Development guide

N32WB452 Series Quick Development Guide

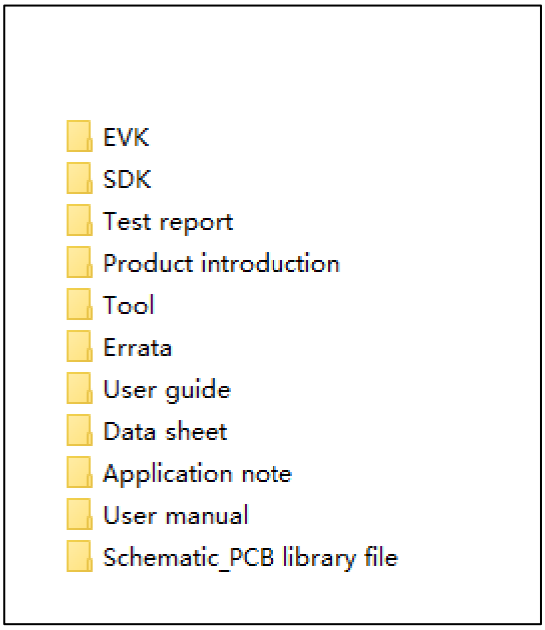
Introduction

The purpose of this document is to allow users to quickly familiarize themselves with the development kits of N32WB452 series MCUs and the related settings of Keil MDK-ARM, so as to reduce the preparation time in the early stage of development and the difficulty of development.

CONTENTS

1. Introduction to the Development Kit

Development kits usually include related product profiles, data sheets, user manuals, application notes and reference guides, as well as development evaluation board EVK, software development kit SDK, related download tools and documentation.



EVK
SDK
Test report
Product introduction
Tool
Errata
User guide
Data sheet
Application note
User manual
Schematic_PCB library file

- **Product introduction:**

  Usually starting with the letter "PB", the chip feature introduction document is used to quickly understand the chip resource overview, order model, package information, etc., and is used in the chip selection stage before project development.

- **Data sheet:**

  Usually starting with the letter "DS", the chip technical parameter document is used to understand the chip module resources, pin definitions, electrical characteristics, packaging information, etc., used in the hardware design stage and software development stage.

- **User manual:**

  Usually starting with the letter "UM", the detailed introduction of each functional module of the chip, including the internal structure of the module, instructions for use, register description, etc., is used as the main reference document in the software development stage.

- **Application note:**

  Usually starting with the letter "AN", the detailed usage instructions of a certain functional module of the chip, including the implementation principle, software process, introduction to the core code implementation, operation instructions, etc., are used as the main reference documents in the software development process.

- **User guide:**

  Usually starting with the letter "UG", the usage guide of the firmware library of the chip-specific function module, including the firmware architecture, software process, core code implementation introduction, operation instructions, etc., as the core reference document in the software development process.

- **EVK:**
  - ■ **N32WB45xL_EVB (QFN88)**

  Contains all the above development board information, including schematic diagram/PCB/development board instruction manual.

- **SDK:**

  Software Development Design Kit, including Keil DFP pack, firmware library (low-level driver code and reference routines).

2. Introduction to the development board

2.1 N32WB45xL_EVB (QFN88) full-function evaluation board

2.1.1 Introduction to N32WB45xL_EVB

N32WB45xL_EVB is a full-function evaluation board for the National Technology BLE 5.0 SOC chip N32WB452LEQ6 (QFN88). The functional structure diagram of the development board is described as follows:

Figure 2-1 Front view of N32WB45xL_EVB V1.1 development board

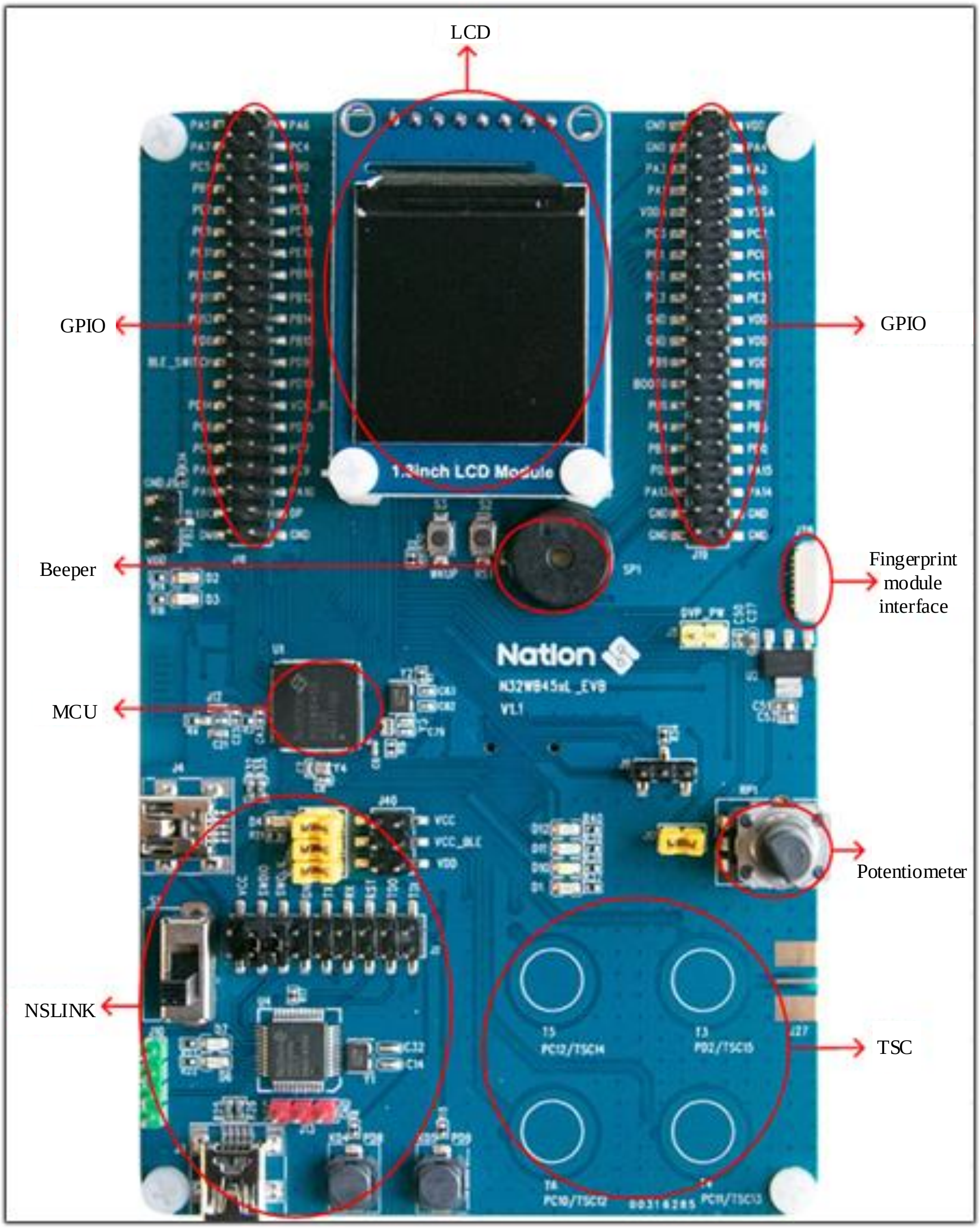
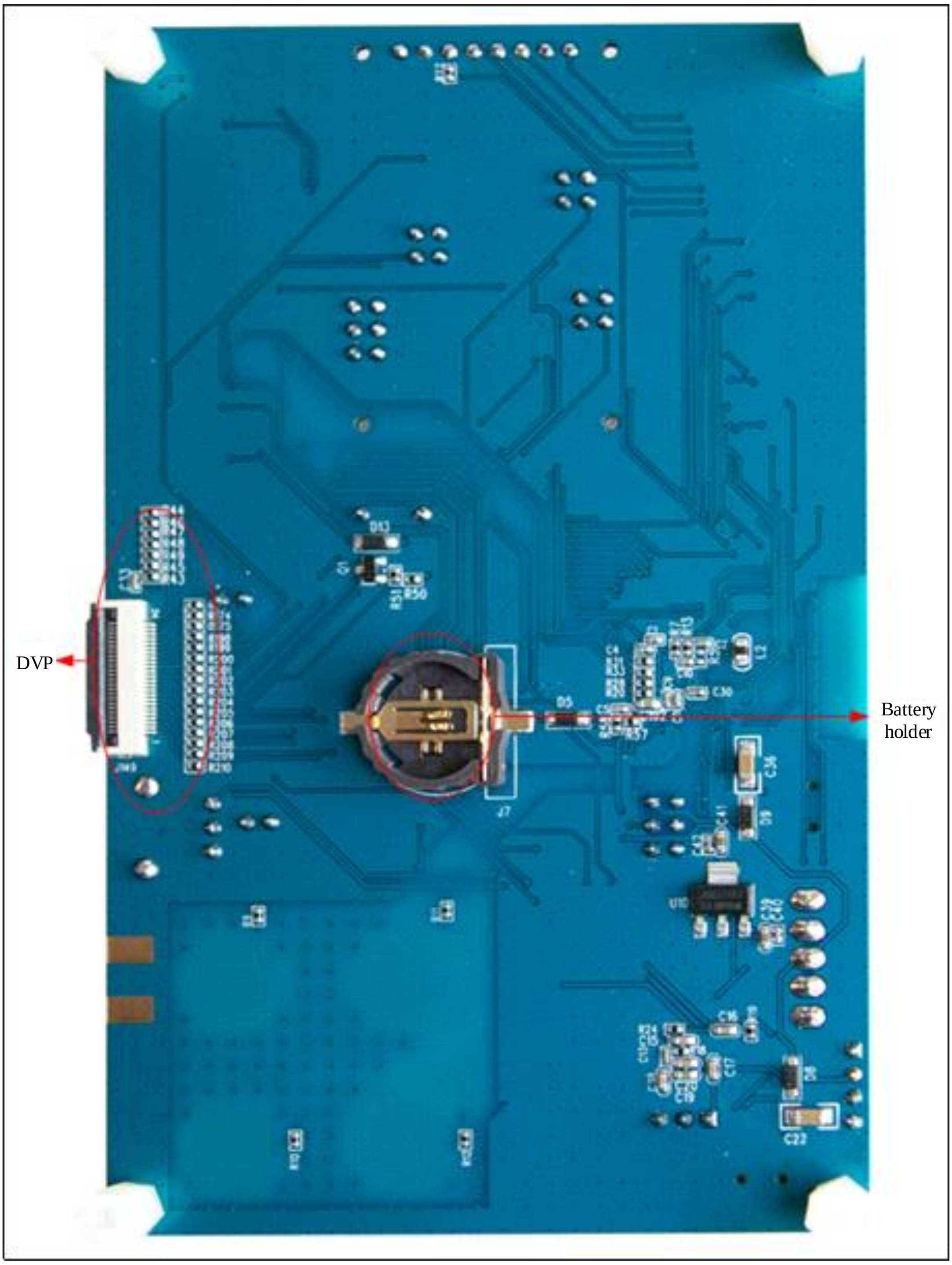


