

# BC45B4523 NFC Reader Development Board Application Note

D/N: AN0573EN

## Introduction

The BC45B4523 is an NFC reader controller supporting a transmission frequency of 13.56MHz. It is compatible with the ISO14443A, ISO14443B, ISO15693 protocols and supports the Crypto\_M encryption and decryption functions. The ISO14443A/B protocols provide four bit rate options, which are 106Kbps, 212Kbps, 424Kbps and 848Kbps. Multiple protocols are able to work with multiple types of NFC tags, making the device an excellent solution for short distance secure transmission.

The BC45B4523 provides a maximum RF output current of 250mA, allowing for an increased sensing distance and also supports a low power automatic card detection mode. The integrated 3.3V LDO, which has a maximum output current of 150mA, can provide power for the master MCU.

This text will show how to use the BC45B4523 along with the BC45B4523 NFC reader development board. Together with the provided program library, this will assist users to get started quickly in using this device.

## Functional Description

This application example includes PC software which is connected to the development board by means of a USB virtual COM. Command procedures, UID read and data access to various NFC tags can be implemented using this software. The software also includes an advanced mode, where users are able to complete the required command procedures for some special NFC tags by sending commands.

A jumper is located on the development board to select the power source for the master MCU and NFC reader transmitter, allowing users to set up different environments.

### Features

- NFC Reader Development Board
  - ◆ Connects with the PC using a USB virtual COM port
  - ◆ UART transmission baud rate = 115200bps
  - ◆ Supports BC45B4523 3.3V and 5V (VUSB) voltage selection
  - ◆ MCU HT32F52241 powered by the BC45B4523's integrated 3.3V LDO or an external 3.3V LDO

- ◆ Provides a buzzer to give an indication sound after finishing tag read/write commands
- ◆ Supports ISO14443A, ISO14443B, ISO15693 and Crypto-M cards
- ◆ Supports hardware card detection function
- ◆ Application sample cards

ISO14443A Type 2: NXP NTAG213 A card, tag IC: NTAG213

ISO14443B Type 3: ST SRT512 B card, tag IC: ST25TB512

ISO15693 Type 5: NXP SL2D2 V card, tag IC: SL2S2002\_SL2S2102

Mifare Classes 1K card: tag IC MF1ICS50

Mifare Classes 4K card: tag IC MF1S70YYX

- NFC Reader Library
  - ◆ Supports ISO14443-1, ISO14443-2, ISO14443-3 protocols
  - ◆ Supports ISO14443A card UID read and Type 2 tag read/write
  - ◆ Supports ISO14443B card UID read
  - ◆ Supports Crypto-M card UID read and card read/write
  - ◆ Supports ISO15693 card UID read and card read/write
  - ◆ Professional mode can be used to support customised commands

## Hardware Description

### Circuit Diagram

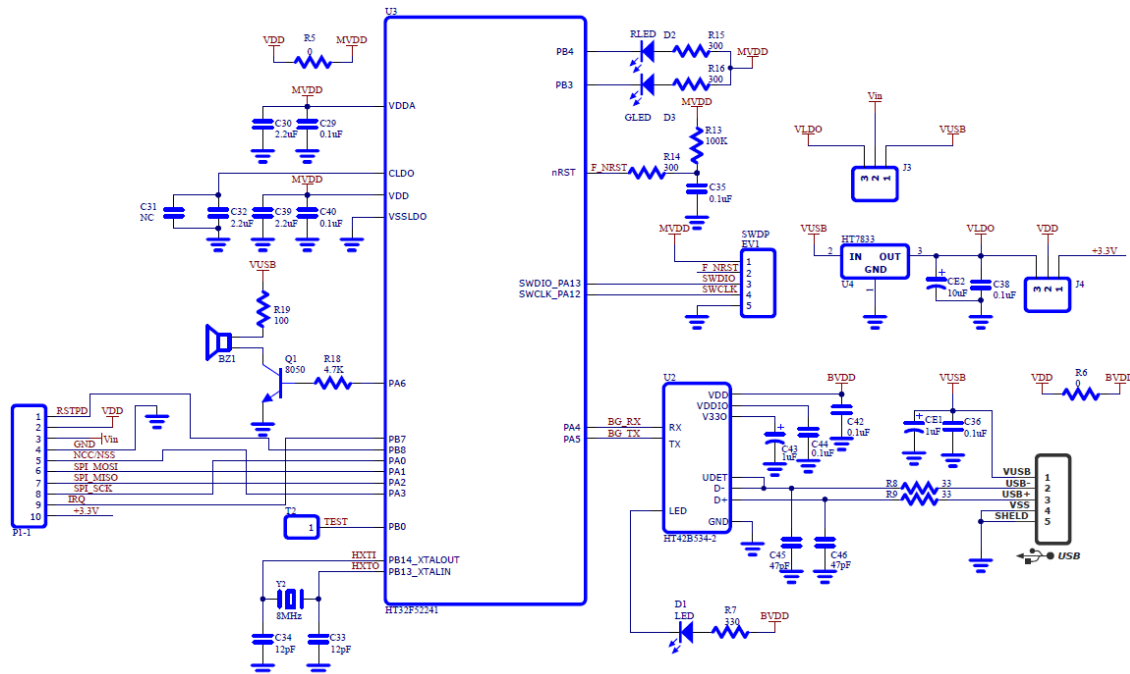


Figure 1

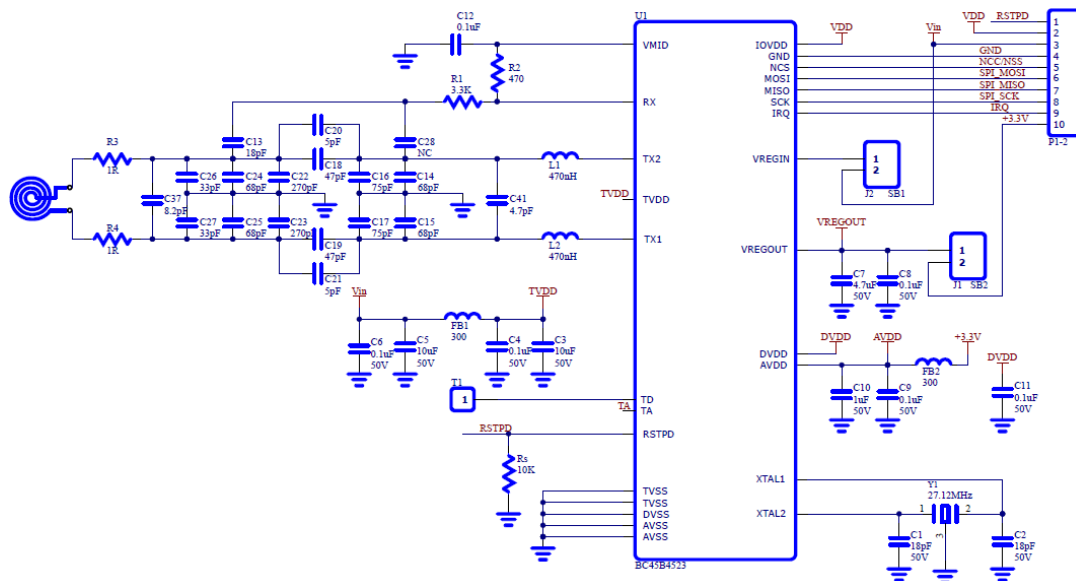


Figure 2

## Circuit Block Diagram

The hardware block diagram is shown in Figure 3. This application example uses a USB interface to connect to the PC. A USB-to-UART bridge IC is used to bridge the master MCU HT32F52241 and USB interface. The HT32F52241 uses an SPI interface to control the BC45B4523 NFC reader. The BC45B4523 circuit includes an RF matching circuit and an NFC antenna.

In addition, the PCB board includes an HT7833 LDO. Users can select either the BC45B4523's internal 3.3V LDO or the HT7833 as the power source for the HT32F52241.

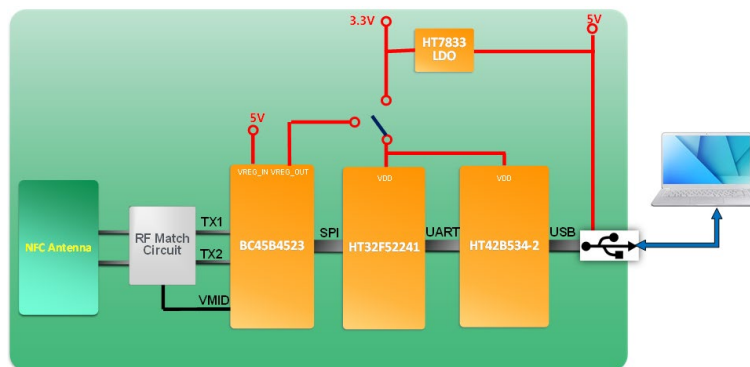


Figure 3. Hardware Block Diagram

## Circuit Description

This section will focus on the BC45B4523 circuit description. Refer to the relevant datasheet for the HT32F52241 and HT42B534-2 application circuits.

- Control Interface and Integrated LDO

As shown in Figure 4, the BC45B4523 uses its SPI interface as the control interface and has an IRQ pin for its interrupt output. It also has an integrated 3.3V LDO with a maximum current output of 150mA, which can be provided for use by external circuits such as the master MCU.

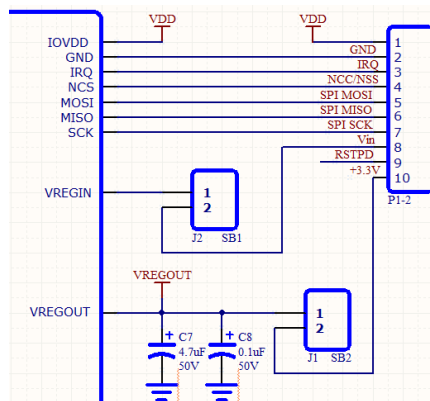


Figure 4. SPI Control Interface

- TX Matching Circuit

The BC45B4523 NFC TX matching circuit is shown below where TX1 and TX2 are the RF differential output pins providing an output frequency of 13.56MHz. Other matching circuit components are described in the following table.

