Chainway Handheld Terminal API Instructions

Catalogue

1. DeviceAPI.dll introduction 8
   - 1.1 WIFI 8
     - 1.1.1 WifiOn() 8
     - 1.1.2 WifiOff() 8
     - 1.1.3 WifiDriver_whether_loaded() 8
     - 1.1.4 WifiGet_signal() 8
   - 1.2 GPRS 9
     - 1.2.1 GPRSPowerOn() 9
     - 1.2.2 GPRSPowerOff() 9
     - 1.2.3 GPRSReset() 9
     - 1.2.4 GPRS_judge_connection_status() 9
     - 1.2.5 GPRS_close_connection() 9
     - 1.2.6 GPRS_judge_modem_status() 10
     - 1.2.7 GPRS_open_serial() 10
     - 1.2.8 GPRS_close_serial() 10
     - 1.2.9 GPRS_send_data() 10
     - 1.2.10 GPRS_receive_data() 10
   - 1.3 GPS 10
     - 1.3.1 Gps_init() 10
     - 1.3.2 Gps_free() 11
     - 1.3.3 Gps_open_serial() 11
     - 1.3.4 Gps_close_serial() 11
     - 1.3.5 Gps_send_data() 11
     - 1.3.6 Gps_receive_data() 11
   - 1.4 1D 11
     - 1.4.1 Barcode1D_init() 11
     - 1.4.2 Barcode1D_free() 11
     - 1.4.3 Barcode1D_scan() 12
   - 1.5 2D hardware decoding 12
     - 1.5.1 Barcode2D_init() 12
     - 1.5.2 Barcode2D_free() 12
     - 1.5.3 Barcode2D_scan() 12
   - 1.6 2D soft decoding 12
     - 1.6.1 SoftDecoding_Init() 12
     - 1.6.2 SoftDecoding_Deinit() 12
     - 1.6.3 SoftDecoding_Select_ScanMode() 13
     - 1.6.4 SoftDecoding_Scan() 13
     - 1.6.5 SoftDecoding_Select_SnapshotMode() 13
     - 1.6.6 SoftDecoding_Snapshot() 13

1.6.7 SoftDecoding_BarcodeType_OnOff() 13
1.7 PSAM 14
1.7.1 Psam_init_Ex() 14
1.7.2 Psam_free_Ex() 14
1.7.3 Psam_powerOn_Ex() 14
1.7.4 Psam_powerOff_Ex() 14
1.7.5 Psam_select_Ex() 15
1.7.6 Psam_command_Ex() 15
1.7.7 Psam_IO_PowerOn() 15
1.7.8 Psam_IO_PowerOff() 15
1.7.9 Psam_IO_GetRandom() 15
1.7.10 Psam_IO_CmdSendAndRecv() 16
1.8 EM125K 16
1.8.1 EM125k_init() 16
1.8.2 EM125k_free() 16
1.8.3 EM125k_OpenAntenna() 16
1.8.4 EM125k_CloseAntenna() 16
1.8.5 EM125k_read() 16
1.8.6 EM125k_UID_REQ() 17
1.8.7 EM125k_ReadHitag() 17
1.8.8 EM125k_WriteHitagPage() 17
1.8.9 EM125k_WriteHitagBlock() 17
1.8.10 EM125k_Read4305() 17
1.8.11 EM125k_Write4305() 18
1.9 ISO 14443A 18
1.9.1 RF_ISO14443A_init() 18
1.9.2 RF_ISO14443A_free() 18
1.9.3 RF_OpenAntenna() 18
1.9.4 RF_CloseAntenna() 19
1.9.5 RF_ISO14443A_request() 19
1.9.6 RF_ISO14443A_request_Ex() 19
1.9.7 RF_ISO14443A_anticoll() 19
1.9.8 RF_ISO14443A_select() 19
1.9.9 RF_ISO14443A_halt() 20
1.9.10 RF_ISO14443A_authentication() 20
1.9.11 RF_ISO14443A_read() 20
1.9.12 RF_ISO14443A_write() 20
1.9.13 RF_ISO14443A_initval() 20
1.9.14 RF_ISO14443A_readval() 21
1.9.15 RF_ISO14443A_decrement() 21
1.9.16 RF_ISO14443A_increment() 21
1.9.17 RF_ISO14443A_restore() 21
1.9.18 RF_ISO14443A_transfer() 22
1.9.19 RF_ISO14443A_ul_anticoll() 22

1.9.20 RF_ISO14443A_ul_write() 22
1.9.21 RF_ISO14443A_cpu_rats() 22
1.9.22 RF_ISO14443A_cpu_reset() 22
1.9.23 RF_FM1216_OnekeyReset() 22
1.9.24 RF_ISO14443A_cpu_command_Ex() 23
1.9.25 RF_ISO14443A_cpu_command() 23
1.9.26 RF_ModeSwitch() 23
1.9.27 RF_ISO14443A_DESFIRE_Cpysel () 23
1.9.28 RF_ISO14443A_DESFIRE_SelApp() 23
1.9.29 RF_ISO14443A_DESFIRE_GetApps () 24
1.9.30 RF_ISO14443A_DESFIRE_DelApp () 24
1.9.31 RF_ISO14443A_DESFIRE_AddApp () 24
1.9.32 RF_ISO14443A_DESFIRE_GetFileIds () 24
1.9.33 RF_ISO14443A_DESFIRE_GetPiccInfo () 25
1.9.34 RF_ISO14443A_DESFIRE_AddStdFile () 25
1.9.35 RF_ISO14443A_DESFIRE_DelFile () 25
1.9.36 RF_ISO14443A_DESFIRE_GetFileSetting () 25
1.9.37 RF_ISO14443A_DESFIRE_ChangeFileSetting () 25
1.9.38 RF_ISO14443A_DESFIRE_Auth () 26
1.9.39 RF_ISO14443A_DESFIRE_ChangeKey () 26
1.9.40 RF_ISO14443A_DESFIRE_GetKeySetting () 26
1.9.41 RF_ISO14443A_DESFIRE_ChangeKeySetting () 26
1.9.42 RF_ISO14443A_DESFIRE_WriteStdFile () 27
1.9.43 RF_ISO14443A_DESFIRE_ReadStdFile () 27
1.9.44 RF_ISO14443A_DESFIRE_AddValueFile () 27
1.9.45 RF_ISO14443A_DESFIRE_GetValueFile () 27
1.9.46 RF_ISO14443A_DESFIRE_CreditValueFile () 28
1.9.47 RF_ISO14443A_DESFIRE_DebitValueFile () 28
1.9.48 RF_ISO14443A_DESFIRE_FormatCard () 28
1.10 ISO 14443B 28
1.10.1 RF_ISO14443B_cpu_reset() 28
1.10.2 RF_ISO14443B_cpu_command() 29
1.10.3 RF_ISO14443B_AT88SC6416_set_user_zone() 29
1.10.4 RF_ISO14443B_AT88SC6416_read_user_zone() 29
1.10.5 RF_ISO14443B_AT88SC6416_read_sys_zone() 29
1.10.6 RF_ISO14443B_AT88SC6416_write_user_zone() 29
1.10.7 RF_ISO14443B_AT88SC6416_write_sys_zone() 30
1.10.8 RF_ISO14443B_AT88SC6416_check_password() 30
1.10.9 RF_ISO14443B_AT88SC6416_Verify_Crypto() 30
1.10.10 RF_ISO14443B_AT88SC6416_Send_Checksum() 31
1.10.11 RF_ISO14443B_AT88SC6416_debug() 31
1.11 ISO15693 31
1.11.1 RF_ISO15693_init() 31
1.11.2 RF_ISO15693_free() 31

1.11.3 RF_ISO15693_inventory() 31
1.11.4 RF_ISO15693_stayQuiet() 32
1.11.5 RF_ISO15693_read_sm() 32
1.11.6 RF_ISO15693_write_sm() 32
1.11.7 RF_ISO15693_lockBlock() 33
1.11.8 RF_ISO15693_select() 33
1.11.9 RF_ISO15693_resetToReady() 34
1.11.10 RF_ISO15693_writeAFI() 34
1.11.11 RF_ISO15693_lockAFI() 34
1.11.12 RF_ISO15693_writeDSFID() 35
1.11.13 RF_ISO15693_lockDSFID() 35
1.11.14 RF_ISO15693_getSystemInformation() 36
1.11.15 RF_ISO15693_getMultipleBlockSecurityStatus() 36
1.12 UHF 36
1.12.1 UHFFlagCrcOn() 36
1.12.2 UHFFlafCrcOff() 36
1.12.3 UHFInit() 37
1.12.4 UHFFree() 37
1.12.5 UHFOpenAndConnect() 37
1.12.6 UHFCloseAndDisconnect() 37
1.12.7 UHFGetPower () 37
1.12.8 UHFSetPower () 37
1.12.9 UHFGetFrequency() 38
1.12.10 UHFSetFrequency_EX () 38
1.12.11 UHFSetFrequency() 38
1.12.12 UHFReadCommand() 39
1.12.13 UHFReadStatus() 39
1.12.14 UHFInventory() 39
1.12.15 UHFGetReceived() 39
1.12.16 UHFStopGet() 40
1.12.17 UHFReadData() 40
1.12.18 UHFWriteData() 40
1.12.19 UHFEraseData() 40
1.12.20 UHFLockMem() 41
1.12.21 UHFKillTag() 41
1.12.22 UHFGetVersion() 41
1.12.23 UHFInventorySingle() 42
1.12.24 UHFReadDataSingle() 42
1.12.25 UHFWriteDataSingle() 42
1.12.26 UHFReadDataNoCnt() 43
1.12.27 UHFReadDataSingleNoCnt() 43
1.12.28 UHFEraseDataSingle() 43
1.12.29 UHFLockMemSingle() 44
1.12.30 UHFKillTagSingle() 44

1.12.31 UHFAddSelect() 44
1.12.32 UHFDeleteSelect() 45
1.12.33 UHFGetSelect() 45
1.12.34 UHFGetSelectReceived() 45
1.12.35 UHFChooseSelect() 45
1.12.36 UHFInventoryAdvanced() 46
1.12.37 UHFGetInventoryAdvancedStatus() 46
1.12.38 UHFReadBuffer() 46
1.12.39 UHFGetBuffer() 46
1.12.40 UHFGetHwType () 47
1.12.41 UHFGetRSSI () 47
1.12.42 UHFSetTempProtect () 47
1.12.43 UHFGetTempProtect () 47
1.12.44 UHFGetTempValue () 48
1.12.45 UHFGetPwm () 48
1.12.46 UHFSetSingelMode () 48
1.12.47 UHFSetPwm () 48
1.12.48 UHFGetSingelMode () 49
1.13 Fingerprint 49
1.13.2 EMFingerFree() 49
1.13.3 EMGetRandomData() 49
1.13.4 EMGetImage() 49
1.13.5 EMGenChar() 50
1.13.6 EMMatch() 50
1.13.7 EMSearch() 50
1.13.8 EMRegModel() 50
1.13.9 EMStorChar() 50
1.13.10 EMLoadChar() 51
1.13.11 EMUpChar() 51
1.13.12 EMDownChar() 51
1.13.13 EMDeletChar() 51
1.13.14 EMEmpty() 51
1.13.15 EMSetReg() 52
1.13.16 EMAutoEnroll() 52
1.13.17 EMAutoMatch() 52
1.13.18 EMSetPSW() 52
1.13.19 EMVfyPSW() 52
1.13.20 EMValidTempleteNum() 53
1.13.21 EMReadChipSN() 53
1.13.22 EMSetManuFacture() 53
1.13.23 EMSetDeviceName() 53
1.13.24 EMReadSysPara() 53
1.13.25 EMUpImage() 53
1.14 Infrared 54

