AN14743

S32M27 Analog to Digital Converter (ADC) Example in S32 Design Studio

Rev. 1.0 — 1 July 2025 Application note

Document information

Information	Content
Keywords	NXP, S32M27, S32 Design Studio, ADC, LLD, RTD
Abstract	Analog to Digital Converter (ADC) implementation using Low Level Drivers (LLD) and Real-Time Drivers (RTD) is a key feature for time-critical applications on S32M2 microcontrollers.The S32M2 microcontrollers from NXP integrates up to two SAR ADCs with three configurable channel groups—Precision, Standard, and External. It supports resolutions from 8 to 14 bits, delivering consistent 15-bit results. Designed for fast and accurate signal conversion, it achieves 1µs sampling at 80 MHz and up to 1M samples/sec, with ±6 LSB accuracy. Key features include multiple conversion modes, DMA and pre-sampling support, analog watchdogs, interrupt flags, calibration, interleaving, averaging, and self-test, making it ideal for automotive analog signal processing.

1 Introduction

The S32M2 microcontroller family from NXP is a highly integrated solution tailored for 12V motor control applications. Built on a system-in-package (SiP) design, it combines high-voltage analog components, such as MOSFET gate pre-drivers, LIN/CAN FD interfaces, and voltage regulators, with a robust embedded MCU core based on the Arm® Cortex®-M4 or M7. This architecture supports functional safety up to ISO 26262 ASIL B and enables advanced motor diagnostics, noise reduction algorithms, and seamless firmware-over-the-air (FOTA) updates through the S32 Automotive Platform.

Analog-to-Digital Converters (ADCs) are fundamental components in embedded systems, especially within the automotive domain, where they serve as the bridge between the physical world and digital control systems. In vehicles, ADCs are used to digitize signals from various sensors—such as temperature, pressure, speed, and position—allowing the microcontroller to interpret and respond to real-world conditions with precision and speed. This capability is essential for functions like engine control, battery management, driver assistance systems, and climate regulation.

The S32M2 microcontroller family from NXP, including the S32M27, is specifically designed for automotive applications, integrating high-performance SAR ADCs that support multiple resolution levels and conversion modes. These ADCs are optimized for fast sampling rates and high accuracy, enabling real-time data acquisition and processing. With features like programmable DMA, analog watchdogs, and hardware averaging, the S32M2xx ADCs provide robust and flexible solutions for demanding automotive environments. Their integration into the S32 Design Studio IDE, along with support for Low-Level Drivers (LLD) and Real-Time Drivers (RTD), facilitates efficient development and deployment of reliable, real-time embedded systems.

2 ADC Design

The S32M27 family of devices supports up to two instances of Analog-to-Digital Converters (ADCs), providing flexible and efficient data acquisition capabilities. The ADC channels are organized into three distinct groups: Precision, Standard, and External. Each group features independent configuration settings, tailored to meet varying accuracy and performance requirements. This design allows developers to optimize ADC behavior according to specific application needs. The configuration details for each ADC group are summarized in the following table.

Instance	S32M27x
ADC_0	Yes
ADC_1	Yes

Table 233. ADC configuration

Feature	ADC_0	ADC_1
No. of precision channels	8	8
No. of standard channels[1]	16	16
No. of special internal channels	1[2]	0
No. of external channels	32	32
BCTU trigger support	Yes	Yes
DMA support	Yes	Yes
Hardware interleaving	Yes	Yes
No. of watchdogs	4	4
No. of Hardware trigger	3	3

Figure 1. S32M27 ADC Instances

NXP S32M27 automotive microcontroller devices are equipped with a 14-bit resolution successive approximation Analog-to-Digital Converter (SAR ADC), designed for the precise acquisition and digitalization of analog input signals. Notably, the conversion result consistently yields a 15-bit wide output, even when the selected resolution is lower, ensuring enhanced accuracy and consistency in signal processing.

