

Index

Chapter 1	Overview	3
§1.1	Controller's working process and programmer's developing procedures .	3
§1.2	Schedule playing sequence	4
§1.3	DLL calling method stdcall and cdecl	5
§1.4	Operation with multiple equipments at the same time	5
§1.5	Transmit of character string parameter while VB6 Calling DLL	6
Chapter 2	PLAYLIST.LY and Config.LY	7
§2.1	PLAYLIST.LY Overview	7
§2.2	PLAYLIST.LY timing info	7
§2.3	PLAYLIST.LY main area index	8
§2.4	PLAYLIST.LY areas number	8
§2.5	PLAYLIST.LY areas' position and size	9
§2.6	PLAYLIST.LY playing items	9
§2.7	PLAYLIST.LY playing item's property	9
§2.8	PLAYLIST.LY Schedule file example:	11
§2.9	Config.LY file	12
Chapter 3	File format	14
§3.1	XMP Image format	14
§3.2	XMP File format	15
§3.3	TMC File format	15
§3.4	Font file format	18
Chapter 4	Communication controlling functions	19
§4.1	Net communication initializing function SCL_NetInitial	19
§4.2	Serial communication initialization SCL_ComInitial	19
§4.3	Setup the receiving object's IP SCL_SetRemoteIP	20
§4.4	Designate receiving object's address code SCL_SetLEDNum	21
§4.5	Change the Target type SCL_TargetSCL2008	21
§4.6	Communication close SCL_Close	21
Chapter 5	Functions for file management and data sending/receiving	22
§5.1	Drivers' Format SCL_FormatDisk	22
§5.2	Fetch driver's free space SCL_FreeSpace	22
§5.3	Fetch the number of directory items SCL_DirItemCount	22
§5.4	Fetch file directory SCL_GetDirItem	22

§5.5	Send files SCL_SendFile	23
§5.6	Receive file SCL_ReceiveFile	23
§5.7	Delete files SCL_RemoveFile	24
§5.8	Create subdirectory SCL_MD	24
§5.9	Delete subdirectory SCL_RD	24
§5.10	Immediately display words SCL_ShowString	24
Chapter 6	Controlling Functions	26
§6.1	Reset controller SCL_Reset	26
§6.2	Play new schedule SCL_Replay	26
§6.3	Check controller clock time SCL_SetTimer	26
§6.4	Setup automatic switching off/on time SCL_SetOnOffTime	26
§6.5	Setup LED screen's brightness SCL_SetBright	27
§6.6	Setup temperature sampling SCL_SetTempOffset	27
§6.7	Setup power mode SCL_SetPowerMode	27
§6.8	Read the running info SCL_GetRunTimeInfo	27
§6.9	Reading the playing info SCL_GetPlayInfo	28
§6.10	Control extended switch SCL_SetExtSW	28
Chapter 7	Automatically organize and arrange data sending/recieving	29
§7.1	Initialize virtual equipment SCL_InitForPackage	29
§7.2	Fetch communication data packet SCL_GetPackage	30
§7.3	Check the response SCL_CheckAnswer	30
§7.4	Shutoff display SCL_LedShow	31
§7.5	Send data to controller's file buffer SCL_SendData	31
§7.6	Retrieve data from controller's buffer SCL_ReceiveData	31
§7.7	save the data from file buffer as a disk file SCL_SaveFile	32
§7.8	Save the Disc file in controller's buffer SCL_LoadFile	32
§7.9	Fetch local file to create DOS date and time SCL_GetFileDosDateTime	32
§7.10	SCL_GetDirItem's enablement	33
§7.11	SCL_SendFile's enablement	33
§7.12	SCL_ReceiveFile enablement	33
§7.13	Broadcast transfer	34
Chapter 8	File format convert	35
§8.1	Image file convert to XMP file SCL_PictToXMPFile	35
§8.2	XMP file size SCL_GetMaxFileSize	35
§8.3	Combine XMP files SCL_AddXMPToXMP	36
Chapter 9	Font	37
§9.1	Download font	37
§9.2	Words message	37
§9.3	Display extension code	38

Chapter 1 Overview

This manual can be suitable for both SuperComm and SCL2008, if anything special, it will be pointed out and described accordingly.

The pixel of the image in the SuperComm has 2 bytes, which is described as 16-bit image; the pixel of the image in SCL2008 has 2-bit, which is described as double-color image. So for the same size of the image, SCL2008 just need 1/8 of the capacity of SuperCOMM.

SuperComm serial/ SCL2008 serial controllers all use FAT16 file management system.

SuperComm controller has a SD card slot and RAM, SCL2008 has FLASH (Disk A) and RAM (Disk C) 2 memories. All the images for displaying are stored as a file in one of the controller's disks, the playing options such as enter method, stayinging time etc are all controlled by the playlist file, the format has to be PLAYLIST.LY.

The controller can store up to 100 schedules, according to the schedules index, the different files and the related images are stored in the different directories. For example, the schedule #00 in SD card is stored in B:\P00 sub-directory, and the #8 schedule in RAM is stored in C:\P08 sub-directory.

The controller's file management follows with the below structure:

1. the controller can store up to 100 schedules, each one is filed according to their sub directory, for example, P00 is stored with #0 schedule; P09 is stored with #9 schedule. When the controller is first powered on, it can only play one schedule, Config.Ly determines which schedule to play.
2. Each schedule is managed by Playlist.Ly, which can orderly arrange 200 programs;
3. Each program can has up to 4 displaying areas and their own time frames(SCL2008 has only 3 displaying areas)
4. Each area can arrange up to 200 images or word files, TXT files, RTF files, tables.and clock time etc.

§1.1 Controller's working process and programmer's developing procedures

If you look at the controller as a Microcomputer under DOS mode, then it will be easy to understand the controller's working process. SuperComm/SCL2008 enables FAT16 mangement. All the pending programs (images) and its playing motheds are all stored as files in one of the controller's disks.

When the controller is first powered on, it will first check Config.LY under the root directory in SD card (SCL2008: FLASH), Config.LY will assign which schedule to play, and check if it is needed to install the fonts or not (this file's use is similar as the Config.sys in DOS), then it will call the related Playlist.Ly to start the schedule (this file's use is similar as Autoexec.bat in DOS, it is not the only one under the root directory, each schedule has its respective PlayList.Ly in the root directory).

In a concrete application project, normally Config.Ly and PlayList.Ly and the related fonts (need font code for displaying characters) are not changable, they can be copied to controller's SD card or FIASH using SD card reader or the edit software. But some other information/data (which don't have to be storage for a long time), can be sent through the user's further developed programs to controller' RAM. And in some actual situration, the

displaying data need be concerned while the LED board is first powered on.

Normally the working process as below:

1. Create a font, and upload it into the Fon subdirectory in SD card or FLASH (or as 'Pxx' subdirectory in SD card or FLASH).
2. Use SuperComm/ SCL2008eidt program to make a schedule for displaying when the LED is first powered on, transfer all the contents into P00 subdirectory in Disk A or B.
3. Manually edit Config.LY, assign to play the program under P00 subdirectory when LED is first powered on, and automatically install the fonts.
4. Manually edit PlayList.Ly, and save it in controller's P01 subdirectory, all the programs in the Playlist.Ly are assigned to save in the disk C.

