unitech

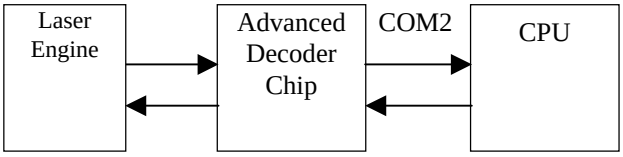
HT660/PA96x/PA982/RH767
Programming Manual

V 1.7 2008-01-15

1. Introduction

1.1. How to download data from scanner

The major difference between the HT660/PA96x/PA982 and a standard HPC/PalmPC is barcode input capability. The WinCE Reference Manual contains no information regarding barcode input. This section will introduce the programming structure of the barcode sub-system and the programming utility library for the HT660/PA96x/PA982. Inside the HT660/PA96x/PA982 there is an advanced decoding chip to control SE900 laser engine and to handle barcode decoding. Below is system diagram for the HT660/PA96x/PA982 barcode:



According to the above diagram, the HT660/PA96x/PA982 communicates with Decoder Chip by mean of serial port COM2. Its communication parameter is fixed on 38400,N,8.1. Normally, the Decoder Chip is in sleep mode when COM2 is not activated. When COM2 is activated, the Decoder Chip will start working, and it will decode the barcode "signal" from the laser engine when the trigger key is pressed. After decoding, barcode data and its symbology type will be sent directly to HT660/PA96x/PA982.

Many programmers find it difficult to control the Decoder Chip via programming language alone, especially if they are not familiar with barcode and serial port controls. Because of this, Unitech provides the following utility library and program for the user or application programmer to control the Decoder Chip:

1. Application program "Scan2Key.exe" is a useful application program that can read input data from the laser scanner and then directly input the data into HT660/PA96x/PA982's keyboard buffer. "Scan2Key.exe" makes barcode data input simple, and can be especially valuable to those programmers not familiar with COM port programming. User program simply reads the barcode data from the keyboard. For barcode symbologies setting, you can run **Scanner Setting** from **Control Panel** to define all of supporting symbologies and delimiter.

2. Utility library:
   For programming control, HT660/PA96x/PA982 provide USI.DLL to let user control scanner input, symbologies setting and profile controlling. Please refer to 2 for detail API lists.

   USI.DLL is Unitech's new scanner function library on HT660/PA96x/PA982. For backward compatible issue, Unitech still provide Scanner3.DLL and ScanKey3.DLL for existing PT930/PT930SA user to port their software into HT660/PA96x/PA982, but several APIs on Scanner3.DLL and ScanKey3.DLL have already been removed on HT660/PA96x/PA982. User can refer to 0and 5 for detail supporting API.

1.2. COM definition for HT660/PA96x/PA982/RH767

COM 1	Physical full RS232 port (ActiveSync)
COM 2	Scanner (Hamster) or RFID reader
COM 3	IrComm
COM 4	USB client
COM 5	IrDA or Bluetooth
COM 6	Reserve
COM 7	Bluetooth Printer
COM 8	Bluetooth Modem
COM 9	Bluetooth ActiveSync

1.3. Useful Sample program

You can get useful sample program for VC, C# and VB.NET from below URL

HT660 series

C# http://w3.tw.ute.com/pub/cs/software/Sample_Program/HT660/c_sharp/c_sharp_sample.zip
Vb.net http://w3.tw.ute.com/pub/CS/software/Sample_Program/HT660/VB.NET/VB.NET_sample.zip
C++ http://w3.tw.ute.com/pub/CS/software/Sample_Program/HT660/evc/evc_sample.zip

PA96x/PA982 series

C# http://w3.tw.ute.com/pub/cs/software/Sample_Program/PA962/c_sharp/c_sharp_sample.zip
Vb.net http://w3.tw.ute.com/pub/CS/software/Sample_Program/PA962/VB.NET/VB.NET_sample.zip
C++ http://w3.tw.ute.com/pub/CS/software/Sample_Program/PA962/evc/evc_sample.zip

1.4. Get SDK from Unitech?

You can get SDK from below URL

HT660 SDK http://w3.tw.ute.com/pub/cs/sdk/ht660/HT660SDK.zip
PA962/963 SDK http://w3.tw.ute.com/pub/cs/sdk/pa962/pa962sdk.zip
PA966/967 SDK http://w3.tw.ute.com/pub/cs/sdk/pa966/pa966sdk.zip
PA982 SDK http://w3.tw.ute.com/pub/cs/sdk/pa982/Pa982SDK.zip
RH767 SDK http://w3.tw.ute.com/pub/cs/sdk/RH767/RH767_CE5_SDK.zip

2. USI.DLL – Unitech Scanner Interface DLL

2.1. Register the application to the USI DLL

Function Description: Register the application to the USI DLL, so that the DLL can communicate with the application. It will also open and initial scanner port (COM2, for example) and set the scanner to the working mode. The application should call USI_Unregister to unregister from the DLL after done with the scanner.

Function call:

```
BOOL USI_Register(HWND hwnd, UINT msgID);
```

Parameter: (input)

hwnd: Handle of the window to which USI DLL will send messages to report all activities, including error messages, scan data ready, etc.

msgID: Specifies the message to be posted. DLL will post messages by calling: PostMessage(hwnd, msgID, msg, param).

The window procedure will receive custom message about msgID and wParam parameter can be one of the followings:

SM_ERROR_SYS: Indicates a system error, which is caused by a call to the system function. Param contains the error code from GetLastEror().

SM_ERROR Indicates an error. Param contains the cause of error, which can be on of followings:

SERR_INVALID_HWND: Invalid window handle.
SERR_INVALID_MSGID: msgID cannot be 0.
SERR_OPEN_SCANNER: Open or initial scanner port failed.
SERR_CHECKSUM: Checksum error in received packet.
SERR_DATALOST: New scan data is lost because data buffer is not empty.
SERR_BUFFEROVERFLOW: Data buffer overflow. The default size is 4K bytes.

SM_REPLY Indicates received a reply. All the responses from the scanner except the scan data will be notified by this message.

SM_DATAREADY Indicates that scan data is successfully decoded and ready to retrieve.

SM_ACK Indicates received a ACK.

SM_NAK Indicates received a NAK.

SM_NOREAD Indicates received a No-Read packet.

Note: Scanner port settings are defined in registry as described below:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Unitech America Inc.\Scanner\Settings]
"COMPORT"="COM2:"
"BAUDRATE"="38400"
"STOPBITS"="1"
"PARITY"="None"
```

"CHECKPARITY"="1"

2.2. Unregister the application from the USI.DLL

Function Description: Unregister the application from the DLL. It will close the scanner port, and by default it will disable the scanner.

Function call: void USI_Unregister();

Return code: None

2.3. Enable / Disable Scanner

Function Description: To start or stop USI function. This function is useful for application to temporarily stop scanner function if it is only need keypad input or keep clear input buffer.

Function call: **BOOL** USI_EnableScan(BOOL bStatus);

Parameter: (input)

bStatus:	TRUE	: Enable Scanner
	FALSE	: Disable Scanner

Return code: BOOL : **TRUE : OK**
FALSE : Failure

2.4. Reset Scanner

Function Description: Set the scanner to the working mode, and reset the communication control.

Function call: BOOL USI_Reset();

Return: Always TRUE

2.5. Get error code

Function Description: Returns the error code (SERR_***).

Function call: DWORD USI_GetError();

Return: Returns the error code (SERR_***), which has been described in USI_Register function.

2.6. Returns the system error code

Function Description: Returns the system error code, which is returned by GetLastError. It will also return the description of the error in buffer if it is not NULL.

Function call: DWORD USI_GetLastSysError(LPTSTR buffer, int len);

Return: Returns the system error code, which is returned by system function GetLastError. It will also return the description of the error in buffer retrieved by system function FormatMessage if it is not NULL.

For a complete list of error codes, refer to the SDK header file WINERROR.H.

2.7. Get scan data

Function Description:

Retrieves the scan data into the buffer. Returns the length of characters. It also returns the barcode type if type is not NULL. Return 0 means that the buffer is too short to hold the data.

USI_GetData should be called when SM_DATAREADY message is received. Or call USI_ResetData to discard the data. Both of them will reset the data buffer so that next scan data can come in.

If the data buffer is not empty and a new scan data occurs, it will be discarded and an error message SM_ERROR with code of SERR_DATALOST will be sent.

Function call:

```
UINT USI_GetData(LPBYTE buffer, UINT len, UINT* type);
```

Parameter: (input)

len : UINT : Len specifies the maximum length of the buffer.

Parameter: (output)

buffer : LPBYTE : Data buffer for storing scanned data

type : UINT : barcode type which is defined on USI.H. Please refer to below list

