ARGOX
Empower the Barcode

Portable Data Terminal

PA-20 / PA-2010

PA-21 / PA-2110

Programming Guide

Version: 1.07B

Preface

To satisfy the user's customized needs, the PA-20 / PA-2010 / PA-21 / PA-2110 provide users to generate programs for their actual demands. This allows users to collect data, execute function expression and store the processed data with the application programs designed by their own.

Developers can use ARM Assembly or C code to create the program flow. And developers can also link standard ANSI C function library to meet the demands through executing the functions of input, output, expression and storage using the functions provided by PA-20 / PA-2010 / PA-21 / PA-2110.

Later in this manual, you'll learn how to write, compile and link program, and also how to download renewed codes and test functions via simulation. Finally, this manual will also conclude the function illustration of PA-20 / PA-2010 / PA-21 / PA-2110 for your reference.

Table of Contents

Program Developing

Development Environment

Folder Structure:

When open the SDK folder in the CD provided with the PA-20 / PA-2010 / PA-21 / PA-2110, it will show the structure as the following:
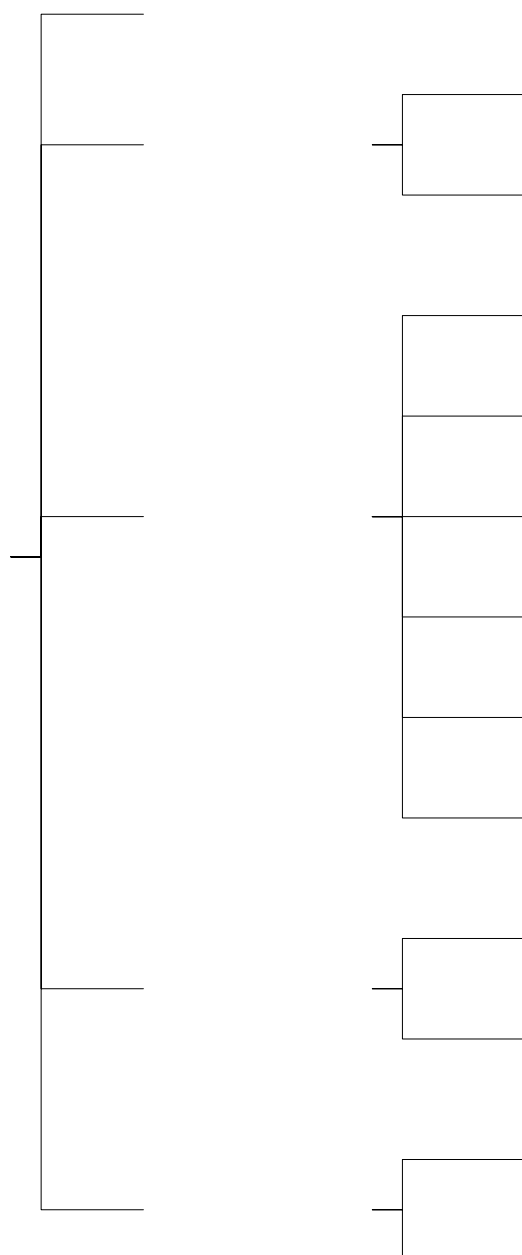
AID MA

Scanner
Upgrad

Folder introduction:

- Upgrade:

  For Fw upgrade, it has two sub folders, "**Ugrade BIN file**" and "**Upgrade BAT file**".

  - ✧ Ugrade BIN file:

    It has a bin file for fw upgrade.

  - ✧ Upgrade BAT file:

    It has a exe file, "DFUSender_PA2.exe", and has three bat files "All Upgrade-COM1.bat", "All Upgrade-COM2.bat", and "All Upgrade-USB.bat".

    If you want to upgrade fw, you have to set PA-20 / PA-2010 / PA-21 / PA-2110
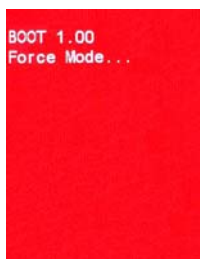
    in force mode or upgrade in supervisor menu
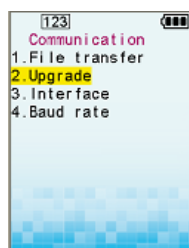
     and double click these bat file for download.

    You can download bin file by PT-FileManager in these status, too.

    Please refer to the "**Upgrade System**" section.

- SDK:

  For SDK develop tools. It has six sub folders, "**Document**", "**PC DLL**", "**Sample Code(For Terminal)**", "**Sample Code(For PC DLL)**", "**Terminal**", and "**Upgrade SDK Tool**".

  - ✧ Document:

    It has two PDF file, "PC DLL.pdf" and "Terminal.pdf", these PDF files are introduct about how to use PC DLL and SDK functions.

  - ✧ PC DLL:

    For PC DLL files.

  - ✧ Sample Code(For Terminal):

    For terminal sample code.

  - ✧ Sample Code(For PC DLL):

    For PC DLL sample code.

  - ✧ Terminal:

For terminal develop environment.

- ✧ Upgrade SDK Tool:

  For upgrade App.bin in Force Mode.

- ■ AID Maker:

  For set PA-20 / PA-2010 / PA-21 / PA-2110 agency ID.

- ■ Scanner FW upgrade:

  For upgrade scanner module fw, if your scanner has something wrong, you can try to upgrade it, Please see "Scanner FW upgrade"

- ■ Utility:

  It has two sub folders, "Font" and "ScanSetting"

  - ✧ Font:

    You can make your font for PA-20 / PA-2010 / PA-21 / PA-2110 to display.

  - ✧ ScanSetting:

    It can make a setting file for scanner, you need use SDK function to set it.(For PA-20 / PA-2010 / PA-21 / PA-2110)

Functions in SDK folder description:

- ■ You have to Install RealView 2.2 in your computer.

- ■ *PA2X_SDK.mcp* is a *RealView Developer Suite* project file, developer need to set *PA2X_SDK.mcp* as RealViews' project file.

- ■ *App.bin* is the binary file ,compiler by RealView 2.2. You can download the binary file by PT-FileManager

- ■ *Source* folder is used to store application files and the PA-20 / PA-2010 / PA-21 / PA-2110 function library used in the programs.

- ■ *Library* folder is used to store PA2X_Lib.a, Startup.s, and other *.inc files for compiler, don't delete this folder.

- ■ *Others* folder is for later use. Don't care this folder.

- ■ *PA2X_SDK_Data* folder is used to store object files.

- ■ *Release* folder is used to store compiler information.

Adding Source File:

All user application program source files has to be placed under *Source* folder, and register needed files' name in the project file of *PA2X_SDK.mcp* before proceeding with compiling and linking process.

Function Library

- PA-20 / PA-2010 / PA-21 / PA-2110 Function Library supports user application program to perform the data collection jobs. PA-20 / PA-2010 / PA-21 / PA-2110 Function Library provides variety of services, and accomplishes special functions according to specific demands.

- When using the PA-20 / PA-2010 / PA-21 / PA-2110 Function Library, please add the import command (#include "SDK.h") into the user program file (*.c) and the function will be imported. In this case, the PA-20 / PA-2010 / PA-21 / PA-2110 Function Library file *PA2X_Lib.a* is needed.

- The PA-20 / PA-2010 / PA-21 / PA-2110 Function Library file *PA2X_Lib.a* is updated occasionally. For most update version, please ask helps from your vendor or the manufacturer.

- PA-20 / PA-2010 / PA-21 / PA-2110 Function Library file *PA2X_Lib.a* is needed during compiling and linking for generating App.bin.

- The update library release is SDK.h. Please refer SDK Library section.

Standard Function Library

- The user application program in the data collector can perform the tasks to combine standard C language function library. The function library is enclosed in the developing environment (RealView Developer Suite). After set up the developing environment, you can find the include head file of standard C language function library in the directory \\ARM\RVCT\Data\2.2\349\include\windows. The following are the available include head file list in standard C language function library:

<assert.h>

    __assert ;

<ctype.h>

    isalnum; isalpha; iscntrl; isdigit; isgraph; islower; ispr; ispunct;

    isspace; isupper; isxdigit; tolower; toupper;

<locale.h>

    setlocale; localeconv;

<math.h>

    acos; asin; atan; atan2; cos; sin; tan; cosh;

    sinh; tanh; exp; frexp; ldexp; log; log10; modf;

    pow; sqrt; ceil; fabs;__d_abs; floor; fmod;

<setjmp.h>

    setjmp; longjmp;

<signal.h>

    signal; raise;

<stdio.h>

    sprintf; sscanf;

<stdlib.h>

    atof; atoi; long atol; strtod; long strtol; strtoul; rand; srand;

    _ANSI_rand; _ANSI_srand; abort; atexit; exit; getenv; system; bsearch;

    qsort; abs; long labs;

<string.h>

    strcpy; strncpy; strcat; strncat; memcmp; strcmp; strncmp; strcoll; strxfrm; strstr;

    memset; strlen;

- If you need to use standard C language functions in the user program, please add #include <header file name> in the top of the file to import the correlated include head files. See following example:

    #include <stdio.h>

#include <stdlib.h>

- The statements listed above will make Compiler and Linker to import all the correlated functions to generate App.bin file.

How to Build Your Program

1. Edit Program:

- Developers may use the *Application.c* file under *Source* folder in the *PA2-T01.00-000 Directory* as the starting file. And you can use *void Application_Main(void)* as the start point to edit the program. And also you can freely create a new source file to proceed structural development.

- For regulations and procedures in the developing procedures, please refer to the "*Development Notice*"

2. Under RealView:

- RealView2.2

    The Development Tool Kit is available from the manufacturer or the suppliers. After installing the Development Tool Kit, run the CodeWarrior for RVDS as figure:
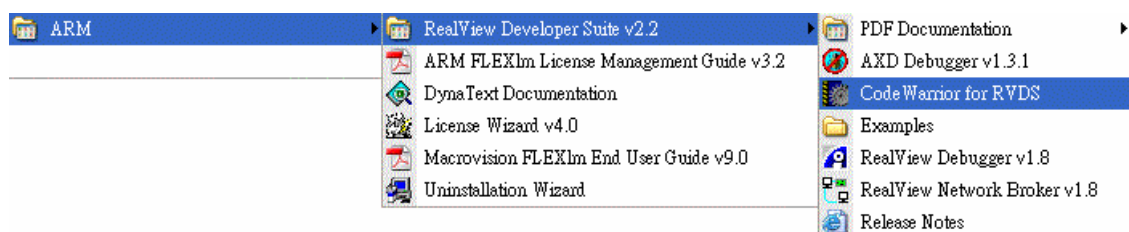
    In RealView CodeWarrior, you can open the RealView project "PA2X_SDK.mcp" by double click as figure:
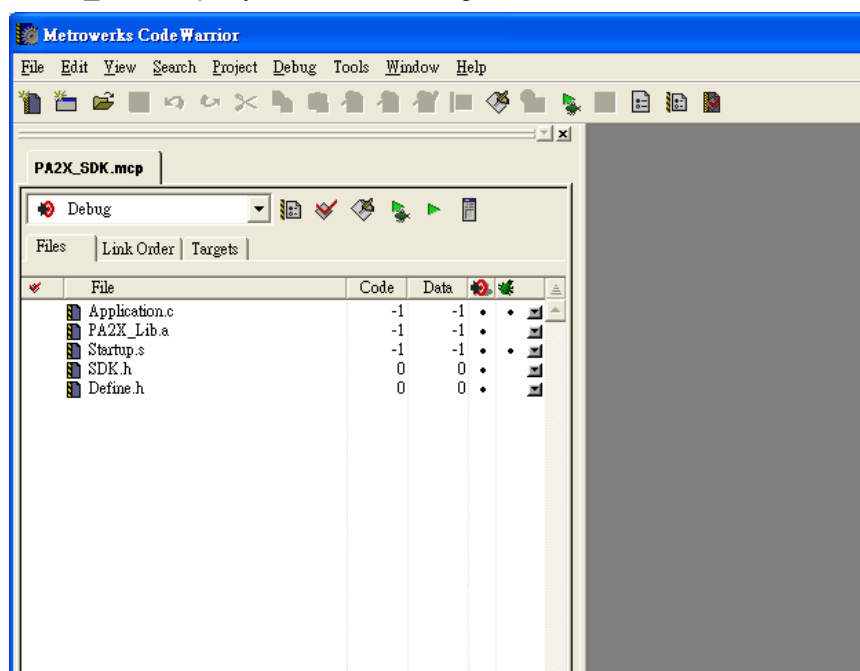
- Add or Delete Files:

    When adding or deleting the source files, you can do the adding and removing directly under the **RealView Developer Suite**.   The files will be displayed both in **Debug** and **Release** file list.

- Compile, Link and Create:

    When finish application code development, you can compile and link application code as figure:
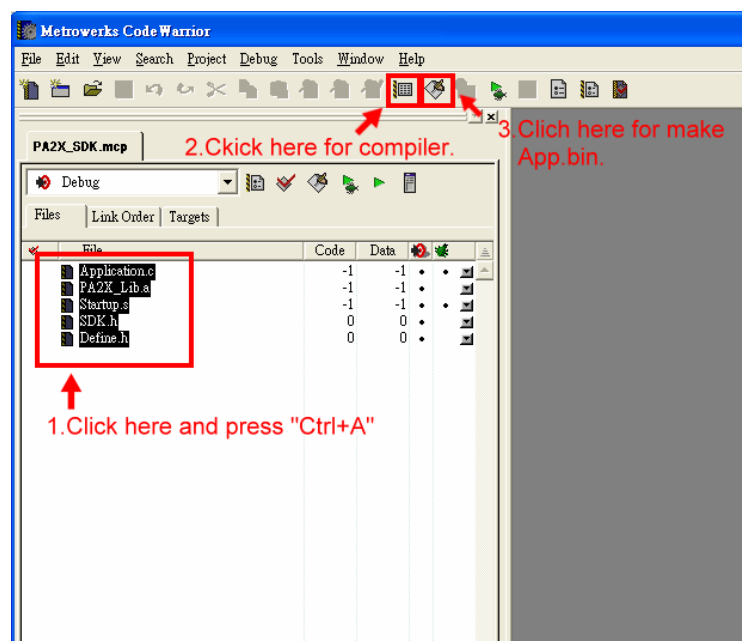
    After thses steps, you can upgrade your system.

3.   Update your System:

Please refer to the "**Upgrade System**" section.

4.   Development Notice:

- **void Application_Main (void)** is the entry program of *Application.c* instead of usual main.c.

- Maximum User Task Stack: 40K bytes

- Maximum global area and Memory allocation(Total): About 22Mbytes.

- Maximum capacity of the Binary file (**App.bin**): About 4MBytes

- System storage:

    ・   Drive C – DDR memory for dynamic access.

    ・   Drive D – NAND flash.

- The developer can exchange files with PC using the communication tool *PT-FileManager*.

Upgrade System

1. System Requirement:

- Software: PT-FileManager
- Hardware: PA-20 / PA-2010 / PA-21 / PA-2110 and PC.
- Firmware: Binary file generated by RealView compiler (App.bin) or Upgrade fw(in Upgrade folder)

2. Upgrade Procedure:

In Force Mode:

- When reset PA-20 / PA-2010 / PA-21 / PA-2110, press combine key "1+3+PW+reset" any time or press combine key "1+3+PW" in power off to into Force

  Mode.
- Then connect the Cradle to the PC and wait for communication.

In Supervisor Menu:

- When reset PA-20 / PA-2010 / PA-21 / PA-2110, press combine key "1+3+0+PW+reset" any time or press combine key "1+3+0+PW" in power off to into Supervisor Menu.
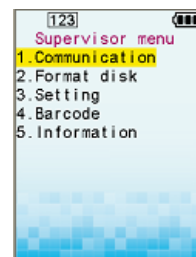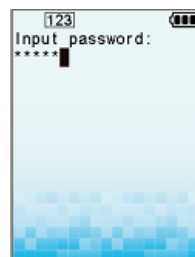
- And input password "00000"   ->Communication
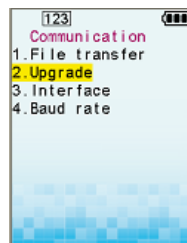
->Upgrade   . Then connect the Cradle to the PC and wait for

communication.

In PTFileManager:

- Execute PTFileManager and select Tool\F/W Update
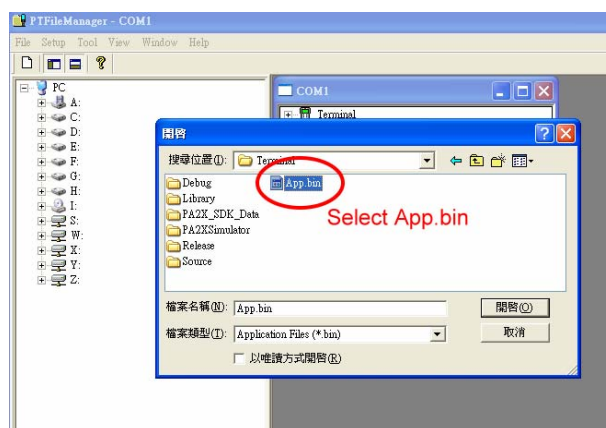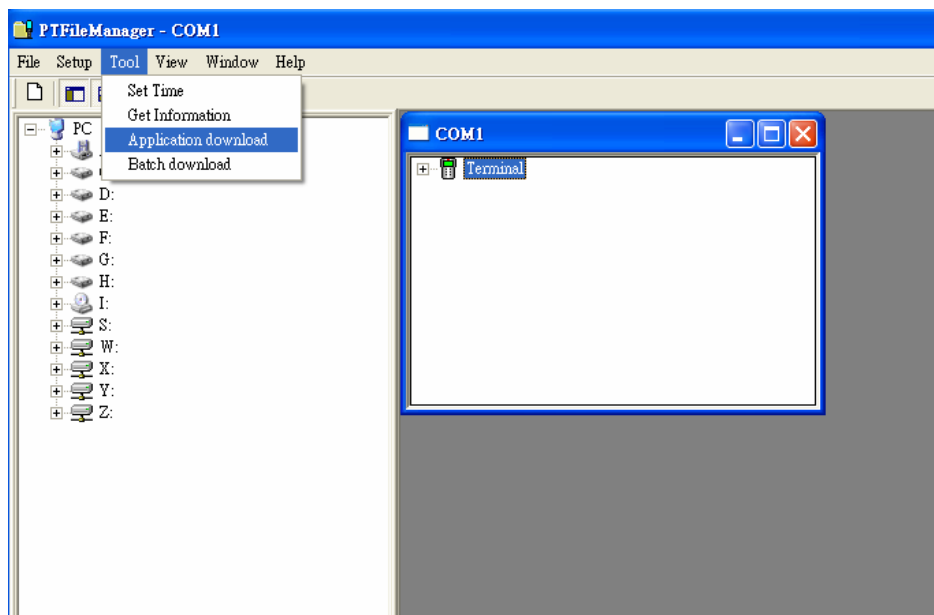
- Select the Binary file and complete the firmware update.

3. Execute System:

- Restart PA-20 / PA-2010 / PA-21 / PA-2110.

Utility & Others

1. AID MAKER

- Select PA-20 / PA-2010 / PA-21 / PA-2110 in communication mode.
- Double click "AgentIDTool.exe" and set connection.
- This application will ask the username and password, each of these word for 4~8 characters.
- After set, you can check the AID by "Check_AID" function in your SDK application.

2. Scanner FW Upgrade

- When scanner cannot work, please use these two files to upgrade scanner fw.
- CCD Version:

    Download "bootall.bin" and "SCN80ALL.bin" to C:\Data in PA-20 / PA-2010 by PTFileManager.

- Laser Version:

    Download "Laser.img" to C:\Data in PA-21 / PA-2110 by PTFileManager.

- After download, go to supervirsor menu, select "Barcode" => "Scanner Fw Upgrade" to upgrade/reset your scanner firmware.

3. Font

This utility "**SDK Tool**" in Font folder can do somethings as follows

- When you need a BMP picture to display, you can make a BMP text by our "SDK Tool". This text file is your BMP file image array, you can copy this array in your code and compiler that.
- When you need a font array in your source code, you can make the font array by "SDK Tool" to make font text.
- When you need a font file for your application, you can make the font file by "SDK Tool", the font generator can help you to make a font file.