When the controller is just powered on, the font should be installed, and display the programs under P00's Playlist.Ly

While making the application programs, first need transfer the image (XMP format) or character string to DISK C (RAM), then call SCL_Replay command, have the controller play the programs under P01 schedule (the programs in this schedule are all in Disk C, which are exactly the images and text files just uploaded). Note: in order to reduce the reading time, after the controller load the program item, it will not repeatedly read the item for the displaying area with only 1 item. If the user wants to display the updated programs in realtime, just need write in 2 same program items, let the controller continually read the updated items to display.

When need communication with a controller, at first, it needs to initialize the communication hardware by calling SCL_NetInitial(net communication) or SCL_ComInitial(serial communication). When the communication is done, it needs to shut off the communication device by calling SCL_Close (for more details, please refer to the chapter "automatic arrange data sending/ receiving")

The programmer only need use the "control functions" in Chapter 6 in the below 3 situations: 1> the hardwares need be used or setup; 2> order controller to reset; 3> designate one program to display. The other remaining jobs are to arrange and organize the programs (images), then use the program managing function to upload the programs to controller's DISKS, (DiskA or B with limited writing times) or RAM (Disk C, data is not savable without power, but with infinite writing times)

IN the development packet, the font creating program is provided, as well as the program for checking XMP, TMC file.

§1.2 Schedule playing sequence

In Disc B (SCL2008: Disc A), there is a CONFIG.LY, which determines which program to display when the controller is just powered on. For a CONFIG.LY file example, the below means to play the PLAYLIST.LY schedule in P03 sub-directory in Disc B(SCL2008: Disc A):

```
[Startup]
```

```
Program_Index=3
```

After the controller is powered on and reset, the steps to search for schedules are as below:

1. first look up the CONFIG.LY in Disc B(SCL2008: Disc A), if it exists in the disk, and has the schedule playing index, then default searching settings is Disk B(SCL2008:A) and the Config.LY assigned index, then go to step 2; otherwise, the default setting will be Disk C, index is 0, go to step 2.

2. In the same driver, under the root directory, if the schedule exists, then start the schedule; otherwise, go to next step.
3. For all the drivers, in order of C, B (SCL2008: C, A), search the schedule in root directory, if it exists, then start it, otherwise, go to next step.
4. For all the drivers, in order of C, B (SCL2008: C,A) search the schedule under P00, if it exists, then start it, otherwise, go to next step;
5. If the schedule were not found, the controller will automatically work under the test mode. In the editing program, it can be set to or automatically display patterns, black LED board or just displays 4 spots on the top right conner.

It will directly access to the above step 2 by calling SCL_Replay function.

§1.3 DLL calling method stdcall and cdecl

The functions for further development are provided to the users by Dynamic link library. There are 2 DLL files. SuperCommSCL2008.Dll includes lower level communication functions, which is together with program editor installed in Windows\System32 file; Based on SuperComSCL2008.Dll, it re-encapsulates SCL_API_Stdcall.Dll and SCL_API_Cdecl.Dll, which provide Stdcall and Cdecl 2 calling method, Except for communication functions, the re-encapsulated DLL still provides some convinient functions to deal with the image data, these functions will be described in detail in this manual

In order for programmers's easy development, together with the development packet, SCL_API_Dll.Pas in Delphi language and SCL_API_Dll.h in C language are provided for the programmer use. It can choose the calling method by changing LIBRARY_CALL_MODE definition. SCL_API_Dll.Pas is an unit, it can be directly imported to Delphi project; SCL_API_Dll.h packet contains function definition and explicit Dll codes, need copy the source code in this file to the project's type definition, initialization and quit 3 sections in programmer's project. Please refer to SCL_API_Dll.h to use it.

The data types used in this manual as below:

BOOL,LongBool: 4 bytes bool type, Non-0: TRUE, 0: FALSE;
 Int,Integer: 4 bytes signed integer
 WORD,Word: 2 bytes unsigned integer
 BYTE,Byte: 1 byte unsigned integer

The programming language in this manual is based on VC++6.0 和 Delphi 7。

When in need, programmers can also refer to <<SCL communication protocol>>, to develop program, to directly communicate with controller.

§1.4 Operation with multiple equipments at the same time

Dynamic link library introduced in this manual can be used for multiple equipments, and run pallelle operations.

In the Dynamic link library, it can deal with up to 256 communication devices at the same

time, for example, multiple serial ports, or multiple UDP terminals. Any different 16-bit code can be used to identify which device for communication,

While using the 3 functions SCL_NetInitial, SCL_ComInitial or SCL_InitForPackage, the programmer need give an equipment code. If the 3 functions return to "true", then the device is successfully initialized, and in the future, programmer need use the same device code to call the communication function.

In the chapter, hereinafter in this manual, when it introduces each function with nDevID in the different chapters, nDevID will stand for a device code, it doesn't mean something else.

§1.5 Transmit of character string parameter while VB6 Calling DLL

DLL is enabled in C language, it uses character string char* type for the data transferring, the string is of 8 bit, ended with a "0". Although the character string in VB is of 16 bit, it can also be simply transferred to Dll in C language, to enable this, it just needs to define the the character string as ByVal in the parameters table.

```
Private Declare Function SCL_SetRemoteIP Lib "SCL_API_StdCall.dll" (ByVal nDevID  
As Integer, ByVal IP As String) As Boolean
```

Then in VB6, it just need directly get the offset of character string in the IP parameters.

```
Dim IPVBStr As String  
Dim RetV As Boolean  
IPStr = "192.168.1.133"  
RetV = SCL_SetRemoteIP(nDevID,IPVBStr)
```

Chapter 2 PLAYLIST.LY and Config.LY

§2.1 PLAYLIST.LY Overview

PLAYLIST.LY is an INI type file. It can be opened and edited by notepad in Windows , and it can also be automatically created by the program.

PLAYLIST.LY basic structure as below:

```
[Program1]
  <Timing info>
  <Main area Index>
  <Areas number>
  [Program1_Screen1]
    <Area 1's location and size>
    <Area 1's playing items>
    <Property of item 1>
    <Property of item 2>
    ...
    <Property of item n>
  [Program1_Screen2]
    <Area 2's loacation and size>
    <Area 2' playing items>
    <Property of item 1>
    <Property of item 2>
    ...
    <Property of item m>
  .....
[Program2]
.....
[Program3]
.....
[ProgramN]
"N" can be up to 200 (namely one schedule can contain Max 200 programs)
```

§2.2 PLAYLIST.LY timing info

Each program can set its time frame. Before the controller runs this program, it will check the timing info and controller's clock, if the current time is in the time frame, then the program will play, otherwise it will not play.

One timing info consists of the below:

TimerX=MMmmDDddWWwwHHNNhhnn, x can be from 1-8, which means it can Max 8 timing frames

MM: the starting month

Mm: the ending month, if the timing is not related to month, then MM=01, mm=12,

which means from January to December.

DD: the starting date;

Dd: the ending date, if the timing is not related to date, then DD=01, dd=31, which means from 1st to 31th;

WW: the starting week date;

Ww: the ending week date, if it is not related to week date, then WW=01,ww=07, which means from Mon. to Sun.

