CipherLab
User Guide

1800 Programming Guide

C Language Programming

For 8 Series Mobile Computers

Version 2.00

CIPHERLAB

CIPHERLAB CO., LTD.
Website: http://www.cipherlab.com

RELEASE NOTES

Version	Date	Notes
2.00	Dec. 18, 2014	▸ New: **1.4** Host Data Message ▸ Modified: **2.6** – RFIDGetSystemVoltage() added with status parameter ▸ Modified: **4.4** – Scan delay time parameters for 1862 added ▸ New: **5.10 Data Output via USB** ▸ New: **Chapter 6 Host Command**
1.00	Sep. 10, 2012	Initial Release

CONTENTS

INTRODUCTION

There are a number of mobile-specific library routines to facilitate the development of the user program. These functions cover a wide variety of tasks, including communications, showing string or bitmap on the buzzer control, scanning, RFID configurations, etc. They are categorized and described in this section by their functions or the resources they work on.

The function prototypes of the library routines, as well as the declaration of the system variables, can be found in the library header file, e.g. **"8XRFGUN.h"**. It is assumed that the programmer has prior knowledge of the C language.

Chapter 1

GLOBAL

IN THIS CHAPTER

1.1 RFID FUNCTION

RFIDInitial	
Purpose	To initialize the RFID Gun.
Syntax	**void RFIDInitial (void*)***
Example	`RFIDInitial ();`
Remarks	The RFID Gun will not work unless it is initialized.
See Also	RFIDHalt

RFIDHalt	
Purpose	To stop the RFID Gun from operating.
Syntax	**void RFIDHalt (void*)***
Example	`RFIDHalt ();`
See Also	RFIDInitial

1.2 PACKET DATA MESSAGE

The following global variables relate to routines that get 1800 packet data message. These variables are declared by the system; users don't have to declare them.

extern PACKETDATA_MSG PacketData_Msg

Syntax

```
typedef struct{
    INT Year;
    INT Month;
    INT Day;
    INT Hour;
    INT Minute;
    INT Second;
    INT Millisecond;
} TIME;

typedef struct{
    TIME Time;
    BYTE RFID_Func;
    INT UL;
    BYTE PC [2];
    INT  EPCLength;
    BYTE EPC [33];
    BYTE CRC [2];
    INT  DataLength;
    BYTE Data [33];
} PACKETDATA_MSG;
```

Remarks

Gotten successfully, the packet data is stored in the PACKETDATA_MSG structure.

RFIDGetPacketDataMessage	
Purpose	Get 1800 packet data message.
Syntax	**INT RFIDGetPacketDataMessage (void)**

Example	`while (1)`
	`{`
	`if (RFIDGetPacketDataMessage ())`
	`printf ("Get 1800 Packet Data");`
	`}`

Return Value	Successful will return 1 or else return 0.
Remarks	Gotten successfully, the event message is stored in the PACKETDATA_MSG structure**.**

The following global variables relate to routines that get 1800 hexadecimal and raw data message. These variables are declared by the system; users don't have to declare them.

extern NONPACKETDATA_MSG NonPacketData_Msg	
Syntax	typedef struct{ **INT** DataLength; **BYTE** *Data [255]*; } NONPACKETDATA_MSG;
Remarks	Gotten successfully, the non-packet data (hexadecimal and raw data) is stored in NONPACKETDATA_MSG structure.

RFIDGetNonPacketDataMessage	
Purpose	Get 1800 hexadecimal and raw data message.
Syntax	**INT RFIDGetNonPacketDataMessage (void)**

Example	`while (1)`
	`{`
	`if (RFIDGetNonPacketDataMessage ())`
	`printf ("Get 1800 hexadecimal and raw Data");`
	`}`

Return Value	Successful will return 1 or else return 0.
Remarks	Gotten successfully, the data is stored in the NONPACKETDADA_MSG structure.

1.3 EVENT MESSAGE

The following global variables relate to routines that get 1800 event message. These variables are declared by the system; users don't have to declare them.

extern EVENT_MSG Event_Msg

Syntax

```
typedef struct{
    BYTE Event_MSB;
    BYTE Event_LSB;
    BYTE GrpCode;
    BYTE CmdCode;
} EVENT_MSG;
```

Remarks Gotten successfully, the event message is stored in the EVENT_MSG structure.

Event Table: (Event_MSB, Event_LSB)

bit		meaning
0	Event_LSB	System will enter power saving mode after the event.
1		System will shut down after this event.
2		*Bluetooth*® will be disconnected after this event.
3		System setting is changed (by command via USB or FN key) indicating that you can send the Group Code and Command Code with the Event packet.
4		Low Battery Alarm. 1800 sends this event every 30 seconds if battery voltage is lower than 5%.
5		Alternate mode← → RFID mode
6		RFID Fail (initialization fail or no response during operating).
7		No tag is found when scan session is timeout in single mode.
8	Event_MSB	Scan session is completed in multi-tag mode. (New tag amount is equal to multi-tag counter)
15~9		Reserved

RFIDGetEventMessage	
Purpose	Get 1800 event message.
Syntax	**INT RFIDGetEventMessage (void)**

Example	`while (1)`
	`{`
	`  if (RFIDGetEventMessage ())`
	`  printf (“Get 1800 Event message”);`
	`}`

Return Value	Successful will return 1 or else return 0.
Remarks	Only the packet data format is applied to issued events; please refer to 5.1 Data Format. Gotten successfully, the event message is stored in the EVENT_MSG structure.

1.4 HOST DATA MESSAGE

The following global variables relate to routines that get 1800 host data message. These variables are declared by the system; users don't have to declare them.

extern HOSTINVENTORYDATA_MSG HostInventoryData_Msg

Purpose	Get 1800 host inventory data message.
Syntax	typedef struct{ **TIME** *Time*; **BYTE** *Index*; **INT** *UL*; **BYTE** *PC[2]*; **INT** *EPCLength*; **BYTE** *EPC[33]*; **BYTE** *CRC[2]*; **INT** *TIDLength*; **BYTE** *TID[33]*; } HOSTINVENTORYDATA_MSG;
Remarks	Gotten successfully, the host inventory data is stored in the HOSTINVENTORYDATA_MSG structure.

extern HOSTREADTAGDATA_MSG HostReadTagData_Msg

Purpose	Get 1800 host tag data message.
Syntax	typedef struct{ **INT** *DataLength*; **BYTE** *Data[255]*; } HOSTREADTAGDATA_MSG;
Remarks	Gotten successfully, the host tag data is stored in the HOSTREADTAGDATA_MSG structure.

RFIDGetHostMessage

Purpose: Get host inventory, tag memory data, and status message message.

Syntax: **INT RFIDGetHostMessage (void)**

Example:

```
INT value;
while (1)
{
    value = RFIDGetHostMessage();
    if (value == 0x01)
        printf (“Get host inventory EPC”);
    else if (value == 0xFF)
        printf (“Finish the operation”);
}
```

Return Value

0x01	Get Inventory EPC message
0x02	Get Inventory EPC+TID message
0x03	Get reading tag memory data message
0xFF	Host Mode Operation finished
0xF0	Host Mode Inventory/Read/Write/Kill/Lock operation stops due to timeout or retry count exceeded
0xF1	Report the tag is locked in response to the host access command
0	No message

Remarks: When the return value is 0x01 or 0x02, the inventory data is stored in the HOSTINVENTORYDATA_MSG structure.

When the return value is 0x03, the tag memory data is stored in the HOSTREADTAGDATA_MSG structure.

Chapter 2

SYSTEM

IN THIS CHAPTER

2.1 SYSTEM INFORMATION

These routines can be used to collect information on the components, either hardware or software, of the reader.

RFIDGetSystemInfo	
Purpose	Get current system information.
Syntax	**INT RFIDGetSystemInfo (BYTE** **Name*, **BYTE** **SN*, **BYTE** **KernelVersion*, **BYTE** **STDVersion*, **BYTE** **MACID*);
Parameters	**BYTE** **Name* Model name.(5 Bytes) **BYTE** **SN* Series Number.(9 Bytes) **BYTE** **KernelVersion* Kernel version.(10 Bytes) **BYTE** **STDVersion* Standard Firmware Version.(10 Bytes) **BYTE** **MACID* *Bluetooth*® MACID.(6 Bytes)

Example	`BYTE Name [6], SN [10], KernelVersion [11], STDVersion [11] , MACID [6];`
	
	`if (RFIDGetSystemInfo (Name, SN, KernelVersion, STDVersion, MACID)==1)`
	`{`
	`  printf ("Model: %s\n", Name);`
	`  printf ("S/N: %s\n", SN);`
	`  printf ("KNL: %s"\n, KernelVersion);`
	`  printf ("USR: %s"\n, STDVersion);`
	`  for (i=0;i<6;i++)  printf ("%02X:", MACID [i]);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

2.2 SYSTEM TIME

RFIDGetSystemTime	
Purpose	Get current system time.
Syntax	**INT RFIDGetSystemTime (INT** *Year, **INT** *Month, **INT** *Day, **INT** *Hour, **INT** *Minute, **INT** *Second**)**
Parameters	**INT** *Year Year **INT** *Month Month **INT** *Day Day **INT** *Hour Hour **INT** *Minute Minute **INT** *Second Second

Example	`INT year, month, day, hour, minute, second;`
	`if(RFIDGetSystemTime(&year,&month,&day,&hour,&minute,&second)==1)`
	`{`
	`  printf ("Time: %02d/%02d/%02d %02d:%02d:%02d", year, month, day,`
	`  hour, minute, second);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetSystemTime

RFIDSetSystemTime	
Purpose	Set system time.
Syntax	**INT RFIDGetSystemTime (INT** *Year*, **INT** *Month*, **INT** *Day*, **INT** *Hour*, **INT** *Minute*, **INT** *Second***)**
Parameters	**INT** *Year* Year.(0~99) **INT** *Month* Month.(1-12) **INT** *Day* Day.(1-31) **INT** *Hour* Hour.(0~23) **INT** *Minute* Minute.(0~59) **INT** *Second* Second.(0~59)
Example	`if (RFIDSetSystemTime (12, 3, 30, 12, 10, 10)==1)` `  printf ("Set System time.”);`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetSystemTime

2.3 SHUT DOWN TIMEOUT

The reader will stay active at power-on, which may be followed by a transition from full CPU speed to low CPU speed (Power-Saving) to auto shutdown (Auto Power Off). 1800Utility allows you to configure the delay time to shut down 1800 RFID Reader.

RFIDGetShutDownTimeout	
Purpose	Get the current auto shutdown delay time.
Syntax	**RFIDGetShutDownTimeout (INT** **Time***)**
Parameters	**INT** **Time* Current system shutdown delay time.

Example	`INT shutTime;`
	`if (RFIDGetShutDownTimeout (&shutTime)==1)`
	`  printf ("Shutdown delay time: %d min", shutTime);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetShutDownTimeout

RFIDSetShutDownTimeout	
Purpose	Set auto shutdown delay time.
Syntax	**RFIDSetShutDownTimeout (INT** *Time***)**
Parameters	**INT** *Time* System shutdown delay time. 0,1~254 (x1 minute) (**default=10**) If Time=0, Auto shutdown disabled.
Example	`if (RFIDSetShutDownTimeout (0)==1)` `  printf ("Disable the shutdown delay time.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetShutDownTimeout

2.4 POWER SAVING TIMEOUT

Power Saving (1~254 min.; 0= Disable): By default, it is set to idle at full-speed for 2 minutes before it enters power saving mode. If this feature is not desired, set it to 0.

