

Porting Guide

N32G45x Series to N32H47x_48x Series

Introduction

This document primarily describes the differences in resources, hardware, and software between the N32G45x series and the N32H47x_48x series, facilitating a rapid transition from the N32G45x series to the N32H47x_48x series for development purposes.

Content

1.	Overview	4
2.	Resource Differences	5
2.1	Comparison between N32G45x and N32H47x	5
2.2	Comparison between N32G45x and N32H48x	6
2.3	Comparison between N32H47x and N32H48x	7
3	Hardware Differences	8
3.1	Power Supply	8
3.2	Boot Pins	8
3.3	ADC Converter	8
3.4	IO Voltage Tolerance	8
3.5	Operational Amplifiers	8
3.6	TSC Touch Sensor	9
3.7	Super High-Resolution Timer (SHRTIM)	9
3.8	CAN	9
3.9	U(S)ART	9
3.10	I2S	9
3.11	XSPI	10
3.12	DVP	10
3.13	USB_HS_DualRole	10
4	Software Differences	11
4.1	Introduction to Software Libraries	11
4.2	Module Overview	14
4.3	Detailed Comparison of Module Driver APIs and Points for Attention	16
4.3.1	RCC	16
4.3.2	PWR	18
4.3.3	DMA	21
4.3.4	CRC	25
4.3.5	SPI/I2S	26
4.3.6	I2C	32
4.3.7	ALGO	34
4.3.8	ETH	35
4.3.9	CORDIC	36
4.3.10	LPTIM	37
4.3.11	GPIO	37
4.3.12	EXTI	40
4.3.13	SDIO	40
4.3.14	FDCCAN	43

4.3.15	IWDG	44
4.3.16	WWDG	44
4.3.17	FLASH	46
4.3.18	MISC	49
4.3.19	USART	50
4.3.20	DBG	53
4.3.21	RTC	54
4.3.22	USBFS	58
4.3.23	USBHS	58
4.3.24	FEMC	58
4.3.25	SHRTIM	58
4.3.26	DVP	58
4.3.27	TIM	59
4.3.28	ADC	70
4.3.29	DAC	76
4.3.30	COMP	78
4.3.31	PGA	80
4.3.32	VREFBUF	83
4.3.33	XSPI	83
4.	Version History	87
6.	Disclaimer	88

1. Overview

The N32G45x series are general-purpose MCUs from Nsing Technologies based on the Cortex-M4 core, while the N32H47x and N32H48x series are the latest high-performance general-purpose MCUs also from Nsing Technologies, developed on the Cortex-M4 core.

Compared to the N32G45x series, the N32H47x and N32H48x series have introduced new modules, optimized and modified some existing module designs, and added new functionalities.

This guide primarily introduces the software differences between the two series of chips, with the following contents:

1. Resource Comparison: It mainly presents a comprehensive comparison of resources across the N32G45x, N32H47x, and N32H48x series.
2. Hardware Comparison: It focuses on the differences in hardware design between the series.
3. Software Library: It introduces the file structure and content overview of the official Software Development Kit (SDK) software library.
4. Module Overview: It provides an overview of the differences in various modules between the N32G45x series and the N32H47x_48x series.
5. Software Differences:
 - a) Driver APIs: It provides a detailed comparison of the functions and input parameters of the driver APIs for each module across the series.
 - b) Application Examples: It introduces the differences in application usage and points to note for each module.

2. Resource Differences

2.1 Comparison between N32G45x and N32H47x

The points to note are as follows:

1. The N32G455CEQ7 does not support USBFSD.
2. N32G452 series only support 2 ADC, N32G455 and N32G457 series support 4 ADC.
3. The N32G452 and N32G455 series do not support the DVP and ETH modules.

Part Number		N32G452CB/C/E N32G455CB/C/E			N32G452RB/C/E N32G457RC/E N32G455RB/C/E			N32G452MB/C/E N32G455MB/C/E N32G457MC/E			N32G452VB/C/E N32G455VB/C/E N32G457VC/E			N32G452QC/E N32G457QE		N32H473KC/U7/8 N32H473KEU7/8		N32H473CCU7/8 N32H473CEU7/8 N32H474CCU7/8 N32H473CCL7/8 N32H473CEL7/8 N32H474RCL7/8 N32H474REL7/8		N32H473RCL7/8 N32H473REL7/8 N32H474RCL7/8 N32H474REL7/8		N32H473MCL7/8 N32H473MEL7/8 N32H474MCL7/8 N32H474REL7/8		N32H473VCL7/8 N32H473VEL7/8 N32H474VCL7/8 N32H474VEL7/8		N32H473QCL7/8 N32H473QEL7/8 N32H474QCL7/8 N32H474QEL7/8			
Flash (KB)		128	256	512	128	256	512	128	256	512	128	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512
General SRAM (KB)		80	144	144	80	144	144	80	144	144	80	144	144	144	144	112	160	112	160	112	160	112	160	112	160	112	160	112	160
CCM SRAM (KB)		0														32													
Backup SRAM (KB)		0														4													
CPU frequency		ARM Cortex-M4F @144MHz/180DMIPS														1.8~3.6V/-40~105°C /125°C													
Working environment		1.8~3.6V/-40~105°C														ARM Cortex-M4F @200MHz/250DMIPS													
Timer	General	TIM2 TIM3 TIM4 TIM5														GTIM1 GTIM2 GTIM3 GTIM4 GTIM5 GTIM6 GTIM7 GTIM8 GTIM9 GTIM10													
	Advanced	TIM1 TIM8														ATIM1 ATIM2 ATIM3													
	Basic	TIM6 TIM7														BTIM1 BTIM2													
	Low Power	NO														LPTIM1 LPTIM2													
Communication interface	SPI	SPI1 SPI2 SPI3														SPI1 SPI2 SPI3 SPI6		SPI1 SPI2 SPI3 SPI4 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6									
	I ² S	I2S2 I2S3																I2S2 I2S3											
	XSPI	QSPI (4线)																XSPI (8线)											
	I ² C	I2C1 I2C2 I2C4	I2C1 I2C2 I2C3 I2C4															I2C1 I2C2 I2C3 I2C4											
	USART	USART1 USART2 USART3																USART1 USART2 USART3 USART4											
	UART	UART4 UART5 UART6	UART4 UART5 UART6 UART7															UART5 UART6 UART7 UART8											
	USB FS Device	0 (1) /USBFS			USBFS													USBFS											
	CAN	CAN1 CAN2																FDCAN1 FDCAN2 FDCAN3(1)											
	SDIO	NO	SDIO															NO											
	DVP	0 (3) /DVP																NO											
	ETH	0 (3) /ETH																NO											
Memory expansion	FEMC	NO																FEMC											
GPIO		37	42	51	65			80			97			26	42/38(2)		52	66		86		107							
DMA		2																2											
Number of channels		16 Channel																16 Channel											
12bit ADC		2 (2) /4																4											
Number of channels		10Channel (2) /16Channel	16Channel (2) /22Channel	16Channel (2) /33 Channel			16Channel (2) /38Channel			18Channel (2) /40Channel			13Channel	21Channel/20Channel(2)	26Channel	38Channel	45Channel	51Channel											
12bit DAC		2																8											
Number of channels		2Channel																8 (4 External/Internal + 4 Internal)											
OPAMP/PGA		OPAMP1 OPAMP2 OPAMP3 OPAMP4																PGA1 PGA2 PGA3 PGA4											
COMP		COMP1 COMP2 COMP3 COMP4	COMP1 COMP2 COMP3 COMP4 COMP5 COMP6 COMP7															COMP1 COMP2 COMP3 COMP4 COMP5 COMP6 COMP7											
Algorithm support		DES/3DES、AES、SHA1/SHA224/SHA256、SM1、SM3、SM4F、SM7、MD5、CRC16/CRC32、TRNG														DES/3DES、AES、SHA1/SHA224/SHA256、SM3、SM4、MD5、CRC16/CRC32、TRNG													
Cordic		NO																Yes											
FMAC		NO																Yes											
Security protection		Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot														Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot													
Package		LQFP48 QFN48	LQFP64		LQFP80		LQFP100		LQFP128		UQFN32	LQFN48 LQFP48		LQFP64		LQFP80		LQFP100		LQFP128									

2.2 Comparison between N32G45x and N32H48x

Part Number		N32G452CB/C/E N32G455CB/C/E			N32G452RB/C/E N32G457RC/E N32G455RB/C/E			N32G452MB/C/E N32G455MB/C/E N32G457MC/E			N32G452VB/C/E N32G455VB/C/E N32G457VC/E			N32G452QC/E N32G457QE		N32H482REL7 N32H487REL7		N32H482VEL7 N32H487VEL7		N32H482ZEL7 N32H487ZEL7	
Flash (KB)		128	256	512	128	256	512	128	256	512	128	256	512	256	512	512		512		512	
General SRAM (KB)		80	144	144	80	144	144	80	144	144	80	144	144	144	144	160		160		160	
CCM SRAM (KB)		0														32					
Backup SRAM (KB)		0														4					
CPU frequency		ARM Cortex-M4F @144MHz,180DMIPS														ARM Cortex-M4F @240MHz,300DMIPS					
Working environment		1.8~3.6V/-40~105℃														1.8~3.6V/-40~105℃					
Timer	General	TIM2 TIM3 TIM4 TIM5														GTIM1 GTIM2 GTIM3 GTIM4 GTIM5 GTIM6 GTIM7 GTIM8 GTIM9 GTIM10					
	Advanced	TIM1 TIM8														ATIM1 ATIM2 ATIM3					
	Basic	TIM6 TIM7														BTIM1 BTIM2					
	Low Power	NO														LPTIM1 LPTIM2					
Communication interface	SPI	SPI1 SPI2 SPI3														SPI1 SPI2 SPI3 SPI4 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6			
	I ² S	I2S2 I2S3														I2S2 I2S3					
	XSPI	QSPI (4线)														XSPI(8线)					
	I ² C	I2C1 I2C2 I2C4	I2C1 I2C2 I2C3 I2C4													I2C1 I2C2 I2C3 I2C4					
	USART	USART1 USART2 USART3														USART1 USART2 USART3 USART4					
	UART	UART4 UART5 UART6	UART4 UART5 UART6 UART7													UART5 UART6 UART7 UART8					
	USB FS Device	0 (1) /USBFS			USBFS											USBFS					
	USBHS Dual Role	NO														USBHS					
	CAN	CAN1 CAN2														FDCAN1 FDCAN2					
	SDIO	NO		SDIO												SDIO					
	DVP	0 (3) /DVP														DVP					
ETH	0 (3) /ETH														ETH						
Memory expansion	FEMC	NO														No		FEMC			
GPIO		37	42	51			65			80			97		54	85		118			
DMA		2														2					
Number of channels		16 Channel														16Channel					
12bit ADC		2 (2) /4														4	4		4		
Number of channels		10Channel (2) /16Channel			16Channel (2) /22Channel			16Channel (2) /33 Channel			16Channel (2) /38Channel			18Channel(2) /40Channel		26Channel	42Channel		51Channel		
12bit DAC		2														2					
Number of channels		2Channel														2 External					
OPAMP/PGA		OPAMP1 OPAMP2 OPAMP3 OPAMP4														NO					
COMP		COMP1 COMP2 COMP3 COMP4		COMP1 COMP2 COMP3 COMP4 COMP5 COMP6 COMP7												NO					
Algorithm support		DES/3DES、AES、SHA1/SHA224/SHA256、SM1、SM3、SM4F、SM7、MD5、CRC16/CRC32、TRNG														DES/3DES、AES、SHA1/SHA224/SHA256、SM3、SM4、MD5、CRC16/CRC32、TRNG					
Cordic		NO														Yes					
FMAC		NO														Yes					
Security protection		Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot														Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot					

Package	LQFP48 QFN48	LQFP64	LQFP80	LQFP100	LQFP128	LQFP64	LQFP100	LQFP144
---------	-----------------	--------	--------	---------	---------	--------	---------	---------

2.3 Comparison between N32H47x and N32H48x

Part Number		N32H482REL7 N32H487REL7		N32H482VEL7 N32H487VEL7		N32H482ZEL7 N32H487ZEL7		N32H473KCU7/8 N32H473KEU7/8		N32H473CCU7/8 N32H473CEU7/8 N32H474CCU7/8 N32H474CEU7/8 N32H473CCL7/8 N32H473CEL7/8 N32H474RCL7/8 N32H474REL7/8		N32H473RCL7/8 N32H473REL7/8 N32H474RCL7/8 N32H474REL7/8		N32H473MCL7/8 N32H473MEL7/8 N32H474MCL7/8 N32H474MEL7/8		N32H473VCL7/8 N32H473VEL7/8 N32H474VCL7/8 N32H474VEL7/8		N32H473QCL7/8 N32H473QEL7/8 N32H474QCL7/8 N32H474QEL7/8	
CPU frequency		1.8~3.6V/-40~105℃								1.8~3.6V/-40~105℃ /125℃									
Working environment		ARM Cortex-M4F @240MHz, 300DMIPS								ARM Cortex-M4F @200MHz, 250DMIPS									
Flash (KB)		512		512		512		256	512	256	512	256	512	256	512	256	512	256	512
Total SRAM	General SRAM	160		160		160		112	160	112	160	112	160	112	160	112	160	112	160
(KB)	CCM SRAM	32								32									
	Backup SRAM	4								4									
Timer	ATIM	ATIM1 ATIM2 ATIM3								ATIM1 ATIM2 ATIM3									
	GTIM	GTIM1 GTIM2 GTIM3 GTIM4 GTIM5 GTIM6 GTIM7 GTIM8 GTIM9 GTIM10								GTIM1 GTIM2 GTIM3 GTIM4 GTIM5 GTIM6 GTIM7 GTIM8 GTIM9 GTIM10									
	BTIM	BTIM1 BTIM2								BTIM1 BTIM2									
	LPTIM	LPTIM1 LPTIM2								LPTIM1 LPTIM2									
Communication interface	SPI/I2S	SPI1 SPI2 SPI3 SPI4 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6		SPI1 SPI2 SPI3 SPI6		SPI1 SPI2 SPI3 SPI6		SPI1 SPI2 SPI3 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6		SPI1 SPI2 SPI3 SPI4 SPI5 SPI6	
	I2S	I2S2 I2S3								I2S2 I2S3									
	I²C	I2C1 I2C2 I2C3 I2C4								I2C1 I2C2 I2C3 I2C4									
	USART	USART1 USART2 USART3 USART4								USART1 USART2 USART3 USART4									
	UART	UART5 UART6 UART7 UART8								UART5 UART6 UART7 UART8									
	USB FS Device	USBFS								USBFS									
	USB HS DualRole	USBHS								NO									
	FDCAN	FDCAN1 FDCAN2								FDCAN1 FDCAN2 FDCAN3(1)									
Ethernet	ETH								NO										
Memory expansion	XSPI	XSPI								XSPI									
	FEMC	No		FEMC				No						FEMC					
	SDIO	SDIO								NO									
Human-machine interaction	DVP	DVP								NO									
GPIO		54		85		118		26		42/38(2)		52		66		86		107	
DMA		2								2									
Number of channels		16Channel								16 Channel									
12bit ADC		4		4		4		4		4		4		4		4		4	
Number of channels		26Channel		42Channel		51Channel		13Channel		21Channel/20Channel(2)		26Channel		38Channel		45Channel		51Channel	
12bit DAC		2								8									
Number of channels		2 External								8 (4 External/Internal + 4 Internal)									
OPAMP/PGA		NO								PGA1 PGA2 PGA3 PGA4									
COMP		NO								COMP1 COMP2 COMP3 COMP4 COMP5 COMP6 COMP7									
Algorithm support		DES/3DES、AES、SHA1/SHA224/SHA256、SM3、SM4、MD5、CRC16/CRC32、TRNG								DES/3DES、AES、SHA1/SHA224/SHA256、SM3、SM4、MD5、CRC16/CRC32、TRNG									
Cordic		Yes								Yes									
FMAC		Yes								Yes									
Security protection		Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot								Read-write protection (RDP/WRP)、storage encryption、partition protection、secure boot									
Package		LQFP64		LQFP100		LQFP144		UQFN32		UQFN48 LQFP48		LQFP64		LQFP80		LQFP100		LQFP128	

3 Hardware Differences

3.1 Power Supply

Comparison between the N32H47x/48x Series and the N32G45x Series, the main design difference lies in the VREF+ power supply.

N32H47x/48x Series (VREF+ Reference Voltage):

- When using the built-in reference source VREFBUF for VREF+, it is recommended to place a 0.1uF and a 1uF capacitor close to the VREF+ pin.
- When VREF+ is powered externally, it is recommended to place a 0.1uF and a 10uF capacitor close to the VREF+ pin.

3.2 Boot Pins

N32G45x Series: BOOT0, BOOT1.

N32H47x/48x Series: BOOT0.

3.3 ADC Converter

N32G45x Series:

With an ADC input clock of 72MHz, the ADC fast channel sampling rate does not exceed 5Msps, and the ADC slow channel sampling rate is recommended to not exceed 2.5Msps.

N32H47x/48x Series:

With an ADC input clock of 80MHz, the ADC fast channel sampling rate does not exceed 4.7Msps, and the ADC slow channel sampling rate is recommended to not exceed 2.5Msps.

3.4 IO Voltage Tolerance

N32G45x Series:

FT: Tolerates 5V; TTa: Tolerates 3.3V, supports analog peripherals; TC: Standard 3.3V I/O.

N32H47x/48x Series:

FT: Tolerates 5V; FTa: Tolerates 5V, supports analog peripherals.

3.5 Operational Amplifiers

N32G45x Series:

Embeds up to 4 independent operational amplifiers (OPAMP) with various working modes such as external amplification, internal follow-up, and programmable gain amplifiers (PGA).

N32H47x Series:

Contains 4 flexible programmable gain amplifiers (PGA) with output options internally connected to ADC or COMP input channels, external amplification is not supported. For details, see the pin multiplexing definitions in the corresponding datasheet.

N32H48x Series:

Does not support operational amplifiers.

3.6 TSC Touch Sensor

N32G45x Series:

Supports a maximum of 24 capacitive touch channels.

N32H47x/48x Series:

Does not support TSC touch functionality.

3.7 Super High-Resolution Timer (SHRTIM)

N32G45x Series:

Does not support a super high-resolution timer.

N32H47x/48x Series:

N32H47x supports one super high-resolution timer. N32H48x does not support it.

3.8 CAN

N32G45x Series:

Supports 2 CAN bus interfaces compatible with specifications 2.0A and 2.0B (active). Full pin mapping is not supported.

N32H47x/48x Series:

Supports up to 3 FDCAN, compatible with CAN 2.0A/B and CAN FD protocols, and Bosch protocols that are non-ISO standard. Full pin mapping is supported.

3.9 U(S)ART

N32G45x Series:

Supports 3 universal synchronous/asynchronous receivers/transmitters (USART1, USART2, and USART3) and 4 universal asynchronous receivers/transmitters (UART4, UART5, UART6, UART7). Full pin mapping is not supported.

N32H47x/48x Series:

Supports 4 USARTs (USART1, USART2, USART3) and 4 UARTs (UART4, UART5, UART6, UART7). Partial pin mapping supports full pin mapping.

3.10 I2S

N32G45x Series:

Only supports half-duplex communication.

N32H47x/48x Series:

Supports full-duplex communication.

3.11 XSPI

N32G45x Series:

Supports Single SPI, Dual SPI, Quad SPI modes.

N32H47x/48x Series:

Supports Single SPI, DUAL SPI, QUAD SPI, Dual-QUAD, OCTAL SPI modes.

3.12 DVP

N32G45x Series:

Supports up to a 14-bit traditional synchronous parallel interface.

N32H47x/48x Series:

Supports an 8-bit, 10-bit, 12-bit, and 16-bit configurable traditional synchronous parallel interface.

3.13 USB_HS_DualRole

N32G45x Series:

Does not support USB High-Speed Dual Role (USB HS Dual Role).

N32H47x/48x Series:

Embeds a USB HS Dual Role interface supporting both Host and Device modes.

4 Software Differences

4.1 Introduction to Software Libraries

Nations will provide an official development kit, which includes SDK software libraries, data manuals, user manuals, usage guides, application notes, and other materials. The file structure of the SDK software library is as follows:

Folder	Subfolder 1	Subfolder 2	Instructions
firmware	CMSIS	core	Kernel file: Must be added to the application project
		device	Startup file and system configuration file: Must be added to the application project
		inc	.h and .c Files of Standard Firmware Libraries for Each Module: Select module drivers based on actual application requirements
		src	
		inc	Security Algorithm Library: Select based on actual application requirements
	n32xxxx_algo_lib	lib	
	n32xxxx_usbfs_driver	inc	USBFS Library: Select based on actual application requirements
		lib	
	n32xxxx_usbhs_driver	device	USBHS Library: Select based on actual application requirements.
		driver	(Note: N32G45x does not have this folder)
middlewares	rt-thread	host	N32H47x_48x does not have the aforementioned USBHS folder
		/	File System (for USB examples in the SDK): Select based on actual application requirements.
	fat_fs	src	(Note: N32G45x does not have this folder)
			FreeRTOS Operating System (for Ethernet examples in the SDK): Select based on actual application requirements.
	FreeRTOSv202212.01	Source	(Note: N32G45x does not have this folder)
projects	lwip-2.2.0	src	lwIP Dedicated Protocol Stack for Ethernet: Select based on actual application requirements.
			(Note: N32G45x does not have this folder)
	n32xxxx_EVAL	bsp	Board-Based Print and Delay Configurations: Select based on actual application requirements
		examples	Typical Application Examples for Each Module: Select module example references based on actual application requirements

Each module drivers (n32xxxx_std_periph_driver) :

> firmware > n32h47x_48x_std_periph_driver > src

名称	修改日期
 misc.c	2024/4/10 19:06
 n32h47x_48x_adc.c	2024/4/18 18:55
 n32h47x_48x_comp.c	2024/4/10 19:06
 n32h47x_48x_cordic.c	2024/4/10 19:06
 n32h47x_48x_crc.c	2024/4/10 19:06
 n32h47x_48x_dac.c	2024/4/10 19:06
 n32h47x_48x_dbg.c	2024/4/10 19:06
 n32h47x_48x_dma.c	2024/4/10 19:06
 n32h47x_48x_dvp.c	2024/4/10 19:06
 n32h47x_48x_eth.c	2024/4/10 19:06
 n32h47x_48x_exti.c	2024/4/10 19:06
 n32h47x_48x_fdcan.c	2024/4/10 19:06
 n32h47x_48x_femc.c	2024/4/10 19:06
 n32h47x_48x_flash.c	2024/4/10 19:06
 n32h47x_48x_fmac.c	2024/4/10 19:06
 n32h47x_48x_gpio.c	2024/4/10 19:06
 n32h47x_48x_i2c.c	2024/4/12 11:21
 n32h47x_48x_iwdg.c	2024/4/10 19:06
 n32h47x_48x_lptim.c	2024/4/10 19:06
 n32h47x_48x_pga.c	2024/4/10 19:06
 n32h47x_48x_pwr.c	2024/4/10 19:06
 n32h47x_48x_rcc.c	2024/4/10 19:06
 n32h47x_48x_rtc.c	2024/4/10 19:06
 n32h47x_48x_sdio.c	2024/4/10 19:06
 n32h47x_48x_shrtim.c	2024/4/10 19:06
 n32h47x_48x_spi.c	2024/4/10 19:06
 n32h47x_48x_tim.c	2024/4/10 19:06
 n32h47x_48x_usart.c	2024/4/12 11:18
 n32h47x_48x_vrefbuf.c	2024/4/10 19:06
 n32h47x_48x_wwdg.c	2024/4/10 19:06
 n32h47x_48x_xspi.c	2024/4/10 19:06

Each module Demos (n32xxxx_std_periph_driver) :

> projects > n32h47x_48x_EVAL > examples

名称	修改日期
AD_GP_BS TIM	2024/4/29 10:28
ADC	2024/4/29 10:28
ALGO	2024/4/29 10:28
COMP	2024/4/29 10:28
CORDIC	2024/4/29 10:28
Cortex-M4F	2024/4/29 10:28
CRC	2024/4/29 10:28
DAC	2024/4/29 10:28
DMA	2024/4/29 10:28
DVP	2024/4/29 10:28
ETH	2024/4/29 10:28
EXTI	2024/4/29 10:28
FDCAN	2024/4/29 10:28
FEMC	2024/4/29 10:28
Flash	2024/4/29 10:28
FMAC	2024/4/29 10:28
GPIO	2024/4/29 10:28
I2C	2024/4/29 10:28
I2S	2024/4/29 10:28
IWDG	2024/4/29 10:28
LPTIM	2024/4/29 10:28
NVIC	2024/4/29 10:28
PGA	2024/4/29 10:28
PWR	2024/4/29 10:28
RCC	2024/4/29 10:28
RTC	2024/4/29 10:28
SDIO	2024/4/29 10:28
SHRTIM	2024/4/29 10:28
SPI	2024/4/29 10:28
USART	2024/4/29 10:28
USBFS	2024/4/29 10:28
USBHS	2024/4/29 10:28
VREFBUF	2024/4/29 10:29
WWDG	2024/4/29 10:29
XSPI	2024/4/29 10:29

4.2 Module Overview

The following table provides an overview of the differences between the modules of the N32G45x series and the N32H47x_48x series. In the section titled "Detailed Comparison of Module Driver APIs and Points for Attention," modules with minor differences will showcase detailed comparisons of the driver API functions and input parameters. For modules with significant differences or new additions that are difficult to compare in detail, only usage guidelines and points for attention are provided. It is recommended to refer to the SDK examples for application development.

N32G45x series	N32H47x_48x series	Overview of Differences
RCC	RCC	System Modules with Significant Differences
PWR	PWR	System Modules: Minor differences with some new features added
FLASH	FLASH	FLASH Module: Minor differences with some new features added
GPIO	GPIO	GPIO Basic and Remapping Configurations: Significant differences with some new features added
MISC	MISC	System Modules: Consistent functions, direct replacement possible
ADC	ADC	Significant differences with multiple new features added
BKP	-	BKP Module has been removed
CAN	FDCAN	The N32G45x series only supports CAN 2.0A/B, while the N32H47x_48x series supports FDCAN, resulting in significant differences
COMP	COMP	Minor differences.
CRC	CRC	Minor differences.
DAC	DAC	Significant functional differences with multiple new features added.
DBG	DBG	The N32H47x_48x series supports SHRTIM debugging.
DMA	DMA	Minor differences with new burst function and some channel request mapping differences.
DVP	DVP	Significant functional differences.
ETH	ETH	IP has been upgraded. Ethernet functionality is unaffected, but driver functions and example project code have significant differences.
EXTI	EXTI	The difference is smaller, the number of external interrupt lines increased from 22 to 32, and the external interrupt supports full pin mapping.
I2C	I2C	The I2C of the N32G45x series does not support FIFO transmit and receive functions, while the N32H47x_48x series does.
IWDG	IWDG	The N32G45x series does not support freeze, while the N32H47x_48x series does.

OPAMP	PGA	Module naming has been changed, resulting in significant functional differences.
QSPI	XSPI	Module naming has been changed. The N32H47x_48x series supports 8-line mode, with significant example differences.
RTC	RTC	The N32G45x series does not support reference clock input. BKP and RTC are separate modules. The N32H47x_48x series supports reference clock input, with BKP and RTC merged into one module, supporting intrusion interrupt connection to EXTI19.
SDIO	SDIO	Minor differences in module functionality, with some register bit definitions adjusted. The SDK has added some interface API functions.
SPI	SPI	The N32G45x series supports 3 SPI interfaces (SPI1~SPI3), with 2 I2S interfaces sharing SPI2 and SPI3. The N32H47x_48x series supports 6 SPI interfaces (SPI1~SPI6), with 2 I2S interfaces sharing SPI2 and SPI3. The SPI of the N32G45x series does not support FIFO functionality, while the N32H47x_48x series does. The I2S audio sampling frequency configuration range of the N32G45x series is from 8KHz to 96KHz, while that of the N32H47x_48x series is from 8KHz to 192KHz.
TIM	TIM	One additional brake line has been added. Separate enable and polarity control bits for brake sources have been added. Bidirectional brake IO functionality has been added. More brake sources have been added. Encoder mode has been added. Significant changes in interconnectivity.
TSC	-	The TSC module has been removed.
USART	USART	Minor differences with some new features added.
WWDG	WWDG	The window value for N32G45x is 7-bit, while it is 14-bit for N32H47x_48x.
ALGO	ALGO	The driver is encapsulated into a library, and interface functions can be called directly.
USBFS	USBFS	Module naming has been changed. The N32H47x_48x series supports crystal-less mode.
-	USBHS	Add USBHS module
-	CORDIC	Add CORDIC module
-	FEMC	Add FEMC module
-	FMAC	Add FMAC module
-	LPTIM	Add LPTIM module

-	SHRTIM	Add SHRTIM module
-	VREFBUF	Add VREFBUF module

4.3 Detailed Comparison of Module Driver APIs and Points for Attention

4.3.1 RCC

Clock control is a system module, and there are significant design differences between the N32G45x series and the N32H47x_48x series. It is recommended that users directly refer to the N32H47x_48x series examples. Before entering the main function, the functions in system_n32h47x_48x.c will be automatically called to initialize the clock. Users can select different clock sources and system clock frequencies through the following macro definitions:

```
#define SYSCLK_USE_HSI      0
#define SYSCLK_USE_HSE      1
#define SYSCLK_USE_HSI_PLL  2
#define SYSCLK_USE_HSE_PLL  3

#ifndef SYSCLK_FREQ
#if defined (N32H473) || defined (N32H474)
#define SYSCLK_FREQ      200000000
#elif defined (N32H475) || defined (N32H482) || defined (N32H487)
#define SYSCLK_FREQ      240000000
#else
#error wrong macro definition
#endif
#endif

#ifndef SYSCLK_SRC
#define SYSCLK_SRC      SYSCLK_USE_HSI_PLL
#endif
```

Select system clock frequency by MCU series

Select system clock source

Meanwhile, the example project code for "RCC_ClockConfig" in the RCC module also provides configurations for other system clocks for reference. Depending on the macro definitions, the system clock source can also be configured as SHRTPLL or USB240M.


```

73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
main.c
log_init(); /* should reinit after sysclk changed */
log_info("==\n");
log_info("msg:\n", msg);
RCC_GetClocksFreqValue(&RCC_ClockFreq);
log_info("SYSCLK: %d\n", RCC_ClockFreq.SysclkFreq);
log_info("PCLK1: %d\n", RCC_ClockFreq.Pclk1Freq);
log_info("PCLK2: %d\n", RCC_ClockFreq.Pclk2Freq);

int main(void)
{
    PrintfClockInfo("After reset");

    /** Select one of the following configuration methods */
    #if SYSCLK_SOURCE_SELECT == SYSCLK_SOURCE_HSI
        /* Method 1 */
        SetSysClockToHSI();
        PrintfClockInfo("HSI->SYSCLK, 8MHz");
    #elif SYSCLK_SOURCE_SELECT == SYSCLK_SOURCE_HSE
        /* Method 2 */
        SetSysClockToHSE();
        PrintfClockInfo("HSE->SYSCLK, 8MHz");
    #elif SYSCLK_SOURCE_SELECT == SYSCLK_SOURCE_PLL
        /* Method 3 */
        SetSysClockToPLL(RCC_PLL_SRC_HSE, 16000000);
        PrintfClockInfo("HSE->PLL->SYSCLK, 160M");
    #elif SYSCLK_SOURCE_SELECT == SYSCLK_SOURCE_SHRTPLL
        /* Method 4 */
        SetSysClockToSHRTPLL(RCC_SHRTPLL_SRC_HSI, HSI_VALUE, (240000000));
        PrintfClockInfo("HSI->SHRTPLL->SYSCLK, 240M");
    #elif SYSCLK_SOURCE_SELECT == SYSCLK_SOURCE_USBHS240M
        /* Method 5 */
        SetSysClockToUSBHS240M();
        PrintfClockInfo("HSE->SHRTPLL->USBHS240M->SYSCLK, 240M");
    #endif

    /* Output HSE clock on MCO pin */
    RCC_EnableAHB1PeriphClk(RCC_AHB1_PERIPHEN_GPIOA, ENABLE);
    RCC_EnableAHB2PeriphClk(RCC_AHB2_PERIPH_AFIO, ENABLE);

    GPIO_InitStructure(&GPIO_InitStructure);
    GPIO_InitStructure.Pin = GPIO_PIN_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_MODE_AF_PP;
    GPIO_InitStructure.GPIO_Alternate = (uint32_t)13;
    GPIO_InitPeripheral(GPIOA, &GPIO_InitStructure);

    /* If the MCO selects PLL, SHRTPLL or USBHS240M, the prescaler can be configured first */
    RCC_ConfigMcoClkPre(RCC_MCO_PLLCLK_DIV10);
    /* Select the corresponding MCO clock source */
    RCC_ConfigMco1(RCC_MCO_SYSCLK);

    while (1);
}

41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
n32h47x_48x_it.h
/*
 * NATIONS products and technologies shall not be used for or incorporated
 * whose manufacture, use, or sale is prohibited under any applicable domestic
 * User shall comply with any applicable export control laws and regulations;
 * the governments of any countries asserting jurisdiction over the parties or
 */

/**
 * \file n32h47x_48x_it.h
 * \author Nations
 * \version v1.0.0
 * \copyright Copyright (c) 2023, Nations Technologies Inc. All rights reserved.
 */

#ifndef _N32H47X_48X_IT_H
#define _N32H47X_48X_IT_H

#ifdef __cplusplus
extern "C" {
#endif

#include "n32h47x_48x.h"
#include "n32h47x_48x_rcc.h"
#include "n32h47x_48x_usart.h"
#include "n32h47x_48x_gpio.h"
#include "n32h47x_48x_flash.h"
#include "misc.h"

#define SYSCLK_SOURCE_HSI 0
#define SYSCLK_SOURCE_HSE 1
#define SYSCLK_SOURCE_PLL 2
#define SYSCLK_SOURCE_SHRTPLL 3
#define SYSCLK_SOURCE_USBHS240M 4

#ifdef SYSCLK_SOURCE_SELECT
#define SYSCLK_SOURCE_SELECT SYSCLK_SOURCE_PLL /*select sysclk source */
#endif

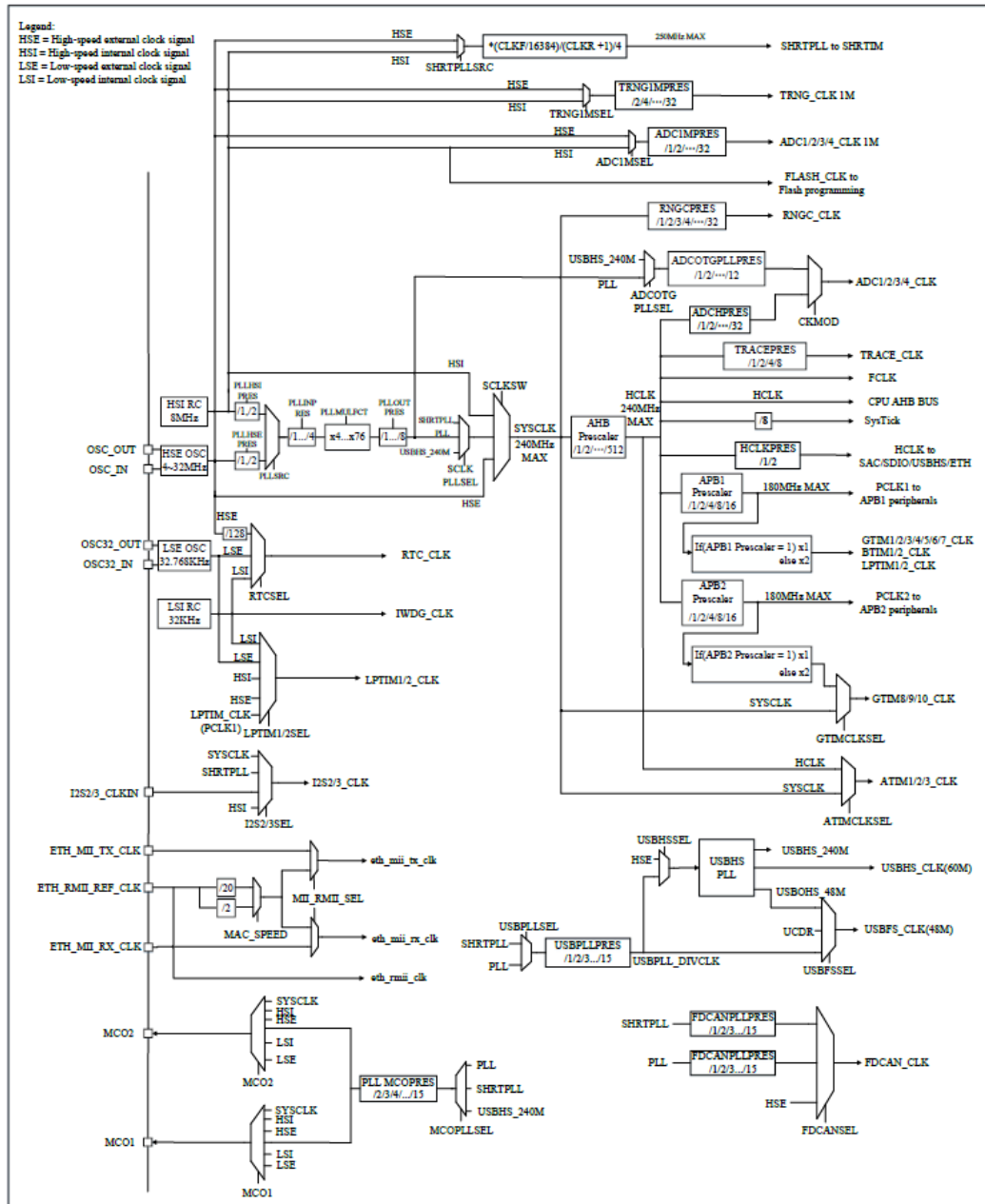
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
void RCC_IRQHandler(void);

#ifdef __cplusplus
}
#endif

#endif /* _N32H47X_48X_IT_H */