1.14.1 Infrared_init() 54
1.14.2 Infrared_free() 54
1.14.3 Infrared_send() 54
1.14.4 Infrared_receive() 54
1.15 Backlight 54
1.15.1 GetBackLightLevel() 54
1.15.2 SetBackLightLevel() 55
1.15.3 BackLight_Control_Switch() 55
1.16 Power 55
1.16.1 module_power_control() 55
1.16.2 system_power_down() 56
1.16.3 power_key_enabled() 56
1.16.4 power_key_disabled() 56
1.16.5 system_restart() 56
1.17 Touch_screen 56
1.17.1 touch_screen_enable() 56
1.17.2 touch_screen_disable() 56
1.18 USB 57
1.18.1 usb_device_get_current_mode() 57
1.18.2 usb_device_close_usb_function() 57
1.18.3 usb_device_storage_card_as_udisk() 57
1.18.4 usb_device_flash_as_udisk() 57
1.18.5 usb_device_set_as_activeSync() 57
1.19 LED 58
1.19.1 led_control_init() 58
1.19.2 led_control_switch() 58
1.19.3 led_control_free() 58
1.20 Extended Serial Port 58
1.20.1 extended_serial_open () 58
1.20.2 extended_serial_close () 58
1.20.3 extended_serial_data_send () 59
1.20.4 extended_serial_data_receive () 59
1.21 Expanded API 59
1.21.1 READ_API_Version() 59
1.21.2 HardwareVersion_Ex() 59
1.21.3 SerialPortSwitch_Ex() 59
1.21.4 SerialPortControl_Ex() 60
1.21.5 extended_interface_power_control() 60
1.21.6 SerialPortOpen_Ex() 60
1.21.7 SerialPortSetBaudRate_Ex() 61
1.21.8 SerialPortFunctionSwitch_Ex() 61
1.22 OS functional API 61
1.22.1 Uniscribe_disable () 61
1.23 3G 61

1.23.1 GPRSPowerOn() 61
1.23.2 GPRSPowerOff() 62
1.23.3 GPRSReset() 62
1.23.4 GPRS_judge_connection_status() 62
1.23.5 GPRS_judge_modem_status() 62
1.23.6 GPRS_open_serial() 62
1.23.7 GPRS_close_serial() 63
1.23.8 GPRS_send_data() 63
1.23.9 GPRS_receive_data() 63
1.24 Bluetooth 63
1.24.1 bluetooth_load () 63
1.24.2 bluetooth_unload () 63
1.25 Light 63
1.25.1 flash_light_power_control() 63
1.26 Printer of C5050 64
1.26.1 printer_power_control() 64
1.26.2 printer_open_serial() 64
1.26.3 printer_close_serial() 64
1.26.4 printer_send_data() 64
1.26.5 printer_receive_data() 64
2. CameraAPI.dll API instruction 65
2.1 GetCameraAPI_Version() 65
2.2 Preview() 65
2.3 CameraInit() 65
2.4 CameraPreview() 65
2.5 TakePicture() 65
2.6 FreeCamera() 66
2.7 Camera_Led_Ctrl() 66

1. DeviceAPI.dll introduction

The handheld data terminal API explains all interface functions included in the device, enabling clients to carry out secondary development easily so as to speed up the R&D progress. We have provided two dynamic links namely DeviceAPI.DLL and cameraAPI.DLL.

- DeviceAPI.DLL

It includes RFID, barcode, PSAM security identification card, GPS, vibration, backlight, fingerprint and other extension functions.

- CameraAPI.DLL

It includes camera operation, preview and snapshot functions.

1.1 WIFI

1.1.1 WifiOn()

Function	BOOL WifiOn()
Description	Load the WIFI driver
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.1.2 WifiOff()

Function	BOOL WifiOff()
Description	unload the WIFI driver
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.1.3 WifiDriver_whether_loaded()

Function	BOOL WifiDriver_whether_loaded()
Description	Judge if WIFI driver already being loaded
Parameter	Null
Return	TRUE: WIFI driver is already loaded
	FALSE: WIFI driver is not loaded yet

1.1.4 WifiGet_signal()

Function	int WifiGet_signal()
Description	Get WIFI signal strength
Parameter	Null
Return	1: not connected
	-100 to 0: increase from -100 to 0, the larger the stronger

1.2 GPRS

Note:

(1) power on the GPRS module before opening the serial port;

(2) close the serial port before powering off the GPRS module;

1.2.1 GPRSPowerOn()

Function	BOOL GPRSPowerOn ()
Description	Power on the GPRS module
Parameter	Null
Return	TRUE : success
	FALSE: failure

1.2.2 GPRSPowerOff()

Function	BOOL GPRSPowerOff()
Description	Power off the GPRS module
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.2.3 GPRSReset()

Note: this function is not recommended by the GPRS module provider which may cause chaos of the module

Function	BOOL GPRSReset()
Description	Reset the GPRS module
Parameter	Null
Return	TRUE : success
	FALSE: failure

1.2.4 GPRS_judge_connection_status()

Function	BOOL GPRS_judge_connection_status(LPCTSTR GprsName)
Description	Judge the status of the GPRS connection with the name of GprsName
Parameter	Input: GprsName, name of the GPRS connection
Return	TRUE: status of the GprsName is connected
	FALSE: status of the GprsName is disconnected

1.2.5 GPRS_close_connection()

Function	void GPRS_close_connection(LPCTSTR GprsName)
Description	Disconnect the GPRS connection with the name of GprsName
Parameter	Input: GprsName, name of the GPRS connection
Return	TRUE: disconnect the GPRS connection with the name of GprsName successfully
	FALSE: disconnect the GPRS connection with the name of GprsName failed

1.2.6 GPRS_judge_modem_status()

Funtion	BOOL GPRS_judge_modem_status ()
Description	Judge if the GPRS module is powered on
Parameter	Null
Return	TRUE: the GPRS module is powered on
	FALSE: the GPRS module is not powered on

1.2.7 GPRS_open_serial()

Function	UINT8 GPRS_open_serial()
Description	Open the GPRS module port
Parameter	Null
Return	0: Failure
	1: Success
	2： being already opened

1.2.8 GPRS_close_serial()

Function	BOOL GPRS_close_serial()
Description	Close the GPRS module serial port
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.2.9 GPRS_send_data()

Function	DWORD GPRS_send_data(UINT8* SendBuffer,DWORD dwLength)
Description	GPRS send data
Parameter	Input: SendBuffer, buffer pointer of the sending data
	Input: dwLength, length of the sending data
Return	The actual length of the sending data

1.2.10 GPRS_receive_data()

Function	DWORD GPRS_receive_data(UINT8*ReceiveBuffer,DWORD dwLength)
Description	GPRS receive data
Parameter	Output: ReceiveBuffer, buffer pointer of the receiving data
	Input: dwLength, length of the receiving data
Return	The actual length of the receiving data

1.3 GPS

1.3.1 Gps_init()

Function	void Gps_init()
Description	Power on GPS module
Parameter	Null
Return	None

1.3.2 Gps_free()

Function	void Gps_free()
Description	Power off GPS module
Parameter	Null
Return	None

1.3.3 Gps_open_serial()

Funtion	HANDLE Gps_open_serial()
Description	Open GPS serial port
Parameter	Null
Return	Return the handle of GPS port

1.3.4 Gps_close_serial()

Function	BOOL Gps_close_serial ()
Description	Close GPS serial port
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.3.5 Gps_send_data()

Funtion	DWORD Gps_send_data(UINT8* SendBuffer,DWORD dwLength)
Description	send data
Parameter	Input: SendBuffer, the buffer pointer of the sending data
	Input: dwLength, length of the sending data
Return	the actual length of the sending data

1.3.6 Gps_receive_data()

Function	DWORD Gps_receive_data(UINT8* ReceiveBuffer,DWORD dwLength)
Description	Receive data
Parameter	Output: ReceiveBuffer, buffer pointer of the receiving date
	Input: dwLength, length of the sending data
Return	The actual length of the receiving data

1.4 1D

1.4.1 Barcode1D_init()

Function	BOOL Barcode1D_init()
Description	Power on 1D barcode scan engine
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.4.2 Barcode1D_free()

Function	void Barcode1D_free ()

Description	Power off 1D barcode scan engine
Parameter	Null
Return	None

1.4.3 Barcode1D_scan()

Function	int Barcode1D_scan(UINT8 *pszData)
Description	Scan the barcode and get the content
Parameter	Output: pszData, return the pointer to save the barcode information
Return	Length of the barcode

1.5 2D hardware decoding

1.5.1 Barcode2D_init()

Function	BOOL Barcode2D_init()
Description	Power on 2D barcode scan engine
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.5.2 Barcode2D_free()

Function	BOOL Barcode2D_ free()
Description	Power off 2D barcode scan engine
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.5.3 Barcode2D_scan()

Function	BOOL Barcode2D_scan (UINT8 *pszData, int iBufferlen)
Description	2D barcode scan and get the barcode content
Parameter	Output: pszData, return the pointer to save the barcode content
	Input: iBufferlen, length of the barcode
Return	Length of the barcode

1.6 2D soft decoding

1.6.1 SoftDecoding_Init()

Function	BOOL SoftDecoding_Init ()
Description	Initialize 2D soft decoding
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.6.2 SoftDecoding_Deinit()

Function	BOOL SoftDecoding_ Deinit()
Description	Power off 2D soft decoding to release the resource
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.6.3 SoftDecoding_Select_ScanMode()

Function	BOOL SoftDecoding_ Select_ScanMode()
Description	Set the decoding mode, use before calling SoftDecoding_Scan, and use only once for uninterrupted modue
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.6.4 SoftDecoding_Scan()

Function	BOOL SoftDecoding_Scan(UINT nTimeout, UCHAR *barcodeData, UINT bufSize, UINT *ReadDataSize)
Description	scan the barcode and decode
Parameter	Input: nTimeout, the timeout time
	Input: barcodeData, storage address to save the barcode data
	Input: bufSize, set it according to the barcode numbers, recommend the 2048 bytes
	Output: ReadDataSize
Return	TRUE: decoding succeeded
	FALSE: decoding failed

1.6.5 SoftDecoding_Select_SnapshotMode()

Function	BOOL SoftDecoding_Select_SnapshotMode()
Description	Set the photographing mode and call before SoftDecoding_Snapshot, call the SoftDecoding_Select_ScanMode before going back to the decoding mode
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.6.6 SoftDecoding_Snapshot()

Function	BOOL SoftDecoding_Snapshot(UINT nTimeout)
Description	Photograph, and create the sdl.jpg under the catalogue of wince
Parameter	Input: nTimeout, the timeout time
Return	TRUE: success
	FALSE: failure

1.6.7 SoftDecoding_BarcodeType_OnOff()

Function	BOOL SoftDecoding_BarcodeType_OnOff(UCHAR* parameter, UINT

	parameterSize)
Description	enable and disable the specified barcode
Parameter	Input: parameter, the types of the barcode to be controled
	Input: parameterSize, the length of the barcode to be controled
Return	TRUE: success
	FALSE: failure

1.7 PSAM

1.7.1 Psam_init_Ex()

Function	BOOL Psam_init_Ex()
Description	Initialize PSAM module
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.7.2 Psam_free_Ex()

Function	BOOL Psam_free_Ex ()
Description	Release PSAM resource
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.7.3 Psam_powerOn_Ex()

Function	int Psam_powerOn_Ex(int iSlot, int iPowerUp, int iBaudrate, UINT8 *pszData)	
Description	Set or select PSAM slot, voltage, baudrate	
Parameter	Input	iSlot, for white slots of C5000, value 1 for the slot2, value 2 for slot1; for black slots, value 1 for 1, value 2 for 2;
		iPowerUp, 0 - 5v power supply; 1 – 3v power supply; 2 – 1.8v power supply
		iBaudrate, range from 0 to 6, 0 – 9600; 1 – 19200; 2 – 38400; 3 – 43000; 4 – 56000; 5 – 57600; 6 – 115200;
	Output	pszData, save the reset information returned by PSAM
Return	TRUE: success	
	FALSE: failure	

1.7.4 Psam_powerOff_Ex()

Function	BOOL Psam_powerOff_Ex()
Description	Power off PSAM module
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.7.5 Psam_select_Ex()

Function	int Psam_select_Ex(int iSlot)	
Description	select PSAM slot	
Parameter	Input: iSlot, select the slot, for white slots of C5000, value 1 for the slot2, value 2 for slot1; for black slots, value 1 for solt 1, value 2 for slot 2.	
Return	TRUE:success	
	FALSE:failure	

1.7.6 Psam_command_Ex()

Function	int Psam_command_Ex(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData)	
Description	send the COS command for dual-PSAM module device	
Parameter	Input	pszCOS, content of the command
		iLenCOS, length of the command
	Output	pszData, receive the array
Return	TRUE: success	
	FALSE: failure	

1.7.7 Psam_IO_PowerOn()

Note: this API is used for C2000 PSAM module powering on, we need to use the RF_ISO14443A_init()to power on the RFID, then call Psam_IO_PowerOn()to power on the PSAM module.

