Compa Family CPLDs Device Slave SPI Interface Configuration and Programming Application Guide

(AN03009, V1.0)

(24.08.2021)

Revisions History

Document Revisions

Version	Date of Release	Revisions
V1.0	24.08.2021	Initial release.

About this Manual

Terms and Abbreviations

Terms and Abbreviations	Meaning
JTAG	Joint Test Action Group
CCS	Configuration Control System
SPI	Serial Peripheral Interface
UID	Unique ID

Related Documentation

The following documentation is related to this manual:

1. ***UG030004_Compa Family CPLDs Configuration User Guide***

Table of Contents

Tables

Figures

Chapter 1 Document Introduction

This document focuses on the application of the Pango Microsystems CPLDs slave SPI configuration mode, the software and hardware environment required for slave SPI configuration, and the setup of the internal special function registers of the CPLDs. The document concludes with an actual application example of configuring CPLDs with MCU, providing users with a complete configuration flow for reference.

Chapter 2 Configuration Details

2.1 Slave SPI Interface Description

With the slave SPI interface, users can configure/reconfigure the CRAM, program embedded FLASH, read/write feature control bits, etc.

The slave SPI interface provides a convenient and universal method for user configuration. Typically, devices such as CPU, MCU, DSP, etc., can act as a master while the CPLDs as a slave. This application can be illustrated by the following figure:

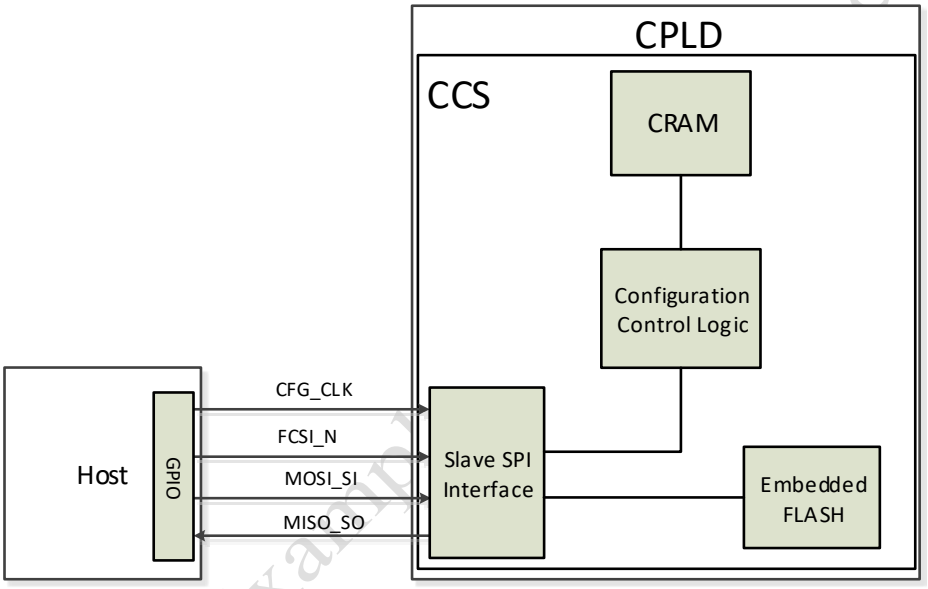

Figure 2-1 Example of Slave SPI Interface Application

The slave SPI pins are defined as shown in the following table:

Table 2-1 Slave SPI Pin Definition

Symbol	I/O	Description
CFG_CLK	I	Configuration clock, with write frequency up to 100MHz and read frequency up to 10MHz
FCSI_N	I	Chip Select signals that are active-low, sampled on the rising edge of CFG_CLK
MOSI_SI	I	Serial Data Input signal, sampled on the rising edge of CFG_CLK
MISO_SO	O	Serial Data Output signal, transmitted on the falling edge of CFG_CLK

When using the slave SPI interface, the CFG_CLK is provided by the master device. When reading data from the slave SPI interface (such as reading IDCODE, feature control bits, bitstream, etc.), the CFG_CLK frequency must not exceed 10MHz; when writing data through the slave SPI interface,

the frequency must not exceed 100MHz.

The slave SPI interface chip select signal FCSI_N is a synchronous signal sampled on the rising edge of CFG_CLK. When FCSI_N is at a high level, MISO_SO is in a high-impedance state. When FCSI_N is at a low level, the CPLDs can sample data through MOSI_SI or output data through MISO_SO. The slave SPI interfaces of different devices and packages of Compa family CPLDs differ in the distribution of pins. For related details, please refer to Table 4-1 or the corresponding device package manual.

Whether the slave SPI interface of the Compa family CPLDs serves as a configuration interface during the configuration or is retained as a configuration interface after the completion of the configuration is controlled by the feature control bits of the chip. See Feature Control Bits.

2.2 Feature Control Bits

The feature control bits of the Comp family CPLDs are stored in the embedded FLASH (Non-Volatile Memory) and are read by the chip CCS each time the chip is powered up. The CCS selects the function of the corresponding configurable multi-function PIN based on the feature control bits.

The CPLDs contains a feature control group CTL which is composed of 32-bit feature control bits. For details on the feature control bits, refer to the document "UG030004_Compa Family CPLDs Configuration User Guide".

The enablement of the salve SPI interface is controlled by the feature control bits cfg_sspi_en_n (CTL[6]) and persist_sspi_n (CTL[15]). When cfg_sspi_en_n (CTL[6]) is 0, the slave SPI interface is enabled in configuration mode; when persist_sspi_n (CTL[15]) is 0, the salve SPI interface is enabled in user mode. Please note that the salve SPI interface shares CFG_CLK, MISO_SO, and MOSI_SI with the master SPI interface. The master SPI and the slave SPI can be enabled simultaneously (the master SPI is controlled by cfg_mspi_en (CTL[4]) and persist_mspi (CTL[28])), but only one interface can be in use at any given moment. It is recommended not to enable the master SPI and the slave SPI simultaneously. If enabled simultaneously, please note when the master SPI interface is active under the following circumstances, the slave SPI interface should not be used by the user:

1. After the chip is powered on, the bitstream from the external SPI FLASH is actively retrieved through the master SPI interface until the chip configuration is completed or the SPI FLASH is fully read.
2. The master and slave SPI interfaces are preserved simultaneously in user mode and the automatic Soft Error Mitigation (SEM) is triggered.

The chip feature control bits can be modified by the following ways:

1. When the PDS (Pango Design Suite) download tool (Fabric Configuration) downloads a sbit file, it writes the feature control bits contained in the sbit file into the chip.
2. Separately program these features control bits of the device in PDS download tool.
3. Execute the JTAG interface instruction "PROGRAM_CTL" when JTAG interface is enabled.
4. Execute the slave SPI interface instruction "PROGRAM_CTL" when the salve SPI interface is enabled.
5. Execute the slave IIC interface instruction "PROGRAM_CTL", when the slave IIC interface is enabled.
6. Use the internal slave APB interface instruction "PROGRAM_CTL".

2.3 Slave SPI Instruction Set

Table 2-2 Configuration of CPLDs-related Slave SPI Instruction Set

Instruction	Description	Op Code	Operation Target
NOP	No Operation	FF	
RDID	Read Identification	A1	
RDUSER	Read Usercode	A2	
RDSR	Read Status Register	A3	
RDUID	Read Unique Identification	A4	
RDLOCK	Read Embedded Flash Lock Information	A5	
CFG	Config Bitstream	50	CRAM
WREN	Write Enable	51	
WRDIS	Write Disable	52	
RESET	Reset CPLDs	60	
ERASE	Erase Bulk	10	EFlash
ERASE_PAGE	Erase Page	11	EFlash
ERASE_CTL	Erase Function Control Page	12	EFlash (Function Control)
PROGRAM	Program Page	20	EFlash
PROGRAM_CTL	Program Function Control Page	22	EFlash (Function Control)
READ	Read	30	EFlash
READ_CTL	Read Function Control Page	31	EFlash (Function Control)
PROGRAM_LOCK	Lock Embedded Flash	40	EFlash
EFlash_SLEEP	Embedded Flash Sleep	70	EFlash
EFlash_WAKEUP	Embedded Flash Wake Up	71	EFlash

The FCSI_N signal needs to be pulled high for at least one clock cycle between instructions, all starting with the highest bit.

2.4 Configuration Flow

2.4.1 Configuration/Reconfiguration

Configuration refers to the process of configuring a CPLDs chip from the start until it enters user mode; Reconfiguration refers to the process of resetting (instruction reset or externally triggered RSTN hard reset), clearing all CRAM, exiting user mode, and then reconfiguring. The flow is as follows:

1. Power-up
2. When the value of INIT_FLAG_N becomes 1, write the RDID instruction to read the device IDCODE
3. Check the value of IDCODE
4. If IDCODE does not match, terminate the operation. If IDCODE matches, write the WREN instruction
5. Write the CFG instruction to load the bitstream
6. After the bitstream transmission is completed, if the salve SPI/slave I^2C interface is released for user use in user mode (with feature control bit persist_sspi_n/persist_si2c_n set to 1), check the values of INIT_FLAG_N and CFG_DONE to conclude the operation. If the slave SPI interface is used as a configuration interface in user mode, write the WRDIS instruction
7. Write the RDSR instruction to read the device status register

The CPLDs device can be cold-started (by setting RSTN to 0) at any time after power-up. After a cold start, users can proceed to step 2 to reconfigure the CPLDs device.

