

GProgrammer User Manual

Version: 2.7

Release Date: 2023-02-03

Shenzhen Goodix Technology Co., Ltd.

Copyright © 2023 Shenzhen Goodix Technology Co., Ltd. All rights reserved.

Any excerption, backup, modification, translation, transmission or commercial use of this document or any portion of this document, in any form or by any means, without the prior written consent of Shenzhen Goodix Technology Co., Ltd. is prohibited.

Trademarks and Permissions

GODIX and other Goodix trademarks are trademarks of Shenzhen Goodix Technology Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Disclaimer

Information contained in this document is intended for your convenience only and is subject to change without prior notice. It is your responsibility to ensure its application complies with technical specifications.

Shenzhen Goodix Technology Co., Ltd. (hereafter referred to as "Goodix") makes no representation or guarantee for this information, express or implied, oral or written, statutory or otherwise, including but not limited to representation or guarantee for its application, quality, performance, merchantability or fitness for a particular purpose. Goodix shall assume no responsibility for this information and relevant consequences arising out of the use of such information.

Without written consent of Goodix, it is prohibited to use Goodix products as critical components in any life support system. Under the protection of Goodix intellectual property rights, no license may be transferred implicitly or by any other means.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: Floor 12-13, Phase B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 Zip Code: 518000

Website: www.goodix.com

Preface

Purpose

This document introduces how to install GProgrammer and operate its functional modules, enabling users to quickly get started with GProgrammer.

Audience

This document is intended for:

- GR551x/GR5526 user
- GR551x/GR5526 developer
- GR551x/GR5526 tester

Release Notes

This document is the thirteenth release of *GProgrammer User Manual*, corresponding to GProgrammer V1.2.32.

Revision History

Version	Date	Description				
1.5	2020-05-30	Initial release				
1.6 2020-06-30		 Updated sector-related description in "Chip Configuration". Added "GR551x_console.exe", introducing a command-line program to erase and download commands; added "GR551x_encrypt_signature.exe" and "User-defined Windows Scripts". Introduced the public key hashes to verify signatures, updated the file name extension for encrypted and signed files, and introduced the firmware signing function in "Encrypt & Sign". 				
1.7	2020-08-30	 Introduced the GR5515I0ND System-on-Chip (SoC) for GR551x SoCs in "SoC/MCU Selection". Changed icons for Delete and Startup in "Firmware". 				
1.8	2020-09-30	Added description on firmware download failure in "Download Firmware".				
1.9	2020-11-26	Updated UI screenshots for software version.				
2.0	2021-01-05 Updated software UI screenshots for SoC/MCU selection and firmware operations.					
2.1	2021-03-02	 Added file modification description to "eFuse Settings". Added file export description to "Import and Export". Updated descriptions concerning operations prior to viewing device logs in "Device Log". Added description of IO_LDO_SEL field to "eFuse Layout". Deleted the parameter of nvds in erase and download commands in "GR551x_console.exe". 				
2.2	2021-05-13	Deleted functionalities for GMF03x series.				
2.3	2021-07-16	Updated software UI figures for SoC selection.				



Version	Date	Description				
2.4	2021-09-06	Updated software UI figures for SoC selection.				
2.5	2022-02-20	 Added GR5526 to descriptions specific to SoC models supported by GProgrammer. Updated the "Firmware" and "Encrypt & Sign" sections where adding HEX firmware files via GProgrammer is supported. Updated the "Firmware" section where Export is added to the Firmware interface and exporting BIN firmware files by GProgrammer is supported. Modified the "External Flash" section where QSPI2 is added to SPI Type and relevant configuration items are modified. 				
2.6	2023-01-19	Updated commands supported by <i>GR5xxx_console.exe</i> and <i>GR5xxx_encrypt_signature.exe</i> . Deleted the GR5515I0ND SoC.				
2.7	2023-02-03	 Updated description in "GR5xxx_console.exe" and added commands supported by <i>GR5xxx_console.exe</i>, including "device" and "load". Added description on the parameter "rand_number" to <i>GR5xxx_console.exe</i> commands. Added description on the software package <i>GProgrammer- Version.tar.bz2</i> (running on Linux). Added a tip in "Flash Configuration" in "External Flash". Added description on viewing/reselecting the chip model in use. Added description on refreshing and choosing the serial number of target boards in SWD connection mode. Added more data file formats (except BIN files) support during data download to Flash memories. Added the function button Parse Para for parsing data in the NVDS area. 				

Contents

1 Introduction 1 2 Installation Instructions 2 2.1 Installation Requirements 2 2.2 Installation Steps 2 3 GProgrammer Use Instructions 4 3.1 Hardware Connection 4 3.2 SoC Selection 6 3.3 GProgrammer GUL 6 3.4 Connection Management 7 3.5 Firmware 9 3.5.1 Downloading Firmware 9 3.5.1 Downloading Firmware 10 3.6 Flash 13 3.6.1 Internal Flash 14 3.6.1.2 Frase Flash 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash 19 3.6.2.3 Flash Configuration 19 3.6.2.4 Flash Configuration 19 3.6.2.7 Back Traper Hash 12 3.7 Encrypt & Sign 22 3.7.1 eFuse Settings 23 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 efuse Layout 27 3.9.1 Init NVDS Area <th>Preface</th> <th>I</th>	Preface	I
2.1 Installation Requirements. 2 2.2 Installation Steps. 2 3 GProgrammer Use Instructions. 4 3.1 Hardware Connection. 4 3.2 SoC Selecton. 6 3.3 GProgrammer GUI. 6 3.4 Connection Management. 7 3.5 Firmware. 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration. 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash. 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash. 22 3.7.1 Eruse Settings. 23 3.7.2 Download 25 3.7.3 Encrypt & Sign. 26 3.8 eruse Layout. 27 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.3 Write.	1 Introduction	1
2.2 Installation Steps. 2 3 GProgrammer Use Instructions. 4 3.1 Hardware Connection 4 3.2 SoC Selection. 6 3.3 GProgrammer GUI. 6 3.4 Connection Management. 7 3.5 Firmware. 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data 17 3.6.2.1 Flash Configuration 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash. 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash 19 3.6.2.3 T Flash Configuration 19 3.6.2.4 External Flash Programming. 21 3.7.1 Erus yet & Sign. 22 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9.1 Init NUDS Area. 30 3.9.2 Read All. 31	2 Installation Instructions	2
3 GProgrammer Use Instructions 4 3.1 Hardware Connection 4 3.2 SoC Selection 6 3.3 GProgrammer GUI. 6 3.4 Connection Management. 7 3.5 Firmware. 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash 14 3.6.1.1 Flash Configuration 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data. 19 3.6.2 External Flash. 19 3.6.2.1 Flash Configuration 19 3.6.2 External Flash. 19 3.6.2.2 External Flash. 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash Programming. 21 3.7 1 encrypt & Sign. 22 3.7.1 efuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.9.4 Add a User Parameter. 30 3.9.5 Modify NVDS Parameters. 34 3.9.5 Modify NVDS Parameters. 34	2.1 Installation Requirements	2
3.1 Hardware Connection 4 3.2 SoC Selection 6 3.3 GProgrammer GUI. 6 3.4 Connection Management. 7 3.5 Firmware. 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.2 Frase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data 19 3.6.2 External Flash. 19 3.6.2.2 External Flash. 19 3.6.2.4 Flash Configuration. 19 3.6.2.5 External Flash. 19 3.6.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.7.4 Download. 25 3.7.3 Encrypt & Sign. 26 3.9.1 Init NVDS Area 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 32 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. </th <th>2.2 Installation Steps</th> <th>2</th>	2.2 Installation Steps	2
3.2 SoC Selection 6 3.3 GProgrammer GUI. 6 3.4 Connection Management. 7 3.5 Firmware 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration. 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2.2 External Flash. 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.7 2 Download. 25 3.7.3 Encrypt & Sign. 26 3.7.4 Download. 25 3.7.5 Download. 25 3.7.6 Download. 25 3.7.7 Download. 27 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32	3 GProgrammer Use Instructions	4
3.3 GProgrammer GU. 6 3.4 Connection Management. 7 3.5 Firmware. 9 3.5.1 Downloading Firmware. 10 3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration. 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.2 External Flash. 19 3.6.2 External Flash. 19 3.6.2.2 External Flash Configuration. 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.7.4 Download. 25 3.7.5 Encrypt & Sign. 26 3.7.6 Download. 25 3.7.7 Download. 25 3.7.8 Encrypt & Sign. 26 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters.<	3.1 Hardware Connection	4
3.4 Connection Management 7 3.5 Firmware 9 3.5.1 Downloading Firmware 10 3.5.2 Action Order 12 3.6 Flash 13 3.6.1 Internal Flash 14 3.6.1.1 Flash Configuration 14 3.6.1.2 Erase Flash 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data 19 3.6.2 External Flash 19 3.6.2 External Flash 19 3.6.2.2 External Flash 19 3.6.2.2 External Flash 19 3.6.2.2 External Flash Programming 21 3.7.1 eFuse Settings 22 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 eFuse Layout 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All 31 3.9.3 Write 32 3.9.4 Add a User Parameter 33 3.9.5 Modify NVDS Parameters 34 3.9.5 Renove a User Parameter 35 3.9.7 Import and Export 36 3.9.8 Parse Data in the NVDS Ar	3.2 SoC Selection	6
3.5 Firmware .9 3.5.1 Downloading Firmware .10 3.5.2 Action Order .12 3.6 Flash .13 3.6.1 Internal Flash .14 3.6.1.1 Flash Configuration .14 3.6.1.2 Erase Flash .15 3.6.1.3 Download Data .17 3.6.1.4 Dump Data .19 3.6.2 External Flash .19 3.6.2.1 Flash Configuration .19 3.6.2.2 External Flash .19 3.6.2.1 Flash Configuration .19 3.6.2.2 External Flash .19 3.6.2.3 Download .22 3.7 Encrypt & Sign .21 3.7.3 Encrypt & Sign .22 3.7.4 Download .25 3.7.3 Encrypt & Sign .26 3.7.4 Download .25 3.7.3 Encrypt & Sign .26 3.8 eFuse Layout .27 3.9 Chip Configuration .29 3.9.1 Init NVDS Area .30 3.9.2 Read All .31 3.9.3 Write .32 3.9.4 Add a User Parameter .33 3.9.5 Modify NVDS Parameters	3.3 GProgrammer GUI	6
3.5.1 Downloading Firmware 10 3.5.2 Action Order 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration 14 3.6.1.2 Frase Flash. 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data 19 3.6.2 External Flash. 19 3.6.2 External Flash 19 3.6.2 External Flash Programming 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.1	3.4 Connection Management	7
3.5.2 Action Order. 12 3.6 Flash. 13 3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration. 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2 External Flash. 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.00 Device Log. 37	3.5 Firmware	9
3.6 Flash 13 3.6.1 Internal Flash 14 3.6.1.1 Flash Configuration 14 3.6.1.2 Erase Flash 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data 19 3.6.2 External Flash 19 3.6.2 External Flash 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash Programming 21 3.7 Encrypt & Sign 22 3.7.1 eFuse Settings 23 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 eFuse Layout 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All 31 3.9.3 Write 32 3.9.4 Add a User Parameter 33 3.9.5 Modify NVDS Parameters 34 3.9.6 Remove a User Parameter 35 3.9.7 Import and Export 36 3.9.8 Parse Data in the NVDS Area 36 3.10 Device Log 37	3.5.1 Downloading Firmware	10
3.6.1 Internal Flash. 14 3.6.1.1 Flash Configuration. 14 3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2 External Flash. 19 3.6.2 External Flash. 19 3.6.2 External Flash 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.5.2 Action Order	
3.6.1.1 Flash Configuration 14 3.6.1.2 Erase Flash 15 3.6.1.3 Download Data 17 3.6.1.4 Dump Data 19 3.6.2 External Flash 19 3.6.2 External Flash Programming 21 3.7 Encrypt & Sign 22 3.7.1 eFuse Settings 23 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 eFuse Layout 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All 31 3.9.3 Write 32 3.9.4 Add a User Parameter 33 3.9.5 Modify NVDS Parameters 34 3.9.6 Remove a User Parameter 35 3.9.7 Import and Export 36 3.9.8 Parse Data in the NVDS Area 36 3.10 Device Log 37	3.6 Flash	
3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2 External Flash 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.1 Internal Flash	
3.6.1.2 Erase Flash. 15 3.6.1.3 Download Data. 17 3.6.1.4 Dump Data. 19 3.6.2 External Flash 19 3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.1.1 Flash Configuration	
3.6.1.4 Dump Data. 19 3.6.2 External Flash 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign 22 3.7.1 eFuse Settings 23 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 eFuse Layout 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All 31 3.9.3 Write 32 3.9.4 Add a User Parameter 33 3.9.5 Modify NVDS Parameters 34 3.9.6 Remove a User Parameter 35 3.9.7 Import and Export 36 3.9.8 Parse Data in the NVDS Area 36 3.10 Device Log 37		
3.6.2 External Flash 19 3.6.2.1 Flash Configuration 19 3.6.2.2 External Flash Programming 21 3.7 Encrypt & Sign 22 3.7.1 eFuse Settings 23 3.7.2 Download 25 3.7.3 Encrypt & Sign 26 3.8 eFuse Layout 27 3.9 Chip Configuration 29 3.9.1 Init NVDS Area 30 3.9.2 Read All 31 3.9.3 Write 32 3.9.4 Add a User Parameter 33 3.9.5 Modify NVDS Parameters 34 3.9.6 Remove a User Parameter 35 3.9.7 Import and Export 36 3.9.8 Parse Data in the NVDS Area 36 3.10 Device Log 37	3.6.1.3 Download Data	17
3.6.2.1 Flash Configuration. 19 3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.1.4 Dump Data	
3.6.2.2 External Flash Programming. 21 3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.2 External Flash	
3.7 Encrypt & Sign. 22 3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.2.1 Flash Configuration	
3.7.1 eFuse Settings. 23 3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.6.2.2 External Flash Programming	
3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37		
3.7.2 Download. 25 3.7.3 Encrypt & Sign. 26 3.8 eFuse Layout. 27 3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.7.1 eFuse Settings	
3.8 eFuse Layout.273.9 Chip Configuration.293.9.1 Init NVDS Area.303.9.2 Read All.313.9.3 Write.323.9.4 Add a User Parameter.333.9.5 Modify NVDS Parameters.343.9.6 Remove a User Parameter.353.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37		
3.9 Chip Configuration. 29 3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.7.3 Encrypt & Sign	
3.9.1 Init NVDS Area. 30 3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.8 eFuse Layout	27
3.9.2 Read All. 31 3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.9 Chip Configuration	29
3.9.3 Write. 32 3.9.4 Add a User Parameter. 33 3.9.5 Modify NVDS Parameters. 34 3.9.6 Remove a User Parameter. 35 3.9.7 Import and Export. 36 3.9.8 Parse Data in the NVDS Area. 36 3.10 Device Log. 37	3.9.1 Init NVDS Area	
3.9.4 Add a User Parameter.333.9.5 Modify NVDS Parameters.343.9.6 Remove a User Parameter.353.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37	3.9.2 Read All	
3.9.5 Modify NVDS Parameters.343.9.6 Remove a User Parameter.353.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37	3.9.3 Write	
3.9.6 Remove a User Parameter.353.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37	3.9.4 Add a User Parameter	
3.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37	3.9.5 Modify NVDS Parameters	
3.9.7 Import and Export.363.9.8 Parse Data in the NVDS Area.363.10 Device Log.37		
3.9.8 Parse Data in the NVDS Area		
3.10 Device Log		
-		
	-	