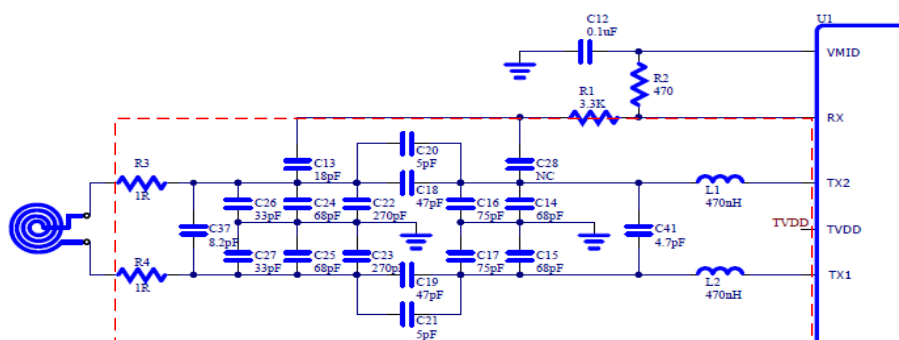


Figure 5. TX Matching Circuit

Components	Main Purpose
L1, L2, C14, C15, C16, C17, C41	EMC filtering
C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C37	Impedance matching
R3, R4	Antenna Q adjustment

Table 1. Component Main Purposes

Figure 6 shows the TX matching circuit measurement method. Connect the SMA connector to the TX1/TX2 terminal, then observe the Smith Chart using a network analyser and adjust the characteristic impedance. The recommended value is  $25\Omega \pm 10\%$ .

The characteristic impedance of this example is  $23.4 + j0.1\Omega$ , as shown in the Smith Chart in Figure 7.

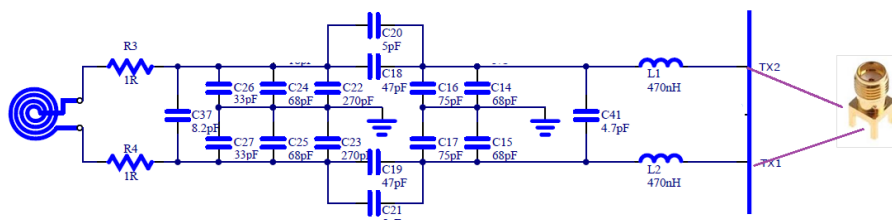


Figure 6. TX Matching Circuit Measurement

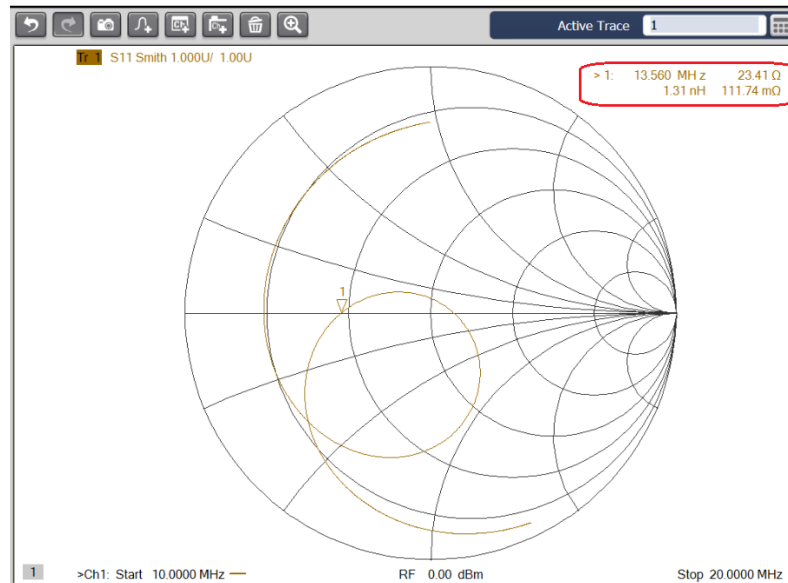


Figure 7. Smith Chart

#### ● Antenna Q Value Adjustment

The antenna's Q value is related to bandwidth. The antenna impedance characteristic measurement is shown in Figure 8. Connect the SMA connector to both terminals of the antenna, measure the resistance (R) and inductance (L) characteristics of the antenna using a network analyser, and adjust the Q value through R3/R4 in the circuit. The recommended Q value for the ISO14443 standard is in a range of 10~30.

Figure 9 shows the antenna characteristics of this example, which is about 588mΩ and 568nH.

$$\begin{aligned}
 Q_{(ant+R3+R4)} &= \omega \times L_{ant} / R_{(ant+R3+R4)} \\
 &= 2 \times \pi \times f_c \times L_{ant} / R_{(ant+R3+R4)} \\
 &= 2 \times \pi \times 13.56M \times 568n / 2.6 \\
 &\doteq 18.6
 \end{aligned}$$

$$\begin{aligned}
 BW &= f_c / Q_{(ant+R3+R4)} \\
 &= 725K
 \end{aligned}$$



Figure 8. Antenna Impedance Characteristic Measurement

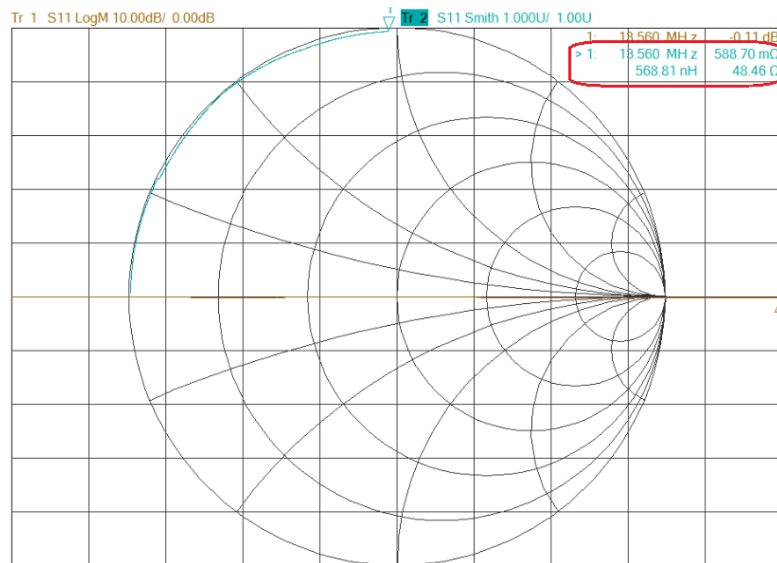


Figure 9. Antenna Characteristics

#### ● RX Receiver Circuit

As shown in the BC45B4523 NFC RX matching circuit below, VMID is a 1.65V DC reference voltage, C12 is a filter capacitor for VMID and RX is an RF input signal receiving pin. Either C13, which is close to the antenna terminal, or C28 which is close to the transmitting terminal, together with R1 and R2 can implement a carrier frequency divider, which is used to send the received signal to the RX pin.

After the RxAutoPD buffer is disabled, the RX input signal should be in a range of 2.5~2.9Vp-p. The calculation formula is as follows:

$$V_{RX} = (V_{TX} \times R2) / (R1 + R2)$$

Here  $V_{TX}$  indicates the voltage of C13 or C28 at the TX matching circuit terminal.

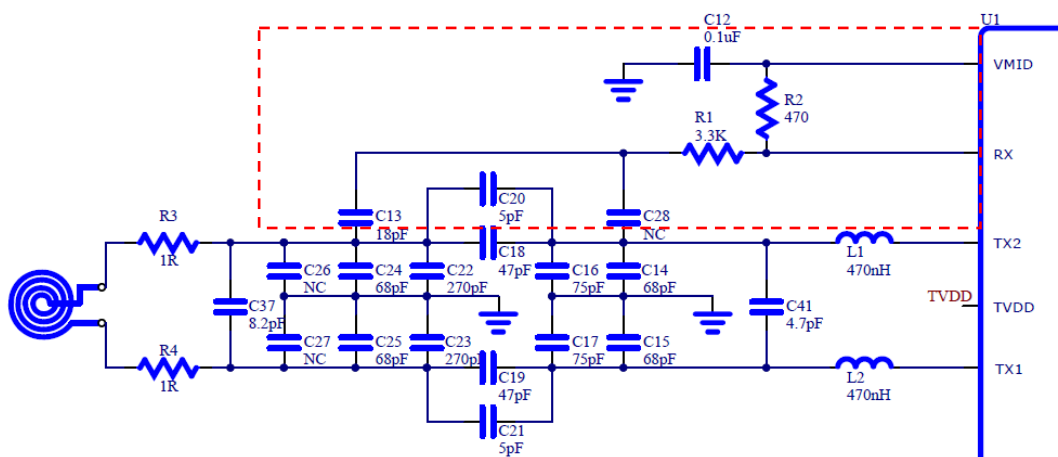


Figure 10. RX Matching Circuit

In this example, the VMID pin voltage is 1.60V and RX Vp-p is 2.76V, as shown in Figure 11.

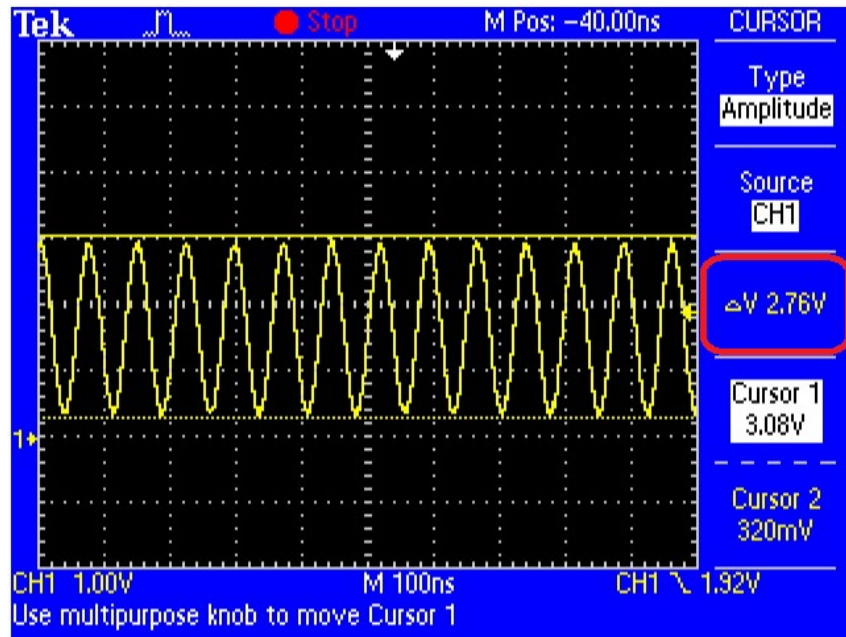


Figure 11. RX Waveform

- RX Layout Considerations

The RX layout routing and components should not be located close to the crystal oscillator (Y1). As shown in Figure 12 below, Y1 should be surrounded by ground surfaces to reduce interference.