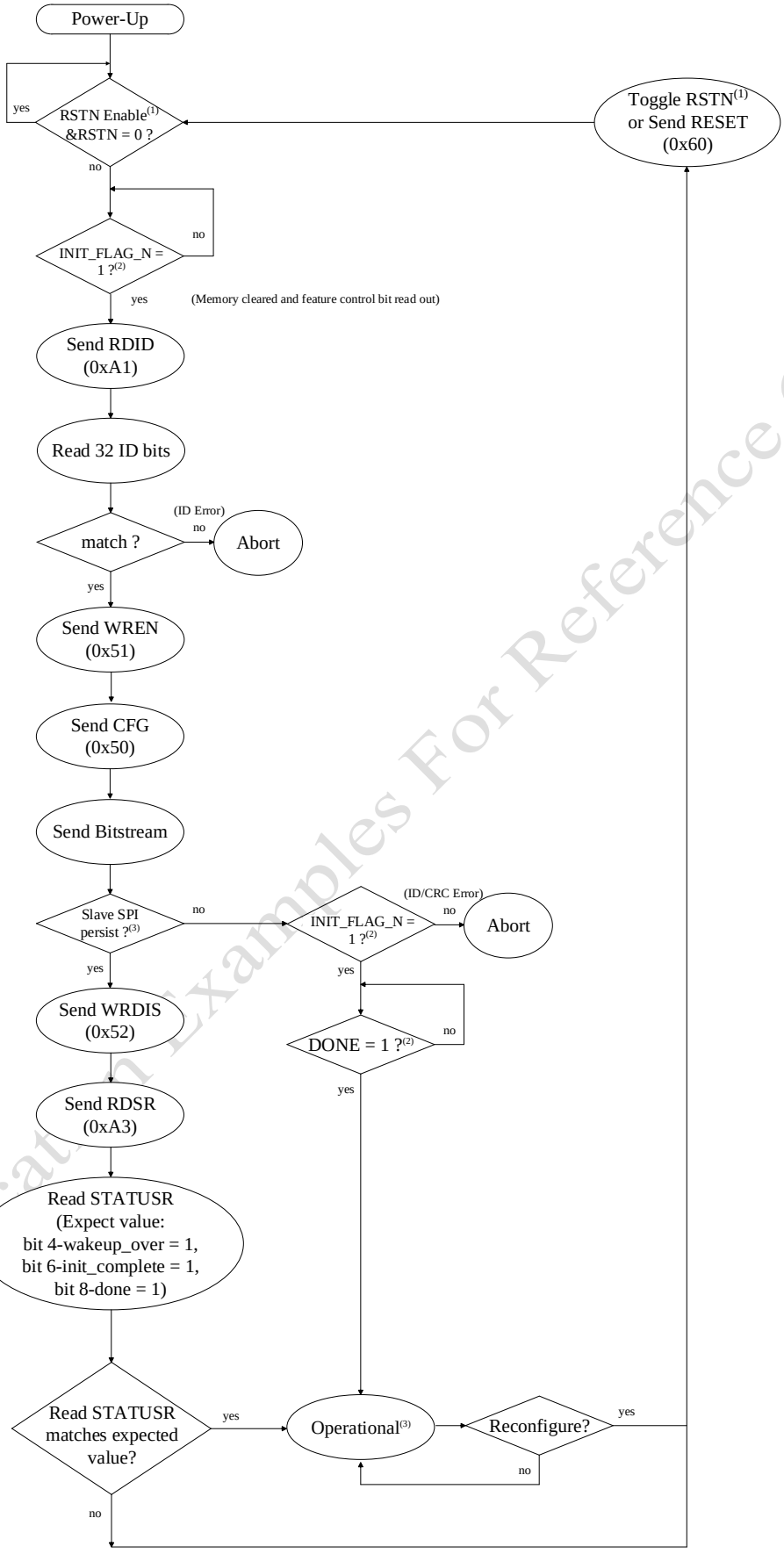

Figure 2-2 Configuration/Reconfiguration Flowchart

Notes:

1. If RSTN is enabled, the device will not be initialized until the RSTN is pulled high.

2. If INIT_FLAG_N is enabled, the pin can be used to indicate whether the initialization is completed or the loading is correct; when DONE is enabled, the pin can be used to indicate whether the loading is completed. However, if RSTN is enabled simultaneously, the evaluation of these two pins needs to be done after the release (pulling high) of RSTN, as these two pins may not accurately indicate the status when RSTN is low.

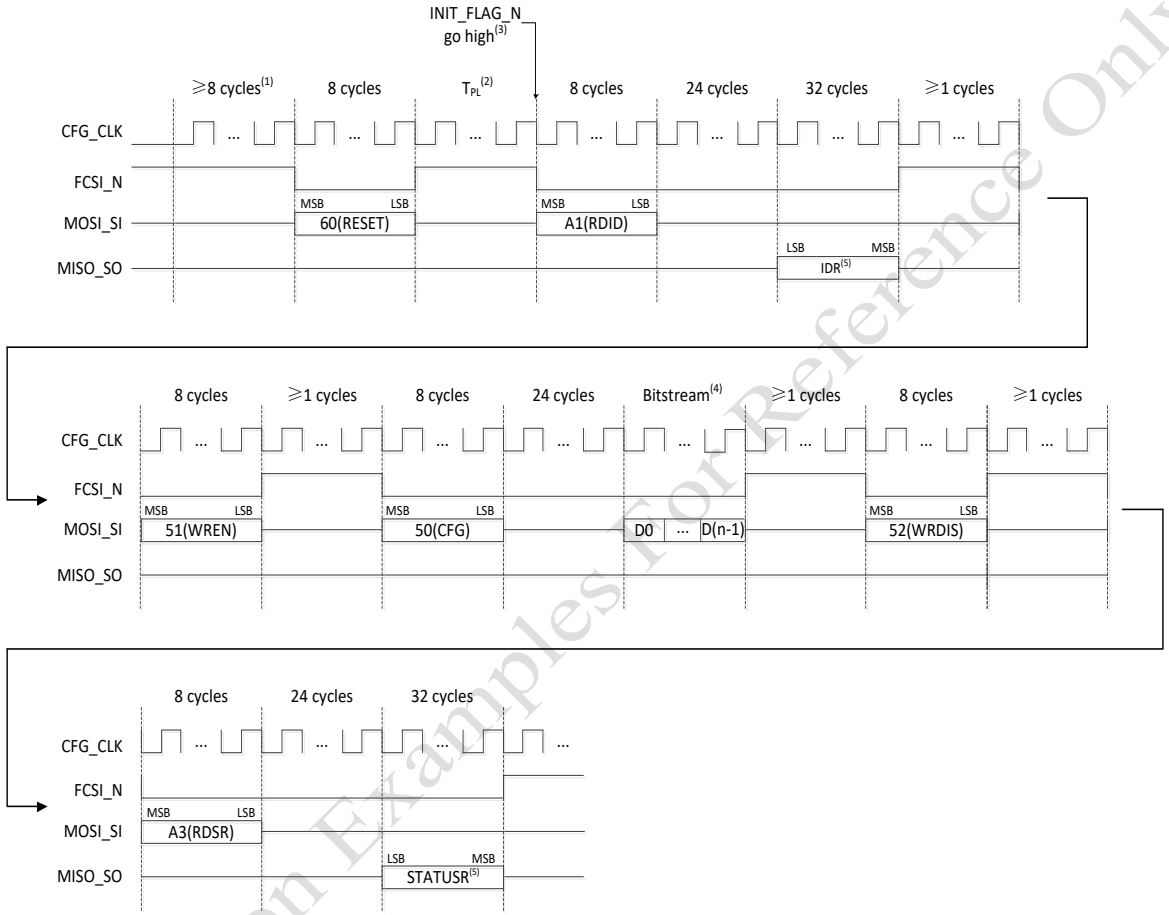

Figure 2-3 Configuration/Reconfiguration Flow and Timing

Notes:

1. After power-up, FCSI_N must be maintained high for at least 8 clock cycles before the first instruction can be executed successfully.

2. See the datasheet for T_{PL}.

3. If INIT_FLAG_N is enabled, the pin can indicate the completion of initialization. This is an example of an instruction reset; If RSTN hard reset is used, INIT_FLAG_N can only be determined after RSTN is released.

4. The supported bitstream formats in this context are bin (recommended), sbit, and sfc.; All of these file formats represent binary streams, and users can transmit them simply from start to finish. If the first byte of the bitstream is 0xAA (1010_1010), then D0=1, D1=0, ….

