AN14136

Enabling Serial Boot in S32G

Rev. 1.0 — 7 March 2024 **Application note**

Document information

Information	Content
Keywords	Serial boot, S32G
Abstract	This application note introduces the procedure of serial boot of NXP S32G2/S32G3 processors and provides a python script to enable non-secure serial boot from UART and FlexCAN interfaces.

1 Introduction

This application note introduces the procedure of serial boot of NXP S32G2/S32G3 processors and provides a python script to enable non-secure serial boot from UART and FlexCAN interfaces.

The script does not target optimization of the time of serial boot, rather it can be used to help the user to understand the mechanism and technical details behind serial boot. The user can use it as a reference to design their script.

The procedure of secure serial boot is also mentioned, but it is not the focus of this document, refer to [3] to get more detailed information about secure serial boot.

The following sections are covered in the document.

- Serial boot overview
- FlexCAN serial boot
- UART serial boot
- Serial boot log
- Enable UART/FlexCAN serial boot with python script

Note: *The attached script and binary are only for showcase purpose, not production grade.*

2 Serial boot overview

S32G2/S32G3 support two boot modes:

- Boot from external flash memory(Quads flash, SD, or MMC)
- Serial Boot mode(UART, FlexCAN)

Serial boot mode enables application code to be directly downloaded to SRAM for execution. It is typically used for development and factory line programming, especially when booting from external flash memory mode can not work, the serial boot mode can be used to write a new image into external flash to recover S32G.

2.1 Serial boot mode enablement

S32G enters serial boot mode under two situations:

- Fail-safe
- Configure specific combinations of FUSE_SEL and BOOTMOD

Errors may be encountered during BOOTROM execution, such as flash memory failure, PCB failure, authentication failure during secure boot and so on. BOOTROM supports a fail-safe mechanism to issue a functional reset in these error conditions. After the number of functional resets ≥8, then the BOOTROM will enter serial boot mode to download application via UART or FlexCAN interfaces.

The user could also configure the below configuration of FUSE_SEL and BOOTMOD to set S32G to enable serial boot:

Table 1. Serial boot mode configurations

Lifecycle	FUSE_SEL	BOOTMOD1	BOOTMOD2	Mode
CUST_DEL or OEM_PROD or IN_FILED	0	0	0	Serial boot + XOSC in Differential mode
	0	0	1	Serial boot + XOSC in Crystal mode or Bypass mode
	1	1	0	Serial boot

Note: *For serial boot, if S32G boot from fuse, DIS_SER_BOOT bit should not be set, the specific XOSC MODE setting in fuse could be referred to Fuse_Map_Tables in [1].*

2.2 Serial boot image structure

The serial boot image is downloaded by the serial download interfaces during the serial download stage. Compared with the normal boot image(SD, emmc, Qflash boot), the serial boot image doesn't have an IVT table, DCD/Self-Test DCD image. The serial boot image can be also distinguished into non-secure serial boot image and secure serial boot image. Compared with the non-secure serial boot image, the secure serial boot image contains additional items for authentication, refer to [2] to get more detailed information about the structure of secure serial boot image. The structures of the two types of serial boot image are shown below.

