ARGOX

Empower the Barcode

Portable Data Terminal

PI-1X

Programming Guide

Version: 2.05

http://www.argox.com

Preface

To satisfy the user’s customized needs, the PI-1X 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 PI-1X.

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 PI-1X for your reference.

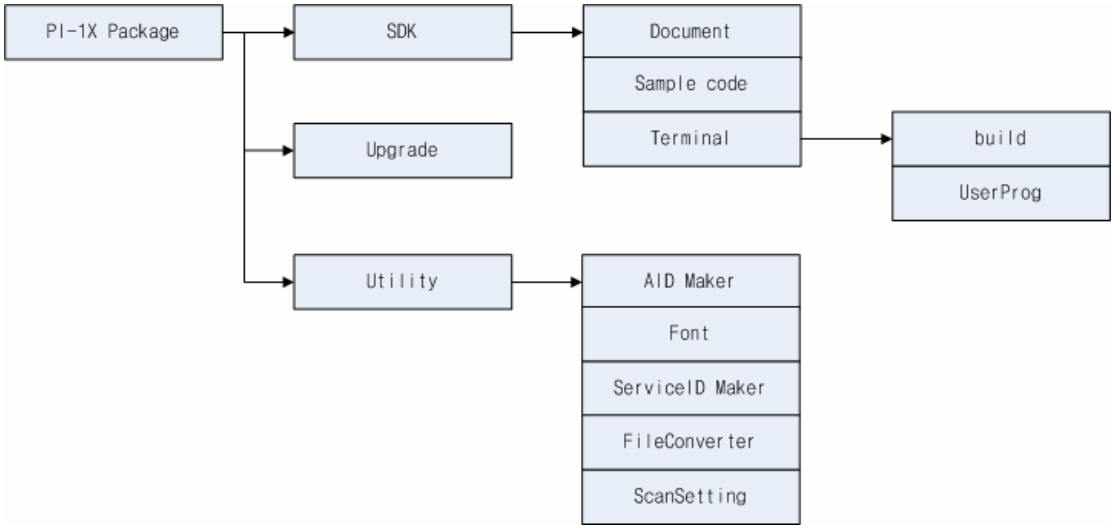
Table of Contents

Program Developing

Development Environment

Folder Structure:

When open the SDK folder in the CD provided with the PI-1X, it will show the structure as the following:



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 firmware upgrade.

  ✧ Upgrade BAT file:

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

  If you want to upgrade firmware, you have to set PI-1X in force mode or upgrade in supervisor menu and double click these bat file for download.

  You can download bin file by PhoenixVoler in these status, too.

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

- SDK:

  For SDK develop tools. It has two sub folders, "**Document**", "**Terminal**" and "**Sample code for Terminal**"

  ✧ Document:

It has one PDF file, “Terminal.pdf”, this PDF file is introduct about how to use SDK functions.

- ✧ Terminal:
  For terminal develop environment.
- ✧ Sample code for Terminal:
  For SDK sample, you can build and test it after download.

- Utility:
  It has 4 sub folders, “**AID Maker**”,” **Font**”,” **ScanSetting**”, “**FileConverter**”, and “**ServiceID Maker**”.
  - ✧ AID Maker:
    For set PI-1X agency ID.
  - ✧ Font:
    You can make your font for PI-1X to display.
  - ✧ ScanSetting:
    To create a file for all scan settings.
  - ✧ ServiceID Maker:
    For set PI-1X Service ID.
  - ✧ FileConverter :
    Convert Source file to Output file with specified format and field.

Adding Source File:

All user application program source files has to be placed under ***UserProg*** folder. If you want to set your source file in other folder, you have to modify the “makefile” under “build” folder.

Function Library

- PI-1X Function Library supports user application program to perform the data collection jobs. PI-1X Function Library provides variety of services, and accomplishes special functions according to specific demands.

- When using the PI-1X Function Library, please add the import command (“LIB_CL.h”, “DBMS.h” and “Define.h”) into the user program file (*.c) and the function will be imported. In this case, the PI-1X Function Library file ***lib_PI1X.a*** is needed.

- The PI-1X Function Library file ***Lib_PI1X.a*** is updated occasionally. For most update version, please ask helps from your vendor or the manufacturer.

- PI-1X Function Library file ***Lib_PI1X.a*** is needed during compiling and linking for generating App.bin.

- The update library release is LIB_CL.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 (GCC cross compiler). After set up the developing environment, you can use the include head file of standard C language function library .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. Install compiler

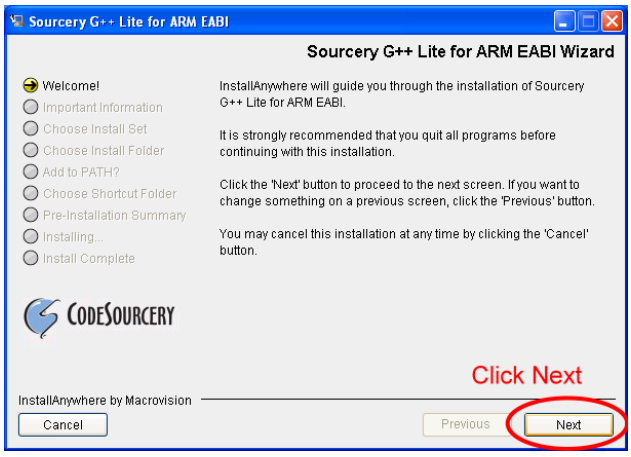
- Sourcery G++ Lite:

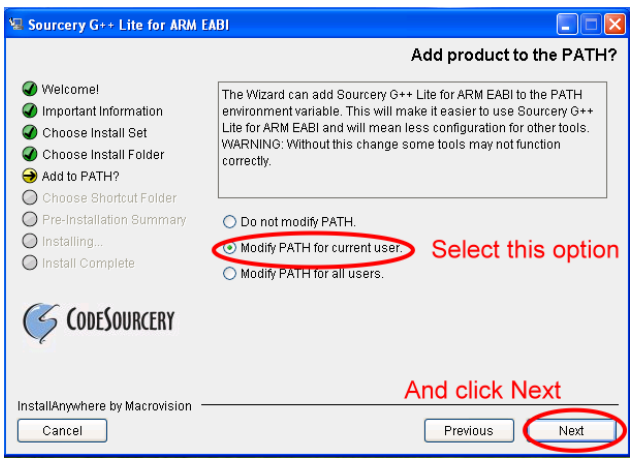
  First, you have to get “Sourcery G++ Lite” and install it. You can get the install file in the path:

  http://www.codesourcery.com/sgpp/lite/arm/portal/package6496/public/arm-none-eabi/arm-2010q1-188-arm-none-eabi.exe

  Double click this file, 

  and you will start install program. All options select default.





  Then you can finish this install.

- Eclipse IDE for C/C++ Developers:

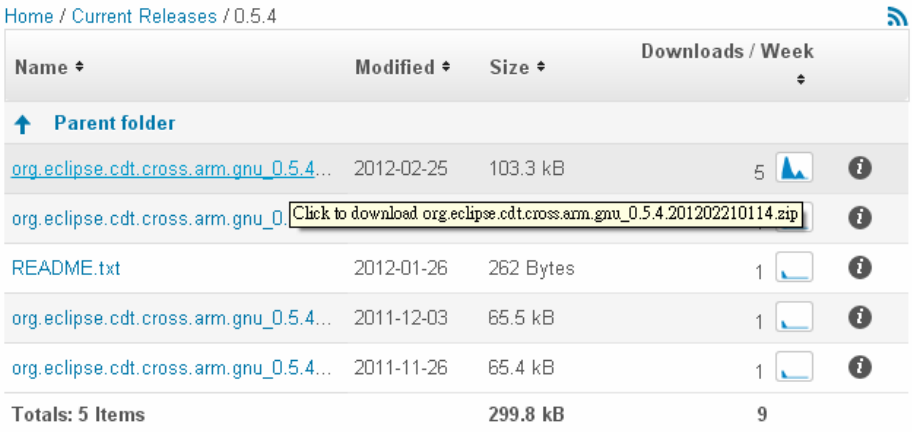
  Before install “Eclipse IDE for C/C++ Developers”, if you don’t install “JAVA virtual machine” yet, you must install “JAVA virtual machine” first.

Please go to this web site

https://java.com/en/

After install “JAVA virtual machine”, you need a plug in file for Eclipse, you can find it in this web site

http://sourceforge.net/projects/gnuarmeclipse/files/Current%20Releases/0.5.4/



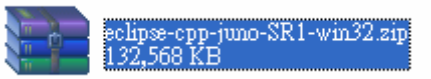
We need the file “org.eclipse.cdt.cross.arm.gnu_0.5.4.201202210114.zip”.

Now, you can download the final file “Eclipse IDE for C/C++ Developers”.

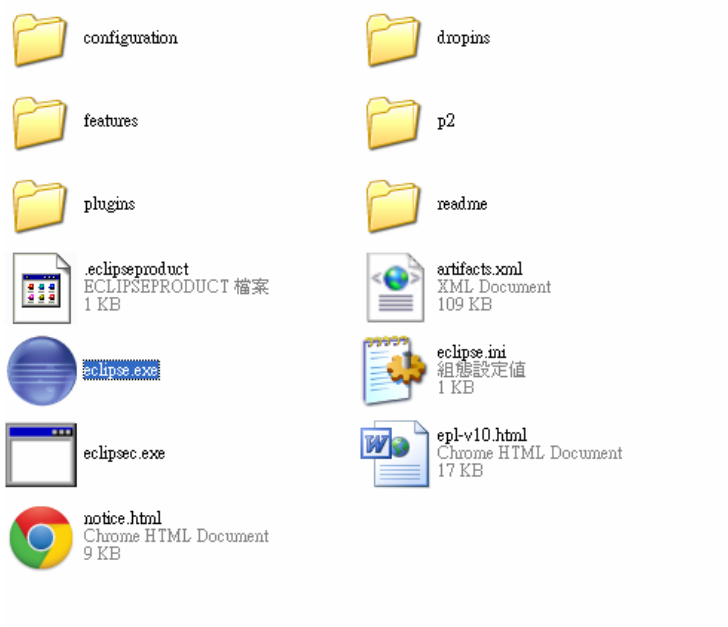
The website is in

http://www.eclipse.org/downloads/packages/release/Juno/SR1

After these actions, we will start to set Eclipse.

Please unzip this file . 

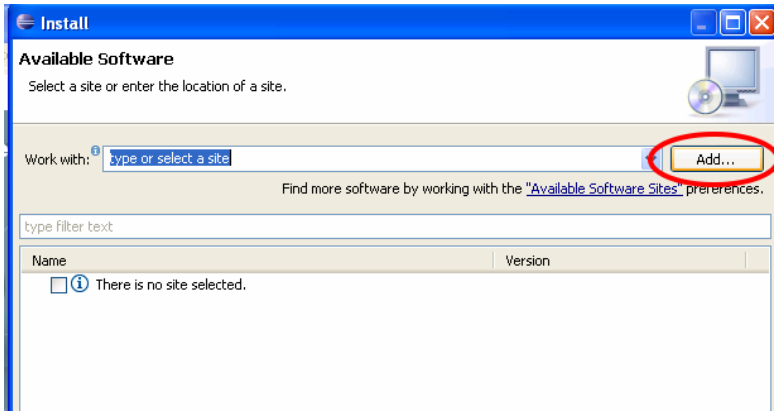
After unzip this file, please double click eclipse.exe.



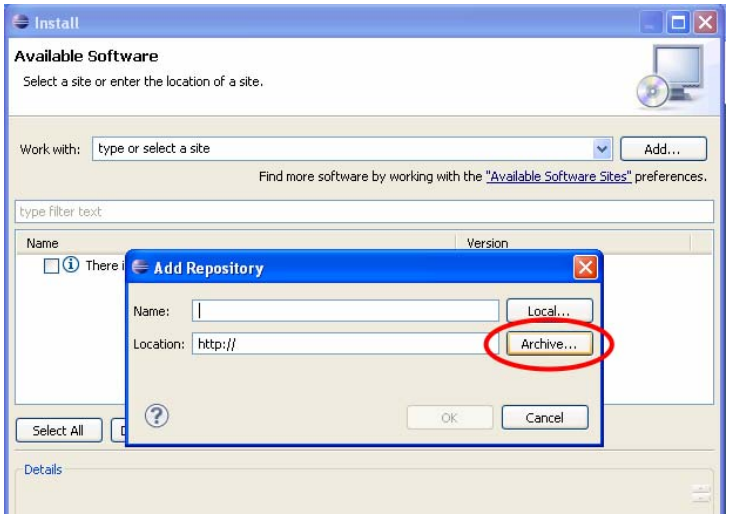
Then you have to install plug in file for compiler.

First, select Help -> Install New Software.

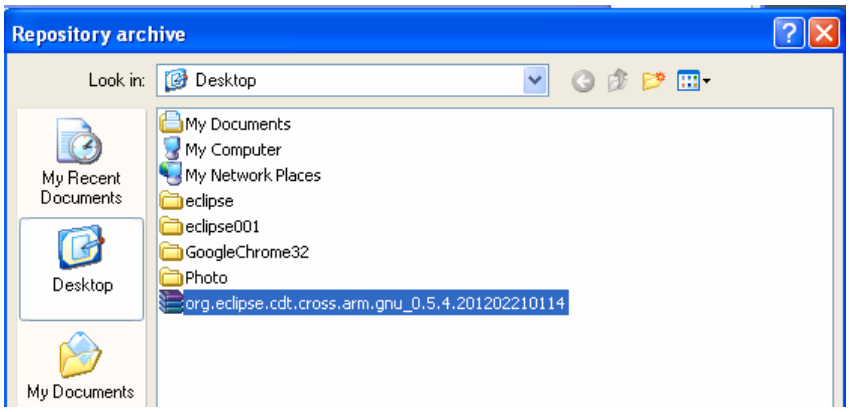
Second, select “Add”



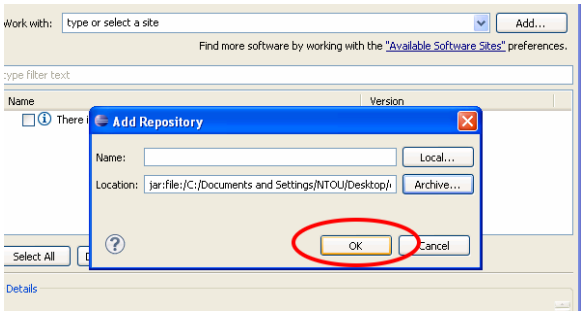
Third, select “Archive”

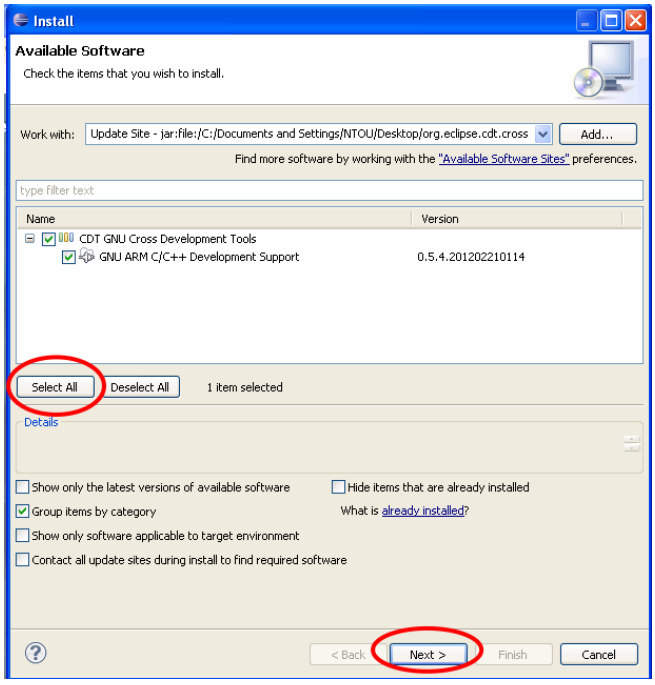


Fourth, select the plug in file.



