

SPI-DirectC v2021.1 User Guide

Introduction

SPI-DirectC is designed to support an embedded In-System Programming for Microchip devices. In-System Programming refers to an external processor on-board programming one of the IGLOO[®]2, SmartFusion[®]2, PolarFire[®] or PolarFire SoC devices using a SPI peripheral interface.

SPI-DirectC supports systems with direct and indirect access to the memory space containing the data file image. With paging support, it is possible to implement the embedded ISP using SPI-DirectC on systems with no direct access to the entire memory space containing the data. Paging support is accomplished by making modifications to the data communication functions defined in dpuser.h, dpcom.c, and dpcom.h.

To use SPI-DirectC v2021.1, you must make some minor modifications to the source code, add the necessary API, and compile the source code and the API together to create a binary executable. The binary executable is downloaded to the system along with the programming data file. The programming data file is a binary file that can be generated by Libero[®] SoC Design Suite version 11.2 or later. For more information on detailed specification of the programming file, see 4. Data File Format.

Supported Device Family

This document describes how to enable microprocessor-based embedded In-System Programming (ISP) on the supported Microchip devices. The following table lists the Microchip devices SPI-DirectC supports.

Device Family	Description
PolarFire®	PolarFire FPGAs deliver the industry's lowest power at mid-range densities with exceptional security and reliability.
PolarFire SoC	PolarFire SoC is the first SoC FPGA with a deterministic, coherent RISC-V CPU cluster, and a deterministic L2 memory subsystem enabling Linux and real-time applications.
SmartFusion [®] 2	SmartFusion2 addresses fundamental requirements for advanced security, high reliability, and low power in critical industrial, military, aviation, communications, and medical applications.
IGLOO [®] 2	IGLOO2 is a low-power mixed-signal programmable solution.

Table 1. Device Family Supported by SPI-DirectC

Table of Contents

Intro	oductio	חמ	1		
1.	System Overview. 3 1.1. Systems with Direct Access to Memory. 3 1.2. Systems with Indirect Access to Memory. 4 1.3. Motorola SPI Protocol. 5				
2.	Gene 2.1.	rating Data Files and Integrating SPI-DirectC Code SPI-DirectC Code Integration	8		
3.	Requi 3.1. 3.2.	ired Source Code Modifications Compiler Switches Hardware Interface Components			
4.	Data 4.1.	File Format DAT File Description for M2GL, M2S, MPF and MPF SoC Devices	15 15		
5.	Sourc	e File Description	17		
6.	Data	File Bit Orientation			
7.	 7. Sample Project				
8.	Error	Messages and Troubleshooting Tips	21		
8. 9.	Error Smar 9.1. 9.2. 9.3.	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform			
8. 9. 10.	Error Smar 9.1. 9.2. 9.3. Revis	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History			
8. 9. 10. Mic	Error Smar 9.1. 9.2. 9.3. Revis rochip	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support			
8. 9. 10. Mic The	Error Smar 9.1. 9.2. 9.3. Revis rochip	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website			
8. 9. 10. Mic The Pro	Error Smar 9.1. 9.2. 9.3. Revis rochip Micro duct C	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service			
8. 9. 10. Mic The Pro Cus	Error Smar 9.1. 9.2. 9.3. Revis rochip Micro duct C	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service Support			
8. 9. 10. Mic Pro Cus Mic	Error Smar 9.1. 9.2. 9.3. Revis rochip duct C stomer rochip	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service Support Devices Code Protection Feature			
8. 9. 10. Mic The Pro Cus Mic Leg	Error Smar 9.1. 9.2. 9.3. Revis rochip duct C stomer rochip rochip	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service Support Devices Code Protection Feature ce			
8. 9. 10. Mic The Pro Cus Mic Leg Tra	Error 9.1. 9.2. 9.3. Revis rochip duct C stomer rochip al Noti	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service Support Devices Code Protection Feature ce			
8. 9. 10. Mic The Pro Cus Mic Leg Tra	Error 9.1. 9.2. 9.3. Revis rochip duct C stomer rochip al Noti demari	Messages and Troubleshooting Tips tFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis Read ID Code Waveform Read FSN waveform Program Frame Waveform ion History FPGA Support chip Website hange Notification Service Support Devices Code Protection Feature ce ks anagement System			

1. System Overview

The system must contain the following parameters to perform the In-System Programming (ISP) for the FPGA.

- A microprocessor with at least 8192 bytes of RAM or a softcore processor implemented in another FPGA.
- SPI IP to interface with the target device. SPI Mode 3 must be used.
- Access to the data file containing the programming data.
- Memory to store and run SPI-DirectC code.

For more information on power requirements for V_{pump} and other power supplies, see your device product device datasheet.

The following table lists the memory requirements.

Table 1-1. Code Memory Requirements- SPI-DirectC Code Size on CM3 16-Bit Mode

Compile Options Enabled	Units are in Bytes				
	ROM Code ¹	ROM Data ²	Read/Write Data ³		
ENABLE_G4_SUPPORT	16902	608	12578		
ENABLE_G5_SUPPORT	20242	1570	12851		
All the above	30414	1576	13639		

Notes:

- 1. ROM Code— This is the compiled code size memory requirements.
- 2. ROM Data— This is the block started by Symbol allocation for variables that do not yet have values, that is, uninitialized data. It is part of the overall data size.
- 3. Read/Write Data— This is the run time memory requirement, that is, the free data memory space required to execute the code.

1.1 Systems with Direct Access to Memory

The following figure shows the overview of a typical system with direct access to the memory space holding the data file.

Figure 1-1. System with Direct Access to Memory



The following table lists the data storage memory requirements.

	Table 1-2. Data	Storage Memor	v Requirements	- Data	Image Size
--	-----------------	---------------	----------------	--------	------------

Data Image Size					
Device	Core/FPGA Array - Encrypt (kB)	Embedded Flash Memory Block - Encrypt (kB)	Core/FPGA Array and Security - Encrypt (kB)		
M2GL005	297	133	851		
M2GL010	557	267	1639		
M2GL025	1197	267	2918		
M2GL050	2364	267	5253		
M2GL090	3564	532	8178		
M2GL150	5997	531	13046		
M2S005	297	137	860		
M2S010	557	272	1648		
M2S025	1197	272	2926		
M2S050	2364	272	5261		
M2S090	3564	536	8186		
M2S150	5997	535	13054		
MPF100	3447	N/A	N/A		
MPF200	5992	N/A	N/A		
MPF300	9256	N/A	N/A		
MPF500	14739	N/A	N/A		
MPFS250T	9542	N/A	N/A		

Note: The total image size is the sum of all the corresponding enabled blocks for the specific target device.

1.2 Systems with Indirect Access to Memory

The following figure is an overview of a system with no direct access to the memory space holding the data file. For example, the programming data might be received via a communication interface peripheral that exists between the processor memory and the remote system holding the data file dpcom.h and dpcom.c must be modified to interface with the communication peripheral.

Figure 1-2. System With Indirect Access to Memory



1.3 Motorola SPI Protocol

Motorola SPI Mode 3 is required to communicate with SmartFusion2, IGLOO2, PolarFire, and PolarFire SoC devices using a dedicated system controller SPI port. See Motorola SPI standard for more information.

The Motorola SPI is a full duplex, four-wire synchronous transfer protocol, which supports programmable clock polarity (SPO) and clock phase (SPH). The state of SPO and SPH control bits decides the data transfer modes as listed in the following table.

Table	1-3.	Data	Transfe	r Modes	

Data Transfer Mode	SPO	SPH
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0
Mode 3	1	1

The SPH control bit determines the clock edge that captures the data.

- When SPH is low, data is captured on the first clock transition.
 - Data is captured on the falling edge of SPI_CLK when SPO = 1.
 - Data is captured on the rising edge of SPI CLK when SPO = 0.
- When SPH is high, data is captured on the second clock transition (rising edge if SPO = 1).
 - Data is captured on the rising edge of SPI_CLK when SPO = 1.
 - Data is captured on the falling edge of SPI_CLK when SPO = 0.

The SPO control bit determines the polarity of the clock and SPS defines the slave select behavior.

- When SPO is low and no data is transferred, SPI CLK is driven to low.
- When SPO is high and no data is transferred, SPI_CLK is driven to high.

Mode	SPS	SPO	SPH	Clock in Idle	Sample Edge	Shift Edge	Select in Idle	Select Between Frames
Motorola	0	0	0	Low	Rising	Falling	High	Pulses between
	0	1	0	High	Falling	Rising	High	all frames
	0	0	1	Low	Falling	Rising	High	Does not pulse between back- to-back frames. Pulses if transmit FIFO empties.
	0	1	1	High	Rising	Falling	High	Does not pulse between back- to-back frames. Pulses if transmit FIFO empties.
	1	0	0	Low	Rising	Falling	High	Stays active
	1	0	1	Low	Falling	Rising	High	until all the frames set by
	1	1	0	High	Falling	Rising	High	frame counter
	1	1	1	High	Rising	Falling	High	

Table 1-4. Summary of the Clock Active Edges in Various SPI Master Modes

Single Frame Transfer - Mode 0: SPO = 0, SPH = 0

The following figure illustrates the single frame transfer using Mode 0 data transfer mode with programmable clock polarity 0 and clock phase 0.