The SAR ADC in the S32MK27 microcontroller family supports selectable resolutions of 8, 10, 12, or 14 bits, yet consistently delivers a 15-bit wide conversion result regardless of the chosen resolution. It achieves high-speed performance with a sampling and conversion time of approximately 1 microsecond using an 80 MHz conversion clock, supporting rates up to 1 million samples per second.

The converter maintains an accuracy of ±6 LSB Total Unadjusted Error (TUE) and offers three conversion modes: Normal (one-shot and scan), Injected (one-shot only), and BCTU (single and list conversions). Each ADC channel is equipped with independent data registers and features programmable DMA and pre-sampling enable options.

Additional capabilities include configurable analog watchdogs for individual channels, distinct interrupt flags for end-of-conversion conditions (single, chain, or BCTU), watchdog threshold violations, and software-triggered calibration. The system also supports hardware interleaving, a hardware averaging function, and a built-in self-test for enhanced reliability.

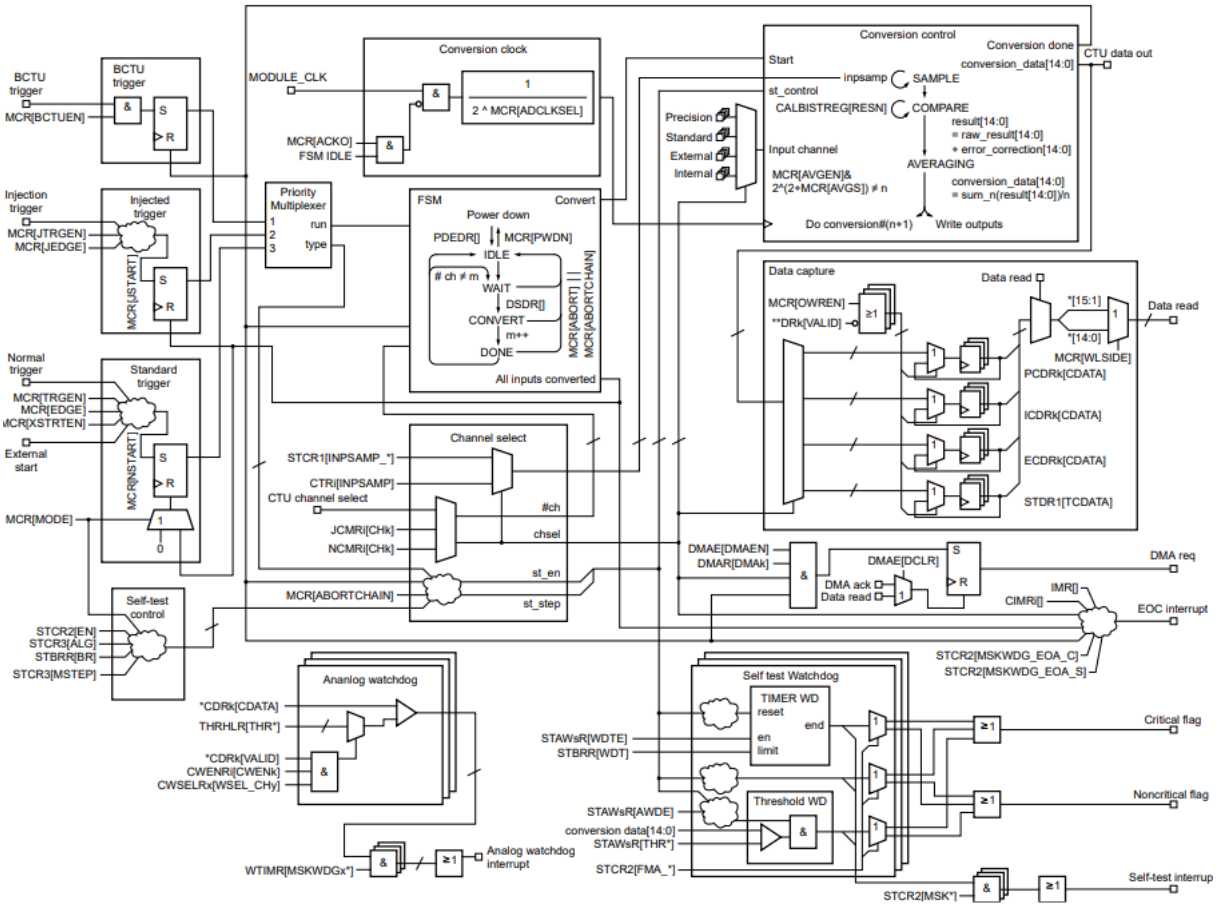

Figure 2. S32M27 Block Diagram