```
BCT_CODE_39                 // Code 39
BCT_CODABAR                 // CodaBar
BCT_CODE_128                // Code 128
BCT_INTERLEAVED_2OF5        // Interleaves 2 of 5
BCT_CODE_93                 // Code 93
BCT_UPC_A                   // UPC A
BCT_UPC_A_2SUPPS            // UPC A with 2 Supps
BCT_UPC_A_5SUPPS            // UPC A with 5 Supps
BCT_UPC_E0                  // UPC E
BCT_UPC_E0_2SUPPS           // UPC E with 2 Supps
BCT_UPC_E0_5SUPPS           // UPC E with 5 Supps
BCT_EAN_8                   // EAN 8
BCT_EAN_8_2SUPPS            // EAN 8 with 2 Supps
BCT_EAN_8_5SUPPS            // EAN 8 with 5 Supps
BCT_EAN_13                  // EAN 13
BCT_EAN_13_2SUPPS           // EAN 13 with 2 Supps
BCT_EAN_13_5SUPPS           // EAN 13 with 5 Supps
BCT_MSI_PLESSEY             // MSI Plessey
BCT_EAN_128                 // EAN 128
BCT_UPC_E1                  // UPC E1
BCT_UPC_E1_2SUPPS           // UPC E1 with 2 Supps
BCT_UPC_E1_5SUPPS           // UPC E1 with 5 Supps
BCT_TRIOPTIC_CODE_39        // TRIOPTIC CODE 39
BCT_BOOKLAND_EAN            // Bookland EAN
BCT_COUPON_CODE             // Coupon Code
BCT_STANDARD_2OF5           // Standard 2 of 5
BCT_CODE_11_TELPEN          // Code 11 Telpen
BCT_CODE_32                 // Code 32
BCT_DELTA_CODE              // Delta Code
BCT_LABEL_CODE              // Label Code IV & V
BCT_PLESSEY_CODE            // Plessey Code
BCT_TOSHIBA_CODE            // Toshiba Code
                            China Postal Code
```

UINT : Data length

-

2.8. Get length of scanned data

Function Description:

Returns the data length of the scan data. When allocate the memory to hold the scan data, add at least one additional byte for string terminator.

Function call:

```
UINT USI_GetDataLength();
```

Return: UNIT : data length

2.9. Get Symbology name

Function Description:

Returns the barcode name of the type.

Function call:

```
LPCTSTR USI_GetBarcodeName(UINT type, LPBYTE buffer, UINT len);
```

Parameter: (input)

type : UINT : barcode type. (refer to 0 for type definition
buffer : LPBYTE : Please refer to below table

Type	Buffer
BCT_CODE_39	Code 39
BCT_CODABAR	Codabar
BCT_CODE_128	Code 128
BCT_INTERLEAVED_2OF5	Interleaved 2 of 5
BCT_CODE_93	Code 93
BCT_UPC_A	UPC A
BCT_UPC_A_2SUPPS	UPC A with 2 Supps.
BCT_UPC_A_5SUPPS	UPC A with 5 Supps.
BCT_UPC_E0	UPC E
BCT_UPC_E0_2SUPPS	UPC E with 2 Supps.
BCT_UPC_E0_5SUPPS	UPC E with 5 Supps.
BCT_EAN_8	EAN 8
BCT_EAN_8_2SUPPS	EAN 8 with 2 Supps.
BCT_EAN_8_5SUPPS	EAN 8 with 5 Supps.
BCT_EAN_13	EAN 13
BCT_EAN_13_2SUPPS	EAN 13 with 2 Supps.
BCT_EAN_13_5SUPPS	EAN 13 with 5 Supps.
BCT_MSI_PLESSEY	MSI Plessey
BCT_EAN_128	EAN 128
BCT_TRIOPTIC_CODE_39	Trioptic Code 39
BCT_BOOKLAND_EAN	Bookland EAN
BCT_COUPON_CODE	Coupon Code
BCT_STANDARD_2OF5	Standard 2 of 5
BCT_CODE_11_TELPEN	Code 11 or Telpen
BCT_CODE_32	Code 32 (Pharmacy Code)
BCT_DELTA_CODE	Delta Code
BCT_LABEL_CODE	Label Code IV & V
BCT_PLESSEY_CODE	Plessey Code
BCT_TOSHIBA_CODE	Toshiba Code (China Postal Code)

len : UINT : length of string on the 2nd parameter buffer

Return: TRUE : if it found name for the barcode type,
FALSE : if not (type may be wrong

-

2.10. Clear scan data system buffer

Function Description:

Reset the data buffer so that next new scan data can come in.

Function call:

```
void USI_ResetData();
```

2.11. Good read indicator

Function Description:

Inform a good receiving of scan data, this will play a sound (wave file scanok.wav) and light the LED lasting for 1 second.

Function call:

```
void USI_ReadOK();
```

Note:

USI will call the function GoodReadLEDOn function exported by the DLL defined in the registry described below (UPI300.DLL is an example) to turn on and off the LED. If the DLL is not defined or the function is not found, USI will bypass the call of GoodReadLEDOn.

[HKEY_LOCAL_MACHINE\SOFTWARE\Unitech America Inc.\Scanner\Settings]
"DLLLEDCONTROL"="UPI300.DLL"

The function prototype of GoodReadLEDOn is:
VOID WINAPI GoodReadLEDOn(BOOL fon);
Turn on when fon is TRUE, and turn off when fon is FALSE.

2.12. Wait for acknowledgement of the last sent command

Function Description:

Wait for acknowledgement of the last sent command until timeout. It is useful when a serial of commands needs to be sent at a time. Before call USI_SendCommand, call USI_WaitForSendEchoTO to make sure that the previous command is done.

Function call:

```
BOOL USI_WaitForSendEchoTO(DWORD timeout);
```

Parameter: (input)

timeout: DWORD : Specifies the timeout in millisecond.

Return:

Returns FALSE if timeout.

2.13. Save setting to profiles

Function Description:

Save current settings of scanner so that the settings will be persistent when the unit get power off and on again.

Function call:

```
BOOL USI_SaveCurrentSettings();
```

Return : TRUE if success,
otherwise FALSE.

-

2.14. Save scanner setting into specified file

Function Description:

Save the current settings to file. The file takes "*.USI" as extension name.

Function call:

BOOL USI_SaveSettingsToFile(LPCTSTR filename)

Parameter: (input)

filename : LPCTSTR: file name for setting profile

Return: TRUE = success
FALSE = error

2.15. Change scanner setting from specified setting profile

Function Description:

Load and activate the settings from file.

Function call:

BOOL USI_LoadSettingsFromFile(LPCTSTR filename, BOOL formulaOnly);

Parameter: (input)

filename: LPCTSTR : name of scanner setting profile (*.USI)
formulaOnly: BOOL: if TRUE, only data editing formulas are load. The other settings remain unchanged

Return: TRUE = success
FALSE = error

2.16. Automatically enable scanner beam with pressing trigger key

Function Description:

Start auto scanning. Scan engine will be automatically triggerrd on.

Function call:

BOOL USI_StartAutoScan(DWORD interval);

Parameter: (input)

interval : DWORD: Specifies the interval in milli-second

Parameter: (output)
Return:

Note: USI will call the function SetScannerOn function exported by the DLL defined in the registry described below (UPI300.DLL is an example) to start and stop the scanner. If the DLL is not defined or the function is not found, then auto scanning is not available.

[HKEY_LOCAL_MACHINE\SOFTWARE\Unitech America Inc.\Scanner\Settings]
"DLLSCANNERCONTROL"="UPI300.DLL"

The function prototype of SetScannerOn is:
VOID WINAPI SetScannerOn(BOOL fon);
Start when fon is TRUE, and stop when fon is FALSE.

-

2.17. Stop auto scanning function

Function Description:

Stop auto scanning

Function call:

void USI_StopAutoScan();

2.18. Check if auto scanning is enable

Function Description:

Check if auto scanning function is enabled or not

Function call: BOOL USI_IsAutoScanning()

Return: BOOL: TRUE : auto-scanning is running
FALSE : auto-scanning is disabled.

2.19. Check if Scan2Key.exe program is running or not

Function Description:

Test whether Scan2Key application is running at background. (It doesn't mean Scan2Key is routing scanner input to keyboard, please call S2K_IsEnabled() to check if routing function is enable or not)

Function call:

HWND S2K_IsLoaded();

Return: NULL : Scan2Key is not running
Non-NULL : indicates scan2key is running. It actually returns window handle for scan2key, but it is for internal use – send messages.

2.20. Test if Scan2Key is enabled

Function Description:

Test whether Scan2Key is enabled. Scan2Key routes scanning input from scanner to keypad buffer, so that barcode data can be input as like from keystrokes on keypad.

Function call:

BOOL S2K_IsEnabled();

Return: TRUE = enabled.
FALSE = disable

2.21. Load/Unload Scan2Key.exe

Function Description:

Load or unload Scan2Key

Function call:

BOOL S2K_Load(BOOL load, DWORD timeout);

Parameter: (input)

load: BOOL: TRUE = load Scan2Key
FALSE = unload Scan2Key

timeout: DWORD: when unload Scan2Key, it will wait until Scan2Key has been removed from memory or timeout specified by this parameter.

Parameter: (output)

Return: TRUE = successfully loaded.

-

2.22. Enable/Disable Scan2Key

Function Description:

Enable or disable Scan2Key to put scanned data to standard keyboard input buffer. Scan2Key is enabled by default.

Function call:

```
BOOL S2K_Enable(BOOL enable, DWORD timeout);
```

Parameter: (input)

enable:	BOOL:	TRUE = Enable scanned data to keyboard buffer FALSE = Disable scanned data to keyboard
timeout:	DWORD:	when enable or disable Scan2Key, it will wait until Scan2Key has been removed from memory or timeout specified by this parameter.

Parameter: (output)

Return: TRUE : if successfully enabled Scan2Key, otherwise FALSE

2.23. Send scanner command to decoding chip

Function Description:

Send scanner command to decoder chip. This command will send a serial of bytes to decoder chip as following: (Esc and BCC will be calculated and added automatically)

Esc, high-length, low-length, command-ID, operation, set, BCC

Please refer to complete command reference on section 4

```
BOOL HAM_SendCommand(BYTE highlen, BYTE lowlen, BYTE cmdID, BYTE op,
BYTE set);
```

Parameter: (input)

highlen:	BYTE:	high byte of command length
lowlen:	BYTE:	low byte of command length
cmdID:	BYTE:	command ID
op:	BYTE:	operation mode for this command
set:	BYTE:	operand for this command

Return:

TRUE = Indicates the command has been successfully sent to queue to output.

2.24. Only send single command decoding chip

Function Description:

Send command to decoder chip. This is a variation of command HAM_SendCommand. It sends following command to Hamster: (note, only two bytes without BCC)

Esc, 0x80+cmd

Function call:

```
BOOL HAM_SendCommand1(BYTE cmd);
```

Parameter: (input)

cmd:	BYTE:	command

Return:

TRUE = indicates the command has been successfully sent to queue to output.

-

2.25. Send command to decoding chip

Function Description:

Send command to decoder chip. This is a variation of command HAM_SendCommand. It will read a number of parameters and packet them as in following format and send it to decoder chip.

Esc, parameter1, parameter2, …, BCC

The total number of parameters is specified by first parameter num.

Function call:

BOOL HAM_SendCommand2(BYTE num, BYTE parameter1, …);

Parameter: (input)

num: BYTE: number of total parameters

parameter*x* BYTE: Parameter

Parameter: (output)

Return:

TRUE = indicates the command has been successfully sent to queue to output.

2.26. 2D imager supporting for PA966/967

2D supporting API is described on individual document. Please get it from
http://w3.tw.ute.com/pub/cs/manual/WinCE_programming_manual/2D_Engine_SDK.pdf

-

3. Control command for decoder chip

Important: This chapter describes low level command for scanner control function. If you already USI to do scanner programming, you don't need to care about this chapter. In general, it is not suggested to use level command to control scanner, because there are timing issue on serial communication programming , and it is always need communication expert to do that and it is hard to explain it on document.

When Host prepare to send a command to hamster, it must first check CTS, if CTS is high, then Host must set the RTS to high then clear RTS to low to wake up the Hamster.

Special Command for control		
command	Format	Comment
Control	Esc,80H+SOH(01H)	Let Hamster enter slaving status. At this status Hamster just receives commands and executes it until it receives Release command or timeout (about 10s). Otherwise, the timeout is about 1s as the interval of commands.
Release	Esc,80H+EOT(04H)	Let Hamster exit from slaving status.
Execute/ Enquiry	Esc,80H+ENQ(05H)	Let Hamster execute the previous saved command and check hamster if there is a result of previous executed command to send to Host. If previous saved command have already executed and no result to send, hamster do not reply until there is a result. If Host receive a result but the BCC is wrong, it can re-send ENQ to re-send result again.
ACK	Esc,80H+ACK(06H)	It is from Hamster to Host. If Hamster receive a command and this command do not need send message back, Hamster reply the ACK.
NAK	Esc,80H+NAK(15H)	It is from Hamster to Host. Hamster require the Host to re-send command again, normally when received a wrong BCC, it can send the NAK. The Hamster sends back NAK whenever it receives a no sense command.

COMMAND FROM HOST TO HMASTER		
Command format:Esc,Lh,Ll,n,m,S1,...,Si,BCC Here: Esc is Escape code(H'1B) Lh/Ll is command's length when the Lh.b7 is 0, Lh is high byte, Ll is low byte, count from n to BCC. When Lh.b7=1 it is a two bytes special command. n is command ID m is operation: Normally for setting commands the 0 means setting, 1 default, 2 read current setting, 3 special operation. When m=1 or 2, the S1 should be 0 for bits or one character setting. If the setting is a string, like pre_amble, the read or default command should not contain any Si byte. The special meaning in a command please refers the command definition. Si is setting/read data. BCC: it equals to XOR of all the bytes before the BCC. Conventions: S1.bj means the number j bit of byte S1. The expression 1~64:2 means that the number is between 1 and 64, the default is 2. Notice: Any interval in a command transmit can not exceed 1 second.		
Command	Format	Comment
Initial/ Warm start	Esc,0,2,0,BCC	Hamster initializes the ports and flags according to the setting in RAM.
Default	Esc,0,2,1,BCC	Reset setting in RAM and initialize
Mpu_idle	Esc,0,4,2,m,S1,BCC	S1 is 0~3:0 is sleep mode,1 is watch mode, 2 is standby mode.
Beep	Esc,0,4,3,m,S1,BCC	S1 0 none,1 low,2 medium,3 high,4

-

		low/high,5high/low
block delay	Esc,0,4,4,m,S1,BCC	S1 is 0 10ms,1 50ms,2 100ms,3 500ms,4 1s,5 3s
char delay	Esc,0,4,5,m,S1,BCC	S1 is 0 none,1 1ms,2 5ms,3 10ms,4 20ms,5 50ms
Function_code	Esc,0,4,6,m,S1,BCC No meaning for you	S1 is 0 off,1_on
Capslock	Esc,0,4,7,m,S1,BCC No meaning for you	S1 is 0_auto trace,1 lower case,2 upper case
Language	Esc,0,4,8,m,S1,BCC No meaning for you	S1 is 0_U.S.,1 U.K.,2 Swiss,3 Swedish, 4 Spanish,5 Norwegian,6 Italian,7 German,8 French,9 Alt Key Mode,A Danish
Baud_rate	Esc,0,4,0D,m,S1,BCC No meaning for you	S1 is 0 300,1 600,2 1200,3 2400,4 4800, 5 9600,6 19200,7 38400
Parity	Esc,0,4,0E,m,S1,BCC No meaning for you	S1 is 0 EVEN,1 ODD,2 MARK,3 SPACE,4_NONE
Data_bits	Esc,0,4,0F,m,S1,BCC No meaning for you	S1 is 0 7,1_8BIT
Handshake	Esc,0,4,10,m,S1,BCC No meaning for you	S1 is 0_IGNORE,1 RTS ENABLE AT POWERUP,2 RTS ENABLE IN COMMUNICATION
Ack_nak	Esc,0,4,11,m,S1,BCC No meaning for you	S1 is 0_OFF,1 ON
BCC_char	Esc,0,4,12,m,S1,BCC No meaning for you	S1 is 0_OFF,1 ON
Data_direction	Esc,0,4,13,m,S1,BCC No meaning for you	S1 is =0_SEND TO HOST,1 SEND TO HOST AND TERMINAL,2 SEND TO TERMINAL
Time_out	Esc,0,4,14,m,S1,BCC No meaning for you	S1 is 0_1S,1 3S,2 10S,3 UNLIMITED
Terminator	Esc,0,4,15,m,S1,BCC	S1 is B1B0=0_ENTER(CR/LF),1 FIELD EXIT(CR),2 RETURN(LF),3 NONE
Code id	Esc,0,4,16,m,S1,BCC	S1 is 0 OFF,1 ON
Verification	Esc,0,4,17,m,S1,BCC	S1 is 0 OFF,1~7 1 to 7 times verification
Scan_mode	Esc,0,4,18,m,S1,BCC	S1 is 0_TRIGGER MODE,1 FLASH_MODE,2 MULTISCAN MODE,3 ONE PRESS ONE SCAN,4~7 reserved
Label type	Esc,0,4,19,m,S1,BCC	S1 is 0 POSITIVE,1 POSITIVE AND NEGATIVE
Aim fuction	Esc,0,4,1a,m,S1,BCC	S1 is 0 DISABLE,1 ENABLE
Scan_pre_data	Esc,0,L,1b,m,S1,…Si,BCC	Si can be 1 to 8 CHARACTERS
Scan_post_data	Esc,0,L,1c,m,S1,…Si,BCC	Si can be 1 to 8 CHARACTERS
Define_code39f	Esc,0,4,1d,m,S1,BCC	define Code 39 full ASCII ID:Here S1 is 1 CHARACTER
Define_code39s	Esc,0,4,1e,m,S1,BCC	define Code 39 standard ID:Here S1 is 1 CHARACTER
Define_EAN13	Esc,0,4,1f,m,S1,BCC	define EAN13 ID:Here S1 is 1 CHARACTER
Define UPCA	Esc,0,4,20,m,S1,BCC	define UPC A ID: Here S1 is 1 CHARACTER
Define EAN8	Esc,0,4,21,m,S1,BCC	define EAN8 ID:Here S1 is 1 CHARACTER
Define UPCE	Esc,0,4,22,m,S1,BCC	define UPC E ID:Here S1 is 1 CHARACTER
Define I25	Esc,0,4,23,m,S1,BCC	define I25 ID:Here S1 is 1 CHARACTER
Define CDB	Esc,0,4,24,m,S1,BCC	define Codabar ID:Here S1 is 1 CHARACTER
Define C128	Esc,0,4,25,m,S1,BCC	define Code128 ID:Here S1 is 1 CHARACTER
Define C93	Esc,0,4,26,m,S1,BCC	define Code93 ID:Here S1 is 1 CHARACTER
Define S25	Esc,0,4,27,m,S1,BCC	define S25 ID:Here S1 is 1 CHARACTER
Define MSI	Esc,0,4,28,m,S1,BCC	define MSI ID:Here S1 is 1 CHARACTER
Define C11	Esc,0,4,29,m,S1,BCC	define Code11 ID:Here S1 is 1 CHARACTER
Define C32	Esc,0,4,2a,m,S1,BCC	define Code32 ID:Here S1 is 1 CHARACTER
Define_DELTA	Esc,0,4,2b,m,S1,BCC	define Delta ID:Here S1 is 1 CHARACTER
Define_LABEL	Esc,0,4,2c,m,S1,BCC	define Label code ID:Here S1 is 1 CHARACTER
Define_PLESSEY	Esc,0,4,2d,m,S1,BCC	define Plessey ID:Here S1 is 1 CHARACTER
Define_TELEPEN	Esc,0,4,2e,m,S1,BCC	define Telepen ID:Here S1 is 1 CHARACTER
Define_TOSHIBA	Esc,0,4,2f,m,S1,BCC	define Toshiba ID:Here S1 is 1 CHARACTER
Define_EAN128	Esc,0,4,30,m,S1,BCC	define EAN128 ID:Here S1 is 1 CHARACTER;IF H'FF, THEN USE "]C1"
Mterminator	Esc,0,4,31,m,S1,BCC No meaning for you	Here S1 is 0_ENTER,1 NONE
Sentinal	Esc,0,4,32,m,S1,BCC No meaning for you	S1 is 0 not send,1 send
Track_selection	Esc,0,4,33,m,S1,BCC No meaning for you	Here S1 is =0_ALL TRACKS,1 TRACK1 AND TRACK2,2 TRACK1 AND TRACK3,3 TRACK2 AND TRACK3,4 TRACK1,5 TRACK2,6 TRACK3
T2_account_only	Esc,0,4,34,m,S1,BCC	S1 is 0 NO,1 YES

-

	No meaning for you	
Separator	Esc,0,4,35,m,S1,BCC No meaning for you	S1 is 1 CHARACTER
Must_have_data	Esc,0,4,36,m,S1,BCC No meaning for you	S1 is 0 YES,1_NO
Track1_sequence	Esc,0,L,37,m,S1,…Si,BCC No meaning for you	Si can be 1 to 16 CHARACTERS
Track2_sequence	Esc,0,L,38,m,S1,…Si,BCC No meaning for you	Si can be 1 to 8 CHARACTERS
Code39_set	Esc,0,4,39,m,S1,BCC	S1.B0 is for Code39_enable,S1.B1 is for Code39_standard,S1.B3B2 for Code39_cd,S1.B4 Code39 ss
Code39_enable	Esc,0,4,3a,m,S1,BCC	S1 is 0 disable,1 enable
Code39_sandard	Esc,0,4,3b,m,S1,BCC	S1 is 0 full ASCII,1 standard
Code39_cd:	Esc,0,4,3c,m,S1,BCC	S1 is 0 calculate&send,1 calculate&not send,2 not calculate
Code39 ss	Esc,0,4,3d,m,S1,BCC	Here S1 is 0 SS send,1 SS not send
Code39 min	Esc,0,4,3e,m,S1,BCC	S1 is 0~48:0 (min<=data len)
Code39 max	Esc,0,4,3f,m,S1,BCC	S1 is 0~48:48 (data len<=max)
I2of5_set	Esc,0,4,40,m,S1,BCC	S1 is S1.B0 is for I2of5_enable,S1.B1 is for I2of5_fixlength,S1.B3B2 is for I2of5_cd,S1.B5B4 is for I2of5 ss
I2of5_enable	Esc,0,4,41,m,S1,BCC	S1 is =0 disable,1 enable
I2of5_fixlength	Esc,0,4,42,m,S1,BCC	S1 is =0 on,1 off (record first 3 record len)
I2of5_cd	Esc,0,4,43,m,S1,BCC	S1 is =0 calculate&send,1 calculate&not send,2 no calculation
I2of5_ss	Esc,0,4,44,m,S1,BCC	S1 is 0 first digit suppressed,1 last digit suppressed,2 not supressed
I25 min	Esc,0,4,45,m,S1,BCC	S1 is 2~64:10 (min<=data len)
I25 max	Esc,0,4,46,m,S1,BCC	S1 is 2~64:64 (data len<=max)
S2of5_set	Esc,0,4,47,m,S1,BCC	S1 is S1.b0 is for S2of5_enable,S1.b1 is for S2of5 fixlength,S1.b3b2 is for S2of5 cd
S2of5_enable	Esc,0,4,48,m,S1,BCC	S1 is 0 disable,1 enable
S2of5_fixlength	Esc,0,4,49,m,S1,BCC	S1 is 0 on,1 off (record first 3 record len)
S2of5_cd	Esc,0,4,4a,m,S1,BCC	S1 is 0 calculate&send,1 calculate&not send, 2 not calculate
S25 min	Esc,0,4,4b,m,S1,BCC	S1 is 1~48:4 (min<=data len)
S25 max	Esc,0,4,4c,m,S1,BCC	S1 is 1~48:48 (data len<=max)
Code32_set	Esc,0,4,4d,m,S1,BCC	S1 is S1.b0 is for Code32_enable,S1.b1 is for Code32 sc,S1.b2 is for Code32 lc
Code32_enable	Esc,0,4,4e,m,S1,BCC	S1 is 0 disable,1 enable
Code32 sc	Esc,0,4,4f,m,S1,BCC	S1 is 0 leading char send,1 not send
Code32 lc	Esc,0,4,50,m,S1,BCC	S1 is 0 tailing char send,1 not send
Telepen	Esc,0,4,51,m,S1,BCC	S1 is S1.b0 is for Telepen_enable,S1.b1 is for Telepen charset