Figure 2-2 Reverse view of N32WB45xL_EVB V1.1 development board



2.1.2 N32WB45xL_EVB development board interface definition description

1) Power supply for the development board

The development board can be powered by the USB interface (J3 or J4), which is connected to the 3.3V LDO input port through the switch S1. The output end is divided into three channels. The first channel is the power supply VCC_MCU for the MCU, and the interface is J6; the second channel is the power supply VCC3.3 for other peripheral chips except MCU, and the interface is J5; the third channel is the power supply VCC_BLE reserved for Bluetooth, and the interface is J15.

2) **USB interface (J3)**

Use MicroUSB interface (J4) to connect to the onboard NSLINK chip (U4), which can be used for program download and debugging. When using it, pay attention to connecting the jumpers J2 and J20.

3) **USB interface (J4)**

Use MicroUSB interface (J4), connect MCU DP DM, can be used for USB interface communication.

4) **Debug interface (J2&J20)**

It is used to connect other debugging tools for program download and debugging, and supports two modes of SWD/JTAG. J20 is the download port in SWD mode, and it needs to be disconnected from the NSLINK chip when using it. J20&J2 can be used as JTAG mode download port.

5) **Reset and wake-up buttons (S2, S3)**

S2 and S3 are reset buttons and wake buttons respectively, which are connected to NRST pins and PA0-WKUP pins of the chip respectively for chip reset and wake functions.

6) **BOOT (J9, J11)**

J9 and J11 are the jumpers of the BOOT0 and BOOT1 pins of the MCU main chip respectively connected to the reserved pull-up and pull-down resistors.

7) **Battery holder J7(BAT)**

The battery holder can hold a CR1220 battery, which is connected to the VBAT pin of the chip to provide power.

8) **GPIO port (J14, J19, J16, J18)**

The GPIO interfaces of the chip are all led out, pins are inserted and GND is reserved. See the schematic diagram for the interface definition.

9) **FLASH chip (U7)**

The development board has an onboard SPI flash chip, bit number U7, which is connected to the MCU SPI pins through the J1, J21, J30, and J32 interfaces.

10) **Temperature and humidity sensor (U8)**

The development board has an onboard temperature and humidity sensor HDC2010, bit number U8, which communicates with the MCU through I2C2.

11) **G-SENSOR (U5)**

The development board has a G-SENSOR chip QMA7981, bit number U5, which communicates with the MCU through I2C1.

12) **DVP interface (J149)**

The development board has an onboard DVP interface, bit number J149, 24pin connector, connected to the camera through a flexible cable.

13) **LCD (U6)**

The development board has an onboard LCD screen, bit number U6, which can be directly used and debugged by customers.

14) Fingerprint module (J28)

The development board has an onboard fingerprint module interface, bit number J28, which can be connected to an external fingerprint module for customer debugging.

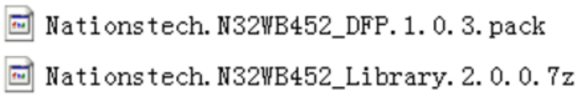
2.1.3 **N32WB45xL_EVB development board jumper function description**

Table 2-1 Development board jumper function description

No.	Jumper bit number	Jumper function	Instructions for use
1	J5、J6、J15	3.3V power supply jumper	J6: MCU power supply interface J5: Power supply interface for other peripherals J15: BLE reserved power supply interface
2	J1、J21、J30、J32	SPI FLASH jumper	If using FLASH, you need to connect J1, J21, J30, J32
3	J8	DVP power supply interface	If using DVP, connect J8
4	J2、J20	Debug interface	J20: SWD mode download port J20&J2: JTAG mode download port
5	J9,J11	BOOT jumper	J9： BOOT0 J11：BOOT1
6	J17	Adjustable resistor jumpers	If the ADC function is used, connect this jumper

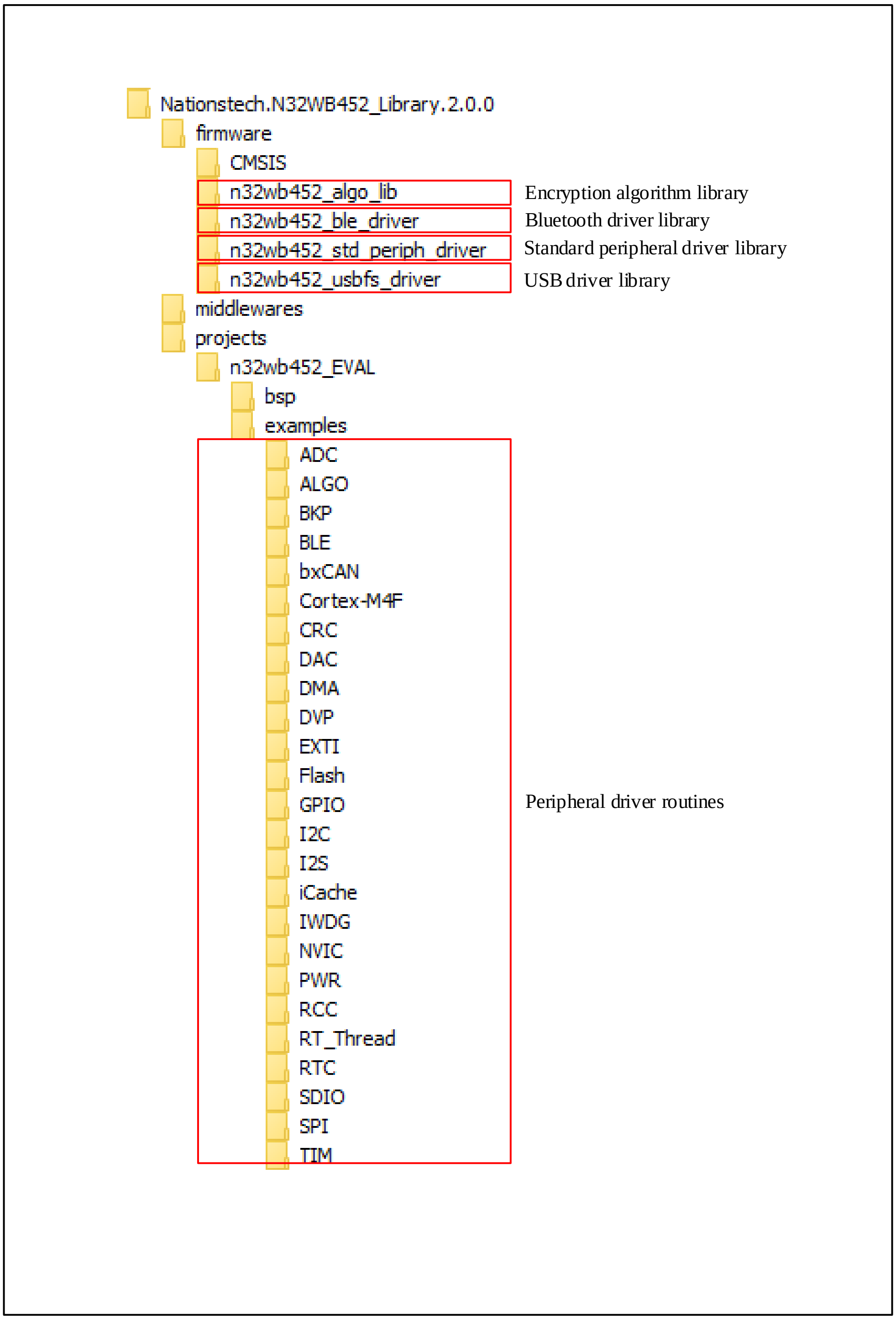
3. Introduction to the firmware development kit