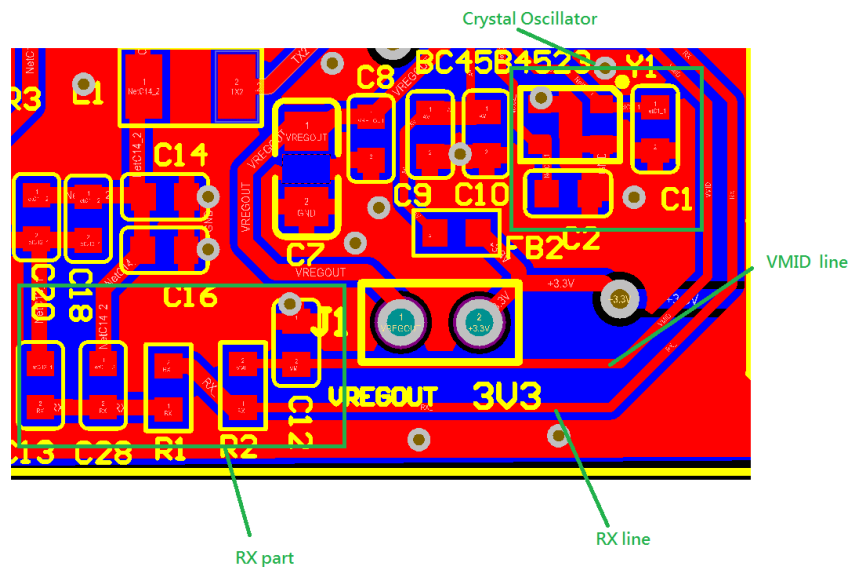
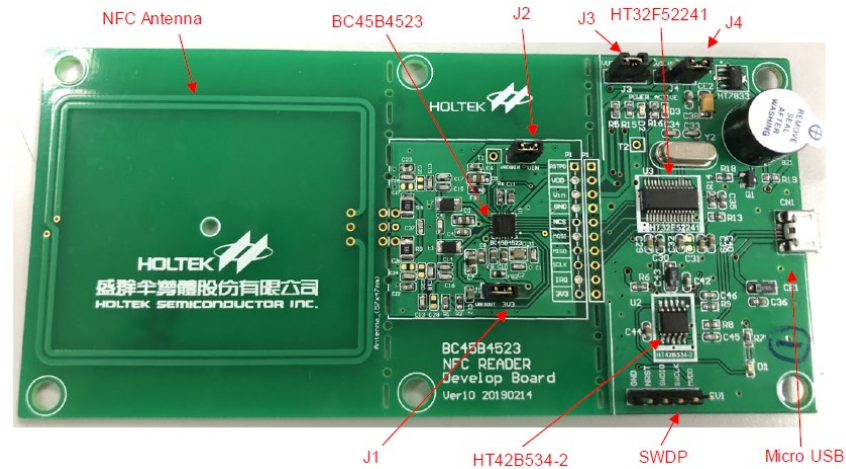


Figure 12. RX Layout

## PCB Appearance and Hardware Functional Description



**Figure 13**

- **Operating Description**
  - (1) Connect to the PC USB port via a Micro USB connector.
  - (2) Use the SWDP interface to edit and program the firmware.
  - (3) Place the tag above the antenna for tag sensing.
  - (4) No metal objects should be located nearby during the tag sensing and card detection calibration.
- **Jumper Description**
  - (1) J1 and J2 should remain shorted.
  - (2) J3 is used to select the power source for the BC45B4523 TVDD (RF transmitter power): 3.3V—provided by LDO (HT7833); 5V—provided by VUSB.
  - (3) J4 is used to select the power source for the 3.3V VDD (MCU VDD / bridge IC VDD): LDO (HT7833) 3.3V output or the BC45B4523's integrated 3.3V LDO output.
- **BC45B4523 Card Detection Current Description**
  - (1) Use J3 to select the power source for the BC45B4523 TVDD (RF transmitter power): HT7833.
  - (2) Use J4 to select the power source for the VDD (MCU VDD / bridge IC VDD): HT7833 3.3V output.
  - (3) **Current Measurement**
    - a) Short connect the BC45B4523's VDD (VDDIO) and VIN, then connect to the HT7833 output. Other signals are connected normally as shown in Figure 14.



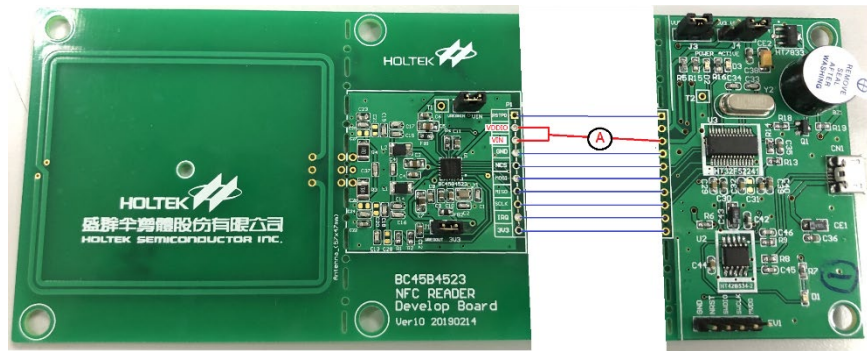


Figure 14

- b) Card detection mode current measurement: select the tag detection mode on the software's Advanced page–Sleep & 500ms.
- c) Use an oscilloscope to examine the antenna transmitter waveform (Figure 15). The RF transmits a signal once every 500ms for about 12.5 $\mu$ s (Figure 15-(B)).
- d) For the stage shown in Figure 15-(A), it is not possible to use the oscilloscope current mode to make measurements due to its small magnitude, measured only in microamperes. A ammeter is used instead to obtain the current value, about 5 $\mu$ A.
- e) For the stage shown in Figure 15-(B), it is possible to use the oscilloscope current mode to make measurements. As shown in Figure 16, there are two intervals of 12.5 $\mu$ s every 1s with an average current of about 54mA for each interval. Averaging the current in these two intervals for 1s gives about 1.35 $\mu$ A.
- f) Adding the step (d) and (e) values gives about 6.35 $\mu$ A.

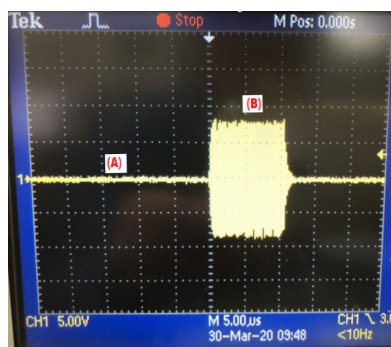


Figure 15

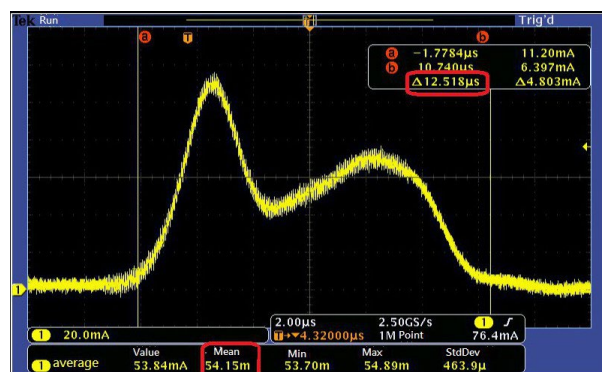


Figure 16

## Software and Program Description

This application example provides the relevant PC software which controls the BC45B4523 NFC reader to implement read and write card operations. The HT32 firmware and library are also provided to assist with rapid user development.

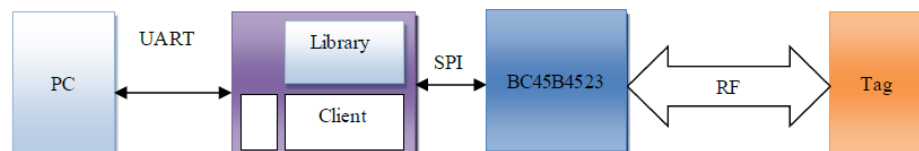
### Software Description

Download the HT32 series library from the Holtek official website: HT32F5xxxx (M0+) Standard Peripheral Firmware Library.

After decompression, create an NFC directory in the Application directory. Decompress the example project and place it in this directory. The project is developed using the Keil uVision 5.

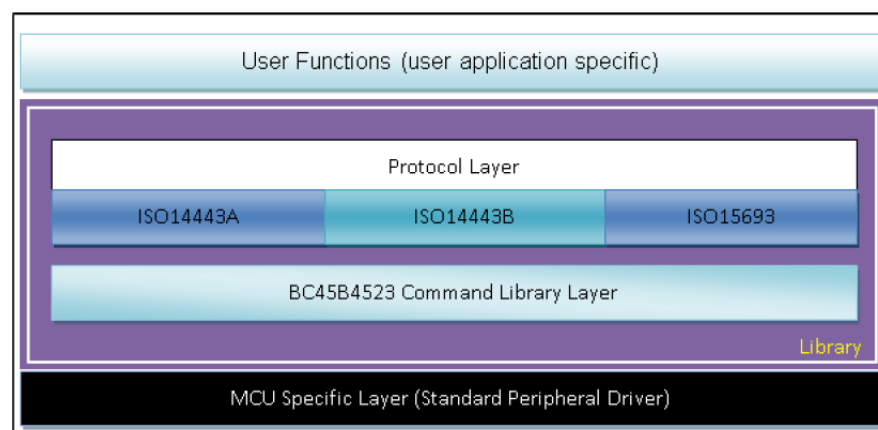
### Architecture Description

The program application block diagram is as follows.



