CipherLab
User Guide

C/C++ Programming

for 9200 Mobile Computer

Version 1.06



Copyright © 2015 CIPHERLAB CO., LTD.
All rights reserved

The software contains proprietary information of CIPHERLAB CO., LTD.; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between CIPHERLAB and the client and remains the exclusive property of CIPHERLAB CO., LTD. If you find any problems in the documentation, please report them to us in writing. CIPHERLAB does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of CIPHERLAB CO., LTD.

For product consultancy and technical support, please contact your local sales representative. Also, you may visit our web site for more information.

The CipherLab logo is a registered trademark of CIPHERLAB CO., LTD.

Other product name mentioned in this manual may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

The editorial use of these names is for identification as well as to the benefit of the owners, with no intention of infringement.

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

RELEASE NOTES

Version	Date	Notes
1.06	Jun 15, 2015	▸ Add 3.7.20 PicklistMode_2D_SE4500 ▸ Add 3.7.21 DisplayMode_2D
1.05	May 4, 2015	▸ Revise *timeout* function for ReadBarcodeData in 3.2.3 Barcode Data ▸ Revise *timeout* function for ReadRfidData and WriteRfidData in 3.2.4 RFID Data ▸ Revise *bidirectionalRedundancy* function for UserPreferences in 3.6.1 Preferences
1.04	Mar 30, 2015	▸ Add UserPreferences_Extend in 2.7.1 Preferences for 2D Scan Engine ▸ Add 5.7 User Preferred Settings
1.03	Oct 24, 2014	▸ Revise unit for Wi-Fi Roam Period
1.02	Dec 2, 2013	▸ Add 4.7 Microphone
1.01	Nov 8, 2013	▸ Modify description for DataOutputSettings code length and tag length (does not incude prefix or suffix) ▸ Update Reader Configuration application name and UI screenshots ▸ Add SCU Wi-Fi APIs: 4.7.2 Radio, 4.7.4 Domain, SDK & MAC Information - GetCurrentDomain, GetSDKVersion, 4.7.8 Configuration Profiles, 4.7.9 Edit Configuration Profile, 4.7.10 Summit Config Settings, 4.7.11 Global Config Settings, 4.7.12 Summit Registry Keys, 4.7.13 Configuration File, 4.7.14 WEP Key, 4.7.15 Pre-Shared Key, 4.7.16 LEAP Credentials, 4.7.17 EAP-FAST Credentials, 4.7.18 PEAP-GTC Credentials, 4.7.19 PEAP-MSCHAP Credentials, 4.7.20 EAP-TLS Credentials, 4.7.21 EAP-TTLS Credentials ▸ Add SCU Wi-Fi data structure:5.4.5 CF10G_STATUS, 5.4.6 CONFIG_FILE_INFO, 5.4.7 SDC_ALL, 5.4.8 SDCConfig, 5.4.9 SDC3rdPartyConfig, 5.4.10 SDCGlobalConfig, 5.4.11 Error Code (SDCERR)
1.00	Aug 19, 2013	Initial release

CONTENTS

RELEASE NOTES ..... - 3 -

INTRODUCTION 1

Development Tool 2

USING READER CONFIGURATION 3

1.1. Windows Message 5

1.1.1. Decode Data Structure – COPYDATASTRUCT 6

1.1.2. Message Queue 6

1.2. Windows Event 7

1.2.1. Decode Data Structure – DecodeMsg 8

1.2.2. Message Queue 8

USING READER DLL 9

2.1. How to Compile 10

2.2. How to Run 10

2.3. Register Decode Message 11

READER API 13

3.1. Initialize/Identify Reader 14

3.1.1. Initialization 14

3.1.2. Active Device 15

3.1.3. Reader Type 16

3.2. Obtain Data 17

3.2.1. Data Output Settings 17

3.2.2. RFID Settings 24

3.2.3. Barcode Data 28

3.2.4. RFID Data 32

3.3. Manipulate Status Indication 39

3.3.1. Notification Settings 39

3.3.2. Beeper (Speaker) 40

3.4. Obtain Essential Information 41

3.4.1. Reader DLL Version 41

3.4.2. Decoder Version 41

3.5. Configure Scan Engine — 1D CCD Scan Engine 42

3.5.1. Preferences 42

3.5.2. UPC_1D_SM1 44

3.5.3. EanJan_1D_SM1 47

3.5.4. Code39_1D_SM1 49

3.5.5. Code93_1D_SM1 51

3.5.6. Interleaved2Of5_1D_SM1 52

3.5.7. Industrial2Of5_1D_SM1 53

3.5.8. Codabar_1D_SM1 54

3.5.9. Msi_1D_SM1 56

3.5.10. GS1_DataBar_1D_SM1 58

3.5.11. Code128_1D_SM1 59

3.6. Configure Scan Engine — 1D Laser Scan Engine 60
3.6.1. Preferences 60
3.6.2. UPC_1D_SE955 64
3.6.3. EanJan_1D_SE955 67
3.6.4. Code39_1D_SE955 69
3.6.5. Code93_1D_SE955 71
3.6.6. Interleaved2Of5_1D_SE955 72
3.6.7. Industrial2Of5_1D_SE955 73
3.6.8. Chinese2Of5_1D_SE955 74
3.6.9. Codabar_1D_SE955 75
3.6.10. Msi_1D_SE955 77
3.6.11. GS1_DataBar_1D_SE955 79
3.6.12. Code128_1D_SE955 80
3.6.13. Code11_1D_SE955 81
3.7. Configure Scan Engine — 2D Scan Engine 82
3.7.1. Preferences 82
3.7.2. Upc_2D_SE4500 90
3.7.3. EanJan_2D_SE4500 93
3.7.4. Code39_2D_SE4500 95
3.7.5. Code93_2D_SE4500 97
3.7.6. Interleaved2Of5_2D_SE4500 98
3.7.7. Industrial2Of5_2D_SE4500 99
3.7.8. Matrix2Of5_2D_SE4500 100
3.7.9. Chinese2Of5_2D_SE4500 101
3.7.10. Codabar_2D_SE4500 102
3.7.11. Msi_2D_SE4500 104
3.7.12. GS1_DataBar_2D_SE4500 106
3.7.13. Code128_2D_SE4500 107
3.7.14. Code11_2D_SE4500 109
3.7.15. PostalCode_2D_4500 110
3.7.16. Composite_2D_SE4500 112
3.7.17. Inverse1D_2D_SE4500 113
3.7.18. Korean3Of5_2D_SE4500 113
3.7.19. Symbologies_2D_SE4500 114
3.7.20. picklistMode_2D_SE4500 117
3.7.21. DisplayMode_2D 118
3.8. Reset Reader 119
SYSTEM API 120
4.1. System Settings 121
4.1.1. Universally Unique Identifier (UUID) 121
4.1.2. Device Name 122
4.1.3. System Information 123
4.1.4. Start Program 123
4.1.5. Soft Reset 123
4.2. Status Indication 124
4.2.1. LED Light 124
4.2.2. Vibrator 125
4.3. LCD Backlight 126
4.3.1. Backlight Control 126
4.3.2. Backlight Level 127

4.3.3. Reset Backlight to Default 128
4.3.4. Set Backlight to Maximum 128
4.3.5. Set Backlight to Minimum 129
4.4. Keypad Backlight 130
4.5. Touch Panel 131
4.5.1. Touch Panel State 131
4.6. Keypad 132
4.6.1. Sensitivity 132
4.6.2. Keypad Lock 133
4.6.3. Keypad Mode 135
4.6.4. Keypad State 137
4.7. Microphone 139
4.7.1. Device Microphone 139
4.7.2. Headset Microphone 141
4.8. Wireless LAN 143
4.8.1. Wi-Fi Power 143
4.8.2. Radio 144
4.8.3. IP Information 145
4.8.4. Domain, SDK & MAC Information 146
4.8.5. Network List 147
4.8.6. Network Status 149
4.8.7. BSSID 150
4.8.8. Configuration Profiles 151
4.8.9. Edit Configuration Profile 152
4.8.10. Summit Config Settings 154
4.8.11. Global Config Settings 155
4.8.12. Summit Registry Keys 156
4.8.13. Configuration File 157
4.8.14. WEP Key 159
4.8.15. Pre-Shared Key 162
4.8.16. LEAP Credentials 163
4.8.17. EAP-FAST Credentials 164
4.8.18. PEAP-GTC Credentials 166
4.8.19. PEAP-MSCHAP Credentials 168
4.8.20. EAP-TLS Credentials 170
4.8.21. EAP-TTLS Credentials 172
4.9. Bluetooth 174
4.9.1. Get/SET Bluetooth POWER 176
4.9.2. Start/Stop Bluetooth 177
4.9.3. Find Devices 177
4.9.4. Pair Devices 178
4.9.5. Available Services & Device Information 179
4.9.6. Bluetooth Connection & Status 182
4.9.7. FTP Service – View Files 185
4.9.8. FTP Service – File Transfer 186
4.9.9. SPP COM Port 188
4.10. Camera 189
4.10.1. Camera Power 190
4.10.2. Camera Settings 191
4.10.3. Preview 193

4.10.4. Still Capture 196
4.10.5. Torch Mode 197
4.11. GPS 198
4.12. Button Assignment 199
4.13. Signature Capture 205
4.13.1. Initialization/Exit 205
4.13.2. Control of Signature Area 206
4.13.3. Background Color 206
4.13.4. Pen Color and Width 207
4.13.5. Image Saving & Loading 208
4.13.6. Signature DLL Version 209
DATA STRUCTURE 211
5.1. System Information Structure 212
5.1.1. SYSINFO 212
5.2. Backlight Structure 214
5.2.1. BKLCTL 214
5.3. Keypad Structure 216
5.3.1. KEYPADBIND 216
5.4. Wi-Fi Structures 217
5.4.1. RAW_DATA 217
5.4.2. WLANADPTINFO 218
5.4.3. WLANCONNECTEDST 219
5.4.4. WLANCTL 220
5.4.5. CF10G_STATUS 222
5.4.6. CONFIG_FILE_INFO 225
5.4.7. SDC_ALL 226
5.4.8. SDCConfig 227
5.4.9. SDC3rdPartyConfig 231
5.4.10. SDCGlobalConfig 233
5.4.11. Error Code (SDCERR) 239
5.5. Bluetooth Structure 240
5.5.1. FTP_FILE_INFO 240
5.6. Camera Structures 241
5.6.1. PICSTATE 241
5.6.2. FLASH 242
5.7. User Preferred Settings 243
5.7.1. UserPreferencesParameters 243
SCAN ENGINE SETTINGS 245
Symbologies Supported 246
RFID Tags Supported 248
INDEX 249

INTRODUCTION

This programming guide contains the necessary info for building applications that can tune reader module(s), capture data, or control built-in hardware on the 9200 Mobile Computer, a multi-functional device powered by Windows Embedded Handheld 6.5.

For data capture, dedicated **Reader Configuration** utility enables streaming decoded data to your applications by either Windows Message or Windows Event. What makes it more flexible is an alternate Reader DLL (dynamic-link library) provided alongside the mobile comptuer so a variety of reader functions can be implemented for your self-created application(s).

For system hardware, a collection of system functions is included herein for your application to exploit in order to control the hardware modules on board the mobile compute.

This document makes it easy to integrate your application(s) to the 9200 platform. We recommend that you keep a copy of this document at hand for quick reference.

Thank you for choosing CipherLab products!

DEVELOPMENT TOOL

Using the API herein requires Windows programming knowledge. For the information on Windows application programming interface (API), essential resource is available on Microsoft website –

http://msdn2.microsoft.com/en-us/library/default.aspx

As the 9200 mobile computer is powered by Windows Embedded Handheld 6.5, the environment and application development tool should be one of the following:

- Visual Studio 2008
- Visual Studio 2005
- 9200 SDK

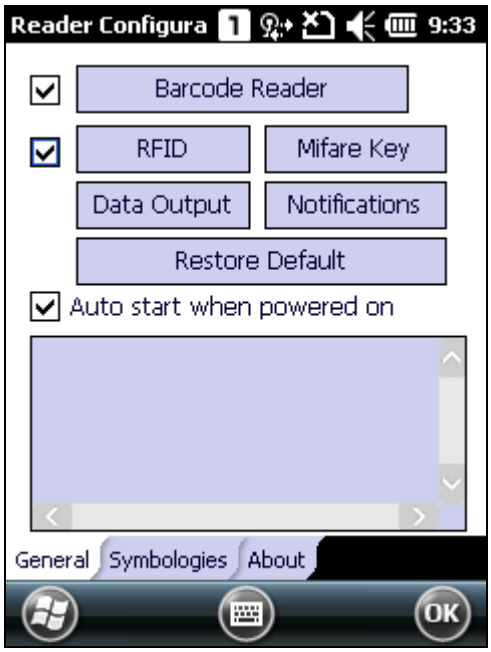
Chapter 1

USING READER CONFIGURATION

Reader Configuration is a utility preinstalled on the mobile computer to configure reader module(s). To open it:

1) On Start screen, tap **Settings** | **System** | **Reader Configuration** .

   The utility opens in context with the available reader(s) on board the mobile computer.



   The utility opens showing **General** tabbed page while hiding **Symbologies** and **About** tabbed pages.

2) Activate the reader(s) to use on **General** tabbed page and define the Symbologies to read on **Symbologies** tabbed page.
3) On **General** tabbed page, tap **Data Output**.

   A page opens providing settings to output decoded data.

4) Sets where to output data to by selecting between **Keyboard Emulation**, **Windows Message** or **Windows Event**.

Setting	Description
Keyboard Emulation	Streams decoded data to the active application as if it was typed. Run your application or any built-in text editor to admit the data.
Windows Message	Sends a Windows message to your application after data decoding. ▸ See Windows Message for details.
Windows Event	Sends a Windows event to your application after data decoding. ▸ See Windows Event for details.

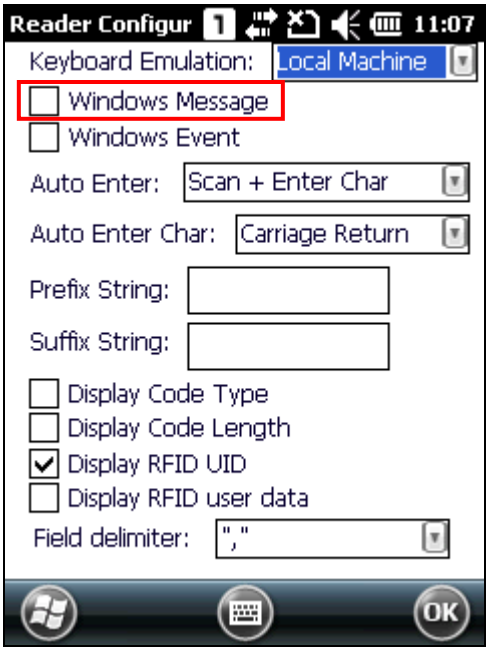
IN THIS CHAPTER

1.1 Windows Message 5
1.2 Windows Event 7

1.1. WINDOWS MESSAGE

To use **Windows Message** to stream data to your application:

1) Open Start screen, tap **Settings** | **System** | **Reader Configuration** | **General** tabbed page | **Data Output**.
2) Select **Windows Message**.



Actions for your application to take using Windows Message:

- Register a Windows Message *WM_DECODE* –
  `DecodeMsg = RegisterWindowMessage(TEXT("WM_DECODE"));`

  Whenever a decoded data comes about, identifier of this Windows message broadcasts, and data gets stored in the Windows message queue.
- Intercept decode message *WM_DECODE*.
- Call Windows API (ReadMsgQueue) to retrieve the decoded data.
  `ReadMsgQueue(hQueue,&buffer,sizeof(buffer),&dwRW,0,&dwFlag);`

1.1.1. DECODE DATA STRUCTURE – COPYDATASTRUCT

Data is stored in *COPYDATASTRUCT* structure, which locates in header file **winuser.h**. The decoded data is passed to your application through *lParam* parameter.

```
Typedef struct tagCOPYDATASTRUCT
      {
DWORD               dwData;
DWORD               cbData;
PVOID               lpData;
} COPYDATASTRUCT, *PCOPYDATASTRUCT;
```

1.1.2. MESSAGE QUEUE

Create a queue named *ReaderConfigQueue*. For example,

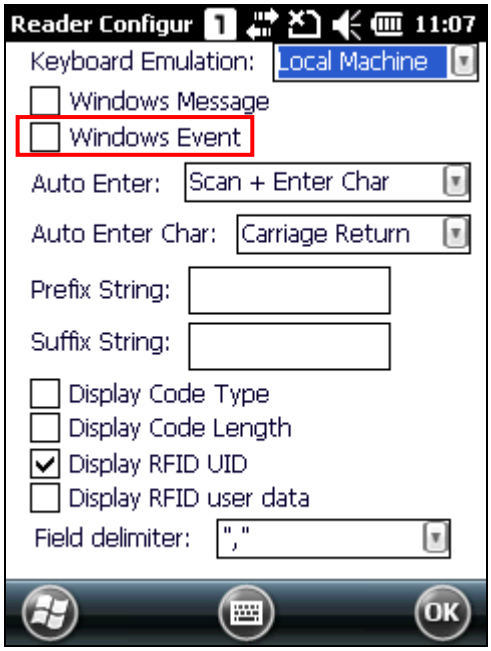
```
HANDLE                  hQueue;
COPYDATASTRUCT          buffer;
MSGQUEUEOPTIONS         QueueOptions;

QueueOptions.dwSize = sizeof(QueueOptions);
QueueOptions.dwMaxMessages = 1;
QueueOptions.cbMaxMessage = sizeof(COPYDATASTRUCT);
QueueOptions.bReadAccess = TRUE;
hQueue = CreateMsgQueue(TEXT("ReaderConfigQueue"),&QueueOptions);
```

1.2. WINDOWS EVENT

To use **Windows Event** to stream data to your application:

1) Open **Start screen** | **Settings** | **System** | **Reader Configuration** | **General** tabbed page | **Data Output**.
2) Select **Windows Event**.



Actions for your application to take:

- Create a Windows event object *DecodeEvent* –
  `hEvent = CreateEvent(NULL,FALSE,FALSE,TEXT("DecodeEvent"));`

  Whenever a decoded data comes about, the Windows event object is signaled, and decoded data gets stored in the Windows message queue.
- Intercept the decode event *DecodeEvent*.
- Call Windows API (ReadMsgQueue) to retrieve the decoded data.
  `ReadMsgQueue(hQueue,&DecodeMsg,sizeof(DecodeMsg),&dwRW,0,&dwFlag);`

1.2.1. DECODE DATA STRUCTURE – DECODEMSG

Data is stored in *DecodeMsg* structure.

```
Typedef struct tagDecodeMsg
      {
BYTE                cDeviceType;
BYTE                cCodeType[4];
INT                 nCodeLen;
BYTE                szRFID_UID[21];

BYTE                Reserve[100];
BYTE                szCodeData[4096+1];
} DECODEMSG, *LPDECODEMSG;
```

1.2.2. MESSAGE QUEUE

Create a queue named *ReaderConfigQueue*. For example,

```
HANDLE                  hQueue;
DECODEMSG               DecodeMsg;
MSGQUEUEOPTIONS         QueueOptions;

QueueOptions.dwSize = sizeof(QueueOptions);
QueueOptions.dwMaxMessages = 1;
QueueOptions.cbMaxMessage = sizeof(DECODEMSG);
QueueOptions.bReadAccess = TRUE;
hQueue = CreateMsgQueue(TEXT("ReaderConfigQueue"),&QueueOptions);
```

Chapter 2

USING READER DLL

Dynamic-link library (DLL) comes as a module with functions and data that are useful and linkable for an application or another DLL. Explicit (run-time) linking makes it possible for an application or another DLL to include only the information that locates an exported DLL function at run-time.

Such DLL is loaded by an application rather than the operating system. And it is loaded in application run-time rather than application load-time. If the DLL to load is absent, function **LoadLibrary()** is to fail.

It is the application that checks for the errors returned by **LoadLibrary()** and transacts accordingly. Upon loading the DLL successfully, the application has to get the addresses of the exported DLL functions by calling **GetProcAddress()**.

Reader DLL works based on said mechanism to deliver a simple control over your reader applications. Find the necessary files on CD-ROM as listed below before starting out writing your reader application(s):

- ReaderDllMobile.dll
- ReaderDll.h

IN THIS CHAPTER

2.1 How to Compile 10
2.2 How to Run 10
2.3 Register Decode Message..... 11

2.1. HOW TO COMPILE

The header file must locate in C source code directory. Specify the path to DLL file in **LoadLibrary()** and **FreeLibrary()**.

1) A process must call **LoadLibrary()** to load Reader DLL at run-time.

   In the parentheses, assigning only the file name of the DLL to load is to provide a relative directory path to the DLL. For example:

   ```
   HINSTANCE hDllInst = LoadLibrary("ReaderDllMobile.dll");
   ```

   In this example, the function is to load a DLL file sharing the same whereabouts with the application. This means the DLL to load locates in the folder where the application is.

   Assigning a full path is another thing, it means providing an absolute direcotry path to the DLL locating somewhere on the mobile computer's storage. For example,

   ```
   HINSTANCE hDllInst = LoadLibrary("\\Windows\\ReaderDllMobile.dll");
   ```

   In this example, the DLL file settles in the mobile computer's "Windows" folder.

2) After loading Reader DLL, the process calls **GetProcAddress()** to get the address of an exported function in the DLL. Then it calls the exported DLL function using the function pointer returned by **GetProcAddress()**.

   For example:

   ```
   InitReader = (INITREADER)GetProcAddress(hDllInst,TEXT("InitReader"));
   ```

   where InitReader is declared as function pointers.

3) When the process ends or the DLL is no more needed, the process must call **FreeLibrary()** to unload the DLL and release the resources so reader module(s) can be powered off.

Note: If a full path is specified in **LoadLibrary()** and **FreeLibrary()**, make sure to copy the DLL to load to the specified path.

2.2. HOW TO RUN

To run your application:

1) Copy the executeable file of your application (.exe) to the mobile computer's internal storage or external storage if the internal storage falls short.

Note: If you decide to store your application on external storage, pay attention to the path assigned to locate the DLL to load.

2) Run the executable file of your applicataion.

   Your application references the path specified in **LoadLibrary()** and loads the DLL.

   For example:

   ```
   HINSTANCE hDllInst = LoadLibrary("\\Windows\\ReaderDllMobile.dll");
   ```

   Your application later quits and unloads the DLL specified in **FreeLibrary()**.

2.3. REGISTER DECODE MESSAGE

To retrieve decoded data, the application has to “register” message *WM_DECODEDATA*.

For example,

```
UNIT nDecodeMsg = RegisterWindowMessage(TEXT(“WM_DECODEDATA”));
```

In this example, *nDecodeMsg* is the message identifier that broadcasts when 1D (laser) reader or RFID reader decodes data.

So the following happens whenever decoded data comes about:

- a *WM_DECODEDATA* message broadcasts when 1D (laser) reader or RFID reader decodes data.
- Your application retrieves message *WM_DECODEDATA* and uses *wParam* to identify decoded data.

For example,

```
 if(wParam == DC_READER_BC) {
    GetDecodeType()
    GetDecodeData()
    GetOutputRecord()
}

else if(wParam == DC_READER_RFID) {
    GetDecodeTagType()
    GetDecodeRfidUID()
    GetDecodeRfidData()
    GetOutputRecord()
}
```

Note: **ReadBarcodeData()** and **ReadRfidData()** scan and retrieve data without intercepting *WM_DECODEDATA*, which is applicable to scan data by tapping a GUI button. All they need to do is to invoke the function in the clicking event handler function.

Chapter 3

READER API

Reader DLL provides a slew of functions for your self-made application to exploit. It implements Windows application programming interface (API). Function prototypes of library routines can be found in library header file. Start a process by **InitReader()**.

Find the necessary files on CD-ROM before starting out writing your application.

Header required:	Lib required:	DLL required:
ReaderDll.h	N/A	ReaderDllMobile.dll

IN THIS CHAPTER

3.1 Initialize/Identify Reader 14
3.2 Obtain Data 17
3.3 Manipulate Status Indication 39
3.4 Obtain Essential Information 41
3.5 Configure Scan Engine — 1D CCD Scan Engine 42
3.6 Configure Scan Engine — 1D Laser Scan Engine 60
3.7 Configure Scan Engine — 2D Scan Engine 82
3.8 Reset Reader 119

3.1. INITIALIZE/IDENTIFY READER

3.1.1. INITIALIZATION

InitReader	
Purpose	Initializes reader module(s).
Syntax	**INT InitReader (VOID);**
Example	`INT nRlt;` `nRlt = InitReader();`
Return Value	If successful, it returns 0. Otherwise, it returns a negative value to indicate a specific error:

-102	E_READER_INIT_FAILED
-103	E_NO_READER_INSTALLED
-111	E_READER_BC_RESET_FAILED
-112	E_READER_RFID_RESET_FAILED (See remarks below.)

Remarks	As this is a function that gets barcode reader ready, it has to be called before any other functions.. ▸ If it fails to access reader settings, it auto-resets the reader. ▸ Call GetReaderType to check the reader module integrated on the mobile computer.
See Also	GetActiveDevice, GetReaderType, SetActiveDevice

3.1.2. ACTIVE DEVICE

GetActiveDevice

Purpose: Gets the currently active reader.

Syntax: **INT GetActiveDevice (VOID);**

Example:
```
INT nRdrType;
nRdrType = GetActiveDevice();
```

Return Value: If successful, it returns the reader type(s) according to the following:

7	DC_READER_BC	Only barcode reader is active.
32	DC_READER_RFID	Only RFID reader is active. (= Barcode reader is inactive!)
255	ALL_DEVICE	Both readers are active.

Otherwise, it returns a negative value to indicate a specific error:

0	NO_ACTIVE_DEVICE
-101	E_READER_NOT_INIT

Remarks: Reader activity allows three possibilities:

- Only (1D or 2D) barcode reader is active.
- Only RFID reader is active.
- Both barcode reader and RFID reader are active.

See Also: GetReaderType, InitReader, SetActiveDevice

SetActiveDevice

Purpose: Activates the reader to use.

Syntax: **INT SetActiveDevice (INT** *actDC***);**

Parameters: *actDC*

[in] A value that specifies the active device.

0	NO_ACTIVE_DEVICE	Deactivates all active readers.
7	DC_READER_BC	Activates only barcode reader.
32	DC_READER_RFID	Activates only RFID reader (with barcode reader deactivated).
255	ALL_DEVICE	Activates both readers.

Example:
```
INT nRlt;
nRlt = SetActiveDevice(ALL_DEVICE);
```

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-101	E_READER_NOT_INIT

Remarks: The trigger key only triggers the active reader to scan. The inactive reader won't incur any event.

An error does not occur if the specified reader type is not found!

See Also: GetReaderType, GetActiveDevice, InitReader

3.1.3. READER TYPE

GetReaderType

Purpose	Gets the reader type on board.
Syntax	**INT GetReaderType (VOID);**
Example	`INT nRdrType;` `nRdrType = GetReaderType();`
Return Value	If successful, it returns the reader type(s) accordingly:

128	ID_MOD_1D_955
256	ID_MOD_MP_RFID
512	ID_MOD_2D_4500
1024	ID_MOD_1D_SM1
384	ID_MOD_1D_955 + ID_MOD_MP_RFID
768	ID_MOD_2D_4500 + ID_MOD_MP_RFID
1280	ID_MOD_1D_SM1 + ID_MOD_MP_RFID

Otherwise, it returns a negative value to indicate a specific error:

0	Could be (1) reader not initialized, or (2) no reader installed!

Remarks

Reader integration allows three possibilities:

- Only (1D or 2D) barcode reader is present.
- Only RFID reader is present.
- Both barcode reader and RFID reader are present.

See Also: InitReader, GetActiveDevice, SetActiveDevice

3.2. OBTAIN DATA

3.2.1. DATA OUTPUT SETTINGS

Processed Data

DataOutputSettings() sets which info to attach to a decoded barcode data:

Info	Length (Byte)	Description
Code Type	1	Barcode Type. See **DataOutputSettings()** parameter *showCodeType*.
Code Length	4	Decoded barcode length (excluding prefix and/or suffix codes). See **DataOutputSettings()** parameter *showCodeLen*.
Prefix Code	0~10	No prefix code if value is 0. See **DataOutputSettings()** parameters *prefixCode* and *prefixCodeLen*.
Decode Data	1~4096	Decoded barcode data
Suffix Code	0~10	No suffix code if value is 0. See **DataOutputSettings()** parameters *suffixCode* and *suffixCodeLen*.

- When keyboard emulation is enabled, barcode data is output to the active application locally.
- When keyboard emulation is disabled, use **GetOutputRecord()** to obtain processed data upon receiving message *WM_DECODEDATA*. Refer to Register Decode Message.

Note: Data fields by sequence may include –
[Code Type], [Code Length], [Prefix Code], [Decode Data], [Suffix Code]

DataOutputSettings() and **RfidSettings()** set which info to attach to a decoded RFID data:

Info	Length (Byte)	Description
Tag Type	1	RFID Tag type. See **DataOutputSettings()** parameter *showCodeType*.
Tag Length	4	Decoded data length (excluding prefix and/or suffix codes). See **DataOutputSettings()** parameter *showCodeLen*.
		Tag length may be - UID only - Data only - UID + Data
Prefix Code	0~10	No prefix code if value is 0. Refer to **DataOutputSettings()** parameter *prefixCode* and *prefixCodeLen*.
Tag UID	1~20	RFID tag's Universal Identification (UID). Refer to **RfidSettings()** parameter *readUid*.
Suffix Code	0~10	No suffix code if value is 0. See **DataOutputSettings()** parameter *suffixCode* and *suffixCodeLen.*
Delimiter	1	The character to separate UID and data. Refer to **RfidSettings()** parameter *useDelim.* No delimiter if value is 0x00.

Tag Data	1~4096	Decoded RFID data. Refer to **RfidSettings()** parameter *readUserData*.

- When keyboard emulation is enabled, RFID data is output to the active application locally.
- When keyboard emulation is disabled, use **GetOutputRecord()** to obtain processed data upon receiving message *WM_DECODEDATA*.

Note: Data fields by sequence may include:
[Tag Type], [Tag Length], [Prefix Code], [Tag UID], [Suffix Code], [Delimiter], [Tag Data]. Note if UID is disabled, output data will not include prefix code, UID, suffix code, and delimiter.

Raw Data

Use the following functions to obtain raw data —

- **GetDecodeData()** and **GetDecodeType()** for printed barcodes.
- **GetDecodeRfidData()**, **GetDecodeRfidUID()** and **GetDecodeTagType()** for RFID tags.

DataOutputSettings

Purpose

Sets where and how to ouptut data.

- "How to output" means which info to attach to a decoded data such as code/tag type, data length, prefix code, suffix code, etc.

Syntax

```
INT DataOutputSettings (INT rw,
                        INT *enableKeyboardEmulation,
                        INT *autoEnterWay,
                        INT *autoEnterChar,
                        INT *showCodeType,
                        INT *showCodeLen,
                        CHAR *prefixCode,
                        INT prefixCodeSize,
                        CHAR *suffixCode,
                        INT suffixCodeSize);
```

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets output settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes output settings

enableKeyboardEmulation

[in][out] A value that specifies whether to emulate data as typed text to subsequently pass it to the active application.