Usually the SDK provides a Keil PACK package, as well as the SDK source code compressed file package.



Nationstech.N32WB452_DFP.1.0.3.pack
Nationstech.N32WB452_Library.2.0.0.7z

The SDK directory is a file named after the firmware library version. The directory structure and description are as follows:



```
Nationstech.N32WB452_Library.2.0.0
    firmware
        CMSIS
        n32wb452_algo_lib                Encryption algorithm library
        n32wb452_ble_driver              Bluetooth driver library
        n32wb452_std_periph_driver       Standard peripheral driver library
        n32wb452_usbfs_driver            USB driver library
    middlewares
    projects
        n32wb452_EVAL
            bsp
            examples
                ADC
                ALGO
                BKP
                BLE
                bxCAN
                Cortex-M4F
                CRC
                DAC
                DMA
                DVP
                EXTI
                Flash
                GPIO                     Peripheral driver routines
                I2C
                I2S
                iCache
                IWDG
                NVIC
                PWR
                RCC
                RT_Thread
                RTC
                SDIO
                SPI
                TIM
```

3.1 Firmware

- **CMSIS:**

  The Microcontroller Software Interface Standard, a vendor-independent hardware abstraction layer for the Cortex-M processor family, CMSIS provides a common interface between the kernel and peripherals, real-time operating systems, and intermediate devices.

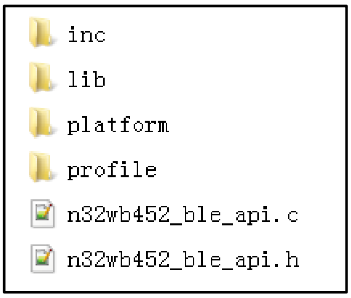
  - Contains the name definition, address definition and configuration function of the register device used to access the kernel. This interface includes debug channel definitions.
  - Provide definitions of all peripherals on-chip, including all peripheral register header files, startup files, and system initialization template files.
  - DSP library, providing optimized signal processing algorithms.

- **n32wb452_algo_lib:**

  - Algorithm library files, including: encryption algorithm, HASH algorithm, random number algorithm, etc.

- **n32wb452_ble driver:**

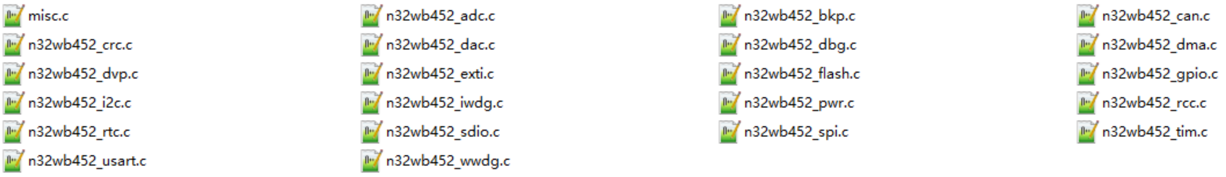
  - BLE driver files, including the header file inc used by the ble algorithm library, the Bluetooth driver library lib, the chip-related resource file platform, and ble frofile. The Bluetooth driver library lib contains the library files of the two development environments of IAR and KEIL, including the Bluetooth protocol stack driver library (host.lib, host_16bit.lib, host_16bit.a) and the Bluetooth underlying driver library (n32wb452_ble.lib, n32wb452_ble_16bit.lib, n32wb452_ble_16bit.a).
  - When using the BLE function, the user can call the API function in n32wb452_ble_api.c to realize the registration, callback, broadcast, connection, etc. of the Bluetooth function.



inc
lib
platform
profile
n32wb452_ble_api.c
n32wb452_ble_api.h

- **n32wb452_std_periph_driver:**

  - Standard driver functions for chip peripherals, including .c source files and .h header files. Users can migrate to the project and quickly complete the use of a peripheral module.



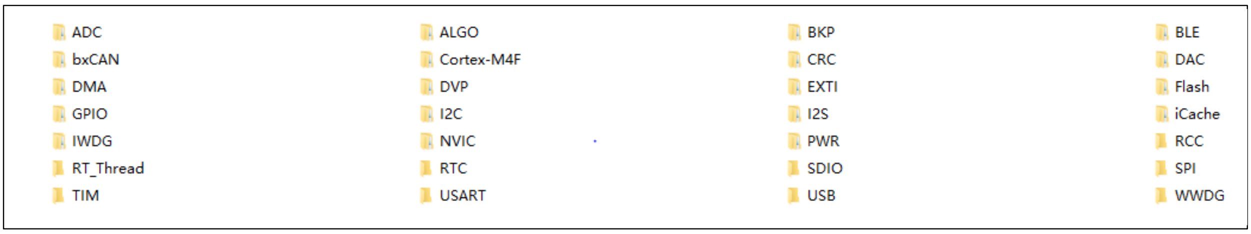
misc.c
n32wb452_adc.c
n32wb452_bkp.c
n32wb452_can.c
n32wb452_crc.c
n32wb452_dac.c
n32wb452_dbg.c
n32wb452_dma.c
n32wb452_dvp.c
n32wb452_exti.c
n32wb452_flash.c
n32wb452_gpio.c
n32wb452_i2c.c
n32wb452_iwdg.c
n32wb452_pwr.c
n32wb452_rcc.c
n32wb452_rtc.c
n32wb452_sdio.c
n32wb452_spi.c
n32wb452_tim.c
n32wb452_usart.c
n32wb452_wwdg.c

- **n32wb452_usbfs_driver:**

  - USB device driver library files, including .c source files and .h header files. The necessary files when developing USB devices and completing the construction of the USB bottom layer protocol stack, users only need to care about the development of the application layer.

3.2 Projects

- **BSP:**
  - Contains the debugging serial port print function file/LOG, which is used to print various debugging information during the debugging process.
- **Examples:**
  - Routine projects including various peripheral function modules are the focus of development engineers, and realize the basic application development of each peripheral module. Users can quickly understand chip usage, programming skills, and module transplantation through these routine projects.

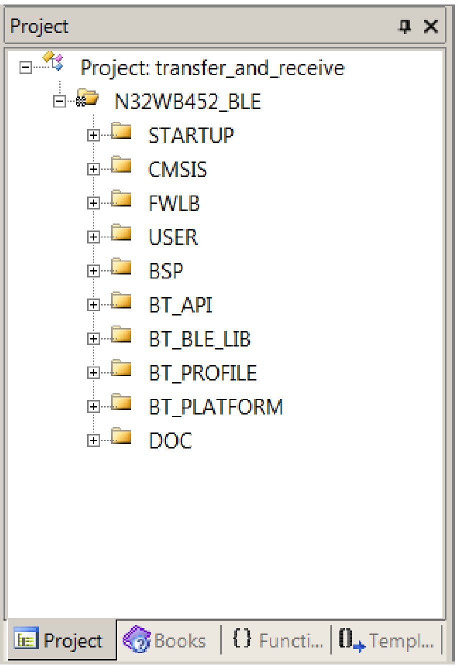


ADC	ALGO	BKP	BLE
bxCAN	Cortex-M4F	CRC	DAC
DMA	DVP	EXTI	Flash
GPIO	I2C	I2S	iCache
IWDG	NVIC	PWR	RCC
RT_Thread	RTC	SDIO	SPI
TIM	USART	USB	WWDG

Under normal circumstances, users can choose a project with the closest application as the initial code for development according to actual needs.