The BC45B4523 NFC reader demo board is composed of a master MCU – HT32F52241, a UART bridge IC – HT42B534-2, and an NFC reader – BC45B4523. After receiving UART commands from the PC, the HT32F52241 controls the BC45B4523 using its SPI interface to transmit ISO14443 and IO15693 protocol radio frequencies. By using this UART solution, users do not need to develop interface software but only need to use general serial port software or a built-in terminal within Windows to send commands to the NFC reader.

The structure of the function library is as follows.

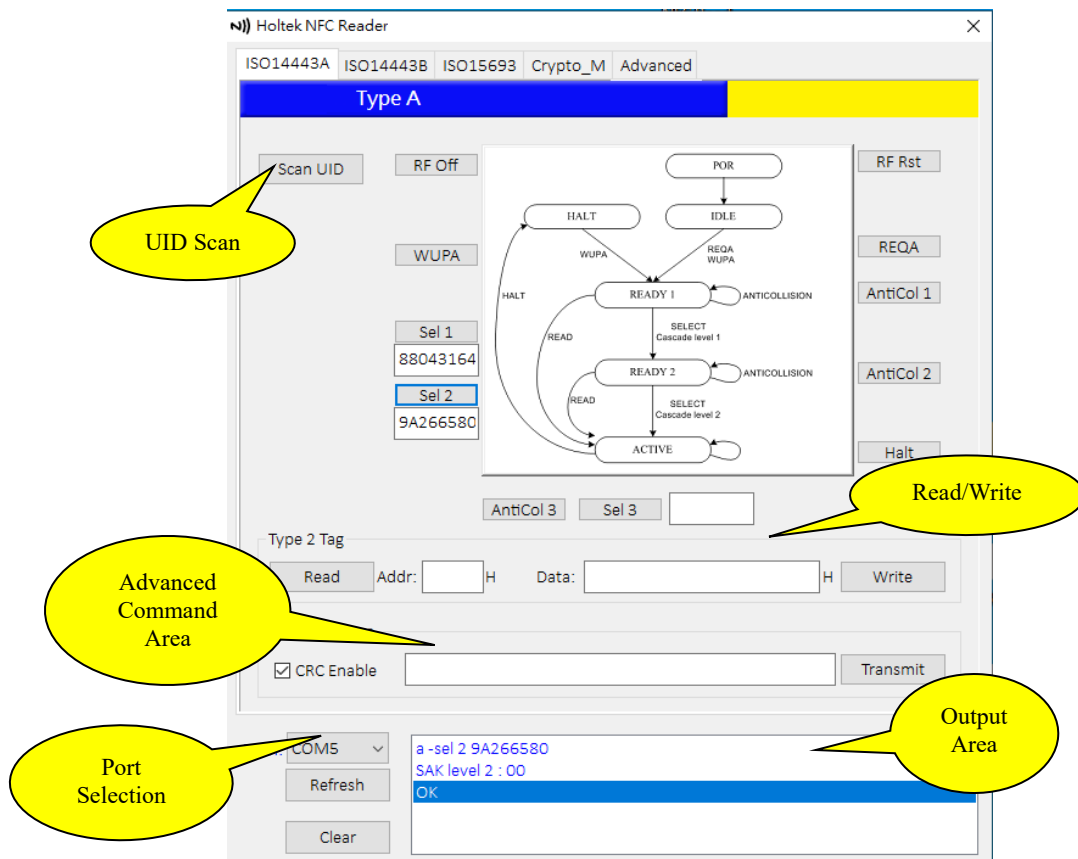


- User Functions
  - (1) IO/SPI/UART/PWM settings
  - (2) UART/Reader command processing
  - (3) UID scanning process

- Protocol Layer
  - (1) Crypto\_M encryption and decryption process
  - (2) ISO14443A RF settings and related commands
  - (3) ISO14443B RF settings and related commands
  - (4) ISO15693 RF settings and related commands
- Command Library Layer
  - (1) BC45B4523 register read/write
  - (2) BC45B4523 FIFO read/write
  - (3) BC45B4523 protocol related functional settings
- Standard Peripheral Driver
  - (1) SPI transmitting/receiving functions
  - (2) UART transmitting/receiving functions
  - (3) System clock settings

## Software Instructions

After the program starts, it will automatically open the first COM port found. If it is not the correct one, reselect another COM port.



- ISO14443A (NFC-A)

(1) UID scan is used to obtain the tag UIDs.

(2) The flowchart on the right shows the decomposition process to help users understand the UID scan actions.

a) Click [RF Rst] to reset RF → click [REQA] then the output area will display the ATQA returned by tag → click [AntiCol 1] then the output area will display the UID Level 1 code returned by tag → click [Sel 1] → click [AntiCol 2] then the output area will display the UID Level 2 code returned by tag → click [Sel 2]. For a 7-byte UID, steps Sel 1 and 2 are sufficient.

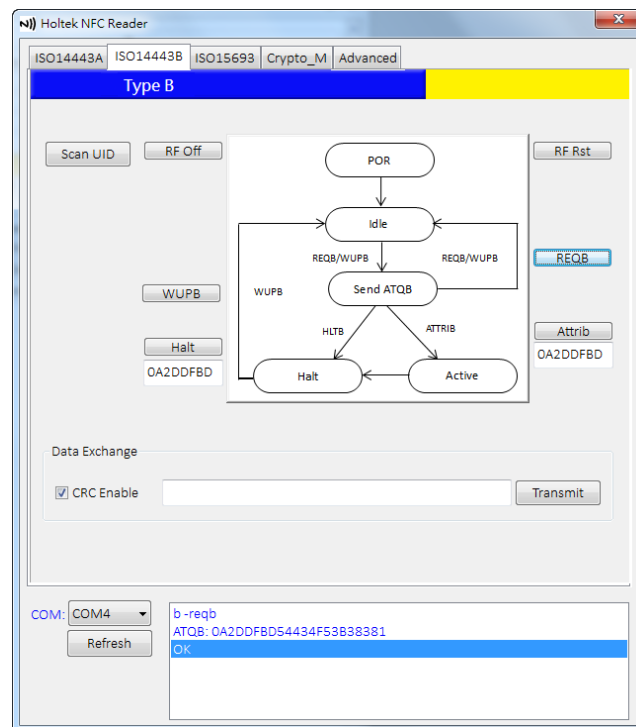
b) For a 10-byte UID, one more [AntiCol 3] step is required to obtain the UID Level 3 code.

(3) After execution to the ACTIVE state, the reader can enter the HALT state by clicking [Halt] and then enter the READY state by clicking [WUPA].

(4) The read/write function presently only supports Type 2 tags with simple protocols.

(5) For other tags with more operation procedures or customised commands, users can use the Data Exchange area to fill in commands and data to implement the required access functions.

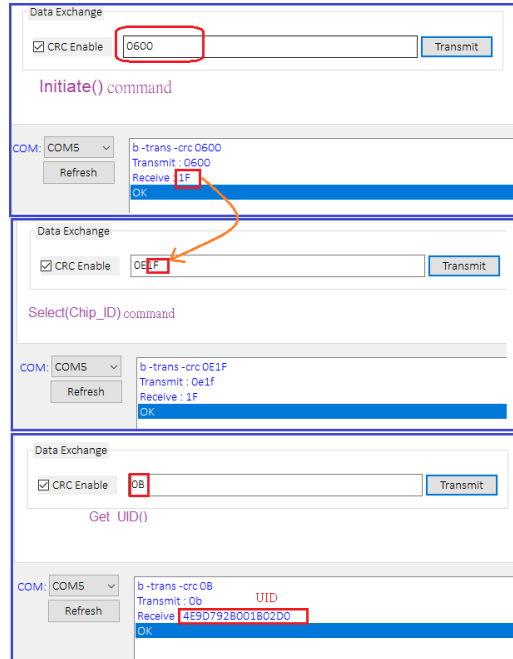
(6) All the execution results will be displayed in the output area.



- ISO14443B (NFC-B)

- (1) The screen function is the same as ISO14443A.
- (2) Read and write operations of Type B cards are not yet supported. Users should execute commands via the Data Exchange function to implement these operations.
- (3) The ST25TB512 is a non-standard Type4 B card in which the UID read and read/write operations are as follows. For detailed commands and procedures, refer to the ST25TB512 datasheet.

a) UID Read Example



The figure illustrates the three-step process for reading a UID from an ISO14443B card using the Data Exchange interface.

