt routine.
   - Task Pre-emption: running 50000 times of benchmark, including 49999 times of task pre-emption.
   - Task Switch: running 50000 times of benchmark, including 50000*2-1=99999 times of task switch.

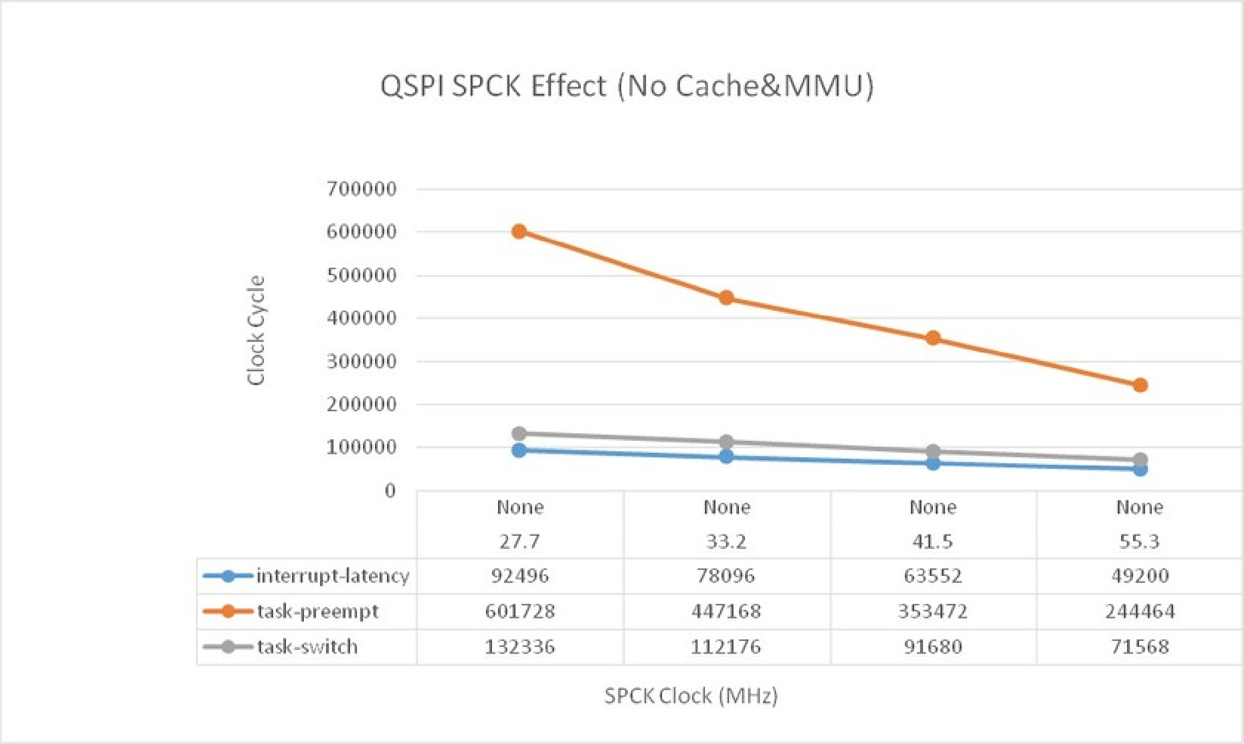
2. Cache&MMU: I/D Cache, L2 Cache and MMU are all enabled.
3. No Cache&MMU: I/D Cache, L2 Cache and MMU are all disabled.
4. Cache Miss: when Cache&MMU is enabled, running benchmark 1 loop.
5. Cache Hit: when Cache&MMU is enabled, running benchmark more than 1 loop.

