AN14156

S32G LLCE Quick sending CAN frame at wakeup

Rev. 1.0 — 21 February 2024 Application note

Document information

Information	Content
Keywords	S32G, LLCE, CAN
Abstract	Explaining quick LLCE CAN frame sending at the wakeup.

1 Introduction

When using a CAN protocol stack, it takes time to send the CAN frame at the wake-up from Standby mode. The boot time is long because of the large size of the CAN communication stack software. However, if your application must send the CAN frame quickly at the wake-up, LLCE and RAM boot can help. You can develop your own "mini LLCE FW" using LLCE FDK (Firmware Development Kit) to perform specific CAN communication. 32 kB Standby SRAM is implemented on S32G, which retains its value during Standby mode. Boot from Standby SRAM (RAM Boot) is faster than the normal boot from Flash (Full Boot). Therefore, if the mini LLCE FW and its bootloader are stored in the Standby SRAM, RAM boot can quickly send the CAN frame at the wake-up as shown in the following figure:

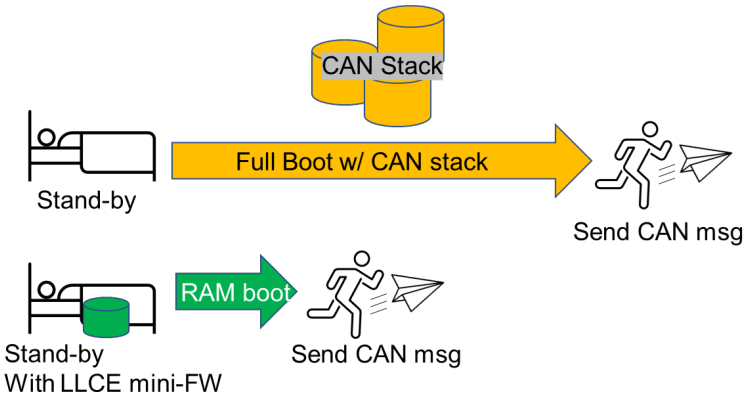

Figure 1. RAM boot with LLCE mini FW enables quick CAN frame sending at wake-up

This document introduces the PoC for such quick CAN frame sending at the wake-up from Standby mode.

2 RAM boot with LLCE mini FW

S32G's power domain has two power domains:

- RUN power domain
- STANDBY power domain

When in Standby mode, the power supply from the full regulator of PMIC to the RUN power domain is cut off. However, a small standby regulator provides a low-power supply to the STANDBY power domain as shown in Figure 2. The 32 KB STANDBY RAM retains the Interrupt Vector Table (IVT) and the application code even during Standby mode. S32G has a boot mode called RAM boot, which boots from the STANDBY RAM. RAM boot is faster than the usual Full boot from Flash. By creating a "Mini LLCE FW" and storing it in the STANDBY RAM, which is loaded to LLCE when wake-up, LLCE executes the code quickly after wake-up.