```

If there are any other specific application requirements, users can configure the clock according to the following clock tree with detailed reference to the user manual:



4.3.2 PWR

The following table compares the driver API functions of the PWR module between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code calling the PWR module driver API functions can be replaced according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
PWR reset	void PWR_DeInit(void)	void PWR_DeInit(void)	YES	None
RTC and backup register access is enabled	void PWR_BackupAccessEnable(FunctionalState Cmd)	void PWR_BackupAccessEnable(FunctionalState Cmd)	YES	None
PVD Enable	void PWR_PvdEnable(FunctionalState Cmd)	void PWR_PvdEnable(FunctionalState Cmd)	YES	None

Configure PVD threshold gear	void PWR_PvdRangeConfig(uint32_t PWR_PVDLevel)	void PWR_PVDLevelConfig(uint32_t level)	NO	The function name and input parameter PWR_PVDLevel are inconsistent: N32G45x serial parameter: PWR_PVDRANGRE_2V2 PWR_PVDRANGRE_2V3 PWR_PVDRANGRE_2V4 PWR_PVDRANGRE_2V5 PWR_PVDRANGRE_2V6 PWR_PVDRANGRE_2V7 PWR_PVDRANGRE_2V8 PWR_PVDRANGRE_2V9 N32H47x_48x serial parameter: PWR_PVD_LEVEL_2V18 PWR_PVD_LEVEL_2V28 PWR_PVD_LEVEL_2V38 PWR_PVD_LEVEL_2V48 PWR_PVD_LEVEL_2V58 PWR_PVD_LEVEL_2V68 PWR_PVD_LEVEL_2V78 PWR_PVD_LEVEL_2V88 PWR_PVD_LEVEL_1V78 PWR_PVD_LEVEL_1V88 PWR_PVD_LEVEL_1V98 PWR_PVD_LEVEL_2V08 PWR_PVD_LEVEL_3V28 PWR_PVD_LEVEL_3V38 PWR_PVD_LEVEL_3V48 PWR_PVD_LEVEL_3V58
Configure PVD detection source	-	void PWR_PVDSourceConfig(uint32_t PVD source)	NO	N32G45x series does not have this API
Wakeup pin enable	void PWR_WakeUpPinEnable(FunctionalState Cmd)	void PWR_WakeUpPinEnable(uint32_t pin, FunctionalState Cmd)	NO	N32H47x_48x series add input parameters pin: WAKEUP_PIN0EN WAKEUP_PIN1EN WAKEUP_PIN2EN WAKEUP_PIN3EN WAKEUP_PIN4EN WAKEUP_PIN5EN
Configure wakeup pin polarity	-	void PWR_WakeUpPinPolarity(uint32_t pin, WAKEUP_PIN_POL polarity)	NO	N32G45x series does not have this API
Enable wake up peripherals	-	void PWR_WakeUpPeriphEnable(uint32_t periph, FunctionalState Cmd)	NO	N32G45x series does not have this API
Enter STOP mode	void PWR_EnterStopState(uint32_t PWR_Regulator, uint8_t PWR_STOPEntry)	void PWR_EnterSTOPMode(uint32_t PWR_Regulator, uint8_t PWR_STOPEntry)	NO	Function name and input parameters PWR_Regulator are inconsistent: PWR_REGULATOR_ON→PWR_REGULATOR_NORMAL
Enter SLEEP mode	void PWR_EnterSLEEPMode(uint8_t SLEEPONEXIT, uint8_t PWR_STOPEntry)	void PWR_EnterSLEEPMode(PWR_SLEEPONEXIT_STATUS SLEEPONEXIT, uint8_t PWR_SLEEPPEEntry)	NO	The input parameters are inconsistent: SLEEPONEXIT: 0→PWR_SLEEP_NOW 1→PWR_SLEEP_ON_EXIT PWR_SLEEPEEntry: PWR_STOPENTRY_WFI→PWR_SLEEPENTRY_WFI PWR_STOPENTRY_WFE→PWR_SLEEPENTRY_WFE
Enter STOP2 mode	void PWR_EnterSTOP2Mode(uint8_t PWR_STOPEntry)	-	NO	N32H47x_48x series does not have this API
Enter STANDBY mode	void PWR_EnterStandbyState(void)	void PWR_EnterSTANDBYMode(void)	NO	Function names are inconsistent

Get PWR status	FlagStatus PWR_GetFlagStatus(uint32_t PWR_FLAG)	FlagStatus PWR_GetFlagStatus(uint32_t PWR_FLAG)	NO	Input parameter PWR_FLAG are inconsistent: N32G45x serial parameter: PWR_WU_FLAG PWR_SB_FLAG PWR_PVDO_FLAG PWR_VBATF_FLAG N32H47x_48x serial parameter: PWR_FLAG_WKUP0 PWR_FLAG_WKUP1 PWR_FLAG_WKUP2 PWR_FLAG_WKUP3 PWR_FLAG_WKUP4 PWR_FLAG_WKUP5 PWR_FLAG_WKUPP PWR_FLAG_STANDBY PWR_FLAG_PVDOUT PWR_FLAG_VBAT
Clear PWR status	void PWR_ClearFlag(uint32_t PWR_FLAG)	void PWR_ClearFlag(uint32_t PWR_FLAG)	NO	Input parameter PWR_FLAG are inconsistent: N32G45x serial parameter: PWR_WU_FLAG PWR_SB_FLAG N32H47x_48x serial parameter: PWR_CLR_WKUP0 PWR_CLR_WKUP1 PWR_CLR_WKUP2 PWR_CLR_WKUP3 PWR_CLR_WKUP4 PWR_CLR_WKUP5 PWR_CLR_WKUPP PWR_CLR_STANDBY PWR_CLR_VBAT
BKR voltage regulation	-	void PWR_ConfigBKROutput(uint32_t voltage output)	NO	N32G45x series does not have this API
Enable IWDG reset	-	void PWR_EnableIWDGReset(FunctionalState Cmd)	NO	N32G45x series does not have this API
Configure Standby mode BRPSRAM remains	-	void PWR_EnableBKPSRAMRetainInStandbyMode(FunctionalState Cmd)	NO	N32G45x series does not have this API
Configure Vbat mode BRPSRAM remains	-	void PWR_EnableBKPSRAMRetainInVbatMode(FunctionalState Cmd)	NO	N32G45x series does not have this API
Enable LSE noise mitigation	-	void PWR_EnableLSENoiseMitigation(FunctionalState Cmd)	NO	N32G45x series does not have this API
Enable LSE_CSS interrupt	-	void PWR_EnableLSECSSCrystalInt(FunctionalState Cmd)	NO	N32G45x series does not have this API
Bypass LSE CSS high limit	-	void PWR_ConfigLSECSSBypassHighLimit(uint32_t high limit)	NO	N32G45x series does not have this API
Configure RTC source switchover when the LSE fails	-	void PWR_EnableLSECSSRTCsourceSwitch(FunctionalState Cmd)	NO	N32G45x series does not have this API
Get LSE CSS statu	-	FlagStatus PWR_GetLSECSSFlagStatus(uint32_t PWR_FLAG)	NO	N32G45x series does not have this API
Clear LSE CSS statu	-	void PWR_ClearLSECSSFlag(void)	NO	N32G45x series does not have this API
Configure LCO clock source	-	void PWR_ConfigLco(uint32_t LCO_source)	NO	N32G45x series does not have this API
Enale LCO	-	void PWR_EnableLCO(FunctionalState Cmd)	NO	N32G45x series does not have this API
Configure NRST pin mode	-	void PWR_ConfigNRSTMode(uint32_t NRST mode)	NO	N32G45x series does not have this API

The following is a comparison of the N32G45x series (left) and N32H47x_48x series (right) "SLEEP" sample

engineering code in the PWR module:

```

/* Serial main program.
*/
int main(void)
{
    /*!< At this stage the microcontroller clock setting is already configured,
    this is done through SystemInit() function which is called from startup
    file (startup_n32g45x_xx.s) before to branch to application main.
    To reconfigure the default setting of SystemInit() function, refer to
    system_n32g45x.c file
    */
    log_init();
    log_info("\r\n MCU Reset! \r\n");
    /* Enable PWR Clock */
    RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
    /* Initialize LEDs on n32g45x-EVAL board */
    LEDInit(LED1_PORT, LED1_PIN);
    LEDInit(LED3_PORT, LED3_PIN);
    LEDOff(LED1_PORT, LED1_PIN);
    LEDOff(LED3_PORT, LED3_PIN);
    /* Initialize Key button Interrupt to wake up the low power*/
    KeyInputExtiInit(KEY_INPUT_PORT, KEY_INPUT_PIN);
    /* Clear the Interrupt flag */
    EXTI_ClrITPendBit(EXTI_LINE0);
    DBG_ConfigPeriph(DBG_SLEEP, ENABLE);
    while (!)
    {
        /* Turn on LED3 */
        LEDBlink(LED3_PORT, LED3_PIN);
        /* Insert a long delay */
        delay(40);
        /* Request to enter SLEEP mode*/
        PWR_EnterSLEEPMode(SLEEP_NOW, PWR_STOPEXIT_WFI);
    }
}

```

```

/*
int main(void)
{
    /* At this stage the microcontroller clock setting is already configured,
    this is done through SystemInit() function which is called from startup
    file (startup_n32h47x_xx.s) before to branch to application main.
    To reconfigure the default setting of SystemInit() function, refer to
    system_n32h47x_48x.c file
    */
    log_init();
    log_info("\r\n --- Reset --- \r\n");
    /* Enable PWR Clock */
    RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
    /* Initialize Key button Interrupt to wake up the low power*/
    KeyInputExtiInit(KEY_INPUT_PORT, KEY_INPUT_PIN);
    /* Clear the Interrupt flag */
    EXTI_ClrITPendBit(KEY_EXTI_LINE);
    /* Enable the DBG_SLEEP to keep debug in low power */
    //DBG_ConfigPeriph(DBG_SLEEP, ENABLE);
    while (!)
    {
        /* Insert a long delay */
        delay(700);
        log_info("Entry SLEEP mode\r\n");
        /* Request to enter SLEEP mode*/
        PWR_EnterSLEEPMode(PWR_SLEEP_NOW, PWR_SLEEPENTRY_WFI);
        log_info("Exit SLEEP mode\r\n");
    }
}
/*
**\name delay.
**\fun Delay function.
*/

```

The PWR configuration process is basically the same. The difference is that the N32H47x_48x series changes the LED wake up prompt to print prompt, and disables the DBG_SLEEP mode configuration. Users can open the PWR mode according to whether they need to advance mode.

4.3.3 DMA

The following table is a comparison table of the driver API functions of the N32G45x series and the N32H47x_48x series in the DMA module. After replacing the driver.c/h file, the application code calls the DMA module driver API function can be replaced according to the following table:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
DMA reset	void DMA_DeInit(DMA_ChannelType* DMAyChx)	void DMA_DeInit(DMA_ChannelType* DMAChx)	YES	None
Initialize DMA	void DMA_Init(DMA_ChannelType* DMAyChx, DMA_InitType* DMA_InitParam)	void DMA_Init(DMA_ChannelType* DMAChx, DMA_InitType* DMA_InitParam)	NO	Add burst function. Before initialization, users should configure BURST_BYPASS, BURST_MODE, and BURST_LEN in the DMA_InitParam parameter as required
Configuration DMA structure parameter initialization	void DMA_StructInit(DMA_InitType* DMA_InitParam)	void DMA_StructInit(DMA_InitType* DMA_InitParam)	YES	None
Enable DMA channel	void DMA_EnableChannel(DMA_ChannelType* DMAyChx, FunctionalState Cmd)	void DMA_EnableChannel(DMA_ChannelType* DMAChx, FunctionalState Cmd)	YES	None
Configure DMA interrupt	void DMA_ConfigInt(DMA_ChannelType* DMAyChx, uint32_t DMAInt, FunctionalState Cmd)	void DMA_ConfigInt(DMA_ChannelType* DMAChx, uint32_t DMAInt, FunctionalState Cmd)	YES	None
Set the current transfer count (number need to be transferred)	void DMA_SetCurrDataCounter(DMA_ChannelType* DMAyChx, uint16_t DataNumber)	void DMA_SetCurrDataCounter(DMA_ChannelType* DMAChx, uint16_t DataNumber)	YES	None
Gets the current transfer count (number remaining to be transferred)	uint16_t DMA_GetCurrDataCounter(DMA_ChannelType* DMAyChx)	uint16_t DMA_GetCurrDataCounter(DMA_ChannelType* DMAChx)	YES	None
Get DMA status	FlagStatus DMA_GetFlagStatus(uint32_t DMAyFlag, DMA_Module* DMAy)	FlagStatus DMA_GetFlagStatus(uint32_t DMAFlag, DMA_Module* DMAy)	NO	The first parameter is inconsistent. The N32H47x_48x series enters one of the following parameters: DMA_FLAG_GL1 DMA_FLAG_TC1 DMA_FLAG_HT1

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
				DMA_FLAG_TE1 DMA_FLAG_GL2 DMA_FLAG_TC2 DMA_FLAG_HT2 DMA_FLAG_TE2 DMA_FLAG_GL3 DMA_FLAG_TC3 DMA_FLAG_HT3 DMA_FLAG_TE3 DMA_FLAG_GL4 DMA_FLAG_TC4 DMA_FLAG_HT4 DMA_FLAG_TE4 DMA_FLAG_GL5 DMA_FLAG_TC5 DMA_FLAG_HT5 DMA_FLAG_TE5 DMA_FLAG_GL6 DMA_FLAG_TC6 DMA_FLAG_HT6 DMA_FLAG_TE6 DMA_FLAG_GL7 DMA_FLAG_TC7 DMA_FLAG_HT7 DMA_FLAG_TE7 DMA_FLAG_GL8 DMA_FLAG_TC8 DMA_FLAG_HT8 DMA_FLAG_TE8
Clear DMA interrupt status	void DMA_ClearFlag(uint32_t DMAFlag, DMA_Module *DMAy)	void DMA_ClearFlag(uint32_t DMAyFlag, DMA_Module* DMAy)	NO	The first parameter is inconsistent. The N32H47x_48x series enters one of the following parameters: DMA_FLAG_GL1 DMA_FLAG_TC1 DMA_FLAG_HT1 DMA_FLAG_TE1 DMA_FLAG_GL2 DMA_FLAG_TC2 DMA_FLAG_HT2 DMA_FLAG_TE2 DMA_FLAG_GL3 DMA_FLAG_TC3 DMA_FLAG_HT3 DMA_FLAG_TE3 DMA_FLAG_GL4 DMA_FLAG_TC4 DMA_FLAG_HT4 DMA_FLAG_TE4 DMA_FLAG_GL5 DMA_FLAG_TC5 DMA_FLAG_HT5 DMA_FLAG_TE5 DMA_FLAG_GL6 DMA_FLAG_TC6 DMA_FLAG_HT6 DMA_FLAG_TE6 DMA_FLAG_GL7 DMA_FLAG_TC7 DMA_FLAG_HT7 DMA_FLAG_TE7 DMA_FLAG_GL8 DMA_FLAG_TC8 DMA_FLAG_HT8 DMA_FLAG_TE8
Get DMA interrupt status	INTStatus DMA_GetIntStatus(uint32_t DMAy_Int, DMA_Module *DMAy)	INTStatus DMA_GetIntStatus(uint32_t DMAy_IT, DMA_Module* DMAy)	NO	The first parameter is inconsistent. The N32H47x_48x series enters one of the following parameters: DMA_INT_GLB1 DMA_INT_TXC1 DMA_INT_HTX1 DMA_INT_ERR1 DMA_INT_GLB2 DMA_INT_TXC2 DMA_INT_HTX2

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
				DMA_INT_ERR2 DMA_INT_GLB3 DMA_INT_TXC3 DMA_INT_HTX3 DMA_INT_ERR3 DMA_INT_GLB4 DMA_INT_TXC4 DMA_INT_HTX4 DMA_INT_ERR4 DMA_INT_GLB5 DMA_INT_TXC5 DMA_INT_HTX5 DMA_INT_ERR5 DMA_INT_GLB6 DMA_INT_TXC6 DMA_INT_HTX6 DMA_INT_ERR6 DMA_INT_GLB7 DMA_INT_TXC7 DMA_INT_HTX7 DMA_INT_ERR7 DMA_INT_GLB8 DMA_INT_TXC8 DMA_INT_HTX8 DMA_INT_ERR8
Clear DAM interrupt pending bit	void DMA_ClrIntPendingBit(uint32_t DMAy_IT, DMA_Module* DMAy)	void DMA_ClrIntPendingBit(uint32_t DMAy_Int, DMA_Module *DMAy)	NO	The first parameter is inconsistent. The N32H47x_48x series enters one of the following parameters: DMA_INT_GLB1 DMA_INT_TXC1 DMA_INT_HTX1 DMA_INT_ERR1 DMA_INT_GLB2 DMA_INT_TXC2 DMA_INT_HTX2 DMA_INT_ERR2 DMA_INT_GLB3 DMA_INT_TXC3 DMA_INT_HTX3 DMA_INT_ERR3 DMA_INT_GLB4 DMA_INT_TXC4 DMA_INT_HTX4 DMA_INT_ERR4 DMA_INT_GLB5 DMA_INT_TXC5 DMA_INT_HTX5 DMA_INT_ERR5 DMA_INT_GLB6 DMA_INT_TXC6 DMA_INT_HTX6 DMA_INT_ERR6 DMA_INT_GLB7 DMA_INT_TXC7 DMA_INT_HTX7 DMA_INT_ERR7 DMA_INT_GLB8 DMA_INT_TXC8 DMA_INT_HTX8 DMA_INT_ERR8
Set channel request mapping	void DMA_RequestRemap(uint32_t DMAy_REMAP, DMA_Module* DMAy, DMA_ChannelType* DMAyChx, FunctionalState Cmd)	void DMA_RequestRemap(uint32_t DMA_Remap, DMA_ChannelType* DMAChx, FunctionalState Cmd)	NO	The N32H47x_48x series does not have the DMAy parameter; The first argument input may vary. When using, users can input parameters as required based on the peripherals in use. There are up to 148 mapping relationships, which are not listed here one by one. For specifics, please refer to the channel selection register in the user manual or the relevant macro definitions in the DMA driver header file.

The N32H47x_48x series has added burst function, and a corresponding example project has been synchronously added to the SDK. If users wish to utilize this function, they can refer to the BURST example in the SDK, as shown in the figure below:

:ts > n32h47x_48x_EVAL > examples > DMA >			▼ ↺	🔍 搜索"DMA"
名称	修改日期	类型		
 BURST	2024/5/13 14:11	文件夹		
 FLASH_RAM	2024/5/13 14:11	文件夹		
 I2C_RAM	2024/5/13 14:11	文件夹		
 SPI_RAM	2024/5/13 14:11	文件夹		

If the function is not to be used, users must follow the relevant descriptions in the "Channel Configuration Process" section of the user manual.

4.3.4 CRC

The following table compares the CRC module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code that calls the CRC module driver API functions can be updated according to the substitutions listed in the table below.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
CRC reset	void CRC32_ResetCrc(void);	void CRC32_ResetCrc(void);	YES	None
Calculate the 32-bit CRC for a given 32-bit data word	uint32_t CRC32_CalcCrc(uint32_t Data);	uint32_t CRC32_CalcCrc(uint32_t Data);	YES	None
Calculate 32-bit CRC(32-bit) for a given data word buffer	uint32_t CRC32_CalcBufCrc(uint32_t pBuffer[], uint32_t BufferLength);	uint32_t CRC32_CalcBufCrc(const uint32_t pBuffer[], uint32_t BufferLength);	YES	None
Get 32bit CRC value	uint32_t CRC32_GetCrc(void);	uint32_t CRC32_GetCrc(void);	YES	None
Stores 8 bits of data in a separate data (ID) register	Void CRC32_SetIDat(uint8_t IDValue);	void CRC32_SetIDat(uint8_t IDValue);	YES	None
Get 8 bits of data in a separate data (ID) register	uint8_t CRC32_GetIDat(void);	uint8_t CRC32_GetIDat(void);	YES	None
Calculate a 16-bit CRC for a given byte of data (8 bits)	uint16_t CRC16_CalcCRC(uint8_t Data);	void PWR_WakeUpPinPolarity(uint32_t pin, WAKEUP_PIN_POL polarity)	YES	None
Calculate 16-bit CRC(16-bit) for a given byte data buffer	uint16_t CRC16_CalcBufCrc(uint8_t pBuffer[], uint32_t BufferLength);	uint16_t CRC16_CalcBufCrc(const uint8_t pBuffer[], uint32_t BufferLength);	YES	None
8 bits of data are written to the 16-bit CRC data register.	-	void __CRC16_SetCrc(uint8_t Data);	NO	N32G45x series does not have this API
Get 16bit CRC value	-	uint16_t __CRC16_GetCrc(void);	NO	N32G45x series does not have this API
Write the initial value of the check register (LRC)	-	void __CRC16_SetLRC(uint8_t Data);	NO	N32G45x series does not have this API
Gets the current check register value (LRC)	-	uint8_t __CRC16_GetLRC(void)	NO	N32G45x series does not have this API
Set the calculate data to little-endian format	-	void __CRC16_SetLittleEndianFmt(void);	NO	N32G45x series does not have this API
Set the calculate data to big-endian format	-	void __CRC16_SetBigEndianFmt(void);	NO	N32G45x series does not have this API
Enable Clean CRC16 value	-	void __CRC16_SetCleanEnable(void);	NO	N32G45x series does not have this API
Disable Clean CRC16 value	-	void __CRC16_SetCleanDisable(void);	NO	N32G45x series does not have this API

CRC use process is basically the same, N32H47x_48x series provides 16bit, 32bit CRC calculation examples, but also provides the DMA transmission data calculation CRC worth examples, users can refer to the example code.

```

183 int main(void)
184 {
185     /* Enable CRC clock */
186     RCC_EnableAHBPeriphClk(RCC_AHB_PERIPHEN_CRC, ENABLE);
187
188     log_init();
189
190     /* Compute the 32bit CRC of "DataBuffer" */
191     CRC32Value = CRC32_CalcBufCrc((uint32_t*)DataBuffer, BUFFER_SIZE);
192
193     /* Compute the 32bit CRC of "DataBuffer" by software*/
194     CRCValue = GetCRC32_Software((uint32_t*)DataBuffer, BUFFER_SIZE, 0xffffffff);
195
196     if( CRC32Value == CRCValue )
197     {
198         printf(" CRC32 calculation is correct! \r\n");
199     }else
200     {
201         printf(" CRC32 calculation error! \r\n");
202     }
203
204     /* Compute the 16bit CRC of an array */
205     CRC16Value = CRC16_CalcBufCrc((uint8_t*)Buffer, 8);
206
207     /* Compute the 16bit CRC of an array by software*/
208     CRCValue = GetCRC16_Software((uint8_t*)Buffer, 8, 0x00);
209
210     if( CRC16Value == CRCValue )
211     {
212         printf("CRC16 calculation is correct! \r\n");
213     }else
214     {
215         printf("CRC16 calculation error! \r\n");
216     }
217
218     while (1)
219     {
220     }
221 }
222

```

```

69 int main(void)
70 {
71     /* Enable CRC and DMA clock */
72     RCC_EnableAHBPeriphClk(RCC_AHB_PERIPHEN_CRC|RCC_AHB_PERIPHEN_DMA1, ENABLE);
73
74     log_init();
75
76     printf(" CRC DMA test start.\r\n");
77
78     CRC32_ResetCrc();
79
80     TestCrc32AndDma(0,2);
81     TestCrc16AndDma(0,2);
82
83     while (1)
84     {
85     }
86 }
87

```

4.3.5 SPI/I2S

The following table compares the driver API functions for the SPI/I2S modules between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code can replace the SPI/I2S module driver API function calls according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
SPI reset	void SPI_I2S_DeInit(SPI_Module* SPIx);	void SPI_I2S_DeInit(const SPI_Module* SPIx);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Initialize SPI	void SPI_Init(SPI_Module* SPIx, SPI_InitType* SPI_InitStruct);	void SPI_Init(SPI_Module* SPIx,const SPI_InitType* SPI_InitStruct);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter:

				SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Initialize I2S	void I2S_Init(SPI_Module* SPIx, I2S_InitType* I2S_InitStruct);	void I2S_Init(SPI_Module* SPIx, const I2S_InitType* I2S_InitStruct);	YES	None
Initialize SPI_InitStruct members	void SPI_InitStruct(SPI_InitType* SPI_InitStruct);	void SPI_InitStruct(SPI_InitType* SPI_InitStruct);	YES	None
Initialize I2S_InitStruct members	void I2S_InitStruct(I2S_InitType* I2S_InitStruct);	void I2S_InitStruct(I2S_InitType* I2S_InitStruct);	YES	None
Enable SPI	void SPI_Enable(SPI_Module* SPIx, FunctionalState Cmd);	void SPI_Enable(SPI_Module* SPIx, FunctionalState Cmd);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Enable I2S	void I2S_Enable(SPI_Module* SPIx, FunctionalState Cmd);	void I2S_Enable(SPI_Module* SPIx, FunctionalState Cmd);	YES	None
Enable SPI/I2S interrupt	void SPI_I2S_EnableInt(SPI_Module* SPIx, uint8_t SPI_I2S_IT, FunctionalState Cmd);	void SPI_I2S_EnableInt(SPI_Module* SPIx, uint8_t SPI_I2S_IT, FunctionalState Cmd);	NO	Input parameter SPIx, SPI_I2S_IT are inconsistent: N32G45x 系列 SPIx 参数: SPI1/SPI2/SPI3 N32H47x_48x 系列 SPIx 参数: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6 N32G45x 系列 SPI_I2S_IT 参数: SPI_I2S_INT_TE SPI_I2S_INT_RNE SPI_I2S_INT_ERR N32H47x_48x 系列 SPI_I2S_IT 参数: SPI_I2S_INT_TE SPI_I2S_INT_RNE SPI_I2S_INT_ERR SPI_I2S_INT_RXONLYC SPI_I2S_INT_RXFIFOH SPI_I2S_INT_RXFIFOHF SPI_I2S_INT_TXFIFOHE
Enable SPIx/I2Sx DMA	void SPI_I2S_EnableDma(SPI_Module* SPIx, uint16_t SPI_I2S_DMAReq, FunctionalState Cmd);	void SPI_I2S_EnableDma(SPI_Module* SPIx, uint16_t SPI_I2S_DMAReq, FunctionalState Cmd);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
SPIx/I2Sx transmit data	void SPI_I2S_TransmitData(SPI_Module* SPIx, uint16_t Data);	void SPI_I2S_TransmitData(SPI_Module* SPIx, uint16_t Data);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Receive SPIx/I2Sx data	uint16_t SPI_I2S_ReceiveData(SPI_Module* SPIx);	void SPI_SetNssLevel(SPI_Module* SPIx, uint16_t SPI_NSSInternalSoft);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Set The NSS level	void SPI_SetNssLevel(SPI_Module* SPIx, uint16_t SPI_NSSInternalSoft);	void SPI_SetNssLevel(SPI_Module* SPIx, uint16_t SPI_NSSInternalSoft);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6

Enable SPI SS output	void SPI_SSOutputEnable(SPI_Module* SPiX, FunctionalState Cmd);	void SPI_SSOutputEnable(SPI_Module* SPiX, FunctionalState Cmd);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Configure SPI data length	void SPI_ConfigDataLen(SPI_Module* SPiX, uint16_t DataLen);	void SPI_ConfigDataLen(SPI_Module* SPiX, uint16_t DataLen);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Transmit SPiX CRC value	void SPI_TransmitCrcNext(SPI_Module* SPiX);	void SPI_TransmitCrcNext(SPI_Module* SPiX, FunctionalState Cmd);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Enable calculate CRC value	void SPI_EnableCalculateCrc(SPI_Module* SPiX, FunctionalState Cmd);	void SPI_EnableCalculateCrc(SPI_Module* SPiX, FunctionalState Cmd);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Get SPI CRC value	uint16_t SPI_GetCRCDat(SPI_Module* SPiX, uint8_t SPI_CRC);	uint16_t SPI_GetCRCDat(const SPI_Module* SPiX, uint8_t SPI_CRC);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Get SPI CRC polynomial register value	uint16_t SPI_GetCRCPoly(SPI_Module* SPiX);	uint16_t SPI_GetCRCPoly(const SPI_Module* SPiX);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Select the data transmission direction in bidirectional mode	void SPI_ConfigBidirectionalMode(SPI_Mod ule* SPiX, uint16_t DataDirection);	void SPI_ConfigBidirectionalMode(SPI_Modul e* SPiX, uint16_t DataDirection);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6
Get SPI/I2S status	FlagStatus SPI_I2S_GetStatus(SPI_Module* SPiX, uint16_t SPI_I2S_FLAG);	FlagStatus SPI_I2S_GetStatus(const SPI_Module* SPiX, uint16_t SPI_I2S_FLAG);	NO	Input parameter SPiX are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/S PI6 N32G45x serial parameter SPI_I2S_FLAG: SPI_I2S_TE_FLAG SPI_I2S_RNE_FLAG SPI_I2S_BUSY_FLAG SPI_I2S_OVER_FLAG SPI_MODERR_FLAG SPI_CRCERR_FLAG I2S_UNDER_FLAG I2S_CHSIDE_FLAG

				N32G45x serial parameter SPI_I2S_FLAG; SPI_I2S_BUSY_FLAG SPI_I2S_OVER_FLAG SPI_MODERR_FLAG SPI_CRCERR_FLAG I2S_UNDER_FLAG I2S_CHSIDE_FLAG SPI_I2S_TE_FLAG SPI_I2S_RNE_FLAG SPI_I2S_RXONLYC_FLAG SPI_I2S_RXFIFO_FLAG SPI_I2S_TXFIFOE_FLAG SPI_I2S_RXFIFOHF_FLAG SPI_I2S_TXFIFOHE_FLAG
Clear SPIx CRC error (CRCERR) flag	void SPI_I2S_ClrCRCErrFlag(SPI_Module* SPIx, uint16_t SPI_I2S_FLAG);	void SPI_I2S_ClrCRCErrFlag(SPI_Module* SPIx, uint16_t SPI_I2S_FLAG);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Get SPI/I2S interrupt flag	INTStatus SPI_I2S_GetIntStatus(SPI_Module* SPIx, uint8_t SPI_I2S_IT);	INTStatus SPI_I2S_GetIntStatus(const SPI_Module* SPIx, uint8_t SPI_I2S_IT);	NO	Input parameter SPIx, SPI_I2S_IT are inconsistent: Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6 N32G45x 系列 SPI_I2S_IT 参数: SPI_I2S_INT_TE SPI_I2S_INT_RNE SPI_I2S_INT_OVER SPI_INT_MODERR SPI_INT_CRCERR I2S_INT_UNDER N32H47x_48x serial parameter SPI_I2S_IT: SPI_I2S_INT_TE SPI_I2S_INT_RNE SPI_I2S_INT_ERR SPI_I2S_INT_RXONLYC SPI_I2S_INT_RXFIFOHF SPI_I2S_INT_TXFIFOHE
Clear SPIx CRC error(CRCERR) interrupt pending bit	void SPI_I2S_ClrITPendingBit(SPI_Module* SPIx, uint8_t SPI_I2S_IT);	void SPI_I2S_ClrITPendingBit(SPI_Module* SPIx, uint8_t SPI_I2S_IT);	NO	Input parameter SPIx are inconsistent: N32G45x serial parameter: SPI1/SPI2/SPI3 N32H47x_48x serial parameter: SPI1/SPI2/SPI3/SPI4/SPI5/SPI6
Enable SPI FIFO	-	void SPI_I2S_FIFO_Cmd(SPI_Module* SPIx, FunctionalState NewState);	NO	The N32G45x series does not have this function
Clear SPIx FIFO bit	-	void SPI_I2S_ClearFIFOBit(SPI_Module* SPIx, uint16_t SPI_I2S_FIFO_Clear);	NO	The N32G45x series does not have this function
Configure SPI Rx FIFO size	-	void SPI_RxFIFOSizeConfig(SPI_Module* SPIx, uint16_t SPI_FIFOSize)	NO	The N32G45x series does not have this function
Configure SPI Tx FIFO size	-	void SPI_TxFIFOSizeConfig(SPI_Module* SPIx, uint16_t SPI_FIFOSize)	NO	The N32G45x series does not have this function
Gets the number active receiving FIFOs	-	uint16_t SPI_RX_FIFO_CNT_GET(const SPI_Module* SPIx);	NO	The N32G45x series does not have this function
Gets the number of active sending FIFOs	-	uint16_t SPI_TX_FIFO_CNT_GET(const SPI_Module* SPIx);	NO	The N32G45x series does not have this function
Set transmit data number		void SPI_TRANSM_SET(SPI_Module* SPIx, uint16_t Data);	NO	The N32G45x series does not have this function