0*	Disables keyboard emulation - No other parameters will take effect unless **GetOutputRecord()** is called.
1	Emulates keyboard input on local machine
2	Emulates keyboard input on remote PC (RDP server via programs like Remote Desktop Connection, Terminal Services Client, etc.)

autoEnterWay

[in][out] A value that specifies whether to auto-affix a character after decoding.

0	Disables
1*	Suffixes the decoded data (= decoded data + Enter-character)
2	Prefixes the decoded data (= Enter-character + decoded data)

autoEnterChar

[in][out] A value that specifies the character to auto-affix.

0	None
1*	Carriage Return (= 0x0d)
2	Tab
3	Space (= 0x20)
4	Comma (= 0x2c)
5	Semicolon (= 0x3b)

showCodeType

[in][out] A value that specifies whether to transmit barcode type or RFID tag type in data records...

0*	Does not transmit
1	Transmits

showCodeLen

[in][out] A value that specifies whether to transmit code length for a barcode or RFID tag in data records.

- In case only UID is to be read, code length is UID length.

0*	Does not transmit
1	Transmits

prefixCode

[in][out] Pointer to a buffer where prefix code is stored.

- For RFID decoding, UID must be enabled so the output includes prefix code, UID, and suffix code.

prefixCodeSize

[in] Size of the buffer where prefix code is stored.

suffixCode

[in][out] Pointer to a buffer where suffix code is stored.

- For RFID decoding, UID must be enabled so the output includes prefix code, UID, and suffix code.

suffixCodeSize

[in] Size of the buffer where suffix code is stored.

Return Value

If successful, it returns 0

Otherwise, it returns 1.

Remarks

Depending on reader type and associated reader settings, the fields of output record may differ.

When data comes from barcode reader, data fields may include:

`[Code Type][Code Length][Prefix Code][Decode Data][Suffix Code]`

[Code Type]:	This field is output only when *showCodeType* value is non-zero.
[Code Length]:	This field is output only when *showCodeLen* value is non-zero. (Prefix/suffix codes are not included.)
[Prefix Code]:	This field is output only when *prefixCode* value is non-zero.
[Decode Data]:	This field is output only when the data has been decoded.
[Suffix Code]:	This field is output only when *suffiixCode* value is non-zero.

When data comes from RFID reader, data fields may include:

- If UID is disabled, output will not include prefix code, UID, suffix code, and delimiter.

`[Tag Type][Tag Length][Prefix Code][Tag UID][Suffix Code][Delimiter][Tag Data]`

[Tag Type]:	This field is output only when *showCodeType* value is non-zero.
[Tag Length]:	This field is output only when *ShowCodeLen* value is non-zero.

	When specified to read UID only, tag length is UID length. (Prefix/suffix codes are not included.)
[Prefix Code]:	This field is output only when *prefixCode* value is non-zero.
[Tag UID]:	This field is output only when **RfidSettings()** *readUID* value is non-zero.
[Suffix Code]:	This field is output only when *suffiixCode* value is non-zero.
[Delimiter]:	This field is output only when **RfidSettings()** *useDelim* value is non-zero.
[Tag Data]:	This field is output only when **RfidSettings()** *readData* value is non-zero.

See Also GetDecodeData, GetDecodeType, GetOutputRecord
GetDecodeRfidData, GetDecodeRfidUID, GetDecodeTagType, RfidSettings

GetOutputRecord

Purpose

Sets how to output data for keyboard emulation.

- Means seting which info to attach to decoded data such as code/tag type, code length, prefix code, suffix code, etc.

Syntax

INT GetOutputRecord (INT *target*,
CHAR **buf*,
INT *bufSize*);

Parameters

Target

[in] A value that specifies where the decoded data comes from.

7	DC_READER_BC	from barcode reader.
32	DC_READER_RFID	from RFID reader.

Buf

[out] Pointer to a buffer where decoded data is stored.

bufSize

[in] Maximum number of bytes to receive; normally equals to buffer size.

Example

```
CHAR buf[2048] = {'\0'};
INT nRlt;
nRlt = GetOutputRecord(DC_READER_BC, buf, sizeof(buf));
nRlt = GetOutputRecord(DC_READER_RFID, buf, sizeof(buf));
```

Return Value

If successful, it returns the number of bytes obtained (excluding terminating null character).

Otherwise, it returns a negative value to indicate a specific error:

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

Remarks

This function is called before another successful decoding to obtain the data just decoded. Depending on reader type and associated reader settings, output may differ.

When data comes from barcode reader, data fields may include:

`[Code Type][Code Length][Prefix Code][Decode Data][Suffix Code]`

[Code Type]:	This field is output only when **DataOutputSettings()** *showCodeType* value is non-zero.
[Code Length]:	This field is output only when **DataOutputSettings()** *showCodeLen* value is non-zero. (Prefix and/or suffix codes are not included.)
[Prefix Code]:	This field is output only when **DataOutputSettings()** *prefixCode* value is non-zero.
[Decode Data]:	This field is output when data gets decoded.
[Suffix Code]:	This field is output only when **DataOutputSettings()** *suffiixCode* value is non-zero.

When data comes from RFID reader, data fields may include:

- Note if UID is disabled, output will not include prefix code, UID, suffix code, and the delimiter.

`[Tag Type][Tag Length][Prefix Code][Tag UID][Suffix Code][Delimiter]`
`[Tag Data]`

[Tag Type]:	This field is output only when **DataOutputSettings()** *showCodeType* value is non-zero.
[Tag Length]:	This field is output only when **DataOutputSettings()** *ShowCodeLen* value is non-zero. When specified to read UID only, it refers to UID length. (Prefix and/or suffix codes are not included.)
[Prefix Code]:	This field is output only when **DataOutputSettings()** *prefixCode* value is non-zero.
[Tag UID]:	This field is output only when **RfidSettings()** *readUID* value is non-zero.
[Suffix Code]:	This field is output only when **DataOutputSettings()** *suffiixCode* value is non-zero.
[Delimiter]:	This field is output only when **RfidSettings()** *useDelim* value is non-zero.
[Tag Data]:	This field is output only when **RfidSettings()** *readData* value is non-zero.

See Also DataOutputSettings, GetDecodeData, GetDecodeType, GetDecodeRfidData, GetDecodeRfidUID, GetDecodeTagType, RfidSettings

3.2.2. RFID SETTINGS

RfidSettings

Purpose

Sets what to collect, where to start collecting from, and how much data to collect from an RFID tag, and sets how to output RFID data.

- "How to output RFID data" means which info to attach to a decoded RFID data such as tag length, prefix code, tag UID, suffix code, delimiter.

Syntax

```
INT RfidSettings (INT rw,
                  INT *readUid,
                  INT *readUserData,
                  INT *userDataStartByte,
                  INT *readUserDataLen,
                  INT *useDelim,
                  INT *timeout,
                  CHAR *mifareLoginKey,
                  INT loginKeyLength,
                  INT *loginKeyType);
```

Parameters

Default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets RFID settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Sets RFID settings

readUID

[in][out] A value that specifies whether to decode UID.

0	Does not decode
1*	Decodes

readUserData

[in][out] A value that specifies whether to decode encoded data.

0*	Does not decode
1	Decodes

userDataStartByte

[in][out] A value that specifies where to start collecting data from.

As different RFID tags may have different capacity, when a value greater than allowed is passed for this parameter, it is auto-adjusted to the maximum value allowed on a tag.

-1*	A value that specifies where to start reading from. "-1" means the RFID reader starts reading from the default page of an RFID tags. The following is a table of various RFID tags' default pages:

Tag type	Standard	Default Page
Mifare	(ISO 14443A)	**4**
ICODE SLI	(ISO 15693)	**3**
LRI512	(ISO 15693)	**0**

SRF55VxxP	(ISO 15693)	**3**
Tag-it	(ISO 15693)	**0**
Others	(ISO 15693)	**0**
ICODE	(Phillips)	**5**

readUserDataLen

[in][out] A value that specifies how much data to collect, which is set to 128 bytes by default. As the capacity varies from tag to tag, when the passed value is greater than allowed, it is auto-adjusted to the maximum value allowed on an RFID tag.

userDelim

[in][out] An ASCII value that specifies the delimiter in use.

0*	No delimiter
1~127	Adds a delimiter between UID and data when both are decoded

timeout

[in][out] A value that specifies the elapsed time before a scanning session* ends.

1~5	3* (second)

mifareLoginKey

[in][out] Pointer to a buffer where the security key (such as Mifare) is stored in Reader DLL. Such key is the authentication to access a specific RFID tag. Set to "FFFFFFFFFFFF" by default.

- This key must be a hex string with 12 bytes length.
- This key applies to all the sectors covered by the range formulated by "userDataStartByte" and "readUserDataLen".

LoginKey Example	*Key String*
0	(Keep the current string)
"F1F2F3F4F5F6"	F1F2F3F4F5F6

loginKeyLength

[in] Pass NULL to ignore this parameter when overwriting the current key string. When you want to read out the current key string stored in Reader DLL, pass a length of the input buffer to this parameter.

- This function does not allow changing security keys (Key A or Key B) of a Mifare card. Instead, you must call **ChangeMifareKey()**.

loginKeyType

[in][out] A value that specifies the login key type.

0*	Key A
1	Key B

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate an error condition:

-253	E_WRONG_READER_TYPE

Remarks

Data fields may include:

`[Tag Type][Tag Length][Prefix Code][Tag UID][Suffix Code][Delimiter][Tag Data]`

[Tag Type]:	This field is output only when **DataOutputSettings()**

	showCodeType value is non-zero.
[Tag Length]:	This field is output only when **DataOutputSettings()** *ShowCodeLen* value is non-zero. When RFID reader is set to read UID only, UID length applies to tag length. (Prefix/suffix codes are not included.)
[Prefix Code]:	This field is output only when **DataOutputSettings()** *prefixCode* value is non-zero.
[Tag UID]:	This field is output only when **RfidSettings()** *readUID* value is non-zero.
[Suffix Code]:	This field is output only when **DataOutputSettings()** *suffiixCode* value is non-zero.
[Delimiter]:	This field is output only when **RfidSettings()** *useDelim* value is non-zero.
[Tag Data]:	This field is output only when **RfidSettings()** *readData* value is non-zero.

- Note if UID is disabled, output will not include prefix code, UID, suffix code, and delimiter.

See Also ChangeMifareKey, GetDecodeRfidData, GetDecodeRfidUID GetDecodeTagType, DataOuputSettings, GetOutputRecord

Note: (1) A scanning session starts when trigger key is hit.
(2) The specified timeout should be greater than the time it takes to press the trigger.
(3) RFID tags may support authentication for security concerns, such as Mifare Standard 1K/4K, and so on.

ChangeMifareKey	
Purpose	Changes Mifare card security key — Key A or Key B of a specific sector.
Syntax	**INT ChangeMifareKey (INT** *loginKeyType*, **INT** *useDefaultKey*, **INT** *defaultKeyIndex*, **INT** *targetSector*, **CHAR** **loginKey*, **CHAR** **newKeyA*, **CHAR** **newKeyB*, **CHAR** **accessControl*);
Parameters	*loginKeyType* Reserved [in] Pass NULL to ignore this parameter. *useDefaultKey* Reserved [in] Pass NULL to ignore this parameter. *defaultKeyIndex* Reserved [in] Pass NULL to ignore this parameter. *targetSector* [in] The sector to change Mifare security key. *loginKey* [in] Pass Key A that logs in the target sector. *newKeyA* [in] New Key A to login to the sector. If not changing this key, pass the old Key A instead. This parameter must be a character array of 6 bytes length. *newKeyB* [in] New Key B to loging to the sector. If not changing this key, pass the old Key B instead. This parameter must be a character array of 6 bytes length. *accessControl* Reserved [in] Pass NULL to ignore this parameter.
Return Value	If successful, it returns 0. Otherwise, it returns 1.
Remarks	Due to security concerns, security keys cannot be read. Make sure the key values are correct.
See Also	RfidSettings

3.2.3. BARCODE DATA

GetDecodeData() and **GetDecodeType()** are the functions to get raw data from a barcode.

GetDecodeData	
Purpose	Gets decoded barcode data upon pressing a physical trigger.
Syntax	**INT GetDecodeData (BYTE** **buf***,** **INT** *bufSize***);**
Parameters	*buf* [out] Pointer to a buffer where decoded data is stored. *bufSize* [in] Maximum number of bytes to receive; normally equals to buffer size.

Example	`INT nRlt;`
	`BYTE szData[128] = {'\0'};`
	`nRlt = GetDecodeData(szData, sizeof(szData));`

Return Value	If successful, it returns the number of obtained bytes (excluding terminating null character). Otherwise, it returns a negative value to indicate a specific error:
Remarks	This function is called to obtain just decoded data right before another successful decoding. The decoded data then hooks up Prefix Code or Suffix Code if they are applicable.
See Also	DataOuputSettings, GetDecodeType, GetOutputRecord

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

GetDecodeType

Purpose: Gets decoded barcode's code type upon pressing a physical trigger.

Syntax: **INT GetDecodeType (VOID);**

Example:

```
INT nType;
nType = GetDecodeType();
```

Return Value: If successful, it returns code type.
Otherwise, it returns a negative value to indicate a specific error:

-101	E_READER_NOT_INIT

Remarks: This function is called to get the currently decoded barcode's code type right before another successful decoding.

Code Type		Symbology
0x40	64 (@)	ISBT 128
0x41	65 (A)	Code 39
0x42	66 (B)	Italian Pharmacode (Code 32)
0x43	67 (C)	French Pharmacode (CIP 39)
0x44	68 (D)	Industrial 25
0x45	69 (E)	Interleaved 25
0x46	70 (F)	Matrix 25
0x47	71 (G)	Codabar (NW7)
0x48	72 (H)	Code 93
0x49	73 (I)	Code 128
0x4A	74 (J)	UPC-E0
0x4B	75 (K)	UPC-E0 with Addon 2
0x4C	76 (L)	UPC-E0 with Addon 5
0x4D	77 (M)	EAN-8
0x4E	78 (N)	EAN-8 with Addon 2
0x4F	79 (O)	EAN-8 with Addon 5
0x50	80 (P)	EAN-13 (also UPC-A on CCD/Laser scan engine)
0x51	81 (Q)	EAN-13 with Addon 2
0x52	82 (R)	EAN-13 with Addon 5
0x53	83 (S)	MSI
0x54	84 (T)	Plessey
0x55	85 (U)	GS1-128 (EAN-128)
0x56	86 (V)	Undefined
0x57	87 (W)	Undefined
0x58	88 (X)	Undefined
0x59	89 (Y)	Undefined

0x5A	90 (Z)	Telepen
0x5B	91 ([ )	GS1 DataBar Omnidirectional (RSS-14)
0x5C	92 (\)	GS1 DataBar Limited (RSS Limited)
0x5D	93 ( ])	GS1 DataBar Expanded (RSS Expanded)
0x5E	94 (^)	UPC-A
0x5F	95 (_)	UPC-A with Addon 2
0x60	96 (‘)	UPC-A with Addon 5
0x61	97 (a)	UPC-E1
0x62	98 (b)	UPC-E1 with Addon 2
0x63	99 (c)	UPC-E1 with Addon 5
0x64	100 (d)	TLC 39 (TCIF Linked Code 39)
0x65	101 (e)	Trioptic (Code 39)
0x66	102 (f)	Bookland (EAN)
0x67	103 (g)	Code 11
0x68	104 (h)	Code 39 Full ASCII
0x69	105 (i)	IATA[Note] (Code 25 used on flight tickets)
0x6A	106 (j)	Industrial 25 (Discrete 25)
0x6B	107 (k)	PDF417
0x6C	108 (l)	MicroPDF417
0x6D	109 (m)	Data Matrix
0x6E	110 (n)	Maxicode
0x6F	111 (o)	QR Code
0x70	112 (p)	US Postnet
0x71	113 (q)	US Planet
0x72	114 (r)	UK Postal
0x73	115 (s)	Japan Postal
0x74	116 (t)	Australian Postal
0x75	117 (u)	Dutch Postal
0x76	118 (v)	Composite Code
0x77	119 (w)	Macro PDF
0x78	120 (x)	Coupon Code
0x79	121 (y)	Chinese 25
0x7A	122 (z)	Aztec
0x7B	123 ({)	MicroQR
0x7C	124 (|)	USPS 4CB / One Code / Intelligent Mail
0x7D	125 (})	UPU FICS Postal

0x7E	126 (~)	Macro MicroPDF417

See Also DataOuputSettings, GetDecodeData

Note: IATA stands for International Air Transport Association, a barcode type used on flight tickets.

ReadBarcodeData

Purpose Reads barcode data by tapping GUI button (without intercepting *WM_DECODEDATA*).

Syntax
```
INT ReadBarcodeData (INT *codeType,
                     CHAR *buf,
                     INT bufSize,
                     INT timeout);
```

Parameters

codeType

[out] Pointer to a variable that stores barcode type.

buf

[out] Pointer to a buffer where barcode data is stored.

bufSize

[in] Buffer size, in characters.

timeout

[in][out] A value that specifies the elapsed time before a decode session ends.

1~9	3* (second)

Example
```
int b1 = 0, codeType = 0;
char outBuf[200];
memset(outBuf, 0, sizeof(outBuf));
b1 = ReadBarcodeData(&codeType, outBuf, sizeof(outBuf), 3);
```

Return Value If successful, it returns the number of bytes obtained (excluding terminating null character).

Otherwise, it returns 1 or a negative value to indicate a specific error:

0	E_READER_UNKNOWN_ERROR
-101	E_READER_NOT_INIT

Remarks This function collects data without intercepting *WM_DECODEDATA.* For example, if users want to trigger scanning by tapping a GUI button, all they need to do is to invoke this function in the clicking event handler function.

See Also DataOutputSettings, GetOutputRecord

3.2.4. RFID DATA

GetDecodeRfidData(), **GetDecodeRfidUID()** and **GetDecodeTagType()** are the funcations to get raw data from an RFID tag. Some RFID tags can be read and written on a page-by-page basis. Use **ReadRfidData()** and **WriteRfidData()** to read or write data to/from an RFID tag.

As different RFID tags may have different page capacity, said functions auto-truncate data to meet page capacity.

An RFID tag's default page to start reading data from, or writing data to:

Tag Type	Standard	Default Block/Page
Mifare	ISO 14443A	4
ICODE SLI	ISO 15693	3
LRI512	ISO 15693	0
SRF55VxxP	ISO 15693	3
Tag-it	ISO 15693	0
Others	ISO 15693	0
ICODE	(Phillips)	5

Note: Different tag types may have different start page and page capacity. See the spec. of the RFID tag to read/write for its memory organization.

GetDecodeRfidData

Purpose	Reads RFID data upon pressing a physical trigger.
Syntax	**INT GetDecodeRfidData (BYTE** **buf*, **INT** *bufSize*);
Parameters	*buf* [out] Pointer to a buffer where decoded data is stored. *bufSize* [in] Maximum number of bytes to receive; normally equals to buffer size.

Example	`INT nRlt;`
	`BYTE szData[128] = {'\0'};`
	`nRlt = GetDecodeRfidData(szData, sizeof(szData));`

Return Value	If successful, it returns the number of bytes obtained (excluding terminating null character). Otherwise, it returns a negative value to indicate an error condition:

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

Remarks	This function is called to obtain the just decoded data right before another successful decoding. The decoded data then hooks up Prefix Code and/or Suffix Code if they are applicable.
See Also	GetDecodeRfidUID, GetDecodeTagType, DataOutputSettings, GetOutputRecord, RfidSettings

GetDecodeRfidUID

Purpose	Reads UID upon pressing a physical trigger.
Syntax	**INT GetDecodeRfidUID (BYTE** **buf*, **INT** *bufSize*);
Parameters	*buf* [out] Pointer to a buffer where UID is stored. *bufSize* [in] Maximum number of bytes to receive; normally equals to buffer size.

Example	`INT nRlt;`
	`BYTE szUID[30] = {'\0'};`
	`nRlt = GetDecodeRfidUID(szUID, sizeof(szUID));`

Return Value	If successful, it returns the number of bytes obtained (excluding terminating null character). Otherwise, it returns a negative value to indicate an error condition:

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

Remarks	This function is called to get the just decoded UID right before another successful decoding. Decoded data then hooks up Prefix Code or/and Suffix Code if they are applicable.
See Also	GetDecodeRfidData, GetDecodeTagType, DataOutputSettings, GetOutputRecord, RfidSettings

GetDecodeTagType

Purpose: Collects RFID tag type upon pressing a physical trigger.

Syntax: **INT GetDecodeTagType (VOID);**

Example:

```
INT nType;
nType = GetDecodeTagType();
```

Return Value: If successful, it returns the tag type.

Return Value	Defined Value	RFID Tag / Standard
0	0x00	Unknown
80	0x50	ISO15693
81	0x51	I-CODE SLI
82	0x52	I-CODE SLI-L
85	0x55	I-CODE SLI–S
87	0x57	55V10P
88	0x58	55V02P
89	0x59	55V10S
91	0x5B	Tag-it HF-I Plus
92	0x5C	Tag-it HF-I Pro
93	0x5D	MB89R118
94	0x5E	LRI 2K
160	0xA0	Mifare DESFire
161	0xA1	Mifare UL (Ultralight)
162	0xA2	Mifare Mini
163	0xA3	Mifare Classic
164	0xA4	ISO14443-4A
165	0xA5	Mifare S50 1K
166	0xA7	Mifare S50 4K
170	0xAA	Jewel
176	0xB0	ISO14443 B
240	0xF0	FeliCa

Otherwise, it returns a negative value to indicate an error condition:

-101	E_READER_NOT_INIT

Remarks: This function is called to get the just decoded tag type right before another successful decoding.

See Also: GetDecodeRfidData, GetDecodeRfidUID, DataOutputSettings, GetOutputRecord, RfidSettings

ReadRfidData

Purpose

Reads RFID data by tapping a GUI button (without intercepting *WM_DECODEDATA*).

Syntax

```
INT ReadRfidData (INT type,
                  BYTE *readData,
                  INT bufSize,
                  INT startByte,
                  INT readLen,
                  INT timeout,
                  CHAR *loginKey,
                  INT loginKeyType);
```

Parameters

type

[in] A value that specifies which data to read (UID or encoded data).

1	READ_UID	Reads UID only.
2	READ_DATA	Reads coded data only.

readData

[out] Pointer to a buffer where decoded data is stored.

bufSize

[in] Maximum number of bytes to read; normally equals to buffer size.

startByte

[in] A value that specifies the byte or page to start reading data from.

-1	Means to start reading from the default page (of a RFID tag). Tabulated below is a number of RFID tags' default page to start reading data from:

Tag Type	Standard	Default Start Page
Mifare	(ISO 14443A)	**4**
ICODE SLI	(ISO 15693)	**3**
LRI512	(ISO 15693)	**0**
SRF55VxxP	(ISO 15693)	**3**
Tag-it	(ISO 15693)	**0**
Others	(ISO 15693)	**0**
ICODE	(Phillips)	**5**

readLen

[in] Number of bytes to read. As different tag types may have different numbers of pages allowed to be read, when you pass a value greater than allowed, it auto-adjusts itself to the maximum value allowed on a tag.

timeout

[in][out] A value that specifies the elapsed time before a decode session ends.

1~10	3* (second)

loginKey

[in] Pointer to a variable that stores the login key, a key to access a specific RFID tag such as Mifare Standard 1K/4K or SLE66R35 tags.

- Such key must be a hex string of 12 bytes length.
- This login key applies to all the sectors covered by the range formulated by "nStartByte" and "nReadLen".

LoginKey Example	*Key String*
0	FFFFFFFFFFFF
"F1F2F3F4F5F6"	F1F2F3F4F5F6

loginKeyType

[in] A value that identifies the login key.

0*	Key A
1	Key B

Example

```
INT nRlt, nRead = 0;
BYTE szRead[128] = {'\0'};
nRlt = ReadRfidData(
READ_UID, szRead, sizeof(szRead), -1, 48, 10, "FFFFFFFFFFFF");
```

Return Value

If successful, it returns the number of bytes obtained (excluding terminating null character).

Otherwise, it returns a negative value to indicate an error condition:

-101	E_READER_NOT_INIT
-272	E_TAG_READ_FAILED
-273	E_TAG_VALUE_OVER_RANGE
-274	E_TAG_INVALID_LENGTH
-278	E_TAG_NO_TAG
-280	E_TAG_CANNOT_READ
-281	E_TAG_READ_TIMEOUT

Remarks

This function collects data without intercepting *WM_DECODEDATA*. For example the scanning triggered by tapping a GUI button. All that is need is to invoke this function in the clicking event handler function. The prospective RFID data then can be collected after decoding.

See Also

DataOutputSettings, GetOutputRecord, RfidSettings, WriteRfidData

WriteRfidData

Purpose: Writes data to an RFID tag upon pressing a physical trigger.

Syntax:

```
INT WriteRfidData (BYTE *writeData,
                   INT startByte,
                   INT writeLen,
                   INT *written,
                   INT timeout,
                   INT mode,
                   CHAR *loginKey,
                   INT loginKeyType);
```

Parameters

writeData

[in] Pointer to a buffer that stores the data to write.

startByte

[in] A value that specifies the byte or page to start writing data to.

-1	Means to start writing from the default page. Tabulated below is a number of RFID tags' default page to start writing data to:

Tag Type	Standard	Default Start Page
Mifare	(ISO 14443A)	**4**
ICODE SLI	(ISO 15693)	**3**
LRI512	(ISO 15693)	**0**
SRF55VxxP	(ISO 15693)	**3**
Tag-it	(ISO 15693)	**0**
Others	(ISO 15693)	**0**
ICODE	(Phillips)	**5**

writeLen

[in] Maximum number of bytes to write. As the number of pages allowed to be written may vary from tag type to tag type, when you pass a value greater than allowed, it auto-adapts to the maximum value allowed on a tag.

written

[out] Pointer to a buffer where stores the actual number of bytes written.

timeout

[in][out] A value that specifies the elapsed time before a decode session ends.

1~10	3* (second)

mode

[in] A value that specifies writing mode.

0	Starts writing immediately.
Non-zero	Starts writing on trigger pressing.

loginKey

[in] Pointer to a variable that stores the login key, a key to access an RFID tag such as Mifare Standard 1K/4K and SLE66R35 tags.

- This key must be a hex string of 12 bytes length.
- This login key applies to all the sectors covered by the range formulated by “nStartByte” and “nWriteLen”.

LoginKey Example	*Key String*
0	FFFFFFFFFFFF
“F1F2F3F4F5F6”	F1F2F3F4F5F6

loginKeyType

[in] A value that identifies the key, key A or B.

0 *	Key A
1	Key B

Example

```
INT nRlt, nWritten = 0;
BYTE szWriteData[] = ”Written by CipherLab”;
nRlt = WriteRfidData(
szWriteData, -1, strlen(szWriteData), &nWritten, 10, 0, 0);
nRlt = Beeper(-1, szPath);
```

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate an error condition:

-101	E_READER_NOT_INIT
-273	E_TAG_VALUE_OVER_RANGE
-274	E_TAG_INVALID_LENGTH
-275	E_TAG_GET_DATA_FAILED
-276	E_TAG_WRITE_FAILED
-277	E_TAG_CANNOT_WRITE
-278	E_TAG_NO_TAG
-279	E_TAG_WRITE_TIMEOUT

Remarks

This function defines what to write to an RFID tag and enforces the writting.

See Also

DataOutputSettings, GetOutputRecord, ReadRfidData, RfidSettings

3.3. MANIPULATE STATUS INDICATION

ReaderDLL signals a successful decoding by a sound or vibration according to **NotificationSettings()**. To signal the receipt of the events other than decoding, call **Beeper()**.

3.3.1. NOTIFICATION SETTINGS

NotificationSettings

Purpose: Configures notification settings.

Syntax:

```
INT NotificationSettings (INT rw,
                          INT *goodRead,
                          INT *enableVibrator,
                          INT *vibrationTime,
                          INT *ledDuration);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

goodRead

[in][out] A value that specifies whether to play a .WAV file to notice a successful reading.

0	Mutes
1~9	Plays sound 2*

enableVibrator

[in][out] A value that specifies whether to vibrate for a successful reading.

0*	Disables Vibrator
1	Enables Vibrator

vibrationTime

[in][out] A value that specifies how long to vibrate.

0	No vibration
1~9	2* (second)

ledDuration

[in][out] A value that specifies whether to light green LED for a successful reading.

0*	Disables
Non-zero	Enables and lighting duration (millisecond)

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

See Also: Beeper

3.3.2. BEEPER (SPEAKER)

Beeper

Purpose: Plays a sound via speaker.

Syntax:

INT Beeper (INT *mode*,
BYTE **path*);

Parameters:

mode

[in] A value that specifies the sound to play.

0	Mutes
1~9	Plays sound 1 to 9
-1	User-definable path to the sound to play

path

[in] Pointer to a buffer where the user-defined path is stored.

- This only applies when **nMode** is -1.

Examp **DataOutputSettings** le:

```
INT nRlt;
BYTE szPath[] = {"\window\test.wav"};
nRlt = Beeper(9, NULL);
nRlt = Beeper(-1, szPath);
```

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

4	ERROR_PARAMETER
-241	E_SOUND_INVALID_INDEX
-242	E_SOUND_INVALID_PATH
-243	E_SOUND_PLAY_FAILED

Remarks: This function plays a sound or WAVE file in an asynchronous manner. To stop playing, use Beeper(0, NULL).

See Also: NotificationSettings

3.4. OBTAIN ESSENTIAL INFORMATION

3.4.1. READER DLL VERSION

GetDllVer

Purpose	Obtains current Reader DLL version.
Syntax	**INT GetDllVer (CHAR** **buf*, **INT** *bufSize*);
Parameters	*buf* [out] Pointer to a buffer where the version number is stored. *bufSize* [in] Size of the buffer, in characters.
Example	`CHAR buf[100];` `GetDllVer(buf, 100);`
Return Value	If successful, it returns 0. Otherwise, it returns 1.

3.4.2. DECODER VERSION

GetDecoderVersion

Purpose: Obtains decoder version.

Syntax: **INT GetDecoderVersion (INT** *target*, **CHAR** **buf*, **INT** *bufSize*);

Parameters:

target

[in] A value that specifies which reader to get decoder version for.

7	DC_READER_BC	Reads barcode reader's decoder version
32	DC_READER_RFID	Reads RFID reader's decoder version.

buf

[out] Pointer to a buffer where version info is stored.

bufSize

[in] Maximum number of bytes to receive; normally equals to buffer size.

Example:

```
CHAR buf[64] = {'\0'};
INT nRlt = 0;
nRlt = GetDecoderVersion(DC_READER_BC, buf, sizeof(buf));
nRlt = GetDecoderVersion(DC_READER_RFID, buf, sizeof(buf));
```

Return Value: If successful, it returns 0.
Otherwise, it returns 1 or a negative value to indicate a specific error:

1	E_READER_UKNOWN_ERROR
-101	E_READER_NOT_INIT

3.5. CONFIGURE SCAN ENGINE — 1D CCD SCAN ENGINE

UserPreferences_1D_SM1() configures the 1D CCD barcode reader while others adjust symbology settings.

3.5.1. PREFERENCES

UserPreferences_1D_SM1	**CCD**

Purpose: Configures 1D CCD barcode reader.

Syntax:

```
int UserPreferences_1D_SM1 (int rw,
                            int *laserOnTime,
                            int *timeoutBetweenSameBarcode,
                            int *redundancyLevel,
                            int *triggerMode);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Sets settings

laserOnTime

[in][out] A value that specifies the maximum time to decode a printed barcode during a scan act.