**Step 1: Initiate() command**  
 - Command: `Initiate()`  
 - CRC: 0600  
 - Transmit: b-trans -crc 0600  
 - Receive: 1F  
 - Status: OK

**Step 2: Select(Chip\_ID) command**  
 - Command: `Select(Chip_ID)`  
 - CRC: 0E1F  
 - Transmit: b-trans -crc 0E1F  
 - Receive: 1F  
 - Status: OK

**Step 3: Get UID() command**  
 - Command: `Get UID()`  
 - CRC: 08  
 - Transmit: b-trans -crc 08  
 - Receive: 4E9D7928001802D0 (UID)  
 - Status: OK

## b) User Area Read/Write Example

☒ CRC Enable

Initiate Command()

COM: COM5
 

Refresh
 

b-trans -crc 0600  
 Transmit : 0600  
 Receive : A6  
 OK

Data Exchange
 

☒ CRC Enable

Random Number

Select Command(0E)+Random Number(A6)

COM: COM5
 

Refresh
 

b-trans -crc 0EA6  
 Transmit : 0ea6  
 Receive : A6  
 OK

Data Exchange
 

☒ CRC Enable

Read Area Command(08)+Address(07)

COM: COM5
 

Refresh
 

b-trans -crc 0807  
 Transmit : 0807  
 Receive : FFFFFFFF  
 OK

Data result

Data Exchange
 

☒ CRC Enable

Write Area Command(09)+Address(07)+Data(4bytes)

COM: COM5
 

Refresh
 

b-trans -crc 090701234567  
 No Response  
 ERROR

No return data so Error report but it is write correctly

Data Exchange
 

☒ CRC Enable

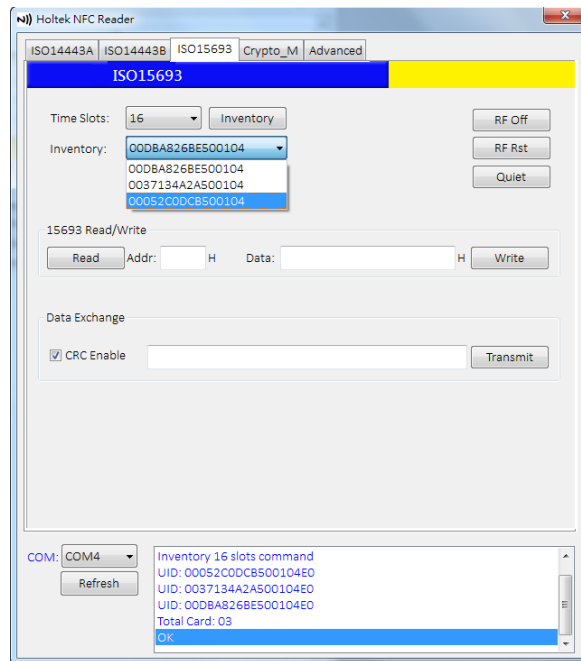
Read Area Command(08)+Address(07)

COM: COM5
 

Refresh
 

b-trans -crc 0807  
 Transmit : 0807  
 Receive : 01234567  
 OK

Data result



## ● ISO15693(NFC-V)

(1) UID(Inventory) scan function: Time Slots=1 and Time Slots=16 two settings.

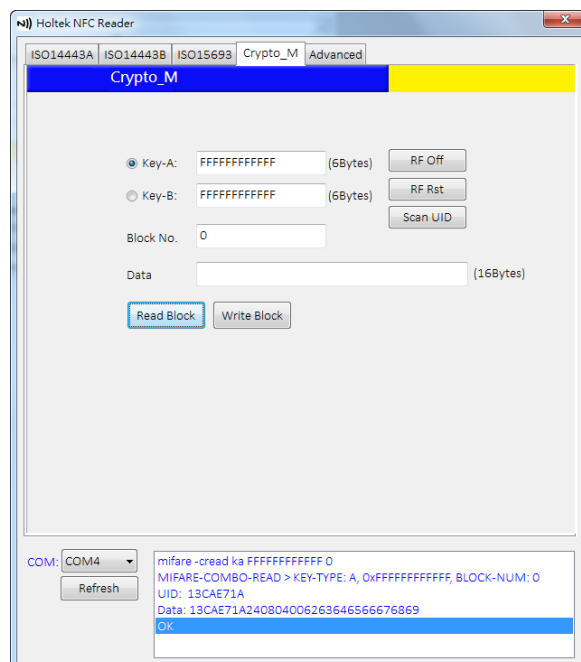
When Time Slots=1, only one tag can be scanned at the same time.

When Time Slots=16, up to 16 tags can be scanned at the same time.

Click [Inventory], now the UIDs will be obtained according to the Time Slots setting and displayed in the Inventory pull-down menu.

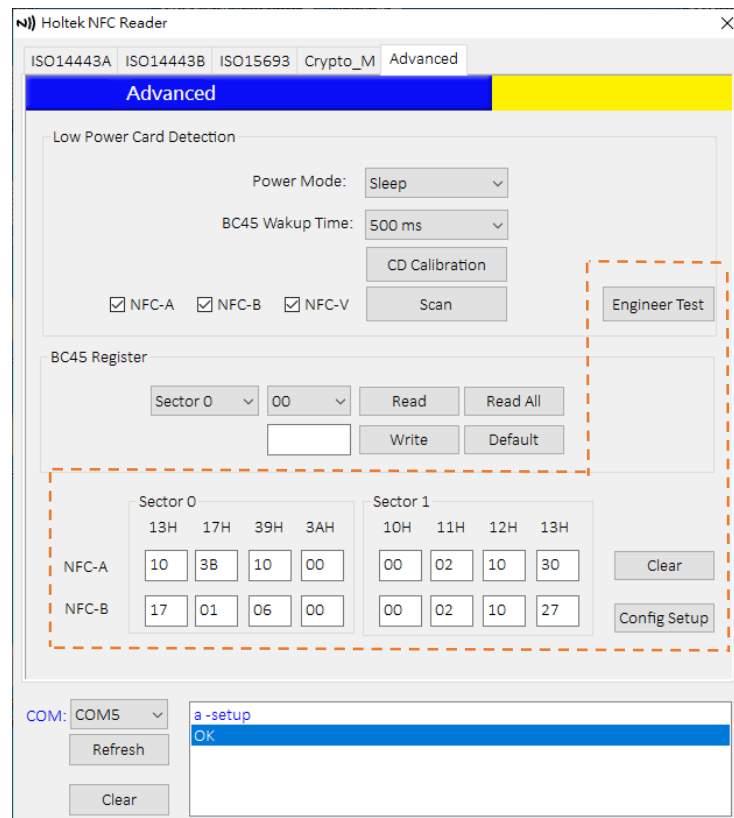
(2) Click [Read]/[Write] button to access the selected inventory tag.

(3) For other tags with customised commands, users should issue relevant commands via the Data Exchange function.



- Crypto\_M

- (1) The Crypto\_M function provided by Holtek is compatible with Mifare tags.
- (2) When implementing read/write operations to Crypto\_M tags, Key-A or Key-B must be provided.
- (3) Different blocks can be set with different permissions, refer to the Mifare tag specification for the permission setting details.



※ The dotted frame area is for engineering testing, it must not be selected or modified.

- Advanced

- (1) Card Detection Mode

Holtek's NFC reader demo board provides three power modes: Deep Sleep1, Sleep and Normal, which refer to the status that the master MCU maintains when a tag has not yet been detected. The task of regularly detecting tags is distributed to the BC45B4523, greatly reducing the power consumption. The BC45B4523 will wake up the master MCU only after a tag is detected.

- a) Before enabling the card detection function of the BC45B4523, click [CD Calibration] to implement RF calibration without placing any tag above the reader.
- b) Configure the master MCU's power mode and the BC45B4523's wake-up time.
- c) After setting the tag type to be detected, click the [Scan] button to enter the tag scan mode.
- d) To change the card detection wake-up time, stop the ongoing operation first and then select the time required.



## (2) BC45B4523 Register Access Function

This is an advanced function which is used to read and write the internal registers of the BC45B4523. Refer to the BC45B4523 datasheet for the functional description of the registers.

[Read]	Read the value of a single register
[Read All]	Read all the registers
[Write]	Write into a single register
[Default]	Write default values into all the registers

## UART Command Description

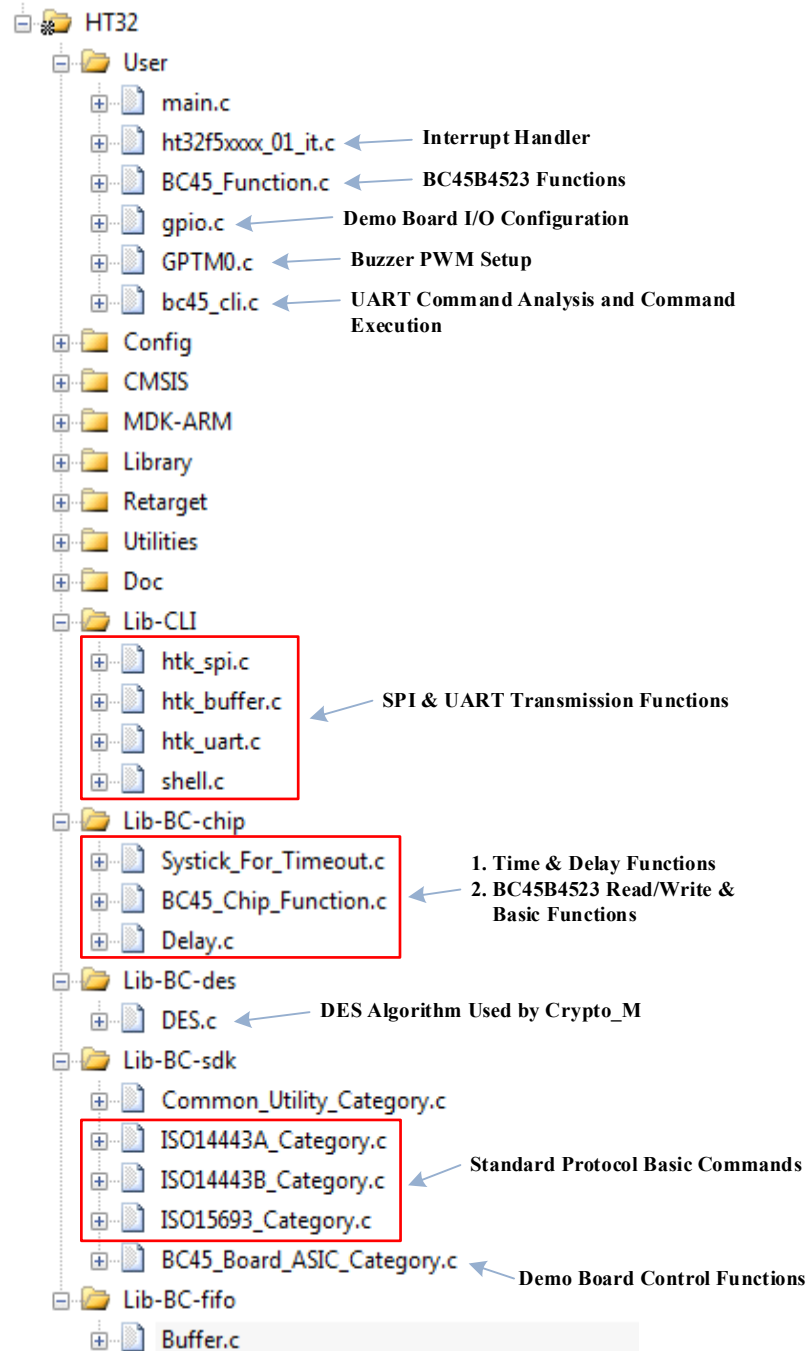
The software provided by Holtek communicates with the NFC reader via UART commands, which are described below.