Get transmit data number	-	uint16_t SPI_TRANSMUM_GET(const SPI_Module* SPIx);	NO	The N32G45x series does not have this function
Configure SPI main clock delay time	-	void SPI_DELAYTIME_SET(SPI_Module* SPIx, uint16_t Data);	NO	The N32G45x series does not have this function
Get SPI main clock delay time	-	uint16_t SPI_DELAYTIME_GET(const SPI_Module* SPIx);	NO	The N32G45x series does not have this function
Set SPI RX FIFO data	-	void SPI_RX_FIFO_SET(SPI_Module* SPIx, uint16_t Data);	NO	The N32G45x series does not have this function
Get SPI RX FIFO data	-	uint16_t SPI_RX_FIFO_GET(const SPI_Module* SPIx);	NO	The N32G45x series does not have this function
Initlize I2S_EXT struct	-	void I2S_EXT_Init(I2S_EXT_Module* I2Sx_EXT, const I2S_InitType* I2S_EXT_InitStruct);	NO	The N32G45x series does not have this function
Enable I2S EXT	-	void I2S_EXT_Enable(I2S_EXT_Module* I2Sx_EXT, FunctionalState Cmd);	NO	The N32G45x series does not have this function
I2Sx_EXT transmit data	-	void I2S_EXT_TransmitData(I2S_EXT_Module* I2Sx_EXT, uint16_t Data);	NO	The N32G45x series does not have this function
Get I2Sx_EXT receive data	-	uint16_t I2S_EXT_ReceiveData(const I2S_EXT_Module* I2Sx);	NO	The N32G45x series does not have this function
Enable I2Sx DMA	-	void I2S_EXT_EnableDma(I2S_EXT_Module* I2Sx, uint16_t I2S_EXT_DMAREq, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enable I2Sx_EXT interrupt	-	void I2S_EXT_EnableInt(I2S_EXT_Module* I2Sx, uint8_t I2S_EXT_IT, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Get I2S_EXT status	-	FlagStatus I2S_EXT_GetStatus(const I2S_EXT_Module* I2Sx, uint16_t I2S_EXT_FLAG);	NO	The N32G45x series does not have this function
Get I2S_EXT interrupt status	-	INTStatus I2S_EXT_GetIntStatus(const I2S_EXT_Module* I2Sx, uint8_t I2S_EXT_IT);	NO	The N32G45x series does not have this function

N32H47x_48x series and N32G45x series SPI use process is basically the same, but their macro definition is very different, it is recommended that users directly refer to the N32H47x_48x series driver.

```

n32g45x_spi.h
15  This parameter can be a value of @ref I2S_Clock_Polarity */
16  I2S_InitType;
17
18  /**
19   * @
20   */
21
22  /** @addtogroup SPI_Exported_Constants
23   * @
24   */
25
26  #define IS_SPI_PERIPH(PERIPH) (((PERIPH) == SPI1) || ((PERIPH) == SPI2) || ((PERIPH) == SPI3))
27
28  #define IS_SPI_OR3_PERIPH(PERIPH) (((PERIPH) == SPI2) || ((PERIPH) == SPI3))
29
30  /** @addtogroup SPI_data_direction
31   * @
32   */
33
34  #define SPI_DIR_DOUBLELINE_FULDDUPLEX ((uint16_t)0x0000)
35  #define SPI_DIR_DOUBLELINE_ROONLY ((uint16_t)0x0400)
36  #define SPI_DIR_SINGLELINE_RX ((uint16_t)0x0800)
37  #define SPI_DIR_SINGLELINE_TX ((uint16_t)0xC000)
38  #define IS_SPI_DIR_MODE(MODE) (((MODE) == SPI_DIR_DOUBLELINE_FULDDUPLEX) || ((MODE) == SPI_DIR_DOUBLELINE_ROONLY) || ((MODE) == SPI_DIR_SINGLELINE_RX) || ((MODE) == SPI_DIR_SINGLELINE_TX))
39
40  /**
41   * @
42   */
43
44  /** @addtogroup SPI_mode
45   * @
46   */
47
48  #define SPI_MODE_MASTER ((uint16_t)0x0104)
49  #define SPI_MODE_SLAVE ((uint16_t)0x0000)
50  #define IS_SPI_MODE(MODE) (((MODE) == SPI_MODE_MASTER) || ((MODE) == SPI_MODE_SLAVE))
51
52  /**
53   * @
54   */
55
56  /** @addtogroup SPI_data_size
57   * @
58   */
59
60  #define SPI_DATA_SIZE_1BITS ((uint16_t)0x0800)
61  #define SPI_DATA_SIZE_8BITS ((uint16_t)0x0000)
62  #define IS_SPI_DATASIZE(DATASIZE) (((DATASIZE) == SPI_DATA_SIZE_1BITS) || ((DATASIZE) == SPI_DATA_SIZE_8BITS))
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
237
```

名称	修改日期	类型	大小	名称	修改日期	类型	大小
CRC	2023/7/17 16:59	文件夹		CRC	2024/4/1 9:59	文件夹	
CRC_Remap	2023/7/17 16:59	文件夹		CRC_FIFO	2024/4/1 9:59	文件夹	
FullDuplex_SoftNSS	2023/7/17 16:59	文件夹		FullDuplex_SoftNSS	2024/4/1 9:59	文件夹	
Simplex_Interrupt	2023/7/17 16:59	文件夹		Simplex_Interrupt	2024/4/1 9:59	文件夹	
SPI_DMA	2023/7/17 16:59	文件夹		SPI_DMA	2024/4/1 9:59	文件夹	
SPI_DMA_T&R	2023/7/17 16:59	文件夹		SPI_DMA_T&R	2024/4/1 9:59	文件夹	
SPI_FLASH	2023/7/17 16:59	文件夹		SPI_FLASH	2024/4/1 9:59	文件夹	
SPI_FLASH_DMA	2023/7/17 16:59	文件夹		SPI_FLASH_DMA	2024/4/1 9:59	文件夹	
SPI_RECORDER	2023/7/17 16:59	文件夹					

able is a comparison

--	--

				I2C_INT_ERR N32H47x 48x serial add two parameters more than N32G45x serial parameter: I2C_REG_BYTENUM I2C_REG_GFLTRCTRL
Enable I2C soft reset	void I2C_EnableSoftwareReset(I2C_Module* I2Cx, FunctionalState Cmd);	void I2C_EnableSoftwareReset(I2C_Module* I2Cx, FunctionalState Cmd);	YES	None
Select the I2C NACK position in master receiver mode	void I2C_ConfigNackLocation(I2C_Module* I2Cx, uint16_t I2C_NACKPosition);	void I2C_ConfigNackLocation(I2C_Module* I2Cx, uint16_t I2C_NACKPosition);	YES	None
Configure I2CSMBus alert	void I2C_ConfigSmbusAlert(I2C_Module* I2Cx, uint16_t I2C_SMBusAlert);	void I2C_ConfigSmbusAlert(I2C_Module* I2Cx, uint16_t I2C_SMBusAlert);	YES	None
Enable I2C send PEC	void I2C_SendPEC(I2C_Module* I2Cx, FunctionalState Cmd);	void I2C_SendPEC(I2C_Module* I2Cx, FunctionalState Cmd);	YES	None
Configure I2C PEC location	void I2C_ConfigPecLocation(I2C_Module* I2Cx, uint16_t I2C_PECPosition);	void I2C_ConfigPecLocation(I2C_Module* I2Cx, uint16_t I2C_PECPosition);	YES	None
Enable calculate PEC value.	void I2C_ComputePec(I2C_Module* I2Cx, FunctionalState Cmd);	void I2C_ComputePec(I2C_Module* I2Cx, FunctionalState Cmd);	YES	None
Get PEC value	uint8_t I2C_GetPec(I2C_Module* I2Cx);	uint8_t I2C_GetPec(const I2C_Module* I2Cx);	YES	None
Enable I2C ARP	I2C_EnableArp(I2C_Module* I2Cx, FunctionalState Cmd);	void I2C_EnableArp(I2C_Module* I2Cx, FunctionalState Cmd);	YES	None
Enable I2C clock extend	void I2C_EnableExtendClk(I2C_Module* I2Cx, FunctionalState Cmd);	void I2C_EnableExtendClk(I2C_Module* I2Cx, FunctionalState Cmd);	YES	None
Configure the I2C fast mode duty cycle	void I2C_ConfigFastModeDutyCycle(I2C_Mod ule* I2Cx, uint16_t FmDutyCycle);	void I2C_ConfigFastModeDutyCycle(I2C_Module* I2Cx, uint16_t FmDutyCycle);	YES	None
I2C check FIFO event	-	ErrorStatus I2C_CheckFifoEvent(I2C_Module* I2Cx, uint32_t I2C_FIFO_EVENT);	NO	The N32G45x series does not have this function
Get I2C last FIFO event	-	uint32_t I2C_GetLastFifoEvent(I2C_Module* I2Cx);	NO	The N32G45x series does not have this function
Clear I2C FIFO	-	void I2C_ClrFIFO(I2C_Module* I2Cx);	NO	The N32G45x series does not have this function
Enable FIFO	-	void I2C_EnableFIFO(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enable DMA FIFO	-	void I2C_EnableDMAFIFO(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enable I2C counting byte	-	void I2C_EnableBYTENUM(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Configure the final state of the master after sending the required data bytes	-	void I2C_SetByteNumLastStartStop(I2C_Module* I2Cx, uint16_t LastStatus);	NO	The N32G45x series does not have this function
Sets the number of bytes of data that the I2C peripheral receiver will receive	-	void I2C_SetReceivedDataBytesNum(I2C_Module* I2Cx, uint16_t Number_Of_bytes);	NO	The N32G45x series does not have this function
Set I2C Tx FIFO threshold	-	void I2C_SetTxFifoThreshold(I2C_Module* I2Cx, uint8_t TxFifoThreshold);	NO	The N32G45x series does not have this function
Set I2C Rx FIFO threshold	-	void I2C_SetRxFifoThreshold(I2C_Module* I2Cx, uint8_t RxFifoThreshold);	NO	The N32G45x series does not have this function
Set I2C FIFO DATA	-	void I2C_SetFIFODAT(I2C_Module* I2Cx, uint32_t I2C_BYTENUM);	NO	The N32G45x series does not have this function
Get I2C FIFO DATA	-	uint8_t I2C_GetFIFODAT(const I2C_Module* I2Cx);	NO	The N32G45x series does not have this function
Enables the minimum timeout period	-	void I2C_EnableLowTimeout(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enables the maximum timeout period	-	void I2C_EnableHighTimeout(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enable SCL analog filter	-	void I2C_EnableSCLAnalogFilter(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Enable SDA analog filter	-	void I2C_EnableSDAAnalogFilter(I2C_Module* I2Cx, FunctionalState Cmd);	NO	The N32G45x series does not have this function
Set SCL analog filter width	-	void I2C_SetSCLAnalogFilterWidth(I2C_Module* I2Cx, uint32_t width);	NO	The N32G45x series does not have this function
Set SDA analog filter width	-	void I2C_SetSDAAnalogFilterWidth(I2C_Module* I2Cx, uint32_t width);	NO	The N32G45x series does not have this function
Set SDA digital filter width	-	void I2C_SetSDADigitalFilterWidth(I2C_Module* I2Cx, uint32_t width);	NO	The N32G45x series does not have this function

Set SCL digital filter width	-	void I2C_SetSCLDigitalFilterWidth(I2C_Module* I2Cx, uint32_t width);	NO	The N32G45x series does not have this function
Check I2Cx event	-	ErrorStatus I2C_CheckEvent(const I2C_Module* I2Cx, uint32_t I2C_EVENT);	NO	The N32G45x series does not have this function
Get I2Cx last event	-	uint32_t I2C_GetLastEvent(const I2C_Module* I2Cx);	NO	The N32G45x series does not have this function
Get I2C flag	-	FlagStatus I2C_GetFlag(const I2C_Module* I2Cx, uint32_t I2C_FLAG);	NO	The N32G45x series does not have this function
Clear I2C flag	-	void I2C_ClrFlag(I2C_Module* I2Cx, uint32_t I2C_FLAG);	NO	The N32G45x series does not have this function
Get I2C interrupt status	-	INTStatus I2C_GetIntStatus(const I2C_Module* I2Cx, uint32_t I2C_IT);	NO	The N32G45x series does not have this function
Clear I2Cx interrupt pending bit	-	void I2C_ClrIntPendingBit(I2C_Module* I2Cx, uint32_t I2C_IT);	NO	The N32G45x series does not have this function

The usage process for the I2C in the N32H47x_48x series and the N32G45x series is largely the same, but there are significant differences in their macro definitions. It is recommended that users directly refer to the drivers for the N32H47x_48x series.

Additionally, the N32H47x_48x series has added FIFO data transmission and reception functionality to its I2C compared to the N32G45x series. Therefore, FIFO-related demos have been included in the demo. As shown in the diagram below, the N32G45x series (left) and the N32H47x_48x series (right):

名称	修改日期	类型	大小	名称	修改日期	类型	大小
EEPROM_DMA	2023/7/17 16:58	文件夹		EEPROM_DMA	2024/4/16 14:21	文件夹	
EEPROM_Int	2023/7/17 16:58	文件夹		EEPROM_Int	2024/4/16 14:21	文件夹	
EEPROM_Polling	2023/7/17 16:58	文件夹		EEPROM_Polling	2024/4/16 14:21	文件夹	
I2C_10bit	2023/7/17 16:59	文件夹		I2C_10bit	2024/4/16 15:52	文件夹	
I2C_Master	2023/7/17 16:59	文件夹		I2C_Master	2024/4/16 14:21	文件夹	
I2C_Master_Int	2023/7/17 16:59	文件夹		I2C_Master_FIFO	2024/4/16 14:21	文件夹	
I2C_Slave	2023/7/17 16:59	文件夹		I2C_Master_Int	2024/4/16 14:21	文件夹	
I2C_Slave_Int	2023/7/17 16:59	文件夹		I2C_Slave	2024/4/16 14:21	文件夹	
				I2C_Slave_FIFO	2024/4/16 14:21	文件夹	
				I2C_Slave_Int	2024/4/16 14:21	文件夹	

4.3.7 ALGO

The ALGO module driver has been encapsulated into a library, and you can use it by simply calling the interface functions. Both the N32H47x_48x series and the N32G45x series provide demo programs for AES, DES, HASH, Rand, and MD5 algorithms. Additionally, the N32H47x_48x series also offers demo programs for SM4 and SM3 invocations. It is recommended that users directly refer to the demo programs in the N32H47x_48x series development kit.

Below is a comparison of the example project code for the "main" function of the ALGO module between the N32H47x_48x series (left) and the N32G45x series (right):

```

73 /**
74  **\name      main.
75  **\fun       Main program.
76  **\param     none
77  **\return    none
78  */
79 int main(void)
80 {
81     log_init();
82     log_info("-----\nAlgorithm demo start.\n");
83
84     SM4_test();
85     AES_128_test();
86     AES_192_test();
87     AES_256_test();
88     DES_test();
89     TDES_2Key_test();
90     TDES_3Key_test();
91     HASH_test();
92     SM3_test();
93     MD5_test();
94     TestRand();
95
96     log_info("\r\nALGO demo all test OK!\r\n");
97     while (1)
98     ;
99
100 }

```

```

371 }
372
373 /**
374  * @brief main function.
375  */
376 int main(void)
377 {
378     log_init();
379     log_info("-----\nAlgorithm demo start.\n");
380
381     // RNG test
382     TestRand();
383
384     // HASH test
385     TestMD5();
386     TestSHA1();
387     TestSHA224();
388     TestSHA256();
389
390     // Cryptogram algorithm
391     TestDES();
392     TestAES();
393
394     log_info("\r\nALGO demo all test OK!\r\n");
395     while (1)
396     ;
397
398 }

```

4.3.8 ETH

The ETH module IP in the N32H47x_48x series has undergone an upgrade, but there are no impacts on its Ethernet functionality. Notably, the ETH registers differ significantly from those in the N32G45x series, leading to substantial differences in the corresponding driver functions, including variations in programming styles. Additionally, when developing for ETH, besides the driver .c/.h files, other necessary files are also required, which are completely different from those in the N32G45x series, and even absent in the N32G45x series. Therefore, during development, it is recommended that users refer to the examples provided in the SDK, as illustrated in the figure below:

ts > n32h47x_48x_EVAL > examples > ETH >	
名称	修改日期
DHCP	2024/5/13 14:11
DNS	2024/5/13 14:11
HTTP_Client	2024/5/13 14:11
HTTP_ControlLEDs	2024/5/13 14:11
HTTP_Server	2024/5/13 14:11
MagicPacketWakeUp	2024/5/13 14:11
NetBIOS	2024/5/13 14:11
PING	2024/5/13 14:11
RemoteWakeUp	2024/5/13 14:11
TCP_Client	2024/5/13 14:11
TCP_Client_RAW	2024/5/13 14:11
TCP_Server	2024/5/13 14:11
TCP_Server_RAW	2024/5/13 14:11
UDP	2024/5/13 14:11
UDP_RAW	2024/5/13 14:11

The examples currently provided include both types with and without an operating system (routines such as NetBIOS, PING, TCP Client RAW, TCP Server RAW, and UDP RAW do not require an operating system, while others are based on FreeRTOS version 202212.01). Users can refer to the corresponding routines according to their needs. Other protocol routines are still under continuous improvement and will be released with subsequent versions.

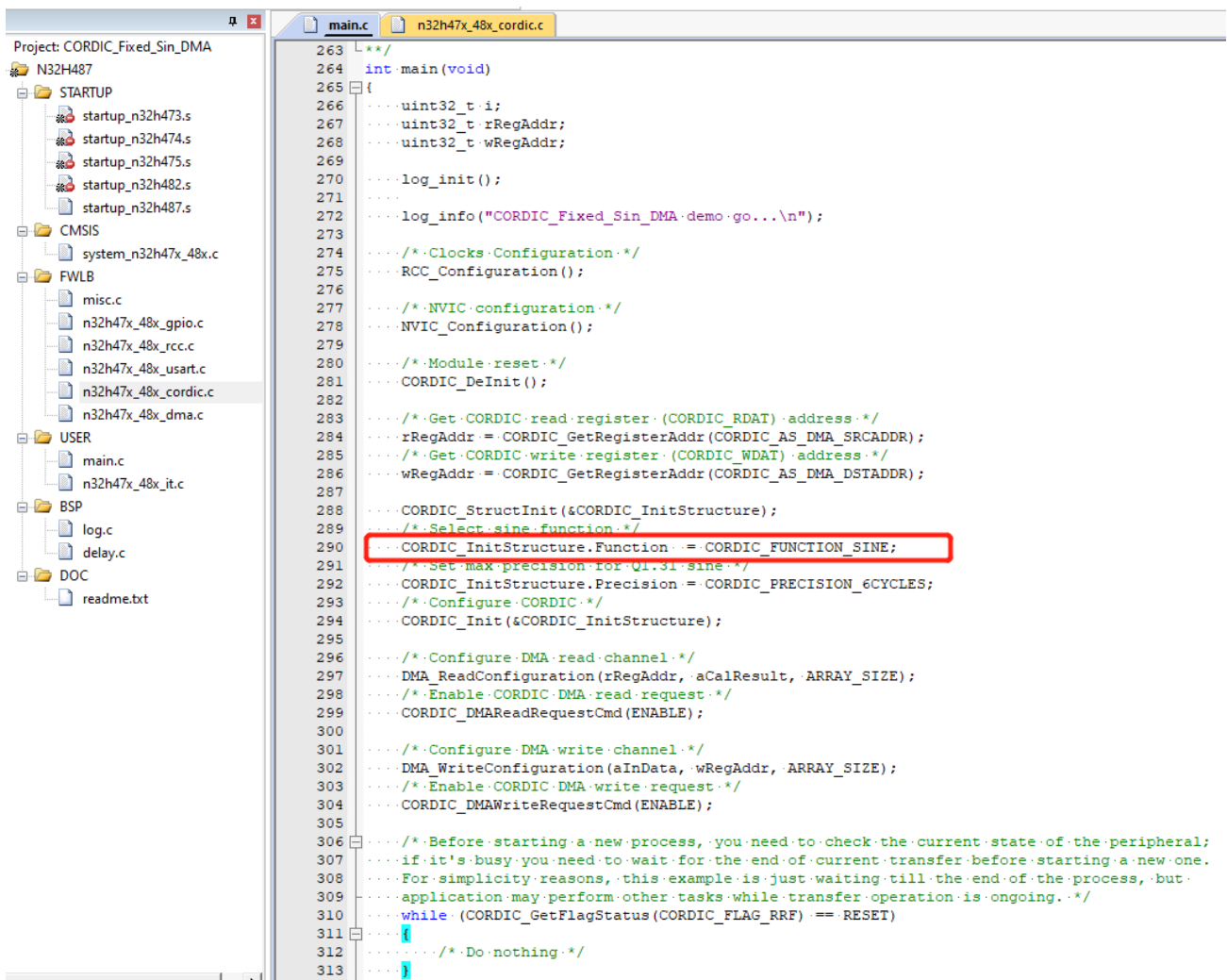
Note: When referring to the examples, it is recommended to pay attention to the precautions mentioned in the readme file of the corresponding routine.

4.3.9 CORDIC

Compared to the N32G45x series, the CORDIC module is a new addition, designed for mathematical function calculations (primarily trigonometric functions). In the SDK, using the sine function as an example, demo projects are provided for three scenarios: "fixed-point input + DMA mode", "fixed-point input + interrupt mode", and "floating-point input + interrupt mode", as shown in the figure below:

名称	修改日期
CORDIC_Fixed_Sin_DMA	2024/5/13 14:11
CORDIC_Fixed_Sin_IT	2024/5/13 14:11
CORDIC_Float_Sin_IT	2024/5/13 14:11

Users can refer to the routines corresponding to their input type and operating mode, and modify them to suit the required mathematical function, for example, by modifying the portion highlighted in red in the routine shown in the figure below.



4.3.10 LPTIM

Compared to the N32G45x series, LPTIM is a newly added module. The timer in this module can operate in all power consumption modes and has the capability to wake up the system (except in VBAT mode). The SDK provides a demo project as shown in the figure below:

cts > n32h47x_48x_EVAL > examples > LPTIM >

名称	修改日期
LPTIM_ENC	2024/5/13 14:11
LPTIM_NENC	2024/5/13 14:11
LPTIM_PulseCounter	2024/5/13 14:11
LPTIM_PWM	2024/5/13 14:11
LPTIM_WakeUp	2024/5/13 14:11

If user need to calculate the number of signal edges in quadrature encoder mode, refer to the "LPTIM_ENC" demo. If user need to calculate the number of signal edges in non-quadrature encoder mode, refer to the "LPTIM_NENC" demo.

If user need to count the number of pulses, refer to the "LPTIM_PulseCounter" demo.

If user need to generate PWM waveforms, refer to the "LPTIM_PWM" demo.

If user need to operate and wake up from low-power modes, refer to the "LPTIM_WakeUp" demo. For information on how to enter other low-power modes besides the one demonstrated, refer to the corresponding demos of the PWR module.

4.3.11 GPIO

The basic configuration and port remapping of GPIO differ significantly. For a detailed comparison of the GPIO module driver API functions between the N32G45x series and the N32H47x_48x series, please refer to the table below. After replacing the driver .c/.h files, you can make corresponding substitutions in your application code when calling the GPIO module driver API functions, according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
GPIO reset	void GPIO_DeInit(GPIO_Module* GPIOx)	void GPIO_DeInit(GPIO_Module* GPIOx)	YES	Input parameter The value range of GPIOx is different: N32H47x_48x series adds GPIOH option
Deinitialize GPIO pin	-	void GPIO_DeInitPin(GPIO_Module* GPIOx, uint32_t Pin)	NO	N32G45x series does not have this API
AFIO reset	void GPIO_AFIOInitDefault(void)	void GPIO_AFIOInitDefault(void)	YES	None
Initialize GPIO	void GPIO_InitPeripheral(GPIO_Module* GPIOx, GPIO_InitType* GPIO_InitStruct)	void GPIO_InitPeripheral(GPIO_Module* GPIOx, GPIO_InitType * GPIO_InitStruct)	NO	The function name is the same as the input parameter, but the configuration methods of the two GPIOs are different, and the structure definition of the input parameter is different. Please refer to the user manual and SDK: N32G45x series structure: Pin, GPIO_Speed, GPIO_Mode N32H47x_48x series structure :Pin, GPIO_Mode, GPIO_Pull, GPIO_Slew_Rate, GPIO_Current, GPIO_Alternate
Initialize GPIO structs	void GPIO_InitStruct(GPIO_InitType* GPIO_InitStruct)	void GPIO_InitStruct(GPIO_InitType* GPIO_InitStruct)	YES	The structure definitions that the input parameters point to vary widely
Get GPIO input data bit	uint8_t GPIO_ReadInputDataBit(GPIO_Module* GPIOx, uint16_t Pin)	uint8_t GPIO_ReadInputDataBit(GPIO_Modul e* GPIOx, uint16_t Pin)	YES	Input parameter The value range of GPIOx is different:

Get GPIO input data	uint16_t GPIO_ReadInputData(GPIO_Module* GPIOx)	uint16_t GPIO_ReadInputData(GPIO_Module* GPIOx)	YES	N32H47x_48x series adds GPIOH option
Get GPIO output data bit	uint8_t GPIO_ReadOutputDataBit(GPIO_Module* GPIOx, uint16_t Pin)	uint8_t GPIO_ReadOutputDataBit(GPIO_Module* GPIOx, uint16_t Pin)	YES	Input parameter The value range of GPIOx is different:
Get GPIO output data	uint16_t GPIO_ReadOutputData(GPIO_Module* GPIOx)	uint16_t GPIO_ReadOutputData(GPIO_Module* GPIOx)	YES	N32H47x_48x series adds GPIOH option
Configure GPIO pin output high level	void GPIO_SetBits(GPIO_Module* GPIOx, uint16_t Pin)	void GPIO_SetBits(GPIO_Module* GPIOx, uint16_t Pin)	YES	Input parameter The value range of GPIOx is different:
Configure GPIO output high level	void GPIO_SetBitsHigh16(GPIO_Module* GPIOx, uint32_t Pin)	-	NO	N32H47x_48x series delet this function
Configure GPIO pin output low level	void GPIO_ResetBits(GPIO_Module* GPIOx, uint16_t Pin)	void GPIO_ResetBits(GPIO_Module* GPIOx, uint16_t Pin)	YES	Input parameter The value range of GPIOx is different:
Configure GPIO pin output level	void GPIO_WriteBit(GPIO_Module* GPIOx, uint16_t Pin, Bit_OperateType BitCmd)	void GPIO_WriteBits(GPIO_Module* GPIOx, uint16_t Pin, Bit_OperateType BitCmd)	YES	N32H47x_48x series adds GPIOH option
Configure GPIO output level	void GPIO_Write(GPIO_Module* GPIOx, uint16_t PortVal)	void GPIO_Write(GPIO_Module* GPIOx, uint16_t data_value)	YES	Input parameter The value range of GPIOx is different:
Toggle GPIO pin level	-	void GPIO_TogglePin(GPIO_Module* GPIOx, uint16_t Pin)	NO	N32H47x_48x series add this function
Lock GPIO pin	void GPIO_ConfigPinLock(GPIO_Module* GPIOx, uint16_t Pin)	void GPIO_ConfigPinLock(GPIO_Module* GPIOx, uint16_t Pin)	YES	Input parameter The value range of GPIOx is different:
Configure EVENT output GPIO pin	void GPIO_ConfigEventOutput(uint8_t PortSource, uint8_t PinSource)	-	NO	N32H47x_48x series adds GPIOH option
Enable EVENT output	void GPIO_CtrlEventOutput(FunctionalState Cmd)	-	NO	The N32H47x_48x series does not need to be enabled
Configure GPIO EXTI line	void GPIO_ConfigEXTILine(uint8_t PortSource, uint8_t PinSource)	void GPIO_ConfigEXTILine(uint8_t LineSource, uint8_t PortSource, uint8_t PinSource)	NO	N32H47x_48x series EXTI input supports full pin mapping, added input parametersLineSource: EXTI_LINE_SOURCE0~15
Configure GPIO pin remapping	void GPIO_ConfigPinRemap(uint32_t RmpPin, FunctionalState Cmd)	void GPIO_ConfigPinRemap(uint8_t PortSource, uint8_t PinSource, uint32_t AlternateFunction)	NO	The N32G45x series must be remapped according to the peripheral port group, while the N32H47x_48x series can be remapped separately for each port, the difference is very big, please refer to the user manual and SDK
Configure SPI NSS port mode	-	void AFIO_Config_SPI_NSS_Mode(uint32_t AFIO_SPIx_NSS, uint32_t NSS_Mode)	NO	N32H47x_48x series add this function
Configure IO filter cycle	-	void AFIO_ConfigIOFilter(uint32_t Filter_Cycle)	NO	N32H47x_48x series add this function
Configure FEMC NADV pin	-	void AFIO_Config_FEMC_NADV_Pin(uint32_t NADV_Pin)	NO	N32H47x_48x series add this function
Configure EXTI analog filter	-	void AFIO_Config_EXTI_Filter(uint32_t EXTI_Filter)	NO	N32H47x_48x series add this function
Configure xSPI XIP data endian	-	void AFIO_Config_XSPI_XIP_BigEndian(uint32_t Endian)	NO	N32H47x_48x series add this function
Configure xSPI half-duplex mode	-	void AFIO_Config_XSPI_HalfDuplexMode(uint32_t HalfDuplex)	NO	N32H47x_48x series add this function
Configure xSPI dual quad mode	-	void AFIO_Config_XSPI_DualQuadMode(uint32_t DualQuad)	NO	N32H47x_48x series add this function

Configure ETH interface mode	void GPIO_ETH_ConfigMediaInterface(uint32_t ETH_ConfigSel)	void AFIO_Config_ETH_Mode(uint32_t ETH_Mode)	NO	Function name and input parameter macrodefinition are different: N32G45x Series Input parameter ETH_ConfigSel: GPIO_ETH_RMII_CFG GPIO_ETH_MII_CFG N32H47x_48x Series Input parameters ETH_Mode: AFIO_ETH_MODE_RMII AFIO_ETH_MODE_MII
Configure the xSPI NSS port input function	-	void AFIO_Config_XSPI_NSS_Input(uint32_t InputEnable, uint32_t InputPin)	NO	N32H47x_48x series add this function
Configure the xSPI RXDS signal sampling delay	-	void AFIO_Config_XSPI_RXDS_SampleDelay(uint32_t Delay)	NO	N32H47x_48x series add this function
Configure xSPI Non-XIP mode read data endian	-	void AFIO_Config_XSPI_BigEndian_Read(uint32_t Endian)	NO	N32H47x_48x series add this function
Configure xSPI Non-XIP mode write data endian	-	void AFIO_Config_XSPI_BigEndian_Write(uint32_t Endian)	NO	N32H47x_48x series add this function
Configure xSPI time extension	-	void AFIO_Config_XSPI_TimeExtension(uint32_t Extension)	NO	N32H47x_48x series add this function
Configure xSPI high level time	-	void AFIO_Config_XSPI_NSS_HighTime(uint32_t clk)	NO	N32H47x_48x series add this function
Enable port 5V tolerance	-	void AFIO_ConfigPinTol5V(GPIO_Module * GPIOx, uint32_t Pin, FunctionalState cmd)	NO	N32H47x_48x series add this function
Configure port digital filtering function	-	void AFIO_ConfigPinFilter(GPIO_Module * GPIOx, uint32_t Pin, FunctionalState cmd)	NO	N32H47x_48x series add this function
Configure GPIOC register read delay function	-	void AFIO_Config_GPIOC_ReadRegDelay(FunctionalState cmd)	NO	N32H47x_48x series add this function
Configure the SHRTIM external event port	-	void AFIO_Config_SHRTIM_EXEV_Pin(uint8_t EXEV_Line, uint8_t PortSource, uint8_t PinSource)	NO	N32H47x_48x series add this function
Configure EMC function	-	void AFIO_Config_EMC_Fun(uint32_t EMC_fun, FunctionalState cmd)	NO	N32H47x_48x series add this function
Configure EMC count	-	void AFIO_Set_EMC_Cnt(uint32_t cnt)	NO	N32H47x_48x series add this function
Get EMC count	-	uint32_t AFIO_Get_EMC_Cnt(void)	NO	N32H47x_48x series add this function

The AFIO section of the GPIO module in the N32H47x_48x series has more new features, with the main differences as follows:

1. Configuration of the global digital filtering period for GPIO ports, with the digital filtering function for each port able to be enabled/disabled individually.
2. External interrupts (EXTI) can be configured on any pin.
3. EXTI analog filtering can be configured.
4. Some pins can be configured for 5V compatibility.
5. The FEMC NADV pin connection can be configured.

6. Remapping of ADC regular/injected conversion triggers is no longer supported; instead, this configuration is handled within the ADC module.
7. Additional features for the xSPI module include dual four-wire mode, half-duplex mode, data endianness mode, and NSS input control in master mode.
8. Added SHRTIM external event pin configuration, supporting full pin mapping.