RFIDGetPowerSavingTimeout	
Purpose	Get the current power saving mode delay time.
Syntax	**INT RFIDGetPowerSavingTimeout (INT** **Time***)**
Parameters	**INT** **Time* Current enter power saving delay time.

Example	`INT  sleepTime;`
	`if (RFIDGetPowerSavingTimeout (&sleepTime)==1)`
	`  printf ("Power saving timeout: %d min", sleepTime);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetPowerSavingTimeout

RFIDSetPowerSavingTimeout	
Purpose	Set power saving mode delay time.
Syntax	**INT RFIDSetPowerSavingTimeout (INT** *Time***)**
Parameters	**INT** *Time* Enter power saving delay time. 0,1~254 (x1 minute) (**default=2**) If Time=0, Never Enter Power Saving Mode
Example	`if (RFIDSetPowerSavingTimeout (1)==1)` `  printf ("Set 1 minute of power saving timeout.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetPowerSavingTimeout

2.5 KEEP ALIVE

If the delay time for system shutdown and power saving mode is not zero, RFIDKeepAlive will prevent the 1800 RFID reader from entering the power saving mode or shutting down.

RFIDKeepAlive	
Purpose	Send Keep Alive command.
Syntax	**INT RFIDKeepAlive (void)**
Example	`if (RFIDKeepAlive ()==1)` `  printf ("Keep Alive Successful.");`
Return Value	If successful, it returns 1.
Remarks	Keeps the system active without entering power saving mode or shutting down.

2.6 VOLTAGE

RFIDGetSystemVoltage	
Purpose	Get the current System Voltage
Syntax	**INT RFIDGetSystemVoltage (INT** **Voltage,* **INT** **Status***);**
Parameters	**INT** **Voltage* System voltage (percentage) **INT** **Status* Charging status

value	Charging status
0	Not charging
1	Charging
2	Charging done

Example	`INT voltage, status;`
	`if (RFIDGetSystemVoltage (&voltage, &status)==1)`
	`{`
	`  printf ("Voltage: %3d%%", voltage);`
	`  printf ("Charging status: %d", status);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	value=60 means 60%

2.7 TRANSMIT BUFFER

By default, transmit buffer is enabled when the 1800 RFID Reader is out of range. However, the host computer may not receive the data immediately if the reader is out of range. With the 2KB transmit buffer, the reader can keep on reading tags until the buffer is fullr, even out of transmission range.

RFIDGetTransmitBuffer

Purpose	Get current transmission buffer setting.
Syntax	**INT RFIDGetTransmitBuffer (INT** **Enable,* **INT** **DelayTime***);**
Parameters	**INT** **Enable* Enable/Disable Tx Buffer. **INT** **DelayTime* Send Tx Buffer Delay.

Example	`INT enable, delayTime;`
	`if (RFIDGetTransmitBuffer (&enable, &delayTime)==1)`
	`{`
	`  printf ("Tx Buffer %d\n", enable);`
	`  printf ("Send Delay:%d\n", delayTime);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetTransmitBuffer

RFIDSetTransmitBuffer

Purpose	Set transmission buffer setting
Syntax	**INT RFIDSetTransmitBuffer (INT** *Enable,* **INT** *DelayTime***);**
Parameters	**INT** *Enable* Enable/Disable Tx Buffer. 0 – Disable, 1 – Enable(**default**) **INT** *DelayTime* Send Tx Buffer Delay.

value	Delay
0	0 (**default**)
1	250ms
2	500ms
3	1 s
4	2 s
5	3 s
6	5 s
7	8 s

Example	`if (RFIDSetTransmitBuffer (1, 3)==1)` `  printf ("Enable transmission buffer and 1 sec buffer delay.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetTransmitBuffer

2.8 BEEPER

RFIDGetBeeperSetting

Purpose	Get current beeper setting
Syntax	**INT RFIDGetBeeperSetting (INT** **BeeperVol*, **INT** **LBAlert*, **INT** **CmdBeep*, **INT** **GoodRdBeep*, **INT** **MTagRptBeep*, **INT** **MTagLstFullBeep*);
Parameters	**INT** **BeeperVol* Volume of Beeper. **INT** **LBAlert* Low Battery Alert. **INT** **CmdBeep* Command Indicating Beeping. **INT** **GoodRdBeep* Good Read Beeping. **INT** **MTagRptBeep* Repeated Tag Beeping in Multi-Tag mode. **INT** **MTagLstFullBeep* Tag List Full Beeping in Multi-Tag mode.

Example	`INT BeeperVol, LBAlert, CmdBeep, GoodRdBeep, MTagRptBeep, MTagLstFullBeep;`
	`if (RFIDGetBeeperSetting (&BZVol, &LVAlarm, &CmdBeep, &GoodRdBeep, &MTagRptBeep, &MTagLstFullBeep)==1)`
	`{`
	`printf ("Beeper Volume: %d\n", BeeperVol);`
	`printf ("Low Battery Alert: %d\n", LBAlert);`
	`printf ("Cmd Beep: %d\n", CmdBeep);`
	`printf ("Good Read Beep: %d\n", GoodRdBeep);`
	`printf ("Repeat Tag Beep: %d\n", MTagRptBeep);`
	`printf ("Tag List Full Beep %d\n", MTagLstFullBeep);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetBeeperSetting

RFIDSetBeeperSetting

Purpose: Set beeper setting

Syntax:

INT RFIDSetBeeperSetting (INT *BeeperVol*,
INT *LBAlert*,
INT *CmdBeep*,
INT *GoodRdBeep*,
INT *MTagRptBeep*,
INT *MTagLstFullBeep*);

Parameters:

INT *BeeperVol*
Volume of Beeper

value	Volume
0	Mute
1	Low
2	Medium
3	High(default)

INT *LBAlert*

Low Battery Alert

0 – Disable, 1 – Enable(**default**)

INT *CmdBeep*

Command Indicating Beeping

0 – Disable(**default**), 1 – Enable

INT *GoodRdBeep*

Good Read Beeping

0 – Disable, 1 – Enable(**default**)

INT *MTagRptBeep*

Repeated Tag Beeping in Multi-Tag mode

0 – Disable(**default**), 1 – Enable

INT *MTagLstFullBeep*

Tag List Full Beeping in Multi-Tag mode

0 – Disable, 1 – Enable(**default**)

Example	`if (RFIDSetBeeperSetting (1, 1, 0, 1, 0, 1)==1)`
	`{`
	`printf ("Beeper volume of low.\n”);`
	`printf ("Enable Low Battery Alert.\n”);`
	`printf ("Disable CMD Beep.\n”);`
	`printf ("Enable Good Read Beep.\n”);`
	`printf ("Disable Repeat Tag Beep.\n");`
	`printf ("Enable Tag List Full Beep.\n");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetBeeperSetting

Note: If the Beeper Volume is muted, there will be no Audio feedback from the reader.

2.9 GOOD READ

RFIDGetGoodReadIndicators

Purpose	Get current settings of good read indicators.
Syntax	**INT RFIDGetGoodReadIndicators (INT** **BeeperFreq*, **INT** **BeeperDuration*, **INT** **LedEnable*, **INT** **LedDuration*, **INT** **VibratorEnable*, **INT** **VibratorDuration*);
Parameters	**INT** **BeeperFreq* Beeper Frequency. **INT** **BeeperDuration* Beeper Length. **INT** **LedEnable* Enable/Disable Good read LED. **INT** **LedDuration* Good read LED Duration. **INT** **VibratorEnable* Enable/Disable Good read Vibrator. **INT** **VibratorDuration* Good read Vibrator Duration.

Example

```
INT beeperFreq, beeperDuration, ledEnable, ledDuration, vibratorEnable, vibratorDuration;

if (RFIDGetGoodReadIndicators (&beeperFreq, &beeperDuration, &ledEnable, &ledDuration, &vibratorEnable, &vibratorDuration)==1)
{
   printf ("Buzzer Freq.: %d\n", beeperFreq);
   printf ("Buzzer Duration: %d\n", beeperDuration);
   printf ("Led %d\n", ledEnable);
   printf ("Led Duration: %d\n", ledDuration);
   printf ("Vibrator %d\n", vibratorEnable);
   printf ("Vibrator Dur.: %d\n", vibratorDuration);
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

See Also

RFIDSetGoodReadIndicators

RFIDSetGoodReadIndicators

Purpose

Set current good read indicators settings

Syntax

INT RFIDGetGoodReadIndicators (INT *BeeperFreq*,
INT *BeeperDuration*,
INT *LedEnable*,
INT *LedDuration*,
INT *VibratorEnable*,
INT *VibratorDuration*);

Parameters

INT *BeeperFreq*

Beeper Frequency.

value	Beeper Frequency
0	8 kHz
1	4 kHz(**default**)
2	2 kHz
3	1 kHz

INT *BeeperDuration*

Beeper Length.

value	Beeper Length
0	Shortest(**default**)
1	Short
2	Longer
3	Longest

INT *LedEnable*

Enable/Disable Good read LED.

0 – Disable, 1 – Enable(**default**)

INT *LedDuration*

Good read LED Duration.

1~254 (x 10 ms). (**default=4**)

INT *VibratorEnable*

Enable/Disable Good read Vibrator.

0 – Disable(**default**), 1 – Enable

INT *VibratorDuration*

Good read Vibrator Duration.

1~254 (x 100 ms) (**default=10**)

Example

```
if (RFIDSetGoodReadIndicators (1, 0, 1, 4, 0, 10)==1)
{
    printf (“Set beeper frequency of 4Hz.\n");
    printf (“Set short of beeper length.\n");
    printf (“Enable good read led.\n");
    printf (“Set 40 ms of good read led duration.\n");
    printf (“Disable the vibrator.\n");
    printf (“Set 1000ms of vibrator duration.\n");
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. it returns 0. Or else refer to Status Codes.

See Also RFIDGetGoodReadIndicators

2.10 MEMORY MODE

RFIDGetMemoryMode

Purpose: Get the current setting of memory mode.

Syntax:
```
INT RFIDGetMemoryMode (INT *Enable,
                       INT *Delay,
                       INT *Fsize);
```

Parameters:

INT **Enable*,

Enable/Disable Memory Mode.

INT **Delay*

Data Transmit Delay.

value	Delay
0	0 (default)
1	250ms
2	500ms
3	1 s
4	2 s
5	3 s
6	5 s
7	8 s

INT **Fsize*

Remaining Free Size of Memory.

Example:
```
INT  enable, delay, fSize;
if (RFIDGetMemoryMode (&enable, &delay, &fSize)==1)
{
    printf ("Memory Mode %d\n", enable);
    printf ("Delay: %d\n", delay);
    printf ("Memory Size: %d KB\n", fSize);
}
```

Return Value: If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

See Also: RFIDSetMemoryMode

RFIDSetMemoryMode

Purpose — Set memory Mode setting.

Syntax — **INT RFIDSetMemoryMode (INT** *Index*,
INT *Value*);

Parameters — **INT** *Index*,

INT *Value*

index	Meaning	Value
1	Enable/Disable Memory Mode	0 – Disable (**default**) 1 – Enable
2	Clear Memory	Always=0
3	Upload Memory Data	Always=0
4	Set Data Transmit Delay	*

* Data Transmit Delay

value	Delay
0	0 (default)
1	250ms
2	500ms
3	1 s
4	2 s
5	3 s
6	5 s
7	8 s

Example

```
INT  RFIDFunc;
BYTE Data [50];
if (RFIDSetMemoryScnner (4, 1, &RFIDFunc, Data)==1)
   printf ("Set Data Transmit delay of 250 ms. ");
```

Return Value — If successful, it returns 1.