Telepen_enable	Esc,0,4,52,m,S1,BCC	S1 is 0 disable,1 enable
Telepen_charset	Esc,0,4,53,m,S1,BCC	S1 is 0 standard,1 numeric
Ean128	Esc,0,4,54,m,S1,BCC	S1 is S1.b0 is for Ean128_id, S1.b1 is for Ean128 id
Ean128_enable	Esc,0,4,55,m,S1,BCC	S1 is 0 disable,1 enable
Ean128 id	Esc,0,4,56,m,S1,BCC	S1 is 0 ID disable,1 ID enable
Ean128_func1	Esc,0,4,57,m,S1,BCC	S1 is 1 char
Code128	Esc,0,4,58,m,S1,BCC	S1 is 0 disable,1 enable
Code128 min	Esc,0,4,59,m,S1,BCC	S1 is 1~64:1 (min<=data len)
Code128 max	Esc,0,4,5a,m,S1,BCC	S1 is 1~64:64 (data len<=max)
Msi_pleasey	Esc,0,4,5b,m,S1,BCC	S1 is S1.b0 is for Msi_p_enable,S1.b1 is for Msi pleasey cd, S1.b3b2 is for Msi p cdmode
Msi_p_enable	Esc,0,4,5c,m,S1,BCC	S1 is 0_disable,1 enable
Msi_pleasey_cd	Esc,0,4,5d,m,S1,BCC	S1 is 0 check digit send,1 not send
Msi_p_cdmode	Esc,0,4,5e,m,S1,BCC	S1 is 0 check digit double module 10,1 check digit module 11 plus 10,2 check digit single module 10
Msi_pleasey_min	Esc,0,4,5f,m,S1,BCC	S1 is 1~64:1 (min<=data len)
Msi_pleasey_max	Esc,0,4,60,m,S1,BCC	S1 is 1~64:64 (data len<=max)
Code93	Esc,0,4,61,m,S1,BCC	S1 is 0 disable,1_enable
Code93 min	Esc,0,4,62,m,S1,BCC	S1 is 1~48:1 (min<=data len)

-

Code93 max	Esc,0,4,63,m,S1,BCC	S1 is 1~48:48 (data len<=max)
Code11	Esc,0,4,64,m,S1,BCC	S1 is S1.b0 is for Code11_enable,S1.b1 is for Code11 cdnumber,S1.b2 Code11 cdsend
Code11_enable	Esc,0,4,65,m,S1,BCC	S1 is 0 disable, 1 enable
Code11_cdnumber	Esc,0,4,66,m,S1,BCC	S1 is 0 one check digit,1 two check digits
Code11_cdsend	Esc,0,4,67,m,S1,BCC	S1 is 0 check digit send,1 not send
Code11 min	Esc,0,4,68,m,S1,BCC	S1 is 1~48:1 (min<=data len)
Code11 max	Esc,0,4,69,m,S1,BCC	S1 is 1~48:48 (data len<=max)
Codabar_set	Esc,0,4,6a,m,S1,BCC	S1 is S1.b0 is for Codabar_enable, S1.b1 is for Codabar_ss, S1.b3b2 is for Codabar_cd, S1.b4 is for Codabar CLSI
Codabar_enable	Esc,0,4,6b,m,S1,BCC	S1 is 0 disable,1 enable
Codabar ss	Esc,0,4,6c,m,S1,BCC	S1 is 0 start&stop char send,1 not send
Codabar_cd	Esc,0,4,6d,m,S1,BCC	S1 is 0 check digit calculate&send,1 check digit calculate but not send,2_check digit not calculate
Codabar_CLSI	Esc,0,4,6e,m,S1,BCC	S1 is 0 CLSI format on,1 off
Codabar min	Esc,0,4,6f,m,S1,BCC	S1 is 3~48:3 (min<=data len)
Codabar max	Esc,0,4,70,m,S1,BCC	S1 is 3~48:48
Label_code	Esc,0,4,71,m,S1,BCC	S1 is S1.b0 is for Label_c_enable,S1.b1 is for Label code cd
Label_c_enable	Esc,0,4,72,m,S1,BCC	S1 is 0 disable,1 enable
Label_code_cd	Esc,0,4,73,m,S1,BCC	S1 is 0 check digit send,1 not send
Upc_a_set	Esc,0,4,74,m,S1,BCC	S1 is S1.b0 is for Upc_a_enable,S1.b1 is for Upc a ld,S1.b2 is for Upc a cd
Upc_a_enable	Esc,0,4,75,m,S1,BCC	S1 is 0 disable,1 enable
Upc a ld	Esc,0,4,76,m,S1,BCC	S1 is 0 leading digit send,1 not send
Upc a cd	Esc,0,4,77,m,S1,BCC	S1 is 0 check digit send,1 not send
Upc_e_set	Esc,0,4,78,m,S1,BCC	S1 is S1.b1 is for Upc_e_enable,S1.b2 is for Upc_e_ld,S1.b3 is for Upc_e_cd,S1.b4 is for Upc e expand,S1.b0 is for Upc e nsc
Upc_e_enable	Esc,0,4,79,m,S1,BCC	S1 is 0 disable,1 enable
Upc e ld	Esc,0,4,7a,m,S1,BCC	S1 is 0 leading digit send,1 not send
Upc e cd	Esc,0,4,7b,m,S1,BCC	S1 is 0 check digit send,1 not send
Upc_e_expand	Esc,0,4,7c,m,S1,BCC	S1 is 0 zero expansion on,1 off
Upc e nsc	Esc,0,4,7d,m,S1,BCC	S1 is 0 disable,1 enable
Ean_13_set	Esc,0,4,7e,m,S1,BCC	S1 is S1.b0 is for Ean_13_enable,S1.b1 is for Ean_13_ld,S1.b2 is for Ean_13_cd,S1.b3 is for Ean 13 bookland
Ean_13_enable	Esc,0,4,7f,m,S1,BCC	S1 is 0 disable,1 enable
Ean 13 ld	Esc,0,4,80,m,S1,BCC	S1 is 0 leading digit send,1 not send
Ean 13 cd	Esc,0,4,81,m,S1,BCC	S1 is 0 check digit send,1 not send
Ean_13_bookland	Esc,0,4,82,m,S1,BCC	S1 is 0 bookland EAN enable,1 disable
Ean_8_set	Esc,0,4,83,m,S1,BCC	S1 is S1.b0 is for Ean_8_enable,S1.b1 is for Ean 8 ld,S1.b2 is for Ean 8 cd
Ean_8_enable	Esc,0,4,84,m,S1,BCC	S1 is 0 disable,1 enable
Ean 8 ld	Esc,0,4,85,m,S1,BCC	S1 is 0 leading digit send,1 not send
Ean 8 cd	Esc,0,4,86,m,S1,BCC	S1 is 0 check digit send,1 not send
Supplement_set	Esc,0,4,87,m,S1,BCC	S1 is S1.b0 is for Supplement_two, s1.b1 is for Supplement_five,S1.b2 is for Supplement_mh, S1.b3 is for Supplement ssi.
Supplement_two	Esc,0,4,88,m,S1,BCC	S1 is 0_off,1 on
Supplement_five	Esc,0,4,89,m,S1,BCC	S1 is 0_off,1 on
Supplement_mh	Esc,0,4,8a,m,S1,BCC	S1 is 0_transmit if present,1 must present
Supplement_ssi	Esc,0,4,8b,m,S1,BCC	S1 is 0 Space been inserted, 1_Space not been inserted
Delta_code_set	Esc,0,4,8c,m,S1,BCC	S1 is S1.b0 is for Delta_c_enable,S1.b1 is for Delta code cdc,S1.b2 is for Delta code cds
Delta_c_enable	Esc,0,4,8d,m,S1,BCC	S1 is 0 disable,1 enable
Delta_code_cdc	Esc,0,4,8e,m,S1,BCC	S1 is 0 check digit calculate,1 not calculate
Delta_code_cds	Esc,0,4,8f,m,S1,BCC	S1 is =0 check digit send,1 not send
Get version	Esc,0,3,90,2,BCC	Get firmware version.
DumpSetting	Esc,Lh,Ll,91,m,S1...Si,BCC	Lh/Ll is command length. Si is in the range of s1 to S255.m=0 is download setting, m=1 is reset the setting area into FF. m=2 is upload

-

		setting. Actually you just need the format as bellow: Download: Esc,1,02,91,0,s1,...,s255,BCC Upload: Esc,0,3,91,2,BCC
EAN128Brace Remove	Esc,0,4,92,m,S1,BCC	S1 is =0_disable,1 enable(Remove the brace)
AimingTime	Esc,0,4,93,m,S1,BCC	S1 is =0 0.5s,1 1s,2 1.5s 3 2s
Exchange data	Esc,Lh,Ll,a3,S1,S2,....,Sn, BCC	• Expect Acknowledge (Esc,80H+ACK(06H)) • Exchange the data between the host and the ICC. • Expected return after issuing Execute/Enquiry command are: Esc,Lh,Ll,0xa3,AH,data,BCC Here: AH=0 Success =1 Timeout =2 No card present data: Response data and status word
Note: Hamster save these commands to buffer and do not execute until it receives an Execute command (Esc,ENQ). Hamster execute the command after receive an "Esc,ENQ" then send back a reply. The Max. Length of data is 264. The m and the reply define as following:		

DATA TO HOST FROM HAMSTER					
Data format: Code_number,Lh,Ll,string Here: The Lh/Ll is string length, Lh is high byte, Ll is low byte, The string length is excluded the Code_number and Lh/Ll. The string contains the Code ID, pre_amble, scanned data,post_amble, and terminator. Code_number is equal to following number plus H'80.					
0 Code 39 full ASCII	1 Code 39 standard or EDP Code			2 EAN 13	3 UPC A
4 EAN 8	5 UPC E	6 I25	7 Codabar	8 Code 128	9 Code 93
10 S25	11 MSI	12 EAN 128	13 Code 32	14 Delta	15 Label
16 Plessey	17 Code 11	18 Toshiba	19 reserved	20 Track 1	21 Track 2
22 Track 3	23 More than 1 track	24 reserved	25 RS232	26 reserved	27 reserved
28 reserved	29 reserved	30 reserved	31 reserved	32 reserved	33 reserved

-

4. Scanner3.DLL – Backward compatible API for PT930/PT930S's Scanner3.dll

"Scanner3.lib" and "scanner3.h" are necessary files for VC programming to compile application. You can find it from standard LIB and INCLUDE folder after installed SDK.

4.1. Enable Decoder

Function Description: This function will open COM2 port, create a thread to get any barcode input from Decoder Chip, and then store input data in the system buffer. Application can use function call **PT_GetBarcode()** to get input data from the system buffer.

Function call:

INT PT_EnableBarcode(VOID);

Return code:

=1	Create new thread fail
=2	Cannot re-enable
=3	Cannot open COM2
=4	Upload parameter from Hamster fail
=0	OK

4.2. Disable Decoder

Function Description:

This function will close COM2 port and then remove thread which is created by **PT_EnableBarcode()**

Function call:

VOID PT_DisableBarcode(VOID);

4.3. Check barcode input

Function Description:

This function is used to check whether there is available barcode data on system buffer which is successfully decoded by decoder chip.

Function call:

BOOL PT_CheckBarcode(VOID);

Return code:

TRUE = There is input data on system buffer.
FALSE = There is no data on system buffer.

-

4.4. Read barcode data

Function Description: Get input barcode data and its type from system buffer.
Function call: BOOL PT_GetBarcode(TCHAR *szBarcodeBuffer,TCHAR *cType);
Parameter: (output)
szBarcodeBuffer : string buffer for storing input data
cType : Type of Input data
=00H Full Code 39
=01H STD Code 39
=02H EAN-13
=03H UPC-A
=04H EAN-8
=05H UPC-E
=06H I-25
=07H CODABAR
=08H Code 128
=09H Code 93
=0Ah STD 25
=0BH MSI
=0CH EAN-128
=0DH Code 32
=0EH DELTA
=0FH LABEL
=10H PLESSEY
=11H Code 11
=12H TOSHIBA
Return code: TRUE = There is barcode input
FALSE = No Barcode Input

4.5. Get DLL version no

Function description:
This function is used to get DLL version no.
Function call:
INT PT_DllVersion(void);
Return :
Integer

4.6. Reset all symbologies to default

Function Description:
This function call will reset decoder chip's symbologies setting to system default value

Function call for VC:
int PT_ SetToDefault (VOID)

Function call for VB:
PT_ SetToDefault

-

5. ScanKey3.DLL – Backward compatible API for PT930/PT930S's ScanKey3.dll

In Technical Binder CD, you can get this file from folder \Programming\scankey. In this folder can also find extra 3 files.

"Scankey3.lib" Used for VC programming
"Scankey3.h" Used for VC programming

5.1. Enable Decoder

Function Description: This function will open COM2 port, create a thread to get any barcode input from Decoder Chip, and then send scanner data to keyboard buffer. User application can get input data just like standard keyboard input.

Function call for VC: int PT_EnableBarToKey(VOID)

Return code:
=1 Create new thread fail
=2 Can not re-enable
=3 Can not open COM2
=4 Upload parameter from Hamster fail
=0 OK

5.2. Disable Decoder

Function Description: This function will close COM2 port and then remove thread which is created by **PT_EnableBarToKey()**

Function call for VC: VOID PT_DisableBarToKey (VOID)

5.3. Get DLL version no

Function description: This function is used to get DLL version number.

Function call for VC: INT PT_Version(void);
Return : Integer

5.4. Disable laser trigger key

Function Description:
This function only stop trigger key to activate laser beam, so COM2 port is still open. This function call is useful when some fields is only allow keyboard input..