5. IDR refers to the Device Identification Register, whose value is the Device ID; STATUSR refers to the Status Register. For details of registers, see "UG030004_Compa Family CPLDs Configuration User Guide" The register value is read starting from the lower bits; for example, if the ID is 0x0049_9899, the shift-out order is 1, 0, 0, 1, 1....

6. Unless otherwise specified, FCSI_N must not be pulled high mid-process in any instruction execution flow (including dummy and data), as pulling high means the termination of the action.

2.4.2 Shutdown Reconfiguration

Shutdown reconfiguration refers to the process of reconfiguring the shutdown device without resetting, clearing all CRAMs, or exiting user mode. Shutdown reconfiguration requires a series of commands to be written to CCS via the CFG instruction. These commands modify the configuration registers to achieve the shutdown reconfiguration based on the bitstream package format.

Table 2-3 Shutdown Reconfiguration Command Flowchart of Slave SPI

Flow	**Data**
Write a CFG instruction to load:	0x50
100-pad bytes	0xFFFFFFFF …… 0xFFFFFFFF
SYNC WORD	0x01332D94
No operation	0xA0000000
Write an RSTCRC instruction to CMDR register	0xA8800001 0x00000001
Write to OPTION0R register (select "wakeup clock in shutdown mode")	0xAE400001 0x00000100
Write to CMASKR register	0xADC00001
Write to CTRLR register (disable masked variable memory readback)	0x00000010
Write a SWAKEDOWN instruction to CMDR register	0xA8800001 0x00000008
20 No operation	0xA0000000 …… 0xA0000000
Write a GDOWN instruction to CMDR register	0xA8800001 0x0000000A
Write a DESYNC instruction to CMDR register	0xA8800001 0x0000000B
Bitstream	Bitstream

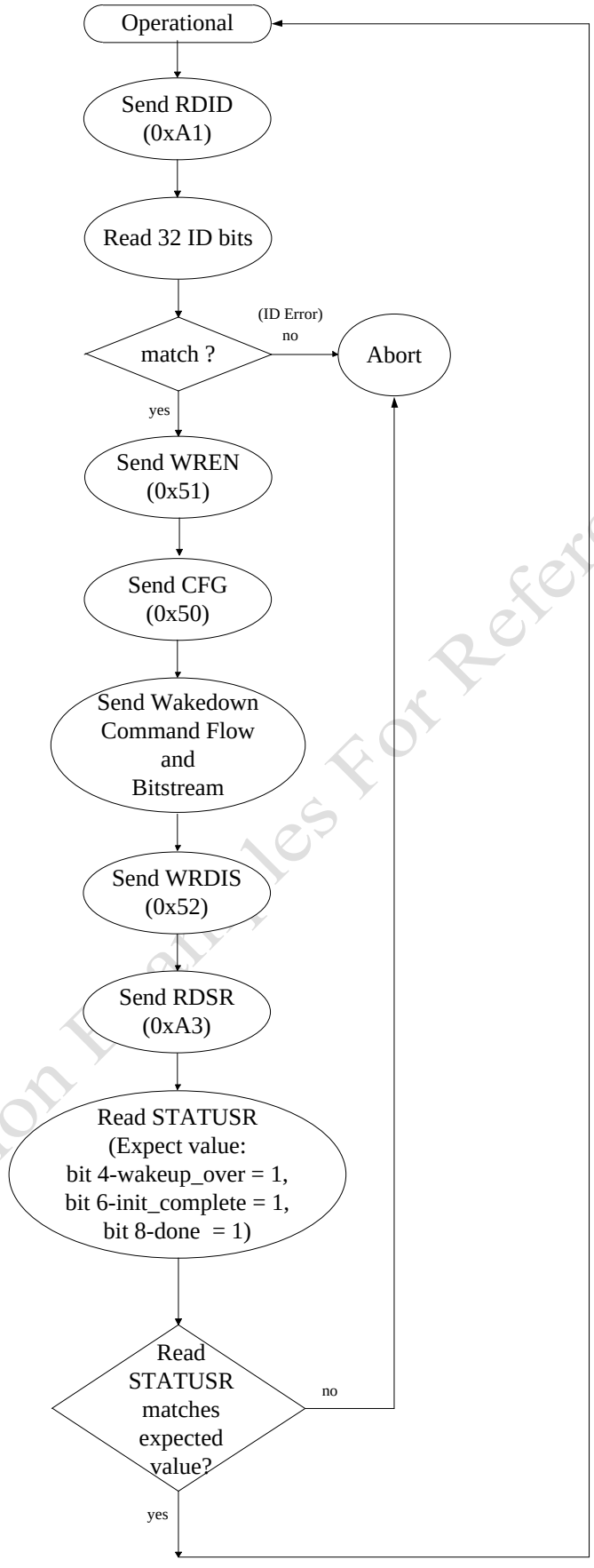

Figure 2-4 Shutdown Reconfiguration Flowchart

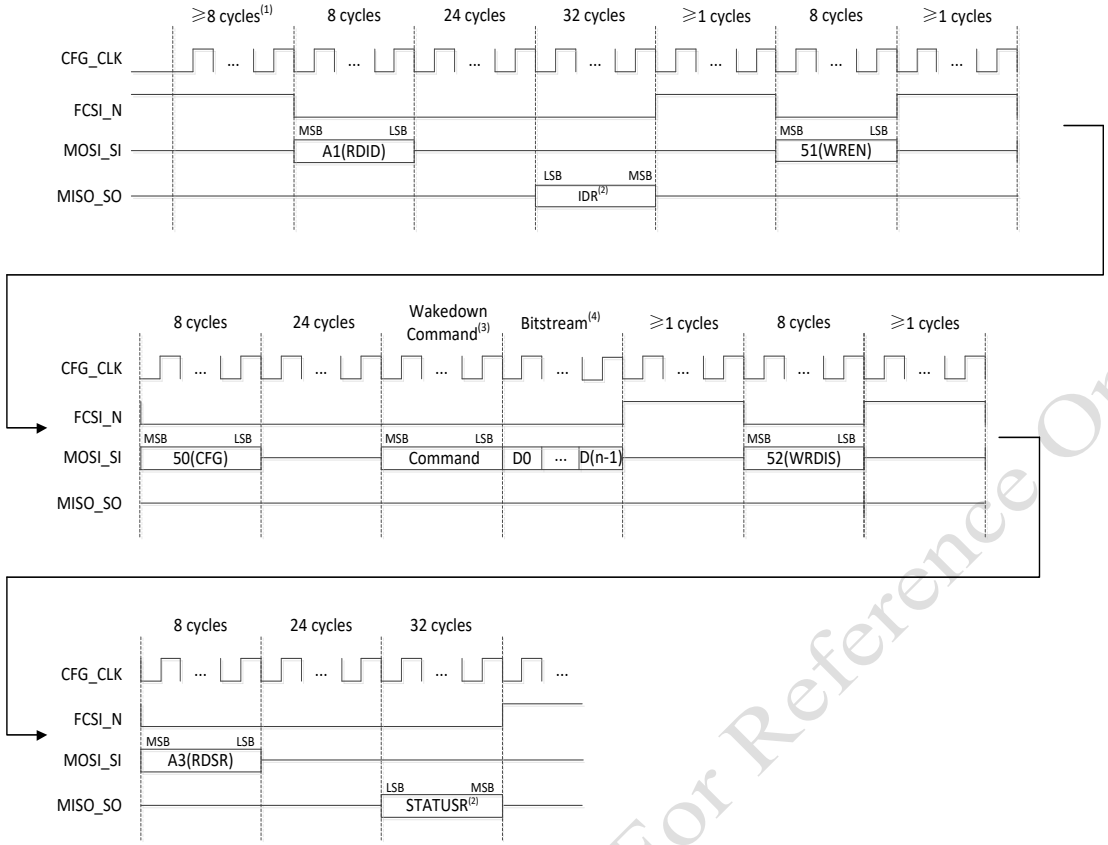

Figure 2-5 Shutdown Reconfiguration Flow and Timing

Notes:

1. After power-up, at least 8 clock cycles are required for the first instruction to be executed properly when FCSI_N is high.

2. IDR refers to the Device Identification Register, whose value is the Device ID; STATUSR refers to the Status Register. For details of registers, see "UG030004_Compa Family CPLDs Configuration User Guide". The register value is read starting from the lower bits; for example, if the ID is 0x0049_9899, the shift-out order is 1, 0, 0, 1, 1....

3. Here, the shutdown instruction stream is shifted-in starting with the highest bit; for instance, with 0x01332D94 (which is binary 0000_0001_0011…), the data shift-in order is 0, 0, 0, 0, 0, 0, 1, 0....