Fifth, select OK, and all plug in and Next.





Now, you finish the compiler tools installation.

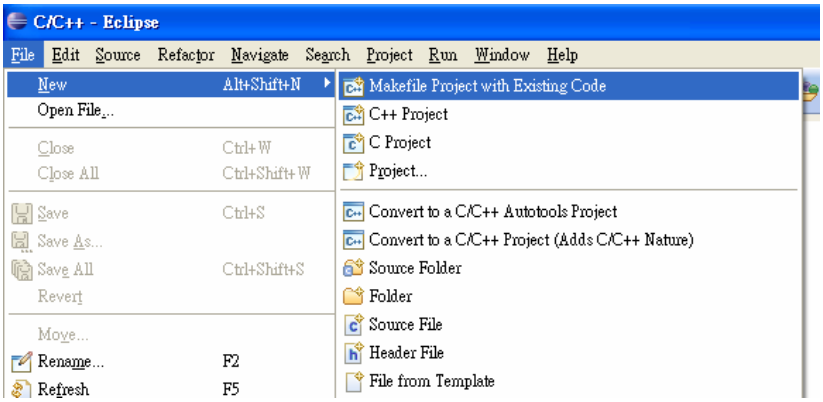
2. Edit Program:

- Developers may use the ***Application.c*** file under *Source* folder in the *PI1-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*"

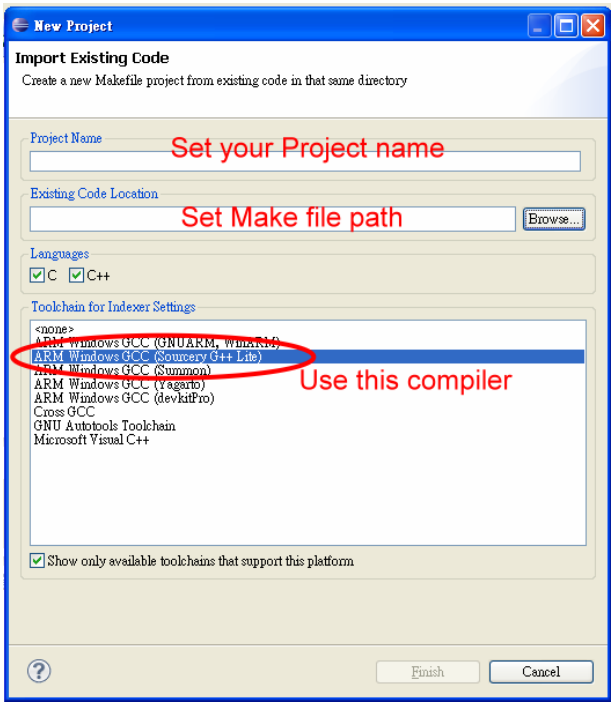
3. Compiler:

- After finish edit program, you can start to compiler.

  First:Open Eclipse -> File -> New -> Makefile Project with Existing Code

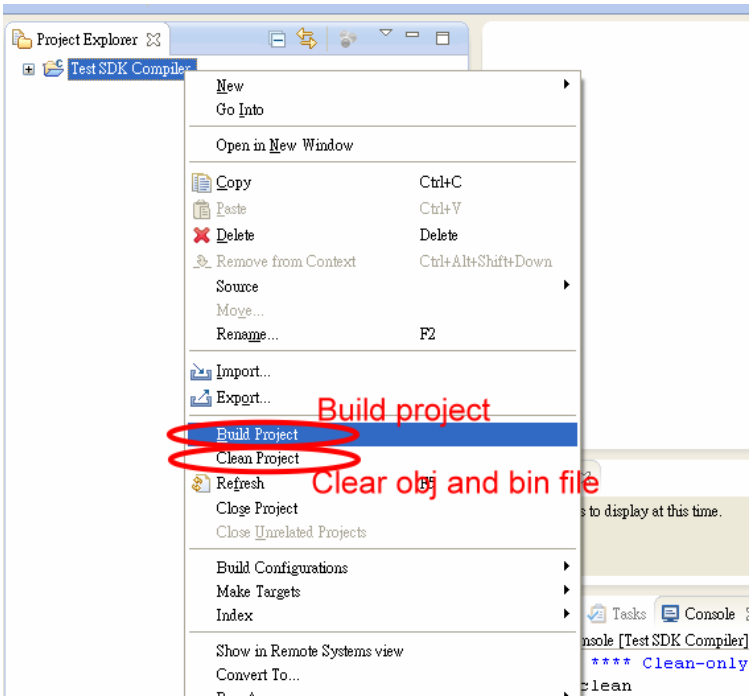


Then you will enter to this setting.



The make file is in our folder. (SDK\Terminal\build)

Select the project you created, and click right mouse buttom



After build project, you can find a file “App.bin” in SDK\Terminal\build\bin, This is our bin file.

4. Update your System:

If your system version is not match this SDK version, please refer to the **“Upgrade System”** section.

5. Development Notice:

- ***void Application_Main (void)*** is the entry program of *Application.c* instead of usual main.c.
- Maximum User Task Stack: About 300K bytes
- Maximum global area and Memory allocation(Total): About 20Mbytes.
- Maximum capacity of the Binary file (***App.bin***): About 2MBytes
- System storage:
  - Drive C – DDR memory for dynamic access.
  - Drive D – NAND flash.
  - Drive E – SD card.
- The developer can exchange files with PC using the communication tool PhoenixVoler.

6. Download APP.bin

- Please download your app.bin by PhoenixVoler, and to the terminal path

"D:\PROGRAM". After download, you can run your application.

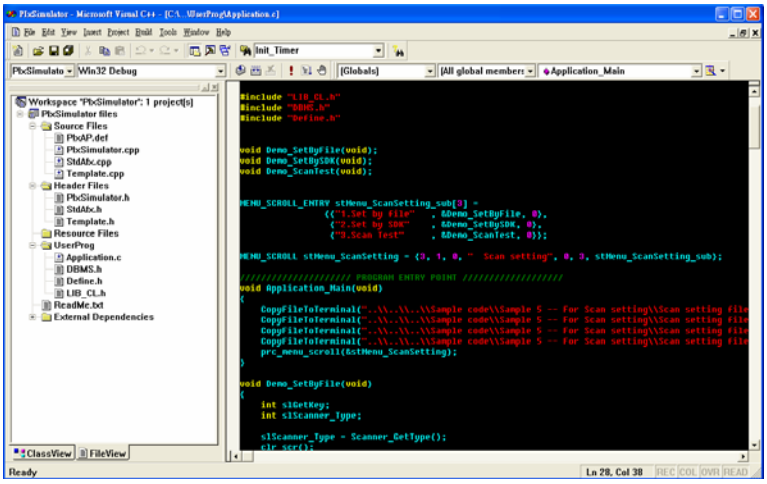
7. Simulator(Only for debug in PC site)

Purpose:

To shorten the development time and increase the program stability, a simulation tool is designed for developer to edit and debug program with ease. With this simulator, developer will know in advance whether there is any error in the program code or whether this program meets actual demands before downloading the program to the collector so that the correction and debugging can be done immediately. The simulator provides a platform, which can simulate the same hardware functionality as a real collector, for example buzzer, LED, scanner, key buttons, memory allocation and LCD display. Developer can identify whether the program meets the demands through the simulation test.

Developing Environment:

Microsoft Visual C++ 6.0 developing tool needs to be installed into the workstation. The developing environment will appear like the image on the right.



How to use:

a. Complete the developing environment setup listed above.

b. Execute \SDK\Terminal\Simulator\PIxSimulator.dsw in the directory then you can open the simulation project file. Under VC++6, execute Build\Set Active Configuration...,select PI1xSimulator - Win32 Debug, then click OK to complete the environment setting.

c. Start Simulating:

   i. Execute program(Function key:Ctrl+F5)

ii. Then a simulator will appear on the desktop

iii. Select “1.Run program” ane press “ENT Button” to run the program.

d. Debug

i. When running simulation, VC++6 will compile and link all the programs and generate a DLL file to link with the simulation file in the Execute directory. When compiling and linking, the error(s) or warning(s) will be displayed on the VC++6 windows to let user know the error messages.

ii. The developer will need to remove all the errors and warnings to ensure the syntax accuracy of the program.

iii. The logical errors of the program need to be debugged using VC++6 debugging environment. This debugging environment provides the functions of line-by-line program execution, variable listing and message hints.

e. The project file links the source file as figure.

Source file	Folder
Application.c	\SDK\Terminal\UserProg\ Application.c
DBMS.h	\SDK\Terminal\UserProg\ DBMS.h
Define.h	\SDK\Terminal\UserProg\ Define.h
LIB_CL.h	\SDK\Terminal\UserProg\ LIB_CL.h

.

Upgrade System

1. System Requirement:

- Software: PhoenixVoler
- Hardware: PI-1010 / PI-1030 and PC.
- Firmware: Bin file for upgrade.(in Upgrade folder)

2. Upgrade Procedure:

In Force Mode:

- When reset PI-1010 / PI-1030, please power off the terminal, and press combine key “1+3” (don’t release this combine key), then press power key to power on. After power on, you can release combine key and enter to Force mode.

- Then connect to PC and wait for communication.
- Force mode is only for firmware upgrade.
- After upgrade in force mode, all of user settings will be reset.

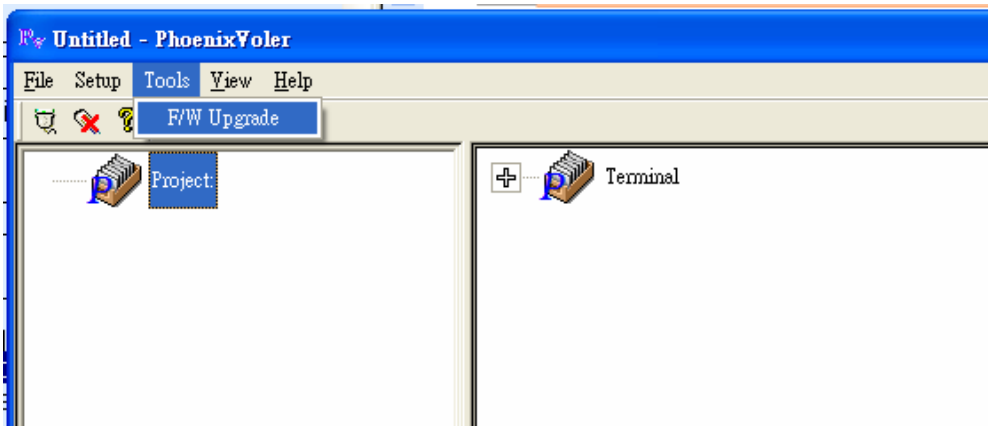
In Supervisor Menu:

- When reset PI-1010 / PI-1030, please power off the terminal, and press combine key “1+3+0” (don’t release this combine key), then press power key to power on. After power on, you can release combine key and enter to Supervisor Menu.

- And input password “00000” -> Remote Link -> Connect. Then connect to the PC and wait for communication.
- Only Supervisor Menu’s Remote Link can upgrade firmware.
- Upgrade firmware under Supervisor Menu’s Remote Link, all of user settings will not be reset.

In PhoenixVoler:

- Execute PhoenixVoler and select Tools\F/W Update



.

- Select the FW upgrade file and complete the firmware update.

3. Execute System:

- Restart PI-1010 / PI-1030.

Utility & Others

1. AID MAKER

- Select PI-1000 / PI-1010 / PI-1030 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.
- Please connect in “Remote Link -> Connect” status.

2. 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.

3. ScanSetting

This utility can make a file for all scanner settings. You can download this file and use SDK function “*ScannerSetFromFile*” to set all scanner settings.

4. ServiceID Maker

- Select PI-1000 / PI-1010 / PI-1030 in communication mode.
- Double click “ServiceIDTool.exe” and set connection.
- This application will ask the Service ID and Project name, each of these word for 4~8 characters.
- Please connect in “Remote Link -> Connect” status.

5. FileConverter

- For **DBMS** function.
- Convert Source file to Output file with specified format and field.
- You can see "FileConverter manual.doc" for how to use this utility.

6. DataMagic

- For Data magic function.(_scanf_DataMagic, DataMagic_Set, DataMagic_Run)
- You can see "PI1X DataMagic Setup Tool User Guide.doc" for how to use this utility.
- And we have C language sample code in SDK=>Sample code=>Sample 2.

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_QueryStatus	Query the scanner(CCD) 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_GetType	To get scanner type for CCD or 2D.
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.
GetBuzzerVol	Get 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.
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_file	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_file	Move file pointer of a DAT file to a new position.
open_file	Open a DAT file and get the file handle of the file for further processing.
read_file	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_file	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.
DiskE_format	Format disk E.(SD card)
DiskC_totalsize	Checking the total space in disk C.
DiskD_totalsize	Checking the total space in disk D.
DiskE_totalsize	Checking the total space in disk E.(SD card)
DiskC_usedsize	Checking the used space in disk C.
DiskD_usedsize	Checking the used space in disk D.
DiskE_usedsize	Checking the used space in disk E.(SD card)
DiskC_freesize	Checking the free space in disk C.
DiskD_freesize	Checking the free space in disk D.
DiskE_freesize	Checking the free space in disk E.(SD card)
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.
CreateDIR	Use CreateDIR can create a directory.
DeleteDIR	Use DeleteDIR can delete a directory.
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, acquiring the appointed field in including this field’s record.
SearchMultiField_GF	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.
SeekRecord	Move the index of searching to the appointed record.
GetRecordNum	Obtain the figure of all records in the file.
DeleteRecord	Delete the appointed record in the file.