Remarks — If the command is “Upload Memory Data” (index=3), you can receive the response Memory Data in Packet Data format using the **RFIDGetPacketDataMessage**.

See Also — RFIDGetMemoryMode

2.11 DEFAULT SETTING

RFIDLoadSystemSetting	
Purpose	Load Default Settings.
Syntax	**INT RFIDLoadSystemSetting (INT** *DefaultSetting***);**
Parameters	**INT** *DefaultSetting*,
	Load Default Setting.
	0 – Load Factory Default Setting.
	1 – Load User Setting.
Example	`if (RFIDLoadSystemSetting (0)==1)` `  printf ("Load factory default setting.");`
Return Value	If successful, it returns 1.
	If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSaveSystemSetting

RFIDSaveSystemSetting	
Purpose	Save User Settings.
Syntax	**INT RFIDSaveSystemSetting (void)**
Example	`if (RFIDSaveSystemSetting ()==1)` `  printf ("Save user setting.");`
Return Value	If successful, it returns 1.
	If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDLoadSystemSetting

2.12 FIRMWARE UPGRADE

Upgrade firmware to a single 1800 RFID Reader at a time.

RFIDSetDownloadInterface	
Purpose	Firmware Upgrade.
Syntax	**INT RFIDSetDownloadInterface (INT** *Interface*);
Parameters	**INT** *Interface,* 1 – BT SPP 2 – USB VCOM
Example	`if (RFIDSetDownloadInterface (2)==1)` `printf ("Set USB VCOM to upgrade firmware.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Note: 1. Ensure the 1800 RFID Reader has a fully charged battery prior to attempting a firmware upgrade.
2. In order to avoid the data loss during firmware upgrade, please save or upload all the data from the flash memory before beginning firmware upgrade.

2.13 SHUTDOWN

RFIDShutdown	
Purpose	System Shutdown.
Syntax	**INT RFIDShutdown (void);**
Example	`if (RFIDShutdown ()==1)` `printf ("System shutdown.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

2.14 EVENT STATUS

RFIDGetEventStatus

Purpose: Get current event status.

Syntax: **INT RFIDGetEventStatus (BYTE** **Event_MSB*
BYTE **Event_LSB*);

Parameters: **BYTE** **Event_MSB*

BYTE **Event_LSB*

Event Table. Refer to Event Table.

Event Table:

bit	**Default**	meaning
0	**1**	System will enter power saving mode after event.
1	**1**	System will shut down after this event.
2	**1**	Bluetooth will be disconnected after this event.
3	**1**	System setting is changed (by command via USB or FN key) indicating that you can send the Group Code and Command Code with the Event packet.
4	**0**	Low Battery Alarm. 1800 sends this event every 30 seconds if battery voltage is lower than 5%.
5	**1**	Alternate mode← → RFID mode
6	**1**	RFID Fail (initialization fail or no response during operating).
7	**0**	No tag is found when scan session is timeout in single mode.
8	**0**	Scan session is completed in multi-tag mode. (New tag amount is equal to multi-tag counter)
15~9	**0**	Reserved

Example:

```
BYTE Event_MSB, Event_LSB;
RFIDGetEventStatus (&Event_MSB, &Event_LSB);
```

Return Value: If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

See Also: RFIDSetEventStatus

RFIDSetEventStatus

Purpose: Set event status.

Syntax: **INT RFIDSetEventStatus (BYTE** *Event_MSB*
BYTE *Event_LSB***);**

Parameters: **BYTE** **Event_MSB*

BYTE **Event_LSB*

Event Table. Refer to Event Table.

0 – Disable, 1 – Enable

Event Table:

bit	**Default**	meaning
0	**1**	System will enter power saving mode after event
1	**1**	System will shut down after this event
2	**1**	Bluetooth will be disconnected after this event
3	**1**	System setting is changed (by command via USB or FN key) indicating that you can send the Group Code and Command Code with the Event packet.
4	**0**	Low Battery Alarm. 1800 sends this event every 30 seconds if battery voltage is lower than 5%
5	**1**	Alternate mode← → RFID mode
6	**1**	RFID Fail (initialization fail or no response during operating)
7	**0**	No tag is found when scan session timeout in single mode
8	**0**	scan session completes in multi-tag mode (New tag amount is equal to multi-tag counter)
15~9	**0**	Reserved

Example:

```
if (RFIDSetEventStatus (0x00, 0x6F)==1)
   printf ("Event status setting successful");
```

Return Value: If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

See Also: RFIDGetEventStatus

Chapter 3

INTERFACE

IN THIS CHAPTER

3.1 USB

RFIDUSBGetType	
Purpose	Get current USB interface type.
Syntax	**INT RFIDUSBGetType (INT** **USBType***)**
Parameters	**INT** **USBType* USB interface type.

Example	`INT  usbType;`
	`if (RFIDUSBGetType (&usbType)==1)`
	`  printf ("Type: %d", usbType);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDUSBSetType

RFIDUSBSetType	
Purpose	Set USB interface type.
Syntax	**INT RFIDUSBGetType (INT** *USBType***)**
Parameters	**INT** *USBType* USB interface type. 127 – Virtual COM CDC (**default**) 128 – Virtual COM
Example	`if (RFIDUSBSetType (128)==1)` `  printf ("Set USB Virtual COM.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDUSBGetType

Note: Create a connection and set the available USB interface type between 1800 RFID Reader and host computer.

3.2 BLUETOOTH®

RFIDBTGetSetting

Purpose	Get current *Bluetooth*® parameters.
Syntax	**INT RFIDBTGetSetting (INT** **Discoverable*, **INT** *Authentication, **INT** *BTPowerSaving, **INT** *SSP);
Parameters	**INT** **Discoverable* Discoverable. **INT** **Authentication* Authentication. **INT** **BTPowerSaving* Bluetooth power saving **INT** **SSP* Secure simple pairing

Example	INT discoverable, authentication, BTPowerSaving, SSP;
	if (RFIDBTGetSetting (&discoverable, &authentication, &BTPowerSaving, &SSP)==1)
	{
	printf ("Discoverable %d\n", discoverable);
	printf ("Authentication %d\n", authentication);
	printf ("Bluetooth Power Saving %d\n", BTPowerSaving);
	printf ("Secure Simple Pairing %d\n", SSP);
	}

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	
See Also	RFIDBTSetSetting

RFIDBTSetSetting

Purpose	Set *Bluetooth*® parameters.
Syntax	**INT RFIDBTSetSetting (INT** *Discoverable*, **INT** Authentication, **INT** BTPowerSaving, **INT** SSP);

Parameters	**INT** *Discoverable* Discoverable. 0 – Disable, 1 – Enable(**default**) **INT** *Authentication* Authentication. 0 – Disable(**default**), 1 – Enable **INT** *BTPowerSaving* Bluetooth power saving 0 – Disable, 1 – Enable(**default**) **INT** *SSP* Secure simple pairing 0 – Disable, 1 – Enable(**default**)

Example	if (RFIDBTSetSetting (1, 0, 1, 1)==1)
	{
	printf ("Enable discoverable.\n");
	printf ("Disable authentication.\n");
	printf ("Enable Bluetooth power saving.\n");
	printf ("Enable secure simple pairing.\n");
	}

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDBTGetSetting

3.3 PIN CODE

A PIN code can be up to 16 characters. If the PIN or passkey is incorrect, any connection requirement will be rejected by 1800 RFID Reader. By default, the PIN code value is "0000".

RFIDBTGetPinCode	
Purpose	Get the current *Bluetooth*® PIN Code.
Syntax	**INT RFIDBTGetPinCode (BYTE** **Pin***)**
Parameters	**BYTE** **Pin* PIN Code
Example	BYTE pin [16]; memset (pin, 0x00, sizeof (pin)); if (RFIDBTGetPinCode (pin)==1) printf ("PIN code: %s", pin);
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	
See Also	RFIDBTSetPinCode

RFIDBTSetPinCode	
Purpose	Set *Bluetooth*® PIN Code.
Syntax	**INT RFIDBTSetPinCode (BYTE** **Pin***)**
Parameters	**BYTE** **Pin* PIN Code PIN : 1~16 Byte characters, 1 Byte =0x00 indicates PIN is not set. (**default="0000"**)
Example	BYTE *pin = "1234"; if (RFIDBTSetPinCode (pin)==1) printf ("Set PIN code is 1234."); Return Value If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDBTGetPinCode

Note: When the authentication and PIN code are changed on the 1800 RFID Reader, you have to remove the 1800 RFID Reader from the paired device list (called unpairing) and go through the whole process to re-establish the connection.

Chapter 4

RFID

IN THIS CHAPTER

4.1 TRIGGER SWITCH

Equipped with a reader switch, 1800 RFID Reader allows you to switch between RFID and Alternate Mode.

RFIDGetTriggerSwitch	
Purpose	Get current trigger switch status.
Syntax	**INT RFIDGetTriggerSwitch (INT** **RFID_Switch****);**
Parameters	**INT** **RFID_Switch* Trigger switch status. 0 – RFID Power Off. Switch is set to EXT. 1800 is in Alternate mode. 1 – RFID Power On. Switch is set to RFID. 1800 is in RFID mode.

Example	`INT RFID_switch;`
	`if (RFIDGetTriggerSwitch (&RFID_switch)==1)`
	`printf ("Trigger switch status: %d" , RFID_switch);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

4.2 SCAN MODE

RFIDGetScanMode

Purpose	Get current scan mode.
Syntax	**INT RFIDGetScanMode (INT** **ScanMod***);**
Parameters	**INT** **ScanMode* Scan mode.

Example	`INT scanMode;`
	`if (RFIDGetScanMode (&scanMode)==1)`
	`  printf ("Scan Mode: %d" , scanMode);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetScanMode

RFIDSetScanMode

Purpose	Set scan mode.
Syntax	**INT RFIDSetScanMode (INT** *ScanMode***);**
Parameters	**INT** *ScanMode* Scan mode 0x06 - Single Mode(**default**) 0x09 - Multi-Tag Mode
Example	`if (RFIDSetScanSettings (0x06)==1)` `  printf ("Set single mode.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetScanMode

Scan Mode Description:

<table>
<tr><th>Scan Mode</th><th colspan="2">Behavior</th></tr>
<tr><td>Single Mode</td><td colspan="2">To Read Tag Once.
1. Condition specified to start the operation: Press and hold the trigger key.
2. Conditions specified to stop the operation：
(1) A tag is read
(2) Trigger key is released.
(3) “Scan Session Timeout” expires while no Tag data is received
(4) New scan mode is set.
3. Release trigger key and press it again to start a new operation cycle. Scanning Timeout will be refreshed.</td></tr>
<tr><td rowspan="2">Multi-Tag Mode</td><td>Multi-Tag counter=0</td><td>To Read Tag continuously
1. Condition specified to start the operation: Press and hold the trigger key.
2. Conditions specified to stop the operation:
(1) Trigger key is released.
(2) New scan mode is set.
3. Scanning speed is controlled by Scanning Delay</td></tr>
<tr><td>Multi-Tag counter≠0</td><td>To Read Tag continuously. Repeated Tags will be ignored and new tag’s EPC will be recorded with the counter increased.
1. Condition to start/continue the operation: Press and hold the trigger key.
2. Condition specified to suspend the operation：
(1) Release trigger key.
3. Conditions specified to stop the operation:
(1) The amount of new Tag is equal to Multi-Tag counter.
(2) New Multi-Tag Counter is set.
(3) New scan mode is set.
4. The counter of read tag can be reset with command and function key.
5. Scanning speed is controlled by Scanning Delay</td></tr>
</table>

4.3 SCANNING TIMEOUT

You have to specify the scanning timeout interval (0~254 sec.; 0= Disable) when the scan mode is set to Single Mode.