4. The supported bitstream formats in this context are bin (recommended), sbit, and sfc.; All of these file formats represent binary streams, and users can transmit them simply from start to finish. If the first byte of the bitstream is 0xAA (1010_1010), then D0=1, D1=0, …

5. Unless otherwise specified, FCSI_N must not be pulled high during any instruction execution (including dummy and data), as pulling high means the termination of the action.

2.5 Programming Embedded Flash

Embedded Flash supports direct operations through JTAG, slave SPI, and slave I^2C interfaces on the Embedded Flash. This section presents the operation flow of the slave SPI.

The complete flow of programming Embedded Flash through a slave SPI is as follows:

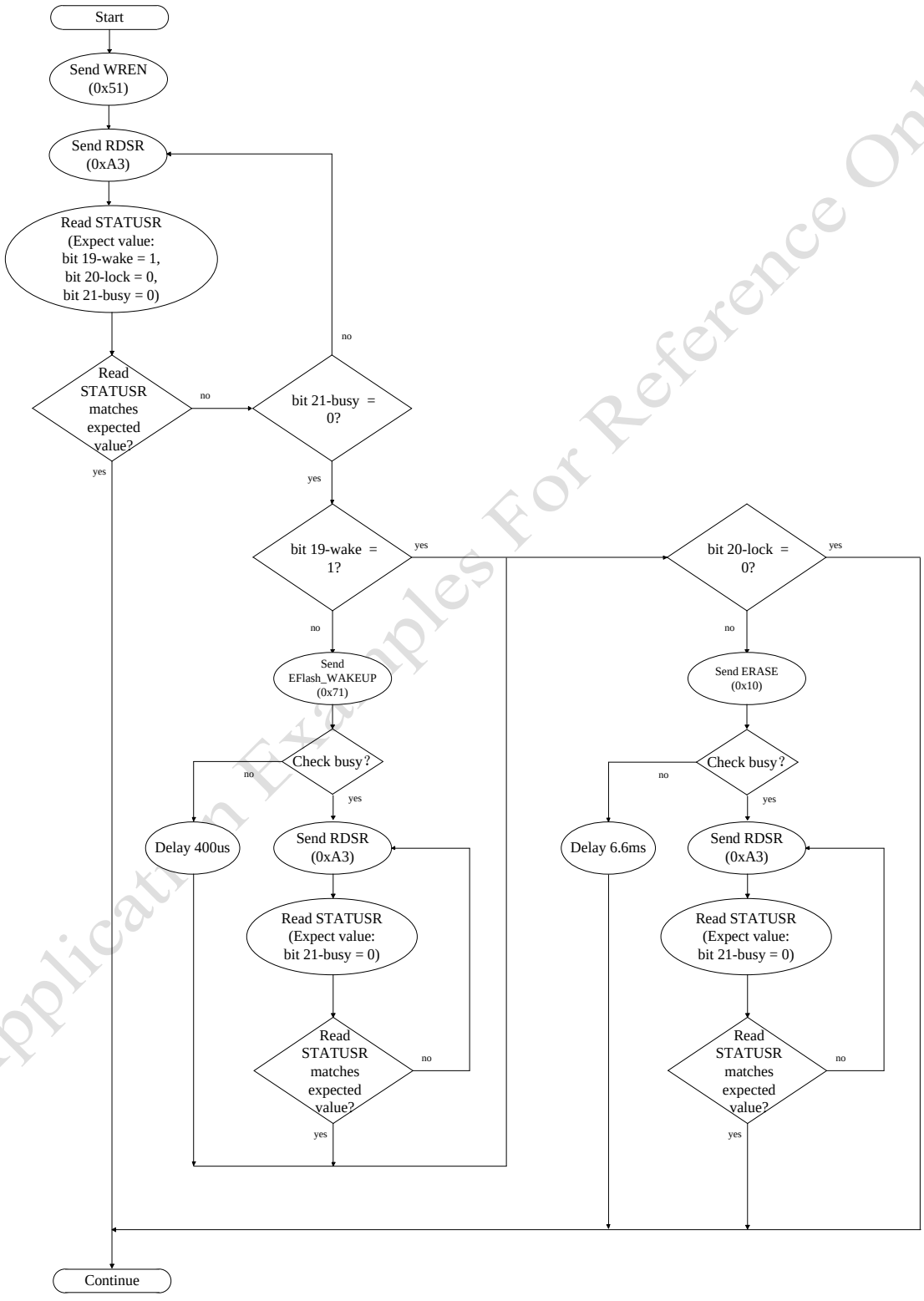

Figure 2-6 Programming Embedded Flash 1

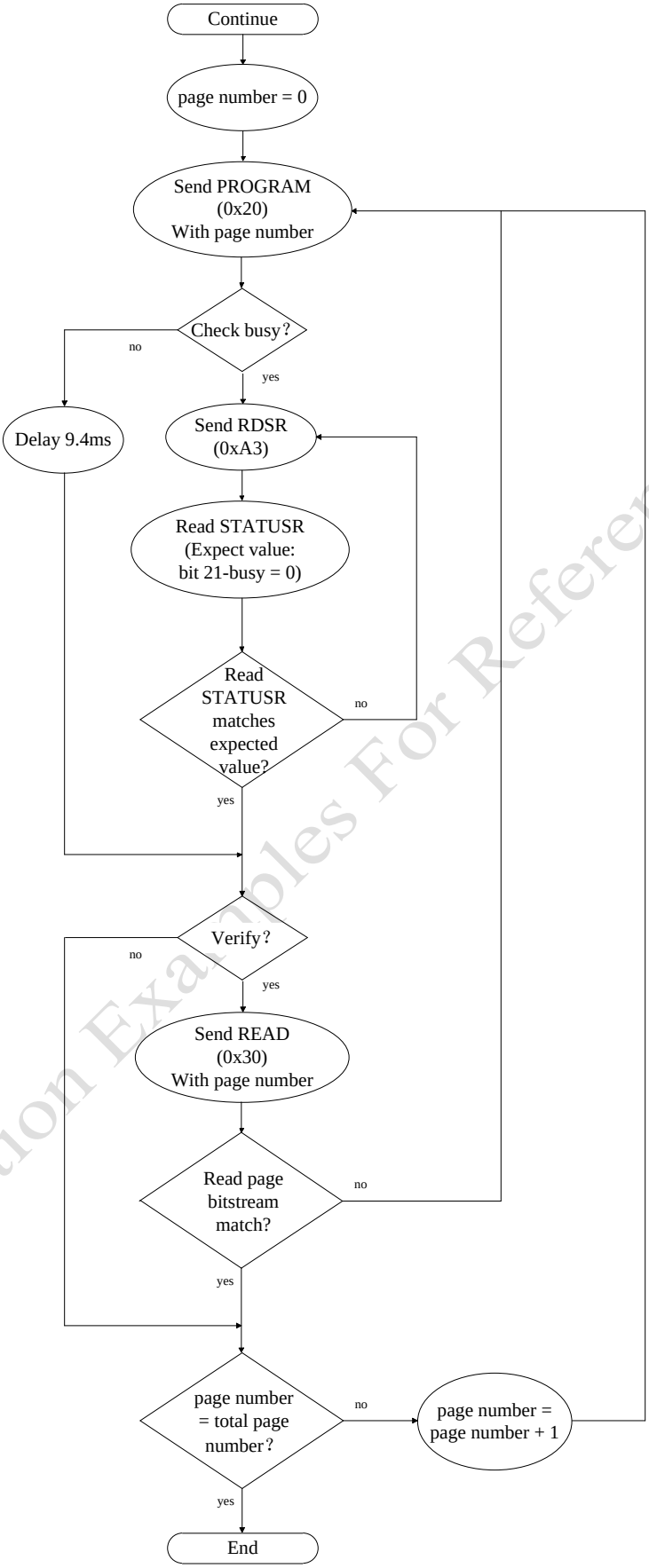

Figure 2-7 Programming Embedded Flash 2

An example of programming timing is shown in the figure below, which assumes that the device is waken-up and EFlash is not in a Lock state.