4.3.12 EXTI

The EXTI modules of the N32G45x series and the N32H47x_48x series are basically the same, with some adjustments made to the register definitions. Additionally, the number of external interrupt lines has been increased from 22 to 32, and full pin mapping for external interrupts is supported (configured in AFIO). For a detailed comparison of the driver API functions, please refer to the table below. After replacing the driver .c/.h files, you can make corresponding substitutions in your application code when calling the EXTI module driver API functions, according to the table below.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
EXTI reset	void EXTI_DeInit(void)	void EXTI_DeInit(void)	YES	None
EXTI Initialize	void EXTI_InitPeripheral(EXTI_InitType* EXTI_InitStruct)	void EXTI_InitPeripheral(EXTI_InitType* EXTI_InitStruct)	YES	None
Initialize EXTI structs	void EXTI_InitStruct(EXTI_InitType* EXTI_InitStruct)	void EXTI_InitStruct(EXTI_InitType* EXTI_InitStruct)	YES	None
Trigger EXTI software interrupts	void EXTI_TriggerSWInt(uint32_t EXTI_Line)	void EXTI_TriggerSWInt(uint32_t EXTI_Line)	YES	The value range of input parameter EXTI_Line is different:
Get EXTI status	FlagStatus EXTI_GetStatusFlag(uint32_t EXTI_Line)	FlagStatus EXTI_GetStatusFlag(uint32_t EXTI_Line)	YES	N32G45x series: EXTI_LINE0 to EXTI_LINE21
Clear EXTI status	void EXTI_ClrStatusFlag(uint32_t EXTI_Line)	void EXTI_ClrStatusFlag(uint32_t EXTI_Line)	YES	N32H47x_48x series: EXTI_LINE0 to EXTI_LINE31
Get EXTI interrupt status	INTStatus EXTI_GetITStatus(uint32_t EXTI_Line)	INTStatus EXTI_GetITStatus(uint32_t EXTI_Line)	YES	The value range of input parameter EXTI_Line is different:
Clear EXTI interrupt pending bit	void EXTI_ClrITPendBit(uint32_t EXTI_Line)	void EXTI_ClrITPendBit(uint32_t EXTI_Line)	YES	N32G45x series: EXTI_LINE0 to EXTI_LINE21
Select the timestamp trigger source	void EXTI_RTCTimeStampSel(uint32_t EXTI_TSEL_Line)	void EXTI_RTCTimeStampSel(uint32_t EXTI_TSEL_Line)	YES	None

4.3.13 SDIO

The SDIO modules of the N32G45x series and the N32H47x_48x series are basically the same, with adjustments made to the register definitions. For a detailed comparison of the driver API functions, please refer to the table below. After replacing the driver .c/.h files, the application code can call the SDIO module driver API functions according to the substitutions outlined in the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
SDIO reset	void SDIO_DeInit(void)	void SDIO_DeInit(void)	YES	None
Initialize SDIO	void SDIO_Init(SDIO_InitType* SDIO_InitStruct)	void SDIO_Init(SDIO_InitType* InitStruct)	NO	Initialization structure member ClkDiv value range has changed: N32G45x series: 0x00 ~ 0xFF N32H47x_48x series: 0x00 ~ 0x1FF Initialization structure member ClkDiv value range has changed:

				N32G45x series: 0x00 ~ 0xFF N32H47x_48x series: 0x00 ~ 0x1FF
Initialize SDIO structs	void SDIO_InitStruct(SDIO_InitType* SDIO_InitStruct)	void SDIO_InitStruct(SDIO_InitType* InitStruct)	NO	Initialization structure member ClkDiv value range has changed: N32G45x series: 0x00 ~ 0xFF N32H47x_48x series: 0x00 ~ 0x1FF Initialization structure member ClkDiv value range has changed: N32G45x series: 0x00 ~ 0xFF N32H47x_48x series: 0x00 ~ 0x1FF
Enable SDIO clock output	void SDIO_EnableClock(FunctionalState Cmd)	void SDIO_EnableClock(FunctionalState Cmd)	YES	None
Set SDIO power	void SDIO_SetPower(uint32_t SDIO_PowerState)	void SDIO_SetPower(uint32_t PowerState)	YES	None
Get SDIO power	uint32_t SDIO_GetPower(void)	uint32_t SDIO_GetPower(void)	NO	The return value is different: N32G45x Series: 0x00, 0x02, 0x03 N32H47x_48x Series: SDIO_POWER_OFF SDIO_POWER_UP SDIO_POWER_ON
Configure SDIO interrupt	void SDIO_ConfigInt(uint32_t SDIO_IT, FunctionalState Cmd)	void SDIO_ConfigInt(uint32_t Int, FunctionalState Cmd)	NO	The first input parameter has changed: The N32H47x_48x series does not support SDIO_INT_CEATAF
Enable SDIO DMA	void SDIO_DMAMCmd(FunctionalState Cmd)	void SDIO_DMAMCmd(FunctionalState Cmd)	YES	None
Send SDIO command	void SDIO_SendCmd(SDIO_CmdInitType* SDIO_CmdInitStruct)	void SDIO_SendCmd(SDIO_CmdInitType* CmdInitStruct)	YES	None
Initialize SDIO command structs	void SDIO_InitCmdStruct(SDIO_CmdInitType* SDIO_CmdInitStruct)	void SDIO_InitCmdStruct(SDIO_CmdInitType* CmdInitStruct)	YES	None
Gets the command index of the last response received	uint8_t SDIO_GetCmdResp(void)	uint8_t SDIO_GetCmdResp(void)	YES	None
Gets the last response received	uint32_t SDIO_GetResp(uint32_t SDIO_RESP)	uint32_t SDIO_GetResp(uint32_t SDIO_RESP)	YES	None
Configure the data transfer control register	void SDIO_ConfigData(SDIO_DataInitType* SDIO_DataInitStruct)	void SDIO_ConfigData(SDIO_DataInitType* DataInitStruct)	YES	None
Data structure initialization	void SDIO_InitDataStruct(SDIO_DataInitType* SDIO_DataInitStruct)	void SDIO_InitDataStruct(SDIO_DataInitType* SDIO_DataInitStruct)	YES	None
Gets the amount of outstanding transfer data	uint32_t SDIO_GetDataCountValue(void)	uint32_t SDIO_GetDataCountValue(void)	YES	None
Read data from the receiving FIFO	uint32_t SDIO_ReadData(void)	uint32_t SDIO_ReadData(void)	YES	None
Write data to send FIFO	void SDIO_WriteData(uint32_t Data)	void SDIO_WriteData(uint32_t Data)	YES	None
Gets the number of words of FIFO data to be written or read	uint32_t SDIO_GetFifoCounter(void)	uint32_t SDIO_GetFifoCounter(void)	YES	None
Enabled read wait function	void SDIO_EnableReadWait(FunctionalState Cmd)	void SDIO_EnableReadWait(FunctionalState Cmd)	YES	None
Start/stop read wait	void SDIO_DisableReadWait(FunctionalState Cmd)	-	NO	The N32H47x_48x series is split into two functions void SDIO_StopReadWait(void) void SDIO_StartReadWait(void)

Stop read wait	-	void SDIO_StopReadWait(void)	NO	N32H47x_48x series new functions
Configure the read wait mode	void SDIO_EnableSdioReadWaitMode(uint32_t SDIO_ReadWaitMode)	void SDIO_EnableReadWaitMode(uint32_t ReadWaitMode)	YES	None
Configure the SD I/O mode	void SDIO_EnableSdioOperation(FunctionalState Cmd)	void SDIO_EnableSdioOperation(FunctionalState Cmd)	YES	None
The SD I/O pause command was configured	void SDIO_EnableSendSdioSuspend(FunctionalState Cmd)	void SDIO_EnableSendSdioSuspend(FunctionalState Cmd)	YES	None
Configure the command completion signal	void SDIO_EnableCommandCompletion(FunctionalState Cmd)	void SDIO_EnableCommandCompletion(FunctionalState Cmd)	YES	None
Enable CE-ATA device interrupted	void SDIO_EnableCEATAInt(FunctionalState Cmd)	-	NO	N32H47x_48x does not support CE-ATA devices
Configure CE-ATA command sending	void SDIO_EnableSendCEATA(FunctionalState Cmd)	-	NO	N32H47x_48x does not support CE-ATA devices
Get SDIO status	FlagStatus SDIO_GetFlag(uint32_t SDIO_FLAG)	FlagStatus SDIO_GetFlag(uint32_t Flag)	NO	The input parameters have changed: The N32H47x_48x series does not support SDIO_FLAG_CEATAF
Get SDIO interrupt status	INTStatus SDIO_GetIntStatus(uint32_t SDIO_IT)	INTStatus SDIO_GetIntStatus(uint32_t Int)	NO	The input parameters have changed: The N32H47x_48x series does not support SDIO_INT_CEATAF
Clear SDIO flag	void SDIO_ClrFlag(uint32_t SDIO_FLAG)	void SDIO_ClrFlag(uint32_t FlagClr)	NO	The input parameters have changed: The N32H47x_48x series does not support SDIO_FLAG_CEATAF
Clear SDIO interrupt pending bit	void SDIO_ClrIntPendingBit(uint32_t SDIO_IT)	-	NO	The N32H47x_48x series was merged into SDIO_ClrFlag
Send the command and get R1 short response	-	uint32_t SDIO_CmdShortResponse1(uint8_t Cmd, uint32_t Arg, uint32_t timeout)	NO	N32H47x_48x 系列新增函数
Send the command and get R3 short response	-	uint32_t SDIO_CmdShortResponse3(uint8_t Cmd, uint32_t Arg)	NO	N32H47x_48x series new functions
Send the command and get an R2 long response	-	uint32_t SDIO_CmdLongResponse2(uint8_t Cmd, uint32_t Arg)	NO	N32H47x_48x series new functions
Send CMD16 and get the response	-	uint32_t SDIO_CmdBlockLength(uint32_t BlockSize)	NO	N32H47x_48x series new functions
Send CMD23 and get the response	-	uint32_t SDIO_CmdBlockCount(uint32_t BlockCnt)	NO	N32H47x_48x series new functions
Send CMD17 and get the response	-	uint32_t SDIO_CmdReadSingleBlock(uint32_t ReadAdd)	NO	N32H47x_48x series new functions
Send CMD18 and get the response	-	uint32_t SDIO_CmdReadMultiBlock(uint32_t ReadAdd)	NO	N32H47x_48x series new functions
Send CMD24 and get the response	-	uint32_t SDIO_CmdWriteSingleBlock(uint32_t WriteAdd)	NO	N32H47x_48x series new functions
Send CMD25 and get the response	-	uint32_t SDIO_CmdWriteMultiBlock(uint32_t WriteAdd)	NO	N32H47x_48x series new functions
Send CMD32 and get the response	-	uint32_t SDIO_CmdSEraseStartAdd(uint32_t StartAdd)	NO	N32H47x_48x series new functions
Send CMD33 and get the response	-	uint32_t SDIO_CmdSEraseEndAdd(uint32_t EndAdd)	NO	N32H47x_48x series new functions
Send CMD35 and get the response	-	uint32_t SDIO_CmdEraseStartAdd(uint32_t StartAdd)	NO	N32H47x_48x series new functions
Send CMD36 and get the response	-	uint32_t SDIO_CmdEraseEndAdd(uint32_t EndAdd)	NO	N32H47x_48x series new functions

Send CMD38 and get the response	-	uint32_t SDIO_CmdErase(void)	NO	N32H47x_48x series new functions
Send CMD42 and get a response	-	uint32_t SDIO_CmdLockUnlock(uint32_t arg)	NO	N32H47x_48x series new functions
Send CMD12 and get the response	-	uint32_t SDIO_CmdStopTransfer(void)	NO	N32H47x_48x series new functions
Send CMD7 and get the response	-	uint32_t SDIO_CmdSelDesel(uint16_t RCA)	NO	N32H47x_48x series new functions
Send CMD0 and get the response	-	uint32_t SDIO_CmdGoIdleState(void)	NO	N32H47x_48x series new functions
Send CMD8 and get the response	-	uint32_t SDIO_CmdOperCond(void)	NO	N32H47x_48x series new functions
Send CMD55 and get the response	-	uint32_t SDIO_CmdAppCommand(uint32_t Arg)	NO	N32H47x_48x series new functions
Send ACMD41 and get the response	-	uint32_t SDIO_CmdAppOperCommand(uint32_t Arg)	NO	N32H47x_48x series new functions
Send ACMD6 and get the response	-	uint32_t SDIO_CmdBusWidth(uint32_t BusWidth)	NO	N32H47x_48x series new functions
Send ACMD51 and get the response	-	uint32_t SDIO_CmdSendSCR(void)	NO	N32H47x_48x series new functions
Send CMD2 and get the response	-	uint32_t SDIO_CmdAllSendCID(void)	NO	N32H47x_48x series new functions
Send CMD10 and get the response	-	uint32_t SDIO_CmdSendCID(uint16_t RCA)	NO	N32H47x_48x series new functions
Send CMD9 and get the response	-	uint32_t SDIO_CmdSendCSD(uint16_t RCA)	NO	N32H47x_48x series new functions
Send CMD3 to SD card to get response and RCA	-	uint32_t SDIO_CmdSetRelAdd(uint16_t *pRCA)	NO	N32H47x_48x series new functions
Send CMD3 to MMC card and get response	-	uint32_t SDIO_CmdSetRelAddMmc(uint16_t RCA)	NO	N32H47x_48x series new functions
Send CMD13 and get the response	-	uint32_t SDIO_CmdSendStatus(uint32_t Arg)	NO	N32H47x_48x series new functions
Send ACMD13 and get the response	-	uint32_t SDIO_CmdStatusRegister(void)	NO	N32H47x_48x series new functions
Send CMD1 and get a response	-	uint32_t SDIO_CmdOpCondition(uint32_t Arg)	NO	N32H47x_48x series new functions
Send CMD6 and get the response	-	uint32_t SDIO_CmdSwitch(uint32_t Arg)	NO	N32H47x_48x series new functions
Send CMD8 and get the response	-	uint32_t SDIO_CmdSendEXTCSD(uint32_t Arg)	NO	N32H47x_48x series new functions
Gets R1 response error status	-	uint32_t SDIO_GetCmdResp1(uint8_t Cmd, uint32_t Timeout)	NO	N32H47x_48x series new functions
Get R2 response error status	-	uint32_t SDIO_GetCmdResp2(void)	NO	N32H47x_48x series new functions
Gets R3 response error status	-	uint32_t SDIO_GetCmdResp3(void)	NO	N32H47x_48x series new functions
Gets the R6 response error status	-	uint32_t SDIO_GetCmdResp6(uint8_t Cmd, uint16_t *pRCA)	NO	N32H47x_48x series new functions
Gets R7 response error status	-	uint32_t SDIO_GetCmdResp7(void)	NO	N32H47x_48x series new functions
Gets the CMD0 send error status	-	static uint32_t SDIO_GetCmdError(void)	NO	N32H47x_48x series new functions

4.3.14 FDCAN

The FDCAN module of the N32H47x_48x series complies with the ISO 11898-1:2015 standard, supporting both CAN 2.0A/B and CAN FD protocols, as well as being compatible with the non-ISO standard Bosch protocol. In contrast, the CAN module of the corresponding N32G45x series only supports CAN 2.0A/B.

Due to significant differences between the two, they cannot be directly compared or modified through direct substitution. When using them, please refer to the FDCAN module section in the user manual and the FDCAN routines in the SDK.

Additionally, the FDCAN port supports full pin mapping, as detailed in the description of multiplexing functions in the GPIO module section of the user manual.

4.3.15 IWDG

The main differences in the IWDG (Independent Watchdog) module are as follows:

1. The N32G45x series does not support freeze functionality, while the N32H47x_48x series does.
2. The IWDG clock source is the LSI (Low-Speed Internal). For the N32G45x series, the LSI clock frequency is 40 kHz, with a minimum reset time of 0.1 ms. In contrast, for the N32H47x_48x series, the LSI clock frequency is 32 kHz, and the minimum reset time is 0.125 ms. Therefore, the timeout value needs to be recalculated.

Below is a comparison table of the IWDG module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code can call the IWDG module driver API functions according to the substitutions outlined in the table. Note that the timeout values need to be converted, and for more details, please refer to the user manual:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
Set the IWDG_PREDIV and IWDG_RELV registers to write protection	void IWDG_WriteConfig(uint16_t IWDG_WriteAccess)	void IWDG_WriteConfig(IWDOG_WRITE_CONFIG IWDG_WriteAccess)	YES	None
Set the clock predivision factor that drives the IWDG	void IWDG_SetPrescalerDiv(uint8_t IWDG_Prescaler)	void IWDG_SetPrescalerDiv(uint8_t IWDG_Prescaler)	YES	None
Set the IWDG reloading value	void IWDG_CntReload(uint16_t Reload)	void IWDG_CntReload(uint16_t Reload)	YES	None
Loads the value in the reload register to the capital counter	void IWDG_ReloadKey(void)	void IWDG_ReloadKey(void)	YES	None
Enable IWDG	void IWDG_Enable(void)	void IWDG_Enable(void)	YES	None
Enable IWDG freeze	-	void IWDG_Freeze_Enable(FunctionalState Cmd)	NO	N32G45x series does not have this API
Get IWDG status	FlagStatus IWDG_GetStatus(uint16_t IWDG_FLAG)	FlagStatus IWDG_GetStatus(IWDG_STATUS_FLAG IWDG_FLAG)	YES	None

4.3.16 WWDG

The main differences in the WWDG (Window Watchdog) module are as follows:

1. The window value for the N32G45x series is 7 bits, while for the N32H47x_48x series, it is 14 bits.
2. The WWDG clock source is APB1/4096. For the N32G45x series, the APB1 clock has a maximum frequency of 36 MHz, resulting in a minimum reset time of 0.113 ms. In contrast, for the N32H47x_48x series, when the main clock frequency is 240 MHz, the APB1 clock frequency is 120 MHz, leading to a minimum reset time of 0.0341 ms. Therefore, the window value needs to be recalculated.

Below is a comparison table of the WWDG module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code can call the WWDG module driver API functions according to the substitutions outlined in the table. Note that the window time needs to be converted, and for more details, please refer to the user manual:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
WWDG reset	void WWDG_DeInit(void)	void WWDG_DeInit(void)	YES	None

Set the pre-division factor for the WWDG driver clock	void WWDG_SetPrescalerDiv(uint32_t WWDG_Prescaler)	void WWDG_SetPrescalerDiv(uint32_t WWDG_Prescaler)	YES	None
Set window value	void WWDG_SetWVValue(uint8_t WindowValue)	void WWDG_SetWVValue(uint16_t WindowValue)	NO	The N32G45x input value is 7bit, and the N32G47x_48x input value is 14bit
Enabled WWDG Early wake up interrupt	void WWDG_EnableInt(void)	void WWDG_EnableInt(void)	YES	None
Set counter value	void WWDG_SetCnt(uint8_t Counter)	void WWDG_SetCnt(uint16_t Counter)	NO	The N32G45x input value is 7bit, and the N32G47x_48x input value is 14bit
The WWDG function was enabled	void WWDG_Enable(uint8_t Counter)	void WWDG_Enable(uint16_t Counter)	NO	The N32G45x input value is 7bit, and the N32G47x_48x input value is 14bit
Gets the WWDG Wake up Early interrupt flag	FlagStatus WWDG_GetEWINTF(void)	FlagStatus WWDG_GetEWINTF(void)	YES	None
Clear the WWDG Wake Up Early interrupt flag	void WWDG_ClrEWINTF(void)	void WWDG_ClrEWINTF(void)	YES	None

4.3.17 FLASH

The following table compares the FLASH module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code can call the FLASH module driver API functions with substitutions according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
Set Latency	void FLASH_SetLatency(uint32_t FLASH_Latency)	void FLASH_SetLatency(uint32_t FLASH_Latency)	YES	N32H47x_48x series add parameter FLASH_LATENCY_5
Get Latency	-	uint8_t FLASH_GetLatency(void)	否	N32H47x_48x series new functions
Set prefetch buffer	void FLASH_PrefetchBufSet(uint32_t FLASH_PrefetchBuf)	void FLASH_PrefetchBufSet(uint32_t FLASH_PrefetchBuf)	YES	None
Icache reset	void FLASH_iCacheRST(void)	void FLASH_iCacheRST(void)	YES	None
Set Icache command	void FLASH_iCacheCmd(uint32_t FLASH_iCache)	void FLASH_iCacheCmd(uint32_t FLASH_iCache)	YES	None
Set FLASH programming mode	void FLASH_SetSMPSELStatus(uint32_t FLASH_smpsel)	-	NO	N32H47x_48x series don't have this API.
FLASH unlock	void FLASH_Unlock(void)	void FLASH_Unlock(void)	YES	None
FLASH lock	void FLASH_Lock(void)	void FLASH_Lock(void)	YES	None
Get FLASH lock status	-	FlagStatus Flash_GetLockStatus(void)	NO	N32H47x_48x series new functions
Option byte unlock	-	void Option_Bytes_Unlock(void)	NO	N32H47x_48x series new functions
Option byte locks	-	void Option_Bytes_Lock(void)	NO	N32H47x_48x series new functions
Get option byte lock status	-	FlagStatus OB_GetLockStatus(void)	NO	N32H47x_48x series new functions
FLASH erase one page	FLASH_STS FLASH_EraseOnePage(uint32_t Page_Address)	FLASH_STS FLASH_EraseOnePage(uint32_t Page_Address)	YES	N32G45x page size: 2KB N32H47x_48x page size: 8KB
FLASH mass erase	FLASH_STS FLASH_MassErase(void)	FLASH_STS FLASH_MassErase(void)	YES	None
FLASH program word	FLASH_STS FLASH_ProgramWord(uint32_t Address, uint32_t Data)	FLASH_STS FLASH_ProgramdoubleWord(uint32_t address, uint32_t data0, uint32_t data1)	NO	Single Word programming to double word programming, you need to enter two Word data at the same time
Set FLASH ROW program	-	void FLASH_RowProgramSet(FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set FLASH ROW area programming	-	void FLASH_RowProgramAreaSet(FunctionalState Cmd)	NO	N32H47x_48x series new functions
FLASH ROW	-	FLASH_STS FLASH_RowProgram(uint32_t address, uint32_t row_num, uint32_t *data)	NO	N32H47x_48x series new functions
Option byte erase	FLASH_STS FLASH_EraseOB(void)	FLASH_STS FLASH_EraseOB(void)	YES	None
Programming Option byte area	FLASH_STS FLASH_ProgramOBData(uint32_t Address, uint32_t Data)	FLASH_STS FLASH_ProgramOB_RUDD(uint32_t option_byte_rpd1, uint32_t option_byte_iwdg, uint32_t option_byte_stop, \n uint32_t option_byte_stdby, uint32_t option_byte_iwdg_stop, uint32_t option_byte_iwdg_stdby, \n uint32_t option_byte_iwdg_sleep, uint32_t option_byte_data0, uint32_t option_byte_data1)	NO	Option byte area programming difference is large, it is recommended to refer to Demo to understand the use
	FLASH_STS FLASH_EnWriteProtection(uint32_t FLASH_Pages)	FLASH_STS FLASH_EnWriteProtection(uint32_t FLASH_Pages)	YES	
	FLASH_STS FLASH_ConfigUserOB(uint16_t OB_IWDG, uint16_t OB_STOP, uint16_t OB_STDBY)	FLASH_STS FLASH_ProgramOB_RU2U3(uint32_t option_byte_rpd2, uint32_t option_byte2_nBOOT0, uint32_t option_byte2_nBOOT1, \n	NO	

		uint32_t option_byte2_nSWBOOT0, uint32_t option_byte2_FlashBoot, \n uint32_t option_byte2_BOR, uint32_t option_byte3_NRST)		
	-	FLASH_STS FLASH_ProgramOB_CCMSRAM(uint32_t option_byte_CCMSRAM)	NO	
Set FLASH L1 Read protection	FLASH_STS FLASH_ReadOutProtectionL1(FunctionalS tate Cmd)	FLASH_STS FLASH_ReadOutProtectionL1(FunctionalSt ate Cmd)	YES	None
Enable FLASH L2 read protection	FLASH_STS FLASH_ReadOutProtectionL2_ENABLE(v oid)	FLASH_STS FLASH_ReadOutProtectionL2_ENABLE(v oid)	YES	None
Gets the status of option byte area USER1	uint32_t FLASH_GetUserOB(void)	uint32_t FLASH_GetUserOB(void)	YES	None
Gets the status of option byte area USER2	-	FlagStatus FLASH_GetUser2(uint32_t option_byte_bit)	NO	N32H47x_48x series new functions
Gets the status of option byte area DATA0	-	uint32_t FLASH_GetOptionBytes_Data0(void)	NO	N32H47x_48x series new functions
Gets the status of option byte area DATA1	-	uint32_t FLASH_GetOptionBytes_Data1(void)	NO	N32H47x_48x series new functions
Get FLASH write protection status	uint32_t FLASH_GetWriteProtectionSTS(void)	uint32_t FLASH_GetWriteProtectionSTS(void)	YES	None
Get status of the read protection status	FlagStatus FLASH_GetReadOutProtectionSTS(void)	FlagStatus FLASH_GetReadOutProtectionSTS(void)	YES	None
Get L2 status of read protection status	FlagStatus FLASH_GetReadOutProtectionL2STS(voi d)	FlagStatus FLASH_GetReadOutProtectionL2STS(void)	YES	None
Get Obtain the prefetched buffer status	FlagStatus FLASH_GetPrefetchBufSTS(void)	FlagStatus FLASH_GetPrefetchBufSTS(void)	YES	None
Get FLASH programming mode status	FLASH_SMPSEL FLASH_GetSMPSELStatus(void)	-	NO	N32H47x_48x series don't have this API
Configure FLASH interrupt	void FLASH_INTConfig(uint32_t FLASH_INT, FunctionalState Cmd)	void FLASH_INTConfig(uint32_t FLASH_INT, FunctionalState Cmd)	YES	FLASH_IT_ERROR→F LASH_INT_ERR N32H47x_48x series add parameters: FLASH_INT_ECC1 FLASH_INT_JS FLASH_INT_ECC2 FLASH_INT_DECC FLASH_INT_RPADD
Get the FLASH flag status	FlagStatus FLASH_GetFlagSTS(uint32_t FLASH_FLAG)	FlagStatus FLASH_GetFlagSTS(uint32_t FLASH_FLAG)	YES	N32H47x_48x series add parameters: : FLASH_FLAG_ECC1E RR FLASH_FLAG_RDKE YERR FLASH_FLAG_RDXK EYERR FLASH_FLAG_NRD XKEYEN FLASH_FLAG_JSERR FLASH_FLAG_RTPKE YERR FLASH_FLAG_ECC2E RR FLASH_FLAG_DECC RDF FLASH_FLAG_DECCE RR FLASH_FLAG_FWOR DF FLASH_FLAG_RPAD DERR
Clear FLASH status	void FLASH_ClearFlag(uint32_t FLASH_FLAG)	void FLASH_ClearFlag(uint32_t FLASH_FLAG)	YES	N32H47x_48x series add parameters: FLASH_FLAG_ECC1E RR

				FLASH_FLAG_JSERR FLASH_FLAG_RTPKEYERR FLASH_FLAG_ECC2ERR FLASH_FLAG_DECCRDF FLASH_FLAG_DECCERR FLASH_FLAG_RPADERR
Get FLASH status	FLASH_STS FLASH_GetSTS(void)	FLASH_STS FLASH_GetSTS(void)	YES	None
Wait for the FLASH operation to complete	FLASH_STS FLASH_WaitForLastOpt(uint32_t Timeout)	FLASH_STS FLASH_WaitForLastOpt(uint32_t Timeout)	YES	N32H47x_48x series add parameters: RowProgramTimeout
Set CCM Write protection	-	void CCM_EnWriteProtection(uint32_t CCM_Pages)	NO	N32H47x_48x series new functions
Get the CCM write protection status	-	uint32_t CCM_GetWriteProtectionSTS(void)	NO	N32H47x_48x series new functions
CCM erase unlock	-	void CCM_Earse_Unlock(void)	NO	N32H47x_48x series new functions
Enabled CCM erase	-	void CCM_EarseEN(void)	NO	N32H47x_48x series new functions
Gets CCM erase status	-	FlagStatus CCM_EarseSTS(void)	NO	N32H47x_48x series new functions
Configure the CCM mode	-	void CCM_ModeSet(FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set XSPI decryption address range	-	void XSPI_DESRangeSet(uint32_t start_add,uint32_t end_add)	NO	N32H47x_48x series new functions
Set FEMC decryption address range	-	void FEMC_DESRangeSet(uint32_t start_add,uint32_t end_add)	NO	N32H47x_48x series new functions
Set XPSI/FEMC decryption KEY	-	void RTP_DESKeySet(uint32_t* DES_key)	NO	N32H47x_48x series new functions
Gets the number of writes decryption KEY	-	uint32_t GetRTP_DESKeyWnum(void)	NO	N32H47x_48x series new functions
Set JTAG Seal	-	void Jtag_SealSet(FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set XPSI/FEMC decryption	-	void XSPI_FEMC_DESSet(FunctionalState Cmd)	NO	N32H47x_48x series new functions
Get the XSPI/FEMC UID	-	uint32_t Get_XFUID(void)	NO	N32H47x_48x series new functions
Get the CCM UID	-	uint32_t Get_CCMUID(void)	NO	N32H47x_48x series new functions

The following diagram shows the example project code for "OptionByte_config" in the FLASH module of the N32H47x_48x series. Before programming the option bytes, it is necessary to first perform FLASH unlocking and option byte unlocking, then erase the option bytes, and finally call the option byte programming functions in sequence to program the option byte areas.


```
int main(void)
{
    FLASH_STS state_value;

    /* USART Init */
    log_init();

    log_info("\nOption Byte configure Test start!\r\n");
    /* Unlocks the FLASH Program Erase Controller */
    FLASH_Unlock();
    /* Unlocks the Option Byte Program Erase Controller */
    Option_Bytes_Unlock();
    /* Erase Option Byte page */
    state_value = FLASH_EraseOB();

    if(state_value == FLASH_EOP)
    {
        /* Start Option Byte program */

        /* Disable read protection L1 */
        state_value = FLASH_ProgramOB_RUDD(FLASH_OB_RDP1_DISABLE, FLASH_OB_IWDG_SOFTWARE, FLASH_OB_STOP_NORST, \
                                           FLASH_OB_STDBY_NORST, FLASH_OB_IWDG_STOP_NOFRZ, FLASH_OB_IWDG_STDBY_NOFRZ, \
                                           FLASH_OB_IWDG_SLEEP_NOFRZ, 0x55, 0xAA);

        if(state_value == FLASH_EOP)
        {
            state_value = FLASH_EraseOB();
        }
        else
        {
            /* no process */
        }

        if(state_value == FLASH_EOP)
        {
            state_value = FLASH_ProgramOB_RU2U3(FLASH_OB_RDP2_DISABLE, FLASH_OB2_NBOOT0_SET, FLASH_OB2_NBOOT1_SET, \
                                                FLASH_OB2_NSBOOT0_SET, FLASH_OB2_FLASHBOOT_SET, 0x00, 0x00);
        }
        else
        {
            /* no process */
        }

        if(state_value == FLASH_EOP)
        {
            state_value = FLASH_ProgramOB_CCMRAM(CCMRAM_RST_NERASE);
        }
        else
        {
            /* no process */
        }

        if(state_value == FLASH_EOP)
        {
            log_info("Option Byte configure OK!\r\n");
        }
        else
        {
            log_info("Option Byte configure failed!\r\n");
        }
    }
    else
    {
        log_info("Option Byte erase failed!\r\n");
    }

    while (1)
    {
    }
}
```

4.3.18 MISC

The following table compares the driver API functions of the misc module between the N32G45x series and the N32H47x 48x series. After replacing the driver .c/.h files, the application code can call the misc module driver API functions with substitutions according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x 48x series		
NVIC priority group	void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)	void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)	YES	None
Configure interrupt priority	void NVIC_Init(NVIC_InitType* NVIC_InitStruct)	void NVIC_Init(NVIC_InitType* NVIC_InitStruct)	YES	None
Configure interrupt vector table location	void NVIC_SetVectorTable(uint32_t NVIC_VectTab, uint32_t Offset)	void NVIC_SetVectorTable(uint32_t NVIC_VectTab, uint32_t Offset)	YES	None
Configure low power mode	void NVIC_SystemLPConfig(uint8_t LowPowerMode, FunctionalState Cmd)	void NVIC_SystemLPConfig(uint8_t LowPowerMode, FunctionalState Cmd)	YES	None

Configure systick clock source	void SysTick_CLKSourceConfig(uint32_t SysTick_CLKSource)	void SysTick_CLKSourceConfig(uint32_t SysTick_CLKSource)	YES	None
--------------------------------	--	--	-----	------

The functionality of the misc module functions is consistent between the N32G45x series and the N32H47x_48x series, allowing for direct substitution and use.

4.3.19 USART

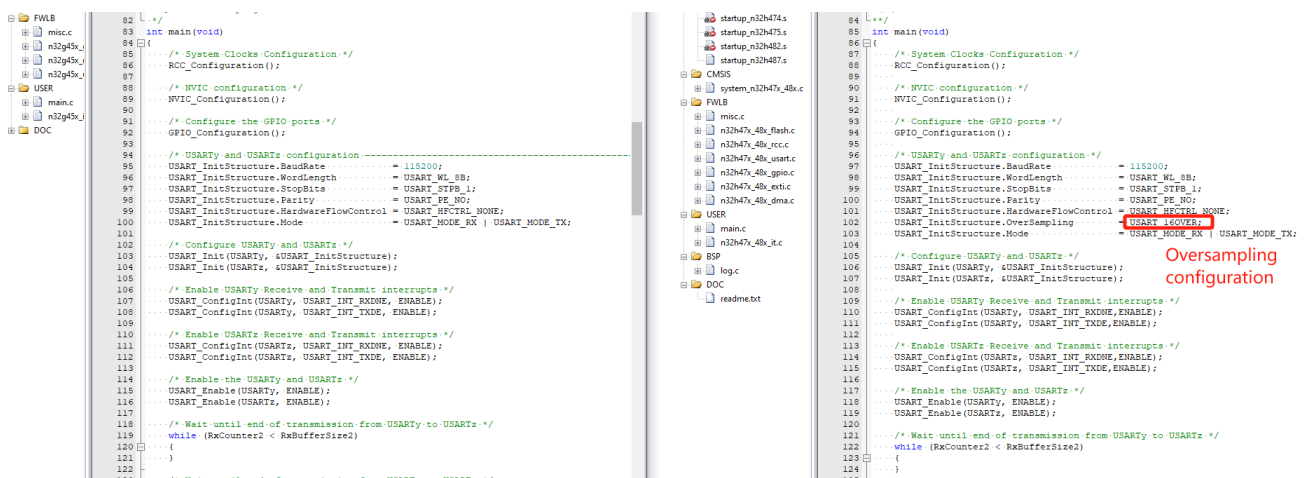
The following table compares the USART module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code can call the USART module driver API functions with substitutions according to the table below. Specifically, for the USARTx parameter differences, they are uniformly described as UART4→USART4, with the addition of UART8.