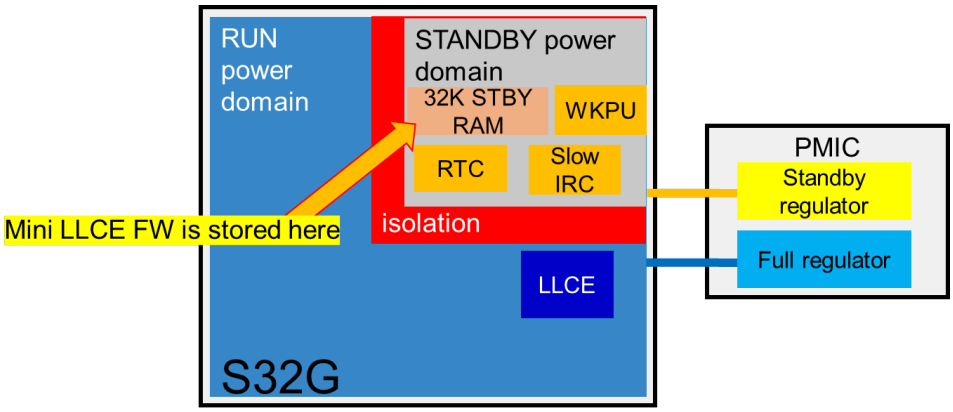

Figure 2. STANDBY power domain

3 Quick CAN frame broadcasting PoC

This chapter explains the PoC of the Quick CAN frame broadcasting. The PoC comprises of two S32DS projects, namely "S32G274Astandby mode" and "standbyramboot". Both codes are flashed in the NOR flash on RDB2. When S32G274Astandby mode is booted from NOR flash, it copies the code of standbyramboot from NOR flash to the Standby-RAM. Also, it creates the Interrupt vector table (IVT) and application header on the Standby-RAM to boot the standbyramboot from the Standby-RAM when waking up from Standby mode.

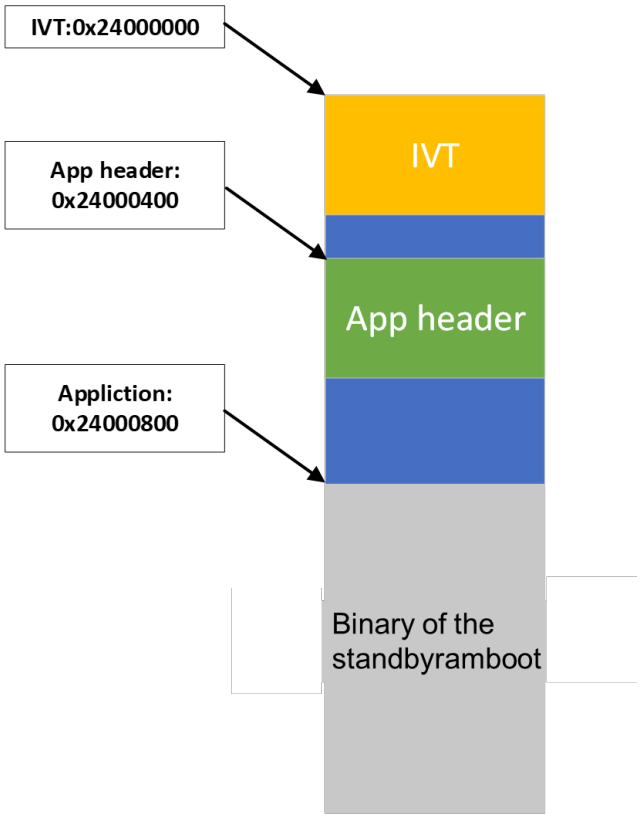

Figure 3. The memory map of Standby-RAM

Then, it brings S32G2 into the Standby mode activated RTC to expire approximately 5 seconds later. After that, the RTC wakes up the SoC, which then performs a RAM boot. The standbyramboot application is executed from Standby-RAM. The standbyramboot binary contains the tiny LLCE FW, which is created using the LLCE FDK. LLCE consists of 4 Cortex-M0+ cores that work in a coordinated multicore processing way. The normal LLCE FW is larger than the 32 KB Standby-RAM area, so it cannot be stored in the Standby-RAM. However, you can create your own LLCE FW using LLCE FDK. In this PoC, LLCE is only needed to send a specific CAN frame broadcasting, so the FW is created with a sufficiently small size to store into the Standby-RAM. It works only on one core of LLCE (that is, the TxPPE core) as shown in the following figure. The FW activates all CAN controllers and sends hard-coded classic CAN frames from odd CAN channels. After the successful TX completion on all those channels, it lights up an onboard LED.