4. ScanSetting

This utility "**ScanSetting**" can help you to set scanner setting faster. After you make a scanner setting file, you have to save the file in PA-20 / PA-2010 / PA-21 / PA-2110, and use SDK function "**ScannerSetFromFile**" to set your scanner setting.

SDK Library

SDK Functions list

Function	Description
Reader	
InitScanner1	Initialize respective scanner port.
Decode	Perform barcode decoding.
SleepScanner1	Set scanner module sleep.
HaltScanner1	Stop the scanner port from operating.
TriggerStatus	To check the scan key status.
Scanner_Reset	Set scanner setting to default.
Scanner_Config_Start	To start scanner setting procedure.
Scanner_Config_End	To end scanner setting procedure.
Scan_SendCommand	Send scanner(CCD) command to change scanner status.
SCAN_SendCommand_Laser	Send scanner(Laser) command to change scanner status.
Scan_QueryStatus	Query the scanner(CCD) current setting.
SCAN_QueryStatus_Laser	Query the scanner(Laser) current setting.
ScannerSetFromFile	Set scanner setting by scanner setting file. This file is made by utility "ScanSetting".
Scanner_Version	Query the scan module version.
Sim_ScanKey_Press	To simulator the "Scan" key press or release.
Scanner_All_Symbologies	To enable or disable all barcode symbologies.
Buzzer	
beeper_status	To see whether a beeper sequence is under going or not.
off_beeper	Terminate beeper sequence.
on_beeper	Assign a beeper sequence to instruct beeper action.
SetBuzzerVol	Set the buzzer volume.
Calender	
DayOfWeek	Get the day of the week information.
get_time	Get current date and time.
set_time	Set new date and time to the calendar chip.
Bluetooth(ForPA-2010 / PA-2110)	
BT_Start	Bluetooth module power enable.
BT_Stop	Bluetooth module power disable.

BT_Open	Bluetooth connect.
BT_Close	Bluetooth disconnect.
BT_Read	Read characters from Bluetooth module.
BT_Write	Write characters to Bluetooth module.
BT_GetLocalInfo	Get Bluetooth information.
BT_SetLocalSet	Set Bluetooth information.
BT_Inquiry	Inquiry other Bluetooth module forPA-2010 / PA-2110 to connect.
BT_DisConnectAlert	For bluetooth disconnect alert.

File Manipulation

__access	Check for file existence.
append	Write a specified number of bytes to bottom (end-of-file position) of a DAT file.
appendln	Write a specified number of bytes to bottom (end-of-file position) of a DAT file.
chsize	Extends or truncates a DAT file.
close	Close a DAT file.
delete_top	Remove a specified number of bytes from top beginning-of-file position) of a DAT file.
delete_topln	Remove a null terminated character string from the top (beginning-of-file position) of a DAT file.
eof	Check if file pointer of a DAT file reaches end of file.
filelength	Get file length information of a DAT file.
filelist	Get file directory information.
lseek	Move file pointer of a DAT file to a new position.
open	Open a DAT file and get the file handle of the file for further processing.
read	Read a specified number of bytes from a DAT file.
read_error_code	Get the value of the global variable fErrorCode.
readln	Read a line terminated by a null character "\0" from a DAT file.
_remove	Delete file.
_rename	Change file name of an existing file.
tell	Get file pointer position of a DAT file.
write	Write a specified number of bytes to a DAT file.
writeln	Write a line terminated by a null character (\0) to a DAT file. The null character is also written to the file. After writing in, file position will update.

DiskC_format	Format disk C.
DiskD_format	Format disk D.
DiskC_totalsize	Checking the total space in disk C.
DiskD_totalsize	Checking the total space in disk D.
DiskC_usedsize	Checking the used space in disk C.
DiskD_usedsize	Checking the used space in disk D.
DiskC_freesize	Checking the free space in disk C.
DiskD_freesize	Checking the free space in disk D.
getDirNum	Get the folder quantity in designate path.
getFileNum	Get the file quantity in designate path.
getDirList	Get the folder information in designate path.
getFileList	Get the file information in designate path.
_fclose	Use _fclose to close a file opened earlier for buffered input/output using _fopen.
_fcloseAll	Use _fcloseAll to close all files opened for buffered input/output with _fopen or tmpfile.
_filelength	Use _filelength to dertimine the length of a file in bytes.
_fopen	Use _fopen to open a file for buffered input/output operations.
_fread	Use _fread to read a specified number of data items, eachof a given size, from the current position in a file opened for buffered input. The current position is updated after the read.
_fseek	Use _fseek to move to a new position in a file opened for buffered input/output.
_fwrite	Use _fwrite to write a specified number of data itmes, each of a given size, from a buffer to the current position in a file opened for buffered output. The current position is updated after the write.
DBMS	
Ini_Search	Use "Ini_Search" can initiate the file search function in disk.
Ini_SearchAdv	Use "Ini_SearchAdv" can initiate the advance file search function in disk
Close_Search	Close the file search function in Disk C and D.
SearchField	Search the designated field.
SearchField_GR	Search the designated field; After searching success, acquiring the record which includes this field.
SearchField_GF	Search the designated field; After searching success,

	standard input file and covert the strings to values of C variables according to specified formats.
_scanf_color_DefaultStr	Use _scanf_color_DefaultStr to set a default string in input and read character strings from the standard input file and covert the strings to values of C variables according to specified formats.
_scanf_ctrl_ScannerStatus	Set scanner on/off when use "_scanf_color" function.
_scanf_ctrl_ScannerSleep	Set scanner sleep on/off when use "_scanf_color" function.
_scanf_ctrl_Vibrate	Set vibrate on/off when use "_scanf_color" function and scanner status on.
_scanf_ctrl_ScanWithENT	Set ENT auto press on/off when use "_scanf_color" function and scanner status on.
_scanf_ctrl_AlphaKey	Set Alpha key function on/off when use "_scanf_color" function.
_scanf_ctrl_AlphaKey_Mode	Set alpha mode when use "_scanf_color" function.
_scanf_ctrl_password	Set display for general or user define when use "_scanf_color" function.
_scanf_ctrl_KeypadLock	Set keypad lock on/off when use "_scanf_color" function.

LCD

clr_eol	Clear from where the cursor is to the end of the line. The cursor position is not affected after the operation.
clr_rect	Clear a rectangular area on the LCD display. The cursor position is not affected after the operation.
clr_scr	Clear LCD display.
fill_rect	Fill a white rectangular area on the LCD display.
fill_rect_color	Fill a user define color rectangular area on the LCD display.
Get_Cursor	Get current cursor status.
Set_Cursor	Turn on or off the cursor of the LCD display.
gotoxy	Move cursor to new position.
wherex	Get x-coordinate of the cursor location.
wherexy	Get x-coordinate and y-coordinate of the cursor location
wherey	Get y-coordinate of the cursor location.
lcd_backlit_Setlv	Set LCD backlight level.
lcd_backlit_SetTimer	Set LCD backlight timer.
lcd_backlit_Getlv	Get LCD backlight level.
lcd_backlit_GetTimer	Get LCD backlight timer.
_printf_color	Use _printf_color to write character strings and volues of C variables, formatted in a specified manner, to display

_putchar	Display a character in color black on the LCD display.
_putchar_color	Display a character in user define color on the LCD display.
_puts	Display a string in color black on the LCD display.
_puts_color	Display a string in color black on the LCD display.
show_image_bmp	Put a rectangular bitmap to the LCD display.

UserFont

| DispFont_SetFont | Set user font from font file. |
| DispFont_GetFontInfo | Get font type,width and height. |

TextBlock

SetTextBlockMode	Set TextBlock mode for single or multi layer.
GetTextBlockMode	Get TextBlock mode.
DefineTextBlock	Define TextBlock setting.
SetTextBlock	Enable the specific TextBlock.
ResetTextBlock	Disable the specific TextBlock.
PrintTextBlock	Print Text to specific TextBlock.
GetTextBlockCur	Get TextBlock current position.
SetTextBlockCur	Set specific TextBlock as active TextBlock and set position.
ShowTextBlockCursor	Show or hide TextBlock cursor.
TextBlock_SetBGColor	Set default background color.
TextBlock_SetBGImage	Set default background image for bmp file.
SwitchTextBlock	Switch TextBlock.

Communication Ports

clear_com	Clear receive buffer
close_com	To close specified communication port
com_cts	Get CTS level
com_eot	To see if any COM port transmission in process (End Of Transmission)
com_overrun	See if overrun error occurred
com_rts	Set RTS signal
nwrite_com	Send a specific number of characters out through RS232 port
open_com	Initialize and enable specified RS232 port
read_com	Read 1 byte from the RS232 receive buffer
write_com	Send a string out through RS232 port
USB_Open	Initialize and enable USB port.
USB_Close	To close USB port

USB_Read	Read specific number of bytes from USB port.
USB_Write	Write specific number of bytes to the PC site.
Remote	
SetRemoteBaud	Setting the RemoteLink baud rate.
GetRemoteBaud	Getting the RemoteLink baud rate.
SetRemoteInterface	Setting the RemoteLink interface.
GetRemoteInterface	Getting the RemoteLink interface.
RemoteLink	Use RemoteLink to call the transmission function for user to upload or download files.
RemoteLink_RealTime	Use RemoteLink_RealTime can transfer file in any state.
System	
SysSuspend	Shut down the system.
SysDelay	Set system delay time.
SetPowerOnState	Set power on status
GetPowerOnState	Get power on status
SetAutoPWOff	Set auto power off timer.
GetAutoPWOff	Get auto power off timer.
SetStatusBAR	Set statusbar display/no display.
GetStatausBAR	Get statusbar display status.
SN_Get	To get the SN.
BIOS_SetDefault	Set BIOS setting default.
Check_AID	Check the agency ID correct or not.
Memory	
Tfree	Use the Tfree to release an allocated storage block to the pool of free memory.
Tmalloc	Use Tmalloc to allocate memory for an array of a given number of bytes, not exceeding 200KB.
TotalHeapSize	Checking the total heap size.
UsedHeapSize	Checking the used heap size.
FreeHeapSize	Checking the free heap size.
Vibrate	
on_vibrator	Use on_vibrator to set vibrator on.
off_vibrator	Use off_vibrator to set vibrator off.
set_vibrator_timer	Use set_vibrator_timer to set vibrator on timer.
get_vibrator_timer	Use get_vibrator_timer to get vibrator on. timer
Other	
prc_menu_color	Create a menu-driven interface.
prc_menu_Set_SelectWithEnt	Set function "prc_menu_color" ENT key status when use

number key to select menu.

prc_menu_Get_SelectWithEnt	Get function "prc_menu_color" ENT key status when use number key to select menu.
Simulator (Only for PC Simulator)	
CopyFileToTerminal	Use BackupDataFiletoPC to copy data file to C:\Data directory in PC.
BackupDataFiletoPC	Use BackupDataFiletoPCA to copy data file to any disc in PC.
Sim_SetScannerType	To setting scan module type.
Data Conversion	
__itoa	Use __itoa to convert an integer value to a null-terminated character string.
__ltoa	Use __ltoa to convert a long integer value to a null-terminated character string.
__ultoa	Use __ultoa to convert an unsigned long integer value to a character string.

Reader

InitScanner1

Purpose：	Initialize respective scanner port.
Syntax：	void InitScanner1(void);
Example call：	InitScanner1( );
Includes：	#include "SDK.h "
Description：	Use InitScanner1 function to initialize scanner port. The scanner port won't work unless it is initialized.
Returns：	None

Decode

Purpose：	Perform barcode decoding.
Syntax：	int Decode(void);
Example call：	while(1){if(Decode( )) break;}
Includes：	#include "SDK.h "
Description：	Once the scanner port is initialized (by use of InitScanner1 function), call this Decode function to perform barcode decoding. This function should be called constantly in user's program loops when barcode decoding is required. If the barcode decoding is not required for a long period of time, it is
	recommended that the scanner port should be stopped by use of the HaltScanner1 function. If the Decode function decodes successfully, the decoded data will be placed in the string variable CodeBuf with a string terminating character appended.
	And the code length will be saved in "CodeLen", the code name type will be saved in "CodeName", and the code ID will be saved in CodeID.
	And we have the other buffer for save all barcode information in "FullCodeBuf", the format of "FullCodeBuf" as follows:

Code name	Pream ble	ID*	Code Lengt h	Barcode data	ID*	Posta mble	Termi nator

The ID position depends on "Code ID position" setting.

Returns：	0 : Fail。
	Other value : Barcode length。

SleepScanner1

Purpose：	Set scanner module sleep.
Syntax：	void SleepScanner1(BOOL bStatus);

Example call： InitScanner1( );

while(1)

{

  if (Decode())

    SleepScanner1(TRUE);

  while(_getchar()==0);

  SleepScanner1(FALSE);

}

Includes： #include "SDK.h "

Description： Use SleepScanner1 function to set scanner in sleep mode. You have not to initial scanner again, and it would be scan again.

Returns： None

HaltScanner1

Purpose： Stop the scanner port from operating.

Syntax： void HaltScanner1(void);

Example call： HaltScanner1( );

Includes： #include "SDK.h "

Description： Use HaltScanner1 function to stop scanner port from operating. To restart a halted scanner port, the initialization function, InitScanner1, must be called. It is recommended that the scanner port should be stopped if the barcode decoding is not required for a long period of time.

Returns： none

TriggerStatus

Purpose： To check the scan key status.

Syntax： int TriggerStatus(void);

Example call： if (TriggerStatus( ))

  _printf_color(COLOR_RED, "Scan key pressed!");

Includes： #include "SDK.h "

Description： This function can check the scan key status, if pressed scan key, this function will return 1, else will return 0.

Returns： 0:Scan key is not pressed.

1:Scan key is pressed.

Scanner_Reset

Purpose： Set scanner setting to default.

Syntax： BOOL Scanner_Reset(void)

Example call： If (Scanner_Reset())

  _printf_color(COLOR_RED, "Scan module reset OK!");

Includes： #include "SDK.h "

Description：	This function can reset scan module,if reset OK,this function will return 1,else will return 0.
Returns：	0:Reset fail.
	1:Reset OK.

Scanner_Config_Start

Purpose：	To start scanner setting procedure.
Syntax：	void Scanner_Config_Start(void);
Example call：	Scanner_Config_Start();
Includes：	#include "SDK.h "
Description：	This function can starting scanner setting procedure.
Returns：	None

Scanner_Config_End

Purpose：	To end scanner setting procedure.
Syntax：	void Scanner_Config_End(void);
Example call：	Scanner_Config_End();
Includes：	#include "SDK.h "
Description：	This function can ending scanner setting procedure.
Returns：	None

Scan_SendCommand

Purpose：	Send scanner(CCD) command to change scanner status.
Syntax：	BOOL SCAN_SendCommand(int Command1,int Command2,char *pValue);
Example call：	char ssValue = 0;
	If(SCAN_SendCommand(6,7,&ssValue))
	_printf_color(COLOR_RED, "Setup complete!");
Includes：	#include "SDK.h "
Description：	This function can send command to set scanner status.
	You can see "Appendix 1" to know about the command setting.
Returns：	0:Send fail.
	1:Send OK.

Scan_SendCommand_Laser

Purpose：	Send scanner(Laser) command to change scanner status.
Syntax：	BOOL SCAN_SendCommand_Laser(int Command1,int Command2,char *pValue);
Example call：	char ssValue = 0;
	If(SCAN_SendCommand_Laser(6,7,&ssValue))
	_printf_color(COLOR_RED, "Setup complete!");
Includes：	#include "SDK.h "

Description： This function can send command to set scanner status.

You can see "Appendix 2" to know about the command setting.

Returns： 0:Send fail.

1:Send OK.

Scan_QueryStatus

Purpose： Query the scanner(CCD) current setting.

Syntax： BOOL SCAN_QueryStatus(int Command1,int Command2,char

*pReturn);

Example call： char ssReturn = 0;

if(SCAN_QueryStatus (6,7, &ssReturn))

_printf_color(COLOR_RED, "Query OK!");

Includes： #include "SDK.h "

Description： This function can query scanner setting.

You can see "Appendix 1" to know about the command setting.

Returns： 0:Query fail.

1: Query OK.

Scan_QueryStatus_Laser

Purpose： Query the scanner(Laser) current setting.

Syntax： BOOL SCAN_QueryStatus_Laser (int Command1,int Command2,char

*pReturn);

Example call： char ssReturn = 0;

if(SCAN_QueryStatus_Laser (6,7, &ssReturn))

_printf_color(COLOR_RED, "Query OK!");

Includes： #include "SDK.h "

Description： This function can query scanner setting.

You can see "Appendix 2" to know about the command setting.

Returns： 0:Query fail.

1: Query OK.

ScannerSetFromFile

Purpose： Set scanner setting by scanner setting file. This file is made by utility

"ScanSetting".

Syntax： BOOL ScannerSetFromFile(char *pssFilePath);

Example call： If(ScannerSetFromFile ("C:\\data\\scan.axs"))

_printf_color(COLOR_RED, "Setting OK!");

Includes： #include "SDK.h "

Description： You can set scanner from "scanner setting file" by this function. This

function can help you set scanner setting easier.

Returns： 0:Load fail.

1: Load OK.

Scanner_Version

Purpose：	Query the scan module version.
Syntax：	BOOL Scanner_Version(char* Returnbuf);
Example call：	If(Scanner_Version (Returnbuf))
	_printf_color(COLOR_RED, "Query module version OK!");
Includes：	#include "SDK.h "
Description：	This function can query the scan module version.
Returns：	0:Query module fail.
	1: Query module OK.

Sim_ScanKey_Press

Purpose：	To simulator the "Scan" key press or release.
Syntax：	void Sim_ScanKey_Press(BOOL bStatus)
Example call：	Sim_ScanKey_Press(TRUE);//Set the scan key pressed.
	Sim_ScanKey_Press(FALSE);//Set the scan key released.
Includes：	#include "SDK.h "
Description：	This function can simulator the scan key status for pressed or released.
Returns：	None

Scanner_All_Symbologies

Purpose：	To enable or disable all barcode symbologies.
Syntax：	BOOL Scanner_All_Symbologies(BOOL bState)
Example call：	Scanner_All_Symbologies(TRUE);//Enable all barcode symbologies
	Scanner_All_Symbologies(FALSE);//Disable all barcode symbologies
Includes：	#include "SDK.h "
Description：	This function can Enable or Disable all barcode symbologies.
Returns：	0:Setting fail.
	1:Setting OK.

Buzzer

beeper_status

Purpose：	To see whether a beeper sequence is under going or not.
Syntax：	int beeper_status(void);
Example call：	while(beeper_status( ));
Includes：	#include "SDK.h "
Description：	The beeper_status function checks if there is a beeper sequence in progress.
Returns：	1 if beeper sequence still in progress, 0 otherwise

off_beeper

Purpose：	Terminate beeper sequence.
Syntax：	void off_beeper(void);
Example call：	off_beeper( );
Includes：	#include "SDK.h "
Description：	The off_beeper function terminates beeper sequence immediately if there is a beeper sequence in progress.
Returns：	none

on_beeper

Purpose：	Assign a beeper sequence to instruct beeper action.
Syntax：	void on_beeper(int *sequence);
Example call：	int beep_twice[50] = {30,10,0,10,30,10,0,0};
	on_beeper(beep_twice);
Includes：	#include "SDK.h "
Description：	A beep frequency is an integer used to specify the frequency (tone) when the beeper activates. The actual frequency that the beeper activates is not the value specified to the beep frequency. It is calculated by the following formula.
	Beep Frequency = 76000 / Actual Frequency Desired
	For instance, to get a frequency of 2000Hz, the value of beep frequency should be 38. If no sound is desired (pause), the beep frequency should beset to 0. A beep with frequency 0 does not terminate the beeper sequence. Suitable frequency for the beeper ranges from 1 to 2700Hz, where peak at 2000Hz.
Returns：	The on_beeper function has no return value.

SetBuzzerVol

Purpose： Set the buzzer volume.

Syntax： void SetBuzzerVol(int slVol);

Example call： SetBuzzerVol(0);//Buzzer close.

Includes： #include "SDK.h "

Description： The SetBuzzerVol function can set the buzzer volume.

slVol	Buzzer vloume
0	close
1	Low
2	Medium
3	High

Returns： None.