5~99	30* (0.1 second)

timeoutBetweenSameBarcode

[in][out] A value that specifies the minimum time for the scan engine to resurrect the ability to read the same barcode for the 2nd time. This helps escape accidently scanning the same barcode twice.

- This parameter applies to Continuous mode.

0~99	10* (0.1 second)

redundancyLevel

[in][out] A value that specifies decode redundancy. Higher redundancy levels should be selected for deteriorated barcode quality.

1*	The following barcodes must be successfully read twice before being decoded:

Barcode Types	Code Length
Codabar	All
MSI	4 characters or less
Industrial 25 (Discrete 25)	8 characters or less
Interleaved 25	8 characters or less

2	All barcodes must be successfully read twice before being decoded.
3	All barcodes must be successfully read twice before being decoded except for the following that must be read three times:

	<table><tr><th>Barcode Types “Excluded”</th><th>Code Length</th></tr><tr><td>MSI</td><td>4 characters or less</td></tr><tr><td>Industrial 25 (Discrete 25)</td><td>8 characters or less</td></tr><tr><td>Interleaved 25</td><td>8 characters or less</td></tr></table>
4	All barcodes must be successfully read three times before being decoded.

triggerMode

[in][out] A value that specifies the mode to scan.

4	Continuous mode
8*	Laser mode

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.2. UPC_1D_SM1

Upc_1D_SM1	**CCD**

Purpose: Configures symbology settings — UPC-A and UPC-E.

Syntax:

```
int Upc_1D_SM1 (int rw,
                int *enableUPC_A,
                int *enableUPC_E,
                int *enableUPC_E1,
                int *UPC_EAN_JAN_decodeSupplementals,
                int *UPC_EAN_JAN_supplementalsRedundancy,
                int *transmitUPC_AcheckDigit,
                int *transmitUPC_EcheckDigit,
                int *transmitUPC_E1checkDigit,
                int *preambleUPC_A,
                int *preambleUPC_E,
                int *preambleUPC_E1,
                int *convertUPC_EtoA,
                int *convertUPC_E1toA);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableUPC_A

[in][out] A value that specifies whether to enable UPC-A.

0	Disables
1*	Enables

enableUPC_E

[in][out] A value that specifies whether to enable UPC-E (=UPC-E0).

0	Disables
1*	Enables

enableUPC_E1

[in][out] A value that specifies whether to enable UPC-E1.

0*	Disables
1	Enables

UPC_EAN_JAN_decodeSupplementals

[in][out] A value that specifies whether to enable UPC Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons
2	Decodes with Addons (= Auto-discriminate)

UPC_EAN_JAN_supplementalsRedundancy

[in][out] A value that specifies decode redundancy when “Decode with Addons (= Auto-discriminate)” is applied.

2~30	7* (times of supplementary decoding)

transmitUPC_AcheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-A data.

0	Does not transmit
1*	Transmits

transmitUPC_EcheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E0 data.

0	Does not transmit
1*	Transmits

transmitUPC_E1checkDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E1 data.

0	Does not transmit
1*	Transmits

preambleUPC_A

[in][out] A value that specifies whether and how to transmit UPC-A preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

preambleUPC_E

[in][out] A value that specifies whether and how to transmit UPC-E0 preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

preambleUPC_E1

[in][out] A value that specifies whether and to transmit UPC-E1 preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

convertUPC_EtoA

[in][out] A value that specifies whether to convert UPC-E0 to UPC-A.

0*	Does not convert
1	Converts

convertUPC_E1toA

[in][out] A value that specifies whether to convert UPC-E1 to UPC-A.

0*	Does not convert
1	Converts

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.3. EANJAN_1D_SM1

EanJan_1D_SM1	**CCD**

Purpose: Configures symbology settings — EAN/JAN-8 and EAN/JAN-13.

Syntax:

```
int EanJan_1D_SM1 (int rw,
                   int *enableEAN8_JAN8,
                   int *enableEAN13_JAN13,
                   int *enableBooklandEAN,
                   int *UPC_EAN_JAN_decodeSupplementals,
                   int *UPC_EAN_JAN_supplementalsRedundancy,
                   int *EAN8_JAN8_extend);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableEAN8_JAN8

[in][out] A value that specifies whether to enable EAN/JAN-8.

0	Disables
1*	Enables

enableEAN13_JAN13

[in][out] A value that specifies whether to enable EAN/JAN-13.

0	Disables
1*	Enables

enableBooklandEAN

[in][out] A value that specifies whether to enable Bookland EAN.

- EAN/JAN-13 must be enabled first!

0	Disables
1*	Enables

UPC_EAN_JAN_decodeSupplementals

[in][out] A value that specifies whether to enable EAN/JAN Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons
2	Decodes with Addons (= Auto-discriminate)

UPC_EAN_JAN_supplementalsRedundancy

[in][out] A value that specifies decode redundancy when "Decode with Addons (= Auto-discriminate)" is applied.

2~30	10* (times of supplementary decoding)

EAN8_JAN8_extend

[in][out] A value that specifies whether to convert EAN/JAN-8 to EAN/JAN-13.

0*	Does not convert
1	Converts

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.4. CODE39_1D_SM1

Code39_1D_SM1	**CCD**

Purpose: Configures symbology settings — Code 39.

Syntax:

```
int Code39_1D_SM1 (int rw,
                   int *enable,
                   int *code39ToCode32,
                   int *code32Prefix,
                   int *checkDigitVerification,
                   int *transmitCheckDigit,
                   int *fullASCII,
                   int *length1,
                   int *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Code 39.

0	Disables
1*	Enables

code39ToCode32

[in][out] A value that specifies whether to convert Code 39 to Code 32 (= Italian Pharmacode).

0*	Does not convert
1	Converts

code32Prefix

[in][out] A value that specifies whether to transmit prefix for Code 32 data.

0*	Does not transmit
1	Transmits

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Does not verify
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

- Check digit verification must be enabled first.

0*	Does not transmit
1	Transmits

fullASCII

[in][out] A value that specifies whether to support Code 39 Full ASCII.

0*	No (Standard Code 39)
1	Yes (Code 39 Full ASCII)

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If “length2” is greater than “length1”, they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.5. CODE93_1D_SM1

Code93_1D_SM1	**CCD**

Purpose: Configures symbology settings — Code 93.

Syntax:

```
int Code93_1D_SM1 (int rw,
                   int *enable,
                   int *length1,
                   int *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Code 93.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.6. INTERLEAVED2OF5_1D_SM1

Interleaved2Of5_1D_SM1	**CCD**

Purpose: Configures symbology settings — Interleaved 25.

Syntax:

```
int Interleaved2Of5_1D_SM1 (int rw,
                            int *enable,
                            int *length1,
                            int *length2,
                            int *checkDigitVerification,
                            int *transmitCheckDigit);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Interleaved 25.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Does not verify
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.7. INDUSTRIAL2OF5_1D_SM1

Industrial2Of5_1D_SM1	**CCD**

Purpose

Configures symbology settings — Industrial 25 (= Discrete 25).

Syntax

int Industrial2Of5_1D_SM1 (int *rw*,
int **enable*,
int **length1*,
int **length2*);

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Industrial 25 (= Discrete 25).

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.8. CODABAR_1D_SM1

Codabar_1D_SM1	**CCD**

Purpose: Configures symbology settings — Codabar.

Syntax:

```
int Codabar_1D_SM1 (int rw,
                    int *enable,
                    int *length1,
                    int *length2,
                    int *clsiEditing,
                    int *notisEditing);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Codabar.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

clsiEditing

[in][out] A value that specifies whether to edit CLSI.

0*	Does not edit
1	Edits

- When applied, the CLSI editing strips the start/stop characters and inserts a space after the first, fifth, and tenth characters of a 14-character Codabar barcode.
- The 14-character barcode length does not include start/stop characters.

notisEditing

[in][out] A value that specifies whether to edit NOTIS.

0*	Does not edit
1	Edits

- NOTIS Editing strips the start/stop characters. It is equally efficacious to "Disable Transmit Start/Stop Characters".

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.9. MSI_1D_SM1

Msi_1D_SM1	**CCD**

Purpose: Configures symbology settings — MSI.

Syntax:

```
int Msi_1D_SM1 (int rw,
                int *enable,
                int *length1,
                int *length2,
                int *checkDigitVerification,
                int *transmitCheckDigit,
                int *checkDigitAlgorithm);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable MSI.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies how to verify check digit.

0*	Verifies one check digit
1	Verifies two check digits

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

checkDigitAlgorithm

[in][out] A value that specifies which algorithm to apply.

0	Modulo 10 / Modulo 11
1*	Double Modulo 10

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.10. GS1_DATABAR_1D_SM1

GS1_DataBar_1D_SM1	**CCD**

Purpose: Configures symbology settings — GS1 DataBar (= RSS family).

Syntax:

```
int GS1_DataBar_1D_SM1 (int rw,
                        int *enableDataBarOmnidirectional,
                        int *enableGS1_DataBarLimited,
                        int *enableGS1_DataBarExpanded);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableDataBarOmnidirectional

[in][out] A value that specifies whether to enable GS1 DataBar Omnidirectional (= RSS-14).

0	Disables
1*	Enables

enableGS1_DataBarLimited

[in][out] A value that specifies whether to enable GS1 DataBar Limited (= RSS Limited).

0	Disables
1*	Enables

enableGS1_DataBarExpanded

[in][out] A value that specifies whether to enable GS1 DataBar Expanded (= RSS Expanded).

0	Disables
1*	Enables

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.5.11. CODE128_1D_SM1

Code128_1D_SM1	**CCD**

Purpose: Configures symbology settings — Code 128.

Syntax:

```
int Code128_1D_SM1 (int rw,
                    int *enableCode128,
                    int *enableGS1_128,
                    int *enableISBT128);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableCode128

[in][out] A value that specifies whether to enable Code 128.

0	Disables
1*	Enables

enableGS1_128

[in][out] A value that specifies whether to enable GS1-128.

0	Disables
1*	Enables

enableISBT128

[in][out] A value that specifies whether to enable ISBT 128.

0	Disables
1*	Enables

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6. CONFIGURE SCAN ENGINE — 1D LASER SCAN ENGINE

UserPreferences_1D_SE955() and **MiscellaneousOption_1D_SE955()** are the functions to configure 1D (laser) barcode reader while others adjust symbology settings.

3.6.1. PREFERENCES

UserPreferences_1D_SE955	**Laser**

Purpose: Configures 1D (laser) barcode reader.

Syntax:

```
INT UserPreferences_1D_SE955 (INT rw,
                              INT *laserOnTime,
                              INT *scanAngle,
                              INT *timeoutBetweenSameBarcode,
                              INT *redundancyLevel,
                              INT *bidirectionalRedundancy,
                              INT *aimDuration,
                              INT *triggerMode);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Sets settings

laserOnTime

[in][out] A value that specifies the maximum time to decode a printed barcode during a scan act.

5~99	30* (0.1 second)

scanAngle

[in][out] A value that specifies scan angle.

0x05	Narrow Angle (35°)
0x06*	Wide Angle (47°)

timeoutBetweenSameBarcode

[in][out] A value that specifies the minimum time for the scan engine to resurrect the ability to read the same barcode for the 2nd time. This helps escape accidently scanning the same barcode twice.

- This parameter applies to Continuous mode.

0~99	10* (0.1 second)

redundancyLevel

[in][out] A value that specifies decode redundancy. Higher redundancy levels should be selected for deteriorated barcode quality.

1*	The following barcodes must be successfully read twice before being decoded:

Barcode Types	Code Length
Codabar	All

	MSI: 4 characters or less Industrial 25 (Discrete 25): 8 characters or less Interleaved 25: 8 characters or less
2	All barcodes must be successfully read twice before being decoded.
3	All barcodes must be successfully read twice before being decoded except for the following that must be read three times: Barcode Types "Excluded": Code Length MSI: 4 characters or less Industrial 25 (Discrete 25): 8 characters or less Interleaved 25: 8 characters or less
4	All barcodes must be successfully read three times before being decoded.

bidirectionalRedundancy

[in][out] A value that specifies whether to enable bidirectional redundancy. When enabled, barcodes must be scanned in both forward and reverse directions in order to be decoded.

- This is valid only when redundancy level is enabled for the specific barcode. For instance, when *redundancyLevel* is set at Level 1, only the barcodes that have redundancy enabled will require both forward and reverse direction scanning. When set at Level 2, all barcodes will require both forward and reverse direction scanning.

0*	Disable
1	Enable

aimDuration Reserved

triggerMode

[in][out] A value that specifies the mode to scan.

4	Continuous mode
8*	Laser mode

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

MiscellaneousOption_1D_SE955	**Laser**

Purpose: Configures more reader settings.

Syntax:

```
INT MiscellaneousOption _1D_SE955 (INT rw,
                                   INT *transmitCodeIdChar,
                                   INT *sendNoReadMessage);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

transmitCodeIdChar

[in][out] A value that specifies whether to transmit Code ID characters.

0*	Transmits none
1	Transmits AIM Code ID Character Being included in the beginning of data, each AIM Code ID contains the three-character string "**]cm**" – ▸] = Flag Character (ASCII 93) ▸ c = Code Character (see below) ▸ m = Modifier Character (see below)

sendNoReadMessage Reserved

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

AIM Code ID — Code Characters

Code Character	Code Type
A	Code 39
C	Code 128
E	UPC/EAN
F	Codabar
G	Code 93
H	Code 11
I	Interleaved 25
M	MSI
S	Industrial 25 (Discrete 25), IATA 2 of 5
X	Code 39 Trioptic, Bookland EAN

AIM Code ID — Modifier Characters

Code Type	Option Value	Option

Code 39	0	No check character or Full ASCII processing.
	1	Check digit has been verified.
	3	Check digit has been verified and stripped.
	4	Full ASCII conversion has been performed.
	5	Result of option values 1 and 4.
	7	Result of option values 3 and 4.
Code 128	0	Standard data packet. No Function Code 1“FNC1” in the first character position.
	1	Function Code 1“FNC1” in the first character position.
	2	Function Code 1“FNC1” in the second character position.
Interleaved 25	0	No check digit processing.
	1	Check digit has been verified.
	3	Check digit has been verified and stripped.
Codabar	0	No check digit processing.
Code 93	0	Always transmit 0.
MSI	0	Modulo 10 check digit verified and transmitted.
	1	Modulo 10 check digit verified but not transmitted.
Industrial 25 (Discrete 25)	0	Always transmits 0.
UPC/EAN	0	Standard data packet in full EAN country code format, which is 13 digits for UPC-A and UPC-E (not including supplemental data).
	1	Two-digit supplemental data only.
	2	Five-digit supplemental data only.
	4	EAN-8 data packet.
	A UPC-A with Addon 2 barcode, 012345678905-10, is transmitted to the host as a 21-character string, **]E0**0012345678905**]E1**10.	
Bookland EAN	0	Always transmits 0.
Trioptic Code 39	0	Always transmits 0.

3.6.2. UPC_1D_SE955

Upc_1D_SE955	**Laser**

Purpose: Configures symbology settings — UPC-A and UPC-E.

Syntax:

```
INT Upc_1D_SE955 (INT rw,
                  INT *enableUpcA,
                  INT *enableUpcE,
                  INT *enableUpcE1,
                  INT *enableAddons,
                  INT *addonsRedundancy,
                  INT *transmitUpcACheckDigit,
                  INT *transmitUpcECheckDigit,
                  INT *transmitUpcE1CheckDigit,
                  INT *preambleUpcA,
                  INT *preambleUpcE,
                  INT *preambleUpcE1,
                  INT *convertUpcEtoA,
                  INT *convertUpcE1toA,
                  INT *uccCouponExtendedCode,
                  INT *upcEanSecurityLevel);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableUpcA

[in][out] A value that specifies whether to enable UPC-A.

0	Disables
1*	Enables

enableUpcE

[in][out] A value that specifies whether to enable UPC-E (=UPC-E0).

0	Disables
1*	Enables

enableUpcE1

[in][out] A value that specifies whether to enable UPC-E1.

0*	Disables
1	Enables

enableAddons

[in][out] A value that specifies whether to enable UPC Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons

2	Decodes with Addons (= Auto-discriminate)

addonsRedundancy

[in][out] A value that specifies decode redundancy when “Decode with Addons (= Auto-discriminate)” is applied.

2~30	7* (times of supplementary decoding)

transmitUpcACheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-A data.

0	Does not transmit
1*	Transmits

transmitUpcECheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E0 data.

0	Does not transmit
1*	Transmits

transmitUpcE1CheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E1 data.

0	Does not transmit
1*	Transmits

preambleUpcA

[in][out] A value that specifies whether and how to transmit UPC-A preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

preambleUpcE

[in][out] A value that specifies whether and how to transmit UPC-E0 preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

preambleUpcE1

[in][out] A value that specifies whether and to transmit UPC-E1 preamble.

0	Does not transmit
1*	Transmits System Number only
2	Transmits both System Number & Country Code

convertUpcEtoA

[in][out] A value that specifies whether to convert UPC-E0 to UPC-A.

0*	Does not convert
1	Converts

convertUpcE1toA

[in][out] A value that specifies whether to convert UPC-E1 to UPC-A.

0*	Does not convert

1	Converts

uccCouponExtendedCode

[in][out] A value that specifies whether to enable UCC Coupon Code: UPC-A barcodes starting with digit "5" and UPC-A Coupon Codes.

0*	Disables
1	Enables

upcEanSecurityLevel

[in][out] A value that specifies the decode security for UPC/EAN barcodes. Higher security levels should be selected for deteriorated barcode quality. Note increasing security level decreases the scan engine's aggressiveness; choose only the security level needed for the application.

0	Level 0 provides sufficient security to decode most "in-spec" UPC/EAN barcodes.
1	Level 1: As barcode quality deteriorates, certain characters become liable to misdecode (i.e. 1, 2, 7, 8). If misdecodes occur due to poor printed quality, and misdecoded character are limited to the said, select this security level.
2*	Level 2: If misdecodes occur due to poor printed quality, and misdecoded characters are not limited to 1, 2, 7 and 8, select this security level.
3	Level 3: If Security Level 2 should fail to prevent misdecodes, select this security level. Note this level is a final act that tackles out-of-spec barcodes by impairing the decoding ability of scan engine. It is therefore recommended to improve barcode's print quality if this level should be needed.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.3. EANJAN_1D_SE955

EanJan_1D_SE955	**Laser**

Purpose: Configures symbology settings — EAN/JAN-8 and EAN/JAN-13.

Syntax:

INT EanJan_1D_SE955 (INT *rw*,
INT **enableEanJan8*,
INT **enableEanJan13*,
INT **enableBooklandEan*,
INT **enableAddons*,
INT **addonsRedundancy*,
INT **enableEanJan8Extended*,
INT **uccCouponExtendedCode*,
INT **upcEanSecurityLevel*);

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableEanJan8

[in][out] A value that specifies whether to enable EAN/JAN-8.

0	Disables
1*	Enables

enableEanJan13

[in][out] A value that specifies whether to enable EAN/JAN-13.

0	Disables
1*	Enables

enableBooklandEan

[in][out] A value that specifies whether to enable Bookland EAN.

- EAN/JAN-13 must be enabled first!

0	Disables
1*	Enables

enableAddons

[in][out] A value that specifies whether to enable EAN/JAN Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons
2	Decodes with Addons (= Auto-discriminate)

addonsRedundancy

[in][out] A value that specifies decode redundancy when "Decode with Addons (= Auto-discriminate)" is applied.

2~30	10* (times of supplementary decoding)

enableEanJan8Extended

[in][out] A value that specifies whether to convert EAN/JAN-8 to EAN/JAN-13.

0*	Does not convert
1	Converts

uccCouponExtendedCode

[in][out] A value that specifies whether to enable UCC Coupon Code: EAN-13 barcodes starting with digits "99".

0*	Disables
1	Enables

upcEanSecurityLevel

[in][out] A value that specifies the decode security for UPC/EAN barcodes. Higher redundancy levels should be selected for deteriorated barcode quality. Note increasing security level decreases the scan engine's aggressiveness; choose only the security level necessary for the application.

0	Level 0 provides sufficient security to decode most "in-spec" UPC/EAN barcodes.
1	Level 1: As barcode quality deteriorates, certain characters become liable to misdecodes (i.e. 1, 2, 7, 8). If misdecodes occur due to poor printed quality, and the misdecoded characters are limited to said characters, select this security level.
2*	Level 2: If misdecodes occur due to poor printed quality, and the misdecoded characters are not limited to 1, 2, 7 and 8, select this security level.
3	Level 3: If Security Level 2 should fail to prevent misdecodes, select this security level. Note this level is a final act that tackles out-of-spec barcodes by impairing the decoding ability of scan engine. It is therefore recommended to improve barcode's print quality if this level should be needed.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.4. CODE39_1D_SE955

Code39_1D_SE955	**Laser**

Purpose: Configures symbology settings — Code 39.

Syntax:

```
INT Code39_1D_SE955 (INT rw,
                     INT *enable,
                     INT *enableTrioptic,
                     INT *convertToCode32,
                     INT *prefixCode32,
                     INT *checkDigitVerification,
                     INT *transmitCheckDigit,
                     INT *fullASCII,
                     INT *length1,
                     INT *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Code 39.

0	Disables
1*	Enables

enableTrioptic

[in][out] A value that specifies whether to enable Trioptic Code 39.

0	Disables
1*	Enables

convertToCode32

[in][out] A value that specifies whether to convert Code 39 to Code 32 (= Italian Pharmacode).

0*	Does not convert
1	Converts

prefixCode32

[in][out] A value that specifies whether to transmit prefix for Code 32 data.

0*	Does not transmit
1	Transmits

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Does not verify
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

- Check digit verification must be enabled first.

0*	Does not transmit
1	Transmits

fullASCII

[in][out] A value that specifies whether to support Code 39 Full ASCII.

0*	No (Standard Code 39)
1	Yes (Code 39 Full ASCII)

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If “length2” is greater than “length1”, they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.5. CODE93_1D_SE955

Code93_1D_SE955	**Laser**

Purpose

Configures symbology settings — Code 93.

Syntax

INT Code93_1D_SE955 (INT *rw*,
INT **enable*,
INT **length1*,
INT **length2*);

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Code 93.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.6. INTERLEAVED2OF5_1D_SE955

Interleaved2Of5_1D_SE955	**Laser**

Purpose: Configures symbology settings — Interleaved 25.

Syntax:

```
INT Interleaved2Of5_1D_SE955 (INT rw,
                              INT *enable,
                              INT *length1,
                              INT *length2,
                              INT *checkDigitVerification,
                              INT *transmitCheckDigit,
                              INT *convertToEAN13);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Interleaved 25.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Does not verify
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

convertToEAN13

[in][out] A value that specifies whether to convert Interleaved 25 to EAN-13.

- Check digit verification must be disabled first! This only works when the barcode has a leading zero and a valid EAN-13 check digit.

0*	Does not convert
1	Converts

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.7. INDUSTRIAL2OF5_1D_SE955

Industrial2Of5_1D_SE955 **Laser**

Purpose

Configures symbology settings — Industrial 25 (= Discrete 25).

Syntax

INT Industrial2Of5_1D_SE955 (INT *rw*,
INT **enable*,
INT **length1*,
INT **length2*);

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Industrial 25 (= Discrete 25).

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.8. CHINESE2OF5_1D_SE955

Chinese2Of5_1D_SE955	**Laser**

Purpose: Configures symbology settings — Chinese 25.

Syntax:

INT Chinese2Of5_1D_SE955 (INT *rw*,
INT **enable*);*

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Chinese 25.

0	Disables
1*	Enables

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.9. CODABAR_1D_SE955

Codabar_1D_SE955	**Laser**

Purpose: Configures symbology settings — Codabar.

Syntax:

```
INT Codabar_1D_SE955 (INT rw,
                      INT *enable,
                      INT *length1,
                      INT *length2,
                      INT *enableCLSI_Editing,
                      INT *enableNOTIS_Editing);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Codabar.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

enableCLSI_Editing

[in][out] A value that specifies whether to edit CLSI.

0*	Does not edit
1	Edits

- When applied, the CLSI editing strips the start/stop characters and inserts a space after the first, fifth, and tenth characters of a 14-character Codabar barcode.
- The 14-character barcode length does not include start/stop characters.

enableNOTIS_Editing

[in][out] A value that specifies whether to edit NOTIS.

0*	Does not edit
1	Edits

- NOTIS Editing strips the start/stop characters. It is equally efficacious to "Disable Transmit Start/Stop Characters".

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.10. MSI_1D_SE955

Msi_1D_SE955	**Laser**

Purpose: Configures symbology settings — MSI.

Syntax:

```
INT Msi_1D_SE955 (INT rw,
                  INT *enable,
                  INT *length1,
                  INT *length2,
                  INT *checkDigitVerification,
                  INT *transmitCheckDigit,
                  INT *checkDigitAlgorithm);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable MSI.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies how to verify check digit.

0*	Verifies one check digit
1	Verifies two check digits

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

checkDigitAlgorithm

[in][out] A value that specifies which algorithm to apply.

0	Modulo 10 / Modulo 11
1*	Double Modulo 10

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.11. GS1_DATABAR_1D_SE955

GS1_DataBar_1D_SE955	**Laser**

Purpose

Configures symbology settings — GS1 DataBar (= RSS family).

Syntax

INT GS1_DataBar_1D_SE955 (INT *rw*,
INT **enableGS1_DataBarOmnidirectional*,
INT **enableGS1_DataBarLimited*,
INT **enableGS1_DataBarExpanded*,
INT **convertToUpcEan*);

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableGS1_DataBarOmnidirectional

[in][out] A value that specifies whether to enable GS1 DataBar Omnidirectional (= RSS-14).

0	Disables
1*	Enables

enableGS1_DataBarLimited

[in][out] A value that specifies whether to enable GS1 DataBar Limited (= RSS Limited).

0	Disables
1*	Enables

enableGS1_DataBarExpanded

[in][out] A value that specifies whether to enable GS1 DataBar Expanded (= RSS Expanded).

0	Disables
1*	Enables

convertToUpcEan

[in][out] A value that specifies whether to convert RSS to UPC/EAN barcodes.

0*	Does not convert
1	Converts

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.12. CODE128_1D_SE955

Code128_1D_SE955	**Laser**

Purpose

Configures symbology settings — Code 128.

Syntax

```
INT Code128_1D_SE955 (INT rw,
                      INT *enableCode128,
                      INT *enableGS1_128,
                      INT *enableISBT128);
```

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enableCode128

[in][out] A value that specifies whether to enable Code 128.

0	Disables
1*	Enables

enableGS1_128

[in][out] A value that specifies whether to enable GS1-128.

0	Disables
1*	Enables

enableISBT128

[in][out] A value that specifies whether to enable ISBT 128.

0	Disables
1*	Enables

Return Value

If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.6.13. CODE11_1D_SE955

Code11_1D_SE955	**Laser**

Purpose: Configures symbology settings — Code 11.

Syntax:

```
INT Code11_1D_SE955 (INT rw,
                     INT *enable,
                     INT *numberOfCheckDigits,
                     INT *transmitCheckDigit,
                     INT *length1,
                     INT *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	**Reader.ReaderEngineAPI.READ_PARAM**	Gets settings
'w'	**Reader.ReaderEngineAPI.WRITE_PARAM**	Writes settings

enable

[in][out] A value that specifies whether to enable Code 11.

0	Disables
1*	Enables

numberOfCheckDigits

[in][out] A value that specifies whether and how to verify check digit.

0*	None
1	One check digit
2	Two check digits

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value: If successful, it returns 0.

Otherwise, it returns a negative value to indicate a specific error:

-253	E_WRONG_READER_TYPE

3.7. CONFIGURE SCAN ENGINE — 2D SCAN ENGINE

UserPreferences_2D_SE4500 and **MiscellaneousOption_2D_SE4500** are the functions to configure 2D imager while the others adjust symbology settings.

3.7.1. PREFERENCES

UserPreferences_2D_SE4500	**2D**

Purpose: Configures reader settings.

Syntax:

```
INT UserPreferences_2D_SE4500 (INT rw,
                               INT *laserOnTime,
                               INT *timeoutBetweenSameBarcode,
                               INT *triggerMode);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

laserOnTime

[in][out] A value that specifies the maximum time to decode a printed barcode during a scan act.

5~99	30* (0.1 second)

timeoutBetweenSameBarcode Reserved

triggerMode

[in][out] A value that specifies the mode to scan.

0*	Level (default)
7	Presentation mode
9	Auto aim

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

UserPreferences_Extend	**2D**

Purpose

Sets LED illumination level.

Syntax

INT UserPreferences_Extend (INT *rw*,
UserPreferencesParameters **param*);

Parameters

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Sets settings

param

[in] Pointer to the structure UserPreferencesParameters where the information is stored.

Return Value

If successful, it returns 1.

Otherwise, it returns 0.

- Call GetErrorCode() to find the error condition encountered:

4	ERROR_PARAMETER

SymbologySecurityLevel_2D_SE4500 **2D**

Purpose

Configures decode redundancy.

Syntax

```
INT SymbologySecurityLevel_2D_SE4500 (INT rw,
                                      INT *redundancyLevel,
                                      INT *securityLevel,
                                      INT *interCharGapSize);
```

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

redundancyLevel

[in][out] A value that specifies decode redundancy. Higher redundancy levels should be selected for deteriorated barcode quality.

1*	The following barcodes must be successfully read twice before being decoded: Barcode Types / Code Length Codabar / 8 characters or less MSI / 4 characters or less Industrial 25 (Discrete 25) / 8 characters or less Interleaved 25 / 8 characters or less
2	All barcodes must be successfully read twice before being decoded.
3	All barcodes must be successfully read twice before being decoded except for the following that must be read three times: Barcode Types "Excluded" / Code Length Codabar / 8 characters or less MSI / 4 characters or less Industrial 25 (Discrete 25) / 8 characters or less Interleaved 25 / 8 characters or less
4	All barcodes must be successfully read three times before being decoded.

securityLevel

[in][out] A value that specifies decode security level, which is useful to fix some printed quality issues when reading delta barcodes such as Code 128, Code 93, UPC/EAN.