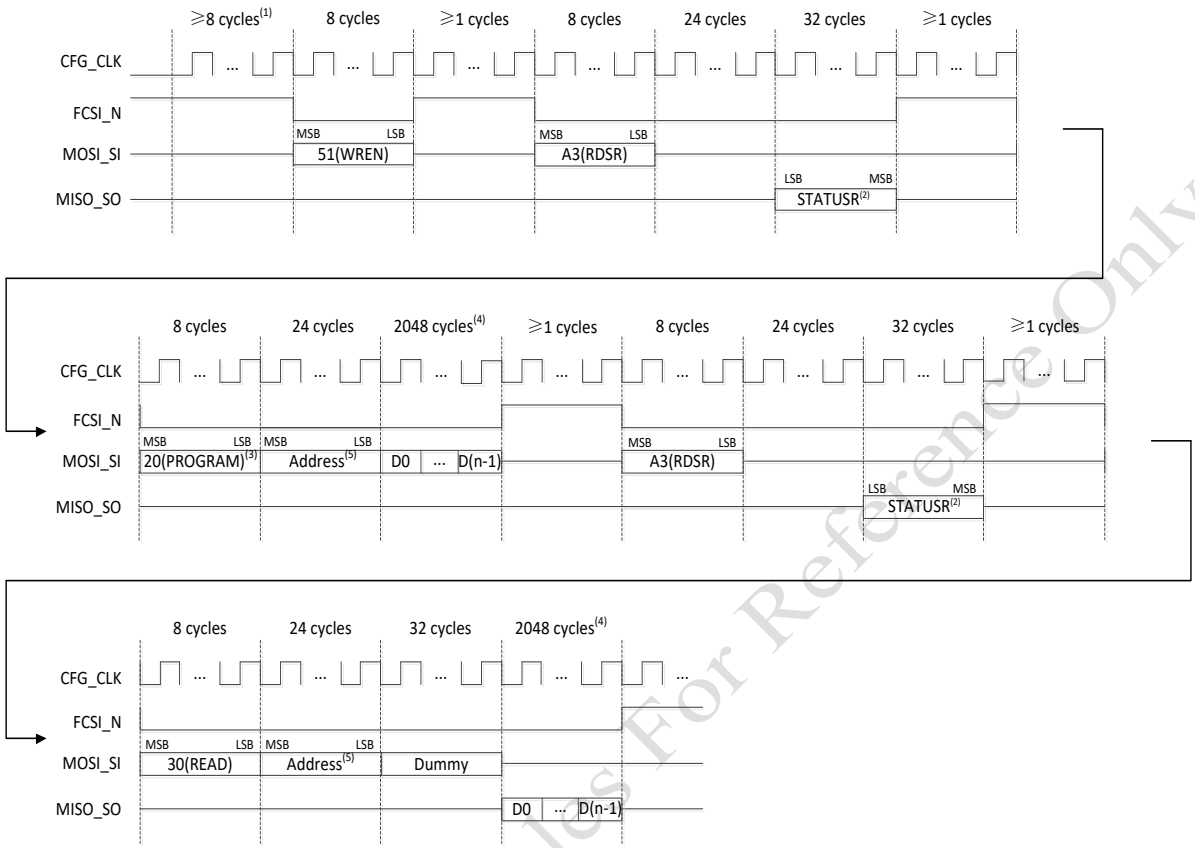

Figure 2-8 Shutdown Reconfiguration Flow and Timing

Notes:

1. After power-up, at least 8 clock cycles are required for the first instruction to be executed properly when FCSI_N is high.
2. STATUSR refers to the Status Register. For details of registers, see "UG030004_Compa Family CPLDs Configuration User Guide". The read of the register value starts from the lower bits.
3. The timing of the PROGRAM instruction varies for different devices, see PROGRAM timing for details.
4. The instructions are executed by page, with each page containing 256Bytes, and data is high-order first in/first out.
5. The Address mapping varies for different devices, see Address for details.
6. Unless otherwise specified, FCSI_N must not be pulled high mid-process in any instruction execution flow (including dummy and data), as pulling high means the termination of the action.

2.5.1 PROGRAM timing

PROGRAM refers to PROGRAM instruction.

Used for programming the normal page of embedded FLASH. The timing differs across devices.

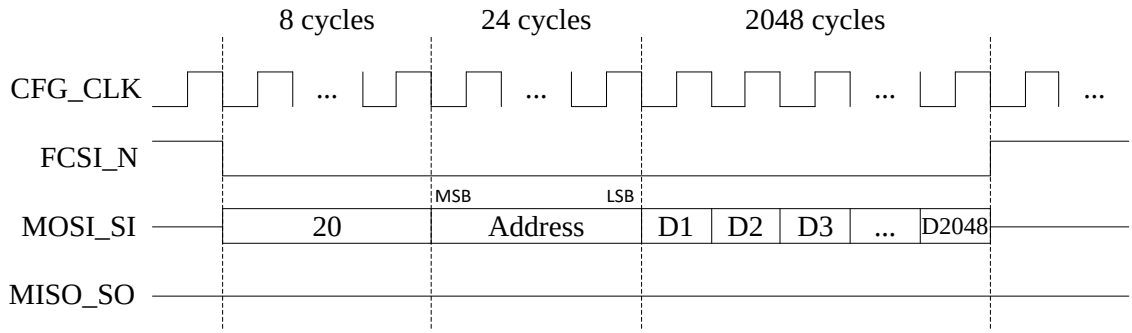

Figure 2-9 1K/2K/4K/7K PROGRAM Timing

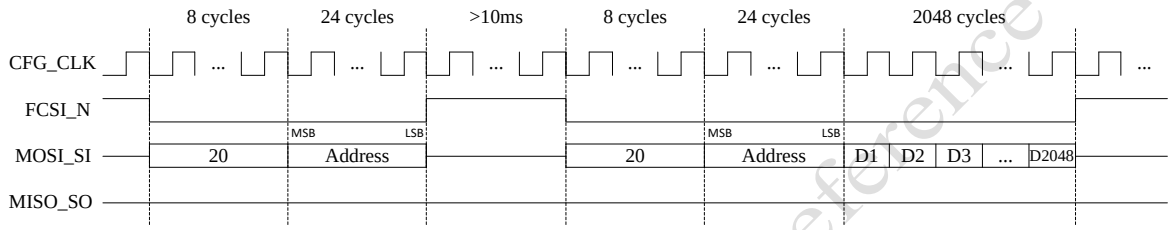

Figure 2-10 10K PROGRAM Timing

2.5.2 Address

The minimum addressing unit of PGC EFlash is a 32-bit word, with each page containing 256Bytes or 64 words. Different devices have different memory sizes, thus the total number of pages varies. For larger devices, EFlash is divided into multiple heaps, hence the Address is non-continuous. The memory size and address mapping of devices are shown below.

Table 2-4 Memory Sizes of Embedded Flash Ordinary Register

Device	**Number of Piles**	**Number of Pages Per Pile**	**Page Size (Bytes)**	**Total Capacity (Bytes)**	**Total Pages of Embedded Flash**
PGC1KG/L	1	332	256	84992	332
PGC2KG/L	1	332	256	84992	332
PGC4KD/L	4	320	256	327680	1280
PGC7KD	4	452	256	462848	1808
PGC10KD	4	640	256	655360	2560

Table 2-5 Definitions of 1K/2K/4K/7K Address

Address	**Item**	**Description**
[23:19]	Reserved	Must be set to 5'b00000
[18:17]	Ba[1:0]	Heap Address For 1K and 2K devices, it is reserved bit, while for the 2K devices, it must be set to 2'b11

Address	Item	Description
[16:8]	Ra[8:0]	Page Address
[7:6]	Reserved	Must be set to 2'b00
[5:0]	Ca[5:0]	32-bit data page internal offset address The PROGRAM instruction should be set to 6'b000000

Table 2-6 Definition of 10K Address

Address	Item	Description
[23:20]	Reserved	Must be set to 5'b00000
[19:18]	Ba[1:0]	Heap Address
[17:8]	Ra[8:0]	Page Address
[7:6]	Reserved	Must be set to 2'b00
[5:0]	Ca[5:0]	32-bit data page internal offset address The PROGRAM instruction should be set to 6'b000000

Chapter 3 Configuration Examples

This section introduces the application of PGC slave SPI configuration, using the example of controlling the embedded FLASH that reads/writes PGC and loading CRAM via STM32F103.
This example is based on the Wildfire STM32F103MINI hardware platform, with the software development environment being Keil5.

3.1 Hardware Connection

The hardware connection consists of two parts: the SPI connection between STM32 and CPLDs using Dupont lines, and the hardware connection between STM32 and the FLASH chip via a PCB circuit (Wildfire STM32MINI on-board STM32F103 and W25Q64 chips).

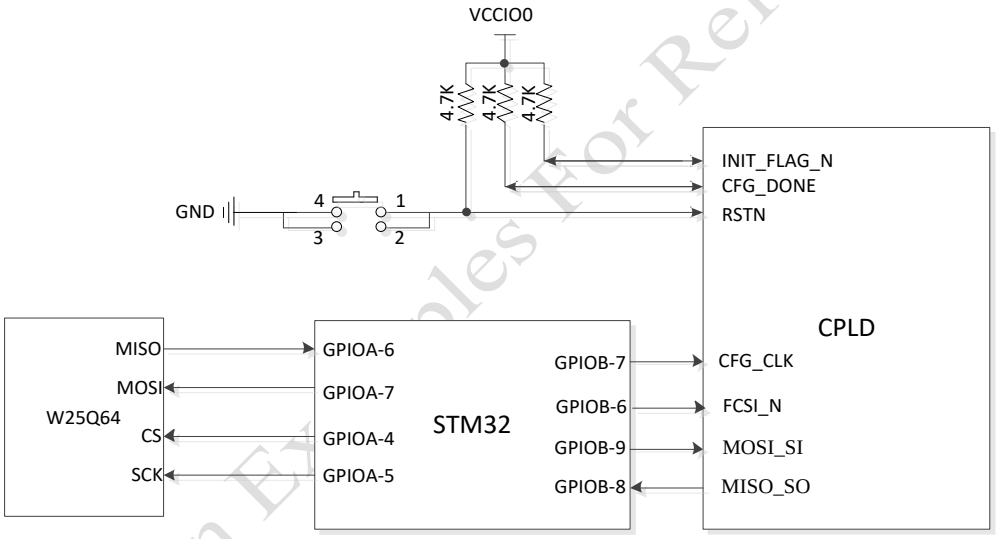

Figure 3-1 Example Hardware Circuit Diagram

During the configuration process, it is necessary to maintain the connection between the microcontroller's GPIO Pin for the simulated SPI interface and the PGC2K's SPI interface.