Function	int Psam_IO_PowerOn()
Description	Power on PSAM module and initialize
Parameter	Null
Return	1: success
	0: failure

1.7.8 Psam_IO_PowerOff()

Note: the API is used for C2000 PSAM module powering off, we need to use the RF_ISO14443A_free() to power off the RFID module after callingthis function.

Function	int Psam_IO_PowerOff()
Description	Power off PSAM module
Parameter	Null
Return	1: success
	0: failure

1.7.9 Psam_IO_GetRandom()

Note: the API is used for C2000

Function	int Psam_IO_GetRandom(UINT8 type, UINT8 *recvBuf, UINT8 *recvLen)	
Description	Get the random data	
Parameter	Input	Type, length of the random data, 4 or 8
	Output	recvBuf, the random arry to be received
		recvLen, length of the random data(1 byte)

Return	1: success
	0: failure

1.7.10 Psam_IO_CmdSendAndRecv()

Note: the API is used for C2000

Function	intPsam_IO_CmdSendAndRecv(UINT8*sendBuf,UINT8sendLen,UINT8 *recvBuf, UINT8 *recvLen)	
Description	Send command and receive the returned value	
Parameter	Input	sendBuf, content of the command
		sendLen, length of the command
	Output	recvBuf, receive the returned data
		recvLen, length returned data(1 byte)
Return	1: success	
	0: failure	

1.8 EM125K

1.8.1 EM125k_init()

Function	bool EM125k_init();
Description	initialize the low frequency module
Parameter	Null
Return	TURE: success
	FAULE: failure

1.8.2 EM125k_free()

Function	void EM125k_free();
Description	Release the low frequency resource
Parameter	Null
Return	None

1.8.3 EM125k_OpenAntenna()

Function	void OpenAntenna();
Description	open the low frequency antenna
Parameter	Null
Return	None

1.8.4 EM125k_CloseAntenna()

Function	void EM125k_CloseAntenna();
Description	close the low frequency antenna
Parameter	Null
Return	None

1.8.5 EM125k_read()

Function	int EM125k_read(int iMode, UINT8 *pszData);
Description	read the data of low frequency tag

Parameter	Input: iMode 0 for ID card ; 1 for FDX animal tag; 2 for HDX animal tag
	Output: pszData tag information, recommend 50 bytes
Return	0: success
	Others: failure

1.8.6 EM125k_UID_REQ()

Function	int EM125k_UID_REQ(int iMode, UINT8 *pszData);
Description	Obtain the UID of hitag
Parameter	Input: iMode Parameter
	Output: pszData UID information
Return	0: success
	Others: failure

1.8.7 EM125k_ReadHitag()

Function	int EM125k_ReadHitag(UCHAR nPage, UCHAR *pszData);
Description	Read the data within one page of the hitag S
Parameter	Input: nPage page number
	Output: pszData, page information, pszData[0] for length, and the data starts from pszData[1]
Return	0: success
	Others: failure

1.8.8 EM125k_WriteHitagPage()

Function	int EM125k_WriteHitagPage(UCHAR nPage, UCHAR *pszData);	
Description	Data written to the specified page of Hitag S (4 bytes)	
Parameter	Input	nPage() page
		pszData the 4bytes data to be written
Return	0: success	
	Others: failure	

1.8.9 EM125k_WriteHitagBlock()

Function	int EM125k_WriteHitagBlock(UCHAR nBlock, UCHAR *pszData);
Description	write data to the specified block(4 pages) of the Hitag S,
Parameter	Input: nBlock block number
	Input: pszData, the hex to be written
Return	0: success
	Others: failure

1.8.10 EM125k_Read4305()

Function	int EM125k_Read4305(UCHAR nPage, UCHAR *pszData);
Description	Read the data within the page (0~31) of EM4305 card
Parameter	Input: nPage page number
	Input: pszData, the read 4 bytes data

Return	0: success
	Others: failure

1.8.11 EM125k_Write4305()

Function	int EM125k_Write4305(UCHAR nPage, UCHAR *pszData);
Description	Write data to the page(0~31) of EM 4305 card; Page 0, 3, 5~13 can be read/written for saving the data; Page 1, the UID storage zone, read-only; Page 4, the parameter configuration zone; Page 14, 15, protection zone
Parameter	Input: nPage, page
	Input: pszData, the 4 bytes to be written
Return	0: success
	Others: failure

1.8.12 HID_ReadID ()

Function	int HID_ReadID(unsigned int *UID,unsigned char *ExtData)
Description	Read ID of HID card
Parameter	Input:Null
	Output: UID ID(5 decimal data) ExtData ExtData[0], ExtData[1] as the ahead 2 bytes, ExtData[2] as the facility code, please refer to the HID standards for more details
Return	0: Success
	Others: Failure

1.9 ISO 14443A

1.9.1 RF_ISO14443A_init()

Function	bool RF_ISO14443A_init()
Description	Power on the HF module and open the serial port, used for 14443A and 14443B
Parameter	Null
Return	TRUE: success, FALSE: falure

1.9.2 RF_ISO14443A_free()

Function	void RF_ISO14443A_free()
Description	Power off theHF module and close the serial port, used for 14443A and 14443B
Parmeter	Null
Return	None

1.9.3 RF_OpenAntenna()

Function	void RF_OpenAntenna();

Description	open the high frequency antanna
Parameter	Null
Return	None

1.9.4 RF_CloseAntenna()

Function	void RF_CloseAntenna();
Description	close the high frequency antenna
Parameter	Null
Return	None

1.9.5 RF_ISO14443A_request()

Note: This API is used only for C5000, RF_ISO14443A_request_Ex needs to be called to be compatiable with C2000, C3000 and C5000

Function	int RF_ISO14443A_request(int iMode, UINT8 *pszATQA);
Description	Request the tag within the antenna sensing area
Parameter	Input: iMode 0 for requesting all the awake tags, 1 for requesting all the tags
	Output: *pszATQA, the returned information(recommend 3 bytes)
Return	0: success
	Others: failure

1.9.6 RF_ISO14443A_request_Ex()

Note: For tag reading, we only recommend this function, waking up, anti-collision and selecting are not needed;

Function	int RF_ISO14443A_request_Ex(int iMode, UINT8 *pszData);
Description	Request all the tags within the antenna sensing area
Parameter	Input: iMode 0 for requesting all the awake tags, 1 for requesting all the tags
	Output: *pszATQA return ATQA(2 bytes)＋length of UID(1 byte)+ UID+SAK
Return	0: success
	Others: failure

1.9.7 RF_ISO14443A_anticoll()

Function	int RF_ISO14443A_anticoll(UINT8 *pszUID);
Description	anti-collision
Parameter	Ouput: *pszATQA UID(recommend 5 bytes)
Return	0: success
	Others: failure

1.9.8 RF_ISO14443A_select()

Function	int RF_ISO14443A_select(UINT8 *pszUID,int iLenUID,UINT8 *pszSAK);	
Description	Select the tag	
Parameter	Input:	*pszUID UID
		iLenUID length of the UID

	Output: pszSAK(2 byte)
Return	0: success
	Others: failure

1.9.9 RF_ISO14443A_halt()

Function	int RF_ISO14443A_halt();
Description	Halt the tags
Parameter	Null
Return	0: success
	Others: failure

1.9.10 RF_ISO14443A_authentication()

Function	int RF_ISO14443A_authentication(int iMode,int iBlock, UINT8 *pszKey,int iLenKey);	
Description	Authentication	
Parameter	Input	int iMode: 0 for key type A, 1 for key type B;
		int iBlock, the absolute block to be authenticated, 0~63;
		UINT8 *pszKey, content of the key(6 bytes)
		int iLenKey ,length of the key, max length 6 bytes;
Return	0	Success
	others	Failure

1.9.11 RF_ISO14443A_read()

Function	int RF_ISO14443A_read(int iBlock, UINT8 *pszData);	
Description	read the 14443A tag	
Parameter	Input	iBlock, the absolute block, 0~63;
	Output	*pszDat, the returned information; max 16 bytes for each block, recommend pszData 17 bytes
Return	0: success	
	Others: failure	

1.9.12 RF_ISO14443A_write()

Function	int RF_ISO14443A_write(int iBlock, UINT8 *pszData, int iLenData);	
Description	Write data to the tag	
Parameter	Input	iBlock the absolute block, 0~63;
		*pszData data to be written
		Max 16 bytes for each block, recommend pszData 16 bytes;
		iLenData length of the data to be wrote
Return	0: success	
	Others: failure	

1.9.13 RF_ISO14443A_initval()

Function	int RF_ISO14443A_initval(int iBlock, int iValue);	
Description	Initialize the value	
Parameter	Input	iBlock, the absolute block, 0~63;
		iValue the initialized value
Return	0: success	
	Others: failure	

1.9.14 RF_ISO14443A_readval()

Function	int RF_ISO14443A_readval(int iBlock, UINT8 *pszValue);	
Description	Read the balance value	
Parameter	Input	iBlock, the absolute block, 0~63;
	Output	*pszValue balance value(4 bytes) (pszData recommend 5 bytes)
Return	0: success	
	Others: failure	

1.9.15 RF_ISO14443A_decrement()

Function	int RF_ISO14443A_decrement(int iBlockValue,int iBlockResult, int iValue);	
Description	Decrement	
Parameter	Input	iBlockValue, block saving the current value;
		iBlockResul, block saving the remaining value;
		iValue (4 bytes)
Return	0: success	
	Others: failure	

1.9.16 RF_ISO14443A_increment()

Function	int RF_ISO14443A_increment(int iBlockValue, int iBlockResult, int iValue);	
Description	Increase	
Parameter	Input	iBlockValue, block saving the current value;
		iBlockResult, block saving the remaining value after increasing;
		iValue(4 bytes);
Return	0: success	
	Others: failure	

1.9.17 RF_ISO14443A_restore()

Function	int RF_ISO14443A_restore(int iBlock);
Description	restore the content in EEPROM to the register
Parameter	Input: int iBlock
Return	0: success
	Others: failure

1.9.18 RF_ISO14443A_transfer()

Function	int RF_ISO14443A_transfer(int iBlock);	
Description	Transfer the content in register to EEPROM;	
Parameter	Input: int iBlock;	
Return	0: success	
	Others: failure	

1.9.19 RF_ISO14443A_ul_anticoll()

Function	int RF_ISO14443A_ul_anticoll(UINT8 *pszUID);	
Description	Ultra light anti-collision	
Parameter	Input: *pszUID UID(7 bytes); recommend 8 bytes for pszUID	
Return	0: success	
	Others: failure	

1.9.20 RF_ISO14443A_ul_write()

Function	int RF_ISO14443A_ul_write(int iBlock, UINT8 *pszData, int iLenData);		
Description	Write data to the tag		
Parameter	Input	IBlock	the absolute block(block 0~3 cannot be written);
		*pszData	the data to be written(recommend 4 bytes)
		iLenData	the length of the data to be written(4bytes for each page)
Return	0: success		
	Others: failure		

1.9.21 RF_ISO14443A_cpu_rats()

Function	int RF_ISO14443A_cpu_rats(UINT8 *pszData);
Description	CPU RATS operation(type A)
Parameter	Output: *pszData, recommend 50 bytes
Return	0: success
	Others: failure

1.9.22 RF_ISO14443A_cpu_reset()

Function	int RF_ISO14443A_cpu_reset(UINT8 *pszData);
Description	CPU reset(type A)
Parameter	Output: *pszData, recommend 50 bytes
Return	0: success
	Others: failure

1.9.23 RF_FM1216_OnekeyReset()

Function	int RF_FM1216_OnekeyReset(UINT8 *pszData);
Description	FM1216 reset
Parameter	Output: *pszData

Return	0: success
	Others: failure

1.9.24 RF_ISO14443A_cpu_command_Ex()

Function	int RF_ISO14443A_cpu_command_Ex(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData);	
Description	CPU T=CL send COS command(customized for Nigeria CPU card)	
Parameter	Input:	UINT8 *pszCOS content of the COS command
		int iLenCOS length of the COS command
	Output:	UINT8 *pszData, the returned information Recommend pszData as 128 bytes
Return	0: Success	
	Others: Failure	

1.9.25 RF_ISO14443A_cpu_command()

Function	int RF_ISO14443A_cpu_command(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData);	
Description	CPU T=CL send COS command	
Parameter	Input:	UINT8 *pszCOS content of the COS command
		int iLenCOS length of the COS command
	Output:	UINT8 *pszData, the returned information
Return	0: success	
	Others: failure	

1.9.26 RF_ModeSwitch()

Function	int RF_ModeSwitch(int iMode)	
Description	RFID module switch(ISO14443A by default)	
Parameter	Input	iMode: 0 for ISO14443A; 1 for ISO14443B; 2 for ISO15693
Return	TRUE: success	
	FALSE: failure	

1.9.27 RF_ISO14443A_DESFIRE_Cpysel ()

Function	void RF_ISO14443A_DESFIRE_Cpysel(unsigned char cpyType)	
Description	DESFIRE encryption types select	
Parameter	Input	cpyType: 0x01 AES; 0x02 DES, DES by default;
Return	Null	

1.9.28 RF_ISO14443A_DESFIRE_SelApp()

Note: all DESFire functions only work on RC663 devices.

