

Compa Family CPLDs Device Slave I2C Interface Configuration and Programming Application Guide

(AN03014, V1.1)

(19.04.2022)

Shenzhen Pango Microsystems Co., Ltd.

All Rights Reserved. Any infringement will be subject to legal action.

Revisions History

Document Revisions

Version	Date of Release	Revisions
V1.1	19.04.2022	Initial release.

Application Examples For Reference Only

About this Manual

Terms and Abbreviations

Terms and Abbreviations	Meaning
JTAG	Joint Test Action Group
CCS	Configuration Control System
I2C	Inter-Integrated Circuit
UID	Unique ID

Related Documentation

The following documentation is related to this manual:

1. UG030004_Compact Family CPLDs Configuration User Guide

Table of Contents

Revisions History	1
About this Manual	2
Table of Contents	3
Tables	4
Figures	5
Chapter 1 Document Introduction	6
Chapter 2 Configuration Details	7
2.1 Slave I2C Interface Description	7
2.2 Feature Control Bits	8
2.3 Slave I2C Instruction Set	8
2.4 Slave I2C timing	9
2.4.1 7-Bit Addressing	10
2.4.2 10-bit addressing	11
2.4.3 PGC10KD PROGRAM	12
2.5 Configuration Flow	12
2.5.1 Configuration/Reconfiguration	12
2.5.2 Shutdown Reconfiguration	14
2.6 Programming Embedded Flash	16
2.6.1 Address	18
Chapter 3 Configuration Examples	20
3.1 Hardware Connection	20
3.2 Bitstream Setting	22
3.2.1 OSC Setting	22
3.2.2 Setting Feature Control Bits	22
3.3 Project Notes	24
3.4 Porting	25
3.4.1 Frequency Setting	25
3.4.2 GPIO Definition	26
3.4.3 Bitstream Storing	26
3.5 Example Demo	26
Disclaimer	28

Tables

Table 2-1 Slave I2C Pin Definition.....	7
Table 2-2 Configuring CPLDs-related Slave I2C Instruction Set.....	8
Table 2-3 Slave I2C Shutdown and Reconfiguration Command Flowchart.....	14
Table 2-4 Memory Sizes of Embedded Flash Ordinary Register	19
Table 2-5 Definitions of 1K/2K/4K/7K Address	19
Table 2-6 Definition of 10K Address.....	19
Table 3-1 List of Supported Functions	20
Table 3-2 List of Supported Devices	20
Table 3-3 Connection Between I2C Pins	21
Table 3-4 FLASH Chip I2C Pins	21
Table 3-5 Project Overview	24
Table 3-6 Project Directory Classification.....	24
Table 3-7 Serial Port Configuration Parameters	26

Figures

Figure 2-1 Example of Slave I2C Interface Application	7
Figure 2-2 Bit Write Timing	10
Figure 2-3 Bit Read Timing	10
Figure 2-4 Bit Write Timing	11
Figure 2-5 Bit Read Timing	11
Figure 2-6 Bit PROGRAM Timing	12
Figure 2-7 Bit PROGRAM timing	12
Figure 2-8 Configuration/Reconfiguration Flowchart	13
Figure 2-9 Shutdown Reconfiguration Flowchart	15
Figure 2-10 Programming Embedded Flash 1	16
Figure 2-11 Programming Embedded Flash 2	17
Figure 2-12 Programming Embedded Flash 3	18
Figure 3-1 Example Hardware Circuit Diagram	21
Figure 3-2 OSC Setting	22
Figure 3-3 Setting Feature Control Bits in Bitstream Settings	23
Figure 3-4 Direct Modification of Feature Control Bits	24
Figure 3-5 Example Code	26
Figure 3-6 Serial Port Debugging	27

Chapter 1 Document Introduction

This document focuses on the application of the Pango Microsystems CPLDs slave I2C configuration mode, the software and hardware environment required for slave I2C 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.

Application Examples For Reference Only

Chapter 2 Configuration Details

2.1 Slave I2C Interface Description

With the slave I2C interface, users can configure/reconfigure CRAM, program embedded FLASH, read/write feature control bits, etc.

The slave I2C 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:

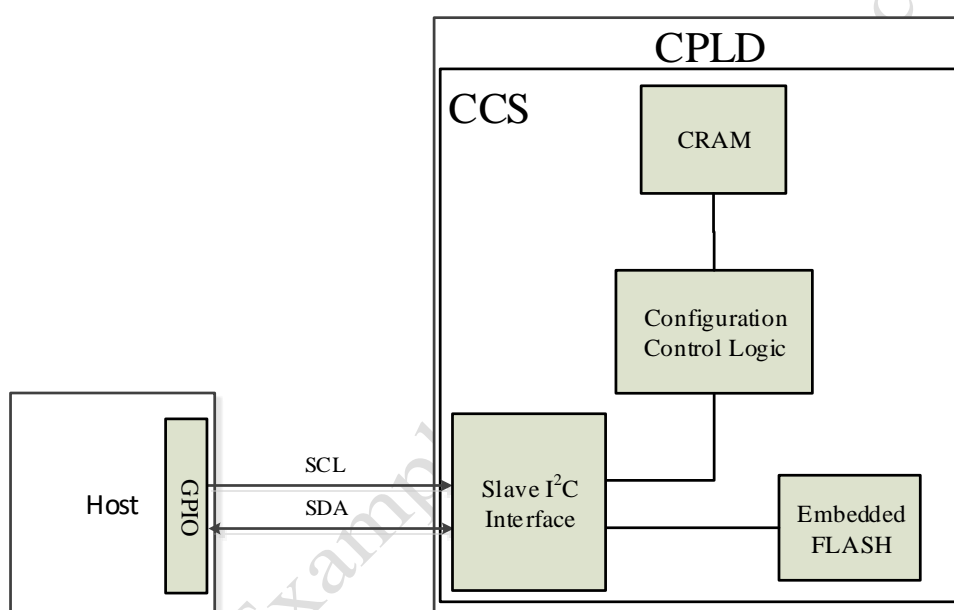


Figure 2-1 Example of Slave I2C Interface Application

The slave I2C pins are defined as shown in the following table:

Table 2-1 Slave I2C Pin Definition

Pin Name	I/O	Description
SCL	I	Serial Clock. For the maximum supported frequency, please refer to the "DS03001_Compa Family CPLDs Device Datasheet".
SDA	IO	Serial Data

The slave I2C interfaces of different devices and packages of Compa family CPLDs are distributed differently on the pins. For details, please refer to the corresponding device package manual.

Whether the slave I2C interface of the Compa family CPLDs serves as a configuration interface during the configuration or is retained as a configuration interface after the configuration is completed is controlled by the feature control bits of the chip. See [Feature Control Bits](#).

Additionally, as the CPLDs uses an internal OSC for decoding during the I2C interface loading process, the OSC needs to remain active during the operation of the interface. If the I2C interface is still required in user mode, the bitstream settings must be modified to keep the OSC active, see [OSC Setting](#).

2.2 Feature Control Bits

The feature control bits of the Compa family CPLDs are stored in the embedded FLASH (Non-Volatile Memory) and are read by the chip CCS each time the chip is initialized. The CCS selects the function of the corresponding configurable multiplexed 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_Comp Family CPLDs Configuration User Guide".