4.1.1. QSPI Clock Effect on Performance

- When Cache&MMU is disabled, or when Cache miss

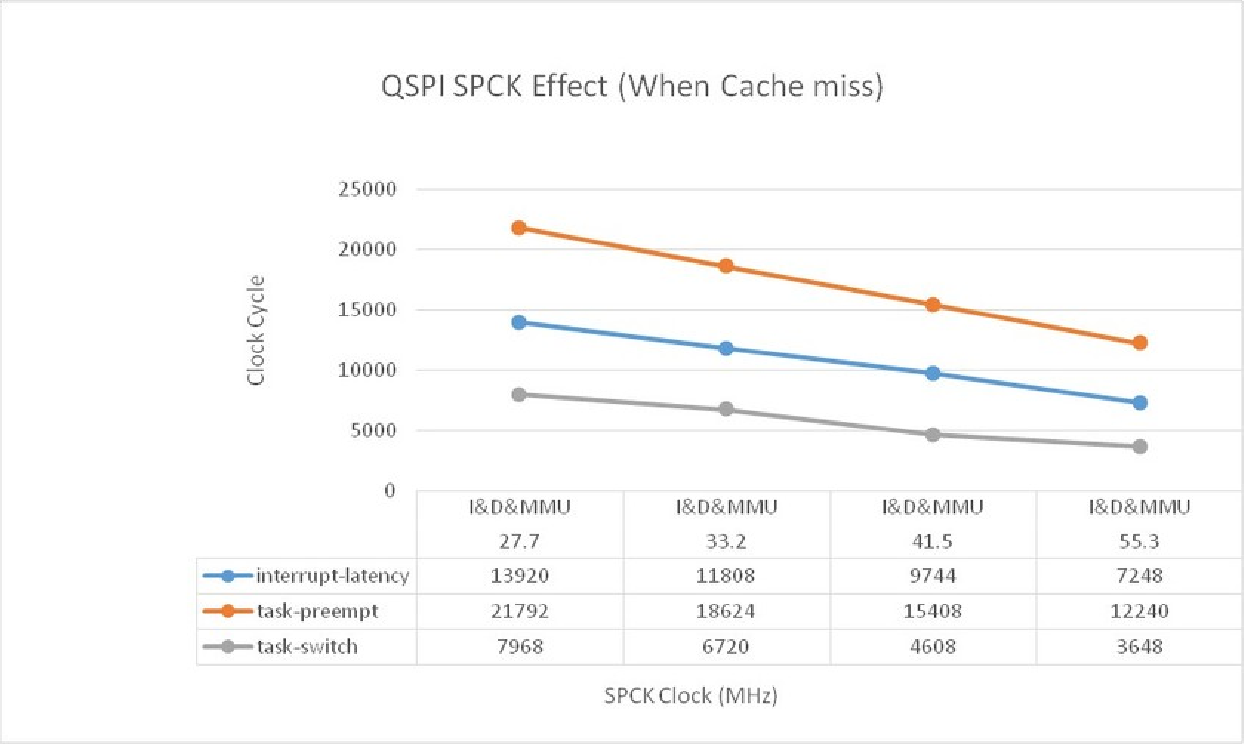
The faster the QSPI clock is, the better the QSPI performance. As shown in the following two diagrams, the QSPI performance is proportional to QSPI clock.

Figure 4-1. SPCK Effect on QSPI Performance when Cache&MMU Disabled



	None 27.7	None 33.2	None 41.5	None 55.3
interrupt-latency	92496	78096	63552	49200
task-preempt	601728	447168	353472	244464
task-switch	132336	112176	91680	71568