Functions	int RF_ISO14443A_DESFIRE_SelApp(unsigned char *AppId)
Descripton	Desfire select APP

Parameter	Input	AppId(3 bytes)
Return	0: success	
	Others: failure	

1.9.29 RF_ISO14443A_DESFIRE_GetApps ()

Function	int RF_ISO14443A_DESFIRE_GetApps(unsigned char *AppNums,unsigned char *AppIDs)	
Description	Get all the Apps	
Parameter	Input	Null
	Output	AppNums: get the numbers(1 byte) AppIDs: buffers to save the ID(bytes of 3*AppNums)
Return	0: success	
	Others: failure	

1.9.30 RF_ISO14443A_DESFIRE_DelApp ()

Function	int RF_ISO14443A_DESFIRE_DelApp(unsigned char *AppId)	
Description	Desfire delete App	
Parameter	Input	AppId(3 bytes)
Return	0: success	
	Others: failure	

1.9.31 RF_ISO14443A_DESFIRE_AddApp ()

Function	int RF_ISO14443A_DESFIRE_AddApp(unsigned char *AppId,unsigned char KeySetting,unsigned char FileNums)	
Description	Desfire add App(max 28 Apps,range from 1~27 except App 0)	
Parameter	Input	AppId(3 bytes) FileNums KeySetting
	output	Null
Return	0: success	
	Others: failure	

1.9.32 RF_ISO14443A_DESFIRE_GetFileIds ()

Function	int RF_ISO14443A_DESFIRE_GetFileIds(unsigned char *FileNums, unsigned char *FileBuf)	
Description	Get the file IDs	
Parameter	Input	Null
	Output	FileNums,1 byte returns by pointer FileBuf, 1 byte for 1 fileID
Return	0: success	
	Others: failure	

1.9.33 RF_ISO14443A_DESFIRE_GetPiccInfo ()

Function	int RF_ISO14443A_DESFIRE_GetPiccInfo(unsigned char *PiccInfo)	
Description	Get Picc Info	
Parameter	Input	Null
	Output	PiccInfo storage buffer including ID and manufacturer info
Return	0: success	
	Others: failure	

1.9.34 RF_ISO14443A_DESFIRE_AddStdFile ()

Function	int RF_ISO14443A_DESFIRE_AddStdFile(unsigned char FileNo,unsigned char CommSet,unsigned char *AccessRight,unsigned int FileSize);	
Description	Add standard file	
Parameter	Input	FileNo 1 byte CommSet communication setting(encryption method) AccessRight FileSize
	Output	Null
Return	0: success	
	Others: failure	

1.9.35 RF_ISO14443A_DESFIRE_DelFile ()

Function	int RF_ISO14443A_DESFIRE_DelFile(unsigned char FileNo)	
Description	Delete file(including standard data file and value file)	
Parameter	Input	FileNo
	Output	Null
Return	0: success	
	Others: failure	

1.9.36 RF_ISO14443A_DESFIRE_GetFileSetting ()

Function	int RF_ISO14443A_DESFIRE_GetFileSetting(unsigned char FileNo, unsigned char *FileAttriBuf)	
Description	Get file setting	
Parameter	Input	FileNo
	Outut	FileAttriBuf
Return	0: success	
	Others: failure	

1.9.37 RF_ISO14443A_DESFIRE_ChangeFileSetting ()

Function	int RF_ISO14443A_DESFIRE_ChangeFileSetting(unsigned char FileNo, unsigned char CommSet,unsigned char *AccessRights)
Description	Change file setting

Parameter	Input	FileNo CommSet AccessRights
	Output	Null
Return	0: success	
	Others: failure	

1.9.38 RF_ISO14443A_DESFIRE_Auth ()

Function	int RF_ISO14443A_DESFIRE_Auth(unsigned char KeyNo,unsigned char *KeyBuf,unsigned char KeyLen)	
Description	Authenticate	
Parameter	Input	KeyNo KeyBuf KeyLen
	Output	Null
Return	0: success	
	Others: failure	

1.9.39 RF_ISO14443A_DESFIRE_ChangeKey ()

Function	int RF_ISO14443A_DESFIRE_ChangeKey(unsigned char KeyNo,unsigned char *KeyBuf,unsigned char KeyLen)	
Description	Change key	
Parameter	Input	KeyNo KeyBuf KeyLen
	Output	Null
Return	0: success	
	Others: failure	

1.9.40 RF_ISO14443A_DESFIRE_GetKeySetting ()

Function	int RF_ISO14443A_DESFIRE_GetKeySetting(unsigned char *KeySetting, unsigned char *KeyNums);	
Description	Get key settings	
Parameter	Input	Null
	Output	KeySetting buffer 1 byte KeyNums 1 byte
Return	0: success	
	Others: failure	

1.9.41 RF_ISO14443A_DESFIRE_ChangeKeySetting ()

Function	int RF_ISO14443A_DESFIRE_ChangeKeySetting(unsigned char KeySetting)
Description	Change key settings

Parameter	Input	KeySetting
Return	0: success	
	Others: failue	

1.9.42 RF_ISO14443A_DESFIRE_WriteStdFile ()

Function	int RF_ISO14443A_DESFIRE_WriteStdFile(unsigned char FileNo,unsigned int OffSet,unsigned int DataSize,unsigned char *DataBuf)	
Description	Write standard file	
Parameter	Input	FileNo OffSet DataSize DataBuf
	Output	Null
Return	0: success	
	Others: failure	

1.9.43 RF_ISO14443A_DESFIRE_ReadStdFile ()

Function	int RF_ISO14443A_DESFIRE_ReadStdFile(unsigned char FileNo,unsigned int OffSet,unsigned int DataSize,unsigned char *DataBuf)	
Description	Read standard file	
Parameter	Input	FileNo OffSet DataSize
	Output	DataBuf
Return	0: success	
	Others: failure	

1.9.44 RF_ISO14443A_DESFIRE_AddValueFile ()

Function	int RF_ISO14443A_DESFIRE_AddValueFile(unsigned char FileNo,unsigned char CommSet,unsigned char *AccessRights,int MinValue, int MaxValue,int InitValue)	
Description	Add value file	
Parameter	Input	FileNo CommSet AccessRights MinValue MaxValue InitValue
	Output	Null
Return	0: success	
	Others: failure	

1.9.45 RF_ISO14443A_DESFIRE_GetValueFile ()

Function	int RF_ISO14443A_DESFIRE_GetValueFile(unsigned char FileNo, int *Value)	
Description	Get the value file Desfire(get the balance)	
Parameter	Input	FileNo
	output	Value
Return	0: success	
	Others: failure	

1.9.46 RF_ISO14443A_DESFIRE_CreditValueFile ()

Function	int RF_ISO14443A_DESFIRE_CreditValueFile(unsigned char FileNo, unsigned int CreValue)	
Description	CreditValue	
Parameter	Input	FileNo CreValue（positive integer）
	Output	Null
Return	0: success	
	Others: failure	

1.9.47 RF_ISO14443A_DESFIRE_DebitValueFile ()

Function	int RF_ISO14443A_DESFIRE_DebitValueFile(unsigned char FileNo, unsigned int DeValue)	
Description	debitValue	
Parameter	Input	FileNo DeValue（positive integer）
	Output	Null
Return	0: success	
	Others: failure	

1.9.48 RF_ISO14443A_DESFIRE_FormatCard ()

Function	int RF_ISO14443A_DESFIRE_FormatCard()	
Description	Format Desfire card	
Parameter	Input	Null
	Output	Null
Return	0: success	
	Others: failure	

1.10 ISO 14443B

1.10.1 RF_ISO14443B_cpu_reset()

Function	int RF_ISO14443B_cpu_reset(UINT8 *pszData);
Description	CPU rest(type B)
Parameter	Output: UINT8 *pszData; recommend 128 bytes
Return	0: success
	Others: failure

1.10.2 RF_ISO14443B_cpu_command()

Function	int RF_ISO14443B_cpu_command(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData);	
Description	CPU T=CL send COS command (type B)	
Parameter	Input:	UINT8 *pszCOS content of the COS command
		int iLenCOS the length of the COS command
	Output: UINT8 *pszData, the returned information, pszData, recommend 128 bytes	
Return	0: success	
	Others: failure	

1.10.3 RF_ISO14443B_AT88SC6416_set_user_zone()

Function	int RF_ISO14443B_AT88SC6416_set_user_zone(UINT8 Data, UINT8 *pszData);	
Description	Set user zone	
Parameter	Input	Data user zone: 0~15;
	Output	pszData, the returned information, pszData, recommend 128 bytes
Return	0: success	
	Others: failure	

1.10.4 RF_ISO14443B_AT88SC6416_read_user_zone()

Function	int RF_ISO14443B_AT88SC6416_read_user_zone(UINT8 addr_H,UINT8 addr_L, UINT8 number, UINT8 *pszData);	
Description	Read the user zone, 512 bytes in total, starting from address 0	
Parameter	Input	addr_H the high 8 addresses of the user zone
		addr_L the low 8 addresses of the user zone
		number numbers of the bytes
	Output	pszData, the returned data information
Return	0: success	
	Others: failure	

1.10.5 RF_ISO14443B_AT88SC6416_read_sys_zone()

Function	int RF_ISO14443B_AT88SC6416_read_sys_zone(UINT8 addr,UINT8 number, UINT8 *pszData);	
Description	Read the system zone, 256 bytes in total, starting from address 0;	
Parameter	Input	addr system_zone address
		number numbers of the bytes
	Output	pszData, the returned information;
Return	0: success	
	Others: failure	

1.10.6 RF_ISO14443B_AT88SC6416_write_user_zone()

Function	int RF_ISO14443B_AT88SC6416_write_user_zone(UINT8 addr_H,UINT8

	addr_L,UINT8 number, UINT8 *pszData);	
Description	Write data to the user zone, 512 bytes for each sector, starting for address 0;	
Parameter	Input	addr_H the high 8 addresses of the user zone
		addr_L the low 8 addresses of the user zone
		Number, numbers of the written bytes
	Output	pszData, the returned data information
Return	0: success	
	Others: failure	

1.10.7 RF_ISO14443B_AT88SC6416_write_sys_zone()

Function	int RF_ISO14443B_AT88SC6416_write_sys_zone(UINT8 addr,UINT8 number, UINT8 *pszData);	
Description	Write data to the system zone, 256 bytes in total, starting from address 0;	
Parameter	Input	addr
		number numbers of the bytes to be written
	Output	pszData, the returned data information
Return	0 : success	
	Others: failure	

1.10.8 RF_ISO14443B_AT88SC6416_check_password()

Function	int RF_ISO14443B_AT88SC6416_check_password(UINT8 password_index,UINT8 *password, UINT8 *pszData);	
Description	Check password	
Parameter	Input	password_index index number, starting from 0 to 7, and 7 as the default
		*password password, 3 bytes in total
	Output	pszData, the returned data information
Return	0: success	
	Others: failure	

1.10.9 RF_ISO14443B_AT88SC6416_Verify_Crypto()

Function	int RF_ISO14443B_AT88SC6416_Verify_Crypto(UINT8 key_index,UINT8 *pQData, UINT8 *pCHData, UINT8 *pszData);		
Description	verify crypto		
Parameter	Input	key_index	00 Secret Seed G0
			01 Secret Seed G1
			02 Secret Seed G2
			03 Secret Seed G3
			10 Session Encryption Key S0
			11 Session Encryption Key S1
			12 Session Encryption Key S2
			13 Session Encryption Key S3

		*pQData 8 bytes in total
		*pCHData 8 bytes in total
	Output	pszData, the returned information;
Return	0 for success	
	Others for failure	