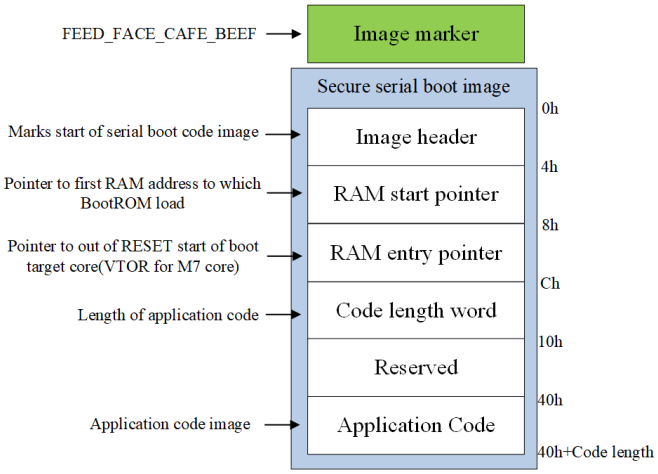


Figure 1. Non-secure serial boot image structure

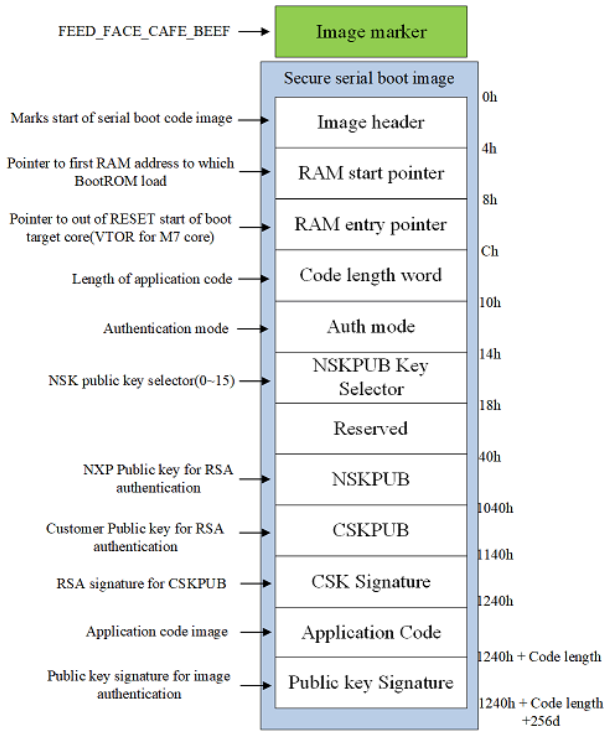


Figure 2. Secure serial boot image structure

2.3 Serial boot download

In the serial boot mode, the BOOTROM will program HSE_H SWT for 60-second timeout and continuously poll for activity on any of the available interfaces: FlexCAN, UART. The first detected active interface is used as the download interface, and then used to execute the serial download protocol procedure. If no activity is detected, a system reset will be issued by BOOTROM after the HSE SWT timeout.

2.3.1 Serial download protocol

The serial download protocol can be distinguished into two modes: non-secure serial download and secure serial download. When the lifecycle is advanced to OEM_PROD or IN_FILED stage, secure serial download mode is enforced.

Non-secure serial download protocol:

1. Initialization phase: BOOTROM configures the clock, interface module, watchdog, and so on, to perform a serial download.
2. Association phase: BOOTROM polls for receiving the FEED_FACE_CAFE_BEEFh image marker.
3. Non-secure header phase: BOOTROM reads the first 40 bytes of a serial boot image (RAM start pointer, entry pointer, code length).
4. Code download phase: BOOTROM downloads the application codes according to the code length.
5. Execution phase: After successful downloading of the entire serial boot image, Cortex_M7_0 core is released with reset vector address.

Secure serial download protocol:

1. Initialization phase: BOOTROM configures the clock, interface module, watchdog, and so on, to perform a serial download.
2. Association phase: BOOTROM polls for receiving the FEED_FACE_CAFE_BEEFh image marker.
3. No-secure header phase: BOOTROM reads the first 40 bytes of a serial boot image (RAM start pointer, entry pointer, code length, and so on).
4. Secure header phase: BOOTROM reads the next 1200h bytes of the image containing the NSK public keys, CSK Public key, and the CSK signature.
5. Code download phase: BOOTROM downloads the application codes according to the code length.
6. Public signature phase: BOOTROM reads the signature (256d bytes) of the image used for authentication of the image.
7. After successful verification of secure serial boot image, Cortex_M7_0 core is released with reset vector address.

3 FlexCAN serial boot

BOOTROM supports serial download from FlexCAN_0. Only Non-FD mode can be used.