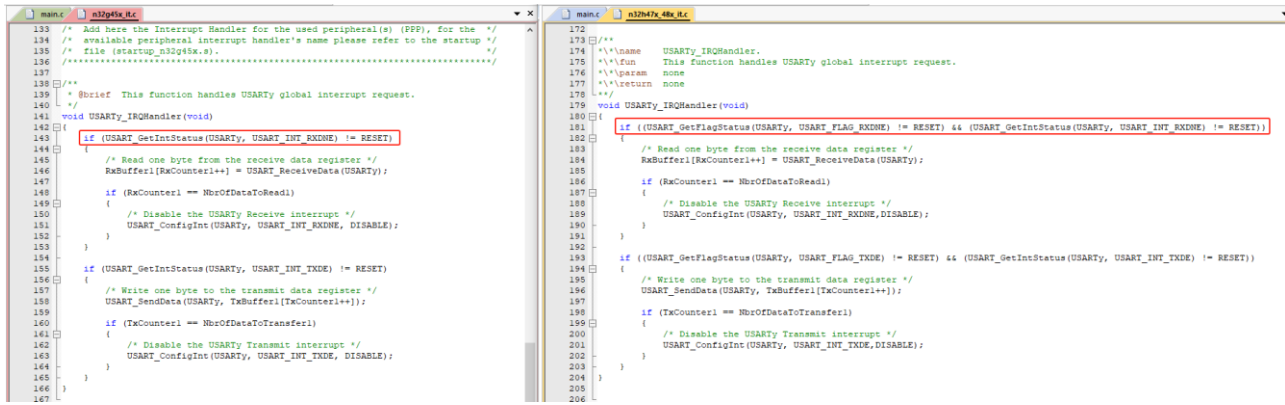
API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
USART reset	void USART_DeInit(USART_Module* USARTx)	void USART_DeInit(USART_Module* USARTx)	YES	None
Initialize USART	void USART_Init(USART_Module* USARTx, USART_InitType* USART_InitStruct)	void USART_Init(USART_Module* USARTx, USART_InitType* USART_InitStruct)	YES	N32H47x_48x maximum BaudRate increases from 1.5M to 15M, and add OverSampling parameter
Initialize USART struct	void USART_StructInit(USART_InitType* USART_InitStruct)	void USART_StructInit(USART_InitType* USART_InitStruct)	YES	N32H47x_48x add OverSampling parameter
Initialize USART clock	void USART_ClockInit(USART_Module* USARTx, USART_ClockInitType* USART_ClockInitStruct)	void USART_ClockInit(USART_Module* USARTx, USART_ClockInitType* USART_ClockInitStruct)	YES	None
Initialize USART clock struct	void USART_ClockStructInit(USART_ClockInitType* USART_ClockInitStruct)	void USART_ClockStructInit(USART_ClockInitType* USART_ClockInitStruct)	YES	None
EnableUSART	void USART_Enable(USART_Module* USARTx, FunctionalState Cmd)	void USART_Enable(USART_Module* USARTx, FunctionalState Cmd)	YES	None
Configure USART interrupt	void USART_ConfigInt(USART_Module* USARTx, uint32_t USART_INT, FunctionalState Cmd)	void USART_ConfigInt(USART_Module* USARTx, uint32_t USART_INT, FunctionalState Cmd)	YES	N32H47x_48x add parameters: USART_INT_RT0E USART_INT_TXFTE USART_INT_RXFTE USART_INT_RXFEE USART_INT_TXFEE USART_INT_RXFFE USART_INT_TXFFE
Enable USART DMA	void USART_EnableDMA(USART_Module* USARTx, uint32_t USART_DMAREq, FunctionalState Cmd)	void USART_EnableDMA(USART_Module* USARTx, uint32_t USART_DMAREq, FunctionalState Cmd)	YES	None
Set USART node address	void USART_SetAddr(USART_Module* USARTx, uint8_t USART_Addr)	void USART_SetAddr(USART_Module* USARTx, uint8_t USART_Addr)	YES	None
Set USART wake-up mode	void USART_ConfigWakeUpMode(USART_Module* USARTx, uint32_t USART_WakeUpMode)	void USART_ConfigWakeUpMode(USART_Module* USARTx, uint32_t USART_WakeUpMode)	YES	None
Set USART Silent mode wake-up	void USART_EnableRcvWakeUp(USART_Module* USARTx, FunctionalState Cmd)	void USART_EnableRcvWakeUp(USART_Module* USARTx, FunctionalState Cmd)	YES	None
Set USART LIN Idle frame length	void USART_ConfigLINBreakDetectLength(USART_Module* USARTx, uint32_t USART_LINBreakDetectLength)	void USART_ConfigLINBreakDetectLength(USART_Module* USARTx, uint32_t USART_LINBreakDetectLength)	YES	None
Set LIN mode	void USART_EnableLIN(USART_Module* USARTx, FunctionalState Cmd)	void USART_EnableLIN(USART_Module* USARTx, FunctionalState Cmd)	YES	None
Send data	void USART_SendData(USART_Module* USARTx, uint32_t Data)	void USART_SendData(USART_Module* USARTx, uint32_t Data)	YES	None
Received data	uint32_t USART_ReceiveData(USART_Module* USARTx)	uint32_t USART_ReceiveData(USART_Module* USARTx)	YES	None

Send break frame	void USART_SendBreak(USART_Module* USARTx)	void USART_SendBreak(USART_Module* USARTx)	YES	None
Set Protection time	void USART_SetGuardTime(USART_Modul e* USARTx, uint8_t USART_GuardTime)	void USART_SetGuardTime(USART_Module* USARTx, uint8_t USART_GuardTime)	YES	None
Set the system clock frequency division	void USART_SetPrescaler(USART_Module* USARTx, uint8_t USART_Prescaler)	void USART_SetPrescaler(USART_Module* USARTx, uint8_t USART_Prescaler)	YES	None
Set smart card mode	void USART_EnableSmartCard(USART_Mo dule* USARTx, FunctionalState Cmd)	void USART_EnableSmartCard(USART_Module * USARTx, FunctionalState Cmd)	YES	None
Set smart card NACK Settings	void USART_SetSmartCardNACK(USART_ Module* USARTx, FunctionalState Cmd)	void USART_SetSmartCardNACK(USART_Mod ule* USARTx, FunctionalState Cmd)	YES	None
Enable USART Half-duplex mode	void USART_EnableHalfDuplex(USART_M odule* USARTx, FunctionalState Cmd)	void USART_EnableHalfDuplex(USART_Modul e* USARTx, FunctionalState Cmd)	YES	None
Set infrared power mode	void USART_ConfigIrDAMode(USART_M odule* USARTx, uint32_t USART_IrDAMode)	void USART_ConfigIrDAMode(USART_Module * USARTx, uint32_t USART_IrDAMode)	YES	None
Set infrared mode	void USART_EnableIrDA(USART_Module* USARTx, FunctionalState Cmd)	void USART_EnableIrDA(USART_Module* USARTx, FunctionalState Cmd)	YES	None
Get the USART status	FlagStatus USART_GetFlagStatus(USART_Modul e* USARTx, uint32_t USART_FLAG)	FlagStatus USART_GetFlagStatus(USART_Module* USARTx, uint32_t USART_FLAG)	YES	N32H47x_48x add parameters: USART_FLAG_FELOSE USART_FLAG_NELOSE USART_FLAG_PELLOSE USART_FLAG_RTO USART_FLAG_TXFT USART_FLAG_RXFT USART_FLAG_RXFE USART_FLAG_TXFE USART_FLAG_RXFF USART_FLAG_TXFF
Clear the USART flag	void USART_ClrFlag(USART_Module* USARTx, uint32_t USART_FLAG)	void USART_ClrFlag(USART_Module* USARTx, uint32_t USART_FLAG)	YES	N32H47x_48x add parameters: USART_FLAG_FELOSE USART_FLAG_NELOSE USART_FLAG_PELLOSE
Clear the USART RTO flag bit	-	void USART_ClrRTOFlag(USART_Module* USARTx)	NO	N32H47x_48x series new functions
Get interrupt status	INTStatus USART_GetIntStatus(USART_Module * USARTx, uint32_t USART_INT)	INTStatus USART_GetIntStatus(USART_Module* USARTx, uint32_t USART_INT)	YES	The meaning of the function has changed: For N32G45x, this function is used to determine whether an interrupt is enabled and whether the corresponding flag bit is set. For N32H47x_48x, this function is only used to determine whether an interrupt is enabled. It needs to be used together with the USART_GetFlagStatus function to achieve the same effect as in N32G45x. N32H47x_48x add parameters: USART_INT_RTOE USART_INT_TXFTE USART_INT_RXFTE USART_INT_RXFEE USART_INT_TXFEE USART_INT_RXFFE USART_INT_TXFFE
Clear the USART interrupt pending bit	void USART_ClrIntPendingBit(USART_Mo dule* USARTx, uint16_t USART_INT)	-	NO	This function is the same as the USART_ClrFlag function
Set USART Idle frame	-	void USART_IdleFrameSet(USART_Module* USARTx, FunctionalState Cmd)	NO	N32H47x_48x series new functions

USART pin interchangeability	-	void USART_PinSwapSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set the start time of the USART drive	-	void USART_CfgDriverAssertTime(USART_Module* USARTx,uint32_t Time)	NO	N32H47x_48x series new functions
Configure USART drive deassert time	-	void USART_CfgDriverdeassertTime(USART_Module* USARTx,uint32_t Time)	NO	N32H47x_48x series new functions
Set the polarity of the USART drive	-	void USART_DriverPolaritySet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART drive mode	-	void USART_DriverModeSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART FEF data discard	-	void USART_FEFDiscardSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART NEF Data discard	-	void USART_NEFDiscardSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART PEF data discard	-	void USART_PEFDiscardSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART Receive timeout	-	void USART_RTOSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Get the number of valid USART Tx FIFO	-	uint32_t USART_GetTxFIFO_Num(USART_Module* USARTx)	NO	N32H47x_48x series new functions
Get the number of valid USART Rx FIFO	-	uint32_t USART_GetRxFIFO_Num(USART_Module* USARTx)	NO	N32H47x_48x series new functions
Configure USART Rx FIFO threshold	-	void USART_CfgRxFIFOThreshold(USART_Module* USARTx,uint32_t threshold)	NO	N32H47x_48x series new functions
Set USART Tx FIFO threshold	-	void USART_CfgTxFIFOThreshold(USART_Module* USARTx,uint32_t threshold)	NO	N32H47x_48x series new functions
Clear USART FIFO	-	void USART_ClrFIFO(USART_Module* USARTx)	NO	N32H47x_48x series new functions
Set USART FIFO mode	-	void USART_FIFOModeSet(USART_Module* USARTx,FunctionalState Cmd)	NO	N32H47x_48x series new functions
Set USART Idle frame width	-	void USART_IdleFrameWidthSet(USART_Module* USARTx,uint32_t Width)	NO	N32H47x_48x series new functions
Configure USART receive timeout	-	void USART_CfgRTOWidth(USART_Module* USARTx,uint32_t Width)	NO	N32H47x_48x series new functions

The following diagram compares the "Interrupt" example project code for the USART module between the N32G45x series (left) and the N32H47x_48x series (right):





The USART configuration process is largely the same between the two series, with the difference that the N32H47x_48x series includes an additional oversampling configuration step. In terms of interrupt judgment, the USART_GetIntStatus function is replaced by USART_GetFlagStatus and USART_GetIntStatus in the N32H47x_48x series.

4.3.20 DBG

The main differences in the DBG module are as follows:

1. The N32H47x_48x series supports SHRTIM debug pause.

The following table compares the driver API functions of the DBG module between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, the application code can call the DBG module driver API functions with substitutions according to the table below. For more details, please refer to the user manual:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
Configure DBG peripheral	void DBG_ConfigPeriph(uint32_t DBG_Periph, FunctionalState Cmd)	void DBG_ConfigPeriph(uint32_t DBG_Periph, FunctionalState Cmd)	NO	The DBG_Periph parts of the function names and input parameters are inconsistent: N32G45x serial parameter: DBG_SLEEP DBG_STOP DBG_STDBY DBG_IWDG_STOP DBG_WWDG_STOP DBG_TIM1_STOP DBG_TIM2_STOP DBG_TIM3_STOP DBG_TIM4_STOP DBG_CAN1_STOP DBG_I2C1SMBUS DBG_I2C2SMBUS DBG_TIM8_STOP DBG_TIM5_STOP DBG_TIM6_STOP DBG_TIM7_STOP DBG_CAN2_STOP N32H47x_48x serial parameter: DBG_SLEEP DBG_STOP DBG_STANDBY DBG_IWDG_STOP DBG_WWDG_STOP DBG_I2C1SMBUS_TIMEOUT DBG_I2C2SMBUS_TIMEOUT DBG_I2C3SMBUS_TIMEOUT DBG_I2C4SMBUS_TIMEOUT DBG_ATIM1_STOP DBG_ATIM2_STOP DBG_ATIM3_STOP DBG_BTIM1_STOP DBG_BTIM2_STOP DBG_GTIM1_STOP DBG_GTIM2_STOP DBG_GTIM3_STOP

				DBG_GTIM4_STOP DBG_GTIM5_STOP DBG_GTIM6_STOP DBG_GTIM7_STOP DBG_GTIM8_STOP DBG_GTIM9_STOP DBG_GTIM10_STOP DBG_SHRTIM1_STOP
Get UCID	void GetUCID(uint8_t *UCIDbuf)	void GetUCID(uint8_t *UCIDbuf)	YES	None
Get UID	void GetUID(uint8_t *UIDbuf)	void GetUID(uint8_t *UIDbuf)	YES	None
Get MCU ID	void GetDBGMCU_ID(uint8_t *DBGMCU_IDbuf)	void GetDBGMCU_ID(uint8_t *DBGMCU_IDbuf)	YES	None
Get chip version	uint32_t DBG_GetRevNum(void)	uint32_t DBG_GetRevNum(void)	YES	None
Get chip model	uint32_t DBG_GetDevNum(void)	uint32_t DBG_GetDevNum(void)	YES	None
Get FLASH size	uint32_t DBG_GetFlashSize(void)	uint32_t DBG_GetFlashSize(void)	YES	None
Gets the SRAM size	uint32_t DBG_GetSramSize(void)	uint32_t DBG_GetSramSize(void)	YES	None

4.3.21 RTC

The main differences in the RTC module are as follows:

1. The N32G45x series does not support a reference clock input, while the N32H47x_48x series does.
2. In the N32G45x series, BKP is a separate module, whereas in the N32H47x_48x series, the backup registers and RTC are integrated into one module.
3. In the N32G45x series, the BKP and RTC modules are separate, with separate functions and drivers. However, in the N32H47x_48x series, the BKP and RTC modules are merged, with combined functions and drivers. Therefore, the function names and parameters related to BKP differ between the N32G45x and N32H47x_48x series.
4. The N32G45x series supports one tamper pin, while the N32H47x_48x series supports three tamper pins.
5. The N32G45x series supports 40 16-bit backup registers, while the N32H47x_48x series supports 20 32-bit backup registers.
6. The tamper interrupt in the N32G45x series is not connected to an EXTI line, whereas the RTC tamper interrupt in the N32H47x_48x series is connected to EXTI19.

The following table compares the RTC module driver API functions between the N32G430 series and the N32G43x series. After replacing the driver .c/.h files, the application code can call the RTC module driver API functions with substitutions according to the table below:

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
RTC reset	ErrorStatus RTC_DeInit(void)	ErrorStatus RTC_Deinitializes(void)	YES	None
Initialize RTC	ErrorStatus RTC_Init(RTC_InitType* RTC_InitStruct)	ErrorStatus RTC_Init(RTC_InitType* RTC_InitStruct)	YES	None
Initialize RTC struct	void RTC_StructInit(RTC_InitType* RTC_InitStruct)	void RTC_StructInit(RTC_InitType* RTC_InitStruct)	YES	None
Enable write protection	void RTC_EnableWriteProtection(FunctionalState Cmd)	void RTC_EnableWriteProtection(FunctionalState Cmd)	YES	None
Enter the initialization mode	ErrorStatus RTC_EnterInitMode(void)	ErrorStatus RTC_EnterInitMode(void)	YES	None
Exit initialization mode	void RTC_ExitInitMode(void)	void RTC_ExitInitMode(void)	YES	None
Wait for synchronization	ErrorStatus RTC_WaitForSynchro(void)	ErrorStatus RTC_WaitForSynchro(void)	YES	None
Enable reference clock	-	ErrorStatus RTC_EnableRefClock(FunctionalState Cmd)	NO	N32G45x series does not have this API
Enable bypass shadow register	void RTC_EnableBypassShadow(FunctionalState Cmd)	void RTC_EnableBypassShadow(FunctionalState Cmd)	YES	None
Set time	ErrorStatus RTC_ConfigTime(uint32_t RTC_Format, RTC_TimeType* RTC_TimeStruct)	ErrorStatus RTC_ConfigTime(uint32_t RTC_Format, RTC_TimeType* RTC_TimeStruct)	YES	None
Initialize time structure	void RTC_TimeStructInit(RTC_TimeType* RTC_TimeStruct)	void RTC_TimeStructInit(RTC_TimeType* RTC_TimeStruct)	YES	None

Acquisition time	void RTC_GetTime(uint32_t RTC_Format, RTC_TimeType* RTC_TimeStruct)	void RTC_GetTime(uint32_t RTC_Format, RTC_TimeType* RTC_TimeStruct)	YES	None
Gets the subsecond value	uint32_t RTC_GetSubSecond(void)	uint32_t RTC_GetSubSecond(void)	YES	None
Set date	ErrorStatus RTC_SetDate(uint32_t RTC_Format, RTC_DateType* RTC_DateStruct)	ErrorStatus RTC_SetDate(uint32_t RTC_Format, RTC_DateType* RTC_DateStruct)	YES	None
Initialize date structure	void RTC_DateStructInit(RTC_DateType* RTC_DateStruct)	void RTC_DateStructInit(RTC_DateType* RTC_DateStruct)	YES	None
Get date	void RTC_GetDate(uint32_t RTC_Format, RTC_DateType* RTC_DateStruct)	void RTC_GetDate(uint32_t RTC_Format, RTC_DateType* RTC_DateStruct)	YES	None
Initialize date structure	void RTC_DateStructInit(RTC_DateType* RTC_DateStruct)	void RTC_Date_Struct_Initializes(RTC_DateType* RTC_DateStruct)	YES	None
Set alarm	void RTC_SetAlarm(uint32_t RTC_Format, uint32_t RTC_Alarm, RTC_AlarmType* RTC_AlarmStruct)	void RTC_SetAlarm(uint32_t RTC_Format, uint32_t RTC_Alarm, RTC_AlarmType* RTC_AlarmStruct)	YES	None
Example Initialize the alarm structure	void RTC_AlarmStructInit(RTC_AlarmType* RTC_AlarmStruct)	void RTC_AlarmStructInit(RTC_AlarmType* RTC_AlarmStruct)	YES	None
Get an alarm clock	void RTC_GetAlarm(uint32_t RTC_Format, uint32_t RTC_Alarm, RTC_AlarmType* RTC_AlarmStruct)	void RTC_GetAlarm(uint32_t RTC_Format, uint32_t RTC_Alarm, RTC_AlarmType* RTC_AlarmStruct)	YES	None
Enable alarm	ErrorStatus RTC_EnableAlarm(uint32_t RTC_Alarm, FunctionalState Cmd)	ErrorStatus RTC_EnableAlarm(uint32_t RTC_Alarm, FunctionalState Cmd)	YES	None
Configure the alarm sub-second	void RTC_ConfigAlarmSubSecond(uint32_t RTC_Alarm, uint32_t RTC_AlarmSubSecondValue, uint32_t RTC_AlarmSubSecondMask)	void RTC_ConfigAlarmSubSecond(uint32_t RTC_Alarm, uint32_t RTC_AlarmSubSecondValue, uint32_t RTC_AlarmSubSecondMask)	YES	None
Get the alarm subsecond	uint32_t RTC_GetAlarmSubSecond(uint32_t RTC_Alarm)	uint32_t RTC_GetAlarmSubSecond(uint32_t RTC_Alarm)	YES	None
Configure the wake clock source	void RTC_ConfigWakeUpClock(uint32_t RTC_WakeUpClock)	void RTC_ConfigWakeUpClock(uint32_t RTC_WakeUpClock)	YES	None
Set the wake counter reload value	void RTC_SetWakeUpCounter(uint32_t RTC_WakeUpCounter)	void RTC_SetWakeUpCounter(uint32_t RTC_WakeUpCounter)	YES	None
Gets the wake up counter reload value	uint32_t RTC_GetWakeUpCounter(void)	uint32_t RTC_GetWakeUpCounter(void)	YES	None
Enable wake up	ErrorStatus RTC_EnableWakeUp(FunctionalState Cmd)	ErrorStatus RTC_EnableWakeUp(FunctionalState Cmd)	YES	None
Configure day-light save	void RTC_ConfigDayLightSaving(uint32_t RTC_DayLightSaving, uint32_t RTC_StoreOperation)	void RTC_ConfigDayLightSaving(uint32_t RTC_DayLightSaving, uint32_t RTC_StoreOperation)	YES	None
Get summer operation records	uint32_t RTC_GetStoreOperation(void)	uint32_t RTC_GetStoreOperation(void)	YES	None
Configure RTC Output	void RTC_ConfigOutput(uint32_t RTC_Output, uint32_t RTC_OutputPolarity)	void RTC_ConfigOutput(uint32_t RTC_Output, uint32_t RTC_OutputPolarity)	YES	None
Configure RTC Output2	-	void RTC_EnableOutput2(FunctionalState Cmd)	否	N32G45x series does not have this API
Enable the Tamper output	-	void RTC_EnableTampOutput(FunctionalState Cmd)	否	N32G45x series does not have this API
Enable calibrating output	void RTC_EnableCalibOutput(FunctionalState Cmd)	void RTC_EnableCalibOutput(FunctionalState Cmd)	YES	None
Configure calibration output	void RTC_ConfigCalibOutput(uint32_t RTC_CalibOutput)	void RTC_ConfigCalibOutput(uint32_t RTC_CalibOutput)	YES	None
Configure smooth calibration	ErrorStatus RTC_ConfigSmoothCalib(uint32_t RTC_SmoothCalibPeriod, uint32_t RTC_SmoothCalibPlusPulses, uint32_t RTC_SmoothCalibMinusPulsesValue)	ErrorStatus RTC_ConfigSmoothCalib(uint32_t RTC_SmoothCalibPeriod, uint32_t RTC_SmoothCalibPlusPulses, uint32_t RTC_SmoothCalibMinusPulsesValue)	YES	None
Enable timestamp	void RTC_EnableTimeStamp(uint32_t RTC_TimeStampEdge, FunctionalState Cmd)	void RTC_EnableTimeStamp(uint32_t RTC_TimeStampEdge, FunctionalState Cmd)	YES	None
Get timestamp	void RTC_GetTimeStamp(uint32_t RTC_Format, RTC_TimeType*)	void RTC_GetTimeStamp(uint32_t RTC_Format, RTC_TimeType*)	YES	None

	RTC_StampTimeStruct, RTC_DateType* RTC_StampDateStruct)	RTC_StampTimeStruct, RTC_DateType* RTC_StampDateStruct)		
Get the timestamp subsecond	uint32_t RTC_GetTimeStampSubSecond(void)	uint32_t RTC_GetTimeStampSubSecond(void)	YES	None
Configure Output type	void RTC_ConfigOutputType(uint32_t RTC_OutputType)	void RTC_ConfigOutputType(uint32_t RTC_OutputType)	YES	None
Configure synchronous shift	ErrorStatus RTC_ConfigSynchroShift(uint32_t RTC_ShiftAddFS, uint32_t RTC_ShiftSub1s)	ErrorStatus RTC_ConfigSynchroShift(uint32_t RTC_ShiftAdd1S, uint32_t RTC_ShiftSubFS)	NO	For N32G45x, it subtracts one second and adds an offset in sub-seconds. For N32H47x_48x, it adds one second and subtracts an offset in sub-seconds.
Configure interrupt	void RTC_ConfigInt(uint32_t RTC_INT, FunctionalState Cmd)	void RTC_ConfigInt(uint32_t RTC_INT, FunctionalState Cmd)	NO	N32H47x_48x has one additional feature compared to the N32G45x: a timestamp interrupt.
Gets the status	FlagStatus RTC_GetFlagStatus(uint32_t RTC_FLAG)	FlagStatus RTC_GetFlagStatus(uint32_t RTC_FLAG)	NO	N32H47x_48x has one additional feature compared to the N32G45x: tamper flag.
Clear flag bit	void RTC_ClrFlag(uint32_t RTC_FLAG)	void RTC_ClrFlag(uint32_t RTC_FLAG)	NO	N32H47x_48x has one additional feature compared to the N32G45x: tamper flag.
Gets the interrupt status	INTStatus RTC_GetITStatus(uint32_t RTC_INT)	INTStatus RTC_GetITStatus(uint32_t RTC_INT)	NO	N32H47x_48x has two additional features compared to the N32G45x: timestamp flag and tamper flag.
Clear the interrupt pending bit	void RTC_ClrIntPendingBit(uint32_t RTC_INT)	void RTC_ClrIntPendingBit(uint32_t RTC_INT)	NO	N32H47x_48x has two additional features compared to the N32G45x: timestamp flag and tamper flag.
Enable TSC wake-up function	RTC_EnableWakeUpTsc	-	NO	N32H47x_48x doesn't have this API
Converts BCD to binary code	static uint8_t RTC_Bcd2ToByte(uint8_t Value)	static uint8_t RTC_Bcd2ToByte(uint8_t Value)	YES	None
Converts binary to BCD code	static uint8_t RTC_ByteToBcd2(uint8_t Value)	static uint8_t RTC_ByteToBcd2(uint8_t Value)	YES	None
The BKP and RTC modules in the N32G45x series are separate, and their drivers are also separate. However, in the N32H47x_48x series, the BKP and RTC modules are merged together, and their drivers are also combined. Therefore, the function names and parameters related to BKP differ between the N32G45x and N32H47x_48x series.				
Configure the tamper triggering mode	-	void RTC_TamperTriggerConfig(uint32_t RTC_Tamper, uint32_t RTC_TamperTrigger)	NO	N32G45x series does not have this API
Intrusion enable	void BKP_TPEnable(FunctionalState Cmd)	void RTC_TamperCmd(uint32_t RTC_Tamper, FunctionalState NewState)	NO	Only one pin of the N32G45x supports tamper detection; The N32H47x_48x has 3 pins support tamper detection
Configure the tamper filter count	-	void RTC_TamperFilterConfig(uint32_t RTC_TamperFilter)	NO	N32G45x series does not have this API
Configure the tamper sampling frequency	-	void RTC_TamperSamplingFreqConfig(uint32_t RTC_TamperSamplingFreq)	NO	N32G45x series does not have this API
Configure tamper pin precharge	-	void RTC_TamperPinsPrechargeDuration(uint32_t RTC_TamperPrechargeDuration)	NO	N32G45x series does not have this API
Enable timestamp triggers the tamper detection event	-	void RTC_TimeStampOnTamperDetectionCmd (FunctionalState NewState)	NO	N32G45x series does not have this API
Enable tamper pull-up	-	void RTC_TamperPullUpCmd(FunctionalState NewState)	NO	N32G45x series does not have this API
Enable tamper erases the backup register	void BKP_ConfigTPLLevel(uint16_t BKP_TamperPinLevel)	void RTC_EnableTampErase(uint32_t RTC_Tamper_Erase, FunctionalState NewState)	NO	N32G45x series does not have this API

Enable tamper activation timestamp	-	void RTC_TamperTAMPTSCmd(FunctionalState NewState)	NO	N32G45x series does not have this API
Enable tamper interrupt	void BKP_TPIntEnable(FunctionalState Cmd)	void RTC_TamperIECmd(uint32_t TAMPxIE, FunctionalState NewState)	NO	Only one pin of the N32G45x supports tamper detection; The N32H47x_48x has 3 pins support tamper detection
Gets the tamper flag bit	FlagStatus BKP_GetTEFlag(void)	FlagStatus RTC_GetFlagStatus(uint32_t RTC_FLAG)	NO	
Clear the tamper flag bit	void BKP_ClrTEFlag(void)	void RTC_ClrFlag(uint32_t RTC_FLAG)	NO	
Gets the tamper interrupt flag bit	INTStatus BKP_GetTINTFlag(void)	INTStatus RTC_GetITStatus(uint32_t RTC_INT)	NO	
Clear the tamper interrupt flag bit	void BKP_ClrTINTFlag(void)	void RTC_ClrIntPendingBit(uint32_t RTC_INT)	NO	
Write backup register	void BKP_WriteBkpData(uint16_t BKP_DAT, uint16_t Data)	void RTC_BKUPRgWrite(uint8_t register_num, uint32_t Data)	NO	The N32G45x has 42 16bit backup registers; The N32H47x_48x has 20 32bit backup registers
Read backup register	uint16_t BKP_ReadBkpData(uint16_t BKP_DAT)	uint32_t RTC_BKUPRgRead(uint8_t register_num)	NO	The N32G45x has 42 16bit backup registers; The N32H47x_48x has 20 32bit backup registers

Below is a comparison of the calendar configuration processes between the N32G45x and N32H47x_48x series, highlighting the differences in the API used to read the BKP (Backup Register).

```

107  /*\fun Main program.
108  /**
109  **/
110  int main(void)
111  {
112      /*!< At this stage the microcontroller clock setting is already configured,
113      this is done through SystemInit() function which is called from startup
114      file (startup_n32g45x_xx.s) before to branch to application main.
115      To reconfigure the default setting of SystemInit() function, refer to
116      system_n32g45x.c file
117      */
118      /* Initialize LEDs on n32g45x-EVAL board */
119      LEDInit(LED1_PORT, LED1_PIN);
120      LEDOff(LED1_PORT, LED1_PIN);
121      /* Initialize USART, TX: PC10 RX: PC11 */
122      log_init();
123      log_info("RTC Init");
124      /* RTC date time alarm default value */
125      RTC_DateAndTimeDefaultVal();
126      /* Enable the PWR clock */
127      RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR | RCC_APB1_PERIPH_BKP, ENABLE);
128      RCC_ClrFlag();
129      /* Allow access to RTC */
130      PWR_BackupAccessEnable(ENABLE);
131      if (USER_WRITE_BKP_DAT1_DATA != BKP_ReadBkpData(BKP_DAT1) )
132      {
133          /* Backup data register value is not correct or not yet programmed (when
134          the first time the program is executed) */
135          log_info("\r\n\r\n RTC not yet configured....");
136          /* RTC clock source select */
137          RTC_CLKSourceConfig(RTC_CLK_SRC_TYPE_LSE, true, true);
138          RTC_PrescalerConfig();
139          log_info("\r\n RTC configured....");
140          /* Adjust time by values entered by the user on the hyperterminal */
141          RTC_DateRegulate();
142          RTC_TimeRegulate();
143          BKP_WriteBkpData(BKP_DAT1, USER_WRITE_BKP_DAT1_DATA);
144      }
145      /* Configure the PA1 pin to generate an EXTI interrupt
146      in which the calendar value is printed (externally feed
147      the I2S signal output on PC13 to PA1 to produce a I2S EXTI interrupt)*/
148      EXTI_PA7_Configuration();
149      EXTI_ClrITPendBit(EXTI_LINE7);
150      /* Calibrate output I2S signal */
151      RTC_ConfigCalibOutput(RTC_CALIB_OUTPUT_I2S);
152      /* Calibrate output config, push pull */
153      RTC_ConfigOutputType(RTC_OUTPUT_PUSH_PULL);
154      /* Calibrate output enable */
155      RTC_EnableCalibOutput(ENABLE);
156      log_info("\r\n RTC Config end....");
157      while (1);
158  }
159  }
160  */
161  */
162  */
163  */
164  */
165  */
166  */
167  */
168  */
169  */
170  */
171  */
172  */
173  */
174  */
175  */
176  */
177  */
178  */
179  */
180  */
181  */
182  */
183  */
184  */
185  */
186  */
187  */
188  */
189  */
190  */
191  */
192  */
193  */
194  */
195  */
196  */
197  */
198  */
199  */
200  */
201  */
202  */
203  */
204  */
205  */
206  */
207  */
208  */
209  */
210  */
211  */
212  */
213  */
214  */
215  */
216  */
217  */
218  */
219  */
220  */
221  */
222  */
223  */
224  */
225  */
226  */
227  */
228  */
229  */
230  */
231  */
232  */
233  */
234  */
235  */
236  */
237  */
238  */
239  */
240  */
241  */
242  */
243  */
244  */
245  */
246  */
247  */
248  */
249  */
250  */
251  */
252  */
253  */
254  */
255  */
256  */
257  */
258  */
259  */
260  */
261  */
262  */
263  */
264  */
265  */
266  */
267  */
268  */
269  */
270  */
271  */
272  */
273  */
274  */
275  */
276  */
277  */
278  */
279  */
280  */
281  */
282  */
283  */
284  */
285  */
286  */
287  */
288  */
289  */
290  */
291  */
292  */
293  */
294  */
295  */
296  */
297  */
298  */
299  */
300  */
301  */
302  */
303  */
304  */
305  */
306  */
307  */
308  */
309  */
310  */
311  */
312  */
313  */
314  */
315  */
316  */
317  */
318  */
319  */
320  */
321  */
322  */
323  */
324  */
325  */
326  */
327  */
328  */
329  */
330  */
331  */
332  */
333  */
334  */
335  */
336  */
337  */
338  */
339  */
340  */
341  */
342  */
343  */
344  */
345  */
346  */
347  */
348  */
349  */
350  */
351  */
352  */
353  */
354  */
355  */
356  */
357  */
358  */
359  */
360  */
361  */
362  */
363  */
364  */
365  */
366  */
367  */
368  */
369  */
370  */
371  */
372  */
373  */
374  */
375  */
376  */
377  */
378  */
379  */
380  */
381  */
382  */
383  */
384  */
385  */
386  */
387  */
388  */
389  */
390  */
391  */
392  */
393  */
394  */
395  */
396  */
397  */
398  */
399  */
400  */
401  */
402  */
403  */
404  */
405  */
406  */
407  */
408  */
409  */
410  */
411  */
412  */
413  */
414  */
415  */
416  */
417  */
418  */
419  */
420  */
421  */
422  */
423  */
424  */
425  */
426  */
427  */
428  */
429  */
430  */
431  */
432  */
433  */
434  */
435  */
436  */
437  */
438  */
439  */
440  */
441  */
442  */
443  */
444  */
445  */
446  */
447  */
448  */
449  */
450  */
451  */
452  */
453  */
454  */
455  */
456  */
457  */
458  */
459  */
460  */
461  */
462  */
463  */
464  */
465  */
466  */
467  */
468  */
469  */
470  */
471  */
472  */
473  */
474  */
475  */
476  */
477  */
478  */
479  */
480  */
481  */
482  */
483  */
484  */
485  */
486  */
487  */
488  */
489  */
490  */
491  */
492  */
493  */
494  */
495  */
496  */
497  */
498  */
499  */
500  */
501  */
502  */
503  */
504  */
505  */
506  */
507  */
508  */
509  */
510  */
511  */
512  */
513  */
514  */
515  */
516  */
517  */
518  */
519  */
520  */
521  */
522  */
523  */
524  */
525  */
526  */
527  */
528  */
529  */
530  */
531  */
532  */
533  */
534  */
535  */
536  */
537  */
538  */
539  */
540  */
541  */
542  */
543  */
544  */
545  */
546  */
547  */
548  */
549  */
550  */
551  */
552  */
553  */
554  */
555  */
556  */
557  */
558  */
559  */
560  */
561  */
562  */
563  */
564  */
565  */
566  */
567  */
568  */
569  */
570  */
571  */
572  */
573  */
574  */
575  */
576  */
577  */
578  */
579  */
580  */
581  */
582  */
583  */
584  */
585  */
586  */
587  */
588  */
589  */
590  */
591  */
592  */
593  */
594  */
595  */
596  */
597  */
598  */
599  */
600  */
601  */
602  */
603  */
604  */
605  */
606  */
607  */
608  */
609  */
610  */
611  */
612  */
613  */
614  */
615  */
616  */
617  */
618  */
619  */
620  */
621  */
622  */
623  */
624  */
625  */
626  */
627  */
628  */
629  */
630  */
631  */
632  */
633  */
634  */
635  */
636  */
637  */
638  */
639  */
640  */
641  */
642  */
643  */
644  */
645  */
646  */
647  */
648  */
649  */
650  */
651  */
652  */
653  */
654  */
655  */
656  */
657  */
658  */
659  */
660  */
661  */
662  */
663  */
664  */
665  */
666  */
667  */
668  */
669  */
670  */
671  */
672  */
673  */
674  */
675  */
676  */
677  */
678  */
679  */
680  */
681  */
682  */
683  */
684  */
685  */
686  */
687  */
688  */
689  */
690  */
691  */
692  */
693  */
694  */
695  */
696  */
697  */
698  */
699  */
700  */
701  */
702  */
703  */
704  */
705  */
706  */
707  */
708  */
709  */
710  */
711  */
712  */
713  */
714  */
715  */
716  */
717  */
718  */
719  */
720  */
721  */
722  */
723  */
724  */
725  */
726  */
727  */
728  */
729  */
730  */
731  */
732  */
733  */
734  */
735  */
736  */
737  */
738  */
739  */
740  */
741  */
742  */
743  */
744  */
745  */
746  */
747  */
748  */
749  */
750  */
751  */
752  */
753  */
754  */
755  */
756  */
757  */
758  */
759  */
760  */
761  */
762  */
763  */
764  */
765  */
766  */
767  */
768  */
769  */
770  */
771  */
772  */
773  */
774  */
775  */
776  */
777  */
778  */
779  */
780  */
781  */
782  */
783  */
784  */
785  */
786  */
787  */
788  */
789  */
790  */
791  */
792  */
793  */
794  */
795  */
796  */
797  */
798  */
799  */
800  */
801  */
802  */
803  */
804  */
805  */
806  */
807  */
808  */
809  */
810  */
811  */
812  */
813  */
814  */
815  */
816  */
817  */
818  */
819  */
820  */
821  */
822  */
823  */
824  */
825  */
826  */
827  */
828  */
829  */
830  */
831  */
832  */
833  */
834  */
835  */
836  */
837  */
838  */
839  */
840  */
841  */
842  */
843  */
844  */
845  */
846  */
847  */
848  */
849  */
850  */
851  */
852  */
853  */
854  */
855  */
856  */
857  */
858  */
859  */
860  */
861  */
862  */
863  */
864  */
865  */
866  */
867  */
868  */
869  */
870  */
871  */
872  */
873  */
874  */
875  */
876  */
877  */
878  */
879  */
880  */
881  */
882  */
883  */
884  */
885  */
886  */
887  */
888  */
889  */
890  */
891  */
892  */
893  */
894  */
895  */
896  */
897  */
898  */
899  */
900  */
901  */
902  */
903  */
904  */
905  */
906  */
907  */
908  */
909  */
910  */
911  */
912  */
913  */
914  */
915  */
916  */
917  */
918  */
919  */
920  */
921  */
922  */
923  */
924  */
925  */
926  */
927  */
928  */
929  */
930  */
931  */
932  */
933  */
934  */
935  */
936  */
937  */
938  */
939  */
940  */
941  */
942  */
943  */
944  */
945  */
946  */
947  */
948  */
949  */
950  */
951  */
952  */
953  */
954  */
955  */
956  */
957  */
958  */
959  */
960  */
961  */
962  */
963  */
964  */
965  */
966  */
967  */
968  */
969  */
970  */
971  */
972  */
973  */
974  */
975  */
976  */
977  */
978  */
979  */
980  */
981  */
982  */
983  */
984  */
985  */
986  */
987  */
988  */
989  */
990  */
991  */
992  */
993  */
994  */
995  */
996  */
997  */
998  */
999  */
1000  */

```

4.3.22 USBFSD

The main differences in the USBFSD module are as follows:

1. **Module Naming Change:** In the N32G45x series, the module is named USB, whereas in the N32H47x_48x series, it is named USBFSD.
2. **Driver Naming Convention:** In the N32G45x series, the USB driver is named usb_xx.c/usb_xx.h. In contrast, in the N32H47x_48x series, the USBFSD driver is named usbfscd_xx.c/usbfscd_xx.h. These drivers can be directly replaced when migrating from one series to the other.
3. **Crystal-Less Mode Support:** The N32G45x series does not support a crystal-less mode. However, the N32H47x_48x series does support this feature. For reference, the HID Keyboard Xtal Less Demo showcases the crystal-less mode implementation in the N32H47x_48x series.
4. **SRAM Allocation:** In the N32G45x series, the USBFS and CAN1 share a common 512-byte SRAM. Conversely, in the N32H47x_48x series, the USBFSD has its dedicated 512-byte SRAM. This allows the N32H47x_48x series to use both the USBFSD and CAN simultaneously without SRAM contention.
5. **System Clock Frequencies with HSE_PLL as USB Clock Source:** When using the HSE_PLL as the USB clock source, the system clock frequencies in the N32G45x series are limited to 144MHz, 96MHz, 72MHz, and 48MHz. In contrast, the N32H47x_48x series supports a broader range of system clock frequencies, including 240MHz, 192MHz, 144MHz, 96MHz, 72MHz, and 48MHz.