1.10.10 RF_ISO14443B_AT88SC6416_Send_Checksum()

Function	int RF_ISO14443B_AT88SC6416_Send_Checksum(UINT8 *pMAC, UINT8 *pszData);	
Description	send the checksum	
Parameter	Input	UINT8 *pMAC 2 bytes in total
	Output	pszData, the returned information, pszData, recommend 128 bytes
Return	0: success	
	Others: failure	

1.10.11 RF_ISO14443B_AT88SC6416_debug()

Function	int RF_ISO14443B_AT88SC6416_debug(UINT8 data_lenth,UINT8 *pdata_send, UINT8 *pszData);	
Description	debug the subpanel program	
Parameter	Input	UINT8 data_lenth data sent to the subpanel, max length 254 bytes
		UINT8 *data_send data sent to the subpanel
	Output	pszData, the returned information, recommend 128 bytes
Return	0: success	
	Others: failure	

1.11 ISO15693

1.11.1 RF_ISO15693_init()

Function	bool RF_ISO15693_init();
Description	Initialize the 15693 mode
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.11.2 RF_ISO15693_free()

Function	bool RF_ISO15693_free();
Description	Release the module resource
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.11.3 RF_ISO15693_inventory()

Function	int RF_ISO15693_inventory(int iMode, int iAFI, UINT8 *pszData);

Description	Read UID		
Parameter	Input	iMode ：	0 multiple tags calling, no AFI
			1 single tag calling, no AFI
			2 multiple tags calling, AFI
			3 single tag calling, AFI
		iAFI: AFI	
	Output: pszData: UID		
Return	0: success		
	Others: failure		

1.11.4 RF_ISO15693_stayQuiet()

Function	int RF_ISO15693_stayQuiet(UINT8 *pszUID, int iLenUID);	
Description	Set the tags to the status of quiet	
Parameter	Input	pszUID: UID, recommend 8 bytes
		iLenUID: length of the UID
Return	0: success	
	Others: failure	

1.11.5 RF_ISO15693_read_sm()

Function	int RF_ISO15693_read_sm(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum, UINT8 *pszData);		
Description	Read the internal data		
Parameter	Input	iMode: (0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
			4 no SELECT, no upload UID TI tag
			5 SELECT, no upload UID TI tag
			6 no SELECT, upload UID TI tag
			7 SELECT, upload UID TI tag
		pszUID: UID	
		iLenUID: UID length	
		Startblock: starting block	
		Blocknum: numbers of the reading blocks(1-10)	
	Output	pszData	
Return	0: success		
	Others: failure		

1.11.6 RF_ISO15693_write_sm()

Function	int RF_ISO15693_write_sm(int iMode, UINT8 *pszUID, int iLenUID, int startblock,

	int blocknum, UINT8 *pszData, int iWriteLen);		
Description	write data to the internal		
Parameter	Input	iMode(0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
			4 no SELECT, no upload UID TI tag
			5 SELECT, no upload UID TI tag
			6 no SELECT, upload UID TI tag
			7 SELECT, upload UID TI tag
		pszUID：UID	
		iLenUID: length of UID	
		Startblock: starting block	
		Blocknum: numbers of the blocks to be written (4 bytes for each block)	
		pszData	
		iWriteLen: length of the data to be written (the integral multiple of 4 bytes)	
Return	0: success		
	Others: failure		

1.11.7 RF_ISO15693_lockBlock()

Function	int RF_ISO15693_lockBlock(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum);		
Description	Lock the block		
Parameter	Input	iMode(value range 0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
			4 no SELECT, no upload UID TI tag
			5 SELECT, no upload UID TI tag
			6 no SELECT, upload UID TI tag
			7 SELECT, upload UID TI tag
		pszUID: UID	
		iLenUID: length of UID	
		startblock	
		blocknum: numbers of the block to be locked	
Return	0: success		
	Others: failure		

1.11.8 RF_ISO15693_select()

Function	int RF_ISO15693_select(UINT8 *pszUID, int iLenUID);
Description	Select the tag

Parameter	Input	pszUID: UID
		iLenUID: length of UID
Return	0: success	
	Others: failure	

1.11.9 RF_ISO15693_resetToReady()

Function	int RF_ISO15693_resetToReady(int iMode, UINT8 *pszUID, int iLenUID);		
Description	ISO15693 reset command and change it to be ready		
Parameter	Input	iMode(value range 0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
		pszUID： UID	
		iLenUID: the length of the UID	
Return	0: success		
	Others: failure		

1.11.10 RF_ISO15693_writeAFI()

Function	int RF_ISO15693_writeAFI(int iMode, UINT8 *pszUID, int iLenUID, int iAFI);		
Description	Write AFI		
Parameter	Input	iMode(value range 0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
			4 no SELECT, no upload UID TI tag
			5 SELECT, no upload UID TI tag
			6 SELECT, upload UID TI tag
			7 SELECT, upload UID TI tag
		pszUID: UID	
		iLenUID: length of UID	
		iAFI: AFI	
Return	0: success		
	Others: failure		

1.11.11 RF_ISO15693_lockAFI()

Function	int RF_ISO15693_lockAFI(int iMode, UINT8 *pszUID, int iLenUID);		
Description	Lock AFI		
Parameter	Input	iMode(value range 0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI

			4	no SELECT, no upload UID TI tag
			5	SELECT, no upload UID TI tag
			6	no SELECT, upload UID TI tag
			7	SELECT, upload UID TI tag
		pszUID: UID		
		iLenUID: length of UID		
Return	0: success			
	Others: failure			

1.11.12 RF_ISO15693_writeDSFID()

Function	int RF_ISO15693_writeDSFID(int iMode, UINT8 *pszUID, int iLenUID, int iDSFID);			
Description	Write DSFID			
Parameter	Input	iMode(value range 0-7)	0	no SELECT, no upload UID NXP I CODE SLI
			1	SELECT, no upload UID NXP I CODE SLI
			2	no SELECT, upload UID NXP I CODE SLI
			3	SELECTstatus, upload UID NXP I CODE SLI
			4	no SELECT, no upload UID TI tag
			5	SELECT, no upload UID TI tag
			6	no SELECT, upload UID TI tag
			7	SELECT, upload UID TI tag
		pszUID: UID		
		ILenUID: length of UID		
		iDSFID: DSFID		
Return	0: success			
	Others: failure			

1.11.13 RF_ISO15693_lockDSFID()

Function	int RF_ISO15693_lockDSFID(int iMode, UINT8 *pszUID, int iLenUID);			
Description	Lock DSFID			
Parameter	Input	iMode(value range 0-7)	0	no SELECT, no upload UID NXP I CODE SLI
			1	SELECT,no uploadUID NXP I CODE SLI
			2	no SELECT, uplpad UID NXP I CODE SLI
			3	SELECT, upload UID NXP I CODE SLI
			4	no SELECT, no upload UID TI tag
			5	SELECT, no upload UID TI tag
			6	no SELECT, upload UID TI tag
			7	SELECT, upload UID TI tag
		pszUID: UID		
		iLenUID: length of UID		
Return	0: success			
	Others: failure			

1.11.14 RF_ISO15693_getSystemInformation()

Function	int RF_ISO15693_getSystemInformation(int iMode, UINT8 *pszUID, int iLenUID, UINT8 *pszData);		
Description	Get the system information		
Parameter	Input	iMode(value range 0-7)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
		pszUID: UID	
		iLenUID: length of UID	
		pszData:	
Return	0: success		
	Others: failure		

1.11.15 RF_ISO15693_getMultipleBlockSecurityStatus()

Function	int RF_ISO15693_getMultipleBlockSecurityStatus(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum, UINT8 *pszData);		
Description	Get the tag information		
Parameter	Input	iMode(value range 0-3)	0 no SELECT, no upload UID NXP I CODE SLI
			1 SELECT, no upload UID NXP I CODE SLI
			2 no SELECT, upload UID NXP I CODE SLI
			3 SELECT, upload UID NXP I CODE SLI
		pszUID: UID	
		iLenUID: length of UID	
		Startblock: starting block	
		Blocknum: numbers of the read blocks	
		pszData: tag information	
Return	0: success		
	Others: failure		

1.12 UHF

1.12.1 UHFFlagCrcOn()

Function	void UHFFlagCrcOn ()
Description	enable CRC (no CRC by default), RAYLINK module only
Parameter	Null
Return	None

1.12.2 UHFFlafCrcOff()

Function	void UHFFlagCrcOn ()
Description	Disable CRC verification, no CRC by default, RAYLINK module only

Parameter	Null
Return	None

1.12.3 UHFInit()

Function	void UHFInit ()
Description	Power on the UHF module
Parameter	Null
Return	None

1.12.4 UHFFree()

Function	void UHFFree ()
Description	Power off the UHF module
Parameter	Null
Return	None

1.12.5 UHFOpenAndConnect()

Function	bool UHFOpenAndConnect(UCHAR* status)
Description	Connect UHF
Parameter	status—obtain the working status of UHF(bit 0 for connecting successfully);
Return	TRUE: success
	FALSE; failure

1.12.6 UHFCloseAndDisconnect()

Function	bool UHFCloseAndDisconnect()
Description	Disconnect
Parameter	Null
Return	TRUE: success
	FALSE; failure

1.12.7 UHFGetPower ()

Function	bool UHFGetPower (UCHAR* uPower)
Description	Get the UHF power
Parameter	uPower——the returned value(10~30dBm);
Return	TRUE: success
	FALSE: failure

1.12.8 UHFSetPower ()

Function	bool UHFSetPower (UCHAR uPower)
Description	set the UHF power
Parameter	uPower——set the UHF power Parameter（10 <= uPower <= 30）
Return	TRUE: success

	FALSE: failure

Note: For C5000 UHF device, the max power we suggest is 24 dBm.

1.12.9 UHFGetFrequency()

Function	bool UHFGetFrequency(UCHAR* uFreMode, UCHAR* uFreBase, UCHAR* uBaseFre, UCHAR* uChannNum, UCHAR* uChannSpc, UCHAR* uFreHop)	
Description	obtain the frequency of UHF module	
Parameter	uFreMode	-frequency mode（0x00-920-925MHz、0x01--845MHz, 0x02-ETSI Standard, 0x03-Fixed frequency(915MHz), 0x04-customization）
	uFreBase	- frequency cardinal(0x00-50KHz, 0x01-125KHZ), the bandwidth cannot be larger than 12MHz when the value is 50KHz, and the bandwidth cannot be larger than 32MHz when it is 125KHz;
	uBaseFre-	Starting frequency(2 bytes, bit 14~5 as the integer part, bit 3~0 as the decimal part)
	uChanNum	Channelnumbers
	uChannSpc-	Bandwidth(bit3~0 as the product)
	uFreHop	Frequency hopping sequency (0x00-random,0x01-high to low,0x10-low to high, FALSE-random)
Return	TRUE: success	
	FALSE: failure	

1.12.10 UHFSetFrequency_EX ()

Function	bool UHFSetFrequency_EX(UCHAR FreMode)
Description	Set the UHF working frequency
Parameter	Input: FreMode, 0x00 –(base frequency-125KHz, starting frequency 840.625MHz, channels-16, bandwidth product-2, random frequency hopping) 0x01 –(base frequency-125KHz, starting frequency920.625MHz, channels-16,bandwidthproduct -2, random frequency hopping) 0x02 –(ETSIStandared, base frequency-50KHz, starting frequency 865.1MHz, channels-12, bandwidth product-4, random frequency hopping) 0x03 –(top frequency, base frequency-125KHz, starting frequency 915MHz, channel-1, bandwidth product-0, random frequency hopping)
Return	TRUE: success
	FALSE: failure

1.12.11 UHFSetFrequency()

Function	bool UHFGetFrequency(UCHAR* uFreMode, UCHAR* uFreBase, UCHAR* uBaseFre, UCHAR* uChannNum, UCHAR* uChannSpc, UCHAR* uFreHop)	
Description	Set the UHF working frequency	
Parameter	uFreMode	-frequency mode (0x00-920-925MHz, 0x01-840-845MHz, 0x02-ETSIStandard, 0x03-Fixed Frequency (915MHz), 0x04-Customization)

	uFreBase	-frequency base number (0x00-50KHz, 0x01-125KHZ),the bandwidth cannot be larger than 12MHz when the base number is 50KHz, the bandwidth cannot belarger than 32MHz when the base number is 125KHz;
	uBaseFre-	The starting frequency (2 bytes, bit 14~5 as the integet part, bit3~0 as the decimal part)
	uChanNum	Channel numbers
	uChannSpc-	Channel bandwidth (bit3~0 product number of the bandwidth)
	uFreHop	Frequency hopping (0x00-random, 0x01-high to low, 0x10-low to high, FALSE-random)
Return	TRUE: success	
	FALSE: failure	