Command	Parameter	Description
reg	-rd [-s0/-s1] <addr> -rd -all -wr [-s0/-s1] <addr> <data> -wr -default	Read addr (address) in sector 0/1 Read all addresses Write data into addr in sector 0/1 Write default values into all addresses
rf	-off -reset	Turn off field Restart field
cd	-cal -lpmcu <deep sleep1/sleep/normal> -wkuptime <100/200/500/1000>	RF calibration Set master MCU's power mode Set BC45B4523 card detection interval
Scan	-start <n>	1. Start card detection according to card type 2. n=0x01: NFC-A n=0x02: NFC-B n=0x04: NFC-V n=0x03: NFC-A and NFC-B ... and so on 3. Enter any byte via UART to end detection
a	-setup -reqa -wupa -halt -anticol <1/2/3> -sel <1/2/3> <id> -getuid -trans <-crc/-nocrc> <data>	Set to ISO14443A information interface Execute ISO14443A reqa command Execute ISO14443A wupa command Execute ISO14443A halt command Execute ISO14443A anti collision command Execute ISO14443A select command Read UID Send data with (without) checksum
b	-setup -reqb -wupb -halt -attrib -getuid -trans <-crc/-nocrc> <data>	Set to ISO14443B information interface Execute ISO14443B reqb command Execute ISO14443B wupb command Execute ISO14443B halt command Execute ISO14443B attrib command Read UID Send data with (without) checksum
v	-setup -inv1 -inv16 -quiet <uid> -rd <-rd> <addr> <uid> -wr <-wr> <addr> <data> <uid> -trans <-crc/-nocrc> <data>	Set to ISO15693 information interface TimeSlots=1 to get inventory TimeSlots=16 to get inventory Issue quiet command to a specific tag Read from the addr address of a tag Write data to the addr address of a tag Send data with (without) checksum
mifare	-cread <ka/kb> <key> <block> -cwrite <ka/kb> <key> <block> <data>	Read from block with ka=key or kb=key Write data to block with ka=key or kb=key
t2t	-rd <addr> -wr <addr> <data>	Read from the addr address of a Type 2 tag Write data to the addr address of a Type 2 tag

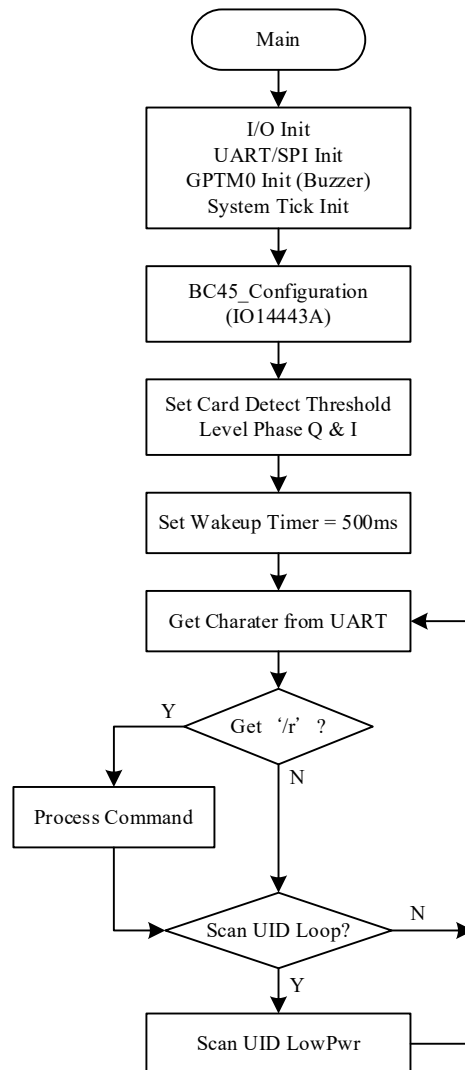
Note: Add \r (return) to the last byte of the command, which means the end of a command.

## Holtek NFC Reader Library Instructions

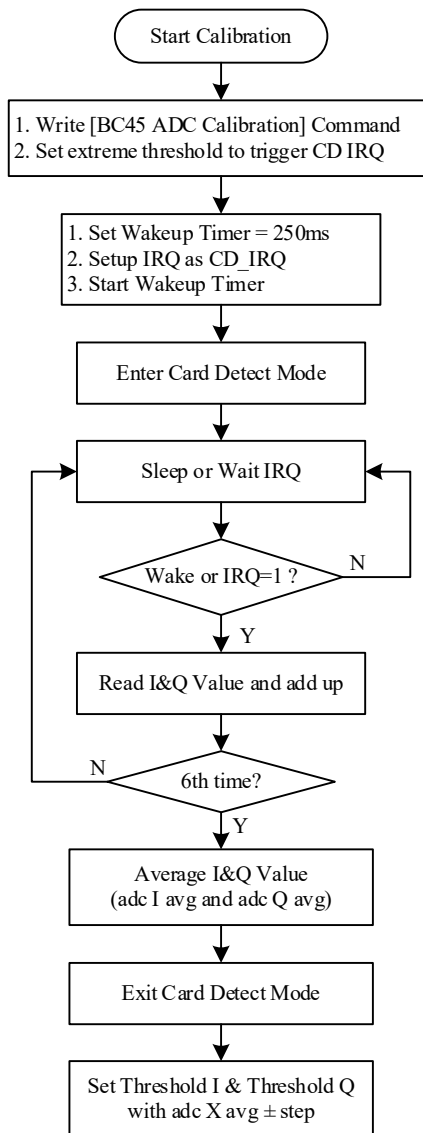
### Directory Structure

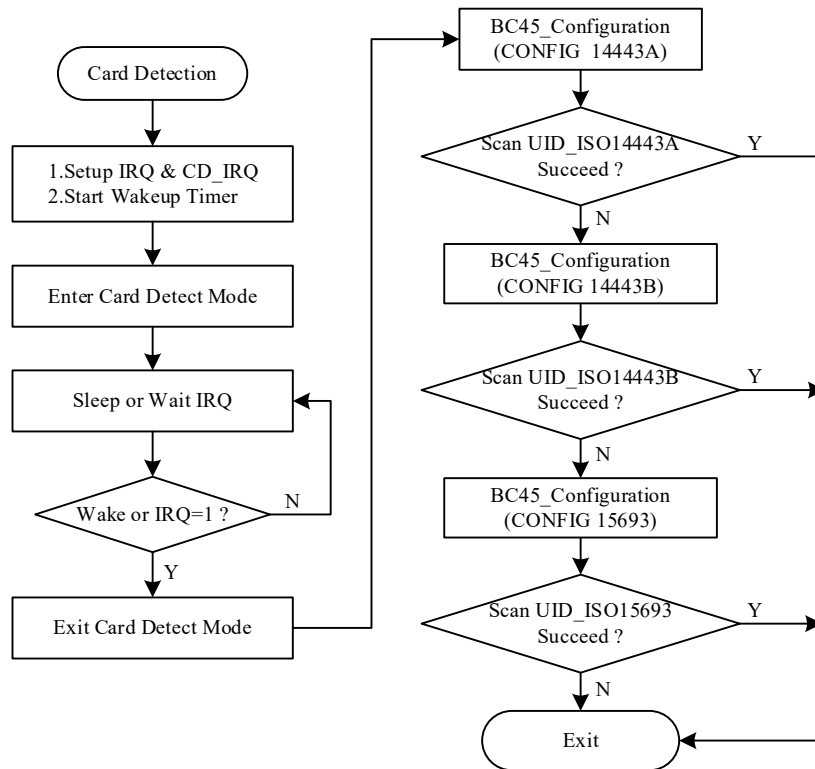


### Main Program Flowchart



### Card Detection Flowchart





## Library Functions

### Category List

	API	Functional Description
ISO14443A Category	ISO14443A_Command	ISO14443A interface command
	ISO14443A_Config	ISO14443A initialisation setup
	ISO14443A_Get_Speed_Reader	Get reader speed configuration
	ISO14443A_Request	Execute ISO14443A Request command
	ISO14443A_WakeUp	Execute ISO14443A Wakeup command
	ISO14443A_Anticoll	Execute ISO14443A Anti Collision command
	ISO14443A_Select	Execute ISO14443A Select command
	ISO14443A_RATS	Execute ISO14443A RATS (Request for Answer To Select) command
	ISO14443A_PPS	Execute ISO14443A PPS (Protocol and Parameter Selection) command
Crypto_M	ISO14443A_Halt	Execute ISO14443A Halt command
	ISO14443A_Load_Key	Load key
	ISO14443A_Authentication	Execute Crypto_M authentication
	ISO14443A_Write_Mifare_Block	Execute Crypto_M Write Block command
	ISO14443A_Read_Mifare_Block	Execute Crypto_M Read Block command
	ISO14443A_Decrement	Execute Crypto_M Decrement command
	ISO14443A_Increment	Execute Crypto_M Increment command
	ISO14443A_Restore	Execute Crypto_M Restore command
ISO14443B Category	ISO14443A_Transfer	Execute Crypto_M Transfer command
	ISO14443B_Command	ISO14443B interface command
	ISO14443B_Config	ISO14443B initialisation setup
	ISO14443B_Get_Speed_Reader	Get reader speed configuration
	ISO14443B_Request	Execute ISO14443B Request command
	ISO14443B_WakeUp	Execute ISO14443B Wakeup command
	ISO14443B_ATTRIB	Execute ISO14443B Attribute command
	ISO14443B_Halt	Execute ISO14443B Halt command