DeleteLastRecord	Delete the last record in the file.
AppendRecord	Increase one record on the file end.
WriteField	Revise the data of appoint field in appointed field record.
WriteRecord	Revise the data of the appointed record.
ReadField	Read the data of appointed field in the appointed record.
ReadRecord	Read data of the appointed record.
LED	
set_led	To set the LED indicators
Keypad	
clr_kb	To clear the keyboard buffer.
dis_alpha	Disable alphabet key stroke processing.
en_alpha	Enable alphabet key stroke processing.
get_alpha_enable_state	Get the status of the alphabet key stroke processing.
set_alpha_mode_state	Set the status of the alphabet mode.
get_alpha_mode_state	Get the status of the alphabet mode.
set_keypad_BL	Set keypad and LCD backlight on/off.
get_keypad_BL	Get keypad and LCD backlight on/off status.
set_keypad_BL_Timer	Set keypad and LCD backlight timer.
get_keypad_BL_Timer	Get keypad and LCD backlight timer.
kbhit	Check keybuffer is empty or not.
_getchar	Get one key stroke from the keyboard buffer.
GetKeyClick	Get current key click status
SetKeyClick	To enable / disable the key click sound.
Def_PKey	Change progrom key 1 ~ 3(P1 ~ P3) key define.
Def_PKey_MultiInput	Change progrom key 1 ~ 3(P1 ~ P3) key define.
FNKey_Reset	To reset all of FN-Key setting.
FNKey_GetState	To check the FN-Key setting that is custom or default.
FNKey_SetUserDef	To set a custom setting for FN-Key.
_scanf	Use _scanf to read character strings from the standard input file and covert the strings to values of C variables according to specified formats.
_scanf_DefaultStr	Use _scanf_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” function.
_scanf_ctrl_ScannerSleep	Set scanner sleep on/off when use “_scanf” function.
_scanf_ctrl_Vibrate	Set vibrate on/off when use “_scanf” function and scanner

	status on.
_scanf_ctrl_ScanWithENT	Set ENT auto press on/off when use "_scanf" function and scanner status on.
_scanf_ctrl_AlphaKey	Set Alpha key function on/off when use "_scanf" function.
_scanf_ctrl_AlphaKey_Mode	Set alpha mode when use "_scanf" function.
_scanf_ctrl_password	Set display for general or user define when use "_scanf" function.
_scanf_ctrl_KeypadLock	Set keypad lock on/off when use "_scanf" function.
_scanf_DataMagic	Use _scanf_ DataMagic to read character strings from the standard input file and covert the strings to values of C variables according to specified formats. After these actions, it will convert strings according to "Data Magic" file.
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.
reverse_rect	To reserve the 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 and keypad backlight level.
lcd_backlit_SetTimer	Set LCD and keypad backlight timer.
lcd_backlit_Getlv	Get LCD and keypad backlight level.
lcd_backlit_GetTimer	Get LCD and keypad backlight timer.
lcd_contrast_Getlv	Get LCD contrast level.
lcd_contrast_Setlv	Set LCD contrast level.
_printf	Use _printf to write character strings and volues of C variables, formatted in a specified manner, to display screen.
_putchar	Display a character in color black on the LCD display.
_puts	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	
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.
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.
USB_Flush	USB reveive and write buffer reset.
Remote	
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. When use this function, file transfer is disable.
RemotePort_SelectIF	Select RemoteLink interface.
RemotePort_GetSelectIF	Get RemoteLink interface.
RemotePort_SetCOM	Set COM port baudrate for RemoteLink.

RemotePort_GetCOM	Get COM port baudrate for RemoteLink.
RemotePort_SetBT	Set Bluetooth function for RemoteLink.
RemotePort_GetBT	Get Bluetooth function for RemoteLink.
RemotePort_SetWIFI	Set WIFI function for RemoteLink.
RemotePort_GetWIFI	Get WIFI function for RemoteLink.
RemotePort_SendMsg	Send message data or Scanner(HID) to PC.
RemotePort_SendBarcode	Send data(Barcode or other input) to PC.
RemotePort_SendBarcode_Status	Return data send temp buffer status.
RemotePort_SendBarcode_Clr	Clear all data send.
RemotePort_ReadBarcode	Read data from read temp buffer..
RemotePort_ReadBarcode_Status	Return data read temp buffer status.
RemotePort_ReadBarcode_Clr	Clear all data read.
RemotePort_SendMsg_1	Send message to PC.
RemotePort_SendMsg_Status_1	Return message send temp buffer status.
RemotePort_ReadMsg_1	Read message from PC send.
RemotePort_FileTran_Status	When using RemoteLink_RealTime, to get the transfer status.
RemotePort_FileTran_Success	When using RemoteLink_RealTime, to get the upload/download quantity of files.
RemotePort_FileTran_StatusClr	Clear all the file transfer information.
RemotePort_FileTran_Info	When using RemoteLink_RealTime, to get the file transfer information.
LinkingPort	
LinkingPort_Open	Start a LinkingPort.
LinkingPort_Close	Stop a LinkingPort.
LinkingPort_SetSelectIF	Set LinkingPort interface select setting.
LinkingPort_GetSelectIF	Get LinkingPort interface select setting.
LinkingPort_SetCOM	Set LinkingPort COM baudrate setting.
LinkingPort_GetCOM	Get LinkingPort COM baudrate setting.
LinkingPort_SetBT	Set LinkingPort Bluetooth function setting.
LinkingPort_GetBT	Get LinkingPort Bluetooth function setting.
LinkingPort_SetWIFI	Set LinkingPort WIFI function setting.
LinkingPort_GetWIFI	Get LinkingPort WIFI function setting.
LinkingPort_Write	Write data to LinkingPort.
LinkingPort_Read	Read data from LinkingPort.

LinkingPort_Write_n	Write data to LinkingPort in BT/WIFI independent task.
LinkingPort_Read_n	Read data from LinkingPort in BT/WIFI independent task.
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.
BIOS_Setting_SaveToKernel	Save BIOS settings to kernel.
Check_AID	Check the agency ID correct or not.
GetKernelVer	To get the Kernel version.
SetDCIn_AlwaysOn	To set power auto off or not status when DC in.
GetDCIn_AlwaysOn	To get power auto off or not status when DC in.
GetBatt_Level	To get the Battery level.
GetWIFI_RSSI	To get the wifi RSSI value.
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. You can use “FreeHeapSize” to check free size for this function
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	Create a menu-driven interface.
prc_menu_scroll	Create a menu-driven interface with scroll function.
prc_menu_Set_SelectWithEnt	Set function “prc_menu” and “prc_menu_scroll” ENT key

	status when use number key to select menu.
prc_menu_Get_SelectWithEnt	Get function “prc_menu” and “prc_menu_scroll” ENT key status when use number key to select menu.
prc_menu_GetMenuSelect	After use function “prc_menu” and “prc_menu_scroll”, it can get what option is selected in these functions.
DataMagic_Set	Set a Data Magic file for function “_scanf_DataMagic” or “DataMagic_Run” to use.
DataMagic_Run	Convert a string by Data Magic file setting.
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.
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.
RF	
RFHost_Open	Start RF module.
RFHost_Close	Stop RF module.
RFHost_CallTagID	Call the tag.
RFHost_GetVersion	Get RF module firmware version.

Reader

InitScanner1

Purpose： Initialize respective scanner port.

Syntax： void InitScanner1(void);

Example call： InitScanner1( );

Includes： #include “LIB_CL.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 “LIB_CL.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”, the code ID will be saved in “CodeID”, and the code type will be saved in “CodeType”.
And we have the other buffer for save all barcode information in “FullCodeBuf”, the format of “FullCodeBuf” as follows:

Code name	Preamble	ID*	Code Length	Barcode data	ID*	Postamble	Terminator

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 “LIB_CL.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 “LIB_CL.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(“Scan key pressed!”);
Includes：	#include “LIB_CL.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 (“Scan module reset OK!”);
Includes：	#include “LIB_CL.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 “LIB_CL.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 “LIB_CL.h ”
Description：	This function can ending scanner setting procedure.
Returns：	None

SCAN_SendCommand

Purpose：	Send scanner(CCD/2D) 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 (“Setup complete!”);

Includes：	#include “LIB_CL.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_QueryStatus

Purpose：	Query the scanner(CCD/2D) current setting.
Syntax：	BOOL SCAN_QueryStatus(int Command1,int Command2,char *pReturn);

Example call：	char ssReturn = 0;
	if(SCAN_QueryStatus (6,7, &ssReturn))
	_printf (“Query OK!”);

Includes：	#include “LIB_CL.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.

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(“Setting OK!”);
Includes：	#include “LIB_CL.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 (“Query module version OK!”);
Includes：	#include “LIB_CL.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 “LIB_CL.h ”
Description：	This function can simulator the scan key status for pressed or released.
Returns：	None

Scanner_GetType

Purpose：	To get scanner type for CCD or 2D.
Syntax：	int Scanner_GetType(void);
Example call：	int i; i = Scanner_GetType( );
Includes：	#include “LIB_CL.h ”

Description： This function can get scanner type for user to check..

Returns： 0:CCD

2:2D

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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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. For example, an array “**int beep[4] = {30,10,0,0};**”, first “30” is for beep frequency, second “10” is for beep time (10 * 0.01s), and third “0”, fourth “0” is for beep end. **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 “LIB_CL.h ”

Description： The SetBuzzerVol function can set the buzzer volume.

slVol	Buzzer vloume
0	close
1	Low
2	Medium
3	High

Returns： None.

GetBuzzerVol

Purpose： Get the buzzer volume.

Syntax： int GetBuzzerVol(void);

Example call： Int i;

i = GetBuzzerVol();

Includes： #include “LIB_CL.h ”

Description： The GetBuzzerVol function can get the buzzer volume.

Returns：

Returns	Buzzer vloume
0	close
1	Low
2	Medium
3	High

Calender

DayOfWeek

Purpose： Get the day of the week information.

Syntax： int DayOfWeek(void);

Example call： day=DayOfWeek( );

Includes： #include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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.

File Manipulation

Unsupport file name : If these symbol “"*+,:;<=>\?[]|” in filename, we cannot support these file for open or transfer.

__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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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_file

Purpose：	Close a DAT file.
Syntax：	int close(int fd);
Example call：	If (close(fd)) _puts(“file closed!\n”);
Includes：	#include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 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 “LIB_CL.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 “LIB_CL.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_file

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 “LIB_CL.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.
8:File not opened
9:Illegal offset value.
10:Illegal origin value.
15:New position is beyond end-of-file.

open_file

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 “LIB_CL.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_file

Purpose：	Read a specified number of bytes from a DAT file.
Syntax：	int read_file(int fd, char *buffer, unsigned count);
Example call：	if ((bytes_read = read_file(fd,buffer,50)) = = -1) _puts(“read error!”);
Includes：	#include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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. 6:File is using.

_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 “LIB_CL.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.

tell

Purpose：	Get file pointer position of a DAT file.
Syntax：	long tell(int fd);
Example call：	current_position = tell(fd);
Includes：	#include “LIB_CL.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_file

Purpose：	Write a specified number of bytes to a DAT file.
Syntax：	int write_file(int fd, char *buffer, unsigned count);
Example call：	Write_file(fd, data_buffer,100);
Includes：	#include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.h ”
Description：	The DiskD_format function formats disk D.
Returns：	0 : Format false。 1 : Format OK。
fErrorCode：	None

DiskE_format

Purpose：	Format disk E.(SD card)
Syntax：	int DiskE_format (void);
Example call：	DiskE_format ( );
Includes：	#include “LIB_CL.h ”
Description：	The DiskE_format function formats disk E.
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 “LIB_CL.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.(KBytes)
fErrorCode：	None

DiskD_totalsize

Purpose：	Checking the total space in disk D.
Syntax：	unsigned int DiskD_ totalsize (void);
Example call：	DiskD_totalsize ( );
Includes：	#include “LIB_CL.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.(KBytes)
fErrorCode：	None

DiskE_totalsize

Purpose：	Checking the total space in disk E.(SD card)
Syntax：	unsigned int DiskE_ totalsize (void);
Example call：	DiskE _totalsize ( );
Includes：	#include “LIB_CL.h ”
Description：	The DiskE _totalsize function returns the total space in disk E.
Returns：	0xffffffff : Disk E unformatted.
	Others : The total space in disk E.(KBytes)
fErrorCode：	None

DiskC_usedsize

Purpose：	Checking the used space in disk C.
Syntax：	unsigned int DiskC_usedsize (void);
Example call：	DiskC_usedsize ( );
Includes：	#include “LIB_CL.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.(KBytes)
fErrorCode：	None

DiskD_usedsize

Purpose：	Checking the used space in disk D.
Syntax：	unsigned int DiskD_usedsize (void);
Example call：	DiskD_usedsize ( );

Includes：	#include “LIB_CL.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.(KBytes)
fErrorCode：	None

DiskE_usedsize

Purpose：	Checking the used space in disk E.(SD card)
Syntax：	unsigned int DiskE_usedsize (void);
Example call：	DiskE _usedsize ( );
Includes：	#include “LIB_CL.h ”
Description：	The DiskE _usedsize function returns the used space in disk E.
Returns：	0xffffffff : Disk E unformatted.
	Others : The used space in disk E.(KBytes)
fErrorCode：	None

DicskC_freesize

Purpose：	Checking the free space in disk C.
Syntax：	unsigned int DiskC_freesize (void);
Example call：	DiskC_freesize( );
Includes：	#include “LIB_CL.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.(KBytes)
fErrorCode：	None

DicskD_freesize

Purpose：	Checking the free space in disk D.
Syntax：	unsigned int DiskD_freesize (void);
Example call：	DiskD_freesize( );
Includes：	#include “LIB_CL.h ”
Description：	The DicskD_freesize function returns the free space in disk D.
Returns：	0xffffffff : Disk D unformatted.
	Others : The free space in disk D.(KBytes)
fErrorCode：	None

DicskE_freesize

Purpose：	Checking the free space in disk E.(SD card)
Syntax：	unsigned int DiskE_freesize (void);
Example call：	DiskE _freesize( );
Includes：	#include “LIB_CL.h ”
Description：	The DiskE _freesize function returns the free space in disk E.

Returns：	0xffffffff : Disk E unformatted.
	Others : The free space in disk E.(KBytes)
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 “LIB_CL.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 “LIB_CL.h ”
Description：	The getFileNum function can get the file quantity in designate path.
Returns：	-1 : path error.
	-2 : disk unformat.
	upward 0 : file 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 “LIB_CL.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 is NULL, it will return buffer size. When pssBuffer is not 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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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.

CreateDIR

Purpose：	Use CreateDIR can create a directory.
Syntax：	int CreateDIR(const char *pssDir_Name);
Example call：	CreateDIR(“C:\\Data\\New_DIR”);
Includes：	#include “LIB_CL.h”
Description：	The CreateDIR function can create a new folder. You can create folder in any disk or folder.
Returns：	0 : Create fail. 1 : Create success.
fErrorCode：	None

DeleteDIR

Purpose：	Use DeleteDIR can delete a directory.
Syntax：	int DeleteDIR(const char *pssDir_Name);
Example call：	DeleteDIR (“C:\\Data\\New_DIR”);
Includes：	#include “LIB_CL.h”
Description：	The DeleteDIR function can delete the folder exist in any disk. Before delete folder, you have to close the using file to avoid something error.
Returns：	0 : Delete fail. 1 : Delete success.
fErrorCode：	None

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：
```
_DBMS fsearch;
_TFILE *filepoint;
unsigned char field_size[5]={6,5,4,5,6};
filepoint = _fopen(“c:\\data\\data.txt”,”r+”);
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	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.
int record_type	When record_type is 0, search for regular length. It has no separate symbols among field and field. Only support regular length search.
int record_length	This argument is each record’s length.
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：
```
_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”,”r+”);
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. When initial, we will make a index file in C disk, so it has to take a few time.
2. 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.
3. 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	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.
	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. Only support regular length search.
	int record_length	When record_type is 0, need to insert this value, not including the symbol of line feed.
	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 “LIB_CL.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

Each special key value:

KEY_UP	0x05	KEY_DOWN	0x06
KEY_LEFT	0x07	KEY_RIGHT	0x0b
KEY_ESC	0x1b	KEY_BS	0x08
KEY_CR	0x0d	KEY_P1	0x15
KEY_P2	0x16	KEY_P3	0x17

clr_kb

Purpose： To clear the keyboard buffer.