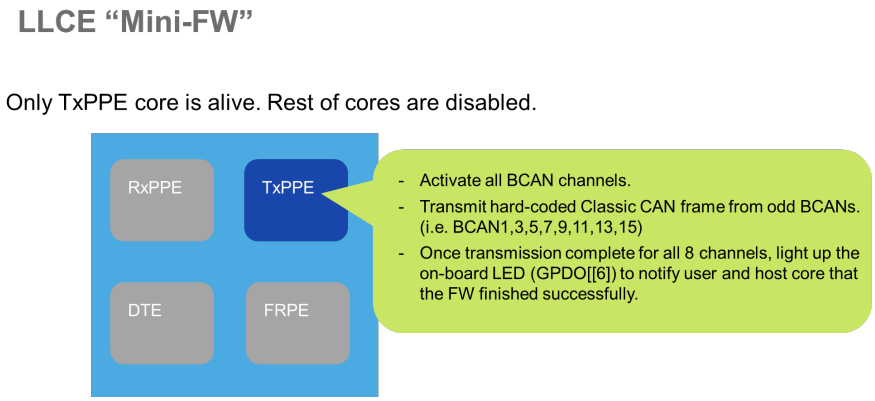

Figure 4. Mini LLCE FW

Upon wake-up, standbyramboot code is booted with RAM boot from the Stand-by RAM. The code copies the mini-FW into the LLCE RAM and starts the TxPPE core. Eventually, the TxPPE core can send the hard-coded CAN frame to the odd CAN channels quickly.

PoC uses S32G-VNP-RDB2 and the connection overview is as below. All LLCE CAN channels are connected between odd CAN channels and even CAN channels (that is, LLCE_CAN0 and LLCE_CAN1, LLCE_CAN2, LLCE_CAN3, and so on). Those CAN channels along with PMIC_STBY_MODE_B and PMIC_VDD_OK are connected to a logic analyzer to see the timing.

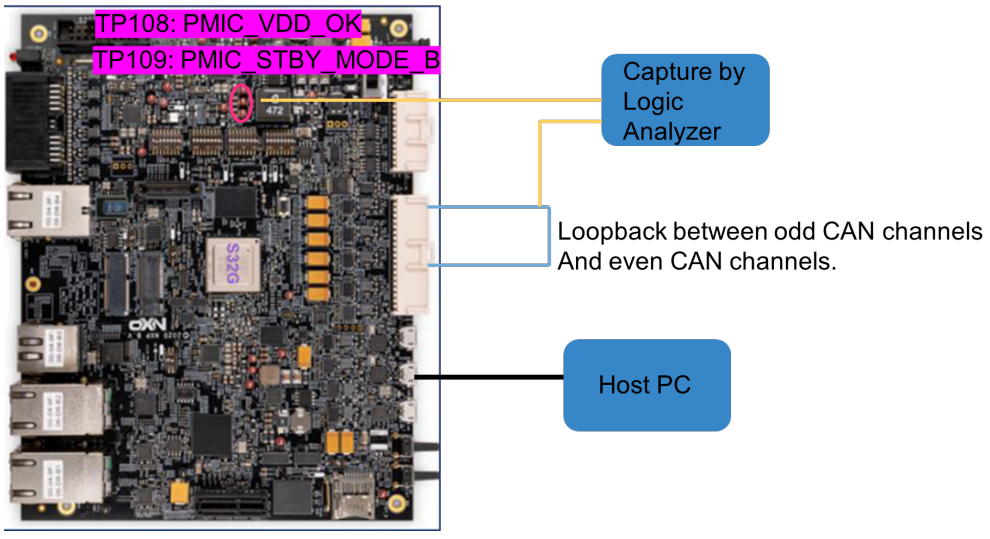

Figure 5. PoC Connection

4 Run the PoC and the result

The Standby-RAM contains the IVT, App header, and binary of the LLCE mini FW and the SoC enters Standby mode. Then, the RTC triggers a wake-up to the PMIC via the signal PMIC_STBY_MODE_B as follows. PMIC then ramps the power supply, which takes 3.75 ms, and asserts PMIC_VDD_OK. After that, the S32G boots the standbyramboot app from standby RAM. The standbyramboot app loads the mini FW code from the Standby RAM to the LLCE RAM and then boots it. The Mini FW then sends the CAN frames on all odd LLCE CAN channels. From the triggering PMIC wake-up to the beginning of sending CAN frames, it takes only 9.38 ms.