Figure 4-2. SPCK Effect on QSPI Performance when Cache Miss

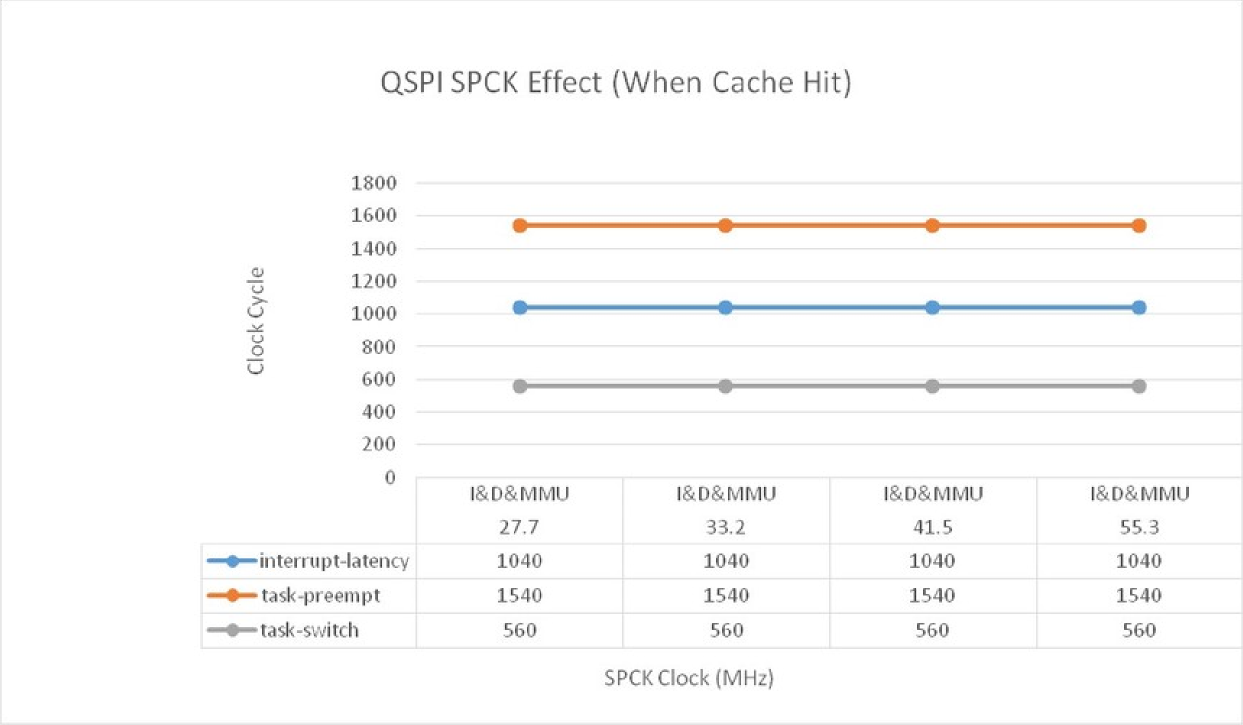


	I&D&MMU 27.7	I&D&MMU 33.2	I&D&MMU 41.5	I&D&MMU 55.3
interrupt-latency	13920	11808	9744	7248
task-preempt	21792	18624	15408	12240
task-switch	7968	6720	4608	3648

- When Cache hit

QSPI performance remains stable no matter what the SPCK is. The reason is that the benchmark code size is in the range of Cache which can be executed directly from Cache after it is loaded from QSPI to Cache the first time.

Figure 4-3. SPCK Effect on QSPI Performance when Cache Hit



	I&D&MMU 27.7	I&D&MMU 33.2	I&D&MMU 41.5	I&D&MMU 55.3
interrupt-latency	1040	1040	1040	1040
task-preempt	1540	1540	1540	1540
task-switch	560	560	560	560