Syntax： void clr_kb(void);

Example call： clr_kb( );

Includes： #include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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(0);
Includes：	#include “LIB_CL.h ”
Description：	This function can set alphabet mode. status = 0 : Numeric input. status = 1 : Lowercase input. status = 2 : Uppercase input.
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 “LIB_CL.h ”
Description：	This function can get alphabet mode status.
Returns：	0 : Numeric input. 1 : Lowercase input. 2 : Uppercase input.

set_keypad_BL

Purpose：	Set keypad and LCD backlight on/off.
Syntax：	void set_keypad_BL(BOOL bStatus);
Example call：	set_keypad_BL(1);//Key backlight on.
Includes：	#include “LIB_CL.h ”
Description：	This function can set keypad backlight on or off. bStatus = 0，off backlight. bStatus = 1，on keypad backlight, LCD backlight is Level 1. bStatus = 2，on keypad backlight, LCD backlight is Level 2. bStatus = 3，on keypad backlight, LCD backlight is Level 3.
Returns：	None

get_keypad_BL

Purpose：	Get keypad and LCD backlight on/off status.
Syntax：	BOOL get_keypad_BL(void);
Example call：	if (get_keypad_BL())

_printf (“Key Backlight on”);

Includes： #include “LIB_CL.h ”

Description： This function can get keypad backlight status.

Returns： 0，off backlight.

1，on keypad backlight, LCD backlight is Level 1.

2，on keypad backlight, LCD backlight is Level 2.

3，on keypad backlight, LCD backlight is Level 3.

set_keypad_BL_Timer

Purpose： Set keypad and LCD backlight timer.

Syntax： void set_keypad_BL_Timer(int slTimer);

Example call： set_keypad_BL_Timer(1);//Set keypad and LCD backlight timer for 1 sec.

Includes： #include “LIB_CL.h ”

Description： This function can set keypad and LCD backlight timer.

Returns： None

get_keypad_BL_Timer

Purpose： Get keypad and LCD backlight timer.

Syntax： int get_keypad_BL_Timer(void);

Example call： int slkeypadimer;

slkeypadimer = get_keypad_BL_Timer();

Includes： #include “LIB_CL.h ”

Description： This function can get keypad and LCD 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 “LIB_CL.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： int _getchar(void);

Example call： c=_getchar ( );

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

else printf(“No key pressed”);

Includes： #include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 ~ 3(P1 ~ P3) 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 “LIB_CL.h ”
Description：	This function can change the program key (P1 ~ P3) to other key define. For example, change P1 key to ENT key or ESC key.
Returns：	none

Def_PKey_MultiInput

Purpose：	Change progrom key 1 ~ 3(P1 ~ P3) key define.

Syntax：	void Def_PKey_MultiInput(int nPKey, char* ssDef, int slKeys);
Example call：	char assKeyDefine[4] = { KEY_UP, KEY_DOWN, KEY_LEFT, 0} Def_PKey_MultiInput(KEY_P1, assKeyDefine, 3);
Includes：	#include “LIB_CL.h ”
Description：	This function can change the program key (P1 ~ P3) to other multi input key define. For example, change P1 key to input up, down and left key.
Returns：	none

FNKey_Reset

Purpose：	To reset all of FN-Key setting.
Syntax：	void FNKey_Reset(void);
Example call：	FNKey_Reset();
Includes：	#include “LIB_CL.h ”
Description：	Reset all FN-key to initial status.
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 “LIB_CL.h ”
Description：	You can check the FN-Key function that is default setting or custom setting. smKeyNum: 0 → FN + 0, 1 → FN + 1, 2 → FN +3 ~~~~~ 9 → FN + 9
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 F0 UserDefine OK!”);

```
    }
    if (FNKey_SetUserDef(0, NULL))
    {
        _printf("Set F0 Default OK!");
    }
}
```

Includes： #include “LIB_CL.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 FN+ 0~9, if you want to set default, please set pslFunction = NULL.

smKeyNum: 0 → FN + 0, 1 → FN + 1, 2 → FN +3 ~~~~~ 9 → FN + 9

Returns： 1 : Set success ◦

0 : Set false ◦

_scanf

Purpose： Use _scanf 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 (const char *format, ...);

Example call： char assBuffer[10];

_scanf (“%s”, assBuffer);

Includes： #include “LIB_CL.h ”

Description： The _scanf 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.

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 to ignore whitespace characters from keyboard. Other nonwhitespace characters, excluding the % sign, cause _scanf 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, there must be a format specification embedded in the *format*. For the complete format specification accepted by the _scanf 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 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_DefaultStr

Purpose： Use _scanf_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_DefaultStr(char* assDefaultStr, const char *format, ...);

Example call： char assBuffer[10] = “ABC”;
_scanf_DefaultStr(assBuffer , “%s”, assBuffer);

Includes： #include “LIB_CL.h ”

Description： The _scanf_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_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_DefaultStr to ignore whitespace characters from keyboard. Other nonwhitespace characters, excluding the % sign, cause _scanf_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_DefaultStr, there must be a format specification embedded in the *format*. For the complete format specification accepted by the _scanf_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_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” function.

Syntax： void _scanf_ctrl_ScannerStatus(BOOL bStatus);

Example call： _scanf_ctrl_ScannerStatus(TRUE);

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function.

Syntax： void _scanf_ctrl_ScannerSleep(BOOL bStatus);

Example call： _scanf_ctrl_ScannerSleep(TRUE);

Includes： #include “LIB_CL.h ”

Description： After use “_scanf” function, this function can set scanner status in sleep mode or stop.

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

TRUE : Scanner sleep.

FALSE : Scanner not sleep.

Returns： none

_scanf_ctrl_Vibrate

Purpose： Set vibrate on/off when use “_scanf” function and scanner status on.

Syntax： void _scanf_ctrl_Vibrate(BOOL bEnable);

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

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function and scanner status on.

Syntax： void _scanf_ctrl_ScanWithENT(BOOL bScanEnt);

Example call： _ scanf_ctrl_ScanWithENT (TRUE);

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function.

Syntax： void _scanf_ctrl_AlphaKey (int status);

Example call： _scanf_ctrl_AlphaKey (TRUE);

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function.

Syntax： void _scanf_ctrl_AlphaKey_Mode(int slAlphaMode);

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

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function.

Syntax： void _scanf_ctrl_password (char ssPassWord);

Example call： _scanf_ctrl_password (‘*’);

Includes： #include “LIB_CL.h ”

Description： When use “_scanf” 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” function.
Syntax：	void _scanf_ctrl_KeypadLock(BOOL bLock);
Example call：	_scanf_ctrl_KeypadLock(FALSE);
Includes：	#include “LIB_CL.h ”
Description：	When use “_scanf” 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

_scanf_DataMagic

Purpose：	Use _scanf_ DataMagic to read character strings from the standard input file and covert the strings to values of C variables according to specified formats. After these actions, it will convert strings according to “Data Magic” file..
Syntax：	int _scanf_DataMagic(char *pssProfile, const char *format, ...);
Example call：	char assBuffer[100]; DataMagic_Set(“C:\\Data\\Sample.dmf”); _scanf_ctrl_ScanWithENT(TRUE); _scanf_DataMagic (“DM_ProFile”, “%s”, assBuffer);
Includes：	#include “LIB_CL.h ”
Description：	The _scanf_ DataMagic function only accepts a string input, from scanner or keyboard. So, this function only support “string” convert. If you use “value”,there will be something wrong. Before use this function, please use “DataMagic_Set” function for initial Data Magic file, and set “_scanf_ctrl_ScanWithENT” for TRUE.
Returns：	0:No input. 1:Success 0xff:Esc exit. -10:Data Magic file is not initial. -11:No profile in this Data Magic file. -12:No match rule to convert.

LCD

The following functions clr_eol, clr_rect, clr_scr, fill_rect, fill_rect, Get_Cursor, Set_Cursor, gotoxy, wherex, wherexy, wherey, _printf, _putchar, _puts 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 "LIB_CL.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 "LIB_CL.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 "LIB_CL.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 “LIB_CL.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

reverse_rect

Purpose： To reserve the rectangular area on the LCD display.

Syntax： void reverse_rect(int left, int top, int width, int height);

Example call： reverse_rect (10,5,30,10);

Includes： #include “LIB_CL.h ”

Description： The reverse_rect function reverse a 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	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

Get_Cursor

Purpose： Get current cursor status.

Syntax： int Get_Cursor(void);

Example call： if (Get_Cursor( ) ==0) _puts(“Cursor Off”);

Includes： #include “LIB_CL.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 “LIB_CL.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 third line of the fourth row */

Includes： #include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.h ”
Description：	The wherey function determines the current y-coordinate location of the cursor.
Returns：	none

lcd_backlit_Setlv

Purpose：	Set LCD and keypad backlight level.
Syntax：	void lcd_backlit_Setlv(int level);
Example call：	lcd_backlit_ Setlv (1);/*Set LCD and keypad backlight level 1*/
Includes：	#include “LIB_CL.h ”
Description：	The lcd_backlit_Setlv sets LCD and keypad 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 3 levels to set. 0，off backlight. 1，on keypad backlight, LCD backlight is Level 1. 2，on keypad backlight, LCD backlight is Level 2. 3，on keypad backlight, LCD backlight is Level 3.
Returns：	None

lcd_backlit_SetTimer

Purpose：	Set LCD and keypad backlight timer.
Syntax：	void lcd_backlit_SetTmer(int timer);
Example call：	lcd_backlit_SetTmer (10);/*Set LCD and keypad backlight on timer for 10 sec.*/
Includes：	#include “LIB_CL.h ”
Description：	The lcd_backlit_SetTimer sets LCD and keypad 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 and keypad backlight level.
Syntax：	int lcd_backlit_Getlv(void);
Example call：	lcd_backlit_Getlv();/*Get LCD backlight level.*/
Includes：	#include “LIB_CL.h ”
Description：	The lcd_backlit_Getlv gets LCD backlight level.
Returns：	0，off backlight. 1，on keypad backlight, LCD backlight is Level 1. 2，on keypad backlight, LCD backlight is Level 2. 3，on keypad backlight, LCD backlight is Level 3.

lcd_backlit_GetTimer

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

lcd_contrast_Getlv

Purpose：	Get LCD contrast level.
Syntax：	int lcd_contrast_Getlv(void);
Example call：	lcd_contrast_Getlv ( );//Get LCD contrast level.
Includes：	#include “LIB_CL.h ”
Description：	The lcd_contrast_Getlv gets LCD contrast level.
Returns：	LCD contrast level for 1~10.

lcd_contrast_Setlv

Purpose：	Set LCD contrast level.
Syntax：	void lcd_contrast_Setlv(int level);
Example call：	lcd_contrast_Setlv(5);//Set LCD contrast level 5.
Includes：	#include “LIB_CL.h ”
Description：	The lcd_contrast_Setlv gets LCD contrast level. Level for 1~10.
Returns：	None.

_printf

Purpose：	Use _printf to write character strings and volues of C variables, formatted in a specified manner, to display screen.
Syntax：	int _printf(char *format, ...);
Example call：	_printf (“The product of %d and %d is %d\n”, x, y, x*y);

Includes：	#include “LIB_CL.h ”
Description：	The _printf 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 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 “LIB_CL.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：	1

_puts

Purpose：	Display a string in color black on the LCD display.
Syntax：	char _puts (char* string)
Example call：	_puts(“Hello World”);
Includes：	#include “LIB_CL.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.

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 “LIB_CL.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

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 “LIB_CL.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 “LIB_CL.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.

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_MONO, 0, 10, 10, 2, 3);

Includes : #include "LIB_CL.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_MONO – for white and black dots. TYPE_MONOREVERSE - for white and black dots, but reversed.
ulBGData	Please set 0.
slColumn	TextBlock column number.
slRow	TextBlock row number.
slXPos	TextBlock left-top X position in pixel(0~159).
slYPos	TextBlock left-top Y position in pixel. Status Bar enable:0~143. Status Bar disable:0~159.

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 “LIB_CL.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 “LIB_CL.h ”

Description： When set slngle 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 “LIB_CL.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 Please set 0.

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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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.

SwitchTextBlock

Purpose：	Switch TextBlock.
Syntax：	BOOL SwitchTextBlock(S32 slBlockNo);
Example call：	SwitchTextBlock(1);
Includes：	#include “LIB_CL.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 “LIB_CL.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：	int close_com(int port);
Example call：	close_com(1);
Includes：	#include “LIB_CL.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：	1:Success 0:Fail

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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 "LIB_CL.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 "LIB_CL.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 "LIB_CL.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 "LIB_CL.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 = USB_Read(ausBuf, 10);
	_printf("Read %d characters => %s", k, ausBuf);

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

USB_Flush

Purpose：	USB reveive and write buffer reset.
Syntax：	void USBHID_Flush (void);
Example call：	USBHID_Flush();
Includes：	#include “LIB_CL.h ”
Description：	The function can flush the USB port receive / write buffer.
Returns：	None

Remote

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 “LIB_CL.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.
When use this function, file transfer is disable.

Syntax： void RemoteLink_RealTime(BOOL bStatus);

Example call： RemoteLink_RealTime(TRUE);

Includes： #include “LIB_CL.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.

When use this function, to avoid modify using file, file transfer protocol is disable.

Returns： None

RemotePort_SelectIF

Purpose： Select RemoteLink interface.

Syntax： S32 RemotePort_SelectIF(unsigned int ulInterface);

Example call： RemotePort_SelectIF(1);

Includes： #include “LIB_CL.h ”

Description： This function can set the RemoteLink port for COM, USB, Bluetooth or WIFI.
Before use this function, you need to disable RemoteLink function.
The ‘ulInterface’ setting as follows:

ulInterface	Port select
1	RS-232
2	USB

3	Bluetooth
4	WIFI

Returns： -1 : OK.

-2 : RemoteLink is using.

-4 : Parameter error.

RemotePort_GetSelectIF

Purpose： Get RemoteLink interface.

Syntax： S32 RemotePort_GetSelectIF(void);

Example call： int i;

i = RemotePort_GetSelectIF();

Includes： #include “LIB_CL.h ”

Description： This function can get the kinds of RemoteLink port for COM, USB , Bluetooth or WIFI.

Returns：

returns	Port select
1	RS-232
2	USB
3	Bluetooth
4	WIFI

RemotePort_SetCOM

Purpose： Set COM port baudrate for RemoteLink.

Syntax： S32 RemotePort_SetCOM(unsigned int ulBaudrate);

Example call： RemotePort_SetCOM(0);

Includes： #include “LIB_CL.h ”

Description： This function can set RemoteLink COM baudrate.

Before use this function, you need to disable RemoteLink function.

The ‘ulBaudrate‘ setting as follows:

ulBaudrate	Baudrate(bps)
0	115200
1	57600
2	38400
3	19200
4	9600
5	4800

Returns： -1 : OK.

-2 : RemoteLink is using.