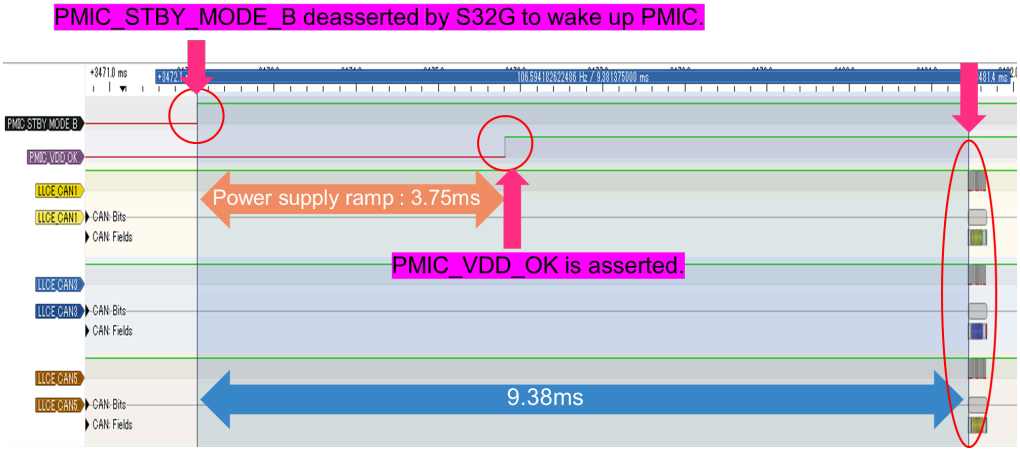

Figure 6. Broadcasting CAN frames in less than 10 ms at wake-up

5 How to play the PoC

Import both the projects (S32G274Astandbymode and standbyramboot) into S32DS and build the projects. Then a .bin file is generated.

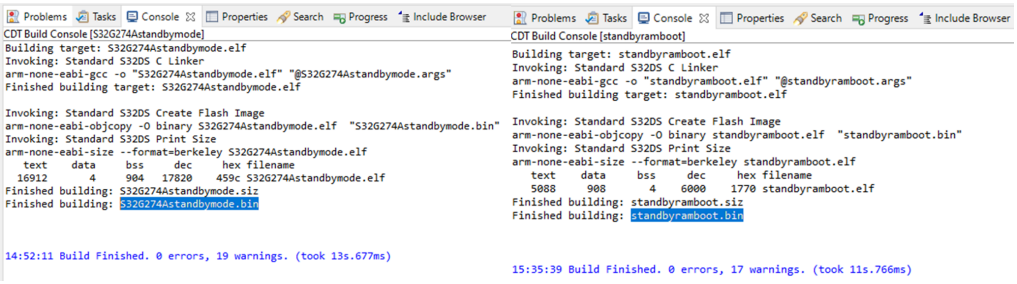

Figure 7. Build S32G274Astandbymode and standbyramboot

For S32G274Astandbymode, you must create a blob binary file that includes the IVT and DCD config using the IVT tool on S32DS. You can refer the following settings and also use a prepared DCD binary file ”DCD_SRAM_INIT.bin” to initialize SRAM. Or, you can skip those procedures if you use the prebuilt blob binary “S32G274Astandbymode_blobImage.bin” located in the S32G274Astandbymode folder for your convenience.