4.3.23 USBHS

The N32G45x series does not support the USBHS module, whereas the N32H47x_48x series does support it. The USBHS module supports dual-role functionality (both host and device). For guidance on using the USBHS, please refer to the provided Demo.

4.3.24 FEMC

N32G45x not support FEMC module, N32H47x_48x support FEMC module, how to use the FEMC can refer to Demo.

4.3.25 SHRTIM

The N32G45x does not support the SHRTIM module. The N32H47x_48x supports the SHRTIM module. For details about SHRTIM usage, see Demo and SHRTIM documents.

4.3.26 DVP

The DVP module in the N32G45x series differs significantly in design from that in the N32H47x_48x series. It is recommended that users directly refer to the example routines for the N32H47x_48x series.

4.3.27 TIM

The following table is a comparison of the TIM module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code calls to the PWR module driver API functions can be replaced according to this table. For interconnectivity-related information, please refer to the user manual for usage.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
TIM reset	void TIM_DeInit(TIM_Module* TIMx)	void TIM_DeInit(TIM_Module* TIMx)	NO	TIM numbers are inconsistent. The ATIM1-3 of N32H47x_48x are advanced timers, while the TIM1/TIM8 of N32G45x are advanced timers. The GTIM1-7 of N32H47x_48x are general-purpose timers, and the TIM2-5 of N32G45x are general-purpose timers. The BTIM1-2 of N32H47x_48x are basic timers, whereas the TIM6/TIM7 of N32G45x are basic timers. The GTIM8-10 of N32H47x_48x are newly added general-purpose timers.
Configure TIM basic, configure timer frequency division, pre-loaded value, counting mode, repeat counter, input signal source	void TIM_InitTimeBase(TIM_Module* TIMx, TIM_TimeBaseInitType* TIM_TimeBaseInitStruct)	void TIM_InitTimeBase(TIM_Module* TIMx, TIM_TimeBaseInitType* TIM_TimeBaseInitStruct)	NO	TIM numbers are inconsistent. Inconsistent Input Signals: The interconnectivity of N32H47x_48x has been significantly enhanced, and it is necessary to consult the user manual for interconnectivity usage.
Configure TIM channel 1, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, complementary output polarity, Output idle state, complementary output idle state	void TIM_InitOc1(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc1(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent. Inconsistent Input Signals: N32H47x_48x add parameter: TIM_OCMODE_OPMOD_RETRIG1 TIM_OCMODE_OPMOD_RETRIG2 TIM_OCMODE_COMBI_PWM1 TIM_OCMODE_COMBI_PWM2
Configure TIM channel 2, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, complementary output polarity, Output idle state, complementary output idle state	void TIM_InitOc2(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc2(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent. Inconsistent Input Signals: N32H47x_48x add parameter: TIM_OCMODE_OPMOD_RETRIG1 TIM_OCMODE_OPMOD_RETRIG2 TIM_OCMODE_COMBI_PWM1 TIM_OCMODE_COMBI_PWM2
Configure TIM channel 3, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, Complementary output polarity, Output idle state,	void TIM_InitOc3(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc3(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent. Inconsistent Input Signals: N32H47x_48x add parameter: TIM_OCMODE_OPMOD_RETRIG1 TIM_OCMODE_OPMOD_RETRIG2 TIM_OCMODE_COMBI_PWM1 TIM_OCMODE_COMBI_PWM2

complementary output idle state				
Configure TIM channel 4, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, Complementary output polarity, Output idle state, complementary output idle state	void TIM_InitOc4(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc4(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent. Inconsistent Input Signals: N32H47x_48x add parameter: TIM_OCMODE_OPMOD_RETRIG1 TIM_OCMODE_OPMOD_RETRIG2 TIM_OCMODE_COMBI_PWM1 TIM_OCMODE_COMBI_PWM2
Configure TIM channel 5, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, Complementary output polarity, Output idle state, complementary output idle state	void TIM_InitOc5(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc5(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent
Configure TIM channel 6, configure output mode, Output Enable, Complementary channel output Enable, Comparison value, Output polarity, Complementary output polarity, Output idle state, complementary output idle state	void TIM_InitOc6(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	void TIM_InitOc6(TIM_Module* TIMx, OCInitType* TIM_OCInitStruct)	NO	TIM numbers are inconsistent
Configure TIM enters the capture	void TIM_ICInit(TIM_Module* TIMx, TIM_ICInitType* TIM_ICInitStruct)	void TIM_ICInit(TIM_Module* TIMx, TIM_ICInitType* TIM_ICInitStruct)	NO	TIM numbers are inconsistent
Configure TIM PWM capture	void TIM_ConfigPwmIc(TIM_Module* TIMx, TIM_ICInitType* TIM_ICInitStruct)	void TIM_ConfigPwmIc(TIM_Module* TIMx, TIM_ICInitType* TIM_ICInitStruct)	NO	TIM numbers are inconsistent
Configure TIM brake	void TIM_ConfigBkdt(TIM_Module* TIMx, TIM_BDTRInitType* TIM_BDTRInitStruct)	void TIM_ConfigBkdt(TIM_Module* TIMx, TIM_BDTRInitType* TIM_BDTRInitStruct)	NO	TIM numbers are inconsistent. The brake configuration of N32H47x_48x has been enhanced with the addition of a second brake (Brake 2), and Brake 1 now includes a bidirectional brake configuration (note that some brake sources have separate polarity control, which is not configured in this function). The brake source configuration for N32H47x_48x needs to be set separately. For specific functional differences, please refer to the user manual and DEMO.

Configure brake 1 filter	-	void TIM_BreakFiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	TIM numbers are inconsistent. N32H47x_48x Added brake 1 filter configuration
Configure brake 2 filter	-	void TIM_Break2FiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	TIM numbers are inconsistent. N32H47x_48x Added brake 2 filter configuration
Enable brake 1 Filter	-	void TIM_BreakFiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent. N32H47x_48x Added the brake 1 filter
Enable Brake 2 Filter	-	void TIM_Break2FiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent. N32H47x_48x Added the brake 2 filter
Enable Brake 1 source	-	void TIM_BreakInputSourceEnable(TIM_Module* TIMx, uint32_t Source, uint32_t Polarity, FunctionalState Cmd)	NO	TIM numbers are inconsistent. N32H47x_48x added brake source select polarity and separate enable
Enable Brake 2 source	-	void TIM_Break2InputSourceEnable(TIM_Module* TIMx, uint32_t Source, uint32_t Polarity, FunctionalState Cmd)	NO	TIM numbers are inconsistent. N32H47x_48x Added brake source select polarity and separate enable
Release Two-way brake 1	-	void TIM_BidirectionDisarm(TIM_Module* TIMx)	NO	TIM numbers are inconsistent. N32H47x_48x added two-way brake function
Enable bidirectional braking 1	-	void TIM_BidirectionRearm(TIM_Module* TIMx)	NO	TIM numbers are inconsistent. N32H47x_48x added two-way brake function
Release Two-way brake 2	-	void TIM_Bidirection2Disarm(TIM_Module* TIMx)	NO	TIM numbers are inconsistent. N32H47x_48x added two-way brake function
Enable bidirectional brake 2	-	void TIM_Bidirection2Rearm(TIM_Module* TIMx)	NO	TIM numbers are inconsistent. N32H47x_48x added two-way brake function
Initialize the structure	void TIM_InitTimBaseStruct(TIM_TimeBaseInitType* TIM_TimeBaseInitStruct)	void TIM_InitTimBaseStruct(TIM_TimeBaseInitType* TIM_TimeBaseInitStruct)	NO	N32H47x_48x structure initialization input parameter changed
Initialize the structure	void TIM_InitOcStruct(OCInitType* TIM_OCInitStruct)	void TIM_InitOcStruct(OCInitType* TIM_OCInitStruct)	NO	-
Initialize the structure	void TIM_InitIcStruct(TIM_ICInitType* TIM_ICInitStruct)	void TIM_InitIcStruct(TIM_ICInitType* TIM_ICInitStruct)	NO	-
Initialize the structure	void TIM_InitBkdtStruct(TIM_BDTRInitType* TIM_BDTRInitStruct)	void TIM_InitBkdtStruct(TIM_BDTRInitType* TIM_BDTRInitStruct)	NO	N32H47x_48x structure initialization input parameter changed
Enable TIM	void TIM_Enable(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_Enable(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Enable TIM master output	void TIM_EnableCtrlPwmOutputs(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_EnableCtrlPwmOutputs(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Enable TIM interrupt	void TIM_ConfigInt(TIM_Module* TIMx, uint16_t TIM_IT, FunctionalState Cmd)	void TIM_ConfigInt(TIM_Module* TIMx, uint32_t TIM_IT, FunctionalState Cmd)	NO	TIM numbers are inconsistent. N32H47x_48x add interrupt input parameter: TIM_INT_CC5 TIM_INT_CC6 TIM_INT_CC7 TIM_INT_CC8 TIM_INT_CC9
Generate TIM event	void TIM_GenerateEvent(TIM_Module* TIMx, uint16_t TIM_EventSource)	void TIM_GenerateEvent(TIM_Module* TIMx, uint32_t TIM_EventSource)	NO	TIM numbers are inconsistent. N32H47x_48x add event input parameter: TIM_EVT_SRC_BREAK2
Configure TIM DMA	void TIM_ConfigDma(TIM_Module* TIMx, uint32_t TIM_DMABase, uint32_t TIM_DMABurstLength)	void TIM_ConfigDma(TIM_Module* TIMx, uint32_t TIM_DMABase, uint32_t TIM_DMABurstLength)	NO	TIM numbers are inconsistent. Modify DMA initial address input parameter: TIM_DMABASE_CTRL1 TIM_DMABASE_CTRL2 TIM_DMABASE_STS

				TIM_DMABASE_EVTGEN TIM_DMABASE_SMCTRL TIM_DMABASE_DMAINTEN TIM_DMABASE_CAPCMPMOD1 TIM_DMABASE_CAPCMPMOD2 TIM_DMABASE_CAPCMPMOD3 TIM_DMABASE_CAPCMPEN TIM_DMABASE_CAPCMPDAT1 TIM_DMABASE_CAPCMPDAT2 TIM_DMABASE_CAPCMPDAT3 TIM_DMABASE_CAPCMPDAT4 TIM_DMABASE_CAPCMPDAT5 TIM_DMABASE_CAPCMPDAT6 TIM_DMABASE_PSC TIM_DMABASE_AR TIM_DMABASE_CNT TIM_DMABASE_REPCNT TIM_DMABASE_BKDT TIM_DMABASE_CAPCMPDAT7 TIM_DMABASE_CAPCMPDAT8 TIM_DMABASE_CAPCMPDAT9 TIM_DMABASE_BKFR TIM_DMABASE_C1FILT TIM_DMABASE_C2FILT TIM_DMABASE_C3FILT TIM_DMABASE_C4FILT TIM_DMABASE_FILTO TIM_DMABASE_INSEL TIM_DMABASE_AF1 TIM_DMABASE_AF2 TIM_DMABASE_BKFR2 TIM_DMABASE_SLIDFPSC DMA length input parameter increased: TIM_DMABURST_LENGTH_ITRANS FER to TIM_DMABURST_LENGTH_35TRANS FERS
Configure DMA request source	void TIM_EnableDma(TIM_Module* TIMx, uint32_t TIM_DMASource, FunctionalState Cmd)	void TIM_EnableDma(TIM_Module* TIMx, uint32_t TIM_DMASource, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Configure TIM internal clock mode	void TIM_ConfigInternalClk(TIM_Module* TIMx)	void TIM_ConfigInternalClk(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Configure TIM internal signals as clocks	void TIM_ConfigInternalTrigToExt(TIM_Module* TIMx, uint16_t TIM_InputTriggerSource)	void TIM_ConfigInternalTrigToExt(TIM_Module* TIMx, uint32_t TIM_InputTriggerSource)	NO	TIM numbers are inconsistent. Internal signal source modification, specific signal source refer to the user manual: TIM_TRIG_SEL_IN_TR0 TIM_TRIG_SEL_IN_TR1 TIM_TRIG_SEL_IN_TR2 TIM_TRIG_SEL_IN_TR3 TIM_TRIG_SEL_IN_TR4 TIM_TRIG_SEL_IN_TR5 TIM_TRIG_SEL_IN_TR6 TIM_TRIG_SEL_IN_TR7 TIM_TRIG_SEL_IN_TR8 TIM_TRIG_SEL_IN_TR9 TIM_TRIG_SEL_IN_TR10 TIM_TRIG_SEL_IN_TR11 TIM_TRIG_SEL_IN_TR12 TIM_TRIG_SEL_IN_TR13 TIM_TRIG_SEL_IN_TR14
Configure TIM external signals as clocks	void TIM_ConfigExtTrigAsClk(TIM_Module* TIMx, uint16_t TIM_TlxExternalCLKSource, uint16_t ICpolarity, uint16_t ICFilter)	void TIM_ConfigExtTrigAsClk(TIM_Module* TIMx, uint32_t TIM_TlxExternalCLKSource, uint32_t ICpolarity, uint32_t ICFilter)	NO	TIM numbers are inconsistent
Configure external clock mode 1	void TIM_ConfigExtClkMode1(TIM_Module* TIMx, uint16_t TIM_ExtTRGPrescaler, uint16_t TIM_ExtTRGPolarity, uint16_t ExtTRGFilter)	void TIM_ConfigExtClkMode1(TIM_Module* TIMx, uint32_t TIM_ETRInputSource, uint32_t TIM_ExtTRGPrescaler, uint32_t TIM_ExtTRGPolarity, uint32_t ExtTRGFilter)	NO	TIM numbers are inconsistent. ETR signal source parameters are added. Refer to the user manual for specific signal sources: TIM_CAPETRSEL_0 TIM_CAPETRSEL_1 TIM_CAPETRSEL_2 TIM_CAPETRSEL_3 TIM_CAPETRSEL_4

				TIM_CAPETRSEL_5 TIM_CAPETRSEL_6 TIM_CAPETRSEL_7 TIM_CAPETRSEL_8 TIM_CAPETRSEL_9 TIM_CAPETRSEL_10 TIM_CAPETRSEL_11 TIM_CAPETRSEL_12 TIM_CAPETRSEL_13
Configure external clock mode 2	void TIM_ConfigExtClkMode2(TIM_Module* TIMx,uint16_t TIM_ExtTRGPrescaler,uint16_t TIM_ExtTRGPolarity,uint16_t ExtTRGFilter)	void TIM_ConfigExtClkMode2(TIM_Module* TIMx,uint32_t TIM_ETRInputSource,uint32_t TIM_ExtTRGPrescaler,uint32_t TIM_ExtTRGPolarity,uint32_t ExtTRGFilter)	NO	TIM numbers are inconsistent. ETR signal source parameters are added. Refer to the user manual for specific signal sources: TIM_CAPETRSEL_0 TIM_CAPETRSEL_1 TIM_CAPETRSEL_2 TIM_CAPETRSEL_3 TIM_CAPETRSEL_4 TIM_CAPETRSEL_5 TIM_CAPETRSEL_6 TIM_CAPETRSEL_7 TIM_CAPETRSEL_8 TIM_CAPETRSEL_9 TIM_CAPETRSEL_10 TIM_CAPETRSEL_11 TIM_CAPETRSEL_12 TIM_CAPETRSEL_13
Configure ETR	void TIM_ConfigExtTrig(TIM_Module* TIMx,uint16_t TIM_ExtTRGPrescaler,uint16_t TIM_ExtTRGPolarity,uint16_t ExtTRGFilter)	void TIM_ConfigExtTrig(TIM_Module* TIMx,uint32_t TIM_ExtTRGPrescaler,uint32_t TIM_ExtTRGPolarity,uint32_t ExtTRGFilter)	NO	TIM numbers are inconsistent
Configure TIM frequency division	void TIM_ConfigPrescaler(TIM_Module* TIMx, uint16_t Prescaler, uint16_t TIM_PSCReloadMode)	void TIM_ConfigPrescaler(TIM_Module* TIMx, uint32_t Prescaler, uint32_t TIM_PSCReloadMode)	NO	TIM numbers are inconsistent
Configure TIM count mode	void TIM_ConfigCntMode(TIM_Module* TIMx, uint16_t CntMode)	void TIM_ConfigCntMode(TIM_Module* TIMx, uint32_t CntMode)	NO	TIM numbers are inconsistent
Configure TIM triggers input source	void TIM_SelectInputTrig(TIM_Module* TIMx, uint16_t TIM_InputTriggerSource)	void TIM_SelectInputTrig(TIM_Module* TIMx, uint32_t TIM_InputTriggerSource)	NO	TIM numbers are inconsistent. Input signal source parameters are added. Refer to the user manual for specific signal sources: TIM_TRIG_SEL_IN_TR0 TIM_TRIG_SEL_IN_TR1 TIM_TRIG_SEL_IN_TR2 TIM_TRIG_SEL_IN_TR3 TIM_TRIG_SEL_IN_TR4 TIM_TRIG_SEL_IN_TR5 TIM_TRIG_SEL_IN_TR6 TIM_TRIG_SEL_IN_TR7 TIM_TRIG_SEL_IN_TR8 TIM_TRIG_SEL_IN_TR9 TIM_TRIG_SEL_IN_TR10 TIM_TRIG_SEL_IN_TR11 TIM_TRIG_SEL_IN_TR12 TIM_TRIG_SEL_IN_TR13 TIM_TRIG_SEL_IN_TR14 TIM_TRIG_SEL_TI1F_ED TIM_TRIG_SEL_TI1FP1 TIM_TRIG_SEL_TI2FP2 TIM_TRIG_SEL_ETRF
Configure encoder mode	void TIM_ConfigEncoderInterface(TIM_Module* TIMx,uint16_t TIM_EncoderMode,uint16_t TIM_IC1Polarity,uint16_t TIM_IC2Polarity)	void TIM_ConfigEncoderInterface(TIM_Module* TIMx,uint32_t TIM_EncoderMode,uint32_t TIM_IC1Polarity,uint32_t TIM_IC2Polarity)	NO	TIM numbers are inconsistent. Encoder mode input parameter increased: TIM_ENCODE_QUA_MODE_SINGLE_TI1 TIM_ENCODE_QUA_MODE_SINGLE_TI2 TIM_ENCODE_DUL_CLKPLUS_MODE1 TIM_ENCODE_DUL_CLKPLUS_MODE2 TIM_ENCODE_SINGLE_CLKPLUS_MODE1 TIM_ENCODE_SINGLE_CLKPLUS_MODE2

Configure channel 1 forces output	void TIM_ConfigForcedOc1(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc1(TIM_Module* TIMx, uint32_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure channel 2 forces output	void TIM_ConfigForcedOc2(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc2(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure channel 3 forces output	void TIM_ConfigForcedOc3(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc3(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure channel 4 Forced output	void TIM_ConfigForcedOc4(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc4(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure channel 5 Forced output	void TIM_ConfigForcedOc5(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc5(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure channel 6 forces output	void TIM_ConfigForcedOc6(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	void TIM_ConfigForcedOc6(TIM_Module* TIMx, uint16_t TIM_ForcedAction)	NO	TIM numbers are inconsistent
Configure AR preload	void TIM_ConfigArPreload(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_ConfigArPreload(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Configure COM controls update	void TIM_SelectComEvt(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_SelectComEvt(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Configure DMA request	void TIM_SelectCapCmpDmaSrc(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_SelectCapCmpDmaSrc(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Enable capture/compare preloaded	void TIM_EnableCapCmpPreloadControl(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_EnableCapCmpPreloadControl(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Enabled CCDAT1 preload	void TIM_ConfigOc1Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc1Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT2 preload	void TIM_ConfigOc2Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc2Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT3 preload	void TIM_ConfigOc3Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc3Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT4 preload	void TIM_ConfigOc4Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc4Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT5 preload	void TIM_ConfigOc5Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc5Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT6 preload	void TIM_ConfigOc6Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	void TIM_ConfigOc6Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	TIM numbers are inconsistent
Enabled CCDAT7 preload	-	void TIM_ConfigOc7Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	N32H47x_48x series new functions
Enabled CCDAT8 preload	-	void TIM_ConfigOc8Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	N32H47x_48x series new functions
Enabled CCDAT9 preload	-	void TIM_ConfigOc9Preload(TIM_Module* TIMx, uint32_t TIM_OCPreload)	NO	N32H47x_48x series new functions
Configures the TIMx Output	void TIM_ConfigOc1Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc1Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent

Compare 1 Fast feature.				
Configures the TIMx Output Compare 2 Fast feature.	void TIM_ConfigOc2Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc2Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent
Configures the TIMx Output Compare 3 Fast feature.	void TIM_ConfigOc3Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc3Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent
Configures the TIMx Output Compare 4 Fast feature.	void TIM_ConfigOc4Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc4Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent
Configures the TIMx Output Compare 5 Fast feature.	void TIM_ConfigOc5Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc5Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent
Configures the TIMx Output Compare 6 Fast feature.	void TIM_ConfigOc6Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	void TIM_ConfigOc6Fast(TIM_Module* TIMx, uint32_t TIM_OCFast)	NO	TIM numbers are inconsistent
Clears OCREF1 signal on an external event	void TIM_ClrOc1Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc1Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clears OCREF2 signal on an external event	void TIM_ClrOc2Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc2Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clears OCREF3 signal on an external event	void TIM_ClrOc3Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc3Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clears OCREF4 signal on an external event	void TIM_ClrOc4Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc4Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clears OCREF5 signal on an external event	void TIM_ClrOc5Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc5Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clears OCREF6 signal on an external event	void TIM_ClrOc6Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	void TIM_ClrOc6Ref(TIM_Module* TIMx, uint16_t TIM_OCClear)	NO	TIM numbers are inconsistent
Clear Channel x output comparison trigger source	-	void TIM_ClrOcRefInputSource(TIM_Module* TIMx, uint32_t OCRefClearInputSelect, uint32_t OCRefClearInputSource)	NO	N32H47x_48x series new functions
Configures the TIMx channel 1 polarity.	void TIM_ConfigOc1Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc1Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx channel 2 polarity.	void TIM_ConfigOc2Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc2Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx channel 3 polarity.	void TIM_ConfigOc3Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc3Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx channel 4 polarity.	void TIM_ConfigOc4Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc4Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx channel 5 polarity.	void TIM_ConfigOc5Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc5Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx channel 6 polarity.	void TIM_ConfigOc6Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	void TIM_ConfigOc6Polarity(TIM_Module* TIMx, uint16_t OcPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx Channel 1N polarity.	void TIM_ConfigOc1NPolarity(TIM_Module* TIMx, uint16_t OcNPolarity)	void TIM_ConfigOc1NPolarity(TIM_Module* TIMx, uint32_t OcNPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx Channel 2N polarity.	void TIM_ConfigOc2NPolarity(TIM_Module* TIMx, uint16_t OcNPolarity)	void TIM_ConfigOc2NPolarity(TIM_Module* TIMx, uint32_t OcNPolarity)	NO	TIM numbers are inconsistent

Configures the TIMx Channel 3N polarity.	void TIM_ConfigOc3NPolarity(TIM_Module* TIMx, uint16_t OcNPolarity)	void TIM_ConfigOc3NPolarity(TIM_Module* TIMx, uint32_t OcNPolarity)	NO	TIM numbers are inconsistent
Configures the TIMx Channel 4N polarity.	-	void TIM_ConfigOc4NPolarity(TIM_Module* TIMx, uint32_t OcNPolarity)	NO	N32H47x_48x series new functions
Enable TIM Capture Compare Channel x	void TIM_EnableCapCmpCh(TIM_Module* TIMx, uint16_t Channel, uint32_t TIM_CCx)	void TIM_EnableCapCmpCh(TIM_Module* TIMx, uint32_t Channel, uint32_t TIM_CCx)	NO	TIM numbers are inconsistent. Add channel parameters: TIM_CH_5 TIM_CH_6
Enable TIM Capture Compare Channel xN	void TIM_EnableCapCmpChN(TIM_Module* TIMx, uint16_t Channel, uint32_t TIM_CCxN)	void TIM_EnableCapCmpChN(TIM_Module* TIMx, uint32_t Channel, uint32_t TIM_CCxN)	NO	TIM numbers are inconsistent. Add channel parameters: TIM_CH_4
Output comparison mode configuration	void TIM_SelectOcMode(TIM_Module* TIMx, uint16_t Channel, uint16_t OcMode)	void TIM_SelectOcMode(TIM_Module* TIMx, uint32_t Channel, uint32_t OcMode)	NO	TIM numbers are inconsistent. Add channel parameters: TIM_CH_5 TIM_CH_6 Add OcMODE parameters: TIM_OCMODE_OPMOD_RETRIG1 TIM_OCMODE_OPMOD_RETRIG2 TIM_OCMODE_COMBI_PWM1 TIM_OCMODE_COMBI_PWM2
Enable update event	void TIM_EnableUpdateEvt(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_EnableUpdateEvt(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Configure update the request source	void TIM_ConfigUpdateRequestIntSrc(TIM_Module* TIMx, uint16_t TIM_UpdateSource)	void TIM_ConfigUpdateRequestIntSrc(TIM_Module* TIMx, uint32_t TIM_UpdateSource)	NO	TIM numbers are inconsistent
Enable HALL	void TIM_SelectHallSensor(TIM_Module* TIMx, FunctionalState Cmd)	void TIM_SelectHallSensor(TIM_Module* TIMx, FunctionalState Cmd)	NO	TIM numbers are inconsistent
Configure single pulse mode	void TIM_SelectOnePulseMode(TIM_Module* TIMx, uint16_t TIM_OPMODE)	void TIM_SelectOnePulseMode(TIM_Module* TIMx, uint32_t TIM_OPMODE)	NO	TIM numbers are inconsistent
Configure TRGO output source	void TIM_SelectOutputTrig(TIM_Module* TIMx, uint16_t TIM_TRGOSource)	void TIM_SelectOutputTrig(TIM_Module* TIMx, uint32_t TIM_TRGOSource)	NO	N32H47x_48x series new functions. Add TRGO signal source: TIM_TRGO_SRC_OC4_7_8_9REF
Configure TRGO2 output	-	void TIM_SelectOutputTrig2(TIM_Module* TIMx, uint32_t TIM_TRGO2Source)	NO	N32H47x_48x series new functions
Configure Slave mode	void TIM_SelectSlaveMode(TIM_Module* TIMx, uint16_t TIM_SlaveMode)	void TIM_SelectSlaveMode(TIM_Module* TIMx, uint32_t TIM_SlaveMode)	NO	TIM numbers are inconsistent. Add slave mode parameters: TIM_SLAVE_MODE_GATED_RESET TIM_SLAVE_MODE_TRIG_RESET
Configure synchronization in primary/secondary mode	void TIM_SelectMasterSlaveMode(TIM_Module* TIMx, uint16_t TIM_MasterSlaveMode)	void TIM_SelectMasterSlaveMode(TIM_Module* TIMx, uint32_t TIM_MasterSlaveMode)	NO	TIM numbers are inconsistent
Set counter	void TIM_SetCnt(TIM_Module* TIMx, uint16_t Counter)	void TIM_SetCnt(TIM_Module* TIMx, uint16_t Counter)	NO	TIM numbers are inconsistent
Set AR value	void TIM_SetAutoReload(TIM_Module* TIMx, uint16_t Autoreload)	void TIM_SetAutoReload(TIM_Module* TIMx, uint32_t Autoreload)	NO	TIM numbers are inconsistent
Set the comparison register value (CCDAT1)	void TIM_SetCmp1(TIM_Module* TIMx, uint16_t Compare1)	void TIM_SetCmp1(TIM_Module* TIMx, uint16_t Compare1)	NO	TIM numbers are inconsistent
Set comparison register value (CCDAT2)	void TIM_SetCmp2(TIM_Module* TIMx, uint16_t Compare2)	void TIM_SetCmp2(TIM_Module* TIMx, uint16_t Compare2)	NO	TIM numbers are inconsistent

Set comparison register value (CCDAT3)	void TIM_SetCmp3(TIM_Module* TIMx, uint16_t Compare3)	void TIM_SetCmp3(TIM_Module* TIMx, uint16_t Compare3)	NO	TIM numbers are inconsistent
Set comparison register value (CCDAT4)	void TIM_SetCmp4(TIM_Module* TIMx, uint16_t Compare4)	void TIM_SetCmp4(TIM_Module* TIMx, uint16_t Compare4)	NO	TIM numbers are inconsistent
Set comparison register value (CCDAT5)	void TIM_SetCmp5(TIM_Module* TIMx, uint16_t Compare5)	void TIM_SetCmp5(TIM_Module* TIMx, uint16_t Compare5)	NO	TIM numbers are inconsistent
Set comparison register value (CCDAT6)	void TIM_SetCmp6(TIM_Module* TIMx, uint16_t Compare6)	void TIM_SetCmp6(TIM_Module* TIMx, uint16_t Compare6)	NO	TIM numbers are inconsistent
Set the comparison register value (CCDAT7)	-	void TIM_SetCmp7(TIM_Module* TIMx, uint16_t Compare7)	NO	N32H47x_48x series new functions
Set comparison register value (CCDAT8)	-	void TIM_SetCmp8(TIM_Module* TIMx, uint16_t Compare8)	NO	N32H47x_48x series new functions
Set comparison register value (CCDAT9)	-	void TIM_SetCmp9(TIM_Module* TIMx, uint16_t Compare9)	NO	N32H47x_48x series new functions
Set the comparison register value (CCDDAT1)	-	void TIM_SetCmp1D(TIM_Module* TIMx, uint16_t compare1D)	NO	N32H47x_48x series new functions
Set comparison register value (CCDDAT2)	-	void TIM_SetCmp2D(TIM_Module* TIMx, uint16_t compare2D)	NO	N32H47x_48x series new functions
Set the comparison register value (CCDDAT3)	-	void TIM_SetCmp3D(TIM_Module* TIMx, uint16_t compare3D)	NO	N32H47x_48x series new functions
Set the comparison register value (CCDDAT4)	-	void TIM_SetCmp4D(TIM_Module* TIMx, uint16_t compare4D)	NO	N32H47x_48x series new functions
Set channel 1 input capture frequency division	void TIM_SetInCap1Prescaler(TIM_Module* TIMx, uint16_t TIM_ICPSC)	void TIM_SetInCap1Prescaler(TIM_Module* TIMx, uint16_t ICPrescaler)	NO	TIM numbers are inconsistent
Set channel 2 input capture frequency division	void TIM_SetInCap2Prescaler(TIM_Module* TIMx, uint16_t TIM_ICPSC)	void TIM_SetInCap2Prescaler(TIM_Module* TIMx, uint16_t ICPrescaler)	NO	TIM numbers are inconsistent
Set channel 3 input capture frequency division	void TIM_SetInCap3Prescaler(TIM_Module* TIMx, uint16_t TIM_ICPSC)	void TIM_SetInCap3Prescaler(TIM_Module* TIMx, uint16_t ICPrescaler)	NO	TIM numbers are inconsistent
Set channel 4 input capture frequency division	void TIM_SetInCap4Prescaler(TIM_Module* TIMx, uint16_t TIM_ICPSC)	void TIM_SetInCap4Prescaler(TIM_Module* TIMx, uint16_t ICPrescaler)	NO	TIM numbers are inconsistent
Set the ETR input source signal	-	void TIM_SelectETRInputSource(TIM_Module* TIMx, uint32_t TIM_ETRInputSource)	NO	N32H47x_48x series new functions
Set TIM clock frequency division	void TIM_SetClkDiv(TIM_Module* TIMx, uint16_t TIM_CKD)	void TIM_SetClkDiv(TIM_Module* TIMx, uint32_t TIM_CKD)	NO	TIM numbers are inconsistent
Get the comparison register value (CCDAT1)	uint16_t TIM_GetCap1(TIM_Module* TIMx)	uint16_t TIM_GetCap1(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get comparison register value (CCDAT2)	uint16_t TIM_GetCap2(TIM_Module* TIMx)	uint16_t TIM_GetCap2(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get comparison register value (CCDAT3)	uint16_t TIM_GetCap3(TIM_Module* TIMx)	uint16_t TIM_GetCap3(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get the comparison register value (CCDAT4)	uint16_t TIM_GetCap4(TIM_Module* TIMx)	uint16_t TIM_GetCap4(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get the comparison register value (CCDAT5)	uint16_t TIM_GetCap5(TIM_Module* TIMx)	uint16_t TIM_GetCap5(TIM_Module* TIMx)	NO	TIM numbers are inconsistent