3.1 FlexCAN serial boot configuration

FlexCAN_0 I/O configuration:

Table 2. FlexCAN_0 I/O

Signal	Pad Number
CAN RX	PAD [43]
CAN TX	PAD [44]

Clock configuration:

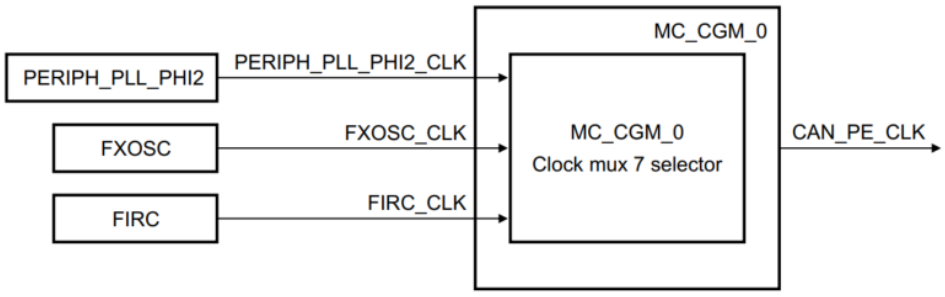


Figure 3. FlexCAN serial boot clock configuration

BOOTROM uses XOSC as the clock source, by default. If XOSC fails to initialize, FIRC is selected as the clock source instead.

Baud rate configuration:

Table 3. FlexCAN_0 I/O

Baud clock frequency	Baud rate
20 MHz(XOSC)	0.5 MBd
40 MHz(XOSC)	1 MBd

3.2 FlexCAN serial boot operation

The message IDs used by the BOOTROM for transfer of frames over the FlexCAN interface are defined, as shown in the following table.

Table 4. FlexCAN CAN message ID

Serial download phase	Rx MB ID	Tx MB ID	Non-secure serial boot	Secure serial boot
Association phase	11 h	01h	√	√
Non-secure header phase	12 h	02h	√	√
Secure header phase	13 h	03h		√
Code download phase	14 h	04h	√	√
Public signature phase	15 h	05h		√

Note:

1. *RX MB ID is the CAN ID of a CAN message, which is transmitted to the target SOC(BOOTROM).*
2. *Tx MB ID is the CAN ID of an echoed CAN message, which is transmitted by the BOOTROM.*
3. *The payload size of the message transmitted is restricted to 8 bytes.*

4 UART serial boot

BOOTROM supports serial boot from the LINFlexD_0 module, configured in UART mode.

4.1 UART serial boot configuration

UART external I/O configuration:

Table 5. UART external I/O

Signal	Pad Number
UART RX	PAD [42]
UART TX	PAD [41]

UART clock configuration:

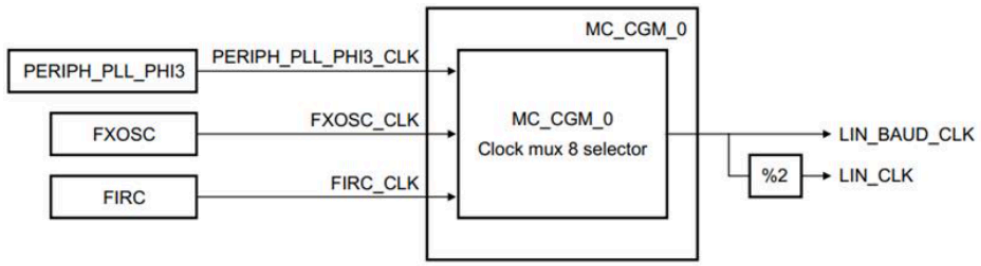


Figure 4. UART serial boot clock configuration

BOOTROM uses XOSC as a clock source for LIN_BAUD_CLK during boot. If XOSC fails to initialize, FIRC is selected as the MUX source instead.