The enablement of the I2C interface is controlled by the feature control bits `cfg_si2c_en_n` (CTL[7]) and `persist_si2c_n` (CTL[16]). When `cfg_si2c_en_n` (CTL[7]) is 0, the slave I2C interface is enabled in configuration mode; when `persist_si2c_n` (CTL[16]) is 0, the slave I2C interface is enabled in user mode.

The chip feature control bits can be modified in 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. The PDS download tool individually programs the device feature control bits.
3. If the JTAG interface is enabled, use the JTAG interface instruction PROGRAM_CTL.
4. If the slave SPI interface is enabled, use the slave SPI interface instruction PROGRAM_CTL.
5. If the slave I2C interface is enabled, use the slave I2C interface instruction PROGRAM_CTL.
6. Use the internal slave APB interface instruction PROGRAM_CTL.

2.3 Slave I2C Instruction Set

Table 2-2 Configuring CPLDs-related Slave I2C Instruction Set

Instruction	Description	Command	Operands	Write Data	Read Data
NOP	No Operation	FF	N/A	N/A	N/A
RDID	Read Identification	A1	00 00 00	N/A	YY*4
RDUSER	Read Usercode	A2	00 00 00	N/A	YY*4
RDSR	Read Status Register	A3	00 00 00	N/A	YY*4

Instruction	Description	Command	Operands	Write Data	Read Data
RDUID	Read Unique Identification	A4	00 00 00	N/A	YY*8
CFG	Config Bitstream	50	00 00 00	XX*n	N/A
WREN	Write Enable	51	N/A	N/A	N/A
WRDIS	Write Disable	52	N/A	N/A	N/A
RESET	Reset CPLDs	60	N/A	N/A	N/A
ERASE	Erase Bulk	10	N/A	N/A	N/A
ERASE_PAGE	Erase Page	11	AAAAAA	N/A	N/A
ERASE_CTL	Erase Function Control Page	12	N/A	N/A	N/A
PROGRAM	Program Page	20	AAAAAA	XX*256	N/A
PROGRAM_CTL	Program Function Control Page	22	00 00 00	XX*4	N/A
READ	Read	30	AAAAAA 00	N/A	YY*4n ⁽²⁾
READ_CTL	Read Function Control Page	31	AAAAAA 00	N/A	YY*4
PROGRAM_LOCK	Lock Embedded Flash	40	AAAAAA	XX*2	N/A
EFlash_SLEEP	Embedded Flash Sleep	70	N/A	N/A	N/A
EFlash_WAKEUP	Embedded Flash Wake Up	71	N/A	N/A	N/A

Notes:

1. In the table, "AA" represents one byte (8bit) of the address; "XX" represents one byte of input data; "YY" represents one byte of output data; "N/A" indicates that the item does not exist.
2. "4n" indicates that it requires reading in the unit of 4bytes (i.e., 32bit), for example, reading back 4bytes, 8bytes, etc., at a time, typically directly reading back 256bytes or one page of data.

2.4 Slave I2C timing

The timing of all slave I2C interface execution flows consists of slave addresses, commands, operands, write data, and read data. A single instruction execution includes at least slave addresses and commands, while different commands contain operands, write data, read data, etc., as detailed in [Slave I2C Instruction Set](#). The following will present the timing diagrams for 7-bit and 10-bit addressing.

Note that as the Eflash programming process (i.e., PROGRAM instruction) of the PGC10KD differs from that of other devices and instructions, the following first introduces the commonly used processes and then that of the PGC10KD.

2.4.1 7-Bit Addressing

Typical read/write timing is illustrated in the figure below:

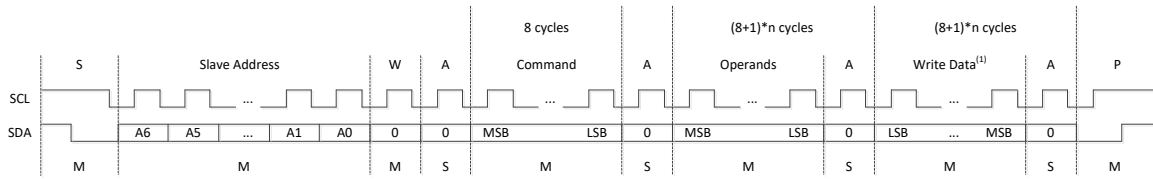


Figure 2-2 Bit Write Timing

Notes: Except for bitstream data, data is all written starting with the lower bits. For example, for feature control bits CTL[31:0], it is sent in the order CTL[0:7], CTL[8:15] ...; For bitstream data that is a binary stream, it is sent from beginning to end. Taking the synchronized word contained in the bitstream (0x01332D94) as an example, the byte is sent in the order 0x01, 0x33..., with corresponding order within the bytes as 00000001, 00110011....

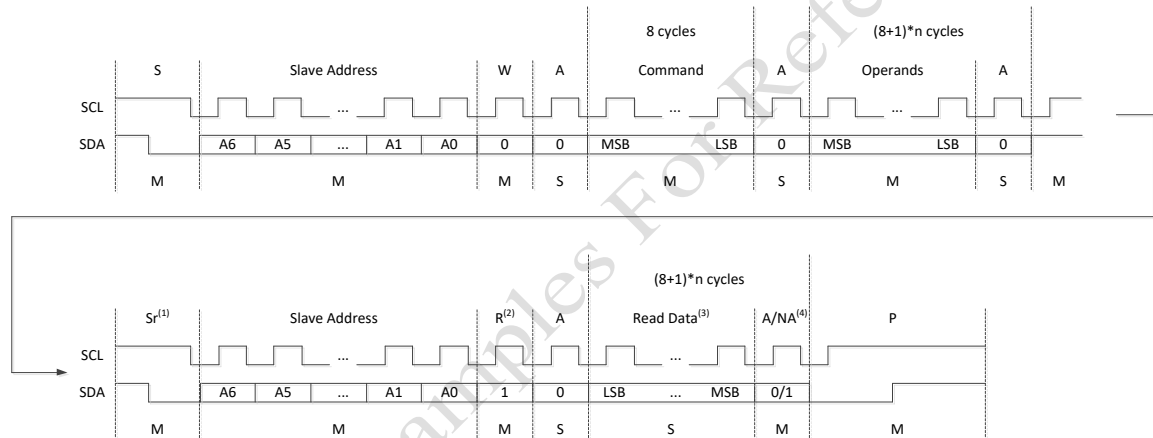


Figure 2-3 Bit Read Timing

Notes:

1. For read operations, a stop condition (i.e., SDA switches from low to high while SCL is high) must not be generated after writing the read-related instructions, otherwise, the instruction flow is considered finished and the device will not output data.
2. Only after the read-related instruction is written and a repeated start is initiated can the read operation be performed; if the read operation is performed directly after an initial start, the device will not respond.
3. Except for bitstream data, data is all read starting with the lower bits. For example, for feature control bits CTL[31:0], it reads back in the order CTL[0:7], CTL[8:15] ...; For bitstream data that is a binary stream, it reads back from beginning to end. Taking the synchronized word contained in the bitstream (0x01332D94) as an example, the byte is read back in the order 0x01, 0x33..., with corresponding order within the bytes as 00000001, 00110011....
4. At the last byte of a read, the master ends the data transfer by not responding to the slave and then generates a stop condition.