Get comparison register value (CCDAT6)	uint16_t TIM_GetCap6(TIM_Module* TIMx)	uint16_t TIM_GetCap6(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get the comparison register value (CCDAT7)	-	uint16_t TIM_GetCap7(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get comparison register value (CCDAT8)	-	uint16_t TIM_GetCap8(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get the comparison register value (CCDAT9)	-	uint16_t TIM_GetCap9(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get the comparison register value (CCDDAT1)	-	uint16_t TIM_GetCap1D(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get the comparison register value (CCDDAT2)	-	uint16_t TIM_GetCap2D(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get the comparison register value (CCDDAT3)	-	uint16_t TIM_GetCap3D(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get the comparison register value (CCDDAT4)	-	uint16_t TIM_GetCap4D(TIM_Module* TIMx)	NO	N32H47x_48x series new functions
Get counter value	uint32_t TIM_GetCnt(TIM_Module* TIMx)	uint32_t TIM_GetCnt(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get the frequency division value	uint16_t TIM_GetPrescaler(TIM_Module* TIMx)	uint16_t TIM_GetPrescaler(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get AR value	uint16_t TIM_GetAutoReload(TIM_Module* TIMx)	uint16_t TIM_GetAutoReload(TIM_Module* TIMx)	NO	TIM numbers are inconsistent
Get the channel status	FlagStatus TIM_GetCCENStatus(TIM_Module* TIMx, uint32_t TIM_CCEN)	FlagStatus TIM_GetCCENStatus(TIM_Module* TIMx, uint32_t TIM_CCEN)	NO	TIM numbers are inconsistent. Add channel parameters: TIM_CC4NEN
Gets flag bit status	FlagStatus TIM_GetFlagStatus(TIM_Module* TIMx, uint32_t TIM_FLAG)	FlagStatus TIM_GetFlagStatus(TIM_Module* TIMx, uint32_t TIM_FLAG)	NO	TIM numbers are inconsistent. Add flag parameters: TIM_FLAG_CC7 TIM_FLAG_CC8 TIM_FLAG_CC9 TIM_FLAG_BREAK2 TIM_FLAG_SYS_BREAK
Clear flag bit	void TIM_ClearFlag(TIM_Module* TIMx, uint32_t TIM_FLAG)	void TIM_ClearFlag(TIM_Module* TIMx, uint32_t TIM_FLAG)	NO	TIM numbers are inconsistent. Add flag parameters: TIM_FLAG_CC7 TIM_FLAG_CC8 TIM_FLAG_CC9 TIM_FLAG_BREAK2 TIM_FLAG_SYS_BREAK
Gets the interrupt flag bit	INTStatus TIM_GetIntStatus(TIM_Module* TIMx, uint32_t TIM_IT)	INTStatus TIM_GetIntStatus(TIM_Module* TIMx, uint32_t TIM_IT)	NO	TIM numbers are inconsistent. Add flag parameters: TIM_INT_CC7 TIM_INT_CC8 TIM_INT_CC9 TIM_INT_BREAK2 TIM_INT_SYS_BREAK

Clear the interrupt flag bit	void TIM_ClrIntPendingBit(TIM_Module* TIMx, uint32_t TIM_IT)	void TIM_ClrIntPendingBit(TIM_Module* TIMx, uint32_t TIM_IT)	NO	TIM numbers are inconsistent. Add flag parameters: TIM_INT_CC7 TIM_INT_CC8 TIM_INT_CC9 TIM_INT_BREAK2 TIM_INT_SYS_BREAK
Set the polarity of capture channel 1, input selection, filtering	static void ConfigTI1(TIM_Module* TIMx, uint16_t ICpolarity, uint16_t ICSelection, uint16_t ICFilter)	void ConfigTI1(TIM_Module* TIMx, uint32_t ICPolarity, uint32_t ICSelection, uint32_t ICFilter)	NO	TIM numbers are inconsistent
Set the polarity of capture channel 2, input selection, filtering	static void ConfigTI2(TIM_Module* TIMx, uint16_t ICpolarity, uint16_t ICSelection, uint16_t ICFilter)	void ConfigTI2(TIM_Module* TIMx, uint32_t ICPolarity, uint32_t ICSelection, uint32_t ICFilter)	NO	TIM numbers are inconsistent
Set the polarity of capture channel 3, input selection, filtering	static void ConfigTI3(TIM_Module* TIMx, uint16_t ICpolarity, uint16_t ICSelection, uint16_t ICFilter)	void ConfigTI3(TIM_Module* TIMx, uint32_t ICPolarity, uint32_t ICSelection, uint32_t ICFilter)	NO	TIM numbers are inconsistent
Set the polarity of capture channel 4, input selection, filtering	static void ConfigTI4(TIM_Module* TIMx, uint16_t ICpolarity, uint16_t ICSelection, uint16_t ICFilter)	void ConfigTI4(TIM_Module* TIMx, uint32_t ICPolarity, uint32_t ICSelection, uint32_t ICFilter)	NO	TIM numbers are inconsistent
Enable center alignment asymmetric mode	-	void TIM_AsymmetricEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable the 4/7/8/9 channel to trigger ADC	-	void TIM_OCxRefTriggerADC(TIM_Module* TIMx, uint32_t OCxRef, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Configure channel 1 Enters the digital filtering	-	void TIM_IC1FiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	N32H47x_48x series new functions
Configure channel 2 Enters the digital filtering	-	void TIM_IC2FiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	N32H47x_48x series new functions
Configure channel 3 Enters the digital filtering	-	void TIM_IC3FiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	N32H47x_48x series new functions
Configure channel 4 Enters the digital filtering	-	void TIM_IC4FiltConfig(TIM_Module* TIMx, TIM_FiltInitType* TIM_FiltInitStruct)	NO	N32H47x_48x series new functions
Enable input digital filtering for channel 1	-	void TIM_IC1FiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable input digital filtering for channel 2	-	void TIM_IC2FiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable input digital filtering for channel 3	-	void TIM_IC3FiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable input digital filtering for channel 4	-	void TIM_IC4FiltEnable(TIM_Module* TIMx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Get channel x input digital filter output status	-	FlagStatus TIM_GetFiltStatus(TIM_Module* TIMx, uint32_t TIM_FiltFlag)	NO	N32H47x_48x series new functions

4.3.28 ADC

The following table compares the ADC module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code calls to the ADC module driver API functions can be substituted according to this table.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
ADC reset	void ADC_DeInit(ADC_Module* ADCx)	void ADC_DeInit(ADC_Module* ADCx)	YES	None
Initialize ADC	void ADC_Init(ADC_Module* ADCx, ADC_InitType* ADC_InitStruct)	ADC_Init(ADC_Module* ADCx, ADC_InitType* ADC_InitStruct)	YES	N32H47x_48x add Resolution configuration parameters
Initialize ADC structure	void ADC_InitStruct(ADC_InitType* ADC_InitStruct)	void ADC_InitStruct(ADC_InitType* ADC_InitStruct)	YES	N32H47x_48x add Resolution configuration parameters
Enable ADC	void ADC_Enable(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_Enable(ADC_Module* ADCx, FunctionalState Cmd)	YES	None
Enable ADC DMA	void ADC_EnableDMA(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_SetDMATransferMode(ADC_Module* ADCx, uint32_t DMAMode)	NO	ADC_MULTI_REG_DMA_E ACH_ADC is equal to ADC_EnableDMA(ADCx,ENABLE).
ConfigureADC interrupt	void ADC_ConfigInt(ADC_Module* ADCx, uint16_t ADC_IT, FunctionalState Cmd)	void ADC_ConfigInt(ADC_Module* ADCx, uint16_t ADC_IT, FunctionalState Cmd)	YES	N32H47x_48x Add parameters: ADC_INT_ENDCA ADC_INT_JENDCA ADC_INT_ENDCERR ADC_INT_RDY ADC_INT_PDRDY ADC_INT_EOSAMP
Start ADC calibration	void ADC_StartCalibration(ADC_Module* ADCx)	void ADC_CalibrationOperation(ADC_Module* ADCx, ADC_CALI_MOD cali_mode)	NO	N32H47x_48x Add parameters: cali_mode The new version distinguishes between single-ended mode calibration and differential mode calibration ADC_StartCalibration(ADCx) equivalent to ADC_CalibrationOperation(ADCx, ADC_CALIBRATION_SIGNAL_MODE)
Gets ADC calibration status	FlagStatus ADC_GetCalibrationStatus(ADC_Module* ADCx)	FlagStatus ADC_GetCalibrationStatus(ADC_Module* ADCx)	YES	None
Reset ADC calibration	无	void ADC_CalibrationReset(ADC_Module* ADCx, ADC_CALI_MOD cali_mode)	NO	N32H47x_48x series new functions
Enable regular channel software conversion	void ADC_EnableSoftwareStartConv(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_EnableSoftwareStartConv(ADC_Module* ADCx, FunctionalState Cmd)	YES	None
Get the rule channel software to start the transition status	FlagStatus ADC_GetSoftwareStartConvStatus(ADC_Module* ADCx)	FlagStatus ADC_GetSoftwareStartConvStatus(ADC_Module* ADCx);	YES	None
Enable the injection channel software conversion	void ADC_EnableSoftwareStartInjectedConv(ADC_Module* ADCx, FunctionalState Cmd);	void ADC_EnableSoftwareStartInjectedConv(ADC_Module* ADCx, FunctionalState Cmd);	YES	None
Get the injection channel software to start the transition status	FlagStatus ADC_GetSoftwareStartInjectedConvCmdStatus(ADC_Module* ADCx);	FlagStatus ADC_GetSoftwareStartInjectedConvCmdStatus(ADC_Module* ADCx);	YES	None
Configures the discontinuous mode for the selected ADC regular	void ADC_ConfigDiscModeChannelCount(ADC_Module* ADCx, uint8_t Number)	void ADC_ConfigDiscModeChannelCount(ADC_Module* ADCx, uint8_t Number);	YES	None

* group channel.				
Enables the discontinuous mode on regular group channel for the specified ADC	void ADC_EnableDiscMode(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_EnableDiscMode(ADC_Module* ADCx, FunctionalState Cmd);	YES	None
Enables the discontinuous mode for injected group channel	void ADC_EnableInjectedDiscMode(ADC_Module* ADCx, FunctionalState Cmd);	void ADC_EnableInjectedDiscMode(ADC_Module* ADCx, FunctionalState Cmd);	YES	None
ConfigureRule group channel	void ADC_ConfigRegularChannel(ADC_Module* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)	void ADC_ConfigRegularChannel(ADC_Module* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)	YES	None
Enable automatic injection	void ADC_EnableAutoInjectedConv(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_EnableAutoInjectedConv(ADC_Module* ADCx, FunctionalState Cmd)	YES	None
Enable rule channel external trigger function	void ADC_EnableExternalTrigConv(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_SetRegularTriggerEdge(ADC_Module* ADCx, uint32_t ExternalRegularTriggerEdge)	NO	N32H47x_48x added the external trigger effective edge configuration ADC_EnableAutoInjectedConv(ADCx, ENABLE) is the same as ADC_SetRegularTriggerEdge(ADCx, ADC_REG_TRIG_EXT_RISING)
Enable injection channel external trigger	void ADC_EnableExternalTrigInjectedConv(ADC_Module* ADCx, FunctionalState Cmd)	void ADC_SetInjectTriggerEdge(ADC_Module* ADCx, uint32_t ExternalInjectTriggerEdge)	NO	N32H47x_48x added the external trigger effective edge configuration ADC_EnableExternalTrigInjectedConv(ADCx, ENABLE) and ADC_SetInjectTriggerEdge(ADCx, ADC_INJ_TRIG_EXT_RISING) are the same
Get single ADC conversion results	uint16_t ADC_GetDat(ADC_Module* ADCx)	uint16_t ADC_GetDat(ADC_Module* ADCx)	YES	None
Get multi-ADC conversion results	uint32_t ADC_GetDualModeConversionDat(ADC_Module* ADCx)	uint32_t ADC_GetMultiModeConversionDat(ADC_Module* ADCx)	NO	N32H47x_48x only modify the function name
Configure an external trigger source for a rule channel	无	void ADC_ConfigExternalTrigRegularConv(ADC_Module* ADCx, uint32_t ADC_ExternalTrigRegularConv)	NO	N32H47x_48x series new functions
Configure the external trigger source for the injection channel	void ADC_ConfigExternalTrigInjectedConv(ADC_Module* ADCx, uint32_t ADC_ExternalTrigInjectConv)	void ADC_ConfigExternalTrigInjectedConv(ADC_Module* ADCx, uint32_t ADC_ExternalTrigInjectConv)	YES	None
Configure Injection group channel	void ADC_ConfigInjectedChannel(ADC_Module* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)	void ADC_ConfigInjectedChannel(ADC_Module* ADCx, uint8_t ADC_Channel, uint8_t Rank, uint8_t ADC_SampleTime)	YES	None
Configures the injection sequence length	void ADC_ConfigInjectedSequencerLength(ADC_Module* ADCx, uint8_t Length)	void ADC_ConfigInjectedSequencerLength(ADC_Module* ADCx, uint8_t Length)	YES	None
Set offset data	void ADC_SetInjectedOffsetDat(ADC_Module* ADCx, uint8_t ADC_InjectedChannel, uint16_t Offset)	void ADC_SetOffsetConfig(ADC_Module* ADCx, uint8_t ADC_Offset, ADC_OffsetType* ADC_OffsetStruct)	NO	For details about how to modify the function, see the user manual The migration of injection channels and regular channels is supported. The input changes are large. You are advised to understand the functions first

Get offset configuration	无	void ADC_GetOffsetConfig(ADC_Module* ADCx, uint8_t ADC_Offset, ADC_OffsetType* ADC_OffsetStruct)	NO	N32H47x_48x series new functions
Gets injection channel transformation data	uint16_t ADC_GetInjectedConversionDat(ADC_Module* ADCx, uint8_t ADC_InjectedChannel)	uint16_t ADC_GetInjectedConversionDat(ADC_Module* ADCx, uint8_t ADC_InjectedChannelOffset)	YES	N32H47x_48x parameter enter has been modified: ADC_INJ_CH_1 corresponds to ADC_INJECT_DATA_OFFSET_1 ... ADC_INJ_CH_4 corresponds to ADC_INJECT_DATA_OFFSET_4
Configure the working mode of the simulated watchdog	void ADC_ConfigAnalogWatchdogWorkChannelType(ADC_Module* ADCx, uint32_t ADC_AnalogWatchdog)	void ADC_ConfigAnalogWatchdog1WorkChannelType(ADC_Module* ADCx, uint32_t ADC_AnalogWatchdog)	YES	None
Set up a channel for guard dogs to monitor	void ADC_ConfigAnalogWatchdogSingleChannel(ADC_Module* ADCx, uint8_t ADC_Channel)	void ADC_ConfigAnalogWatchdog1SingleChannel(ADC_Module* ADCx, uint8_t ADC_Channel)	YES	None
Set the watchdog threshold	void ADC_ConfigAnalogWatchdogThresholds(ADC_Module* ADCx, uint16_t HighThreshold, uint16_t LowThreshold)	void ADC_ConfigAnalogWatchdogThresholds(ADC_Module* ADCx, ADC_AWDG_Awdg, uint16_t HighThreshold, uint16_t LowThreshold)	否	N32H47x_48x parameter enter has been modified: The original function configures watchdog 1 only, the new function can choose to configure watchdog 1/2/3
Enable the temperature sensor and the built-in reference voltage	void ADC_EnableTempSensorVrefint(FunctionalState Cmd)	void ADC_EnableTempSensorVrefint(FunctionalState Cmd)	YES	None
Get ADC status	FlagStatus ADC_GetFlagStatus(ADC_Module* ADCx, uint8_t ADC_FLAG)	FlagStatus ADC_GetFlagStatus(ADC_Module* ADCx, uint16_t ADC_FLAG)	YES	N32H47x_48x parameter enter has been modified: - ADC_FLAG_EOC_ANY - ADC_FLAG_JEOC_ANY - - ADC_FLAG_ENDC_ERROR - ADC_FLAG_RDY - ADC_FLAG_PDRDY - ADC_FLAG_EOSAMP - ADC_FLAG_TCFLAG
Clear ADC status	void ADC_ClearFlag(ADC_Module* ADCx, uint8_t ADC_FLAG)	void ADC_ClearFlag(ADC_Module* ADCx, uint16_t ADC_FLAG)	YES	None
Get interrupt status	INTStatus ADC_GetIntStatus(ADC_Module* ADCx, uint16_t ADC_IT)	INTStatus ADC_GetIntStatus(ADC_Module* ADCx, uint16_t ADC_IT)	YES	N32H47x_48x parameter enter has been modified: - ADC_INT_EOC_ANY - ADC_INT_JEOC_ANY - ADC_INT_ENDC_ERROR - ADC_INT_RDY - ADC_INT_PDRDY - ADC_INT_EOSAMP - ADC_INT_TCFLAG
Clear interrupt state	void ADC_ClearIntPendingBit(ADC_Module* ADCx, uint16_t ADC_IT)	void ADC_ClearIntPendingBit(ADC_Module* ADCx, uint16_t ADC_IT)	YES	N32H47x_48x parameter enter has been modified: - ADC_INT_EOC_ANY - ADC_INT_JEOC_ANY - ADC_INT_ENDC_ERROR - ADC_INT_RDY - ADC_INT_PDRDY - ADC_INT_EOSAMP - ADC_INT_TCFLAG
Set the differential mode or single-ended mode for a channel	void ADC_SetDifChs(ADC_Module* ADCx, uint32_t DifChs)	void ADC_SetChannelSingleDiff(ADC_Module* ADCx, uint32_t Channel, uint32_t SingleDiff)	NO	N32H47x_48x modify the parameters of the function, refer to the parameter enumeration of the function
Set ADC additional function	void ADC_InitEx(ADC_Module* ADCx, ADC_InitTypeEx* ADC_InitStructEx)	void ADC_SetConvResultBitNum(ADC_Module* ADCx, uint32_t ResultBitNum)	NO	N32H47x_48x delete the function and split the functionality of the function

Gets ADC ready status	FlagStatus ADC_GetFlagStatusNew(ADC_Module* ADCx, uint8_t ADC_FLAG_NEW)	无	NO	N32H47x_48x this function is merged with ADC_GetFlagStatus
Set whether the ADC bypass mode	void ADC_SetBypassCalibration(ADC_Module* ADCx, FunctionalState en)	void ADC_SetBypassCalibration(ADC_Module* ADCx, FunctionalState Cmd)	YES	None
Set resolution	void ADC_SetConvResultBitNum(ADC_Module* ADCx, uint32_t ResultBitNum)	void ADC_SetConvResultBitNum(ADC_Module* ADCx, uint32_t ResultBitNum)	YES	None
Set the clock mode of the ADC	void ADC_AHB_Clock_Mode_Config(ADC_Module* ADCx) void ADC_PLL_Clock_Mode_Config(ADC_Module* ADCx)	void ADC_SelectClockMode(ADC_Module* ADCx, uint32_t ClockMode)	NO	N32H47x_48x optimization function writing
Set ADC clock mode and frequency division	void ADC_ConfigClk(ADC_CTRL3_CKMOD ADC_ClkMode, uint32_t RCC_ADCHCLKPrescaler)	void ADC_ConfigClk(ADC_CTRL3_CKMOD ADC_ClkMode, uint32_t RCC_ADCHCLKPrescaler)	YES	None
Set the delay time of the multi-ADC cross mode	-	void ADC_SetMultiTwoSamplingDelay(ADC_Module* ADCx, uint32_t MultiTwoSamplingDelay)	NO	N32H47x_48x series new functions
Stop the injection channel conversion	-	void ADC_StopInjectedConv(ADC_Module* ADCx)	NO	N32H47x_48x series new functions
Stop regular channel conversion	-	void ADC_StopRegularConv(ADC_Module* ADCx)	NO	N32H47x_48x series new functions
Get the gain compensation enable status	-	FlagStatus ADC_GetGainCompensationCmdStatus(ADC_Module* ADCx)	NO	N32H47x_48x series new functions
Set the number of filters for watchdog 1	-	void ADC_SetAWDG1FilteringConfig(ADC_Module* ADCx, uint32_t FilteringCount)	NO	N32H47x_48x series new functions
Set the watchdog 2/3 monitoring channel group	-	void ADC_SetAnalogWatchdog23MonitChannels(ADC_Module* ADCx, uint8_t AWDG_RegEnOffset, uint32_t AWDG_ChannelGroup)	NO	N32H47x_48x series new functions
Gets 2/3 of the watchdog's monitoring channel groups	-	uint32_t ADC_GetAnalogWatchdog23MonitChannels(ADC_Module* ADCx, uint8_t AWDG_RegEnOffset)	NO	N32H47x_48x series new functions
Set the watchdog 2/3 interrupt channel	-	void ADC_SetAnalogWatchdog23IntConfig(ADC_Module* ADCx, uint8_t AWDG_RegIntEnOffset, uint32_t AWDG_ChannelEn)	NO	N32H47x_48x series new functions
Gets watchdog 2/3 interrupt channel	-	uint32_t ADC_GetAnalogWatchdog23IntConfig(ADC_Module* ADCx, uint8_t AWDG_RegEnOffset)	NO	N32H47x_48x series new functions
Gets the watchdog 2/3 status flag	-	uint32_t ADC_GetAnalogWatchdog23StatusFlag(ADC_Module* ADCx, uint8_t AWDG_RegSTSOFFset)	NO	N32H47x_48x series new functions
Clear 2/3 of watchdog status flags	-	void ADC_ClearAnalogWatchdog23StatusFlag(ADC_Module* ADCx, uint8_t AWDG_RegSTSOFFset, uint32_t AWDG_ChannelFlag)	NO	N32H47x_48x series new functions
Set the ADC calibration factor	-	void ADC_SetCalibrationFactor(ADC_Module* ADCx, uint32_t SingleDiff, uint32_t CalibrationFactor)	NO	N32H47x_48x series new functions
Enable gain compensation and set the gain compensation value	-	void ADC_SetGainCompensation(ADC_Module* ADCx, uint32_t GainCompensationValue)	NO	N32H47x_48x series new functions
Enable the discontinuous	-	void ADC_EnableOverSamplingDiscont(ADC_Module* ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions

oversampling mode				
Set the working range for oversampling	-	void ADC_SetOverSamplingScope(ADC_Module *ADCx, uint32_t OversampleScope)	NO	N32H47x_48x series new functions
Set the oversampling rate and the number of right shifts	-	void ADC_ConfigOverSamplingRatioAndShift(ADC_Module *ADCx, uint32_t Ratio, uint32_t Shift)	NO	N32H47x_48x series new functions
Enable deep sleep mode	-	void ADC_EnableDeepSleepMode(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable battery voltage	-	void ADC_EnableBatteryVoltageMonitor(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable calibration value can be loaded automatically	-	void ADC_EnableCalibrationAutoLoad(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable Whether the positive end of channel 1 is connected to the PGA	-	void ADC_EnableCH1PositiveEndConnetPGA_P(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable Whether the negative end of channel 1 is connected to the PGA	-	void ADC_EnableCH1NegativeEndConnetPGA_N(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable the positive end of channel 2 is connected to the PGA	-	void ADC_EnableCH2PositiveEndConnetPGA_P(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable the ADC FIFO function	-	void ADC_EnableFIFO(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Clearing FIFO data	-	void ADC_ClearFIFO(ADC_Module *ADCx)	NO	N32H47x_48x series new functions
Get the FIFO waterline	-	void ADC_ConfigFIFOWaterLevel(ADC_Module *ADCx, uint32_t FIFO_Level)	NO	N32H47x_48x series new functions
Get the number of valid FIFOs	-	uint8_t ADC_GetFIFOInvalidDataCount(ADC_Module *ADCx)	NO	N32H47x_48x series new functions
Get the FIFO status flag	-	FlagStatus ADC_GetFIFOFlagStatus(ADC_Module *ADCx, uint16_t ADC_FIFOFLAG)	NO	N32H47x_48x series new functions
Clear the FIFO status flag	-	void ADC_ClearFIFOFlag(ADC_Module *ADCx, uint16_t ADC_FIFO_FLAG)	NO	N32H47x_48x series new functions
Configure ADC FIFO interrupts	-	void ADC_ConfigFIFOInt(ADC_Module *ADCx, uint16_t ADC_FIFO_IT, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Clear the FIFO interrupt state	-	void ADC_ClearFIFOIntPendingBit(ADC_Module *ADCx, uint16_t ADC_FIFO_IT)	NO	N32H47x_48x series new functions
Select the specific EXTI line as the trigger source for the ADC rule channel	-	void ADC_ConfigRegularExtLineTrigSource(ADC_Module *ADCx, uint32_t ADC_trigger)	NO	N32H47x_48x series new functions
Select the specific EXTI line as the trigger source for the ADC injection channel	-	void ADC_ConfigInjectedExtLineTrigSource(ADC_Module *ADCx, uint32_t ADC_trigger)	NO	N32H47x_48x series new functions
Enable automatic loading of ADC	-	void ADC_EnableCalibrationAutoLoad(ADC_Module *ADCx, FunctionalState Cmd)	NO	N32H47x_48x series new functions

calibration values				
Set the ADC power supply	-	void ADC_ConfigADCPowerSupport(ADC_Mode* ADCx, uint32_t ADC_PowerSupportSel)	NO	N32H47x_48x series new functions

The following diagram highlights the numerous new features in the ADC module of the N32H47x_48x series, including oversampling, offset compensation, gain compensation, full-channel monitoring of watchdog timers 2/3, EXTI line triggering, and more.

1. The EXTI line triggering demo has significant differences, and the selection of EXTI lines is shown in the diagram below:

```

73  /**\return none
74  **/
75  void ADC_Init(void)
76  {
77      /* ADC1 configuration */
78      ADC_InitStructure.WorkMode = ADC_WORKMODE_INDEPENDENT;
79      ADC_InitStructure.MultiChEn = ENABLE;
80      ADC_InitStructure.ContinueConvEn = DISABLE;
81      ADC_InitStructure.ExtTrigSelect = ADC_EXT_TRIG_REG_CONV_EXT_INT0_15;
82      ADC_InitStructure.DataAlign = ADC_DAT_ALIGN_R;
83      ADC_InitStructure.ChsNumber = 2;
84      ADC_InitStructure.Resolution = ADC_DATA_RES_12BIT;
85      ADC_Init(ADC1, &ADC_InitStructure);
86      /* ADC1 regular channel4-12 configuration */
87      ADC_ConfigRegularChannel(ADC1, ADC1_Channel_04_PA1, 1, ADC_SAMP_TIME_CYCLES_239_5);
88      ADC_ConfigRegularChannel(ADC1, ADC1_Channel_12_PB1, 2, ADC_SAMP_TIME_CYCLES_239_5);
89
90      /* Regular discontinuous mode channel number configuration */
91      ADC_ConfigDiscModeChannelCount(ADC1, 1);
92      /* Enable regular discontinuous mode */
93      ADC_EnableDiscMode(ADC1, ENABLE);
94
95      /* Set injected sequencer length */
96      ADC_ConfigInjectedSequencerLength(ADC1, 2);
97      /* ADC1 injected channel configuration */
98      ADC_ConfigInjectedChannel(ADC1, ADC1_Channel_14_PB11, 1, ADC_SAMP_TIME_CYCLES_239_5);
99      ADC_ConfigInjectedChannel(ADC1, ADC1_Channel_11_PB12, 2, ADC_SAMP_TIME_CYCLES_239_5);
100     /* ADC1 injected external trigger configuration */
101     ADC_ConfigExternalTrigInjectedConv(ADC1, ADC_EXT_TRIG_INJ_CONV_EXT_INT0_15);
102
103     /* Enable JENDC interrupt */
104     ADC_ConfigInt(ADC1, ADC_INT_JENDC, ENABLE);
105
106     /* Enable ADC1 DMA */
107     ADC_SetDMATransferMode(ADC1, ADC_MULTI_REG_DMA_EACH_ADC);
108
109     ADC_ConfigRegularExtLineTrigSource(ADC1, Regular_ExtLine_TrigSource);
110     ADC_ConfigInjectedExtLineTrigSource(ADC1, Injected_ExtLine_TrigSource);
111
112     /* Enable ADC1 */
113     ADC_Enable(ADC1, ENABLE);
114     /* Check ADC Ready */
115     while (ADC_GetFlagStatus(ADC1, ADC_FLAG_RDY) == RESET)
116     {
117         /* Start ADC1 calibration */
118         ADC_CalibrationOperation(ADC1, ADC_CALIBRATION_SIGNAL_MODE);
119         /* Check the end of ADC1 calibration */
120         while (ADC_GetCalibrationStatus(ADC1))
121         {
122             ;
123         }
124     }
125 }
126
127 /**
128  * @brief Configures the different EXTI lines.
129  */
130 void EXTI_Configuration(void)
131 {
132     EXTI_InitType EXTI_InitStructure;
133
134     GPIO_ConfigEXTILine(GPIOE_PORT_SOURCE, GPIO_PIN_SOURCE11);
135     /* EXTI line11 configuration */
136     EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Event;
137     EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
138     EXTI_InitStructure.EXTI_Line = EXTI_LINE11;
139     EXTI_InitStructure.EXTI_LineCmd = ENABLE;
140     EXTI_InitPeripheral(&EXTI_InitStructure);
141
142     /* Select the EXTI Line15 the GPIO pin source */
143     GPIO_ConfigEXTILine(GPIOE_PORT_SOURCE, GPIO_PIN_SOURCE15);
144     /* EXTI line15 configuration */
145     EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Event;
146     EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
147     EXTI_InitStructure.EXTI_Line = EXTI_LINE15;
148     EXTI_InitStructure.EXTI_LineCmd = ENABLE;
149     EXTI_InitPeripheral(&EXTI_InitStructure);
150 }
151
152 /**

```

2. For other new features, please refer to the specific Demos provided.

4.3.29 DAC

The following table is a comparison of the DAC module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code calls to the DAC module driver API functions can be substituted according to this table.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
DAC reset	void DAC_DeInit(void)	void DAC_DeInit(DACX DACx)	NO	N32H47x_48x add new DACx parameters
Initialize DAC	void DAC_Init(uint32_t DAC_Channel, DAC_InitType* DAC_InitStruct)	void DAC_Init(DACX DACx, DAC_InitType* DAC_InitStruct)	NO	N32H47x_48x add new DACx parameters DAC_InitStruct add members DAC_ConnectExternalPin DAC_ConnectOnChipPeripheral DAC_SignedFormat DAC_DMADoubleDataMode DAC_TriggerEnable and other differences, specific reference driver code
Initialize DAC structure	void DAC_ClearStruct(DAC_InitType* DAC_InitStruct)	void DAC_StructInit(DAC_InitType* DAC_InitStruct)	NO	N32H47x_48x DAC_InitStruct add members DAC_ConnectExternalPin DAC_ConnectOnChipPeripheral DAC_SignedFormat DAC_DMADoubleDataMode DAC_TriggerEnable and other differences, specific reference driver code
EnableDAC	void DAC_Enable(uint32_t DAC_Channel, FunctionalState Cmd)	void DAC_Enable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x add new DACx parameters
Enable DAC DMA	void DAC_DmaEnable(uint32_t DAC_Channel, FunctionalState Cmd)	void DAC_DmaEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x add new DACx parameters
Enable DAC software triggers	void DAC_SoftTrgEnable(uint32_t DAC_Channel, FunctionalState Cmd)	void DAC_SoftTrgEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x add new DACx parameters
Enable software triggers both Dacs simultaneously	void DAC_DualSoftwareTrgEnable(FunctionalState Cmd)	void DAC_DualSoftwareTrgEnable(DAC_Module *Dual_DACx, FunctionalState Cmd)	NO	N32H47x_48x add new DACx parameters
Enable DAC sawtooth step trigger	无	void DAC_SoftTrgSawStepEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable simultaneous stepping trigger of the DAC sawtooth wave	无	void DAC_DualSoftwareTrgSawStepEnable(DAC_Module *Dual_DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable DAC waveform generation	void DAC_WaveGenerationEnable(uint32_t DAC_Channel, uint32_t DAC_Wave, FunctionalState Cmd)	void DAC_WaveGenerationConfig(DACX DACx, uint32_t DAC_Wave)	NO	N32H47x_48x add new DACx parameters and function optimization
DAC holds register writes	void DAC_SetCh1Data(uint32_t DAC_Align, uint16_t Data) void DAC_SetCh2Data(uint32_t DAC_Align, uint16_t Data)	void DAC_SetData(DACX DACx, uint32_t DAC_Align, uint16_t Data)	NO	Function optimization
DAC dual channel hold register write simultaneously	void DAC_SetDualChData(uint32_t DAC_Align, uint16_t Data2, uint16_t Data1)	void DAC_SetDualChData(DAC_Module *Dual_DACx, uint32_t DAC_Align, uint16_t Data2, uint16_t Data1)	NO	N32H47x_48x the new parameter Dual_DACx due to the number of DACx
Gets the value of the DAC output data register	uint16_t DAC_GetOutputDataVal(uint32_t DAC_Channel)	uint16_t DAC_GetOutputDataVal(DACX DACx)	NO	N32H47x_48x add new DACx parameters and function optimization
Set the sawtooth reset value	-	void DAC_SetSawtoothResetValue(DACX DACx, uint16_t ResetValue)	NO	N32H47x_48x series new functions
Set the sawtooth step value	-	void DAC_SetSawtoothStepValue(DACX DACx, uint16_t StepData)	NO	N32H47x_48x series new functions

Enable DAC calibration	-	void DAC_CaliEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable the DAC output to be connected to the chip	-	void DAC_ConnetToOnChipEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable the DAC output to connect to external I/OS	-	void DAC_ConnetToExternalPinEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable DAC DMA dual data mode	-	void DAC_DMADoubleDataModeEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable output in signed format	-	void DAC_SignFormatModeEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable to output high drive capability	-	void DAC_HighDriveAbilityEnable(DACX DACx, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Get DAC status	-	FlagStatus DAC_GetFlagSts(DACX DACx, uint32_t DAC_FLAG)	NO	N32H47x_48x series new functions
Clear the DAC status bit	-	void DAC_ClearFlag(DACX DACx, uint32_t DAC_FLAG)	NO	N32H47x_48x series new functions
ConfigureDAC interrupt	-	void DAC_ConfigInt(DACX DACx, uint32_t DAC_IT, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Gets the DAC interrupt status flag bit	-	FlagStatus DAC_GetIntSts(DACX DACx, uint32_t DAC_IT)	NO	N32H47x_48x series new functions
Clear the DAC interrupt status flag bit	-	void DAC_ClearITPendingBit(DACX DACx, uint32_t DAC_IT)	NO	N32H47x_48x series new functions
ConfigureDAC frequency division coefficient	-	void DAC_ConfigClkPrescaler(DAC_Module* DACx, uint8_t Prescaler)	NO	N32H47x_48x series new functions
Set the DAC high frequency mode	-	void DAC_SetHighFrequencyMode(DAC_Module* DACx, uint32_t mode)	NO	N32H47x_48x series new functions
DAC sets the user calibration value	-	void DAC_SetUserTrimming(DACX DACx, uint8_t TrimmingValue)	NO	N32H47x_48x series new functions
Get the base address through DACx	-	static DAC_Module* DAC_BaseAddrGet(DACX DACx)	NO	N32H47x_48x series new functions Note: This function is only available in .C

The following diagram highlights the numerous new features of the DAC module in the N32H47x_48x series, including internal/external output configuration for DAC, DAC sawtooth wave generation, DMA dual-data mode, DAC interrupt, DAC calibration, and more.