HHNN: starting hour and minute

Hhnn: The ending hour and minute, if it is not related to hour and minute, then HHNN=0000, hhnn=2359, which means from 00:00 to 23:59;

For example, the program plays from 8:00am and 12:00pm on Saturday and Sunday from 10th to 25th of July and August, the timing info should be as below:

Timer1=07081025060708001200

Timer1-Timer8 all should be listed, none can be missed, otherwise the controller will spend a lot of time to search for the timing info, which causes the controller spend a long time to switch the programs. In case, the other timer is not in use, then it should be written as below:

Timer1=01120131010700002359

Timer2=00000000000000000000

Timer3=00000000000000000000

Timer4=00000000000000000000

Timer5=00000000000000000000

Timer6=00000000000000000000

Timer7=00000000000000000000

Timer8=00000000000000000000

§2.3 PLAYLIST.LY main area index

Each program should have one main playing area, the playing time is up to the main area's playing time. If the playing items in the main area are complete, then the program is complete. If the playing items in other areas are complete, but those in main areas are still playing, then the items in other areas will restart from the beginning.

The definition format of the main area is as below:

MainScreen=n, here the 'n' means one area's index , it can be from 1 to 4 (SCL2008:1-3), which should be less than the areas number.

§2.4 PLAYLIST.LY areas number

One program can have several independent areas for their respective displaying. The number of areas defines the quantity of independent areas. SuperComm can control 4 independent areas, SCL2008 can control 3 independent areas. The definition of the areas number as below:

Screen=n, here, the n can be 1-4(SCL2008:1-3)

§2.5 PLAYLIST.LY areas' position and size

The size and location of each area can be assigned at user's discretion. If the areas overlapped, then latter defined area will cover the former one, unless there isn't any updated items in the latter defined area, and the former one has updated items.

The definition of the area location and size as below:

Position=Left,Top,Width,Height

Left, Top are the topleft coordinate of the area, Width and Height are respectively the width and height of the area.

The Max range the controller can control is 960 columns (For SCL2008: the Max columns is 4096, but the actual usable Max column is 4032), 512 rows. Normally the controller is fixed on the right side (from front view), so the left coordinate need move to right. The most left coordinate is the Max controlling range--the width of LED board. Suppose the LED board is 256 columns wide, then the most left coordinate is $960-256=704$ (SCL2008: $4096-256=3840$).

§2.6 PLAYLIST.LY playing items

One area can play up to 200 playing items, the definition of the number of playing items as below:

ItemCount=n, the n is the actual number of playing items, it can be from 0-200.

§2.7 PLAYLIST.LY playing item's property

The playing item is a display-able document. The format to define the playing item as below:

ItemN=type,Drv:PathFileName,Entermode,Step,Staytime

ItemN: can be Item1 – Item255, the digits can not be greater than the previous defined playing items number, otherwise the playing item can not be played.

Type: file type, the value can be letter 'F', 'P', 'T', 'X' or 'S'.

'F'	FLASH images, which need be quickly and directly copied to displaying areas
'P'	Normal images
'T'	Clock item, which can display date and time, temperature, humidity, serial port info, countdown etc.
'X'	Text file, which need pre-upload font and be installed in Config.LY, text can include extension code, please refer to Chapter 7.
'S'	System command, which enable the program skip forward and backward (it supports multiple skips forward, but only one skip backward).

Drv: the driver of the current playing items, which can be A (FLASH), B (SD card) or C (RAM), if the file is in the same driver where PLAYLIST.LY is, then Drv is omissible.

Path: Path, whose format is "\Pxx\" or "\" (x is a digit character from 0-9), the former one is a subdirectory, the later is directory. If no path, then it means that the

file and schedules are in the same subdirectory.....

FileName: Playing item name. SuperComm/SCL2008 controllers only support the short file name in 8.3 format; if type is a system command, then here can be 2 command styles: one is to directly write the name RETURN, which means the program will run on before returning to program schedule; another is to have path and file written as a playlist file under another directory, for example, P14\PLAYLST.LY, which means to switch to play another program. If the SD card has P00\PLAYLIST.LY and P03\PLAYLIST.LY, and in P03\PLAYLIST.LY, the main area of the last program is a RETURN system command, while the system is processing P00\PLAYLIST.LY, if send SCL_Replay(1,3), then controller will switch to play P03\PLAYLIST.LY; but when the system receives a RETURN command during playing P03\PLAYLIST.LY, the controller will automatically return to resume the program in P00\PLAYLIST.LY. This process can be used to insert playing programs.

Entermode: Entermode, the value can be from 0-15, the effect can be defined as below:

Entering mode code	Entering effect	
	SuperComm	SCL2008
0	Random	Random
1	Directly	Directly
2	Move to top	Move to top
3	Move to left	Move to left
4	Melt in	Melt in
5	Cover to top	—
6	Cover to left	—
7	Expand from center	Expand form center
8	Move to bottom	Move to bottom
9	Move right	Move to right
10	Cover to bottom	—
11	Cover to right	—
12	Blinds horizontal	Blinds horizontal
13	Blinds vertically	Blinds vertically
14	Fan shaped	—
15	Diamond shaped	—
15	Flash	Flash

Step: Moving speed can be set from 1-8, the value is greater, the moving is faster.

Staytime: is the staying time on the screen after the pattern or words enter in, the time unit is : second.

Entering mode, moving speed and staying time are ineffective to flash, and entering mode and moving speed are ineffective to clock.

Item6=F,zzydq.s06,1,1,0

It indicates the 6th item is a Flash animation, file and PLAYLIST>LY are under the same directory in the same dirver.

§2.8 PLAYLIST.LY Schedule file example:

For one schedule including 3 programs, the complete PLAYLIST.LY file is as below
(Program1 has three areas, Program2 has two areas, program3 has one area):

[Program1]

Timer1=01120131010700002359

Timer2=00000000000000000000

Timer3=00000000000000000000

Timer4=00000000000000000000

Timer5=00000000000000000000

Timer6=00000000000000000000

Timer7=00000000000000000000

Timer8=00000000000000000000

MainScreen=1

Screen=3

[Program1_Screen1]

Position=576,0,263,86

ItemCount=1

Item1=P,SUPERC~1.d01,4,1,5

[Program1_Screen2]

Position=576,90,228,167

ItemCount=2

Item1=P,m71a.j01,0,1,3

Item2=P,zr29a.j01,0,1,3

[Program1_Screen3]

Position=808,90,153,167

ItemCount=1

Item1=T,H8W1C1AQ.tmc,1,1,65535

[Program2]

Timer1=01120131010700002359

Timer2=00000000000000000000

Timer3=00000000000000000000

Timer4=00000000000000000000

Timer5=00000000000000000000

Timer6=00000000000000000000

Timer7=00000000000000000000

Timer8=00000000000000000000

MainScreen=2

Screen=2

[Program2_Screen1]

Position=576,0,384,29

ItemCount=2

Item1=P,0002a.j01,1,1,5

Item2=P,0003a.j01,1,1,5

[Program2_Screen2]

```
Position=576,30,384,226
ItemCount=3
Item1=P,34a.x01,0,1,5
Item2=P,ANIMAL~1.j01,0,1,5
Item3=P,auto255c.j01,0,1,5
```

[Program3]

```
Timer1=01120131010700002359
Timer2=00000000000000000000
Timer3=00000000000000000000
Timer4=00000000000000000000
Timer5=00000000000000000000
Timer6=00000000000000000000
Timer7=00000000000000000000
Timer8=00000000000000000000
MainScreen=1
Screen=1
```

[Program3_Screen1]

```
Position=576,0,384,256
ItemCount=1
Item1=P,zr29a~00.j01,0,1,5
```

§2.9 Config.LY file

Configuration file is also an INI type file. It can be opened and edited in the notebook, in some situation, the programs editing software can automatically create this file (designate a non-0 playing index, or setup the temperature for LED screen automatic switching on/off or SW2 etc.)