G@DiX

3.11.1 GR5xxx_console.exe	39
3.11.2 GR5xxx_encrypt_signature.exe	42
3.11.3 User-defined Windows Scripts	43
3.12 Help	44

1 Introduction

GProgrammer supports programming of Flash memories on GR551x and GR5526 System-on-Chips (SoCs). It provides the following features.

- Connection via SWD and UART
- Firmware download
- Flash programming & erasing
- Inputting product information (ID, name, description, and value)
- Downloading files to eFuse
- Viewing eFuse contents
- Firmware encryption and signing
- Configuring Non-Volatile Data Storage (NVDS) parameters
- Displaying device logs
- Programming on GR5xxx_console

Figure 1-1 shows the Graphical User Interface (GUI) of GProgrammer.

9	GProgrammer						- 🗆 🗙
۲	Firmware						Ø
ŧ	0x010F FFFF		Firmware File				
ß		ble_app_ancs	User App Firmware:	E:\Firmware\ble_app_hts_fw.b			Export
			(i) Image Info				
(I)			Image Name:	ble_app_hts_	Run Address:		
	Unused		Version:	1	Size(Byte):	81024	
	NVDS		SPI Access Mode:		Boot Delay:	🖲 Yes 🔘 No	
B	Existed		CheckSum:	7990386	Check Image:	🖲 Yes 🔘 No	
	Download	ble_app_hrs_	Load Address:				
•	Delete	ble_app_bps					Update
	Overlapping						
	🧧 Update	ble_app_hrs	Unfinished Eve				
			No. Action	Description	ble_app_ancs_c_fw.bin		×
			2 add		ble_app_hts_fw.bin		×
			3 startup	Start up ble_app_ht			×
	0x0100 2000	ble_app_hts_					
		Refresh Add Delete Startup					Commit

Figure 1-1 GProgrammer GUI

2 Installation Instructions

This chapter describes the environment requirements as well as installation steps for installing GProgrammer.

2.1 Installation Requirements

• Hardware environment

Table 2-1 Hardware environment

Name	Description			
CPU	1.6 GHz and faster			
RAM	1 GB and larger			

• Operating system

Table 2-2 Operating system

Name	Description
Windows	Windows 7/Windows 10 (32-bit/64-bit)
Linux	Ubuntu 20.04 (64-bit)

2.2 Installation Steps

GProgrammer runs on Windows with an executable installation package: *GProgrammer Setup Version.exe*. Users can follow the steps below when installing GProgrammer:

Note:

To run GProgrammer on Linux (64-bit), you only need to unzip the software package *GProgrammer- Version.tar.bz2* and double-click the portable .exe program to launch GProgrammer.

Double-click *GProgrammer Setup Version.exe*, and follow the steps in the **GProgrammer Setup** wizard (see Figure 2-1).

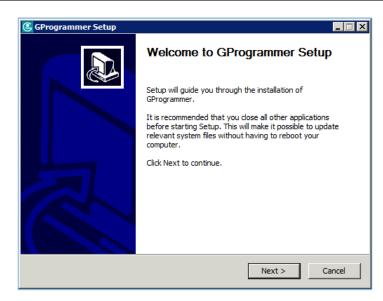


Figure 2-1 GProgrammer Setup installation wizard

Note:

Version indicates the GProgrammer software version number.

2. After installing GProgrammer, you are prompted to install J-Link on demand. See Figure 2-2.

🕒 GProgrammer Setup	
	Completing GProgrammer Setup
	GProgrammer has been installed on your computer. Click Finish to close Setup.
	🗹 Install JLink
凤	
	< Back Finish Cancel

Figure 2-2 Prompt to install J-Link

Tip:

For users who have installed J-Link on their PCs before installing GProgrammer, clear **Install J-Link** in the installation wizard.

3. After installing J-Link, you can start the GProgrammer by clicking the GProgrammer shortcut on desktop or **Start** menu.

3 GProgrammer Use Instructions

This chapter elaborates on how to use functional modules of GProgrammer.

3.1 Hardware Connection

Before starting GProgrammer, make sure the host (PC) is correctly connected to the target board. You can establish the connection in either SWD mode or UART mode.

• SWD mode

In SWD mode, users need a J-Link emulator with one end connecting to the host through a Micro USB cable and the other end connecting to SoC pins of the target board through Dupont wire cables.

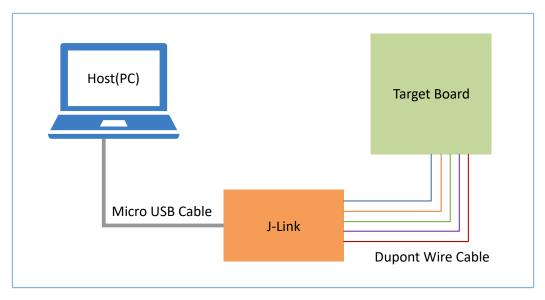


Figure 3-1 Host-target-board connection in SWD mode

The table below lists the mapping relations between J-Link emulator pins and SoC pins.

J-Link Emulator Pin	SoC Pin
VCC	VCC
GND	GND
SWDIO	GPIO_1
SWCLK	GPIO_0

🛄 Note:

- For target boards that have been integrated with J-Link emulator chips, you can connect the host to the target board directly through a Micro USB cable.
- For Goodix Starter Kit (SK) Boards, you cannot connect an SK Board to a PC directly via the on-board J-Link port for firmware programming because the integrated ROM upgrade program in the SoC shall be implemented based on a baud rate of 921600, a value which the integrated J-Link emulator chips on the SK Board fails to support.
- UART mode

In UART mode, users need a USB-to-serial converter with one end connecting to the host through a Micro USB cable and the other end connecting to SoC pins of the target board through Dupont wire cables.