The ADC operates exclusively in Functional mode, with its behavior primarily governed by the Main Configuration Register (MCR). Most of the ADC's features are configured through this register, and adjustments to its field values should only be made when the ADC is in the Idle state—except for the ABORTCHAIN and ABORT fields, which may be modified at any time. To conserve power, the ADC can be placed into Power-Down mode by setting the MCR[PWDN] bit to 1. Additionally, clock signal gating can be achieved by setting MCR[ACKO] to 1, also while the ADC is in the Idle state. For the S32M27 family of devices the following DCM GPR bits have been used:

ADC channel	GPR_bit	Connected pad
ADC0 Standard channel 8 (ADC0 S8)	DCM.DCMRWF4[1]	0: GPIO PAD0 (default) 1: GPIO PAD45
ADC0 Standard channel 9 (ADC0 S9)	DCM.DCMRWF4[2]	0: GPIO PAD1 (default) 1: GPIO PAD46
ADC 1 standard channel 14 (ADC1 S14)	DCM.DCMRWF4[3]	0: GPIO PAD69 (default) 1: GPIO PAD32
ADC 1 standard channel 15 (ADC1 S15)	DCM.DCMRWF4[4]	0: GPIO PAD4 (default) 1: GPIO PAD33
ADC1 Standard channel 22 (ADC1 S22)	DCM.DCMRWF4[5]	0: GPIO PAD124 (default) 1: GPIO PAD145
ADC1 Standard channel 23 (ADC1 S23)	DCM.DCMRWF4[6]	0: GPIO PAD125 (default) 1: GPIO PAD146