The basic file format of Configuration file as below:

```
[Startup]
Program_Index=n
[Font]
FontCount=x
Font1=FontFileName1,CharCount1,Mode1,width1,height1[,Byte2Start1[,Code1Start1]]
Font2=FontFileName2,CharCount2,Mode2,width2,height2[,Byte2Start2[,Code1Start2]]
.....
Fontx=FontFileNameX,CharCountX,ModeX,widthX,heightX[,Byte2StartX[,Code1StartX]]
```

Above, 'n' indicates the playing program index when the controller is reset or powered on, it can be 0-99. In the SD card, it is corresponding to the schedule of P00-P99

'x' indicates the number of installed fonts, which is less than 8; FontFileName is a font file name with its path. Except for Pxx path, the font file can also be stored in a specified Font path; CharCount is the characters number of a font; Mode indicates the look-up method, it can be designated as either of 'C' or 'A', 'C' means double bytes coding, 'A' means the direct look-up ASCII characters. Width is the character's width, height is character's height.

For Chinese characters and other local language characters, they are normally double bytes coding, and the code of the first byte is greater than 0x80. The 2 bytes of the simplified Chinese characters start coding from 0xA0. Byte2Start gives the starting code of the second

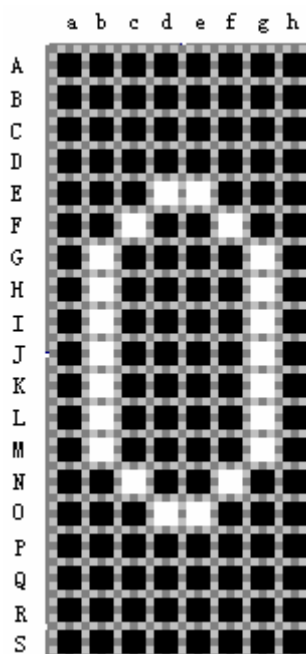
byte, for example, the simplified font is 0xA0, then here directly write A0, the traditional Chinese font is 0x40, then directly write 40; Byte1start gives the first byte's starting code, for example, the simplified Chinese character font is 0xA0, then here directly write AO, the traditional Chinese character font is 0x80, then here write 80. Their default value is 0x80.

Byte2Start and Byte1Start can reduce the font file length, and it can be more easily compatible with other language code.

Chapter 3 File format

§3.1 XMP Image format

The storing sequence of a XMP image's pixels is: row first, then column. For example, for the below picture, the pixels' storing sequence is below:



Aa,Ba,Ca,Da,.....Sa,
Ab,Bb,Cb,Db,.....Sb,
.....
Ah,Bh,Ch,Dh,.....Sh

16-bit XMP: in the SuperComm controller, every 2 bytes indicate a pixel. For an image with the height H , width W , there will need $H*W*2$ bytes for storage of a XMP image.

Double-color XMP without grayscale: in SCL2008 controller, every 2 bits indicate 1 pixel, each byte can show 4 pixels in vertical direction, for an image with height H , width W , there will need $(H+3)/4*W$ bytes for storage of a XMP image (here the '/' means divide exactly, for a few amount of rows, it is stored in a low order location, d7, d5, d3, d1 indicate green, d6,d4,d2,d0 indicate red, the adjacent 2 bits indicate one pixel. For the above picture, the color in A row is stored in d1, d0; the color in B row is stored in d3, d2; the color in C row is stored in d5,d4...

Single color XMP without grayscale: for font, 1 bit indicates 1 pixel of XMP image, each byte can store 8 pixels in vertical direction, for a character with height H and width W , it will use $(H+7)/8*W$ to show a character (here the '/' indicates divide exactly). The lines from beginning will be stored from the high order of the byte. For the above character, #A line is stored in d7, #B line is stored in D6, #C line is stored in d5.....

§3.2 XMP File format

Each XMP format file has 6 bytes of file head:

BYTE	XMPTType
BYTE	PictureCount
WORD	Height
WORD	Width

IN the above:

XMPTType:	file's format type
PictureCount:	the images number included in the XMP file
Height:	the height of the biggest image, Unit: Pixel
Width:	the width of the biggest image, Unit: Pixel

If XMPTType=0, it means a 16-bit color XMP image; after XMP file header, there are 4 bytes used for the image's height and width before the XMP image.

If XMPTType=1, it means a double color XMP image without grayscale, after XMP file header, there are 4 bytes used for the image's height and width before XMP image;

If XMPTType=2, it means a 16-bit images, after XMP file header, there orderly arrange the XMP images.

§3.3 TMC File format

In the schedule, the playing item marked with 'T' is time clock item, the corresponding file is a time item file, and normally its suffix is "TMC".

This consists of the below 4 parts:

[File header][Displaying item1][Displaying item2]..... [Font1][Font2]..... [Background picture]

[File header], 12 bytes:

[File version], 2 bytes

- 1: SuperComm's TMC file, the background picture is a XMP image with 3 16-bit colors;
- 2: SuperComm's TMC file, the background picture is a XMP image with 2 16-bit colors
- 3: SCL2008's TMC file, the background picture is a 2-color XMP image without grayscale;

[Height of the background picture], 2 bytes

[Width of the background picture], 2 bytes

[Displaying items of the file], 2 bytes

[Fonts number included in the file], 2 bytes

[Offset of the background picture], 2 bytes

[Displaying itme], 24 bytes:

[Displaying item type], 1byte。 Type code as below:

1:year,2:month,3:date,4:week,5:hour,6:minute,7:second,8:temperature,9:Humidity,1

0: countdown,11:serial data

[Displaying color], 3 bytes, First byte is red, the second byte is green, the third byte is blue;

[offset of the used font in the file], 2 bytes
 [Displaying bits], 1 byte
 [Displaying item property], 1 byte, for the countdown item's direction, counting forward (0), counting backward (1); for the serial data, it is the parameter serial number(1-8); For 'hour' data, it is a signed number, which is used for timezone's offset, for example, suppose the controller's hour value is '3', and the field is '-2', then it will display '1'; for the temperature, celsius (0) Fahrenheit(1).
 [Initial value of Countdown], 4 bytes, long integer.
 [Serial port number of serial data], 1 byte, 0: serial port 1, 1: serial port 2, 2: serial port 3
 [The mark hiding the leading 0], 1 byte, 0: display leading 0, 1: hide the leading 0
 [Change the item's year], 1 byte, to modulo 100
 [Change the item's month], 1 byte
 [Change the item's date], 1 byte. The above 3 items match the default value, which is used for controller's automatic countdown.
 [Unused], 1 byte
 [the output vertical coordinate of displaying item], 2 bytes
 [the output horizontal coordinate of displaying item], 2 bytes
 [Unused], 2 bytes

[Font]:

[Character height], 2 bytes
 [Character width], 2 bytes
 [bytes number used by each character], 2 bytes
 [Character dot matrix], the rule is "row first, then column, each byte indicates 8 pixels in the vertical direction", orderly store the characters '0'-'9', '+', '-', space, decimal's dot matrix (each character's dot matrix is a single color picture)

[Background picture]:

It is a XMP image conforming the [height] and [width] of [File header]'s image. If [File version]=2, then it is 16-bit color XMP image; if [File version]=3, then it is a double color XMP image.