Table 3-1 Connection Between SPI Pins

STM32 Pins	**PGC2KG Pins**
GPIOB-6	FCSI_N
GPIOB-7	CFG_CLK
GPIOB-8	MISO_SO
GPIOB-9	MOSI_SI

In the example program, the bitstream of PGC2K is stored on the FLASH chip of the STM32 circuit board and the FLASH chip uses a standard SPI interface. The pin connection between the

microcontroller and the FLASH chip is shown in the following table.

Table 3-2 FLASH Chip SPI Pins

FLASH Chip Pins	**STM32 Pins**
CS	GPIOA-4
SCK	GPIOA-5
MISO	GPIOA-6
MOSI	GPIOA-7

When downloading the demonstration bit stream, it is necessary to connect the FLASH pins from Table 2-5 to the outside, writing the PGC2K bit stream starting from Address 0. Note that the reset button on the board must be kept pressed during the FLASH programming. Users can use Pango Fabric Configuration to download FLASH, simply by connecting these corresponding FLASH pins to the Pango USB Cable.

3.2 Setting Feature Control Bits

As described in Section 2.2, the feature control bits determine whether the slave SPI is enabled. To use the SPI slave, ensure that the feature control bits are set to enable the slave SPI interface. The chip is shipped with the slave SPI enabled by default. If it is accidentally disabled, users can enable it by modifying the feature control bits through reserved interfaces such as JTAG or the slave I2C interface. The following shows how the slave SPI is enabled via the JTAG interface by the PDS download tool.

When loading CRAM, the feature control bits are written by default. Users can enable the slave SPI (select Dedicated IO) in the bitstream settings, then the slave SPI will be enabled when users use the download tool Fabric Configuration to download the bitstream.

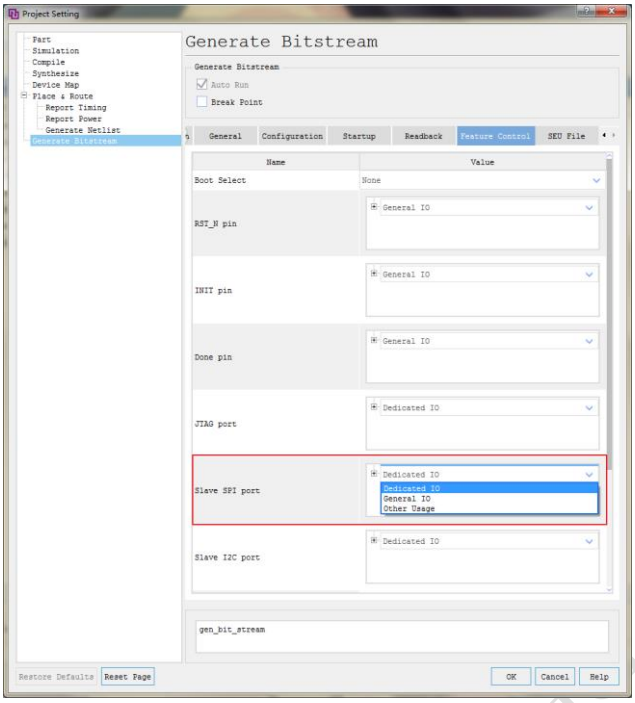

Figure 3-2 Setting Feature Control Bits in Bitstream Settings

Alternatively, users can use the download tool Fabric Configuration to directly modify the feature control bits (deselect the option outlined in the red box in the figure below), and reset or re-power the device to make the modification effective.

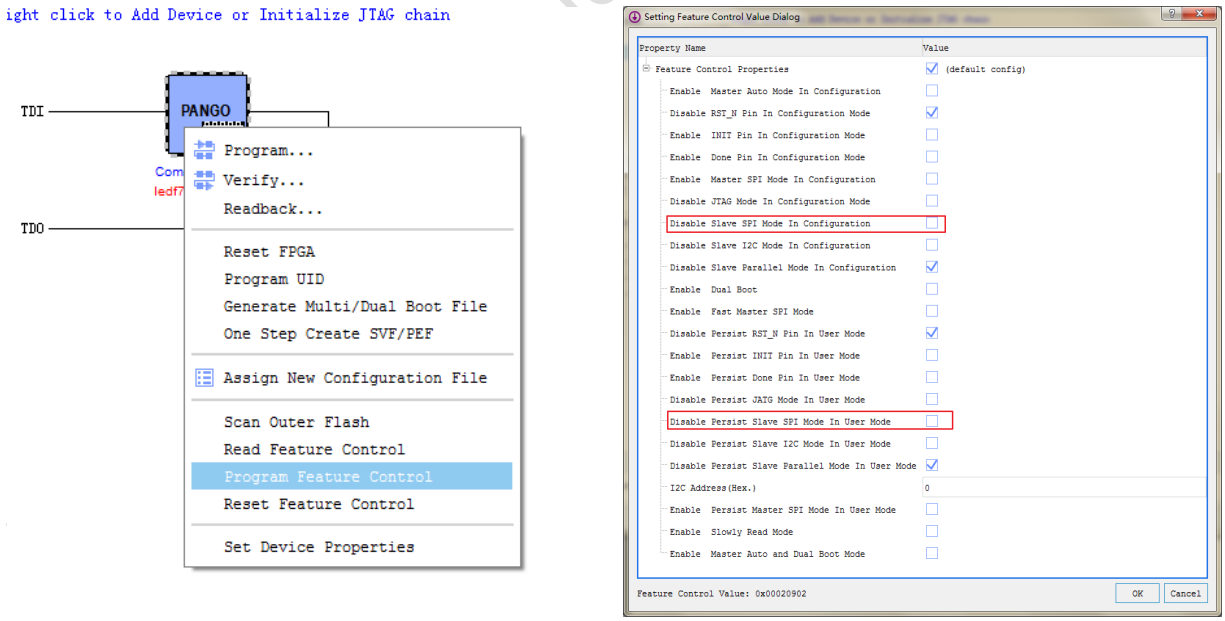

Figure 3-3 Direct Modification of Feature Control Bits

3.3 Project Notes

Table 3-3 Project Overview

Application Range	
Supported Devices	The routine is run by STM32F103 and can be ported to other microcontroller platforms with minor modifications.
Language Used	C
Provided Design Files	
Circuit Diagram of the Example Board	PDF file
C Language Source Program	.c and .h files
keil5 Project File	RAR compressed file
Development Tools Provided	
Design Tools	Keil5 uVision5 version

The project directory can be used for the following purposes:

Table 3-4 Project Directory Classification

DOC	Documents
Libraries	Standard library files for STM32
Listing	Files related to the listing output by Keil5
Output	Intermediate files generated by Keil5 compilation, etc.
Project	Location where Keil5 project files are placed
User	Path for storing program source code

3.4 Example Demo

The Demo program is a routine of data interaction based on serial port control, where users control the embedded system by inputting instructions through the serial port to configure the CPLDs. The use of it requires the installation of a serial port debugging assistant, which is configured as shown in the table below.

Table 3-5 Serial Port Configuration Parameters

Configuration Item	**Parameter**
Baud Rate	115200
Checksum Bits	None
Data Bits	8
Stop Bits	1
HEX Display	Off
HEX Send	Off

After the STM32 is reset, the serial port debugging assistant appears as shown in the figure. Enter the number for the corresponding function in the sending box and then send, then the microcontroller will execute the corresponding operation.

Figure 3-4 Serial Port Debugging

3.5 Function Description

3.5.1 Bottom Layer Functions

These lower-level functions are quite streamlined. They are usually used in combination and the running results can only be seen by calling serial port output or by other means. These functions correspond to the functionality in Appendix 3 of the "Compa Family CPLDs Configuration User Guide".

3.5.1.1 RDID

Table 3-6 Read Device ID Function

Function Name	unsigned int SSPI_RDID(void);
Function Description	Used to read the chip ID number, which is the return value and corresponds to the chip model. Please consult the supplier for the latest corresponding table.
Input Parameter Description	
Return Value	None

Usage	Refer to SSPI_FPGA_Init()
Actual Application Scenarios	Check whether the current chip model is consistent with the bitstream

Used to read back the IDCODE of CPLDs devices, starting with the lower bits. The timing diagram is shown below.