Function call for VC:
int PT_StopScan (VOID)

5.5. Enable laser trigger key

Function Description: This function only stop trigger key to activate laser beam, so COM2 port is still open. This function call is useful when some fields is only allow keyboard input..

Function call for VC: int PT_StartScan (VOID)

5.6. Reset all symbologies to default

Function Description: This function call will reset decoder chip's symbologies setting to system default value

Function call for VC: int PT_ SetToDefault (VOID)

Function call for VB: PT_ SetToDefault

-

6. UnitechAPI.DLL

In HT660/PA96x/PA982, Unitech create UnitechAPI.DLL to provide some special function call which are different from standard Microsoft API. For example, RS232 is defined as host communication port with PC via ActiveSync, so it will automatically envoke ActiveSync program to do communication with PC when RS232 cable is plugged into HT660/PA96x/PA982. However, it will make RS232 port useless if user want to connect HT660/PA96x/PA982 with any device with RS232 interface. RS232Event.DLL provides function call for user to disable ActiveSync function over RS232 port to let user directly control RS232 port.

Unitech also provide several function to enable/disable several system icon and task bar. For WinCE system, it just like Windows OS platform, user can directly tap "Start" button from task bar to setup terminal or execute any application on WinCE terminal, so it mean that operator can change, modify or delete any setting. If system developer don't want operator to do any extra operation beside application, Unitech provide function call to provides ability to disable/enable task bar, keyboard and etc.

You can get demo program from HT660/PA96x/PA982 technical binder zip files from \programming\UnitechAPI

6.1. Disable ActiveSync

Function Description:

After called this function, HT660/PA96x/PA982 will not automatically execute ActiveSync program("repllog.exe") when user plug RS232 cable into HT660/PA96x/PA982.

Function call:

BOOL RS232EventEnable (VOID);

Return code:

=TRUE	OK
=FALSE	Fail

6.2. Enable ActiveSync

Function Description:

After called this function, HT660/PA96x/PA982 will automatically execute ActiveSync program ("repllog.exe") again when user plug RS232 cable into HT660/PA96x/PA982.

Function call:

BOOL RS232EventEnable (LPTSTR);

Parameter (Input):

String buffer and content should be "REPLLOG.EXE". If user assign other program, it will invoke user defined program rather than "REPLLOG.EXE"

Return code:

=1	OK
=2	File not found

6.3. Suspend

Function Description:

After called this function, HT660/PA96x/PA982 will automatically suspend itself.

Function call:

void Suspend (void);

6.4. Disable TaskBar

Function Description:

This function will hide "TaskBar" and it doesn't like "Auto Hide" function which is set from **Start → Settings → TaskBar**. "TaskBar" can not be show again when tap button of LCD screen. It need to execute "Enable_TaskBar()" to enable it again

Function call:

BOOL DisableTaskbar (VOID);

Return code:

-

=TRUE OK
=FALSE Fail

6.5. Enable TaskBar

Function Description:

This function will show “TaskBar” again after ”Disable_TaskBar()” was executed to hide taskbar.

Function call:

BOOL EnableTaskbar (VOID);

Return code:

=TRUE OK
=FALSE Fail

6.6. Disable Desktop

Function Description:

This function will hide all icons on desktop, it mean that any short-cut or files cannot be accessed or executed.

Function call:

BOOL DisableDesktop (VOID);

Return code:

=TRUE OK
=FALSE Fail

6.7. Enable Desktop

Function Description:

This function will show all icons which had already showed on desktop before executed DisableDesktop().

Function call:

BOOL EnsableDesktop (VOID);

Return code:

=TRUE OK
=FALSE Fail

6.8. Disable toolbar on windows explorer

Function Description: This function will hide windows explorer’s toolbar
Function call: BOOL DisableExploreToolbar (VOID);
Return code: =TRUE OK
=FALSE Fail

6.9. Enable toolbar on windows explorer

Function Description:

This function will enable windows explorer’s toolbar again

Function call:

BOOL EnsableExploreToolbar (VOID);

Return code:

=TRUE OK
=FALSE Fail

-

6.10. Disable Connection

Function Description:

This function will disable the specify connection in “Settings\Network and Dial-up Connections”.

Function call:

BOOL DisableConnection (LPTSTR);

Parameter (Input):

Specify the connection name in parameter.

Return code:

=TRUE	Success
=FAULE	Fail

6.11. Enable Connection

Function Description:

This function will enable the specify connection in “Settings\Network and Dial-up Connections”.

Function call:

BOOL EnableConnection (LPTSTR);

Parameter (Input):

Specify the connection name in parameter.

Return code:

=TRUE	Success
=FAULE	Fail

-

7. SysIOAPI.DLL

This DLL provide hardware relative API for user to control scanner, LED, back-light and PC card slot. API functions are provided through DLL to assist programmer to write application for HT660/PA96x/PA982. Two files are essential and provided in SDK, SysIOAPI.LIB and SysIOAPI.H.

7.1. Keypad Related Functions

7.1.1. Get CAPS lock status *(This function call is reserved for OS using, it is not suggested to be used on application if you are not fully understand OS operation behavior)*

Function Description:

To check if CAPS is lock or unlock

Function call:

BOOL GetCapsLock (void)

Return code:

BOOL: TRUE : CAPS lock
FALSE : CAPS unlock

7.1.2. Get SHIFT status *(This function call is reserved for OS using, it is not suggested to be used on application if you are not fully understand OS operation behavior)*

Function Description:

To check if SHIFT key is lock or not

Function call:

BOOL GetShift (void)

Return code:

TRUE : Shift lock
FALSE : Shift unlock

7.1.3. Get keypad type *(This function call is reserved for OS using, it is not suggested to be used on application if you are not fully understand OS operation behavior)*

Function Description:

PA962/PA982 only have 22 keys configuration, PA96x will have two keypad type, 22 keys and 36 keys. In HT660, there is only 36 keys configuration.The following function returns current keypad type.

Function call:

int GetKeypadType (void)

Return code:

0 = no keypad
1 = 22-key keypad (For PA962/PA966/PA982 using)
2 = 36-kwy keypad (For PA966/HT660 using)

-

7.1.4. Disable/enable power button

Function Description:

To enable / disable power button

Function call:

VOID DisablePowerButton (BOOL)

Parameter (Input)

TRUE = Disable power button.
FALSE = Enable power button.

Return code:

None

7.1.5. Set keypad utility input mode

Function Description:

In terminal, there is a utility to emulate full alpha key input, called GetVK. The input mode can be switched by pressing “alpha” key, or by following function.

Function call:

void SetGetVKWorkingMode(int)

Parameter (input)

For HT660 :

0 = normal
1 = lower case.
2 = upper case.

For PA962/PA966/PA982

0 = hide the selection window.
1 = show lower case selection window.
2 = show upper case selection window.

Return code:

None

7.1.6. Get keypad utility input mode (For HT660 only)

Function Description:

This function is used to check alpha key input mode.

Function call:

BYTE GetAlphaKeyWorkingMode(void);

Return code:

0 = **normal** .
1 = **lower case**.
2 = **upper case.**

7.1.7. Check Alpha key is pressing (For PA962/PA966/PA982 only)

Function Description:

This function is used to check if alpha key is pressed or not.

Function call:

BOOL GetKeypadAlphaKeyStatus(void);

Return code:

TRUE = Alpha key is pressed.
FALSE = Alpha key is released.

-

7.2. Scanner Related Functions

To save power, the decoder IC is disabled when scanner is not in use. It can be enabled through USI functions. Following functions are meaningful only if decode IC is enabled.

7.2.1. Enable/Disable Scanner trigger key

Function Description:

This function enables/disables trigger keys.

Function call:

void EnableScannerTrigger(BOOL fOn)

Parameter (Input)

fON: BOOL: TRUE = enable trigger keys.
FALSE = disable trigger keys.

Return code:

7.2.2. Turn on/off Scan Engine

Function Description:

This function emulates trigger keys to turn scan engine on or off. It functions even if trigger keys are disabled.

Function call:

void SetScannerOn(BOOL fON)

Parameter(Input)

fON: BOOL: TRUE = turn scan engine on.
False= turn scan engine off.

Return code: none

7.2.3. Get Trigger keys Status

Function Description:

This function returns enable/disable status of trigger keys.

Function call:

BOOL GetScannerTrigger(void)

Return code:

TRUE = trigger keys are enabled.
FALSE = trigger keys are disabled.

7.2.4. Get Scanner Status

Function Description:

This function returns the status of scan engine, or trigger key.

Function call:

BOOL GetScannerStatus(void)

Return code:

TRUE = scan engine is on, or trigger key is pressed.
FALSE = scan engine is off, or trigger key is released.

-

7.2.5. Control trigger key's key event.

Function Description:

This function is used to inform system if necessary to generate key event for trigger key.

Function call:

Void EnableTriggerKeyEvent (BOOL fON)

Parameter(Input)

fON: BOOL: TRUE = Enable key event.
False= Don't generate key event.

Return code: none

Note:

Trigger key activity will generate an event named EXT("KeybdTriggerChangeEvent"). Fast, repeated event generation may cause some trouble for AP By passing FALSE to this function can prevent upcoming event generation

-

7.2.6. Check Trigger key is pressing

Function Description:

This function is used to check if left or right trigger key is pressed or not.

Function call:

BOOL TriggerKeyStatus(int key);

Parameter(Input)

key: int: LEFT_TRIGGER_KEY : left trigger key
RIGHT_TRIGGER_KEY : right trigger key.

Return code:

TRUE = trigger is pressed.
FALSE = trigger is released.

Example:

```
#define kKeybdTriggerEventName          TEXT("KeybdTriggerChangeEvent")
#define kKeybdAlphaKeyEventName         TEXT("KBDAlphaKeyChangeEvent")
#define LEFT_TRIGGER_KEY   1
#define RIGHT_TRIGGER_KEY 2
gKeyEvents[0] = CreateEvent(NULL, TRUE, FALSE, kKeybdTriggerEventName);
gKeyEvents[1] = CreateEvent(NULL, TRUE, FALSE, kKeybdAlphaKeyEventName);

while (1)
{
        WaitForMultipleObjects(2, gKeyEvents, FALSE, INFINITE);

        TriggerKeyStatus(LEFT_TRIGGER_KEY);
        TriggerKeyStatus(RIGHT_TRIGGER_KEY);
}
```

7.3. LED related function

Function Description:

There are two LEDs above the screen of HT660/PA96x/PA982, red and green LEDs. Only the green LED can be controlled by programmer.

Function call:

void GoodReadLEDOn(BOOL fON)

Parameter(Input)

fON: BOOL: TRUE = turn on LED.
FALSE = turn off green LED.

7.4. Backlight related function

There are two backlight controls, screen backlight and keypad backlight. They are controlled separately. For screen backlight, you can adjust brightness of backlight also.

7.4.1. Screen Backlight Control

Function Description:

This function turns screen backlight on or off.

Function call:

void BacklightOn(BOOL fON)

Parameter(Input)

fON: BOOL: TRUE = turn on screen backlight.
FALSE= turn off backlight.

Return code:

-

7.4.2. Get Screen Backlight Status

Function Description:

This function returns the status of screen backlight.

Function call:

BOOL GetBacklightStatus(void)

Return code:

TRUE = screen backlight is on.
FALSE = screen backlight is off.

7.4.3. Keypad Backlight Control (For PA966/PA962/PA982 only)

Function Description:

This function turns keyoad backlight on or off.

Function call:

void KeypadLightOn(BOOL fON)

Parameter(Input)

fON: BOOL: TRUE = turn on keypad backlight.
FALSE = turn off backlight.

Return code:

7.4.4. Get Keypad Backlight Status (For PA966/PA962/PA982 only)

Function Description:

This function returns the status of keyoad backlight.

Function call:

BOOL GetKeypadLightStatus(void)

Return code:

TRUE = keypad backlight is on.
FALSE = keypad backlight is off.

7.4.5. Screen Backlight Brightness Control

Function Description:

This function adjusts screen backlight brightness.

Function call:

void BrightnessUp(BOOL fup)

Parameters(Input)

Fup: BOOL: TRUE = adjust one step up.
FALSE = adjust one step down.

Return code:

-

7.5. PCMCIA/CF slot related functions

In HT660, it only support CF slot and PA96x/PA982 can support both CF and PCMCIA slot. So, please note that PCMCIA function is not work on following API in this section.

7.5.1. Get physical slot ID

Function Description:

PA96x/PA982 has two PC card slots, slot 0 and slot 1, for PCMCIA and CF. this function return which slot for PCMCIA or CF

Function call:

UINT GetPCMCIASlotID(UINT)

Parameters(Input)

0 = PCMCIA. (For PA962/PA966/PA982 only)
1 = CF.

Return code:

Physical slot ID.

7.5.2. Enable/Disable PCMCIA or CF slot

Function Description:

This function enables/disables PCMCIA or CF slot. PA96x/PA982 assigns physical slot 0 to CF and slot1 to PCMCIA, which is reversed compared with previous products. The following function is kept for compatible reason. It takes the same uSocket value as previous products, but reversed internally.

Function call:

void EnablePCMCIASlot(UINT uSocket, BOOL bEnable)

Parameters(Input)

uSocket:	UINT:	0 = PCMCIA slot. ***(PA962/PA966/PA982 only)***
		1 = Compact flash slot.
bEnable :	BOOL:	TRUE = enable specified slot.
		FALSE = disable specified slot.

7.5.3. Enable/Disable IO slots

Function Description:

This function enables/disables IO slots. It is recommended to use with function GetPCMCIASlotID() for platform independent reason.

Function call:

void EnablePCMCIASlot1(UINT uSocket, BOOL bEnable)

Parameters(Input)

uSocket:	UINT:	slot to be applied.
bEnable :	BOOL:	TRUE = enable specified slot.
		FALSE = disable specified slot.

Example

To disable PCMCIA slot and enable CF slot,