1.12.12 UHFReadCommand()

Function	bool UHFReadCommand (UCHAR* uCmd)	
Description	get the current command	
Parameter	uCmd	
Return	TRUE: success	
	FALSE: failure	

1.12.13 UHFReadStatus()

Function	bool UHFReadStatus (UCHAR* uStatus)	
Description	read the UHF working status, RAYLINK module only	
Parameter	uStatus:get the working status(bit0 for connecting successfully)	
Return	TRUE: success	
	FALSE: failure	

1.12.14 UHFInventory()

Function	bool UHFInventory (UCHAR flagAnti, UCHAR initQ)	
Description	send the cycling UHF recognition command	
Parameter	flagAnti	0x01-enable the anti-collition mode,0x00-disable the anti-collision
	initQ	Q value (valid flagAnti=1), value range (0~15), scan the target tag continuously by the times of 2 to the Q, and time is in direct proportion to the index of Q value for each anti-collision
Return	TRUE: success	
	FALSE: failure	

1.12.15 UHFGetReceived()

Function	bool UHFGetReceived (int* uLenUii, UCHAR* uUii)	
Description	Receive the returned UII	
Parameter	uLenUii	Length of the uUii
	uUii	The UII should be at least 66 bytes
Return	TRUE: success	

	FALSE: failure

1.12.16 UHFStopGet()

Function	bool UHFStopGet()
Description	Stop the UHF scanning
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.12.17 UHFReadData()

Function	bool UHFReadData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uReadData, UCHAR* uErrorCode)		
Description	Read the tag data (specified UII)		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address
		uCnt	Length of the read data (two bytes as a unit, cannot be 0)
		uUii	UII
	Output	uReadData	The read data (at least ucnt*2 bytes)
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.18 UHFWriteData()

Function	bool UHFWriteData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uWriteData, UCHAR* uErrorCode)		
Description	Write data (the specified UII)		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address
		uCnt	Length of the data (two bytes as a unit, cannot be 0)
		uUii	UII
		uWriteData	The written data (cnt*2 bytes)
	Output	uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.19 UHFEraseData()

Function	bool UHFEraseData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uErrorCode)		
Description	Erase the data		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Types of the tag (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address (EBV format)
		uCnt	Length of the data (two bytes as a unit, cannot be 0)
		uUii	UII
	Output	uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.20 UHFLockMem()

Function	bool UHFLockMem (UCHAR* uAccessPwd, UCHAR* uLockData, UCHAR* uUii, UCHAR* uErrorCode)		
Description	lock the specified data segment		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uLockData	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uUii	UII
	Output	uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.21 UHFKillTag()

Function	bool UHFKillTag (UCHAR* uKillPwd, UCHAR* uUii, UCHAR* uErrorCode)		
Description	Kill the specified tag		
Parameter	Input	uKillPwd	KillPassWord (4 bytes)
		uUii	UII
	Output	uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.22 UHFGetVersion()

Function	bool UHFGetVersion (UCHAR* uSerial, UCHAR* uVersion)		
Description	Get the hardware S/N and software version		
Parameter	Output	uSerial	Serial number of the UHF hardware(6 bytes)
		uVersion	UHF software version (3 bytes)

Return	TRUE: success	
	FALSE: failure	

1.12.23 UHFInventorySingle()

Function	bool UHFInventorySingle (UCHAR* uLenUii, UCHAR* uUii)		
Description	Single recognition(only operate once regardless of success or failure)		
Parameter	Output	uLenUii	Length of the UII
		uUii	UII
Return	TRUE: success		
	FALSE: failure		

1.12.24 UHFReadDataSingle()

Function	bool UHFReadDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uReadData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
Description	Read data (no specified UII)		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address (EBV format)
		uCnt	Length of the read data (two bytes as a unit, cannot be 0)
	Output	uReadData	Read data (at least uCnt*2 bytes)
		uUii	UII
		uLenUii	Length of the UII (unit: byte)
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.25 UHFWriteDataSingle()

Function	bool UHFWriteDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uWriteData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
Description	Write data (no specified UII)		
Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address (EBV format)
		uCnt	Length of the read data (two bytes as a unit, cannot be 0)
		uUii	UII
		uWriteData	Data to be written
	Output	uUii	UII
		uLenUii	Length of the (unit: byte)

		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.26 UHFReadDataNoCnt()

Function	bool UHFReadDataNoCnt (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR* uUii, UCHAR* Data_len, UCHAR* uReadData, UCHAR* uErrorCode)		
Description	Read the data when the CNT is 0 (specified UII)		
Parameter	Input	uAccessPwd	ACCESS PASSWORD (4 bytes)
		uBank	Type of the data (0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address (EBV format)
		uUii	UII—UII of the tag
	Output	Data_len	Length of the read data
		uReadData	the read data
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.27 UHFReadDataSingleNoCnt()

Function	bool UHFReadDataSingleNoCnt (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR* Data_len, UCHAR* uReadData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
Description	Read the data when the CNT is 0 (no specified UII)		
Parameter	Input	uAccessPwd	ACCESS PASSWORD (4 bytes)
		uBank	Type of the data(0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER)
		uPtr	Offset of the starting address (EBVformat)
	Output	Data_len	Length of the read data
		uReadData	the read data
		uUii	UII
		uLenUii	Length of the UII
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.28 UHFEraseDataSingle()

Function	bool UHFEraseDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uErrorCode)
Description	Erase the data(no specified UII)

Parameter	Input	uAccessPwd	ACCESSPASSWORD (4 bytes)
		uBank	Type of the data segment(0x00-RESERVED, 0x01-EPC, 0x02-TID, 0x03-USER)
		uPtr	Offset of the starting address (EBV format)
		uCnt	Length of the data to be erased(two bytes as the unit)
	Output	uUii	Return the UII
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.29 UHFLockMemSingle()

Function	bool UHFLockMem (UCHAR* uAccessPwd, UCHAR* uLockData, UCHAR* uUii, UCHAR* uErrorCode)		
Description	lock the specified data segment(no specified UII)		
Parameter	Input	uAccessPwd	ACCESSPASSWORD(4 bytes)
		uLockData	LockData(3bytes)
	Output	uUii	Return the UII
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.30 UHFKillTagSingle()

Function	bool UHFKillTagSingle (UCHAR* uKillPwd, UCHAR* uUii, UCHAR* uErrorCode)		
Description	Kill the tag(no specified UII)		
Parameter	Input	uKillPwd	The Kill Password(4 bytes)
	Output	uUii	Return the UII
		uErrorCode	Error Code(only works when returning failure and the uErrorcode is not 0xff)
Return	TRUE: success		
	FALSE: failure		

1.12.31 UHFAddSelect()

Function	bool UHFAddSelect (UCHAR* precord, UCHAR* STATUS)		
Description	Add the select record, RAYLINK module only		
Parameter	Input	precord	The starting address of the SRECORD structure(please refer to the SRECORD definition)
	Output	STATUS	Return the status information (0x00-success, 0x01-SINDEX out of the range, 0x02-SINDEX=0 read-only SRECORD, 0x03-undefined)
Return	TRUE: success		
	FALSE: failure		

1.12.32 UHFDeleteSelect()

Function	bool UHFDeleteSelect (UCHAR SINDEX, UCHAR* STATUS)		
Description	Delete the Select record, RAYLINK module only		
Parameter	Input	SINDEX	Select serial number to be deleted
	Output	STATUS	Return the status information (0x00-success, 0x01-SINDEX out of the range, 0x02-SINDEX=0 read-only SRECORD, 0x03-undefined)
Return	TRUE: success		
	FALSE: failure		

1.12.33 UHFGetSelect()

Function	bool UHFGetSelect (UCHAR SINDEX, UCHAR SNUM, UCHAR* STATUS)		
Description	Read the Select record, RAYLINK module only		
Parameter	Input	SINDEX	The Select serial number to be read
		SNUM	Numbers of the Select to be read
	Output	STATUS	Return the status information (0x00-success, 0x01-SINDEX out of the range, 0x02-SINDEX=0 read-only SRECORD, 0x03-undefined)
Return	TRUE: success		
	FALSE: failure		

1.12.34 UHFGetSelectReceived()

Function	bool UHFGetSelectReceived(UCHAR* STATUS, UCHAR* precord)		
Description	get the selected record, RAYLINK module only		
Parameter	Output	STATUS	Return the status information (0x00- success, 0x01-SINDEX out of the range, 0x02-SINDEX=0 read-only SRECORD, 0x03-undefined)
		psRecord	Return the pointer and the structureof SRECORD (please refer to the SRECORD declaration)
Return	TRUE: success		
	FALSE: failure		

1.12.35 UHFChooseSelect()

Function	bool UHFChooseSelect (UCHAR SINDEX, UCHAR SNUM, UCHAR* STATUS)		
Description	send the parameter of Select command, RAYLINK module only		
Parameter	Input	SINDEX	the starting command (0~15) of Select command
		SNUM	SNUM—numbers of the SRECORD to the selected
	Output	STATUS	Return the status information (0x00-success, 0x01-SINDEX out of the range, 0x02-SINDEX=0 read-only SRECORD, 0x03-undefined)
Return	TRUE: success		
	FALSE: failure		

1.12.36 UHFInventoryAdvanced()

Function	bool UHFInventoryAdvanced(UCHAR SINDEX, UCHAR SNUM, UCHAR SST, UCHAR Q)		
Description	Advanced recognition command, RAYLINK module only		
Parameter	Input	SINDEX	-the starting command (0~15) of Select command
		SNUM	SNUM—send the numbers of the Select commands in SRECORD
		SST	send the parameter of Sel, Session,Target
		Q	anti-collision Q value
Return	TRUE: success		
	FALSE: failure		

1.12.37 UHFGetInventoryAdvancedStatus()

Function	bool UHFGetInventoryAdvancedStatus(UCHAR* STATUS, int* TNUM)		
Description	get the returned status of the advanced recognition and numbers of the recognized tags, RAYLINK module only		
Parameter	Input	STATUS	the advanced recognition status
	Output	TNUM	the recognized numbers of the tags
Return	TRUE: success		
	FALSE: failure		

1.12.38 UHFReadBuffer()

Function	bool UHFReadBuffer (int TINDEX, int TLEN, UCHAR* STATUS)		
Description	Read the buffer, RAYLINK module only		
Parameter	Input	TINDEX	The starting address (2 bytes)
		TLEN	Return the numbers (2 bytes)
		TMODE	TIMEsegment option (1-response TRECORDincluding TIME, 0-no TIME)
	Output	STATUS	Return the status information (0x00-follow-up response, 0x01-no follow-up response, finish the TLEN TRECORDstransmission, 0x03-no follow-up, the sequence does not exist)
Return	TRUE: success		
	FALSE: failure		

1.12.39 UHFGetBuffer()

Function	bool UHFGetBuffer (UCHAR* precord, UCHAR* STATUS)
Description	get the buffer data, RAYLINK module only

Parameter	Input	precord	Information fileld
	Output	STATUS	Return the operation status information(0x00-follow-up response、0x01-no follow-up response, finish TLEN TRECORDs transmission, 0x03-no follow-up response, the sequence does not exist)
Return	TRUE: success		
	FALSE: failure		

1.12.40 UHFGetHwType ()

Function	bool UHFGetHwType(UCHAR* version)		
Description	Get the hardware version		
Parameter	Input	Null	
	Output	version	version[0] length of the version version[1]- version[len] version data
Return	TRUE: Success		
	FALSE: Failure		

1.12.41 UHFGetRSSI ()

Function	bool UHFGetRSSI(int *RssiVal)		
Description	Get RSSI value R2000 module only		
Parameter	Input	Null	
	Output	RssiVal	RssiVal value
Return	TRUE: Success		
	FALSE: Failure		

1.12.42 UHFSetTempProtect ()

Function	bool UHFSetTempProtect(unsigned int Flag)		
Description	Set temperature protect R2000 module only		
Parameter	Input	Flag	Flag==1 as temperature protect , 0 as no
	Output	Null	
Return	TRUE: Success		
	FALSE: Failure		

1.12.43 UHFGetTempProtect ()

Function	bool UHFGetTempProtect(unsigned int *Flag)		
Description	Get temperature protect R2000 module only		
	Input	Null	