0*	Security Level 0 – The default. It allows the scan engine to operate aggressively enough to decode most "in-spec" barcodes.
1	Security Level 1 – Select this option if misdecodes occur. This level should fix most misdecodes.
2	Security Level 2 – Select this option if Security Level 1 fails to fix

	misdecodes.
3	Security Level 3 – Select this option if Security Level 2 also fails to fix misdecodes. However, selecting this option impairs the decoding ability of the scan engine. If this level of security is necessary, try to improve the barcode quality.

interCharGapSize

[in][out] A value that specifies the intercharacter gap size for Code 39 and Codabar, which is typically quite small. Due to various barcode printing technologies, this gap can grow larger than the maximum size allowed and prevent the scan engine from decoding a barcode. If this problem occurs, set it to "Large Intercharacter Gaps" to tolerate these out-of-specification barcodes.

0x06*	Normal intercharacter gaps
0x0A	Large intercharacter gaps

Return Value

If successful, it returns 0.

Otherwise, it returns 1.

MiscellaneousOption_2D_SE4500	**2D**

Purpose — Configures more reader settings.

Syntax

```
INT MiscellaneousOption_2D_SE4500 (INT rw,
                                   INT *transmitCodeIdChar,
                                   INT *sendNoReadMessage);
```

Parameters — A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

transmitCodeIdChar

[in][out] A value that specifies whether to transmit Code ID characters.

0*	None
1	AIM Code ID Character Being included in the beginning of data, each AIM Code ID contains the three-character string "**]cm**" – ▸] = Flag Character (ASCII 93) ▸ c = Code Character (see below) ▸ m = Modifier Character (see below)

sendNoReadMessage Reserved

Return Value — If successful, it returns 0.

Otherwise, it returns 1.

AIM Code ID — Code Characters

Code Character	Code Type
A	Code 39, Code 39 Full ASCII, Code 32
C	Code 128, Coupon (Code 128 portion)
d	Data Matrix
E	UPC/EAN, Coupon (UPC portion)
e	GS1 DataBar (RSS)
F	Codabar
G	Code 93
H	Code 11
I	Interleaved 25
L	PDF417, Macro PDF417, Micro PDF417
M	MSI
Q	QR Code
S	Industrial 25 (Discrete 25), IATA 2 of 5
U	Maxicode

X	Code 39 Trioptic, Bookland EAN, US Postnet, US Planet, UK Postal, Japan Postal, Australian Postal, Dutch Postal

AIM Code ID — Modifier Characters

Code Type	Option Value	Option
Code 39	0	No check character or Full ASCII processing.
	1	Check digit has been verified.
	3	Check digit has been verified and stripped.
	4	Full ASCII conversion has been performed.
	5	Result of option values 1 and 4.
	7	Result of option values 3 and 4.
Code 128	0	Standard data packet. No Function Code 1“FNC1” in the first character position.
	1	Function Code 1“FNC1” in the first character position.
	2	Function Code 1“FNC1” in the second character position.
Interleaved 25	0	No check digit processing.
	1	Check digit has been verified.
	3	Check digit has been verified and stripped.
Codabar	0	No check digit processing.
Code 93	0	Always transmits 0.
MSI	0	Modulo 10 check digit verified and transmitted.
	1	Modulo 10 check digit verified but not transmitted.
Industrial 25 (Discrete 25)	0	Always transmits 0.
UPC/EAN	0	Standard data packet in full EAN country code format, which is 13 digits for UPC-A and UPC-E (not including supplemental data).
	3	Standard data packet with two-digit or five-digit supplemental data.
	4	EAN-8 data packet.
	A UPC-A with Addon 2 barcode, 012345678905-10, is transmitted to the host as a 18-character string, **]E3**001234567890510.	
Bookland EAN	0	Always transmits 0.
Trioptic Code 39	0	Always transmits 0.
Code 11	0	Single check digit (has been verified.)
	1	Two check digits (has been verified.)
	3	Check digit has been verified but not transmitted.
GS1 DataBar (RSS)	0	Always transmits 0.
	RSS-14 and RSS Limited will be transmitted with an Application Identifier “01”. For example, an RSS-14 barcode, 10012345678902, is transmitted	

	as **]e0**0110012345678902.	
EAN.UCC Composites (RSS, GS1-128, 2D portion of UPC composite)	Native mode transmission	
	0	Standard data packet
	1	Data packet containing the data following an encoded symbol separator character.
	2	Data packet containing the data following an escape mechanism character. The data packet does not support the ECI protocol.
	3	Data packet containing the data following an escape mechanism character. The data packet supports the ECI protocol.
	GS1-128 emulation	
	1	Data packet is a GS1-128 barcode (i.e. data is preceded with "]JC1"). ▸ In GS1-128 emulation mode, RSS is transmitted using Code 128 rules (i.e. "]C1"). ▸ UPC portion of composite is transmitted using UPC rules.
PDF417, Micro PDF417	0	Scan engine is set to conform to protocol defined in 1994 PDF417 symbology specifications. ▸ When this option is transmitted, the receiver cannot reliably determine whether ECIs have been invoked or whether data byte 92_{DEC} has been doubled in transmission.
	1	Scan engine is set to follow the ECI protocol (Extended Channel Interpretation). All data characters 92_{DEC} are doubled.
	2	Scan engine is set for Basic Channel operation (no escape character transmission protocol). Data characters 92_{DEC} are not doubled. ▸ When decoders are set to this mode, unbuffered Macro symbols and symbols requiring the decoder to convey ECI escape sequences cannot be transmitted.
	3	The barcode contains a GS1-128 symbol, and the first codeword is 903-907, 912, 914, 915.
	4	The barcode contains a GS1-128 symbol, and the first codeword is in the range 908-909.
	5	The barcode contains a GS1-128 symbol, and the first codeword is in the range 910-911.
	A PDF417 barcode, ABCD, with no transmission protocol enabled, is transmitted as]L2ABCD.	
Data Matrix	0	ECC 000-140, not supported.
	1	ECC 200.
	2	ECC 200, FNC1 in first or fifth position.
	3	ECC 200, FNC1 in second or sixth position.
	4	ECC 200, ECI protocol implemented.

	5	ECC 200, FNC1 in first or fifth position, ECI protocol implemented.
	6	ECC 200, FNC1 in second or sixth position, ECI protocol implemented.
Maxicode	0	Mode 4 or 5
	1	Mode 2 or 3
	2	Mode 4 or 5, ECI protocol implemented.
	3	Mode 2 or 3, ECI protocol implemented in secondary message.
QR Code	0	Model 1
	1	Model 2, ECI protocol not implemented.
	2	Model 2, ECI protocol implemented.
	3	Model 2, ECI protocol not implemented, FNC1 implied in first position.
	4	Model 2, ECI protocol implemented, FNC1 implied in first position.
	5	Model 2, ECI protocol not implemented, FNC1 implied in second position.
	6	Model 2, ECI protocol implemented, FNC1 implied in second position

3.7.2. UPC_2D_SE4500

Upc_2D_SE4500	**2D**

Purpose: Configures symbology settings — UPC-A and UPC-E.

Syntax:

```
INT Upc_2D_SE4500 (INT rw,
                   INT *enableUpcA,
                   INT *enableUpcE,
                   INT *enableUpcE1,
                   INT *enableAddons,
                   INT *addonsRedundancy,
                   INT *transmitUpcACheckDigit,
                   INT *transmitUpcECheckDigit,
                   INT *transmitUpcE1CheckDigit,
                   INT *preambleUpcA,
                   INT *preambleUpcE,
                   INT *preambleUpcE1,
                   INT *convertUpcEtoA,
                   INT *convertUpcE1toA,
                   INT *uccCouponExtendedCode);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableUpcA

[in][out] A value that specifies whether to enable UPC-A.

0	Disables
1*	Enables

enableUpcE

[in][out] A value that specifies whether to enable UPC-E (=UPC-E0).

0	Disables
1*	Enables

enableUpcE1

[in][out] A value that specifies whether to enable UPC-E1.

0*	Disables
1	Enables

enableAddons

[in][out] A value that specifies whether to enable UPC/EAN/JAN Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons

2	Decodes with Addons (= Auto-discriminate)

addonsRedundancy

[in][out] A value that specifies decode redundancy when “Decode with Addons (= Auto-discriminate)” is applied.

2~30	10* (times of supplementary decoding)

transmitUpcACheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-A data.

0	Does not transmit
1*	Transmits

transmitUpcECheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E0 data.

0	Does not transmit
1*	Transmits

transmitUpcE1CheckDigit

[in][out] A value that specifies whether to transmit check digit for UPC-E1 data.

0	Does not transmit
1*	Transmits

preambleUpcA

[in][out] A value that specifies whether and how to transmit UPC-A preamble.

0	Does not transmit
1*	Transmits System Number
2	Transmits System Number & Country Code

preambleUpcE

[in][out] A value that specifies whether and how to transmit UPC-E0 preamble.

0	Does not transmit
1*	Transmits System Number
2	Transmits System Number & Country Code

preambleUpcE1

[in][out] A value that specifies whether and how to transmit UPC-E1 preamble.

0	Does not transmit
1*	Transmits System Number
2	Transmits System Number & Country Code

convertUpcEtoA

[in][out] A value that specifies whether to convert UPC-E0 to UPC-A.

0*	Does not convert
1	Converts

convertUpcE1toA

[in][out] A value that specifies whether to convert UPC-E1 to UPC-A.

0*	Does not convert

1	Converts

uccCouponExtendedCode

[in][out] A value that specifies whether to enable UCC Coupon Code: UPC-A barcodes starting with digit "5" and UPC-A Coupon Codes.

0*	Disables
1	Enables

Return Value

If successful, it returns 0.

Otherwise, it returns 1.

3.7.3. EANJAN_2D_SE4500

EanJan_2D_SE4500	**2D**

Purpose: Configures symbology settings — EAN/JAN-8 and EAN/JAN-13.

Syntax:

```
INT EanJan_2D_SE4500 (INT rw,
                      INT *enableEanJan8,
                      INT *enableEanJan13,
                      INT *enableBooklandEan,
                      INT *enableAddons,
                      INT *addonsRedundancy,
                      INT *enableEanJan8Extended,
                      INT *uccCouponExtendedCode,
                      INT *booklandIsbnFormat,
                      INT *EnableIssnEan);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableEanJan8

[in][out] A value that specifies whether to enable EAN/JAN-8.

0	Disables
1*	Enables

enableEanJan13

[in][out] A value that specifies whether to enable EAN/JAN-13.

0	Disables
1*	Enables

enableBooklandEan

[in][out] A value that specifies whether to enable Bookland EAN.

- EAN/JAN-13 must be enabled first!

0	Disables
1*	Enables

enableBooklandEan

[in][out] A value that specifies whether to enable Bookland EAN.

- EAN/JAN-13 must be enabled first!

0*	Disables
1	Enables

enableAddons

[in][out] A value that specifies whether to enable EAN/JAN Addon 2 and Addon 5.

0*	Ignores Addons
1	Decodes only with Addons
2	Decodes with Addons (= Auto-discriminate)

addonsRedundancy

[in][out] A value that specifies decode redundancy when “Decode with Addons (= Auto-discriminate)” is applied.

2~30	10* (times of supplementary decoding)

enableEanJan8Extended

[in][out] A value that specifies whether to convert EAN/JAN-8 to EAN/JAN-13.

0*	Does not convert
1	Converts

uccCouponExtendedCode

[in][out] A value that specifies whether to enable UCC Coupon Code: EAN-13 barcodes starting with digits "99".

0*	Disables
1	Enables

booklandIsbnFormat

[in][Out] If you enable Bookland EAN, select one of the following formats for Bookland data.

0*	Bookland ISBN-10
1	Bookland ISBN-13

EnableIssnEan

[in][Out] A value that specifies the whether to enable Issn Ean.

0*	Disables
1	Enables

Return Value

If successful, it returns 0.

Otherwise, it returns 1.

3.7.4. CODE39_2D_SE4500

Code39_2D_SE4500	**2D**

Purpose: Configures symbology settings — Code 39.

Syntax:

```
INT Code39_2D_SE4500 (INT rw,
                      INT *enable,
                      INT *enableTrioptic,
                      INT *convertToCode32,
                      INT *prefixCode32,
                      INT *checkDigitVerification,
                      INT *transmitCheckDigit,
                      INT *fullASCII,
                      INT *length1,
                      INT *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Code 39.

0	Disables
1*	Enables

enableTrioptic

[in][out] A value that specifies whether to enable Trioptic Code 39.

0*	Disables
1	Enables

convertToCode32

[in][out] A value that specifies whether to convert Code 39 to Code 32 (= Italian Pharmacode).

0*	Does not convert
1	Converts

prefixCode32

[in][out] A value that specifies whether to transmit prefix for Code 32 data.

0*	Does not transmit
1	Transmits

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Not verifies
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

- Check digit verification must be enabled first.

0*	Does not transmit
1	Transmits

fullASCII

[in][out] A value that specifies whether to support Code 39 Full ASCII.

0*	No (Standard Code 39)
1	Yes (Code 39 Full ASCII)

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If “length2” is greater than “length1”, they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.

Otherwise, it returns 1.

3.7.5. CODE93_2D_SE4500

Code93_2D_SE4500	**2D**

Purpose: Configures symbology settings — Code 93.

Syntax:

```
INT Code93_2D_SE4500 (INT rw,
                      INT *enable,
                      INT *length1,
                      INT *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Code 93.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification. (see above)

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

3.7.6. INTERLEAVED2OF5_2D_SE4500

Interleaved2Of5_2D_SE4500	**2D**

Purpose: Configures symbology settings — Interleaved 25.

Syntax:

```
INT Interleaved2Of5_2D_SE4500 (INT rw,
                               INT *enable,
                               INT *length1,
                               INT *length2,
                               INT *checkDigitVerification,
                               INT *transmitCheckDigit,
                               INT *convertToEAN13);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Interleaved 25.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Disables
1	USS Check Digit
2	OPCC Check Digit

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

convertToEAN13

[in][out] A value that specifies whether to convert Interleaved 25 to EAN-13.

- Check digit verification must be disabled first! This only works when the

barcode has a leading zero and a valid EAN-13 check digit.

0*	Does not convert
1	Converts

Return Value

If successful, it returns 0.
Otherwise, it returns 1.

3.7.7. INDUSTRIAL2OF5_2D_SE4500

Industrial2Of5_2D_SE4500 **2D**

Purpose

Configures symbology settings — Industrial 25 (= Discrete 25).

Syntax

INT Industrial2Of5_2D_SE4500 (INT *rw*,
INT **enable*,
INT **length1*,
INT **length2*);

Parameters

A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Industrial 25 (= Discrete 25).

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value

If successful, it returns 0.
Otherwise, it returns 1.

3.7.8. MATRIX2OF5_2D_SE4500

Matrix2Of5_2D_SE4500	**2D**

Purpose: Configures symbology settings — Matrix 25.

Syntax:

```
INT Matrix2Of5_2D_SE4500 (INT rw,
                          INT *enable,
                          INT *length1,
                          INT *length2,
                          INT *redundancy,
                          INT *checkDigitVerification,
                          INT *transmitCheckDigit);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Matrix 25.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

redundancy

[in][out] A value that specifies whether to enable decode redundancy.

0*	Disables
1	Enables

checkDigitVerification

[in][out] A value that specifies whether to verify check digit.

0*	Not verifies
1	Verifies

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

Return Value
If successful, it returns 0.
Otherwise, it returns 1.

3.7.9. CHINESE2OF5_2D_SE4500

Chinese2Of5_2D_SE4500 **2D**

Purpose
Configures symbology settings — Chinese 25.

Syntax
INT Chinese2Of5_2D_SE4500 (INT *rw*,
INT **enable*);*

Parameters
A default value comes with an asterisk "*".
rw
[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable
[in][out] A value that specifies whether to enable Chinese 25.

0	Disables
1*	Enables

Return Value
If successful, it returns 0.
Otherwise, it returns 1.

3.7.10. CODABAR_2D_SE4500

Codabar_2D_SE4500	**2D**

Purpose: Configures symbology settings — Codabar.

Syntax:

```
INT Codabar_2D_SE4500 (INT rw,
                       INT *enable,
                       INT *length1,
                       INT *length2,
                       INT *enableCLSI_Editing,
                       INT *enableNOTIS_Editing);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Codabar.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

enableCLSI_Editing

[in][out] A value that specifies whether to edit CLSI.

0*	Disables
1	Enables

- When applied, the CLSI editing strips the start/stop characters and inserts a space after the first, fifth, and tenth characters of a 14-character Codabar barcode.
- The 14-character barcode length does not include start/stop characters.

enableNOTIS_Editing

[in][out] A value that specifies whether to edit NOTIS.

0*	Disables
1	Enables

- NOTIS Editing strips the start/stop characters. It is equally efficacious to "Disable Transmit Start/Stop Characters".

Return Value	If successful, it returns 0. Otherwise, it returns 1.

3.7.11. MSI_2D_SE4500

Msi_2D_SE4500	**2D**

Purpose: Configures symbology settings — MSI.

Syntax:

```
INT Msi_2D_SE4500 (INT rw,
                   INT *enable,
                   INT *length1,
                   INT *length2,
                   INT *checkDigitVerification,
                   INT *transmitCheckDigit,
                   INT *checkDigitAlgorithm);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable MSI.

0	Disables
1*	Enables

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

checkDigitVerification

[in][out] A value that specifies how to verify check digit.

0*	Verifies one check digit
1	Verifies two check digits

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

checkDigitAlgorithm

[in][out] A value that specifies which algorithm to apply.

0	Modulo 10 / Modulo 11
1*	Double Modulo 10

Return Value

If successful, it returns 0.
Otherwise, it returns 1.

3.7.12. GS1_DATABAR_2D_SE4500

GS1_DataBar_2D_SE4500 **2D**

Purpose: Configures symbology settings — GS1 DataBar (= RSS family).

Syntax:

```
INT GS1_DataBar_2D_SE4500 (INT rw,
                           INT *enableGS1_DataBarOmnidirectional,
                           INT *enableGS1_DataBarLimited,
                           INT *enableGS1_DataBarExpanded,
                           INT *convertToUpcEan);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableGS1_DataBarOmnidirectional

[in][out] A value that specifies whether to enable GS1 DataBar Omnidirectional (= RSS-14).

0	Disables
1*	Enables

enableGS1_DataBarLimited

[in][out] A value that specifies whether to enable GS1 DataBar Limited (= RSS Limited).

0	Disables
1*	Enables

enableGS1_DataBarExpanded

[in][out] A value that specifies whether to enable GS1 DataBar Expanded (= RSS Expanded).

0	Disables
1*	Enables

convertToUpcEan

[in][out] A value that specifies whether to convert RSS to UPC/EAN barcodes.

0*	Does not convert
1	Converts

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

3.7.13. CODE128_2D_SE4500

Code128_2D_SE4500	**2D**

Purpose: Configures symbology settings — Code 128.

Syntax:

```
INT Code128_2D_SE4500 (INT rw,
                       INT *enableCode128,
                       INT *enableGS1_128,
                       INT *enableISBT128,
                       INT *enableIsbtConcatenation,
                       INT *isbtConcatenationRedundancy);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableCode128

[in][out] A value that specifies whether to enable Code 128.

0	Disables
1*	Enables

enableGS1_128

[in][out] A value that specifies whether to enable GS1-128.

0	Disables
1*	Enables

enableISBT128

[in][out] A value that specifies whether to enable ISBT 128.

0	Disables
1*	Enables

enableIsbtConcatenation

[in][out] A value that specifies whether to decode and concatenate pairs of ISBT barcodes.

0*	Disables concatenation. ▸ Pairs of ISBT barcodes are not concatenated.
1	Enables concatenation. ▸ However unless there are two ISBT barcodes, decoding and concatenation won't be performed. Single ISBT barcode won't be decoded.
2	Enables auto-discrimination. ▸ Pairs of ISBT barcodes are decoded and concatenated immediately. If there is only a single ISBT barcode, the scanner must decode 10 times before transmitting its data to confirm that there is no additional ISBT barcode.

isbtConcatenationRedundancy

	[in][out] A value that specifies concatenation redundancy (2~20 times) when auto-discriminate of ISBT concatenation is enabled. It is set to 10 times by default.
Return Value	If succesful, it returns 0. Otherwise, it returns 1.

3.7.14. CODE11_2D_SE4500

Code11_2D_SE4500	**2D**

Purpose: Configures symbology settings — Code 11.

Syntax:

```
INT Code11_2D_SE4500 (INT rw,
                      INT *enable,
                      INT *numberOfCheckDigits,
                      INT *transmitCheckDigit,
                      INT *length1,
                      INT *length2);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable Code 11.

0	Disables
1*	Enables

numberOfCheckDigits

[in][out] A value that specifies whether to verify check digit.

0*	None
1	One check digit
2	Two check digits

transmitCheckDigit

[in][out] A value that specifies whether to transmit check digit.

0*	Does not transmit
1	Transmits

length1

[in][out] A value that specifies length qualification.

0~55	4*

length2

[in][out] A value that specifies length qualification.

0~55	55*

- If "length2" is greater than "length1", they are applied as Max/Min length; otherwise a fixed length applies.

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

3.7.15. POSTALCODE_2D_4500

PostalCode_2D_SE4500	**2D**

Purpose — Configures symbology settings — Postal Codes.

Syntax

```
INT PostalCode_2D_SE4500 (INT rw,
                          INT *enableUSPostnet,
                          INT *enableUSPlanet,
                          INT *enableUKPostal,
                          INT *transmitUKCheckDigit,
                          INT *enableJapanPostal,
                          INT *enableAustralianPostal,
                          INT *enableDutchPostal,
                          INT *transmitUSCheckDigit);
```

Parameters — A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableUSPostnet

[in][out] A value that specifies whether to enable US Postnet.

0	Disables
1*	Enables

enableUSPlanet

[in][out] A value that specifies whether to enable US Planet.

0	Disables
1*	Enables

enableUKPostal

[in][out] A value that specifies whether to enable UK Postal.

0	Disables
1*	Enables

transmitUKCheckDigit

[in][out] A value that specifies whether to transmit check digit for UK Postal data.

0	Does not transmit
1*	Transmits

enableJapanPostal

[in][out] A value that specifies whether to enable Japan Postal.

0	Disables
1*	Enables

enableAustralianPostal

[in][out] A value that specifies whether to enable Australian Postal.

0	Disables
1*	Enables

enableDutchPostal

[in][out] A value that specifies whether to enable Dutch Postal.

0	Disables
1*	Enables

transmitUSCheckDigit

[in][out] A value that specifies whether to transmit check digit for US Postal data.

0	Does not transmit
1*	Transmits

Return Value

If successful, it returns 0.
Otherwise, it returns 1.

3.7.16. COMPOSITE_2D_SE4500

Composite_2D_SE4500	**2D**

Purpose: Configures symbology settings — Composite barcodes.

Syntax:

```
INT Composite_2D_SE4500 (INT rw,
                         INT *enableCC_C,
                         INT *enableCC_AB,
                         INT *enableTLC39,
                         INT *enableUpcMode,
                         INT *beepMode,
                         INT *enableEmulationMode);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enableCC_C

[in][out] A value that specifies whether to enable Composite CC-C.

0	Disables
1*	Enables

enableCC_AB

[in][out] A value that specifies whether to enable Composite CC-A/B.

0*	Disables
1	Enables

enableTLC39

[in][out] A value that specifies whether to enable Composite TLC 39 (= TCIF Linked Code 39).

0*	Disables
1	Enables

enableUpcMode

[in][out] A value that specifies whether to enable UPC Composite Mode.

- UPC barcodes can be "linked" with a 2D barcode during transmission as if they were one barcode.

0	Never links UPC (= discards 2D portion if there is any)
1*	Always links UPC (= discards whole barcode if there is no 2D)
2	Auto-discriminates UPC Composites

beepMode Reserved

enableEmulationMode

[in][out] A value that specifies whether to enable GS1-128 Emulation Mode for UCC/EAN Composite Codes.

0*	Disables
1	Enables

Return Value
If successful, it returns 0.
Otherwise, it returns 1.

3.7.17. INVERSE1D_2D_SE4500

Inverse1D_2D_SE4500	**2D**

Purpose
Configures symbology settings — 1D Inverse.

Syntax
INT Inverse1D_2D_SE4500 (INT *rw*, **INT ***_enable1DInverse_**)**

Parameters
rw
[in] A value that specifies the operation.
enable1DInverse
[in][out] A value that specifies whether to enable 1D Inverse.

0*	Regular Only - the decoder decodes regular 1D bar codes only
1	Inverse Only - the decoder decodes inverse 1D bar codes only
2	Inverse Autodetect - the decoder decodes both regular and inverse 1D bar codes

Return Value
If successful, it returns 0.
Otherwise, it returns 1.

3.7.18. KOREAN3OF5_2D_SE4500

Korean3Of5_2D_SE4500	**2D**

Purpose
Configures symbology settings — Korean 3 Of 5.

Syntax
INT Korean3Of5_2D_SE4500 (INT *rw*,
INT ***_enable_)**

Parameters
rw
[in] A value that specifies the operation.
enable
[in][out] A value that specifies whether to enable Korean 3 Of 5.

0*	Disables
1	Enables

Return Value
If successful, it returns 0.
Otherwise, it returns 1.

3.7.19. SYMBOLOGIES_2D_SE4500

Symbologies_2D_SE4500	**2D**

Purpose: Configures 2D symbology settings.

Syntax:

```
INT Symbologies_2D_SE4500 (INT rw,
                           INT *enablePDF417,
                           INT *enableMicroPDF417,
                           INT *enableCode128Emulation,
                           INT *enableDataMatrix,
                           INT *enableDataMatrixInverse,
                           INT *decodeMirrorImage,
                           INT *enableMaxicode,
                           INT *enableQRCode,
                           INT *enableQRCodeInverse,
                           INT *enableMicroQR,
                           INT *enableAztec,
                           INT *enableAztecInverse);
```

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enablePDF417

[in][out] A value that specifies whether to enable PDF417.

0	Disables
1*	Enables

enableMicroPDF417

[in][out] A value that specifies whether to enable MicroPDF417.

0*	Disables
1	Enables

enableCode128Emulation

[in][out] A value that specifies whether to enable Code 128 Emulation for certain MicroPDF417 barcodes.

0*	Disables
1	Enables

- AIM Code Identifier must be enabled first!
- When applied, the MicroPDF417 barcodes are transmitted as if they were encoded in Code 128 barcodes with one of these prefixes:

The first codeword of MicroPDF417 is 903-907, 912, 914, 915:
The original Code ID "]L3" will be changed to "]C1".

The first codeword of MicroPDF417 is 908 or 909:
The original Code ID "]L4" will be changed to "]C2".
The first codeword of MicroPDF417 is 910 or 911:
The original Code ID "]L5" will be changed to "]C0".

enableDataMatrix

[in][out] A value that specifies whether to enable Data Matrix.

0	Disables
1*	Enables

enableDataMatrixInverse

[in][out] A value that specifies whether to enable Data Matrix Inverse.

0*	Regular Only: decode regular Data Matrix barcodes only
1	Inverse Only: decode inverse Data Matrix barcodes only
2	Inverse Autodetect: decode both regular and inverse Data Matrix barcodes

decodeMirrorImage

[in][out] A value that specifies whether to enable Data Matrix Inverse.

0*	Never
1	Always
2	Auto

enableMaxicode

[in][out] A value that specifies whether to enable Maxicode.

0	Disables
1*	Enables

enableQRCode

[in][out] A value that specifies whether to enable QR Code.

0	Disables
1*	Enables

enableQRCodeInverse

[in][out] A value that specifies whether to enable QR Code Inverse.

0*	Regular Only: decodes regular QR Code only
1	Inverse Only: decodes inverse QR Code only
2	Inverse Autodetect: decodes both regular and inverse QR Code

enableMicroQR

[in][out] A value that specifies whether to enable MicroQR.

0	Disables
1*	Enables

enableAztec

[in][out] A value that specifies whether to enable Aztec.

0	Disables
1*	Enables

enableAztecInverse

[in][out] A value that specifies whether to enable Aztec Inverse.

0*	Regular Only: decodes regular Aztec barcodes only
1	Inverse Only: decodes inverse Aztec barcodes only
2	Inverse Autodetect: decodes both regular and inverse Aztec barcodes

Return Value

If successful, it returns 0.
Otherwise, it returns 1.

3.7.20. PICKLISTMODE_2D_SE4500

PicklistMode_2D_SE4500	**2D**

Purpose: Selects whether to enable picklist mode for decoding accuracy.

Syntax:

PicklistMode_2D_SE4500 (int *rw*,
ref int *enable*);

Parameters: The default value (if there is) for each setting is indicated by an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Sets settings

enable

[in][out] A value that specifies whether to enable picklist mode.

0*	Disables
2	Enables

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

- Call GetErrorCode() to find the error condition encountered:

-201	E_IS_INITIAL_MODE

3.7.21. DISPLAYMODE_2D

DisplayMode_2D	**2D**

Purpose: Selects whether to enable display mode.

Syntax:

DisplayMode_2D (int *rw*,
ref int *enable*);

Parameters: A default value comes with an asterisk "*".

rw

[in] A value that specifies the operation.

'r'	Reader.ReaderEngineAPI.READ_PARAM	Gets settings
'w'	Reader.ReaderEngineAPI.WRITE_PARAM	Writes settings

enable

[in][out] A value that specifies whether to enable display mode.

0*	Disables
1	Enables

Return Value: If successful, it returns 0.

Otherwise, it returns 1.

- Call GetErrorCode() to find the error condition encountered:

-201	E_IS_INITIAL_MODE

3.8. RESET READER

ResetReaderToDefault

Purpose	Resets reader module(s).
Syntax	**INT ResetReaderToDefault (INT** *readerType***);**
Parameters	*readerType* [in] A value that specifies the reader type to reset.

7	DC_READER_BC	Resets barcode reader only.
32	DC_READER_RFID	Reset RFID reader only.
255	ALL_DEVICE	Resets both.

Example

```
INT nRlt;
nRlt = ResetReaderToDefault(DC_READER_BC);
```

Return Value

If successful, it returns 1.
Otherwise, it returns a negative value to indicate a specific error:

-101	E_READER_NOT_INIT
-111	E_READER_BC_RESET_FAILED
-112	E_READER_RFID_RESET_FAILED

Remarks: It takes approximately 2 seconds to enforce the reset.

See Also: InitReader

Chapter 4

SYSTEM API

Dynamic linking is strongly recommended for backward compatibility. For .lib, we do not guarante its compatiblility with the .lib releases to come, and thus don't rule out the possibility that you need to re-compile your application to link the latest .lib file.

For RS-232 data connection, use COM 0 on the mobile computer.

Find the header and library usable by your application on the bundled CD-ROM:

- **SystemMobile.h**
- **SystemMobile.lib**

Note: System DLL is accessible within OS directory at **"\Windows\SystemMobile.dll"**.

IN THIS CHAPTER

4.1 System Settings 121
4.2 Signature Capture 124
4.3 LCD Backlight 126
4.4 Keypad Backlight 130
4.5 Touch Panel 131
4.6 Keypad 132
4.7 Microphone 139
4.8 Wireless LAN 143
4.9 Bluetooth 174
4.10 Camera 189
4.11 GPS 198
4.12 Button Assignment 199
4.13 Signature Capture 205

4.1. SYSTEM SETTINGS

4.1.1. UNIVERSALLY UNIQUE IDENTIFIER (UUID)

GetHALUUID

Purpose	To get information about the Universally Unique Identifier (UUID) that is based on an OEM-defined device hardware identifier.
Syntax	**DWORD GetHALUUID (GUID** **guid***);**
Parameters	*guid* [out] Pointer to the structure GUID where the UUID information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	UUID not accessible
4	ERROR_PARAMETER (Wrong parameter)

Remarks	A Universally Unique Identifier (UUID) is an identifier standard that provides a unique 128-bit (16 byte) value used to identify objects. The Globally Unique Identifier (GUID) is a Microsoft implementation of the UUID standard.