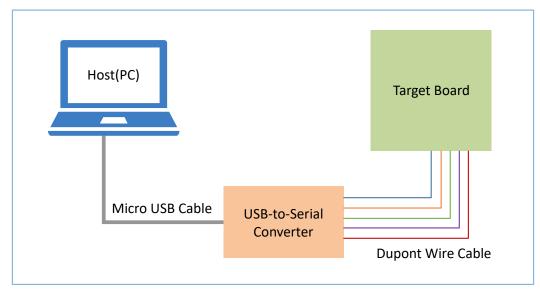


Figure 3-2 Host-target-board connection in UART mode

The table below lists the mapping relations between USB-to-serial converter pins and SoC pins.

USB-to-Serial Converter Pin	SoC Pin
VCC	VCC
GND	GND
ТХ	GPIO_1
RX	GPIO_0
RTS	CHIP_EN

Table 3-2 Mapping relations between USB-to-serial converter pins and SoC pins

△Tip:

- For target boards that have been integrated with USB-to-serial converter chips, you can connect the host to the target board directly through a Micro USB cable.
- When the SWD interface is closed, firmware information in the Flash is to be erased after the Host and the target board is connected through UART.

3.2 SoC Selection

Start GProgrammer. Prior to other operations, you are required to choose the SoC model on your target board and click **OK**.

△Tip:

By default, GProgrammer opens the SoC selection interface when being started.

T Filter Settings	C	FILL	l Products						
Products	~	GR	GR551x series is a high-performance system-on-chip (SoC) supporting Bluetooth 5.1, making it an ideal choice for mobile devices, wearables, and Internet of Things (IoT) products. The series allows users to develop Bluetooth Low Energy (LE) applications and products serving as a Central and/or a Peripheral. GR5515I0NDA is available in QFN56 package.						
# Part Number	>								
Series	>	renpire							
Core	>								
🛗 Memory	>								
Package	>	lten	n list 12 items						
< Peripheral	>	_	Part Number	Core	Frequency	RAM	Flash	Package	Peripherals
Kits	>		GR5515IGND	Cortex-M4F	64MHz	256KB	1024KB	QFN56	2 x QSPI, 2 x SPI, 2 x I2C, 2 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
			GR5515RGBD	Cortex-M4F	64MHz	256KB	1024KB	BGA68	2 x QSPI, 2 x SPI, 2 x I2C, 2 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
			GR5515GGBD	Cortex-M4F	64MHz	256KB	1024KB	BGA55	0 x QSPI, 2 x SPI, 2 x I2C, 2 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
				Carton 100	682.0	600	00	(2715)	2 × 509, 2 × 59, 2 × 50, 2 × 58, 2 × 5487, 1 × 402, 1 × 502193, 4 × 7000
			GR5513BEND	Cortex-M4F	64MHz	128KB	512KB	QFN40	1 x QSPI, 2 x SPI, 2 x I2C, 1 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
			GR5515IENDU	Cortex-M4F	64MHz	256KB	512KB	QFN56	2 x QSPI, 2 x SPI, 2 x I2C, 2 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
		۲	GR5515I0NDA	Cortex-M4F	64MHz	256KB	OKB	QFN56	2 x QSPI, 2 x SPI, 2 x I2C, 2 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
			GR5513BENDU	Cortex-M4F	64MHz	128KB	512KB	QFN40	1 x QSPI, 2 x SPI, 2 x I2C, 1 x I2S, 2 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM
			GR5526VGBIP	Cortex-M4F	96MHz	512KB	1024KB	BGA83	3 x QSPI, 2 x SPI, 6 x I2C, 2 x I2S, 6 x UART, 1 x ADC, 1 x ISO7816, 6 x PWM, 1 x DSPI, 1 x USB
									ОК

Figure 3-3 SoC selection interface

On the SoC selection interface, the left pane lists **Products** and **Kits** options, and the right pane shows the available choices. You can select an SoC by defining its **Part Number**, **Series**, **Core**, **Memory**, **Package**, or **Peripheral**.

△Tip:

Peripherals listed on the SoC selection interface are only part of the peripherals of an SoC. For details of all peripherals, see the datasheet corresponding to SoC series.

3.3 GProgrammer GUI



After you choose an SoC, the main operational interface opens, as shown in the figure below.

9	GProgrammer					- 🗆 ×
	Firmware			8	SWD	UART
	0x010F FFFF		Firmware File User App Firmware: Image Info Image Name: Version: SPI Access Mode: CheckSum:	Run Address: Size(Byte): Boot Delay: Check Image:	Device: Speed:	Cortex-M4 4000 V Connect
6	NVDS		Load Address:			
	0x0100 2000 Refresh Add Dele	te Startup				

Figure 3-4 GProgrammer GUI

The GUI comprises a functional navigation bar on the left (see Table 3-3) and a function operational zone on the right.

lcon	Function Name	Description
C	-	 View/Reselect the chip model in use. You can view the selected chip model by moving the cursor onto this icon. You can select another chip model by clicking this icon to return to the SoC selection interface. The current device is to be disconnected after clicking OK in the lower-right corner.
Ö	Firmware	Display firmware-related operations.
ซิ	Flash	Display operations related to Flash memory.
ê	Encrypt & Sign	Display operations related to firmware encryption and signing.
(1)	eFuse Layout	Display eFuse layout.
ö	Chip Configuration	Display operations related to chip configurations.
Ē	Device Log	Display device logs.
Ð	Help	Display help information.

Table 3-3 Options on the functional navigation bar

3.4 Connection Management

GProgrammer helps users manage and control the connection between your host and target board.

Click
Cl

GProgrammer supports two connection modes: SWD and UART.

• SWD

Users need to configure parameters below and click **Connect** to connect the target board to the host.

Table 3-4 Parameter description

Parameter	Description			
Device	PU of the on-board chip. It is "Cortex-M4" by default and cannot be modified.			
Speed	ata transfer rate. The default value is 4000 .			
SN	 Serial number of the target board. The default value is 0. When the PC is connected with only one target board, you can keep the default value "0" or obtain the corresponding serial number by clicking Refresh. When the PC is connected with multiple target boards, you should obtain the corresponding serial numbers by clicking Refresh, and then choose the target one. In this case, if you keep the default value "0" and start device connection, a window will pop up to inform you of choosing the target board S/N when GProgrammer runs on Windows. 			

SWD		UART
Device:	Cortex-	M4
Speed:	4000	\sim
SN:	0	\sim
R	efresh	Connect

Figure 3-5 GProgrammer SWD connection

UART

Users need to configure **Port** (click **Refresh** and select a correct **Port** value) and **Baudrate** on demand. The default configurations of other parameters (**Parity**, **DataBits**, **StopBits**, and **FlowControl**) cannot be modified.

After setting these parameters, click **Connect** to connect the target board to the host.

8	SWD	UART		
	Port:	COM1	\vee	
	Baudrate:	921600		
	Parity:	None	\sim	
ss:	DataBits:	8	\sim	
ick:	StopBits:	1	\sim	
ed:	FlowControl:	none	\sim	
:			-	
ge: Mode	Re	fresh	Connect	

Figure 3-6 GProgrammer UART connection

After the connection is successfully established, the connection management window automatically hides with the
the
the
the
the
the
the transportation stablishment.

To disconnect the host from the board, click <a>o to open the connection management window, and click <a>Disconnect.

Ø	SWD	UART
	Device:	Cortex-M4
	Speed:	4000 V
	SN:	0 ~
dress:		
e):	Refre	sh Disconnect
lay:		

Figure 3-7 Clicking Disconnect on GProgrammer

3.5 Firmware

Click on the left side of the main interface of GProgrammer to open the **Firmware** interface.



e a	GProgrammer						- 🗆 ×
٠	Firmware						Ø
Ŧ	0x010F FFFF		Firmware File				
٩			User App Firmware:	E:\Firmware\ble_app_hts_fw.b			Export
2			i Image Info				
11			Image Name:	ble_app_hts_	Run Address:		
	Unused		Version:	1	Size(Byte):	81024	
	NVDS		SPI Access Mode:		Boot Delay:	🖲 Yes 🔵 No	
E	Existed		CheckSum:	7990386	Check Image:	🖲 Yes 🔵 No	
	Download		Load Address:				
ð	Delete	ble_app_bps					Update
	Overlapping						
	🧧 Update	ble_app_hrs 🎈	Unfinished Eve				
			No. Action	Description Add and download	d ble_app_ancs_c_fw.bin		×
			2 add		d ble_app_hts_fw.bin		×
			3 startup	Start up ble_app_h	ts_		×
	0x0100 2000	ble_app_hts_					
		Refresh Add Delete Sta	artup				Commit

Figure 3-8 GProgrammer Firmware interface

You can download your application firmware to the contiguous space of Flash memories, ranging from 0x01002000 to 0x010FFFF.

Note:

The start and end addresses of Flash memories to which firmware can be downloaded vary depending on the Flash size of the specific SoC.

3.5.1 Downloading Firmware

GProgrammer graphically displays the Flash memory space layout occupied by firmware (see Figure 3-9), which helps you easily learn the Flash occupation status.



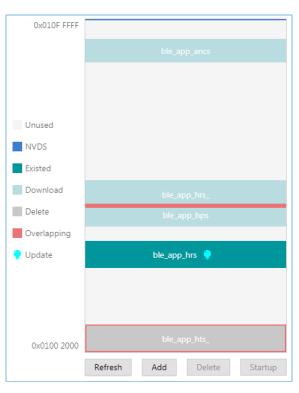


Figure 3-9 Flash firmware layout

- represents Flash space to which data can be downloaded.
- represents default NVDS area to which firmware cannot be downloaded.
- indicates space for storing to-be-deleted firmware. Example: ble_app_hts_.
- indicates space for storing to-be-downloaded firmware. Example: ble_app_ancs.
- Indicates space for storing downloaded firmware in Flash memories. Example: ble_app_hrs.
- Indicates space overlapped by two pieces of firmware. Examples: ble_app_hrs_ and ble_app_bps.

Follow the steps below to download firmware to a Flash memory by using GProgrammer:

- Click Add to add a local firmware file (HEX/BIN) to GProgrammer. GProgrammer presents details of the added firmware such as firmware directory (User App Firmware) and Image Info.
 In the Firmware File area, click Export to convert the imported firmware file to an unencrypted BIN file that can be used by the SoC.
- 2. Click **Commit** to download the firmware to Flash memories.

After downloading, the color of the firmware turns from to , indicating the firmware has been successfully downloaded.

🛄 Note:

- 1. GProgrammer automatically reads firmware existing in the Flash memories after being connected a target board.
- If J-Link cannot be connected when you download firmware, connection/firmware download to the SK Board fails. At this moment, the SoC may be in sleep mode (the firmware keeps running in sleep mode). You can press RESET on the SK Board, wait for around one second, and re-download the firmware. If this approach does not work, erase the Flash and re-download the firmware.

3.5.2 Action Order

You can execute multiple actions at a time. For example, download multiple pieces of firmware to Flash memories and set one piece of firmware as **Startup**. The user-defined actions are executed by clicking **Commit**. The action orders are displayed in **Unfinished Events**, as shown in Figure 3-10.