```
#define PCMCIA_SOCKET        0        (PA966/PA962/PA982 only)
#define CF_SOCKET            1
EnablePCMCIASlot1(GetPCMCIASlotID(PCMCIA_SLOT),FALSE);
EnablePCMCIASlot1(GetPCMCIASlotID(CF_SLOT),TRUE);
```

-

7.5.4. Inquire PCMCIA/CF slot status

Function Description:

This function returns PCMCIA/CF slot enable/disable status. Terminal assigns physical slot 0 to CF and slot1 to PCMCIA, which is reversed compared with previous products. The following function is kept for compatible reason. It takes the same uSocket value as previous products, but reversed internally.

Function call:

BOOL GetPCMCIAStatus(UINT uSocket)

Parameters(Input)

uSocket: UINT: 0 = PCMCIA slot.
(PA966/PA962/PA982 only)
1 = Compact flash slot.

Return

bEnable : BOOL: TRUE = Slot is enabled.
FALSE = Slot is disable.

7.5.5. Inquire IO slot status

Function Description:

This function returns slot enable/disable status. It is recommended to use with function GetPCMCIASlotID() for platform independent reason.

Function call:

BOOL GetPCMCIAStatus1(UINT uSocket)

Parameters(Input)

uSocket: UINT: slot to be applied.

Return

bEnable : BOOL: TRUE = specified slot is enabled.
FALSE = specified slot is disable.

Example To check PCMCIA slot status,

```
#define PCMCIA_SOCKET    0          (PA966/PA962/PA982 only)
#define CF_SOCKET        1