1. On the left side of the diagram is the configuration method for outputting two triangular waves to I/O in the G45x series, while the right side shows the configuration method for the N32H47x_48x series. Beyond the differences outlined below, the N32H47x_48x series also requires additional configurations for the DAC clock prescaler (DAC_ConfigClkPrescaler()) and DAC frequency mode (DAC_SetHighFrequencyMode()). For specific details, please refer to the "TwoDACsTriangleWave" demo example.

```

87 /**
88  * @brief DAC channel1 Configuration.
89  */
90 void DAC_ChannelsConfig(void)
91 {
92     DAC_InitType DAC_InitStructure;
93
94     /* DAC Periph clock enable */
95     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_DAC, ENABLE);
96     /* DAC channel1 Configuration */
97     DAC_InitStructure.Trigger = DAC_TRG_T5_TRGO;
98     DAC_InitStructure.WaveGen = DAC_WAVEGEN_TRIANGLE;
99     DAC_InitStructure.LfsrUnMaskTriAmp = DAC_TRIAMP_2047;
100     DAC_InitStructure.BufferOutput = DAC_BUFFOUTPUT_DISABLE;
101     DAC_Init(DAC_CHANNEL_1, &DAC_InitStructure);
102
103     /* DAC channel2 Configuration */
104     DAC_InitStructure.LfsrUnMaskTriAmp = DAC_TRIAMP_1023;
105     DAC_Init(DAC_CHANNEL_2, &DAC_InitStructure);
106
107     /* Enable DAC Channel1: Once the DAC channel1 is enabled, PA.04 is
108     automatically connected to the DAC converter. */
109     DAC_Enable(DAC_CHANNEL_1, ENABLE);
110
111     /* Enable DAC Channel2: Once the DAC channel2 is enabled, PA.05 is
112     automatically connected to the DAC converter. */
113     DAC_Enable(DAC_CHANNEL_2, ENABLE);
114
115     /* Set DAC dual channel DHR12CH register */
116     DAC_SetDualChData(DAC_ALIGN_R_12BIT, 0x100, 0x100);
117 }
118
119 /**
120  * @name DAC_Configuration
121  * @fun Configures the DAC1 module.
122  * @return none
123  */
124 void DAC_Configuration(void)
125 {
126     DAC_InitType DAC_InitStructure;
127
128     DAC_StructInit(&DAC_InitStructure);
129     /* DAC1 Configuration */
130     DAC_InitStructure.DAC_Trigger = DAC_Trigger_ATIM1_TRGO;
131     DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_Triangle;
132     DAC_InitStructure.DAC_LFSRUnmask_TriangleAmplitude = DAC_TriangleAmplitude_2047;
133     DAC_InitStructure.DAC_OutputBuffer = DISABLE;
134     DAC_InitStructure.DAC_ConnectOnChipPeripheral = DISABLE;
135     DAC_InitStructure.DAC_ConnectExternalPin = ENABLE;
136     DAC_InitStructure.DAC_TriggerEnable = ENABLE;
137     DAC_Init(DAC1, &DAC_InitStructure);
138
139     /* DAC2 Configuration */
140     DAC_InitStructure.DAC_LFSRUnmask_TriangleAmplitude = DAC_TriangleAmplitude_1023;
141     DAC_Init(DAC2, &DAC_InitStructure);
142
143     /* Set DAC1 DAC2 DHR12R register */
144     DAC_SetDualChData(DAC12, DAC_ALIGN_R_12BIT, 0x100, 0x100);
145     /* Enable DAC1 */
146     DAC_Enable(DAC1, ENABLE);
147     /* Enable DAC2 */
148     DAC_Enable(DAC2, ENABLE);
149 }

```

- User trimming can be performed when the operating conditions differ from the nominal factory trim conditions, especially when changes occur in VDDA voltage, temperature, or VREF+ value. This trimming can be done via software at any point in the application. For specific trimming procedures, please refer to the "UserBufferCalibration" demo example.

4.3.30 COMP

The following table compares the COMP module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code calls to the COMP module driver API functions can be substituted according to this table.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
COMP reset	void COMP_DeInit(void)	void COMP_DeInit(void)	YES	None
Initialize COMP structure	void COMP_StructInit(COMP_InitType* COMP_InitStruct)	void COMP_StructInit(COMP_InitType* COMP_InitStruct)	YES	N32H47x_48x parameter enumeration is different. Comparison is missing InpDOOutSelacConnect OuOutSeltSel (related to design changes)
Initialize COMP	void COMP_Init(COMPX COMPx, COMP_InitType* COMP_InitStruct)	void COMP_Init(COMPX COMPx, COMP_InitType* COMP_InitStruct)	YES	N32H47x_48x parameter enumeration is different, specific reference code

Enable COMP	void COMP_Enable(COMPX COMPx, FunctionalState en)	void COMP_Enable(COMPX COMPx, FunctionalState Cmd)	YES	None
Select COMP positive input	void COMP_SetInpSel(COMPX COMPx, COMP_CTRL_INPSEL VpSel)	void COMP_SetInpSel(COMPX COMPx, COMP_CTRL_INPSEL VpSel)	YES	None
Select COMP negative input	void COMP_SetInmSel(COMPX COMPx, COMP_CTRL_INMSEL VmSel)	void COMP_SetInmSel(COMPX COMPx, COMP_CTRL_INMSEL VmSel)	YES	None
Select Output to TIM	void COMP_SetOutTrig(COMPX COMPx, COMP_CTRL_OUTTRIG OutTrig)	void COMP_OutToTimEnable(uint32_t TimEn, FunctionalState Cmd)	NO	N32H47x_48x function changes, due to design changes, the original 1-to-1 output to TIM, changed to 1-to-many output mode
Set COMP lock	void COMP_SetLock(uint32_t Lock)	void COMP_SetLock(uint32_t Lock)	YES	None
Enable interrupt function	void COMP_SetIntEn(uint32_t IntEn)	void COMP_SetIntEn(uint32_t IntEn, FunctionalState Cmd)	NO	Enable and disable compatible interrupts
Gets the interrupt status flag	uint32_t COMP_GetIntSts(void)	无	NO	Delete redundant function
Gets the interrupt status of COMP	FlagStatus COMP_GetIntStsOneComp(COMPX COMPx)	FlagStatus COMP_GetIntStsOneComp(COMPX COMPx)	YES	None
Clear an interrupt for COMP	-	void COMP_ClearIntStsOneComp(COMPX COMPx)	NO	N32H47x_48x series new functions
Set the frequency division coefficient with moderate filtering	void COMP_SetFilterPrescaler(COMPX COMPx, uint16_t FilPreVal)	void COMP_SetFilterPrescaler(COMPX COMPx, uint16_t FilPreVal)	YES	None
Set filter control	void COMP_SetFilterControl(COMPX COMPx, uint8_t FilEn, uint8_t TheresNum, uint8_t SampPW)	void COMP_SetFilterControl(COMPX COMPx, uint8_t FilEn, uint8_t TheresNum, uint8_t SampPW)	YES	None
Set the hysteresis level	void COMP_SetHyst(COMPX COMPx, COMP_CTRL_HYST HYST)	void COMP_SetHyst(COMPX COMPx, COMP_CTRL_HYST HYST)	YES	N32H47x_48x parameter enumeration is different.
Set the blanking source	void COMP_SetBlanking(COMPX COMPx, COMP_CTRL_BLKING BLK)	void COMP_SetBlanking(COMPX COMPx, COMP_CTRL_BLKING BLK)	YES	N32H47x_48x parameter enumeration is different.
Get the result of the comparator before filtering	-	FlagStatus COMP_GetOutStatus(COMPX COMPx)	YES	When filtering is enabled, the function returns the result before filtering
Set 6bitDAC control output	void COMP_SetRefSel(uint8_t Vv2Trim, bool Vv2En, uint8_t Vv1Trim, bool Vv1En)	void COMP_SetRefSel(uint8_t Vv3Trim, bool Vv3En, uint8_t Vv2Trim, bool Vv2En, uint8_t Vv1Trim, bool Vv1En)	YES	None
Get the filter result of the comparator	FlagStatus COMP_GetOutStatus(COMPX COMPx)	FlagStatus COMP_GetFilterOutStatus(COMPX COMPx);	NO	N32H47x_48x distinguishes between post-filtering and pre-filtering results
Enable window mode	-	void COMP_WindowModeEnable(uint32_t WinModeEn, FunctionalState Cmd)	NO	N32H47x_48x series new functions
Enable VFLAG control	-	void COMP_SetVflagEnable(COMPX COMPx, FunctionalState Cmd);	NO	N32H47x_48x series new functions

The following diagram highlights the significant changes in the COMP module for the N32H47x_48x series. The main additions include the status output before and after COMP filtering, hysteresis levels, and the ability for COMP output to be simultaneously routed to multiple TIMs in a one-to-many configuration.

1. On the left side of the diagram is the configuration method for outputting to TIM in the G45x series, while the right side shows the configuration method for the N32H47x_48x series.

```

/**
 * @brief Configures the comp module.
 */
void COMP_Configuratorin(void)
{
    COMP_InitType COMP_Initial;

    /*Set dac2, dac1, because dac1/PA4 is share pin line, so only PB0 puls 0/1, can find out puls*/
    COMP_SetRefSel(16, true, 32, true);
    /*Initial comp*/
    COMP_StructInit(&COMP_Initial);
    COMP_Initial.InpSel = COMP1_CTRL_INPSEL_PB10;
    COMP_Initial.InmSel = COMP1_CTRL_INMSEL_DAC1_PA4;
    COMP_Initial.SampWindow = 18; // (0~32)
    COMP_Initial.Thresh = 30; // (0~32)
    COMP_Init(COMP1, &COMP_Initial);
    /*trig initial as tim1&tim8 break*/
    COMP_SetOutTrig(COMP1, COMP1_CTRL_OUTSEL_TIM1_BKIN_TIM8_BKIN);
    /*enable comp1*/
    COMP_Enable(COMP1, ENABLE);
}

/**
 * @name COMP_Configuratorin
 * @fun Configures the comp module.
 * @return none
 */
void COMP_Configuratorin(void)
{
    COMP_InitType COMP_Initial;

    /*Initial comp*/
    COMP_StructInit(&COMP_Initial);
    COMP_Initial.InpSel = COMP1_CTRL_INPSEL_PA1;
    COMP_Initial.InmSel = COMP1_CTRL_INMSEL_PA4;
    COMP_Initial.SampWindow = 18; // (0~32)
    COMP_Initial.Threshold = 30; // (0~32)
    COMP_Init(COMP1, &COMP_Initial);
    COMP_Enable(COMP1, ENABLE);
    /*Enable comp1 output to timer*/
    COMP_OutToTimEnable(COMP1_TIM_EN, ENABLE);
}

```

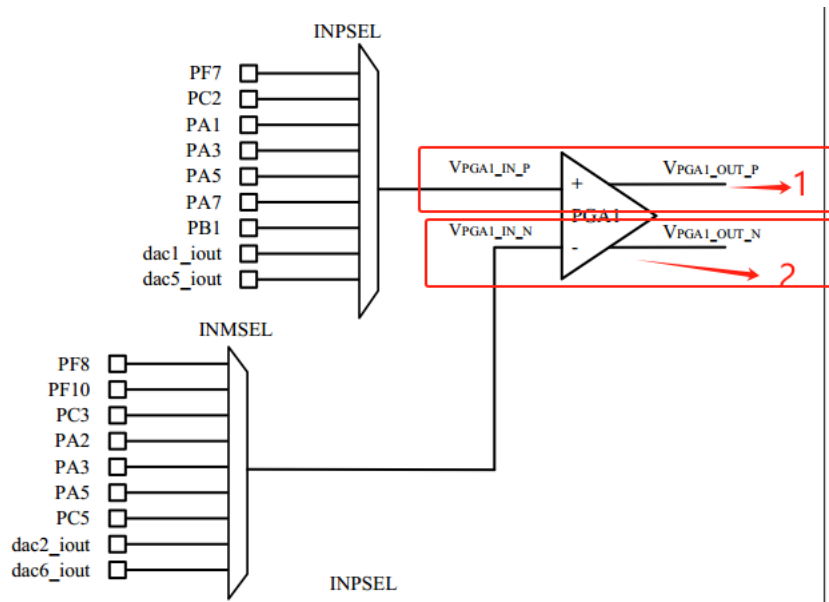
4.3.31 PGA

The PGA module in the N32H47x_48x series has undergone significant functional changes compared to the OPA module in the N32G45x series, making direct comparison or substitution impractical. It is recommended to refer to the PGA demo for testing purposes.

The output of the PGA in the N32H47x_48x series is directly connected to the internal channels of the ADC and cannot be configured to connect to external I/O. The PGA can operate in both single-ended and differential modes:

4.3.31.1 Single-ended:

1. In the case of PGA1, it can be split into two single-ended PGAs for use. INPSEL allows you to select a channel as the input, and the output is directly connected to ADC1_VINP1.



Using PA1 as the input and a gain of 2 as an example, the corresponding configuration code is shown in the diagram below:

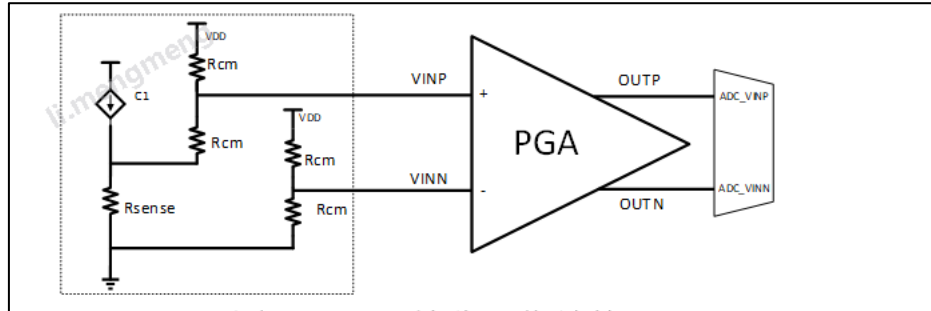
```

102  /**
103  **/
104  void PGA_Configuration(void)
105  {
106      PGA_InitType PGA_Init;
107
108      /*Initial comp*/
109      PGA_StructInit(&PGA_Init);
110      PGA_Init.DiffModeEn = DISABLE;
111      PGA_Init.Gain1 = PGA_CTRL_CH1GAIN_2_DIFF_4;
112      PGA_Init.Vpsel = PGA1_CTRL_VPSEL_PA1;
113      PGA_Init.TimeAutoMuxEn = DISABLE;
114      PGA_Init.En1 = DISABLE;
115      PGA_Init.En2 = DISABLE;
116      PGA_Init(PGA1, &PGA_Init);
117      /*enable PGA1 CH1 and CH2 */
118      PGA_Enable(PGA1, PGA_Channel1, ENABLE);
119  }
120
121  /**
122  **/
123  /**\name ADC_Init
124  **\fun ADC_Init program.
125  **\return none
126  **/
127  void ADC_Init(void)
128  {
129      /* ADCx configuration */
130      ADC_InitStructure.WorkMode = ADC_WORKMODE_INDEPENDENT;
131      ADC_InitStructure.MultiChEn = DISABLE;
132      ADC_InitStructure.ContinueConvEn = ENABLE;
133      ADC_InitStructure.ExtTrigSelect = ADC_EXT_TRIG_REG_CONV_SOFTWARE;
134      ADC_InitStructure.DataAlign = ADC_DATA_ALIGN_R;
135      ADC_InitStructure.ChsNumber = 1;
136      ADC_InitStructure.Resolution = ADC_DATA_RES_12BIT;
137      ADC_Init(ADC1, &ADC_InitStructure);
138
139      ADC_ConfigRegularChannel(ADC1, ADC_CH_1, 1, ADC_SAMP_TIME_CYCLES_55_5);
140      /* Enable ADC1 channel 1 positive end connecting to output of PGA1 positive end */
141      ADC_EnableCH1PositiveEndConnetPGA_P(ADC1, ENABLE);
142      /* Enable ADC1 */
143      ADC_Enable(ADC1, ENABLE);
144      /* Check ADC1 Ready */
145      while(ADC_GetFlagStatus(ADC1, ADC_FLAG_RDY) == RESET)
146      {
147          /* Start ADC1 single-ended calibration */
148          ADC_CalibrationOperation(ADC1, ADC_CALIBRATION_SIGNAL_MODE);
149          /* Check the end of ADC1 calibration */
150          while(ADC_GetCalibrationStatus(ADC1))
151          {
152          }
153      }
154  }

```

4.3.31.2 Differential mode:

1. In the case of PGA1, it can be configured as a differential PGA. INPSEL and INMSEL can select a pair of differential channels as inputs, and the output is directly connected to the ADC differential input channels ADC1_VINP1 and ADC1_VINN1.
2. The external connection diagram is shown below.



Note: The VINN of the PGA requires a fixed voltage bias of $VDDA/2$.

3. Using PA1 as the positive input and PA2 as the negative input, with a differential gain of 2 as an example, the corresponding configuration code is shown in the diagram below.

```

107 void PGA_Configuration(void)
108 {
109     PGA_InitType PGA_Init;
110
111     /*Initial comp*/
112     PGA_StructInit(&PGA_Init);
113     PGA_Init.DiffModeEn = ENABLE;
114     PGA_Init.Gain1 = PGA_CTRL_CH1GAIN_1_DIFF_2;
115     PGA_Init.Vmsel = PGA1_CTRL_VMSEL_PA2;
116     PGA_Init.Vpsel = PGA1_CTRL_VPSEL_PA1;
117     PGA_Init.TimeAutoMuxEn = DISABLE;
118     PGA_Init.En1 = DISABLE;
119     PGA_Init.En2 = DISABLE;
120     PGA_Init(&PGA_Init);
121     /*enable PA1-CH1 and CH2 */
122     PGA_Enable(PGA1, PGA_Channel1, ENABLE);
123     PGA_Enable(PGA1, PGA_Channel2, ENABLE);
124 }
125
126 /**
127  * \name    ADC_Init.
128  * \fun     ADC_Init program.
129  * \return  none
130  */
131 void ADC_Init(void)
132 {
133     /* ADCx configuration */
134     ADC_InitStructure.WorkMode = ADC_WORKMODE_INDEPENDENT;
135     ADC_InitStructure.MultiChEn = DISABLE;
136     ADC_InitStructure.ContinueConvEn = ENABLE;
137     ADC_InitStructure.ExtTrigSelect = ADC_EXT_TRIG_REG_CONV_SOFTWARE;
138     ADC_InitStructure.DataAlign = ADC_DAT_ALIGN_R;
139     ADC_InitStructure.ChsNumber = 1;
140     ADC_InitStructure.Resolution = ADC_DATA_RES_12BIT;
141     ADC_Init(ADC1, &ADC_InitStructure);
142
143     ADC_ConfigRegularChannel(ADC1, ADC_CH_1, 1, ADC_SAMP_TIME_CYCLES_55_5);
144     /* Enable ADC1 channel 1 positive end connecting to output of PGA1 positive end. */
145     ADC_EnableCH1PositiveEndConnectPGA_P(ADC1, ENABLE);
146     /* Enable ADC1 channel 1 negative end connecting to output of PGA1 negative end. */
147     ADC_EnableCH1NegativeEndConnectPGA_N(ADC1, ENABLE);
148     /* Enable ADC1 channel 1 work on differential mode. */
149     ADC_SetChannelSingleDiff(ADC1, ADC_CH_1, ADC_DIFFERENTIAL_ENDED);
150     /* Enable ADC1 */
151     ADC_Enable(ADC1, ENABLE);
152     /* Check ADC1 Ready */
153     while (ADC_GetFlagStatus(ADC1, ADC_FLAG_RDY) == RESET)
154     {
155     }
156     /* Start ADC1 differential calibration */
157     ADC_CalibrationOperation(ADC1, ADC_CALIBRATION_DIFF_MODE);
158     /* Check the end of ADC1 calibration */
159     while (ADC_GetCalibrationStatus(ADC1))
160     {
161     }
162 }

```

4.3.31.3 User TRIM:

Due to the characteristics of the amplifier in the PGA, when there is a significant difference between the calibration environment and the user's working environment, user trimming is supported to calibrate the PGA in order to compensate for errors introduced by various components. For specific details, please refer to the "PGA_UserTrimming" example project.

4.3.32 VREFBUF

The VREFBUF module in the N32H47x_48x series is a newly added feature, and it is recommended to refer to the VREFBUF demo for testing.

Note: In certain packages, if the VREF+ pin is connected to the VDDA pin (please refer to the package pin description in the datasheet for specifics), the voltage reference buffer is not available and must remain disabled. Additionally, the RCC_VREFCTRL.HIM bit needs to be set to 1.

The following diagram shows part of the configuration code for the "ADC VREFBUF" example project for the VREFBUF module in the N32H47x_48x series:

```

158 /**
159  * \name ..... VREFBUF_CONFIG.
160  * \fun ..... Configures vrefbuf work mode.
161  * \param ..... VrefBufferMode.
162  * \..... VREFBUF_MODE_OUTPUT
163  * \..... VREFBUF_MODE_INPUT:
164  * \param ..... voltage
165  * \..... VREFBUF_VOLTAGE_SCALE_2_048V..
166  * \..... VREFBUF_VOLTAGE_SCALE_2_5V..
167  * \..... VREFBUF_VOLTAGE_SCALE_2_9V..
168  * \return none
169  */
170 void VREFBUF_CONFIG(VREFBUF_MODE VrefBufferMode, uint32_t voltage)
171 {
172     ....
173     if (VrefBufferMode == VREFBUF_MODE_OUTPUT)
174     {
175         *(uint32_t *)VREFBUF_EN_CTRL |= (1<<10);
176         VREFBUF_SetVoltageScale(voltage);
177         VREFBUF_EnableHIM(DISABLE);
178         VREFBUF_Enable(ENABLE);
179         while (VREFBUF_IsVREFReady() == RESET);
180     }
181     else
182     {
183         *(uint32_t *)VREFBUF_EN_CTRL &= ~(1<<10);
184         VREFBUF_EnableHIM(ENABLE);
185         VREFBUF_Enable(DISABLE);
186     }
187 }
188 /**

```

Note: When using the VREFBUF output, the VREF+ pin cannot be externally connected to a voltage source.

4.3.33 XSPI

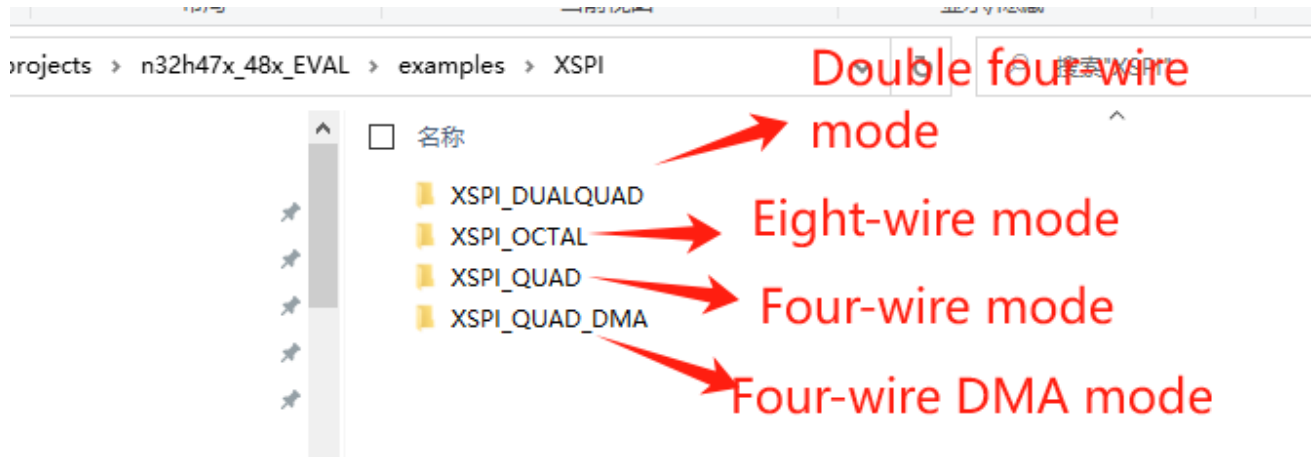
The following table compares the XSPI module driver API functions between the N32G45x series and the N32H47x_48x series. After replacing the driver .c/.h files, application code calls to the XSPI module driver API functions can be substituted according to the table below.

API description	API name		Is the API consistent	Difference declaration
	N32G45x series	N32H47x_48x series		
Enable XSPI	void QSPI_Cmd(bool cmd)	void XSPI_Cmd(FunctionalState cmd)	NO	Function names are inconsistent
Enable XIP	void QSPI_XIP_Cmd(bool cmd)	void XSPI_XIP_Cmd(FunctionalState cmd)	NO	Function names are inconsistent
XSPI reset	void QSPI_DeInit(void)	void XSPI_DeInit(void)	NO	Function names are inconsistent
Initialize XSPI	void QspiInitConfig(QSPI_InitType* QSPI_InitStruct)	void XSPI_Init(XSPI_InitType* XSPI_InitStruct)	NO	The function name and input parameter structure are inconsistent: QSPI_InitStruct→XSPI_InitStruct
Initialize XSPI structure	-	void XSPI_StructInit(XSPI_InitType* XSPI_InitStruct)	NO	N32G45x series does not have this API
Pin configuration	void QSPI_GPIO(QSPI_NSS_PORT_SEL qspi_nss_port_sel, bool IO1_Input, bool IO3_Output)	-	NO	N32H47x_48x series does not have this API

Configure DMA sending threshold	void QSPI_Tx_DMA_CTRL_Config(uint8_t Cmd, uint8_t TxDataLevel)	void XSPI_ConfigDMATxLevel(uint32_t TxDataLevel) void XSPI_EnableDMA(XSPI_DMAREQ_TX, FunctionalState Cmd)	NO NO	The N32H47x_48x series splits the DMA send threshold and enable into two functions, adding the XSPI_DMAREQ_TX parameter
Configure DMA receiving threshold	void QSPI_Rx_DMA_CTRL_Config(uint8_t Cmd, uint8_t RxDataLevel)	void XSPI_ConfigDMARxLevel(uint32_t RxDataLevel) void XSPI_EnableDMA(XSPI_DMAREQ_RX, FunctionalState Cmd)	NO NO	The N32H47x_48x series splits the DMA receive threshold and enable into two functions, adding the XSPI_DMAREQ_RX parameter
Configure interrupt	-	void XSPI_ConfigInt(uint16_t XSPI_IT, FunctionalState Cmd)	NO	N32G45x series does not have this API
Get interrupt status	uint16_t QSPI_GetITStatus(uint16_t FLAG)	uint16_t XSPI_GetINTStatus(uint16_t FLAG)	NO	Function name and input parameter FLAG are inconsistent: N32G45x serial parameter: QSPI_ISTS_TXFEIS QSPI_ISTS_TXFOIS QSPI_ISTS_RXFUIS QSPI_ISTS_RXFOIS QSPI_ISTS_RXFFIS QSPI_ISTS_MMCIS QSPI_ISTS_XRXOIS N32H47x_48x serial parameter: XSPI_ISTS_TXFEIS XSPI_ISTS_TXFOIS XSPI_ISTS_RXFUIS XSPI_ISTS_RXFOIS XSPI_ISTS_RXFFIS XSPI_ISTS_MMCIS XSPI_ISTS_XRXOIS XSPI_ISTS_TXUIS
Clear interrupt state flag	void QSPI_ClearITFLAG(uint16_t FLAG)	void XSPI_ClearITBit(uint16_t XSPI_IT)	NO	Function name and input parameter FLAG are inconsistent: N32G45x serial parameter: QSPI_ISTS_TXFEIS QSPI_ISTS_TXFOIS QSPI_ISTS_RXFUIS QSPI_ISTS_RXFOIS QSPI_ISTS_RXFFIS QSPI_ISTS_MMCIS QSPI_ISTS_XRXOIS QSPI_XIP_RXFOI_CLR_XRXFOIC N32H47x_48x serial parameter: XSPI_IT_TXUIM XSPI_IT_XRXOIM XSPI_IT_MMCIM XSPI_IT_RXFFIM XSPI_IT_RXFOIM XSPI_IT_RXFUM XSPI_IT_TXFOIM XSPI_IT_TXFEIM
	void QSPI_XIP_ClearITFLAG(uint16_t FLAG)		NO	
Get Flag status	-	FlagStatus XSPI_GetFlagStatus(uint32_t XSPI_FLAG)	NO	N32G45x series does not have this API
Get busy status	bool GetQspiBusyStatus(void)	FlagStatus XSPI_GetBusyStatus(void)	NO	Function names are inconsistent
Gets the full state of the sent FIFO	bool GetQspiTxDataBusyStatus(void)	FlagStatus XSPI_GetTxDataBusyStatus(void)	NO	Function names are inconsistent
Gets the empty state of the sent FIFO	bool GetQspiTxDataEmptyStatus(void)	FlagStatus XSPI_GetTxDataEmptyStatus(void)	NO	Function names are inconsistent
Get receive FIFO empty state	bool GetQspiRxHaveDataStatus(void)	FlagStatus XSPI_GetRxHaveDataStatus(void)	NO	Function names are inconsistent
Get the received	bool GetQspiRxDataFullStatus(void)	FlagStatus XSPI_GetRxDataFullStatus(void)	NO	Function names are inconsistent

FIFO full state				
Get data conflict status	bool GetQspiDataConflictErrorStatus(void)	FlagStatus XSPI_GetDataConflictErrorStatus(void)	NO	Function names are inconsistent
Send 1 word data	void QspiSendWord(uint32_t SendData)	void XSPI_SendData(uint32_t SendData)	NO	Function names are inconsistent
Read 1 word data	uint32_t QspiReadWord(void)	uint32_t XSPI_ReceiveData(void)	NO	Function names are inconsistent
Gets the DAT0 register pointer	uint32_t QspiGetDataPointer(void)	uint32_t XSPI_GetDataPointer(void)	NO	Function names are inconsistent
Gets the number of Fifos received	uint32_t QspiReadRxFifoNum(void)	uint32_t XSPI_ReadRxFifoNum(void)	NO	Function names are inconsistent
Gets the number of data sent FIFOs	-	uint32_t XSPI_ReadTxFifoNum(void)	NO	N32G45x series does not have this API
Clear the receiving FIFO	void ClrFifo(void)	void XSPI_ClrFifo(void)	NO	Function names are inconsistent
Read N word data	uint32_t GetFifoData(uint32_t* pData, uint32_t Len)	uint32_t XSPI_GetFifoData(uint32_t* pData, uint32_t Len)	NO	Function names are inconsistent
Send N words and get the returned data	void QspiSendAndGetWords(uint32_t* pSrcData, uint32_t* pDstData, uint32_t cnt)	void XSPI_SendAndGetWords(uint32_t* pSrcData, uint32_t* pDstData, uint32_t cnt)	NO	Function names are inconsistent
Send 1 word of data and get the returned data	uint32_t QspiSendWordAndGetWords(uint32_t WrData, uint32_t* pRdData, uint8_t LastRd)	uint32_t XSPI_SendWordAndGetWords(uint32_t WrData, uint32_t* pRdData, uint8_t LastRd)	NO	Function names are inconsistent
Set transmission type	-	void XSPI_SetTransType(uint32_t TransType)	NO	N32G45x series does not have this API
Set waiting period	-	void XSPI_SetWaitCycles(uint32_t WAITCYCLES)	NO	N32G45x series does not have this API
Set the received FIFO full threshold	-	void XSPI_SetRXFIFOLevel(uint32_t fifo_len)	NO	N32G45x series does not have this API
Set the null threshold for sending FIFO	-	void XSPI_SetTXFIFOLevel(uint32_t fifo_len)	NO	N32G45x series does not have this API
Set the threshold for transmitting FIFO to start transmission	-	void XSPI_SetTXFIFOStartLevel(uint32_t fifo_len)	NO	N32G45x series does not have this API
Get the received FIFO full threshold	-	uint8_t XSPI_GetRXFIFOLevel(void)	NO	N32G45x series does not have this API
Get the empty threshold of the sent FIFO	-	uint8_t XSPI_GetTXFIFOLevel(void)	NO	N32G45x series does not have this API

The example project code for the XSPI module differs significantly between the N32G45x series and the N32H47x_48x series. It is recommended to directly refer to the following N32H47x_48x series routines and complete application development based on the code and the readme file.



4. Version History

Version	Date	Changes
V1.0.0	2024.11.12	Initial version

6. Disclaimer

This document is the exclusive property of NSING TECHNOLOGIES PTE. LTD. (Hereinafter referred to as NSING). This document, and the product of NSING described herein (Hereinafter referred to as the Product) are owned by NSING under the laws and treaties of Republic of Singapore and other applicable jurisdictions worldwide. The intellectual properties of the product belong to Nations Technologies Inc. and Nations Technologies Inc. does not grant any third party any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only. NSING reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NSING and obtain the latest version of this document before placing orders. Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product.

NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, 'Insecure Usage'. Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to supporter sustain life. All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage Any express or implied warranty with regard to this document or the Product, including, but not limited to. The warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law. Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.