AN3443

Execute-In-Place (XIP) with QSPI on Cortex-M7 MCUs Using MPLAB Harmony v3

Introduction

The parallel data communication interface was very effective in terms of performance; however, it increased the pin count and lead to complex designs. To overcome these challenges, the parallel interface peripherals played a crucial role, and the Quad Serial Peripheral Interface (QSPI) is one of these peripherals.

This document describes the Execute-In-Place (XIP) feature of the QSPI on a Arm®Cortex®-M7 based MCU (SAM E70), and discusses the implementation of an application using the MPLAB® Harmony v3 software framework. It explains how to generate an application binary to execute in the QSPI memory region, and it also shows how to execute the application from the QSPI region.

Table of Contents

1. Hardware and Software Requirements

1.1 SAM E70 Xplained Ultra Evaluation Kit

The SAM E70 Xplained Ultra Evaluation Kit is a development kit for evaluating the SAME70 microcontroller (MCU). The SAM E70 is based on the Cortex-M7, and is capable of running at 300 MHz. The evaluation kit includes an on-board embedded debugger, which eliminates the need for external tools to program or debug the SAME70. The evaluation kit also offers external connectors to extend the features of the board and ease the development of custom designs.

The SAM E70 Xplained Ultra Evaluation Kit is available at Microchip Direct.

1.2 MPLAB X Integrated Development Environment (IDE) and XC Compilers

MPLAB X IDE is an expandable, highly configurable software program that incorporates powerful tools to help users to discover, configure, develop, debug, and qualify embedded designs for most of the Microchip's microcontrollers.

MPLAB X IDE is available at the Microchip Website. This document uses MPLAB X IDE version 5.30.

MPLAB XC Compilers are available at the Microchip Website. This document uses MPLAB XC32 version 2.30.

1.3 MPLAB Harmony v3

MPLAB Harmony v3 is a fully-integrated embedded software development framework that provides flexible and interoperable software modules that enables the user to dedicate resources to create applications for 32-bit PIC® and SAM devices, rather than dealing with device details, complex protocols, and library integration challenges.

It includes MPLAB Harmony Configurator (MHC), an easy-to-use development tool with a Graphical User Interface (GUI) that simplifies device set up, library selection, configuration and application development. MHC is available as a plug-in that directly integrates with MPLAB X IDE and has a separate Java executable for stand-alone use with other development environments.

The examples used in this document use the following MPLAB Harmony v3 repositories, which can be downloaded from GitHub:

- CSP (Chip Support Package)
- DEV_PACKS (MPLAB Harmony v3 Product Database)
- MHC (MPLAB Harmony v3 Configurator)

OR

Use the MPLAB Harmony v3 Content Manager to download the repositories.

2. Introduction to QSPI

The Quad SPI Interface (QSPI) is a synchronous serial interface to communicate with external devices or memories. QSPI is similar to Serial Parallel Interface (SPI) protocol except it has four data lines. Because data is sent over multiple lines, it increases bandwidth and performance compared to the standard SPI protocol.

The QSPI supports single, quad, or dual I/O based on the mode selected.

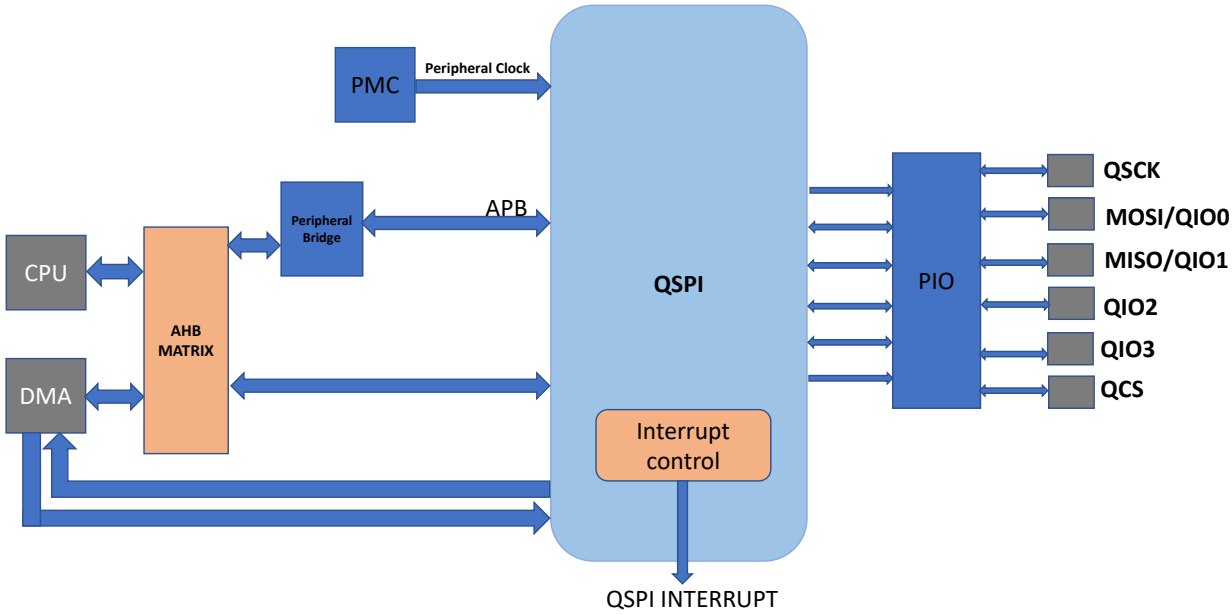
The QSPI can be used in the following modes:

- **SPI mode**: Acts as a regular SPI Master mode. Interfaces to serial peripherals, such as the ADC, DAC, LCD controllers, CAN controllers and sensors.
  Note: The scope of this document is limited to the QSPI Serial Memory mode. For a detailed description on the operation and configuration of the QSPI in SPI mode, refer to the specific device data sheet.
- **Serial Memory mode:** Interfaces to serial Flash memories.