- Operation will stop if the operation time equals Scan Session Timeout and No Tag data is received.
- The timeout value is in the range of 0~254 in seconds. When the timeout is set to '0', the operation will not stop.

RFIDGetScanningTimeout	
Purpose	Get current scanning timeout.
Syntax	**INT RFIDGetScanningTimeout (INT** **ScanningTimeout***);**
Parameters	**INT** **ScanningTimeout* Scanning timeout.

Example	`INT scanningTimeout;`
	`if (RFIDGetScanningTimeout (&scanningTimeout)==1)`
	`printf ("Timeout: %d" , scanningTimeout);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetScanningTimeout

RFIDSetScanningTimeout	
Purpose	Set scanning timeout.
Syntax	**INT RFIDSetScanningTimeout (INT** *ScanningTimeout***);**
Parameters	**INT** *ScanningTimeout* Scanning timeout. 0~254 (sec) (**default=0**)
Example	`if (RFIDSetScanningTimeout (0)==1)` `printf ("Set scanning timeout of 0 sec.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	1. Operation stops if operation time equals Scan Session Timeout and No Tag data is received. 2. The timeout value is in the range of 0~254 in seconds. When it is set to 0, the operation will not stop. 3. Operation time will not be refreshed if a new timeout value is set.
See Also	RFIDGetScanningTimeout

4.4 SCANNING DELAY TIME

You can set 1800 RFID Reader to scan continuously or set an interval time between each scan. Specify the scanning delay time when the scan mode is set to Multi-Tag Mode.

RFIDGetScanningDelayTime

Purpose	Get current scanning delay.
Syntax	**INT RFIDGetScanningDelayTime (INT** **ScanningDelayTime***);**
Parameters	**INT** **ScanningDelayTime* Scanning delay.

Example	`INT  scanningDelayTime;`
	`if (RFIDGetScanningDelay (&scanningDelayTime)==1)`
	`  printf ("Delay Time: %d", scanningDelayTime);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetScanningDelay

RFIDSetScanningDelayTime

Purpose	Set scanning delay.
Syntax	**INT RFIDSetScanningDelayTime (INT** *ScanningDelayTime***);**
Parameters	**INT** *ScanningDelayTime* Scanning delay.

	1861	1862
value	Delay Time	
0	100ms (**default**)	20ms
1	200ms	50ms (**default**)
2	400ms	100ms
3	800ms	250ms
4	1 sec	500ms
5	2 sec	---
6	3 sec	---
7	5 sec	---

Example	`if (RFIDSetScanningDelayTime (0)==1)` `  printf ("Set 1861 scanning delay to 100ms. ");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetScanningDelay

4.5 MULTI-TAG

RFIDGetMultiTagModeSetting

Purpose	Get current multi-tag mode setting.
Syntax	**INT RFIDGetMultiTagModeSetting (INT** **Count***);**
Parameters	**INT** **Count* Counter limit for unique tag read in Multi-Tag

Example	`INT count;`
	`if (RFIDGetMultiTagModeSetting (&count)==1)`
	`  printf ("Counter limit for unique tag read: %d", count);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetMultiTagModeSetting

RFIDSetMultiTagModeSetting

Purpose	Set multi-tag mode setting.
Syntax	**INT RFIDSetMultiTagModeSetting (INT** *Count***);**
Parameters	**INT** *Count* Counter limit for unique tag read in Multi-Tag 0, 1~128. (**default=128**)
Example	`if (RFIDSetMultiTagModeSetting (128)==1)` `  printf ("Set multi-tag mode settings.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	1. Set a non-zero Multi-Tag Counter to create a tag list in the 1800. If scan mode is set to Multi-Tag mode, any received tag will be compared to the content of the list. The tag of which EPC is not repeated will be accepted and stored in the list. Otherwise, it will be ignored. When the tag list is full, scan session completes and stops. 2. The Set Multi-Tag Counter command will clear the tag list to start a new session. If the counter is set to 0, any received tag will be accepted.
See Also	RFIDGetMultiTagModeSetting

4.6 RFID FUNCITON

RFIDGetAccessMode

Purpose	Get current access mode.
Syntax	**INT RFIDGetAccessMode (INT** **AccessMode*);**
Parameters	**INT** **AccessMode* Access Mode.

Example	`INT  AccessMode;`
	`if (RFIDGetAccessMode (&AccessMode)==1)`
	`  printf ("Access Mode: %d", AccessMode);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetAccessMode

RFIDSetAccessMode

Purpose	Set Access Mode.
Syntax	**INT RFIDSetAccessMode (INT** *AccessMode*);**
Parameters	**INT** *AccessMode* Access Mode. 0x00-Inventory (**default**) 0x01-Read Tag Memory 0x02-Write Tag Memory
Example	`if (RFIDSetAccessMode (0x01)==1)` `  printf ("Set Access Mode of Read tag memory.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetAccessMode

RFID Function Description

<table>
<tr><th>RFID Function</th><th>Description</th></tr>
<tr><td>Inventory</td><td>1. Get Tag's EPC Data (PC+EPC+CRC)
2. Supported by all scan modes
3. Affected by following parameters
<table>
<tr><th rowspan="2">Parameter</th><th colspan="2">Scan Mode</th></tr>
<tr><th>Single</th><th>Multi-Tag</th></tr>
<tr><td>Scanning Timeout</td><td>•</td><td></td></tr>
<tr><td>Scanning Delay</td><td></td><td>•</td></tr>
<tr><td>Multi-Tag Counter</td><td></td><td>•</td></tr>
<tr><td>Filter - Included EPC</td><td>•</td><td>•</td></tr>
<tr><td>Filter - Excluded EPC</td><td>•</td><td>•</td></tr>
</table></td></tr>
<tr><td>Read Tag Memory</td><td>1. Read data from specified memory bank of Tag.
2. Supported by all scan modes
3. Affected by following parameters
<table>
<tr><th rowspan="2">Parameter</th><th colspan="2">Scan Mode</th></tr>
<tr><th>Single</th><th>Multi-Tag</th></tr>
<tr><td>Scanning Timeout</td><td>•</td><td></td></tr>
<tr><td>Scanning Delay</td><td></td><td>•</td></tr>
<tr><td>Multi-Tag Counter</td><td></td><td>•</td></tr>
<tr><td>Filter - Included EPC</td><td>•</td><td>•</td></tr>
<tr><td>Filter - Excluded EPC</td><td>•</td><td>•</td></tr>
<tr><td>Access Password</td><td>•</td><td>•</td></tr>
<tr><td>Access Memory Bank</td><td>•</td><td>•</td></tr>
<tr><td>Access Start Address</td><td>•</td><td>•</td></tr>
<tr><td>Access Data Length</td><td>•</td><td>•</td></tr>
</table></td></tr>
<tr><td>Write Tag Memory</td><td>1. Write data to specified memory bank of Tag.
2. Only Trigger mode and Alternate mode support this function.
3. Affected by following parameters
<table>
<tr><th rowspan="2">Parameter</th><th colspan="2">Scan Mode</th></tr>
<tr><th>Single</th><th>Multi-Tag</th></tr>
<tr><td>Scanning Timeout</td><td>•</td><td></td></tr>
<tr><td>Scanning Delay</td><td></td><td>•</td></tr>
<tr><td>Multi-Tag Counter</td><td></td><td>•</td></tr>
<tr><td>Filter - Included EPC</td><td>•</td><td>•</td></tr>
</table></td></tr>
</table>

	Filter - Excluded EPC	•	•
	Access Password	•	•
	Access Memory Bank	•	•
	Access Start Address	•	•
	Access Data Length	•	•
	Data Buffer for Writing	•	•

4.7 TAG ACCESS

RFIDGetTagAccessParameter

Purpose	Get current access parameters.
Syntax	**INT RFIDGetTagAccessParameter (BYTE** **AP*, **INT** **MB*, **INT** **SA*, **INT** **DL*);
Parameters	**BYTE** **AP* Access Password. **INT** **MB* Memory Bank. **INT** **SA* Starting Address Byte pointer. **NT** **DL* Data Length Byte to Access.

Example	`INT MB, SA, DL;`
	`BYTE AP [5];`
	`if (RFIDGetTagAccessParameter (AP, &MB, &SA, &DL)==1)`
	`{`
	`printf ("AccPwd: %s\n", AP);`
	`printf ("Memory Bank: %d\n", MB);`
	`printf ("Start Addr: %d\n", SA);`
	`printf ("Data Length: %d\n", DL);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetTagAccessParameter

RFIDSetTagAccessParameter

Purpose: Set access parameters.

Syntax:
INT RFIDSetTagAccessParameter (BYTE **AP*,
INT *MB*,
INT *SA*,
INT *DL*);

Parameters:

BYTE **AP*

Access Password. (**default=null**) (0~4 Byte(s))

INT *MB*

Memory Bank. Value 0~3

value	Bank
0	Reserved Bank
1	EPC Bank (**default**)
2	TID
3	User

INT *SA*

Starting Address Byte pointer. (**default=0**)

NT *DL*

Data Length Byte to Access. (**default=0**)

Example:

```
if (RFIDSetTagAccessParameter ((BYTE*)“1234”, 1, 4, 20)==1)
{
    printf ("Set Access password of 1234.\n");
    printf ("Set Memory bank of EPC bank.\n");
    printf ("Set Starting address byte pointer of 4.\n");
    printf ("Set Data length of 20 bytes.\n");
}
```

Return Value: If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks: Starting Address (SA) and Data Length (DL) must be Even numbers.

See Also: RFIDGetTagAccessParameter

4.8 DATA IN BUFFER

RFIDGetDataInBufferForWrite

Purpose	Get current data in buffer.
Syntax	**INT RFIDGetDataInBufferForWrite (INT** **DL*, **BYTE** **BUF*);
Parameters	**INT** **DL* Data Length Byte. **BYTE** **BUF* Data stored in buffer for writing.