UART baud rates:

Table 6. UART baud rate for S32G2

Baud clock frequency	Baud rate
20 MHz (XOSC)	24,000 Bd
40 MHz (XOSC)	48,000 Bd
44 MHz (FIRC)	48,000 Bd

Table 7. UART baud rate for S32G3

Baud clock frequency	Baud rate
20 MHz (XOSC)	57,600 Bd
40 MHz (XOSC)	115,200 Bd
44 MHz (FIRC)	115,200 Bd

Note: *Since the S32G3 uses higher baud rate, the user need to integrate some delay for each byte transmission during the association phase and non-secure header phase to leave the BOOTROM more time to parse the received packets.*

5 Serial boot log

During the serial download phase, the received packets are either echoed or not, determined by the NO_ECHO bit of code length word. For non-secure serial boot, if the NO_ECHO bit is set to 1, starting from the code download phase, received packets are not echoed. At the association phase and non-secure header phase, the received packets will still be echoed irrespetively.

The echoed received packets contain the same payload with the transmit packets. The user can receive the echoed packets to check if the communication with BOOTROM work fine. For example:

For FlexCAN non-secure serial boot:

At the association phase:

1. The Transceiver sends a CAN message to BOOTROM with FlexCAN_0:
   Can TX id: 0x11h CAN TX data: 0xFEEDFACECAFEBEEF
2. The transceiver receives echoed message from BOOTROM with FlexCAN_0:
   Can RX id: 0x1h Can RX data: 0xFEEDFACECAFEBEEF

For UART non-secure serial boot:

At the association phase:

1. The transceiver sends a marker to BOOTROM with LINFlexD_0:
   UART TX data: 0xFEEDFACECAFEBEEF
2. The transceiver receives an echoed message from BOOTROM with LINFlexD_0:
   UART RX data: 0xFEEDFACECAFEBEEF

6 Enable UART/FlexCAN serial boot with python script

The attached python script implements an enablement of non-secure serial boot on S32G-VNP-RDB2 and S32G-VNP-RDB3. The user could use the python script as a reference to build their project. More information about the procedure of secure serial boot can be referred to [3].

The attached script uses the PEAK-CAN probe as a USB-to-CAN device to transmit the CAN message from a PC. Please install the PEAK-CAN driver in the local PC before executing the attached script .

6.1 Hardware and software prerequisites

In this demo, the following hardware and software packages are used.

- S32G-VNP-RDB3 / S32G-VNP-RDB2
- PEAK-CAN probe
- UART cable line
- S32 design studio 3.4.3
- Tera term
- Python 3 and pyserial, python-can library

6.2 Generate a non-secure serial boot image

For installing S32 design studio, RTD package for S32DS, and build application image for S32G-VNP-RDB2 or S32G-VNP-RDB3, refer to [4] and [5]. This section focuses on how to add an image marker and non-secure header for application image. The attached prebuilt serial boot image - *'s32g3_non_secure_serial_boot_image.bin'* is an application demo, which lights up the RGB LED(U128) of S32G-VNP-RDB3.

To add an image marker and non-secure header for an application image, follow the steps as below:

1. Open S32 Design Studio and click *'ConfigTools'* and select `*'IVT'.*

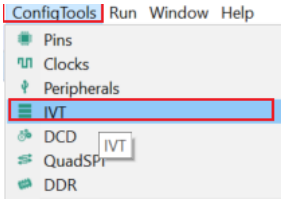


Figure 5. S32 Configuration tools

2. Choose the generated application image at the application bootloader column and fill in the RAM start pointer address and RAM entry pointer address.