The QSPI allows the system to use high-performance serial Flash memories which are small and inexpensive, in place of larger and more expensive parallel Flash memories.

The following figure illustrates the block diagram of the QSPI.

Figure 2-1. QSPI Block Diagram



The QSPI can be switched in between SPI mode or Serial Memory mode by using the SMM bit in the Mode register (QSPI_MR). The QSPI operates on the clock controlled by the internal programmable baud rate generator. The clock phase and polarity can be configured in the Serial Clock register (QSPI_SCR). The delays listed below are programmable through the QSPI_MR. These delays allow the QSPI to be synchronized to the interfaced peripherals based on their speed and timing.

- Transfer Delays between Consecutive Transfers
- Delay between Clock and Data
- Delay between Deactivation and Activation of Chip Select

2.1 QSPI Serial Memory Mode

In Serial Memory mode, the QSPI acts as a serial Flash memory controller. To activate this mode, the SMM bit must be set in the QSPI_MR. Once enabled, the peripheral appears as memory-mapped device at QSPI memory space

0x8000_0000. The data is read or written to the address 0x8000_0000 in Serial Memory mode. In this mode, the data cannot be transferred by the QSPI_TDR or QSPI_RDR. The QSPI can be used to read data from the serial Flash memory allowing the CPU to execute code from it (XIP). The QSPI can also be used to control the serial Flash memory (Program, Erase, Lock, and so on) by sending specific commands. In Serial Memory mode, the QSPI is compatible with the following modes:

- Single-Bit SPI
- Dual SPI
- Quad SPI

2.1.1 Single-Bit SPI

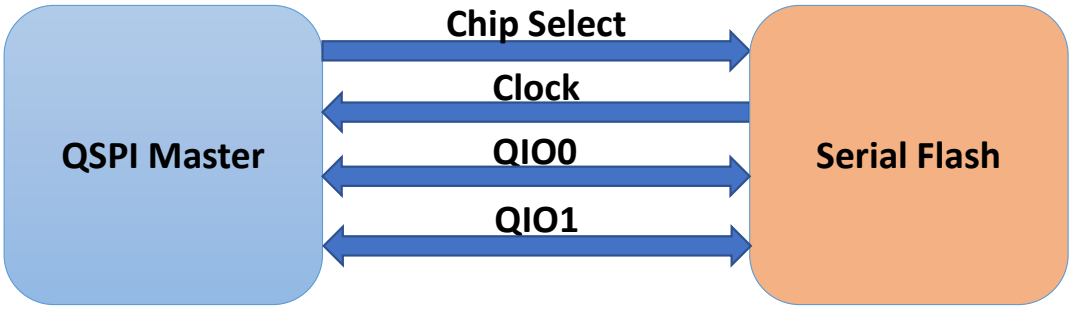
Single-Bit SPI uses two data lines: Master Out Slave In (MOSI) and Master In Slave Out (MISO) for communicating with the serial Flashes.

Figure 2-2. Single-Bit SPI

2.1.2 Dual SPI

Dual SPI uses two bidirectional QIO lines, QIO0, and QIO1, for communicating with the serial Flashes or external memories.

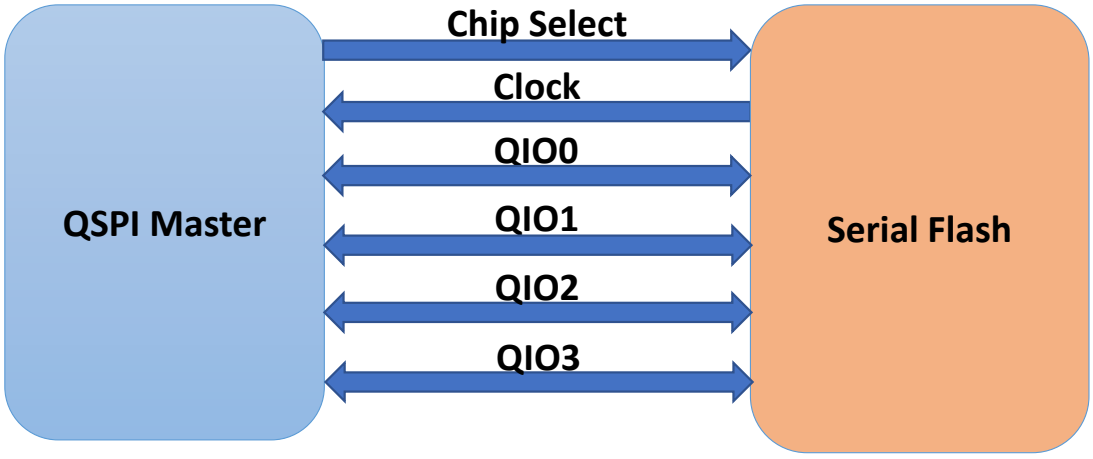
Figure 2-3. Dual SPI



2.1.3 Quad SPI

Quad SPI mode uses four bidirectional QIO lines for communicating with the serial Flashes or external memories.

Figure 2-4. Quad SPI



2.2 Instruction Frame

The QSPI sends the instructions, such as READ, WRITE, PROGRAM, ERASE, LOCK and so on to control serial Flash memories. All these instructions sets are implemented by serial Flash memory vendors. To support all serial Flashes, the QSPI includes a complete Instruction Frame register (QSPI_IFR), which makes it flexible and compatible with all serial Flash memories.