Figure 3. S32M27 ADC Mux Mode

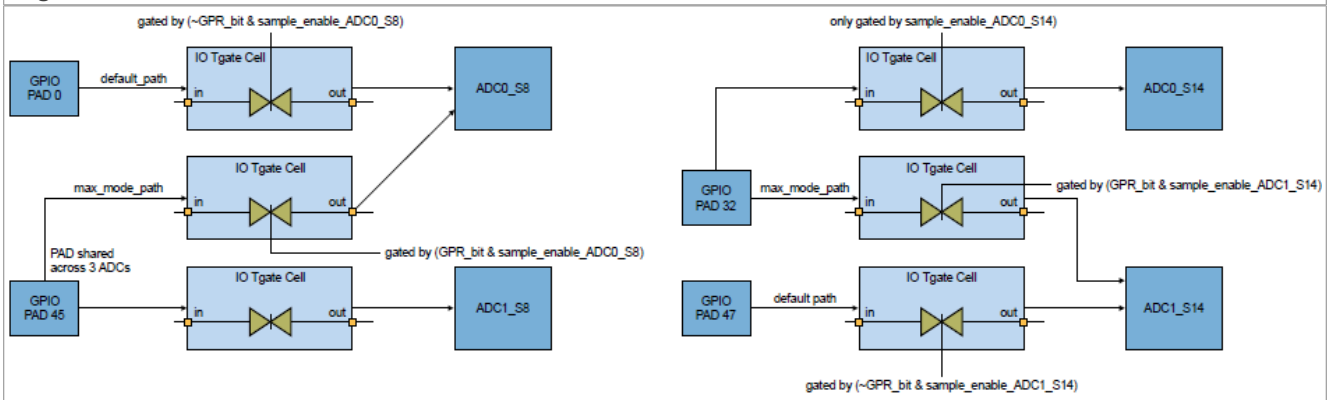

Figure 4. S32M27 ADC Channel Interleaving

The total conversion time of the ADC is directly influenced by the conversion clock frequency, designated as AD_clk. This frequency is configured through the MCR[ADCLKSEL] bit within the Main Configuration Register. The relationship between the clock configuration and the ADC's performance characteristics is illustrated in the accompanying figure, which depicts the ADC_n clocking setup for each MCU.

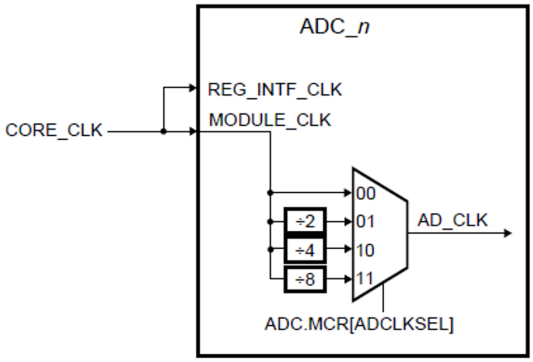

Figure 5. S32M27 ADC Clocking

S32M27 ADC Normal Conversion

To initiate a normal conversion in the ADC module, the Normal Conversion Mask Registers (NCMRn) and the Main Configuration Register (MCR) must first be properly configured. Once these settings are in place, a software-triggered conversion can begin by setting the MCR[NSTART] bit. Every conversion sequence includes both sampling and conversion phases. The ADC supports two operation modes: One-Shot mode (MODE = 0), where the sequential conversion is executed only once, and Scan mode (MODE = 1), which enables continuous sequential conversions.

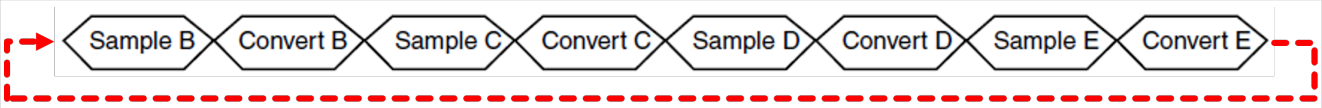

Figure 6. S32M27 ADC Normal Convertion Sequence

At the conclusion of each ADC conversion, the resulting value is stored in the corresponding register—PCDRn, ICDRn, or ECDRn—depending on the type of conversion performed. Simultaneously, the End of Conversion (EOC) flag is set, and, if enabled via the Interrupt Mask Register (IMR[MSKEOC]), an EOC interrupt is triggered. Once the final conversion in a defined chain is completed, the End of Chain (ECH) flag is asserted, and an ECH interrupt is issued if it has been enabled through IMR[MSKECH].

S32M27 ADC Injected Convertion

To initiate an injected conversion, the Injected Conversion Enable Inputs Registers (JCMRn) must be configured for the desired group—whether precision, standard, or external. Once set, a software-triggered conversion can be launched by setting the MCR[JSTART] bit. Injected conversions are restricted to insertion within an active Normal conversion chain and must operate in One-Shot mode exclusively. Upon completion of each injected conversion, an End of Injected Conversion (JEOC) interrupt is issued, while the conclusion of the entire injected chain triggers an End of Injected Chain (JECH) interrupt.