2.4.2 10-bit addressing

Typical read/write timing is illustrated in the figure below:

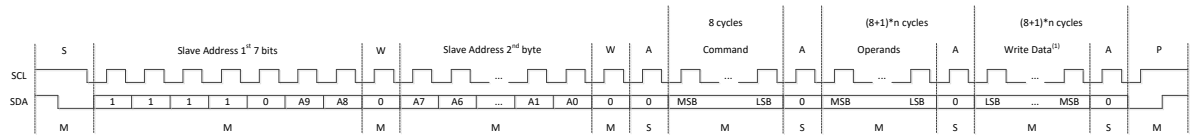


Figure 2-4 Bit Write Timing

Notes: Except for bitstream data, data is all written starting with the lower bits. For example, for feature control bits CTL[31:0], it is sent in the order CTL[0:7], CTL[8:15] ...; For bitstream data that is a binary stream, it is sent from beginning to end. Taking the synchronized word contained in the bitstream (0x01332D94) as an example, the byte is sent in the order 0x01, 0x33..., with corresponding order within the bytes as 00000001, 00000011....

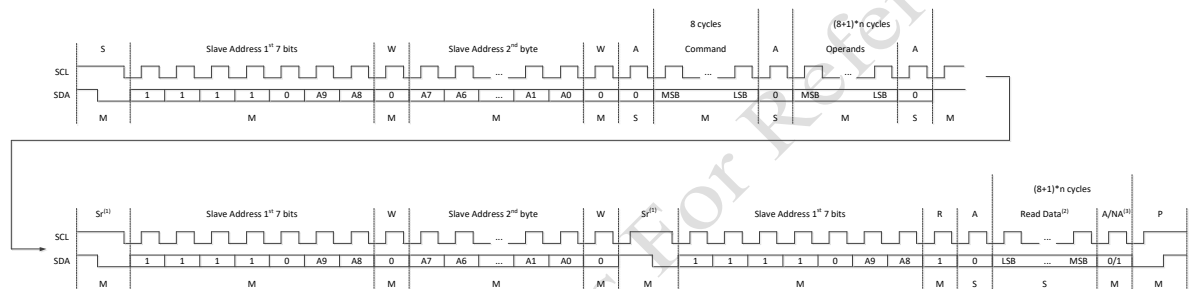


Figure 2-5 Bit Read Timing

Notes:

- 1 For read operations, a stop condition (i.e., SDA switches from low to high while SCL is high) must not be generated after writing the read-related instructions, otherwise, the instruction flow is considered finished and the device will not output data.
- 2 Only after the read-related instruction is written and a repeated start is initiated can the read operation be performed; if the read operation is performed directly after an initial start, the device will not respond.
- 3 Except for bitstream data, data is all read starting with the lower bits. For example, for feature control bits CTL[31:0], it reads back in the order CTL[0:7], CTL[8:15] ...; For bitstream data that is a binary stream, it reads back from beginning to end. Taking the synchronized word contained in the bitstream (0x01332D94) as an example, the byte is read back in the order 0x01, 0x33..., with corresponding order within the bytes as 00000001, 00110011....
- 4 At the last byte of readback, the master ends the data transfer by not responding to the slave and then generates a stop condition.

2.4.3 PGC10KD PROGRAM

The PGC10KD PROGRAM process requires a 10ms delay, and the timing diagram is as follows:

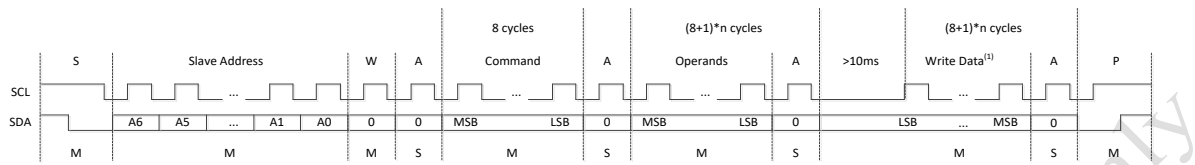


Figure 2-6 Bit PROGRAM Timing

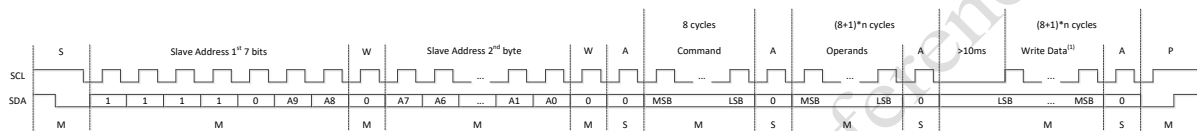


Figure 2-7 Bit PROGRAM timing

2.5 Configuration Flow

2.5.1 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 after the CPLDs chip enters user mode. 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 slave I2C interface is released for user use in user mode (with feature control bit persist_si2c_n set to 1), check the values of INIT_FLAG_N and CFG_DONE to conclude the operation. If the slave I2C 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 reset (by setting RSTN to 0) at any time after power-up. After reset, users can proceed to step 2 to reconfigure the CPLDs device.