	API	Functional Description
ISO15693 Category	ISO15693_Command	ISO15693 interface command
	ISO15693_Config	ISO15693 initialisation setup
	ISO15693_Get_Speed_Reader	Get reader speed configuration
	ISO15693_Inv_Req_1_Slot	Execute ISO15693 Inventory (1 Slot) command
	ISO15693_Inv_Req_16_Slots	Execute ISO15693 Inventory (16 Slots) command
	ISO15693_Stay_Quiet	Execute ISO15693 Stay Quiet command
	ISO15693_Select	Execute ISO15693 Select command
	ISO15693_Reset_to_Ready	Execute ISO15693 Reset to Ready command
	ISO15693_Read_Single_Block	Execute ISO15693 Read Single Block command
	ISO15693_Write_Single_Block	Execute ISO15693 Write Single Block command
	ISO15693_Lock_Block	Execute ISO15693 Lock Block command
	ISO15693_Read_Multiple_Blocksgfs	Execute ISO15693 Read Multiple Blocks command
	ISO15693_Write_Multiple_Blocks	Execute ISO15693 Write Multiple Blocks command
	ISO15693_Write_AFI	Execute ISO15693 Write AFI command
	ISO15693_Lock_AFI	Execute ISO15693 Lock AFI command
	ISO15693_Write_DSFID	Execute ISO15693 Write DSFID command
	ISO15693_Lock_DSFID	Execute ISO15693 Lock DSFID command
	ISO15693_Get_System_Information	Execute ISO15693 Get System Information command
	ISO15693_Get_Multiple_Block_Security_Status	Read the security status of multiple blocks of VICC
BC45B4523 Chip Function	Transparent_With_CRC	Transmit/receive data with CRC
	Transparent_Without_CRC	Transmit/receive data without CRC
Demo Board Client Function	ScanUID_ISO14443ATagType	Scan the UID of ISO14443A tag type
	ScanUID_ISO14443BTagType	Scan the UID of ISO14443B tag type
	ScanUID_ISO15693TagType	Scan the UID of ISO15693 tag type

## ISO14443A Category

Name	uint8_t ISO14443A_Command (uint8_t Command, uint8_t *Param, uint16_t LenParam, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Follows the ISO14443A and Crypto_M interface commands to support all the functions below. For functional definitions, refer to the ISO14443A_Category.h file.	
Input	Command	Command related to ISO14443A/MIFARE
	*Param	Parameter related to the command
	LenParam	Length of the input parameter
Output	*Data_Resp	Response to the operation command
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Config (uint8_t Speed)	
Function	Configures buffer and RF communication speed for ISO14443A operations	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
Output	None	
Return	Return the success status	

Name	uint8_t ISO14443A_Get_Speed_Reader (uint8_t *Speed, uint16_t *LenSpeed)	
Function	Returns the current RF speed configuration	
Input	None	
Output	*Speed	High nibble: TX speed; Low nibble: RX speed
	*LenSpeed	Length of the command output
Return	Return the success status	

Name	uint8_t ISO14443A_Request (uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Request command according to the ISO14443A standard	
Command	26H	
Input	None	
Output	*Data_Resp	If the command succeeds, reply with the corresponding command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_WakeUp (uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes WakeUp command according to the ISO14443A standard	
Command	52H	
Input	None	
Output	*Data_Resp	If the command succeeds, reply with the WakeUp command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Anticoll (uint8_t Level, uint8_t CollMask_Value, uint8_t *NumColl, uint8_t *Coll_Pos, uint8_t *UID, uint16_t *LenUID)	
Function	Executes Anti Collision command according to the ISO14443A standard, only one card can be selected in the field	
Command	93H;95H;97H	
Input	Level	Selects Cascade Level
	CollMask_Value	Collision bit setup, refer to the ISO14443A_Anticoll function for detailed functional definition
Output	*NumColl	Number of collisions
	*Coll_Pos	Bit position when collision occurs
	*UID	Selects a card (UID) to operate on
	*LenUID	Length of the UID received in anti-collision
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Select (uint8_t Level, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Select command according to the ISO14443A standard	
Command	93H;95H;97H	
Input	Level	Selects Cascade Level (1~3)
	*UID	Selects a card (UID) to operate on
Output	*Data_Resp	If the command succeeds, reply with the command data of the card; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_RATS (uint8_t Parameter_Byte, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes RATS command of the ISO14443A-4 standard	
Command	E0H	
Input	Parameter_Byte	High nibble: FSDI (maximum packet length that can be received by PCD); Low nibble: CID (logical number of the addressed card)
Output	*Data_Resp	If the command succeeds, reply with the RATS command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_PPS (uint8_t CID, uint8_t PPS0, uint8_t PPS1, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes PPS command of the ISO14443A-4 standard	
Command	D0H	
Input	CID	Logical number of the addressed card
	PPS0	Defines whether to transmit PPS1
	PPS1	“0000”+DRI(2-bit) + DSI(2-bit) DSI: Divisor integer selected from card to reader DRI: Divisor integer selected from reader to card
Output	*Data_Resp	If the command succeeds, reply with the PPS command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Halt (uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes a Halt command according to the ISO14443A standard	
Command	50H	
Input	None	
Output	*Data_Resp	If the command succeeds, there is no need to respond with data; If the Halt fails, reply with a HALT message; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

### Crypto\_M

Name	uint8_t ISO14443A_Load_Key (uint8_t *Key)	
Function	Loads key into the BC45B4523 master key buffer	
Input	Key	6-byte key data
Output	None	
Return	State after the function is processed	

Name	uint8_t ISO14443A_Authentication (uint8_t Select_Key, uint8_t Block_Num, uint8_t *UID, uint8_t LenUID)	
Function	Executes Authentication command of Crypto_M	
Command	60H;61H	
Input	Select_Key	Selects Key_A(0x00) or Key_B(0x01)
	Block_Num	Data block to be authenticated
	*UID	Selects a card (UID) to operate on
	LenUID	Length of the UID
Output	None	
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Write_Mifare_Block (uint8_t Block_Num, uint8_t *Data_Wr, uint8_t *Data_Resp, uint16_t *LenData_Resp, uint8_t Check_Crypto)	
Function	Executes Write Block command of Crypto_M	
Command	A0H	
Input	Block_Num	Address to be written with a data block
	*Data_Wr	16-byte data to be written
Output	*Data_Resp	If the command succeeds, no need to respond with data; If the write operation fails, reply with a NACK message; For other errors, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
	Check_Crypto	Check whether the authentication has passed before execution
Return	For the state after the function is processed, refer to the comments in the program	



Name	uint8_t ISO14443A_Read_Mifare_Block (uint8_t Block_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp, uint8_t Check_Crypto)	
Function	Executes Read Block command of Crypto_M	
Command	30H	
Input	Block_Num	Address of the data block to be read
Output	*Data_Resp	If the command succeeds, reply with the data read out; If the read operation fails, reply with a NACK message; For other errors, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
	Check_Crypto	Check whether the authentication has passed before execution
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Decrement (uint8_t Block_Num, uint8_t *Decrement_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	C0H	
Command	Executes Decrement command of Crypto_M	
Input	Block_Num	Data block to implement decrement
	Decrement_Value	4-byte subtrahend
Output	*Data_Resp	If the command succeeds, no need to respond with data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Increment (uint8_t Block_Num, uint8_t *Increment_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Increment command of Crypto_M	
Command	C1H	
Input	Block_Num	Data block to implement increment
	Increment_Value	4-byte addend
Output	*Data_Resp	If the command succeeds, no need to respond with data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Restore (uint8_t Block_Num, uint8_t *Restore_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Restore command of Crypto_M	
Command	C2H	
Input	Block_Num	Reads out the value of the data block
	Restore_Value	No real value, any value can be entered
Output	*Data_Resp	If the command succeeds, no need to respond with data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443A_Transfer (uint8_t Block_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Transfer command of Crypto_M	
Command	B0H	
Input	Block_Num	Data block to be set with a new value
Output	*Data_Resp	If the command succeeds, no need to respond with data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

**ISO14443B Category**

Name	uint8_t ISO14443B_Command (uint8_t Command, uint8_t *Param, uint16_t LenParam, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Follows the ISO14443B interface commands to support all the functions below. For functional definitions, refer to the ISO14443B_Category.h file.	
Input	Command	Command related to ISO14443B
	*Param	Parameter related to the command
	LenParam	Length of the input parameter
Output	*Data_Resp	Response to the operation command
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443B_Config (uint8_t Speed)	
Function	Configures buffer and RF communication speed for ISO14443B operations	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
Output	None	
Return	Return the success status	

Name	uint8_t ISO14443B_Get_Speed_Reader (uint8_t *Speed, uint16_t *LenSpeed)	
Function	Returns the current RF speed configuration	
Input	None	
Output	*Speed	High nibble: TX speed; Low nibble: RX speed
	*LenSpeed	Length of the command output
Return	Return the success status	

Name	uint8_t ISO14443B_Request (uint8_t AFI, uint8_t Num_Slots_N, uint8_t *Slot_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Request command according to the ISO14443B standard	
Input	AFI	PICC application type locked by PCD
	Num_Slots_N	Number of slots, refer to the ISO14443B specification for details
Output	*Slot_Num	The first slot number to find a card
	*Data_Resp	If the command succeeds, reply with the corresponding command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443B_WakeUp (uint8_t AFI, uint8_t Num_Slots_N, uint8_t *Slot_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes WakeUp command according to the ISO14443B standard	
Input	AFI	PICC application type locked by PCD
	Num_Slots_N	Number of slots, refer to the ISO14443B specification for details
Output	*Slot_Num	The first slot number to find a card
	*Data_Resp	If the command succeeds, reply with the WakeUp command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443B_ATTRIB (uint8_t *PUPi, uint8_t *Param, uint8_t *Higher_Layer, uint16_t LenHigher_Layer, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Command	1DH	
Function	Executes ATTRIB command according to the ISO14443B standard	
Input	PUPi	Unique PICC identifier (4-byte)
	Param	4-byte parameter which follows ISO14443B
	Higher_Layer	Higher level information which follows ISO14443B, refer to the ISO14443B specification
	LenHigher_Layer	Length of the obtained higher level information
Output	*Data_Resp	If the command succeeds, reply with the ATTRIB command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO14443B_Halt (uint8_t *PUPi, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Halt command according to the ISO14443B standard	
Command	50H	
Input	PUPi	Unique PICC identifier (4-byte)
Output	*Data_Resp	If the command succeeds, reply with the Halt command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