4.1.2. Performance QSPI vs. SRAM vs. DDR3L

- When Cache&MMU is disabled, or when Cache miss

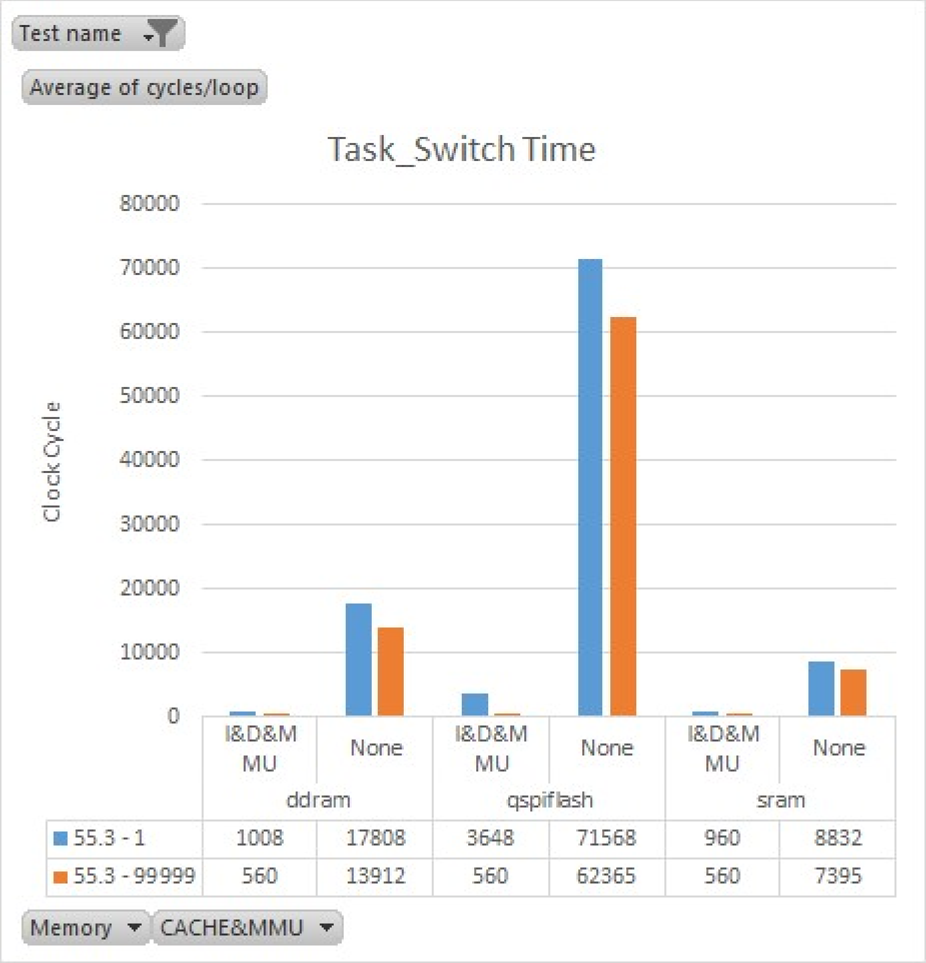
The time used for each benchmark item is stable for a fixed type of memory (QSPI, SRAM or DDR3L), which is only affected by the memory speed. When Cache miss, QSPI performance is greatly boosted, but its speed is still slower than SRAM and DDR3L. Details are shown in Table "Benchmark Result Overview".

- When Cache hit

Performance is greatly boosted for the three types of memory, especially QSPI. The benchmark result of QSPI is the same as that of SRAM and DDR3L because all the code is executed from Cache.

The following diagrams show the results of benchmark times on three different memories (QSPI, DDR3L and SRAM) under two conditions (without I/D Cache&MMU, with I/D Cache&MMU):