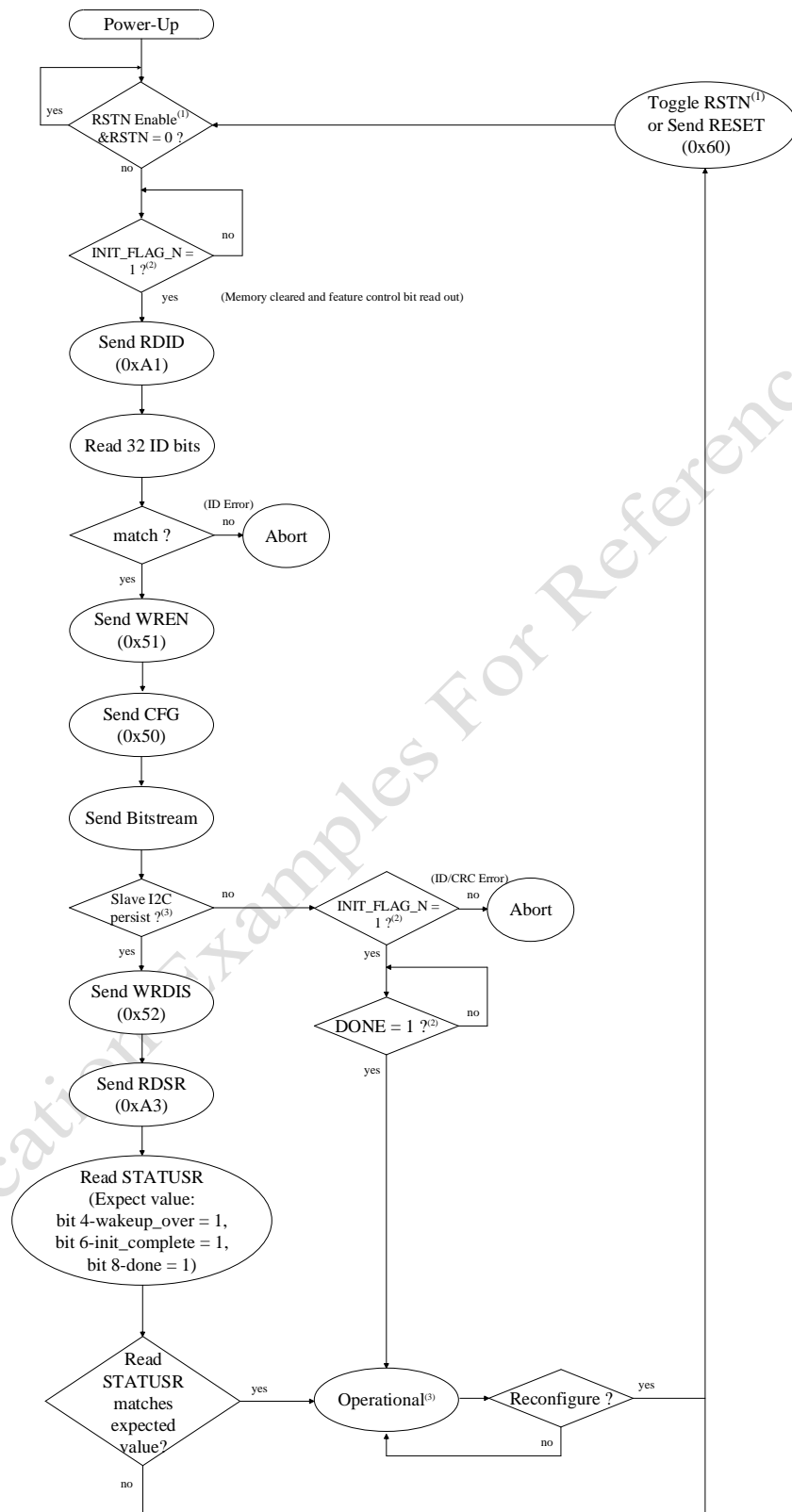


Figure 2-8 Configuration/Reconfiguration Flowchart

Notes:

1. If RSTN is enabled, the device will not be initialized until the RSTN is 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 RSTN turns high, as these two pins may not accurately indicate the status when RSTN is low.
3. "Slave I2C persist" means retaining the I2C interface as a configuration interface. To do so, the feature control bits and OSC must be set, see [Bitstream Setting](#).

2.5.2 Shutdown Reconfiguration

Shutdown reconfiguration refers to the process of shutting down and reconfiguring 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 Slave I2C Shutdown and Reconfiguration Command Flowchart

Flow	Data
Write a CFG instruction to load:	0x50
100-pad words	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 0x00000300
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

Note: For register descriptions mentioned in the table, refer to "UG030004_Compa Family CPLDs Configuration User Guide".

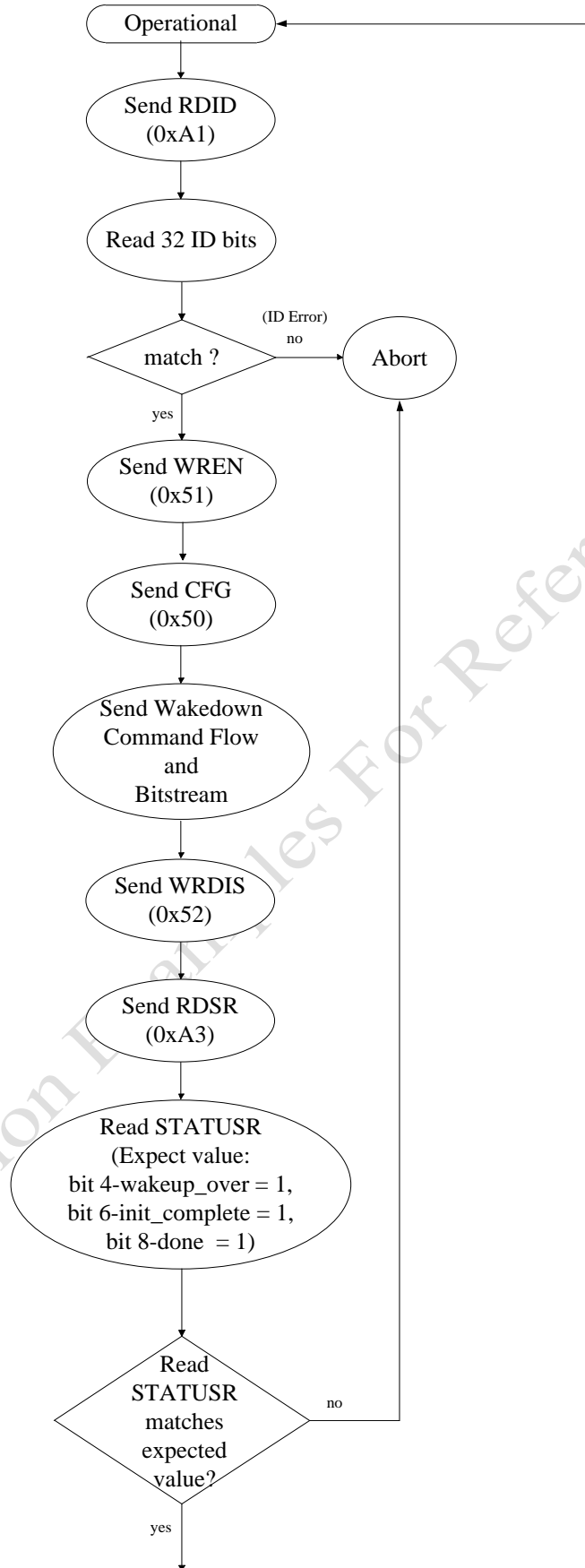


Figure 2-9 Shutdown Reconfiguration Flowchart

2.6 Programming Embedded Flash

Operations on the embedded FLASH can be performed directly through the slave I²C interface.

The complete process for programming the embedded FLASH via slave I2C is as follows:

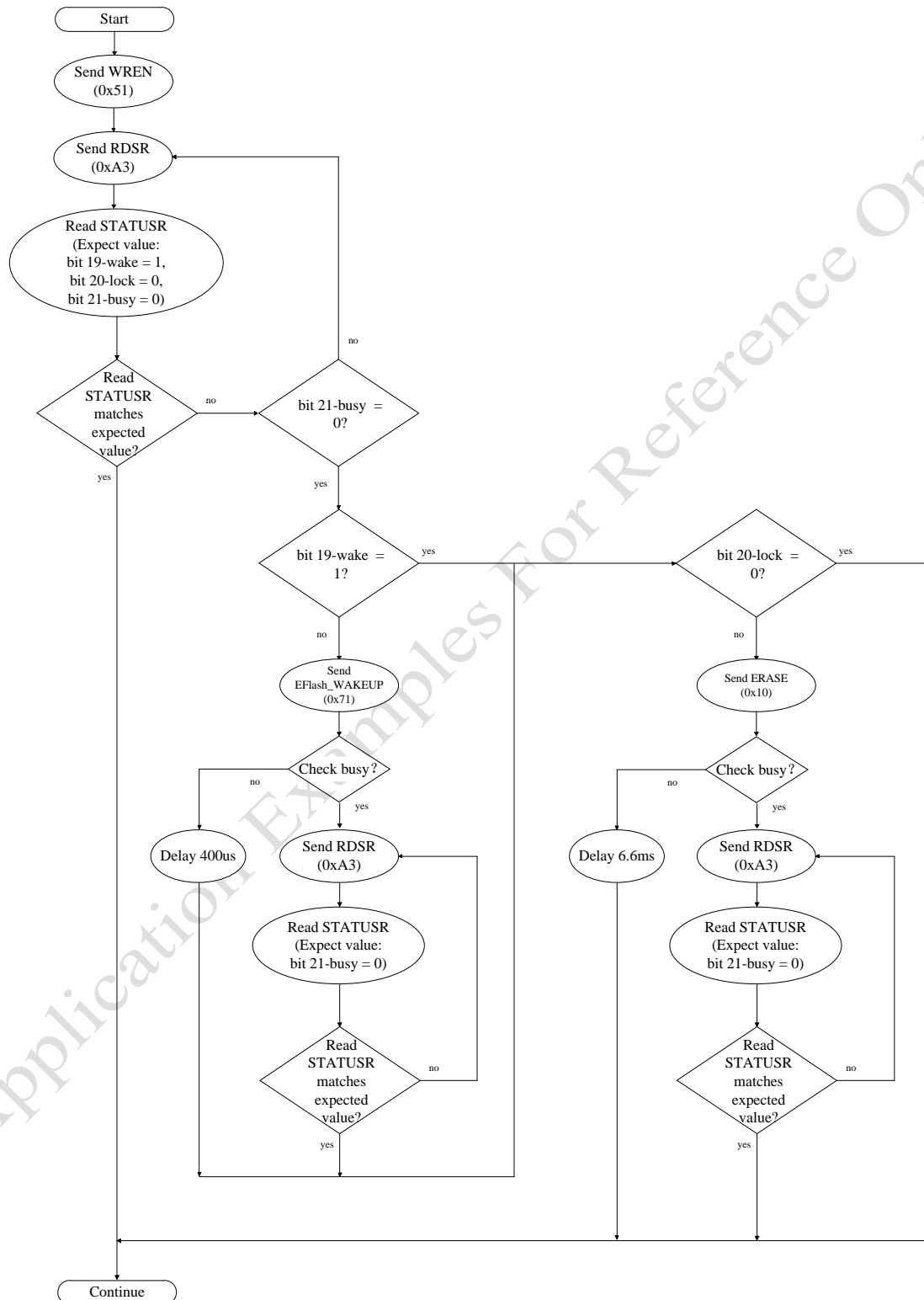


Figure 2-10 Programming Embedded Flash 1

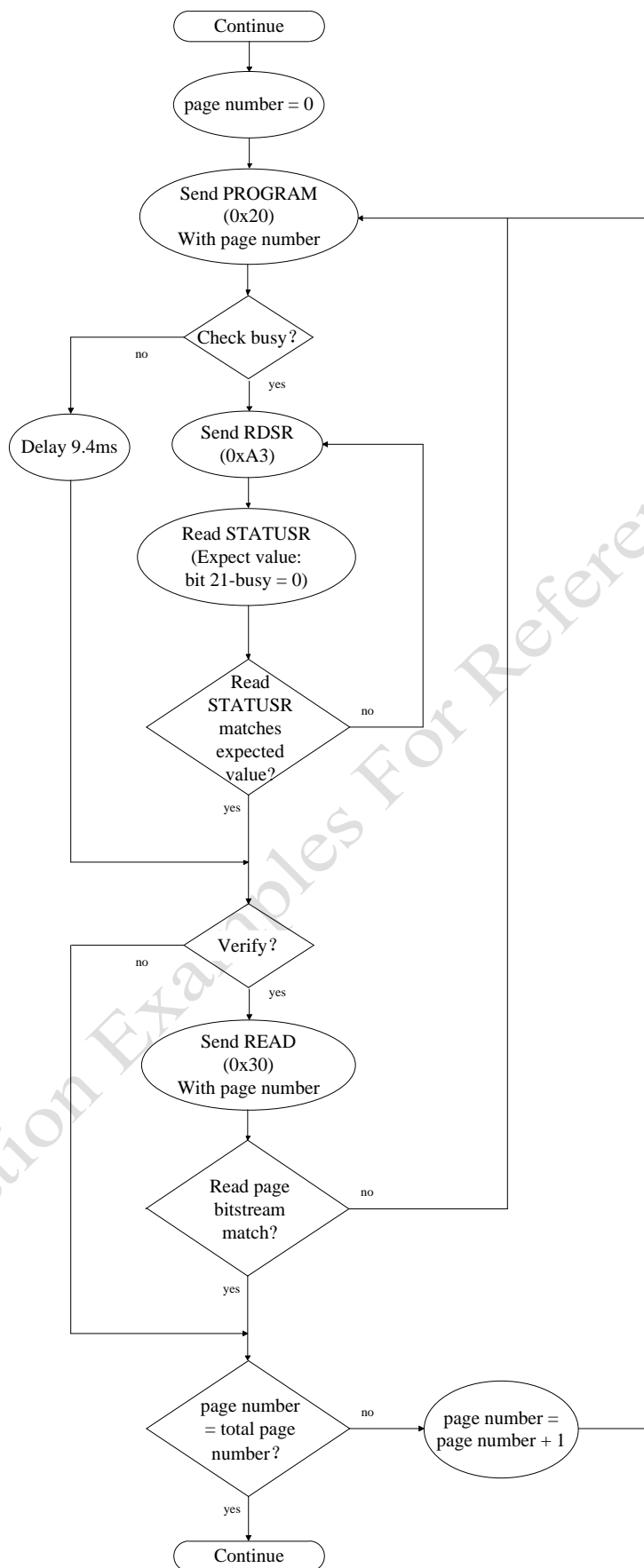


Figure 2-11 Programming Embedded Flash 2

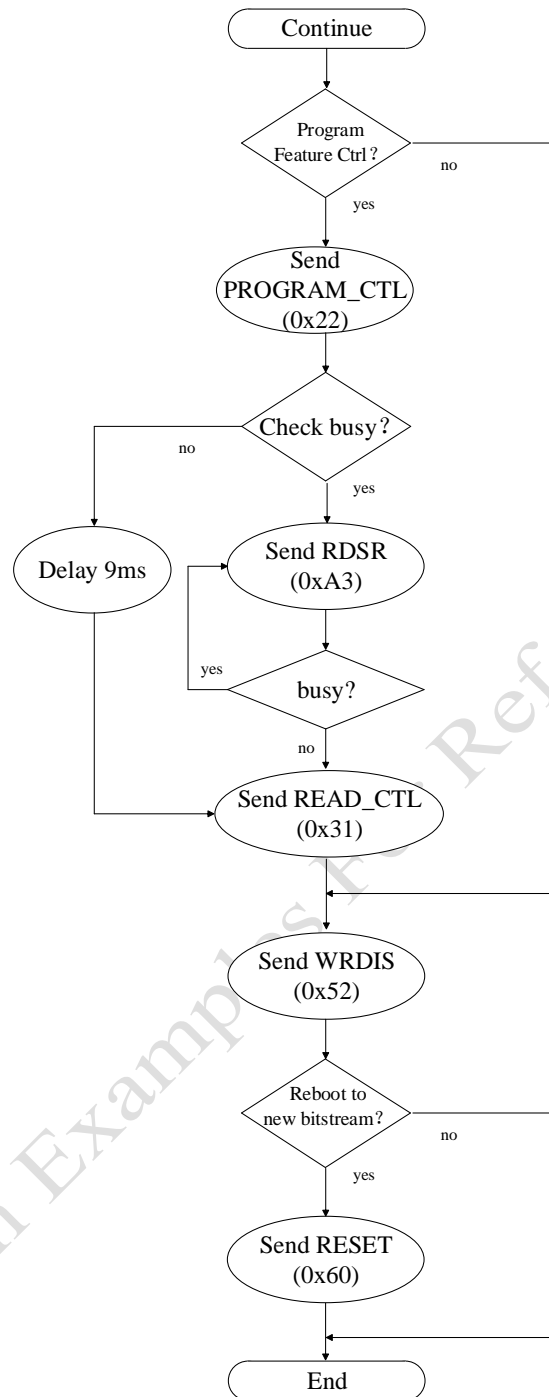


Figure 2-12 Programming Embedded Flash 3

2.6.1 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
[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 For 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 4'b0000
[19:18]	Ba[1:0]	Heap Address
[17:8]	Ra[9:0]	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 I2C interface configuration, using the example of controlling the read/write of embedded FLASH and CRAM loading via STM32F103.

This example (CFG_DEMO_I2C) is based on the Wildfire STM32F103MINI hardware platform, with Keil5 as the software development environment.

The supported functions and devices in the example are shown in the following table:

Table 3-1 List of Supported Functions

S. No.	Supported Functions
1	Read device information, including IDCode, UID, status register, and feature control bits
2	Program UID
3	Configure CRAM
4	Program EFlash