In the TMC files, all the structures' formats in C language are as below:

```
#define WORD unsigned short;
#define BYTE unsigned char;
struct Tmc_Head
{
    WORD Flag;
    WORD AreaHeight;
    WORD AreaWidth;
    WORD ItemCount;
```

```

        WORD          FontCount;
        WORD          BkGroundOffset;
};
struct Item_Struct
{
    BYTE             AType;
    BYTE             Color_Red;
    BYTE             Color_Green;
    BYTE             Color_Blue;
    WORD             FontOffset;
    BYTE             BitCount;
    Signed char      CountDir; // when AType=5 is 'hour', this field is timezone offset.
    long             Value;
    BYTE             ComPort;
    BYTE             HideZero;
    BYTE             Year;
    BYTE             Month;
    BYTE             Day;
    BYTE             NC0;
    short            YPosi;
    short            XPosi;
    WORD             NC1;
};
struct Font_Struct
{
    WORD             CharHeight;
    WORD             CharWidth;
    WORD             ByteCount;
    BYTE             Font[];
};

```

In the TMC files, all the structures' formats in Delphi language are as below:

```

Tmc_Head = Packed Record
    Flag           : WORD;
    AreaHeight     : WORD;
    AreaWidth      : WORD;
    ItemCount      : WORD;
    FontCount      : WORD;
    BkGroundOffset : WORD;
End;

Item_Struct = Packed Record
    Atype          : BYTE;
    Color_Red      : BYTE;
    Color_Green    : BYTE;
    Color_Blue     : BYTE;
    FontOffset     : WORD;

```

```

    BitCount          : BYTE;
    CountDir          : short int; // when AType=5 is 'hour', this field is timezone
offset.
    Value             : Integer;
    ComPort           : BYTE;
    HideZero          : BYTE;
    Year              : BYTE;
    Month             : BYTE;
    Day               : BYTE;
    NC0               : BYTE;
    Yposi             : Smallint;
    Xposi             : Smallint;
    NC1               : WORD;
End;
Font_Struct = Packed Record
    CharHeight       : WORD;
    CharWidth        : WORD;
    ByteCount        : WORD;
    Font              : Array Of BYTE;
End;

```

§3.4 Font file format

All the single color XMP images without grayscale are orderly arranged as font file.

Chapter 4 Communication controlling functions

This chapter will introduce the communication initializing functions. Under the net mode, computer and controller adopt UDP protocol to exchange data, it will occupy the adjacent terminal (P, P+1). When under communication state, the computer will send the data through terminal P to controller's P+1 terminal; and controller will also send the data through Terminal P to the computer's terminal P+1. Before communication with controller, it has to call either of the 2 functions SCL_NetInitial or SCL_ComInitial; when the communication is done, it has to call SCL_Close to release communication equipment. The functions are in the parameter table, if there is a parameter nDevID, which is an equipment number, please refer to §1.4 for the detailed definition.

§4.1 Net communication initializing function SCL_NetInitial

C language:	<code>BOOL SCL_NetInitial(WORD nDevID, char *Password, char *RemotelIP, int SecTimeOut, int Retry, WORD UDPPort, BOOL bSCL2008);</code>
Delphi language:	<code>Function SCL_NetInitial(nDevID:WORD; Password:PChar; RemotelIP:PChar; SecTimeOut, RetryTimes:Integer; UDPPort:Word; bSCL2008:LongBool):LongBool;</code>
Function:	create Socket, and save the parameters like timeout, retry times, controller's IP and controller type etc.
Enter: nDevID:	it is a 16-bit equipment ID number given by the programmer. If the initialization is successful, then it has to use the same equipment ID while calling this communication function.
Password:	character string, password visiting
RemotelIP:	Character string, controller's IP address
SecTimeOut:	Communication timeout threshold value, if there was no response in the time of threshold value, the communication is regarded as failure, unit: second
RetryTimes:	Retry times after the failure of communication
UDPPort:	Create terminal for UDP communication
bSCL2008:	TRUE indicates the controller is SCL2008, FALSE indicates the controller is SuperComm
Exit:	If it is successful creating Socket, return to TRUE (Non-0), otherwise return to FALSE (0) (this is probably because the 256 communication equipments managed by DLL have been occupied, or the equipment has the same number as others, or the UDP port is repeated etc.)

§4.2 Serial communication initialization SCL_ComInitial

C language:	<code>BOOL SCL_ComInitial(WORD nDevID; int ComPort, int Baudrate, int LedNum, int SecTimeOut, int RetryTimes, BOOL bSCL2008);</code>
-------------	--

Delphi language: `Function SCL_ComInitial(nDevID: WORD; ComPort, Baudrate, LedNum, SecTimeOut, RetryTimes:Integer; bSCL2008:LongBool): LongBool;`

Function: Initialize serial communication resource, save the communication parameters: timeout, retry times, controller's number. Controller's serial communication uses 8 data bits, 1 stop bit, no checksum. For the controller's serial port, it has to setup the same format of data bit, stop bit and checksum, and choose 'Data Transfer'

Entrance: nDevID: It is a 16-bit equipment number given by programmers. If the initialization is successful, then it has to use the same equipment ID while calling this communication function in future.

ComPort: Computer's serial port number in the communication, 1-255;

Baudrate: Serial communication Baudrate, the value can be 2400、4800、9600、19200、38400 and 57600

LedNum: It is the controller's serial communication address code, 0-254. After the settings of IP address, the controller's address code will default the IP's last segment, for example, if the controller's IP is: 10.1.1.127, then its code is 127. Note: the code 255 is the broadcast code, any controller will responde to it when it receives the data for controller #255.

Note: SCL2008's default IP address is 10.1.1.100, so the address code is 100, but in the SCL2008Edit software, it starts from 0. Programmer still need use the number greater than 100.

SecTimeOut: Please refer to SCL_NetInitial

RetryTimes: Please refer to SCL_NetInitial

bSCL2008: Please refer to SCL_NetInitial

Exit: If initialization is successful, then return to TRUE(NON-0), otherwise return to FALSE(0) (it is probably because all the 256 communication equipments managed by DLL are occupied; or its number is the same as another equipment's; or the serial port is occupied; or the serial port doesn't exist at all; or the baudrate is not correct).

§4.3 Setup the reciving object's IP SCL_SetRemotelP

C language: `BOOL SCL_SetRemotelP(WORD nDevID,char *RemotelP);`

Delphi language: `Function SCL_SetRemotelP(nDevID:WORD; RemotelP:PChar):LongBool;`

Function: It is to setup the controller's IP address, and it has to call SCL_NetInitial function before using this function; and when an equipment communicates with multiple controllers, and when it need exchange data with one of the objects, it can designate the corresponding object. This way can avoid repeated calling SCL_Close, and calling SCL_NetInitial to repeatedly create Socket.