Parameter	Output	Flag	Flag==1 as temperature protect , 0 as no
Return	TRUE: Success		
	FALSE: Failure		

1.12.44 UHFGetTempValue ()

Function	bool UHFGetTempValue(unsigned int *Flag)		
Desctiption	Get temperature value R2000 module only		
Parameter	Input	Null	Get temperature value
	Output	Flag	Temperature data
Return	True: success		
	FALSE: failure		

1.12.45 UHFGetPwm ()

Function	bool UHFGetPwm(unsigned int *WorkTime,unsigned int *WaitTime)		
Description	Get Pwm R2000 module only		
Parameter	Input	Null	
	Output	WorkTime	Work time ms
		WaitTime	Wait time ms
Return	True: success		
	FALSE: failure		

1.12.46 UHFSetSingelMode ()

Function	bool UHFSetSingelMode(unsigned char mode)		
Description	Set single mode R2000 module only		
Parameter	Input	mode	0 as dual mode 1 as single mode
	Output	Null	
Return	True: success		
	FALSE: failure		

1.12.47 UHFSetPwm ()

Function	bool UHFSetPwm(unsigned int WorkTime,unsigned int WaitTime)		
Description	Set Pwm R2000 module only		
Parameter	Input	WorkTime WaitTime	Work time ms Wait time ms

	Output	Null	
Return	True: success		
	FALSE: failure		

1.12.48 UHFGetSingelMode ()

Function	bool UHFGetSingelMode(unsigned char *mode)		
Description	Get single mode R2000 module only		
Parameter	Input	Null	
	Output	mode	0 as dual mode 1 as single mode
Return	True: success		
	FALSE: failure		

1.13 Fingerprint

1.13.1 EMFingerInit()

Function	bool EMFingerInit(int iPort, int iBaudRate)	
Description	Initializa the fingerprint module	
Parameter	Input	iPort: port(1,2,3…)
		iBaudRate: 57600 as the default, it can be set;
Return	True: success	
	FALSE: failure	

1.13.2 EMFingerFree()

Function	void EMFingerFree (int iPort)	
Description	power off the fingerprint module port	
Parameter	Input	iPort: port(1,2,3…)
Return	TRUE: success	
	FALSE: failure	

1.13.3 EMGetRandomData()

Function	bool EMGetRandomData(UCHAR *pszData)	
Description	get the random data	
Parameter	Output	pszData： length +data information
Return	True: success	
	FALSE: failure	

1.13.4 EMGetImage()

Function	bool EMGetImage()
Description	get the image information and save in imgae buffer zone
Parameter	Null

Return	True: success	
	FALSE: failure	

1.13.5 EMGenChar()

Function	bool EMGenChar(UCHAR GenID)	
Description	get the image character files and save in GenID	
Parameter	Input	GenID: value 1 or 2
Return	True: success	
	FALSE: failure	

1.13.6 EMMatch()

Function	bool EMMatch(int *MatchScore)	
Description	compare the fingerprint character file templates	
Parameter	Output	MatchScore
Return	True: success	
	FALSE: failure	

1.13.7 EMSearch()

Function	bool EMSearch(UCHAR BuffID, int StartPage, int PageNum, int *PageID, int *MatchScore)	
Description	Compare and find out the data with the same character value	
Parameter	Input	BuffID: bufferID(1 or 2)
		StartPage
		PageNum: number of the ID to be searched
	Output	PageID: the searched page
		MatchScore
Return	True: success	
	FALSE: failure	

1.13.8 EMRegModel()

Function	bool EMRegModel()
Description	Combine template 1 with template 2, generate the template and save in templte buffer zone;
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.13.9 EMStorChar()

Function	bool EMStorChar(UCHAR BuffID, int PageID)	
Description	Save the template file within the specified BufferID to the fingerprint library of PageID;	
Parameter	Input	BuffID: buffer ID(1 or 2);

		PageID: (0~32768), which is assigned by the user automatically;
Return	TRUE: success	
	FALSE: failure	

1.13.10 EMLoadChar()

Function	bool EMLoadChar(UCHAR BuffID, int PageID)	
Description	Load the template in the specified PageID to the bufferID	
Parameter	Input	BuffID： bufferID(1 or 2)
		PageID: Page ID(0~32768) assigned by the user;
Return	TRUE: success	
	FALSE: failure	

1.13.11 EMUpChar()

Function	bool EMUpChar(UCHAR BuffID, UCHAR *pszData)	
Description	Save the template files in the BufferID to pszData array(512 bytes)	
Parameter	Input	BuffID: buffer ID(1 or 2)
	Output	pszData
Return	TRUE: success	
	FALSE: failure	

1.13.12 EMDownChar()

Function	bool EMDownChar(UCHAR BuffID, UCHAR *pszData)	
Description	download the file to bufferID	
Parameter	Input	BuffID： buffer ID(1 or 2)
		pszData—(512 bytes)
Return	TRUE: success	
	FALSE: failure	

1.13.13 EMDeletChar()

Function	bool EMDeletChar(int PageID, int nNum)	
Description	Delete the templates starting from the specified PageID;	
Parameter	Input	PageID: the starging Page ID
		nNum: numbers of the tempalates to be deleted;
Return	TRUE: success	
	FALSE: failure	

1.13.14 EMEmpty()

Function	bool EMEmpty()
Description	Clear all the templates in fingerprint library
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.13.15 EMSetReg()

Function	bool EMSetReg(UCHAR RegID, UCHAR nValue)	
Description	set the regester	
Parameter	Input	RegID
		nValue
Return	TRUE: success	
	FALSE: failure	

1.13.16 EMAutoEnroll()

Function	bool EMAutoEnroll(UCHAR nTime, int UserID, int *RUserID)	
Description	enroll the fingerprint twice, finish, combine and regedit automatically	
Parameter	Input	nTime: time of enrolling the fingerprint automatically
		IuserID: register ID
	Output	RuserID: user ID
Return	TRUE: success	
	FALSE: failure	

1.13.17 EMAutoMatch()

Function	bool EMAutoMatch(UCHAR nFlag, int StartPage, int PageNum, int *PageID, int *MatchScore)	
Description	record the template once, and finish the comparison automatically	
Parameter	Input	nFlag: buffer zone (1 or 2)
		StartPage
		PageNum
	Output	PageID
		MatchStore
Return	TRUE: success	
	FALSE: failure	

1.13.18 EMSetPSW()

Function	bool EMSetPSW(UCHAR *PassWord)	
Description	Set the password	
Parameter	Input	PassWord：4 bytes
Return	TRUE: success	
	FALSE: failure	

1.13.19 EMVfyPSW()

Function	bool EMVfyPSW (UCHAR *PassWord)	
Description	verify the char password	
Parameter	Input	PassWord：4 bytes
Return	TRUE: success	

	FALSE: failure

1.13.20 EMValidTempleteNum()

Function	bool EMValidTempleteNum(int *nValueModel)	
Description	Read the valid template numbers in fingerprint library	
Parameter	Output	nValueModel: the valid numbers
Return	TRUE: success	
	FALSE: failure	

1.13.21 EMReadChipSN()

Function	bool EMReadChipSN(UCHAR *SerialNum)	
Description	Reand the S/N of the chip and feedback (as the unique information of the product)	
Parameter	Output	SerialNum: 1 byte+S/N(ASCII)
Return	TRUE: success	
	FALSE: failure	

1.13.22 EMSetManuFacture()

Function	bool EMSetManuFacture(UCHAR *SerialNum)	
Description	set the S/N	
Parameter	Input	DeviceName：8 bytes (ASCII)
Return	TRUE: success	
	FALSE: failure	

1.13.23 EMSetDeviceName()

Function	bool EMSetDeviceName(UCHAR *DeviceName)	
Description	Set device name	
Parameter	Input	DeviceName：8 bytes(ASCII）
Return	TRUE: success	
	FALSE: failure	

1.13.24 EMReadSysPara()

Function	bool EMReadSysPara(UCHAR *pszData)	
Description	Obtain system Parameter	
Parameter	Output	pcData：35 bytes system information: version of fingerprint module + model (ASCII)
Return	TRUE: success	
	FALSE: failure	

1.13.25 EMUpImage()

Function	bool EMUpImage(unsigned char Mode, LPCTSTR lpFileName, unsigned int *TotalLen)
Description	upload image

Parameter	Input	mode—check if it is bmp file, 0 for not, others for bmp file
		pFile--file handle
	Output	TotalLen： total length of the obtained file (byte)
Return	TRUE: success	
	FALSE: failure	

1.14 Infrared

Note: It is only supported by C5000.

1.14.1 Infrared_init()

Function	BOOL Infrared_init()
Descrition	Power on the infrared module and open the port
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.14.2 Infrared_free()

Function	BOOL Infrared_free()
Description	Power off the infrared module and close the port
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.14.3 Infrared_send()

Function	DWORD Infrared_send(UINT8* SendBuffer,DWORD dwLength)
Description	Send data
Parameter	Input: SendBuffer, the buffer pointer of the data
	Input: dwLength, length of the data to be sent
Return	the actual length of the data to be sent

1.14.4 Infrared_receive()

Function	DWORD Infrared_receive(UINT8* ReceiveBuffer)
Description	Receive data
Parameter	Ouput: ReceiveBuffer, the buffer pointer of the data to be received
Return	the actual length of the data to be received

1.15 Backlight

1.15.1 GetBackLightLevel()

Function	DWORD GetBackLightLevel()
Description	obtain the level value of backlight
Parameter	Null
Return	Backlight level, range from 1 to 10

1.15.2 SetBackLightLevel()

Function	bool SetBackLightLevel(int iLevel)
Description	set the backlight level
Parameter	Input: iLevel, range from 1 to 10
Return	TRUE: success
	FALSE: failure

1.15.3 BackLight_Control_Switch()

Function	BOOL BackLight_Control_Switch (BOOL Switch)
Description	Open/close backlight
Parameter	Inpt: Switch, TRUE to open, FALSE to close
Return	TRUE: success
	FALSE: failure

1.16 Power

1.16.1 module_power_control()

Different using environment between different products:

C2000:

HF module power on/off; 1D module power on/off, scan enginer control; 2D hardware module power on/off, scan engine control;

C3000:

HF module powr on/off; 1D module power on/off, scan engine control;

C5000:

LF/HF/Infrared module power on/off; 1D module power on/off, scan engine control; 2D hardware module power on/off, scan engine control;

Function	BOOL module_power_control(UINT8 module,UINT8 action)
Description	Control of the power of the specified module
Parameter	Input: module,different modules, details as follows: LF: 0 HF: 1 1D: 3 2D hardware: 4 Fingerprint: 8 Infrared: 9
	Input: action, indicates the actions to the modules, like powering off 1D power , powering on 1D power, detailed as follows: Values for LF, HF, 1D, 2D hardware, fingerprint and infrared powering down are 0, powering on are 1. Values for 1D and 2D hardware that the light does not come out are 2,the opposite is 3.
Return	FALSE: failure TRUE: success

1.16.2 system_power_down()

Function	BOOL system_power_down()
Description	Power down the system
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.16.3 power_key_enabled()

Function	bool power_key_enabled()
Description	Enable power key
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.16.4 power_key_disabled()

Function	BOOL power_key_disabled()
Description	Disable power key
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.16.5 system_restart()

Function	BOOL system_restart(void)
Description	Restart the system
Parameter	Null
Return	TRUE: success
	FALSE: failure

Note: For C2000, this is supported only for the OS released later than 2013/03/13; For C5000, this is supported only for the OS released later than 2013/04.