Table 3-2 List of Supported Devices

S. No.	Supported Devices
1	PGC1KG
2	PGC1KL
3	PGC2KG
4	PGC2KL
5	PGC4KD
6	PGC4KL
7	PGC7KD
8	PGC10KD

3.1 Hardware Connection

The hardware connection consists of two parts: the I2C 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).

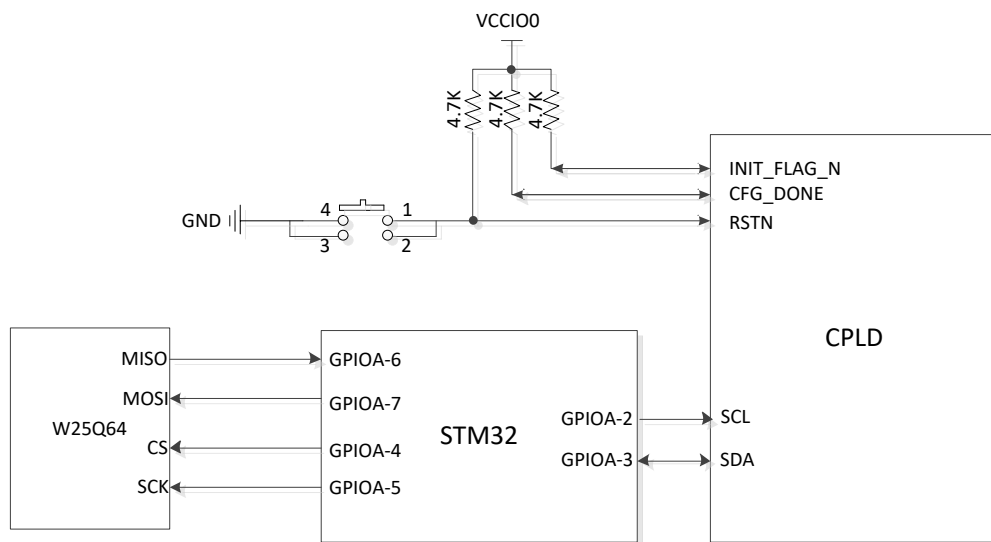


Figure 3-1 Example Hardware Circuit Diagram

During the configuration process, it is necessary to maintain the connection between the microcontroller's GPIO Pin for the simulated I2C interface and the CPLDs' I2C interface.

Table 3-3 Connection Between I2C Pins

STM32 Pins	CPLDs Pins
GPIOA-2	SCL
GPIOA-3	SDA

In the example program, the bitstream of CPLDs 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 3-4 FLASH Chip I2C Pins

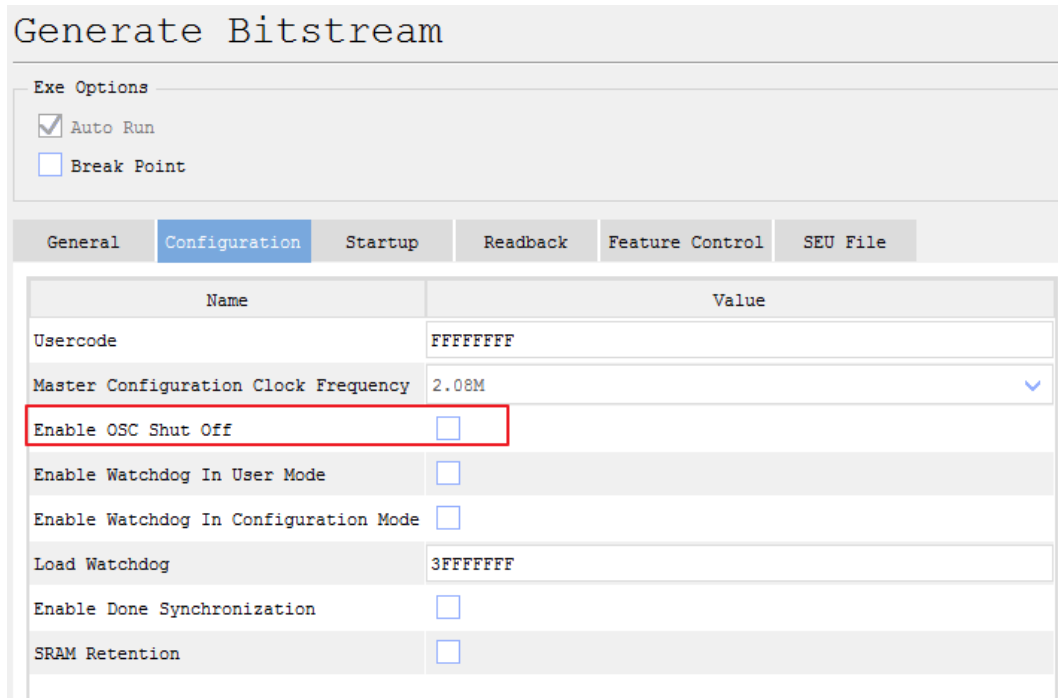
FLASH Chip Pins	STM32 Pins
CS	GPIOA-4
SCK	GPIOA-5
MISO	GPIOA-6
MOSI	GPIOA-7

When downloading the demonstration bitstream, it is necessary to connect the FLASH pins from [Figure 2-5](#) to the outside, writing the CPLDs bitstream 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 Bitstream Setting