-4 : Parameter error.

RemotePort_GetCOM

Purpose： Get COM port baudrate for RemoteLink.

Syntax： S32 RemotePort_GetCOM(void);

Example call： int i;

i = RemotePort_GetCOM();

Includes： #include “LIB_CL.h ”

Description： This function can Get RemoteLink COM baudrate.

Returns： 0: 115200bps

1: 57600bps

2: 38400bps

3: 19200bps

4: 9600bps

5: 4800bps

RemotePort_SetBT

Purpose： Set Bluetooth function for RemoteLink.

Syntax： S32 RemotePort_SetBT(_BT_INFO* stBT_Set);

Example call： _BT_INFO stSetBT;

RemotePort_SetBT(&stSetBT);

Includes： #include “LIB_CL.h ”

Description： This function can set RemoteLink Bluetooth settings.

You can use “RemotePort_GetBT” first to get internal settings and modify it.

Before use this function, you need to disable RemoteLink function.

The ‘stSetBT ‘ struct setting as follows:

```
typedef struct __BT_INFO
{
    char assLocalAddress[16];     // PI-1010 / PI-1030 Bluetooth MAC address.(Cannot change.)
    char assLocalName[20];        // PI-1010 / PI-1030 Bluetooth device name
    int stInquiryTimeout;         // PI-1010 / PI-1030 Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).
    int stInquirytMaxResponse;    // PI-1010 / PI-1030 Bluetooth inquiry max response, the value from 1 to 9.
    BOOL bLocalSecurity;          // PI-1010 / PI-1030 Bluetooth security mode, set TRUE(on) or FALSE(off)
    BOOL bLocalEncryption;        // PI-1010 / PI-1030 Bluetooth encryption mode, set TRUE(on) or FALSE(off)
    char assLinkAddress[16];      //Set link device address.
```

```
        char assPinCode[20];        //Set PIN code.
} _BT_INFO;
```

Returns： -1 : OK.

-2 : RemoteLink is using.

-4 : Parameter error.

RemotePort_GetBT

Purpose： Get Bluetooth function for RemoteLink.

Syntax： S32 RemotePort_GetBT(_BT_INFO* stBT_Set);

Example call：

```
_BT_INFO stGetBT;
RemotePort_ GetBT (&stGetBT);
```

Includes： #include “LIB_CL.h ”

Description： This function can get RemoteLink Bluetooth settings.

The ‘stSetBT ‘ struct setting as follows:

```
typedef struct __BT_INFO
{
    char assLocalAddress[16];   // PI-1010 / PI-1030 Bluetooth MAC address.(Cannot change.)
    char assLocalName[20];      // PI-1010 / PI-1030 Bluetooth device name
    int stInquiryTimeout;       // PI-1010 / PI-1030 Bluetooth inquiry timeout set, the value from 1(1.28 seconds) to 48(61.44 seconds).
    int stInquirytMaxResponse;  // PI-1010 / PI-1030 Bluetooth inquiry max response, the value from 1 to 9.
    BOOL bLocalSecurity;        // PI-1010 / PI-1030 Bluetooth security mode, set TRUE(on) or FALSE(off)
    BOOL bLocalEncryption;      // PI-1010 / PI-1030 Bluetooth encryption mode, set TRUE(on) or FALSE(off)
    char assLinkAddress[16];    //Set link device address.
    char assPinCode[20];        //Set PIN code.
} _BT_INFO;
```

Returns： -1 : OK.

-4 : Parameter error.

RemotePort_SetWIFI

Purpose： Set WIFI function for RemoteLink.

Syntax： S32 RemotePort_SetWIFI(_WIFI_INFO* stWIFI_Set);

Example call：

```
_WIFI_INFO stSetWIFI;
```

RemotePort_SetWIFI(&stSetWIFI);

Includes： #include “LIB_CL.h ”

Description： This function can set RemoteLink WIFI settings.

You can use “RemotePort_GetWIFI” first to get internal settings and modify it.

Before use this function, you need to disable RemoteLink function.

The ‘stSetWIFI ‘ struct setting as follows:

```
typedef struct __WIFI_INFO
{
    int stDhcpEnable;      //TCP/IP stack in manual or DHCP modes.
    unsigned char assIpAddr[16];     //Local IP address(if DHCP off.)
    unsigned char assSubnetMask[16];     //Subnet Mask(if DHCP off.)
    unsigned char assGateway[16]; //Address of Gateway(if DHCP off.)
    char assSSID[36];     //WIFI access point (SSID name)
    int stTxPower;   //AP TxPower: 0.Low 1.Medium 2.High
    BOOL stPWRSave;   //0:Power saving disable 1:Power saving
                        Enable
    int stKeyType;   //0:disable 1:Wep 2:Wpa2
    unsigned char assConnIpAddr[16];     //Connect IP address
    int stConnRemotePort;       //Connect remote port
    unsigned char assSecurityKey[64];     //Security key
    char assMAC[12];      //MAC address of the module.(Read only.)
    char assVER[6];       //firmware version in the module.(Read only.)
    BOOL bAdHoc;       //0:Ad-hoc disable 1:Ad-hoc enable.(RemotePort
always disable & only support LinkingPort.)
}_WIFI_INFO;
```

Returns： -1 : OK.

-2 : RemoteLink is using.

-4 : Parameter error.

RemotePort_GetWIFI

Purpose： Get WIFI function for RemoteLink.

Syntax： S32 RemotePort_GetWIFI(_WIFI_INFO* stWIFI_Set);

Example call： _WIFI_INFO stGetWIFI;

RemotePort_ GetWIFI(&stGetWIFI);

Includes： #include “LIB_CL.h ”

Description： This function can get RemoteLink WIFI settings.

The ‘stSetWIFI ‘ struct setting as follows:

```
typedef struct __WIFI_INFO
```

```
{
    int stDhcpEnable;      //TCP/IP stack in manual or DHCP modes.
    unsigned char assIpAddr[16];    //Local IP address(if DHCP off.)
    unsigned char assSubnetMask[16];    //Subnet Mask(if DHCP off.)
    unsigned char assGateway[16]; //Address of Gateway(if DHCP off.)
    char assSSID[36];     //WIFI access point (SSID name)
    int stTxPower;   //AP TxPower: 0.Low 1.Medium 2.High
    BOOL stPWRSave;   //0:Power saving disable 1:Power saving
                        Enable
    int stKeyType;   //0:disable 1:Wep 2:Wpa2
    unsigned char assConnIpAddr[16];    //Connect IP address
    int stConnRemotePort;      //Connect remote port
    unsigned char assSecurityKey[64];    //Security key
    char assMAC[12];     //MAC address of the module.(Read only.)
    char assVER[6];      //firmware version in the module.(Read only.)
    BOOL bAdHoc;      //0:Ad-hoc disable 1:Ad-hoc enable.(RemotePort
always disable & only support LinkingPort.)
}_WIFI_INFO;
```

Returns： -1 : OK.

-4 : Parameter error.

RemotePort_SendMsg

Purpose： Send message data or Scanner(HID) to PC.

Syntax： S32 RemotePort_SendMsg(char *pssMsg, char ssType, char* pssTargetEID);

Example call： Char sendData[104];

RemotePort_SendMsg(sendData, 1, "0015");

Includes： #include “LIB_CL.h ”

Description： This function can send message Scanner(HID) to PC.

The ‘pssMsg’ is for message of Scanner(HID) data.

The ‘ssType‘ setting as follows:

ssType	description
0	Barcode type
1	Text message

When ‘ssType’ is 0, pssTargetEID can set any value.

When ‘ssType’ is 1, pssTargetEID is the send target EID.

Returns： 1 : OK.

0 : FALSE.

RemotePort_SendBarcode

Purpose：	Send data(Barcode or other input) to PC.
Syntax：	int RemotePort_SendBarcode(char *pssBarcode, int slBarcodeSize, char* pssTargetEID);
Example call：	RemoteLink_RealTime(TRUE); RemotePort_SendBarcode(“01234567”, 8, “0123”)
Includes：	#include “LIB_CL.h ”
Description：	This function can send data(Barcode or other input) to PC. It has 20 gropus for temp your data.When remote connect, the data will be auto send. Before call this function, you have to call “RemoteLink_RealTime” function to enable remote connect. **char *pssBarcode:** The data you want to send, max length is 2047. **int slBarcodeSize:** Send data length. **char* pssTargetEID:** Terget EID(Equipment ID). Only 4 characters. If set “NULL”, the data will only send to PC.
Returns：	0: Remote Link is not enable. 1: Data is send to temp buffer and waiting for send. -1:Parameter error. -2:Temp buffer is full, cannot send data this time.

RemotePort_SendBarcode_Status

Purpose：	Return data send temp buffer status.
Syntax：	int RemotePort_SendBarcode_Status(void);
Example call：	int i; i = RemotePort_SendBarcode_Status();
Includes：	#include “LIB_CL.h ”
Description：	This function can return send data temp buffer(Barcode or other input) status.
Returns：	0:Data send complete. -1:Temp data send buffer is full. 1~19:How many datas in send temp buffer.

RemotePort_SendBarcode_Clr

Purpose：	Clear all data send.
Syntax：	void RemotePort_SendBarcode_Clr(void);
Example call：	RemotePort_SendBarcode_Clr();
Includes：	#include “LIB_CL.h ”
Description：	Clear all send data in send data buffer.
Returns：	None.

RemotePort_ReadBarcode

Purpose：	Read data from read temp buffer..

Syntax：	int RemotePort_ReadBarcode(char *pssSource, char *pssDateTime ,char *pssBarcode, int *pslBarcodeSize);
Example call：	char assSource[8]; char assDateTime[20]; char assBarcode[200]; int slSize; RemotePort_ReadBarcode(assSource, assDateTime, assBarcode, &slSize);
Includes：	#include “LIB_CL.h ”
Description：	This function can read data(Barcode or other input) from temp read buffer. It has 20 gropus for temp your data.When remote connect, the data will be auto read. If data buffer is full, the oldest data will be deleted. **char *pssSource:**.Data source EID(Equipment ID). **char *pssDateTime:** Reveive datetime.. **char *pssBarcode:** The data you want to read, max length is 2047. **int *pslBarcodeSize:** Return the “pssBarcode” string size.
Returns：	0:No receive data 1:Read a data. -1:Parameter error.

RemotePort_ReadBarcode_Status

Purpose：	Return data read temp buffer status.
Syntax：	int RemotePort_ReadBarcode_Status(void);
Example call：	int i; i = RemotePort_ReadBarcode_Status ();
Includes：	#include “LIB_CL.h ”
Description：	This function can return read data temp buffer status.
Returns：	0:No receive data. 1~20:Data need to be read.

RemotePort_ReadBarcode_Clr

Purpose：	Clear all data read.
Syntax：	void RemotePort_ReadBarcode_Clr(void);
Example call：	RemotePort_ReadBarcode_Clr();
Includes：	#include “LIB_CL.h ”
Description：	Clear all read data in read data buffer.
Returns：	None

RemotePort_SendMsg_1

Purpose：	Send message to PC.
Syntax：	int RemotePort_SendMsg_1(char *pssMsg , int slMsgSize, char* pssTargetEID);
Example call：	RemoteLink_RealTime(TRUE); RemotePort_SendMsg_1 (“01234567”, 8, “9000”)
Includes：	#include “LIB_CL.h ”
Description：	This function can send message to PC. It has 20 gropus for temp your data.When remote connect, the data will be auto send. Before call this function, you have to call “RemoteLink_RealTime” function to enable remote connect. **char *pssMsg:** The message you want to send, max length is 99. **int slMsgSize:** Send data length. **char* pssTargetEID:** Terget EID(Equipment ID). Only 4 characters.
Returns：	0: Remote Link is not enable. 1: Data is send to temp buffer and waiting for send. -1:Parameter error. -2:Temp buffer is full, cannot send data this time.

RemotePort_SendMsg_Status_1

Purpose：	Return message send temp buffer status.
Syntax：	int RemotePort_SendMsg_Status_1(void);
Example call：	RemotePort_SendMsg_Status_1();
Includes：	#include “LIB_CL.h ”
Description：	This function can return send message temp buffer status.
Returns：	0:Data send somplete. -1:Temp data send buffer is full. 1~19: How many datas in send temp buffer.

RemotePort_ReadMsg_1

Purpose：	Read message from PC send.
Syntax：	int RemotePort_ReadMsg_1(int *pslType, char *pssSource, char *pssDateTime ,char *pssMsg, int *pslMsgSize);
Example call：	int slType; char assSource[8]; char assDateTime[20]; char assMsg[100]; int slMsgSize; RemotePort_ReadMsg_1(&slType, assSource, assDateTime ,assMsg, &slMsgSize);
Includes：	#include “LIB_CL.h ”

Description：	This function can read message from temp read buffer. It has only 1 buffer for read. So, it returns only the latest message. **int *pslType:** Message type. 0: Priority Message 1: Normal Message **char *pssSource:**.Data source EID(Equipment ID). **char *pssDateTime:** Reveive datetime.. **char *pssMsg:** The data you want to read, max length is 99. **int *pslMsgSize:** Return the “pssBarcode” string size.
Returns：	0:No receive data 1:Read a data. -1:Parameter error.

RemotePort_FileTran_Status

Purpose：	When using RemoteLink_RealTime, to get the transfer status.
Syntax：	int RemotePort_FileTran_Status(void);
Example call：	int i; i = RemotePort_FileTran_Status();
Includes：	#include “LIB_CL.h ”
Description：	When using RemoteLink_RealTime, this function can get the file transfer status.
Returns：	0:No file in transferring. 1:File is transfering. -1:Not open the RemoteLink_RealTime.

RemotePort_FileTran_Success

Purpose：	When using RemoteLink_RealTime, to get the upload/download quantity of files.
Syntax：	int RemotePort_FileTran_Success(int slType);

Example call：	int slupload, sldownload;
	slupload = RemotePort_FileTran_Success(1);
	sldownload = RemotePort_FileTran_Success(2);

Includes：	#include “LIB_CL.h ”
Description：	When using RemoteLink_RealTime, this function can get the upload/download quantity of files. You can select to get upload or download quantity of files. **int slType:** 1: return upload quantity of files 2: return download quantity of files. **Attention:** The maximum return is 20. When this function returned 20 and you didn’t clear the information, the newest file information will over covered the oldest file information.

Returns： -1:Not open the RemoteLink_RealTime.
0~20: Quantity of files.

RemotePort_FileTran_StatusClr

Purpose： Clear all the file transfer information.

Syntax： void RemotePort_FileTran_StatusClr(void);

Example call： RemotePort_FileTran_StatusClr();

Includes： #include “LIB_CL.h ”

Description： This function can clear all the file transfer information.

Returns： None

RemotePort_FileTran_Info

Purpose： When using RemoteLink_RealTime, to get the file transfer information.