Calender

DayOfWeek

Purpose： Get the day of the week information.

Syntax： int DayOfWeek(void);

Example call： day=DayOfWeek( );

Includes： #include "SDK.h "

Description： The DayOfWeek function returns the day of week information based on current date.

Returns： The DayOfWeek function returns an integer indicating the day of week information. A value of 1 to 6 represents Monday to Saturday accordingly. And a value of 7 indicates Sunday.

get_time

Purpose： Get current date and time

Syntax： int get_time(char *cur_time);

Example call： char system_time[16];

get_time(system_time);

Includes： #include "SDK.h "

Description： The get_time function reads current date and time from the calendar chip and copies them to a character array specified in the argument cur_time. The character array cur_time allocated must have a minimum of 15 bytes to accommodate the date, time, and the string terminator. The format of the system date and time is listed below.

"YYYYMMDDhhmmss"

YYYY	year, 4 digits
MM	month, 2 digits
DD	day, 2 digits
hh	hour, 2 digits
mm	minute, 2 digits
ss	second, 2 digits

Returns： Normally the get_time function always returns an integer value of 0. If the calendar chip malfunctions, the get_time function will then return 1 to indicate error.

set_time

Purpose： Set new date and time to the calendar chip.

Syntax： int set_time(char *new_time);

Example call： set_time("20030401223035");

Includes： #include "SDK.h "

Description： The set_time function set a new system date and time specified in the argument new_time to the calendar chip. The character string new_time must have the following format,

"YYYYMMDDhhmmss"

YYYY	year, 4 digits
MM	month, 2 digits, 1-12
DD	day, 2 digits, 1-31
hh	hour, 2 digits, 0-23
mm	minute, 2 digits, 0-59
ss	second, 2 digits, 0-59

Ps. When it execute in simulator, the time will not change.

Returns： Normally the set_time function always returns an integer value of 1. If the calendar chip malfunctions, the set_time function will then return 0 to 0 error. Also, if the format is illegal (e.g. set hour to 25), the operation is simply denied and the time is not changed.

Bluetooth(For PA-2010 / PA-2110)

These functions only for PA-2010 / PA-2110, and our Bluetooth module only support SPP mode(Serial Port Profile).

BT_Start	
Purpose：	Bluetooth module power enable.
Syntax：	int BT_Start(void);
Example call：	BT_Start();
Includes：	#include "SDK.h "
Description：	This function can enable Bluetooth module power. After use this function, the left led will flash blue light.If you want to use other Bluetooth function, you must run this function first.
Returns：	BT_NOMODULE : The terminal is notPA-2010 / PA-2110.
	BT_START : Bluetooth module has been power enable, please run BT_Stop.
	BT_OK : Bluetooth module power enable.

BT_Stop	
Purpose：	Bluetooth module power disable.
Syntax：	int BT_Stop(void);
Example call：	BT_Stop();
Includes：	#include "SDK.h "
Description：	This function can disable Bluetooth module power. Each Bluetooth function will return "BT_NOSTART" after use this function, and the left led will stop flashing.
Returns：	BT_NOMODULE : The terminal is notPA-2010 / PA-2110.
	BT_NOSTART : Bluetooth module power disable, please run BT_Start.
	BT_OK : Bluetooth module power disable.

BT_Open	
Purpose：	Bluetooth connect.
Syntax：	int BT_Open(void);
Example call：	BT_Open();
Includes：	#include "SDK.h "
Description：	This function can connect to other Bluetooth device.Befor use this function, you have to set the target Bluetooth MAC address by using "BT_SetLocalSet" function.
Returns：	BT_NOMODULE : The terminal is notPA-2010 / PA-2110.
	BT_NOSTART : Bluetooth module power disable, please run BT_Start.

BT_CONNECT : Bluetooth has connected, please run BT_Close.

BT_OK : Bluetooth connect ok.

BT_Close

Purpose：	Bluetooth disconnect.
Syntax：	int BT_Close(void);
Example call：	BT_Close();
Includes：	#include "SDK.h "
Description：	This function can disconnect Bluetooth. If you want to disconnect, you can use BT_Stop(Bluetooth module power off) or this function.
Returns：	BT_NOMODULE : The terminal is notPA-2010 / PA-2110.
	BT_NOSTART : Bluetooth module power disable, please run BT_Start.
	BT_OK : Bluetooth disconnect ok.

BT_Read

Purpose：	Read characters from Bluetooth module.
Syntax：	int BT_Read(char* pssBuf, int slReadSize, int* pslGetSize);
Example call：	int slGetSize;
	char assBuf[20];
	BT_Read(assBuf, 10, &slGetSize);
Includes：	#include "SDK.h "
Description：	If Bluetooth is connected, this function can read characters from Bluetooth module.The parameter "slReadSize" can set how many characters you will read, and the parameter "pslGetSize" will tell you how many characters you read.
Returns：	BT_NOMODULE : The terminal is notPA-2010 / PA-2110.
	BT_NOSTART : Bluetooth module power disable, please run BT_Start.
	BT_DISCONNECT : Bluetooth not connect to other bluetooth device, please run BT_Open.
	BT_ERROR : Parameter error, please check your parmeter.
	BT_OK :Read OK.

BT_Write

Purpose：	Write characters to Bluetooth module.
Syntax：	int BT_Write(char* pssBuf, int slWriteSize, int* pslPutSize);
Example call：	int slWriteSize;
	char assBuf[20] = "1234567890";
	BT_Write(assBuf, 10, & slWriteSize);
Includes：	#include "SDK.h "
Description：	If Bluetooth is connected, this function can wriet characters to other Bluetooth device.The parameter "slWriteSize" tell the function how many

characters will be writed to other Bluetooth device, and "pslPutSize" will
tell you how meny characters send to other Bluetooth device.

Returns： BT_NOMODULE : The terminal is notPA-2010 / PA-2110.

BT_NOSTART : Bluetooth module power disable, please run BT_Start.

BT_DISCONNECT : Bluetooth not connect to other bluetooth device,
please run BT_Open.

BT_ERROR : Parameter error, please check your parmeter.

BT_OK :Write OK.

BT_GetLocalInfo

Purpose： Get Bluetooth information.

Syntax： int BT_GetLocalInfo(_BT_INFO* stInfo);

Example call： _BT_INFO stInfo;

BT_GetLocalInfo(&stInfo);

Includes： #include "SDK.h "

Description： This function will return thePA-2010 / PA-2110 Bluetooth information in
structure.

The structure describe as follows:

typedef struct __BT_INFO

{

    char assLocalAddress[16];   //PA-2010 / PA-2110 Bluetooth MAC
address.(Cannot change.)

    char assLocalName[20];   //PA-2010 / PA-2110 Bluetooth device
name

    BOOL bLocalSecurity;   //PA-2010 / PA-2110 Bluetooth
security mode, set TRUE(on) or
FALSE(off)

    BOOL bLocalEncryption;   //PA-2010 / PA-2110 Bluetooth
encryption mode, set TRUE(on) or
FALSE(off)

    int stInquiryTimeout;   //PA-2010 / PA-2110 Bluetooth inquiry
timeout set, the value from 1(1.28
seconds) to 48(61.44 seconds).

    int stInquirytMaxResponse;   //PA-2010 / PA-2110 Bluetooth inquiry
max response, the value from 1 to 9.

    char assLinkAddress[16];   //Set link device address.

char assPinCode[20];　　　//Set PIN code.

　　　　}_BT_INFO;

Returns：　BT_NOMODULE : The terminal is notPA-2010 / PA-2110.

　　　　BT_OK : Get information OK.

BT_SetLocalSet

Purpose：　Set Bluetooth information.

Syntax：　int BT_SetLocalSet(_BT_INFO* stSet);

Example call：　_BT_INFO stInfo;

　　　　BT_GetLocalInfo(&stInfo);

　　　　stInfo. stInquiryTimeout = 10;

　　　　stInfo. stInquirytMaxResponse = 5;

　　　　BT_SetLocalSet(&stSet);

Includes：　#include "SDK.h"

Description：　This function can help you to set our Bluetooth module parameters.

　　　　The structure describe as follows:

　　　　typedef struct __BT_INFO

　　　　{

　　　　　　char assLocalAddress[16];　//PA-2010 / PA-2110 Bluetooth MAC address.(Cannot change.)

　　　　　　char assLocalName[20];　//PA-2010 / PA-2110 Bluetooth device name

　　　　　　BOOL bLocalSecurity;　//PA-2010 / PA-2110 Bluetooth security mode, set TRUE(on) or FALSE(off)

　　　　　　BOOL bLocalEncryption;　//PA-2010 / PA-2110 Bluetooth encryption mode, set TRUE(on) or FALSE(off)

　　　　　　int stInquiryTimeout;　//PA-2010 / PA-2110 Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).

　　　　　　int stInquirytMaxResponse;　//PA-2010 / PA-2110 Bluetooth inquiry max response, the value from 1 to 9.

　　　　　　char assLinkAddress[16];　//Set link device address.

　　　　　　char assPinCode[20];　//Set PIN code.

　　　　}_BT_INFO;

Returns：　BT_NOMODULE : The terminal is notPA-2010 / PA-2110.

BT_NOSTART : Bluetooth module power disable, please run BT_Start.

BT_CONNECT : Bluetooth has connected, please run BT_Close.

BT_ERROR : Parameter error, please check your parmeter.

BT_TIMEOUT : Set fail, please Set again.

BT_OK : Set OK.

BT_Inquiry

Purpose：　Inquiry other Bluetooth module forPA-2010 / PA-2110 to connect.

Syntax：　int BT_Inquiry(_BT_DEVINFO* stBT_DevInfo, BOOL bGetDevName);

Example call：　_BT_DEVINFO stBT_DevInfo;

BT_Inquiry(&stBT_DevInfo, TRUE);

Includes：　#include "SDK.h"

Description：　This function can search other Bluetooth device forPA-2010 / PA-2110 to

connect, the struct "stBT_DevInfo" will return how many device are found

and these devices MAC address. The parameter "bGetDevName" will tell

the "BT_Inquiry" function that will return device name or not.If

bGetDevName is "TRUE", then "BT_Inquiry" will return device name in

struct "stBT_DevInfo".

Returns：　BT_NOMODULE : The terminal is notPA-2010 / PA-2110.

BT_NOSTART : Bluetooth module power disable, please run BT_Start.

BT_CONNECT : Bluetooth has connected, please run BT_Close.

BT_TIMEOUT : Search fail, please search again.

BT_OK :Search OK.

BT_DisConnectAlert

Purpose：　For bluetooth disconnect alert.

Syntax：　Void BT_DisConnectAlert(BOOL bBeep, int slTimeGap);

Example call：　BT_DisConnectAlert(TRUE, 0);//Only beep one time for disconnect.

Includes：　#include "SDK.h"

Description：　This function can prompt you that bluetooth disconnect after connect.

If bBeep is TRUE, the buzzer will sound for disconnect after connect,

else, the buzzer will do nothing for disconnect after connect.

And, if bBeep is TRUE, the slTimeGap will be set for buzzer sound time

gap. For example, if slTimeGap is 2, the buzzer will sound for each 2

seconds.

Returns：　None.

File Manipulation

__access

Purpose：	Check for file existence.
Syntax：	int __access(char *filename);
Example call：	if(__access("C:\\data\\store.dat") _puts("store.dat exist!!");
Includes：	#include "SDK.h "
Description：	Check if the file specified by filename.
Returns：	If the file specified by filename exist, access returns an integer value of 1, 0 otherwise. In case of error, access will return an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and theirinterpretation are listed below.
fErrorCode：	1: filename is a NULL string.

append

Purpose：	Write a specified number of bytes to bottom (end-of-file position) of a DAT file.
Syntax：	int append(int fd, char *buffer, int count);
Example call：	append(fd,"ABCDE",5);
Includes：	#include "SDK.h "
Description：	The append function writes the number of bytes specified in the argument count from the character array buffer to the bottom of a DAT file whose file handle is fd. Writing of data starts at the end-of-file position of the file, and the file pointer position is unaffected by the operation. The append function will automatically extend the file size of the file to hold the data written.
Returns：	The append function returns the number of bytes actually written to the file. In case of error, append returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	2 File specified by fd does not exist.
	8 File not opened
	9 The value of count is negative.
	10 No more free file space for file extension.

appendln

Purpose：	Write a null terminated character string to the bottom (end-of-file position) of a DAT file.
Syntax：	int appendln(int fd, char *buffer);
Example call：	appendln(fd, data_buffer);
Includes：	#include "SDK.h "
Description：	The appendln function writes a null terminated character string from the character array buffer to a DAT file whose file handle is fd. Characters are written to the file until a null character (\0) is encountered. The null character is also written to the file. Writing of data starts at the end-of-file position. The file pointer position is unaffected by the operation. The appendln function will automatically extend the file size of the file to hold the data written.
Returns：	The appendln function returns the number of bytes actually written to the file (includes the null character). In case of error, appendln returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	2:File specified by fd does not exist.
	8:File not opened
	10:No more free file space for file extension.
	11:Can not find string treminator in buf.

chsize

Purpose：	Extends or truncates a DAT file.
Syntax：	int chsize(int fd, long new_size);
Example call：	if (chsize(fd, 0)) _puts("file truncated!\n");
Includes：	#include "SDK.h "
Description：	The chsize function truncates or extends the file specified by the argument fd to match the new file length in bytes given in the argument new_size. If the file is truncated, all data beyond the new file size will be lost. If the file is extended, no initial value is filled to the newly extended area.
Returns：	If chsize successfully changes the file size of the specified DAT file, it returns an integer value of 1. In case of error, chsize will return an integer value of 0 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.

fErrorCode： 2:File specified by fd does not exist.

8:File not opened

10:No more free file space for file extension.

close

Purpose： Close a DAT file.

Syntax： int close(int fd);

Example call： If (close(fd)) _puts("file closed!\n");

Includes： #include "SDK.h "

Description： Close a previously opened or created DAT file whose file handle is fd.

Returns： close returns an integer value of 1 to indicate success. In case of error, close returns an integer value of 0 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.

fErrorCode： 2:File specified by fd does not exist.

8:File not opened

delete_top

Purpose： Remove a specified number of bytes from top (beginning-of-file position) of a DAT file.

Syntax： int delete_top(int fd, int count);

Example call： delete_top(fd,100);

Includes： #include "SDK.h "

Description： The delete_top function removes the number of bytes specified in the argument count from a DAT file whose file handle is fd. Removing of data starts at the beginning-of-file position of the file. The file pointer position is adjusted accordingly by the operation. For instance, if initially the file pointer points to the tenth character, after removing 8 character from the file, the new file pointer will points to the second character of the file. The delete_top function will resize the file size automatically.

Returns： The delete_top function returns the number of bytes actually removed from the file. In case of error, delete_top returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation.

fErrorCode： 2:File specified by fd does not exist.

8:File not opened

9:The value of count is negative.

10:No more free file space for file extension.

delete_topln

Purpose：	Remove a null terminated character string from the top (beginning-of-file position) of a DAT file.
Syntax：	int delete_topln(int fd);
Example call：	delete_topln (fd);
Includes：	#include "SDK.h "
Description：	The delete_topln function removes a line terminated by a null character file until a null character (\0) or end-of-file is encountered. The null character is also removed from the file. Removing of data starts at the top (beginning-of-file position) of the file, and the file pointer position is adjusted accordingly. The delete_topln function will resize the file size automatically.
Returns：	The delete_topln function returns the number of bytes actually removed from the file (includes the null character). In case of error, delete_topln returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	2:File specified by fd does not exist.
	8:File not opened
	9:The value of count is negative.
	10:No more free file space for file extension.

eof

Purpose：	Check if file pointer of a DAT file reaches end of file.
Syntax：	int eof(int fd);
Example call：	if (eof(fd)) _puts("end of file reached!\n");
Includes：	#include "SDK.h "
Description：	The eof function checks if the file pointer of the DAT file whose file handle is specified in the argument fd, points to end-of-file.
Returns：	The eof function returns an integer value of 1 to indicate an end-of-file and a 0 when not. In case of error, eof returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered.
fErrorCode：	2:File specified by DBF_fd does not exist.
	8:File not opened

filelength

Purpose：	Get file length information of a DAT file.
Syntax：	long filelength(int fd);
Example call：	datasize = filelength(fd);
Includes：	#include "SDK.h "
Description：	The filelength function returns the size in number of bytes of the DAT file whose file handle is specified in the argument fd.
Returns：	The long integer value returned by filelength is the size of the DAT file in number of bytes. In case of error, filelength returns a long value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation.
fErrorCode：	2:File specified by fd does not exist. 8:File not opened

filelist

Purpose：	Get file directory information.
Syntax：	int filelist(char * file_list);
Example call：	total_file = filelist(file_list);
Includes：	#include "SDK.h "
Description：	The filelist function copies the file name, file type, and file size information (separated by a blank character) of all files in existence into a character array specified in the argument dir. When char * file_list = NULL，it will pass the length that the file string needs back. For example, if there are two files in "C:\Data", the filename are StoreIn.dat and StoreOut, and their filesize are 100bytes and 150bytes, the data in the return buffer is "C:\Data\StoreIn.dat dat 100 C:\Data\StoreOut.dat dat 150"
Returns：	When "char*file_list" is NULL, it will pass the size of memory back. When "char*file_list" is not NULL, it will pass the quantity of file back.
fErrorCode：	None

lseek

Purpose：	Move file pointer of a DAT file to a new position.
Syntax：	long lseek(int fd, long offset, int origin);
Example call：	lseek (fd, 512, 0);
Includes：	#include "SDK.h "

Description： The lseek function moves the file pointer of a DAT file whose file handle is specified in the argument fd to a new position within the file. The new position is specified with an offset byte address to a specific origin. The offset byte address is specified in the argument offset which is a long integer. There are 3 possible values for the argument origin.

The values and their interpretations are listed below.

Value of origin	Interpretation
1	beginning of file
0	current file pointer position
-1	end of file

Returns： When successful, lseek returns the new byte offset address of the file pointer from the beginning of file. In case of error, lseek returns a long value of -1L and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.

fErrorCode： 2:File specified by fd does not exist.

9:Illegal offset value.

10:Illegal origin value.

15:New position is beyond end-of-file.

open

Purpose： Open a DAT file and get the file handle of the file for further processing.

Syntax： int open(char *filename);

Example call： if (fd = open("C:\\data\\store.dat")>0)

_puts("store.dat opened!");

Includes： #include "SDK.h "

Description： The open function opens a DAT file specified by filename and gets the file handle of the file. A file handle is a positive integer (excludes 0) used to identify the file for subsequent file manipulations on the file. If the file specified by filename does not exist, it will be created first. If filename exceeds 8 characters, it will be truncated to 8 characters long. After the file is opened, the file pointer points to the beginning of file.

Returns： If open successfully opens the file, it returns the file handle of the file being opened. In case of error, open will return an integer value of -1

and an error code is set to the global variable

fErrorCode to indicate the error condition encountered. Possible

error codes and their interpretation are listed below.

fErrorCode： 1:filename is a NULL string.

6:Can't create file. Because the maximum number of files allowed in the

system is exceeded.

read

Purpose： Read a specified number of bytes from a DAT file.

Syntax： int read(int fd, char *buffer, unsigned count);

Example call： if ((bytes_read = read(fd,buffer,50)) = = -1)

_puts("read error!");

Includes： #include "SDK.h "

Description： The read function copies the number of bytes specified in the

argument count from the DAT file whose file handle is fd to the

array of characters buffer. Reading starts at the current position of

the file pointer, which is incremented accordingly when the

operation is completed.

Returns： The read function returns the number of bytes actually read from

the file. In case of error, read returns an integer value of -1 and an

error code is set to the global variable fErrorCode to indicate the

error condition encountered. Possible error codes and their

interpretation are listed below.

fErrorCode： 2:File handle is NULL.

7:fd is not a file handle of a previously opened file.

read_error_code

Purpose： Get the value of the global variable fErrorCode.

Syntax： int read_error_code( );

Example call： if (read_error_code( ) = = 2) _puts("File not exist!");

Includes： #include "SDK.h "

Description： The read_error_code function gets the value of the global variable

fErrorCode and returns the value to the calling program. The

programmer can use this function to get the error code of the file

manipulation routine previously called. However, the global

variable fErrorCode can be directly accessed without making a

call to this function.