🕞 Unf	🗐 Unfinished Events					
No.	Action	Description				
1	update	Update ble_app_bps image info	×			
2	delete	Delete ble_app_ancs	×			
3	add	Add and download ble_app_hrs_fw.bin	×			
4	add	Add and download ble_app_hts_fw.bin	×			
5	add	Add and download ble_app_T3u_fw.bin	×			
6	startup	Start up ble_app_hrs_	×			

Figure 3-10 Action order

Executable actions for users are listed in the table below.

Table 3-5 Executable actions for users on GProgrammer

Name	Button/Icon	Description
		Click Add to add a local firmware file to GProgrammer.
		Alternatively, you can add a local firmware file to GProgrammer by directly dragging the file to
Add firmware	Add	GProgrammer from Windows/File Explorer.
		Note:
		Do not click Open after dragging the file to GProgrammer.
		Click Refresh to obtain the information of firmware downloaded in the Flash memories of a
		target board.
Refresh firmware	Refresh	Unexecuted actions of Flash firmware on the living target board in the Unfinished Events pane,
		such as those labeled as startup or update are withdrawn with modified parameters being reset
		to values before refresh.



Name	Button/Icon	Description
Delete firmware	Delete	Click the Delete button to delete existing firmware in Flash memories. Select firmware to be deleted in the Flash firmware layout, and click Delete . The firmware color turns to An action labelled as delete is added to the Unfinished Events . Note: Delete operations result in deleting only image info of the selected firmware. The firmware information stored in the area will not be deleted.
Start execution	Startup	Set firmware as startup to run the firmware immediately. Select firmware in the Flash firmware layout, and click the Startup button. displays on the right of the firmware. An action labelled as startup is added to the Unfinished Events . The host automatically disconnects from the target board after running the firmware.
Update firmware information	Update	Click the Update button to update the information of existing firmware in Flash memories on a target board. Select firmware to be updated in the Flash firmware layout, and modify the firmware information (the color of modified parameters turns to). Click Update , and the icon displays on the right side of the firmware. An action labelled as update is added to the Unfinished Events . Execute update actions, and all parameters involved are locked. No editing is allowed. If modification is required, withdraw the previous update action.

Note:

- In the action order list, you can withdraw an action by clicking imes on the right side of the action.
- For two associated actions, withdrawal of the associated action may lead to automatic withdrawal of the previous action. For example, add a firmware file to Flash memories, and set it as **startup**. Withdrawal of **Add** leads to withdrawal of **Startup**.

In addition, if there is overlapped space for firmware, **Commit** will not be available until the conflict is resolved.

Note:

For two pieces of firmware totally overlapping with each other, you can click the overlapping space to select one piece of firmware and double-click the space to select the other.

3.6 Flash

Click on the left side of the main interface of GProgrammer to open the **Flash** interface.



9	GProgrammer		- c	×
	Flash			Ø
Ū	0x010F FFFF		Flash Configuration	
₽			Internal Flash External Flash ID: Flash Size: 1 M V Conf	ïg
11				
			Erase Flash	
E	Unused		C Erase Sector 0	
0	Boot NVDS		Erase Specified Area 0x 01002000 to 0x 0102FFFF Erase	se
	Firmware	ble_app_bps	🕹 Download Data	
			File Path:	B
			File Size(Byte): Download Address: 0x 00000000 Download	ad
		ble_dfu_boot	Dump Data	
	0x0100 0000		Starting Address: 0x 0000000 Size(Byte): 0 Dum	qr

Figure 3-11 GProgrammer Flash interface

GProgrammer allows users to program internal and external Flash memories of SoCs. Detailed programming actions include **Erase Flash**, **Download Data**, and **Dump Data**.

Similar to the firmware layout, the Flash module presents the Flash space occupation in a graphic manner.

- unused Flash space
- space for NVDS
- Boot info space (0x01000000 to 0x01002000, specific to SoC model). The Boot info space is automatically loaded and displayed when users choose internal Flash memories.
- space for storing downloaded firmware in Flash memories. Example: ble_app_bps
- space to be operated, such as Flash space to be erased

3.6.1 Internal Flash

3.6.1.1 Flash Configuration

Select Internal Flash in the Flash Configuration list to program internal Flash memories.

The Flash layout on the left side of the **Flash** interface automatically synchronizes with updated firmware layout information to obtain the firmware, NVDS, and Boot info space.



9	GProgrammer			- 🗆 X
	Flash			Ø
∵	0x010F FFFF		Flash Configuration Internal Flash External Flash ID: Flash Size: 1 M V	Config
ا ا ≀			Erase Flash	
•	Unused Boot NVDS		Erase Sector 0 • Erase Specified Area 0x 0102FFFF	Erase
	Firmware	ble_app_bps	Download Data File Path: File Size(Byte): Download Address: 0x 0000000 Dc	Download
	0x0100 0000	ble_dfu_boot	Dump Data Starting Address: 0x 00000000 Size(Byte): 0	Dump

Figure 3-12 Selecting Internal Flash

3.6.1.2 Erase Flash

GProgrammer provides three Flash erasing mechanisms: Erase All, Erase Sector, and Erase Specified Area.

• Erase All

The mechanism helps erase all Flash spaces.

The Boot info and NVDS space is cleared with all firmware deleted.



<u></u>	GProgrammer			- 🗆 ×
	Flash			Ø
U	0x010F FFFF		🔁 Flash Configuration	
٨			Internal Flash Flash ID: Flash Size:	Config
11			Erase Flash	
0			Erase All	
E	Unused		C Erase Sector 0	
	Boot		Erase Specified Area 0x 01002000 to 0x 0102FFFF	Erase
1	NVDS Firmware	ble_app_bps	Warning X	
	Firmware		Download Dewnload Frase all data in internal flash, are you sure to continue?	Þ
				Download
	0.0100.0000		Dump Data Starting Address: 0x 00000000 Size(Byte): 0	Dump
	0x0100 0000			

Figure 3-13 Erase All on GProgrammer

Erase Sector

The mechanism helps erase a specified Flash sector (size: 4 KB).

9	GProgrammer		- 🗆 🗙
	Flash		Ø
€	0x010F FFFF	Flash Configuration	
۵		Internal Flash External Flash ID: Flash Size: 1 M Y	Config
UI)			
		Erase Flash	
Ē	Unused	Erase Sector 100 Erase Specified Area 0x 01010000 to 0x 01020000	Erase
i	Boot NVDS		
		🕹 Download Data	
		File Path:	B
		File Size(Byte): Download Address: 0x 00000000	Download
		Dump Data	
	0x0100 0000	Starting Address: 0x 0000000 Size(Byte): 0	Dump

Figure 3-14 Erase Sector on GProgrammer

• Erase Specified Area

The mechanism helps erase an area within a specified address range, by sector.



9	GProgrammer	- 🗆 ×
•	Flash	Ø
	0x010F FFFF	Flash Configuration Internal Flash External Flash Flash Size: 1 M Y Config
© = •	Unused Boot NVDS	Erase Flash Erase All Erase Sector 0 Erase Specified Area 0x 01010000 to 0x 0x 01020000
		Download Data File Path: Download Address: 0x 0000000 Download Download Address: 0x 0000000 Download Starting Address: 0x 0000000 Size(Byte): 0 Dump
	0x0100 0000	Starting Audress. UN 0000000 Size(byte): 0 Dump

Figure 3-15 Erase Specified Area on GProgrammer

3.6.1.3 Download Data

When downloading data to Flash memories on GProgrammer, users only need to view and add the data file, as well as set a start address for downloading in **Download Address**.

△Tip:

The download address shall be 4 KB-aligned.



، 2	GProgrammer		- 🗆 ×
٠	Flash		Ø
Ð	0x010F FFFF	Image: Blass Gopen × Flass ← → × ↑ _ Construction	
₽ III	ble_app	ncs Organize ▼ New folder Extern OneDrive This PC Name Date Type Date Type Date Name Date Sin File Date Sin File Date Sin File	Config
٥		SD Objects SD Objects Destop Destop Document_publ	
8 1	Unused Boot NVDS	 Erase : Bocuments Erase : Downloads Music ■ Pictures Software 	Erase
	Firmware	Dow Image Videos File Path: weekly report File Size(B File name: ble_app_bps_fw.bin Image (*.bin) Open Cancel	Download
	0x0100 0000	Dump Data Starting Address: 0x 0000000 Size(Byte): 0	Dump

Figure 3-16 Viewing and selecting a data file to be downloaded

A Flash overflow error occurs when the downloaded file size is excessively large or the start address is out of range.

9	GProgrammer			- 🗆 ×
۲	Flash			Ø
	0x010F FFFF		Flash Configuration Internal Flash External Flash Flash ID: Flash Size:	Config
© = •	Unused Boot NVDS	ble_app_bps	Erase Flash Erase All Failed Cut of range. CK	Erase
	Firmware	or obhaba	Download Data File Path: F\新产品开发\GRProgrammer测试\ble_app_T3u_fw.bin File Size(Byte): 85728 Download Address: 0x 010EFFFF D	lownload
	0x0100 0000	ble_dfu_boot	Dump Data Starting Address: 0x 00000000 Size(Byte): 0	Dump

Figure 3-17 Flash overflow error

↓Tip:

- Users are allowed to forcibly download data to the Boot info space of GR551x series in SWD connection mode only. In UART mode, force download to the Boot info space is prohibited.
- Downloading data to the Boot info space is not supported by GR5526 series.

3.6.1.4 Dump Data

Users can dump any data in Flash memories to a local file by specifying a starting dump address and the data size.

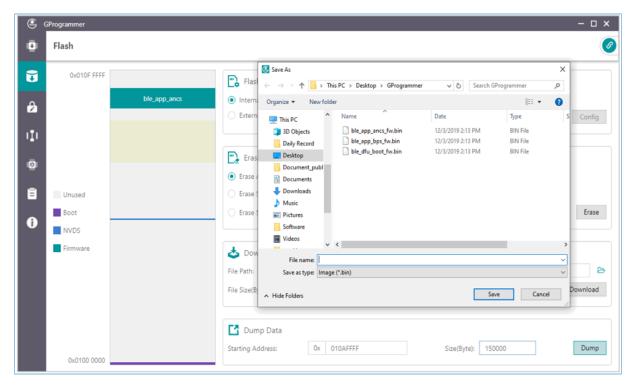


Figure 3-18 Dump Data on GProgrammer

3.6.2 External Flash

3.6.2.1 Flash Configuration

Select **External Flash** in the **Flash Configuration** list to program external Flash memories. Click **Config** to configure the SPI Type and pins based on actual demands.

Click **Apply** to complete the configuration.

G@DiX

↓Tip:

- Before clicking Apply, make sure external Flash memories are correctly connected to the target board in accordance with pin configurations. Incorrect connections lead to failures in communications between external Flash and the board.
- GPIO_0 and GPIO_1 are for device connection, and if they are configured as external Flash pins, GProgrammer will disconnect from the target board.