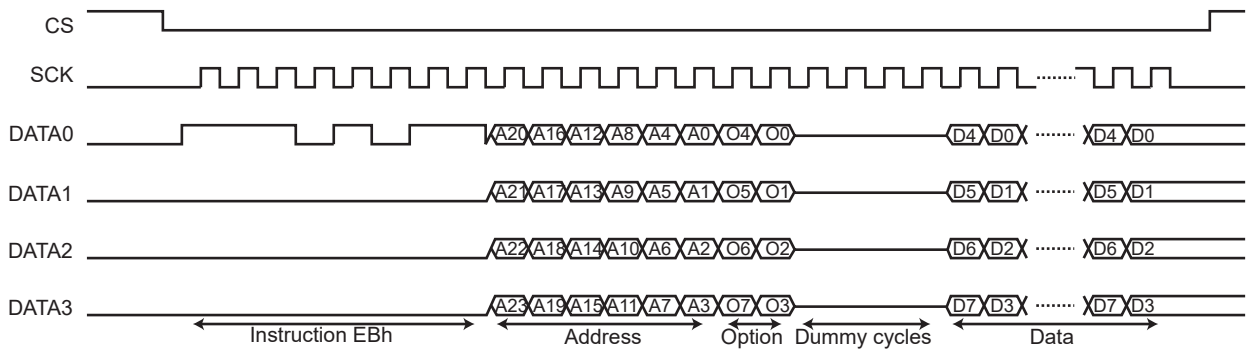
The following table represents the structure of instruction frame.

Table 2-1. Instruction Frame Structure

Instruction Frame field	Description
Instruction code (size: 8 bits)	The instructions as listed by the serial Flash memory. It is optional in some cases.
address (size: 24 bits or 32 bits)	The address is optional, but is required by instructions such as, READ, PROGRAM, ERASE, and 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).
option code (size: 1/2/4/8 bits)	This is useful to activate the XIP mode or the Continuous Read mode for READ instructions in some serial Flash memory devices. These modes improve the data read latency.
Dummy cycles	Dummy cycles are required by READ instructions.
Data Bytes	Data bytes are present for data transfer instructions, such as READ or PROGRAM.

The following figure displays a typical QSPI mode instruction frame.

Figure 2-5. QSPI Instruction Frame



2.2.1 Instruction Frame Configuration

The Instruction frame must be configured based on the commands to be sent to the external Flash memory. Refer to the data sheet of the respective external Flash memory for a list of supported commands. The registers to be configured are as follows:

- The Instruction Frame register (QSPI_IFR)
- The Instruction Address register (QSPI_IAR)
- The Instruction Code register (QSPI_ICR)

2.2.1.1 Instruction Frame Register

The QSPI_IFR must be written based on the command to be sent. If the instruction frame does not include any data, writing to this register triggers the instruction transmission over the QSPI. If the instruction frame includes data, the instruction frame will be transferred by the first data access in the QSPI memory space. The QSPI_IFR includes the following configurable fields:

Table 2-2. Instruction Frame Register

Field [bits]	Description
WIDTH [2:0]	To configure which data lanes (single-bit, Dual or QUAD) must be used to send the instruction code, the address, the option code, and to transfer the data.
INSTEN [4]	Enable to send instruction code.
ADDREN [5]	Enable to send address after instruction code.
OPTEN [6]	Enable to send option code after address.
DATAEN [7]	Enable to receive/send data during READ or Program instruction.
OPTL [9:8]	Length of the option code. The length must be consistent with the WIDTH configuration.
ADDRL [10]	Address length (24 bit or 32 bit).
TFRTYP [13:12]	Type of data transfer to be performed.
CRM [14]	Enable Continuous Read mode.
NBDUM [20:16]	Number of dummy cycles to be added when reading from serial Flash memory.

Transfer Types (TFRTYP)

- **TFRTYP = 0:** To read serial memory, such as JEDEC-ID or Serial Memory Status Register, but not to read data stored in memory.
- **TFRTYP = 1:** To read serial memory data. The address of the first instruction frame is the first read access over the QSPI memory space (0x80000000). For non-sequential read access, a new instruction frame is sent with the last system read access.
- **TFRTYP = 2:** To write serial memory, such as Configuration register or Status register, but not to write memory data over QSPI space
- **TFRTYP = 3:** To write to serial memory space

Note: For TFRTYP = 0/2/3: The address sent in the instruction frame is the address of the first system bus accesses. The addresses of the next accesses are not used by the QSPI.

2.2.1.2 Instruction Address Register

If the instruction frame includes only an address and no data, the address to send to must be written to the Instruction Address register (QSPI_IAR). For example, the BLOCK ERASE command would need only the address and does not need any data. When data is present, the address of the instruction is defined by the address of the data accesses in the QSPI memory space, not by QSPI_IAR.

2.2.1.3 Instruction Code Register

If the instruction frame includes the instruction code and/or the option code, the INST and OPT fields in the Instruction Code register (QSPI_ICR) must be configured.

2.2.1.4 End of Instruction Frame

When data transfer is not enabled, the end of the instruction frame is indicated when the INSTRE flag in QSPI_SR rises. When data transfer is enabled, the user must indicate when the data transfer is completed in the QSPI memory space by setting the LASTXFR bit in the QSPI_CR register. The end of the instruction frame is indicated when the INSTRE flag in the QSPI_SR rises.

3. Execute-In-Place

Execute-In-Place (XIP) is a method of executing code directly from the serial Flash memory without copying the code to the RAM. The serial Flash memory is seen as another memory in the MCU's memory address map.

XIP is achieved by configuring the QSPI in Quad SPI Serial Memory mode. Because Quad SPI mode uses four lines for data transfer, it allows the system to use high-performance serial Flash memories which are small and inexpensive, in place of larger and more expensive Flash memories. XIP on serial Flash is achieved by the ability of the QSPI to read data at random addresses allowing the CPU to execute code directly from it.

In the SAME70 device, the XIP is enabled by configuring the QSPI in Continuous Read Mode (The CRM bit in the QSPI_IFR register), setting DATAEN = 1 and TFRTYP = 1 in the QSPI_IFR register and sending the instruction to the serial Flash memory.

In XIP mode, the code instruction executed from the serial Flash is mapped to the QSPI memory space (0x80000000).