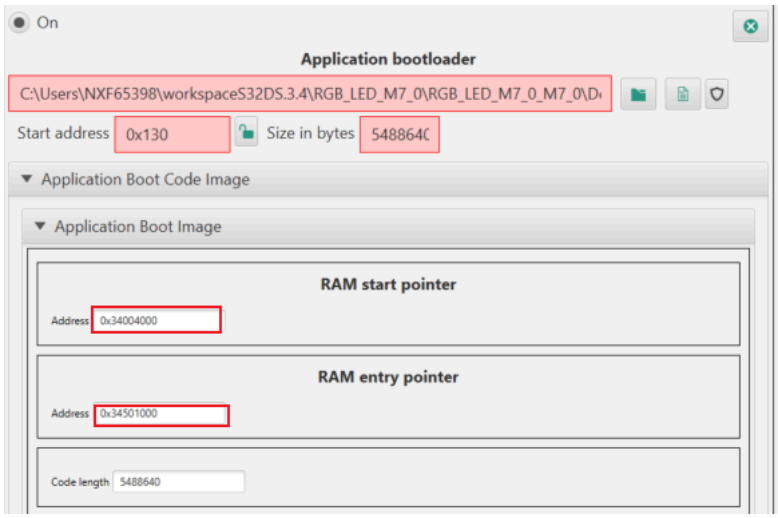


Figure 6. Application bootloader configuration column

3. Click *'Serial Boot'* and check the *'Serial Boot Image'* and *'Include Transmission Marker'* options.

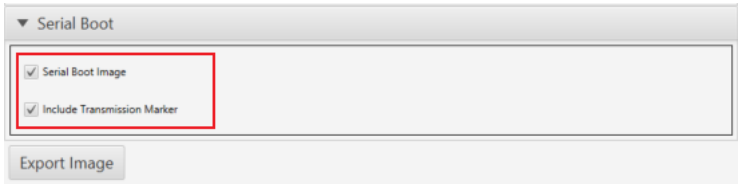


Figure 7. S32 Serial Boot configuration column A

4. Click *'Export Image'* and save the generated non-secure serial boot image.

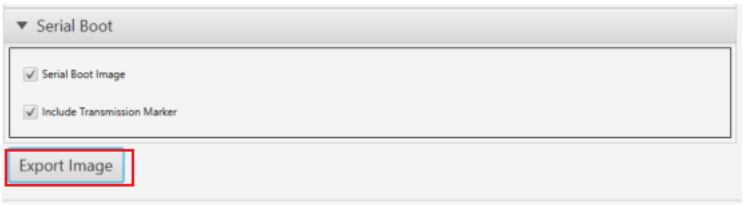
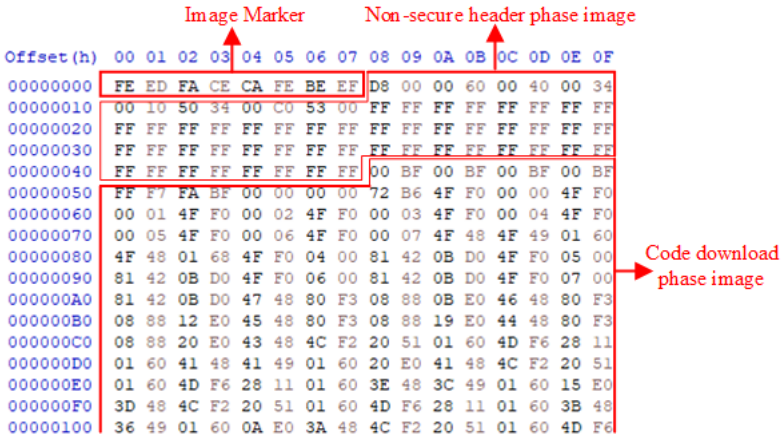