SPI Type	2: ((J SPI () QS		QSPI1 () QS
	GPIO Type	GPIO P	IN	PIN MUX
CS:	NORMAL \vee	GPIO_6	\sim	MUX_2 V
CLK:	NORMAL \vee	GPIO_3	\sim	MUX_2 V
MOSI:	NORMAL \vee	GPIO_4	\sim	MUX_2 V
MISO:	NORMAL V	GPIO_5	\sim	MUX_2 V

Figure 3-19 SPI configurations

SPI Typ	e:	0 9	SPI 💿 QSPIO	_ QS	PI1 O QS	SPI2
	GPIO Typ	De	GPIO PIN		PIN MU	Х
CS:	AON	\sim	AON_GPIO_1	\sim	MUX_5	$^{\vee}$
CLK:	NORMAL	\sim	GPIO_24	\sim	MUX_5	\vee
IO0:	NORMAL	\sim	GPIO_25	\sim	MUX_5	$^{\vee}$
IO1:	NORMAL	\sim	GPIO_16	\sim	MUX_5	$^{\vee}$
IO2:	NORMAL	\sim	GPIO_17	\sim	MUX_5	$^{\vee}$
IO3:	NORMAL	\sim	GPIO_31	\sim	MUX_5	$^{\vee}$

Figure 3-20 QSPI0 configurations

• Configure Flash Size

After users apply the pin configurations, GProgrammer reads and displays the external **Flash ID** based on which the **Flash Size** is automatically set.

Users need to manually set the **Flash Size** when GProgrammer fails to get the Flash size based on the accessed Flash ID.

9	GProgrammer		- 🗆 🗙
•	Flash		Ø
U	0x000F FFFF	Flash Configuration Internal Flash	
₽		External Flash ID: FFFFFF Flash Size: 1 M	Config
		Warning Erase Fla Erase All Unknown flash ID and you need to config flash size.	
8		Erase Secto OK Erase Specified Area VX O1002000 to VX O1002000 to VX O10020FFFF	Erase
i	Unused		
		🕹 Download Data	
		File Path: F:\GRProgrammer测试\ble_app_T3u_fw.bin	D
		File Size(Byte): 85728 Download Address: 0x 01009000	Download
		Dump Data	
	0×0000 0000	Starting Address: 0x 010AFFFF Size(Byte): 150000	Dump

Figure 3-21 Unknown Flash ID

3.6.2.2 External Flash Programming

GProgrammer allows users to program Flash memories (erase Flash, download data to Flash, and dump data to a local file) within a valid address range.



9	GProgrammer		- 🗆 ×
۵	Flash		Ø
	0x003F FFFF	Flash Configuration Internal Flash External Flash Flash ID: FFFFFF Flash Size:	Config
:-' :		Erase Flash Erase All Erase Sector 10	
0	Unused	Erase Specified Area 0x 01002000 to 0x 0102FFFF	Erase
		File Path: F:\GRProgrammer觊珑\ble_app_T3u_fw.bin	B
		File Size(Byte): 85728 Download Address: 0x 00050000	Download
		Dump Data	
	0x0000 0000	Starting Address: 0x 010AFFFF Size(Byte): 150000	Dump

Figure 3-22 Download Data to external Flash on GProgrammer

△Tip:

No operation on external Flash is allowed before completing pin configurations.

3.7 Encrypt & Sign

Click on the left side of the main interface of GProgrammer to open the Encrypt & Sign interface.



<u>e</u> (GProgrammer		- 🗆 ×
٠	Encrypt & Sign		8
Ŧ	eFuse Settings		
	Name:	ID:	
	Firmware Key:	Using Random Key Select Key	B
1 <u>I</u> 1	Security Mode:	● Open ○ Close SWD: ● Open ○ Close	
	Batch eFuse:	2	
۵			Generate eFuse File
B	Download		
	Encrypt Key Info:		B
•	Mode Control:		B
			Download to eFuse
	Encrypt and Sign		
	Product Info:		B
	Random Number:	Using Random Number Select Number	Ъ
	Firmware:		B
			Encrypt Encrypt and Sign

Figure 3-23 GProgrammer Encrypt & Sign interface

The selected SoCs support Security Mode and Non-security Mode. The mode is determined by the security mode of the product written in eFuse. When Security Mode is enabled, only firmware that has been encrypted and signed can be downloaded to Flash memories.

3.7.1 eFuse Settings

eFuse is a one-time programmable (OTP) memory with random access interfaces on SoCs. The eFuse stores product configurations, security mode control information, and keys for encryption and signing.

When using GProgrammer, users can generate eFuse files by specifying product names, IDs, and firmware keys, and by configuring security mode and SWD interfaces.

Name:	test		ID: 1	
irmware Key:	O Using Random Key	 Select Key 	F:\GR551X update\tools\GProgrammer\test\firmware.key	(
ecurity Mode:	● Open ○ Close		SWD:	
Batch eFuse:	3			

Figure 3-24 Setting eFuse parameters

Dote:

- Firmware keys can be random keys generated by GProgrammer. Users can also add key files on demand.
- When **Security Mode** is enabled, users can choose to **Open** or **Close** the SWD interface.

GProgrammer allows users to generate multiple *Encrypt_key_info.bin* files in batches by checking **Batch eFuse**. The generated files are unique, meeting requirements of scenarios demanding one key for one device. For example, when users input "3" in the **Batch eFuse** box, GProgrammer generates three *Encrypt_key_info.bin* files: *Encrypt_key_info.bin*, 2_*Encrypt_key_info.bin*, and 3_*Encrypt_key_info.bin*.

Generated files are listed in the figure below:

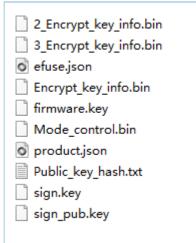


Figure 3-25 Generated files

- *efuse.json*: a temporary file
- Encrypt_key_info.bin, 2_Encrypt_key_info.bin, and 3_Encrypt_key_info.bin: files to be downloaded to eFuse, covering information on products, encryption, and signing. These files shall be downloaded to and stored in eFuse.
- *firmware.key*: a private key for encrypting firmware
- *Mode_control.bin*: an eFuse file covering information on security mode and SWD. This file shall be downloaded to and stored in eFuse.
- *product.json*: a product information file. This file shall be imported to a GProgrammer when encrypting or signing firmware.
- *sign.key*: a private key to generate signatures
- *sign_pub.key*: a public key to verify signatures
- Public_key_hash.txt: a public key hash file to verify signatures

To make files download to eFuse or firmware encryption and signing user-friendly, GProgrammer automatically loads the paths of the *Encrypt_key_info.bin* file and the *Mode_control.bin* file to the **Download** area, and the path of the *product.json* file to the **Product Info** pane in the **Encrypt and Sign** area, as shown in the figure below.



(<u> </u>	Programmer								- 🗆 ×
	Encrypt & Sign								×
÷	eFuse Settings								
	Name:	test			ID:	1111			
2	Firmware Key:	 Using Random Key 	🔿 Select Key						B
III	Security Mode:	● Open 🔵 Close	Success			×	Close		
'1'	✓ Batch eFuse:	3		Complete to ge	enerate efuse fil	e.			
۵	Only Data Key is different	between batch eFuse files.							Generate eFuse File
B	Download			OK]		
В	Encrypt Key Info:	C:\eFuse\Encrypt_key_info.bin							Þ
•	Mode Control:	C:\eFuse\Mode_control.bin							D
									Download to eFuse
	Encrypt and Sign								
	Product Info:	C:\eFuse\product.json							B
	Random Number:	 Using Random Number 	🔿 Select Ni	umber					B
	Firmware:								Ď
								Encryp	Encrypt and Sign

Figure 3-26 Paths for automatically loaded files

Note:

No modification of eFuse-generated files is allowed because any modification may lead to firmware encryption and signing failures.

3.7.2 Download

For users who have clicked **Generate eFuse File** to generate *Encrypt_key_info.bin* and *Mode_control.bin* files in the **eFuse Settings** pane, select **Encrypt Key Info** and **Mode Control** in the **Download** pane, and click **Download to eFuse** to download the files to eFuse.

Otherwise, users need to manually add *Encrypt_key_info.bin* and *Mode_control.bin* files before downloading the files to eFuse.

	GØ	DiX
--	----	-----

9	Programmer		- o x
•	Encrypt & Sign		8
Ð	eFuse Settings		
	Name:	test ID: 1111	
2	Firmware Key:	Using Random Key Select Key	B
11	Security Mode:	● Open ○ Close SWD: ● Open ○ Close	
'4'	 Batch eFuse: 	3	
	Only Data Key is different	between batch eFuse files.	Generate eFuse File
E	Download		
	 Encrypt Key Info: 	C:\eFuse\Encrypt_key_info.bin	B
ð	✓ Mode Control:	C:\eFuse\Mode_control.bin	B
			Download to eFuse
	Encrypt and Sign		
	Product Info:	C:\eFuse\product.json	B
	Random Number:	Using Random Number Select Number	B
	Firmware:		//
			Encrypt Encrypt and Sign

Figure 3-27 Downloading files to eFuse

Note:

eFuse information cannot be repeatedly downloaded to firmware.

3.7.3 Encrypt & Sign

When Security Mode is enabled, only firmware that has been encrypted and signed can be downloaded to Flash memories. GProgrammer allows users to encrypt and sign, or to sign multiple firmware files (HEX/BIN) by using one set of product information (**Product Info**) and one random number (**Random Number**).

The Random Number can be manually set by users or generated by GProgrammer.

When adding more than one firmware file, separate each file path with a semicolon (;), as shown in Figure 3-28.

Encrypt and Sign				
Product Info:	C:\eFuse\product.json			Þ
Random Number:	● Using Random Number	🔘 Select Number		Þ
Firmware:	C:\firmware\ble_app_ancs_fw.bin	;C:\firmware\ble_app_bps_fw	v.bin;C:\firmware\ble_app_hrs_fw.bin;C:\firmware\ble_app_hts_fw.bin	þ
			Z Encrypt	Encrypt and Sign

Figure 3-28 Adding more than one firmware file

To encrypt and sign the firmware, check the **Encrypt** box, and the button changes from **Sign** to **Encrypt and Sign**; to sign the firmware only, clear the **Encrypt** box, and the button changes back to **Sign**. Choose the directory to save the (encrypted and) signed firmware, and click **Encrypt and Sign/Sign**.

Files after being encrypted and signed are generated in BIN formats with details listed below:

ble_app_ancs_fw_encryptandsign.bin
 ble_app_bps_fw_encryptandsign.bin
 ble_app_hrs_fw_encryptandsign.bin
 ble_app_hts_fw_encryptandsign.bin
 random.bin



Files after being signed are listed below:

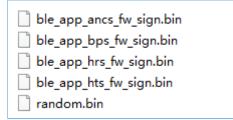


Figure 3-30 GProgrammer-generated files after signing

Note:

The random number generated by GProgrammer is for encryption algorithms. After users perform encryption and signing of firmware files, the *random.bin* file is stored in the same directory as encrypted and signed firmware files. Users can view and add the *random.bin* file to GProgrammer next time they use the random number for firmware encryption and signing.

