AN14602

IPMS Code Architecture to use CodeWarrior Debugger

Rev. 1.0 — 21 July 2025 Application note

Document information

Information	Content
Keywords	CodeWarrior, debug, IDE, IPMS, code
Abstract	This application note explains how to write a software project that can execute inside and outside a debug session.

1 Introduction

CodeWarrior IDE offers the possibility to debug projects. When the debug session is active, the user can place one breakpoint at a time to stop the program on specific instructions, view the content of the memory, check the value of variables, and so on.

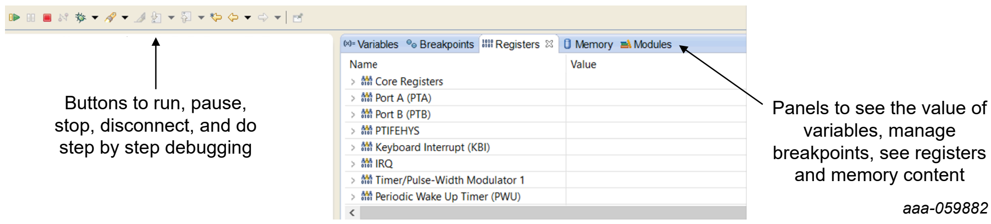

Figure 1. Debug session

However, several points must be considered when writing a software project to be able to debug it. If these points are not taken into account, the project will not work as expected while the debug session is active. The reason is that it is not possible to maintain a debug session while the device is in STOP1. STOP1 is a Deep-Sleep mode, so while a debug session is active, the MCU automatically enters STOP4 instead of STOP1. Entering STOP4 instead of STOP1 changes the application flow. This means the MCU behavior during a debug session is not the same as outside a debug session. It is possible to write the application main() in such way that the program flow is equivalent inside and outside a debug session. The following explains how this can be implemented.

First, the user must understand why the application flow is different when the MCU enters STOP4 instead of STOP1. Table 1 summarizes the differences.

Table 1. Differences between STOP4 and STOP1

	Exit from STOP	Interrupt vector of the interrupt that woke up the MCU from STOP	SPMSC2_PDF bit	SIMSES register
STOP4	The program jumps to the next instruction.	Is accessed as interrupts are enabled after STOP4 exit.	Is clear upon exit from STOP4.	Is unchanged upon exit from STOP4.
STOP1	The program takes the reset vector and goes back to the beginning of the main(). MCU and GPIOs must be configured again.	Is accessed after interrupts have been enabled in the application, if the block generating the interrupt is not reset upon STOP1 exit. Refer to the user manual for information specific to each block. ***Note:*** *The interrupts are disabled upon STOP1 exit.*	Is set upon exit from STOP1.	The corresponding bit of the interrupt that occurred is set upon exit from STOP1.

1.1 Behavior example inside and outside a debug session

The following is an example of main() having the same behavior inside and outside a debug session:

```
void main(void)
{
    vfnSetupMCU();
    vfnSetupGPIO();
    EnableInterrupts;
    if (CLEAR == SPMSC2_PDF)
        {
        /* Actions not performed after STOP1 exit (performed after POR, COP
 reset etc… */
        }
        for (;;) {

        _RESET_WATCHDOG();

        if ( ((IPMS_INTERRUPT_FLAG & LFR_INTERRUPT_FLAG) == LFR_INTERRUPT_FLAG)
 || (SET == SIMSES_LFF))
        {
        /* Actions on LF interrupt */
        }

        if ((SET == SIMSES_RFF) || (SET == RF_INTERRUPT_FLAG))
        {
        /* Actions on RF interrupt */
        }

        if ( ((IPMS_INTERRUPT_FLAG & PWU_INTERRUPT_FLAG) == PWU_INTERRUPT_FLAG)
 || (SET == SIMSES_PWUF))
        {
        /* Actions on PWU interrupt */
        }

        SPMSC2 |= SPMSC2_PDC_MASK;
        SPMSC2_PPDACK = SET;

        RF_INTERRUPT_FLAG = CLEAR;
        IPMS_INTERRUPT_FLAG = CLEAR;

        vfnSetSTOPMode(STOP1); /* If a debug session is active, STOP4 will
 automatically be entered */

        EnableInterrupts;

        asm STOP;

    } // End for loop
}
```

2 Code architecture

The following explains how this works in terms of code architecture and flags checked by the application.

Figure 2 shows the application main() flow inside and outside a debug session. The flow of the implemented code architecture is equivalent in both cases, meaning the project can be used inside or outside a debug session.