Figure 8. S32 Serial Boot configuration column B

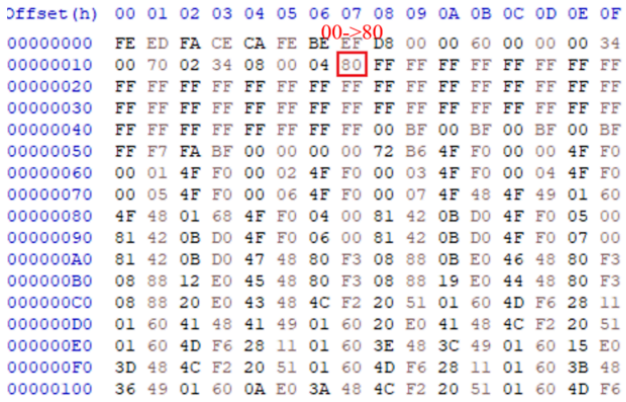
5. Open the generated non-secure serial boot image and check the content of the image as shown here exemplary.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	FE	ED	FA	CE	CA	FE	BE	EF	D8	00	00	60	00	40	00	34
00000010	00	10	50	34	00	C0	53	00	FF	FF	FF	FF	FF	FF	FF	FF
00000020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000040	FF	FF	FF	FF	FF	FF	FF	FF	00	BF	00	BF	00	BF	00	BF
00000050	FF	F7	FA	BF	00	00	00	00	72	B6	4F	F0	00	00	4F	F0
00000060	00	01	4F	F0	00	02	4F	F0	00	03	4F	F0	00	04	4F	F0
00000070	00	05	4F	F0	00	06	4F	F0	00	07	4F	48	4F	49	01	60
00000080	4F	48	01	68	4F	F0	04	00	81	42	0B	D0	4F	F0	05	00
00000090	81	42	0B	D0	4F	F0	06	00	81	42	0B	D0	4F	F0	07	00
000000A0	81	42	0B	D0	47	48	80	F3	08	88	0B	E0	46	48	80	F3
000000B0	08	88	12	E0	45	48	80	F3	08	88	19	E0	44	48	80	F3
000000C0	08	88	20	E0	43	48	4C	F2	20	51	01	60	4D	F6	28	11
000000D0	01	60	41	48	41	49	01	60	20	E0	41	48	4C	F2	20	51
000000E0	01	60	4D	F6	28	11	01	60	3E	48	3C	49	01	60	15	E0
000000F0	3D	48	4C	F2	20	51	01	60	4D	F6	28	11	01	60	3B	48
00000100	36	49	01	60	0A	E0	3A	48	4C	F2	20	51	01	60	4D	F6

Figure 9. Serial boot image content

6. (Optional) To improve the performance of serial boot, enable a non-echo bit of serial boot image. Disable non-echo bit may need to set more delay time during the application image download phase.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	FE	ED	FA	CE	CA	FE	BE	EF	D8	00	00	60	00	00	00	34
00000010	00	70	02	34	08	00	04	80	FF	FF	FF	FF	FF	FF	FF	FF
00000020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000040	FF	FF	FF	FF	FF	FF	FF	FF	00	BF	00	BF	00	BF	00	BF
00000050	FF	F7	FA	BF	00	00	00	00	72	B6	4F	F0	00	00	4F	F0
00000060	00	01	4F	F0	00	02	4F	F0	00	03	4F	F0	00	04	4F	F0
00000070	00	05	4F	F0	00	06	4F	F0	00	07	4F	48	4F	49	01	60
00000080	4F	48	01	68	4F	F0	04	00	81	42	0B	D0	4F	F0	05	00
00000090	81	42	0B	D0	4F	F0	06	00	81	42	0B	D0	4F	F0	07	00
000000A0	81	42	0B	D0	47	48	80	F3	08	88	0B	E0	46	48	80	F3
000000B0	08	88	12	E0	45	48	80	F3	08	88	19	E0	44	48	80	F3
000000C0	08	88	20	E0	43	48	4C	F2	20	51	01	60	4D	F6	28	11
000000D0	01	60	41	48	41	49	01	60	20	E0	41	48	4C	F2	20	51
000000E0	01	60	4D	F6	28	11	01	60	3E	48	3C	49	01	60	15	E0
000000F0	3D	48	4C	F2	20	51	01	60	4D	F6	28	11	01	60	3B	48
00000100	36	49	01	60	0A	E0	3A	48	4C	F2	20	51	01	60	4D	F6

Figure 10. Serial boot image with non-echo

Note: *Since the non-secure serial boot image generated by S32DS includes the image marker, the offset of each component of the serial boot image structure should add 8 bytes.*