3.2.1 OSC Setting

If the I2C interface is needed in user mode, the OSC must be kept on, which means deselecting the option shown below (automatic shutdown of OSC):



The screenshot shows the 'Generate Bitstream' application window. The 'Configuration' tab is selected. Under 'Exe Options', 'Auto Run' is checked and 'Break Point' is unchecked. The main configuration table lists various settings:

Name	Value
Usercode	FFFFFFFF
Master Configuration Clock Frequency	2.08M
Enable OSC Shut Off	<input type="checkbox"/>
Enable Watchdog In User Mode	<input type="checkbox"/>
Enable Watchdog In Configuration Mode	<input type="checkbox"/>
Load Watchdog	3FFFFFFF
Enable Done Synchronization	<input type="checkbox"/>
SRAM Retention	<input type="checkbox"/>

Figure 3-2 OSC Setting

3.2.2 Setting Feature Control Bits

As described in section 2.2, the feature control bits determine whether the slave I2C is enabled. To use the slave I2C, ensure the feature control bits are set to enable the slave I2C interface. The chip is shipped with the slave I2C 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 SPI interface. The following shows how the slave I2C is enabled via the JTAG interface by the PDS download tool.

When configuring CRAM, the feature control bits are written by default. Enable the slave I2C in the bitstream settings (select Dedicated IO) and set the I2C address. Then when users use the download tool Fabric Configuration to download the bitstream, the Slave I2C interface will be enabled.

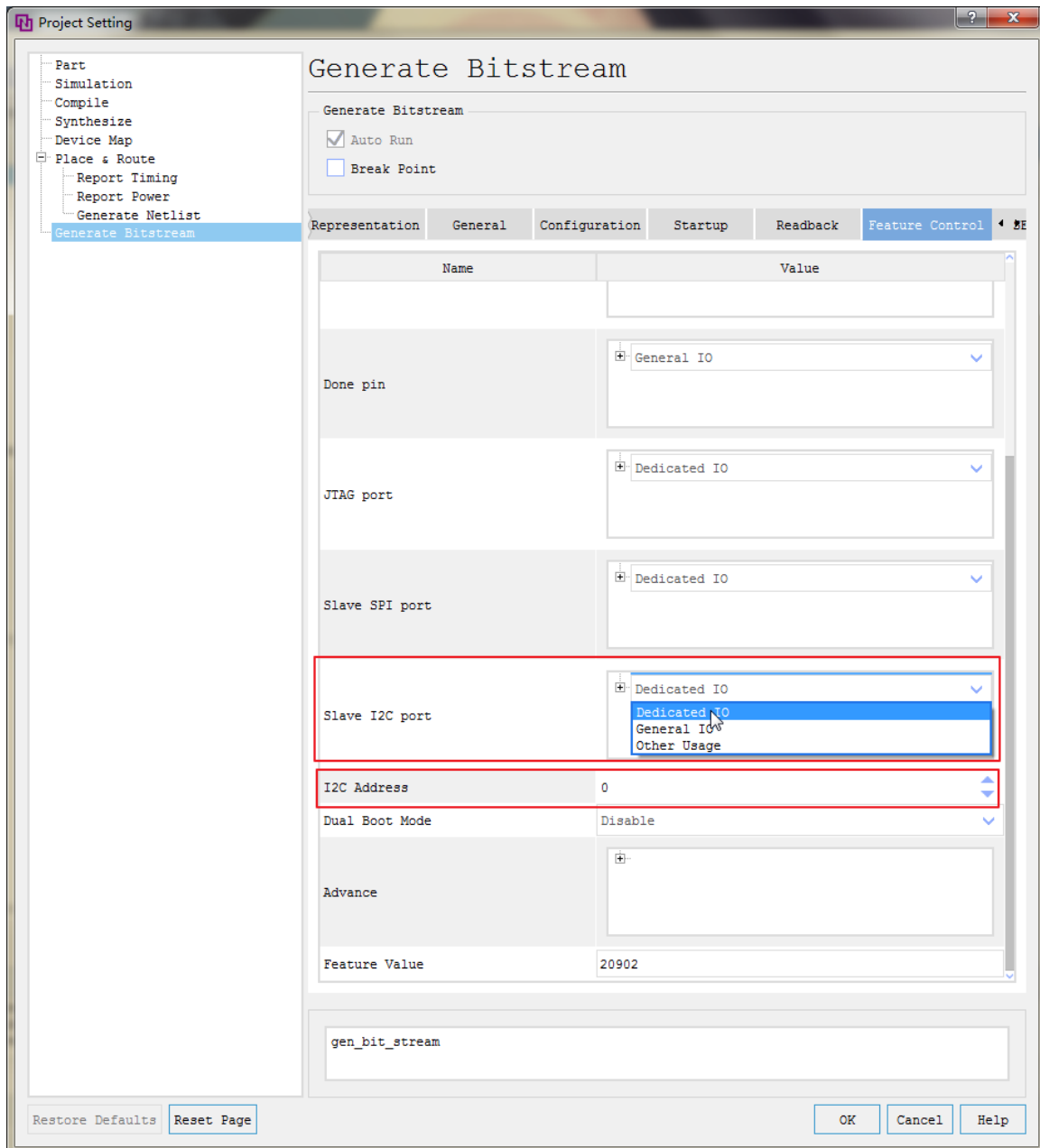


Figure 3-3 Setting Feature Control Bits in Bitstream Settings

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.

Right click to Add Device or Initialize JTAG chain

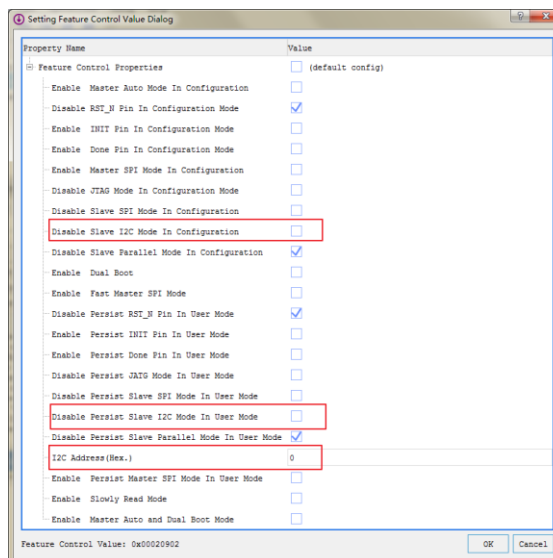
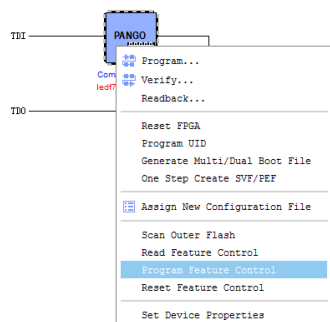


Figure 3-4 Direct Modification of Feature Control Bits

3.3 Project Notes

Table 3-5 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 3-6 Project Directory Classification

Doc	Documents
Project	Location where Keil5 project files are placed
Libraries	Platform-related codes, including the library files and module drivers such as spi, uart, etc. required by the example platform
Source	<p>Program key source codes</p> <p>It includes the following parts:</p> <p>pgc_def.h: Definitions of parameters related to the PGC family devices</p> <p>pango_i2c.c/h: Implementation of the I2C interface protocol</p> <p>pgc_i2c_basic.c/h: Implementation of basic operation for the PGC family I2C interface</p> <p>pgc_i2c_demo.c: Implementation of practical example functions for the PGC family I2C Interface</p> <p>hardware.c: Implementation of platform-related function functions</p>

3.4 Porting

This example project can be ported to other platforms with simple modifications. Porting mainly refers to modifying the "Source" or the core code in the project directory for use on other platforms. The following explains the main concerns during the porting process.