Figure 1-3. Motorola SPI Mode 0



Multiple Frame Transfer - Mode 0: SPO = 0, SPH = 0

The following figure illustrates the multiple frame transfer using the Mode 0 data transfer mode with programmable clock polarity 0 and clock phase 0.

Figure 1-4. Motorola SPI Mode 0 Multiple Frame Transfer



Notes:

- Between frames, the slave selects (SPI_SS[x]) a signal that is asserted for the duration of the clock pulse.
- Between frames, the clock (SPI_CLK) is low.
- Data is transferred to Most Significant Bit (MSB) first.
- The output enables (SPI_DOE_N) a signal that is asserted during the transmission and deasserted at the end of the transfer (after the last frame is sent).

Single Frame Transfer - Mode 1: SPO = 0, SPH = 1

The following figure illustrates the single frame transfer using the Mode 1 data transfer mode with programmable clock polarity 0 and clock phase 1.

Figure 1-5. Motorola SPI Mode 1



Single Frame Transfer - Mode 2: SPO = 1, SPH = 0

The following figure illustrates the single frame transfer using the Mode 2 data transfer mode with programmable clock polarity 1 and clock phase 0.

Figure 1-6. Motorola SPI Mode 2



Single Frame Transfer - Mode 3: SPO = 1, SPH = 1

The following figure illustrates the single frame transfer using the Mode 3 data transfer mode with programmable clock polarity 1 and clock phase 1.

Figure 1-7. Motorola SPI Mode 3



2. Generating Data Files and Integrating SPI-DirectC Code

The following chapter describes the flows for data file generation and SPI-DirectC code integration. To generate the DAT file:

- 1. Launch the Libero SoC Design Suite and open the project.
- 2. Expand the Handoff Design for Production tree on the Design Flow tab.
- 3. Double click **Export Bitstream**. The **Export Bitstream** dialog box opens. The dialog box options depend on the device family, **Custom Security settings**, and **Permanent Locks** for the production settings. For more information on working with the **Export Bitstream**, see the Libero SoC Design Flow User Guide.
- 4. Program the DAT file into the storage memory.

2.1 SPI-DirectC Code Integration

The following figure shows the SPI-DirectC integration use flow.

Figure 2-1. Importing SPI-DirectC Files



To use SPI-DirectC code integration:

1. Import the SPI-DirectC files into your development environment as shown in the following figure. Figure 2-2. SPI-DirectC Files to Import into your Development Environment



- 2. Modify the SPI-DirectC code.
 - Add the SPI driver (available with the processor used to run SPI-DirectC).
 - Modify the hardware interface functions (do_SPI_SCAN_in and do_SPI_SCAN_out) to use the hardware API functions designed to control the SPI port.
 - Modify memory access functions to access the data blocks within the image file programmed into the system memory. See 6. Data File Bit Orientation for more details.
 - Call dp_top with the action code desired.
- 3. Compile the source code. This creates a binary executable that is downloaded to the system for execution.

3. Required Source Code Modifications

You must modify the dpuser.h, dpDUTspi.c, dpcom.c, and dputil.c files when using the SPI-DirectC source code, which contains a short description of SPI-DirectC source code and their functions.

The following table lists the functions that must be modified.

Table 3-1. Modified Functions

Function	Source File	Purpose
do_SPI_SCAN_in	dpspi.c	Hardware interface function used to scan data in using the SPI driver
do_SPI_SCAN_out	dpspi.c	Hardware interface function used to scan data out using the SPI driver
dp_get_page_data	dpcom.c	Programming file interface function
dp_display_text	dpuser.c	Function to display text to an output device
dp_display_value	dpuser.c	Function to display value of a variable to an output device
dp_delay	dputil.c	Delay function

3.1 Compiler Switches

The following table lists the compiler switches.

Table 3-2. Compiler Switches

Function	Source File	Purpose
USE_PAGING	dpuser.h	Enables paging implementation for memory access.
ENABLE_G4M_SUPPORT	dpuser.h	Enables M2S/M2GL programming support.
ENABLE_G5M_SUPPORT	dpuser.h	Enables MPF/MPF SoC programming support.
PERFORM_CRC_CHECK	dpuser.h	Enables CRC check of the programming data prior to performing the desired action.
ENABLE_DISPLAY	dpuser.h	Enables display to hyper terminal or other output devices.

3.2 Hardware Interface Components

3.2.1 Hardware Interface Function (dpDUTspi.c)

do_SPI_SCAN_in and do_SPI_SCAN_out functions are used to interface with the SPI port to clock data into and out of the target device.

Note: These functions must use the SPI driver API available for the targeted device processor.

dp_SPI_SCAN_in Function

This function takes three arguments:

- Command: 8-bit variable holding the command value.
- Data_bits: The number of bits to clock into the device.
- input_buffer: Pointer to the buffer which holds valid data to be clocked into the device.

dp_SPI_SCAN_OUT Function

This function takes four arguments:

- Command bits: The number of bits to clock in for the command portion of the frame. This value must be 8 as all SPI commands are 8 bit long.
- · Command: 8-bit variable holding the command value
- · Data_bits: The number of bits to read from the device.
- Ouput_buffer: Pointer to the buffer to hold the data read from the targeted device.

3.2.2 Display Functions

Four functions, dp_display_array, dp_display_array_reverse, dp_display_text, and dp_display_value are available to display text as well as numeric values. Note: You must modify dp_display_text and dp_display_value functions for proper operation.

3.2.3 Memory Interface Functions

All accesses to the memory blocks within the data file are done through the dp_get_data function within the DirectC code. This is true for all system types.

This function returns an address pointer to the byte containing the first requested bit. The dp_get_data function takes two arguments:

- var_ID: An integer variable, which contains an identifier specifying which block within the data file needs to be accessed.
- bit_index: The bit index addressing the bit to address within the data block specified in Var_ID. Upon
 completion of the function, it is expected that return_bytes indicates the total number of valid bytes available
 for the client of the function.

See 3.2.4 Systems with Direct Access to the Memory Containing the Data File and 3.2.5 Systems with Indirect Access to the Data File for more details.

3.2.4 Systems with Direct Access to the Memory Containing the Data File

The memory space holding the data file is accessible by the microprocessor. It can be treated as an array of unsigned characters. In this case:

- 1. Disable the USE PAGING compiler switch.
- 2. Assign the physical address pointer to the first element of the data memory location (image_buffer defined in dpcom.c). Image_buffer is used as the base memory for accessing the information in the programming data in storage memory.

The dp_get_data function calculates the address offset to the requested data and adds it to image_buffer.
Return bytes is the requested data.

```
Example 3-1. dp_get_data Function Implementation
DPUCHAR* dp_get_data (DPUCHAR var_ID, DPULONG bit_index)
```

```
{
    DPULONG image_requested_address;
    if (var_ID == Header_ID)current_block_address = 0;
    else
    dp_get_data_block_address(var_ID);
    if ((current_block_address ===0) && (var_ID != Header_ID))
    {
        return_bytes = 0;
        return NULL;
    }
    /* Calculating the relative address of the data block needed within the image
*/
```

```
image_requested_address = current_block_address + bit_index / 8;
return_bytes=image_size - image_requested_address;
return image_buffer+image_requested_address;
}
```

3.2.5 Systems with Indirect Access to the Data File

These systems access programming data indirectly via a paging mechanism. Paging is a method of copying a certain range of data from the memory containing the data file and pasting it into a limited size memory buffer that DirectC can access.

To implement paging:

- 1. Enable the USE PAGING compiler option.
- 2. Define Page buffer size. The minimum buffer size is 16 bytes.
- 3. Modify the dp_get_page_data function.

This function copies the requested data from the external memory device into the page buffer. See 6. Data File Bit Orientation for additional information. For correct operation:

- 1. Fill the entire page buffer unless the end of the image is reached. See 4. Data File Format.
- 2. Update return_bytes to reflect the number of valid bytes in the page.

Every time access to a data block within the image data file is needed, SPI-DirectC programming functions call the dp_get_data function. The dp_get_data function calculates the relative address location of the requested data and checks, if it already exists in the current page data. The paging mechanism is triggered, if the requested data is not within the page buffer.

3.2.6 dp_get_page_data Function Implementation

dp_get_page_data is the only function that must interface with the communication peripheral of the image data file. As the requested data blocks may not be contiguous, it must have random access to the data blocks. Its purpose is to fill the page buffer with valid data.