6.3 Configure serial boot configuration Json file

The attached python script configures the runtime parameters of the CAN/UART interface using a json file. The user must edit the json file with the correct configuration before executing the python script. The architecture of the configuration json file is shown below.

```
{
    "SerBL_Cfg": {
        "SOC":{
            "S32G274A":{
                "config":  {"option":"n"}
            },
            "S32G399A":{
                "config":  {"option":"y"}
            }
        },
        "Download_Interface":{
            "UART":{
                "config": {"option":"y","baudRate": "115200","port":"COM15"}
            },
            "FlexCAN":{
                "config": {"option":"n", "baudRate":"1000000","port":"PCAN_USBBUS1"}
            }
        },
        "SerBL_Image":{
            "config": {"option":"y","path":"gpio-g3-serial_download.bin"}
        },
        "Echo_enable":{
            "config": {"option":"y"}
        }
    }
}
```

Figure 11. Serial boot configuration json file

Since the supported baud rate of the UART interface differs between S32G2 and S32G3, the user must ensure that the filled baud rate does not exceed the limitation of each soc. Refer to Table 6 and Table 7.

Echo_enable option is recommended for receiving and printing UART or CAN messages from BOOTROM during the association phase and non-secure boot header phase. It is used to check whether the communication between PC and BOOTROM of the test board works fine.

6.4 Enable non-secure serial boot with test board

Before executing the attached python script, ensure if the python3, pyserial, python-can are installed. Use the below commands to install pyserial and python-can library.

- *pip3 install pyserial*
- *pip3 install python-can*

Since the software and hardware environment may differ between users, it is recommended to adjust and fine tune the delay time in the script according to the user's actual situation, especially the delay time of 'downloading App_code_image' stage in the script. The user should focus on the serial boot enabling process in the script.

6.4.1 Enable non-secure serial boot with UART interface

1. Connect the PC with the UART0 port of S32G-VNP-RDB3 using the UART cable line.
2. Refer to Figure 14 to configure the DIP switch as serial boot mode for S32G-VNP-RDB3 and power on. Set SW11 = ON to power on the RGB LED (U128).
3. Modify the json file to configure UART interface node according to the actual situation. FlexCAN interface node should be disabled.
4. Execute '*python3 Enable_non_secure_serial_boot.py -j s32gxx_serbl.json*' on the PC terminal.
5. The terminal log should progress and complete as shown below.
6. Watching the S32G-VNP-RDB3 to check if the RGB LED lights up as expected.

```
$ python Enable_non_secure_serial_boot.py -j s32gxx_serbl.json
uptime library not available, timestamps are relative to boot time and not to Epoch UTC
INFO:root:download_interface: UART
INFO:root:uart_com_port: COM7
INFO:root:uart_baudrate: 115200
INFO:root:serial_boot_image_path: s32g3_non_secure_serial_boot_image.bin
INFO:root:echo_enable_flag: False
INFO:root:Serial Port: COM7 open OK
INFO:root:Downloading Image Marker ...
INFO:root:Downloading Non-secure header...
INFO:root:Downloading App_code_image ...
INFO:root:Done
INFO:root:Close Serial Port: COM7
INFO:root:Time cost: 27s
```