3.1 Continuous Read Mode

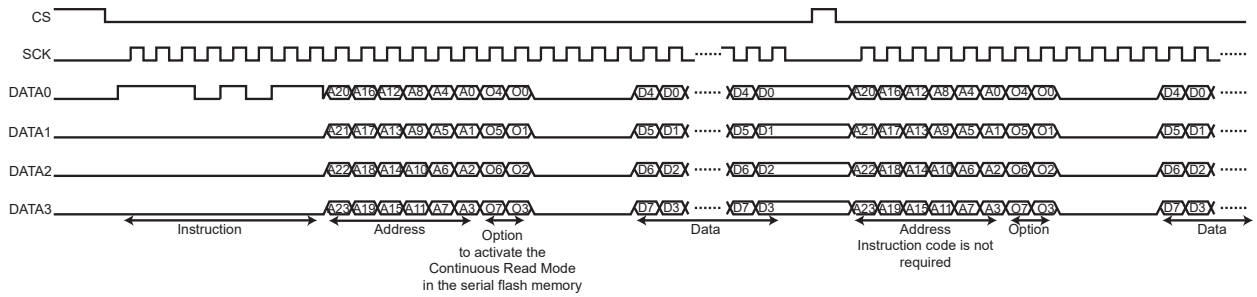
The QSPI supports Continuous Read mode which is implemented in some serial Flash memories. Continuous read mode is used when reading data from the memory (TFRTYP = 1). The addresses of the system bus read accesses are often non-sequential and this leads to many instruction frames that have the same instruction code. When the Continuous Read mode is activated in a serial Flash memory by a specific option code, the instruction code is stored in the memory. For the next instruction frames, the instruction code is not required as the memory uses the stored one. By disabling the send of the instruction code, the Continuous Read mode reduces the access time of the data and also instruction overhead.

Continuous read mode must be enabled in both the QSPI and the serial Flash memory. It is enabled in the QSPI by setting the bit CRM in the QSPI_IFR (TFRTYP field value must equal 1). It is enabled in the serial Flash memory by sending a specific option code.

Note: XIP is not possible if the Continuous Read mode is not supported by the serial Flash memory.

The following figure illustrates the QSPI Continuous Read mode.

Figure 3-1. QSPI Mode Continuous Read Mode



4. MPU Configuration for QSPI

The Cortex-M7 processor features a Memory Protection Unit, which allows the memory map to be divided into several regions with privilege permissions and access rules. It helps in providing fine grain memory control, enabling applications to utilize multiple privilege levels, separating and protecting code, data and stack, on a task-by-task basis.

The SAM E70 devices manage up to 16 regions with the MPU for safety or critical applications. The following table summarizes the available MPU attributes in the Cortex-M7.

Table 4-1. MPU Attributes

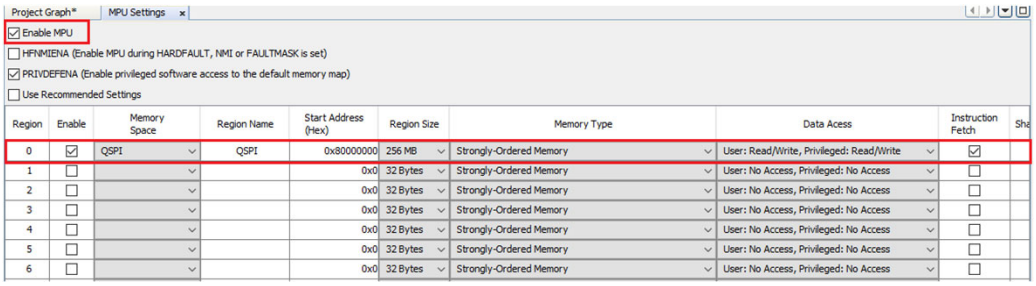
Memory type	Shareability	Attributes	Description
Strongly ordered	N/A	N/A	All access occurs in program order. No concurrent access can be done until the current access is completed.
Device	Shared	N/A	All access occurs in program order. The memory mapped peripheral is shared by several masters.
	Non-shared	N/A	All access occurs in program order. The memory mapped peripheral is shared by a single master.
Normal	Shared	Non-cacheable Write-through cacheable Write-back cacheable	Normal memory shared by several masters.
	Non-shared	Non-cacheable Write-through cacheable Write-back cacheable	Normal memory shared by single master.

When the QSPI is accessed by the Cortex-M7 processor for programming operations, the QSPI memory space must be defined in the Cortex-M7 memory protection unit (MPU).

For Programming operations, the QSPI memory space must be defined in the MPU with the attribute 'Device' or 'Strongly Ordered'. For Fetch or Read operations, the QSPI memory space must be defined in the MPU with the attribute 'Normal' in order to benefit from the internal cache.

The following figure shows the MPU configuration for the QSPI memory region using the “MPU Settings” window in the MPLAB Harmony v3 Configurator.

Figure 4-1. MPLAB Harmony v3 MPU Settings



5. Linker Script Customization

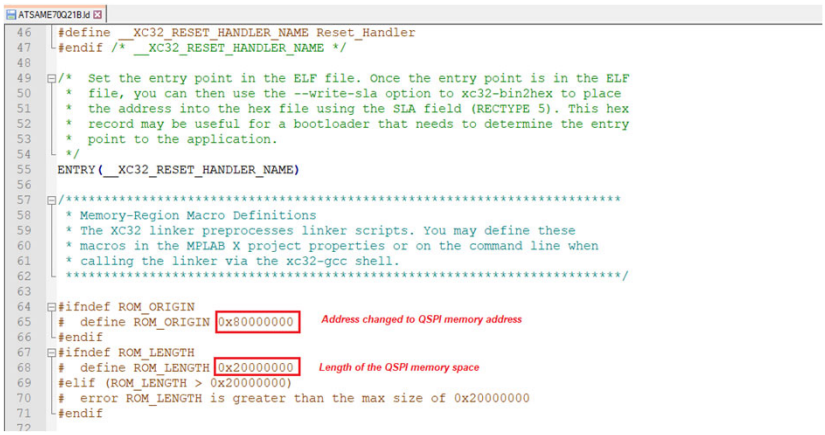
To execute from the QSPI memory space, the application needs to be linked to the QSPI address space. The linker file plays an important role in linking.