3.3 Projects framework

Take the BLE Slave project as an example:



```
Project: transfer_and_receive
  N32WB452_BLE
    STARTUP
    CMSIS
    FWLB
    USER
    BSP
    BT_API
    BT_BLE_LIB
    BT_PROFILE
    BT_PLATFORM
    DOC
```

- **STARTUP:**
  - Contains the startup file of N32WB452.

- **CMSIS:**
  - Contains the system_n32wb452.c source file.
- **FWLIB:**
  - Contains the driver files for n32wb452.
- **USER:**
  - Contains the development board carrying the user application layer files.
- **BSP:**
  - Contains the BSP file of n32wb452.
- **BT_API:**
  - Contains BLE API interface files.
- **BT_BLE_LIB:**
  - Contains BLE Bluetooth protocol stack driver library and underlying driver library files.
- **BT_PROFILE:**
  - Contains the BLE Profile file.
- **BT_PLATFORM:**
  - Contains source files for BLE using n32wb452 chip related resources.
- **DOC:**
  - Routine brief description document

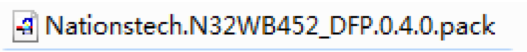
4. Compilation environment and configuration

4.1 Compile environment installation

Please install the KEIL MDK-ARM development environment, and the version requirement is V5.26 or later. The CMSIS architecture of MDKV5 and later versions can support the online update function. In the future, Nations will place the latest CMSIS version online, and users can update directly through the Pack Installer in the Keil environment. (If the amount of compiled code exceeds 32K, you need to purchase a KEIL product license key).

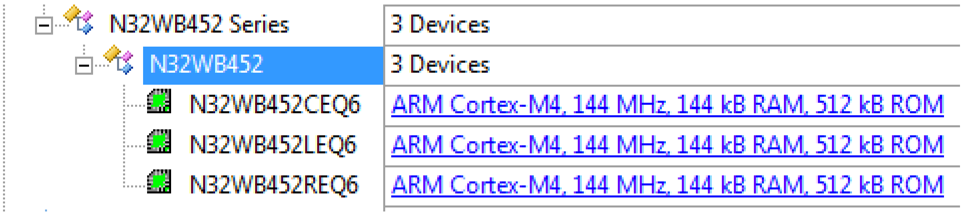
4.2 Firmware support package installation

Please install the firmware support package:



Nationstech.N32WB452_DFP.0.4.0.pack

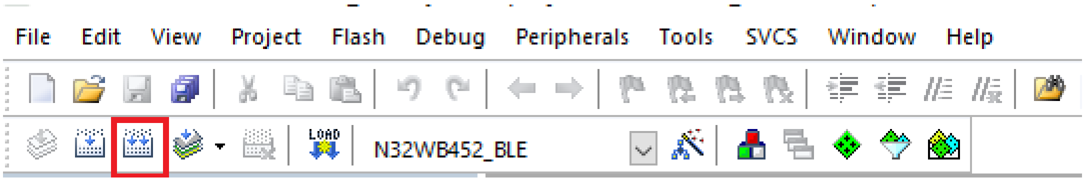
After the installation is successful, you can see the list of supported devices in the Pack Installer interface of KEIL:



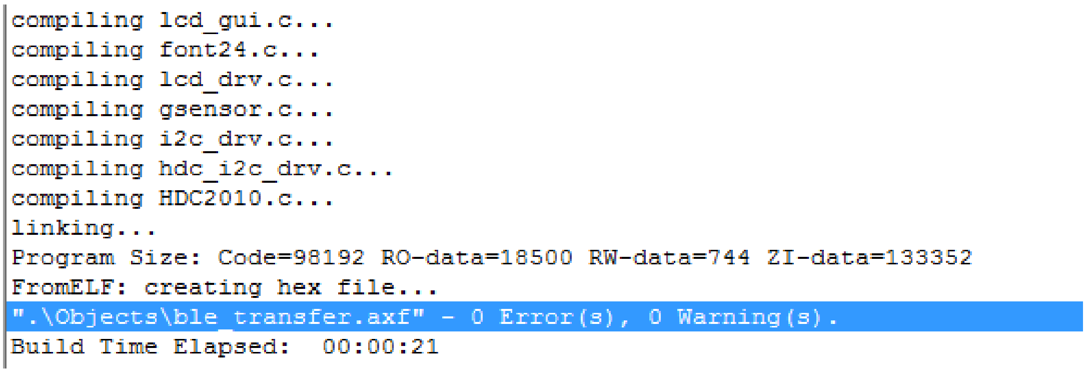
N32WB452 Series	3 Devices
N32WB452	3 Devices
N32WB452CEQ6	ARM Cortex-M4, 144 MHz, 144 kB RAM, 512 kB ROM
N32WB452LEQ6	ARM Cortex-M4, 144 MHz, 144 kB RAM, 512 kB ROM
N32WB452REQ6	ARM Cortex-M4, 144 MHz, 144 kB RAM, 512 kB ROM

4.3 Software compilation

Select a project in the SDK and open it, click Global Compile.



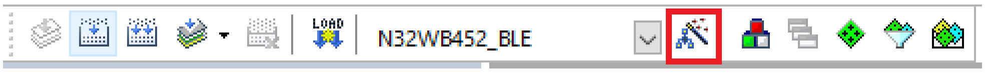
The default unmodified project can be compiled successfully.



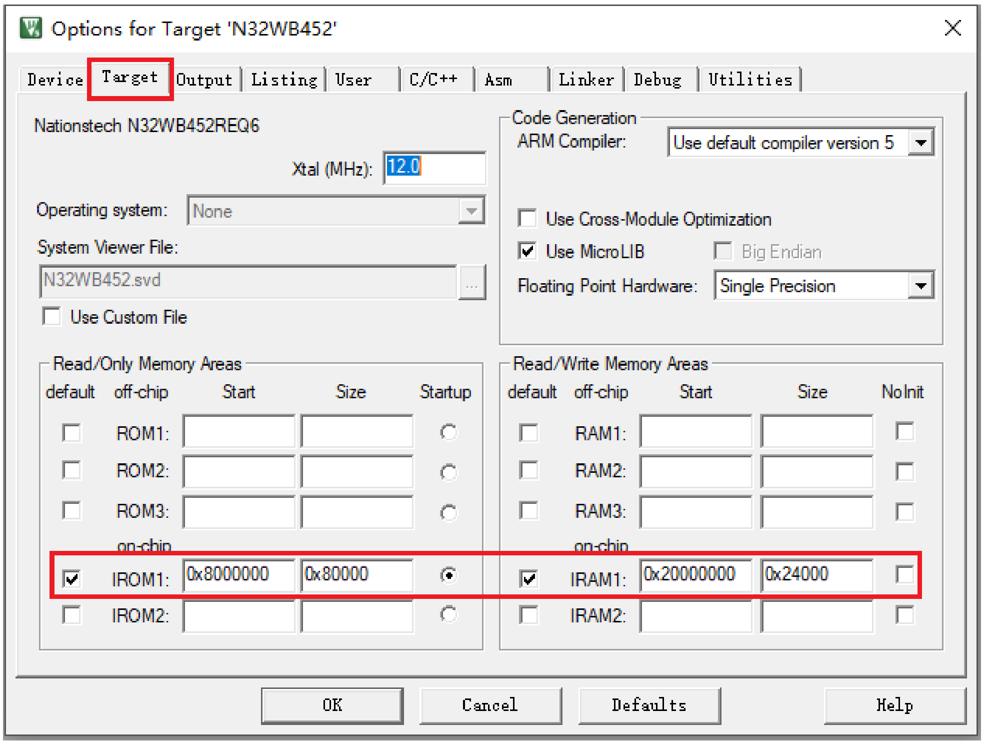
```
compiling lcd_gui.c...
compiling font24.c...
compiling lcd_drv.c...
compiling gsensor.c...
compiling i2c_drv.c...
compiling hdc_i2c_drv.c...
compiling HDC2010.c...
linking...
Program Size: Code=98192 RO-data=18500 RW-data=744 ZI-data=133352
FromELF: creating hex file...
".\Objects\ble_transfer.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:21
```

4.4 Compile environment configuration

Click the magic wand to open the OPTION setting interface.



