User Guide

N32H7xx series boot interface command user guide

Introduction

This user guide mainly describes the BOOT interface commands of the N32H7xx series MCUs, facilitating development and firmware downloading with the Nationz Technology BOOT Loader.

Content

1 Boot Overview

This user guide applies to the N32H7xx series chips and provides user download functionality, details are as follows:

1) Interface Support:

A. USART1 is supported; refer to Section 2.2.1 for the specific list of supported baud rates.

The list of physical interfaces is as follows:

Series/Model	USART-TX	USART-RX
N32H73x/N32H76x/N32H78x	PA9	PA10

After power-on, when the host computer communicates with the general-purpose MCU via the serial port, it first uses a baud rate of 9600 bps. The baud rate can be reconfigured using the CMD_SET_BR command, which takes effect upon receiving a success response. If the specified baud rate is not supported, a failure status will be returned.

2) Supports data or program downloading function;
3) Supports CRC32 verification for downloaded data;
4) Supports software reset operation of the chip;
5) Supports JTAG_KEY update;
6) Supports encrypted downloading;
7) Supports jumping to the user area for program execution;
8) Supports jumping to the SRAM area for program execution;
9) Supports configuration of the Option Bytes (OB);
10) Supports allocation of PFOER and Security areas;
11) Supports WRP configuration.

This document provides a detailed description of the functions, implementation and usage instructions of the BOOT module for general-purpose MCU chips.

2 BOOT Flow and Command Processing

The BOOT program supports downloading user programs and data via the USART interface. The relevant command processing flow is described below.

2.1 Commands and Data Structures

2.1.1 Command List

Table 2-1 Command definition

Command Name	Value	Description
CMD_SET_BR	0x01	Set the baud rate of the serial port (Valid only when serial ports are used)
CMD_GET_INF	0x10	Read MCU model index, BOOT version number, MCU ID
CMD_GET_RNG	0x20	Get Random Number
CMD_KEY_UPDATE	0x04	Update JTAG_KEY
CMD_FLASH_DWNLD	0x31	Download user programs to FLASH/SRAM
CMD_DATA_CRC_CHECK	0x32	CRC verification download user program
CMD_OPT_RW	0x40	Read/Configure Option Bytes (including read protection level, FLASH page write protection, PFOER, and Security)
CMD_OTP_OPT_RW	0x41	Read OTP configuration
CMD_SYS_RESET	0x50	The system reset
CMD_APP_GO	0x51	Jump to user area to execute the program

2.1.2 Data structure

This section describes some conventions described in the following sections. "<>" represents fields that must be included, and "()" represents the fields that included according to different parameters.

1. Upper and lower instruction data structure

1. Upper instruction structure:

<CMD_H + CMD_L + LEN + Par> + (DAT).

CMD_H indicates the level-1 command field, and CMD_L indicates the level-2 command field. LEN indicates the length of data to be sent.Par represents a four-byte command parameter; DAT represents the specific data sent from the upper level instruction to the lower level;

2. Lower response structure:

< CMD_H + CMD_L + LEN > + (DAT) + <CR1+CR2>.

CMD_H indicates the level-1 command field, and CMD_L indicates the level-2 command field. The command fields at the lower level are the same as those at the upper level. LEN indicates the length of data to be sent. DAT indicates the specific data that the lower layer replies to the upper layer. CR1+CR2 indicates the command execution result returned to the upper layer. If the level-1 and level-2 command fields do not belong to any command, BOOT replies CR1=0xBB and CR2 = 0xCC.

2. Command data structures supported by the serial port:

1. The host computer issues the upper instruction:

STA1 + STA2 + {Upper instruction structure} + XOR.

STA1 and STA2 are the start bytes of commands sent through the serial port. STA1=0xAA and STA2=0x55. Used for MCU identification upper computer to send serial data stream.

XOR represents the XOR operation value of the previous command byte (STA1 + STA2 + {Upper instruction structure}).

2. The upper computer receives the lower response:

STA1 + STA2 + {Lower response structure} + XOR.

STA1 and STA2 are the start bytes of commands sent through the serial port. STA1=0xAA and STA2=0x55. It is used for the host computer to identify MCU and send serial port data stream

XOR represents the XOR operation value of the previous command byte (STA1 + STA2 + {Lower response structure}).

2.2 Command description

2.2.1 CMD_ SET_BR

This command is used to change the baud rate of the serial port.