In addition, this function must maintain start_page_address, end_page_address, and return_bytes. These global variables contain the range of data currently in the page as well as the number of valid bytes.

dp get page data takes one argument:

address_offset— Contains the relative address of the needed element within the data block of the image file.

```
Example 3-2. dp_get_page_data Function Implementation

void dp_get_page_data(DPULONG image_requested_address)
{
DPULONG image_address_index;
start_page_address=0;
image_address_index=image_requested_address;
return_bytes = PAGE_BUFFER_SIZE;
if (image_requested_address + return_bytes > image_size)
return_bytes = image_size - image_requested_address;
while (image_address_index < image_requested_address + return_bytes)
{
page_global_buffer[start_page_address]=image_buffer[image_address_index];
start_page_address = image_requested_address;
end_page_address = image_requested_address + return_bytes - 1;
return;
}</pre>
```

3.2.7 Main Entry Function

The main entry function is dp_top defined in dpalg.c. It must be called to initiate the programming operation. Prior to calling the function, a global variable Action_code must be assigned a value as defined in dpuser.h. Action codes are listed in the following codeblock.

#define DP_DEVICE_INFO_ACTION_CODE 1
#define DP_READ_IDCODE_ACTION_CODE 2
#define DP_ERASE_ACTION_CODE 3
#define DP_PROGRAM_ACTION_CODE 4
#define DP_VERIFY_ACTION_CODE 5
#define DP_ENC_DATA_AUTHENTICATION_ACTION_CODE 6
#define DP_VERIFY_DIGEST_ACTION_CODE 7
#define DP_READ_DEVICE_CERTIFICATE_ACTION_CODE 30u
#define DP_ZEROIZE_LIKE_NEW_ACTION_CODE 31u
#define DP_ZEROIZE_UNRECOVERABLE_ACTION_CODE 32u

Note:

Programming of individual blocks, such as array only, eNVM only, or security only is not possible with one data file because of how the data is constructed. If you wish to use such a feature, you must generate multiple data files.

3.2.8 Data Type Definitions

Microchip uses DPUCHAR, DPUSHORT, DPUINT, DPULONG, DPBOOL, DPCHAR, DPINT, and DPLONG in the SPI-DirectC source code. Change the corresponding variable definition, if different data type names are used.

```
*****
/******
/* DPCHAR -- 8-bit Windows (ANSI) character */
/* that is 8-bit signed integer */
/* DPINT -- 16-bit signed integer */
/* DPLONG -- 32-bit signed integer */
/* DPBOOL -- boolean variable (0 or 1) */
/* DPUCHAR -- 8-bit unsigned integer */
/* DPUSHORT -- 16-bit unsigned integer */
/* DPUINT -- 16-bit unsigned integer */
/* DPULONG -- 32-bit unsigned integer */
\/*******
                                            *****/
typedef unsigned char DPUCHAR;
typedef unsigned short DPUSHORT;
typedef unsigned int DPUINT
typedef unsigned long DPULONG;
typedef unsigned char DPBOOL;
typedef char DPCHAR;
typedef int DPINT;
typedef long DPLONG;
```

3.2.9 Supported Actions

The following table lists the supported actions and devices.

Table 3-3. Supported Actions and Devices

Action	Supported Devices	Description
DP_DEVICE_INFO_ACTION	All	Displays device security settings.
DP_READ_IDCODE_ACTION	All	Reads and displays the content of the IDCODE register.
DP_ERASE_ACTION	All	Erases all supported blocks in the data file.
DP_PROGRAM_ACTION	All	Performs erase, program and verify operations for all the supported blocks in the data file.
DP_VERIFY_ACTION	All	Performs verify operation for all the supported blocks in the data file.
DP_ENC_DATA_AUTHENTICATION_AC TION	All	Performs data authentication of the bitstream within the data file.

continued					
Action	Supported Devices	Description			
DP_VERIFY_DIGEST_ACTION_CODE	All	Checks the digest of a programmed target device.			
DP_READ_DEVICE_CERTIFICATE_ACTI ON_CODE	All	Reads and displays device certificate.			
DP_ZEROIZE_LIKE_NEW_ACTION_CO DE	PolarFire/PolarFire Soc	Performs zeroization action. Device is recoverable.			
DP_ZEROIZE_UNRECOVERABLE_ACTI ON_CODE	PolarFire/PolarFire Soc	Performs zeroization action. Device is not recoverable.			

4. Data File Format

The following chapter contains information related to data file format.

4.1 DAT File Description for M2GL, M2S, MPF and MPF SoC Devices

The data file contains the following sections:

Header Block

Contains information identifying the type of the binary file and data size blocks.

Table 4-1. DAT Image Description

Header Section of DAT File			
Information	# of Bytes		
Designer Version Number	24		
Header Size	1		
Image Size	4		
DAT File Version	1		
Tools Version Number	2		
Map Version Number	2		
Feature Flag	2		
Device Family	1		

Constant Data Block

Includes device ID, silicon signature, and other information needed for programming.

Table 4-2. DAT Image Description

Constant Data Block				
Information	# of Bytes			
Device ID	4			
Device ID Mask	4			
Silicon Signature	4			
Checksum	2			
Number of BSR Bits	2			
Number of Components	2			
Data Size	2			
Erase Data Size	2			
Verify Data Size	2			
ENVM Data Size	2			
ENVM Verify Data Size	2			
UEK1_EXISTS	1			

continued			
Constant Data Block			
Information	# of Bytes		
UEK2_EXISTS	1		
SEC_ERASE	1		
UEK3_EXISTS (M2S, M2GL only)	1		
Number of Records	1		

Data Lookup Table

Contains records identifying the starting relative location of all the different data blocks used in the SPI-DirectC code and data size of each block. The following table lists the format.

Table 4-3. DAT Image Description

Look-Up-Table			
Information	# of Bytes		
Data Identifier # 1	1		
Pointer to data 1 memory location in the data block section	4		
# Of bytes of data 1	4		
Data Identifier # 2	1		
Pointer to data 2 memory location in the data block section	4		
# Of bytes of data 2	4		
Data Identifier # x	1		
Pointer to data x memory location in the data block section	4		
# Of bytes of data x	4		

Data Block

Contains the raw data for all the different variables specified in the look-up-table.

Table 4-4. DAT Image Description

Data Block			
Information	# of Bytes		
Binary Data	Variable		
CRC of the entire image	2		

5. Source File Description

dpuser.h	Contains definitions of all action codes as well as possible error codes that are reported within SPI-DirectC code.
dpalg.c and dpalg.h	Contain the main entry function dp_top and device ID check function.
dpg4alg.c and dpg4alg.h	Contain the main entry function dp_top_g4 and all other functions common to M2S and MGL families.
dpg5alg.c and dpg5alg.h	Contain the main entry function dp_top_g5 and all other functions common to the MPF and MPF SoC family of devices.
dpdutspi.c and dpdutspi.h	Contain the SPI interface function declaration and definition to the target device. SPI Mode 3 must be used to program the target device. See SPI IP block used for proper initialization.
dpg4spi.c and dpg4spi.h	Contain the SPI interface function declaration and definition to the target device specific to M2S and M2GL device families.
dpg5spi.c and dpg5spi.h	Contain the SPI interface function declaration and definition to the target device specific to MPF and MPF SoC device families.
dputil.c and dputil.h	Contain utility functions needed in the SPI-DirectC code.

6. Data File Bit Orientation

This chapter specifies the data orientation of the binary data file generated by the Libero SoC Design Suite. The SPI-DirectC implementation must be in sync with the specified data orientation. The following table lists how the data is stored in the binary data file. For more information, see 4. Data File Format.

Table 6-1. Binary Data File Example

Byte 0	Byte 1	Byte 2	Byte 3	 	Byte N
Bit7.Bit0	Bit15.Bit8	Bit23.Bit16	Bit35.Bit24	 	Bit(8N+7).Bit(8 N)
Valid Data	Valid Data	Valid Data	Valid Data	 	o <-Valid Data

If the number of bits in a data block is not a multiple of eight, the rest of the Most Significant Bit (MSB) in the last byte are filled with zeros. The following example shows a given 70-bit data to be shifted into the target shift register from the Least Significant Bit (LSB) to the Most Significant Bit (MSB).

The following figure shows a binary representation of the same data.

Figure 6-1. Binary representation of data

20E60A9AB06FAC78A6	tdi
10000011100110 00001010100110101010000011011	tdi
Bit 69	Bit 0

This data is stored in the data block section. The following table lists how the data is stored in the data block.

Table 6-2. Data Block Section Example

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4		Byte 8
Bit7Bit0	Bit15.Bit8	Bit23.Bit16	Bit31.Bit24	Bit43.Bit32		Bit71Bit64
10100110	01111000	10101100	01101111	10110000	-	00100000
A6	78	AC	6F	B0	-	20

7. Sample Project

The sample project, IAR_SPI_SlaveDirectC.zip, available with this release of SPI-DirectC is based on IAR Embedded Workbench version 7.40. It is designed to work on M2GL_M2S-EVAL-KIT with SmartFusion2 M2S025-FGG484 device.