Syntax： int RemotePort_FileTran_Info(int slType, int slNumber, char* assPath, char* assFileName, char* assTime);

Example call：
```
char assPath[24];
char assFileName[20];
char assTime[24];
int i;
i = RemotePort_FileTran_Info(1, 0, assPath, assFileName, assTime);
```

Includes： #include “LIB_CL.h ”

Description： When using RemoteLink_RealTime, this function can get the file transfer information.

int slType:

1: Get upload information 2: Get download information.

int slNumber:

Which file to get. For example, if there are three files transferred success, when you want to get the first file information, the parameter will be filled in 0.

char* assPath:

To get the file path,you have to give it 24bytes to get path.(Path in PI-1X)

char* assFileName:

To get the file name, you have to give it 16bytes to get name.

char* assTime:

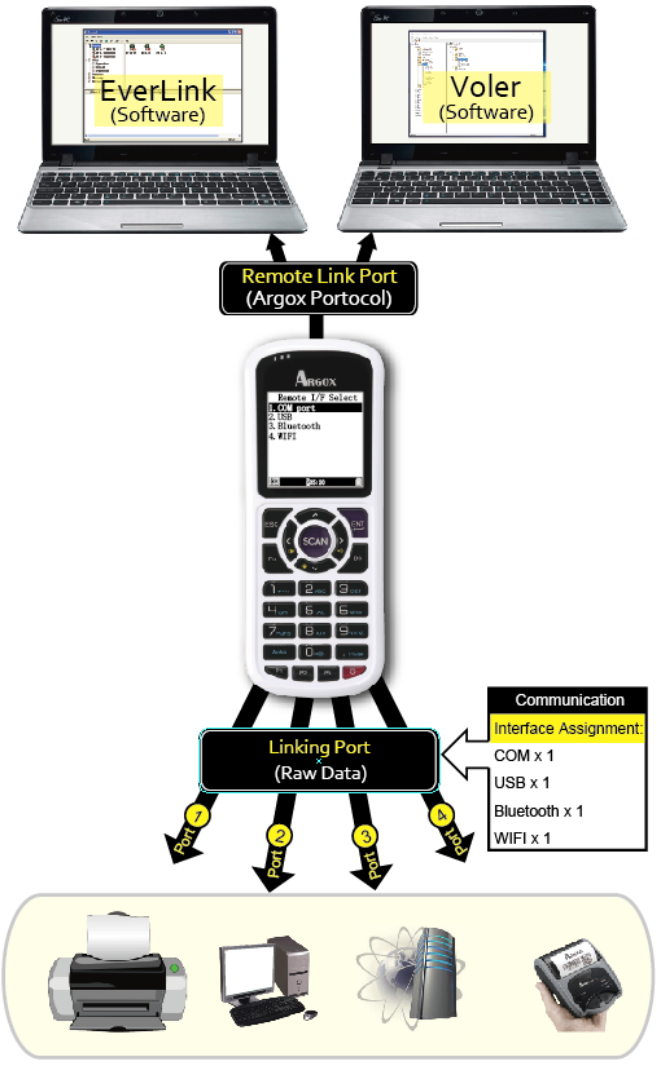
To get the file upload/download time, you have to give it 16bytes to get time.

Returns： 0:No information returned.
1:Get a file information.
-1: Not open the RemoteLink_RealTime.

-2:Parameter error.

LinkingPort

LinkingPort is a new setting. We have four LinkingPorts, port 1 to port 4. Each LinkingPort has its settings, you can set each port for COM, USB, Bluetooth , WIFI setting. After you finish these setting, you can easy to open, close, read, write these ports.



LinkingPort_Open

Purpose： Start a LinkingPort.

Syntax： S32 LinkingPort_Open(U32 ulPortNumber);

Example call： LinkingPort_Open(1);

Includes： #include “LIB_CL.h”

Description： Use this function can start a LinkingPort. Before use this function, you have to set LinkingPort’s setting.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

Returns： -1: Open LinkingPort success.

-2: Selected LinkingPort is using.

-3: Selected LinkingPort’s connect interface is using.

-4: Parameter error.

-5: Connect fail.

-7: LinkingPort is not set.

LinkingPort_Close

Purpose： Stop a LinkingPort.

Syntax： S32 LinkingPort_Close(U32 ulPortNumber);

Example call： LinkingPort_Close(1);

Includes： #include “LIB_CL.h”

Description： Use this function can stop a LinkingPort.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

Returns： -1: Close LinkingPort success.

-6: LinkingPort is not open.

-7: LinkingPort is not set.

LinkingPort_SetSelectIF

Purpose： Set LinkingPort interface select setting.

Syntax： S32 LinkingPort_SetSelectIF(U32 ulPortNumber, U32 ulInterface);

Example call： LinkingPort_SetSelectIF(1, 3);

Includes： #include “LIB_CL.h”

Description： Use this function can select a LinkingPort’s interface. Before use this function, you have to close LinkingPort.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.The parmeter “ulInterface” is LinkingPort’s interface.

0: None

1 :RS-232

2: USB

3: BlueTooth

4: WIFI

Returns： -1: Success.

-2: Selected LinkingPort is using.

-4: Parameter error.

LinkingPort_GetSelectIF

Purpose： Get LinkingPort interface select setting.

Syntax： S32 LinkingPort_GetSelectIF(U32 ulPortNumber);

Example call： int i;

i = LinkingPort_GetSelectIF(1);

Includes： #include “LIB_CL.h”

Description： Use this function can get a LinkingPort interface.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

Returns： 0: None

1: RS-232

2: USB

3: BlueTooth

4: WIFI

-4: Parameter error.

LinkingPort_SetCOM

Purpose： Set LinkingPort COM baudrate setting.

Syntax： S32 LinkingPort_SetCOM(U32 ulPortNumber, U32 ulBaudrate);

Example call： LinkingPort_SetCOM(1, 0);

Includes： #include “LIB_CL.h”

Description： Use this function can set LinkingPort’s COM baudrate. Before use this function, you have to close LinkingPort.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

The parmeter “ulBaudrate” is LinkingPort’s COM baudrate.

0: 115200 bps

1: 57600 bps

2: 38400 bps

3: 19200 bps

4: 9600 bps

5: 4800 bps

Returns： -1: Success.

-2: Selected LinkingPort is using.

-4: Parameter error.

LinkingPort_GetCOM

Purpose： Get LinkingPort COM baudrate setting.

Syntax： S32 LinkingPort_GetCOM(U32 ulPortNumber);

Example call： int i;

i = LinkingPort_GetCOM(1);

Includes： #include “LIB_CL.h”

Description： Use this function can get LinkingPort's baudrate.
The parmeter "ulPortNumber" is LinkingPort's port number. The value is form 1 to 4.

Returns： 0: 115200 bps
1: 57600 bps
2: 38400 bps
3: 19200 bps
4: 9600 bps
5: 4800 bps
-4: Parameter error.

LinkingPort_SetBT

Purpose： Set LinkingPort Bluetooth function setting.

Syntax： S32 LinkingPort_SetBT(U32 ulPortNumber, _BT_INFO* stBT_Set);

Example call：
```
_BT_INFO stBT_Set;
LinkingPort_GetBT(1, &stBT_Set);
stBT_Set. bLocalEncryption = FALSE;
LinkingPort_SetBT(1, &stBT_Set);
```

Includes： #include "LIB_CL.h"

Description： Use this function can set LinkingPort's Bluetooth settings. Before use this function, you have to close LinkingPort.
The parmeter "ulPortNumber" is LinkingPort's port number. The value is form 1 to 4.
The parmeter "stBT_Set" is LinkingPort's Bluetooth settings. About this structure, please see "RemotePort_SetBT" function.

Returns： -1: Success.
-2: Selected LinkingPort is using.
-4: Parameter error.

LinkingPort_GetBT

Purpose： Get LinkingPort Bluetooth function setting.

Syntax： S32 LinkingPort_GetBT(U32 ulPortNumber, _BT_INFO* stBT_Set);

Example call：
```
_BT_INFO stBT_Set;
LinkingPort_GetBT(1, &stBT_Set);
```

Includes： #include "LIB_CL.h"

Description： Use this function can get LinkingPort's Bluetooth settings.
The parmeter "ulPortNumber" is LinkingPort's port number. The value is form 1 to 4.
The parmeter "stBT_Set" is LinkingPort's Bluetooth settings. About this structure, please see "RemotePort_SetBT" function.

Returns： -1: Success.

-4: Parameter error.

LinkingPort_SetWIFI

Purpose： Set LinkingPort WIFI function setting.

Syntax： S32 LinkingPort_SetWIFI(U32 ulPortNumber, _WIFI_INFO* stWIFI_Set);

Example call： _WIFI_INFO stWIFI_Set;

LinkingPort_GetWIFI(1, &stWIFI_Set);

stWIFI_Set. stDhcpEnable= TRUE;

LinkingPort_SetWIFI(1, &stWIFI_Set);

Includes： #include “LIB_CL.h”

Description： Use this function can set LinkingPort’s WIFI settings. Before use this function, you have to close LinkingPort.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

The parmeter “stWIFI_Set” is LinkingPort’s WIFI settings. About this structure, please see “RemotePort_SetWIFI” function.

Returns： -1: Success.

-2: Selected LinkingPort is using.

-4: Parameter error.

LinkingPort_GetWIFI

Purpose： Get LinkingPort WIFI function setting.

Syntax： S32 LinkingPort_GetWIFI(U32 ulPortNumber, _WIFI_INFO* stWIFI_Set);

Example call： _WIFI_INFO stWIFI_Set;

LinkingPort_GetWIFI(1, &stWIFI_Set);

Includes： #include “LIB_CL.h”

Description： Use this function can get LinkingPort’s WIFI settings.

The parmeter “ulPortNumber” is LinkingPort’s port number. The value is form 1 to 4.

The parmeter “stWIFI_Set” is LinkingPort’s WIFI settings. About this structure, please see “RemotePort_SetWIFI” function.

Returns： -1: Success.

-4: Parameter error.

LinkingPort_Write

Purpose： Write data to LinkingPort.

Syntax： S32 LinkingPort_Write(U32 ulPortNumber, U8* ausBuf, U32 ulWriteSize);

Example call： LinkingPort_Write(1, “12345”, 5);

Includes： #include “LIB_CL.h”

Description： After opening LinkingPort, you can write data to that LinkingPort.

Returns： -4: Parameter error.
-5: Connect fail.
-6: LinkingPort is not open.
Over zero or zero: Bytes to be written.

LinkingPort_Write_n

Purpose： Write data to LinkingPort in BT/WIFI independent task.

Syntax： S32 LinkingPort_Write_n(U32 ulPortNumber, U8* ausBuf, U32 ulWriteSize);

Example call： LinkingPort_Write(1, “12345”, 5);

Includes： #include “LIB_CL.h”

Description： After opening LinkingPort, you can write data to that LinkingPort.

Returns： -4: Parameter error.
-5: Connect fail.
-6: LinkingPort is not open.
-8: LinkingPort is connecting.
Over zero or zero: Bytes to be written.

LinkingPort_Read

Purpose： Read data from LinkingPort.

Syntax： S32 LinkingPort_Read(U32 ulPortNumber, U8* ausBuf, U32 ulReadSize);

Example call： unsigned char ausBuff[20];
LinkingPort_Read(1, ausBuff, 20);

Includes： #include “LIB_CL.h”

Description： After opening LinkingPort, you can read data from that LinkingPort.

Returns： -4: Parameter error.
-5: Connect fail.
-6: LinkingPort is not open.
Over zero or zero: Bytes to be read.

LinkingPort_Read_n

Purpose： Read data from LinkingPort in BT/WIFI independent task.

Syntax： S32 LinkingPort_Read_n(U32 ulPortNumber, U8* ausBuf, U32 ulReadSize);

Example call： unsigned char ausBuff[20];
LinkingPort_Read(1, ausBuff, 20);

Includes： #include “LIB_CL.h”

Description： After opening LinkingPort, you can read data from that LinkingPort.

Returns： -4: Parameter error.
-5: Connect fail.
-6: LinkingPort is not open.

-8: LinkingPort is connecting.

Over zero or zero: Bytes to be read.

LinkingPort_Flush

Purpose： Flush the LinkingPort data buffer.

Syntax： S32 LinkingPort_Flush(U32 ulPortNumber);

Example call： LinkingPort_Flush(1);

Includes： #include “LIB_CL.h”

Description： After opening LinkingPort, you can flush the LinkingPort’s read and write data buffer.

Returns： -1: Open LinkingPort success.

-6: LinkingPort is not open.

-7: LinkingPort is not set.

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 10 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 "LIB_CL.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 "LIB_CL.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 "LIB_CL.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 ("Power on for restart:");
Includes：	#include "LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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(“Statusbar on”);
Includes：	#include “LIB_CL.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 PI-1X.
Syntax：	void SN_Get(char *pssSNBuffer);
Example call：	SN_Get(SNBuffer);
Includes：	#include “LIB_CL.h ”
Description：	The function cab get the SN of PI-1X. 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 “LIB_CL.h “
Description：	This function can set BIOS setting to default setting. It takes several seconds.
Returns：	None

BIOS_Setting_SaveToKernel

Purpose：	Save BIOS settings to kernel.
Syntax：	void BIOS_Setting_SaveToKernel(void);
Example call：	BIOS_Setting_SaveToKernel ();
Includes：	#include “LIB_CL.h “
Description：	This function can save BIOS setting to kernel. When you change some settings, for example, Bluetooth, LinkingPort, Backlight, Scanner settings, if you don’t use this function, when you back to main menu(User menu), these settings will lose. If you call this function before back to main menu, these settings will be saved.
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 (“AID Correct!!”);
Includes：	#include “LIB_CL.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.

GetKernelVer

Purpose：	To get the Kernel version.
Syntax：	void GetKernelVer(char * StrBuf);
Example call：	GetKernelVer(StrBuf);
Includes：	#include “LIB_CL.h ”
Description：	The function can get the Kernel version of PI-1X. The string buffer size must longer than 10 bytes.
Returns：	None

SetDCIn_AlwaysOn

Purpose：	To set power auto off or not status when DC in.
Syntax：	void SetDCIn_AlwaysOn(int state);
Example call：	SetDCIn_AlwaysOn (1);
Includes：	#include “LIB_CL.h ”
Description：	The function can set the power auto off status when DC in. If set “1”, and when DC input, PDC will not auto power off, but when DC not input, PDC will auto power off after auto power off timer count end. If set “0”, and when DC input, PDC will auto power off after auto power off timer count end.
Returns：	None

GetDCIn_AlwaysOn

Purpose：	To get power auto off or not status when DC in.
Syntax：	unsigned int GetDCIn_AlwaysOn(void);
Example call：	int i; i = GetDCIn_AlwaysOn ( );
Includes：	#include “LIB_CL.h ”
Description：	The function can get the “DC in always on” status.
Returns：	0:DC in or not, PDC will auto power off after auto off timer count end. 1:When DC in, PDC will not auto power off.

GetBatt_Level

Purpose：	To get the Battery level.
Syntax：	int GetBatt_Level(void);
Example call：	int i; i = GetBatt_Level( );
Includes：	#include “LIB_CL.h ”
Description：	The function can get the battery level status, you can check bettery level by this function
Returns：	0: Level zero.(Low battery) 1: Level 1. 2: Level 2

	3: Level 3
	4: Battery is charging.
	5: When power on, battery will not detect in 3 seconds, it will return this.