Upper instruction:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x01 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of send data: 0x00,0x00							
4~7(Par)	Par[0~3] : Set baud rate parameters							
(DAT)	None							

- Par[0~3], the serial port baud rate can be set to a typical value;

Table 2-2 Baud Rate Definition List

Par[0~3]	**Switch to specified baud rate (bps)**
0x000F4240	1000000
0x000E15C4	923076
0x0008CA00	576000
0x0003E800	256000
0x0001F400	128000
0x0001C200	115200
0x0000E100	57600
0x00009600	38400
0x00004B00	19200
0x00003840	14400
0x00002580	9600
0x000012C0	4800
0x00000960	2400

- Reserved value: 0x00;

Lower response:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x01 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of send data: 0x00,0x00							
(DAT)	None							
4(CR1)	Status byte 1							
5(CR2)	Status byte 2							

● Status bytes (CR1 and CR2) are divided into the following types according to command execution:

1. Return success: status flag bit (0xA0, 0x00).
2. Return failure: status flag bit (0xB0, 0x00).

● Serial Port Baud Rate Support for N32H7xx Series MCU BOOT V1.0

Table 2-3 Supported Baud Rate List

2400	**4800**	**9600**	**14400**	**19200**	**38400**	**57600**	**115200**	**128000**	**256000**	**576000**	**921600**	**923076**	**1M**
√	√	√	√	√	√	√	√	√	√	√	√	√	√

Note: "√" indicates supported, "/" indicates unsupported.

2.2.2 CMD_GET_INF

The function provided by this command is to read the BOOT version number, MCU model index, and MCU ID.

Upper instructions:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x10 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3 (LEN)	Length of send data: 0x00, 0x00							
4~7(Par)	Reserved, 0x00, 0x00, 0x00, 0x00							
(DAT)	None							

- LEN Send data length: 0x0000(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] +(LEN[1]<<8).

Lower response:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x10 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3 (LEN)	The length of the data 0x1D 0x00							
4~32(DAT)	BOOT version, MCU model index, MCU ID							
33(CR1)	Status byte 1							
34(CR2)	Status byte 2							

- The procedure byte (CMD_H) corresponds to the upper instruction (CMD_H).
- LEN is the data length: 0x1D(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- DAT[0] : 0x01, Chip Model Index

Product Number: 0x0A, N32H73x_76x_78x Series

- DAT[1] : 0xXY, BOOT command version (BCD code)

0x10: Indicates the command set version used by BOOT, representing the V1.0 command set version

- DAT[2] : BOOT code versionV1.0
- DAT[3~18] : 16Byte UCID (for example: 36 10 10 0C 0F 54 36 56 36 32 34 30 30 02 14 30).
- DAT[19-22] : 4Byte DBGMCU_IDCODE (UID) (for example: 59 5C 78 10).

  For the specific definitions of UCID/UID/DBGMCU_IDCODE, refer to 《UM_N32H7xx_Series_User_Manual_Vx.x.pdf》.
- DAT[23~28]: No meaning.
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

1. Return success: status flag bit (0xA0, 0x00).
2. Return failure: status flag bit (0xB0, 0x00).

2.2.3 CMD_KEY_RNG

Gets the random number of the key that the user needs to verify.

Upper-level instructions:

Bit Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x20 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of data sent:0x00, 0x00							
4~7(Par)	Reserved, 0x00, 0x00, 0x00, 0x00							
(DAT)	None							

- Par: 0x00000000;
- LEN Send data length: 0x00(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).

Lower layer response:

Bit Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x20 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of data sent							
4~19(DAT)	16Bytes pseudo-random number							

20(CR1)	Status byte 1
21(CR2)	Status byte 2

- LEN Send data length: 0x10(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- 16Byte pseudo-random number is generated by software algorithm.
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:
  1. Return success: status flag bit (0xA0, 0x00).
  2. Return failure: status flag bits (0xB0, 0x00).

2.2.4 CMD_KEY_UPDATE

Update JTAG_KEY.

Upper-level instructions:

Bit / Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x04 Level-1 Command field							
1(CMD_L)	0x00 Level-2 command field:							
2~3(LEN)	Length of data sent: 0x10 0x00							
4~7(Par)	Reserved value: 0x02 0x00 0x00 0x00							
8~23(DAT)	DAT[0~15] : 16Bytes old key authentication value							

- LEN Send data length: 0x10(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- Par: 0x00000002.
- DAT[0~15] : Encrypted data of JTAG_KEY.

Note:

The JTAG_KEY occupies 4 bytes, but the encryption algorithm requires 16 bytes. Therefore, the composition of the data value before encryption is as follows:

{Reserve[15:4], JTAG_KEY[3:0]}. The value is then encrypted using AES to obtain DAT[15:0].

Nonce and Key used for encryption:

Nonce = {0x61B5342B_E158D23F_7B6D61E1_7D466F06}

Key = {0x6DE52840_510EE9D2_1DE1CE61_753BD930}

Lower layer response:

Bit / Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x04 Level-1 Command field							

1(CMD_L)	0x00 Level-2 command field
2~3(LEN)	Length of data sent: 0x00 0x00
(DAT)	None
4(CR1)	Status byte 1
5(CR2)	Status byte 2

- LEN Send data length: 0x00(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:
  1. Return success: status flag bit (0xA0, 0x00).
  2. Return failure: status flag bits (0xB0, 0x00).