7.1 Project Requirements

You need the following hardware and software to run the sample project:

- Hardware:
 - a. SmartFusion2 Security Evaluation Kit with SmartFusion2 M2S090-FGG484 device
 - b. jLink from IAR
 - c. Target board with Microchip device to be programmed
- Software:
 - a. IAR Embedded Workbench version 7.4
 - b. UART Host Loader available with this release package

7.2 Procedure

- 1. Program the evaluation kit with SPI_DC_top.job file included under M2S Eval Kit Files directory. The M2S090 design connects SPI1 port to certain pins of J1 header. Although not needed for this project, it also maps out specific MSS IOs to other J1 header pins for JTAG access.
- 2. Connect the SPI pins as described in HeaderPinAssignment.xlsx available under M2S Eval Kit Files directory. Ignore JTAG portion of the header.
- 3. Connect the Mini USB (J18) to your PC. The mini-USB is connected to FTDI FT4232h device used as a USB to UART bridge.
- 4. Make sure the appropriate drivers are installed on your PC to communicate with this chip.
- 5. Run UARTHostLoader application available with this release package.
- 6. There should be four com ports available in the serial port setup window. Select the fourth one from the list and configure the Baud Rate as shown in the following figure. If more than four ports are available, disconnect the J18 header and refresh the com ports in the UARTHostLoader application to identify exiting ports. Reconnect the J18 header and refresh the USB ports. Select the fourth port from the newly generated port list.
- 7. Click Initialize Port to establish connection with the selected COM port.
- 8. Select the programming file and desired action.
- 9. Click Run. The UARTHostLoader application waits for data from the SmartFusion2 evaluation kit.
- 10. The programming file programmed into the evaluation kit has a SPI-DirectC sample project that supports SmartFusion2, IGLOO2, PolarFire, and PolarFire SoC devices. Resetting the board runs the embedded application and performs the action selected. To run another action or select a different programming file, select it from the UARTHostLoader. Click **Run**.
- 11. To make changes to the embedded project, run IAR workbench and modify the compile options as required. You can download the embedded application using jLink as follows:
 - a. Connect jLink to RVI/IAR header.
 - b. Set the JTAG select jumper low.
 - c. Click **download** and run from IAR.

	st Loader		- 0
COM Port	Operations		Microson
COM Ports	: COM34 ~		Power Matte
Baud Rate	921600 🗸		
	Initialize Port		
Port Status	Refresh Port Close Port		
JTAG/SPI	DirectC		
DAT File:	C:\dat\G5.dat		Browse
Action:	PROGRAM	~	
HandSha	Run Abort		ĺ
HandSha **** Sma **** Sma ****** ID read Checkin Expecte Calcula	Run Abot Abot Abot Abot Active Active Active Active Active Active Active Abot Abot Active Abot Abot Active Abot Abot Active Abot Abot Active Abot Active Abot Active	ave Programming De	***** mp *** *****
HandSha **** Sma ******* ID read Checkin Expecte Calcula	Run Abot Abot Active passed Active passed Ac	ave Programming De	****** mo *** ******

Figure 7-1. UART Host Loader

8. Error Messages and Troubleshooting Tips

The information in this chapter may help you solve or identify a problem when using SPI-DirectC code. See the following table for a description of exit codes and their solutions.

Exit Code	Error Message	Action/Solution
0	This code does not indicate an error.	This message indicates success
2	Data processing failed.	 Solution: Check the V_{pump} level. Try with a new device. Measure SPI pins and noise or reflection. Load the correct DAT file.
6	The IDCODE of the target device does not match the expected value in the DAT file image.	 Possible Causes: The data file loaded is compiled for a different device. Example: M2S010 DAT file loaded to program M2S050 device. Noise or reflections on one or more of the SPI pins causing incorrect read-back of the SDO bits. Solution: Choose the correct DAT file for the target device. Cut down the extra length of ground connection.
7	Device polling error.	 Solution: Check the V_{pump} level Try with a new device Measure SPI pins and noise or reflection. Load the correct DAT file.
8	FPGA failed during the Erase operation.	 Possible Causes: The device is secured, and the corresponding data file is not loaded. The device has been permanently secured and cannot be unlocked. Solution: Load the correct DAT file.
10	Failed to program device.	 Solution: Check V_{pump} level. Try with a new device. Measure the SPI pins and noise or reflection.

Table 8-1. Exit Code

continued				
Exit Code	Error Message	Action/Solution		
11	FPGA failed verify.	 Possible Cause: The device is secured, and the corresponding DAT file is not loaded. The device is programmed with an incorrect design. Solution: Load the correct DAT file. Check V_{pump} level. Measure the SPI pins and noise or reflection. 		
18	Failed to authenticate the encrypted data.	Make sure the AES key used to encrypt the data matches the AES key programmed in the device.		
25	Device initialization failure.	 Solution: Check V_{pump} level. Try with a new device. Measure the SPI pins and noise or reflection. 		
100	CRC data error. Data file is corrupted or pro- gramming on system board is not successful.	Solution:Regenerate data file.Reprogram data file into system memory.		
150	Request action is not found.	Check spelling.		
151	Action is not supported because required data block is missing from the data file.	Regenerate DAT file with the needed block/feature support.		

9. SmartFusion2 and IGLOO2 SPI-Slave Programming Waveform Analysis

9.1 Read ID Code Waveform

- 1. Checking hardware status.
 - Figure 9-1. Hardware Status Check



2. Checking hardware status.

Figure 9-2. Hardware Status Check



3. Clocking in read_id command.



Figure 9-3. Clock in read_id Command (0x21)

Clocking in 16 bytes of zero values - Byte 0
 Figure 9-4. Clock in 16 Bytes of Zero Values - Byte 0

MS0-X 2024A, MY52490979: Tue Jul 26 17:50:15 2016 20.125 200.0%/ Stop D, TTL KEYSIGHT Normal 125MSa/s Channels DC 10.0:1 DC DC Cursors MISO ΔХ: +1.318000us MOSI 1/ΔX: +758.73kHz (C) CS Cursors Menu Mode Source Cursors Units <None> X1 X2 Manual Clock in 16 bytes of zero values - Byte 1.

5.

IM2	U-X 2024A	, MY52490	979: Tue Jul	26 17:50:24	2016									
1		2	3	4			21.44≦	200).O\$/	Stop)	₹ D ₀	TÌ	l
		Ŷ					X1 = 21.6	46000us			ì	**	KEYSI TECHNOL	
-							X2 = 22.9	64000us				:: Д	cquisition	า ::
												1	Normal 25MSa/s	
													Channels	
												DC		10.0:1
												DC		10.0:1
												UL Inc		10.0:1
												::	Cursors	::
D3	MISO				-							ΔΧ:	0010010	
													+1.318	000us
D ₂	MOSI				1							1/ΔX:		
					-								+758.	73kHz
D,	SCK													
Dn	CS													
Cu	rsors Mei	าน												
~	⊃ Moo Man	de ual	् Sou <no< td=""><td>irce one></td><td>Curso X1 X</td><td>ors 2</td><td>Ur</td><td>nits M</td><td>X1: 2 X2: 2</td><td>1.64600 2.96400</td><td>)Ous)Ous</td><td></td><td></td><td></td></no<>	irce one>	Curso X1 X	ors 2	Ur	nits M	X1: 2 X2: 2	1.64600 2.96400)Ous)Ous			

Figure 9-5. Clock in 16 Bytes of Zero Values - Byte 1

Clocking in 16 bytes of zero values - Byte 2.
 Figure 9-6. Clock in 16 Bytes of Zero Values - Byte 2

MS0-X 2024A, MY52490979: Tue Jul 26 17:50:34 2016



7. Clocking in 16 Bytes of Zero Values - Byte 3.



Figure 9-7. Clock in 16 bytes of zero values - Byte 3

Clocking in 16 bytes of zero values - Byte 4.
 Figure 9-8. Clock in 16 Bytes of Zero Values - Byte 4

MS0-X 2024A, MY52490979: Tue Jul 26 17:50:54 2016



9. Clocking in 16 bytes of zero values - Byte 5.

M	MS0-X 2024A, MY52490979: Tue Jul 26 17:51:05 2016										
1		2	3	4		26.	705 20	0.0°/	Stop	t D	o TTL
		~				X1 = 2	6.902000us				KEYSIGHT
						X2 = 2	B.220000us				Acquisition 🛛 🗧
											Normal 125MSa/s
											Channels ::
										DC	10.0:1
										DC	10.0:1
										DC	10.0:1
										DC	10.0.1
D,	MISO									8 • • • •	Cursors #
										ΔХ:	11 910000
n	MOST									1/47	+1.318000us
^U 2										ΠΔΛ	+758 73kHz
					1						1700.70KH2
D1	SCK									i .	
D _o	CS										
Ci	ursors Men	iu									
	🗿 Mod	le	🕤 🛛 Sour	ce	🕤 Cursors		Units	X1: 26.	902000us		
i.	Manu	ıal	<nor< td=""><td>ne></td><td>X1 X2</td><td></td><td>+</td><td>X2: 28.</td><td>220000us</td><td></td><td></td></nor<>	ne>	X1 X2		+	X2: 28.	220000us		