1) Set the size of ROM and RAM in the Target interface (set according to different models)



Options for Target 'N32WB452'

Nationstech N32WB452REQ6

Xtal (MHz): 12.0

Operating system: None

System Viewer File: N32WB452.svd

□ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

□ Use Cross-Module Optimization

☑ Use MicroLIB □ Big Endian

Floating Point Hardware: Single Precision

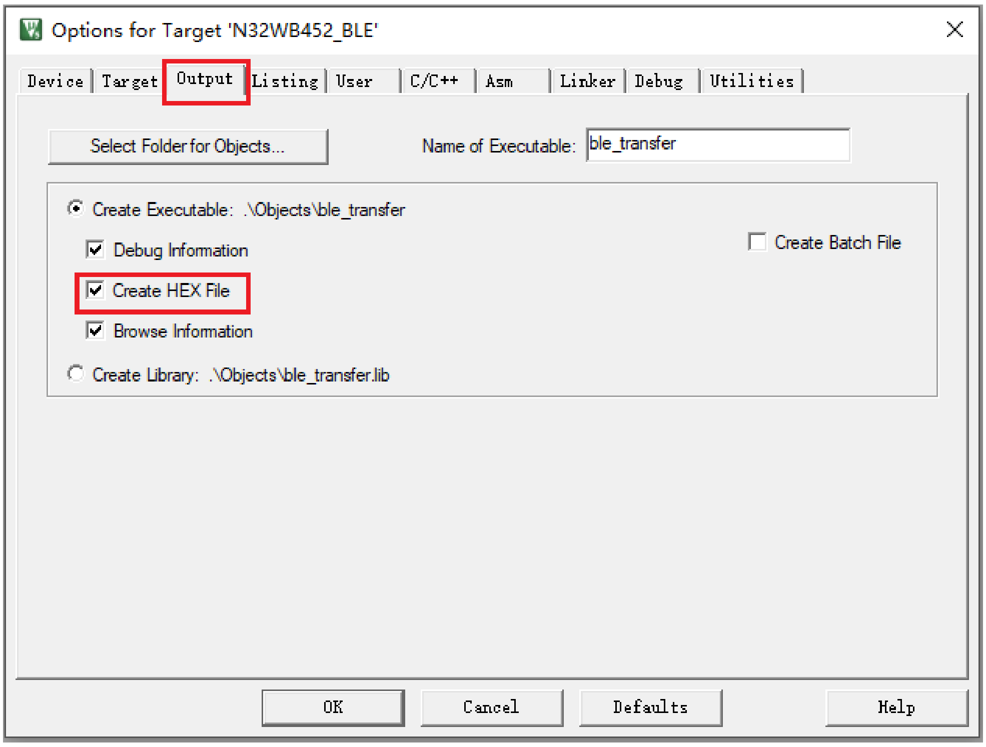
Read/Only Memory Areas

default	off-chip	Start	Size	Startup
□	ROM1:			○
□	ROM2:			○
□	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x80000	●
□	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
□	RAM1:			□
□	RAM2:			□
□	RAM3:			□
	on-chip			
☑	IRAM1:	0x20000000	0x24000	□
□	IRAM2:			□

2) Select the hex output file in the Output interface



Options for Target 'N32WB452_BLE'

Name of Executable: ble_transfer

● Create Executable: .\Objects\ble_transfer

☑ Debug Information

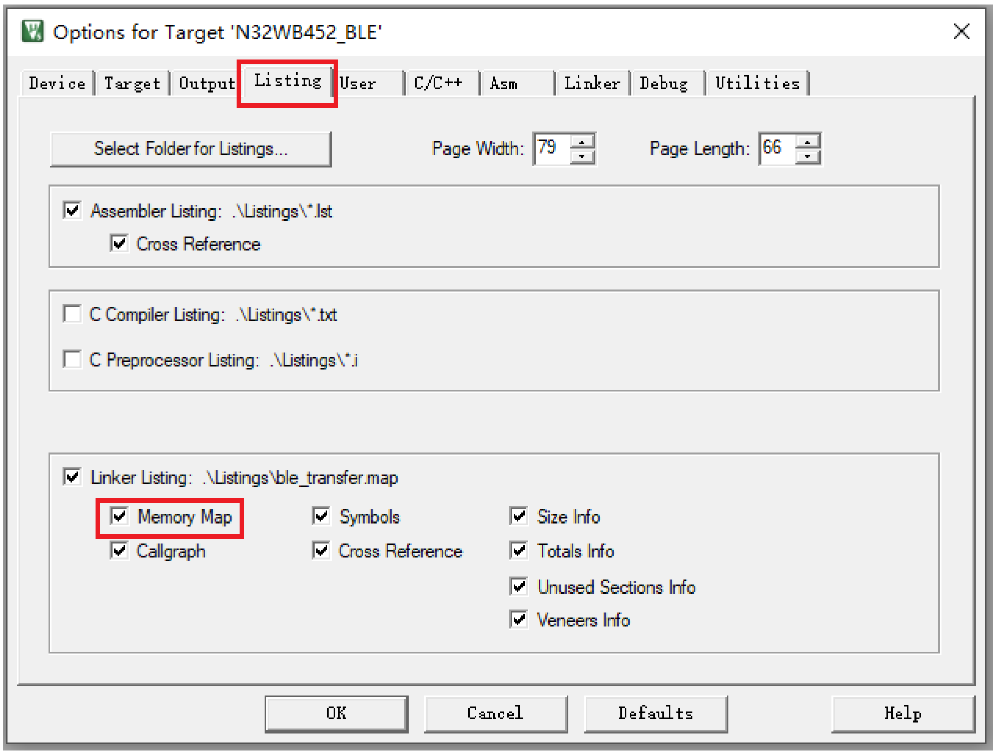
☑ Create HEX File

☑ Browse Information

○ Create Library: .\Objects\ble_transfer.lib

☐ Create Batch File

3) Memory Map file generation



Options for Target 'N32WB452_BLE'