Entrance: RemotelPS: Controller's IP address

Exit: If the equipment is initialized, then return to TRUE (NON-0), otherwise return to FALSE (0)

§4.4 Designate receiving object's address code SCL_SetLEDNum

C Language: `BOOL SCL_SetLEDNum (WORD nDevID,int LedNum);`
Delphi Language: `Function SCL_SetLEDNum (nDevID:WORD; LedNum:Integer):LongBool;`
Function: It is to setup the the controller's serial communication address code. It has to call SCL_ComInitial before using this function; if using RS422/RS485 cable bus through a single serial port to communicate with several controllers, then it can call this function to give the controllers' address code. This way is to avoid repeated calling SCL_Close and SCL_ComInitial to repeatedly initialize the serial port device.
Entrance: LedNum: controller's number
Exit: If the corresponding number of equipment is initialized, then return to TRUE(NON-0), otherwise return to FALSE(0)

§4.5 Change the Target type SCL_TargetSCL2008

C Language: `BOOL SCL_TargetSCL2008(WORD nDevID,BOOL bSCL2008);`
Delphi Language: `Function SCL_TargetSCL2008(nDevID:Word; bSCL2008:LongBool):LongBool;`
Function: It is to switch the controller's type. When one equipment communicates with both SCL2008 and SuperComm, this function will advise DLL to communicate or pack the data packet according to the corresponding controller type.
Entrance: bSCL2008 : TRUE indicates the equipment corresponding to nDevID will communicate with SCL2008 controller. FALSE indicates it will communicate with SuperComm controller.
Exit: If the corresponding number of the equipment is successfully initialized, then return to TRUE(NON-0), otherwise return to FALSE(0)

§4.6 Communiacion close SCL_Close

C Language: `BOOL SCL_Close(WORD nDevID);`
Delphi Language: `Function SCL_ComClose(nDevID):LongBool;`
Function: it is to close communicatin, and release the corresponding hardwae resource.
Entrance: NO
Exit: Normally it will always return to TRUE(NON-0)

Chapter 5 Functions for file management and data sending/recieving

IN this chapter or the chapter afterwards, 'Driver's Number' parameter: 0 indicates Disc A (FLASH), 1 indicates Disc B (SD card), 2 indicates Disc C (RAM)

§5.1 Drivers' Format SCL_FormatDisk

C Language: `BOOL SCL_FormatDisk(WORD nDevID,int Drv);`
Delphi Language: `Function SCL_FormatDisk(nDevID:Word; Drv:integer):LongBool;`
Function: This function is to format the controller's drivers, it can quickly delete all the files in one of the controller's drivers
Entrance: Drv: Driver's number. It doesn't support to format SD card, if the SD care need to format, it has to use a PC with a SD card reader to do that, and the format has to be FAT16.
Exit: Format successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§5.2 Fetch driver's free space SCL_FreeSpace

C Language: `int SCL_FreeSpace(WORD nDevID,int Drv);`
Delphi Language: `Function SCL_FreeSpace(nDevID:Word; Drv:Integer):Integer;`
Function: release free space of controller's driver
Entrance: Drv: Driver's number;
Exit: the dirver's free space is counted in unit: byte, -1 indicates failure in releasing the free space

§5.3 Fetch the number of directory items SCL_DirItemCount

C Language: `int SCL_DirItemCount(WORD nDevID,int drv,char *Path);`
Delphi Language: `Function SCL_DirItemCount(nDevID:Word; drv:Integer; Path:PChar): Integer;`
Function: It is to download the directories' items of a directory or a subdirectory to file buffer, respond with the number of the files or subdirectories.
Entrance: drv: Driver's number
Path: Subdirectory name, blanck string indicates the file under directory and the number of subdirectories.
Exit: the number of files and subdirectories, -1 indicates failure in getting the items number.

§5.4 Fetch file directory SCL_GetDirItem

C Language: `BOOL SCL_GetDirItem(WORD nDevID,int ItemCount,char* Buff);`
Delphi Language: `Function SCL_GetDirItem(nDevID:Word; ItemCount:Integer; Buff:Pointer): LongBool;`

Function: It is to fetch the file and subdirectory items of the path corresponding to SCL_DirItemCount. It has to call this function before any other net manipulations. Normally it needs first to call SCL_DirItemCount to fetch the directory items, then apply for buffer based on the fetched directory items, then immediately call SCL_GetDirItem to fetch directory and save it into the buffer.

Entrance:ItemCount: items number, normally it is the returned value of SCL_DirItemCount function; Buff: receive the returned file and the space pointer of the subdirectory name, each item has 32 bytes which should meet FAT16 definition, Buff's space should be or greater than ItemCount*32 bytes. (Regarding to FAT16 format, please refer to DLL's header file)

Exit: successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§5.5 Send files SCL_SendFile

C Language: `BOOL SCL_SendFile(WORD nDevID,int drv,char *Path,char *LocalFileName);`

Delphi Language: `Function SCL_SendFile(nDevID: Word; drv: Integer; Path, LocalFileName: Pchar): LongBool;`

Function: This function is to send the local files to controller's drv Driver under Path directory, the file name should be corresponding to file name in PC.

Entrance: drv: Driver's number
Path: Controller's subdirectory, if it is blank, then it indicates to send to directory; if Path is not blank, and the subdirectory does not exist, then the function returns to FALSE.

LocalFileName: Local file name with its path, note that local file name has to meet format 8.3, namely it cannot be a long file name.

Exit: if sending successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§5.6 Receive file SCL_ReceiveFile

C Language: `int SCL_ReceiveFile(WORD nDevID,int drv,char* RemoteName,char* LocalName);`

Delphi Language: `Function SCL_ReceiveFile(nDevID: Word; drv:integer; RemoteName, LocalName: Pchar):Integer;`

Function: it is to retrieve the files from Controller to local driver

Entrance: drv: driver's number
RemoteName: File name with path in 8.3 format in the controller, for example, "P02\11.xmp" (files under subdirectory P02), or "11.xmp" (file under root directory);
LocalName: Local file name with its path

Exit: if successful, return to file length, otherwise return to -1

§5.7 Delete files SCL_RemoveFile

C Language: `BOOL SCL_RemoveFile(WORD nDevID,int drv,char* filename);`
Delphi Language: `Function SCL_RemoveFile(nDevID:Word; drv:Integer;filename:Pchar):LongBool;`
Function: it is delete some file on the controller
Entrance: drv: driver's number
 filename: File name with path in 8.3 format in the controller, for example "P02\11.xmp" (file name under subdirectory P02), or "11.xmp" (file under directory)
Exit: If successfule , return to TRUE(Non-0) , otherwise return to FALSE(0)

§5.8 Create subdirectory SCL_MD

C Language: `BOOL SCL_MD (WORD nDevID,int drv,char* path);`
Delphi Language: `Function SCL_MD(nDevID:Word; drv:Integer; path:Pchar):LongBool;`
Function: Create a subdirectory under controller's directory
Entrance: drv: Driver's number
 path: "FON" or subdirectory file name of "Pxx" format, x indicates the digits from 0-9.
Exit if successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§5.9 Delete subdirectory SCL_RD

C Language: `BOOL SCL_RD (WORD nDevID, int drv,char* path);`
Delphi Language: `Function SCL_RD(nDevID:Word; drv:Integer; path:Pchar):LongBool;`
Function: Delete one subdirectory under the root directory in one of controller's drivers, it can only be deleted when it is empty.
Entrance: drv: Driver's number
 path: "FON" or subdirectory file name of "Pxx" format, x indicates the digits from 0-9
Exit: If successfule, then return to TRUE(Non-0), otherwise return to FALSE(0)

§5.10 Immediately display words SCL_ShowString

C Language: `BOOL SCL_ShowString(WORD nDevID,short *info, char *str);`
Delphi Language: `Function SCL_ShowString(nDevID:Word; Var info:WORD, str:PChar):LongBool;`
Function: It is to immediately display the character string, it needs the controller preload the fonts, and gives the fonts installation requirement in Config.LY file. Please refer to the chapter "about the font for more details.

Entrance: info: parameters of word displaying, please refer to the section : word displaying info;

str: The character string for displaying, including extension info, refer to section "extension code". Str length cannot exceed 1000 characters.