4.1.2. DEVICE NAME

GetSysDevName

Purpose	To get the device name that is identical to the one displayed on **Start Screen** | **Settings** | **System** – Device Name tab on the mobile computer.
Syntax	**DWORD GetSysDevName (WCHAR** **deviceName*, **BYTE** *nameSize*);
Parameters	*deviceName* [out] Pointer to a buffer where the device name is stored. *nameSize* [in] The size of the output buffer.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

2	ERROR_NOSPACE (Wrong buffer size)
4	ERROR_PARAMETER (Wrong parameter)

Remarks: By default, the device name is provided by the manufacturer.

SetSysDevName

Purpose	To change the device name.
Syntax	**DWORD SetSysDevName (WCHAR** **deviceName*, **BYTE** *nameSize*);
Parameters	*deviceName* [in] Pointer to a buffer where the device name is stored. *nameSize* [in] The string length of device name. The maximum length of the device name is 15 characters. No spaces are allowed.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks: By default, the device name is provided by the manufacturer.

- To apply the new name, you must warm boot the system.
- The device name may include up to 15 characters from the following ranges: a-z, A-Z, 0-9, or the '-' or '_' character. It must start with 'a-z' or 'A-Z', and cannot end with '-' or '_'.

4.1.3. SYSTEM INFORMATION

GetSysInfo

Purpose	To get the system information, such as serial number, manufacture info, and OS version.
Syntax	**DWORD GetSysInfo (SYSINFO** **sysInfo***);**
Parameters	*sysInfo* [out] Pointer to the structure SYSINFO where the system information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

4.1.4. START PROGRAM

SetInitLoaderBind

Purpose	To bind a program that will be loaded after the system is initialized.
Syntax	**DWORD SetInitLoaderBind (KEYPADBIND** **keypadBind***);**
Parameters	*keypadBind* [in] Pointer to the structure KEYPADBIND where the program information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to open registry
2	Fail to set registry
4	ERROR_PARAMETER (Wrong parameter)

Remarks	The program you bind will be executed before AutoRun.ini and overwrite AutoRun.exe (= AutoRun.exe will never be run).

4.1.5. SOFT RESET

SystemSoftReset

Purpose	To perform software reset (warm boot).
Syntax	**DWORD SystemSoftReset (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)

4.2. STATUS INDICATION

4.2.1. LED LIGHT

SetLEDLight

Purpose: To manipulate the status light (LED).

Syntax:

DWORD SetLEDLight (BYTE *color*,
BOOL *onOff*);

Parameters:

color

[in] BYTE variable

0x00	Red LED
0x01	Green LED
0x02	Amber LED

onOff

[in] Boolean variable

0	Turn off status light
1	Turn on status light

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

Remarks: The time interval required between on and off must be greater than 400 milliseconds.

4.2.2. VIBRATOR

GetVibratorPower

Purpose	To find out the current vibrator state.
Syntax	**DWORD GetVibratorPower (BYTE** **onOff***);**
Parameters	*onOff* [out] Pointer to a buffer where the information is stored.

0	Vibrator OFF
1	Vibrator ON

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

SetVibratorPower

Purpose	To set the vibrator state.
Syntax	**DWORD SetVibratorPower (BYTE** *onOff***);**
Parameters	*onOff* [in] BYTE variable

0	Vibrator OFF
1	Vibrator ON

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

4.3. LCD BACKLIGHT

4.3.1. BACKLIGHT CONTROL

GetBacklightCTL

Purpose	To get the current settings of backlight control.
Syntax	**DWORD GetBacklightCTL (BKLCTL** **bklCtl***);**
Parameters	*bklCtl* [out] Pointer to the structure BKLCTL where the settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetBacklightCTL

Purpose	To change the settings of backlight control.
Syntax	**DWORD SetBacklightCTL (BKLCTL** **bklCtl***);**
Parameters	*bklCtl* [in] Pointer to the structure BKLCTL where the settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)
16	Out of range, backlight level in battery mode.
32	Out of range, backlight level in AC mode.

4.3.2. BACKLIGHT LEVEL

GetBacklightLV

Purpose: To get the current backlight level.

Syntax: **DWORD GetBacklightLV (BYTE** *byFromAC*,
BYTE **byLevel*);

Parameters:

byFromAC
[in] BYTE variable

0	Backlight level for battery mode
1	Backlight level for AC mode

pbyLevel
[out] Pointer to a buffer where the information is stored.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks: The range of backlight level is from 0 (dark) to 11 (bright).
System defaults are level 6 in battery mode and level 11 in AC mode.

SetBacklightLV

Purpose: To set the backlight level.

Syntax: **DWORD SetBacklightLV (BYTE** *byFromAC*,
BYTE *byLevel*);

Parameters:

byFromAC
[in] BYTE variable

0	Backlight level for battery mode
1	Backlight level for AC mode

byLevel
[in] BYTE variable – Level 0 (dark) to Level 11 (bright)

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER (Wrong parameter)

4.3.3. RESET BACKLIGHT TO DEFAULT

SetBacklightDefault

Purpose	To reset the settings of backlight control to defaults.
Syntax	**DWORD SetBacklightDefault (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
16	Out of range, backlight level in battery mode.
32	Out of range, backlight level in AC mode.

Remarks

System defaults are –
DWORD batttimeout=30;
DWORD lightlevel=3;
DWORD actimeout=60;
DWORD aclightlevel=6;
DWORD backlightontap=1;
DWORD acbacklightontap=1;
DWORD usebattery=1;
DWORD useext=0;

See Also: BKLCTL

4.3.4. SET BACKLIGHT TO MAXIMUM

SetBacklightMax

Purpose	To set backlight control for maximum performance.
Syntax	**DWORD SetBacklightMax (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
16	Out of range, backlight level in battery mode.
32	Out of range, backlight level in AC mode.

Remarks

The values for maximum performance are –
DWORD batttimeout=300;
DWORD lightlevel=10;
DWORD actimeout=600;
DWORD aclightlevel=10;
DWORD backlightontap=1;
DWORD acbacklightontap=1;
DWORD usebattery=1;
DWORD useext=1;

See Also: SetBacklightDefault, BKLCTL

4.3.5. SET BACKLIGHT TO MINIMUM

SetBacklightMin	
Purpose	To set backlight control for minimum performance.
Syntax	**DWORD SetBacklightMin (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
16	Out of range, backlight level in battery mode.
32	Out of range, backlight level in AC mode.

Remarks	The values for minimum performance are – DWORD batttimeout=30; DWORD lightlevel=0; DWORD actimeout=60; DWORD aclightlevel=0; DWORD backlightontap=0; DWORD acbacklightontap=1; DWORD usebattery=1; DWORD useext=1;
See Also	SetBacklightDefault, BKLCTL

4.4. KEYPAD BACKLIGHT

GetKeylightOnOff

Purpose: To find out the keypad backlight status.

Syntax: **DWORD GetKeylightOnOff (BYTE** **onOff***);**

Parameters: *onOff*

[out] Pointer to a buffer where the information is stored.

0	Keypad backlight is off
1	Keypad backlight is on

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetKeylightOnOff

Purpose: To set the keypad backlight.

Syntax: **DWORD SetKeylightOnOff (BYTE** *onOff***);**

Parameters: *onOff*

[in] BYTE variable

0	Set keypad backlight off
1	Set keypad backlight on

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

4.5. TOUCH PANEL

4.5.1. TOUCH PANEL STATE

GetTchLock

Purpose	To find out the lock state of touch panel.
Syntax	**DWORD GetTchLock (BYTE** **lockState***);**
Parameters	*lockState* [out] Pointer to a buffer where the information is stored.

0	Touch panel unlocked
1	Touch panel locked

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get touch panel handle
4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

SetTchLock

Purpose	To set the lock state of touch panel.
Syntax	**DWORD SetTchLock (BYTE** *lockState***);**
Parameters	*lockState* [in] BYTE variable

0	Touch panel unlocked
1	Touch panel locked

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get touch panel handle
4	ERROR_PARAMETER (Wrong parameter)
5	ERROR_API_FUNCTION (Fail to call System API))

Note: You cannot lock both the touch panel and keypad at the same time!

4.6. KEYPAD

4.6.1. SENSITIVITY

GetKeypadSensitivity

Purpose	To find out the repeat rate of a key being held down for more than one second.
Syntax	**DWORD GetKeypadSensitivity (DWORD** **value***);**
Parameters	*value* [out] Pointer to a buffer where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

Remarks	The greater the value is, the more times the character repeats itself (per second) when you hold down a key for more than one second. ▸ By default, it is set to 6 times per second.

SetKeypadSensitivity

Purpose	To set the repeat rate of a key being held down for more than one second.
Syntax	**DWORD SetKeypadSensitivity (DWORD** *value***);**
Parameters	*value* [in] 32-bit unsigned integer variable The greater the value is, the more times the character repeats itself (max. 20 times per second) when you hold down a key for more than one second.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

4.6.2. KEYPAD LOCK

GetKeypadLock

Purpose: To find out the lock state of keypad.

Syntax:

DWORD GetKeypadLock (DWORD *key*,
DWORD **lockState*);

Parameters:

key

[in] 32-bit unsigned integer variable

This parameter cannot be a combination of the following values:

1	KEY_GENERAL	General key state
2	KEY_ALPHA	[Alpha] key state
4	KEY_FN	[Fn] key state
32	KEY_SCAN	[Scan] key state
64	KEY_POWER	[Power] key state
128	KEY_LSTRIGGER	Left side trigger state
256	KEY_RSTRIGGER	Right side trigger state
512	KEY_SHIFT	[Shift] key state
0xFFFFFFFF	KEY_ALL	Whole keypad state

lockState

[out] Pointer to a buffer where the information is stored.

0	Unlocked
1	Locked

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

Remarks: KEY_ALL = KEY_GENERAL + KEY_ALPHA + KEY_FN + KEY_SCAN + KEY_POWER + KEY_LSTRIGGER + KEY_RSTRIGGER + KEY_SHIFT

SetKeypadLock

Purpose

To set the lock state of keypad.

Syntax

DWORD SetKeypadLock (DWORD *key*,
DWORD *lockState*);

Parameters

key

[in] 32-bit unsigned integer variable

This parameter can be a combination of the following values:

1	KEY_GENERAL	General key state
2	KEY_ALPHA	[Alpha] key state
4	KEY_FN	[Fn] key state
32	KEY_SCAN	[Scan] key state
64	KEY_POWER	[Power] key state
128	KEY_LSTRIGGER	Left side trigger state
256	KEY_RSTRIGGER	Right side trigger state
512	KEY_SHIFT	[Shift] key state
0xFFFFFFFF	KEY_ALL	Whole keypad state

lockState

[in] 32-bit unsigned integer variable

0	Unlock
1	Lock

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
2	Touch panel locked
4	ERROR_PARAMETER (Wrong parameter)

Remarks

KEY_ALL = KEY_GENERAL + KEY_ALPHA + KEY_FN + KEY_SCAN + KEY_POWER + KEY_LSTRIGGER + KEY_RSTRIGGER + KEY_SHIFT

4.6.3. KEYPAD MODE

- Auto Resume mode — The function mode is toggled on by pressing the function key; it is toggled off by pressing the second key of the key combination. A status icon is displayed on the screen to indicate the status.
- Toggle mode — The function mode is toggled on by pressing the function key; it can only be toggled off by pressing the function key again. A status icon is displayed on the screen to indicate the status.
- Multi-Key mode — For any key combination, it requires pressing two keys at the same time, or holding down the function key followed by the second key.

GetKeypadMode

Purpose: To find out the key mode for a special key.

Syntax:

DWORD GetKeypadMode (DWORD *key*,
DWORD **mode*);

Parameters:

key

[in] 32-bit unsigned integer variable

4	KEY_FN	To get [Fn] key mode.

mode

[out] Pointer to a buffer where the information is stored.

0	Toggle mode
1	Auto Resume mode
2	Multi-Key mode

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

SetKeypadMode

Purpose

To set the key mode for a special key.

Syntax

DWORD SetKeypadMode (DWORD *key*,
DWORD *mode*);

Parameters

key

[in] 32-bit unsigned integer variable

4	KEY_FN	To set [Fn] key mode.

mode

[in] 32-bit unsigned integer variable

0	Toggle mode
1	Auto Resume mode
2	Multi-Key mode

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

4.6.4. KEYPAD STATE

GetKeypadState

Purpose: To find out the key state for a special key.

Syntax:

DWORD GetKeypadState (DWORD *key*,
DWORD **state*);

Parameters:

key

[in] 32-bit unsigned integer variable

2	KEY_ALPHA	To get [Alpha] key state.
4	KEY_FN	To get [Fn] key state.

state

[out] Pointer to a buffer where the information is stored.

	Alpha key	FN key
0	Numeric	Inactive
1	N/A	Active
2	Lower-case	N/A

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
4	ERROR_PARAMETER (Wrong parameter)

SetKeypadState

Purpose

To set the key state for a special key.

Syntax

DWORD SetKeypadState (DWORD *key*,
DWORD *state*);

Parameters

key

[in] 32-bit unsigned integer variable

2	KEY_ALPHA	To set [Alpha] key state.
4	KEY_FN	To set [Fn] key state.

state

[in] 32-bit unsigned integer variable

	Alpha key	FN key
0	Numeric	Inactive
1	N/A	Active
2	Lower-case	N/A

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get key handle
2	Key locked
4	ERROR_PARAMETER (Wrong parameter)

4.7. MICROPHONE

4.7.1. DEVICE MICROPHONE

SetAMicGain

Purpose: To set a customized device microphone gain level.

Syntax: **DWORD SetAMicGain (unsigned int** *level***);**

Parameters: *level*

[in] Unsigned integer variable

A value that specifies the level of device microphone gain.

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

GetAMicGain

Purpose

To get the customized device microphone gain level.

Syntax

DWORD GetAMicGain (unsigned int **level***);**

Parameters

level

[out] Unsigned integer variable

A value that specifies the level of device microphone gain.

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

- Before using this function, set a default level first with SetAMicGain.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

4.7.2. HEADSET MICROPHONE

SetHSMicGain

Purpose: To set a customized headset microphone gain level.

Syntax: **DWORD SetHSMicGain (unsigned int** *level***);**

Parameters:

level

[in] Unsigned integer variable

A value that specifies the level of headset microphone gain.

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

GetHSMicGain

Purpose: To get the customized headset microphone gain level.

Syntax: **DWORD GetHSMicGain (unsigned int** *_level_**);**

Parameters: *level*

[out] Unsigned integer variable

A value that specifies the level of headset microphone gain.

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

- Before using this function, set a default level first with SetHSMicGain.

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

4.8. WIRELESS LAN

4.8.1. WI-FI POWER

GetWiFiPower

Purpose	To find out the current power state of the Wi-Fi module.
Syntax	**DWORD GetWiFiPower (BYTE** **onOff***);**
Parameters	*onOff* [out] Pointer to a buffer where the information is stored.

0	Turn off the power to the Wi-Fi module
1	Turn on the power to the Wi-Fi module

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to get Wi-Fi power state
4	ERROR_PARAMETER (Wrong parameter)

SetWiFiPower

Purpose	To set the power state of the Wi-Fi module.
Syntax	**DWORD SetWiFiPower (BYTE** *onOff***);**
Parameters	*onOff* [in] BYTE variable

0	Turn off the power to the Wi-Fi module
1	Turn on the power to the Wi-Fi module

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to open Wi-Fi device handle
2	Fail to set Wi-Fi power
4	ERROR_PARAMETER (Wrong parameter)

4.8.2. RADIO

RadioDisable	
Purpose	To disable the radio.
Syntax	**SDCERR RadioDisable (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

RadioEnable	
Purpose	To enable the radio.
Syntax	**SDCERR RadioEnable (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.3. IP INFORMATION

GetWlanIpInfo	
Purpose	To get the IP information for wireless networking.
Syntax	**DWORD GetWlanIpInfo (WLANADPTINFO** **adpt***);**
Parameters	*adpt* [out] Pointer to the structure WLANADPTINFO where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks	When DHCP is enabled, this function will get DHCP status only.

SetWlanIpInfo	
Purpose	To set the IP information for wireless networking.
Syntax	**DWORD SetWlanIpInfo (WLANADPTINFO** **adpt***);**
Parameters	*adpt* [in] Pointer to the structure WLANADPTINFO where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

4.8.4. DOMAIN, SDK & MAC INFORMATION

GetCurrentDomain

Purpose	To find out the regulatory domain or domains for which the radio is configured.
Syntax	**REG_DOMAIN GetCurrentDomain (VOID);**
Return Value	If successful, it returns the regulatory domain:

0	REG_FCC	(Americas)
1	REG_ETSI	(Europe)
2	REG_TELEC	(Japan)
3	REG_WW	(Worldwide)
4	REG_KCC	(Korea)

Remarks	This function reads the values from SROM and the execution time is significant. Therefore, it must not be called frequently

GetSDKVersion

Purpose	To find out Summit radio SDK version.
Syntax	**SDCERR GetSDKVersion (DWORD** **version***);**
Parameters	*pVersion* [out] Pointer to a buffer where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

GetWiFiMac

Purpose	To get the MAC address of Summit radio.
Syntax	**DWORD GetWiFiMac (ULONGLONG** **wifiMac***);**
Parameters	*wifiMac* [out] Pointer to a buffer where the MAC address is stored.
Return Value	If successful, it returns 0. Otherwise, it returns one of the following values to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
2	ERROR_API_FUNCTION (Fail to call System API))
3	Wi-Fi power not ready
4	ERROR_PARAMETER (Wrong parameter)

4.8.5. NETWORK LIST

AddWlanSsidToPreferredList

Purpose	To add a wireless network (via an access point) to the preferred list.
Syntax	**DWORD AddWlanSsidToPreferredList (WLANCTL&** *wlanCtl1***);**
Parameters	*wlanCtl1* [in] Structure variable. Refer to the structure WLANCTL.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)
8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)
16	ERROR_WLAN_SSID_REPEAT

See Also	GetWlanPreferredList, ResetWlanPreferredList

GetWlanAvailableList

Purpose	To get the list of available wireless networks (via access points).
Syntax	**DWORD GetWlanAvailableList (RAW_DATA*** *pAvailableList***);**
Parameters	*pAvailableList* [in][out] Pointer to the structure RAW_DATA where the list is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

2	ERROR_NOSPACE (Buffer is too small)
4	ERROR_PARAMETER (Wrong parameter)
8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)
19	ERROR_WLAN_NO_AVAILABLE_LIST

Remarks	Refer to System API's sample code.
See Also	GetWlanPreferredList

GetWlanPreferredList	
Purpose	To get the list of preferred wireless networks (via access points).
Syntax	**DWORD GetWlanPreferredList (RAW_DATA*** *pPreferredList***);**
Parameters	*pPreferredList* [in][out] Pointer to the structure RAW_DATA where the list is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

2	ERROR_NOSPACE (Buffer is too small)
4	ERROR_PARAMETER (Wrong parameter)
8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)
17	ERROR_WLAN_NO_PRSSID_LIST

Remarks	Refer to System API's sample code.
See Also	AddWlanSsidToPreferredList, GetWlanAvailableList, ReconnectWlanPreferredList, ResetWlanPreferredList

ResetWlanPreferredList	
Purpose	To clear the list of preferred wireless networks (via access points).
Syntax	**DWORD ResetWlanPreferredList (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)

Remarks	It will terminate the current connection.
See Also	RAW_DATA, AddWlanSsidToPreferredList

4.8.6. NETWORK STATUS

GetWlanConnectedStatus

Purpose	To get the status of current connection.
Syntax	**DWORD GetWlanConnectedStatus (WLANCONNECTEDST*** *pWlanConnSt***);**
Parameters	*pWlanConnSt* [in][out] Pointer to the structure WLANCONNECTEDST where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER (Wrong parameter)
8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)
18	ERROR_WLAN_NO_ASSOCIATED
20	ERROR_WLAN_NO_CONNECTED

ReconnectWlanPreferredList

Purpose	To reconnect to an access point on the list of preferred wireless networks.
Syntax	**DWORD ReconnectWlanPreferredList (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

8	ERROR_WLAN_QUIRY_INTERFACE (Fail to inquiry interface)

See Also	RAW_DATA, GetWlanPreferredList

4.8.7. BSSID

ScanWiFiBSSID	
Purpose	To scan available access points.
Syntax	**DWORD ScanWiFiBSSID (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER (Wrong parameter)

GetWiFiBSSIDList	
Purpose	To obtain the BSSID list.
Syntax	**DWORD GetWiFiBSSIDList (BYTE** **buffer***);**
Parameters	*buffer* [out] Pointer to a buffer where the information is stored. Buffer size must be greater than sizeof (NDISUIO_QUERY_OID) + sizeof (NDIS_802_11_BSSID_LIST_EX) + 50 * sizeof (NDIS_WLAN_BSSID_EX)
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER (Wrong parameter)

4.8.8. CONFIGURATION PROFILES

ActivateConfig

Purpose	To activate a configuration profile.
Syntax	**SDCERR ActivateConfig (CHAR ****name***);**
Parameters	*name* [in] Pointer to a buffer where the profile name is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

GetCurrentConfig

Purpose	To find out the configuration currently in use.
Syntax	**SDCERR GetCurrentConfig (DWORD ****num***,** **CHAR ****name***);**
Parameters	*num* [out] Pointer to a buffer where the information is stored.

0	ThirdPartyConfig is active **(Reserved)**
1~	Identifier of the active configuration profile

name

[out] Pointer to a buffer where the name of configuration profile is stored.

- Buffer size must be greater than CONFIG_NAME_SZ.

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

GetNumConfigs

Purpose	To get the total number of available configuration profiles.
Syntax	**SDCERR GetNumConfigs (DWORD ****num***);**
Parameters	*num* [out] Pointer to a buffer where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.9. EDIT CONFIGURATION PROFILE

AddConfig	
Purpose	To add a configuration profile.
Syntax	**SDCERR AddConfig (SDCConfig** **config***);**
Parameters	*config* [in] Pointer to the structure SDCConfig where the configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

DeleteConfig	
Purpose	To delete a configuration profile.
Syntax	**SDCERR DeleteConfig (CHAR** **name***);**
Parameters	*name* [in] Pointer to a buffer where the profile name is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	▸ You are not allowed to delete the active configuration. ▸ NULL is not a valid name.

SetDefaultConfigValues	
Purpose	To set default values for new configurations.
Syntax	**SDCERR SetDefaultConfigValues(SDCConfig** **cfg***);**
Parameters	*cfg* [in] Pointer to the structure SDCConfig where the configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

CreateConfig	
Purpose	To create a configuration from the default values.
Syntax	**SDCERR CreateConfig(SDCConfig** **cfg***);**
Parameters	*cfg* [in] Pointer to the structure SDCConfig where the configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	▸ You must add the config after it is created. ▸ You must allocate the config memory.

GetConfig	
Purpose	To get a configuration profile.
Syntax	**SDCERR GetConfig (CHAR** **name***,** **SDCConfig** **config***);**
Parameters	*name* [in] Pointer to a buffer where the profile name is stored. *config* [out] Pointer to the structure SDCConfig where configuration is stored. ▸ Cannot be NULL.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

ModifyConfig	
Purpose	To modify a configuration profile.
Syntax	**SDCERR ModifyConfig (CHAR** **name***,** **SDCConfig** **config***);**
Parameters	*name* [in] Pointer to a buffer where the profile name is stored. Name of the configuration to update. *config* [in] Pointer to the structure SDCConfig where configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.10. SUMMIT CONFIG SETTINGS

GetAllConfigs	
Purpose	To retrieve all configuration profiles.
Syntax	**SDCERR GetAllConfigs (SDCConfig** **config*, **DWORD** **num*);
Parameters	*config* [out] Pointer to the structure SDCConfig where configuration is stored. ▸ The total elements of this array must be greater than MAX_CFGS defined in the header file. *num* [out] Pointer to a buffer where the number of elements is stored.

SetAllConfigs	
Purpose	To overwrite all configuration profiles.
Syntax	**SDCERR SetAllConfigs (DWORD** *num*, **SDCConfig** **config*);
Parameters	*num* [in] 32-bit unsigned integer variable ▸ A value that specifies the number of elements in array. *config* [in] Pointer to the structure SDCConfig where configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.11. GLOBAL CONFIG SETTINGS

GetGlobalSettings	
Purpose	To retrieve the global configuration settings.
Syntax	**SDCERR GetGlobalSettings (SDCGlobalConfig** **configGlobal***);**
Parameters	*configGlobal* [out] Pointer to the structure SDCGlobalConfig where configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetGlobalSettings	
Purpose	To overwrite the global configuration settings and restart the Wi-Fi module.
Syntax	**SDCERR SetGlobalSettings (SDCGlobalConfig** **configGlobal***);**
Parameters	*configGlobal* [in] Pointer to the structure SDCGlobalConfig where configuration is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.12. SUMMIT REGISTRY KEYS

FlushConfigKeys

Purpose	To flush the registry keys of a configuration into the storage.
Syntax	**SDCERR FlushConfigKeys (INT** *configNumber***);**
Parameters	*configNumber* [in] Integer variable

-1	Flush global settings
0	Flush the third party configuration **(Reserved)**
1 ~ MAX_CFGS	Flush the configuration profile by identifier.

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	This function forces a flush so the Summit parameters are saved if a power cycle occurs before the system flushes the registry.

FlushAllConfigKeys

Purpose	To flush all registry keys related to Summit configuration.
Syntax	**SDCERR FlushAllConfigKeys (VOID);**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	This function forces a flush so the Summit parameters are saved if a power cycle occurs before the system flushes the registry.

4.8.13. CONFIGURATION FILE

GetConfigFileInfo	
Purpose	To retrieve information of a Summit configuration file.
Syntax	**SDCERR GetConfigFileInfo (CHAR** **fileName***,** **CONFIG_FILE_INFO** **info***);**
Parameters	*fileName* [in] Pointer to a buffer where the filename is stored. *info* [out] Pointer to the structure CONFIG_FILE_INFO where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

ExportSettings	
Purpose	To export configurations and global settings to a specified file.
Syntax	**SDCERR ExportSettings (CHAR** **fileName***,** **SDC_ALL** **configAll***);**
Parameters	*fileName* [in] Pointer to a buffer where the filename is stored. *configAll* [in] Pointer to the structure SDC_ALL where the all the settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	Refer to structure SDC_ALL for details. - configGlobal – Either NULL to skip global config export or a valid pointer - configThirdParty – Either NULL to skip third party config export or a valid pointer - configs – Either NULL to skip config export or a valid pointer to one or more SDCConfig structures - numConfigs 0 – To skip all SDCConfigs or the number of configurations (SDCConfig) to export. Do not include configGloball or configThirdParty in this count.

ImportSettings	
Purpose	To import configurations and global settings from a specified file.
Syntax	**SDCERR ImportSettings (CHAR** **fileName*, **SDC_ALL** **configAll*);
Parameters	*fileName* [in] Pointer to a buffer where the filename is stored. *configAll* [out] Pointer to the structure SDC_ALL where the all the settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).
Remarks	Refer to structure SDC_ALL for details. ▸ configGlobal – Either NULL to skip global configuration import or a valid pointer to an allocated structure ▸ configThirdParty – Either NULL to skip ThirdPartyConfig import or a valid pointer to an allocated structure ▸ configs – Either NULL to skip config import or a valid pointer to one or more allocated SDCConfig structures ▸ numConfigs – Set to the number of configurations (SDCConfig) exported. This count doesn't include configGlobal or configThirdParty. You must allocate the memory in the SDC_ALL structure-configGlobal, configThirdParty and up to MAX_CFGS configs.

4.8.14. WEP KEY

GetWEPKey

Purpose: To retrieve a WEP key.

Syntax:

```
SDCERR GetWEPKey (SDCConfig *config,
                  INT keyIndex,
                  WEPLEN *keyLength,
                  UNSIGNED CHAR *wepKey,
                  BOOLEAN *keyState);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

keyIndex

[in] Integer variable

1 ~ 4	Index of the WEP key to be retrieved

keyLength

[out] A value that specifies the key length; if NULL, this parameter is ignored.

0	WEPLEN_NOT_SET	(Clear the key.)
1	WEPLEN_40BIT	(The key must be 10 hex characters.)
2	WEPLEN_128BIT	(The key must be 26 hex characters.)

wepKey

[out] Pointer to a buffer where the WEP key is stored.

- Buffer size must be greater than 26 bytes.

keyState

[out] Boolean variable

0	The key is not activated
1	The key is activated

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetWEPKey

Purpose: To set a WEP key.

Syntax:

```
SDCERR SetWEPKey (SDCConfig *config,
                  INT keyIndex,
                  WEPLEN keyLength,
                  UNSIGNED CHAR *wepKey,
                  BOOLEAN keyState);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

keyIndex

[in] Integer variable

1 ~ 4	Index of the WEP key to be set

keyLength

[in] A value that specifies the key length.

0	WEPLEN_NOT_SET	(Clear the key.)
1	WEPLEN_40BIT	(The key must be 10 hex characters.)
2	WEPLEN_128BIT	(The key must be 26 hex characters.)

wepKey

[in] Pointer to a buffer where the WEP key is stored.

- Buffer size must be greater than 26 bytes.

keyState

[in] Boolean variable

0	Do not activate the key
1	Activate the key

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

GetMultipleWEPKeys

Purpose

To retrieve all four WEP keys.