After the compiler generates the object files, they must be correctly linked per the memory map of the target device. All the object files use relative addressing, and the final address mapping is performed at link time. A linker combines input files (object file format) into a single output file (executable). The linker files are different for each compiler. The linkers make use of a linker script or command file to place different code and data sections into the appropriate memory.

The default linker file has a memory segment for Flash and a SRAM region. In addition, to be able to link the application to the QSPI memory space, a memory segment with the respective address and length for the QSPI region must be defined. Once a memory region is defined, the linker script can direct the linker to place the specific output sections into that memory region.

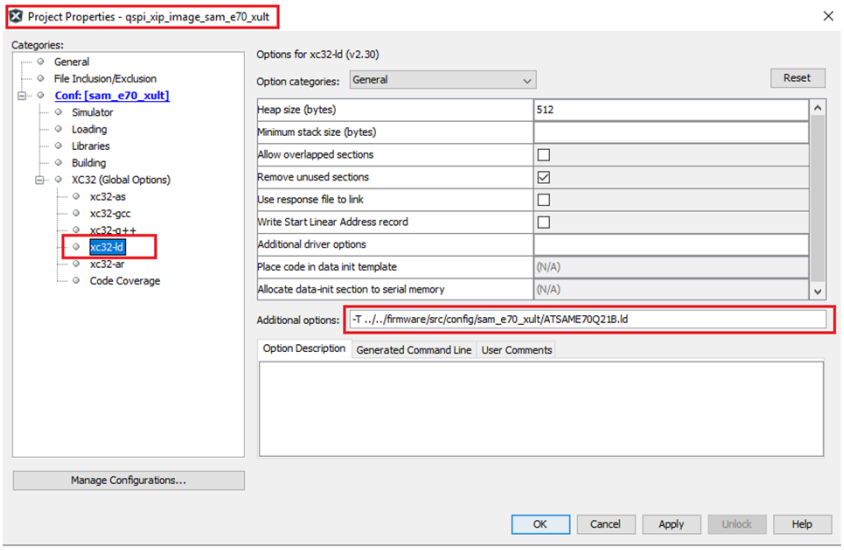
The following figure shows the custom linker file modified to link the application to the QSPI address space.

Figure 5-1. Custom Linker ID File



The following figure shows the MPLAB X IDE linker configuration options to enable the custom linker script.

Figure 5-2. Linker File Linking to the Project

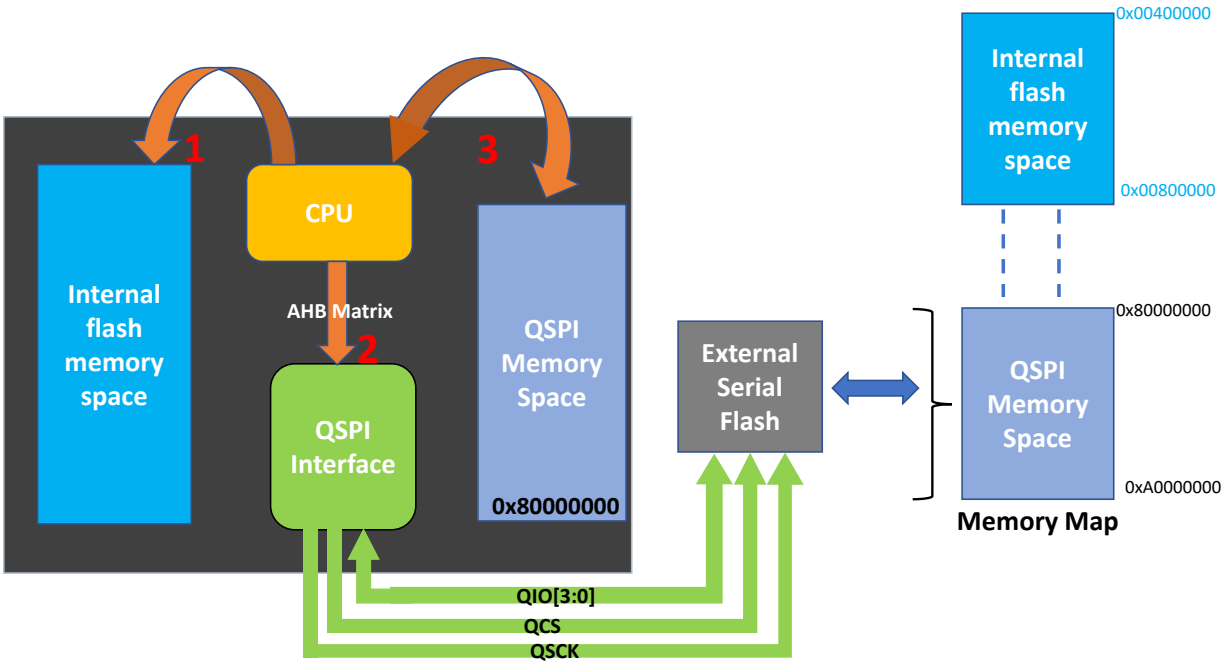


Note: To enable the custom linker script, In MPLAB X IDE, navigate to Project Properties and select the **xc32-ld** option.

6. XIP with QSPI Example Using MPLAB Harmony v3

To implement XIP with QSPI using MPLAB Harmony v3, refer to 1. Hardware and Software RequirementsThe . SAM E70 Xplained Ultra Board contains 4-MB QSPI Flash (SST26VF032BA) interfaced to QSPI lines. Refer to the SST26VF032BA data sheets to know commands and instructions to communicate to the serial Flash.