Exit: If successful, return to TRUE(Non-0), otherwise , return to FALSE(0)

Chapter 6 Controlling Functions

§6.1 Reset controller SCL_Reset

C Language: `BOOL SCL_Reset(WORD nDevID);`
Delphi language: `Function SCL_Reset(nDevID:Word):LongBool;`
Function: reset controller
Entrance: No
Exit: if successful, then return to TRUE(Non-0), otherwise return to FALSE(0)

§6.2 Play new schedule SCL_Replay

C Language: `BOOL SCL_Replay(WORD nDevID,int drv, int Index);`
Delphi Language: `Function SCL_Replay(nDevID:Word; drv:Integer; Index:Integer):LongBool.`
Function: when all programs or images are transferred to controller from PC, then it need call this function for the controller to excute the new schedule.
Entrance: drv: driver number
Index: Schedule number, 0-99. For example, `SCL_Replay(nDevID,2,8)` can order the controller to play the PLAYLIST.LY under C:\P08; `SCL_Replay(nDevID,1,3)` will order the controller to play the PLAYLIST.LY under B:\P03.
Exit: If the controller plays new schedule, then return to TRUE(Non-0), otherwise return to FALSE(0)

§6.3 Check controller clock time SCL_SetTimer

C Language: `BOOL SCL_SetTimer(WORD nDevID);`
Delphi Language: `Function SCL_SetTimer(nDevID:Word):LongBool;`
Function: automatically fetch PC's time to check/adjust controller's time.
Entrance: No
Exit: If successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§6.4 Setup automatic switching off/on time SCL_SetOnOffTime

C Language: `BOOL SCL_SetOnOffTime(WORD nDevID,unsigned short OnTime,unsigned short OffTime);`
Delphi Language: `Function SCL_SetOnOffTime(nDevID:Word; OnTime,OffTime:Word):LongBool;`
Function: It is to setup the automatic switching on/off time. When It need force the LED screen to switch off, it just need setup both the swithing on time/off time as 0; if need force the LED screen to switch on, then it

can setup both the switching on/off time as 23:59.

Entrance: OnTime:Switch on time
OffTime: switch off time。
OnTime and OffTime's format is hour*100+minute. For example, 9:28 is written as 928.
Exit: if successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§6.5 Setup LED screen's brightness SCL_SetBright

C Language: `BOOL SCL_SetBright(WORD nDevID,unsigned short brightness);`
Delphi Language: `Function SCL_SetBright(nDevID:Word; brightness:Word):LongBool;`
Function: setup LED screen's brightness。
Entrance: bright: Brightness value, which is 0-31, and the value '31' indicates to automatically adjust the brightness according to the environment's brightness.
Exit: if successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§6.6 Setup temperature sampling SCL_SetTempOffset

C Language: `BOOL SCL_SetTempOffset(WORD nDevID,short offset);`
Delphi Language: `Function SCL_SetTempOffset(nDevID:Word; offset:smallint):LongBool;`
Function: It is to amend the sampling data from the temperature sensor DS18B20. DS18B20 sensor's tolerance is $\pm 2^{\circ}\text{C}$, this amending can have the LED display more accurate temperature.
Entrance: Offset is the amending for the sampling data from temperature sensor, its range is from -7 to +7.
Exit: if successful, then return to TRUE(Non-0), otherwise return to FALSE(0)

§6.7 Setup power mode SCL_SetPowerMode

C Language: `BOOL SCL_SetPowerMode(WORD nDevID,int PowerMode);`
Delphi Language: `Function SCL_SetPowerMode(nDevID:Word; PowerMode:Integer);`
Function: it is to setup the power mode for LED screen, enable the remote control on the power source of LED screen.
Entrance: PowerMode: new power mode, 0: switch off power source, 1: switch on power source, 2: automatic switch on/off according to SCL_SetOnOffTime
Exit: if the setting is successful, return to TRUE(Non-0), otherwise return to FALSE(0)

§6.8 Read the running info SCL_GetRunTimeInfo

C Language: `BOOL SCL_GetRunTimeInfo(WORD nDevID, BYTE *Buff512Bytes);`

Delphi Language: `Function SCL_GetRunTimeInfo(nDevID:Word; Buff512Bytes:Byte):LongBool;`

Function: it is to fetch the controller's current running info.

Entrance: Buff512Bytes: save running info into a buffer which is at least 512 bytes.

Exit: If successful, return to TRUE (Non-0), Buff512Bytes buffer is effective; otherwise, return to FALSE(0)

Regarding to the data format in Buff512Bytes, please refer to the DLL header file in C language and Delphi language.

§6.9 Reading the playing info SCL_GetPlayInfo

C Language: `BOOL SCL_GetPlayInfo(WORD nDevID, BYTE *PlayInfo);`

Delphi Language: `Function SCL_GetPlayInfo(nDevID:Word; PlayInfo:BYTE):LongBool;`

Function: it is to read the controller's current playing info.

Entrance: PlayInfo: 7 bytes of storage for the return data.

Exit: if successful, then return to TRUE (Non-0), PlayInfo data is effective; otherwise return to FALSE(0).

PlayInfo[0]: the driver number of the current playing schedule;

PlayInfo[1]: the index number of the current playing schedule from 00-99, it is numbered from 0;

PlayInfo[2]: the current playing program item of PLAYLIST.LY, it is numbered from 0;

PlayInfo[3]: the index of current playing program item in area#1, it is numbered from 0

PlayInfo[4]: the index of current playing program item in area#2, it is numbered from 0

PlayInfo[5]: the index of current playing program item in area#3, it is numbered from 0

PlayInfo[6]: the index of current playing program item in area#4, it is numbered from 0

§6.10 Control extended switch SCL_SetExtSW

C Language: `BOOL SCL_SetExtSW(WORD nDevID,WORD OnOff);`

Delphi Language: `Function SCL_SetExtSW(nDevID:Word; OnOff:Word):LongBool;`

Function: It is to control the SW2's state for Supercom full version (SCL2008: SW1)

Entrance: OnOff: 0 switch off, 1 switch on

Exit: if successful, then return to TRUE(non-0), otherwise return to FALSE(0).

When it is successfully setup, and when OnOff is 0, then controller's LSW2 indicator will turn off, SW2(SCL2008:SW1) output will have no voltage fall; if OnOff is 1, then controller's LSW2 indicator will turn on, SW2(SCL2008:SW1) output will have a signal with 4V, 20mA, which is used to control the solid state relay.

Chapter 7 Automatically organize and arrange data sending/receiving

In some cases, programmers just want DLL to provide functions to enable the encapsulation of communication protocol, and then have themselves to initialize the communication equipments and manage the data sending/receiving. This chapter will introduce how to realize these requirements.

It is to call SCL_InitForPackage function to initialize virtual net or serial equipment, and have DLL just encapsulate the data packet, not really use the communication equipment to send/receive data.

Under net mode, PC and controller exchange their data by UDP protocol, which occupy 2 adjacent terminals. After getting the data packet, if PC send data to controller through terminal P, then the target (controller) terminal should be P+1; meanwhile, controller also sends data to PC through terminal P, PC need use terminal P+1 to receive response.

In the file management functions, data sending/receiving functions and control functions, except for the 3 functions SCL_GetDirItem, SCL_SendFile and SCL_ReceiveFile need exchange multiple data packets, the other functions just need exchange one data packet.

This chapter will introduce how to encapsulate data packet, and check if the controller's responses are correct or not after the data packets are sent to the controller.