2.2.5 CMD_FLASH_DWNLD

This command allows users to download codes into the specified FLASH/SRAM. The download address must be 16-byte aligned, and the data length must also be 16-byte aligned (if insufficient, the host computer will automatically pad with 0xFF). Both the address and length are provided by upper-layer commands.

Upper-level instructions:

Bit / Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x31 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field: Download partition number							
2~3(LEN)	The length of the send data must be a multiple of 16 plus 4 bytes for CRC. The minimum length is 16 + 4 bytes of CRC = 20 bytes, and the maximum length is 128 + 4 bytes of CRC = 132 bytes							
4~7(Par)	Start address for downloading to FLASH/SRAM, 16-byte aligned.							
8~8+N+4(DAT)	DAT[0:N]: N+1 bytes of code data. DAT[N+1:N+4]: specifies the 4-byte CRC32 check value for unencrypted data.							

- CMD_L: 0
- LEN send data length: 0xXX(LEN[0]), 0xXX(LEN[1]), LEN = LEN[0] + (LEN[1]<<8)
- Par[3:0]: download the starting address of the FLASH/SRAM, synthetic rules to Address = Par[0] | Par[1]<<8 | Par[2]<<16 | Par[3]<<24.
- DAT[0~N] : download specific data, the total number of data is N+1

USART: up to 128 bytes, 15<=N<=127. N+1 must be a multiple of 16.

- DAT[N+1~N+4] : 4Byte CRC32 check value of unencrypted data

Lower layer response:

Bit Byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x31 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field: Download partition number							
2(LEN)	Length of data sent: 0x00 0x00							
(DAT)	None							
3(CR1)	Status byte 1							
4(CR2)	Status byte 2							

- LEN Send data length: 0x00(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

  1. Download success: status flag bit (0xA0, 0x00).
  2. Download failed: status flag bit (CR1, CR2).

     1) (0xB0, 0x21) : Incorrect data start address

     2) (0xB0, 0x20) : Incorrect data length.

     3) (0xB0, 0x10) : Data CRC32 verification error;

     4) (0xB0, 0x30) : Data write failure;

2.2.6 **CMD_DATA_CRC_CHECK**

This command is used to verify whether the downloaded data is correct. Considering the download speed and the relatively low probability of download failure, a unified CRC verification is performed after the data download is completed. The upper-layer command shall provide the CRC value of the downloaded data, as well as the verification start address and verification length.

When authentication is enabled, it is necessary to use CMD_KEY_RNG to obtain a random number and perform authentication before CRC verification.

Upper-level instructions:

Byte \ Bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x32 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field: Parity partition number							
2~3(LEN)	Length of data sent: 0x08 0x00							
4~7(Par)	32-bit CRC check value							
8~15(DAT)	DAT[0:3] : Verification start address DAT[4:7] : Verification data length (calculated in bytes)							

- CMD_L: 0x00
- LEN Send data length: 0x08(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8);
- Par[3:0] : 32 bit CRC check value, the synthetic rules for CRC32 = Par[0] | Par[1]<<8 | Par[2]<<16 | Par[3]<<24;
- DAT[3:0] : check the starting address, the synthesis rules to Address = DAT[0] | DAT[1]<<8 | DAT[2]<<16 | DAT[3]<<24, the Address is only within the scope of the FLASH/SRAM;
- DAT[7:4] : check length, its synthesis rules for CRC_LEN = DAT[4] | DAT[5]<<8 | DAT[6]<<16 | DAT[7]<<24, CRC_LEN must be within the valid range;

Lower layer response:

Byte \ Bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x32 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of data sent: 0x00 0x00							
(DAT)	None							

4(CR1)	Status byte 1
5(CR2)	Status byte 2

- LEN Send data length: 0x00(LEN[0]), 0x00(LEN[1]), LEN = LEN[0] + (LEN[1]<<8).
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