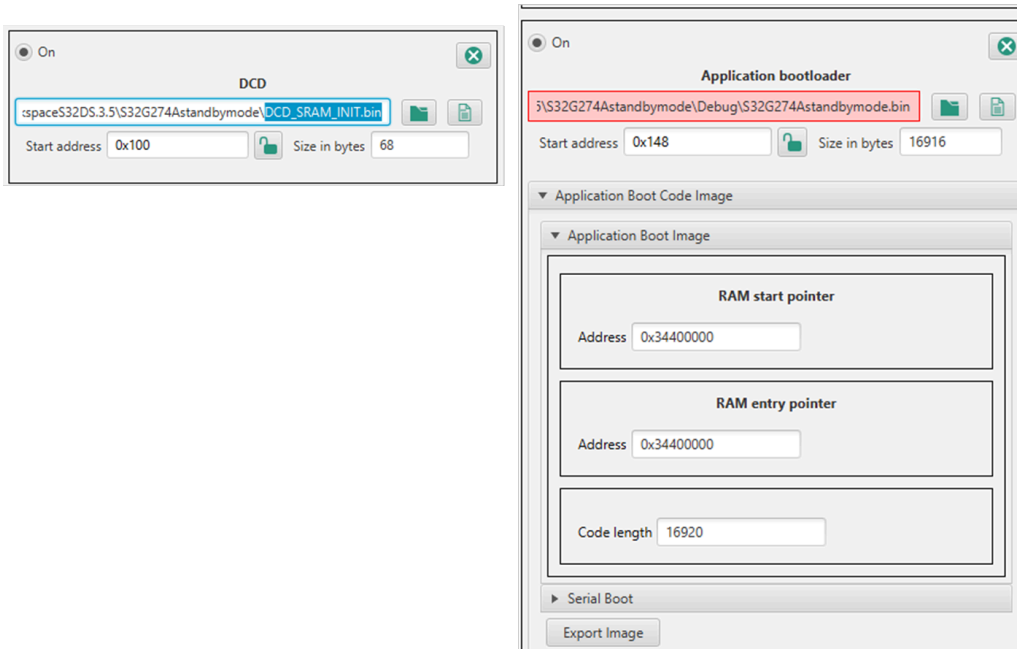

Figure 8. IVT tool config to create blob binary of S32G274Astandbymode

Then, flash the binaries into the NOR flash on the RDB2 as follows.

Step 1: Set S32G to Serial Boot Mode.

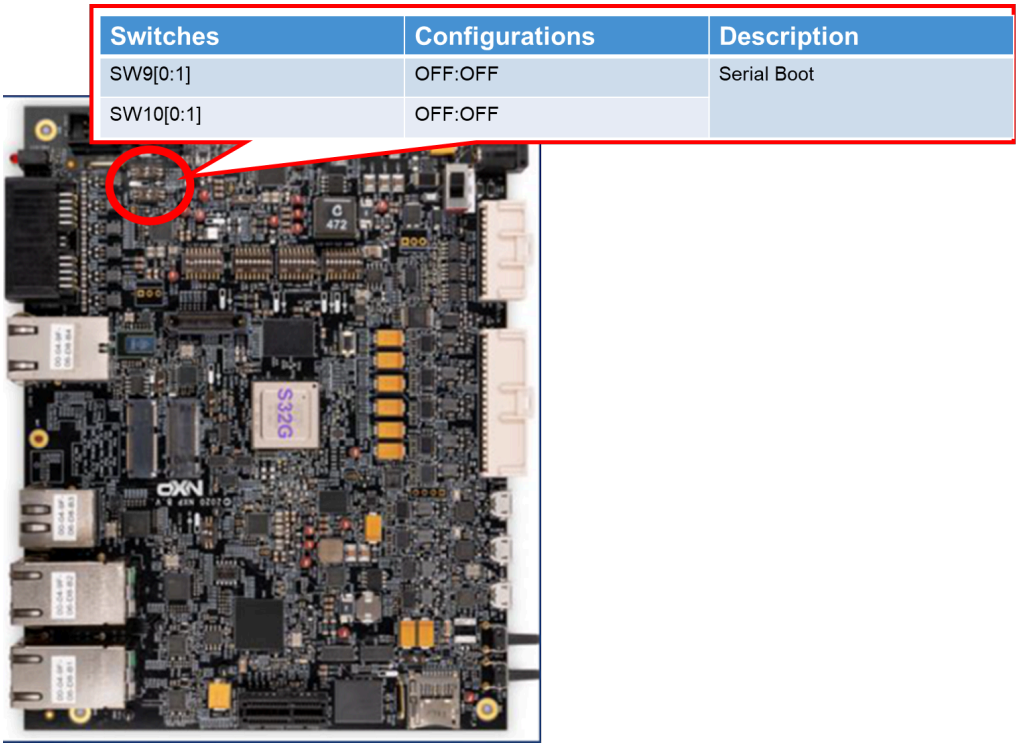

Figure 9. Set serial Boot Mode

Step 2: Connect your PC and RDB2 via USB (UART0).

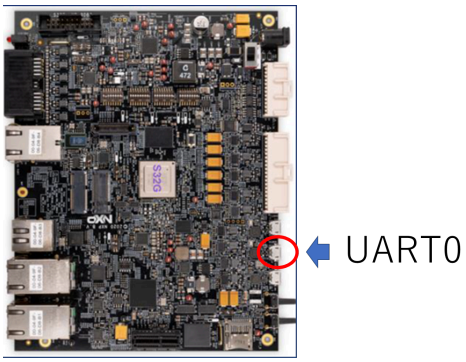

Figure 10. Connect your PC and RDB2 via USB (UART0)

Step 3: Run S32 Flash Tool (s32ft.exe).