Returns： The read_error_code function returns the value of the global

variable fErrorCode.

fErrorCode： None

readln

Purpose：	Read a line terminated by a null character "\0" from a DAT file.
Syntax：	int readln(int fd, char *buffer, unsigned max_count);
Example call：	readln(fd, buffer, 50);
Includes：	#include "SDK.h "
Description：	The readln function reads a line from the DAT file whose file handle is fd and stores the characters in the character array buffer. Characters are read until end-of-file encountered, a null character (\0) encountered, or the total number of characters read equals the number specified in max_count. The readln function then returns the number of bytes actually read from the file. The null character (\0) is also counted if read. If the readln function completes its operation not because a null character is read, there will be no null character stored in buffer. Reading starts at the current position of the file pointer, which is incremented accordingly when the operation is completed.
Returns：	The readln function returns the number of bytes actually read from the file (includes the null character if read). In case of error, readln returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	2:File handle is NULL.
	7:fd is not a file handle of a previously opened file.

_remove

Purpose：	Delete file.
Syntax：	int _remove(char *filename);
Example call：	if (_remove(C:\\data\\store.dat) _puts("store.dat deleted");
Includes：	#include "SDK.h "
Description：	Delete the file specified by filename. If filename exceeds 8 characters, it will be truncated to 8 characters long. If the file to be deleted is a DBF file, the DBF file and all the index (key) files associated to it will be deleted altogether.
Returns：	If remove deletes the file successfully, it returns an integer value of 1. In case of error, remove will return an integer value of 0 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretations are listed below.

fErrorCode： 1:filename is a NULL string.

2:File specified by filename does not exist.

_rename

Purpose：	Change file name of an existing file.
Syntax：	int _rename(char *old_filename, char *new_filename);
Example call：	if (_rename("C:\\data\\store.dat"," C:\\data\\text.dat")
	_puts("store.dat renamed");
Includes：	#include "SDK.h "
Description：	Change the file name of the file specified by old_filename to new_filename. But the route does not change.
Returns：	If rename successfully changes the file name, it returns an integer value of 1. In case of error, rename will return an integer value of 0, and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	1:Either old_filename or new_filename is a NULL string.
	2:File specified by old_filename does not exist.
	3:A file with file name new_filename already exists.
	4:File path is error
	5:Filename is too long.
	6:File is using.
	7:Filename is error
	8:Other error

tell

Purpose：	Get file pointer position of a DAT file.
Syntax：	long tell(int fd);
Example call：	current_position = tell(fd);
Includes：	#include "SDK.h "
Description：	The tell function returns the current file pointer position of the DAT file whose file handle is specified in the argument fd. The file pointer position is expressed in number of bytes from the beginning of file. For instance, if the file pointer points to the beginning of file, the file pointer position will be 0.
Returns：	The long integer value returned by tell is the current file pointer position in file. In case of error, tell returns a long value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.

| fErrorCode： | 2:File handle is NULL. |
| | 7:fd is not a file handle of a previously opened file. |

write

Purpose：	Write a specified number of bytes to a DAT file.
Syntax：	int write(int fd, char *buffer, unsigned count);
Example call：	write(fd, data_buffer,100);
Includes：	#include "SDK.h "
Description：	The write function writes the number of bytes specified in the argument count from the character array buffer to a DAT file whose file handle is fd. Writing of data starts at the current position of the file pointer, which is incremented accordingly when the operation is completed.
	If the end-of- file condition is encountered during the operation, the file will be extended automatically to complete the operation.
Returns：	The write function returns the number of bytes actually written to the file. In case of error, write returns an integer value of -1 and an error code is set to the global variable fErrorCode to indicate the error condition encountered. Possible error codes and their interpretation are listed below.
fErrorCode：	2:File handle is NULL.
	7:fd is not a file handle of a previously opened file.
	10:No more free file space for file extension.

writeln

Purpose：	Write a line terminated by a null character (\0) to a DAT file. The null character is also written to the file. After writing in, file position will update.
Syntax：	int writeln(int fd, char *buffer);
Example call：	writeln(fd, data_buffer);
Includes：	#include "SDK.h "
Description：	The writeln function writes a line terminated by a null character from the character array buffer to a DAT file whose file handle is fd. Characters are written to the file until a null character (\0) is encountered. The null character is also written to the file. Writing of data starts at the current position of the file pointer, which is incremented accordingly when the operation is completed. If the end-of-file condition is encountered during the operation, the file will be extended automatically to complete the operation.
Returns：	The writeln function returns the number of bytes actually written

to the file (includes the null character). In case of error, writeln

returns an integer value of -1 and an error code is set to the global

variable fErrorCode to indicate the error condition encountered.

Possible error codes and their interpretation are listed below.

fErrorCode： 2:File handle is NULL.

7:fd is not a file handle of a previously opened file.

9:no null character found in buffer

10:No more free file space for file extension.

DiskC_format

Purpose：	Format disk C.
Syntax：	int DiskC_format(void);
Example call：	DiskC_format ( );
Includes：	#include "SDK.h "
Description：	The DiskC_format function formats disk C.
Returns：	0 : Format false。
	1 : Format OK。
fErrorCode：	None

DiskD_format

Purpose：	Format disk D.
Syntax：	int DiskD_format (void);
Example call：	DiskD_format ( );
Includes：	#include "SDK.h "
Description：	The DiskC_format function formats disk D.
Returns：	0 : Format false。
	1 : Format OK。
fErrorCode：	None

DiskC_totalsize

Purpose：	Checking the total space in disk C.
Syntax：	unsigned int DiskC_totalsize (void);
Example call：	DiskC_totalsize ( );
Includes：	#include "SDK.h "
Description：	The DicskC_totalsize function returns the used space in disk C.
Returns：	0xffffffff : Disk C unformatted.
	Others : The total space in disk C.(Bytes)
fErrorCode：	None

DiskD_totalsize

Purpose：	Checking the total space in disk D.
Syntax：	unsigned int DiskD_ totalsize (void);

Example call：	DiskD_totalsize ( );
Includes：	#include "SDK.h "
Description：	The DicskD_totalsize function returns the total space in disk D.
Returns：	0xffffffff : Disk D unformatted.
	Others : The total space in disk D.(Bytes)
fErrorCode：	None

DiskC_usedsize

Purpose：	Checking the used space in disk C.
Syntax：	unsigned int DiskC_usedsize (void);
Example call：	DiskC_usedsize ( );
Includes：	#include "SDK.h "
Description：	The DicskC_usedsize function returns the used space in disk C.
Returns：	0xffffffff : Disk C unformatted.
	Others : The used space in disk C.(Bytes)
fErrorCode：	None

DiskD_usedsize

Purpose：	Checking the used space in disk D.
Syntax：	unsigned int DiskD_usedsize (void);
Example call：	DiskD_usedsize ( );
Includes：	#include "SDK.h "
Description：	The DicskD_usedsize function returns the used space in disk D.
Returns：	0xffffffff : Disk D unformatted.
	Others : The used space in disk D.(Bytes)
fErrorCode：	None

DicskC_freesize

Purpose：	Checking the free space in disk C.
Syntax：	unsigned int DiskC_freesize (void);
Example call：	DiskC_freesize( );
Includes：	#include "SDK.h "
Description：	The DicskC_freesize function returns the free space in disk C.
Returns：	0xffffffff : Disk C unformatted.
	Others : The free space in disk C.(Bytes)
fErrorCode：	None

DicskD_freesize

Purpose：	Checking the free space in disk D.
Syntax：	unsigned int DiskD_freesize (void);
Example call：	DiskD_freesize( );
Includes：	#include "SDK.h "

Description：	The DicskD_freesize function returns the free space in disk D.
Returns：	0xffffffff : Disk C unformatted.
	Others : The free space in disk D.(Bytes)
fErrorCode：	None

getDirNum

Purpose：	Get the folder quantity in designate path.
Syntax：	int getDirNum(char *pssPath);
Example call：	int Dir_Num;
	Dir_Num = getDirNum("C:\\");
Includes：	#include "SDK.h "
Description：	The getDirNum function can get the folder quantity in designate path.
Returns：	-1 : path error.
	-2 : disk unformat.
	upward 0 : folder quantity.
fErrorCode：	None

getFileNum

Purpose：	Get the file quantity in designate path.
Syntax：	int getFileNum(char *pssPath);
Example call：	int File_Num;
	File_Num = getFileNum("C:\\Data\\");
Includes：	#include "SDK.h "
Description：	The getFileNum function can get the file quantity in designate path.
Returns：	-1 : path error.
	-2 : disk unformat.
	upward 0 : folder quantity.
fErrorCode：	None

getDirList

Purpose：	Get the folder information in designate path.
Syntax：	int getDirList(char *pssPath, char *pssBuffer);
Example call：	int DirNum;
	char assBuffer[100];
	DirNum = getDirList ("C:\\", assBuffer);
Includes：	#include "SDK.h "
Description：	The getDirList function can get the folder quantity and name in designate path.
	When pssBuffer = NULL, this function will return the buffer size.
	For example, the path "D:\" has three folders "Program", "Fonts", "Lookup", then the buffer will get folder information like "Program Fonts

Lookup".

Returns：	-1 : path error.
	-2 : disk unformat.
	upward 0 : When pssBuffer = NULL, it will return buffer size. When
	pssBuffer != NULL, it will return folder quantity.
fErrorCode：	None

getFileList

Purpose：	Get the file information in designate path.
Syntax：	int getFileList(char *pssPath, char *pssBuffer);
Example call：	int File_Num;
	char assBuffer[200];
	File_Num = getFileList("D:\\Lookup\\", assBuffer);
Includes：	#include "SDK.h"
Description：	The getFileList function can get the file quantity and name in designate
	path.
	When pssBuffer = NULL, this function will return the buffer size.
	For example, the path "C:\Data\" has two files "StoreIn.dat",
	"StoreOut.dat", and their size are 1128 bytes and 564 bytes, then the
	buffer will get file information like "StoreIn.dat dat 1128 StoreOut.dat dat
	564 ".
Returns：	-1 : path error.
	-2 : disk unformat.
	upward 0 : When pssBuffer = NULL, it will return buffer size. When
	pssBuffer != NULL, it will return file quantity.
fErrorCode：	None

_fclose

Purpose：	Use _fclose to close a file opened earlier for buffered input/output using
	_fopen.
Syntax：	int _fclose(_TFILE *file_pointer);
Example call：	_fclose(infile);
Includes：	#include "SDK.h"
Description：	The _fclose function closes the file specified by the argument file_pointer.
	This pointer must have been one returned earlier when the file was opened
	by _fopen. If the file is opened for writing, the contens of the buffer
	associated with the file are flushed before the file is closed. The buffer is
	then released.
Returns：	If the file is successfully closed, _fclose returns a zero. In case of an error,
	the return value is equal to the constant EOF.

_fcloseAll

Purpose：	Use _fcloseAll to close all files opened for buffered input/output with _fopen or tmpfile.
Syntax：	void _fcloseAll(void);
Example call：	_fcloseAll();
Includes：	#include "SDK.h"
Description：	The _fcloseAll function closes all files that have been opened by _fopen or tmpfile for buffered I/O. Buffers associated with files opened for writing are written out to the corresponding file before closing.

_filelength

Purpose：	Use _filelength to dertimine the length of a file in bytes.
Syntax：	size_t _filelength(_TFILE* file_pointer);
Example call：	file_size = _filelength(infile);
Includes：	#include "SDK.h"
Description：	The _filelength function returns the size in number of bytes of the file specified in the argument file_pointer. This pointer should be the return value of earlier opened file by _fopen.
Returns：	The integer value returned by _filelength is the size of the file in number of bytes.

_fopen

Purpose：	Use _fopen to open a file for buffered input/output operations.
Syntax：	_TFILE* _fopen(const char*filename, const char *access_mode);
Example call：	input_file = _fopen("c:\\data\\order.dat", "r");
Includes：	#include "SDK.h"
Description：	The fopen function opens the file specified in the argument filename. The type of operations you intend to perform on the file must be given in the argument access_mode. The following table explains the values that the access_mode string can take:

Access Mode String	Interpretation
r	Opens file for read operations only. The *_fopen* function fails if the file does not exist.
w	Opens a new file for writing. If the file exists, its contents are destroyed.
r+	Opens an existing file for both read and write operations. Error is returned if file does not exist.

w+	Creates a file and opens it for both reading and writing. If file exists, current contents are destroyed.

Returns： If the file is opened successfully, _fopen returns a pointer to the file. Actually, this is a pointer to a structure of type _TFILE, which is defined in the header file. The actual structure is allocated elsewhere and you do not have to allocate it. In case of an error, _fopen returns a NULL.

_fread

Purpose： Use _fread to read a specified number of data items, eachof a given size, from the current position in a file opened for buffered input. The current position is updated after the read.

Syntax： size_t _fread(const void *buffer, size_t size, size_t count, _TFILE *file_pointer);

Example call： Numread = _fread(buffer, sizeof(char), 80, infile);

Includes： #include "SDK.h"

Description： The fread function reads count data items, each of size bytes, starting at the current read position of the file specified by the argument file_pointer. After the read is complete, the current position is updated. You must allocate storage for a buffer to hold the number of bytes that you expect to read. This buffer is a pointer to a void data type.

Returns： The _fread function returns the number of items it successfully read.

_fseek

Purpose： Use _fseek to move to a new position in a file opened for buffered input/output.

Syntax： int _fseek(_TFILE *file_pointer, long offset, int origin);

Example call： _fseek(infile, 0, SEEK_SET); /* Go to the beginning */

Includes： #include "SDK.h"

Description： The fseek function sets the current read or write position of the file specified by the argument file_pointer to a new value indicated by the arguments "off-set" and "origin". The "offset" is a long integer indicating how far away the new position is from a specific location given in "origin". The following table explains the possible value of "origin".

Origin	Interpretation
SEEK_SET	Beginning of file.
SEEK_CUR	Current position in the file.

Returns： When successful, _fread returns a zero. In case of error, _fread returns a

non-zero value.

_fwrite

Purpose：Use _fwrite to write a specified number of data itmes, each of a given size, from a buffer to the current position in a file opened for buffered output. The current position is updated after the write.

Syntax：size_t _fwrite(const void *buffer, size_t size, size_t count, _TFILE *file_pointer);

Example call：numwrite = _fwrite(buffer, sizeof(char), 80, outfile);

Includes：#include "SDK.h"

Description：The _fwrite function writes count data items, each of size bytes, to the file specified by the argument file_pointer, starting at the current position. After the write operation is complete, the current position is updated. The data to be written is in the buffer whose address is passed to _fwrite in the argument buffer.

Returns：The _fwrite function returns the number of items it actually wrote.

DBMS

Ini_Search

Purpose： Use "Ini_Search" can initiate the file search function in disk.

Syntax： int Ini_Search(_TFILE* filehd,_DBMS* F_Search, unsigned char *pusFielddlt, int record_type, int record_length, int total_field_no, int total_record_no);

Example call： **Example 1: Variable field length**

_DBMS fsearch;

_TFILE *filepoint;

unsigned char ausFielddlt[2]=",";

filepoint = _fopen("c:\\data\\data.txt","r+");

Ini_Search(filepoint,&fsearch, ausFielddlt,1,0,5,0);

Example 2: Regular field length

_DBMS fsearch;

_TFILE *filepoint;

unsigned char field_size[5]={6,5,4,5,6};

filepoint = _fopen("c:\\data\\data.txt","a+");

Ini_Search(filepoint,&fsearch, field_size,0,26,5,0);

Includes： #include "DBMS.h "

Description： This function can initialize a work of searching file. After inserting every argument, you can use _ DBMS* F _ Search to search files. Several introduces the argument as follows:

argument	description
_TFILE* filehd	An opened file handle.
_DBMS* F_Search	One of _DBMS start address has already declared. Originally after the beginning success this argument was used for written into various kinds of search.

	unsigned char *pusFielddlt	This argument has two kinds of meanings. When record _ Type is 0, search for regular length. This function needs to insert the unsigned char array; the array represents the length of every field. When record _ Type is1, search for variable length, this function need to insert one character to represent separate symbol.
	int record_type	When record _ Type is 0, search for regular length. It has no separate symbols among field and field. When record _ Type is 1, search for variable length. It needs a separate symbol among field and field.
	int record_length	This argument is each record's length. When record _ Type is 0, need to insert this value, not including the symbol of line feed. When record _ Type is 1, this field can insert any value.
	int total_field_no	This argument is the field's quantity of each record.
	int total_record_no	Total amount of records in the file. If does not know the total amount, you can insert -1, that will calculate automatically by the system.
Returns：	0: Initialize defeat. 1: Initialize success.	

Ini_SearchAdv

Purpose：	Use "Ini_SearchAdv" can initiate the advance file search function in disk
Syntax：	int Ini_SearchAdv(_TFILE* filehd,_DBMS* F_Search, unsigned char *pusFielddlt, unsigned char*pusKeyIdx, int slKeyField_Num, int record_type, int record_length, int total_field_no, int total_record_no);
Example call：	**Example 1: Variable field length**
	_DBMS fsearch;
	_TFILE *filepoint;
	unsigned char ausFielddlt[2]=",";

filepoint = _fopen("c:\\data\\data.txt","r+");

Ini_Search(filepoint, &fsearch, ausFielddlt, NULL, 0, 1, 0, 5, 0);

Example 2: Regular field length

_DBMS fsearch;

_TFILE *filepoint;

unsigned char field_size[5]={6,5,4,5,6};

unsigned char keyfield[2] = {0, 2};

int keyfieldNum = 2;

filepoint = _fopen("c:\\data\\data.txt","a+");

Ini_Search(filepoint, &fsearch, field_size, keyfield , keyfieldNum, 0, 26, 5,
 0);

Includes： #include "DBMS.h "

Description： This function can initialize a work of advance searching file. After inserting

every argument, you can use _ DBMS* F _ Search to search files.

When using this function to initial a DBMS search, you have to take care

for:

1. This function cannot support Variable field length search.

2. When initial, we will make a index file in C disk, so it has to take a few

time.

3. The index filename will be similar to origin file. For example, the

lookup file name is "AAA.txt", the index filename will be "AAA.idx". So,

you have to check the duplicate filename to avoid error fo making

index file.

4. You have to reserve some space for the function to make index file in

C disk.

Several introduces the argument as follows:

argument	description
_TFILE* filehd	An opened file handle.
_DBMS* F_Search	One of _DBMS start address has already declared. Originally after the beginning success this argument was used for written into various kinds of search.

	unsigned char *pusFielddlt	This argument has two kinds of meanings. When record _ Type is 0, search for regular length. This function needs to insert the unsigned char array; the array represents the length of every field. When record _ Type is1, search for variable length, this function need to insert one character to represent separate symbol.
	unsigned char *pusKeyIdx	This argument can give max. 8 key fields for search. We will make a checksum index file for these key fields.
	int slKeyField_Num	This argument can give the sum of pusKeyIdx size, for sum of key fields.
	int record_type	When record _ Type is 0, search for regular length. It has no separate symbols among field and field. When record _ Type is 1, search for variable length. It needs a separate symbol among field and field.
	int record_length	This argument is each record's length. When record _ Type is 0, need to insert this value, not including the symbol of line feed. When record _ Type is 1, this field can insert any value.
	int total_field_no	This argument is the field's quantity of each record.
	int total_record_no	Total amount of records in the file. If does not know the total amount, you can insert -1, that will calculate automatically by the system.
Returns：	0: Initialize defeat.	
	1: Initialize success.	
	-1: Argument "pusKeyIdx" or "slKeyField_Num" is error, please check it.	
	-2: Cannot make a IDX file, please check your lookup filename or C disk size.	

Close_Search

Purpose： Use "Close _ Search" can close the file search function in Disk C and D.

Syntax：	int Close_Search(_DBMS* F_Search);
Example call：	Close_Search(&F_Search);
Includes：	#include "DBMS.h"
Description：	When want to finish the file searching state, you can use this function.
Returns：	0: Close defeat.
	1: Close success.