Figure 9-9. Clock in 16 Bytes of Zero Values - Byte 5

Clocking in 16 bytes of zero values - Byte 6. Figure 9-10. Clock in 16 Bytes of Zero Values - Byte 6

MS0-X 2024A, MY52490979: Tue Jul 26 17:51:14 2016



11. Clocking in 16 bytes of zero values - Byte 7.



Figure 9-11. Clock in 16 Bytes of Zero Values - Byte 7

Clocking in 16 bytes of zero values - Byte 8. Figure 9-12. Clock in 16 Bytes of Zero Values - Byte 8

MS0-X 2024A, MY52490979: Tue Jul 26 17:51:43 2016



13. Clocking in 16 bytes of zero values - Byte 9.



Figure 9-13. Clock in 16 Bytes of Zero Values - Byte 9

Clocking in 16 bytes of zero values - Byte 10.
 Figure 9-14. Clock in 16 Bytes of Zero Values - Byte 10

MS0-X 2024A, MY52490979: Tue Jul 26 17:52:04 2016



15. Clocking in 16 bytes of zero values - Byte 11.



Figure 9-15. Clock in 16 Bytes of Zero Values - Byte 11

16. Clock in 16 bytes of zero values - Byte 12.

Figure 9-16. Clock in 16 bytes of Zero Values - Byte 12

MS0-X 2024A, MY52490979: Tue Jul 26 17:52:26 2016



17. Clock in 16 bytes of zero values - Byte 13.



Figure 9-17. Clock in 16 Bytes of Zero Values - Byte 13

18. Clock in 16 bytes of zero values - Byte 14.

Figure 9-18. Clock in 16 Bytes of Zero Values - Byte 14

MS0-X 2024A, MY52490979: Tue Jul 26 17:52:46 2016



19. Clock in 16 bytes of zero values - Byte 15.



Figure 9-19. Clock in 16 Bytes of Zero Values - Byte 15

20. Checking hardware status.

Figure 9-20. Hardware Status Check

MS0-X 2024A, MY52490979: Tue Jul 26 17:53:22 2016



21. Checking hardware status.

Figure 9-21. Hardware Status Check

MS0-X 2024A, MY52490979; Tue Jul 26 17:53:30 2016

1	2	3 4	2010	55.985	200.0%/	Stop	₹ D _n	TTL
				(Delay = 55.98	4000us		**	KEYSIGHT TECHNOLOGIES
							<u> </u>	cquisition ∷ Normal 25MSa/s
								Channels ::
							DC	10.0:1 10.0:1
							DC	10.0:1
n. MISO							:	Cursors #
-							ΔX:	+1.318000us
D ₂ MOSI							1/ΔX:	+758.73kHz
D, SCK								
D _o CS								
Cursors Menu	u)	Source	Cursors	Units	X1: 46.	164000us		
Cursors Menu Mode Manua	u e)(O	Source <none></none>	 Cursors X1 X2 	Units	X1: 46. X2: 47.	464000us 782000us		

22. Checking hardware status.

Figure 9-22. Hardware Status Check

MS0-X 2024A, MY52490979: Tue Jul 26 17:53:38 2016



23. Clocking in read command (0x5).



24. Reading out 16 Bytes of data – Byte 0 = 0xCF

Figure 9-23. Clock in Read Command (0x5)

Figure 9-24. Reading out 16 Bytes of Data – Byte 0 = 0xCF

MS0-X 2024A, MY52490979: Tue Jul 26 17:54:13 2016



25. Reading out 16 Bytes of data – Byte 1 = 0x21.



Figure 9-25. Reading out 16 Bytes of Data – Byte 1 = 0x21

26. Reading out 16 Bytes of data – Byte 2 = 0x80.

Figure 9-26. Reading out 16 Bytes of Data – Byte 2 = 0x80

MS0-X 2024A, MY52490979: Tue Jul 26 17:54:35 2016



27. Reading out 16 Bytes of data – Byte 3 = 0x3F.



Figure 9-27. Reading out 16 Bytes of Data – Byte 3 = 0x3F

28. Reading out 16 Bytes of data – Byte 4 = 0x0.

Figure 9-28. Reading out 16 Bytes of Data – Byte 4 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:54:57 2016



29. Reading out 16 Bytes of data - Byte 5 = 0x0.


Figure 9-29. Reading out 16 Bytes of Data – Byte 5 = 0x0

30. Reading out 16 Bytes of data – Byte 6 = 0x0.

Figure 9-30. Reading out 16 Bytes of Data – Byte 6 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:55:21 2016



31. Reading out 16 Bytes of data - Byte 7 = 0x0.



Figure 9-31. Reading out 16 Bytes of Data – Byte 7 = 0x0

32. Reading out 16 Bytes of data – Byte 8 = 0x0.

Figure 9-32. Reading out 16 Bytes of Data – Byte 8 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:55:41 2016



33. Reading out 16 Bytes of data - Byte 9 = 0x0.



Figure 9-33. Reading out 16 Bytes of Data – Byte 9 = 0x0

Reading out 16 Bytes of data – Byte 10 = 0x0.
 Figure 9-34. Reading out 16 Bytes of Data – Byte 10 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:56:01 2016



35. Reading out 16 Bytes of data – Byte 11 = 0x0.

IVI	M50-X 2024A, MY52490979; Tue Jul 26 17:56; TU 2016																	
1		2	3		4					92.7	3s 3s	200	.0°,		Stop	ť	Do	TTL
		Ý							Ď	(1 = 92	.898	000us						KEYSIGHT TECHNOLOGIES
									Þ	(2 = 94	.216	UUUus		1			4	oquisición :: Normal
																	1	25MSa/s
																		Channels ::
																DC	3	10.0:1
																DO		10.0:1
														+		DU		10.0:1
																UL	j.	10.0:1
D3	MISO															:: A`	<i>.</i>	Lursors #
																	^. 	+1.318000us
D_2	MOSI						-									1/	ΔΧ:	
																—		+758.73kHz
D	SCK										4							
Do	CS																	
Сц	irsors Mer	nu																
	Moc	le	0	Source	е	0	Curs	ors			Units		X1	92.8	98000u			
	Manu	lal	<	<none.< td=""><td>></td><td></td><td>- X I)</td><td>(2</td><td></td><td></td><td>+</td><td></td><td>_[X2</td><td>: 94.2</td><td>160000</td><td>S</td><td>1</td><td></td></none.<>	>		- X I)	(2			+		_[X2	: 94.2	160000	S	1	

Figure 9-35. Reading out 16 Bytes of Data – Byte 11 = 0x0

36. Reading out 16 Bytes of data – Byte 12 = 0x0.
Figure 9-36. Reading out 16 Bytes of Data – Byte 12 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:56:20 2016



37. Reading out 16 Bytes of data - Byte 13 = 0x0.

1 2 3 4 95.36₅ 200.0≩/ Stop t l	D ₀ TTL
X1 = 95.520000us	
X2 = 96 838000us	Acquisition 🛛 🗧
	Normal 125MSa/s
	Channels 🛛 🗄
	10.0:1
	10.0:1
	10.0:1
	Fursors ::
D ₃ MISO	
	+1.318000us
	X:
	+758.73kHz
D, ISCK	
Mode Source Cursors Units X1: 95.520000us	
Manual <none> X1 X2 - X2: 96.838000us</none>	

Figure 9-37. Reading out 16 Bytes of Data – Byte 13 = 0x0

38. Reading out 16 Bytes of data – Byte 14 = 0x0.
Figure 9-38. Reading out 16 Bytes of Data – Byte 14 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:56:39 2016



39. Reading out 16 Bytes of data – Byte 15 = 0x0.



Figure 9-39. Reading out 16 Bytes of Data – Byte 15 = 0x0

9.2 Read FSN waveform

- 1. Checking hardware status.
- Figure 9-40. Hardware Status Check

MS0-X 2024A, MY52490979: Tue Jul 26 16:57:30 2016



2. Checking hardware status.



3. Clocking in $read_{FSN}$ command (0x18).



Clocking in 16 bytes of zero values - Byte 0.
 Figure 9-43. Clock in 16 Bytes of Zero Values - Byte 0

MS0-X 2024A, MY52490979: Tue Jul 26 17:01:58 2016



5. Clocking in 16 bytes of zero values - Byte 1.



Figure 9-44. Clock in 16 Bytes of Zero Values - Byte 1

Clocking in 16 bytes of zero values - Byte 2.
 Figure 9-45. Clock in 16 Bytes of Zero Values - Byte 2

24.085 200.0%/ Stop D_0 TTL Ŧ KEYSIGHT TECHNOLOGIES Normal 1.00GSa/s DC 10.0:1 10.0:1 MISO D. AX: +1.318000us MOSI 1/ΔX: D. +758.73kHz SCK D. CS D, Cursors Menu X1: 24.256000us X2: 25.574000us Mode Source Cursors Units 5 Manual <None> X1 X2

MS0-X 2024A, MY52490979: Tue Jul 26 17:02:37 2016

Clocking in 16 bytes of zero values - Byte 3.
 Figure 9-46. Clock in 16 Bytes of Zero Values - Byte 3