GetWIFI_RSSI

Purpose：	To get the wifi RSSI value.
Syntax：	int GetWIFI_RSSI(int slPort);
Example call：	int i; i = GetWIFI_RSSI(1);
Includes：	#include “LIB_CL.h ”
Description：	The function can get the wifi RSSI status, you can get it by this function. “slPort” is for the RemoteLink port’s or Linking port’s WIFI RSSI. slPort = 0, for Remote port. slProt = 1~4, for Linking port 1~4. When you use Linking port 1 in WIFI, you need to set the “slPort” value “1”. When you close all the port for WIFI(Remote and Linking port), you can set the “slPort” value for 0~4 to get Remote or Linking port’s WIFI RSSI.
Returns：	>0: WIFI RSSI. -1: Get error. -2: The port is opening or connecting. -3: The port is not using port(when WIFI is opened).

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 “LIB_CL.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. You can use “FreeHeapSize” to check free size for this function
Syntax：	void* Tmalloc(unsigned int num_bytes);
Example call：	buffer = (char *)Tmalloc(100*sizeof(char));
Includes：	#include “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.h ”
Description：	The UsedHeapSize function can get the used heap size.
Returns：	The free 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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.h ”
Description：	Use get_vibrator_timer function can get vibrator on timer.
Returns：	Vibrator on timer.

Other

prc_menu

Purpose： Create a menu-driven interface.

Syntax： void prc_menu(MENU *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 Menu_01 = {0,1,"1.Test Menu 01",&FuncMenu_01,0};
MENU_ENTRY Menu_02 = {0,2,"2.Test Menu 02",&FuncMenu_02,0};
MENU_ENTRY Menu_03 = {0,3,"3.Test Menu 03",&FuncMenu_03,0};

void prc_menu_Test(void)
{
    MENU Menu_Test = {3,1,0,"Menu Test!!", {&Menu_01, &Menu_02,
    &Menu_03}};
     prc_menu (&Menu_Test);
}
```

Includes： #include “LIB_CL.h ”

Description： The prc_menu_color function is used to create a user-defined menu. SMENU and SMENU_ENTRY structures are defined in ”LIB_CL.h”. Users can just fill the SMENU_ENTRY structure and call the prc_menu function to build a hierarchy menu-driven user interface.

```
struct SMENU
{
    int total_entry;
    //For total selections. For example, you have a title and three
    selections, this value has to be '3'.
    int selected_entry;
    //For first select. For example, you have three selections, and this
    value is '2', the first selects is selection 2.
    int ReturnFlag;
    //For return mode. If this value is 0, the menu will not return until "ESC"
    is pressed. If this value is 1, the menu will return after any selection
    process end.
    char* title;
    //Title string.
    struct SMENU_ENTRY* entry_list[9];
};

struct SMENU_ENTRY
{
    int text_x;
    //text x position.
    int text_y;
    //text y position.
    char* text;
    //text string for selection name.
    void (*func)(void);
    //Function point for selection.
    struct SMENU *sub_menu;
    //Menu point for selection.
    If menu point is not NULL, the function point must be NULL.
    If function point is not NULL, the menu point must be NULL.
};

typedef struct SMENU MENU;
typedef struct SMENU_ENTRY MENU_ENTRY;
```

Returns： None

prc_menu_scroll

Purpose： Create a menu-driven interface with scroll function.

Syntax： void prc_menu_scroll(MENU_SCROLL *stMenu_Scroll);

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_SCROLL_ENTRY stMenu_TestSub[3] =
 {{"1.Test Menu 01", & FuncMenu_01, 0},
  {"2.Test Menu 02, &FuncMenu_02, 0 },
  {"3.Test Menu 03", & FuncMenu_03, 0}};

MENU_SCROLL stMenu_Test = {3, 1, 0, "    Menu Test ", 0, 3,
stMenu_TestSub};

void prc_menu_Test(void)
{
        prc_menu_scroll(&stMenu_Test);
}
```

Includes： #include “LIB_CL.h ”

Description： The prc_menu_scroll function is used to create a user-defined menu. SMENU_SCROLL and SMENU_SCROLL_ENTRY structures are defined in ”LIB_CL.h”. Users can just fill the SMENU_SCROLL_ENTRY structure and call the prc_menu function to build a hierarchy menu-driven user interface with scroll function.

```
struct SMENU_SCROLL
{
    int total_entry;
  //For total selections. For example, you have a title and three
  selections, this value has to be '3'.
    int selected_entry;
  //For first select. For example, you have three selections, and this
  value is '2', the first selects is selection 2.
    int ReturnFlag;
  //For return mode. If this value is 0, the menu will not return until "ESC"
  is pressed. If this value is 1, the menu will return after any selection
  process end.
    char* title;
  //Title string.
    int title_y;
  //Title y position.
    int scroll_line;
  //The range for selections display. For example, you have 3 selections
  but set scroll_line = 2, it will display 2 selections, and press up or down
  can select the last selection.
    struct SMENU_SCROLL_ENTRY* entry_list;
};

struct SMENU_SCROLL_ENTRY
{
   char *text;
  //text string.
    void (*func)(void);
  //Function point for selection.
    struct SMENU_SCROLL *sub_menu;
  //Menu point for selection.
  If menu point is not NULL, the function point must be NULL.
  If function point is not NULL, the menu point must be NULL.
};

typedef struct SMENU_SCROLL MENU_SCROLL;
typedef struct SMENU_SCROLL_ENTRY MENU_SCROLL_ENTRY;
```

Returns： None

prc_menu_Set_SelectWithEnt

Purpose： Set function “prc_menu” and “prc_menu_scroll” 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 “LIB_CL.h ”

Description： Use prc_menu_Set_SelectWithEnt function can set prc_menu and prc_menu_scroll ENT key status when use number key to select menu.

Returns： None

prc_menu_Get_SelectWithEnt

Purpose： Get function “prc_menu” and “prc_menu_scroll” 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 “LIB_CL.h ”

Description： Use prc_menu_Get_SelectWithEnt function can get prc_menu and prc_menu_scroll ENT key status when use number key to select menu.

Returns： Get prc_menu_color with ENT key status.

prc_menu_GetMenuSelect

Purpose： After use function “prc_menu” and “prc_menu_scroll”, it can get what option is selected in these functions.

Syntax： int prc_menu_GetMenuSelect(void);

Example call： int i;
i = prc_menu_GetMenuSelect();

Includes： #include “LIB_CL.h ”

Description： After get prc_menu or prc_menu_scroll, use prc_menu_GetMenuSelect function can get what option is selected in these functions.
For example, a menu have 3 options, and you select first option, after select option and exit menu or enter other function by menu setting, use this function will get a return value “1”.

Returns： -1:Esc for exit prc_menu or prc_menu_scroll.
Others: Selected option.

DataMagic_Set

Purpose： Set a Data Magic file for function “_scanf_DataMagic” or “DataMagic_Run” to use.

Syntax： int DataMagic_Set(char *pssPath);

Example call： int i;

i = DataMagic_Set(“C:\\Data\\PI-1X.dmf”);

Includes： #include “LIB_CL.h ”

Description： This function can select the DataMagic file that you want to use.

After select a file, you can use this function to change other DataMagic file.

Returns： 1:Success

-1:No DataMagic file in path.

-2:DataMagic file is opened.

-3:DataMagic file is not correct.

-4:Version is not match, please update kernel.

-5:PI-10(CCD)/PI-12(2D) is not match. Please create a correct DataMagic file for your terminal(CCD or 2D)

-6:DataMagic file has some problem, cannot run.

DataMagic_Run

Purpose： Convert a string by Data Magic file setting.

Syntax： int DataMagic_Run(char* assProFile_Name, int slGetFrom, short smBarcodeType, char* pssStr);

Example call：

```
int i;
char assBuffer[100];
InitScanner1();
while(1)
{
   if Decode()
   {
      strcpy(assBuffer, CodeBuf);
      i = DataMagic_Run(“ProFile”, 1, (short)CodeType, assBuffer);
   }
}
```

Includes： #include “LIB_CL.h ”

Description： Use this function can convert a string by DataMagic file setting.

char* assProFile_Name:In DataMagic file, it has profile name, this parameter use to select the profile.

int slGetFrom: 0:By other input 1:By scanner. If you set check barcode type in DataMagic file, but set this parameter “0”, this function will not check barcode type. If this parameter is “1”, this function will check barcode type.

short smBarcodeType: For check barcode type.Please put the barcode type whitch get from scanner.

char* pssStr: String for this function to convert.

Returns： 1:Success

-10:No DataMagic file is set.

-11:No profile, convert fail.

-12:In profile, no rule can be converted.

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 “LIB_CL.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 “LIB_CL.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

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 “LIB_CL.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 “LIB_CL.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 “LIB_CL.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).

RF

RFHost_Open

Purpose： Start RF module.

Syntax： int RFHost_Open(void)

Example call： RFHost_Open();

Includes： #include “LIB_CL.h”

Description： The function will start the RF module work.

When open RF module, the module need 1 second to start.

Returns： 0:Open fail.

1:RF module is opened.

2:RF module is opening.

RFHost_Close

Purpose： Stop RF module.

Syntax： void RFHost_Close(void);

Example call： RFHost_Close();

Includes： #include “LIB_CL.h”

Description： The function will stop the RF module work.

Returns： None

RFHost_CallTagID

Purpose： Call the tag.

Syntax： int RFHost_CallTagID(char *pssID, short smBuzzer);

Example call： RFHost_CallTagID(“00F1F2F3”, 0x155);

Includes： #include “LIB_CL.h”

Description： The function can call tag by ID, and tag will beep.

The ‘pssID’ is for ID of tag.

The ‘smBuzzer ‘ setting as follows: (2 byte of beeping pattern.)

Bit	15	14	13	12	11	10	9	8
Function	Reserve						Loop	
Bit	7	6	5	4	3	2	1	0
Function	Stop		Beep3		Beep2		Beep1	

Loop:For beep loop times, 0 and 1=> 1time, 2 =>2times, 3=>3times

Beep1~3:For beep time. 0 and 1=> 100ms, 2=>200ms, 3=>300ms

Stop: For beep stop time. 0 and 1=> 100ms, 2=>200ms, 3=>300ms

Returns： 1: Success.

	0: Fail.
	-1: Not open RF.
	-2: ID is incorrect.

RFHost_GetVersion

Purpose：	Get RF module firmware version.
Syntax：	int RFHost_GetVersion(char *pssVersion)
Example call：	char assVersion[12];
	memset(assVersion, 0x0, 12);
	RFHost_GetVersion(assVersion);
Includes：	#include “LIB_CL.h”
Description：	The function can get the RF module firmware version.
Returns：	1: Success.
	0: Fail.
	-1: Not open RF.

APPENDIX 1：

Scan Module (CCD) Configuration Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
6 Transmission	1 Preamble transmission	0: Disable 1: Enable *
	2 Postamble transmission	0: Disable 1: Enable *
	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	1 Scanning Mode	1:Momentary(*) 4:Continue 6:Continue-with aim light
	2 Standby duration	1 ~ 99(default: 6)
	4 Double confirm	0 ~ 9 0 *
	6 Global min. code length	0 ~ 99 4 *
	7	0 ~ 99

	Global max. code length	63 *
	8 Inverted image scan	0: Disable * 1: Enable
8 String setting	1 Prefix characters setting	0 * 0x00 ~ 0xff ASCII code 12 characters.
	2 Suffix characters setting	0 * 0x00 ~ 0xff ASCII code 12 characters.
	3 Preamble characters settings	0 * 0x00 ~ 0xff ASCII code 12 characters.
	4 Postamble characters settings	0 * 0x00 ~ 0xff ASCII code 12 characters.
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/One digit * 2:Disable/Two digits 3:Enable/One digit 4:Enable/Two digits
	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 transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /Enable

	4 Max. code length	0 ~ 64 0 *
	5 Min. code length	0 ~ 64 1 *
	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 transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /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 transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /Enable
	4 Max. code length	0 ~ 64 0 *

	5 Min. code length	0 ~ 64 1 *
	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 transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /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	1	0: Disable

EAN 8	Read	1: Enable *
	2 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 *
	2 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 ISBN/ISSN conversion	0: Disable * 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 transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /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	0:Disable/Disable *

	Check-sum transmit /verify	1:Disable/Enable 2:Enable /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)
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:N/disable * 1:N/MOD 10 2:N/Mod 10,10 3:N/mod 11,10 4:Y/ Mod10 5:Y/ Mod 10,10 6:Y/ Mod 11/10
	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)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable 1:Disable/Enable * 2:Enable /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)
22 Telepen	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /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 *
	2 Check-sum transmission	0: Disable 1: Enable *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8	<A>

	Code ID setting	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 *
	2 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 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check-sum transmit /verify	0:Disable/Disable * 1:Disable/Enable 2:Enable /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	<B> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
28 China post	1 Read	0: Disable * 1: Enable
	4 Max. code length	0 ~ 64 11 *
	5 Min. code length	0 ~ 64 11 *
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<t> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
29 RSS 14	1 Read	0: Disable * 1: Enable
	6 Truncate leading	0 ~ 15 0 *
	7 Truncate ending	0 ~ 15 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 Truncate ending	0 ~ 15 0 *
	8 Code ID setting	<p> 0x00 ~ 0xff ASCII code(1 or 2 bytes)
	10 Leading “A”	0: Disable * 1: Enable

CCD Barcode Type Table:

value	Barcode type	value	Barcode type
100	Code 11	113	UPCA
101	Code 39	114	UPCE
102	Code 93	115	Matrix 25
103	Code 128	118	China Post
104	Codabar	119	RSS 14
105	EAN8	120	RSS Limited
106	EAN13	121	RSS Expanded
107	Industrial 2 of 5	122	Pharama code39
108	Interleaved 2 of 5		
109	Standard 2 of 5		
110	MSI Plessey		
111	UK Plessey		
112	Telepen		

APPENDIX 2：

Scan Module (2D) Configuration Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
	4 Vibrator	0: Disable 1: Enable *
6 Transmission	8 Transmit Code ID	0: None(*) 1: AIM Code ID 2: Symbol Code ID
7 Scan	17 Timeout	5~99(0.1 sec.) Default: 99(9.9 sec)
	20 Trigger Mode	0:Trigger(*) 7: Hand-Free Mode 9:Auto
	21 Picklist Mode	0: Disable(*) 2: Enable
	22 Same Barcode Timeout	5~99(0.1 sec.) Default: 6 (0.6 sec)
	23 Mobile Phone/Display Mode	0: Disable(*) 3: Enable
	27 Illumination Power Level	1~10 (default: 10)
	28 Decoding Illumination	0: Disable 1: Enable(*)
	29 Decode Aiming Pattern	0: Disable 2: Enable(*)
	32 Inverse 1D	0: Regular(*) 1: Inverse 2: Inverse Autodetect
10 Code 11	1 Read	0: Disable(*) 1: Enable
	2 Check Digit Verification	0: Disable(*) 1: 1 Check Digit