SearchField

Purpose：	SearchField can search the appointed field that begin from the appointed record and compare with importing string. If agreeing, pass back to the first record.
Syntax：	int SearchField(_DBMS* F_Search, char* field, int search_fieldno, int recordno, int flag);
Example call：	char str[8]="abcdefg";
	int Record_Num;
	Record_Num =SearchField(&fsearch, str,0,0,FORWARD);
Includes：	#include "DBMS.h"
Description：	Several describe the argument as follows:

argument	description
_DBMS* F_Search	The file's searching structure that has been initialized.
char* field	String data wanted to match.
int search_fieldno	Field wanted to search.
int recordno	Begin to search from which data.
int flag	FORWARD => Search from forward to backward
	BACKWARD => Search from backward to forward
	As success of searching, the file index will stay in successful record front. When search defeat, the file index will not be moved.

Returns：	-1: Search defeat.
	Other value: Match the record position of data

SearchField_GR

Purpose：	SearchField_GR can search the appointed field that begin from the appointed record and compare with importing string. If agreeing, it will copy the record which included the field to buffer.
Syntax：	int SearchField_GR(_DBMS* F_Search, char* field, int search_fieldno, int

recordno, char* R_Buffer, int flag);

Example call： char str[8]="abcdefg",str_buffer[60];

SearchField_GR(&fsearch, str,0,0, str_buffer,FORWARD);

Includes： #include "DBMS.h"

Description： This function can search and contrast the data of appointed field. After success, reading the record which includes this field.

Several describe the argument as follows:

argument	description
_DBMS* F_Search	The file's searching structure that has been initialized.
char* field	String data wanted to match.
int search_fieldno	Field wanted to search.
int recordno	Begin to search from which data.
char* R_Buffer	After contrast success, it will write record which included this field into buffer.
int flag	FORWARD => Search from forward to backward
	BACKWARD => Search from backward to forward
	As success of searching, the file index will stay in successful record front. When search defeat, the file index will not be moved.

Returns： When "R _ Buffer = NULL", pass back – 1: Search defeat; Pass other value back: That is the size of space for buffer.

When "R _ Buffer ≠ NULL", pass back – 1: Search defeat; Pass other value back: That is the record position which confirm to contrast data.

SearchField_GF

Purpose： Search the designated field. After success, acquiring the appointed field in including the field's record.

Syntax： int SearchField_GF(_DBMS* F_Search, char* field, int search_fieldno, int recordno, int get_field_no, char* F_Buffer, int flag);

Example call： char str[8]="abcdefg",str_buffer[60];

SearchField_GF(&fsearch, str,0,0,1,str_buffer,FORWARD);

Includes： #include "DBMS.h"

Description： Search the correctly appointed field. After search success, acquiring another appointed field which including record of this field.

Several describe the argument as follows:

argument	description
_DBMS* F_Search	The file's searching structure that has been initialized.
char* field	String data wanted to match.
int search_fieldno	Field wanted to search.
int recordno	Begin to search from which data.
int get_field_no	After contrasting success, acquiring the data of appointed field in this record.
char* F_Buffer	After contrast success, it will write record which included this field into buffer.
int flag	FORWARD => Search from forward to backward
	BACKWARD => Search from backward to forward
	As success of searching, the file index will stay in successful record front. When search defeat, the file index will not be moved.

Returns： When "F _ Buffer = NULL", pass back – 1: Search defeat; Pass other value back: That is the size of space for buffer.

When "F _ Buffer ≠ NULL", pass back – 1: Search defeat; Pass other value back: That is the record position which confirm to contrast data.

SearchMultiField_GF

Purpose： Search the designated field. The field's information include field string and field number.You can write many fields in this field buffer. After searching success, acquiring the appointed field in including this field's record.

Syntax： int SearchMultiField_GF(_DBMS* F_Search, char* multi_field, int recordno, int get_field_no, char* F_Buffer, int flag);

Example call： char str[20]="00001,0;abcdefg,1",str_buffer[60];
SearchMultiField_GF(&fsearch, str, 0,3,str_buffer,FORWARD);

Includes： #include "DBMS.h"

Description： Search the correctly appointed field. After search success, acquiring another appointed field which including record of this field.

Several describe the argument as follows:

argument	description
_DBMS* F_Search	The file's searching structure that has been initialized.
char* multi_field	String data wanted to match. The string form is "*field string 0, field number 0; field string 1, field number 1;...*". Each field string and field number use separate symbol ",", behind the field number use separate symbol ";", last field number don't use any separate symbol.
int recordno	Begin to search from which data.
int get_field_no	After contrasting success, acquiring the data of appointed field in this record.
char* F_Buffer	After contrast success, it will write record which included this field into buffer.
int flag	FORWARD => Search from forward to backward BACKWARD => Search from backward to forward As success of searching, the file index will stay in successful record front. When search defeat, the file index will not be moved.

Returns： When "F _ Buffer = NULL", pass back – 1: Search defeat; Pass other value back: That is the size of space for buffer.

When "F _ Buffer ≠ NULL", pass back – 1: Search defeat; Pass other value back: That is the record position which confirm to contrast data.

SeekRecord

Purpose： Move the searching index to the appointed record.

Syntax： long SeekRecord(_DBMS* F_Search,int recordno);

Example call： SeekRecord(&fsearch,10);//move file index to eleventh record。

Includes： #include "DBMS.h"

Description： Use this function can move the search index to appointed record. The number of first record is 0. The number of second record is 1.

Returns： -1: The index move is defeated.

Other value: the present address of searching index

GetRecordNum

Purpose： Use this function can read the total amount of records storing in the file at present. .

Syntax：	int GetRecordNum(_DBMS* F_Search);
Example call：	int record_num;
	record_num= GetRecordNum(&fsearch);
Includes：	#include "DBMS.h"
Description：	GetRecordNum can pass back the amount of record storing in the file at present.
Returns：	Amount of record that stores in the file

DeleteRecord

Purpose：	Use this function can delete the appointed record in the file.
Syntax：	int DeleteRecord(_DBMS* F_Search,int recordnum);
Example call：	DeleteRecord(&fsearch,2);//delete the third data of this file。
Includes：	#include "DBMS.h"
Description：	"DeleteRecord" can delete the appointed record, and change the size of the file.
	As success of deleting, file index will stay in the deleting record front. As deleting defeat, file index will not move.
Returns：	0: Delete defeat. 1: Delete success.

DeleteLastRecord

Purpose：	Use this function can delete the last record in the file.
Syntax：	int DeleteLastRecord(_DBMS* F_Search);
Example call：	DeleteLastRecord(&fsearch);
Includes：	#include "DBMS.h"
Description：	"DeleteLastRecord" can delete the last record in the file, and change the size of the file.
	As success of deleting, file index will stay in deleting record front. As deleting defeat, file index will not move.
Returns：	0: Delete defeat. 1: Delete success.

AppendRecord

Purpose：	Use this function can increase a new record on the file end.
Syntax：	int AppendRecord(_DBMS* F_Search,char* record);
Example call：	char str_record[25]="A1357924680,PA-20,3500";
	AppendRecord(&fsearch, str_record);
Includes：	#include "DBMS.h"
Description：	"AppendRecord" can increase a new record on the file end, the data of record is introduced by char * record.
	As increasing success, file index will be moved to the front of increasing record.
Returns：	-1: Write into defeat.

Other value: the quantity of the data.

WriteField

Purpose： Use this function can revise the designated record in the existed file.

Syntax： int WriteField(_DBMS* F_Search, int recordno, int fieldno, char* field);

Example call： char str_field[10]="123456789";

WriteField(&fsearch,0,1, str_field);// Revise the second field of the first

data to "str_field".

As revising success, file index will be moved to the front of the record

included revising field.

Includes： #include "DBMS.h"

Description： Using WriteField function can copy the field of appointed record. If the file

in disc D that you want to write, it will not allow to write.

Returns： -1: Write into defeat.

Other value: Write into the amount of data.

WriteRecord

Purpose： Using this function can copy the existed record.

Syntax： int WriteRecord(_DBMS* F_Search, int recordno, char* record);

Example call： char str_record[20]="A123456,PA-20,2330";

WriteRecord(&fsearch,0, str_record);// Revise the first record to char

str_record。

Includes： #include "DBMS.h"

Description： Use WriteRecord function can copy the existed record, but unable to

increase a new record.

As revising success, file index will be moved to revise the front of revising

record. If the file in disc D that you want to write, it will not allow to write.

Returns： -1: Write into defeat.

Other value: Write into the amount of data.

ReadField

Purpose： Use this function to read the data of appointed field in the appointed

record.

Syntax： int ReadField(_DBMS* F_Search, int recordno, int fieldno, char* buffer);

Example call： char str_buffer[30];

ReadField(&fsearch,5,0,str_buffer);//Reading the data of first field in the

sixth record, and store to "str_buffer".

Includes： #include "DBMS.h"

Description： int recordno : Read of record position.

int fieldno : Read of field position.

char* buffer : Read the storing space of field。

Returns： When char * buffer = NULL, functions will pass the data size back. Read
defeat: Pass back - 1.

When char * buffer ≠ NULL. Read succeed: Pass 1 back; Read defeat:
Pass back - 1.

ReadRecord

Purpose： Use this function to read the data of appointed record.

Syntax： int ReadRecord(_DBMS* F_Search, int recordno, char* buffer);

Example call： char str_buffer[30];

ReadRecord (&fsearch,5,str_buffer);//Reading the data of sixth record,

and store to "str_buffer".

Includes： #include "DBMS.h"

Description： int recordno : Read of record position。

char* buffer : Read the storing space of field。

Returns： When char * buffer = NULL, functions will pass materials size back. Read
defeat. Pass back - 1.

When char * buffer does not equal NULL. Read succeed. Passing 1 back;
Read defeat. Pass back - 1.

LED

set_led

Purpose：	To set the LED indicators
Syntax：	void set_led(int led, int mode, int duration);
Example call：	set_led(LED_RED, LED_FLASH, 30);
Includes：	#include "SDK.h "

Description：

led	description
LED_GREEN	LED moving display green light.
LED_RED	LED moving display red light.
LED_ORANGE	LED moving display orange light.

mode	description
LED_OFF	off for (duration X 0.01) seconds then on
LED_ON	on for (duration X 0.01) seconds then off
LED_FLASH	flash, on then off each for (duration X 0.01) seconds then repeat

Returns： none

Keypad

clr_kb

Purpose：	To clear the keyboard buffer.
Syntax：	void clr_kb(void);
Example call：	clr_kb( );
Includes：	#include "SDK.h "
Description：	The clr_kb function clears the keyboard buffer. This function is automatically called by the system program upon power up.
Returns：	none

dis_alpha

Purpose：	Disable alphabet key stroke processing.
Syntax：	void dis_alpha(void);
Example call：	dis_alpha( );
Includes：	#include "SDK.h "
Description：	The dis_alpha function disables the alphabet key stroke processing. If the alpha lock status is on prior to calling this function, it will become off after calling this function.
Returns：	none

en_alpha

Purpose：	Enable alphabet key stroke processing.
Syntax：	void en_alpha(void);
Example call：	en_alpha( );
Includes：	#include "SDK.h "
Description：	The en_alpha function enables the alphabet key stroke processing.
Returns：	none

get_alpha_enable_state

Purpose：	Get the status of the alphabet key stroke processing.
Syntax：	void get_alpha_enable_state (void);
Example call：	get_alpha_enable_state ( );
Includes：	#include "SDK.h "
Description：	This routine gets the current status, enable/disable, of the alphabet key stroke processing. The default is enabled.
Returns：	1, if the alphabet key stroke processing is enabled. 0, if disabled.

set_alpha_mode_state

Purpose：	Set the status of the alphabet mode status.

Syntax：	void set_alpha_mode_state(int status);
Example call：	set_alpha_mode_state(ALPHA_123);
Includes：	#include "SDK.h "
Description：	This function can set alphabet mode.
Returns：	none

get_alpha_mode_state

Purpose：	Get the status of the alphabet mode status.
Syntax：	int get_alpha_mode_state(void);
Example call：	get_alpha_mode_state();
Includes：	#include "SDK.h "
Description：	This function can get alphabet mode status.
Returns：	ALPHA_123　　: Numeric input.
	ALPHA_abc　　: Lowercase input.
	ALPHA_ABC　　: Uppercase input.

set_keypad_BL

Purpose：	Set keypad backlight on/off.
Syntax：	void set_keypad_BL(BOOL bStatus);
Example call：	set_keypad_BL(TRUE);//Key backlight on.
Includes：	#include "SDK.h "
Description：	This function can set keypad backlight on or off.
Returns：	None

get_keypad_BL

Purpose：	Get keypad backlight on/off status.
Syntax：	BOOL get_keypad_BL(void);
Example call：	if (get_keypad_BL())
	_printf_color(COLOR_BLACK, "Key Backlight on");
Includes：	#include "SDK.h "
Description：	This function can get keypad backlight status.
Returns：	TRUE: Key backlight on.
	FALSE: Key backlight off.

set_keypad_BL_Timer

Purpose：	Set keypad backlight timer.
Syntax：	void set_keypad_BL_Timer(int slTimer);
Example call：	set_keypad_BL_Timer(1);//Set keypad backlight timer for 1 sec.
Includes：	#include "SDK.h "
Description：	This function can set keypad backlight timer.

Returns： None

get_keypad_BL_Timer

Purpose： Get keypad backlight timer.

Syntax： int   get_keypad_BL_Timer(void);

Example call： int slkeypadimer;

slkeypadimer = get_keypad_BL_Timer();

Includes： #include "SDK.h "

Description： This function can get keypad backlight timer.

Returns： 0: Keypad backlight always on

Other: The timer for keypad backlight(sec.).

kbhit

Purpose： Check keybuffer is empty or not.

Syntax： int kbhit(void);

Example call： kbhit();

Includes： #include "SDK.h "

Description： This function can check keybuffer is empty or not.

Returns： 0: Keybuffer is empty.

1: Keybuffer is not empty.

_getchar

Purpose： Get one key stroke from the keyboard buffer.

Syntax： char _getchar(void);

Example call： c=_getchar ( );

if (c > 0) _printf("Key %d pressed",c);

else printf("No key pressed");

Includes： #include "SDK.h "

Description： The getchar function reads one key stroke from the keyboard buffer and

then removes the key stroke from the keyboard buffer. It will pass the

value back, and clear the buffer. If there is no any key press before, it will

pass NULL(0X00) back.

Returns： The getchar function returns the key stroke read from the keyboard

buffer. If the keyboard buffer is empty, a null character (0x00) is returned.

The keystroke returned is the ASCII code of the key being pressed.

GetKeyClick

Purpose： Get current key click status

Syntax： int GetKeyClick(void);

Example call： state = GetKeyClick( );

Includes：	#include "SDK.h "
Description：	The function returns an integer indicates the key click staus.The default is enabled.
Returns：	1, if key click sound is enabled.
	0, if key click sound is disabled.

SetKeyClick

Purpose：	To enable / disable the key click sound.
Syntax：	void SetKeyClick(int status);
Example call：	SetKeyClick(1); /* enable the key click sound */
Includes：	#include "SDK.h "
Description：	This routine truns on or off the key click sound
	1, if key click sound is enabled.
	0, if key click sound is disabled.
Returns：	none

Def_PKey

Purpose：	Change progrom key 1 & 2(P1 & P2) key define.
Syntax：	void Def_PKey(int nPKey, char ssDef);
Example call：	Def_PKey (KEY_P1, KEY_CR); /*Change P1 key to ENT key*/
Includes：	#include "SDK.h "
Description：	This function can change the program key (P1 & P2) to other key define.
	For example, change P1 key to ENT key or ESC key.
Returns：	none

FNKey_Reset

Purpose：	To reset all of FN-Key setting.
Syntax：	void FNKey_Reset(void);
Example call：	FNKey_Reset();
Includes：	#include "SDK.h "
Description：	將所有功能鍵設定值回復預設值(Null)
	If you want to set default for all FNKey.
Returns：	none

FNKey_GetState

Purpose：	To check the FN-Key setting that is custom or default.
Syntax：	char FNKey_GetState(short smKeyNum);
Example call：	if (FNKey_GetState(0))
	_printf("FN + 0 key is custom setting");

Includes：	#include "SDK.h "
Description：	You can check the FN-Key function that is default setting or custom setting.
	smKeyNum： 0 → F1, 1 → F2, 2 → F3, 3 → F4, 4 → F5, 5 → F6
Returns：	1 : Custom Setting。
	0 : Default Setting。
	-1: Error。

FNKey_SetUserDef

Purpose：	To set a custom setting for FN-Key.
Syntax：	char FNKey_SetUserDef(short smKeyNum, void (*pslFunction)(void));
Example call：	

```
void Sample01FN(void)
{
    _printf("This is Test!!");
}
void SetFNKey(void)
{
    if (FNKey_SetUserDef(0, Sample01FN))
    {
        _printf("Set F1 UserDefine OK!");
    }
    if (FNKey_SetUserDef(0, NULL))
    {
        _printf("Set F1 Default OK!");
    }
}
```

Includes：	#include "SDK.h "
Description：	The function is used to set the FN-Key. After set successed, the FN-Key is changed for custom setting function. You can set F1~6, if you want to set default, please set pslFunction = NULL.
	smKeyNum： 0 → F1, 1 → F2, 2 → F3, 3 → F4, 4 → F5, 5 → F6
Returns：	1 : Set success。
	0 : Set false。

SetScanKeyPwOn

Purpose：	To set power on by scan key.
Syntax：	BOOL SetScanKeyPwOn(BOOL state);
Example call：	SetScanKeyPwOn(TRUE);
Includes：	#include "SDK.h "
Description：	This function can set power on by scan key.

| Returns： | FALSE:set fail |
| | TRUE:set success |

GetScanKeyPwOn

Purpose：	Get state for power on by scan key.
Syntax：	BOOL GetScanKeyPwOn(void);
Example call：	Int state;
	state = GetScanKeyPwOn();
Includes：	#include "SDK.h "
Description：	This function can get scan key power on state.
Returns：	FALSE:scan key not power on.
	TRUE:scan key power on.

_scanf_color

Purpose：	Use _scanf_color to read character strings from the standard input file and covert the strings to values of C variables according to specified formats.
Syntax：	int _scanf_color(int color, const char *format, ...);
Example call：	char assBuffer[10];
	_scanf_color(COLOR_RED, "%s", assBuffer);
Includes：	#include "SDK.h "
Description：	The _scanf_color function accepts a variable number of arguments, which it interprets as addresses of C variables, and reads character strings, representing their values. It converts them to their internal representations using formatting commands embedded in the argument *format*, which must be present in a call to _scanf_color.
	The interpretation of the variables depends on the *forma*. The formatting command for each variable begins with a % sign and can contain other characters as well. A whitespace character (a blank space, a tab, or a new line) may cause _scanf_color to ignore whitespace characters from keyboard. Other nonwhitespace characters, excluding the % sign, cause _scanf_color to ignore each matching character from the input. It begins to interpret the first nonmatching character as the value of variable that is being read.
	For each C variable whose address is included in the argument list to _scanf_color, there must be a format specification embedded in the *format*. For the complete format specification accepted by the _scanf_color function, please refer to the *scanf* function in Turbo C++.
	If you want input a float value, the value type is " double ", not " float ".
Returns：	The _scanf_color function returns the number of input items that were

successfully read, converted, and saved in variables. A return value equal to EOF means that an end-of-file was encountered during the read operation.

_scanf_color_DefaultStr

Purpose： Use _scanf_color_DefaultStr to set a default string in input and read character strings from the standard input file and covert the strings to values of C variables according to specified formats.

Syntax： int _scanf_color_DefaultStr(int color, char* assDefaultStr, const char *format, ...);

Example call： char assBuffer[10] = "ABC";

_scanf_color_DefaultStr(COLOR_RED, assBuffer , "%s", assBuffer);

Includes： #include "SDK.h "

Description： The _scanf_color_DefaultStr function accepts a variable number of arguments, which it interprets as addresses of C variables, and reads character strings, representing their values. It converts them to their internal representations using formatting commands embedded in the argument *format*, which must be present in a call to _scanf_color_DefaultStr.

The interpretation of the variables depends on the *forma*. The formatting command for each variable begins with a % sign and can contain other characters as well. A whitespace character (a blank space, a tab, or a new line) may cause _scanf_color_DefaultStr to ignore whitespace characters from keyboard. Other nonwhitespace characters, excluding the % sign, cause _scanf_color_DefaultStr to ignore each matching character from the input. It begins to interpret the first nonmatching character as the value of variable that is being read.