3.8 eFuse Layout

Click 💷 on the left side of the main interface of GProgrammer to open the **eFuse Layout** interface.

	Offset	Name		Value	I	Length	Comments
	0x015E	Product ID		00 00		2	product identity
	0x0158	Chip ID		00 00 00 00 00 00 00		6	chip identity
	0x0152	EncMode		00 00		2	encrypted or not
	0x0150	SWDDisable		00 00		2	enable SWD or not
^	0x014C	Config		00 00 00 00		4	chip configuration
	0x013C	Chip UID		54 53 4D 14 04 50 52 59 57 34 38 30	30 08 6F 22	16	
	0x012A	ХО		00 00		2	xo offset
	0x0124	BT_MAC		00 00 00 00 00 00		6	
	0x0112	Package Type		00		1	0:Unused, 1:GR5515RGBD, 2:GR5515GGBD, 3:GR5515IGND, 4:GR5515I0ND, 5:GR5513BEND, 6:GR5515BEND, 7:GR5513BEN
^	0x0111	IO_LDO_SEL		08		1	
	Name		Value		Comments		Operate
	IO_PWR_S	SRC	1 ~		0: Internal, 1: External		Write

Figure 3-31 eFuse Layout interface

GProgrammer presents users with eFuse layout information: Offset, Value, Length, and Comments of fields including but not limited to Product ID, Chip ID, EncMode, SWDDisable, Config, and IO_LDO_SEL. Among them, the Config and IO_LDO_SEL fields contain multiple bit fields.

Click **Refresh** to obtain the values of all fields or bit fields.

Click $^{\circ}$ before **Offset** of **Config** or **IO_LDO_SEL** to expand the detailed bits, as shown in the figure below. Click $^{\circ}$ or double-click **Config** or **IO_LDO_SEL** to collapse the detailed bits.

You can change the **IO_PWR_SRC** value in the **IO_LDO_SEL** field to set the power source of peripherals.

Note:

You can only change the **IO_PWR_SRC** value from "0" to "1". The contrary direction is not allowed.



	eFuse Lay	out				
are	Offset	Name	Value		Length	Comments
	0x015E	Product ID	00 00		2	product identity
	0x0158	Chip ID	00 00 00 00 00 00 00		б	chip identity
	0x0152	EncMode	00 00		2	encrypted or not
I.	0x0150	SWDDisable	00 00		2	enable SWD or not
	∨ 0x014C	Config	00 00 00 00		4	chip configuration
	0	upgrade_disable	0		1	
	1	boot_clk	000		3	0: PLL-64MHz, 1: PLL-48MHz, 2: XO-16MHz, 3: PLL-24MHz, 4: PLL-16MHz, 5: PLL-32MHz
	4	dpad_while_disable	0		1	
	5	rx_sample_delay	00		2	
	7	flash_power_up_delay	0000		4	
	11	spi_mode	00		2	mode 0, 1, 2, 3
	13	clk_fls_ctrl	0000		4	0: 64MHz, 1: 48MHz, 2: 32MHz, 3: 24MHz, 4: 16MHz, 5: 16MHz
	Name	Va	lue	Comments		Operate
	IO_PWR_S	SRC	1 🗸	0: Internal, 1: Exter	rnal	Write

Figure 3-32 Expanded Offset

Note:

The fields and bit fields listed in the interface are stored in the *efuse_config.json* file in the config folder. Information stored in eFuse is more than just the listed fields and bit fields.

3.9 Chip Configuration

Click on the left side of the main interface of GProgrammer to open the **Chip Configuration** interface.

<u>e</u> e	GProgramn	mer								- 0	×
•	Chip (Configurati	on					=			×
÷	Init NV	'DS Area									
	Start Ad	ldress: 0x	0107F000					Sec	tors: 1		
۶	Parame	eters									
	All	ID	Parameter Name	Description	Length(Byte)	Value		Value In Chip			
I.	USEF	R Parameters								+ 🗎	~ ^
0	RON	1 Parameters									~
- * *		0xC001	BD_ADDRESS	Device Address	б	01:23:45:67:89:AB					
Ē		0xC002	DEVICE_NAME	Device Name	4	name					
		0xC007	LPCLK_DRIFT	Sleep Clock Accuracy	2	500					
•		0xC085	CODED_PHY_500	Prefer LE Coded PHY 500K	1	0×00					
ð		0xCOB1	RF_XO_OFFSET	XO offset	2	0x0100					
	Unfin	ished				Import	Export	Write	Read All	Parse F	ara

Figure 3-33 GProgrammer Chip Configuration interface

GProgrammer allows users to set the parameters (including **USER Parameters** and **ROM Parameters**) stored in the NVDS area.

- USER Parameters: user-defined parameters that can be added, deleted, and modified
- **ROM Parameters**: ROM parameters stored on GR551x SoCs, which can be modified only by users. Neither parameter addition nor deletion is allowed.

🛄 Note:

- The default ROM parameters listed in the interface are stored in the *nvds_config.json* file in the config folder. The parameters are not results accessed in real time from the NVDS area. For more information about ROM parameters, see Table 3-6.
- Click ⇒ in the upper-right corner of the **Chip Configuration** interface to enable display of complete value contents of a parameter.
- Look up parameters quickly by using the \mathbb{Z} screening box in the upper-right corner of the interface.

ID	Parameter Name	Description
0xC001	BD_ADDRESS	This parameter sets the Bluetooth device address.
0xC002	DEVICE_NAME	This parameter sets the device name.
0xC007	LPCLK_DRIFT	This parameter sets the Sleep Clock Accuracy (SCA); range: 10 ppm to 500 ppm
0xC085	CODED_PHY_500	This parameter sets the default Coded PHY value; Value 0: 125 kbps; Value 1: 500 kbps
0xC0B1	RF_XO_OFFSET	This parameter sets the clock calibration byte; range: 0x000 to 0x1FF

Table 3-6 NVDS ROM parameters

3.9.1 Init NVDS Area

Prior to configuring NVDS parameters, users need to specify a start address (4 KB aligned) and the number of occupied sectors in the NVDS area.

Init NVDS Ar	ea			
Start Address:	0x	010FF000	Sectors:	1

Figure 3-34 Setting the start address and sector quantity in the NVDS area

NVDS initialization fails when the configured NVDS area overlaps with the existing firmware area.



<u> </u>	GProgrammer									- 0	⊐ × □
•	Chip Configurat	ion						⇒ (Ø
•	Init NVDS Area										
	Start Address: 0x	010FF000							Sectors: 1		
	Parameters										
	All ID	Parameter Name	error			×		Value In Ch	ip		
١Į١	USER Parameters		Can not init NV	'DS in the area	firmware exi	sts.				+ 🖻	^
ø	ROM Parameters										~
- 11	0xC001	BD_ADDRESS		ОК							
Ē	0xC002	DEVICE_NAME	Sleep Clock Accuracy	2	500						
	0xC085	CODED_PHY_500	Prefer LE Coded PHY 500K	1	0x00						
ð	0xCOB1	RF_XO_OFFSET	XO offset	2	0x0100						
	Unfinished					Import	Export	Write	Read All	Parse F	Para
					l	import	Export	wille	Read All	raiser	ala

Figure 3-35 NVDS initialization failure

3.9.2 Read All

GProgrammer can read all parameters in the current NVDS area and display them in the **Parameters** pane.

To prevent operation failures in user applications due to parameter overlapping in the NVDS area, users are recommended to click **Read All** after connecting the target board to the host.

GProgrammer provides three parameter states: **Unfinished**, **Same**, and **Different**, which help you quickly identify the parameter state in the current NVDS. Details are listed below:

- **Unfinished**: Parameters in unfinished state are presented in black. These parameters are either new ones different from the default listed parameters after users click **Read All** (example: 0x4000 in Figure 3-36) or ones that have been listed in the NVDS area but with a different parameter length (example: 0x4001 in Figure 3-36).
- **Same**: Parameters in same state are presented in green, indicating the parameters already exist in the NVDS area and have the same length and value as those in the default list (example: 0x4002 in Figure 3-36)
- Different: Parameters in different state are presented in orange, indicating the parameters already exist in the NVDS area and have the same length as but a different value from default listed parameters (example: 0x4003 in Figure 3-36)



Chip	Configurati	on					
Init N	VDS Area						
Start A	Address: 0x	10ff000				Sectors: 1	
Paran	neters						
All	ID	Parameter Name	Description	Length(Byte)	Value	Value In Chip	
US	ER Parameters						+
	0x4001	test1	test1	1	1	01:00	
	0x4003	test3	test3	1	2	1	
	0x4000			0		01	
	0x4002	test2	test2	1	1	1	
RO	M Parameters						
	0xC001	BD_ADDRESS	Device Address	6	01:23:45:67:89:AB	N/A	
	0xC002	DEVICE_NAME	Device Name	4	name	N/A	
	0xC007	LPCLK_DRIFT	Sleep Clock Accuracy	2	500	N/A	
	0xC085	CODED_PHY_500	Prefer LE Coded PHY 500K	1	0x00	N/A	
	0xCOB1	RF_XO_OFFSET	XO offset	2	0x0100	0x008c	
■ Unfi	inished 🔳 Same	e 📕 Different			Import Exp	Write Read All	Parse Para

Figure 3-36 Read All interface

3.9.3 Write

Select parameters to be written to NVDS, and click Write.

Init N	VDS Area						
	ddress: 0x	10ff000				Sectors: 1	
Param	neters						
All	ID	Parameter Name	Warning		×	Value In Chip	
USI	ER Parameters		The following v	alue will be w	riten in chip:		+ 🛍
	0x4001	test1	0xC002 : name			01:00	
	0x4003	test3	Are you sure to	continue?		1	
	0x4000		ОК	Cancel		01	
	0x4002	test2	OK	Cancer		1	
	M Parameters						
	0xC001	BD_ADDRESS	Device Address	6	01:23:45:67:89:AB	N/A	
~	0xC002	DEVICE_NAME	Device Name	4	name	N/A	
	0xC007	LPCLK_DRIFT	Sleep Clock Accuracy	2	500	N/A	
	0xC085	CODED_PHY_500	Prefer LE Coded PHY 500K	1	0x00	N/A	
	0xC0B1	RF_XO_OFFSET	XO offset	2	0x0100	0x008c	

Figure 3-37 Write parameters to NVDS

GODiX

↓Tip:

- Parameters in unfinished state cannot be written to NVDS directly.
- You can select more than one parameter to implement a batch write.
- When an unfinished parameter is selected, **Write** is unavailable.

3.9.4 Add a User Parameter

Follow the steps below to add a user parameter to NVDS.