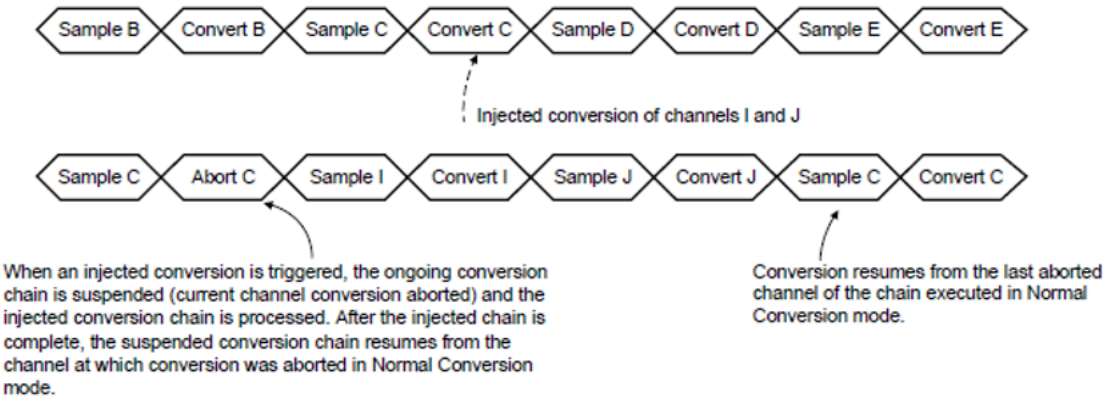

Figure 7. S32M27 ADC Injected Conversion Sequence

3 Code Description

This project configures the ADC driver with the RTD 4.0.0 P01.

The RTD interface is used for the peripheral configuration.

For the implementation of this application, a **S32M27XEVB-C064** will be used.

Function Description

This function set the clock configuration according to pre-defined structure.

```
Clock_Ip_StatusType Clock_Ip_Init(Clock_Ip_ClockConfigType const * Config)
```

This function enable a clock for a given peripheral.

```
void Clock_Ip_EnableModuleClock(Clock_Ip_NameType ClockName)
```

This function configures the pins with the options provided in the given structure.

```
Siul2_Port_Ip_PortStatusType Siul2_Port_Ip_Init(uint32 pinCount, const
 Siul2_Port_Ip_PinSettingsConfig config[])
```

This function initializes the configured interrupts at interrupt controller level.

```
IntCtrl_Ip_StatusType IntCtrl_Ip_Init(const IntCtrl_Ip_CtrlConfigType *pIntCtrlCtrlConfig)
```

This function configures the global functionalities of a BCTU instance.

```
void Bctu_Ip_Init(const uint32 u32Instance, const Bctu_Ip_ConfigType * const pConfig)
```

This function initializes the ADC_SAR module by configuring all available features.

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_Init(const uint32 u32Instance, const Adc_Sar_Ip_ConfigType * const
 pConfig)
```

This function sets the state of the Global Trigger Enable flag.

```
void Bctu_Ip_SetGlobalTriggerEn(const uint32 u32Instance, const boolean bState)
```

This function enables notifications selected by the given mask on the selected BCTU instance.

```
void Bctu_Ip_EnableNotifications(const uint32 u32Instance, const uint32 u32NotificationMask)
```

This function triggers a conversion or list of conversions associated with a trigger index.

```
void Bctu_Ip_SwTriggerConversion(const uint32 u32Instance, const uint8 u8TrigIdx)
```

This function returns the conversion result data stored in the BCTU result FIFO at the given index.

```
uint16 Bctu_Ip_GetFifoData(const uint32 u32Instance, const uint8 u8FifoIdx)
```

This function writes the given pin from a port, with the given value ('0' represents LOW, '1' represents HIGH).