For each C variable whose address is included in the argument list to _scanf_color_DefaultStr, there must be a format specification embedded in the *format*. For the complete format specification accepted by the _scanf_color_DefaultStr function, please refer to the *scanf* function in Turbo C++.

If you want input a float value, the value type is " double ", not " float ".

Returns： The _scanf_color_DefaultStr function returns the number of input items that were successfully read, converted, and saved in variables. A return value equal to EOF means that an end-of-file was encountered during the read operation.

_scanf_ctrl_ScannerStatus

Purpose： Set scanner on/off when use "_scanf_color" function.

Syntax： void _scanf_ctrl_ScannerStatus(BOOL bStatus);

Example call： _scanf_ctrl_ScannerStatus(TRUE);

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set scanner status.

TRUE : Scanner on.

FALSE : Scanner off.

Returns： none

_scanf_ctrl_ScannerSleep

Purpose： Set scanner sleep on/off when use "_scanf_color" function.

Syntax： void _scanf_ctrl_ScannerSleep(BOOL bStatus);

Example call： _scanf_ctrl_ScannerSleep(TRUE);

Includes： #include "SDK.h "

Description： After use "_scanf_color" function, this function can set scanner status in sleep mode or stop.

If use this function and set status "TRUE", when left "_scanf_color" function, scanner will sleep, and when you use "_scanf_color" function next time, the scanner will not reinitial. That can make the "_scanf_color" function speed up.

TRUE : Scanner sleep.

FALSE : Scanner not sleep.

Returns： none

_scanf_ctrl_Vibrate

Purpose： Set vibrate on/off when use "_scanf_color" function and scanner status on.

Syntax： void _scanf_ctrl_Vibrate(BOOL bEnable);

Example call： _scanf_ctrl_Vibrate(TRUE);//Enable vibrate

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set vibrate on/off after scanner read.

TRUE : Vibrate on.

FALSE : Vibrate off.

Returns： none

_scanf_ctrl_ScanWithENT

Purpose： Set ENT auto press on/off when use "_scanf_color" function and scanner status on.

Syntax： void _scanf_ctrl_ScanWithENT(BOOL bScanEnt);

Example call： _ scanf_ctrl_ScanWithENT (TRUE);

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set auto press ENT

key after scanner read.

TRUE : Auto press ENT on.

FALSE : Auto press ENT off.

Returns： none

_scanf_ctrl_AlphaKey

Purpose： Set Alpha key function on/off when use "_scanf_color" function.

Syntax： void _scanf_ctrl_AlphaKey (int status);

Example call： _scanf_ctrl_AlphaKey (TRUE);

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set enable/disable

alpha key when key input.

TRUE : Enable alpha key.

FALSE : Disable alpha key.

Returns： none

_scanf_ctrl_AlphaKey_Mode

Purpose： Set alpha mode when use "_scanf_color" function.

Syntax： void _scanf_ctrl_AlphaKey_Mode(int slAlphaMode);

Example call： _scanf_ctrl_AlphaKey_Mode(ALPHA_123);//Set keypad input for number.

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set alpha mode when

key input.

ALPHA_123 : For input number.

ALPHA_abc : For input lower character.

ALPHA_ABC : For input upper character.

Returns：

_scanf_ctrl_password

Purpose： Set display for general or user define when use "_scanf_color" function.

Syntax： void _scanf_ctrl_password (char ssPassWord);

Example call： _scanf_ctrl_password ('*');

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set enable/disable

alpha key when key input.

0 : Input character nomoral display.

others : Input character display define word.

Returns： none

_scanf_ctrl_KeypadLock

Purpose： Set keypad lock on/off when use "_scanf_color" function.

Syntax： void _scanf_ctrl_KeypadLock(BOOL bLock);

Example call： _scanf_ctrl_KeypadLock(FALSE);

Includes： #include "SDK.h "

Description： When use "_scanf_color" function, this function can set keypad   input lock on/off except ENT key ,ESC key and Scan key.

TRUE : Keypad lock

FALSE : Keypad unlock.

Returns： none

LCD

The following functions clr_eol, clr_rect, clr_scr, fill_rect, fill_rect_color, Get_Cursor, Set_Cursor, gotoxy, wherex, wherexy, wherey, _printf_color, _putchar, _puts, _puts_color and show_image_bmp only effect the current TextBlock. The parameters of those function will base on TextBlock's size and position.

clr_eol	
Purpose：	Clear from where the cursor is to the end of the line. The cursor position is not affected after the operation.
Syntax：	void clr_eol(void);
Example call：	clr_eol( );
Includes：	#include "SDK.h "
Description：	The clr_eol function clears from where the cursor is to the end of the line, and then moves the cursor to the original place.
Returns：	None

clr_rect	
Purpose：	Clear a rectangular area on the LCD display. The cursor position is not affected after the operation.
Syntax：	void clr_rect(int left, int top, int width, int height);
Example call：	clr_rect(10,5,30,10);
Includes：	#include "SDK.h "
Description：	The clr_rect function clears an rectangular area on the LCD display whose top left position and size are specified by left, top, width, and height. The cursor position is not affected after the operation. Several introduces the argument as follows:
	left　　　　Clear form the start point of X-axis.
	top　　　　Clear form the start point of Y-axis.
	width　　　Clear the width form the start point.
	height　　　Clear the high form the start point.
Returns：	None

clr_scr	
Purpose：	Clear LCD display.
Syntax：	void clr_scr(void);
Example call：	clr_scr( );
Includes：	#include "SDK.h "
Description：	The clr_scr function clears the LCD display and places the cursor at the first column of the first line, that is (0,0) as expressed with the coordinate

system.

Returns：	None

fill_rect

Purpose：	Fill a white rectangular area on the LCD display.
Syntax：	void fill_rect(int left, int top, int width, int height);
Example call：	fill_rect (10,5,30,10);
Includes：	#include "SDK.h "
Description：	The fill_rect function fills a rectangular area white on the LCD display whose top left position and size are specified by left, top, width, and height. The cursor position is not affected after the operation.Several introduces the argument as follows:

left	Fill form the start point of X-axis.
top	Fill form the start point of Y-axis.
width	Fill the width form the start point.
height	Fill the high form the start point.

Returns：	None

fill_rect_color

Purpose：	Fill a user define color rectangular area on the LCD display.
Syntax：	void fill_rect_color(int left, int top, int width, int height, int color);
Example call：	fill_rect_color(10,5,30,10,COLOR_RED);
Includes：	#include "SDK.h "
Description：	The fill_rect_ color function fills a rectangular area for user define on the LCD display whose top left position and size are specified by left, top, width, and height. The cursor position is not affected after the operation.Several introduces the argument as follows:

left	Fill form the start point of X-axis.
top	Fill form the start point of Y-axis.
width	Fill the width form the start point.
height	Fill the high form the start point.
color	Fill color choice.

Returns：	None

Get_Cursor

Purpose：	Get current cursor status.
Syntax：	int Get_Cursor(void);
Example call：	if (Get_Cursor( ) ==0) _puts("Cursor Off");
Includes：	#include "SDK.h "
Description：	The Get_Cursor function checks if the cursor is visible or not.

| Returns： | The Get_Cursor function returns an integer of 1 if the cursor is visible (turned on), 0 if not. |

Set_Cursor

Purpose：	Turn on or off the cursor of the LCD display.
Syntax：	void Set_Cursor(int status);
Example call：	Set_Cursor (0);//Cursor off
Includes：	#include "SDK.h "
Description：	The Set_Cursor function displays or hides the cursor of the LCD display according to the value of status specified. If status equals 1, 2, or 3, the cursor will be turned on to show the current cursor position. If status equals 0, the cursor will be invisible.

status	Curser status
0	Cursor off.
1	Cursor on, and cursor type is a line as _.
2	Cursor on, and cursor type is a line as |.
3	Cursor on, and cursor type is a block as ■.

| Returns： | None |

gotoxy

Purpose：	Move cursor to new position.
Syntax：	int gotoxy(int x_position, int y_position);
Example call：	gotoxy(3,2);/* Move to second line of the third row */
Includes：	#include "SDK.h "
Description：	The gotoxy function moves the cursor to a new position whose coordinate is specified in the argument x_position and y_position.
Returns：	Normally the gotoxy function will return an integer value of 1 when operation completes. In case of LCD fault, 0 is returned to indicate error.

wherex

Purpose：	Get x-coordinate of the cursor location.
Syntax：	int wherex(void);
Example call：	x_position = wherex( );
Includes：	#include "SDK.h "
Description：	The wherex function determines the current x-coordinate location of the cursor.
Returns：	The wherex function returns the x-coordinate of the cursor location.

wherexy

| Purpose： | Get x-coordinate and y-coordinate of the cursor location |
| Syntax： | int wherexy(int* column, int* row); |

Example call：	wherexy(&x_position,&y_position);
Includes：	#include "SDK.h "
Description：	The wherexy function copies the value of x-coordinate and y-coordinate of the cursor location to the variables whose address is specified in the arguments column and row.
Returns：	None

wherey

Purpose：	Get y-coordinate of the cursor location.
Syntax：	int wherey(void);
Example call：	y_position = wherey( );
Includes：	#include "SDK.h "
Description：	The wherey function determines the current y-coordinate location of the cursor.
Returns：	none

lcd_backlit_Setlv

Purpose：	Set LCD backlight level.
Syntax：	void lcd_backlit_Setlv(int level);
Example call：	lcd_backlit_ Setlv (1);/*Set LCD backlight level 1*/
Includes：	#include "SDK.h "
Description：	The lcd_backlit_Setlv sets LCD backlight level. When any key is pressed, the backlight will turn on, and the light will be the level that you set. The back light level has 5 levels to set.
Returns：	None

lcd_backlit_SetTimer

Purpose：	Set LCD backlight timer.
Syntax：	void lcd_backlit_SetTmer(int timer);
Example call：	lcd_backlit_SetTmer (10);/*Set LCD backlight on timer for 10 sec.*/
Includes：	#include "SDK.h "
Description：	The lcd_backlit_SetTimer sets LCD backlight on timer. If set timer 0, the backlight always not light, others will set backlight on timer for sec. The max timer will be 65535 sec.
Returns：	None

lcd_backlit_Getlv

Purpose：	Get LCD backlight level.
Syntax：	int lcd_backlit_Getlv(void);
Example call：	lcd_backlit_Getlv();/*Get LCD backlight level.*/

Includes：	#include "SDK.h "
Description：	The lcd_backlit_Getlv gets LCD backlight level.
Returns：	LCD backlight level for 1~5.

lcd_backlit_GetTimer

Purpose：	Get LCD backlight timer.
Syntax：	int lcd_backlit_GetTimer(void);
Example call：	lcd_backlit_GetTimer( );//Get LCD backlight timer.
Includes：	#include "SDK.h "
Description：	The lcd_backlit_GetTimer gets LCD backlight timer.
Returns：	LCD backlight timer for 0~65535.

_printf_color

Purpose：	Use _printf_color to write character strings and volues of C variables, formatted in a specified manner, to display screen.
Syntax：	int _printf_color(int color, char *format, ...);
Example call：	_printf_color(COLOR_RED, "The product of %d and %d is %d\n", x, y, x*y);
Includes：	#include "SDK.h "
Description：	The _printf_color function accepts a variable number of arguments and prints them out to display screen. The value of each argument is formatted according to the codes embedded in the format specification format_string. If the format_string does not contain a % character (except for the pair %%, which appears as a single % in the output), no argument is expected and the format_string is written out to display screen. For the complete format specification accepted by the _printf function, please refer to the same function in Turbo C++.
Returns：	The _printf_color function returns the number of characters it has printed. In case of error, it returns EOF

_putchar

Purpose：	Display a character in color black on the LCD display.
Syntax：	int _putchar(char c);
Example call：	_putchar('A');
Includes：	#include "SDK.h "
Description：	The putchar function sends the character specified in the argument c to the LCD display at the current cursor position and moves the cursor accordingly.
Returns：	None

_putchar_color

Purpose：	Display a character in user define color on the LCD display.

Syntax： int _putchar_color(int color, char c);

Example call： _putchar_color(COLOR_BLACK, 'A');

Includes： #include "SDK.h "

Description： The putchar function sends the character specified in the argument c to the LCD display at the current cursor position and moves the cursor accordingly.

Returns： None

_puts

Purpose： Display a string in color black on the LCD display.

Syntax： char _puts (char* string)

Example call： _puts("Hello World");

Includes： #include "SDK.h "

Description： The puts function sends a character string whose address is specified in the argument string to the LCD display starting from the current cursor position. The cursor is moved accordingly as each character of string is sent to the LCD display. The operation continues until a terminating null character is encountered.

Returns： The puts function returns the number characters sent to the LCD display.

_puts_color

Purpose： Display a string in color black on the LCD display.

Syntax： char _puts_color(int colorindex, char* string);

Example call： _puts_ color (COLOR_RED, "Hello World");

Includes： #include "SDK.h "

Description： The puts function sends a character string in user define color whose address is specified in the argument string to the LCD display starting from the current cursor position. The cursor is moved accordingly as each character of string is sent to the LCD display. The operation continues until a terminating null character is encountered.

Returns： The puts function returns the number characters sent to the LCD display in user define color.

show_image_bmp

Purpose： Put a rectangular bitmap to the LCD display.

Syntax： void show_image_bmp(int left, int top, int width, int height, const void *pat);

Example call： show_image_ bmp (10,5,60,30,buffer);

Includes： #include "SDK.h "

Description： The showet_image function displays a rectangular bitmap specified by pat to the LCD display. The rectangular's top left position and size are specified by left, top, width, and height. The cursor position is not affected after the operation.

left	Display form the start point of X-axis.
top	Display form the start point of Y-axis.
width	Display the width form the start point.
height	Display the high form the start point.
pat	The buffer that you want to display data of image.

Returns： none

Notice： If you want to show a two bits file of BMP, you can change the format by using SDKUtility, and write into the buffer. After that, it will show on PA-20 /PA-2010 / PA-21 / PA-2110's LCD.

UserFont

DispFont_SetFont

Purpose：	Set user font from font file.
Syntax：	BOOL DispFont_SetFont(S32 slSelFont, const char *filename)
Example call：	DispFont_SetFont(2,"D:\\Fonts\\Font16.cft");
Includes：	#include "SDK.h "
Description：	slSelFont User Font 2~9
	filename User font file
Returns：	TRUE : success
	FALSE : fail

DispFont_GetFontInfo

Purpose：	Get font type,width and height.
Syntax：	BOOL DispFont_GetFontInfo(S32 slSelFont, S32* slType, S32* slWidth, S32* slHeight)
Example call：	DispFont_GetFontInfo(2,&Type,&Width,&slHeight);
Includes：	#include "SDK.h "
Description：	This function copies the slSelFont(0~9) info of font type ,width and height to the variables whose address is specified in the arguments slType , slWidth and slHeight.
Returns：	TRUE : success
	FALSE : fail
Notice：	Font type 0: 1 byte font file
	1: 2 bytes font file
	2: 2 bytes + 1bytes font file

TextBlock

TextBlock is a floating window printing rectangle area on screen. TextBlock defines it's activated area anywhere within LCD screen display. A out of display area definition is not allowed.

Each TextBlock has individual attribute definition for position, size, font , background color or bmp. There are total 16 TextBlocks. TextBlock(0) is system default Window. The setting of TextBlock(0) can't change. TextBlock(1~15) are user defined.

SetTextBlockMode	
Purpose：	Set TextBlock mode for single or multi layer.
Syntax：	void SetTextBlockMode(int slMode);
Example call：	SetTextBlockMode(0);
Includes：	#include "SDK.h "
Description：	This function can set the TextBlock mode for single or multi layer.
	Single layer is the same as PT-20, only backup TextBlock in only one layer. Multi layer can backup each TextBlock in it's buffers.
	When you change the mode setting, all the TextBlock will be reset.
Returns：	None
Notice：	slMode = 0, set mode for single layer.
	slMode = 1, set mode for multi layer.

GetTextBlockMode	
Purpose：	Get TextBlock mode.
Syntax：	int GetTextBlockMode(void);
Example call：	GetTextBlockMode();
Includes：	#include "SDK.h "
Description：	This function can get the TextBlock mode status.
Returns：	0:Single mode.
	1.Multi mode.

DefineTextBlock	
Purpose：	Define TextBlock setting.
Syntax：	BOOL DefineTextBlock(S32 slBlockNo,S32 slSelFont,S32 slBGType,U32* ulBGData,S32 slColumn,S32 slRow,S32 slXPos,S32 slYPos)
Example call：	DefineTextBlock(1,0,TYPE_IMAGE,(U32*)gausImage_PA20LOGO,10,8, 40,20);
Includes：	#include "SDK.h "

Description： The DefineTextBlock function defines font,background graph,size and position. There are total 15 Windows.

slBlockNo TextBlock number(1~15).

slSelFont Defined Font:

0~1 system font and 2~9 user font.

slBGType TYPE_COLOR -using background color.

TYPE_IMAGE -using background bmp.

TYPE_ORIGINAL –using default background.

ulBGData pointer of background color or bmp.

slColumn TextBlock column number.

slRow TextBlock row number.

slXPos TextBlock left-top X position in pixel(0~239).

slYPos TextBlock left-top Y position in pixel.

Status Bar enable:0~295.

Status Bar disable:0~319.

Returns： TRUE : success

FALSE : fail

Notice： It must define user font(2~9) before select user font number(2~9). The function define a TextBlock start at point(slXPos,slYPos) , width equal to slColumn * FontSize(width) and height equal to slRow * FontSize(height). The TextBlock also has background color or bmp and the specific font.

SetTextBlock

Purpose： Enable the specific TextBlock.

Syntax： BOOL SetTextBlock(S32 slBlockNo,BOOL bSF);

Example call： SetTextBlock(1,TRUE);

Includes： #include "SDK.h "

Description： This function will assign TextBlock area attribute and optionally store existing screen in TextBlock occupied area. This TextBlock will become the active TextBlock. A regular print function is assigned to this TextBlock.

slBlockNo TextBlock number(1~15).

bSF Save flag for save screen or not.

Returns： TRUE : success

FALSE : fail

Notice： When sets Multi layer mode, every TextBlocks will save their screens, whatever "bSF" is TRUE or FALSE.

ResetTextBlock

Purpose： Disable the specific TextBlock.

Syntax： void ResetTextBlock (S32 slBlockNo);

Example call： ResetTextBlock(1);

Includes： #include "SDK.h "

Description： When set sIngle layer mode, ResetTextBlock will disable the specific TextBlock and set the active TextBlock to 0. If an opened TextBlock has screen save, this function will restore previous screen from buffer in TextBlock occupied area.

When set multi layer mode, ResetTextBlock will disable the specific TextBlock and set the active TextBlock to last active TextBlock. If an opened TextBlock has screen save, this function will restore previous screen from buffer in TextBlock occupied area.

slBlockNo　　　TextBlock number(1~15).

Returns： None.

Notice： When set single layer mode, you only have one layer buffer to backup screen(in TextBlock internal program).If you set single layer mode and your TextBlock is overlapping, your backup screen will be covered by the last TextBlock. If you want to save screen, when Set TextBlock(SetTextBlock**),** please set the save flag(save screen) TRUE.

When set Multi layer mode, you have multi layer buffer to backup screen. System supports all TextBlock screen buffer. Each TextBlock screen will be saved.

PrintTextBlock

Purpose： Print Text to specific TextBlock.

Syntax： POINT PrintTextBlock(S32 slBlockNo, S32 slColumn, S32 slRow, char* string, U32 ulFontColor)

Example call： pt = PrintTextBlock(0,5,7,string,COLOR_BLACK);

Includes： #include "SDK.h "

Description： This function displays colorful text to specific TextBlock. The active TextBlock is not changed.

slBlockNo　　　TextBlock number(0~15).

slColumn　　　column.

slRow　　　row.

string　　　Text data pointer.

ulFontColor　　　Text color.

Returns： Function success: return the next position.

Function fail: return (-1,-1).

GetTextBlockCur

Purpose：	Get TextBlock current position.
Syntax：	POINT GetTextBlockCur(S32 slBlockNo);
Example call：	pt = GetTextBlockCur(3);
Includes：	#include "SDK.h "
Description：	This function can get position in specific TextBlock.
	slBlockNo TextBlock number(0~15).
Returns：	Function success: return the current position.
	Function fail: return (-1,-1).