if (GetPCMCIAStatus1(GetPCMCIASlotID(PCMCIA_SLOT))) {
}
```

7.5.6. Disable PCMCIA/CF slot when resume

Function Description:

This function will disable the specified slot after resume even though that slot is enabled before suspend..

Function call:

void DisablePCMCIAUponResume(UINT uSocket, BOOL bDisable);

Parameters(Input)

uSocket: UINT: 1 = physical socket 1
0 = for physical socket 0

bDisable: BOOL: TRUE disable on resume
FALSE enable on resume

Return none

-

7.6. Check battery type

Function Description:

Check if HT660/PA96x/PA982 is installed smart battery and battery ID.

Function call:

BYTE GetSmartBatteryID(void);

Return

0	: Not Smart Battery
Other	: Smart battery it

7.7. Enable/Disable LCD screen

Function Description:

Turn on / off LCD screen

Function call:

void PowerOnColorLCD(BOOL fON)

Parameters(Input)

fON:	BOOL:	TRUE	= Power on LCD screen
		FALSE	= Power off LCD screen

Return None

-

8. BlueTooth relative API - BTAPI.DLL

This DLL provide BlueTooth relative API. Two files are essential and provided in SDK, BTAPI.LIB and BTAPI.H.

8.1. Enable/Disable Bluetooth Power status

Function Description:

Enable Bluetooth Module Power ON/OFF

Function call:

void BT_PowerEnable (BOOL bEnable)

Parameter (Input)

bON: BOOL: TRUE = Enable

FALSE = Disable

Return code:

FALSE PowerOFF Module

8.2. Get BT Power status

Function Description:

Get Bluetooth Module Power Status

Function call:

BYTE BT_PowerStatus (void)

Return code:

BYTE: 1 = Bluetooth Module is Power ON
0 = Bluetooth Module is Power OFF

8.3. DLL Version

Function Description:

Get BTAPI.dll Version

Function call:

DWORD BT_DllVersion (void)

Return code:

DWORD: Version number

-

9. *RH767 HF reader*

To programming RH767 HF reader, it need C++ DLL “RDINT.dll” and RDINT.h”. Please get it from below URL.
http://w3.tw.ute.com/pub/cs/sdk/RH767/RH767_HF_SDK.zip

9.1. Get library version

Function Description:

To get the library version.

Function Call:

INT32 RDINTsys_GetAPIVersionString (LPWSTR strVersion);

Parameter:

strVersion: Get the library version.

Return code:

Please refer to section 9.10.

9.2. Connect to RFID reade

Function Description:

To create a connection with the reader before control it.

Function Call:

INT32 RDINTsys_OpenReader (BYTE u8COMPort，UINT32 u32Baudrate，CONST LPTSTR strAccessCode，BYTE u1SecurityMode，UINT32 u32OpenDelayMs，PUINT32 pu32Baudrate)

Parameter:

u8COMPort: The reader’s COM port number (1 – 255)
u32Baudrate: The reader’s baud rate and the defualt is 19200.It supports 9600, 19200, 38400 and 115200.
strAccessCode: The reader’s access code, default is “00000000”
u1SecurityMode: To set use security mode or not.
TURN_ON : Open.
TURN_OFF : Close.
u32OpenDelayMs: The delay time for wait reader initial, we suggest this value is 700.
pu32Baudrate: Receive the current reader’s baud rate, if it is NULL then it will not receive the value.

Return code:

Please refer to section 9.10.

9.3. Select Card type

Function Description:

This API change the reader working type with different card type and this should be called before read the card.

Function Call:

INT32 RDINT_WorkingType (BYTE u8COMPort, BYTE u8Type);

Parameter:

u8COMPort: The reader’s COM port number (1 – 255)
u8Type: WT_ISO14443_TypeA
WT_ISO14443_TypeB
WT_ISO15693
WT_SR176_SRIX4K

Return code:

Please refer to section 9.10.

9.4. Get Reader Information

Function Description:

Get the reader’s serial number and firmware version.

Function Call:

INT32 RDINTv2_ReaderInfo (BYTE u8COMPort, LPBYTE pu8SerialNum, LPBYTE

-

pu8FirmwareVer);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)
pu8SerialNum: Get reader's serial number (Length: READER_SERIAL_LEN)
pu8FirmwareVer: Get reader's firmware version (Length: FIRMWARE_VER_LEN)

Return code:

Please refer to section 9.10.

9.5. Antenna Control

Function Description:

Enable/Disable antenna to save power.

Function Call:

INT32 RDINTv2_AntennaControl (BYTE u8COMPort, BYTE u1OnOff);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)
u1OnOff: TURN_ON
TURN_OFF

Return code:

Please refer to section 9.10.

9.6. Close Reader

Function Description:

To finish controlling the reader.

Function Call:

INT32 RDINTsys_CloseReader (BYTE u8COMPort);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)

Return code:

Please refer to chapter 9.10.

9.7. Set ISO-15693 Inventory Parameter

Function Description:

Set inventory parameter for ISO-15693.

Function Call:

INT32 RDINT_ISO15693AutoInventoryParam (BYTE u8COMPort, BYTE u8Flag, BYTE u8Afi, BYTE u8MaskLen, LPBYTE pu8Mask);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)
u8Flag: It specifies the actions to be performed by the VICC and whether corresponding fields are present or not.
u8Afi: Application Family Identifier parameter.
u8MaskLen: The mask length indicates the number of significant bits of the mask value. It can have any value between 0 and 60 when 16 slots are used and any value between 0 and 64 when 1 slot is used. LSB shall be transmitted first.
pu8Mask: The mask value is contained in an integer number of bytes. LSB shall be transmitted first.

Return code:

Please refer to section 9.10.

9.8. ISO-15693 Inventory

Function Description:

Read ISO-15693 Tag.

Function Call:

INT32 RDINT_ISO15693AutoInventory4Antennas (BYTE u8COMPort, LPBYTE pu8GRLUid);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)

-

pu8GRLUid: Receive the read data's length which in the reader's buffer.

Return code:

Please refer to section 9.10.

9.9. Get Data From Reader

Function Description:

Get the data from the reader's buffer.

Function Call:

INT32 RDINT_GetReturnDataArray (BYTE u8COMPort, BYTE u8Indx, BYTE u8Offset, LPBYTE pu8Data);

Parameter:

u8COMPort: The reader's COM port number (1 – 255)
u8Indx: The start index of pu8Data.
u8Offset: The data's length which get from ISO15693AutoInventory4Antennas
pu8Data: The point which will receive the data from reader. The front four bytes mean which antenna get how many tags, the others are the data.

Return code:

Please refer to section 9.10.

9.10. Error Code

Name	Value	Description
LRSUCCESS	0x00	Successful completion of request
LRSYSTEM	0x01	Unknown error
LRLASTCARD	0x02	Last Card Still Present
LRNOCARD	0x03	Card is not present
LRCTYPE	0x04	Card Type error
LRPARAM	0x05	Request Parameter error
LRACCESS	0x06	Card access error
LRREAD	0x07	Card read error
LRWRITE	0x08	Card write error
LRINCR	0x09	Purse increment error
LRDECR	0x0a	Purse decrement error
LRTRANSFER	0x0b	Purse value transfer error
LRRESTORE	0x0c	Purse restore error
LRPURSE	0x0d	Purse value corrupt
LRMADERR	0x0e	Card Directory error
LRFIXERR	0x0f	Purse fix error
LRFIXED	0x10	Purse found corrupt but fixed
LRNOTOPEN	0x11	Card not open
LRNOFILE	0x12	File not found
LRBADSIZE	0x13	Bad file size
LRABORTED	0x14	Request aborted
LRMANYCARD	0x15	Too many card present
LRFORMAT	0x16	Card format error
LRCREATE	0x17	Card file create error
LRDELETE	0x18	Card file delete error
LRALREADOPEN	0x19	Card has been opened already
LRALREADCLOSED	0x1a	Card has been closed already
LRMSTRKEYLOAD	0x1b	Cannot load master keys
LRAPPKEYLOAD	0x1c	Cannot load application
LRKEYCARD	0x1d	Keycard Error
LRUNFORMAT	0x1e	Card has files on it
LRNOKBDCHAR	0x20	No keyboard character
LRNOTIMPL	0x7f	Function not implemented
LRUNKNOWN	0x80	Unknown error
LRCCRBUSY	0xbb	Reader is busy

-

LRNOINIT	0xff	Reader has not been opened
LRCRDNOTOPEN	0xfa	Card has not been opened
LRINUSE	0xfb	Card in use by another applications
LRAPPLICERR	0xfc	API system error
LRLINKLOST	0xfd	Link to Reader has been lost
LRBADCOMPORT	0xfe	COM port cannot be accessed
LRNOCRYPTBOX	0xf8	Key-Box not found
LRBADAPPACCESS	0xf7	Invalid Application access code
LRNOMAIDFILE	0xf6	Cannot open MAID definition
LRBOXREAD	0xf5	Cannot read from Key-Box
LRBOXWRITE	0xf4	Cannot write to Key-Box
LRBOXNOKEYS	0xf3	No of Keys in Box is zero or
LRSECURE	0xf2	Comms MAC checking failed
LRERRSELREADER	0xf1	Cannot change to selected reader

-

10. RH767 UHF reader

To programming RH767 UHF reader, it need C# DLL “MPR DLL.dll”. Please get it from below URL.
http://w3.tw.ute.com/pub/cs/sdk/RH767/RH767_UHF_SDK.zip

10.1. Class “MPRReader”

This is the main class instantiated by Applications. Manages a single WJ Multi-Protocol Reader. Provides properties and methods for accessing features of the MPR. Talks to the MPR via an MPRComm object. Generates request frame payloads for MPR API commands. Parses response frame payloads from MPR API commands. Fires events when MPR public properties change.

10.2. The Parameter in MPRReader

- byte **ActiveAntenna**
  This parameter to set and get the active antenna on reader and this value should be 0 on RH767.
- byte **TxPower**
  This parameter to set and get the current antenna power and this value should between 18-30.
- int **InvUpdateGap**
  To set and get the time between two inventories. Please set this value to 0 to get good performance.
- TimeSpan **PersistTime**
  How long a tag that has been read will persist in the inventory, without being read, while an inventory is running. If an inventory is stopped, tags do not expire. If a tag is re-read, it will live at least another PersistTime.
- bool **Class0InventoryEnabled**
  Whether to perform EPC Class 0 inventories.
- bool **Class1InventoryEnabled**
  Whether to perform EPC Class 1 inventories.
- bool **Gen2InventoryEnabled**
  Whether to perform EPC UHF Gen2 inventories.
- bool **IsConnected**
  To check the connection with reader.
- bool **InvTimerEnabled**
  To get or set inventory status.

-

10.3. The Parameter in MPRReader

10.3.1. Connect to RFID Reader

Function Description:
To create a connection with the reader before control it.
Function Call:
bool Connect(string SerialPortName, string BaudRate)
Parameter:
SerialPortName: The reader's COM port number (COM1: – COM255:)
BaudRate: The baud rate with the reader, the default is "57600"
Return code:
True : Connect success.
False : Connect fail.

10.3.2. Disconnect with RFID Reader

Function Description:
Close the connection and disable inventory with reader.
Function Call:
void Disconnect()
Parameter:
None.
Return code:
None.

10.3.3. Clear All Tags In The Reader

Function Description:
Remove all tags in the buffer of reader.
Function Call:
void ClearInventory()
Parameter:
None.
Return code:
None.

10.3.4. The Event in MPRReader

- **EventHandler InvTimerEnabledChanged**
  Fired when manufacturing information is read from the reader.
- **TagEventHandler TagAdded**
  Fired when a new tag is added to the inventory.
- **TagEventHandler TagRemoved**
  Fired when a tag expires, i.e. hasn't been read for the persist time period.

-

11. Useful function call – without include SysIOAPI.DLL

Below API maybe useful for you to control HT660/PA96x

11.1.1. Warm-boot. Cold-boot and power off

```
#include        <pkfuncs.h>
        #include        "oemioctl.h"

         //     Warn boot
        KernelIoControl(IOCTL_HAL_REBOOT, NULL, 0, NULL, 0, NULL);

        // Cold boot
        KernelIoControl(IOCTL_COLD_BOOT, NULL, 0, NULL, 0, NULL);

        // Power off
        {
        DWORD dwExtraInfo=0;
        BYTE bScan=0;
        keybd_event( VK_OFF, bScan, KEYEVENTF_SILENT, dwExtraInfo );
        keybd_event( VK_OFF, bScan, KEYEVENTF_KEYUP, dwExtraInfo );
        }
```

-

12. *Get Device ID*

In HT660/PA96x, an unique ID had been burnt into terminal, user can check it by pressing “Func”+”9”.

The sample code for read device ID as follow,

```
//////////////////////////////////////////////////////////////
    HWND hDeviceId = GetDlgItem(hWnd, IDC_DEVICEID);

    PDEVICE_ID pDeviceID = NULL;
    TCHAR outBuf[512], deviceID[200];
    DWORD bytesReturned;
    char platformID[64];

    pDeviceID = (PDEVICE_ID)outBuf;
    pDeviceID->dwSize = sizeof(outBuf);
    if (KernelIoControl(IOCTL_HAL_GET_DEVICEID, NULL, 0, outBuf, sizeof(outBuf), &bytesReturned))
    {
        // Platform ID
        memcpy((PBYTE)platformID, (PBYTE)pDeviceID + pDeviceID->dwPlatformIDOffset, pDeviceID->dwPlatformIDBytes);
        // Device ID
        memcpy((PBYTE)deviceID, (PBYTE)pDeviceID + pDeviceID->dwPresetIDOffset, pDeviceID->dwPresetIDBytes);
    }
//////////////////////////////////////////////////////////
```

The code will have platformID holds Platform ID, and deviceID holds Device ID.

13. *Get OEM Info*

In HT660/PA96x, an OEM ID had been burnt into terminal, user can check it by pressing “Func”+”9”.

The sample code for read OEM ID as follow,

```
//////////////////////////////////////////////////////////////
    {
        TCHAR szBuff[500];

        ZeroMemory(szBuff, sizeof(szBuff));

        SystemParametersInfo(SPI_GETOEMINFO, 500, (LPVOID)szBuff, 0);

        MessageBox(szBuff);
    }
//////////////////////////////////////////////////////////
```

-

14. Update notes

V1.0 The first version
V1.1 Wrong URL link for C# on chapter 1.4
V1.2 PA982 support
V1.3 Add RH767 HF/UHF programming on chapter 9 & 10
V1.4 Modify RH767 HF programming on chapter 9.
V1.6 Change logo
V1.7 Modify SDK URL

-