  1. Check succeeded: status flag bit (0xA0, 0x00).
  2. Check failure: status flag bits (CR1, CR2)

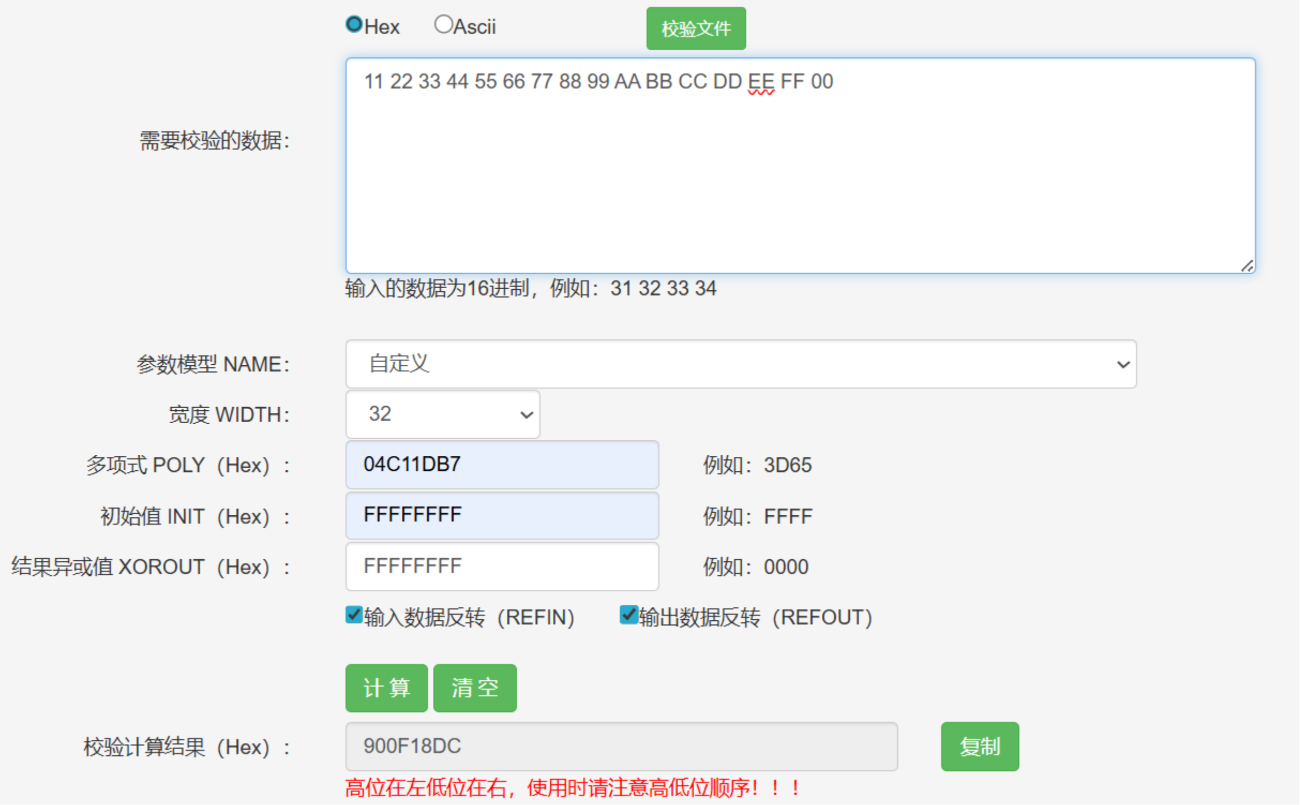
     1) (0xB0, 0x21) : Incorrect start address.

     2) (0xB0, 0x20) : Incorrect data length;

     3) (0xB0, 0x10) : CRC verification failure;

◆ **The CRC32 model is as follows:**

The input data does not need to be bit-reversed. For example, if you need to calculate the CRC32 value of 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF, 0x00, you only need to input the data into the following model in the corresponding order.



● Hex ○ Ascii

需要校验的数据： 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00

输入的数据为16进制，例如：31 32 33 34

参数模型 NAME： 自定义

宽度 WIDTH： 32

多项式 POLY（Hex）： 04C11DB7 例如：3D65

初始值 INIT（Hex）： FFFFFFFF 例如：FFFF

结果异或值 XOROUT（Hex）： FFFFFFFF 例如：0000

☑输入数据反转（REFIN） ☑输出数据反转（REFOUT）

校验计算结果（Hex）： 900F18DC

高位在左低位在右，使用时请注意高低位顺序！！！

◆ **The software implementation code is as follows:**

```
static unsigned long bitrev(unsigned long input, int bw){
    int i;
    unsigned long var;
    var = 0;
    for (i = 0; i < bw; i++){
        if (input & 0x01){
            var |= 1 << (bw - 1 - i);
        }
        input >>= 1;
    }
    return var;
}
unsigned long GetCRC32_Software(unsigned char* buf, unsigned long Length, unsigned long initcrc){
    unsigned int i, j = 0;
    unsigned char* data = buf;
    unsigned long crc = initcrc;
    while ((Length--) != 0){
        data[j] = bitrev(data[j],8);  /*The input inversion */
        crc ^= (unsigned long)data[j] << 24;
        j++;
        for (i = 0; i < 8; ++i){
            if ((crc & 0x80000000) != 0){
                crc = (crc << 1) ^ CRC32_DFE_POLY;  //0x04C11DB7
            }
            else{
                crc <<= 1;
            }
        }
    }
```

2.2.7 CMD_OPT_RW

This command is used for reading and writing option bytes (including read protection level, FLASH page write protection, Data0/1 configuration, and USER configuration).