MS0-X 2024A, MY52490979: Tue Jul 26 17:02:22 2016



8. Clocking in 16 bytes of zero values - Byte 4.



Figure 9-47. Clock in 16 Bytes of Zero Values - Byte 4

Clocking in 16 bytes of zero values - Byte 5.
 Figure 9-48. Clock in 16 Bytes of Zero Values - Byte 5



MS0-X 2024A, MY52490979: Tue Jul 26 17:03:01 2016

Clocking in 16 bytes of zero values - Byte 6. Figure 9-49. Clock in 16 Bytes of Zero Values - Byte 6

MS0-X 2024A, MY52490979: Tue Jul 26 17:03:12 2016



11. Clocking in 16 bytes of zero values - Byte 7.



Figure 9-50. Clock in 16 bytes of zero values - Byte 7

Clocking in 16 bytes of zero values - Byte 8.
 Figure 9-51. Clock in 16 Bytes of Zero Values - Byte 8



MS0-X 2024A, MY52490979: Tue Jul 26 17:03:38 2016

Clocking in 16 bytes of zero values - Byte 9. Figure 9-52. Clock in 16 Bytes of Zero Values - Byte 9

MS0-X 2024A, MY52490979: Tue Jul 26 17:03:48 2016



14. Clocking in 16 bytes of zero values - Byte 10.



Figure 9-53. Clock in 16 Bytes of Zero Values - Byte 10

15. Clocking in 16 bytes of zero values - Byte 11.



Clocking in 16 bytes of zero values - Byte 12. Figure 9-55. Clock in 16 Bytes of Zero Values - Byte 12

MS0-X 2024A, MY52490979: Tue Jul 26 17:04:21 2016



17. Clocking in 16 bytes of zero values - Byte 13.



Figure 9-56. Clock in 16 Bytes of Zero Values - Byte 13

18. Clocking in 16 bytes of zero values - Byte 14. Figure 9-57. Clock in 16 Bytes of Zero Values - Byte 14



MS0-X 2024A, MY52490979: Tue Jul 26 17:04:42 2016

Clocking in 16 bytes of zero values - Byte 15. Figure 9-58. Clock in 16 Bytes of Zero Values - Byte 15

MS0-X 2024A, MY52490979: Tue Jul 26 17:04:56 2016



20. Checking hardware status.



Figure 9-59. Hardware Status Check

21. Checking hardware status.

Figure 9-60. Hardware Status Check



MS0-X 2024A, MY52490979: Tue Jul 26 17:25:43 2016

SmartFusion2 and IGLOO2 SPI-Slave Programming Wave...

22. Clock in read co	ommand (0x5)							
Figure 9-61. Cl	ock in Read (Command	(0x5)					
MSO-X 2024A, MY5249	0979: Tue Jul 26 1	7:26:00 2016	6					
1 2	3	4		20.625	200.0\$/	Stop	₹ D _o	TTL
Ĭ								KEYSIGHT TECHNOLOGIES
							∷ Ac	quisition 🛛 🗄
							N N	lormal
							1.l	JUGSa/s
								hannale ::
								10 10 1
							DC	10.0:1
							DC	10.0:1
					to to to to to		DC	
D. MISO							# C	ursors ::
							ΔΧ:	
D MOCT								+1.318UUUus
				╧┥╴┍─┤─			17ΔX:	י דבט דטוע-
2			Ŧ					+700.70KHZ
D ₁ SCK								
D _o CS			-					
Cursors Menu								
📀 Mode	Source	C I	Cursors	Units	X1: 2	0.782000us		
Manual	<none></none>		X1 X2	. +	X2: 2	2. TUUUUUus		

23. Reading out 16 Bytes of FSN data – Byte 0 = 0x14.



Figure 9-62. Reading out 16 Bytes of FSN Data – Byte 0 = 0x14

24. Reading out 16 Bytes of FSN data – Byte 1 = 0x0.

Figure 9-63. Reading out 16 Bytes of FSN Data – Byte 1 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:26:25 2016



25. Reading out 16 Bytes of FSN data – Byte 2 = 0x12.

Figure 9-64. Reading out 16 Bytes of FSN Data – Byte 2 = 0x12

MS0-X 2024A, MY52490979: Tue Jul 26 17:26:39 2016



26. Reading out 16 Bytes of FSN data – Byte 3 = 0x.



Figure 9-65. Reading out 16 Bytes of FSN Data – Byte 3 = 0x

27. Reading out 16 Bytes of FSN data – Byte 4 = 0x13.

Figure 9-66. Reading out 16 Bytes of FSN Data – Byte 4 = 0x13



MS0-X 2024A, MY52490979: Tue Jul 26 17:27:01 2016

28. Reading out 16 Bytes of FSN data – Byte 5 = 0x0. Figure 9-67. Reading out 16 Bytes of FSN Data – Byte 5 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:27:12 2016



29. Reading out 16 Bytes of FSN data – Byte 6 = 0x44.



Figure 9-68. Reading out 16 Bytes of FSN Data – Byte 6 = 0x44

30. Reading out 16 Bytes of FSN data – Byte 7 = 0x0.

Figure 9-69. Reading out 16 Bytes of FSN Data – Byte 7 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:27:32 2016



Reading out 16 Bytes of FSN data – Byte 8 = 0x5A Figure 9-70. Reading out 16 Bytes of FSN Data – Byte 8 = 0x5A

MS0-X 2024A, MY52490979: Tue Jul 26 17:27:43 2016



32. Reading out 16 Bytes of FSN data – Byte 9 = 0xCD.



Figure 9-71. Reading out 16 Bytes of FSN Data – Byte 9 = 0xCD

33. Reading out 16 Bytes of FSN data – Byte 10 = 0x0.

Figure 9-72. Reading out 16 Bytes of FSN Data – Byte 10 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:28:03 2016



34. Reading out 16 Bytes of FSN data – Byte 11 = 0x0.

Figure 9-73. Reading out 16 Bytes of FSN Data – Byte 11 = 0x0

MS0-X 2024A, MY52490979: Tue Jul 26 17:28:13 2016



35. Reading out 16 Bytes of FSN data – Byte 12 = 0x4.



Figure 9-74. Reading out 16 Bytes of FSN Data – Byte 12 = 0x4

36. Reading out 16 Bytes of FSN data – Byte 13 = 0xD8.

Figure 9-75. Reading out 16 Bytes of FSN Data – Byte 13 = 0xD8



MS0-X 2024A, MY52490979: Tue Jul 26 17:28:38 2016

37. Reading out 16 Bytes of FSN data – Byte 14 = 0x88.

Figure 9-76. Reading out 16 Bytes of FSN Data – Byte 14 = 0x88

MS0-X 2024A, MY52490979: Tue Jul 26 17:28:48 2016



38. Reading out 16 Bytes of FSN data – Byte 15 = 0x13.



Figure 9-77. Reading out 16 Bytes of FSN Data – Byte 15 = 0x13

9.3 **Program Frame Waveform**

When performing Program, Verify or Authenticate actions, the data is clocked into the device starts at the beginning of the datastream block as shown in the following figure. This data is different depending on the device and the design, but in all cases, the data is clocked in 16 bytes at a time.

The following scope plots show how the first data frame is clocked.

Figure 9-78. Datastream Block

00000000000000000000000000000000000000
2000407E93B00500640000200000
datastream
7C5D1C2B3D7519B3924AABEE4ED56C6286545B7B3266FDF5975C868564A76E06DC5E14F6C16E6DE0
C7F14DFC0DA0412A0003000200000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
8C41036EF3EF6CC81E93FD29C4F2890B6FBA831F5F4A2BAB62092E6A9C604E7E9AC405860A7B097A
5B80D956D163A0012389AC4DB418B59558AB1A2F1CF5703EFB1915B8C20DA85FB053B7DA68E5490C
6B3E0239D719E17F9F2869F240AC43034122933EC8A0D2F58CBBD85B5AA8B224749635569CD51E95
CD18AC8359DCA64994043B10DA5B9A7596CC21727F01D3C8D20279283AD77D2726F1549822F1B52A
010300020000000000000000000000000000000
00000000000000000000000000000000000000

In this example, the following data is clocked.

SmartFusion2 and IGLOO2 SPI-Slave Programming Wave...

Table 9-1. Clocked Data

Bytes															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7C	5D	1C	2B	3D	75	19	B3	92	4A	AB	EE	4E	D5	6C	62

Mode 3 of the SPI mode is used and the data is clocked byte 0 MSB first. Note the following:

- 1. Before performing any data shift, the target device SPI buffer status is checked by shifting 0xff. This is the only instruction that is 8 bit long, and the data is read out at the same time as it is shifted in. The result of the first shift is ignored.
- 2. When shifting data, into the device, the first byte is the command followed by 16 bytes of data. 16 bytes of zero value must be shifted for commands that do not require data.
- 3. Shifting data out from the device is a two steps operation. The command is clocked into the device first and then the data is clocked out using a read command of 0x5.
- All operations except for SPI hardware status check are made of one byte of command followed by 16 bytes of data. Chip Select (CS) line must be driven low before clocking the command and should remain low until the last bit of data is shifted in. Then, it must be driven high to execute the loaded instruction.
 Note: 1, 2, and 3 are taken care of by the programming algorithm.
- 1. Checking hardware status.