Figure 4-4. Task Switch Time Diagram

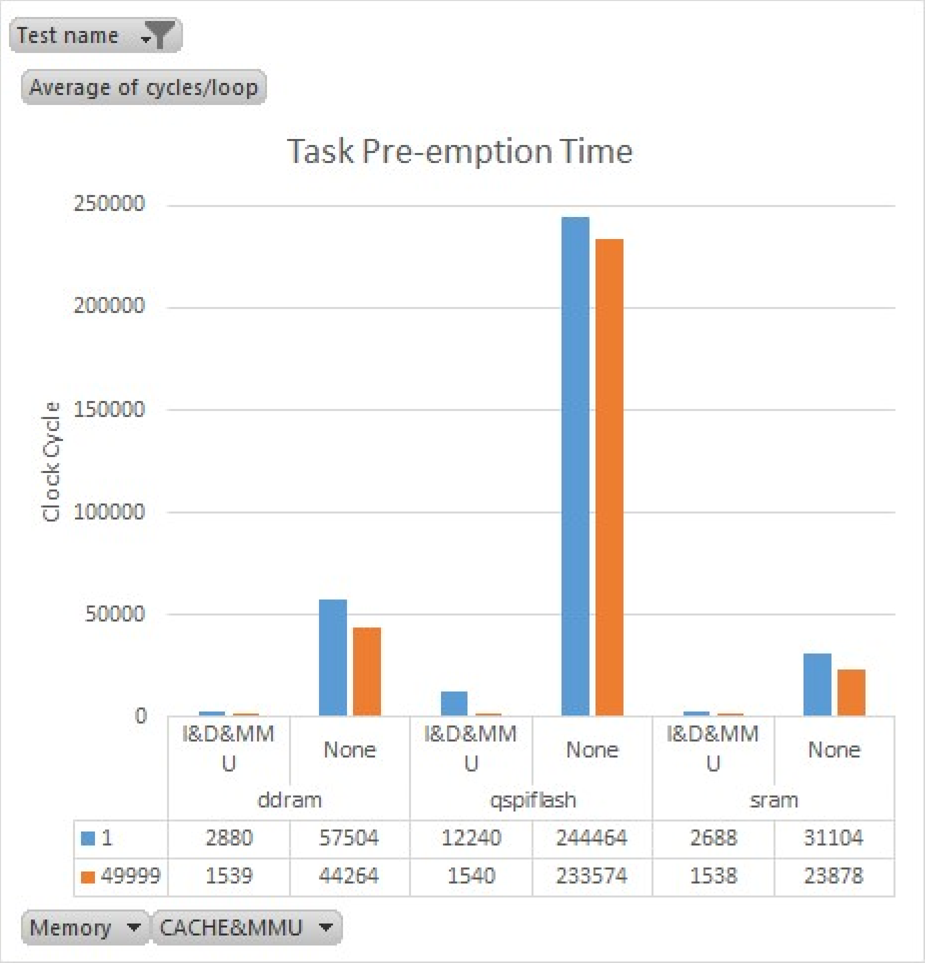


	ddram		qspiflash		sram	
	I&D&MMU	None	I&D&MMU	None	I&D&MMU	None
🟦 55.3 - 1	1008	17808	3648	71568	960	8832
🟧 55.3 - 99999	560	13912	560	62365	560	7395

Note:

1. Row “55.3-1” means the task switch is run once (QSPI SPCK@55.3 MHz).
2. Row “55.3-99999” means the benchmark item is run 50000 times, which includes 99999 times of task switch operation, and gets the average performance (QSPI SPCK@55.3 MHz).