- 1. Click + to open the Add USER Parameter window.
- 2. Specify the **ID**, **Parameter Name**, **Description**, **Type**, **Length(Byte)**, **Value**, and data presentation format (**dec** or **hex**).

Add USER Paramete	er	×
ID	0x 4000~40FF	
Parameter Name		
Description		
Туре	Unsigned Integer	
Length(Byte)	0	
Value		
	● dec ○ hex	¢
	OK Cancel	

Figure 3-38 Adding a user parameter to NVDS

3. Click **OK** to complete the adding.

GODiX

🛄 Note:

- You cannot input a parameter ID that is identical with those listed in the **Parameters** pane. Otherwise, a warning dialog box pops up, as shown in Figure 3-39.
- If the added ID is different from those existing in the NVDS, the added parameter is directly written to NVDS.
- If the ID of a to-be-added parameter already exists in NVDS and the two parameters with the same ID are of the same length, the to-be-added parameter is written to NVDS.
- If the ID of a to-be-added parameter already exists in NVDS but the two parameters with the same ID are of different lengths, the to-be-added parameter is not written to NVDS. Users need to modify the parameter length before writing it to NVDS.

	0x4001 e	existed	existed parameter	1	10
RON	Add USER Parame	ter	×		
	ID	0x 4001	Warning		× ⁸
	10	0X 4001	User Paramete	er ID must be ide	ntical
	Parameter Name	duplicated			in the second seco
	Description	duplicated paramter		OK	
	Туре	Unsigned Integer			
Jnfin	Length(Byte)	1			
	Value	20			
			💿 dec 🔵 hex		
		OK Cance	1		

Figure 3-39 Failure to add a user parameter due to an identical parameter ID

3.9.5 Modify NVDS Parameters

Users can modify both the USER Parameters and ROM Parameters.

ROM Parameters: You can modify the **Parameter Name**, **Description**, and **Value** of a ROM parameter. The modification on a parameter value does not lead to changes in the parameter length (except varying-length character strings).

USER Parameters: For user parameters in same and different states, the **Parameter Name**, **Description**, and **Value** can be modified. For user parameters in unfinished state, the **Type** and **Length(Byte)** can be modified.

Double-click a parameter to be modified, and edit the parameter information in the pop-up window. Click **OK** to write the modifications into NVDS.



ID	0 x 4001
Parameter Name	ABC
Description	abc
Туре	Address
Length(Byte)	6
Value	ΑΑ:ΑΑ:ΑΑ:ΑΑ:ΑΑ:ΑΑ

Figure 3-40 Edit Parameter Value window

Note:

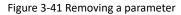
Parameters in unfinished state with a modified length that is different from that in the NVDS remain unfinished. Such parameters cannot be automatically written into the NVDS.

3.9.6 Remove a User Parameter

Users can remove user parameters only.

Select a parameter to be removed, and click Delete to remove the parameter from the NVDS.

9	GProgram	mer						- 0	⊐ ×
•	Chip	Configurati	on				⇒		Ø
•	Init NV	/DS Area							
÷	Start Ad	ddress: 0x	10ff000				Sectors: 1		
۶	Param	eters							
	All	ID	Parameter Name	Warning		×	Value In Chip		
I.	USE	R Parameters		The selected parameter	ers will be de	eted from chip.		+ 🛍	~
		0x4001	testl	The selected parameter are you sure to contin			01:00		
		0x4003	test3				1		
E		0x4000		OK	Cancel		01		
	~	0x4002	test2	test2	1	1	1		
ì		VI Parameters							~
		0xC001	BD_ADDRESS	Device Address	6	01:23:45:67:89:AB	N/A		
		0xC002	DEVICE_NAME	Device Name	4	name	N/A		
		0xC007	LPCLK_DRIFT	Sleep Clock Accuracy	2	500	N/A		
		0xC085	CODED_PHY_500	Prefer LE Coded PHY 500K	1	0x00	N/A		
		0xC0B1	RF_XO_OFFSET	XO offset	2	0x0100	0x008c		
	Unfir	nished 📕 Same	e 📕 Different			Import	Export Write Read All	Parse F	ara



G@DiX

△Tip:

- You can select more than one parameter and click Delete to implement a batch removal.
- When a ROM parameter is selected, **Remove** is unavailable (^{Delete} is in grey).

3.9.7 Import and Export

GProgrammer allows users to export the selected parameter data (**Parameter Name**, **Description**, **Length**, and **Value**) to a local JSON configuration file and import local JSON configuration files to GProgrammer.

	Init NVDS Area		💽 Open				×		
	Start Address: 0x	10ff000		sktop > GProgrammer	✓ δ Search GProgram	nmer J		Sectors: 1	
	Parameters		Organize • New folde				D		
	All ID	Parameter Na	OneDrive	Name	Date	Туре	Value In Chi	2	
Ŀ	USER Parameters	Parante del 14	This PC	efusejson	2019/10/29 19:14	JSON 文件	voice in chi	P	+ 🛍
			3D Objects	productjson	2019/10/29 19:14	JSON 文件	01:00		
	0x4001	test1	Daily Record Desktop						
	0x4003 0x4000	test3	Document_publ				1 01		
	0x4000	test2	Documents				1		
		testz	Downloads Music						
	ROM Parameters		Pictures						
	0xC001	BD_ADDRESS	Software				N/A		
	0xC002	DEVICE_NAM					N/A		
	0xC007	LPCLK_DRIFT	weekly report	<			> N/A		
	0xC085	CODED_PHY_	File na	me	 NVDS Config (*,j) 	json) `	N/A		
	0xC0B1	RF_XO_OFFSE			Open	Cancel	0x008c		
						Imp	ort config file		
	Unfinished Sa	ame 📕 Different				I	mport Export	Write	Read

Figure 3-42 Importing local JSON configuration files to GProgrammer

△Tip:

- Parameters in the imported JSON files replace all those listed in the Parameters pane.
- Export modified parameter data to a local JSON file to prevent repeated modification.
- **Export** is unavailable when parameters in unfinished state exist.

3.9.8 Parse Data in the NVDS Area

GProgrammer provides users with data parsing functionality **Parse Para**. It supports parsing data read from the NVDS area or loaded from a local data file.



٩	GProgrammer				- 0	I X
	Chip Configuration					Ø
÷	Init NVDS Area					
	Start Address: 0x 010	FF000	Se	ectors: 1		
₽	Parameters					
	All ID	Parameters	×	, ,		
1 <u>1</u> 1	USER Parameters	Init NVDS Area			+ 🛍	~
Ö	ROM Parameters					~
	0xC001	Internal Flash Start Address: 0x 010FF000 Sectors: 1				
E	0xC002	O Parse File	B			
	0xC007	Deve en et eu				
i	0xC085	Parameters				
	0xC0B1	ID Length(Byte) Value In Chip				
	 Unfinished 		Parse	Read All	Parse Pa	ara
			- disc			_

Figure 3-43 Configuring to-be-parsed data

• To parse data read from the NVDS area, choose **Internal Flash**, and then set the start address of the NVDS area (4 KB aligned) as well as the number of occupied sector(s).

The area shall be in the range configured for NVDS parameters, with start address and the number of occupied sector(s) as detailed in "Section 3.9.1 Init NVDS Area". Otherwise, data parsing fails.

• To parse data loaded from a local data file, choose **Parse File**, and then select an exported NVDS data file locally.

🛄 Note:

- This functionality is applicable to non-encrypted data only.
- When parsing starts by clicking **Parse**, the sequence of data parsing and result display is identical with that of Flash memory data. **Value in Chip** is in little-endian mode.

3.10 Device Log

Click on the left side of the main interface of GProgrammer to open the **Device Log** interface.



e 6	Program	mer –	п×
•	Devic	ie Log	Ø
Ŧ	ID	CONTENT	ascii
÷	A001	HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-0000000A LR-01020581 PC-01015FEC XPSR-61000011	
β.	A002	HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-0000000 R12-0D266465 LR-010058AD PC-01015FEC XPSR-61000011	
	A003	HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-0D2E6465 LR-010058AD PC-01015FEC XPSR-61000011	
III	A004		
'4'	A005		
۵	A006		
	A007		
E	A008		
	A009		
1	A00A		
	A00B		
	A00C		
	A00D		
	A00E		
	A00F		
	A010		
			Read

Figure 3-44 Device Log interface

Users can view device logs, mainly error information during SoC running, on GProgrammer. Click **Read** to retrieve the device logs.

Note:

Prior to viewing device logs, make sure you have performed the following:

- Write device error code into the NVDS by using the application firmware (NVDS ID: A001–A010).
- Initialize the NVDS area correctly on GProgrammer, and the initialization result is identical with the value defined in the application firmware.

In the interface, click • or • in the upper-right corner to switch the mode in displaying device logs between ASCII and stream.

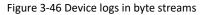
- Image: The device logs are displayed by ASCII character as shown in Figure 3-45.
 - The device logs are displayed by byte stream as shown in Figure 3-46.

ø	Devic	e Log	Ø
Đ	ID A001	CONTENT HARDFAULT CALLSTACK INFC: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-0000000A LR-01020581 PC-01015FEC XPSR-61000011	ascii
٩		HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-0000000 R4-01020501 PC-01015FEC XPSR-0100011 HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-00020000 R2-001025501 PC-01015FEC XPSR-0100011	
	A003	HARDFAULT CALLSTACK INFO: R0-00000000 R1-00000000 R2-00000000 R3-00000000 R12-0D2E6465 LR-010058AD PC-01015FEC XPSR-61000011	

Figure 3-45 Device logs in ASCII characters



ID	CONTENT
A001	48 41 52 44 46 41 55 4C 54 20 43 41 4C 4C 53 54 41 43 48 20 49 4E 46 4F 3A 20 52 30 2D 30 30 30 30 30 30 30 30 20 52 31 2D 30 30 30 30 30 30 30 30 20 52 32 2D 30 30 30 30 30 30 30 30 30 30 30 30 30
A002	48 41 52 44 46 41 55 4C 54 20 43 41 4C 4C 53 54 41 43 48 20 49 4E 46 4F 3A 20 52 30 2D 30 30 30 30 30 30 30 30 30 20 52 31 2D 30 30 30 30 30 30 30 20 52 32 2D 30 30 30 30 30 30 30 30 30 30 30 30 30
A003	48 41 52 44 46 41 55 4C 54 20 43 41 4C 4C 53 54 41 43 4B 20 49 4E 46 4F 3A 20 52 30 2D 30 30 30 30 30 30 30 30 20 52 31 2D 30 30 30 30 30 30 30 30 20 52 32 2D 30 30 30 30 30 30 30 30 30 30 30 30 30



3.11 Command-line Programs

Goodix provides two command-line programs in the GProgrammer installation directory: *GR5xxx_console.exe* and *GR5xxx_encrypt_signature.exe*.