```
void Siul2_Dio_Ip_WritePin(Siul2_Dio_Ip_GpioType * const base, Siul2_Dio_Ip_PinsChannelType pin,
 Siul2_Dio_Ip_PinsLevelType value)
```

Project Creation

1. Download and open S32 Design Studio for S32 Platform.
2. Download the S32M2xx Development Package version 3.6.0 from S32DS:
   a. Select Help > S32DS Extensions and Updates > S32M2xx development package
   b. Click on Install button and follow the instructions.
3. Select File > New > S32DS Application Project.
4. Write a project name without spaces. For example: *S32M2xx_ADC_Example*.
5. Open folder *Family S32M2xx*, select S32M276 in the *Processors* section and click next.

6. Click on the three dots (…) to select a Software Development Kit (SDK), click ok and click finish.

Pins ConfigTool

1. Expand your project folder in the *Project Explorer* view.
2. Double click on the <Project name>.mex file.
3. Click on the *Pins* view on the toolbar if not already selected.
4. In the pins tab, type PTD15 in the search bar and select the check box on the corresponding pin name to open pin alternatives.
5. Select alternative *SIUL2:gpio,111* > *Output* > *OK* > *Done*.
6. Search PTD16 by repeating step 4 and select *SIUL2:gpio,112* > *Output* > *OK* > *Done*
7. Search PTE6 by repeating step 4 and select *ADC1:adc1_p6* > *Done*
8. (Optional) In the Routing Details tab, is recommended to give a meaningful identifier to the pin and configure electrical characteristics as needed.

Peripherals ConfigTool

1. Click on the *Peripherals* view near the *Pins* view in the toolbar.
2. In the components tab at the left side, click the Drivers plus "+" button.
3. Select "All" in the components that should be offered in the top center.
4. Type *Bctu_Ip* in the search bar and doble click to add the component.
   a. Select BCTU_FIFO1 in Data Destination field.
   b. Disable field in Enable HW Triggering field.
   c. Select List in BCTU ADC Command List Operating field.
   d. Type "2" in ADC Target Mask field.
   e. Add 3 items in Bctu LIST items.
   f. Select P6_ChanNum6 in ADC Channel ID.
   g. Select VREFH_ChanNum55 in ADC Channel ID.
   h. Select VREFL_ChanNum54 in ADC Channel ID and enable Last channel check box.
   i. Add an item in Result FIFOs.
   j. Enable interrupt notifications check box.
   k. Type "*Bctu_Notification*" at Watermark notification field.
5. Add *Adc_Sar_Ip* component by repeating steps 2 to 4.
   a. Open Adc_Sar_Ip component by doble click.
   b. Select ADC1 in Adc Hardware Unit field.
   c. Select Scan in Adc Conversion Mode field.
   d. Select Trigger Mode in Adc Ctu mode tab.
   e. Select P6_ChanNum6 in Adc Physical Channel Name of Channel configurations array section.
6. *Add Siul2_Dio* component by repeating steps 2 to 4.
7. *Add Siul2_Port* component by repeating steps 2 to 4.
8. Open *Siul2_Port*.
   a. Click on *PortConfigSet* tab.
   b. In *PortPin Mscr* type "111".
   c. Click on the plus "+" button to add a new port pin.
   d. In *PortPin Mscr* type "112".
   e. Click on the plus "+" button to add a new port pin.
   f. In *PortPin Mscr* type "134".
9. Add *IntCtrl_Ip* component by repeating steps 2 to 4
10. Open *IntCtrl_Ip*.
    a. Click on *General Configuration* tab.

b. Disable *Development errors detection* check box.
c. Click on *Interrupt Controller* tab.
d. Add an item by clicking the plus "+" button.
e. Add a new interrupt by clicking the plus "+" button in *PlatformIsrConfig*.
f. For this example, select interrupt name *BCTU_IRQn*.
g. Click on the Interrupt Enabled check box.
h. In *Priority* type "3".
i. In *Handler* type "Bctu_0_Isr".

11. Update the code by clicking *Update Code* button in the toolbar and click OK.

Main Code

1. To go back to the main file click on *S32DS C/C++* in the toolbar.
2. Replace the auto-generated code with the following:

```
/* Including necessary configuration files. */
#include "Clock_Ip.h"
#include "IntCtrl_Ip.h"
#include "Siul2_Port_Ip.h"
#include "Siul2_Dio_Ip.h"
#include "Adc_Sar_Ip.h"
#include "Bctu_Ip.h"