Page Width: 79

Page Length: 66

☑ Assembler Listing: .\Listings*.lst

☑ Cross Reference

☐ C Compiler Listing: .\Listings*.txt

☐ C Preprocessor Listing: .\Listings*.i

☑ Linker Listing: .\Listings\ble_transfer.map

☑ Memory Map

☑ Callgraph

☑ Symbols

☑ Cross Reference

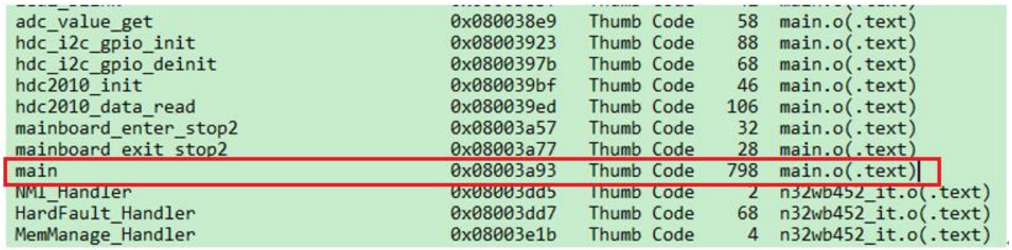
☑ Size Info

☑ Totals Info

☑ Unused Sections Info

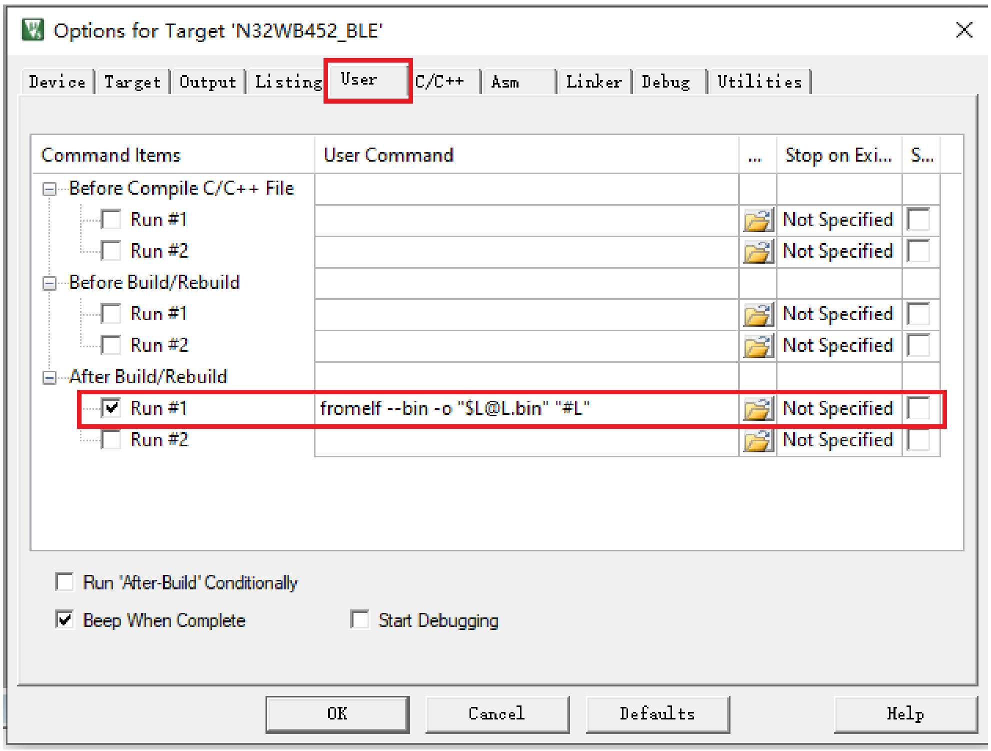
☑ Veneers Info

The .map file records the usage of program storage space, and it is convenient to view the address space and size of variables, constants, and functions, as shown in the following figure.



```
adc_value_get                            0x080038e9   Thumb Code    58  main.o(.text)
hdc_i2c_gpio_init                        0x08003923   Thumb Code    88  main.o(.text)
hdc_i2c_gpio_deinit                      0x0800397b   Thumb Code    68  main.o(.text)
hdc2010_init                             0x080039bf   Thumb Code    46  main.o(.text)
hdc2010_data_read                        0x080039ed   Thumb Code   106  main.o(.text)
mainboard_enter_stop2                    0x08003a57   Thumb Code    32  main.o(.text)
mainboard_exit_stop2                     0x08003a77   Thumb Code    28  main.o(.text)
main                                     0x08003a93   Thumb Code   798  main.o(.text)
NMI_Handler                              0x08003dd5   Thumb Code     2  n32wb452_it.o(.text)
HardFault_Handler                        0x08003dd7   Thumb Code    68  n32wb452_it.o(.text)
MemManage_Handler                        0x08003e1b   Thumb Code     4  n32wb452_it.o(.text)
```

4) **If you need to generate a burning file in bin format, copy fromelf --bin -o "$L@L.bin" "#L" to the following image**



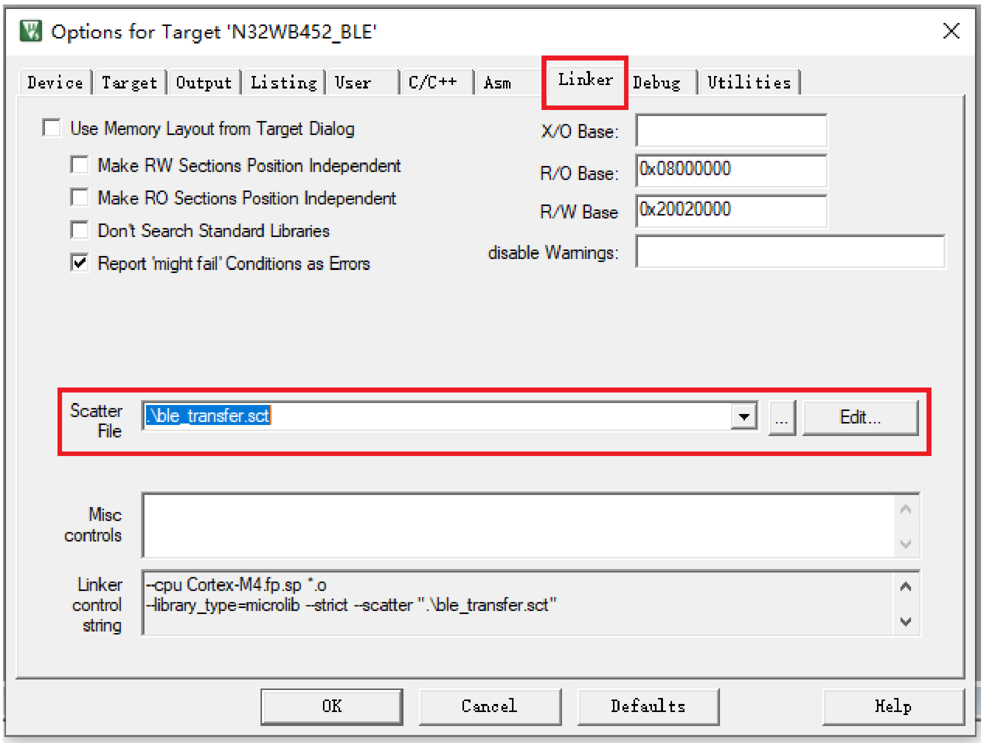
Options for Target 'N32WB452_BLE'

Command Items	User Command	...	Stop on Exi...	S...
Before Compile C/C++ File				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
Before Build/Rebuild				
□ Run #1			Not Specified	□
□ Run #2			Not Specified	□
After Build/Rebuild				
☑ Run #1	fromelf --bin -o "$L@L.bin" "#L"		Not Specified	□
□ Run #2			Not Specified	□

□ Run 'After-Build' Conditionally

☑ Beep When Complete   □ Start Debugging

5) **Linker Scatter File settings (set according to the actual needs of the project)**



Options for Target 'N32WB452_BLE'

□ Use Memory Layout from Target Dialog

□ Make RW Sections Position Independent

□ Make RO Sections Position Independent

□ Don't Search Standard Libraries

☑ Report 'might fail' Conditions as Errors

X/O Base:

R/O Base: 0x08000000

R/W Base 0x20020000

disable Warnings:

Scatter File: .\ble_transfer.sct

Misc controls:

Linker control string:

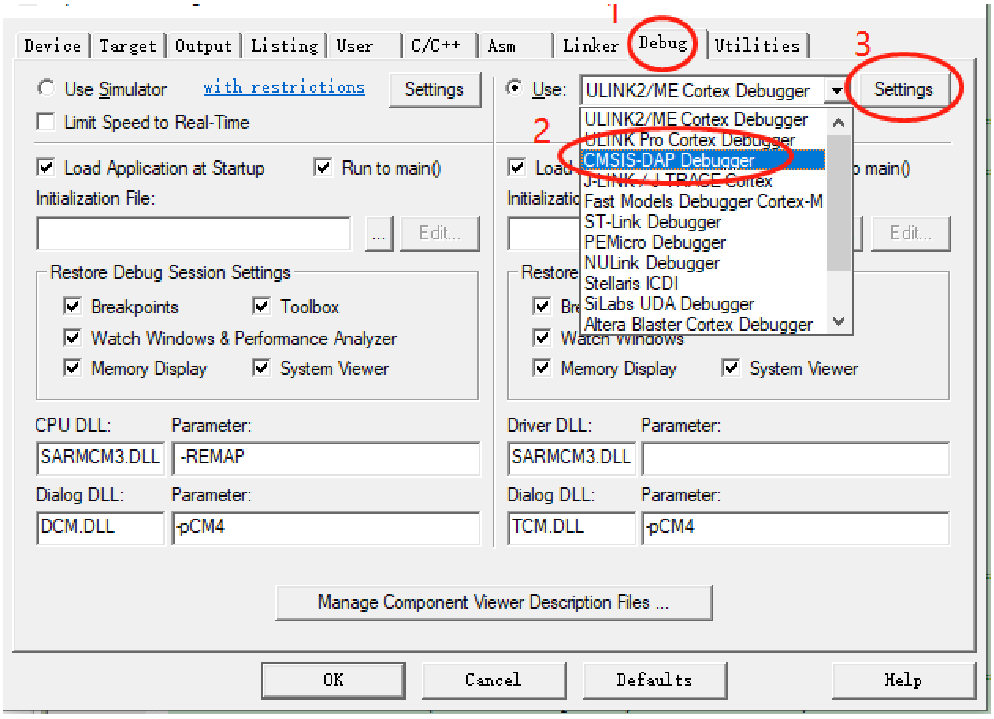
```
--cpu Cortex-M4.fp.sp *.o
--library_type=microlib --strict --scatter ".\ble_transfer.sct"
```

.sct is a text file that specifies how the ARM linker allocates the storage addresses of RO, RW, ZI and other data when generating an image file by writing a scatter-loading file. Under normal circumstances, the ARM linker will generate image files by default.

In some cases, we want to put some data at a specified address.

- There is a complex address mapping: for example, code and data need to be stored in multiple areas (usually used in boot code).
- There are multiple memory types: including Flash, ROM, SDRAM, etc. According to the characteristics of code and data, they are stored in different memories.
- Use the Scatter file to place a function at a fixed address, regardless of whether its application has been changed or recompiled.
- Memory-mapped IO: A data segment can be placed at a precise address by using scatter file.