Figure 6-1. QSPI Application Block Diagram



1. The CPU starts executing from the internal MCU Flash and initializes QSPI in Serial Memory mode.
2. Programs the external serial Flash with binary generated to run from QSPI memory region through QSPI and configures QSPI peripheral to run in Continuous Read mode.
3. In Serial Memory mode, the serial Flash mapped to 0x80000000 appears as other another address mapped memory to CPU. The CPU starts executing from QSPI memory region.

XIP with QSPI example comprises of the following three applications:

- “QSPI XIP Main” MPLAB Harmony v3 application
- “QSPI Image” MPLAB Harmony v3 application
- “Hex Image generation” Python application

6.1 QSPI XIP Main MPLAB Harmony v3 Application

The QSPI XIP Main application acts as an image loader to program and execute from the serial Flash memory. This application configures the QSPI:

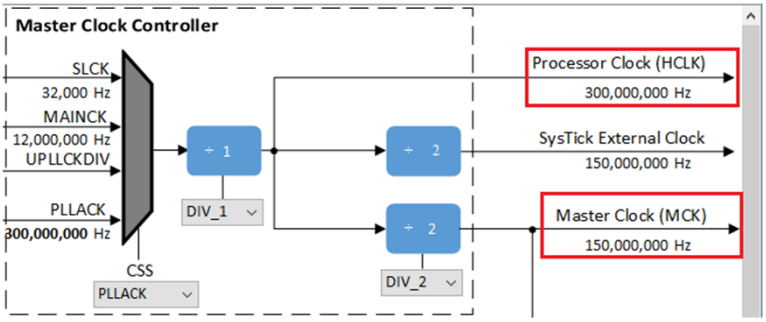
- To load the image generated (QSPI Image).
- Configures the QSPI to Continuous Read mode to execute the code from the serial Flash.
- This application includes the header file (`xip_image_pattern_hex.h`), which contains the firmware of “QSPI Image” represented in hex values, which are automatically written into the header file when the user runs the custom script.

The development sequence and flow of the QSPI XIP Main application is as follows:

1. Configure the MPU for the QSPI memory regions. The QSPI must be configured as 'Strongly ordered' for the programming operation as shown in MPLAB Harmony v3 MPU Settings.

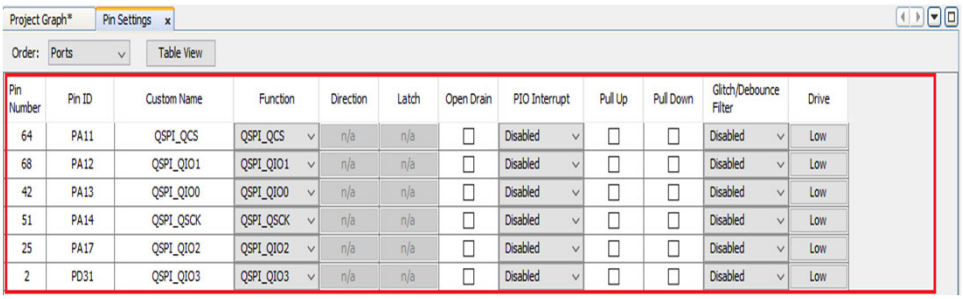
2. Verify Master and Processor clocks from the MHC Clock Configuration window.

Figure 6-2. QSPI XIP Main Clock Configurations



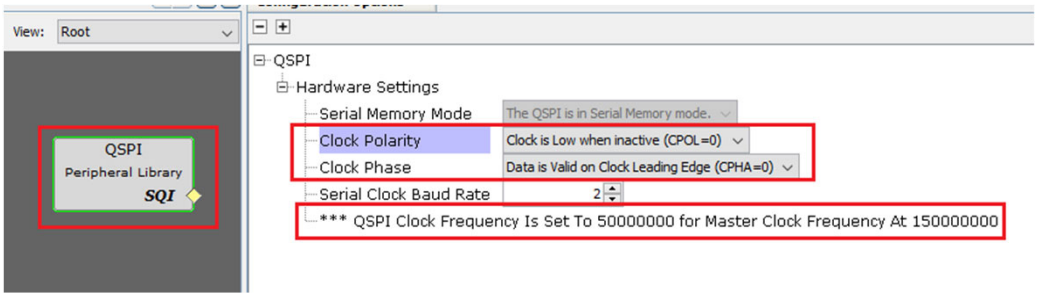
3. Configure the GPIO pins (QSCK, QCS, QIO [3:0]) for the QSPI peripheral from the MHC Pin Settings window.

Figure 6-3. QSPI XIP Main QSPI Pin Configuration



4. Configure the QSPI with clock and polarity settings.

Figure 6-4. QSPI XIP MAIN QSPI Configuration



5. Enable the Quad SPI mode in the application for better performance (For serial Flash SST26VF032BA, send command “0x38”).
6. Erase the serial memory by executing the appropriate ERASE command (For serial Flash SST26VF032BA, send command “0xD8”). The application provides `APP_BulkErase` or `APP_Erase` API functions to perform the erase.
7. Send the ‘Page program’ command (For serial Flash SST26VF032BA, the command is “0x02”) with the input buffer containing the hex values extracted from the QSPI Image binary file (explained in the following section).
8. Read the first 32 bytes from the QSPI memory to extract the Stack Pointer and reset handler address of the QSPI Image application in Quad SPI mode.
9. Enable 'Continuous Read Mode' to enter into the XIP by using the API `APP_MemoryReadContinuous()`.
10. After reading the data from the QSPI memory, verify the read content with the input buffer.

11. If verification passes, that is data written to and read from QSPI are matching, then configure the Stack Pointer and reset handler of the QSPI Image application programmed in the QSPI Flash memory extracted in step 8.