Figure 12. Enabling non-secure UART serial boot log

6.4.2 Enable non-secure serial boot with FlexCAN interface

1. Connect the PC with the LLCE_CAN0 cable of S32G-VNP-RDB3 using PCAN-Probe.
2. Configure the DIP switch as serial boot mode for S32G-VNP-RDB3 and power on. Refer to Figure 14 . Set SW11 = ON to power on the RGB LED (U128).
3. Modify the json file to configure FlexCAN interface node according to the actual situation. UART interface node should be disabled.
4. Execute '*python3 Enable_non_secure_serial_boot.py -j s32gxx_serbl.json*' on the PC terminal.
5. The terminal log should progress and complete as shown below.
6. Watching the S32G-VNP-RDB3 to check if the RGB LED lights up as expected.

```
$ python Enable_non_secure_serial_boot.py -j s32gxx_serbl.json
uptime library not available, timestamps are relative to boot time and not to Epoch UTC
INFO:root:download_interface: FlexCAN
INFO:root:pcan_channel: PCAN_USBBUS1
INFO:root:flexcan_baudrate: 1000000
INFO:root:serial_boot_image_path: s32g3_non_secure_serial_boot_image.bin
INFO:root:echo_enable_flag: False
INFO:root:flexcan channel: PCAN_USBBUS1 open OK
INFO:root:Downloading Image Marker ...
INFO:root:Downloading Non-secure header...
INFO:root:Downloading App_code_image ...
INFO:root:Done
INFO:root:Close flexcan channel: PCAN_USBBUS1
INFO:root:Time cost: 56s
```

Figure 13. Enabling non-secure FlexCAN serial boot log

Note: *FlexCAN serial boot uses PAD[43] and PAD[44] as FlexCAN_0 I/O. Since these pads are connected to LLCE_CAN0 cables on S32G-VNP-RDB3, so the PCAN-probe must connect with the LLCE_CAN0 cable, not the FlexCAN_0 cable.*

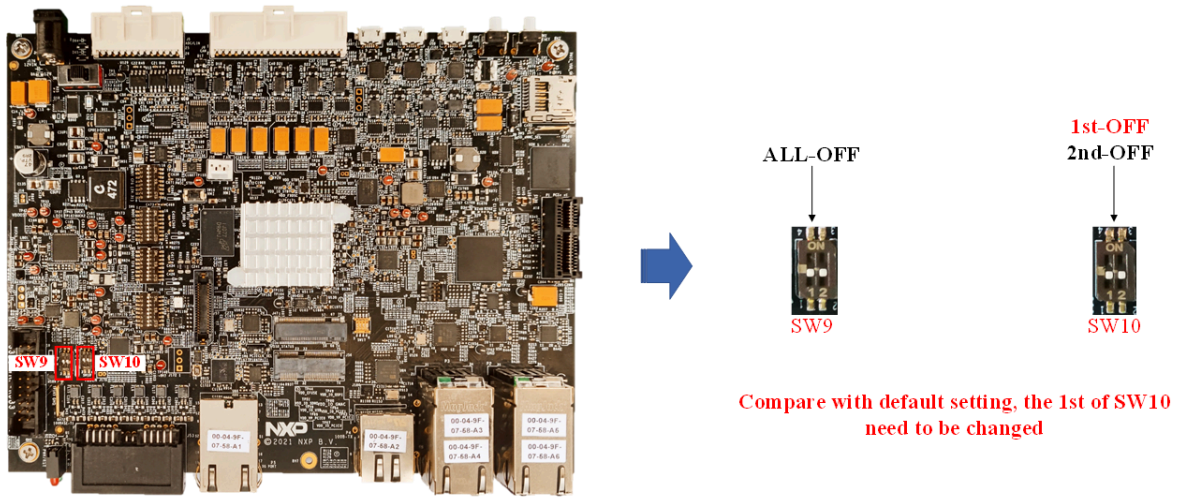


Figure 14. Serial boot DIP switch configuration

7 Reference documents

For further details, refer to the following documents.

Table 8. Reference documents

Serial number	Document name	Reference	Author(s)/Issuer(s)
• [1]	S32Gx Reference Manuals	Device-specific Reference Manual	NXP
• [2]	HSE firmware Reference Manual	Version.1.8 August 2023	NXP
• [3]	Serial bootloader authentication with HSE	Application Note AN13146	NXP
• [4]	S32G-VNP-RDB3 Software Enablement Guide	User guide	NXP
• [5]	S32G-VNP-RDB3 real time Driver Example Enablement Guide	User guide	NXP

8 Revision history

Table 9. Revision history

Document ID	Release date	Description
AN14136 v. 1.0	7 March 2024	Initial release

9 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: https://www.nxp.com

Date of release: 7 March 2024
Document identifier: AN14136