6) Simulation related settings in the Debug interface



○ Use Simulator

☐ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter: -REMAP

Dialog DLL: DCM.DLL Parameter: -pCM4

● Use: ULINK2/ME Cortex Debugger

☑ Load ☑ Run to main()

Initialization File:

Restore Debug Session Settings

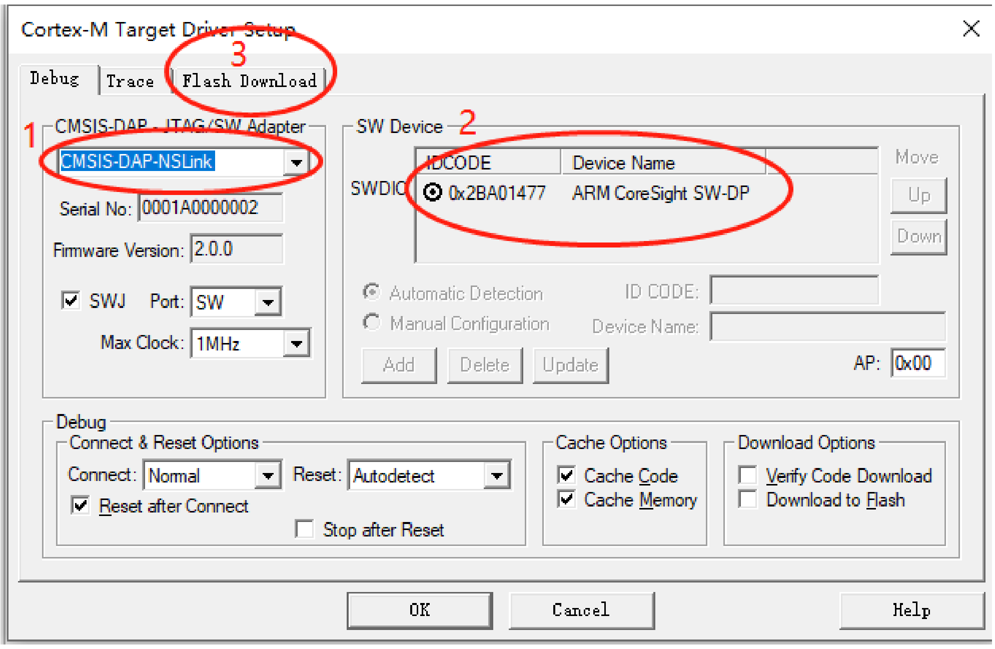
☑ Breakpoints

☑ Watch Windows

☑ Memory Display ☑ System Viewer

Driver DLL: SARMCM3.DLL Parameter:

Dialog DLL: TCM.DLL Parameter: -pCM4



CMSIS-DAP - JTAG/SW Adapter

CMSIS-DAP-NSLink

Serial No: 0001A0000002

Firmware Version: 2.0.0

☑ SWJ Port: SW

Max Clock: 1MHz

SW Device

	IDCODE	Device Name
SWDIO	● 0x2BA01477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:

○ Manual Configuration Device Name:

AP: 0x00

Debug

Connect & Reset Options

Connect: Normal Reset: Autodetect

☑ Reset after Connect

☐ Stop after Reset

Cache Options

☑ Cache Code

☑ Cache Memory

Download Options

☐ Verify Code Download

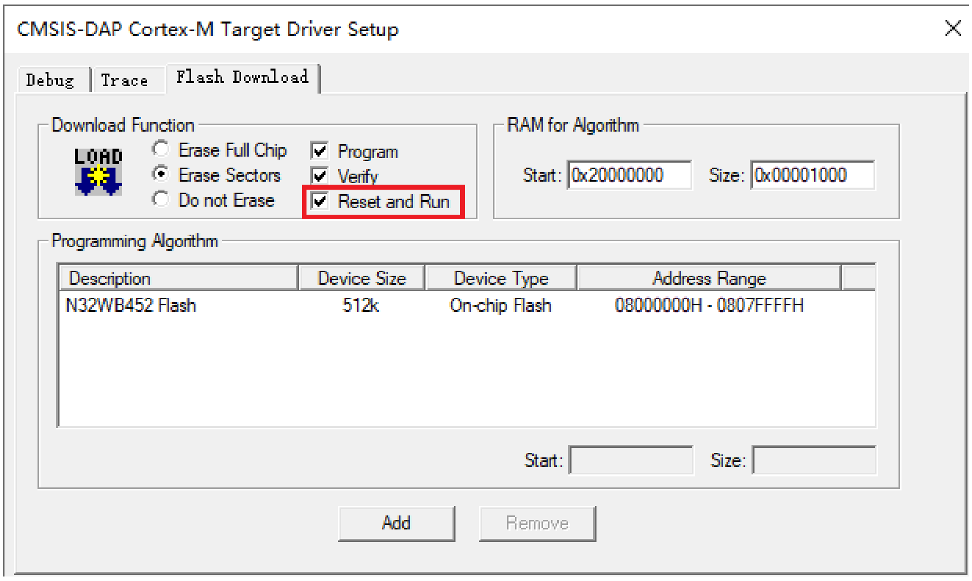
☐ Download to Flash

a) Select the NSLINK emulator (if connecting to other LINKs, select the corresponding LINK type).

b) If the connection is successful, the device serial number of the emulator can be displayed in position 2.

c) In the Flash Download interface, check the reset operation.

Click Flash Download settings, check "Reset and Run", the code will automatically reset and run after downloading.


CMSIS-DAP Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Download Function
- ○ Erase Full Chip
- ● Erase Sectors
- ○ Do not Erase
- ☑ Program
- ☑ Verify
- ☑ Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
N32WB452 Flash	512k	On-chip Flash	08000000H - 0807FFFFH

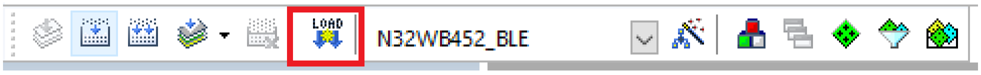
Start: Size:

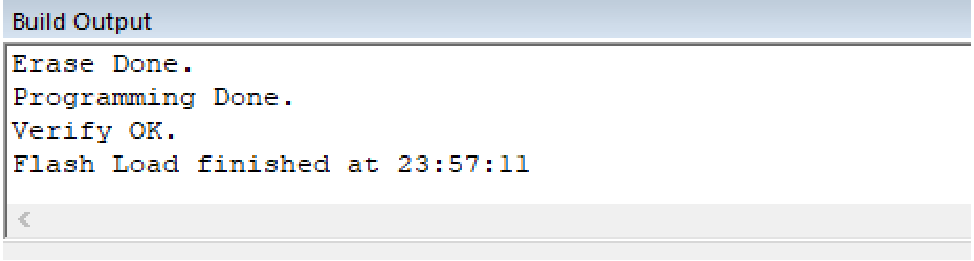
Add Remove

The compilation environment settings are completed here.

4.5 Download and simulation

4.5.1 Download firmware



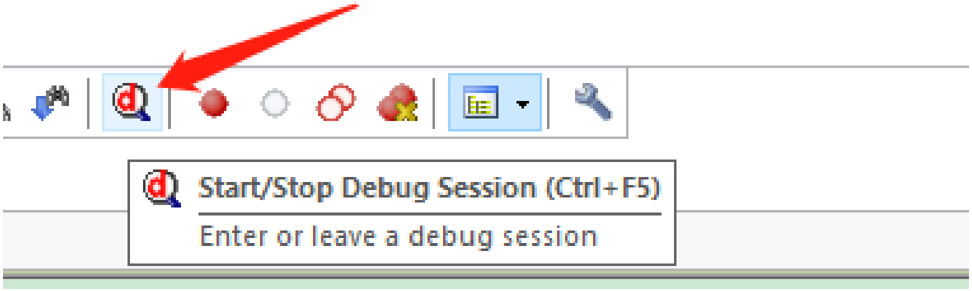

Build Output

```
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 23:57:11
```

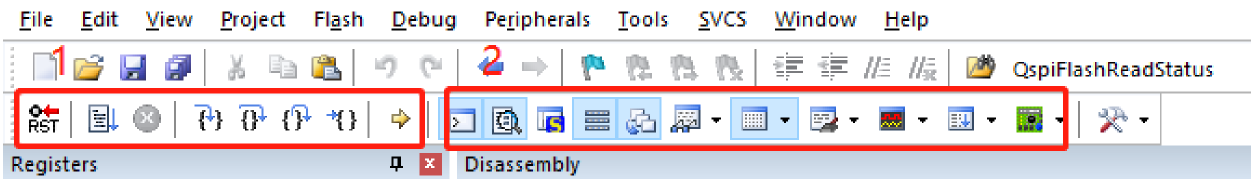
As shown in the figure above, the development board/debugger board is already running the program at this time.