Figure 6-5. SP and PC Configuring

```
/* Set PC and SP of the XIP Image application programmed in QSPI Flash memory*/

/* Pointer to Reset Handler of the XIP Image Application */
__start_new = (uint32_t(*) (void)) xip_image_buffer[1];
__start_sp = xip_image_buffer[0];


__set_MSP(__start_sp);

/* Rebase the vector table base address */
SCB->VTOR = ((uint32_t) QSPIMEM_ADDR & SCB_VTOR_TBLOFF_Msk);

/* Run XIP Image Application */
__start_new();

/* Should never reach here */
break;
```

12. Following the execution of the previous steps, the control jumps to the QSPI Image and the application runs from the QSPI memory region.

Note: On Power-on Reset (POR), the QSPI XIP Main application executes according to the flow in the previous steps.

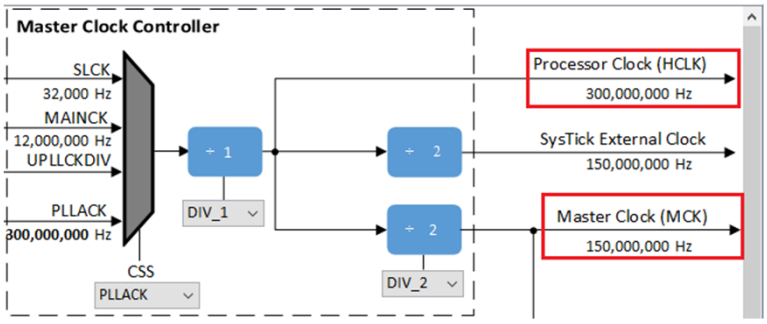
6.2 QSPI Image MPLAB Harmony v3 Application

The QSPI Image application is a simple application which blinks a LED every one second. To execute the application from the QSPI memory region, the QSPI Image application's binary file must be linked to the QSPI address space. A QSPI image application uses the custom linker script to blink the LED continuously.

The development sequence and flow of the QSPI Image application is as follows:

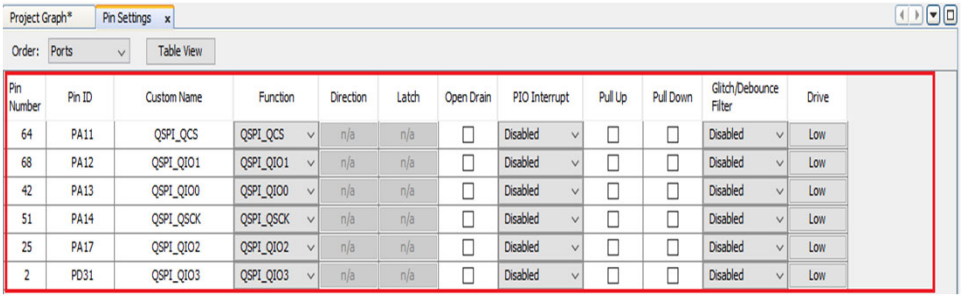
1. Verify Master and Processor clocks from the MHC Clock Configuration window.

Figure 6-6. QSPI Image Clock Configurations



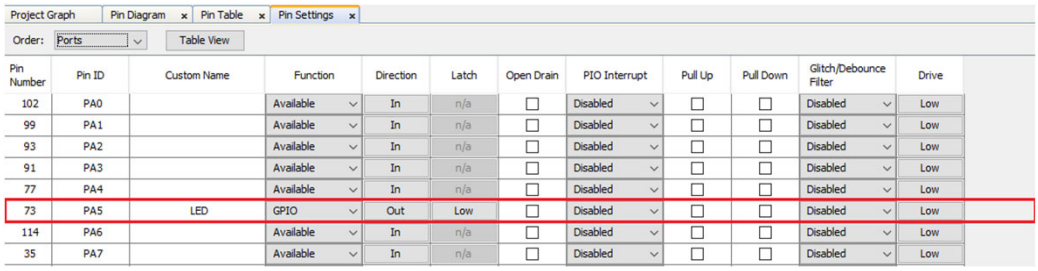
2. Configure the GPIO pins (QSCK, QCS, QIO [3:0]) for the QSPI peripheral from the MHC Pin Settings window.

Figure 6-7. QSPI Image QSPI Pin Configurations



3. Configure the LED pin as a GPIO.

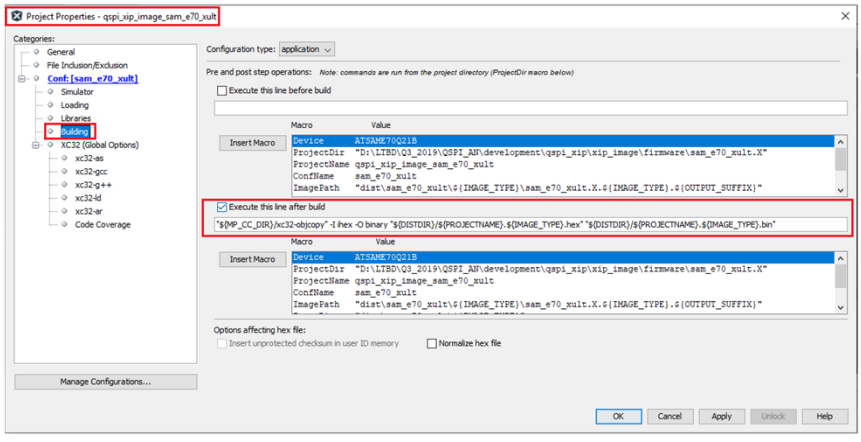
Figure 6-8. QSPI Image LED Pin Configuration



4. Link the custom linker script as specified in Linker Script Customization section above. This custom linker file is modified to run from the QSPI memory region.
5. Generate a binary file, which will be used by a python application to convert it into an array of hex values.

To generate the output in binary format, make the change in the QSPI Image application's Project Properties as shown in the following figure.