SetTextBlockCur

Purpose：	Set specific TextBlock as active TextBlock and set position.
Syntax：	void SetTextBlockCur(S32 slBlockNo,S32 slColumn,S32 slRow)
Example call：	SetTextBlockCur(3,1,1);
Includes：	#include "SDK.h "
Description：	This function sets active TextBlock become to slBlockNo. The position of slBlockNo sets to (slColumn, slRow).
	slBlockNo TextBlock number(0~15)
	slColumn Column
	slRow Row
Returns：	None.

ShowTextBlockCursor

Purpose：	Show or hide TextBlock cursor.
Syntax：	void ShowTextBlockCursor(S32 slBlockNo, BOOL bShow, S32 slType)
Example call：	ShowTextBlockCursor(1,TRUE, 3);
Includes：	#include "SDK.h "
Description：	This function defines cursor type. Only the active TextBlock can show cursor.
	slBlockNo TextBlock number(0~15)
	bShow TRUE:show cursor
	FALSE:Hide cursor
	slType 0: Cursor off.
	1: Cursor on, and cursor type is a line as _.
	2: Cursor on, and cursor type is a line as |.
	3: Cursor on, and cursor type is a Window as ■.
Returns：	None.

TextBlock_SetBGColor

Purpose：	Set default background color.
Syntax：	void TextBlock_SetBGColor(S32 slColor);
Example call：	TextBlock_SetBGColor(COLOR_BLUE);

Includes： #include "SDK.h "

Description： This function can help you to set background color.

After use this function, all TextBlock will be reset.

TextBlock_SetBGImage

Purpose： Set default background image for bmp file.

Syntax： BOOL TextBlock_SetBGImage(S8* pssPath);

Example call： TextBlock_SetBGImage("D:\\Font\\BGImage.bmp");

Includes： #include "SDK.h "

Description： This function can help you to set background image.

If you want change the background image, you have to save a bmp file to

disk first. The bmp file need for 240 * 320 pixel.

After use this function, all TextBlock will be reset.

SwitchTextBlock

Purpose： Switch TextBlock.

Syntax： BOOL SwitchTextBlock(S32 slBlockNo);

Example call： SwitchTextBlock(1);

Includes： #include "SDK.h "

Description： This function can help you to switch the TextBlock that you want.

But the TextBlock has to be set and active before this function.

Returns： TRUE:Switch success.

FALSE:Switch fail. Please check the TextBlock is actived or not.

Communication Ports

clear_com

Purpose：	Clear receive buffer
Syntax：	void clear_com(int port);
Example call：	clear_com(1);
Includes：	#include "SDK.h "
Description：	This routine is used to clear all data stored in the receive buffer. This can be used to avoid mis-interpretation when overrun or other error occurred. Use the argument "port" as the connect port which is chosen to open . You can choose 1(RS232).
Returns：	None

close_com

Purpose：	To close specified communication port
Syntax：	void close_com(int port);
Example call：	close_com(1);
Includes：	#include "SDK.h "
Description：	The close_com disables the communication port specified. Use the argument "port" as the connect port which is chosen to open . You can choose 1(RS232).
Returns：	None

com_cts

Purpose：	Get CTS level
Syntax：	int close_com(int port);
Example call：	if (com_cts(1) = = 0) _printf("COM 1 CTS is space); else _printf("COM 1 CTS is mark");
Includes：	#include "SDK.h "
Description：	This routine is used to check current CTS level. Use the argument "port" as the connect port which is chosen to open. Now we only can choose 1(RS232).
Returns：	1 : allow to deliver 0 : not allow to deliver

com_eot

Purpose：	To see if any COM port transmission in process (End Of Transmission)
Syntax：	int com_eot(int port);
Example call：	while (com_eot(1) != 0x00); write_com(1,"NEXT STRING");
Includes：	#include "SDK.h "

Description:	This routine is used to check if prior transmission is still in process or not. Use the argument "port" as the connect port which is chosen to open . You can choose 1(RS232).
Returns:	0, prior transmission still in course
	1, transmission completed
	-1, the transmitting port choices error

com_overrun

Purpose:	See if overrun error occurred
Syntax:	int com_overrun(int port);
Example call:	if (com_overrun(1) > 0) clear_com(1);
Includes:	#include "SDK.h "
Description:	This routine is used to see if overrun met. The overrun flag is automatically cleared after examined. You can choose 1(RS232) or.
Returns:	1, overrun error met
	0, OK
	-1, the transmitting port choices error

com_rts

Purpose:	Set RTS signal
Syntax:	void com_rts(int port, int val);
Example call:	com_rts(1,1);
Includes:	#include "SDK.h "
Description:	This routine is used to control the RTS signal. It works even when the CTS flow control is selected. However, RTS might be changed by the background routine according to receiving buffer status. It is strongly recommended not to use this routine if CTS control is utilized. Use the argument "port" as the connect port which is chosen to open. Now we only can choose 1(COM).
	The argument "val" is set up RTS, 1 is ok for receiving data; 0 is error.
Returns:	None

nwrite_com

Purpose:	Send a specific number of characters out through RS232 port
Syntax:	int nwrite_com(int port, char *s, int count);
Example call:	char s[20]={"Hello World\n"};
	nwrite_com(1,s,5);/*send string "Hello" to connect port*/
Includes:	#include "SDK.h "
Description:	This routine is used to send a specific number of characters specified by count through RS232 ports. If any prior transmission is still in process, it is terminated then the current transmission resumes. The character string is

transmitted one by one until the specified number of character is sent. Use the argument "port" as the connect port which is chosen to open. You can choose 1(RS232). The argument "count" is the number of words of sending data.

If you select RS-232 and set hardware flow control(CTS/RTS), this function will wait while data be transmitted.

Returns： -1 : error

Other value: the number of words that success writing into.

open_com

Purpose： Initialize and enable specified RS232 port

Syntax： int open_com(int com_port, int setting);

Example call： open_com(1,0x0b);/*openCOM1，baud rate 38400,8 data bits,no parity,no handshake*/

Includes： #include "SDK.h "

Description： The open_com function initializes the specified RS-232 port. It clears the receive buffer, stops any data transmission under going, reset the status of the port, and set the RS-232 specification according to parameters set. Use the argument "port" as the connect port which is chosen to open. You can choose 1(RS232).

Each bit of the argument "setting"：

D0	baud rate	0 :115200	1-2 : 57600
~		3 : 38400	4 : 19200
D2		5 : 9600	6-7 : 4800
D3	data bits	0 : 7bits	1 : 8bits
D4	Parity enable	0 : disable	1 : enable
D5	even / odd	0 : odd	1 : even
D6	flow control	0 : disabe	1 : enable
D7	flow control method	0 : CTS/RTS	1 : Reserved

Returns： 0 : Open fail

1 : Open success

Remark： When flow control set up disable and flow control method set up CTS/RTS.

When you use IR port, the baud rate only can be setted 115200, flow control disable.

read_com

Purpose： Read 1 byte from the RS232 receive buffer

Syntax： int read_com(int port, char *c);

Example call： char c;

int i;

> i = read_com(1,c);
>
> if (i) _printf("char %c received from COM1",*c);

Includes：	#include "SDK.h "
Description：	This routine is used to read one byte from the receive buffer and then remove it from the buffer. However, if the buffer is empty, no action is taken and 0 is returned. Use the argument "port" as the connect port which is chosen to open. You can choose 1(RS232).
Returns：	1, available or 0 if buffer is empty

write_com

Purpose：	Send a string out through RS232 port
Syntax：	int write_com(int port, char *s);
Example call：	char s[20]={"Hello World\n"};
	write_com(1,s);
Includes：	#include "SDK.h "
Description：	This routine is used to send a string through RS232 ports. If any prior transmission is still in process, it is terminated then the current transmission resumes. The character string is transmitted one by one until a NULL character is met. A null string can be used to terminate prior transmission. Use the argument "port" as the connect port which is chosen to open. You can choose 1(RS232).
	If you select RS-232 and set hardware flow control(CTS/RTS), this function will wait while data be transmitted.
Returns：	-1 : error
	Other value: the number of words that success writing into.

USB_Open

Purpose：	Initialize and enable USB port.
Syntax：	void USB_Open(void);
Example call：	USB_Open();
Includes：	#include "SDK.h "
Description：	The USB_Open function initializes and enable USB port.
Returns：	None

USB_Close

Purpose：	To close USB port
Syntax：	void USB_Close(void);
Example call：	USB_Close();
Includes：	#include "SDK.h "
Description：	The USB_Close function disable and suspend USB port.
Returns：	None

USB_Read

Purpose：	Read specific number of bytes from USB port.
Syntax：	int USBRead(unsigned char *rbuf, unsigned int rLength);
Example call：	int k;
	unsigned char ausBuf[12];
	k = USBRead(ausBuf, 10);
	_printf("Read %d characters => %s", k, ausBuf);
Includes：	#include "SDK.h "
Description：	The function can write specific number of bytes from USB port.
Returns：	The USBRead function returns the number characters from the PC site

USB_Write

Purpose：	Write specific number of bytes to USB port.
Syntax：	void USBWrite(unsigned char *wbuf, unsigned int wLength);
Example call：	USBWrite("0123456789", 10);
Includes：	#include "SDK.h "
Description：	The function can write specific number of bytes to USB port.
Returns：	None

Remote

SetRemoteBaud

Purpose：	Setting the RemoteLink baud rate.
Syntax：	void SetRemoteBaud(int slBaud);
Example call：	SetRemoteBaud (0);
Includes：	#include "SDK.h "
Description：	This function can set RemoteLink baud rate(when use com port), the baud rate setting as follows:

slBaud	Baud rate(bps)
0	115200
1	57600
2	38400
3	19200
4	9600
5	4800

Returns：	None

GetRemoteBaud

Purpose：	Getting the RemoteLink baud rate.
Syntax：	int GetRemoteBaud(void);
Example call：	Switch(GetRemoteBaud())

```
{
  Case 0:
     _printf_color(COLOR_RED, "RemoteLink baud rate=115200");
     Break;
  Case 1:
     _printf_color(COLOR_RED, "RemoteLink baud rate=57600");
     Break;
  Case 2:
     _printf_color(COLOR_RED, "RemoteLink baud rate=38400");
     Break;
  Case 3:
     _printf_color(COLOR_RED, "RemoteLink baud rate=19200");
     Break;
  Case 4:
     _printf_color(COLOR_RED, "RemoteLink baud rate=9600");
     Break;
```

Case 5:

    _printf_color(COLOR_RED, "RemoteLink baud rate=4800");

    Break;

}

Includes：	#include "SDK.h "
Description：	This function can get RemoteLink baud rate(when use com port).
Returns：	0:115200 bps
	1:57600 bps
	2:38400 bps
	3:19200 bps
	4:9600 bps
	5:4800 bps

SetRemoteInterface

Purpose：	Setting the RemoteLink interface.
Syntax：	void SetRemoteInterface(int slInterface);
Example call：	SetRemoteInterface(int slInterface);
Includes：	#include "SDK.h "
Description：	This function can set RemoteLink connect interface, the interface setting as follows:

slInterface	Interface
0	RS-232
1	USB

Returns：	None

GetRemoteInterface

Purpose：	Getting the RemoteLink interface.
Syntax：	int GetRemoteInterface(void);
Example call：	Switch(GetRemoteInterface ())
	{
	Case 0:
	_printf_color(COLOR_RED, "RemoteLink Interface is RS-232");
	Break;
	Case 1:
	_printf_color(COLOR_RED, "RemoteLink Interface is USB");
	Break;
	}
Includes：	#include "SDK.h "
Description：	This function can set RemoteLink connect interface.

Returns： 0：RS-232

1：USB

RemoteLink

Purpose：	Use RemoteLink to call the transmission function for user to upload or download files.
Syntax：	void RemoteLink(void);
Example call：	RemoteLink( );
Includes：	#include "SDK.h "
Description：	The RemoteLink function provides the transmission environment to link with PT-FilaManager and make file uploading or downloading.
Returns：	Use RemoteLink to call the transmission function for user to upload or download files.

RemoteLink_RealTime

Purpose：	Use RemoteLink_RealTime can transfer file in any state.
Syntax：	void RemoteLink_RealTime(BOOL bStatus);
Example call：	RemoteLink_RealTime(TRUE);
Includes：	#include "SDK.h "
Description：	This function can set real time RemoteLink enable/disable, the 'bStatus' setting as follows:

FALSE	Real time RemoteLink disable
TRUE	Real time RemoteLink enable.

Returns：	None

System

Two time variables are declared by the system, which can be used for counting time. As they are updated by system clock, Don't write to them.

· **extern volatile unsigned long sys_msec;**   // in unit of 5 ms

· **extern volatile unsigned long sys_sec;**   // in unit of 1 second

These two variables are cleared to 0 upon power up.

SysSuspend	
Purpose：	Shut down the system.
Syntax：	void SysSuspend(void);
Example call：	SysSuspend( );
Includes：	#include "SDK.h "
Description：	This function will shut down the system.When power on, the system will resume or restart itself, depending on the system setting.
Returns：	None.

SysDelay	
Purpose：	Set system delay time.
Syntax：	void SysDelay(unsigned int ulTime);
Example call：	SysDelay(6000);//delay 6 seconds.
Includes：	#include "SDK.h "
Description：	The function can delay system, its unit is millisecond.
Returns：	None

SetPowerOnState	
Purpose：	Set power on for resume or restart.
Syntax：	void SetPowerOnState(int slState);
Example call：	SetPowerOnState(0);//Power on for resume.
Includes：	#include "SDK.h "
Description：	This function can set power on status for resume or restart.
	slState for 0:Set power on resume.
	slState for 1:Set power on restart.
Returns：	None

GetPowerOnState	
Purpose：	Get power on status
Syntax：	int GetPowerOnState(void);
Example call：	if (GetPowerOnState())
	_printf_color(COLOR_BLACK, "Power on for restart:");

Includes：	#include "SDK.h "
Description：	This function can get power on status for resume or restart.
Returns：	0:Resume
	1:Restart

SetAutoPWOff

Purpose：	Set auto power off timer.
Syntax：	void SetAutoPWOff(int slTimer);
Example call：	SetAutoPWOff(0);//Always on
Includes：	#include "SDK.h "
Description：	This function can set auto power off timer. The range is from 30 to
	65535(s),
	If the value is 0, that means always on.
Returns：	None

GetAutoPWOff

Purpose：	Get auto power off timer.
Syntax：	int GetAutoPWOff(void);
Example call：	int slAPO;
	slAPO = GetAutoPWOff( );
Includes：	#include "SDK.h "
Description：	This function can get auto power off timer.
Returns：	Auto power off timer(s).

SetStatusBAR

Purpose：	Set statusbar display/no display.
Syntax：	void SetStatusBAR(int slStatus);
Example call：	SetStatusBAR(TRUE);//Statusbar on.
Includes：	#include "SDK.h "
Description：	This function can set statusbar display or on display.
	If use this function, all of the TextBlock setting will be reset.
Returns：	None.

GetStatausBAR

Purpose：	Get statusbar display status.
Syntax：	int GetStatusBAR(void);
Example call：	if (GetStatusBAR())
	_printf_color(COLOR_BLACK, "Statusbar on");
Includes：	#include "SDK.h "
Description：	This function can get statusbar display status.
Returns：	TRUE: Statusbar display
	FALSE: Statusbar no display

SN_Get

Purpose：	To get the SN of PT10/12.
Syntax：	void SN_Get(char *pssSNBuffer);
Example call：	SN_Get(SNBuffer);
Includes：	#include "SDK.h "
Description：	The function cab get the SN of PA-20 /PA-2010 / PA-21 / PA-2110. The string buffer size must longer than 8 bytes.
Returns：	None

BIOS_SetDefault

Purpose：	Set BIOS setting default.
Syntax：	void BIOS_SetDefault(void);
Example call：	BIOS_SetDefault();
Includes：	#include "SDK.h "
Description：	This function can set BIOS setting to default setting. It takes several seconds.
Returns：	None

Check_AID

Purpose：	Check the agency ID correct or not.
Syntax：	BOOL Check_AID(char* pssUser, char* pssPassword);
Example call：	if (Check_AID("USER01", "AAAAAAAA")) _printf_color(COLOR_BLACK, "AID Correct!!");
Includes：	#include "SDK.h "
Description：	This function can check the agency ID correct or not. You can write the agency ID by "AID Maker". The pssUser and pssPassword need 4~8 characters.
Returns：	TRUE: AID correct. FALSE: AID not correct.

Memory

Tfree	
Purpose：	Use the Tfree to release an allocated storage block to the pool of free memory.
Syntax：	void Tfree(void *mem_address);
Example call：	Tfree(buffer);
Includes：	#include "SDK.h"
Description：	The Tfree function returns to the pool of free memory a blockof memory that was allocated earlier by Tmalloc. The address of the block is specified by the argument mem_address, which is a pointer to the starting byte of the block. A NULL pointer argument is ignored by Tfree.
Returns：	None

Tmalloc	
Purpose：	Use Tmalloc to allocate memory for an array of a given number of bytes, not exceeding 256KB.
Syntax：	void* Tmalloc(U32 num_bytes);
Example call：	buffer = (char *)Tmalloc(100*sizeof(char));
Includes：	#include "SDK.h"
Description：	The Tmalloc function allocates the number of bytes requested in the argument num_bytes by calling internal Turbo C heap management routines. The Tmalloc function will work properly for all memory models.
Returns：	The Tmalloc function returns a pointer that is the starting address of the memory allocated. The allocated memory is properly aligned (the address of the first byte meets the requirements for storing any type of C variable). If the memory allocation is unsuccessful because of insufficient space or bad values of the arguments, a NULL is returned.
Comments：	Note that when using Tmolloc to allocate storage for a specific data type, you should cast the returned void pointer to that type.

TotalHeapSize	
Purpose：	Checking the total heap size.
Syntax：	int TotalHeapSize(void);
Example call：	totalsize = TotalHeapSize();
Includes：	#include "SDK.h "
Description：	The TotalHeapSize function can get the total heap size.
Returns：	The total heap size in units of Kbytes.

UsedHeapSize	

Purpose：	Checking the used heap size.
Syntax：	int UsedHeapSize(void);
Example call：	usedsize = UsedHeapSize();
Includes：	#include "SDK.h "
Description：	The UsedHeapSize function can get the used heap size.
Returns：	The used heap size in units of Kbytes.

FreeHeapSize

Purpose：	Checking the free heap size.
Syntax：	int FreeHeapSize(void);
Example call：	freesize = FreeHeapSize();
Includes：	#include "SDK.h "
Description：	The UsedHeapSize function can get the used heap size.
Returns：	The total heap size in units of Kbytes.

Vibrate

on_vibrator

Purpose：	Use on_vibrator to set vibrator on.
Syntax：	void on_vibrator(void);
Example call：	on_vibrator( );
Includes：	#include "SDK.h "
Description：	Use on_vibrator function can enable vibrator. On timer is set by set_vibrator_timer function.
Returns：	None.

off_vibrator

Purpose：	Use off_vibrator to set vibrator off.
Syntax：	void off_vibrator(void);
Example call：	off_vibrator( );
Includes：	#include "SDK.h "
Description：	Use off_vibrator function can daisable vibrator.
Returns：	None.

set_vibrator_timer

Purpose：	Use set_vibrator_timer to set vibrator on timer.
Syntax：	void set_vibrator_timer(unsigned char usTimer);
Example call：	set_vibrator_timer(10);//Set vibrator on timer for 1 sec.
Includes：	#include "SDK.h "
Description：	Use set_vibrator_timer function can set vibrator on timer. For example, set 10 for on 1 sec.
Returns：	None.

get_vibrator_timer

Purpose：	Use get_vibrator_timer to get vibrator on. timer
Syntax：	unsigned char get_vibrator_timer(void);
Example call：	i = get_vibrator_timer( );
Includes：	#include "SDK.h "
Description：	Use get_vibrator_timer function can get vibrator on timer.
Returns：	Vibrator on timer.

Other

prc_menu_color

Purpose： Create a menu-driven interface.