During the porting process, first add the platform-independent files (pgc_def.h, pango_i2c.c/h, pgc_i2c_basic.c/h) from the "Source" directory to the target project; then call the functions by referring to pgc_i2c_demo.c file, or just add some of the functions to the target project; finally, modify the hardware.c file with the following main points in mind:

3.4.1 Frequency Setting

The frequency of CPUs across various platforms differs significantly, affecting some functions of the example code, mainly the IO toggling rate of the I2C interface, the delay in certain processes, etc..

Considering the impact on the I2C interface rate, the PGC family JTAG interface supports a rate up to 400KHz. It is necessary to ensure that the interface rate does not exceed 400KHz, which can be determined by observing the I2C output. If it exceeds 400KHz, a delay must be added to the IO control interface. For example, the IO control function:

```
void pango_I2CSetSCLHigh ()
{
    GPIO_SetBits(pango_GPIO_PORT_I2C, pango_I2C_SCL_PIN);
}
```

The toggling rate can be reduced by setting the IO to the same value in a loop:

```
void pango_I2CSetSCLHigh ()
{
    unsigned int IdleTime = 5; // Loop 5 times for delay
    unsigned int i = 0;
    for(i = 0; i < IdleTime; i++)
        GPIO_SetBits(pango_GPIO_PORT_I2C, pango_I2C_SCL_PIN);
}
```

Considering delays within the process, the I2C frequency should not exceed 400KHz after the adjustment in the last step. Some processes in the example project require delays, which will be adjusted based on the global variable g_user_clk. This parameter can simply be modified based on the actual frequency.

3.4.2 GPIO Definition

To control IO toggling across different platforms, it generally requires defining the IO location and initializing the IO, as shown in the hardware.c example:

```

29  /***** I2C connected pin define *****/
30  #define pango_GPIO_PORT_I2C      GPIOA
31  #define pango_RCC_I2C_PORT      RCC_APB2Periph_GPIOA
32  #define pango_I2C_SCL_PIN      GPIO_Pin_2
33  #define pango_I2C_SDA_PIN      GPIO_Pin_3

36  void pango_I2CIOInit()
37  {
38      GPIO_InitTypeDef GPIO_InitStructure;
39
40      RCC_APB2PeriphClockCmd(pango_RCC_I2C_PORT, ENABLE); /*Enable GPIO clock*/
41
42      GPIO_InitStructure.GPIO_Pin = pango_I2C_SCL_PIN | pango_I2C_SDA_PIN;
43      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
44      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD; /*Open-drain output*/
45      GPIO_Init(pango_GPIO_PORT_I2C, &GPIO_InitStructure);
46  }

```

Figure 3-5 Example Code

Users can modify this part based on the specific platform and then execute the read/ write interface functions (pango_I2CSetSCLHigh, pango_I2CReadSDA ...).

3.4.3 Bitstream Storing

The example project stores the bitstream in an external SPI Flash and utilizes the driver implemented in the bsp_spi.c file to read the bitstream. During the porting process, just modify the function pango_BufferRead based on the bitstream acquisition method supported by the actual platform. As shown in the definition, the function needs to support two key inputs: ReadAddr, which specifies the read address; and NumByteToRead, which is the number of bytes to read at one time.

3.5 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 3-7 Serial Port Configuration Parameters

Configuration Item	Parameter
Baud Rate	115200
Checksum Bits	None
Data Bits	8
Stop Bits	1

Configuration Item	Parameter
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.

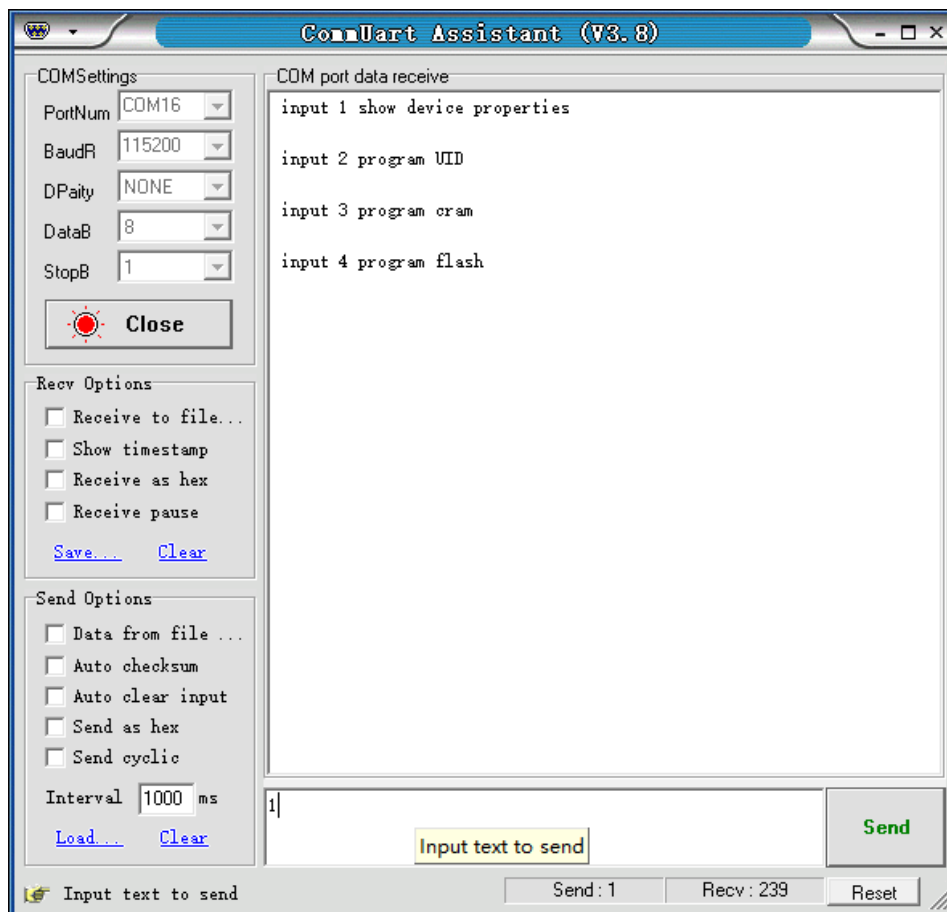


Figure 3-6 Serial Port Debugging

Disclaimer

Copyright Notice

This document is copyrighted by Shenzhen Pango Microsystems Co., Ltd., and all rights are reserved. Without prior written approval, no company or individual may disclose, reproduce, or otherwise make available any part of this document to any third party. Non-compliance will result in the Company initiating legal proceedings.

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.