Syntax

```
SDCERR GetMultipleWEPKey (SDCConfig *config,
                          INT *keyIndex,
                          WEPLEN *keyLength1,
                          UNSIGNED CHAR *wepKey1,
                          WEPLEN *keyLength2,
                          UNSIGNED CHAR *wepKey2,
                          WEPLEN *keyLength3,
                          UNSIGNED CHAR *wepKey3,
                          WEPLEN *keyLength4,
                          UNSIGNED CHAR *wepKey4);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

keyIndex

[out] Integer variable

1 ~ 4	Index of the active WEP key

keyLength1(~4)

[out] A value that specifies the key length.

0	WEPLEN_NOT_SET	(Clear the key.)
1	WEPLEN_40BIT	(The key must be 10 hex characters.)
2	WEPLEN_128BIT	(The key must be 26 hex characters.)

wepKey1(~4)

[out] Pointer to a buffer where the WEP key 1(~4) is stored.

Buffer size must be greater than 26 bytes.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetMultipleWEPKeys

Purpose

To set all four WEP keys.

Syntax

SDCERR SetMultipleWEPKey (SDCConfig **config*,
INT *keyIndex*,
WEPLEN *keyLength1*,
UNSIGNED CHAR **wepKey1*,
WEPLEN *keyLength2*,
UNSIGNED CHAR **wepKey2*,
WEPLEN *keyLength3*,
UNSIGNED CHAR **wepKey3*,
WEPLEN *keyLength4*,
UNSIGNED CHAR **wepKey4*);

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

keyIndex

[in] Integer variable

1 ~ 4	

keyLength1(~4)

[in] A value that specifies the key length.

0	WEPLEN_NOT_SET	(Clear the key.)
1	WEPLEN_40BIT	(The key must be 10 hex characters.)
2	WEPLEN_128BIT	(The key must be 26 hex characters.)

wepKey1(~4)

[in] Pointer to a buffer where the WEP key 1(~4) is stored.

- Buffer size must be greater than 26 bytes.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.15. PRE-SHARED KEY

GetPSK	
Purpose	To retrieve the Pre-Shared Key.
Syntax	**SDCERR GetPSK (SDCConfig** **config*, **CHAR** **psk*);
Parameters	*config* [in] Pointer to the structure SDCConfig where configuration is stored. *psk* [out] Pointer to a buffer where the Pre-Shared Key is stored. ▸ Buffer size must be greater than PSK_SZ (bytes) defined in the header file.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetPSK	
Purpose	To set the Pre-Shared Key.
Syntax	**SDCERR SetPSK (SDCConfig** **config*, **CHAR** **psk*);
Parameters	*config* [in] Pointer to the structure SDCConfig where configuration is stored. *psk* [in] Pointer to a buffer where the Pre-Shared Key is stored. ▸ Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than PSK_SZ (bytes) defined in the header file. ▸ For PSK, the string must be 64 hex characters. ▸ For Passphrase, the string must be 8~63 printable ASCII characters.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.16. LEAP CREDENTIALS

GetLEAPCred	
Purpose	To retrieve the LEAP credentials.
Syntax	**SDCERR GetLEAPCred (SDCConfig** **config*, **CHAR** **username*, **CHAR** **password*);
Parameters	*config* [in] Pointer to the structure SDCConfig where configuration is stored. *username* [out] Pointer to a buffer where the username is stored. ▸ Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. *password* [out] Pointer to a buffer where the password is stored. Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file. ▸ Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetLEAPCred	
Purpose	To set the LEAP credentials.
Syntax	**SDCERR SetLEAPCred (SDCConfig** **config*, **CHAR** **username*, **CHAR** **password*);
Parameters	*config* [in] Pointer to the structure SDCConfig where configuration is stored. *username* [in] Pointer to a buffer where the username is stored. ▸ Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_NAME_SZ (bytes) defined in the header file. *password* [in] Pointer to a buffer where the password is stored. ▸ Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_PWD_SZ (bytes) defined in the header file.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.17. EAP-FAST CREDENTIALS

GetEAPFASTCred	
Purpose	To retrieve the EAP-FAST credentials.
Syntax	**SDCERR GetEAPFASTCred (SDCConfig** **config*, **CHAR** **username*, **CHAR** **password*, **CHAR** **pacFileName*, **CHAR** **pacPassword*);
Parameters	*config* [in] Pointer to the structure SDCConfig where configuration is stored. *username* [out] Pointer to a buffer where the username is stored. ▸ Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. *password* [out] Pointer to a buffer where the password is stored. ▸ Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file. *pacFileName* [out] Pointer to a buffer where the PAC filename is stored. ▸ Buffer size must be greater than CRED_PFILE_SZ (bytes) defined in the header file. *pacPassword* [out] Pointer to a buffer where the PAC password is stored. ▸ Buffer size must be greater than CRED_PFILE_SZ (bytes) defined in the header file.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetEAPFASTCred	
Purpose	To set the EAP-FAST credentials.
Syntax	**SDCERR SetEAPFASTCred (SDCConfig** **config*, **CHAR** **username*, **CHAR** **password*, **CHAR** **pacFileName*, **CHAR** **pacPassword*);

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[in] Pointer to a buffer where the username is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_NAME_SZ (bytes) defined in the header file.

password

[in] Pointer to a buffer where the password is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_PWD_SZ (bytes) defined in the header file.

pacFileName

[in] Pointer to a buffer where the PAC filename is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than CRED_PFILE_SZ (bytes) defined in the header file.

pacPassword

[in] Pointer to a buffer where the PAC password is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than CRED_PFILE_SZ (bytes) defined in the header file.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.18. PEAP-GTC CREDENTIALS

GetPEAPGTCCred

Purpose: To retrieve the PEAP-GTC credentials.

Syntax:

```
SDCERR GetPEAPGTCCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CERTLOCATION *certLocation,
                       CHAR *caCert);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[out] Pointer to a buffer where the username is stored.

Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

password

[out] Pointer to a buffer where the password is stored.

Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

certLocation

[out] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[out] Pointer to a buffer where CA certificate is stored.

- Buffer size must be greater than CRED_CERT_SZ (bytes) defined in the header file. CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetPEAPGTCCred

Purpose

To set the PEAP-GTC credentials.

Syntax

SDCERR SetPEAPGTCCred (SDCConfig **config*,
CHAR **username*,
CHAR **password*,
CERTLOCATION *certLocation*,
CHAR **caCert*);

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[in] Pointer to a buffer where the username is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_NAME_SZ (bytes) defined in the header file.

password

[in] Pointer to a buffer where the password is stored.

- Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_PWD_SZ (bytes) defined in the header file.

certLocation

[in] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[in] CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.19. PEAP-MSCHAP CREDENTIALS

GetPEAPMSCHAPCred

Purpose: To retrieve the PEAP-MSCHAP credentials.

Syntax:

```
SDCERR GetPEAPMSCHAPCred (SDCConfig *config,
                          CHAR *username,
                          CHAR *password,
                          CERTLOCATION *certLocation,
                          CHAR *caCert);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[out] Pointer to a buffer where the username is stored.

- Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

password

[out] Pointer to a buffer where the password is stored.

- Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

certLocation

[out] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[out] Pointer to a buffer where CA certificate is stored.

- Buffer size must be greater than CRED_CERT_SZ (bytes) defined in the header file. CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetPEAPMSCHAPCred

Purpose

To set the PEAP-MSCHAP credentials.

Syntax

SDCERR SetPEAPMSCHAPCred (SDCConfig **config*,
CHAR **username*,
CHAR **password*,
CERTLOCATION *certLocation*,
CHAR **caCert*);

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[in] Pointer to a buffer where the username is stored.

- Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

password

[in] Pointer to a buffer where the password is stored.

- Buffer size must be greater than USER_PWD_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

certLocation

[in] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[in] Pointer to a buffer where CA certificate is stored.

- Buffer size must be greater than CRED_CERT_SZ (bytes) defined in the header file. CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.20. EAP-TLS CREDENTIALS

GetEAPTLSCred

Purpose

To retrieve the EAP-TLS credentials.

Syntax

```
SDCERR GetEAPTLSCred (SDCConfig *config,
                      CHAR *username,
                      CHAR *userCert,
                      CERTLOCATION *certLocation,
                      CHAR *caCert);
```

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[out] Pointer to a buffer where the username is stored.

- Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

userCert

[out] Pointer to a buffer where the user certificate is stored.

- Buffer size must be greater than 20 bytes. “userCert” is a 20-byte hash value representing one specific certificate from the Microsoft certificate store. Pass NULL to ignore this parameter.

certLocation

[out] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	“caCert” is set to NULL. Don’t validate the server.
1	CERT_FILE	“caCert” is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	“caCert” is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	“caCert” is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[out] Pointer to a buffer where CA certificate is stored.

- Buffer size must be greater than CRED_CERT_SZ (bytes) defined in the header file. CA certificate depends on the value of “certLocation” above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetEAPTLSCred

Purpose

To set the EAP-TLS credentials.

Syntax

SDCERR SetEAPTLSCred (SDCConfig **config*,
CHAR **username*,
CHAR **userCert*,
CERTLOCATION *certLocation*,
CHAR **caCert*);

Parameters

config

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[in] Pointer to a buffer where the username is stored.

Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_NAME_SZ (bytes) defined in the header file.

userCert

[in] Pointer to a buffer where the password is stored.

Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_PWD_SZ (bytes) defined in the header file.

certLocation

[in] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[in] CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.8.21. EAP-TTLS CREDENTIALS

GetEAPTTLSCred

Purpose

To retrieve the EAP-TTLS credentials.

Syntax

```
SDCERR GetEAPTTLSCred (SDCConfig *cfg,
                       char *username,
                       char *password,
                       CERTLOCATION certLocation,
                       char *caCert);
```

Parameters

cfg

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[out] Pointer to a buffer where the username is stored.

Buffer size must be greater than USER_NAME_SZ (bytes) defined in the header file. Pass NULL to ignore this parameter.

password

[out] Pointer to a buffer where the user certificate is stored.

Buffer size must be greater than 20 bytes. “userCert” is a 20-byte hash value representing one specific certificate from the Microsoft certificate store. Pass NULL to ignore this parameter.

certLocation

[out] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	“caCert” is set to NULL. Don’t validate the server.
1	CERT_FILE	“caCert” is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	“caCert” is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	“caCert” is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[out] Pointer to a buffer where CA certificate is stored.

Buffer size must be greater than CRED_CERT_SZ (bytes) defined in the header file. CA certificate depends on the value of “certLocation” above. Pass NULL to ignore this parameter.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

SetEAPTTLSCred

Purpose

To set the EAP-TTLS credentials.

Syntax

```
SDCERR SetEAPTTLSCred (SDCConfig *cfg,
                       char *username,
                       char *password,
                       CERTLOCATION certLocation,
                       char *caCert);
```

Parameters

cfg

[in] Pointer to the structure SDCConfig where configuration is stored.

username

[in] Pointer to a buffer where the username is stored.

Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_NAME_SZ (bytes) defined in the header file.

password

[in] Pointer to a buffer where the password is stored.

Its value is a null-terminated string, and will be cleared when NULL is passed to the argument. The string length must be no greater than USER_PWD_SZ (bytes) defined in the header file.

certLocation

[out] A value that specifies whether to use a CA certificate to validate an authentication server. Pass NULL to ignore this parameter.

certLocation		caCert
0	CERT_NONE	"caCert" is set to NULL. Don't validate the server.
1	CERT_FILE	"caCert" is the filename of root certificate authority (CA) digital certificate; string length must be no greater than CRED_CERT_SZ (bytes).
2	CERT_FULL_STORE	"caCert" is set to NULL. It will search for a valid certificate from the entire Microsoft certificate store.
3	CERT_IN_STORE	"caCert" is a 20-byte hash value representing one specific certificate from the Microsoft certificate store.

caCert

[in] CA certificate depends on the value of "certLocation" above. Pass NULL to ignore this parameter.

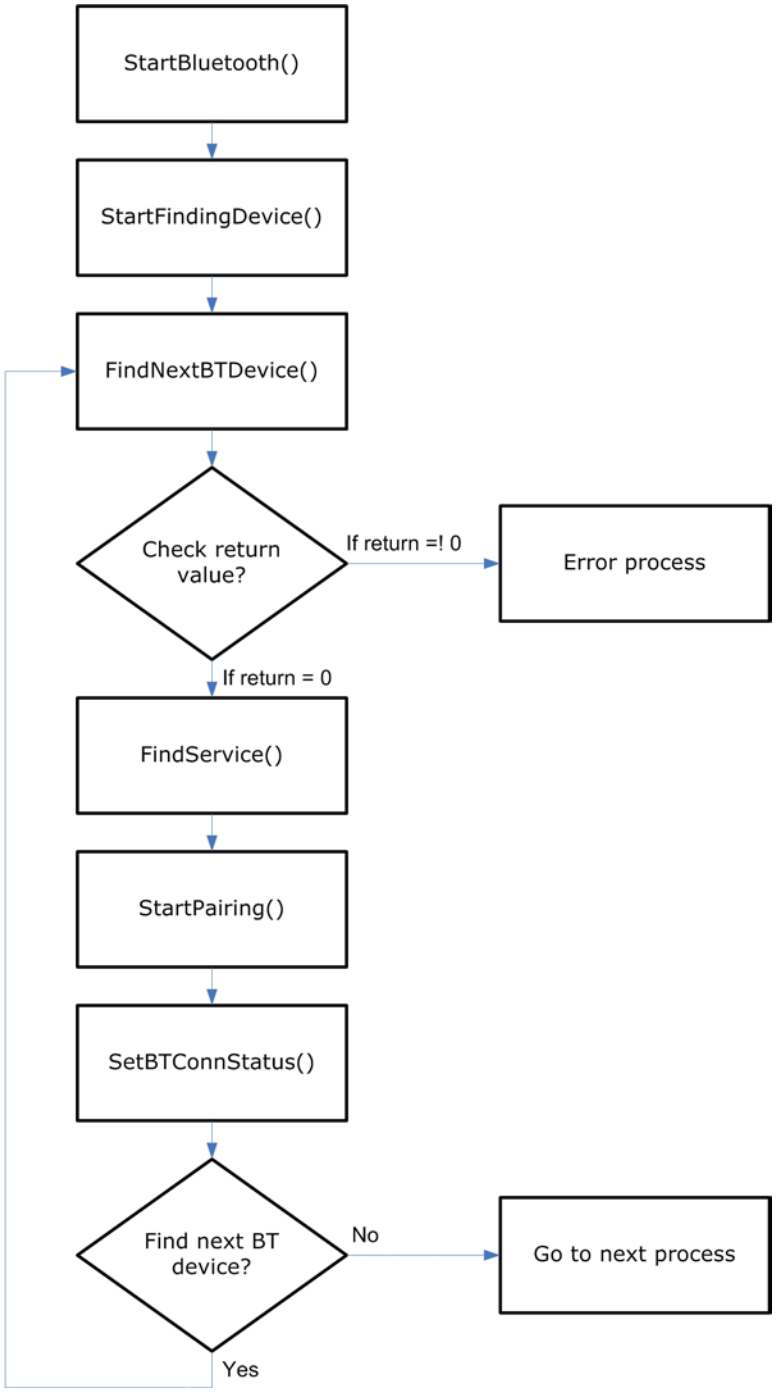
Return Value

If successful, it returns 0.

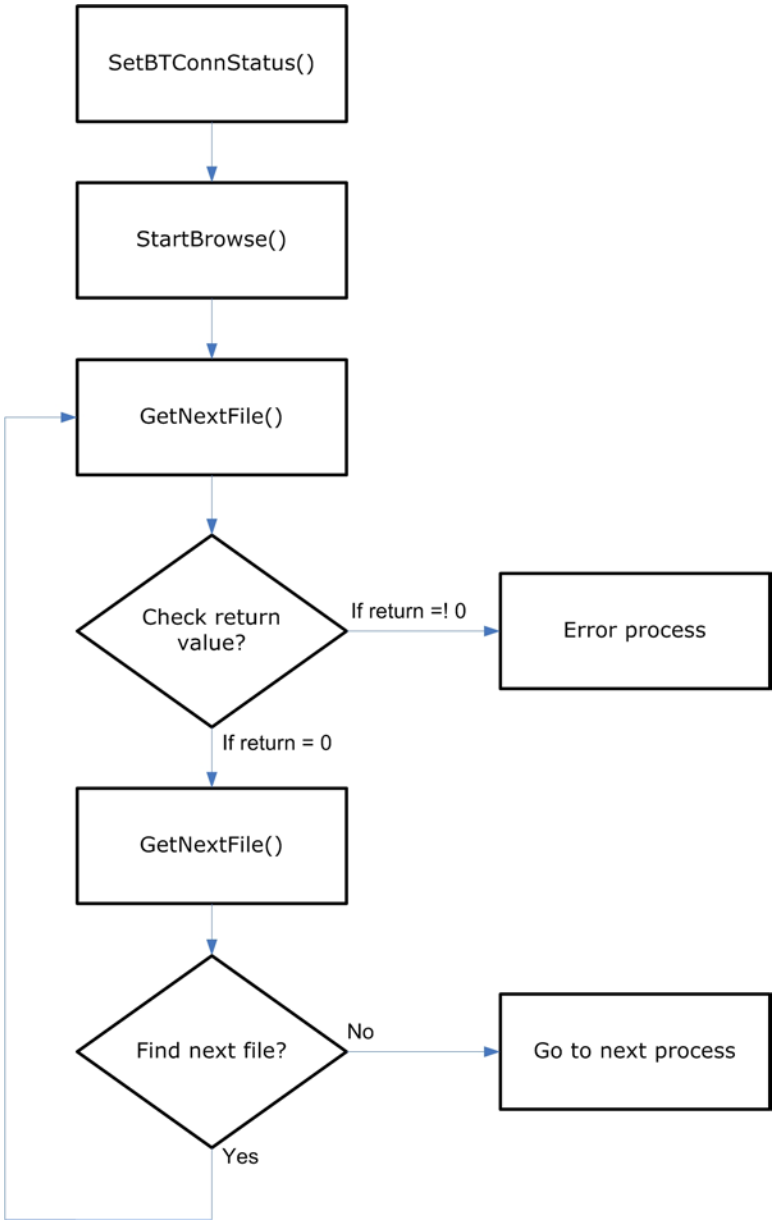
Otherwise, it returns a non-zero value to indicate a specific error condition. Refer to Error Code (SDCERR).

4.9. BLUETOOTH

Here is the programming flow for Bluetooth initialization.



After initialization, you may start with a desired Bluetooth service by calling SetBTConnStatus(). Below is the programming flow for using the File Transfer service (FTP).


SetBTConnStatus()
StartBrowse()
GetNextFile()
Check return value?
If return =! 0
Error process
If return = 0
GetNextFile()
Find next file?
No
Go to next process
Yes

4.9.1. GET/SET BLUETOOTH POWER

GetBTPower

Purpose	Gets current BT module power status.
Syntax	**DWORD GetBTPower (BYTE** **onOff***);**
Parameters	*onOff* [out] Pointer to a buffer where the information stores.

0	BT module off
1	BT_module on

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	DeviceIOControl Fail
2	CreateFile Error

SetBTPower

Purpose	Sets BT module power state.
Syntax	**DWORD SetBTPower (BYTE** *onOff***);**
Parameters	*onOff* [in] Power state to set.

0	Turns off BT module power
1	Turns on BT module power

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to set power

4.9.2. START/STOP BLUETOOTH

StartBluetooth

Purpose: To enable the access of Bluetooth services.

Syntax: **DWORD StartBluetooth (VOID);**

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

32	ERROR_OPERATION_FAIL
33	ERROR_MACHINE_NOT_SUPPORTED

StopBluetooth

Purpose: To disable the access of Bluetooth services.

Syntax: **DWORD StopBluetooth (VOID);**

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

32	ERROR_OPERATION_FAIL

4.9.3. FIND DEVICES

StartFindingDevice

Purpose: To initialize the search procedure.

Syntax: **DWORD StartFindingDevice (VOID);**

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

32	ERROR_OPERATION_FAIL

FindNextBTDevice

Purpose: To search Bluetooth devices after StartFindingDevice() is called.

Syntax: **DWORD FindNextBTDevice (INT** **deviceInfo****);**

Parameters: *deviceInfo*
[out] Pointer to a variable that stores the device information.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

30	ERROR_NO_STARTFINDING
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

4.9.4. PAIR DEVICES

StartPairing

Purpose: To pair with a spcified Bluetooth device.

Syntax:
DWORD StartPairing (INT *deviceInfo*,
CHAR **pinCode*,
INT *pinCodeLen*);

Parameters:
deviceInfo
[in] A value returned by FindNextBTDevice().
pinCode
[in] Pointer to a buffer where the PIN code is stored.
pinCodeLen
[in] A value that specifies the PIN code length.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER
21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

See Also: FindNextBTDevice

StartUnpairing

Purpose: To unpair with a spcified Bluetooth device.

Syntax: **DWORD StartUnpairing (INT** *deviceInfo*);

Parameters:
deviceInfo
[in] A value returned by FindNextBTDevice().

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

4.9.5. AVAILABLE SERVICES & DEVICE INFORMATION

FindService

Purpose: To find out Bluetooth services available on the connected device.

Syntax:
DWORD FindService (INT *targetService*,
INT *deviceInfo*,
INT **serviceInfo*);

Parameters:

targetService
[in] A value that specifies which Bluetooth service is inquired.

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM 0 (client)
3	BT_SPP_SERVICE	COM 0 (client)
4	BT_OPP_SERVICE	
5	(Reserved)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

deviceInfo
[in] A value returned by FindNextBTDevice().
serviceInfo
[out] Pointer to a variable that stores the information.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

See Also: GetServiceType

GetDeviceAddress

Purpose	To get the MAC address of the connected device.
Syntax	**DWORD GetDeviceAddress (INT** *deviceInfo*, **ULONGLONG** **btAddr*);
Parameters	*deviceInfo* [in] A value returned by FindNextBTDevice(). *btAddr* [out] Pointer to a buffer where the MAC address is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
31	ERROR_WRONG_ARG

See Also FindNextBTDevice

GetDeviceName

Purpose	To get the name of the connected device.
Syntax	**DWORD GetDeviceName (INT** *deviceInfo*, **CHAR** **buf*, **INT** *bufSize*);
Parameters	*deviceInfo* [in] A value returned by FindNextBTDevice(). *buf* [out] Pointer to a buffer where the device name is stored. *bufSize* [in] The size of the buffer, in characters.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
31	ERROR_WRONG_ARG

See Also FindNextBTDevice

GetServiceType	
Purpose	To get the Bluetooth service that is currently in use.
Syntax	**DWORD GetServiceType (INT** *serviceInfo*, **INT** **serviceType*);
Parameters	*serviceInfo* [in] A value returned by FindService(). *serviceType* [out] Pointer to a variable that stores the information.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
31	ERROR_WRONG_ARG

See Also FindService

4.9.6. BLUETOOTH CONNECTION & STATUS

ConnectByMacAddr

Purpose: To establish Bluetooth connection by MAC address and service type

Syntax:

```
DWORD ConnectByMacAddr (ULONGLONG deviceMacAddr,
                        INT targetService,
                        CHAR *pinCode,
                        INT sppComPort,
                        INT *deviceInfo,
                        INT *serviceInfo);
```

Parameters

deviceMacAddr
[in] A value that specifies the MAC address of the target device.
targetService
[in] A value that specifies which Bluetooth service to connect.

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM 0 (client)
3	BT_SPP_SERVICE	COM 0 (client)
4	(Reserved)	
5	(Reserved)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

pinCode
[in] Pointer to a buffer where the PIN code is stored.
sppComPort
[in] A value that specifies the serial port used for connecting SPP service.
deviceInfo
[out] Pointer to a variable that stores the information.
serviceInfo
[out] Pointer to a variable that stores the information.

Return Value

If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER
21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
27	ERROR_WRONG_SERVICE_TYPE
32	ERROR_OPERATION_FAIL

GetBTConnStatus

Purpose	To find out the status of the current Bluetooth connection.
Syntax	**DWORD GetBTConnStatus (INT** *serviceInfo*, **INT** **connStatus*);
Parameters	*serviceInfo* [in] A value returned by FindService(). *connStatus* [out] Pointer to a variable that stores the information.

0	Disconnect with the target Bluetooth device
1	Connect with the target Bluetooth device

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER
22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ

See Also: FindService

SetBTConnStatus

Purpose

To set Bluetooth connection.

Syntax

DWORD SetBTConnStatus (INT *serviceType*,
INT *onOff*,
INT *deviceInfo*,
INT *serviceInfo*);

Parameters

serviceType

[in] A value that specifies which Bluetooth service is desired.

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM 0 (client)
3	BT_SPP_SERVICE	COM 0 (client)
4	(Reserved)	
5	(Reserved)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

connStatus

[in] A value that specifies whether to establish a connection using the specified Bluetooth service.

0	Disconnect with the target Bluetooth device
1	Connect with the target Bluetooth device

deviceInfo

[in] A value returned by FindNextBTDevice().

serviceInfo

[in] A value returned by FindService().

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
28	ERROR_DEVICE_NOT_HAS_SERVICE
32	ERROR_OPERATION_FAIL

See Also

FindNextBTDevice, FindService

4.9.7. FTP SERVICE – VIEW FILES

StartBrowse

Purpose: To initialize the browse procedure.

Syntax:
DWORD StartBrowse (INT *serviceInfo*,
CHAR **targetPath*);

Parameters:
serviceInfo
[in] A value returned by FindService().
targetPath
[in] Pointer to a buffer where the directory path is stored.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
32	ERROR_OPERATION_FAIL

See Also: FindService

GetNextFile

Purpose: To view files after StartBrowse() is called.

Syntax:
DWORD GetNextFile (INT *serviceInfo*,
FTP_FILE_INFO **fileInfo*);

Parameters:
serviceInfo
[in] A value returned by FindService().
fileInfo
[out] Pointer to the structure FTP_FILE_INFO where the information is stored.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

See Also: FindService

4.9.8. FTP SERVICE – FILE TRANSFER

GetFile

Purpose	To download a file from the FTP share folder.
Syntax	**DWORD GetFile (INT** *serviceInfo*, **FTP_FILE_INFO** **fileInfo*, **HWND** *progressBar*);
Parameters	*serviceInfo* [in] A value returned by FindService(). *fileInfo* [out] Pointer to the structure FTP_FILE_INFO where the information is stored. *progressBar* [in] A Windows handle of a progress bar that is used to display the progress of uploading a file. You may pass NULL to this argument.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

See Also: FindService, GetNextFile, StartBrowse

PutFile

Purpose

To upload a file to the FTP share folder.

Syntax

DWORD PutFile (INT *serviceInfo*,
CHAR **filePath*,
HWND *progressBar*);

Parameters

serviceInfo

[in] A value returned by FindService().

filePath

[in] Pointer to a buffer where the file path is stored.

progressBar

[in] A Windows handle of a progress bar that is used to display the progress of uploading a file. You may pass NULL to this argument.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

See Also

FindService, GetNextFile, StartBrowse

4.9.9. SPP COM PORT

GetSppComPort

Purpose	To find out the serial port used for connecting SPP service.
Syntax	**DWORD GetSppComPort (INT** *serviceInfo*, **INT** **comPort*);
Parameters	*serviceInfo* [in] A value returned by FindService(). *comPort* [in] Pointer to a buffer where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

4	ERROR_PARAMETER
22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE

SetSppComPort

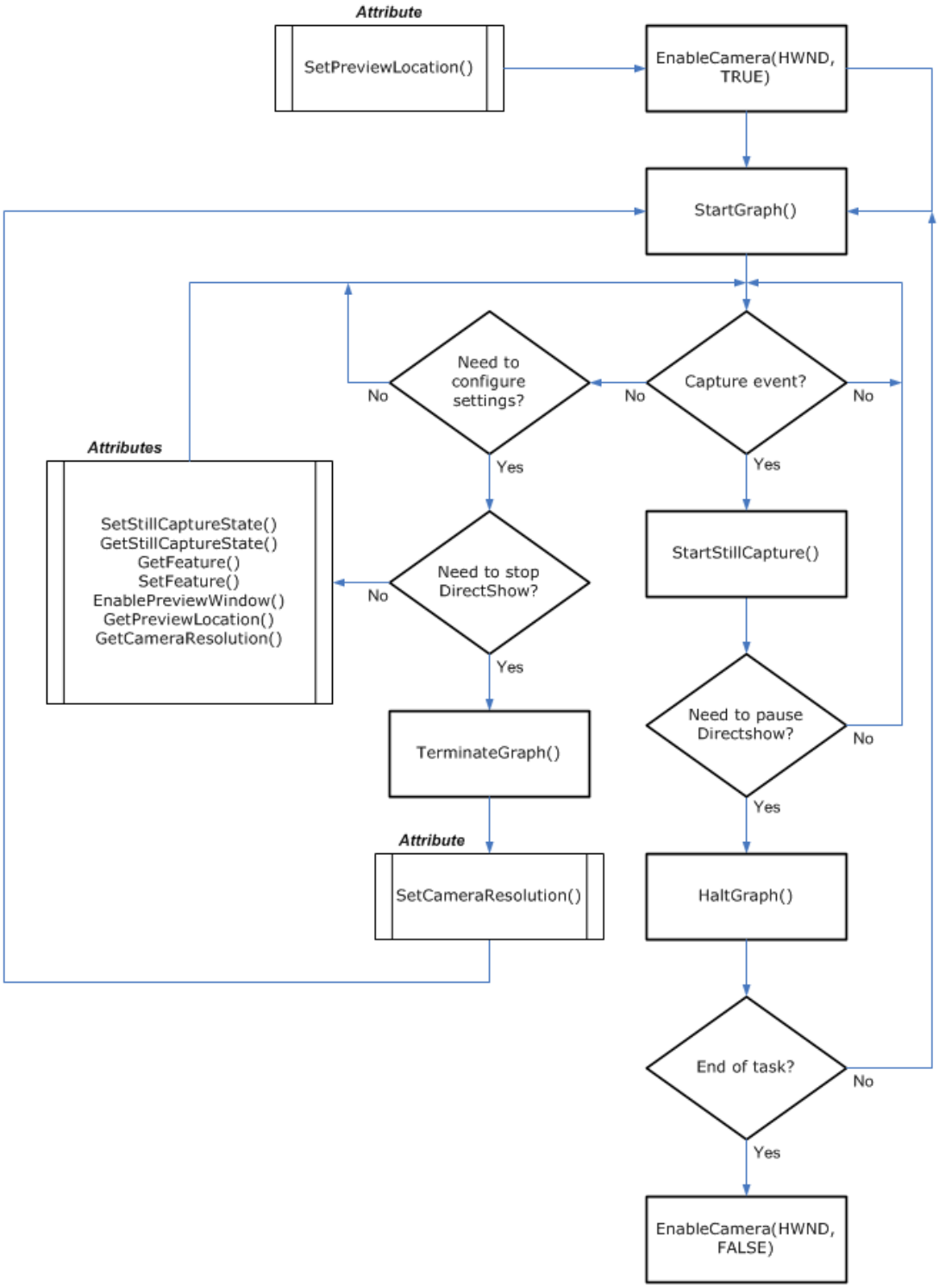
Purpose	To set the serial port used for connecting SPP service.
Syntax	**DWORD SetSppComPort (INT** *serviceInfo*, **INT** *comPort*);
Parameters	*serviceInfo* [in] A value returned by FindService(). *comPort* [in] Always uses COM 0 (Client).
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE

Remarks	By default, only COM 0 is the serial port used for connecting SPP service. This function must be called before SetBTConnStatus().

4.10. CAMERA

Here is the programming flow for camera operation.



4.10.1. CAMERA POWER

EnableCamera

Purpose	To turn on the camera or turn it off.
Syntax	**DWORD EnableCamera (HWND** *wnd*, **BOOL** *onOff*);
Parameters	*wnd* [in] HWND variable *onOff* [in] BOOL variable

0	Disable (Turn off the camera.)
1	Enable (Turn on the camera.)

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

See Also	EnablePreviewWindow, SetPreviewLocation

4.10.2. CAMERA SETTINGS

GetCameraResolution

Purpose	To get the camera resolution.
Syntax	**DWORD GetCameraResolution (DWORD** **resValue***);**
Parameters	*resValue* [out] Pointer to a buffer where the information is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetCameraResolution

Purpose	To set the camera resolution.
Syntax	**DWORD SetCameraResolution (DWORD** *resValue***);**
Parameters	*resValue* [in] 32-bit unsigned integer variable

0	640 x 480 pixels
1	1280 x 960 pixels
2	1600 x 1200 pixels
3	2048 x 1536 pixels

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks	TerminateGraph() must be called before this function. The new resolution will not take effect until StartGraph() is called again.
See Also	StartGraph, TerminateGraph

GetFlashLight	
Purpose	Gets camera flashlight setting.
Syntax	**DWORD GetFlashLight (FLASH** **flashLight***);**
Parameters	*flashLight* [out] Pointer to structure FLASH where flashlight setting is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetFlashLight	
Purpose	Sets if camera beams a light for the picture to take.
Syntax	**DWORD SetFlashLight (FLASH** **flashLight***);**
Parameters	*flashLight* [in] Pointer to structure FLASH where flashlight setting is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

See Also	EnableCamera, StartStillCapture

4.10.3. PREVIEW

EnablePreviewWindow

Purpose: To show or hide the preview window.

Syntax: **DWORD EnablePreviewWindow (BOOL** *onOff***);**

Parameters: *onOff*
[in] BOOL variable

0	Disable (= No preview)
1	Enable (= Display a preview image)

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)

See Also: SetPreviewLocation

GetPreviewLocation

Purpose: To get the position of the preview window.

Syntax: **DWORD GetPreviewLocation (INT** **posX*,
INT **posY*);

Parameters: *posX* Reserved
[out] Integer variable
posY
[out] Integer variable

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetPreviewLocation

Purpose: To set the position of the preview window.

Syntax: **VOID SetPreviewLocation (INT** *posX*,
INT *posY*);

Parameters: *posX* Reserved
[in] Integer variable
posY
[in] Integer variable

Remarks: The size of the preview window is 240 x 180 pixels.

See Also: EnablePreviewWindow

GetFeature

Purpose	To get the special effect setting.
Syntax	**DWORD GetFeature (DWORD** **level***);**
Parameters	*level* [out] 32-bit unsigned integer variable
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetFeature

Purpose	To apply special effect to the preview image.
Syntax	**DWORD SetFeature (DWORD** *level***);**
Parameters	*level* [in] 32-bit unsigned integer variable

0	Mono
1	Normal*
2	Negative
3	Sepia
4	Posterize
5	Whiteboard
6	Blackboard
7	Aqua

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

StartGraph

Purpose	To run DirectShow filters and start a graph in the preview window.
Syntax	**DWORD StartGraph ();**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)

See Also	HaltGraph, TerminateGraph

TerminateGraph	
Purpose	To stop running DirectShow filters and flush the preview window.
Syntax	**DWORD TerminateGraph ();**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:
	1 ERROR_NORESOURCE (Fail to get resource)
See Also	HaltGraph, StartGraph

HaltGraph	
Purpose	To pause a graph in the preview window until StartGraph() is called again.
Syntax	**DWORD HaltGraph ();**
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:
	1 ERROR_NORESOURCE (Fail to get resource)
See Also	StartGraph, TerminateGraph

4.10.4. STILL CAPTURE

GetStillCaptureState

Purpose	To get the capture settings.
Syntax	**DWORD GetStillCaptureState (PICSTATE** **picInfo***);**
Parameters	*picInfo* [out] Pointer to the structure PICSTATE where the capture settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetStillCaptureState

Purpose	To set the capture settings.
Syntax	**DWORD SetStillCaptureState (PICSTATE** **picInfo***);**
Parameters	*picInfo* [in] Pointer to the structure PICSTATE where the capture settings are stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks	This function must be called before StartStillCapture().
See Also	StartStillCapture

StartStillCapture

Purpose	To start with still image capture.
Syntax	**DWORD StartStillCapture (INT** *playSound***);**
Parameters	*playSound* [in] Integer variable

0	No notification.
1	Play a sound to notify the image has been captured.

Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

Remarks	Upon completion of the capture, it will send a Windows message WM_CAMERACAPTUREFINISH.
See Also	SetStillCaptureState

4.10.5. TORCH MODE

GetTorchMode

Purpose	Gets current torch mode.
Syntax	**DWORD GetTorchMode (DWORD ***dwMode**);**
Parameters	*dwMode* [out] Pointer to a buffer where the information is stored.

0	Disables
1	Enables

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

SetTorchMode

Purpose	To enable or disable torch light.
Syntax	**DWORD SetTorchMode (DWORD** *dwMode***);**
Parameters	*dwMode* [in] 32-bit unsigned integer variable

0	Disables
1	Enables

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	ERROR_NORESOURCE (Fail to get resource)
4	ERROR_PARAMETER (Wrong parameter)

4.11. GPS

The serial interface protocol is based on the national Marine Electronics Association's NMEA 0183 ASCII interface specification. For details, please refer to NMEA 0183 Version 3.01, which may be obtained from NMEA, www.nmea.org.

The table below lists the standard characteristics of the NMEA 0183 data transmissions:

Serial Configuration	NMEA Standard
Baud Rate	4800
Data Bits	8
Parity	None
Stop bits	1

The GPS receiver hosted on the mobile computer uses COM 7 to output NMEA messages, as long as the COM port is opened. You may use Microsoft's standard API to open the serial port and receive NMEA format data. The following URLs are documentations related to serial port programming techniques:

http://msdn.microsoft.com/en-us/library/bb202722.aspx
http://msdn.microsoft.com/en-us/library/ms810467.aspx
http://msdn.microsoft.com/en-us/library/bb201943.aspx

4.12. BUTTON ASSIGNMENT

GetButtonAssignment

Purpose: To find out the assignment of a user-definable key.

Syntax: **DWORD GetButtonAssignment (INT** *buttonID*,
INT **keyCode*,
TCHAR **buffer*);

buttonID
[in] Integer variable

1	Side-Trigger-Left	(default: Scan)
2	Side-Trigger-Right	(default: Scan)
3	(Reserved)	(Reserved)
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	(Reserved)	(Reserved)
7	Send	(default: Send)
8	End	(default: End)
9	(Reserved)	(Reserved)
10	(Reserved)	(Reserved)
11	Up	(default: Up)
12	Down	(default: Down)
13	Left	(default: Left)
14	Right	(default: Right)
15	ESC	(default: Esc)
16	TAB	(default: Tab)
17	Backspace	(default: Backspace)
18	Volume Up	(default: Volume Up)
19	Volume Down	(default: Volume Down)
20	Application (programming key)	(default:VK_OEM_KEY_1)
21	*Shift	(default: Shift)
	This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	
22	(Reserved)	(Reserved)
23	Enter	(default: Enter)
24	*Alpha	(default: Alpha)
	This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	

Parameters

25	*Fn	(Default: Fn)
	This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	
26	(Reserved)	(Reserved)
27	(Reserved)	(Reserved)
28	-	(default: -)
29	.	(default: .)
107	(Reserved)	(Reserved)
108	(Reserved)	(Reserved)
109	(Reserved)	(Reserved)
110	(Reserved)	(Reserved)
111	(Reserved)	(Reserved)
112	(Reserved)	(Reserved)
113	(Reserved)	(Reserved)
114	(Reserved)	(Reserved)
115	(Reserved)	(Reserved)
116	(Reserved)	(Reserved)
117	(Reserved)	(Reserved)

keyCode

[out] Pointer to a buffer where the key assignment information is stored.

Buffer

[out] Pointer to a buffer where user-defined key code or the full file path of the program is stored.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to open registry
2	Fail to set registry
4	ERROR_PARAMETER (Wrong parameter)

SetButtonAssignment

Purpose

Assigns a user-definable key to act as another key or serve as a shortcut key for launching a specific program.

Syntax

DWORD SetButtonAssignment (INT *buttonID*,
INT *keyCode*,
TCHAR **buffer*);

Parameters

buttonID

[in] Integer variable

1	Side-Trigger-Left	(default: Scan)
2	Side-Trigger-Right	(default: Scan)
3	(Reserved)	(Reserved)
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	(Reserved)	(Reserved)
7	Send	(default: Send)
8	End	(default: End)
9	(Reserved)	(Reserved)
10	(Reserved)	(Reserved)
11	Up	(default: Up)
12	Down	(default: Down)
13	Left	(default: Left)
14	Right	(default: Right)
15	ESC	(default: Esc)
16	TAB	(default: Tab)
17	Backspace	(default: Backspace)
18	Volume Up	(default: Volume Up)
19	Volume Down	(default: Volume Down)
20	Application (programming key)	(default:VK_OEM_KEY_1)
21	*Shift	(default: Shift)
	▸ This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	
22	(Reserved)	(Reserved)
23	Enter	(default: Enter)
24	*Alpha	(default: Alpha)
	▸ This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	
25	*Fn	(Default: Fn)
	▸ This is a special key code which is not assignable; lock/unlock available only. Unlocked after system reset.	

26	(Reserved)	(Reserved)
27	(Reserved)	(Reserved)
28	-	(default: -)
29	.	(default: .)
107	(Reserved)	(Reserved)
108	(Reserved)	(Reserved)
109	(Reserved)	(Reserved)
110	(Reserved)	(Reserved)
111	(Reserved)	(Reserved)
112	(Reserved)	(Reserved)
113	(Reserved)	(Reserved)
114	(Reserved)	(Reserved)
115	(Reserved)	(Reserved)
116	(Reserved)	(Reserved)
117	(Reserved)	(Reserved)

keyCode

[in] Integer variable

0	User-defined key code (0x01~0xFF)
1	Launch the specified program
2	Enter
3	Scan
4	Esc
5	Delete
6	Backspace
7	Space
8	Tab
9 **10** **...** **20**	F1 F2 ... F12
21	Start Menu
22	Alt
23	OEM_Key1 (0xE9)
24	OEM_Key2 (0xEA)
25	OEM_Key3 (0xEB)
26	OEM_Key4 (0xEC)
27	OEM_Key5 (0xED)

28	OEM_Key6 (0xEE)
29	OEM_Key7 (0xEF)
30	OEM_Key8 (0xF0)
31	OEM_Key9 (0xF1)
32	OEM_Key10 (0x2A)
33	*
34	#
35	Send(VK_TTALK)
36	End(VK_END)
37	(Reserved)
38	Up
39	Down
40	Left
41	Right
42	TAB
43	Volume Up
44	Volume Down
45	(Reserved)
46	(Reserved)
47	OK
48	(Reserved)
49	(Reserved)
50	Home(VK_HOME)
51	End(VK_END)
52	Fn+ESC(0xF5)
53 **54** **...** **64**	F13 F14 ... F24
65	Home(VK_LWIN+ VK_APP2)
66	(Reserved)
67	-
68	.
997	Lock key ▸ Locks keys; the locked key will have no function under any mode (Shift+key, Blue+key, Orange+key)
998	Unlock key

buffer

[in] Pointer to a buffer which stores the user-defined key code or the full path to the application to launch. Pass Null if this parameter is not used.

Return Value

If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	Fail to open registry
2	Fail to set registry
4	ERROR_PARAMETER

4.13. SIGNATURE CAPTURE

4.13.1. INITIALIZATION/EXIT

InitSigScreen

Purpose: To initialize and show the signature area.

Syntax:

```
INT InitSigScreen (INT x,
                   INT y,
                   INT width,
                   INT height,
                   HWND parentWnd);
```

Parameters:

x

[in] A value that specifies the X-coordinate of the start position for the area that allows signature capture.

y

[in] A value that specifies the Y-coordinate of the start position for the area that allows signature capture.

width

[in] A value that specifies the width of the area.

height

[in] A value that specifies the height of the area.

parentWnd

[in] A handle to the parent or owner window where the signature capture functionality is implemented.

Return Value: If successful, it returns 0.

Otherwise, it returns a non-zero value to indicate a specific error:

1	SIGERROR_UNKNOWN
2	SIGERROR_MACHINE_NOT_SUPPORTED

Remarks: This function must be called before any other signature-related functions. The signature area is enabled upon completion of initialization.

CloseSigScreen

Purpose: To exit the signature area.

Syntax:

```
VOID CloseSigScreen (VOID);
```

Remarks: The signature-related functions will not take effect until InitSigScreen() is called again.

4.13.2. CONTROL OF SIGNATURE AREA

ClearSigArea

Purpose: To clear the entire signature.

Syntax: **VOID ClearSigArea (VOID);**

EnableSigScreen

Purpose: To enable or disable the signature area.

Syntax: **VOID EnableSigScreen (INT** *enabled*);

Parameters: *enabled*

[in] Integer variable

0	Disable
1	Enable

ShowSigScreen

Purpose: To show or hide the signature area.

Syntax: **VOID ShowSigScreen (INT** *showArea*);

Parameters: *showArea*

[in] Integer variable

0	Hide
1	Show

4.13.3. BACKGROUND COLOR

SetSigBgColor

Purpose: To set the background color of the signature area.

Syntax: **VOID SetSigBgColor (INT** *redOfRGB*,
INT greenOfRGB,
INT blueOfRGB);

Parameters: *redOfRGB*

[in] Integer variable

0~255	255*

greenOfRGB

[in] Integer variable

0~255	255*

blueOfRGB

[in] Integer variable

0~255	255*

4.13.4. PEN COLOR AND WIDTH

SetSigLineColor

Purpose: To set the pen color.

Syntax:
```
VOID SetSigLineColor (INT redOfRGB,
                      INT greenOfRGB,
                      INT blueOfRGB);
```

Parameters:

redOfRGB
[in] Integer variable

0~255	0*

greenOfRGB
[in] Integer variable

0~255	0*

blueOfRGB
[in] Integer variable

0~255	0*

SetSigLineWidth

Purpose: To set the pen width, in pixels.

Syntax:
```
VOID SetSigLineWidth (INT width);
```

Parameters:

width
[in] Integer variable

1~5	1*

4.13.5. IMAGE SAVING & LOADING

LoadSigImage

Purpose	To load the signature image.
Syntax	**INT LoadSigImage (WCHAR** **filePath***);**
Parameters	*filePath* [in] Pointer to a buffer where the full file path of the image is stored.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

1	SIGERROR_UNKNOWN
3	SIGERROR_SIG_WINDOW_NOT_CREATED
4	SIGERROR_FILE_NOT_FOUND
6	SIGERROR_HIDDEN_WINDOW

Remarks	We suggest not calling this function while initializing the signature area. Also, it cannot load an image with a resolution greater than 640 x 480 pixels.

SaveSigImage

Purpose	To save the signature image at a resolution no greater than 640 x 480 pixels.
Syntax	**INT SaveSigImage (INT** *imageFormat***,** **WCHAR** **filePath***);**
Parameters	*imageFormat* [in] Integer variable

0	BMP_FORMAT	To save in BMP file format.
1	JPG_FORMAT	To save in JPEG file format.
2	LOCUS_FORMAT	To save in LOCUS file format.

filePath
[in] Pointer to a buffer where the full file path of the image is stored.

Return Value: If successful, it returns 0.
Otherwise, it returns a non-zero value to indicate a specific error:

1	SIGERROR_UNKNOWN
3	SIGERROR_SIG_WINDOW_NOT_CREATED
6	SIGERROR_HIDDEN_WINDOW

4.13.6. SIGNATURE DLL VERSION

GetSigDllVer	
Purpose	To obtain the current Signature DLL version.
Syntax	**INT GetSigDllVer (CHAR** **buf*, **INT** bufSize**);**
Parameters	*buf* [out] Pointer to a buffer where the version information is stored. *bufSize* [in] The size of the buffer, in characters.
Return Value	If successful, it returns 0. Otherwise, it returns a non-zero value to indicate a specific error:

5	SIGERROR_ILLEGAL_PARAMETER

Chapter 5

DATA STRUCTURE

IN THIS CHAPTER

5.1 System Information Structure 212
5.2 Backlight Structure 214
5.3 Keypad Structure 216
5.4 Wi-Fi Structures 217
5.5 Bluetooth Structure 219
5.6 Camera Structures 241

5.1. SYSTEM INFORMATION STRUCTURE

5.1.1. SYSINFO

This structure stores the system information, which is identical to the settings displayed on the mobile computer under **Start Screen** | **Settings** | **System** | **Device Information**.

```
typedef struct _SYSINFO
{
TCHAR        DevDesc[16];
TCHAR        DevCFG[16];
TCHAR        SerialNum[16];
TCHAR        SerialNumReadOnly[16];
TCHAR        SerialNumPCBA[16];
TCHAR        Manufactory[32];
TCHAR        ManuDate[16];
UCHAR        OsVer[32];
UCHAR        BootloaderVer[32];
TCHAR        MicroPVer[32];
UCHAR        BTAddress[32];
UCHAR        WiFiAddress[32];
CHAR         IMEI[32];
} SYSINFO, *PSYSINFO;
```

Data Type	Member Name	Description
TCHAR	DevDesc[16]	Device Model
TCHAR	DevCFG[16]	(Reserved)
TCHAR	SerialNum[16]	Serial Number
TCHAR	SerialNumReadOnly[16]	Serial number for factory use[Note]
TCHAR	SerialNumPCBA[16]	Serial number of PCBA[Note] — always 16 characters "XXXXXXXXXXXXXXX"
TCHAR	Manufactory[32]	Manufacturer
TCHAR	ManuDate[16]	Manufacturer Date
UCHAR	OsVer[32]	OS Version
UCHAR	BootloaderVer[32]	Bootloader version
TCHAR	MicroPVer[32]	MicroP firmware version
UCHAR	BTAddress[32]	Bluetooth MAC address
UCHAR	WiFiAddress[32]	Wi-Fi MAC address
TCHAR	IMEI[32]	Unique GSM number

Note: (1) By default, SerialNum and SerialNumReadOnly are the same.
(2) Information on SerialNumReadOnly and SerialNumPCBA can be accessed by calling GetSysInfo() only.

5.2. BACKLIGHT STRUCTURE

5.2.1. BKLCTL

This structure stores information about the backlight control, which is identical to the settings displayed on the mobile computer under **Start screen** | **Settings** | **System** | **Backlight.**

Battery mode - Battery power in use

AC mode - The mobile computer is connected to an AC power source via the cradle.

```
typedef struct _BKLCTL
{
DWORD        batttimeout;
DWORD        lightlevel;
DWORD        actimeout;
DWORD        aclightlevel;
DWORD        backlightontap;
DWORD        acbacklightontap;
DWORD        usebattery;
DWORD        useext;
} BKLCTL, *PBKLCTL;
```

Data Type	Member Name	Description
DWORD	batttimeout	In battery mode, turn off backlight after a specified period of idle time – 15, 30, 60, 120, 300 (in units of second). ▸ "usebattery" must be enabled!
DWORD	lightlevel	Backlight level in battery mode – 0~11 from dark to bright
DWORD	actimeout	In AC mode, turn off backlight after a specified period of idle time – 15, 30, 60, 120, 300 (in units of second). ▸ "useext" must be enabled!
DWORD	aclightlevel	Backlight level in AC mode – 0~11 from dark to bright
DWORD	backlightontap	When backlight is off in battery mode, it will be automatically turned on when you tap the touch screen or press a key – 0 Disable backlight on tap in battery mode 1 Enable backlight on tap in battery mode
DWORD	acbacklightontap	When backlight is off in AC mode, it will be automatically turned on when you tap the touch screen or press a key – 0 Disable backlight on tap in AC mode

<table>
<tr><td></td><td></td><td><table><tr><td>1</td><td>Enable backlight on tap in AC mode</td></tr></table></td></tr>
<tr><td>DWORD</td><td>usebattery</td><td>Power-saving mode for backlight when using battery power –<table><tr><td>0</td><td>Disable power-saving in battery mode</td></tr><tr><td>1</td><td>Enable power-saving in battery mode</td></tr></table>
When enabled, proceed to define "batttimeout" and "backlightontap".</td></tr>
<tr><td>DWORD</td><td>useext</td><td>Power-saving mode for backlight when using external power –<table><tr><td>0</td><td>Disable power-saving in AC mode</td></tr><tr><td>1</td><td>Enable power-saving in AC mode</td></tr></table>
When enabled, proceed to define "actimeout" and "acbacklightontap".</td></tr>
</table>

5.3. KEYPAD STRUCTURE

5.3.1. KEYPADBIND

This structure stores information about the application to launch upon system initialization.

```
typedef struct

typedef struct
{
TCHAR        szClass[64];
} KEYPADBIND, *PKEYPADBIND;
```

Data Type	Member Name	Description
TCHAR	szClass[64]	Name of the application to launch.

5.4. WI-FI STRUCTURES

5.4.1. RAW_DATA

This structure stores information about WLAN networks.

```
typedef struct
{
DWORD        dwDataLen;
BYTE*        pData;
} RAW_DATA, *PRAW_DATA;
```

Data Type	Member Name	Description
DWORD	dwDataLen	The length of pData.
BYTE*	pData	Refer to the structure PWZC_802_11_CONFIG_LIST defined in wzcsapi.h.

5.4.2. WLANADPTINFO

Structure WLANADPTINFO stores the information about the mobile computer's Wi-Fi networking which is viewable at **Start screen** | **Settings** | **Connections** | **Wi-Fi** | **Network Adapters**.

```
typedef struct _WLANADPTINFO
{
DWORD        fUseDHCP;
DWORD        IPAddr;
DWORD        SubnetMask;
DWORD        Gateway;
DWORD        DNSAddr;
DWORD        DNSAltAddr;
DWORD        WINSAddr;
DWORD        WINSAltAddr;
} WLANADPTINFO, *PWLANADPTINFO;
```

Data Type	Member Name	Description
DWORD	fUseDHCP	Whether to use DHCP server – 0 Disables DHCP 1 Enables DHCP
DWORD	IPAddr	IP address of the mobile computer
DWORD	SubnetMask	IP address of Subnet Mask
DWORD	Gateway	IP address of default Gateway
DWORD	DNSAddr	DNS server address – Primary DNS
DWORD	DNSAltAddr	DNS server address – Secondary DNS
DWORD	WINSAddr	WINS server address– Primary WINS
DWORD	WINSAltAddr	WINS server address– Secondary WINS

Note: When DHCP is enabled, the other IP-related information is set to 0.

5.4.3. WLANCONNECTEDST

This structure stores information about wireless connection status, which is identical to the information displayed on the Wireless Manager settings page.

```
typedef struct _WLANCONNECTEDST
      {
CHAR                SSID[32];
DWORD               SSIDLength;
BYTE                BSSID[6];
} WLANCONNECTEDST, *PWLANCONNECTEDST;
```

Data Type	Member Name	Description
CHAR	SSID[32]	Service Set ID of the access point, maximum 32 characters
DWORD	SSIDLength	Length of SSID, including spaces
BYTE	BSSID[6]	MAC address of the access point connected

5.4.4. WLANCTL

This structure stores information about new wireless networks, which is identical to the information added by tapping **Start Screen** | **Settings** | **Connections** | **Wi-Fi** | **Wireless** tab | **Add New…** button.

```
typedef struct _WLANCTL
      {
CHAR                SSID[32];
DWORD               authentication;
DWORD               encryption;
DWORD               adhoc;
DWORD               eap;
WCHAR               key[80];
} WLANCTL, *PWLANCTL;
```

Data Type	Member Name	Description
CHAR	SSID[32]	Maximum 32 characters
DWORD	authentication	Authentication in use – 0 – Open (Ndis802_11AuthModeOptn) 1 – Share key (Ndis802_11AuthModeShared) 3 – WPA (Ndis802_11AuthModeWPA) 4 – WPA-PSK (Ndis802_11AuthModeWPAPSK) 5 – WPA-NONE (Ndis802_11AuthModeWPANone) 6 – WPA2 (Ndis802_11AuthModeWPA2) 7 – WPA2-PSK (Ndis802_11AuthModeWPA2PSK)
DWORD	encryption	Encryption in use – 0 – WEP (Ndis802_11WEPEnabled) 1 – None (Ndis802_11WEPDisabled) 4 – TKIP (Ndis802_11Encryption2Enabled) 6 – AES (Ndis802_11Encryption3Enabled)
DWORD	adhoc	Network type – 0 – infrastructure 1 – ad hoc network
DWORD	eap	EAP type – 13 – TLS

<table>
<tr><td></td><td></td><td><table><tr><td>25</td><td>PEAP</td></tr></table></td></tr>
<tr><td>WCHAR</td><td>key[80]</td><td>WEP key in any of the following formats –<table>
<tr><td>1/0x1234567890</td><td>[index=1,40-bit(10-digit hexa)]</td></tr>
<tr><td>4/zxcvb</td><td>[index=4, 40-bit(5-char)]</td></tr>
<tr><td>3/0x12345678901234567890123456</td><td>[index=3, 104-bit(26-digit hexa)]</td></tr>
<tr><td>2/abcdefghij123</td><td>[index=2, 104-bit(13-char)]</td></tr>
<tr><td>auto</td><td>[key comes from successful EAP]</td></tr>
</table></td></tr>
</table>

5.4.5. CF10G_STATUS

```
typedef struct _CF10G_STATUS
{
    CARDSTATE       cardState;
    CHAR            configName[CONFIG_NAME_SZ];
    UCHAR           client_MAC[6];
    UCHAR           client_IP[4];
    CHAR            clientName[CLIENT_NAME_SZ];
    UCHAR           AP_MAC[6];
    UCHAR           AP_IP[4];
    CHAR            APName[CLIENT_NAME_SZ];
    EAPTYPE         eapType;
    DWORD           channel;
    INT             rssi;
    BITRATE         bitRate;
    INT             txPower;
    DWORD           driverVersion;
    RADIOTYPE       radioType;
    DWORD           DTIM;
    DWORD           beaconPeriod;
    DWORD           beaconsReceived;
} CF10G_STATUS;
```

<table>
<tr><th>Data Type</th><th>Member Name</th><th>Description</th></tr>
<tr><td>CARDSTATE</td><td>cardState</td><td>The association status —<table><tr><td>0</td><td>CARDSTATE_NOT_INSERTED</td></tr><tr><td>1</td><td>CARDSTATE_NOT_ASSOCIATED</td></tr><tr><td>2</td><td>CARDSTATE_ASSOCIATED</td></tr><tr><td>3</td><td>CARDSTATE_AUTHENTICATED</td></tr><tr><td>4</td><td>CARDSTATE_FCCTEST</td></tr><tr><td>5</td><td>CARDSTATE_NOT_SDC</td></tr></table></td></tr>
<tr><td>CHAR</td><td>configName[CONFIG_NAME_SZ]</td><td>Name of active profile.
No more than 33 characters. See CONFIG_NAME_SZ defined in the header file.</td></tr>
<tr><td>UCHAR</td><td>client_MAC[6]</td><td>MAC address of Summit radio.</td></tr>
<tr><td>UCHAR</td><td>client_IP[4]</td><td>IP address of Summit radio.</td></tr>
</table>

<table>
<tr><td>CHAR</td><td>clientName[CLIENT_NAME_SZ]</td><td>Name of Summit radio
No more than 17 characters. See CLIENT_NAME_SZ defined in the header file.</td></tr>
<tr><td>UCHAR</td><td>AP_MAC[6]</td><td>MAC address of the access point</td></tr>
<tr><td>UCHAR</td><td>AP_IP[4]</td><td>IP address of the access point (= Null, if not supported by AP)</td></tr>
<tr><td>CHAR</td><td>APName[CLIENT_NAME_SZ]</td><td>Name of the access point (= Null, if not supported by AP)
No more than 17 characters. See CLIENT_NAME_SZ defined in the header file.</td></tr>
<tr><td>EAPTYPE</td><td>eapType</td><td>The EAP type in use —<table><tr><td>0</td><td>EAP_NONE</td></tr><tr><td>1</td><td>EAP_LEAP</td></tr><tr><td>2</td><td>EAP_EAPFAST</td></tr><tr><td>3</td><td>PEAPMSCHAP</td></tr><tr><td>4</td><td>PEAPGTC</td></tr><tr><td>5</td><td>EAPTLS</td></tr></table></td></tr>
<tr><td>DWORD</td><td>channel</td><td>Channel in use — information on WLAN connection between Summit radio and AP.</td></tr>
<tr><td>INT</td><td>rssi</td><td>Signal strength — information on WLAN connection between Summit radio and AP.</td></tr>
<tr><td>BITRATE</td><td>bitRate</td><td>Data rate — information on WLAN connection between Summit radio and AP.<table><tr><td>0</td><td>BITRATE_AUTO</td></tr><tr><td>2</td><td>BITRATE_1</td></tr><tr><td>4</td><td>BITRATE_2</td></tr><tr><td>11</td><td>BITRATE_5_5</td></tr><tr><td>12</td><td>BITRATE_6</td></tr><tr><td>18</td><td>BITRATE_9</td></tr><tr><td>22</td><td>BITRATE_11</td></tr><tr><td>24</td><td>BITRATE_12</td></tr><tr><td>36</td><td>BITRATE_18</td></tr><tr><td>48</td><td>BITRATE_24</td></tr><tr><td>72</td><td>BITRATE_36</td></tr><tr><td>96</td><td>BITRATE_48</td></tr><tr><td>108</td><td>BITRATE_54</td></tr></table></td></tr>
<tr><td>INT</td><td>txPower</td><td>Transmit power — information on WLAN connection between Summit radio and AP.<table><tr><td>0</td><td>TXPOWER_MAX</td></tr><tr><td>1</td><td>TXPOWER_1</td></tr></table></td></tr>
</table>

		5 TXPOWER_5 10 TXPOWER_10 20 TXPOWER_20 30 TXPOWER_30 50 TXPOWER_50
DWORD	driverVersion	Driver version
RADIOTYPE	radioType	The radio type in use — 0 RADIOTYPE_BG 1 RADIOTYPE_ABG 100 RADIOTYPE_NOT_SDC 101 RADIOTYPE_NOT_SDC_1
DWORD	DTIM	A multiple of the beacon period that specifies how often the beacon contains a delivery traffic indication message (DTIM), which tells power-save client devices that a packet is waiting for them (e.g. a DTIM interval of 3 means that every third beacon contains a DTIM).
DWORD	beaconPeriod	Amount of time between access point beacons in kilomicroseconds, where one Kμsec equals 1024 microseconds.
DWORD	beaconsReceived	Beacons received.

5.4.6. CONFIG_FILE_INFO

```
typedef struct _CONFIG_FILE_INFO
{
    DWORD            numConfigs;
    BOOLEAN          globalConfigPresent;
    BOOLEAN          thirdPartyConfigPresent;
    DWORD            sdkVersion;
} CONFIG_FILE_INFO;
```

<table>
<tr><th>Data Type</th><th>Member Name</th><th>Description</th></tr>
<tr><td>DWORD</td><td>numConfigs</td><td>Number of profiles
No more than 20. See MAX_CFGS defined in the header file.</td></tr>
<tr><td>BOOLEAN</td><td>globalConfigPresent</td><td>Whether global configuration is present —<table><tr><td>0</td><td>No</td></tr><tr><td>1</td><td>Yes</td></tr></table></td></tr>
<tr><td>BOOLEAN</td><td>thirdPartyConfigPresent</td><td>(Function reserved)
Whether 3rd party configuration is present —<table><tr><td>0</td><td>No</td></tr><tr><td>1</td><td>Yes</td></tr></table></td></tr>
<tr><td>DWORD</td><td>sdkVersion</td><td>SDK version for Summit radio</td></tr>
</table>

5.4.7. SDC_ALL