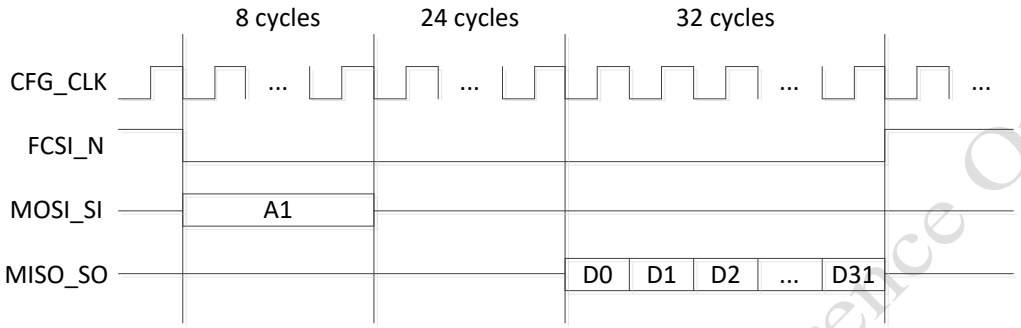

Figure 3-5 RDID Timing Diagram

3.5.1.2 RDUID

Table 3-7 Read UID

Function Name	unsigned int PG_SSPI_ReadOutUIDLow(void); unsigned int PG_SSPI_ReadOutUIDHigh(void)
Function Description	Used to read the chip UID. As the chip UID is 64 bits, two functions are used to read the 32 upper bits and the 32 lower bits respectively.
Input Parameter Description	
Return Value	None
Usage	Refer to PGC2KG_demo.h PGC2K_ReadAllUid(PGC2KDevice ***dev**)

Used to read back the UID of CPLDs devices, starting with the lower bits. The timing diagram is shown below.

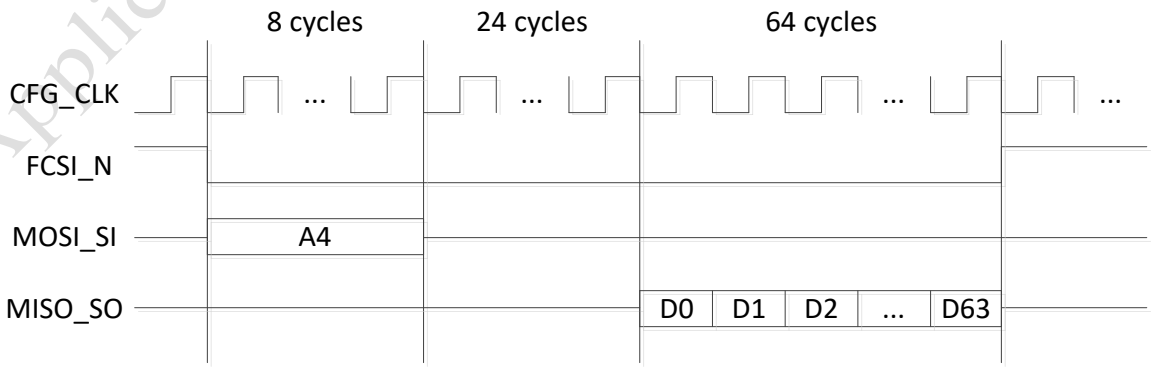

Figure 3-6 RDUID Timing Diagram

3.5.1.3 RDSR

Table 3-8 Read Status Register

Function Name	unsigned int SSPI_RDSR(void);
Function Description	It is recommended to be used for reading the current feature control bits after the wake-up of the embedded FLASH
Input Parameter Description	None
Return Value	The feature control bits of the current chip
Usage	SSPI_IsEFLASHBusy(void)
Description	After the following instructions are inputted: ERASE PROGRAM READ PROGRAM_LOCK ERASE_PAGE RDSR can be used directly for checking the status of the current embedded FLASH, without the need to perform WRDIS

Used to read back the status register, starting with the lower bits. The timing diagram is shown below.

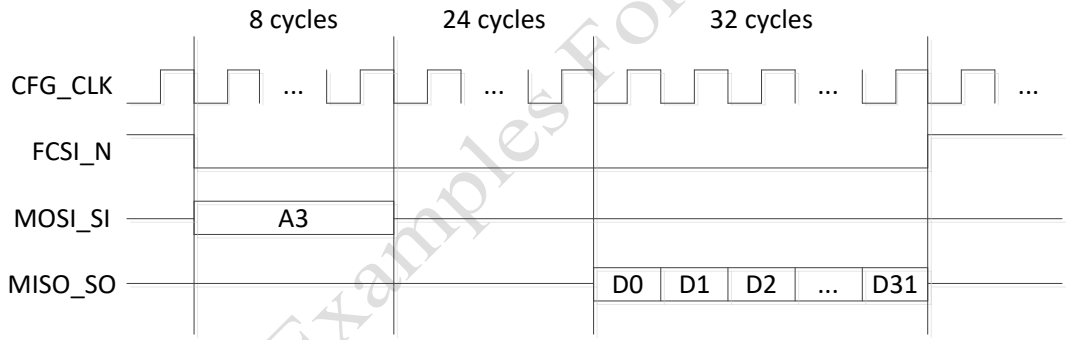

Figure 3-7 RDSR Timing Diagram

3.5.1.4 PROGRAM

Table 3-9 Write Data to Embedded FLASH

Function Name	SSPI_Write(unsigned int addr,unsigned char *dataoutput)
Function Description	Write one page of data to the EFLASH at the specified address
Input Parameter Description	addr: The address of the page to be written *dataoutput: Data to be written, which should be less than 256 bytes
Return Value	None
Usage	PG2K_SSPI_WriteAPage (unsigned int PageAddr,unsigned char *datain)

Used for programming the normal pages of embedded FLASH. The timing diagram is shown below.

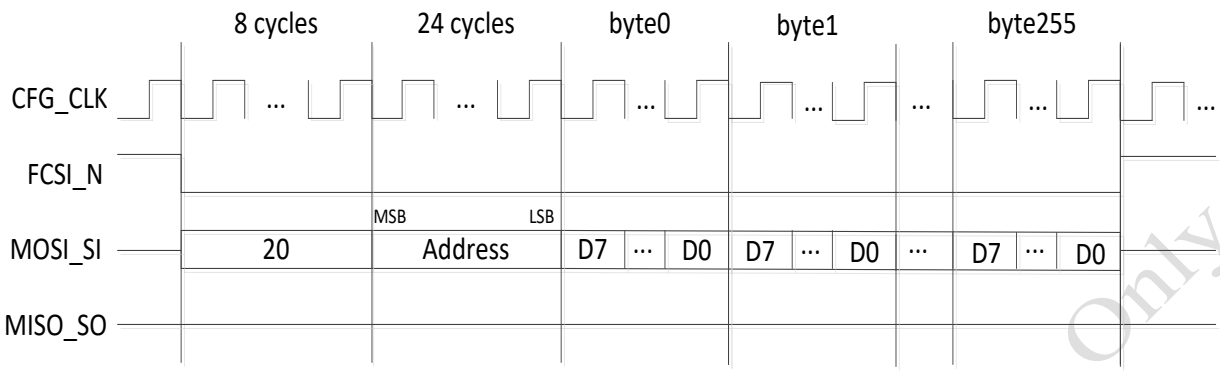

Figure 3-8 PROGRAM Timing Diagram

3.5.2 Application Functions

Application functions are direct applications of bottom layer functions, corresponding to the functions in Appendix 4 of the"Compa Family CPLDs Configuration User Guide".

3.5.2.1 Wake Up the Embedded FLASH

Table 3-10 Wake Up Embedded FLASH

Function Name	IntPG_SSPI_WakeUpEFLASH(void);
Function Description	Wake up the current embedded FLASH
Input Parameter Description	None
Return Value	None
Usage	Refer to main.c of the demo program
Description of calls	After identifying the chip that needs to be configured using the RDID function, enable the sleep mode

Waking up the embedded FLASH is a mandatory operation before performing actions such as writing UID, and reading and writing feature control bits, FLASH, or the Master Self Configuration bitstream. Therefore, before executing the above operations, users should first check the status register to confirm if the embedded FLASH is in a sleep state. If so, wake it up before proceeding.

After each power-up and completion of the Master Self Configuration loading of bitstream, the CPLDs will actively put the embedded FLASH into sleep mode to optimize power consumption. In other situations that require lower power consumption, it is also recommended to put the embedded FLASH that is not in use into sleep mode.