Figure 6-9. QSPI Image Binary Output Setting



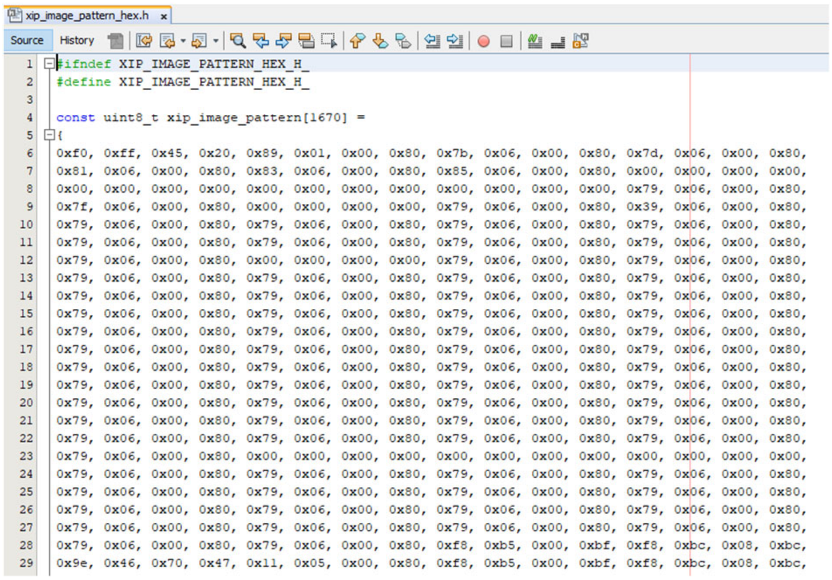
Note: The following code is the command to convert the hex file into a binary file:

```
${MP_CC_DIR}/xc32-objcopy" -I ihex -O binary "${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.hex ${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.bin
```

6.3 Hex Image Generation Python Application

The Hex Image generation python application is a custom python script used to convert the binary (QSPI Image) file to a `.hex` format and store it in the header file (`xip_image_pattern_hex.h`), as an array of hex values used in the QSPI XIP Main application.

Figure 6-10. XIP QSPI Image Header File



```
#ifndef XIP_IMAGE_PATTERN_HEX_H_
#define XIP_IMAGE_PATTERN_HEX_H_

const uint8_t xip_image_pattern[1670] =
{
0xf0, 0xff, 0x45, 0x20, 0x89, 0x01, 0x00, 0x80, 0x7b, 0x06, 0x00, 0x80, 0x7d, 0x06, 0x00, 0x80,
0x81, 0x06, 0x00, 0x80, 0x83, 0x06, 0x00, 0x80, 0x85, 0x06, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79, 0x06, 0x00, 0x80,
0x7f, 0x06, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x79, 0x06, 0x00, 0x80, 0x39, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80,
0x79, 0x06, 0x00, 0x80, 0x79, 0x06, 0x00, 0x80, 0xf8, 0xb5, 0x00, 0xbf, 0xf8, 0xbc, 0x08, 0xbc,
0x9e, 0x46, 0x70, 0x47, 0x11, 0x05, 0x00, 0x80, 0xf8, 0xb5, 0x00, 0xbf, 0xf8, 0xbc, 0x08, 0xbc,
```

Note: The QSPI XIP example is available in MPLAB Harmony v3 `csp` repository.

To run the QSPI XIP example follow these steps:

1. Open the QSPI image project (`<MPLAB Harmony v3 download path>/csp/apps/qspi/qspi_xip/xip_image/firmware/sam_e70_xult.X`) in the MPLAB X IDE.
2. Build the project using the MPLAB X IDE and do not program.
3. Run the python script (`<MPLAB Harmony v3 download path>/csp/apps/qspi/qspi_xip/xip_image_pattern_gen.py`) using the following command in the command prompt to extract the hex code from the binary file generated in the QSPI Image application and to store it to the header file (`<MPLAB Harmony v3 download path>/csp/apps/qspi/qspi_xip/xip_main/firmware/src/config/sam_e70_xult/xip_image_pattern_hex.h`) as an array of hex values.
   Command: **`python xip_image_pattern_gen.py`**
4. Open the QSPI XIP main project (`<MPLAB Harmony v3 download path>/csp/apps/qspi/qspi_xip/xip_main/firmware/ sam_e70_xult.X`) in the MPLAB X IDE.
5. Build and program the application using the MPLAB X IDE.
6. The main application programs the QSPI image and executes the code from the serial Flash memory. The QSPI image application starts blinking the LED continuously.

7. Performance

The QSPI allows the system to use high-performance serial Flash memories which are small and inexpensive, in place of larger and more expensive parallel Flash memories. Therefore, the performance of the QSPI plays an important role. The Performance of the QSPI is bounded by the QSPI speed, Flash capabilities and so on.

Note: Refer to the application note: “Execute-In-Place with QSPI using ASF” as described in the section 9. References for details on the performance numbers on the QSPI in XIP mode. The application referred to in the document “Execute-In-Place with QSPI using ASF” is not developed using MPLAB Harmony v3 and performance numbers in the document may vary with respect to compiler settings and optimization levels.

8. Conclusion

Developing a QSPI application in XIP requires an understanding of the QSPI protocols, MPU settings and linker scripts. MPLAB Harmony v3 provides a flexible, abstracted and fully integrated firmware development platform for 32-bit SAM, and PIC microcontrollers. This document describes how to use the XIP mode in the QSPI to work with the external Flash memories.

9. References

Refer to the document “Execute-In-Place (XIP) with Quad SPI Interface (QSPI) using ASF”, which is available for download at the following location:

http://ww1.microchip.com/downloads/en/AppNotes/Atmel-44065-Execute-in-Place-XIP-with-Quad-SPI-Interface-SAM-V7-SAM-E7-SAM-S7_Application-Note.pdf

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-5868-5

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820