Example	`INT DL, i;`
	`BYTE BUF [33];`
	`if (RFIDGetDataInBufferForWrite (&DL, BUF)==1)`
	`{`
	`  printf ("Data Length: %d\n", DL);`
	`  printf ("Data:);`
	`  for (i=0;i<DL;i++) printf ("%c", BUF [i]);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetDataInBufferForWrite

RFIDSetDataInBufferForWrite

Purpose	Set data in buffer.
Syntax	**INT RFIDSetDataInBufferForWrite (INT** *DL*, **BYTE** **BUF*);
Parameters	**INT** *DL* Data Length Byte. (**default=0**) **BYTE** **BUF* Data stored in buffer for writing. 0~32 Bytes (**default=null**)
Example	`if (RFIDSetDataInBufferForWrite (6, (BYTE*)"123ABC")==1)` `  printf ("Set data is 123ABC.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Data Length (DL) can only be set to an Even number
See Also	RFIDGetDataInBufferForWrite

4.9 FILTER

RFIDFilterGetIncludeEPC

Purpose	Get current included EPC filter setting.
Syntax	**INT RFIDFilterGetIncludeEPC (INT** **Enable*, **INT** **EPCScheme*, **INT** **SBIT*, **INT** **BL*, **BYTE** **EPC1*, **BYTE** **EPC2*);
Parameters	**INT** **Enable* Enable/Disable Filter. 0 – Disable (**default**), 1 – Enable range filter **INT** **EPCScheme Activated Scheme for EPC Filter **INT** **SBIT* Start bit of EPC. (0~255) **INT** **BL* Bits of pattern length. (0~255) **BYTE** **EPC1*, **EPC2* Included EPC pattern

Example	`INT enable, EPCScheme, SBit, BL;`
	`BYTE EPC1 [33], EPC2 [33];`
	`if (RFIDFilterGetIncludeEPC (&enable, &EPCScheme, &SBit, &BL, EPC1, EPC2)==1)`
	`{`
	`  printf ("Enable: %d\n", enable);`
	`  printf ("EPC Scheme: 0x%2X\n", EPCScheme);`
	`  printf ("Start bit: %d\n", SBit);`
	`  printf ("Length of pattern bits: %d\n", BL);`
	`  printf ("EPC1: %s\n", EPC1);`
	`  printf ("EPC2: %s\n", EPC2);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remark

If Range Filter is set to “1” (enabled), it indicates the tag which EPC is in the range of EPC1~EPC2 will be accepted (EPC Pattern1 ≤ Tag EPC ≤ EPC Pattern2). The length of pattern bits (BL) is shared with EPC1 and EPC2

The EPC Scheme parameter is shared with included and excluded EPC filters. If both filters are used simultaneously, be sure to set the same EPC pattern.

Note: EPC Scheme

The value of EPC Scheme and meaning.

0x2C	GDTI-96
0x2D	GSRN-96
0x2F	USDoD-96
0x30	SGTIN-96 **(default)**
0x31	SSCC-96
0x32	SGLN-96
0x33	GRAI-96
0x34	GIAI-96
0x35	GID-96
0x36	SGTIN-198
0x37	GRAI-170
0x38	GIAI-202
0x39	SGLN-195
0x3A	GDTI-113
0x3B	ADI

See Also

RFIDFilterSetIncludedEPC

RFIDFilterSetIncludeEPC	
Purpose	Set included EPC filter setting.
Syntax	**INT RFIDFilterSetIncludeEPC (INT** *Enable*, **INT** EPCScheme, **INT** SBIT, **INT** BL, **BYTE** *EPC1, **BYTE** *EPC2);
Parameters	**INT** **Enable* Enable/Disable Filter. 0 – Disable (**default**), 1 – Enable range filter **INT** *EPCScheme Activate Scheme for EPC Filter **INT** *SBIT* Start bit of EPC. (**default=0**) (0~255) **INT** *BL* Length of pattern bits. (**default=0**) (0~255) **BYTE** **EPC1, *EPC2* Include EPC pattern. max=32 Byte(**default=null**)

Example	`if (RFIDFilterSetIncludeEPC (1, 0x2F, 12, 48, (BYTE*)"230585", (BYTE*) "230590")==1)`
	`{`
	`printf ("Enable range of the included EPC pattern filter.\n");`
	`printf ("Activate USDoD-96 tag type for EPC filter.\n");`
	`printf ("Start bit of EPC is 12.\n");`
	`printf ("Pattern length is 48 bits.\n");`
	`printf ("Included EPC pattern range is 230585~230590.\n");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

SBIT + BL must be less than or equal to 256.

If Range Filter is set to “1” (enabled), it indicates the tag which EPC is in the range of EPC1~EPC2 will be accepted (EPC Pattern1 ≤ Tag EPC ≤ EPC Pattern2). The length of pattern bits (BL) is shared with EPC1 and EPC2.

The EPC Scheme parameter is shared with included and excluded EPC filters. If both filters are used simultaneously, be sure to set the same EPC pattern.

Note: EPC Scheme

The value of EPC Scheme and meaning.

0x2C	GDTI-96
0x2D	GSRN-96
0x2F	USDoD-96
0x30	SGTIN-96 **(default)**
0x31	SSCC-96
0x32	SGLN-96
0x33	GRAI-96
0x34	GIAI-96
0x35	GID-96
0x36	SGTIN-198
0x37	GRAI-170
0x38	GIAI-202
0x39	SGLN-195
0x3A	GDTI-113
0x3B	ADI

See Also

RFIDFilterGetIncludeEPC

RFIDFilterGetExcludeEPC

Purpose	Get current excluded EPC filter setting.
Syntax	**INT RFIDFilterGetExcludeEPC (INT** **Enable*, **INT** **EPCScheme*, **INT** **SBIT*, **INT** **BL*, **BYTE** **EPC1*, **BYTE** **EPC2*);
Parameters	**INT** **Enable* Enable/Disable Filter. 0 – Disable (**default**), 1 – Enable range filter **INT** *EPCScheme Activated Scheme for EPC Filter **INT** **SBIT* Start bit of EPC. (0~255) **INT** **BL* Length of pattern bits. (0~255) **BYTE** **EPC1, *EPC2* Excluded EPC pattern.

Example	`INT enable, EPCScheme, SBit, BL;`
	`BYTE EPC1 [33], EPC2 [33];`
	`if (RFIDFilterGetExcludeEPC (&enable, &EPCScheme, &SBit, &BL, EPC1, EPC2)==1)`
	`{`
	`  printf ("Enable: %d\n", enable);`
	`  printf ("EPC Scheme: 0x%2X\n", EPCScheme);`
	`  printf ("Start bit: %d\n", SBit);`
	`  printf ("Length of pattern bits: %d\n", BL);`
	`  printf ("EPC1: %s\n", EPC1);`
	`  printf ("EPC2: %s\n", EPC2);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remark

If Range Filter is set to “1” (enabled), it indicates the tag which EPC is in the range of EPC1~EPC2 will be rejected. The length of pattern bits (BL) is shared with EPC1 and EPC2.

The EPC Scheme parameter is shared with included and excluded EPC filters. If both filters are used simultaneously, be sure to set the same EPC pattern.

Note: EPC Scheme

The value of EPC Scheme and meaning.

0x2C	GDTI-96
0x2D	GSRN-96
0x2F	USDoD-96
0x30	SGTIN-96 **(default)**
0x31	SSCC-96
0x32	SGLN-96
0x33	GRAI-96
0x34	GIAI-96
0x35	GID-96
0x36	SGTIN-198
0x37	GRAI-170
0x38	GIAI-202
0x39	SGLN-195
0x3A	GDTI-113
0x3B	ADI

See Also

RFIDFilterSetExcludeEPC

RFIDFilterSetExcludeEPC	
Purpose	Set excluded EPC filter setting.
Syntax	**INT RFIDFilterSetExcludeEPC (INT** *Enable*, **INT** *EPCScheme*, **INT** *SBIT*, **INT** *BL*, **BYTE** **EPC1*, **BYTE** **EPC2*);
Parameters	**INT** **Enable* Enable/Disable Filter. 0 – Disable (**default**), 1 – Enable range filter **INT** *EPCScheme Activate Scheme for EPC Filter **INT** *SBIT* Start bit of EPC. (**default=0**) (0~255) **INT** *BL* Length of pattern bits. (**default=0**) (0~255) **BYTE** **EPC1, *EPC2* Excluded EPC pattern. max=32 Byte (**default=null**)

Example	`if (RFIDFilterSetExcludeEPC (1, 0x2F, 12, 48, (BYTE*)"230585", (BYTE*)"230590")==1)`
	`{`
	`printf ("Enable range of the excluded EPC pattern filter.\n");`
	`printf ("Activate USUDoD-96 tag type for EPC filter.\n");`
	`printf ("Start bit of EPC is 12.\n");`
	`printf ("Pattern length is 48 bits.\n");`
	`printf ("Excluded EPC pattern range is 230585~230590.\n");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

SBIT + BL must be less than or equal to 256.

If Range Filter is set to “1” (enabled), EPC2 is needed. It indicates the tag which EPC is in the range of EPC1~EPC2 will be rejected. The length of pattern bits (BL) is shared with EPC1 and EPC2.

The EPC Scheme parameter is shared with included and excluded EPC filters. If both filters are used simultaneously, be sure to set the same EPC pattern.

Note: EPC Scheme

The value of EPC Scheme and meaning.

0x2C	GDTI-96
0x2D	GSRN-96
0x2F	USDoD-96
0x30	SGTIN-96 **(default)**
0x31	SSCC-96
0x32	SGLN-96
0x33	GRAI-96
0x34	GIAI-96
0x35	GID-96
0x36	SGTIN-198
0x37	GRAI-170
0x38	GIAI-202
0x39	SGLN-195
0x3A	GDTI-113
0x3B	ADI

See Also

RFIDFilterGetExcludeEPC

RFIDGetTagTypeActivation

Purpose

Get current the setting of accepted EPC tag type.

Syntax

INT RFIDGetTagTypeActivation (BYTE **Scheme1*,
BYTE **Scheme2*,
BYTE **Scheme3*,
BYTE **Scheme4*);

Parameters

BYTE *Scheme1~*Scheme4

Accepted EPC encoding scheme. Each bit represents an individual encoding scheme.