4.5.2 Simulation debugging

When the emulator is connected and the program is compiled successfully, click the emulate button.


Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

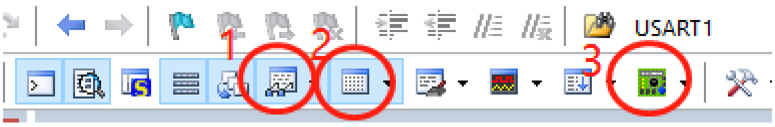
Enter the simulation interface as shown below, and the simulation tool button appears in the toolbar.



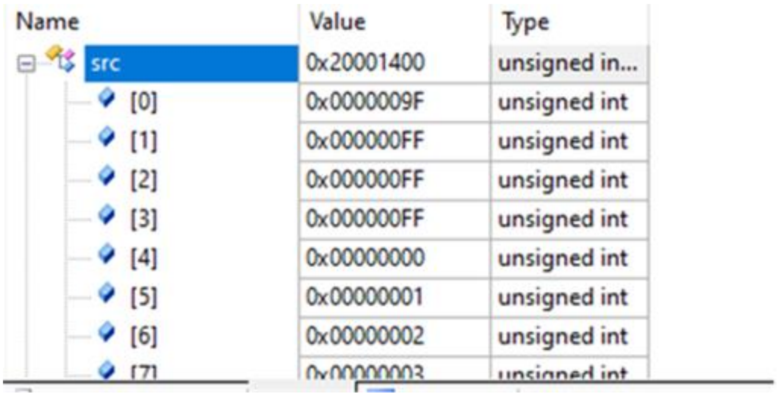
1) Execute simulation steps and control the execution process (single step, line by line, full speed, reset, etc.).

2) Monitor the simulation status, such as registers, variables, FLASH and other information.

KEIL common simulation tools

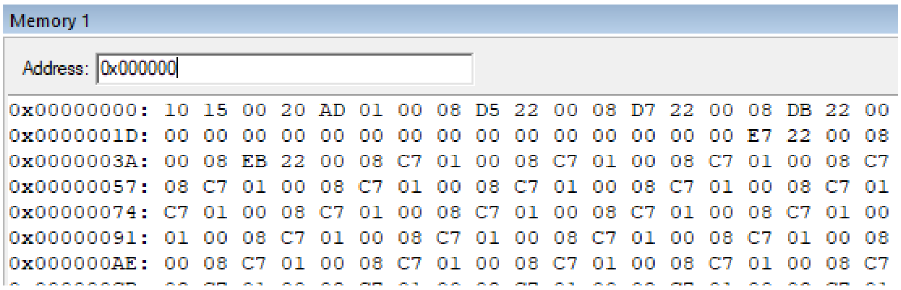


1) Click button 1 to open the variable view window



Name	Value	Type
src	0x20001400	unsigned in...
[0]	0x0000009F	unsigned int
[1]	0x000000FF	unsigned int
[2]	0x000000FF	unsigned int
[3]	0x000000FF	unsigned int
[4]	0x00000000	unsigned int
[5]	0x00000001	unsigned int
[6]	0x00000002	unsigned int
[7]	0x00000003	unsigned int

2) Click button 2 to open the memory view window

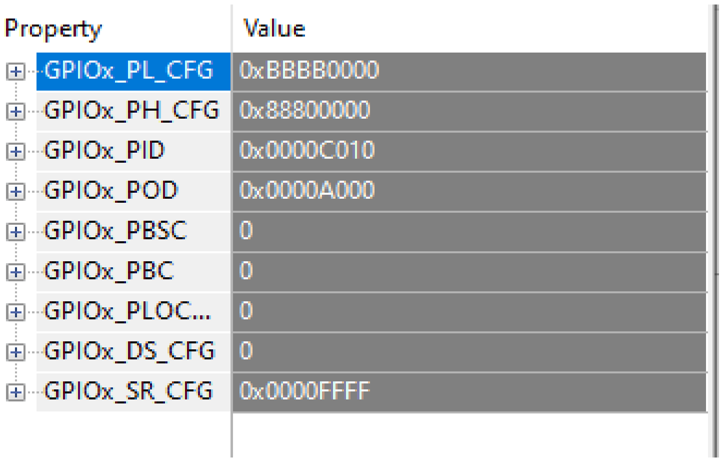


Memory 1

Address: 0x000000

```
0x00000000: 10 15 00 20 AD 01 00 08 D5 22 00 08 D7 22 00 08 DB 22 00
0x0000001D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E7 22 00 08
0x0000003A: 00 08 EB 22 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7
0x00000057: 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01
0x00000074: C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00
0x00000091: 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08
0x000000AE: 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7 01 00 08 C7
```

3) Click button 3 to open the register viewing window.



Property	Value
GPIOx_PL_CFG	0xBBBB0000
GPIOx_PH_CFG	0x88800000
GPIOx_PID	0x0000C010
GPIOx_POD	0x0000A000
GPIOx_PBSC	0
GPIOx_PBC	0
GPIOx_PLOC...	0
GPIOx_DS_CFG	0
GPIOx_SR_CFG	0x0000FFFF

Through the above methods, the running process and status of the program can be observed intuitively during the simulation process, and the development progress can be accelerated.

5. NS-LINK debugging tool

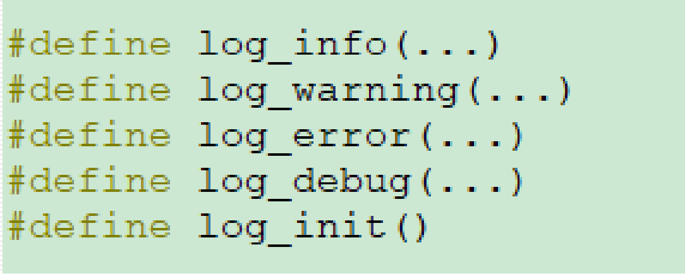
NS-LINK adopts ARM's standard software debugging access interface: CMSIS-DAP, which supports program download, emulation and serial port debugging functions. All Nations development boards integrate an NS-LINK circuit that supports SWD debugging.

5.1 Download/simulation interface

Users can easily download and debug programs with only one USB MINI cable. The SWD interface can be connected to the core MCU chip of the development board through a jumper cap, or it can be connected to other debug versions through a DuPont cable, which can be used as an independent LINK debugging tool.

5.2 Virtual serial port

NS-LINK supports the USB serial port function, and is connected to the chip USART2 through a jumper cap by default. Users can call the following output functions through the LOG file included in the BSP folder to output the serial port debugging information conveniently.



```
#define log_info(...)
#define log_warning(...)
#define log_error(...)
#define log_debug(...)
#define log_init()
```

5.3 Common interface problems

- **Connect to debug USB, without any device enumeration**

  It may be that the NS-LINK control chip has not been programmed with firmware, please contact the Nations technical support team to solve it.

- **The jumper cap is not properly connected, so the interface cannot work**

  Consult the development board documentation in the "EVT" folder and connect as required.

- **The serial port cannot print information**

  Possibility 1: The code uses USART2 for other purposes, so the LOG printing information cannot be output.

  Possibility 2: The serial port jumper cap is not properly connected.

  Possibility 3: The serial port communication is blocked, and the serial port assistant needs to be reopened.

- **After downloading the program, you cannot continue to download it again**

  Possibility 1: SWDIO/SWDCLK is used for other purposes in the code, causing the emulation interface to fail. In general, it is not recommended to use emulation pins for other functional designs

  Possibility 2: The chip enters the low-power power-down mode, and the emulation interface is invalid at this time, and it needs to be downloaded in the wake-up state.

6. **Version history**

Date	**Version**	**Remark**
2020-06-15	V1.0	Create documentation
2020-07-30	V1.1	Upgrade N32WB45xL-EVB development board to V1.1
2022-06-24	V1.2	1. Modified the wrong interface and jumper description of USB, debug interface, GPIO, SPI FLASH 2. Updated sample screenshots and related descriptions that do not match the current SDK 3. Modify the text file in Linker Scatter File Setting to .sct

7. NOTICE

This document is the exclusive property of Nations Technologies Inc. (Hereinafter referred to as NATIONS). This document, and the product of NATIONS described herein (Hereinafter referred to as the Product) are owned by NATIONS under the laws and treaties of the People's Republic of China and other applicable jurisdictions worldwide.

NATIONS does not grant 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.

NATIONS reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NATIONS 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 support or 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.