1.17 Touch_screen

1.17.1 touch_screen_enable()

Function	BOOL touch_screen_enable()
Description	enable the touch_screen
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.17.2 touch_screen_disable()

Function	BOOL touch_screen_disable()

Description	Disable touch_screen
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.18 USB

1.18.1 usb_device_get_current_mode()

Function	UINT8 usb_device_get_current_mode()
Description	obtain the current working mode of USB slave
Parameter	Null
Return	0: USB Slave is disabled 1: storage card as U disk 2: Flash Disk as U disk 3: ActiveSync

1.18.2 usb_device_close_usb_function()

Function	BOOL usb_device_close_usb_function()
Description	Disable USB Slave
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.18.3 usb_device_storage_card_as_udisk()

Function	BOOL usb_device_storage_card_as_udisk()
Description	Set the storage card as U disk
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.18.4 usb_device_flash_as_udisk()

Function	BOOL usb_device_flash_as_udisk()
Description	Set flash disk as U disk
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.18.5 usb_device_set_as_activeSync()

Function	BOOL usb_device_set_as_activeSync()
Description	Disable USB Slave
Parameter	Null
Return	TRUE: success

	FALSE: failure

1.19 LED

1.19.1 led_control_init()

Function	bool led_control_init()
Description	Initialize the LED control
Parameter	Null
Return	TRUE: success
	FALSE: failure

1.19.2 led_control_switch()

Function	bool led_control_free()	
Description	control if the LED light on or off	
Parameter	Input	LedNum, fo the specified LED, like 1 for LED1
		Switch, for controlling on or off, true for on, false for off
Return	TRUE: success	
	FALSE: failure	

1.19.3 led_control_free()

Function	bool led_control_free()
Description	release the LED control resource
Parameter	Null
Return	TRUE: releasing successfully
	FALSE: releasing failed

1.20 Extended Serial Port

1.20.1 extended_serial_open ()

Function	BOOL extended_serial_open(UINT8 ComIndex,int iBaudRate)
Description	open the serial port by specified baudrate
Parameter	ComIndex：ComIndex=0 for COM3, ComIndex=1 for COM4, only support C5000 currently; iBaudRate: include 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 and 115200.
Return	TRUE for success
	FALSE for failure

1.20.2 extended_serial_close ()

Function	BOOL extended_serial_close(void)
Description	Close the extended serial port
Parameter	Null

Return	TRUE: success
	FALSE: failure

1.20.3 extended_serial_data_send ()

Function	int extended_serial_data_send (UINT8 *pszData, int iLength)	
Description	send the data	
Parameter	Input	pszData, data to be sent
		iLength, length of the data to be sent
Return	The actual length of the data to be sent	

1.20.4 extended_serial_data_receive ()

Function	int extended_serial_data_receive (UINT8 *pszBuffer, int iLength)	
Description	receive the data	
Parameter	Input	pszBuffer, data to be sent
		iLength, length of the data to be received
Return	The actual length of the data to be received	

1.21 Expanded API

1.21.1 READ_API_Version()

Function	void READ_API_Version(UINT8 *pszData)	
Description	Get the DeviceAPI.dll version information	
Parameter	Output	pszData, include the version information, like 'DeviceAPI V2.21(2013.03.04)'
Return	Null	

1.21.2 HardwareVersion_Ex()

Function	int HardwareVersion_Ex(UINT8 *pszData)	
Description	Obtain the version information of the LF and HF	
Parameter	Output	pszData, the hardare information
Return	length of the version (byte)	

1.21.3 SerialPortSwitch_Ex()

Note: the API is only available for C5000;

Function	bool SerialPortSwitch_Ex(UINT8 ComID)	
Description	Switch to the port of the module	
Parameter	Input	ComID, switch to the port of the module 0: switch to port of RFID LF/HF 1: switch to UHF/Infrared/Fingerprint/Expanded GPS(module being integraded the GPS and BD2) 2: 1D/2D hardware decording 3: GPS

Return	TRUE: success
	FALSE: failure

1.21.4 SerialPortControl_Ex()

Note: the API is only available for C5000;

Function	bool SerialPortControl_Ex(UINT8 uPortID, UINT8 uValue)	
Description	control the power on/off of the port	
Parameter	Input	uPortID： port number to be controlled
		uValue： 0 for setting the port low level, 1 for setting the port high level;
Return	TRUE: success	
	FALSE: failure	

1.21.5 extended_interface_power_control()

Note: the API is only available for C5000.

Function	BOOL extended_interface_power_control(UINT8 PinInterface, UINT8 InterfaceControl)
Description	Power control and GPIO control of extended module
Parameter	PinInterface, input the pin to be controlled: 1: 3.3V pin; 2: RFID 5V, supported by the new C5000 subpanel; 3: VBAT of RFID (Q302 the third pin, TP307 and the underside can be controlled—supported by the new C5000 subpanel; 4: GPIO1 of RFID---TP311, supported by the new C5000 subpanel; 5 and 6: reserved 7: 5V of the extended port 8: VBAT of the extended port (Q303 the third pin---TP301) can be controlled---supported by the new C5000 subpanel; 9: EN1 of the extended port; 10: EN2; 11: IO2-OC, the pin should be connected to the extended module or power, the output voltage is 0 when the value is 0, the output voltage is exterded module power or power voltage when the value is 1. InterfaceControl, input the action to be done, on or off the 5V. TRUE for on, FALSE for off.
Return	TRUE： success
	FALSE： failure

1.21.6 SerialPortOpen_Ex()

Note: the API is only available for C5000;

Function	bool SerialPortOpen_Ex(int iBaudRate)
Description	Open COM4 by the specified baudrate

Parameter	Input	iBaudRate: optional 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600and 115200
Return	TRUE: success, FALSE:failure	

1.21.7 SerialPortSetBaudRate_Ex()

Note: the API is only available for C5000;

Function	bool SerialPortSetBaudRate_Ex(int iBaudRate)	
Description	Set the baudrate	
Parameter	Input	iBaudRate: optional 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600and 115200
Return	TRUE: success, FALSE:failure	

1.21.8 SerialPortFunctionSwitch_Ex()

Note: the API is only available for C5000;

Function	bool SerialPortFunctionSwitch_Ex(int iModule)	
Description	Call after powering on, for RFID/Barcode switch	
Parameter	Input	iModule: 0RFID，1barcode
Return	TRUE: success, FALSE:failure	

1.22 OS functional API

1.22.1 Uniscribe_disable ()

Unicode Processor for Complex Scripts was added to the OS which is mainly used for supporting the showing language from right to left(like Arabic). Please be aware that the OS would run slower because of adding this component. This function Uniscribe_disable is used for enabling/disabling Unicode Processor for Complex Script, it works after the device is rebooted.

Function	BOOL Uniscribe_disable(BOOL Switch)
Description	If enabling the Uniscribe
Parameter	Input: Switch=FALSEfor enabling;Switch=TRUEfor disabling;
Return	FALSE: failure TURE: success

1.23 3G

Notes for 3G development:

(1)3G module would be powered on automatically once the PDA is booted;

(2)It would take at least 8 seconds to connect once the 3G module is waken up.

1.23.1 GPRSPowerOn()

Function	BOOL GPRSPowerOn ()
Description	Power on 3G module
Parameters	None
Return	TRUE: success

	FALSE：failure

Note: GPRSPowerOn() is not supported by the current OS.

1.23.2 GPRSPowerOff()

Function	BOOL GPRSPowerOff()
Description	Power off 3G module
Parameter	None
Return	TRUE：success
	FALSE：failure

Note: GPRSPowerOff() is not supported by the current OS.

1.23.3 GPRSReset()

Function	BOOL GPRSReset()
Description	Reset the module
Parameter	None
Return	TRUE：success
	FALSE：failure

Note: GPRSReset() is not supported by the current OS.

1.23.4 GPRS_judge_connection_status()

Function	BOOL GPRS_judge_connection_status(LPCTSTR GprsName)
Description	Juge the connection status with the name of GprsName
Parameter	Input：GprsName, name of GPRS connection
Return	TRUE：being connected
	FALSE：being disconnected

Note:GPRS_judge_connection_status()is not supported by the current OS.

1.23.5 GPRS_judge_modem_status()

Function	BOOL GPRS_judge_modem_status ()
Description	Judge if 3G module is powered on
Parameter	None
Return	TRUE：success
	FALSE：failure

Note: GPRS_judge_modem_status() is not supported by the current OS.

1.23.6 GPRS_open_serial()

Function	UINT8 GPRS_open_serial()
Description	Open serial port
Parameter	None
Return	0：failure
	1：success
	2：the serial port is already opened

1.23.7 GPRS_close_serial()

Function	BOOL GPRS_close_serial()
Description	Close serial port
Parameter	None
Return	TRUE：success
	FALSE：failure

1.23.8 GPRS_send_data()

Function	DWORD GPRS_send_data(UINT8* SendBuffer,DWORD dwLength)
Description	Send data
Parameter	Output: SendBuffer
	Input：dwLength
Return	Lenth of the data

1.23.9 GPRS_receive_data()

Function	DWORDGPRS_receive_data(UINT8*ReceiveBuffer,DWORD dwLength)
Description	Receive data
Parameter	Output：ReceiveBuffer
	Input: dwLength
Return	Length of the data

1.24 Bluetooth

1.24.1 bluetooth_load ()

Function	BOOL bluetooth_load()
Description	Init Bluetooth and upload the driver
Parameter	Null
Return	TRUE：success
	FALSE：failure

Note: C3000 does not support it;

1.24.2 bluetooth_unload ()

Function	BOOL bluetooth_unload()
Description	Free Bluetooth and unload the driver
Parameter	Null
Return	TRUE：success
	FALSE：failure

Note: C3000 does not support it;

1.25 Light

1.25.1 flash_light_power_control()

Function	BOOL flash_light_power_control(UINT8 action)
Description	Light control of C3000

Parameter	Input	Switch: 0 for close, 1 for open, 2 for close when the device is standby, 3 for open when device is standby
Return	TRUE：success	
	FALSE：failure	

Note: function supported by the API released later than 23rd May 2014.

1.26 Printer of C5050

Note: Only supported by C5050

1.26.1 printer_power_control()

Function	BOOL printer_power_control(BOOL Switch)
Description	Power control
Parameter	TRUE for power on, FALSE for power off
Return	TURE: success; FALSE: failure;

1.26.2 printer_open_serial()

Function	DWORD printer_open_serial(void)
Description	Open serial port
Parameter	Null
Return	0：failure
	1：success
	2：already being opened

1.26.3 printer_close_serial()

Function	BOOL printer_close_serial(void)
Description	Close the serial port
Parameter	Null
Return	TRUE：success
	FALSE：failure

1.26.4 printer_send_data()

Function	DWORD printer_send_data(UINT8* SendBuffer,DWORD dwLength)
Description	Send data
Parameter	Input: SendBuffer
	Output: dwLength
Return	Length of the data

1.26.5 printer_receive_data()

Function	DWORD printer_receive_data(UINT8* ReceiveBuffer,DWORD dwLength)
Description	Receive the printer data, like paperjam, overheating
Parameter	Input：ReceiveBuffer
	Input：dwLength
Return	Length of the data

2. CameraAPI.dll API instruction

2.1 GetCameraAPI_Version()

Function	void GetCameraAPI_Version(UINT8 *pszData)	
Description	Obtain the CameraAPI.dll version information including the CameraAPI name, version and data, like "CameraAPI V1.0(2012.12.22)"	
Parameter	output	pszData point to the obtained version information
Return	Null	

2.2 Preview()

Function	BOOL Preview(HWND hVideoWnd, LPCTSTR fileName,int resolution)	
Description	Initialize the camera module and preview	
Parameter	Input	hVideoWnd：show the control handle of the preview, m_hWnd for MFC, Handle for C#
		filename: file path to save the phote
		Resolution: (0 for 320*240, 1 for 640*480, 2 for 1024*768, 3 for 2048*1536)
Return	TRUE: success	
	FALSE: failure	

Note: this API functions include camera module initialization and preview, but we do not recommend it since it is unreasonable, and now we use CameraInit and CamearPrevies as follows:

2.3 CameraInit()

Function	void CameraInit(HWND hVideoWnd)	
Description	Initialize the camera module, use it when opening the camera application	
Parameter	Input	hVideoWnd: show the control handle, m_hWnd for MFC, Handle for C#
Return	no	

2.4 CameraPreview()

Function	void CameraPreview(int resolution)	
Description	preview	
Parameter	Input	resolution：(0 for 320*240, 1 for 640*480, 2 for 1024*768, 3 for 2048*1536)
Return	None	

2.5 TakePicture()

Function	BOOL TakePicture(LPCTSTR fileName)	
Description	photograph	
Parameter	Input	fileName：file path to save the photo
Return	TRUE: success	
	FALSE: failure	

2.6 FreeCamera()

Function	void FreeCamera()
Description	Release the camera resource
Parameter	Null
Return	None

2.7 Camera_Led_Ctrl()

Function	BOOL Camera_Led_Ctrl(BOOL swich)	
Description	C3000 flash control of camera	
Parameter	Input	Swich, TRUE for open, FALSE for close
Return	TRUE: Open/Close successfully FALSE: Open/Close failed	

Note: This function is supported by the API released later than 23rd Jan, 2014.