Upper instructions:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x40 Level-1 command field							
1(CMD_L)	Level-2 command field							
2~3 (LEN)	Length of send data; refer to the description below for the command format							
4~7(Par)	Parameters；refer to the description below for the command format							
8~n (DAT)	Configurations;refer to the description below for the command format							

- CMD_L:
  1. 0x00: Read option bytes
  2. 0x01: Configure option bytes
- LEN: Length of send data; (LEN[0])、0x00(LEN[1])，LEN = LEN[0] + (LEN[1]<<8)
- Par[3:0] : refer to the description below for the command format
- DAT : refer to the description below for the command format

Lower response:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x40 Level-1 command field							
1(CMD_L)	Level-2 command field							
2~3 (LEN)	The length of the data, refer to the description below for the command format							
4~7(DAT)	OTP data							
8(CR1)	Status byte 1							

9(CR2)	Status byte 2

- CMD_L：
  1. 0x00: Read option bytes
  2. 0x01: Configure option bytes
- LEN：The lengths of send data (LEN[0])、0x00(LEN[1])，LEN = LEN[0] + (LEN[1]<<8), refer to the description below for the command format
- DAT : refer to the description below for the command format
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

1． Check succeeded: status flag bit (0xA0, 0x00).

2． Check failure: status flag bits (CR1, CR2)

1) (0xB0, 0x00): Failure returned.

2.2.7.1 **RDP**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0**	**CR1/CR2**
Read	0x40	0x00	0x0000	0x00000001	-	-
Read_Rsp	0x40	0x00	0x0001	-	RDP= 0/1/2;	0xA000/0xB000
Write	0x40	0x01	0x0001	0x00000001	RDP= 0/1/2;	-
Write_Rsp	0x40	0x01	0x0001	-	RDP= 0/1/2;	0xA000/0xB000

RDP: 0 denotes Level 0 (L0) protection, 1 denotes Level 1 (L1) protection, and 2 denotes Level 2 (L2) protection.

2.2.7.2 **WRP**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0 D1 D2 D3**	**CR1/CR2**
Read	0x40	0x00	0x0000	0x00000002	-	-
Read_Rsp	0x40	0x00	0x0004	-	WRP = 0x_D3_D2_D1_D0	0xA000/0xB000

Write	0x40	0x01	0x0004	0x00000002	WRP = 0x_D3_D2_D1_D0	-
Write_Rsp	0x40	0x01	0x0004	-	WRP = 0x_D3_D2_D1_D0	0xA000/0xB000

WRP: Corresponding write-protected sectors, 32 bits in total. Each bit represents 128 KB. BIT0 is meaningless. BIT1 represents the address range 0x15000000 ~ 0x1501FFFF, and BIT31 represents the address range 0x153C0000 ~ 0x153DFFFF.

2.2.7.3 **BOOTADDR_M7**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0 D1 D2 D3**	**CR1/CR2**
Read	0x40	0x00	0x0000	0x00000003	-	-
Read_Rsp	0x40	0x00	0x0004	-	M7 Addr = 0x_D3_D2_D1_D0	0xA000/0xB000
Write	0x40	0x01	0x0004	0x00000003	M7 Addr = 0x_D3_D2_D1_D0	-
Write_Rsp	0x40	0x01	0x0004	-	M7 Addr = 0x_D3_D2_D1_D0	0xA000/0xB000

M7 Addr: Boot address of the M7 core

2.2.7.4 **BOOTADDR_M4**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0**	**CR1/CR2**
Read	0x40	0x00	0x0000	0x00000004	-	-
Read_Rsp	0x40	0x00	0x0004	-	M4 Addr = 0x_D3_D2_D1_D0	0xA000/0xB000

Write	0x40	0x01	0x0004	0x00000004	M4 Addr = 0x_D3_D2_D1_D0	-
Write_Rsp	0x40	0x01	0x0004	-	M4 Addr = 0x_D3_D2_D1_D0	0xA000/0xB000

M4 Addr: Boot address of the M4 core

2.2.7.5 **SEC_CFG**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0 D1 D2 D3**	**CR1/CR2**
Cfg Sec	0x40	01	0x0004	0x00000105	Cfg = 0x_D3_D2_D1_D0	-
DME Del Sec	0x40	01	0x0004	0x01000105	Cfg = 0x_D3_D2_D1_D0	-
ME Del Sec	0x40	01	0x0004	0x02000105	Cfg = 0x_D3_D2_D1_D0	-
Read	0x40	00	0x0000	0x00000105	-	-
Cfg Sec Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
DME Del Sec Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
ME Del Sec Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
Read Rsp	0x40	00	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000

Cfg Sec: Configures the Security region

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMES	Reserved		SEC_END[12:0]												
rw									rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	SEC_BEGIN[12:0]
	rw

Writing 1 to the DMES bit enables the function of deleting the security region. DME Del Sec deletes the security region by means of read protection downgrading, while ME Del Sec erases the security region of the Flash via Mass Erase. Res[30:29] and Res[15:13] are reserved bits. The read data format is Read = 0x_D3_D2_D1_D0.

Start address of the Flash security region = 0x15000000 + SEC_BEGIN[12:0] * 4K;

End address of the Flash security region = 0x15000000 + 4K + SEC_END[12:0] * 4K - 1;

Note: The minimum size of the security user region is 8KB.