		2: 2 Check Digits
	3 Transmit Check Digit(s)	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 4)
	5 Length 2※1	0 ~ 55 (default: 55)
11 Code 39	1 Read	0: Disable 1: Enable(*)
	2 Check Digit Verification	0: Disable(*) 1: Enable
	3 Transmit Check Digit	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 2)
	5 Length 2※1	0 ~ 55 (default: 55)
	10 Full ASCII Conversion	0: Disable(*) 1: Enable
	19 Code 32 Prefix	0: Disable(*) 1: Enable
	20 Trioptic Code 39	0: Disable(*) 1: Enable
	21 Convert Code 39 to Code 32	0: Disable(*) 1: Enable
12 Code 93	1 Read	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 4)
	5 Length 2※1	0 ~ 55 (default: 55)
13 Code 128	1 Read	0: Disable 1: Enable(*)
	4 Length 1※1	0 ~ 55 (default: 0)
	5 Length 2※1	0 ~ 55 (default: 0)
	14 ISBT 128	0: Disable 1: Enable(*)
	15	0: Disable

	GS1-128	1: Enable(*)
	19 ISBT Concatenation	0: Disable(*) 1: Enable 2: Auto
	20 Check ISBT Table	0: Disable 1: Enable(*)
	21 ISBT Concatenation Redundancy	2 ~ 20 (default: 10)
14 Codabar	1 Read	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 5)
	5 Length 2※1	0 ~ 55 (default: 55)
	14 CLSI Editing	0: Disable(*) 1: Enable
	15 NOTIS Editing	0: Disable(*) 1: Enable
15 EAN 8/JAN 8	1 Read	0: Disable 1: Enable(*)
	20 Zero Extend	0: Disable(*) 1: Enable
16 EAN 13/JAN 13	1 Read	0: Disable 1: Enable(*)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable(*)
	2 Check Digit Verification	0: Disable(*) 1: USS Check Digit 2: OPCC Check Digits
	3 Transmit Check Digit	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 14)
	5 Length 2※1	0 ~ 55 (default: 0)
	9 Convert to EAN 13	0: Disable(*) 1: Enable
20 MSI Plessey	1 Read	0: Disable(*) 1: Enable
	2	0: One Check Digit(*)

	Check Digits	1: Two Check Digits
	3 Transmit Check Digit	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 4)
	5 Length 2※1	0 ~ 55 (default: 55)
	9 Check Digit Algorithm	0: MOD 10/MOD 11 1: MOD 10/MOD 10(*)
23 UPCA	1 Read	0: Disable 1: Enable(*)
	3 Transmit Check Digit	0: Disable 1: Enable(*)
	20 UPC-A Preamble	0: Disable 1: System Character Only(*) 2. System Character and Country Code
24 UPCE	1 Read	0: Disable 1: Enable(*)
	3 Transmit Check Digit	0: Disable 1: Enable(*)
	13 UPC-E1	0: Disable(*) 1: Enable
	14 Convert UPC-E to UPC-A	0: Disable(*) 1: Enable
	20 Transmit UPC-E1 Check Digit	0: Disable 1: Enable(*)
	21 Convert UPC-E1 to UPC-A	0: Disable(*) 1: Enable
	22 UPC-E Preamble	0: Disable 1: System Character Only(*) 2. System Character and Country Code
	23 UPC-E1 Preamble	0: Disable 1: System Character Only(*) 2. System Character and Country Code
25 Matrix 2 of 5	1 Read	0: Disable(*) 1: Enable
	2 Check Digit	0: Disable(*) 1: Enable
	3	0: Disable(*)

	Transmit Check Digit	1: Enable
	4 Length 1※1 5 Length 2※1	0 ~ 55 (default: 14) 0 ~ 55 (default: 0)
26 PDF-417	1 Read	0: Disable 1: Enable(*)
33 MicroPDF	1 Read	0: Disable(*) 1: Enable
	11 Code 128 Emulation	0: Disable(*) 1: Enable
35 UPC/EAN	1 Bookland EAN	0: Disable(*) 1: Enable
	2 Bookland ISBN Format	0: Bookland ISBN-10(*) 1: Bookland ISBN-13
	3 UCC Coupon Extended Code	0: Disable(*) 1: Enable
	5 Supplemental	0: Ignore supplemental (*) 1: Decode with Supplemental only 2: Auto discriminate Supplemental 3: Smart Supplemental Mode ※ Applies to EAN-13 starting with any prefix listed previously 4: Enable 378/379 Supplemental 5: Enable 978/979 Supplemental ※ If you select 978 Supplemental Mode and are scanning Bookland EAN bar codes, you should enable Bookland EAN, and select a format using Bookland ISBN Format. 6: Enable 414/419/434/439 Supplemental 7: Enable 977 Supplemental 8: Enable 491 Supplemental 9: Supplemental User-Programmable Type 1 ※ Applies to EAN-13 bar codes starting with a 3-digit user-defined prefix. Set this 3-digit prefix using User-Programmable Supplemental. 10: Supplemental User-Programmable Type 1 and 2 ※ Applies to EAN-13 bar codes starting with either of two 3-digit user-defined prefixes. Set the 3-digit prefixes using User-Programmable Supplemental. 11: Smart Supplemental Plus User-Programmable 1 ※ Applies to EAN-13 bar codes starting with any prefix listed previously or the user-defined

		prefix set using User-Programmable Supplemental. 12: Smart Supplemental Plus User-Programmable 1 and 2 ※ Applies to EAN-13 bar codes starting with any prefix listed previously or one of the two user-defined prefixes set using *User-Programmable Supplemental*.
	6 Supplemental Redundancy	2 ~ 30 (default: 10)
	7 User-Programmable Supplemental 1	-1 ~ 999(default:-1)
	8 User-Programmable Supplemental 2	-1 ~ 999(default:-1)
	9 UPC/EAN/JAN Supplemental AIM ID Format	0: Separate 1: Combined(*) 2: Separate Transmission
	10 Coupon Report	0: Old Coupon Symbols 1: New Coupon Symbols(*) 2: Both Coupon Formats
	11 ISSN EAN	0: Disable(*) 1: Enable
45 Australia Post	1 Read	0: Disable 1: Enable(*)
	2 Format	0: Auto(*) 1: Raw Format 2: Alphanumeric Encoding 3: Numeric Encoding
48 Japan Postal	1 Read	0: Disable 1: Enable(*)
49 KIX Code	1 Read	0: Disable 1: Enable(*)
51 USPS	1 Read	0: Disable(*) 1: Enable
52 UPU	1 Read	0: Disable(*) 1: Enable
53 Aztec	1 Read	0: Disable 1: Enable(*)
	10 Inverse	0: Regular(*) 1: Inverse 2: Auto
54	1	0: Disable

Data Matrix	Read	1: Enable(*)
	6 Inverse	0: Regular(*) 1: Inverse 2: Auto
55 Maxicode	1 Read	0: Disable 1: Enable(*)
58 OCR	1 Read	0: OCR off (*) 1: OCR-A 2: OCR-B 3. US Currency 4. MICR E13B
	2 OCR-A Variant ※2	0: OCR-A Full ASCII(*) 1: OCR-A Reserved 1 2: OCR-A Reserved 2 3: OCR-A Banking
	3 OCR-B Variant ※3	0: OCR-B Full ASCII(*) 1: OCR-B Banking 2: OCR-B Limited 3: OCR-B Travel Document Version 1 (TD1) 3 Line ID Cards 4: OCR-B Passport 6: OCR-B ISBN 10-Digit Book Numbers 7: OCR-B ISBN 10 or 13-Digit Book Numbers 8: OCR-B Travel Document Version 2 (TD2)2-Line ID Cards 9: OCR-B Visa Type A 10: OCR-B Visa Type B 14:Travel Document 2 or 3-Line ID Cards Auto-Detect
	4 OCR Orientation	0: OCR Orientation 0(*) 1: OCR Orientation 270 Clockwise 2: OCR Orientation 180 Clockwise 3: OCR Orientation 90 Clockwise 4: OCR Orientation Omnidirectional
	5 OCR Lines	1: OCR 1 Line(*) 2: OCR 2 Line 3: OCR 3 Line
	6 OCR Minimum Characters	3(*) Range:3~100
	7 OCR Maximum Characters	100(*) Range:3~100
	8	50(*)

	OCR Quiet Zone	Range:20~99
	9 Inverse OCR	0: Regular Only(*) 1: Inverse Only 2: Autodiscriminate
59 Discrete 2 of 5	1 Read	0: Disable(*) 1: Enable
	4 Length 1※1	0 ~ 55 (default: 12)
	5 Length 2※1	0 ~ 55 (default: 0)
60 Chinese 2 of 5	1 Read	0: Disable(*) 1: Enable
61 GS1 Data Bar	1 GS1 DataBar-14	0: Disable 1: Enable(*)
	2 GS1 DataBar Limited	0: Disable(*) 1: Enable
	3 GS1 DataBar Expanded	0: Disable(*) 1: Enable
	4 Convert to UPC/EAN	0: Disable(*) 1: Enable
	5 GS1 DataBar Limited Security Level	1: Level 1 2: Level 2 3: Level 3(*) 4: Level 4
62 Korean 3 of 5	1 Read	0: Disable(*) 1: Enable
63 Postal codes	1 US Postnet	0: Disable 1: Enable(*)
	2 US Planet	0: Disable 1: Enable(*)
	3 Transmit US Postal Check Digit	0: Disable 1: Enable(*)
	4 UK Postal	0: Disable 1: Enable(*)
	5 Transmit UK Postal Check Digit	0: Disable 1: Enable(*)
64 Composite	1 Composite CC-C	0: Disable(*) 1: Enable
	2 Composite CC-A/B	0: Disable(*) 1: Enable

	3 Composite TLC-39	0: Disable(*) 1: Enable
	4 UPC Composite Mode	0: UPC Never Linked(*) 1: UPC Always Linked 2: Auto
	5 GS1-128 Emulation Mode for UCC/EAN Composite Codes	0: Disable(*) 1: Enable
65 QR Code	1 Read	0: Disable 1: Enable(*)
	2 Inverse	0: Regular(*) 1: Inverse 2: Auto
66 Micro QR	1 Read	0: Disable 1: Enable(*)
68 Symbology Specific Security Levels	1 Redundancy Level	1: Level 1(*) 2: Level 2 3: Level 3 4: Level 4
	2 Security Level	0: Level 0 1: Level 1(*) 2: Level 2 3: Level 3
	3 Intercharacter Gap Size	6:Normal(*) 10:Large

※1:

- **One Discrete Length**: To limit the decoding of Barcode to specific length, assigned this length to **Length1** and 0 to **Length2**. For example, for fixed length 14, set **Length 1 = 14, Length2 = 0**.
- **Two Discrete Lengths**: To limit the decoding of Barcode to either of two specific lengths, assigned greater length to **Length1** and lesser to **Length2**. For example, to decode barcode codes of either 2 or 14 characters only, set **Length 1 = 14, Length2 = 2**.
- **Length Within Range**: To decode Barcode that fall within a specific length range, assigned lesser length to **Length1** and greater to **Length2**. For example, to decode barcode codes of length 4 through 12 characters, set **Length 1 = 4, Length2 = 12**.

※2:

OCR-A Variant

OCR-A supports the following variants:

- **OCR-A Full ASCII:**
  !"#$()*+,-./0123456789<>ABCDEFGHIJKLMNOPQRSTUVWXYZ\^

- **OCR-A Reserved 1:**
  $*+-./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **OCR-A Reserved 2**
  $*+-./0123456789<>ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **OCR-A Banking**
  -0123456789<> ϒ⊓ſ
  - ϒ outputs as f
  - ⊓ outputs as c
  - ſ outputs as h

※3:

OCR-B Variant

OCR-B supports the following variants:

- **OCR-B Full ASCII**
  !#$%()*+,-./0123456789<>ABCDEFGHIJKLMNOPQRSTUVWXYZ^|Ñ
- **OCR-B Banking**
  #+-0123456789<>JNP|
- **OCR-B Limited**
  +,-./0123456789<>ACENPSTVX
- **OCR-B ISBN 10-Digit Book Numbers**
  -0123456789>BCEINPSXz
- **OCR-B ISBN 10 or 13-Digit Book Numbers**
  -0123456789>BCEINPSXz
- **OCR-B Travel Document Version 1 (TD1) 3-Line ID Cards**
  -0123456789<ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **OCR-B Travel Document Version 2 (TD2) 2-Line ID Cards**
  -0123456789<ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **OCR-B Travel Document 2 or 3-Line ID Cards Auto-Detect**
  !#$%()*+,-./0123456789<>ABCDEFGHIJKLMNOPQRSTUVWXYZ^|Ñ
- **OCR-B Passport**
  -0123456789<ABCDEFGHIJKLMNOPQRSTUVWXYZÑ
- **OCR-B Visa Type A**
  -0123456789<ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **OCR-B Visa Type B**
  -0123456789<ABCDEFGHIJKLMNOPQRSTUVWXYZÑ

Variant OCR Lines Setting

Passport	2
TD1 ID Cards	3
TD2 ID Cards	2
Visa Type A	2
Visa Type B	2

2D Barcode Type Table

Code Type	Code ID	Code Name
100	H	Code 11
101	B	Code 39
102	E	Code 93
103	D	Code 128
104	C	Codabar
105	A	EAN8
106	A	EAN13
108	F	Interleaved 2 of 5
110	J	MSI Plessey
113	A	UPC-A
115	S	Matrix 2 of 5
116	X	PDF-417
123	X	Micro PDF
125	G	IATA
133	T	TLC-39
134	P04	Planet (US)
135	P03	Postnet (US)
136	P09	Postal (Australia)
138	P07	Postbar (CA)
139	P05	Postal (Japan)
142	P0A	4State US
143	P0B	4State US4
144	z	Aztec
145	P00	DataMatrix
146	P02	Maxicode
147	P01	Micro QR Code
149	None	OCR
152	G	Discrete 2 of 5
153	M	Code 39 Trioptic
156	A	UPCE1
157	L	Bookland
158	N	Coupon Code
159	R	GS1 DataBar-14
160	R	GS1 DataBar Limited
161	R	GS1 DataBar Expanded
162	D	ISBT-128
209	A	UPC-E
215	K	GS1-128
219	B	Code 39 Full ASCII
229	X	Micro PDF CCA
232	B	Code 32
233	D	ISBT-128 Concat.
236	P08	Postal (Dutch)
239	P06	Postal (UK)
240	X	Macro PDF

241	X	Macro QR
244	P01	QR Code
246	z	Aztec Rune
254	X	ISSN
272	A	UPC-A + 2
273	A	UPC-E + 2
274	A	EAN-8 + 2
275	A	EAN-13 + 2
280	A	UPCE1 + 2
281	T	CC-A + GS1-128
282	T	CC-A + EAN-13
283	T	CC-A + EAN-8
284	T	CC-A + GS1 DataBar Expanded
285	T	CC-A + GS1 DataBar Limited
286	T	CC-A + GS1 DataBar-14
287	T	CC-A + UPC-A"
288	T	CC-A + UPC-E
289	T	CC-C + GS1-128
297	T	CC-B + GS1-128
298	T	CC-B + EAN-13
299	T	CC-B + EAN-8
300	T	CC-B + GS1 DataBar Expanded
301	T	CC-B + GS1 DataBar Limited
302	T	CC-B + GS1 DataBar-14
303	T	CC-B + UPC-A
304	T	CC-B + UPC-E
314	U	Chinese 2 of 5
315	V	Korean 3 of 5
336	A	UPC-A + 5
337	A	UPC-E + 5
338	A	EAN-8 + 5
339	A	EAN-13 + 5
344	A	UPCE1 + 5
354	X	Macro Micro PDF
380	R	GS1 DataBar Coupon