Figure 4-5. Task Pre-emption Time Diagram

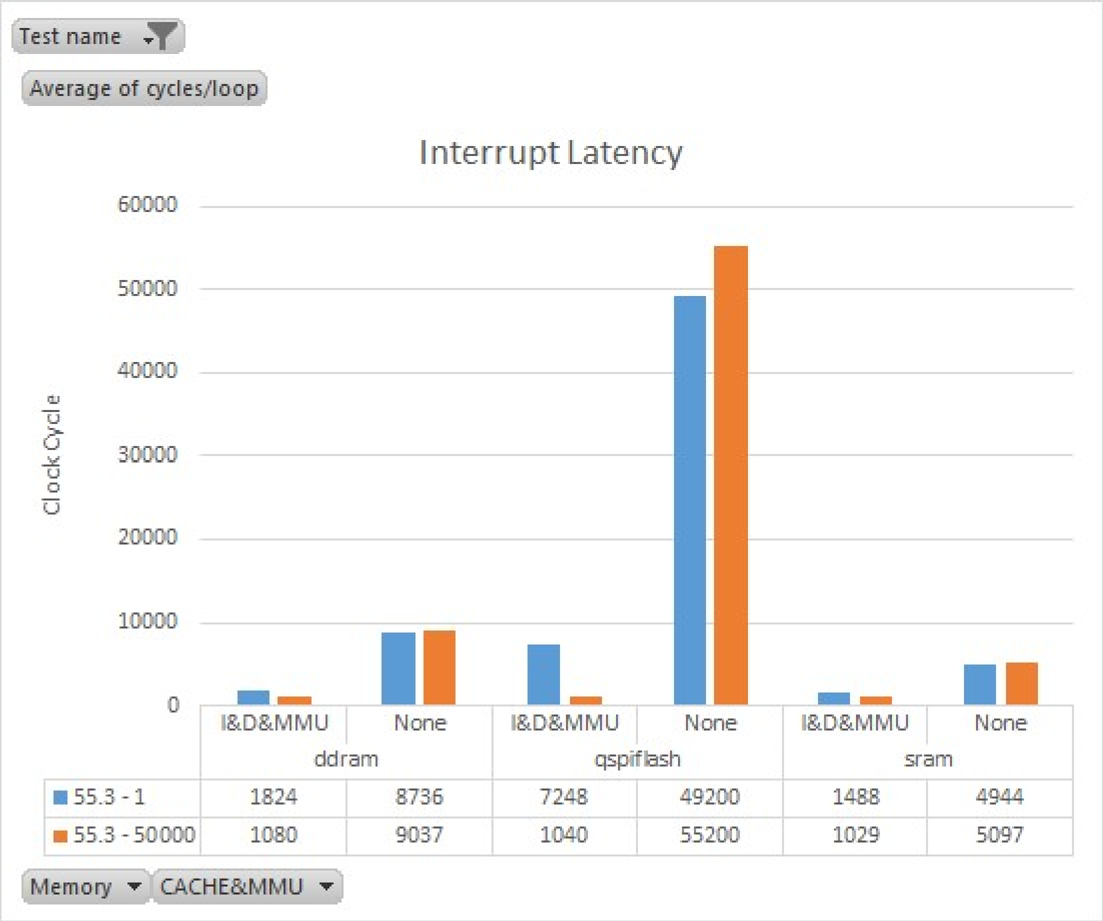


	ddram		qspiflash		sram	
	I&D&MMU	None	I&D&MMU	None	I&D&MMU	None
🟦 1	2880	57504	12240	244464	2688	31104
🟧 49999	1539	44264	1540	233574	1538	23878

Note:

1. Row “1” means the task switch is run once (QSPI SPCK@55.3 MHz).
2. Row “49999” means the benchmark item is run 50000 times, which includes 49999 times of task switch operation, and gets the average performance (QSPI SPCK@55.3 MHz).

Figure 4-6. Interrupt Latency Diagram



	ddram		qspiflash		sram	
	I&D&MMU	None	I&D&MMU	None	I&D&MMU	None
🟦 55.3 - 1	1824	8736	7248	49200	1488	4944
🟧 55.3 - 50000	1080	9037	1040	55200	1029	5097

Note:

1. Row “55.3-1” means the interrupt is generated once (QSPI SPCK@55.3 MHz).
2. Row “55.3-50000” means the interrupt is generated 50000 times and gets the average latency (QSPI SPCK@55.3 MHz).

5. Result Summary

5.1. Without Cache&MMU

- QSPI performance is proportional to its clock (SPCK).
- QSPI performance is about 10%~12% of SRAM.
- QSPI performance is about 20%~26% of DDR3.

5.2. With Cache&MMU

When Cache miss:

- QSPI performance can improve by about 7~20 times (compared with No Cache&MMU).
- QSPI performance still lower than SRAM and DDR3, which is about 20% to 26% of SRAM and 23%~27% of DDR3.

When Cache hit:

- QSPI performance boost about 50~150 times (compared with No Cache&MMU).
- QSPI performance equal to SRAM and DDR3.

So we can reach the conclusion that if the executed code size can fit in Cache, the performance of QSPI can compete with DDR or SRAM, which is a good choice to run a small size application with a DDR-less system.

6. Revision History

Table 6-1. Revision History

Doc. Rev.	Date	Changes
A	4-Aug-16	First release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.