S32ft.exe is found under the S32DS installed folder as shown in the following figure:

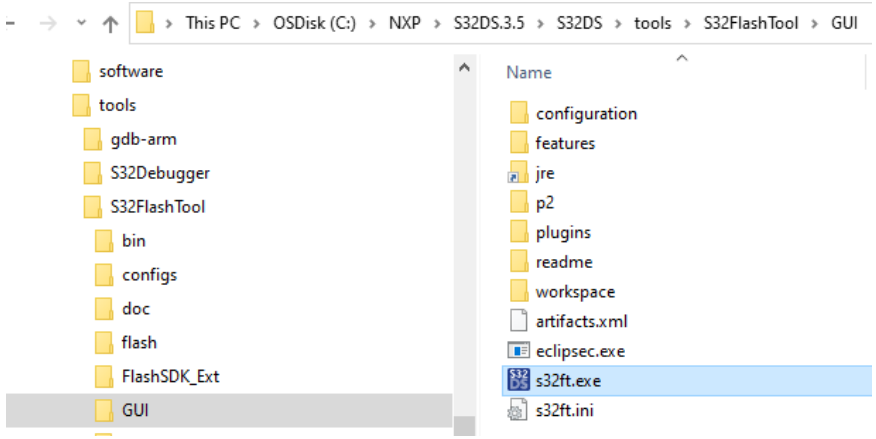

Figure 11. Flash Tool

Step 4: Fill the parameters in the Flash Tool.

Target: S32G274A

Algorithm: MX25UM51245G

COM: Check the com number of your PC for UART0.

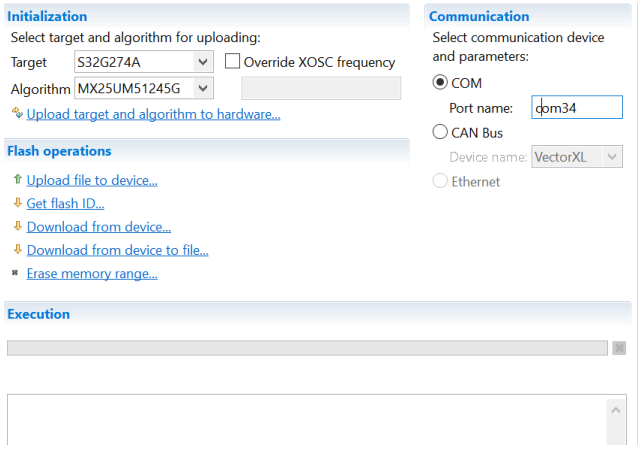

Figure 12. Flash Tool

Step 5: Initialize the hardware.

Click “Upload target and algorithm to hardware…”

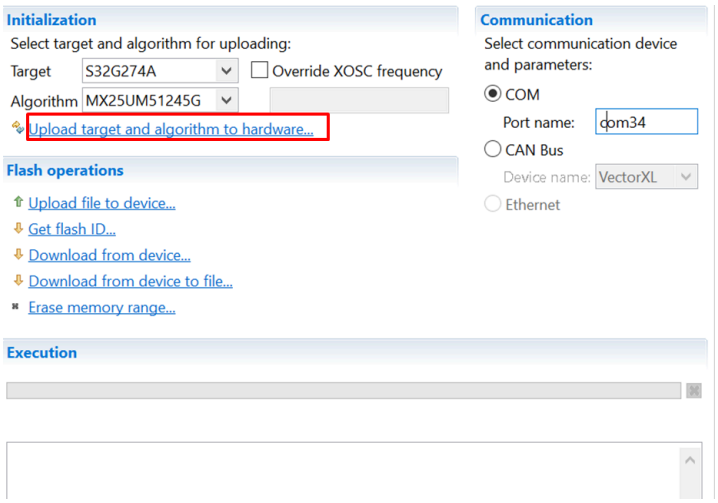

Figure 13. Initialize HW

Step 6: Erase memory range.