3.5.2.2 Put the Embedded FLASH into Sleep Mode

Table 3-11 Put the Embedded FLASH into Sleep Mode

Function Name	intPG_SSPI_SleepEFLASH(void);
Function Description	The function is used to put the embedded FLASH into sleep mode
Input Parameter Description	
Return Value	None
Usage	Refer to main.c of the demo program
Description of calls	After identifying the chip that needs to be configured using the RDID function, enable the sleep mode

3.5.2.3 Read UID

Table 3-12 Read UID

Function Name	unsigned intPG_SSPI_ReadOutUIDHigh(void); unsigned int PG_SSPI_ReadOutUIDLow(void);
Function Description	Read UID
Input Parameter Description	None
Return Value	The 32 upper bits and the 32 lower bits of the current chip Note: 8bit is an upper bit, thus PG_SSPI_ReadOutUIDHigh(void) returns the 8 upper bits of the UID
Usage	Refer to main.c of the demo program

3.5.2.4 Program the Master Self Configuration Bitstream

Table 3-13 Program the Master Self Configuration Bitstream

Function Name	Int PG_SSPI_Configuration(const char* cfg_file_name)
Function Description	Write the Master Self Configuration bitstream
Input Parameter Description	The parameter here is the filename used for supporting embedded systems with a file system. The call of it in the routine does not use a file system, it can be left blank. Call it as follows: PG_SSPI_Configuration("");
Return Value	Write successful 1 Write Fail 0
Usage	Refer to main.c of the demo program

3.5.2.5 Erase the Master Self Configuration Bitstream

Table 3-14 Erase the Master Self Configuration Bitstream

Function Name	PG2K_SSPI_EraseSomePage(unsigned int StartPage,unsigned int StopPage)
Function Description	Erase the Master Self Configuration bitstream

Input Parameter Description	StartPage and StopPage are the starting and ending pages respectively for the erasure.
Return Value	None
Usage	Refer to main.c of the demo program

Erasing the Master Self Configuration bitstream is essentially a special form of erasing embedded FLASH by page. Users can erase the data of specified pages by inputting the starting and ending pages of the bitstream into this function.

Chapter 4 Appendix

Table 4-1 SPI Interface Number for Compa Family CPLDs

Device	Package	Name	Pin Number
PGC1KG	MBG256	DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_21N/MISO_SO	T6
		DIFFI_B2_1P/FCSI_N	R12
		DIFFI_B2_21P/CFG_CLK	P6
	FBG256	DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_21N/MISO_SO	T6
		DIFFI_B2_1P/FCSI_N	R12
		DIFFI_B2_21P/CFG_CLK	P6
	LPG144	DIFFI_B2_1N/MOSI_SI	71
		DIFFI_B2_21N/MISO_SO	45
		DIFFI_B2_1P/FCSI_N	70
		DIFFI_B2_21P/CFG_CLK	44
	LPG100	DIFFI_B2_1N/MOSI_SI	49
		DIFFI_B2_21N/MISO_SO	32
		DIFFI_B2_1P/FCSI_N	48
		DIFFI_B2_21P/CFG_CLK	31
PGC1KL	UWG36	DIFFI_B2_0N/MOSI_SI	F1
		DIFFI_B2_10N/MISO_SO	E4
		DIFFI_B2_0P/FCSI_N	E1
		DIFFI_B2_10P/CFG_CLK	F5
PGC2KG	MBG256	DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_21N/MISO_SO	T6
		DIFFI_B2_1P/FCSI_N	R12
		DIFFI_B2_21P/CFG_CLK	P6
	FBG256	DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_21N/MISO_SO	T6
		DIFFI_B2_1P/FCSI_N	R12
		DIFFI_B2_21P/CFG_CLK	P6
	LPG144	DIFFI_B2_1N/MOSI_SI	71
		DIFFI_B2_21N/MISO_SO	45
		DIFFI_B2_1P/FCSI_N	70
		DIFFI_B2_21P/CFG_CLK	44
	LPG100	DIFFI_B2_1N/MOSI_SI	49
		DIFFI_B2_21N/MISO_SO	32

Device	Package	Name	Pin Number
		DIFFI_B2_1P/FCSI_N	48
		DIFFI_B2_21P/CFG_CLK	31
PGC2KL	UWG49	DIFFI_B2_1N/MOSI_SI	G1
		DIFFI_B2_21N/MISO_SO	F5
		DIFFI_B2_1P/FCSI_N	G2
		DIFFI_B2_21P/CFG_CLK	F6
	SSBG256	DIFFI_B2_1N/MOSI_SI	N5
		DIFFI_B2_21N/MISO_SO	R11
		DIFFI_B2_1P/FCSI_N	M5
		DIFFI_B2_21P/CFG_CLK	T12
PGC7KD	FBG484	DIFFI_B2_1N/MOSI_SI	W19
		DIFFI_B2_31N/MISO_SO	Y7
		DIFFI_B2_1P/FCSI_N	AA20
		DIFFI_B2_31P/CFG_CLK	AB6
	MBG400	DIFFI_B2_1N/MOSI_SI	W20
		DIFFI_B2_31N/MISO_SO	Y6
		DIFFI_B2_1P/FCSI_N	Y20
		DIFFI_B2_31P/CFG_CLK	W6
	MBG256	DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_31N/MISO_SO	T6
		DIFFI_B2_1P/FCSI_N	R12
		DIFFI_B2_31P/CFG_CLK	P6
	MBG332	DIFFI_B2_1N/MOSI_SI	V17
		DIFFI_B2_31N/MISO_SO	W7
		DIFFI_B2_1P/FCSI_N	W17
		DIFFI_B2_31P/CFG_CLK	V7
	LPG144	DIFFI_B2_1N/MOSI_SI	71
		DIFFI_B2_31N/MISO_SO	45
		DIFFI_B2_1P/FCSI_N	70
		DIFFI_B2_31P/CFG_CLK	44
PGC4KD	FBG256	DIFFI_B2_27N/MISO_SO	T6
		DIFFI_B2_27P/CFG_CLK	P6
		DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_1P/FCSI_N	R12
	MBG256	DIFFI_B2_27N/MISO_SO	T6
		DIFFI_B2_27P/CFG_CLK	P6
		DIFFI_B2_1N/MOSI_SI	P13
		DIFFI_B2_1P/FCSI_N	R12
	MBG324	DIFFI_B2_27N/MISO_SO	U6

Device	Package	Name	Pin Number
		DIFFI_B2_27P/CFG_CLK	T6
		DIFFI_B2_1N/MOSI_SI	V17
		DIFFI_B2_1P/FCSI_N	U16
	MBG332	DIFFI_B2_27N/MISO_SO	W7
		DIFFI_B2_27P/CFG_CLK	V7
		DIFFI_B2_1N/MOSI_SI	V17
		DIFFI_B2_1P/FCSI_N	W17
	LPG144	DIFFI_B2_1N/MOSI_SI	71
		DIFFI_B2_31N/MISO_SO	45
		DIFFI_B2_1P/FCSI_N	70
		DIFFI_B2_31P/CFG_CLK	44
PGC4KL	UWG81	DIFFI_B2_1N/MOSI_SI	H1
		DIFFI_B2_27N/MISO_SO	J7
		DIFFI_B2_1P/FCSI_N	G1
		DIFFI_B2_27P/CFG_CLK	H7
	SSBG256	DIFFI_B2_1N/MOSI_SI	N5
		DIFFI_B2_21N/MISO_SO	R11
		DIFFI_B2_1P/FCSI_N	M5
		DIFFI_B2_21P/CFG_CLK	T12
PGC4KLS	SBG81	DIFFI_B2_1N/MOSI_SI	H9
		DIFFI_B2_27N/MISO_SO	J4
		DIFFI_B2_1P/FCSI_N	J8
		DIFFI_B2_27P/CFG_CLK	J3
PGC10KD	MBG484	DIFFI_B2_1N/MOSI_SI	AA21
		DIFFI_B2_35N/MISO_SO	U9
		DIFFI_B2_1P/FCSI_N	AB21
		DIFFI_B2_35P/CFG_CLK	T9
	MBG400	DIFFI_B2_1N/MOSI_SI	W20
		DIFFI_B2_35N/MISO_SO	Y6
		DIFFI_B2_1P/FCSI_N	Y20
		DIFFI_B2_35P/CFG_CLK	W6

Disclaimer

Copyright Notice

Disclaimer

1. This document only provides information in stages and may be updated at any time based on the actual situation of the products without further notice. The Company assumes no legal responsibility for any direct or indirect losses caused by improper use of this document.
2. This document is provided "as is" without any warranties, including but not limited to warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranties mentioned in proposals, specifications, or samples. This document does not grant any explicit or implied intellectual property usage licence, whether by estoppel or otherwise.
3. The Company reserves the right to modify any documents related to its family products at any time without prior notice.
4. The information provided in this document is intended to assist users in resolving application issues; however, it does not guarantee flawlessness. The Company disclaims any responsibility for functional abnormalities or performance degradation that may occur due to the user's failure to follow the methods outlined in this document, and such issues cannot be acknowledged as product-related. Moreover, the solutions presented in this document are merely one of several possible options, and no guarantee is provided for their applicability to all application scenarios. In the event of any functional abnormalities or performance degradation resulting from user implementation of the instructions outlined in this document, the Company will not accept responsibility or acknowledge them as product-related issues.