Figure 9-79. Hardware Status Check

MS0-X 2024A, MY52490979: Mon Jun 06 13:02:09 2016

1	2				-990.0°s	500.	Os/	Stop	ť	1.44V
									*	KEYSIGHT
										Acquisition 🛛 🗧
										Normal
										1.00GSa/s
									::	Channels ::
									DC	10 0.1
									DC	10.0:1
									DC	10.0:1
									DC	10.0:1
D,	MISO			+						Cursors #
0									ΔΧ:	. 44 10000
n.	MOSI			Ţ					1/ ΔΥ	+44.100000us
2									ПДА	+22.676kHz
D	SCK			Ť.						
U ₁	JUN									
Do	CS			-						
Pr	int Configurat	ion Menu				-				
•	Print to A R METERS		ptions	€ Palette	Network	Setup				Press to
3	(話 NETPHI		etup info	Lolor						Print

2. Checking hardware status.



3. Shift in the first frame. Command = 0x1. Data to follow. Note CS signal.

Figure 9-81. Shift in the First Frame. Command = 0x1. Data to Follow. Note CS Signal



MS0-X 2024A, MY52490979: Mon Jun 06 13:05:48 2016

SmartFusion2 and IGLOO2 SPI-Slave Programming Wave...



5. Data Byte 1 = 0x5D



Data Byte 2 = 0x1C Figure 9-84. Data Byte 2 = 0x1C





SmartFusion2 and IGLOO2 SPI-Slave Programming Wave...



8. Data Byte 4 = 0x3D


Data Byte 5 = 0x75

Figure 9-87. Data Byte 5 = 0x75



10. Data Byte 6 = 0x19

Figure 9-88. Data Byte 6 = 0x19

MS0-X 2024A, MY52490979: Mon Jun 06 13:11:11 2016



11. Data Byte 7 = 0xB3

Figure 9-89. Data Byte 7 = 0xB3

MS0-X 2024A, MY52490979: Mon Jun 06 13:11:46 2016



12. Data Byte 8 = 0x92



13. Data Byte 9 = 0x4A

Figure 9-91. Data Byte 9 = 0x4A



MS0-X 2024A, MY52490979: Mon Jun 06 13:12:45 2016

14. Data Byte 10 = 0	xAB 22 Byte 10 = 0xAl	8				
MS0-X 2024A MY52490	979: Mon Jun 116 1 3:1:	3 R09.2016				
1 2	3	4	33.30 5	200.0°J/	Stop	₹ 1 1.44V
Ý						
						# Acquisition #
						Normal
						1.00038/5
						። Channels ።
						DC 10.0:1
						DC 10.0:1
				<u></u>		
D MISO						: Cursors :
03 M130						ΔΧ:
n MOST						+1.318000us
						+758 73kHz
D, CS						
Save to file = [scope_	_35					
Save	Recall	Default/Erase				Press to
						3416

15. Data Byte 11 = 0xEE



16. Data Byte 12 = 0x4E

Figure 9-94. Data Byte 12 = 0x4E



MS0-X 2024A, MY52490979: Mon Jun 06 13:14:00 2016

17. Data Byte 13 = 0.	xD5 • Bute 12 = 0xD6					
MS0-X 2024& MV524909	а Буце 13 – 0х D а 179: Mont.lun 06 13:14	• 1-20-2016				
1 2	3	4	37.24s	200.0\$/	Stop	₹ 1 1.44V
						* Acquisition *
						Normal 1.00GSa/s
						:: Channels ::
						DC 10.0:1
						DC 10.0:1
					t t. t.	DC 10.0.1
						a Cursors a
						ΔX: +1.318000us
						+758.73kHz
D ₁ SCK						
D, CS		+++++++++++++++++++++++++++++++++++++++				
Save to file = [scope_	_38		_			
Save	Hecall	Default/Erase				Press to Save

18. Data Byte 14 = 0x6C



At this point, the first frame of data is clocked in. The next operation is to check the status.
 Figure 9-97. Data Byte 15 = 0x62. Note CS signal



MS0-X 2024A, MY52490979: Mon Jun 06 13:15:15 2016



21. Framing status command. Command = 0x4. Note CS signal.



Figure 9-99. Frame Status command. Command = 0x4. Note CS signal

22. Framing status command. Command = 0x4. Note CS signal.

Figure 9-100. Frame Status Command. Command = 0x4. Note CS signal



MS0-X 2024A, MY52490979: Mon Jun 06 13:19:45 2016

23.	Data B Figure	yte 0 = 0 9-101. E	x0)ata Byt e	e 0 = 0x0							
M	SO-X 2024/	A, MY5249	0979: Mon	Jun 06 13:20:0	8 2016						
1		2	3	4		20.12	š 200.	Os/	Stop	t 1	1.44V
		Y								*	KEYSIGHT
										:	Acquisition 🛛 🗄
											Normal
											1.UUGSa/s
										·	Channala
										DC	nutrianners :: 10 ∩·1
											10.0.1
										DC	10.0:1
										DC	
n	MISO									::	Cursors 🛛
^D 3	11200									ΔΧ:	
	MOCT									4.1.437	+1.318000us
U ₂	MUSI									_1/ΔX	
											+/58./3kHz
D_1	SCK										
Do	CS									e	
Sa	we to file	=[scope	e_42								
	Sai	/e	F	Recall	Default/Eras	е					Press to
36	-			+	+						Save

24. Data Byte 1 = 0x0



25. Data Byte 2 = 0x0

Figure 9-103. Data Byte 2 = 0x0



MS0-X 2024A, MY52490979: Mon Jun 06 13:20:47 2016



27. Data Byte 4 = 0x0



28. Data Byte 5 = 0x0

Figure 9-106. Data Byte 5 = 0x0



MS0-X 2024A, MY52490979: Mon Jun 06 13:21:33 2016

29.	Data By	te 6 = 0x(C							
	Figure 9	9-107. Da	ta Byte (6 = 0x0						
MS	30-X 2024A,	MY524909	179: Mon Ju	n 06 13:21:4	5 2016					
1		2	3	4		28.01皆	200.0\$/	Stop	₹ 1	1.44V
										KEYSIGHT
									: I	Acquisition 🛛 🗄
										Normal
										1.UUGSa/s
										Channele ::
									DC .	
									DC	10.0:1
									DC	10.0:1
									DC	10.0.1
D.	MISO								::	Cursors 🛛
- 0									ΔX:	. 4 040000
n	MAST								1/AV-	+1.318000us
02										+758 73kHz
										3 7 00.7 0KHZ
U ₁	SUK									
D_0	CS _				+					
Sa	ve to file =	scope_	48		D-4					D
			He	call	Default/Erase					Press to
						3				0000

30. Data Byte 7 = 0x0



31. Data Byte 8 = 0x0

Figure 9-109. Data Byte 8 = 0x0



MS0-X 2024A, MY52490979: Mon Jun 06 13:22:26 2016

32.	Data By	/te 9 = 0x(C							
	Figure	9-110. Da	ta Byte 9	= 0x0						
М	SO-X 2024A	, MY524909	179: Mon Jur	n 06 13:22:4	2 2016					
1		2	3	4		31.955	200.0\$/	Stop	¥ 1	1.44V
									*	KEYSIGHT TECHNOLOGIES
									11 - A	Acquisition 🛛 🗄
										Normal
										1.UUGSa/s
										Channale ::
									DC	
									DC	10.0:1
									DC	10.0:1
									DC	10.0:1
D.	MISO									Cursors #
- 0									AX:	. 4 040000
n	MOST									+1.318000us
02										+758 73kHz
	CCK				-					1700.70km
U ₁	SUK .					┛				
Do	CS .									
Sa	we to file =	= scope_	51	211	D-1					Description
	294	8	Hec	air	Default/Erase					Press to Save
		1				3				Jaile

33. Data Byte 10 = 0x0



34. Data Byte 11 = 0x0 Figure 9-112. Data Byte 11 = 0x0



MS0-X 2024A, MY52490979: Mon Jun 06 13:23:05 2016

35.	Data By	te 12 = 0	x0										
	Figure 9	9-113. Da	ta Byte	12 = 0x0									
M	SO-X 2024A,	MY524909	979: Mon Ji	un 06 13:23:1	9 2016								
1		2	3	4			35.905	200).0°s/	Stop	ť	: 1	1.44V
		Ý									•	**	KEYSIGHT TECHNOLOGIES
												Ą	cquisition 🛛 🗄
												1	Normal .00GSa/s
													Channels :::
											D	C	10.0:1
											D	C	10.0:1
	- te te te										D	C	10.0:1
											U	IC	10.0:1
D3	MISO				-							X٠.	Cursors ::
													+1.318000us
D_2	MOSI										1	/ΔΧ:	
D.	SCK				-								+758.73kHz
D.	cs												
Sa	we to file =	scope	54	014-									
	Save International Save		Re	call	Default/	Erase							Press to Save