Click Erase memory range.

Fill the “memory range dialog box”.

Start Address: 0x00000

Size: 0x20000

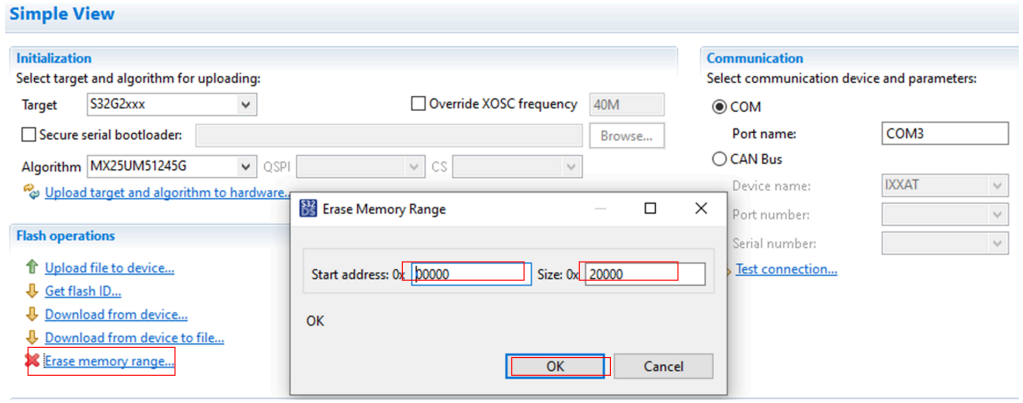

Figure 14. Erase memory range

Step 7: Upload S32G274Astandbymode binary to device.

Click “Upload file to device”.

Fill the Dialog box.

Start Address: 0x00000

Keep “Verify” checked.

File; Choose the S32G274Astandbymode blob image.

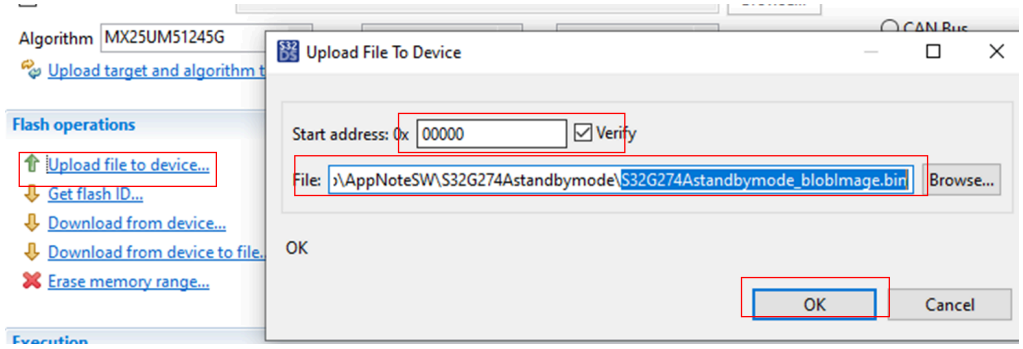

Figure 15. Upload S32G274Astandbymode binary

Step 8: Upload standbyramboot binary to device

Click “Upload file to device”.

Fill the Dialog box.

Start Address: 0x8000

Keep “Verify” checked.

File; Choose the standbyramboot binary.

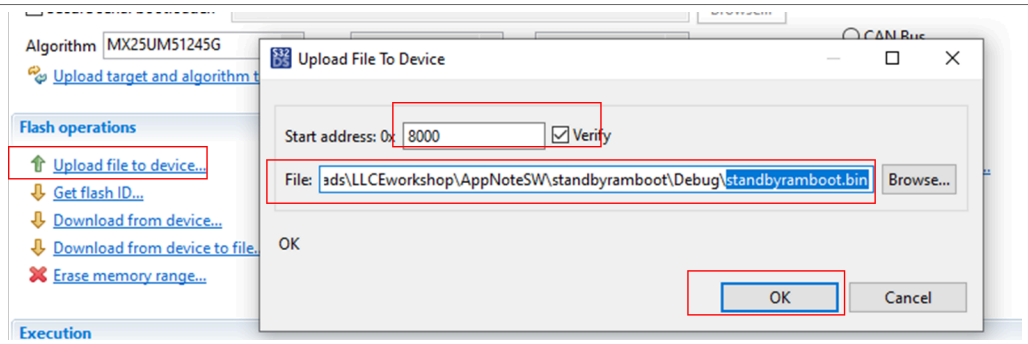

Figure 16. Upload standbyramboot binary

Step 9: Set the DIP switch as shown in the following figure:

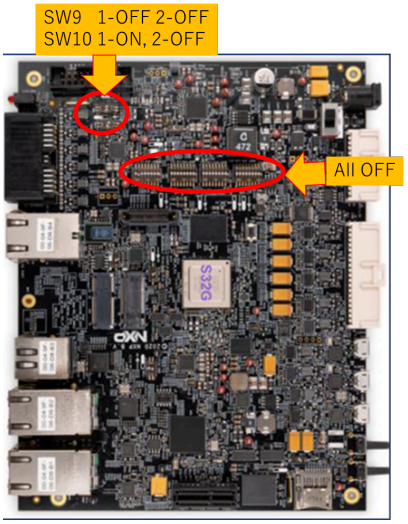

Figure 17. Set DipSW

Step 10: Close the Flash tool and open the terminal on the PC for UART0.

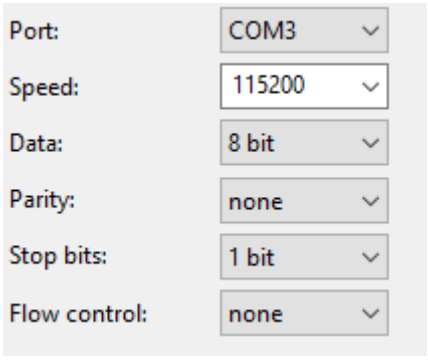

Figure 18. Terminal setting

Step 11: Ensure the connection as shown in the following figure and power on the board and then push any key to start on the terminal.

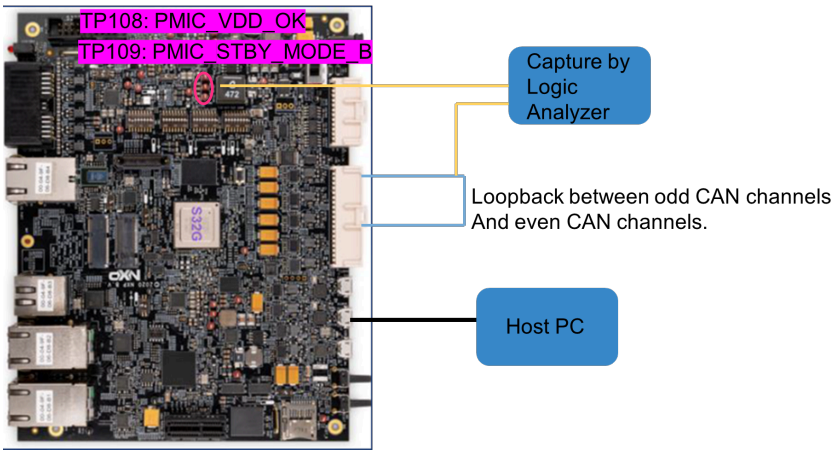

Figure 19. Connection

Approximately 5 seconds later, you see the waveform as previously described.

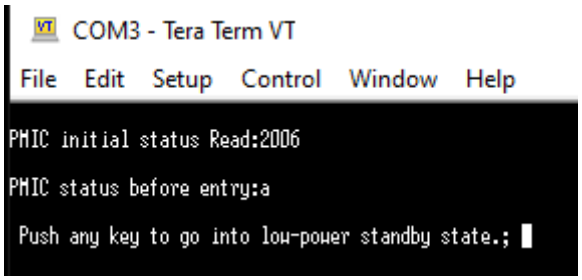

Figure 20. Push any key to enter the standby

6 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN14156 v.1.0	21 February 2024	Initial release

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Freescale — is a trademark of NXP B.V.

Tables

Figures

Contents

Date of release: 21 February 2024
Document identifier: AN14156