#define clockConfig      &Clock_Ip_aClockConfig[0]
#define BCTU_Instance_0  (0U)
#define ADC_Instance_1   (1U)

volatile uint8_t flag = 0;

volatile uint16_t data[3] = {0, 0, 0};
volatile uint8_t index = 0;

void Bctu_Notification(void)
{
    for(index = 0; index < 3; index++)
    {
        data[index] = Bctu_Ip_GetFifoData(BCTU_Instance_0, 0U);
    }
    flag = 1;
}

int main(void)
{
    /* Initial Clock */
    Clock_Ip_Init(clockConfig);
    Clock_Ip_EnableModuleClock(ADC1_CLK);
    Clock_Ip_EnableModuleClock(BCTU0_CLK);

    /* Initial Pin */
    Siul2_Port_Ip_Init(NUM_OF_CONFIGURED_PINS_PortContainer_0_BOARD_InitPeripherals,
    g_pin_mux_InitConfigArr_PortContainer_0_BOARD_InitPeripherals);

    /*Initial ISR*/
    IntCtrl_Ip_Init(&IntCtrlConfig_0);

    /* BCTU */
    Bctu_Ip_Init(BCTU_Instance_0, &BctuHwUnit_0);

    /* ADC */
    Adc_Sar_Ip_Init(ADC_Instance_1, &AdcHwUnit_0);

    /* Notification and Trigger of BCTU */
    Bctu_Ip_SetGlobalTriggerEn(BCTU_Instance_0, 1U);
    Bctu_Ip_EnableNotifications(BCTU_Instance_0, BCTU_IP_NOTIF_FIFO1);
    Bctu_Ip_SwTriggerConversion(BCTU_Instance_0, 0U);