Example

```
BYTE Scheme1, Scheme2, Scheme3, Scheme4;
if (RFIDGetTagTypeActivation (&Scheme1, &Scheme2, &Scheme3, &Scheme4)==1)
{
   printf ("Get current accepted EPC tag type");
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

	Bit	Meaning
Scheme 1	0	Accept EPC encoded by GDTI-96
	1	Accept EPC encoded by GSRN-96
	2	Accept EPC encoded by USDoD-96
	3	Accept EPC encoded by SGTIN-96
	4	Accept EPC encoded by SSCC-96
	5	Accept EPC encoded by SGLN-96
	6	Accept EPC encoded by GRAI-96
	7	Accept EPC encoded by GIAI-96
Scheme 2	0	Accept EPC encoded by GID-96
	1	Accept EPC encoded by SGTIN-198
	2	Accept EPC encoded by GRAI-170
	3	Accept EPC encoded by GIAI-202
	4	Accept EPC encoded by SGLN-195
	5	Accept EPC encoded by GDTI-113
	6	Accept EPC encoded by ADI
	7	Reserved. Always read and write as 1
Scheme 3		Reserved. Always read and write as 0xFF
Scheme 4		Reserved. Always read and write as 0xFF

See Also

RFIDSetTagTypeActivation

RFIDSetTagTypeActivation

Purpose	Set the setting of accepted EPC tag type.
Syntax	**INT RFIDSetTagTypeActivation (BYTE** *Scheme1*, **BYTE** *Scheme2*, **BYTE** *Scheme3*, **BYTE** *Scheme4*);
Parameters	**BYTE** Scheme1~Scheme4 Accepted EPC encoding scheme. Each bit represents an individual encoding scheme. The vlues of Scheme 1~4 set to 0xFF means all tags are accepted. Otherwise, the reader only accepts the tag which EPC is encoded by the specific scheme if the corresponding bit is set to 1. (**default: Scheme 1~4 values are set to 0xFF**)

Example	`if (RFIDSetTagTypeActivation ((BYTE*)"11111111", (BYTE*) "11111111", (BYTE*) "11111111", (BYTE*) "11111111")==1)`
	`{`
	`printf ("Set all accepted EPC tag type");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

	Bit	Meaning
Scheme 1	0	Accept EPC encoded by GDTI-96
	1	Accept EPC encoded by GSRN-96
	2	Accept EPC encoded by USDoD-96
	3	Accept EPC encoded by SGTIN-96
	4	Accept EPC encoded by SSCC-96
	5	Accept EPC encoded by SGLN-96
	6	Accept EPC encoded by GRAI-96
	7	Accept EPC encoded by GIAI-96
Scheme 2	0	Accept EPC encoded by GID-96
	1	Accept EPC encoded by SGTIN-198
	2	Accept EPC encoded by GRAI-170
	3	Accept EPC encoded by GIAI-202
	4	Accept EPC encoded by SGLN-195
	5	Accept EPC encoded by GDTI-113
	6	Accept EPC encoded by ADI
	7	Reserved. Always read and write as 1
Scheme 3		Reserved. Always read and write as 0xFF
Scheme 4		Reserved. Always read and write as 0xFF

See Also	RFIDGetTagTypeActivation

The greater power level value is specified for the stronger signal. By default, the level is set to '3' for the strongest broadcast.

RFIDGetPowerLevel	
Purpose	Get current RFID power level.
Syntax	**INT RFIDGetPowerLevel (INT** **Level****);**
Parameters	**INT** **Level*
	RFID module Power Level.

Example	`INT level;`
	`if (RFIDGetPowerLevel (&level)==1)`
	`  printf ("Power Level: %d ", level);`

Return Value	If successful, it returns 1.
	If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetPowerLevel

RFIDSetPowerLevel	
Purpose	Set RFID power level.
Syntax	**INT RFIDSetPowerLevel (INT** *Level****);**
Parameters	**INT** *Level*
	RFID module Power Level.
	Valid range=0~3 (**default=3**)
Example	`if (!RFIDSetPowerLevel (3))` `  printf ("RFID module power level of 3.");`
Return Value	If successful, it returns 1.
	If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetPowerLevel

Chapter 5

OUTPUT

IN THIS CHAPTER

5.1 DATA FORMAT

You have to define the output data format firstly before capturing the data transmission from 1800 RFID Reader. There are three output formats specified to various *Bluetooth*® interfaces.

RFIDGetOutputDataFormat

Purpose	Get current output data format.
Syntax	**INT RFIDGetOutputDataFormat (INT** **Format***);**
Parameters	**INT** **Format* Output Data Format.

Example	`INT  FMT;`
	`if (RFIDGetOutputDataFormat (&FMT)==1))`
	`  printf ("Data Format: %d " , FMT);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetOutputDataFormat

RFIDSetOutputDataFormat

Purpose	Set output data format.
Syntax	**INT RFIDSetOutputDataFormat (INT** *Format***);**
Parameters	**INT** *Format* Output Data Format.

Format	BT SPP
0 (default)	Packet Data
1	Hexadecimal
2	Raw Data

Example	`if (RFIDSetOutputDataFormat (1)==1)` `  printf ("Output Data Format of Hexadecimal.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetOutputDataFormat

5.2 OUTPUT DATA SEQUENCE

RFIDGetOutputDataSequence	
Purpose	Get current output data sequence.
Syntax	**INT RFIDGetOutputDataSequence (INT** **Section1*, **INT** **Section2*, **INT** **Section3*);
Parameters	**INT** **Section1* Data Section output in sequence1. **INT** **Section2* Data Section output in sequence2. **INT** **Section3* Data Section output in sequence3.

Example	`INT section1, section2, section3;`
	`if (RFIDGetOutputDataSequence (&section1, &section2, &section3)==1)`
	`{`
	`  printf ("Sequence 1: %d\n", section1);`
	`  printf ("Sequence 2: %d\n", section2);`
	`  printf ("Sequence 3: %d\n", section3);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data.
See Also	RFIDSetOutputDataSequence

RFIDSetOutputDataSequence	
Purpose	Set output data sequence.
Syntax	**INT RFIDSetOutputDataSequence (INT** *Section1*, **INT** *Section2*, **INT** *Section3*);
Parameters	**INT** *Section1* Data Section output in sequence1. (**default=1**) **INT** *Section2* Data Section output in sequence2. (**default=2**) **INT** *Section3* Data Section output in sequence3. (**default=3**) **Data Section:** 0 – Disable 1 – Counter Section 2 – Timestamp Section 3 – EPC Tag Section

Example	`if (RFIDSetOutputData Sequence (1, 2, 0)==1)`
	`{`
	`printf ("Data Section output in sequence1 of Counter Section.\n");`
	`printf ("Data Section output in sequence2 of Timestamp Section.\n");`
	`printf ("Disable Data Section output in sequence3.\n");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data. Each section is individual and at least one section has to be enabled.
See Also	RFIDGetOutputDataSequence

5.3 DATA COUNTER

There are three events supported to reset the counter. When the parameter is '1', the output data will be counted from the beginning for each event occurrence.

RFIDGetDataCounterContent	
Purpose	Get the setting of current counter section.
Syntax	**INT RFIDGetDataCounterContent (BYTE** **PAD*, **INT** **GetResetCounterCMD*, **INT** **UHFPowerON*, **INT** **NewConnection*,
Parameters	**INT** **PAD* Character to be padded before data counter. **INT** **GetResetCounterCMD* Reset data counter events when getting Reset Conter Command. **INT** **UHFPowerOn* Reset data counter events when powering on UHF. **INT** **NewConnection* Reset data counter events when establishing a new connection.

Example	`BYTE PAD;`
	`INT GetResetCounterCMD , UHFPowerOn, NewConnection;`
	`if (RFIDGetDataCounterContent (&PAD, &GetResetCounterCMD , &UHFPowerOn, &NewConnection)==1)`
	`{`
	`printf ("PAD: %c\n", PAD);`
	`printf ("Reset events when Get Reset Counter Command: %d\n", GetResetCounterCMD);`
	`printf ("Reset events when UHF Power On: %d\n", UHFPowerOn);`
	`printf ("Reset events when New Connection: %d\n", NewConnection);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data.
See Also	RFIDSetDataCounterContent

RFIDSetDataCounterContent	
Purpose	Set the content of counter section.
Syntax	**INT RFIDSetDataCounterContent (BYTE** *PAD*, **INT** *GetResetCounterCMD*, **INT** *UHFPowerOn*, **INT** *NewConnection*);
Parameters	**INT** *PAD* Character to be padded before data counter. **(default=0x20(space))** **INT** *GetResetCounterCMD* Reset data counter events when getting Reset Conter Command. 0 – Disable, 1 – Enable **(default)** **INT** *UHFPowerOn* Reset data counter events when powering on UHF. 0 – Disable **(default)**, 1 – Enable **INT** *NewConnection* Reset data counter events when establishing a new connection. 0 – Disable, 1 – Enable **(default)**

Example	`if (RFIDSetDataCounterContent (0x20, 1,0,1)==1)`
	`{`
	`printf ("Padded before data counter of space character.\n");`
	`printf ("Reset data counter events when Get Reset Counter Command and New Connection.\n");`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data.
See Also	RFIDGetDataCounterContent

5.4 RESET DATA COUNTER

RFIDResetDataCounter	
Purpose	Reset data counter.
Syntax	**INT RFIDResetDataCounter (void)**
Example	`if (RFIDResetDataCounter ()==1)` `  printf ("Reset data counter.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data.

Note: Data Counter will always be reset when system powers up.

5.5 TIMESTAMP

RFIDGetTimestampContent	
Purpose	Get the settings of current timestamp.
Syntax	**INT RFIDGetTimestampContent (INT** **FD1*, **INT** **FD2*, **INT** **FD3*, **INT** **FD4*, **INT** **FD5*, **INT** **FD6*, **INT** **FD7*, **BYTE** **SC1*, **BYTE** **SC2*, **BYTE** **SC3*, **BYTE** **SC4*, **BYTE** **SC5*, **BYTE** **SC6*, **INT** **YD*, **INT** **MS*);
Parameters	**INT** **FD1* ~ *FD7 Time Information in field1~field7. **BYTE** **SC1* ~ **SC6* Separating character between field1~field7. **INT** **YD* Digits of year. **INT** **MS* The 'Second' field displays in milliseconds.

Example

```
INT  FD1, FD2, FD3, FD4, FD5, FD6, FD7, YD, MS;
BYTE SC1, SC2, SC3, SC4, SC5, SC6;
if (RFIDGetTimestampContent (&FD1, &FD2, &FD3, &FD4, &FD5, &FD6, &FD7,
&SC1, &SC2, &SC3, &SC4, &SC5, &SC6, &YD, &MS)==1)
{
   printf ("Field 1: %d\n", FD1);
   printf ("Field 2: %d\n", FD2);
   printf ("Field 3: %d\n", FD3);
   printf ("Field 4: %d\n", FD4);
   printf ("Field 5: %d\n", FD5);
   printf ("Field 6: %d\n", FD6);
   printf ("Field 7: %d\n", FD7);
   printf ("Separator 1: %c\n", SC1);
   printf ("Separator 2: %c \n", SC2);
   printf ("Separator 3: %c \n", SC3);
   printf ("Separator 4: %c \n", SC4);
   printf ("Separator 5: %c \n", SC5);
   printf ("Separator 6: %c \n", SC6);
   printf ("Digits of Year: %d\n", YD);
   printf ("Show millisecond: %d\n", MS);
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

Only effective when Output Data Format is set to HEX and Raw Data.

See Also

RFIDSetTimestampContent

RFIDSetTimestampContent

Purpose: Set the contents of timestamp.

Syntax:

```
INT RFIDSetTimestampContent (INT FD1,
                             INT FD2,
                             INT FD3,
                             INT FD4,
                             INT FD5,
                             INT FD6,
                             INT FD7,
                             BYTE SC1,
                             BYTE SC2,
                             BYTE SC3,
                             BYTE SC4,
                             BYTE SC5,
                             BYTE SC6,
                             INT YD,
                             INT MS);
```

Parameters

INT *FD1* ~ FD7

Time Information in field1~field7.

Time Info	value
Disable	0
Year	1
Month	2
Day	3
Weekday	4
Hour	5
Minute	6
Second	7

BYTE *SC1 ~ SC6*

Separating character between field1~field7.

INT *YD*

Digits of year.

0 – 2 digits

1 – 4 digits, shown as “20xx”

INT *MS*

The ‘Second’ field displays in milliseconds.