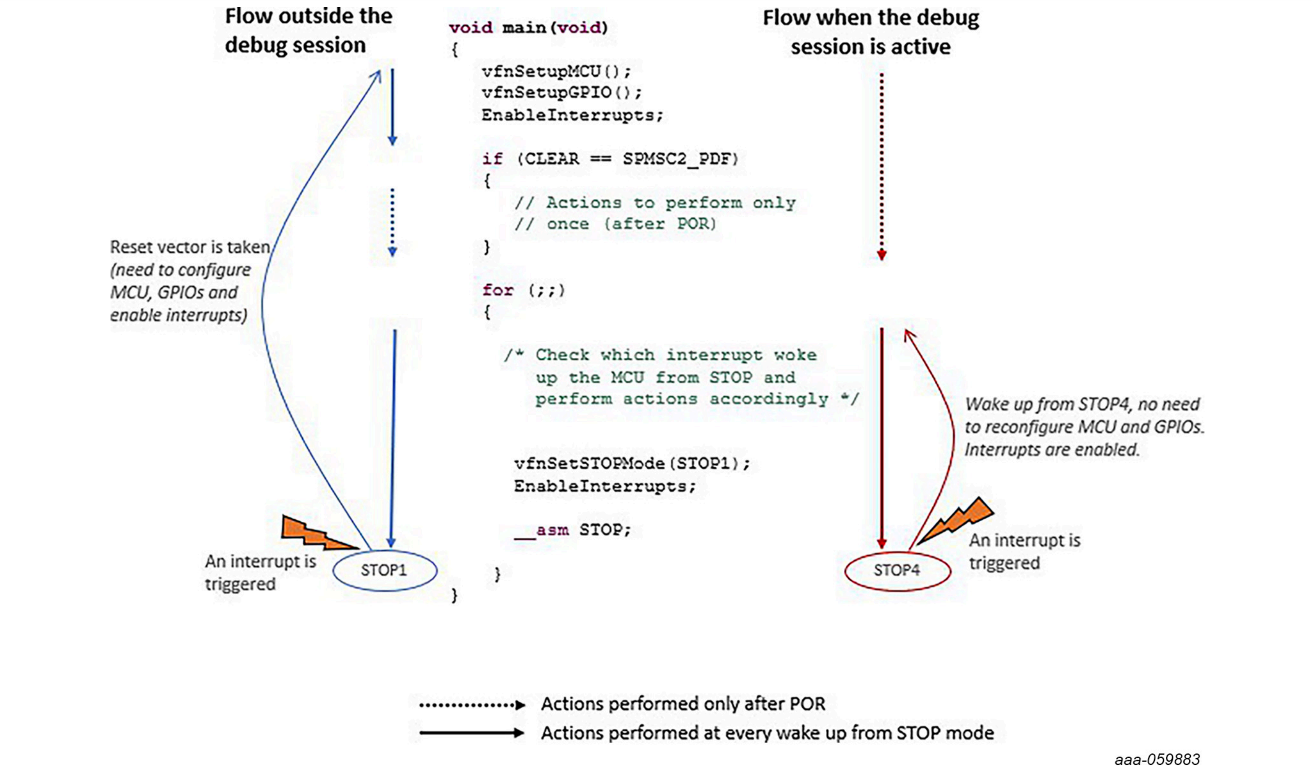

Figure 2. Application flow inside and outside a debug session

3 Flags checked by the application

In the infinite for loop, see Figure 2, inside the main() function, flags are checked to know which interrupt woke the MCU from Stop mode. The user must ensure that the flags checked are updated on exit from both stop modes to have an application working on exit from STOP1 or STOP4.

Table 2 describes the flags holding the information of which interrupt woke the MCU from stop.

Table 2. Flags updated upon STOP exit

Block generating the interrupt	Flags indicating which interrupt woke the MCU from STOP1	Flags indicating which interrupt woke the MCU from STOP4
PWU, LF	• SIMSES register flags. • IPMS_INTERRUPT_FLAG (if interrupts have been enabled)	• IPMS_INTERRUPT_FLAG.
KBI	• SIMSES register flags. (KBI block is reset upon STOP1 exit, which clears the hardware flag. As a result, the KBI interrupt vector is not access upon STOP1 exit).	• IPMS_INTERRUPT_FLAG.
RF	• SIMSES register flags.	Hardware flags are cleared in the interrupt vector when the interrupt is acknowledged. The user can create flags in the application to check them in the main().
Low-voltage detection, RTI	• IPMS_INTERRUPT_FLAG (if interrupts have been enabled).	• IPMS_INTERRUPT_FLAG.

For most interrupts, IPMS_INTERRUPT_FLAG can be checked.

Note: *This flag is updated inside the interrupt vectors, so upon exit from STOP1, interrupts must be enabled for the program to access the interrupt vector and update the flag.*

If interrupts are not enabled, IPMS_INTERRUPT_FLAG will not be updated upon exit from STOP1.

If STOP1 was exited on RF interrupt, the SIMSES register is updated and can be checked by the application. However, upon exit from STOP4, the hardware flag indicating that an RF interrupt occurred is cleared in the interrupt vector when the interrupt is acknowledged. As a result, the application cannot check the hardware flag once the interrupt vector has been exited. In that case, the application can create its own flag and update it to 1 in the RF interrupt vector. In the main() code given previously, the RF_INTERRUPT_FLAG has been created by the user, it is declared in the main() and set to 1 by the user application in the RF interrupt vector. In this way, the application can check the RF_INTERRUPT_FLAG to know if an RF interrupt woke the MCU from STOP4. It also works upon exit from STOP1 if interrupts are enabled at the beginning of the main() to allow the program to access the interrupt vector.

Note: *In the example code given, for each interrupt, both SIMSES register and IPMS_INTERRUPT_FLAG or user interrupt flag are checked. However, it is not compulsory to check both SIMSES and the interrupt flag. If interrupts are enabled right after a STOP1 exit, then only checking the interrupt flags is enough, as they will be updated when the interrupts are enabled.*

4 Note about the source code in this document

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

5 Revision history

Table 3. Revision history

Document ID	Release date	Description
AN14602 v.1.0	21 July 2025	Initial version

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

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.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

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.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

Tables

Figures

Contents