    for(;;)
    {
        if(1U == flag)
```

```
        {
            flag = 0;
            if(4095 >= data[0])
            {
                Siul2_Dio_Ip_WritePin(LED1_PORT, LED1_PIN, 0);
                Siul2_Dio_Ip_WritePin(LED2_PORT, LED2_PIN, 0);
            }
            else if(8191 >= data[0])
            {
                Siul2_Dio_Ip_WritePin(LED1_PORT, LED1_PIN, 0);
                Siul2_Dio_Ip_WritePin(LED2_PORT, LED2_PIN, 1);
            }
            else if(12287 >= data[0])
            {
                Siul2_Dio_Ip_WritePin(LED1_PORT, LED1_PIN, 1);
                Siul2_Dio_Ip_WritePin(LED2_PORT, LED2_PIN, 0);
            }
            else
            {
                Siul2_Dio_Ip_WritePin(LED1_PORT, LED1_PIN, 1);
                Siul2_Dio_Ip_WritePin(LED2_PORT, LED2_PIN, 1);
            }
            Bctu_Ip_SwTriggerConversion(BCTU_Instance_0, 0U);
        }
    }
}
```

3. Click the hammer icon in the toolbar or "Ctrl+b" to build the project.
4. Connect the board.
5. Click the bug icon in the toolbar to debugg the project as the standard debug configuration.
6. (Optional) Check if Debug Configurations are as follows:
   a. Click on the arrow next to the debug icon.
   b. Select Debug Configurations.
   c. Expand GDB PEMicro Interface Debugging.
   d. Select <project name>_Debug_FLASH_PNE.
   e. Click on the *PEmicro Debugger* tab and review the options in the figure.
   f. Click on Apply and then Debug.
7. Click the play icon in the toolbar to run the application.

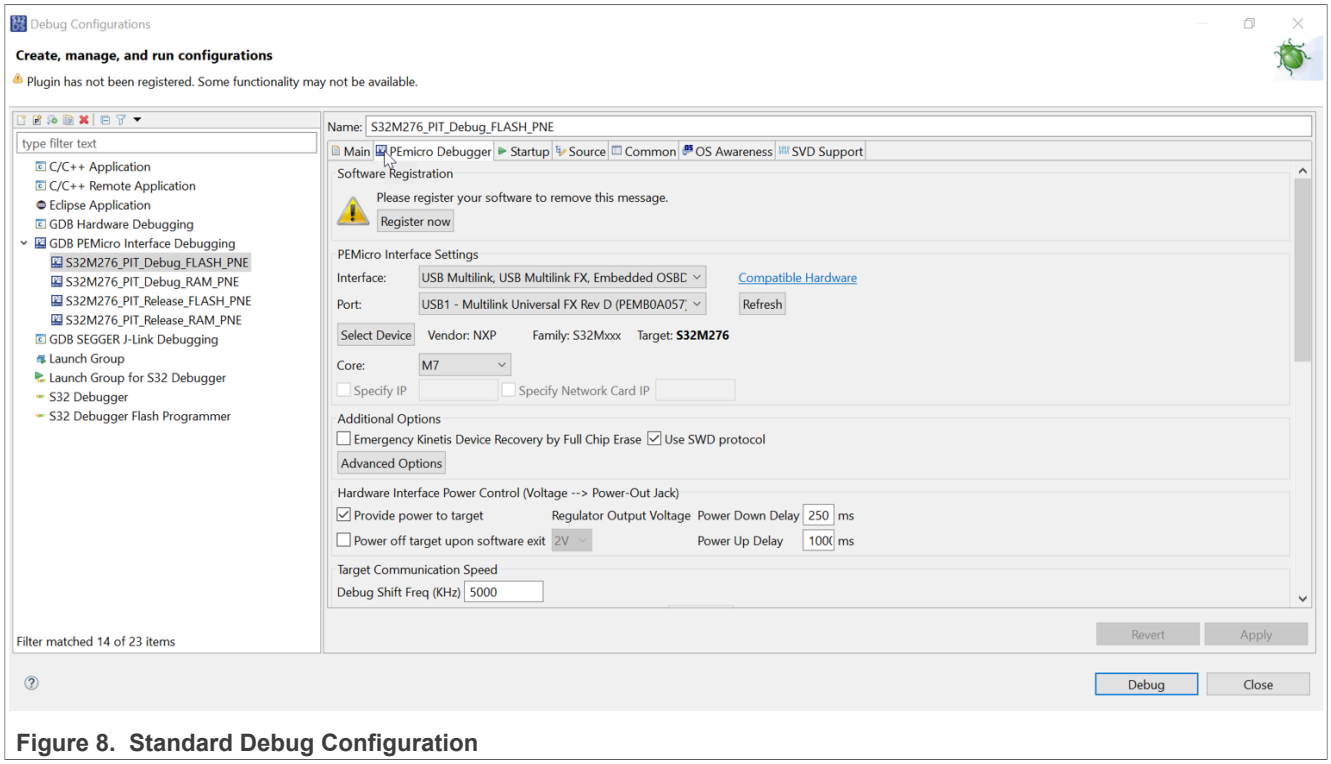

Figure 8. Standard Debug Configuration

4 Conclusion

In conclusion, the ADC module in the NXP S32MK27x family of automotive microcontrollers delivers a high-performance, flexible, and feature-rich solution for analog signal acquisition and digital conversion. With its configurable resolution, rapid conversion rates, and multi-mode operation—including Normal, Injected, and BCTU conversions—it supports a wide range of application requirements. The comprehensive control provided by registers such as MCR, NCMRn, and JCMRn allows for precise configuration, while the intelligent interrupt system and diagnostic features like self-test and watchdogs enhance both responsiveness and reliability. Overall, this ADC architecture is engineered to offer robust performance, optimized power management, and extensive configurability, making it well-suited for demanding automotive environments.

5 Note about the source code in the document

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

Copyright 2025 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.

6 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN14743 v.1.0	01 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.

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.

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.

Contents