2.2.7.6 **PFOER_CFG**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0 D1 D2 D3**	**CR1/CR2**
Cfg Pfoer	0x40	01	0x0004	0x00000106	Cfg = 0x_D3_D2_D1_D0	-
DME Del Pfoer	0x40	01	0x0004	0x01000106	Cfg = 0x_D3_D2_D1_D0	-
ME Del Pfoer	0x40	01	0x0004	0x02000106	Cfg = 0x_D3_D2_D1_D0	-
Read	0x40	00	0x0000	0x00000106	-	-
Cfg Pfoer Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
DME Del Pfoer Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
ME Del Pfoer Rsp	0x40	01	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000
Read Rsp	0x40	00	0x0004	-	Cfg = 0x_D3_D2_D1_D0	0xA000/0xB000

Cfg Pfoer: Configures the Pfoer region

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMEP	Reserved		PFOER_END[12:0]												
rw			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PFOER_BEGIN[12:0]												
			rw												

Writing 1 to the DMEP bit enables the function of deleting the FFOER region. DME Del Pfoer deletes the FFOER region by means of read protection downgrading, while ME Del Pfoer erases the FFOER region of the Flash via Mass Erase. Res[30:29] and Res[15:13] are reserved bits. The read data format is Read = 0x_D3_D2_D1_D0.

Start address of the Flash FFOER region = 0x15000000 + FFOER_BEGIN[12:0] * 4K;

End address of the Flash FFOER region = 0x15000000 + 4K + FFOER_END[12:0] * 4K - 1;

Note: The minimum size of the FFOER region is 8KB.

2.2.7.7 **SEC_MODE**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0**	**CR1/CR2**
Read	0x40	0x00	0x0000	0x00000007	-	-
Read_Rsp	0x40	0x00	0x0002	-	DAT = 0x_D1_D0	0xE5CE/Others
Write	0x40	0x01	0x0002	0x00000007	DAT = 0x_ D1_D0	-
Write_Rsp	0x40	0x01	0x0002	-	DAT = 0x_D1_D0	0xE5CE/Others

DAT Parameters: 0xE5CE = Enabled; Others = Disabled

Note: Configuration of the security region can only be performed after the security mode is enabled.

2.2.7.8 **CRYPTION**

Command	**Cmd_H**	**Cmd_L**	**Len**	**Para**	**D0**	**CR1/CR2**
Read	0x40	00	0x0000	0x00000008	-	-
Read_Rsp	0x40	0x00	0x0002	-	DAT= 0x_D1_D0	0xE00B/0Xebeb / Others
Write	0x40	01	0x0002	0x00000008	DAT= 0x_D1_D0	-
Write_Rsp	0x40	0x01	0x0002	-	DAT= 0x_D1_D0	0xE00B/0xEBEB / Others

DAT Parameter:

0xE00B: Low-level encryption (unbound encryption);

0xEBEB: High-level encryption (bound encryption);

Others: No encryption.

2.2.8 **CMD_EXT_EXT_OTP_RW**

This command is used for option byte reading and writing. It includes configuring the boot mode lock function (OTP_SYS_CFG_BTM), BOR threshold (OTP_SYS_CFG_BORLS), and hardware watchdog (OTP_SYS_CFG_IWD).

For the configuration of the boot mode lock function, please refer to the documentation specified in 《CN_UG_N32H7xx_Series_Software_Development_User_Guide_x.x.pdf》.

Upper instructions:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x41 Level-1 command field							
1(CMD_L)	Level-2 command field							
2~3 (LEN)	Length of send data: 0x00, 0x00							

4~7(Par)	Parameters
8~n (DAT)	Configurations

- CMD_L:
  1. 0x00: Read option bytes
  2. 0x01: Configure option bytes
- LEN:
  1. When reading OTP, no DAT field is required, and LEN = 0x00 0x00
  2. When writing OTP, LEN = 0x04 0x00, where LEN = LEN[0] | (LEN[1] << 8)
- Par[0~3]:
  1. 0x02: OTP_SYS_CFG_BTM
  2. 0x03: OTP_SYS_CFG_BORLS
  3. 0x04: OTP_SYS_CFG_IWDG
- DAT: Data field for reading or writing option bytes
  1. Configure boot mode lock (OTP_SYS_CFG_BTM); Default value: 0xFFFFFFFF

Locked Mode Status	**Description**	**DAT[0]**	**DAT[1]**	**DAT[2]**	**DAT[3]**
Enable	Code boots only from FLASH	0xA5	0x5A	0x5A	0xA5
	Code boots only from BOOT	0x84	0x48	0x48	0x84
Disable	The code boot mode depends on the BOOT pin level	Others	Others	Others	Others