```
typedef struct _SDC_ALL
{
    DWORD                   numConfigs;
    SDCConfig               *configs;
    SDC3rdPartyConfig       *configThirdParty;
    SDCGlobalConfig         *configGlobal;
} SDC_ALL;
```

Data Type	Member Name	Description
DWORD	numConfigs	Number of profiles
		No more than 20. See MAX_CFGS defined in the header file.
SDCConfig	*configs	Refer to SDCConfig structure.
SDC3rdPartyConfig	*configThirdParty	**(Function reserved)** Refer to SDC3rdPartyConfig structure.
SDCGlobalConfig	*configGlobal	Refer to SDCGlobalConfig structure.

5.4.8. SDCCONFIG

```
typedef struct _SDCConfig
{
    CHAR            configName[CONFIG_NAME_SZ];
    CHAR            SSID[SSID_SZ];
    CHAR            clientName[CLIENT_NAME_SZ];
    INT             txPower;
    AUTH            authType;
    EAPTYPE         eapType;
    POWERSAVE       powerSave;
    WEPTYPE         wepType;
    BITRATE         bitRate;
    RADIOMODE       radioMode;
    CRYPT           userName;
    CRYPT           userPwd;
    CRYPT           PSK;
    CRYPT           WEPKeys;
} SDCConfig;
```

<table>
<tr><th>Data Type</th><th>Member Name</th><th>Description</th></tr>
<tr><td>CHAR</td><td>configName[CONFIG_NAME_SZ]</td><td>Name of profile.
No more than 33 characters. See CONFIG_NAME_SZ defined in the header file.</td></tr>
<tr><td>CHAR</td><td>SSID[SSID_SZ]</td><td>Service Set Identifier (SSID) which Summit radio will connect to.
No more than 33 characters. See SSID_SZ defined in the header file.</td></tr>
<tr><td>CHAR</td><td>clientName[CLIENT_NAME_SZ]</td><td>Name of Summit radio.
No more than 17 characters. See CLIENT_NAME_SZ defined in the header file.</td></tr>
<tr><td>INT</td><td>txPower</td><td>The TX power in use —<table><tr><td>0</td><td>Maximum defined for the current regulatory domain</td></tr><tr><td>1</td><td>1 mW</td></tr><tr><td>5</td><td>5 mW</td></tr><tr><td>10</td><td>10 mW</td></tr><tr><td>20</td><td>20 mW</td></tr><tr><td>30</td><td>30 mW</td></tr><tr><td>50</td><td>50 mW</td></tr></table></td></tr>
</table>

<table>
<tr><td>AUTH</td><td>authType</td><td>The authentication type in use —<table><tr><td>0</td><td>Open</td></tr><tr><td>1</td><td>Shared-key</td></tr><tr><td>2</td><td>LEAP (Network-EAP)</td></tr></table></td></tr>
<tr><td>EAPTYPE</td><td>eapType</td><td>The authentication type in use —<table><tr><td>0</td><td>EAP_NONE</td></tr><tr><td>1</td><td>EAP_LEAP</td></tr><tr><td>2</td><td>EAP_EAPFAST</td></tr><tr><td>3</td><td>PEAPMSCHAP</td></tr><tr><td>4</td><td>PEAPGTC</td></tr><tr><td>5</td><td>EAPTLS</td></tr></table></td></tr>
</table>

<table>
<tr><td>POWERSAVE</td><td>powerSave</td><td>The power saving setting —
Constantly Awake Mode (CAM) keeps the client adapter powered up continuously so there is little lag in message response time. It consumes the most power but offers the highest throughput. It is recommended when AC power is in use.
Max Power Savings (Max PSP) causes the access point to buffer incoming messages for the client adapter, which wakes up periodically and polls the access point to see if any buffered messages are waiting for it. The client adapter can request each message and then go back to sleep. It conserves the most power but offers the lowest throughput. It is recommended when battery power is in use.
Power Save Mode (Fast PSP) switches between the two modes described above, depending on network traffic. This mode switches to CAM when retrieving a large number of packets and switches back to PSP (= PS-Poll Procedure) after the packets have been retrieved. It is recommended when power consumption is a concern but you need greater throughput than that allowed by Max PSP.
<table><tr><td>0</td><td>Constantly awake mode (CAM)</td></tr><tr><td>1</td><td>Maximum power saving</td></tr><tr><td>2</td><td>Fast power save mode</td></tr></table></td></tr>
<tr><td>WEPTYPE</td><td>wepType</td><td>The WEP type in use —<table><tr><td>0</td><td>WEP_OFF</td></tr><tr><td>1</td><td>WEP_ON</td></tr><tr><td>2</td><td>WEP_AUTO</td></tr><tr><td>3</td><td>WEP_PSK</td></tr><tr><td>4</td><td>WEP_TKIP</td></tr><tr><td>5</td><td>WEP2_PSK</td></tr><tr><td>6</td><td>WEP2_AES</td></tr><tr><td>7</td><td>CCKM_TKIP</td></tr><tr><td>8</td><td>WEP_CKIP</td></tr><tr><td>9</td><td>WEP_AUTO_CKIP</td></tr><tr><td>10</td><td>CCKM_AES</td></tr></table></td></tr>
<tr><td>BITRATE</td><td>bitRate</td><td>Data rate used by Summit radio when interacting with AP.<table><tr><td>0</td><td>BITRATE_AUTO</td></tr><tr><td>2</td><td>BITRATE_1</td></tr><tr><td>4</td><td>BITRATE_2</td></tr><tr><td>11</td><td>BITRATE_5_5</td></tr><tr><td>12</td><td>BITRATE_6</td></tr><tr><td>18</td><td>BITRATE_9</td></tr></table></td></tr>
</table>

		22 BITRATE_11 24 BITRATE_12 36 BITRATE_18 48 BITRATE_24 72 BITRATE_36
RADIOMODE	radioMode	The radio mode in use — 0 RADIOMODE_B_ONLY 1 RADIOMODE_BG 2 RADIOMODE_G_ONLY 3 RADIOMODE_BG_LRS 4 RADIOMODE_A_ONLY 5 RADIOMODE_ABG 6 RADIOMODE_BGA 7 RADIOMODE_ADHOC
CRYPT	userName	Username for authentication (EAP) No more than 72 characters. See CRED_CA_POS defined in the header file.
CRYPT	userPwd	Password for authentication (EAP) No more than 72 characters. See CRED_UCA_POS defined in the header file.
CRYPT	PSK	Pre-Shared Key for encryption.
CRYPT	WEPKeys	WEP Keys for encryption.

5.4.9. SDC3RDPARTYCONFIG

```
typedef struct _SDC3rdPartyConfig
{
    CHAR            clientName[CLIENT_NAME_SZ];
    POWERSAVE       powerSave;
    INT             txPower;
    BITRATE         bitRate;
    RADIOMODE       radioMode;
} SDC3rdPartyConfig;
```

<table>
<tr><th>Data Type</th><th>Member Name</th><th>Description</th></tr>
<tr><td>CHAR</td><td>clientName[CLIENT_NAME_SZ]</td><td>Name of Summit radio
No more than 17 characters. See CLIENT_NAME_SZ defined in the header file.</td></tr>
<tr><td>POWERSAVE</td><td>powerSave</td><td>The power saving setting —
▸ Constantly Awake Mode (CAM) keeps the client adapter powered up continuously so there is little lag in message response time. It consumes the most power but offers the highest throughput. It is recommended when AC power is in use.
▸ Max Power Savings (Max PSP) causes the access point to buffer incoming messages for the client adapter, which wakes up periodically and polls the access point to see if any buffered messages are waiting for it. The client adapter can request each message and then go back to sleep. It conserves the most power but offers the lowest throughput. It is recommended when battery power is in use.
▸ Power Save Mode (Fast PSP) switches between the two modes described above, depending on network traffic. This mode switches to CAM when retrieving a large number of packets and switches back to PSP (= PS-Poll Procedure) after the packets have been retrieved. It is recommended when power consumption is a concern but you need greater throughput than that allowed by Max PSP.
<table><tr><td>0</td><td>Constantly awake mode (CAM)</td></tr><tr><td>1</td><td>Max Power Savings (Max PSP)</td></tr><tr><td>2</td><td>Power Save Mode (Fast PSP)</td></tr></table></td></tr>
<tr><td>INT</td><td>txPower</td><td>The TX power in use —
<table><tr><td>0</td><td>Maximum defined for the current regulatory</td></tr></table></td></tr>
</table>

		domain
		1 mW
		5 mW
		10 mW
		20 mW
		30 mW
		50 mW
BITRATE	bitRate	Data rate used by Summit radio when interacting with AP.
		0 BITRATE_AUTO
		2 BITRATE_1
		4 BITRATE_2
		11 BITRATE_5_5
		12 BITRATE_6
		18 BITRATE_9
		22 BITRATE_11
		24 BITRATE_12
		36 BITRATE_18
		48 BITRATE_24
		72 BITRATE_36
RADIOMODE	radioMode	The radio mode in use —
		0 RADIOMODE_B_ONLY
		1 RADIOMODE_BG
		2 RADIOMODE_G_ONLY
		3 RADIOMODE_BG_LRS
		4 RADIOMODE_A_ONLY
		5 RADIOMODE_ABG
		6 RADIOMODE_BGA
		7 RADIOMODE_ADHOC

5.4.10. SDCGLOBALCONFIG

```
typedef struct _SDCGlobalConfig
{
    DWORD           fragThreshold;
    DWORD           RTSThreshold;
    RX_DIV          RxDiversity;
    TX_DIV          TxDiversity;
    ROAM_TRIG       roamTrigger;
    ROAM_DELTA      roamDelta;
    ROAM_PERIOD     roamPeriod;
    PREAMBLE        preamble;
    GSHORTSLOT      g_shortslot;
    BT_COEXIST      BTcoexist;
    PING_PAYLOAD    pingPayload;
    DWORD           pingTimeout;
    DWORD           pingDelay;
    DWORD           radioState;
    DWORD           displayPasswords;
    DWORD           adminOverride;
    DWORD           txMax;
    FCC_TEST        FCCtest;
    DWORD           testChannel;
    BITRATE         testRate;
    TXPOWER         testPower;
    DWORD           regDomain;
    DWORD           ledUsed;
    DWORD           txTestTimeout;
    DWORD           WMEenabled;
    DWORD           CCXfeatures;
    CHAR            certPath[MAX_CERT_PATH];
    CRYPT           adminPassword;
    DWORD           bLRS;
    DWORD           avgWindow;
    DWORD           probeDelay;
    DWORD           polledIRQ;
    DWORD           keepAlive;
    DWORD           trayIcon;
    DWORD           aggScanTimer;
    DWORD           authTimeout;
```

```
    DWORD           autoProfile;
    DWORD           Reserved0[6];
    DWORD           txMaxA;
    DWORD           adminFiles;
    DWORD           DFSchannels;
    DWORD           interferenceMode;
    DWORD           authServerType;
    DWORD           Reserved1[4];
} SDCGlobalConfig;
```

Data Type	Member Name	Description
DWORD	fragThreshold	If packet size exceeds threshold, then it is fragmented — ▸ 256 ~ 2346 (bytes) See FRAG_LOW and FRAG_HIGH defined in the header file.
DWORD	RTSThreshold	Packet size above which RTS/CTS is required on link — ▸ 0 ~ 2347 (bytes) See RTS_LOW and RTS_HIGH defined in the header file.
RX_DIV	RxDiversity	How to handle antenna diversity when receiving data from AP — 0 — Use main antenna only 1 — Use auxiliary antenna only 2 — On startup, use auxiliary antenna 3 — On startup, use main antenna
TX_DIV	TxDiversity	How to handle antenna diversity when transmitting data to AP — 0 — Use main antenna only 1 — Use auxiliary antenna only 2 — --- 3 — Use diversity
ROAM_TRIG	roamTrigger	When the moving average RSSI from the current AP is weaker than Roam Trigger, radio does a roam scan where it probes for an AP with a signal that is at least Roam Delta dBm stronger — 50 — -50 dBm 55 — -55 dBm 60 — -60 dBm 65 — -65 dBm 70 — -70 dBm 75 — -75 dBm

		80 -80 dBm 85 -85 dBm 90 -90 dBm
ROAM_DELTA	roamDelta	When Roam Trigger is met, a second AP's signal strength (RSSI) must be Roam Delta dBm stronger than the moving average RSSI for the current AP before radio will attempt to roam to the second AP — 5 5 dBm 10 10 dBm 15 15 dBm 20 20 dBm 25 25 dBm 30 30 dBm 35 35 dBm 40 40 dBm 45 45 dBm 50 50 dBm 55 55 dBm 60 60 dBm
ROAM_PERIOD	roamPeriod	After association or roam scan (with no roam), radio will collect RSSI scan data from Roam Period seconds before considering roaming — 5 5 sec 10 10 sec 15 15 sec 20 20 sec 25 25 sec 30 30 sec 35 35 sec 40 40 sec 45 45 sec 50 50 sec 55 55 sec 60 60 sec
PREAMBLE	preamble	(Not implemented)
GSHORTSLOT	g_shortslot	(Not implemented)
BT_COEXIST	BTcoexist	(Not implemented)

<table>
<tr><td>PING_PAYLOAD</td><td>pingPayload</td><td>The amount of data to be transmitted on a pin —<table><tr><td>32</td><td>32 bytes</td></tr><tr><td>64</td><td>64 bytes</td></tr><tr><td>128</td><td>128 bytes</td></tr><tr><td>256</td><td>256 bytes</td></tr><tr><td>512</td><td>512 bytes</td></tr><tr><td>1024</td><td>1024 bytes</td></tr></table></td></tr>
<tr><td>DWORD</td><td>pingTimeout</td><td>The amount of time that elapses without a response before ping request is considered a failure —
0 ~ 30000 (milliseconds)
See PTIME_LOW and PTIME_HIGH defined in the header file.</td></tr>
<tr><td>DWORD</td><td>pingDelay</td><td>The amount of time that elapses between successive ping requests —
0 ~ 7200000 (milliseconds)
See PDELAY_LOW and PDELAY_HIGH defined in the header file.</td></tr>
<tr><td>DWORD</td><td>radioState</td><td>Whether to enable radio or not —<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
<tr><td>DWORD</td><td>displayPasswords</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>adminOverride</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>txMax</td><td>(Not implemented)</td></tr>
<tr><td>FCC_TEST</td><td>FCCtest</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>testChannel</td><td>(Not implemented)</td></tr>
<tr><td>BITRATE</td><td>testRate</td><td>(Not implemented)</td></tr>
<tr><td>TXPOWER</td><td>testPower</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>regDomain</td><td>Check the regulatory domain in use —<table><tr><td>0</td><td>FCC (Americas)</td></tr><tr><td>1</td><td>ETSI (Europe)</td></tr><tr><td>2</td><td>TELEC (Japan)</td></tr><tr><td>3</td><td>Worldwide</td></tr></table></td></tr>
<tr><td>DWORD</td><td>ledUsed</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>txTestTimeout</td><td>(Not implemented)</td></tr>
<tr><td>DWORD</td><td>WMEenabled</td><td>Whether to allow the use of Wi-Fi Multimedia Extensions (WME) or not —<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
<tr><td>DWORD</td><td>CCXfeatures</td><td>Whether to allow the use of three CCX features (AP-assisted</td></tr>
</table>

		roaming, AP-specified maximum transmit power, and radio management) — 0 Disable 1 Enable (for Cisco APs only)
CHAR	certPath[MAX_CERT_PATH]	File path where the certificate is stored. No more than 65 characters. See MAX_CERT_PATH defined in the header file.
CRYPT	adminPassword	(Not implemented)
DWORD	bLRS	(Not implemented)
DWORD	avgWindow	(Not implemented)
DWORD	probeDelay	(Not implemented)
DWORD	polledIRQ	(Not implemented)
DWORD	keepAlive	Specify how often a null packet gets sent in seconds in CAM mode (= no power saving) — 0 Never 9 Default
DWORD	trayIcon	Whether to enable the system tray icon or not — 0 Disable 9 Enable
DWORD	aggScanTimer	Aggressive scanning complements and works in conjunction with the standard scanning that is configured through the Roam Trigger, Roam Delta, and Roam Period settings. It is recommended that aggressive scanning is enabled unless there is significant co-channel interference because of overlapping coverage from APs that are on the same channel 0 Disable 1 Enable
DWORD	authTimeout	For EAP credentials: 8 seconds by default
DWORD	autoProfile	(Not implemented)
DWORD	Reserved0[6]	(Reserved)
DWORD	txMaxA	(Not implemented)
DWORD	adminFiles	Whether to allow import/export settings to file — 0 No 1 Yes
DWORD	DFSchannels	(Not implemented)
DWORD	interferenceMode	Specify the interference mode — 0 Off 1 Non-WLAN

		2 WLAN 3 Auto
DWORD	authServerType	Specify the authentication server type — 0 ACS (type 1) 1 SBR (type 2)
DWORD	Reserved1[4]	(Reserved)

5.4.11. ERROR CODE (SDCERR)

Error Code	SDCERR Description
1	SDCERR_FAIL
2	SDCERR_INVALID_NAME
3	SDCERR_INVALID_CONFIG
4	SDCERR_INVALID_DELETE
5	SDCERR_POWERCYCLE_REQUIRED
6	SDCERR_INVALID_PARAMETER
7	SDCERR_INVALID_EAP_TYPE
8	SDCERR_INVALID_WEP_TYPE
9	SDCERR_INVALID_FILE

5.5. BLUETOOTH STRUCTURE

5.5.1. FTP_FILE_INFO

```
type struct
{
INT                 fileType;
WCHAR         filename[200];
INT                 fileSize;
} FTP_FILE_INFO;
```

Data Type	Member Name	Description
INT	fileType	Specify type of the file to save— 0 File 1 Folder
WCHAR	fileName	The name of the file to save.
INT	fileSize	Size of the file name.

5.6. CAMERA STRUCTURES

5.6.1. PICSTATE

```
typedef struct
{
INT                 AutoExposure;
INT                 AutoWhiteBalance;
INT                 ImageFormat;
WCHAR         lpPathName[500];
INT                 PathSize;
WCHAR         lpFileName[500];
INT                 FileSize;
} PICSTATE, *PPICSTATE;
```

Data Type	Member Name	Description
INT	AutoExposure	Reserved
INT	AutoWhiteBalance	Reserved
INT	ImageFormat	The format for the image to capture– 0 JPEG 1 BMP
WCHAR	lpPathName	The directory for saving the image to capture.
INT	PathSize	Size of the path name.
WCHAR	lpFileName	The name of the image to save.
INT	FileSize	Size of the file name.

5.6.2. FLASH

```
type struct
      {
DWORD               PreviewFlash;
DWORD               CaptureFlash;
} FLASH, *PFLASH;
```

Data Type	Member Name	Description
DWORD	PreviewFlash	Whether to enable flashlight (for example, when previewing an image) – 0 Disables flashlight 1 Enables flashlight
DWORD	CaptureFlash	Whether to flash a light for the picture to take – 0 Not flashes 1 Flashes 2 Leaves flashlight on automatic

5.7. USER PREFERRED SETTINGS

5.7.1. USERPREFERENCESPARAMETERS

This structure stores the information about user preferred settings, such as LED illumination power level.

```
struct UserPreferencesParameters
{
    int             iVersion
    int             iSize
    int             illuminationPowerLevel
}
```

Data Type	Member Name	Description
int	iVersion	Current version number: 1
int	iSize	Size of the structure (UserPreferencesParameters).
int	illuminationPowerLevel	Level of LED illumination, in the range of 0 to 10. A larger value indicates higher illumination level. Default value is 10.

Appendix I

SCAN ENGINE SETTINGS

9200 Series Mobile Computer supports the following reader types. Reader availability depends on the hardware integration of the mobile computer.

Scan Engine		ID
1D	*CCD*	*SM1*
1D	*Laser*	*SE955*
2D	*2D Imager (Software decode)*	*SE4500*
RFID	*ID_MOD_MP_RFID*	

Reader combination allowed is either 1D+RFID or 2D+RFID. Each combination is able to initialize both readers at the same time (dual mode operation). For example, when you hit the physical scan key, the mobile computer reads a printed barcode in position or an RFID tag in proximity depending on which one comes first.

Note: (1) 1D and 2D scan engines don't coexist on the mobile computer because they are both barcode readers and the mobile computer allows only one barcode reader on board.

(2) Run only one reader-controlling application at a time. For example, while running MIRROR Browser, avoid running Reader Configuration, Terminal Emulation, or any other application that uses ReaderDLL.

IN THIS CHAPTER

Symbologies Supported..... 246
RFID Tags Supported..... 248

SYMBOLOGIES SUPPORTED

According to the scan engine installed, the supported symbologies are listed below.

		CCD	Laser	2D
Codabar		✓	✓	✓
Code 11		✘	✓	✓
Code 39	Code 39	✓	✓	✓
	Trioptic Code 39	✘	✓	✓
	Italian Pharmacode (Code 32)	✓	✓	✓
Code 93		✓	✓	✓
Code 128	Code 128	✓	✓	✓
	GS1-128 (EAN-128)	✓	✓	✓
	ISBT 128	✓	✓	✓
Code 2 of 5	Chinese 25	✘	✓	✓
	Industrial 25 (Discrete 25)	✓	✓	✓
	Interleaved 25	✓	✓	✓
	Convert Interleaved 25 to EAN-13	✘	✓	✓
	Matrix 25	✘	✘	✓
Composite Code	Composite CC-A/B	✘	✘	✓
	Composite CC-C	✘	✘	✓
	Compostie TLC 39	✘	✘	✓
GS1 DataBar (RSS)	GS1 DataBar-14 (RSS-14)	✓	✓	✓
	GS1 DataBar Limited (RSS Limited)	✓	✓	✓
	GS1 DataBar Expanded (RSS Expanded)	✓	✓	✓
	Convert to UPC/EAN	✘	✓	✓
Inverse	Inverse 1D barcodes	✘	✘	✓
Korean 3 of 5		✘	✘	✓
MSI		✓	✓	✓
Postal Codes	Australian Postal	✘	✘	✓
	Japan Postal	✘	✘	✓
	Netherlands KIX Code	✘	✘	✓
	US Postnet	✘	✘	✓
	US Planet	✘	✘	✓
	UK Postal	✘	✘	✓
EAN/UPC	EAN-8	✓	✓	✓
	EAN-8 Extend	✓	✓	✓

	EAN-13	✓	✓	✓
	Bookland EAN (ISBN)	✓	✓	✓
	UCC Coupon Extended Code	✘	✓	✓
	ISSN EAN	✘	✘	✓
	UPC-A	✓	✓	✓
	UPC-E	✓	✓	✓
	Convert UPC-E to UPC-A	✓	✓	✓
	UPC-E1	✓	✓	✓
	Convert UPC-E1 to UPC-A	✓	✓	✓
2D Symbologies	Aztec	✘	✘	✓
	Data Matrix	✘	✘	✓
	Maxicode	✘	✘	✓
	MicroPDF417	✘	✘	✓
	MicroQR	✘	✘	✓
	PDF417	✘	✘	✓
	QR Code	✘	✘	✓

RFID TAGS SUPPORTED

RFID reader supports both reading and writing RFID data. Supported labels include ISO 14443A, ISO 15693 and ISO 18092.

The supported RFID tags are:

ID_MOD_MP_RFID		UID Only	Read Page	Write Page
ISO 14443A	Mifare Standard S50 1K	✓	✓	✓
	Mifare Standard S70 4K	✓	✓	✓
	Mifare Ultralight UL/ULC	✓	✓	✓
	Mifare DESFire	✓	---	---
	SLE66R35	✓	✓	✓
ISO 15693	ICODE SLI	✓	✓	✓
	SRF55V02P	✓	✓	✓
	SRF55V02S	✓	---	---
	SRF55V10P	✓	✓	✓
	TI Tag-it HF-I Pro/Plus	✓	✓	✓
	LRI512	✓	✓	✓
ISO 18092	Felica	✓	---	---

Note: Study the specifications of the RFID tags to read before reading.

INDEX

ActivateConfig, 148
AddConfig, 149
AddWlanSsidToPreferredList, 144
Backlight Structure, 210
Beeper, 40
BKLCTL, 210
Camera Structure, 237
CF10G_STATUS, 218
ChangeMifareKey, 27
Chinese2Of5_1D_SE955, 74
Chinese2Of5_2D_SE4500, 100
ClearSigArea, 202
CloseSigScreen, 201
Codabar_1D_SE955, 75
Codabar_1D_SM1, 54
Codabar_2D_SE4500, 101
Code11_1D_SE955, 81
Code11_2D_SE4500, 108
Code128_1D_SE955, 80
Code128_1D_SM1, 59
Code128_2D_SE4500, 106
Code39_1D_SE955, 69
Code39_1D_SM1, 49
Code39_2D_SE4500, 94
Code93_1D_SE955, 71
Code93_1D_SM1, 51
Code93_2D_SE4500, 96
Composite_2D_SE4500, 111
CONFIG_FILE_INFO, 221
ConnectByMacAddr, 178
COPYDATASTRUCT, 6
CreateConfig, 150
DataOutputSettings, 19
DecodeMsg, 8
DeleteConfig, 149
Development Tool, 2
EanJan_1D_SE955, 67
EanJan_1D_SM1, 47
EanJan_2D_SE4500, 92
EnableCamera, 186
EnablePreviewWindow, 189
EnableSigScreen, 202
Error Code (SDCERR), 235
ExportSettings, 154
FindNextBTDevice, 173
FindService, 175
FLASH, 238
FlushAllConfigKeys, 153
FlushConfigKeys, 153
FTP_FILE_INFO, 236
GetActiveDevice, 15
GetAllConfigs, 151
GetAMicGain, 137
GetBacklightCTL, 123
GetBacklightLV, 124
GetBTConnStatus, 179
GetButtonAssignment, 195
GetCameraResolution, 187
GetConfig, 150
GetConfigFileInfo, 154
GetCurrentConfig, 148
GetCurrentDomain, 143
GetDecodeData, 28
GetDecodeRfidData, 33
GetDecodeRfidUID, 33
GetDecoderVersion, 41
GetDecodeTagType, 34
GetDecodeType, 29
GetDeviceAddress, 176
GetDeviceName, 176
GetDllVer, 41
GetEAPFASTCred, 161
GetEAPTLSCred, 167
GetEAPTTLSCred, 169
GetFeature, 190
GetFile, 182
GetFlashLight, 188
GetGlobalSettings, 152
GetHALUUID, 118
GetHSMicGain, 139
GetKeylightOnOff, 127
GetKeypadLock, 130
GetKeypadMode, 132
GetKeypadSensitivity, 129
GetKeypadState, 134
GetLEAPCred, 160
GetMultipleWEPKeys, 157
GetNextFile, 181
GetNumConfigs, 148
GetOutputRecord, 22
GetPEAPGTCCred, 163
GetPEAPMSCHAPCred, 165
GetPreviewLocation, 189
GetPSK, 159
GetReaderType, 16

GetSDKVersion, 143
GetServiceType, 177
GetSigDllVer, 205
GetSppComPort, 184
GetStillCaptureState, 192
GetSysDevName, 119
GetSysInfo, 120
GetTchLock, 128
GetTorchMode, 193
GetVibratorPower, 122
GetWEPKey, 156
GetWiFiBSSIDList, 147
GetWiFiMac, 143
GetWiFiPower, 140
GetWlanAvailableList, 144
GetWlanConnectedStatus, 146
GetWlanIpInfo, 142
GetWlanPreferredList, 145
GS1_DataBar_1D_SE955, 79
GS1_DataBar_1D_SM1, 58
GS1_DataBar_2D_SE4500, 105
HaltGraph, 191
ImportSettings, 155
Industrial2Of5_1D_SE955, 73
Industrial2Of5_1D_SM1, 53
Industrial2Of5_2D_SE4500, 98
InitReader, 14
InitSigScreen, 201
Interleaved2Of5_1D_SE955, 72
Interleaved2Of5_1D_SM1, 52
Interleaved2Of5_2D_SE4500, 97
Inverse1D_2D_SE4500, 112
Keypad Structure, 212
KeyPadBind, 212
Korean3Of5_2D_SE4500, 112
LoadSigImage, 204
Matrix2Of5_2D_SE4500, 99
Message Queue, 6, 8
MiscellaneousOption_1D_SE955, 62
MiscellaneousOption_2D_SE4500, 85
ModifyConfig, 150
Msi_1D_S, 56
Msi_1D_SE955, 77
Msi_2D_SE4500, 103
NotificationSettings, 39
PICSTATE, 237
PostalCode_2D_SE4500, 109
PutFile, 183
RadioDisable, 141
RadioEnable, 141
RAW_DATA, 213
ReadBarcodeData, 31
ReadRfidData, 35
ReconnectWlanPreferredList, 146
ResetReaderToDefault, 116
ResetWlanPreferredList, 145
RfidSettings, 24
SaveSigImage, 204
ScanWiFiBSSID, 147
SDC_ALL, 222
SDC3rdPartyConfig, 227
SDCConfig, 223
SDCGlobalConfig, 229
SeKeypadLock, 131
SetActiveDevice, 15
SetAllConfigs, 151
SetAMicGain, 136
SetBacklightCTL, 123
SetBacklightDefault, 125
SetBacklightLV, 124
SetBacklightMax, 125
SetBacklightMin, 126
SetBTConnStatus, 180
SetButtonAssignment, 197
SetCameraResolution, 187
SetDefaultConfigValues, 149
SetEAPFASTCred, 161
SetEAPTLSCred, 168
SetEAPTTLSCred, 170
SetFeature, 190
SetFlashLight, 188
SetGlobalSettings, 152
SetHSMicGain, 138
SetInitLoaderBind, 120
SetKeylightOnOff, 127
SetKeypadMode, 133
SetKeypadSensitivity, 129
SetKeypadState, 135
SetLEAPCred, 160
SetLEDLight, 121
SetMultipleWEPKeys, 158
SetPEAPGTCCred, 164
SetPEAPMSCHAPCred, 166
SetPreviewLocation, 189
SetPSK, 159
SetSigBgColor, 202
SetSigLineColor, 203
SetSigLineWidth, 203
SetSppComPort, 184
SetStillCaptureState, 192
SetSysDevName, 119
SetTchLock, 128
SetTorchMode, 193
SetVibratorPower, 122
SetWEPKey, 156
SetWiFiPower, 140

SetWlanIpInfo, 142
ShowSigScreen, 202
StartBluetooth, 173
StartBrowse, 181
StartFindingDevice, 173
StartGraph, 190
StartPairing, 174
StartStillCapture, 192
StartUnpairing, 174
StopBluetooth, 173
Symbologies_2D_SE4500, 113
SymbologySecurityLevel_2D_SE4500, 83
SysInfo, 208
System Information Structure, 208
SystemSoftReset, 120
TerminateGraph, 191
Upc_1D_SE955, 64
Upc_1D_SM1, 44
Upc_2D_SE4500, 89
UserPreferences_1D_SE955, 60
UserPreferences_1D_SM1, 42
UserPreferences_2D_SE4500, 82
Wi-Fi Structure, 213
Windows Event, 7
Windows Message, 5
WLANADPTINFO, 214
WLANCONNECTEDST, 215
WLANCTL, 216
WriteRfidData, 37