0 – Disable, 1 – Enable

Example

```
if (RFIDSetTimestampContent (1, 2, 3, 0, 5, 6, 7,
0x2D, 0x2D, 0x20, 0x20, 0x3A, 0x20, 0, 1)==1)
{
    printf ("Set timestamp default setting. ");
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

	Default
FD1	1
FD2	2
FD3	3
FD4	0
FD5	5
FD6	6
FD7	7
SC1	0x2D (-)
SC2	0x2D (-)
SC3	0x20 (SPACE)
SC4	0x20 (SPACE)
SC5	0x3A (:)
SC6	0x20 (SPACE)
YD	0
MS	1

Only effective when Output Data Format is set to HEX and Raw Data

See Also

RFIDGetTimestampContent

5.6 DATA CONTENT

RFIDGetEPCTagContent

Purpose	Get settings of the current EPC Tag section.
Syntax	**INT RFIDGetEPCTagContent (INT** **FD1*, **INT** **FD2*, **INT** **FD3*, **INT** **FD4*, **INT** **FD5*, **BYTE** **SC1*, **BYTE** **SC2*, **BYTE** **SC3*, **BYTE** **SC4*, **INT** **DLType*);
Parameters	**INT** **FD1* ~ *FD5 EPC Tag in field1~field5. **BYTE** **SC1* ~ **SC4* Separating character between field1~field5. **INT** **DLType* Data Length Type.

Example	`INT FD1, FD2, FD3, FD4, FD5, DLType;`
	`BYTE SC1, SC2, SC3, SC4;`
	`if (RFIDGetEPCTagContent (&FD1, &FD2, &FD3, &FD4, &FD5,`
	`&SC1, &SC2, &SC3, &SC4, DLType)==1)`
	`{`
	`printf ("Field 1: %d\n", FD1);`
	`printf ("Field 2: %d\n", FD2);`
	`printf ("Field 3: %d\n", FD3);`
	`printf ("Field 4: %d\n", FD4);`
	`printf ("Field 5: %d\n", FD5);`
	`printf ("Separator 1: %c\n", SC1);`
	`printf ("Separator 2: %c \n", SC2);`
	`printf ("Separator 3: %c \n", SC3);`
	`printf ("Separator 4: %c \n", SC4);`
	`printf ("Data Length Type: %d\n", DLType);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data

See Also RFIDSetEPCTagContent

RFIDSetEPCTagContent

Purpose Set contents of the EPC Tag section.

Syntax **INT RFIDGetEPCTagContent (INT** *FD1*,
INT *FD2*,
INT *FD3*,
INT *FD4*,
INT *FD5*,
BYTE *SC1*,
BYTE *SC2*,
BYTE *SC3*,
BYTE *SC4*,
INT *DLType*);

Parameters **INT** *FD1* ~ FD5

EPC Tag in field1~field5.

EPC Tag	value
Disable	0
CRC	1
PC	2
EPC	3
Memory Data	4
Data Length	5

BYTE *SC1* ~ *SC4*

Separating character between field1~field5.

INT *DLType*

Data Length Type.

0 – Total Length (PC+EPC+CRC{+Memory Data})

1 – EPC Length

2 – Memory Data Length

Example
```
if (RFIDSetEPCTagContent (2, 3, 1, 4, 0, 0x00, 0x00, 0x00, 0x00, 0)==1)
    printf ("Set EPC Tag section default setting.");
```

Return Value If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

	Default
FD1	2
FD2	3
FD3	1
FD4	4
FD5	0
SC1	0x00 (NULL)
SC2	0x00 (NULL)
SC3	0x00 (NULL)
SC4	0x00 (NULL)
DL	0

Only effective when Output Data Format is set to HEX and Raw Data

See Also RFIDGetEPCTagContent

5.7 PREFIX FORMAT

RFIDGetPrefixFormat	
Purpose	Get prefix.
Syntax	**INT RFIDGetPrefixFormat (INT** *PSEC*, **BYTE** **PFX*);
Parameters	**INT** *PSEC* Section. 1 – Counter Section 2 – Timestamp Section 3 – EPC Tag Section **BYTE** **PFX* Get prefix characters of selected section.

Example	`INT PSEC;`
	`BYTE PFX[9];`
	`if (RFIDGetPrefixFormat (1, PFX)==1)`
	`printf ("Get counter section prefix characters of %s, PFX");`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data
See Also	RFIDSetPrefixFormat

RFIDSetPrefixFormat

Purpose	Set prefix.
Syntax	**INT RFIDSetPrefixFormat (INT** *PSEC*, **BYTE** **PFX*);
Parameters	**INT** *PSEC* Section. 1 – Counter Section 2 – Timestamp Section 3 – EPC Tag Section **BYTE** **PFX* Set prefix characters of selected section. Max length is 8 Bytes. Set one Byte as 0x00 to disable the prefix.
Example	`if (RFIDSetPrefixFormat (1, (BYTE*)"AB")==1)` `  printf ("Set counter section prefix characters of AB. ");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	See table below Only effective when Output Data Format is set to HEX and Raw Data
See Also	RFIDGetPrefixFormat

Section	Default
Prefix of Counter Section	0x00 (NULL)
Prefix of Timestamp Section	0x20 (SPACE)
Prefix of EPC Tag Section	0x00 (NULL)

5.8 SUFFIX FORMAT

RFIDGetSuffixFormat	
Purpose	Get suffix.
Syntax	**INT RFIDGetSuffixFormat (INT** *SSEC*, **BYTE** **SFX*);
Parameters	**INT** *SSEC* Section. 1 – Counter Section 2 – Timestamp Section 3 – EPC Tag Section **BYTE** **SFX* Get suffix characters of selected section.

Example	`INT SSEC;`
	`BYTE SFX[9];`
	`if (RFIDGetSuffixFormat (2, SFX)==1)`
	`  printf ("Get timestamp section suffix characters of %s, SFX");`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	Only effective when Output Data Format is set to HEX and Raw Data
See Also	RFIDSetSuffixFormat

RFIDSetSuffixFormat

Purpose	Set suffix.
Syntax	**INT RFIDSetSuffixFormat (INT** *SSEC*, **BYTE** **SFX*);
Parameters	**INT** *SSEC* Section. 1 – Counter Section 2 – Timestamp Section 3 – EPC Tag Section **BYTE** **SFX* Set suffix characters of selected section. Max length is 8 Bytes. Set one Byte as 0x00 to disable the suffix.
Example	`if (RFIDSetSuffixFormat (2, (BYTE*)"CD")==1)` `  printf ("Set timestamp section suffix characters of CD.");`
Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	See table below. Only effective when Output Data Format is set to HEX and Raw Data
See Also	RFIDGetSuffixFormat

Section	Default
Suffix of Counter Section	0x2E (.)
Suffix of Timestamp Section	0x20 (SPACE)
Suffix of EPC Tag Section	0x0D (CR)

5.9 PROGRAMMABLE KEYS

RFIDGetProgrammableKey

Purpose	Get settings of current Programmable Keys.
Syntax	**INT RFIDGetProgrammableKey (INT** *StrID*, **INT** **Key*, **BYTE** **String*);
Parameters	**INT** *StrID* Output String Buffer ID. (Range:**1~6**) **INT** **Key* Get Action. **BYTE** **String* Get the string to be output via BT when specified key action occurs in Alternate mode. Max length is 10 Bytes.

Example	`INT Key;`
	`BYTE String [11]`
	`if (RFIDGetProgrammableKey (1, &Key, String)==1)`
	`{`
	`printf ("Get output string buffer 1\n");`
	`printf ("Key: %d\n", Key);`
	`printf ("String: %s\n", String);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetProgrammableKey

RFIDSetProgrammableKey

Purpose: Set settings of Programmable Keys.

Syntax:

```
INT RFIDSetProgrammableKey (INT StrID,
                            INT Key,
                            BYTE *String);
```

Parameters:

INT *StrID*

Set Output String Buffer ID. (Range:**1~6**)

INT *Key*

Action.

Key	Action
0	Disable
1	Trigger is pressed
2	Trigger is released
3	F1 is pressed
4	F1 is released
5	F2 is pressed
6	F2 is released
7	F1 + Trigger are pressed
8	F2 + Trigger are pressed

BYTE **String*

String to be output via BT when specified key action occurs in Alternate mode. Max length is 10 Bytes.

Set one Byte as 0x00 to clear the string buffer.

Example:

```
if (RFIDSetProgrammableKey (1, 1, (BYTE*)“TRIGON\r”)==1)
{
    printf ("Set output string buffer 1.\n");
    printf ("action of trigger is pressed.\n");
    printf ("String to be output of  “TRIGON\r”.\n");
}
```

Return Value: If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

	Default
String 1	“TRIGON\r”
String 2	“TRIGOFF\r”
String 3	0x00(NULL)
String 4	0x00(NULL)
String 5	0x00(NULL)
String 6	0x00(NULL)
KEY1	1
KEY2	2
KEY3	0
KEY4	0
KEY5	0
KEY6	0

See Also RFIDGetProgrammableKey

5.10 DATA OUTPUT VIA USB

RFIDGetDataOutputViaUSB

Purpose	Get the status of outputting data to USB virtual COM.
Syntax	**INT RFIDGetDataOutputViaUSB (INT** **Enable***);**
Parameters	**INT** **Enable* Enable/Disable outputting data to USB virtual COM.

Example	`INT enable;`
	`if (RFIDGetDataOutputViaUSB (&enable)==1)`
	`{`
	`  printf ("Output data via USB: %d\n", enable);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDSetDataOutputViaUSB

RFIDSetDataOutputViaUSB

Purpose	Set the status of outputting data to USB virtual COM.
Syntax	**INT RFIDSetDataOutputViaUSB (INT** *Enable***);**
Parameters	**INT** *Enable* Enable/Disable outputting data to USB virtual COM.

Example	`if (RFIDSetDataOutputViaUSB (1)==1)`
	`{`
	`  printf ("Enable data output to USB virtual COM.”);`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetDataOutputViaUSB

Chapter 6

HOST COMMAND

IN THIS CHAPTER

6.1 HOST MODE

RFIDGetHostMode

Purpose	Get the current Host Mode status.
Syntax	**INT RFIDGetHostMode (INT** **Status***);**
Parameters	**INT** **Status* Host Mode status

Example	`INT status;`
	`if (RFIDGetHostMode (&status)==1)`
	`printf ("Host mode status: %d", status);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Else refer to Status Codes.
See Also	RFIDSetHostMode

RFIDSetHostMode

Purpose	Set the Host Mode status.
Syntax	**INT RFIDSetHostMode (INT** *Status***);**
Parameters	**INT** *Status* Host Mode status

Value	Status
0	Normal Mode
1	Host Mode

Example	`INT status;`
	`if (RFIDSetHostMode (1)==1)`
	`printf ("Host Mode is enabled.”);`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
See Also	RFIDGetHostMode

6.2 INVENTORY EPC/TID

RFIDHostGetInventoryEPC

Purpose	Get inventory EPC in Host Mode.
Syntax	**INT RFIDHostGetInventoryEPC (INT** *Count***);**
Parameters	**INT** *Count* Operation count ranges from 0 to 255. Zero means to stop the operation.

Example	`INT status, value;`
	`if (RFIDHostGetInventoryEPC (10)==1)`
	`{`
	`printf ("Operation…”);`
	`}`
	`while (1)`
	`{`
	`value = RFIDGetHostMessage ();`
	`if (value == 0x01)`
	`{`
	`printf (“Get host inventory EPC.”);`
	`break;`
	`}`
	`else if (value == 0xFF)`
	`{`
	`printf (“Finish the operation.”);`
	`break;`
	`}`
	`else if (value == 0xF0)`
	`{`
	`printf (“Operation stops due to timeout.”);`
	`break;`
	`}`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	1. It works regardless of the settings of Scan Mode and RFID function. 2. Scanning Dealy is still effective even after Scan Session timeout expires. 3. Multi-tag Counter is not effective. 4. Accepted EPC Encoding Scheme and Select EPC/Eliminated EPC filters are still available.
See Also	RFIDGetHostMessage

RFIDHostGetInventoryTID	
Purpose	Get inventory EPC+TID in Host Mode.
Syntax	**INT RFIDHostGetInventoryTID (INT** *Count***);**
Parameters	**INT** *Count* Operation count ranges from 0 to 255. Zero means to stop the operation.

Example	`INT status, value;`
	`if (RFIDHostGetInventoryTID (10)==1)`
	`{`
	`  printf ("Operation…”);`
	`}`
	`while (1)`
	`{`
	`  value = RFIDGetHostMessage ();`
	`  if (value == 0x02)`
	`  {`
	`  printf (“Get host inventory EPC and TID.”);`
	`  break;`
	`  }`
	`  else if (value == 0xFF)`
	`  {`
	`  printf (“Finish the operation.”);`
	`  break;`
	`  }`
	`  else if (value == 0xF0)`
	`  {`
	`  printf (“Operation stops due to timeout.”);`
	`  break;`
	`  }`
	`}`

Return Value	If successful, it returns 1. If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.
Remarks	1. It works regardless of the settings of Scan Mode and RFID function. 2. Scanning Dealy is still effective even after Scan Session timeout expires. 3. Multi-tag Counter is not effective. 4. Accepted EPC Encoding Scheme and Select EPC/Eliminated EPC filters are still available.
See Also	RFIDGetHostMessage

6.3 READ TAG MEMORY

RFIDHostReadTagMemoryByEPC

Purpose: Read tag memory data by EPC in Host Mode.

Syntax:

```
INT RFIDHostReadTagMemoryByEPC (BYTE *AP,
                                BYTE *PC,
                                BYTE *EPC,
                                INT EL,
                                INT MB,
                                INT SA,
                                INT DL,
                                INT Retry);
```

Parameters

BYTE **AP*

Access Password (4 Bytes)

BYTE **PC*

Protocol Control (2 Bytes)

BYTE **EPC*

EPC (Max. EPC size: 32 Bytes)

INT *EL*

EPC Length byte (Max. EPC size: 32 Bytes)

INT *MB*

Memory Bank

Value	Bank
0	Reserved Bank
1	EPC Bank
2	TID
3	User

INT *SA*

Starting Address byte pointer

INT *DL*

Data Length byte to access

INT *Retry*

Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE PC [2] = {0x30, 0x00};
BYTE EPC [12] = {0x2C, 0x70, 0xAF, 0xEC, 0x2B, 0x08, 0xAE, 0x00, 0x00,
0x00, 0x00, 0x07};


if (RFIDHostReadTagMemoryByEPC (AP, PC, EPC, 12, 3, 0, 8, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0x03)
    {
        printf (“Get host tag memory data”);
        break;
    }
    else if (value == 0xFF)
    {
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to read the tag\r\n”);
        printf (“Retry counter exceeded”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

Starting Address (SA) and Data Length (DL) must be even numbers.

See Also

RFIDGetHostMessage

RFIDHostReadTagMemoryByTID

Purpose: Read tag memory data by TID in Host Mode.

Syntax:

```
INT RFIDHostReadTagMemoryByTID (BYTE *AP,
                                BYTE *TID,
                                INT TL,
                                INT MB,
                                INT SA,
                                INT DL,
                                INT Retry);
```

Parameters

BYTE **AP*

Access Password (4 Bytes)

BYTE **TID*

Tag ID (Max. EPC size: 64 Bytes)

INT *TL*

Tag ID Length byte (Max. EPC size: 64 Bytes)

INT *MB*

Memory Bank

Value	Bank
0	Reserved Bank
1	EPC Bank
2	TID
3	User

INT *SA*

Starting Address byte pointer

INT *DL*

Data Length byte to access

INT *Retry*

Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE TID [8] = {0xE2, 0x00, 0x60, 0x01, 0x01, 0x1F, 0xCD, 0x93};

if (RFIDHostReadTagMemoryByTID (AP, TID, 8, 3, 0, 8, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0x03)
    {
        printf (“Get host tag memory data”);
        break;
    }
    else if (value == 0xFF)
    {
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to read the tag\r\n”);
        printf (“Retry counter exceeded”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

Starting Address (SA) and Data Length (DL) must be even numbers.

See Also

RFIDGetHostMessage

6.4 WRITE TAG MEMORY

RFIDHostWriteTagMemoryByEPC

Purpose: Write tag memory data by EPC in Host Mode.

Syntax:

```
INT RFIDHostWriteTagMemoryByEPC (BYTE *AP,
                                 BYTE *PC,
                                 BYTE *EPC,
                                 INT EL,
                                 INT MB,
                                 INT SA,
                                 BYTE *DT,
                                 INT DL,
                                 INT Retry);
```

Parameters

BYTE *AP*

Access Password (4 Bytes)

BYTE *PC*

Protocol Control (2 Bytes)

BYTE *EPC*

EPC (Max. EPC size: 32 Bytes)

INT *EL*

EPC Length byte (Max. EPC size: 32 Bytes)

INT *MB*

Memory Bank

Value	Bank
0	Reserved Bank
1	EPC Bank
2	TID
3	User

INT *SA*

Starting Address byte pointer

BYTE *DT*

Data to write (Max. data size: 32 Bytes)

INT *DL*

Data Length byte to access

INT *Retry*

Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE PC [2] = {0x30, 0x00};
BYTE EPC [12] = {0x2C, 0x70, 0xAF, 0xEC, 0x2B, 0x08, 0xAE, 0x00, 0x00,
0x00, 0x00, 0x07};


if (RFIDHostWriteTagMemoryByEPC (AP, PC, EPC, 12, 3, 0,  (*BYTE)”ABCD” ,
4, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0xFF)
    {
        printf (“Write successful\r\n”);
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to write the tag\r\n”);
        printf (“Retry counter exceeded”);
        break;
    }
    else if (value == 0xF1)
    {
        printf (“The tag is locked”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

Starting Address (SA) and Data Length (DL) must be even numbers.

See Also

RFIDGetHostMessage

RFIDHostWriteTagMemoryByTID

Purpose: Write tag memory data by TID in Host Mode.

Syntax:

```
INT RFIDHostReadTagMemoryByTID (BYTE *AP,
                                BYTE *TID,
                                INT TL,
                                INT MB,
                                INT SA,
                                BYTE *DT,
                                INT DL,
                                INT Retry);
```

Parameters

BYTE *AP

Access Password (4 Bytes)

BYTE *TID

Tag ID (Max. EPC size: 64 Bytes)

INT *TL*

Tag ID Length byte (Max. EPC size: 64 Bytes)

INT *MB*

Memory Bank

Value	Bank
0	Reserved Bank
1	EPC Bank
2	TID
3	User

INT *SA*

Starting Address byte pointer

BYTE *DT

Data to write (Max. data size: 32 Bytes)

INT *DL*

Data Length byte to access

INT *Retry*

Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE TID [8] = {0xE2, 0x00, 0x60, 0x01, 0x01, 0x1F, 0xCD, 0x93};

if (RFIDHostWriteTagMemoryByTID (AP, TID, 8, 3, 0,  (*BYTE)”ABCD” , 4, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0xFF)
    {
        printf (“Write successful\r\n”);
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to write the tag\r\n”);
        printf (“Retry counter exceeded”);
        break;
    }
    else if (value == 0xF1)
    {
        printf (“The tag is locked”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

Starting Address (SA) and Data Length (DL) must be even numbers.

See Also

RFIDGetHostMessage

6.5 KILL TAG MEMORY

RFIDHostKillTagByEPC

Purpose	Kill tag by EPC in Host Mode.
Syntax	**INT RFIDHostKillTagByEPC (BYTE** **AP*, **BYTE** **PC*, **BYTE** **EPC*, **INT** *EL*, **INT** *Retry*);
Parameters	**BYTE** **AP* Access Password (4 Bytes) **BYTE** **PC* Protocol Control (2 Bytes) **BYTE** **EPC* EPC (Max. EPC size: 32 Bytes) **INT** *EL* EPC Length byte (Max. EPC size: 32 Bytes) **INT** *Retry* Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE PC [2] = {0x30, 0x00};
BYTE EPC [12] = {0x2C, 0x70, 0xAF, 0xEC, 0x2B, 0x08, 0xAE, 0x00, 0x00,
0x00, 0x00, 0x07};


if (RFIDHostKillTagByEPC (AP, PC, EPC, 12, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0xFF)
    {
        printf (“Kill successful\r\n”);
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to kill the tag\r\n”);
        printf (“Retry counter exceeded”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

See Also

RFIDGetHostMessage

6.6 LOCK TAG MEMORY

RFIDHostLockTagByEPC

Purpose: Lock tag by EPC in Host Mode.

Syntax:

INT RFIDHostReadTagMemoryByTID (BYTE **AP*,
BYTE **PC*,
BYTE **EPC*,
INT *EL*,
INT *MB*,
INT *Lock*,
INT *Retry*);

Parameters:

BYTE **AP*

Access Password (4 Bytes)

BYTE **PC*

Protocol Control (2 Bytes)

BYTE **EPC*

EPC (Max. EPC size: 32 Bytes)

INT *EL*

EPC length byte (Max. EPC size: 32 Bytes)

INT *MB*

Memory Bank

Value	Target
0	Kill Password
1	Access Password
2	EPC Bank
3	TID Bank
4	User Bank

INT *Lock*

Lock settings

Value	Command
0	Unlock
1	Reserved
2	Password Access*
3	Permanent Lock*

INT *Retry*

Operation retry count (ranging from 0 to 9)

Example

```
INT value;
BYTE AP [4] = {0x00, 0x00, 0x00, 0x00};
BYTE PC [2] = {0x30, 0x00};
BYTE EPC [12] = {0x2C, 0x70, 0xAF, 0xEC, 0x2B, 0x08, 0xAE, 0x00, 0x00,
0x00, 0x00, 0x07};


if (RFIDHostLockTagMemoryByEPC (AP, PC, EPC, 12, 4, 2, 9)==1)
{
    printf ("Operation…”);
}
while (1)
{
    value = RFIDGetHostMessage ();
    if (value == 0xFF)
    {
        printf (“Lock successful\r\n”);
        printf (“Finish the operation”);
        break;
    }
    else if (value == 0xF0)
    {
        printf (“Unable to write the tag”);
        printf (“Retry counter exceeded”);
        break;
    }
}
```

Return Value

If successful, it returns 1.

If *Bluetooth*® is disconnected, it returns 0. Or else refer to Status Codes.

Remarks

MB	Password Access	Permanent Lock
0, 1	Associated password location is ***readable*** and ***writeable*** in the **secured** state, not the **open** state.	Associated password location is not ***readable*** and ***writeable*** in any state.
2, 3, 4	Associated memory bank is ***writeable*** in the **secured** state, not the **open** state.	Associated memory bank is not ***writeable*** in any state.

See Also

RFIDGetHostMessage

Appendix I

STATUS CODES

Value	Meaning
0xFF	Invalid Command - Error in getting/setting bit settings in the op code field. - Error settings for request/response/event bit.
0xFE	Incorrect Packet Length - Error packet length upon a correct packet checksum. This will only occur in the Response of Get Parameter Command.
0xFD	Invalid Parameter - Error data in the Data Field
0xFC	Invalid Mode Sending Host commands in Normal Mode
0xFB	Last operation is still running
0xFA	Invalid transmit buffer setting
0xEF	RFID Fail - Failed to configure the reader when setting Q value and module power level
0xEE	Conflicts in Scan Mode and RFID Function - Set scan mode to Continuous Mode or Test Mode when the RFID Function was configured to Write Tag Memory. - Set RFID fucntion to Write Tag Memory when the scan mode is in Continuous Mode or Test Mode.
0xED	Unsupported UHF command The UHF module firmware needs to be upgraded.
0xDF	Fail to Set System Time - Configuring Real Time Clock has failed.

Note: The status codes table above offers detailed information of the errors encountered.

INDEX