### ISO15693 Category

Name	uint8_t ISO15693_Command (uint8_t Command, uint8_t *Param, uint16_t LenParam, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Follows the ISO15693 interface commands to support all the functions below. For functional definitions, refer to the ISO15693_Category.h file.	
Input	Command	Command related to ISO15693
	*Param	Parameter related to the command
	LenParam	Length of the input parameter
Output	*Data_Resp	Response to the operation command
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Config (uint8_t Speed)	
Function	Configures buffer and RF communication speed for ISO15693 operations	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
Output	None	
Return	Return the success status	

Name	uint8_t ISO15693_Get_Speed_Reader (uint8_t *Speed, uint16_t *LenSpeed)	
Function	Returns the current RF speed configuration	
Input	None	
Output	*Speed	High nibble: TX speed; Low nibble: RX speed
	*LenSpeed	Length of the command output
Return	Return the success status	

Name	uint8_t ISO15693_Inv_Req_1_Slot (uint8_t Speed, uint8_t Inv_Mode, uint8_t AFI, uint8_t Mask_Len, uint8_t *Mask_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Inventory (1 Slot) command according to the ISO15693 standard	
Command	01H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + AFI mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	AFI	VICC application type locked by VCD
	Mask_Len	Length of mask code indicating the effective number of bits
	*Mask_Value	Mask code placed in the byte array
Output	*Data_Resp	If the command succeeds, reply with the Inventory(01H) command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Inv_Req_16_Slots (uint8_t Speed, uint8_t Inv_Mode, uint8_t AFI, uint8_t Mask_Len, uint8_t *Mask_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Inventory (16 Slots) command according to the ISO15693 standard	
Command	01H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + AFI mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	AFI	VICC application type locked by VCD
	Mask_Len	Length of mask code indicating the effective number of bits
	*Mask_Value	Mask code placed in the byte array
Output	*Data_Resp	If the command succeeds, reply with the Inventory(01H) command data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Stay_Quiet (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Stay Quiet command according to the ISO15693 standard	
Command	02H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, there is no need to respond with data; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Select (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Select command according to the ISO15693 standard	
Command	25H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, reply with the Select (25H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Reset_to_Ready (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp);	
Function	Executes the Reset to Ready according to the ISO15693 standard, VICC will return to Ready state	
Command	26H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, reply with the Reset (26H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Read_Single_Block (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t Block_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Read Single Block command according to the ISO15693 standard	
Command	20H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	Block_Num	Serial number of the data block to be read
Output	*Data_Resp	If the command succeeds, reply with the Read Single Block (20H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Write_Single_Block (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t Block_Size, uint8_t *Write_Block_Param, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Write Single Block command according to the ISO15693 standard	
Command	21H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	Block_Size	Size of data block (byte)(different brands may be different)
	*Write_Block_Param	1st byte: serial number of the data block to be written 2nd...end: data to be written
Output	*Data_Resp	If the command succeeds, reply with the Write Single Block (21H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Lock_Block (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t Block_Num, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Lock Block command according to the ISO15693 standard	
Command	22H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	Block_Num	Serial number of data block to be locked
Output	*Data_Resp	If the command succeeds, reply with the Lock Block (22H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Read_Multiple_Blocks (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Read_Multi_Block_Param, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Read Multiple Blocks command according to the ISO15693 standard	
Command	23H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	*Read_Multi_Block_Param	1st byte: serial number of the first data block to be read 2nd byte: number of data blocks to be read
Output	*Data_Resp	If the command succeeds, reply with the Read Multiple Blocks (23H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Write_Multiple_Blocks (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t Block_Size, uint8_t *Write_Multi_Block_Param, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Write Multiple Blocks command according to the ISO15693 standard	
Command	24H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	Block_Size	Size of data block (byte)(different brands may be different)
	*Write_Multi_Block_Param	1st byte: serial number of the first data block to be written 2nd byte: number of data blocks to be written 3rd byte...Nth byte: data to be written
Output	*Data_Resp	If the command succeeds, reply with the Write Multiple Blocks (24H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Write_AFI (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t AFI_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Write AFI command according to the ISO15693 standard	
Command	27H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	AFI_Value	VICC application type locked by VCD
Output	*Data_Resp	If the command succeeds, reply with the Write AFI (27H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Lock_AFI (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Lock AFI command according to the ISO15693 standard, always locking the AFI value into VICC memory	
Command	28H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, reply with the Lock AFI (28H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Write_DSFD (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t DSFD_Value, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Write DSFD (Data Storage Format Identifier) command according to the ISO15693 standard	
Command	29H	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	DSFD_Value	Data storage format identifier value
Output	*Data_Resp	If the command succeeds, reply with the Write DSFD (29H) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Lock_DSFDID (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Executes Lock DSFDID (Data Storage Format Identifier) command according to the ISO15693 standard, always locking the DSFDID value into VICC memory	
Command	2AH	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, reply with the Lock DSFDID (2AH) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Get_System_Information (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Gets ICC system information	
Command	2BH	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
Output	*Data_Resp	If the command succeeds, reply with the Get System Information (2BH) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t ISO15693_Get_Multiple_Block_Security_Status (uint8_t Speed, uint8_t Non_Inv_Mode, uint8_t *UID, uint8_t *Get_Multi_Block_Secure_Param, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Gets the security status of multiple blocks of VICC	
Command	2CH	
Input	Speed	High nibble: TX speed; Low nibble: RX speed
	Non_Inv_Mode	RFU_Flg(1 bit) + Option Flg(1 bit) + Protocol Extension Flg(1 bit) + '0' + Operation mode(4 bits), refer to the ISO15693 specification and ISO15693_Category.h file for details
	*UID	Unique ISO15693 identifier (8-byte)
	*Get_Multi_Block_Secure_Param	1st byte: serial number of the first data block to be read 2nd byte: number of data blocks to be read
Output	*Data_Resp	If the command succeeds, reply with the Get Multiple Block Security Status (2CH) command data; If a Flag error occurs, respond with the error code of the ISO15693 standard; If the command fails, reply with an error message; For other states, no data is present
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	



**BC45B4523 Chip Function**

Name	uint8_t Transparent_With_CRC (uint8_t *Data_Tx, uint16_t LenData_Tx, uint8_t TimeOut, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Transmits data with the calculated CRC byte and checks the received response data using the CRC. If the received CRC is correct, the Data_Resp will not show the CRC byte.	
Input	*Data_Tx	Data to be transmitted
	LenData_Tx	Length of data to be transmitted
	TimeOut	RF data response waiting time 0x00: use the last setting 0x01: 1ms 0x02: 2ms 0x03: 4ms 0x04: 8ms 0x05: 16ms 0x06: 32ms 0x07: 64ms 0x08: 128ms 0x09: 256ms 0x0A: 512ms 0x0B: 1s 0x0C: 2s 0x0D: 4s 0x0E: 8s 0x0F: 16s 0x10: 32s
Output	*Data_Resp	Response data
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

Name	uint8_t Transparent_Without_CRC (uint8_t *Data_Tx, uint16_t LenData_Tx, uint8_t TimeOut, uint8_t *Data_Resp, uint16_t *LenData_Resp)	
Function	Transmits data without the CRC byte and do not check CRC when receiving response data	
Input	*Data_Tx	Data to be transmitted
	LenData_Tx	Length of data to be transmitted
	TimeOut	Refer to Transparent_With_CRC for details
Output	*Data_Resp	Response data
	*LenData_Resp	Length of the response data
Return	For the state after the function is processed, refer to the comments in the program	

**Demo Board Client Function**

Name	uint8_t ScanUID_ISO14443ATagType(void)	
Function	Scans the UID of ISO14443A tag type	
Input	None	
Output	None	
Return	For the state after the function is processed, refer to the comments in the program.	

Name	uint8_t ScanUID_ISO14443BTagType(void)	
Function	Scans the UID of ISO14443B tag type	
Input	None	
Output	None	
Return	For the state after the function is processed, refer to the comments in the program.	

Name	uint8_t ScanUID_ISO15693TagType(void)	
Function	Scans the UID of ISO15693 tag type	
Input	None	
Output	None	
Return	For the state after the function is processed, refer to the comments in the program.	

## Conclusion

Using a demo board example, this text has introduced how to use the BC45B4523 NFC reader, including hardware and software descriptions, which can help users with their rapid product development.

## Reference Material

Reference files: BC45B4523, HT32F52241, HT42B534-2 datasheet.

For more details, refer to the Holtek website: [www.holtek.com](http://www.holtek.com).

## Versions and Modification Information

Date	Author	Issue	Modify Information
2022.02.21	王冠中	V1.10	Update the attachment program.
2020.06.11	王冠中	V1.00	First version.

## Disclaimer

All information, trademarks, logos, graphics, videos, audio clips, links and other items appearing on this website ('Information') are for reference only and is subject to change at any time without prior notice and at the discretion of Holtek Semiconductor Inc. and its related companies (hereinafter 'Holtek', 'the company', 'us', 'we' or 'our'). Whilst Holtek endeavors to ensure the accuracy of the Information on this website, no express or implied warranty is given by Holtek to the accuracy of the Information. Holtek shall bear no responsibility for any incorrectness or leakage.

Holtek shall not be liable for any damages (including but not limited to computer virus, system problems or data loss) whatsoever arising in using or in connection with the use of this website by any party. There may be links in this area, which allow you to visit the websites of other companies. These websites are not controlled by Holtek. Holtek will bear no responsibility and no guarantee to whatsoever Information displayed at such sites. Hyperlinks to other websites are at your own risk.

### Limitation of Liability

In no event shall Holtek Limited be liable to any other party for any loss or damage whatsoever or howsoever caused directly or indirectly in connection with your access to or use of this website, the content thereon or any goods, materials or services.

### Governing Law

The Disclaimer contained in the website shall be governed by and interpreted in accordance with the laws of the Republic of China. Users will submit to the non-exclusive jurisdiction of the Republic of China courts.

**Update of Disclaimer**

Holtek reserves the right to update the Disclaimer at any time with or without prior notice, all changes are effective immediately upon posting to the website.