🛄 Note:

GR5xxx, representing the name of SoC series, includes GR551x and GR5526.

- *GR5xxx_console.exe* supports firmware download and Flash programming in SoCs in a command-line interface.
- *GR5xxx_encrypt_signature.exe* supports firmware encryption and (or) signing in a command-line interface.

3.11.1 GR5xxx_console.exe

Follow the steps below to run *GR5xxx_console.exe*:

- 1. Open the **Command Prompt** window from the **Start** menu or by entering **cmd** in the **Run** window.
- 2. Navigate to the GProgrammer installation directory by using cd command.
- 3. Type the GR5xxx_console.exe command to complete corresponding operations. The details about the command are shown in Table 3-7.

Command	Functional Description	Command Format and Parameter Description	Remarks
program	Programs firmware files to internal SoC Flash memories.	<pre>program <firmware file="" path=""> <run immediately:y="" n="" =""> <flash address(hex)="" start=""> <flash size=""> <product type=""> <s n(optional)=""> Parameter description: <firmware file="" path="">: It sets the path of the to-be- downloaded firmware file. <run immediately:y="" n="" ="">: It decides on whether to run the firmware immediately after downloading.</run></firmware></s></product></flash></flash></run></firmware></pre>	The following parameters apply to all commands: • <flash start<br="">address(hex)>: It sets the start address in the Flash memories to which firmware files are downloaded. Value:</flash>

Table 3-7 GR5xxx_console supported	l commands
------------------------------------	------------



Command	Functional Description	Command Format and Parameter Description	Remarks
erase	Erases Flash memory data within an SoC based on a specified address range.	 erase <start address<hex="">> <end address<hex="">><force< li=""> erase when conflict with firmware/bootinfo:y n> <flash address(hex)="" start=""> <flash size=""> <product type=""></product></flash></flash> <s n(optional)=""></s> Parameter description: <start address<(hex)="">>: It represents the start address of the storage area to be erased (in hexadecimal).</start> <end address<(hex)="">>: It represents the end address of the storage area to be erased (in hexadecimal).</end> <force bootinfo:y="" conflict="" erase="" firmware="" n="" when="" with="" ="">: This parameter decides whether to forcibly erase the flash memory data when its address</force> </force<></end></start>	 0x01000000: for GR551x 0x00200000: for GR5526 <flash size="">: lt indicates the Flash size (unit: KB) of the</flash>
eraseall	Erases all Flash memory data within an SoC.	conflicts with that of firmware, Boot info, or NVDS. eraseall <product type=""> <s n(optional)=""> download <data file="" path=""> <start address<(hex)="">></start></data></s></product>	selected SoC. For value details, see the Flash column in Figure 3-3. Note:
download	Downloads data files to internal SoC Flash memories.	<pre><force bootinfo:y="" conflict="" download="" firmware="" n="" when="" with="" =""> <flash address(hex)="" start=""> <flash size=""> <product type=""> <s n(optional)=""> Parameter description: • <data file="" path="">: It sets the path of the to-be- downloaded data file. • <start address<(hex)="">>: It represents the start address of the download area (in hexadecimal). • <force bootinfo:y="" conflict="" download="" firmware="" n="" when="" with="" ="">: This parameter decides whether to forcibly download the data files to internal SoC Flash memories when their addresses conflict with that of firmware or Boot info.</force></start></data></s></product></flash></flash></force></pre>	 For SoCs with 0 KB Flash, the external Flash size applies. <product type="">: It indicates the SoC series. Valid value and description: 0: GR551x 1: GR5526 </product> <s n(optional)="">: Serial number to identify devices to be operated, which is required only when a PC is connected with</s>
writeefuse	Writes Encrypt Key Info and Mode Control files to eFuse.	 writeefuse <encrypt file="" info="" key="" path=""> <mode control<="" li=""> file Path> <product type=""> <s n(optional)=""></s></product> Parameter description: <encrypt file="" info="" key="" path="">: It sets the path of Encrypt Key Info file.</encrypt> <mode control="" file="" path="">: It sets the path of Mode Control file.</mode> </mode></encrypt>	PC is connected with multiple DUTs.



Command	Functional Description	Command Format and Parameter Description	Remarks
generate BIN fi	Converts firmware files into BIN files that can be used	generate <input file="" firmware="" path=""/> <output firmware<br="">file path> <flash address(hex)="" start=""> <flash size=""> <product type=""> Parameter description: • <input file="" firmware="" path=""/>: It indicates the path of an</product></flash></flash></output>	
	by the SoC	 imported file (HEX/BIN file generated via integrated development environment tools). <output file="" firmware="" path="">: It indicates the output file of an exported file.</output> 	
device	Obtains J-Link device information, such as device name and serial number.	 device <display count(optional)=""></display> Parameter description: <display count(optional)="">: It sets the number of device information parameters to be displayed.</display> 	
load	Loads firmware, such as Flash and RAM firmware.	 load <firmware file="" path=""> <erase type=""> <run< li=""> immediately:y n> <s n(optional)=""></s> Parameter description: <firmware file="" path="">: It sets the path of the to-be-downloaded firmware file.</firmware> <erase type="">: It sets the type to erase Flash, with the following options: sector: Erases Flash space occupied by boot info and firmware. chip: Erases all Flash spaces. none: Erases the firmware space only. </erase> <run immediately:y="" n="" ="">: It decides on whether to run the firmware immediately after downloading.</run> </run<></erase></firmware>	
help	Displays all help information.	help	

Take GR551x SoC as an example. The code below shows how to use the program command to download a firmware file to SoC Flash memories and run the firmware immediately after downloading. Command line:

GR5xxx_console.exe program "D:/test/test_fw.bin" y "0x01000000" 1024 0

The parameter descriptions are listed below:

- "D:/test/test_fw.bin": It indicates the path for the to-be-downloaded firmware BIN file
- "0x01000000" 1024 0: It represents the start address in Flash to which the firmware is downloaded (0x01000000), the Flash size (1024 KB), and SoC model (GR551x) respectively.

The downloading progress is displayed in real time during executing the program command.

Note:

You cannot operate *GR5xxx_console.exe* while GProgrammer is running.

3.11.2 GR5xxx_encrypt_signature.exe

Follow the steps below to run *GR5xxx_encrypt_signature.exe*:

- 1. Open the **Command Prompt** window from the **Start** menu or by entering **cmd** in the **Run** window.
- 2. Navigate to the GProgrammer installation directory by using cd command.
- Type GR5xxx_encrypt_signature.exe --parameter to complete corresponding operations.
 For most frequently used parameters, see Table 3-8. To view all parameters, enter GR5xxx_encrypt_signature.exe --help.

Parameter	Description	Remarks
operation	Indicates the operation type. Options:encryptandsign: Encrypt and sign firmware.sign: Sign firmware only	
firmware_key	Shows the directory of <i>firmware.key</i> , which is used for firmware encryption and signing, or signing only.	
signature_key	Shows the directory of <i>sign.key</i> , which is used for firmware encryption and signing, or signing only.	T he d'action of the
signature_pub_key	Shows the directory of <i>sign_pub.key</i> , which is used for firmware encryption and signing, or signing only.	The directories correspond to the paths you have set when you click Generate eFuse File in "Section 3.7.1 eFuse Settings".
product_json_path	Shows the directory of <i>product.json</i> , which is used for firmware encryption and signing, or signing only.	
rand_number	Shows the directory of <i>random.bin</i> , which is used for firmware encryption and signing, or signing only.	Jettings .
ori_firmware	Shows the directory that saves the firmware before encryption and signing, or signing only.	
output	Shows the directory that saves the firmware after encryption and signing, or signing only.	
Sets the start address in the Flash memories to which firmware files are downloaded. Value: • 0x01000000: for GR551x • 0x00200000: for GR5526		

Table 3-8 Frequently used parameters for *GR5xxx_encrypt_signature.exe*



Parameter	Description	Remarks
	Indicates the Flash size (unit: KB) of the selected SoC. For value details,	
flash size	see the Flash column in Figure 3-3.	
flash_size	Note:	
	For SoCs with 0 KB Flash, the external Flash size applies.	
	Indicates the SoC series. Valid value and description:	
product_type	• 0: GR551x	
	• 1: GR5526	
help	Displays help information.	

Take GR551x SoC as an example. The code below shows how to encrypt and sign firmware by using

GR5xxx_encrypt_signature.exe:

```
GR5xxx_encrypt_signature.exe --operation="encryptandsign" --firmware_key="D:/test/eFuse/
firmware.key" --signature_key="D:/test/eFuse/sign.key" --signature_pub_key="D:/test/
eFuse/sign_pub.key" --product_json_path="D:/test/eFuse/product.json" --rand_number="D:/
test/eFuse/random.bin" --ori_firmware="D:/test/firmware/test_fw.bin" --output="D:/
test/firmware_encryptAndSign/test_fw_encryptAndSign.bin" --random_output="D:/test/
firmware_encryptAndSign/random.bin" --base_addr="0x01000000" --flash_size="1024" --
product_type="0"
```

In the code snippet above, the **D:/test/eFuse/** directories show the user-defined folders where files are saved after users click **Generate eFuse File**, as described in "Section 3.7.1 eFuse Settings". For descriptions of other parameter, see Table 3-8.

- --ori_firmware="D:/test/firmware/test_fw.bin": the directory of the firmware before any operation
- --output="D:/test/firmware_encryptAndSign/test_fw_encryptAndSign.bin": the directory of the encrypted and signed firmware
- --base_addr="0x01000000" --flash_size="1024" --product_type="0": the start address in Flash to which the firmware is downloaded (0x01000000), the Flash size (1024 KB), and SoC model (GR551x) respectively
- --rand_number: Refer to "Section 3.7.3 Encrypt & Sign" for specific descriptions on "Random Number".

Run the command to encrypt and sign the firmware.

3.11.3 User-defined Windows Scripts

Users can also write custom scripts on Windows to call command-line programs. Two sample script files are provided in the GR5xxx_script file in the GProgrammer installation directory.

🛄 Note:

GR5xxx, representing the name of SoC series, includes GR551x and GR5526.

encryptAndSignatureFirmware.bat can encrypt and sign firmware with firmware_origin.bin in the same directory and the files saved in the eFuse directory. The encrypted and signed firmware is available in firmware_encryptAndS ign\firmware_encryptAndSign.bin.

GODIX

program_Firmware_EncryptAndSign.bat can erase all internal Flash memories, and download the firmware firmw are_encryptAndSign\firmware_encryptAndSign.bin and save the firmware file in the internal Flash memories.

3.12 Help

Click on the left side of the main interface of GProgrammer to open the **Help** interface.

GProgrammer offers help and support to users.

About GProgrammer

This section provides version information and features of GProgrammer.

• Feedback

If you have any questions or suggestions, please send an email to *software@reg.goodix.com*.

About Goodix

For more information, please visit Goodix official website: <u>www.goodix.com</u>.