Syntax： void prc_menu_color(MENU_COLOR *menu);

Example call： void FuncMenu_01(void)

{

/*to do :add your own program code here*/

}

void FuncMenu_02(void)

{

/*to do :add your own program code here*/

}

void FuncMenu_03(void)

{

/*to do :add your own program code here*/

}

MENU_ENTRY_COLOR Menu_01 = {0,1,"1.Test Menu

01",&FuncMenu_01,0};

MENU_ENTRY_COLOR Menu_02 = {0,2,"2.Test Menu

02",&FuncMenu_02,0};

MENU_ENTRY_COLOR Menu_03 = {0,3,"3.Test Menu

03",&FuncMenu_03,0};

void prc_menu_Test(void)

{

    MENU_COLOR Menu_Test = {3,1,0,"Menu Test!!", COLOR_BROWN,

    COLOR_BLACK, COLOR_LIGHTBLUE, {&Menu_01, &Menu_02,

    &Menu_03}};

     prc_menu_color (&Menu_Test);

}

Includes： #include "SDK.h "

Description： The prc_menu_color function is used to create a user-defined menu.

SMENU_COLOR and SMENU_ENTRY_COLOR structures are defined

in "SDK.h". Users can just fill the SMENU_ENTRY_COLOR structure and

call the prc_menu function to build a hierarchy menu-driven user interface.

Returns：	None

Purpose：	Set function "prc_menu_color" ENT key status when use number key to select menu.
Syntax：	void prc_menu_Set_SelectWithEnt(BOOL bENT);
Example call：	prc_menu_Set_SelectWithEnt(TRUE);
Includes：	#include "SDK.h "
Description：	Use prc_menu_Set_SelectWithEnt function can set prc_menu_color ENT key status when use number key to select menu.
Returns：	None

Purpose：	Get function "prc_menu_color" ENT key status when use number key to select menu.
Syntax：	BOOL prc_menu_Get_SelectWithEnt(void);
Example call：	i = prc_menu_Get_SelectWithEnt ( );
Includes：	#include "SDK.h "
Description：	Use prc_menu_Get_SelectWithEnt function can get prc_menu_color ENT key status when use number key to select menu.
Returns：	Get prc_menu_color with ENT key status.

Simulator (Only for PC Simulator)

CopyFileToTerminal

Purpose：	Use BackupDataFiletoPC to copy data file to C:\Data directory in PC.
Syntax：	void CopyFileToTerminal(char *pssPCFileName, char *pssPDTFileName);
Example call：	CopyFileToTerminal("..\\Lookup\\MenuLook.dat",
	"D:\\Lookup\\MenuLook.dat");
Includes：	#include "SDK.h"
Description：	The CopyFileToTerminal function copies the PC file path specified by pssPCFileName pointer to the simulator path specified by pssPDTFileName pointer.
Returns：	None

BackupDataFiletoPC

Purpose：	Use BackupDataFiletoPCA to copy data file to any disc in PC.
Syntax：	void BackupDataFiletoPC(char *pTerminalFile, char *pPCFileName);
Example call：	BackupDataFiletoPC("c:\\data\\test1.dat","f:\\sample\\test1.dat ");
Includes：	#include "SDK.h"
Description：	The BackupDataFiletoPC function copies the simulator datafile path specified by pTerminalFile to the pFileName in PC, and you need to store with the same file name.
Returns：	None

Sim_SetScannerType

Purpose：	To setting scan module type.
Syntax：	void Sim_SetScannerType(int type);
Example call：	Sim_SetScannerType(1); //Scan module type is Laser
Includes：	#include "SDK.h"
Description：	"type" is an integer variable indicating the scan module type.

type	Meaning
0	CCD module
1	Laser module

Returns：	None

Data Conversion

__itoa

Purpose： Use __itoa to convert an integer value to a null-terminated character string.

Syntax： char * __itoa (int value, char *string, int radix);

Example call： __itoa(32, buffer, 16); /* buffer will contain "20" */

Includes： #include "SDK.h"

Description： The __itoa function converts the int argument value into a null-terminated character string using the argument radix as the base of the number system. The resulting string with a length of up to 17 bytes is saved in the buffer whose address is given in the argument string. You must allocate enough room in the buffer to hold all digits of the converted string plus the terminating null character (\0). For radixes other than 10, the sign bit is not interpreted; instead, the bit pattern of value is simply expressed in the requested radix. The argument radix specifies the base (between 2 and 36) of the number system in which the string representation of value is expressed. For example, using either 2, 8, 10, or 16 as radix, you can convert value into its binary, octal, decimal, or hexadecimal representation, respectively. When radix is 10 and the value is negative, the converted string will start with a minus sign.

Returns： The __itoa function returns the pointer to the string of degits (i.e., it returns the argument string).

__ltoa

Purpose： Use __ltoa to convert a long integer value to a null-terminated character string.

Syntax： char * __ltoa (long value, char *string, int radix);

Example call： __ltoa(0x10000, string, 10); /* string = "65536" */

Includes： #include "SDK.h"

Description： The __ltoa function converts the long argument value into acharacter string using the argument radix as the base of the number system. A long integer has 32 bits when expressed in radix 2, so the string can occupy a maximum of 33 bytes with the terminating null character. The resulting string is returned in the buffer whose address is given in the argument string. The argument radix specifies the base (between 2 and 36) of the number system in which the string representation of value is expressed. For example, using either 2, 8, 10,or 16 as radix, you can convert value into its binary, octal, decimal, or hexadecimal representation, respectively.

When radix is 10 and the value is negative, the converted string will start

with a minus sign.

Returns： The __ltoa function returns the pointer to the converted string (i.e., it

returns the argument string).

__ultoa

Purpose： Use __ultoa to convert an unsigned long integer value to a character string.

Syntax： char * __ultoa (unsigned long value, char *string, int radix);

Example call： __ultoa(0x20000, string, 10); /* string = "131072" */

Includes： #include "SDK.h"

Description： The __ultoa function converts the unsigned long argument value into a

null-terminated character string using the argument radix as the base of the

number system. A long integer has 32 bits when expressed in radix 2, so

the string can occupy a maximum of 33 bytes with the terminating null

character. The resulting string is returned by __ultoa in the buffer whose

address is given in the argument string. The argument radix specifies the

base (between 2 and 36) of the number system in which the string

representation of value is expressed. For example, using either 2, 8, 10, or

16 as radix, you can convert value into its binary, octal, decimal, or

hexadecimal representation, respectively.

Returns： The __ultoa function returns the pointer to the converted string (i.e., it

returns the argument string).

APPENDIX 1：

Scan Module (CCD) Configuration Table

Command1	Command2	Value
5 Indication	1 Power on alert	0:On * 1:None 2:Off
	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	7 Code ID position	0: Before code data * 1: After code data
	8 Code ID transmission	0: Disable * 1: Proprietary ID 2: AIM ID
	9 Code length transmission	0: Disable * 1: Enable
	10 Code name transmission	0: Disable * 1: Enable
	11 Case conversion	0: Disable * 1: Upper case 2. Lower case
7 Scan	4 Double confirm	0 ~ 9 0 *
	6 Global min. code length	0 ~ 63 4 *
	7 Global max. code length	0 ~ 63 63 *
	8 Inverted image scan	0: Disable * 1: Enable
8 String setting	2 Suffix characters setting	0 * 0x00 ~ 0xff ASCII code 22 characters.

	3 Preamble characters settings	0 * 0x00 ~ 0xff ASCII code 22 characters.
	4 Postamble characters settings	0 * 0x00 ~ 0xff ASCII code 22 characters.
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: One digit 2: Two digits
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<O> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 20 0 *
	7 Truncate ending	0 ~ 15 0 *

	8 Code ID setting	<*> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: Full ASCII
	13 Start/stop transmission	0: Disable * 1: Enable
12 Code 93	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<&> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
13 Code 128	1 Read	0: Disable 1: Enable *
	2 Check-sum verification	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *

	8 Code ID setting	<#> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Format	0: Standard * 1: UCC.EAN 128
	12 UCC/EAN 128 ID setting	<#> 0x00 ~ 0xff ASCII code(1 bytes)
	13 Concatenation code	0x1D * 0x00 ~ 0xff ASCII code(1 bytes)
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check-sum verification	0: Disable * 1: Enable
	3 Check-sum transmission	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 0 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<%> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Start/stop type	0: ABCD/ABCD * 1: abcd/abcd 2: ABCD/TN*E 3: abcd/tn*e
	11 Start/stop transmission	0: Disable * 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *

		7 Truncate ending	0 ~ 15 0 *
		8 Code ID setting	<FF> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
		10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
		11 Truncation/expansion	0: None * 1: Truncate leading zero 2: Expand to EAN 13
		12 Expansion	0: Disable * 1: Enable
16 EAN 13		1 Read	0: Disable 1: Enable *
		3 Check-sum transmission	0: Disable 1: Enable *
		6 Truncate leading	0 ~ 15 0 *
		7 Truncate ending	0 ~ 15 0 *
		8 Code ID setting	<F> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
		10 Supplement digits	0: None * 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
		12	0: Disable *

		ISBN/ISSN conversion	1: Enable
17 Industrial 2 of 5	1 Read	0:Disable * 1:Enable	
	4 Max. code length	0 ~ 64 0 *	
	5 Min. code length	0 ~ 64 0 *	
	6 Truncate leading	0 ~ 15 0 *	
	7 Truncate ending	0 ~ 15 0 *	
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)	
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *	
	2 Check-sum verification	0: Disable * 1: Enable	
	3 Check-sum transmission	0: Disable * 1: Enable	
	4 Max. code length	0 ~ 64 0 *	
	5 Min. code length	0 ~ 64 0 *	
	6 Truncate leading	0 ~ 15 0 *	
	7 Truncate ending	0 ~ 15 0 *	
	8 Code ID setting	<i> 0x00 ~ 0xff ASCII code(1 or 2 bytes)	
19 Standard 2 of 5	1 Read	0: Disable * 1: Enable	
	2 Check-sum verification	0: Disable * 1: Enable	
	3 Check-sum transmission	0: Disable * 1: Enable	
	4	0 ~ 64	

		Max. code length	0 *
	5		0 ~ 64
		Min. code length	0 *
	6		0 ~ 15
		Truncate leading	0 *
	7		0 ~ 15
		Truncate ending	0 *
	8		<i>
		Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
20 MSI Plessey	1		0: Disable *
		Read	1: Enable
	2		0: Disable
		Check-sum verification	1: Mod 10 *
			2: Mod 10/10
			3: Mod 11/10
	3		0: Disable *
		Check-sum transmission	1: Enable
	4		0 ~ 64
		Max. code length	0 *
	5		0 ~ 64
		Min. code length	0 *
	6		0 ~ 15
		Truncate leading	0 *
	7		0 ~ 15
		Truncate ending	0 *
	8		<@>
		Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
21 UK Plessey	1		0: Disable *
		Read	1: Enable
	2		0: Disable
		Check-sum verification	1: Enable *
	3		0: Disable *
		Check-sum transmission	1: Enable
	4		0 ~ 64
		Max. code length	0 *
	5		0 ~ 64

		Min. code length	0 *
	6	Truncate leading	0 ~ 15 0 *
	7	Truncate ending	0 ~ 15 0 *
	8	Code ID setting	<@> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
22 Telepen	1	Read	0: Disable * 1: Enable
	2	Check-sum verification	0: Disable * 1: Enable
	3	Check-sum transmission	0: Disable * 1: Enable
	4	Max. code length	0 ~ 64 0 *
	5	Min. code length	0 ~ 64 0 *
	6	Truncate leading	0 ~ 15 0 *
	7	Truncate ending	0 ~ 15 0 *
	8	Code ID setting	<S> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10	Format	0: Numeric * 1: Full ASCII
23 UPCA	1	Read	0: Disable 1: Enable *
	3	Check-sum transmission	0: Disable 1: Enable *
	6	Truncate leading	0 ~ 15 0 *
	7	Truncate ending	0 ~ 15 0 *
	8	Code ID setting	<A> 0x00 ~ 0xff ASCII code(1 or 2 bytes)

	10 Supplement digits	0: None *
		1: 2 digits
		2. 5 digtis
		3: 2, 5 digits
		4: UCC/EAN 128
		5: 2, UCC/EAN 128
		6: 5, UCC/EAN 128
		7: All
	11 Truncate/expansion	0: None
		1: Truncate leading zero *
		2: Expand to EAN 13
24 UPCE	1 Read	0: Disable
		1: Enable *
	3 Check-sum transmission	0: Disable
		1: Enable *
	6 Truncate leading	0 ~ 15
		0 *
	7 Truncate ending	0 ~ 15
		0 *
	8 Code ID setting	<E>
		0x00 ~ 0xff ASCII code(1 or 2
		bytes)
	10 Supplement digits	0: None *
		1: 2 digits
		2: 5 digits
		3: 2, 5 digits
		4: UCC/EAN 128
		5: 2, UCC/EAN 128
		6: 5, UCC/EAN 128
		7: All
	11 Truncate/expansion	0: None *
		1: Truncate leading zero
		2: Expand to EAN 13
		3: Expand to UPCA
	12 Expansion	0: Disable *
		1: Enable
	13 UPCE-1	0: Disable *
		1: Enable

25	1	0: Disable *
Matrix 25	Read	1: Enable
	2	0: Disable *
	Check-sum verification	1: Enable
	3	0: Disable *
	Check-sum transmission	1: Enable
	4	0 ~ 64
	Max. code length	0 *
	5	0 ~ 64
	Min. code length	0 *
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
	Truncate ending	0 *
	8	<B>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
28	1	0: Disable *
China post	Read	1: Enable
	4	0 ~ 64
	Max. code length	11 *
	5	0 ~ 64
	Min. code length	11 *
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
	Truncate ending	0 *
	8	<t>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
29	1	0: Disable *
RSS 14	Read	1: Enable
	6	0 ~ 15
	Truncate leading	0 *
	7	0 ~ 15
	Truncate ending	0 *
	8	<R4>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2

		bytes)	
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable	
30 RSS Limited	1 Read	0: Disable * 1: Enable	
	6 Truncate leading	0 ~ 15 0 *	
	7 Truncate ending	0 ~ 15 0 *	
	8 Code ID setting	<RL> 0x00 ~ 0xff ASCII code(1 or 2 bytes)	
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable	
31 RSS Expanded	1 Read	0: Disable * 1: Enable	
	4 Max. code length	0 ~ 99 99 *	
	5 Min. code length	0 ~ 99 1 *	
	6 Truncate leading	0 ~ 15 0 *	
	7 Truncate ending	0 ~ 15 0 *	
	8 Code ID setting	<RX> 0x00 ~ 0xff ASCII code(1 or 2 bytes)	
	11 UCC/EAN 128 emulation	0: Disable * 1: Enable	
32 Italian Pharmacode 39	1 Read	0: Disable * 1: Enable	
	4 Max. code length	0 ~ 64 12 *	
	5 Min. code length	0 ~ 64 9 *	
	6 Truncate leading	0 ~ 15 0 *	

	7	0 ~ 15
	Truncate ending	0 *
	8	<p>
	Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10	0: Disable *
	Leading "A"	1: Enable

APPENDIX 2：

Scan Module (Laser) Configuration Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	7 Code ID position	0: Before code data * 1: After code data
	8 Code ID transmission	0: Disable * 1: Enable
	9 Code length transmission	0: Disable * 1: Enable
	10 Code name transmission	0: Disable * 1: Enable
	12 AIM ID	0: Disable * 1: Enable
7 Scan	6: Global min. code length	0~80 (0: Disable) 3*
	9: Global lock code length	0~80 (0: Disable) 0*
	10: Configurable code length #1	See Note1 (4 characters)
	11: Configurable code length #2	See Note1 (4 characters)
	12: Configurable code length #3	See Note1 (4 characters)
	13: Configurable code	See Note1 (4 characters)

	length #4	
	14: Configurable code length #5	See Note1 (4 characters)
	15: Configurable code length #6	See Note1 (4 characters)
	16: Configurable code length #7	See Note1 (4 characters)
	17 : Scan Timeout (Sec)	1~30 See Note2 2*
	18: Idle Timeout (Sec)	0~255 (0: Disable) See Note2 5*
8 String setting	3 Preamble characters settings	0x00 ~ 0xff ASCII code (1-10 characters) 00 *
	4 Postamble characters settings	0x00 ~ 0xff ASCII code (1-10 characters) 00 *
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check Digit	1: One digit * 2: Two digits
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1

			bytes)
	10 Full ASCII	0: Disable * 1: Enable	
	13 Transmit Start/Stop Characters	0: Disable * 1: Enable	
	14 Italian Pharmacode	0: Disable * 1: Enable	
	15 Italian Pharmacode Only	0: Disable 1: Enable *	
12 Code 93	1 Read	0: Disable 1: Enable *	
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)	
13 Code 128	1 Read	0: Disable 1: Enable *	
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)	
	10 UCC/EAN 128	0: Disable * 1: Enable	
	14 ISBT 128	0: Disable * 1: Enable	
	15 ISBT 128 Transmit Identifier Data	0: Disable 1: Enable *	
	16 ISBT 128 Pre-Defined Concatenation	0: Disable 1: Enable *	
	17 ISBT 128 Form:	0: =A + =% * 1: =A + &; 2: =A + &! 3: =< + => 4: =< + &> 5: &< + => 6: &< + &>	

14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	11 Transmit Start/Stop Characters	0: Disable * 1: Enable
	12 Dual Field	0: Disable * 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	13 Convert to EAN-13	0: Disable * 1: Enable
16 EAN 13	1 Read	0: Disable 1: Enable *
	3 Check-sum transmission	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	12 ISBN conversion	0: Disable * 1: Enable
	13 ISBN Supplement Required	0: Disable * 1: Enable
	14 ISMN conversion	0: Disable * 1: Enable

	15 ISMN Supplement Required	0: Disable * 1: Enable
17 Industrial 2 of 5	1 Read	0:Disable * 1:Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable 1: Mod 10 * 2: Mod 10/10
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
22 Telepen	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1

		bytes)
	10 Alpha Telepen	0: Disable * 1: Enable
23 UPCA	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	12 Convert to EAN-13	0: Disable * 1: Enable
	13 Coupon	0: Disable * 1: Enable See Note 3
	14 Coupon Transmit "]C1"	0: Disable 1: Enable *
	15 Transmit Number System	0: Disable 1: Enable *
24 UPCE	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	14 Convert to UPC-A	0: Disable * 1: Enable
	15 Transmit Number System	0: Disable 1: Enable *
25 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8	0: Disable*

	Code ID setting	0x20 ~ 0xff ASCII code(1 bytes)
35 UPC/EAN General	1 Supplements Required	0: Disable * 1: Enable
	2 Two Digit Supplements	0: Disable * 1: Enable
	3 Five Digit Supplements	0: Disable * 1: Enable
	4 Two Digit Redundancy	0: Disable * 1: Enable
	5 Five Digit Redundancy	0: Disable * 1: Enable
	6 GTIN Formatting	0: Disable * 1: Enable
36 IATA 2 of 5	1 Read	0: Disable * 1: Enable See Note 4
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
37 TRI-OPTIC	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable* 0x20 ~ 0xff ASCII code(1 bytes)
	10 Conversion	0: Disable * 1: Enable
38 RSS	1 Read	0: Disable * 1: Enable
	2 Transmit Application ID	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	4 Transmit Symbology ID "]e0"	0: Disable * 1: Enable

Note1:

There are seven bar code lock lengths available. Specific code type can be assigned to a lock length.

Code type:

	Hex
CODE11	64
CODE39	65
CODE93	66
CODE128	67
CODABAR	68
INDUSTRIAL_25	6B
INTERLEAVED_25	6C
MSI_PLESSEY	6E
UK_PLESSEY	6F
TELEPEN	70
MATRIX_25	73
TRIOPTIC	7E

Length:

0~80 (0: Disable)

Default setting:

"0000"

Example:

Cmd1	Cmd2	Value
7	10	"670C"
7	11	"6514"
7	12	"6710"

Code 128: Can only read 12 or 16 the length of the barcode.
Code39: Can only read 20 the length of the barcode.

Note2:

ScanTimeout:

The maximum time, in seconds, during which the laser remains on without decoding any barcode.

IdleTimeout:

The maximum time, in seconds, during which the scanner remains idle without any action.

Note3:

When this setting is enable the UCC / EAN 128 barcode can not be read.

Note4:

IATA 2 of 5 only support 13, 15 Digit.