  2. Set BOR level (OTP_SYS_CFG_BORLS): Default value = 0xFFFF0002

NO	**Rising**	**Falling**	**DAT[0]**	**DAT[1]**	**DAT[2]**	**DAT[3]**
0	2.2V	2.1V	0x02	0x00	0xFD	0xFF
1	2.5V	2.4V	0x03	0x00	0xFC	0xFF
2	2.8V	2.7V	0x04	0x00	0xFB	0xFF
3	3.1V	3.0V	0x05	0x00	0xFA	0xFF
4	3.45V	3.35V	0x06	0x00	0xF9	0xFF

3. Configure watchdog and NRST options (OTP_SYS_CFG_IWDG): Default value = 0xC0003FFF;

Data Field	**Bit Field Description**
DAT3	~nDAT[1]
DAT2	~nDAT[0]
DAT1	Bit14~Bit15：2b’00 Bit13：nRST_STOP_C_M7 Bit12：nRST_STOP_C_M4 Bit11：nRST_STANDBY_C_M7 Bit10：nRST_STANDBY_C_M4 Bit9：IWDG_SW_M7 Bit8：IWDG_SW_M4
DAT0	Bit7：IWDG_STOP0_M7 Bit6：IWDG_STOP0_M4 Bit5：IWDG_STOP2_M7 Bit4：IWDG_STOP2_M4 Bit3：IWDG_STANDBY_M7 Bit2：IWDG_STANDBY_M4 Bit1：IWDG_SLEEP_M7 Bit0：IWDG_SLEEP_M4

Lower response:

bit / byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x41 Level-1 command field							
1(CMD_L)	Level-2 command field							
2~3 (LEN)	The length of the data 0x04 0x00							
4~7(DAT)	OTP data							
8(CR1)	Status byte 1							
9(CR2)	Status byte 2							

- CMD_L：

1. 0x00: Read option bytes
2. 0x01: Configure option bytes

- LEN：LEN=0x04 0x00，LEN = LEN[0] | (LEN[1]<<8)
- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

1. Return success: status flag bit (0xA0, 0x00).
2. Return failure: status flag bit (0xB0, 0x00).

2.2.9 **CMD_SYS_RESET**

This command is used to reset the BOOT program.

Upper instruction:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x50 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of send data：0x00, 0x00							
4~7(Par)	Reserved							
(DAT)	None							

- Reserved value: 0x00;

Lower response:

bit byte	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x50 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of send data：0x00, 0x00							
(DAT)	None							
4(CR1)	Status byte 1							

5(CR2)	Status byte 2

- Status bytes (CR1 and CR2) are divided into the following types according to command execution:
    1. Return success: status flag bit (0xA0, 0x00).
    2. Return failure: status flag bit (0xB0, 0x00).

2.2.10CMD_APP_GO

This command is used to jump to and execute the application program after the BOOT loader has downloaded the application program to the internal Flash/RAM.

Upper instruction:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x51 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3(LEN)	Length of send data: 0x00，0x00							
4~7(Par)	Jump Address							
(DAT)	None							

- Par: Jump Address, supporting Flash/AHB SRAM/TCM/AXI SRAM

Lower response:

byte \ bit	b7	b6	b5	b4	b3	b2	b1	b0
0(CMD_H)	0x51 Level-1 command field							
1(CMD_L)	0x00 Level-2 command field							
2~3 (LEN)	Length of send data: 0x00，0x00							
(DAT)	None							
4(CR1)	Status byte 1							
5(CR2)	Status byte 2							

- Status bytes (CR1 and CR2) are divided into the following types according to command execution:

1. Return success: status flag bit (0xA0, 0x00).
2. Return failure: status flag bit (0xB0, 0x00).

2.3 Returns the status word description

2.3.1 Returns the success status word

Return success: status flag bit (0xA0, 0x00). Indicates that the command issued by the upper layer is executed successfully, and returns a success status word.

Contains the success return value of the read, update, configure, and other commands.

2.3.2 Returns the failure status word

Return failure: status flag bits (0xB0, 0x00). Indicates that the command issued by the upper layer fails to execute due to other reasons (command acceptance format error or timeout, etc.), and the failure status word is returned.

2.3.3 Return other status words

The following return status words also return failure. The second byte status word indicates a different error type.

1) (0xB0, 0x10) : CRC check error;

2) (0xB0, 0x20) : Length error

3) (0xB0, 0x21) : Address error.

4) (0xB0, 0x30) : Download failure

3 BOOT User Guide

3.1 Host Computer Control Flow

The host computer supports user code download and integrity verification of the downloaded code. By reading the configuration information, the host computer automatically identifies the address range for erasing, downloading and verification entered by the user.

The host computer supports users to update the JTAG_KEY.

The host computer supports users to read and modify option bytes.

The host computer supports software reset commands.