36. Data Byte 13 = 0x0



37. Data Byte 14 = 0x0

Figure 9-115. Data Byte 14 = 0x0



MS0-X 2024A, MY52490979: Mon Jun 06 13:23:49 2016

38.	Data Byte 15	5 = 0x0	E - 0.40						
M	CLY 2024A MV5	7/10/1070 Mon Ju	5 - UXU • 06 13:24:10	12016					
1	20-X 2024A, M10	2430373. MOH 30 3	1100 13.24.10 4	12010	39.85s	200.0\$/	Stop	₹ 1	1.44V
				+				•	KEYSIGHT TECHNOLOGIES
								:: А	cquisition =
								1	Normal .00GSa/s
								::	Channels ==
								DC DC DC	10.0:1 10.0:1 10.0:1
								DC	10.0:1
D3	MISO			(**) (**)				:: AV-	Cursors =
								ΔΛ.	+1.318000us
D_2	MOSI			-				1/ΔX:	
D1	SCK								+758.73kHz
D _o	cs								
Sa	ve to file = [sco	ope_57							
	Save +	Rec	all	Default/Erase					Press to Save

39. Instruction is loaded. Issue read instruction using 0x5 command.



Figure 9-117. Hardware Status Check

40. Checking hardware status.

Figure 9-118. Hardware Status Check



MS0-X 2024A, MY52490979: Mon Jun 06 13:47:21 2016



42. Reading Data Byte 0.



43. Reading Data Byte 1.

Figure 9-121. Data Byte 1 read



MS0-X 2024A, MY52490979: Mon Jun 06 13:48:21 2016



45. Reading Data Byte 3.



46. Reading Data Byte 4.

Figure 9-124. Data Byte 4 Read

27.15s Stop 1.44V 200.0%/ KEYSIGHT TECHNOLOGIES X1 = 27.344000us X2 = 28.662000us Normal 1.00GSa/s DC 10.0:1 10.0:1 MISO AX: +1.318000us MOSI 1/ΔX: D. +758.73kHz SCK D. CS D, Save to file = [scope_65 Save Recall Default/Erase Press to Save

MS0-X 2024A, MY52490979: Mon Jun 06 13:49:07 2016

47. Reading Data Byte 5. Figure 9-125. Data Byte 5 Read MS0-X 2024A, MY52490979: Mon Jun 06 13:49:20 2016 28.465 200.0%/ Stop 1.44V 1 2 4 ł KEYSIGHT Normal 1.00GSa/s 10.0:1 DC 10.0:1 MISO n ΔX: +1.318000us MOSI 1/ΔX: D., +758.73kHz SCK D, CS Do Save to file = scope_66 Recall Save Default/Erase Press to -Save -

48. Reading Data Byte 6.



49. Reading Data Byte 7.

Figure 9-127. Data Byte7 Read



MS0-X 2024A, MY52490979: Mon Jun 06 13:49:44 2016

50. Reading Data Byte 8.

Figure 9-128. Data Byte 8 Read

MS0-X 2024A, MY52490979: Mon Jun 06 13:49:54 2016



51. Reading Data Byte 9.



52. Reading Data Byte 10.

Figure 9-130. Data Byte 10 Read



MS0-X 2024A, MY52490979: Mon Jun 06 13:50:19 2016



54. Reading Data Byte 12.



55. Reading Data Byte 13.

Figure 9-133. Reading Data Byte 13



MS0-X 2024A, MY52490979: Mon Jun 06 13:50:52 2016

56. Reading Data Byte 14.

Figure 9-134. Data Byte 14 Read

MS0-X 2024A, MY52490979: Mon Jun 06 13:51:04 2016



57. Reading Data Byte 15. Figure 9-135. Data Byte 15 Read



10. Revision History

Revision	Date	Description
A	09/2021	 Migrated to the Microchip standard template format. Updated for the new version numbering schema (v202x.x) for DirectC solutions.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call 800.262.1060
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, 650.318.8044

The Microchip Website

Microchip provides online support via our website at 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 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: 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 specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.

- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- · Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "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 is provided for the sole purpose of designing with and using Microchip products. Information 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.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". 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 ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. 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, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, 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, 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, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, 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, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, 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.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-8876-7

Quality Management System

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


Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	Australia - Sydnev	India - Bangalore	Austria - Wels
2355 West Chandler Blvd.	Tel: 61-2-9868-6733	Tel: 91-80-3090-4444	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	China - Beiiing	India - New Delhi	Fax: 43-7242-2244-393
Tel: 480-792-7200	Tel: 86-10-8569-7000	Tel: 91-11-4160-8631	Denmark - Copenhagen
Fax: 480-792-7277	China - Chengdu	India - Pune	Tel: 45-4485-5910
Technical Support:	Tel: 86-28-8665-5511	Tel: 91-20-4121-0141	Fax: 45-4485-2829
www.microchip.com/support	China - Chongging	Japan - Osaka	Finland - Espoo
Web Address:	Tel: 86-23-8980-9588	Tel: 81-6-6152-7160	Tel: 358-9-4520-820
www.microchip.com	China - Dongguan	Japan - Tokvo	France - Paris
Atlanta	Tel: 86-769-8702-9880	Tel: 81-3-6880- 3770	Tel: 33-1-69-53-63-20
Duluth. GA	China - Guangzhou	Korea - Daegu	Fax: 33-1-69-30-90-79
Tel: 678-957-9614	Tel: 86-20-8755-8029	Tel: 82-53-744-4301	Germany - Garching
Fax: 678-957-1455	China - Hangzhou	Korea - Seoul	Tel: 49-8931-9700
Austin. TX	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	Germany - Haan
Tel: 512-257-3370	China - Hong Kong SAR	Malaysia - Kuala Lumpur	Tel: 49-2129-3766400
Boston	Tel: 852-2943-5100	Tel: 60-3-7651-7906	Germany - Heilbronn
Westborough, MA	China - Naniing	Malavsia - Penang	Tel: 49-7131-72400
Tel: 774-760-0087	Tel: 86-25-8473-2460	Tel: 60-4-227-8870	Germany - Karlsruhe
Eax: 774-760-0088	China - Qingdao	Philippines - Manila	Tel: 49-721-625370
Chicago	Tel: 86-532-8502-7355	Tel: 63-2-634-9065	Germany - Munich
Itasca II	China - Shanghai	Singanore	Tel: 49-89-627-144-0
Tel: 630-285-0071	Tel: 86-21-3326-8000	Tel: 65-6334-8870	Fax: 49-89-627-144-44
Eax: 630-285-0075	China - Shenyang	Taiwan - Hsin Chu	Germany - Rosenheim
Dallas	Tel: 86-24-2334-2829	Tel: 886-3-577-8366	Tel: 49-8031-354-560
Addison TX	China - Shenzhen	Taiwan - Kaobsiung	Israel - Ba'anana
Tel: 072-818-7/23	Tel: 86-755-8864-2200	Tel: 886-7-213-7830	
Fax: 072-818-2024	China - Suzhou	Taiwan - Tainei	Italy - Milan
Detroit	Tel: 86-186-6233-1526	Tel: 886-2-2508-8600	Tel: 39-0331-742611
Novi MI	China - Wuhan	Thailand - Bangkok	Fax: 39-0331-466781
Tel: 248-848-4000	Tel: 86-27-5980-5300	Tel: 66-2-604-1351	Italy - Padova
Houston TX	China - Xian	Vietnam - Ho Chi Minh	Tel: 30-040-7625286
Tel: 281-894-5983	Tel: 86-29-8833-7252	Tel: 84-28-5448-2100	Netherlands - Drupen
Indiananolis	China - Xiamen	101. 04-20-0440-2100	Tel: 31-416-690399
Noblesville IN	Tel: 86-592-2388138		Eax: 31-416-690340
Tel: 317-773-8323	China - Zhuhai		Norway - Trondheim
Eax: 317-773-5453	Tel: 86-756-3210040		Tel: 47-72884388
Tel: 317-536-2380			Poland - Warsaw
l os Angeles			Tel: 48-22-3325737
Mission Vielo CA			Romania - Bucharest
Tal: 040 462 0523			
Eax: 040 462 0608			Spain Madrid
Tak. 949-402-9000			Tol: 34 01 708 08 00
Balaigh NC			Eax: 34 01 708 08 01
Tal: 010 844 7510			Sweden Cothenhora
			Tel: 46-31-704-60 40
Tel: 631-435-6000			Sweden - Stockholm
San lose CA			Tel: 46-8-5000-4654
Jan JUSE, CA			181. 40-0-3090-4034
Tol: 400-7-30-9110			Tal: 14 118 021 5900
161. 400-430-4270			101. 44-110-921-3000
			rax: 44-118-921-5820
Fax. 900-090-2010			