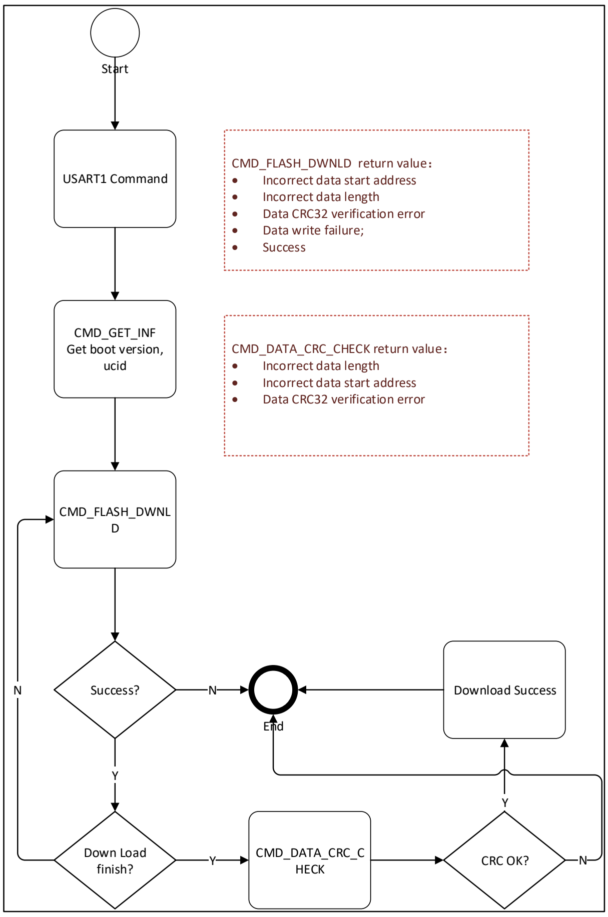
Enter BOOT: Enter the BOOT mode, where interaction with the PC TOOL is available via the USART1 interface.

Command Set Interaction: The PC TOOL sends different commands according to the command set supported by BOOT to invoke corresponding functions:

1. Read the BOOT version number, chip model index and chip ID;
2. Obtain 16-byte true random numbers;
3. Update the JTAG_KEY;
4. Download the user program to FLASH;
5. Download the user program to RAM;
6. Perform CRC verification on the downloaded user program;
7. Read/configure option bytes (including read protection level, FLASH page write protection, PFOER, SEC);
8. System reset, which can reset and re-run the BOOT program;
9. Jump to the target area (FLASH/SRAM) to execute the user code.

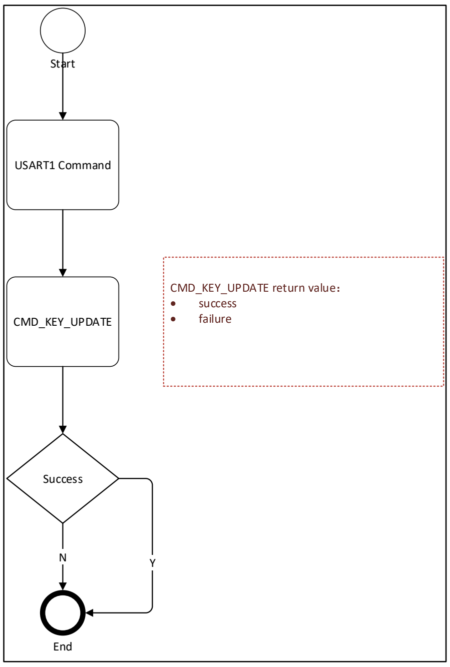
3.1.1 Download Command Control Flow Chart

Figure 3-1 Download Command Control Flow Chart



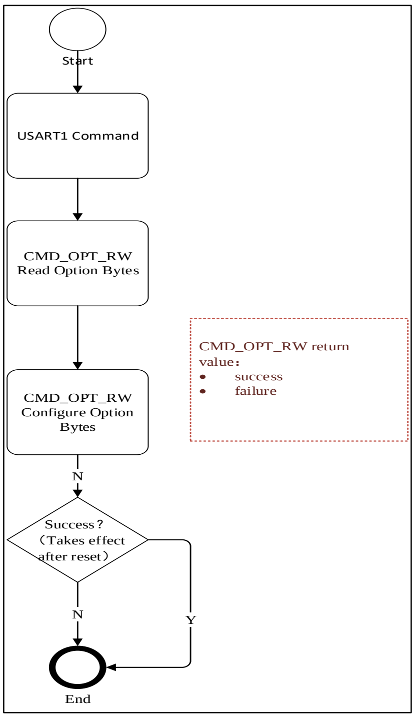
3.1.2 JTAG Key Update Command Control Flow Chart

Figure 3-2 JTAG Key Update Command Control Flow Chart



3.1.3 Option Bytes Read/Write Command Control Flow Chart

Figure 3-3 Option Bytes Read/Write Command Control Flow Chart



3.2 BOOT Usage Notes

1) Most configurations take effect only after a reset.

4 History versions

Version	Date	Notes
V1.0.0	2025.12.5	Initial version

5 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 NSING Technologies Inc. and NSING 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 NSING has attempted to provide accurate and reliable information, NSING 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 NSING 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. NSING 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 NSING and hold NSING 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 NSING, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.