The function calling steps as below:

1. Call SCL_InitForPackage, initialize a virtual net or serial communication equipment.
2. Based on the request, call SCL_SetRemoteIP or SCL_SetLEDNum and SCL_TargetSCL2008 functions
3. Call file sending/receiving functions and control functions
4. Call SCL_GetPackage, fetch the encapsulated data packet of the communication protocol.
5. Programmer organizes and arranges the data sending /receiving.
6. Call SCL_CheckAnswer, check if the controller's response is correct or not, if it is correct, it can use this response, or continue to send other data to the controller.

As the 3 functions SCL_GetDirItem, SCL_SendFile and SCL_ReceiveFile need exchange multiple data packets with controller, so these 3 functions can not be directly called when the programmer organizes and arranges the data receiving /sending. And this chapter will also introduce how to enable these 3 functions.

§7.1 Initialize virtual equipment SCL_InitForPackage

C Language: `int SCL_InitForPackage(WORD nDevID, BOOL bNet, BOOL bSCL2008);`

Delphi Language: `Function SCL_InitForPackage(WORD nDevID; bNet, bSCL2008: LongBool): LongBool;`

Function: It is to initialize the virtual communication equipment for encapsulated data packet

Entrance: nDevID: it is a 16-bit equipment number given by the programmer. If it was successfully initialized, it will call communication and fetch communication data packet, it has to use the same equipment

number while checking the response function.

- bNet:** It indicates if it is under net communication mode or serial communication mode; under different communication mode, the way to encapsulate the data packet is different. TRUE: net mode; FALSE: serial port mode.
- bSCL2008:** It indicates if it is to encapsulate the data packet for SCL2008 controller or not. FALSE: SuperComm; TRUE: SCL2008
- Exit:** If it is successfully initialized, return to TRUE (Non-0), otherwise return to FALSE (0) (it is probably because the 256 communication equipments managed by DLL have been occupied, or it uses the same equipment number as other's).

After being successfully initialized, it can call SCL_TargetSCL2008 to change the controller type, if it is serial communication, it can also call SCL__SetLEDNum to change controller's address code (controller's address code is encapsulated in the serial communication data packet). The default address code is 0 (SCL2008's default address code is 100)

§7.2 Fetch communication data packet SCL_GetPackage

- C Language:** `int SCL_GetPackage(WORD nDevID, BYTE *Data, int *AnswerCount);`
- Delphi Language:** `Function SCL_GetNetPackage(nDevID:Word; Var Data:Byte; Var AnswerCount: Integer):Integer;`
- Function:** Fetch communication data packet
- Entrance:** Data: the memory pointing at stored data packet. Normally this space need be defined as 1100 bytes (For SuperComm/ SCL2008, each data packet will have less than 1024 usable bytes; with protocol header, the Max will be 1060 bytes) after calling this function, DLL will copy the encapsulated data packet to this space.
- AnswerCount:** The saved data length of the returned and expected response data. After calling this function, DLL will give the adequate bytes to the response packet according to the protocol, it is for programmers to deal with the receiving process.
- Exit :** Data bytes in the buffer.

§7.3 Check the response SCL_CheckAnswer

- C Language:** `BOOL SCL_CheckAnswer(WORD nDevID, int *AnswerCount, BYTE *AnswerData);`
- Delphi Language:** `Function SCL_CheckAnswer(nDevID:Word; Var AnswerCount:Integer; Var AnswerData: BYTE):LongBool;`
- Function:** Check if the response from controller is correct or not.
- Entrance:** AnswerCount: the bytes number of received response ;
AnswerData: received response data
- Exit:** if response is correct, then return to TRUE(Non-0), if it is incorrect, then return to FALSE(0)

If it is return to TRUE, then AnswerCount will amend as a validated data length with the communication protocol stripping off.

AnswerData will also amend as a validated data length with the communication protocol stripping off.

The below functions are used to enable SCL_GetDirItem, SCL_SendFile and SCL_ReceiveFile

§7.4 Shutoff display SCL_LedShow

C language: `BOOL SCL_LedShow(WORD nDevID,BOOL OnOff);`
Delphi Language: `Function SCL_LedShow(nDevID:Word; OnOff:LongBool):LongBool;`
Function: to Shutoff LED Screen, and stop controller's playing process. Controller runs a time sharing operating system, when there is a large amount of data in transfer, it need stop playing process, make CUP fully work on data reciving/sending for the satisfied transferring speed. When in serial communication, it doesn't have to call this function.
Entrance: OnOff: TRUE indicates switching on, FALSE indicates switching off.
Exit: if it is successful, then return to TRUE(Non-0), fail, then return to FALSE(0)

§7.5 Send data to controller's file buffer SCL_SendData

C Language: `BOOL SCL_SendData(WORD nDevID,int Offset,int SendBytes, BYTE *Buff);`
Delphi Language: `Function SCL_SendData(nDevID:Word;Offset:integer; SendBytes: Integer; Var Buff: BYTE):LongBool;`
Function: It is to send a data packet to controller's file buffer, which is 2M bytes, repeatedly to call this function can separately send a file to the buffer
Entrance: Offset: the address offset of the data written in the buffer;
SendBytes: Bytes of the data;
Buff: the buffer for storage of data;
Exit: if successfule, then return to TRUE(Non-0), Fail, then return to FALSE(0)

§7.6 Retrieve data from controller's buffer SCL_ReceiveData

C Language: `BOOL SCL_ReceiveData(WORD nDevID,int Offset,int ReceiveBytes, BYTE *Buff);`
Delphi Language: `Function SCL_ReceiveData(nDevID:Word; Offset:integer; ReceiveBytes Integer; Var Buff: BYTE):LongBool;`
Function: It is to retrieve one group of data from controller's file buffer. After one file is read in the file buffer, this function can separately retrieve the data back to PC.

Entrance: Offset: the address offset of the retrieved data in the controller's buffer;

ReceiveBytes: the request bytes of the retrieved data;

Buff: the buffer for storage of data;

Exit: if successful, then return to TRUE(Non-0), fail, then return to FALSE(0)

§7.7 save the data from file buffer as a disk file SCL_SaveFile

C Language: `BOOL SCL_SaveFile(WORD nDevID,int DrvNo,char *Name, int Len,int Date,int Time);`

Delphi Language: `Function SCL_SaveFile(nDevID:Word;DrvNo:Integer; Name:PChar; Len,Date,Time: Integer):LongBool;`

Function: It is to save the data from controller's buffer as a Disk file.

Entrance: DrvNo: driver number;

Name: the file name with its path in 8.3 format;

Len: the data length (unit: byte) of the saved file;

Date, Time: the time when the file is created (it can be obtained from SCL_GetFileDosDateTime)

Date:

<u>d15 d14 d13 d12 d11 d10 d9</u>	<u>d8 d7 d6 d5</u>	<u>d4 d3 d2 d1 d0</u>
year-1980	Month	date

Time:

<u>d15 d14 d13 d12 d11 d10 d9 d8 d7 d6 d5</u>	<u>d4 d3 d2 d1 d0</u>
hour	minute
	second/10

Exit: If the file is successfully saved, then return to TRUE(Non-0); fail, then return to FALSE(0)

§7.8 Save the Disc file in controller's buffer SCL_LoadFile

C Language: `BOOL SCL_LoadFile(WORD nDevID,int DrvNo,char *Name);`

Delphi Language: `Function SCL_LoadFile(nDevID:Word;DrvNo:Integer; Name:Pchar):LongBool;`

Function: it is to download one disk file to the controller's buffer.

Entrance: DrvNo: driver's number

Name: file name with its path in 8.3 format

Exit: if successful, then return to TRUE(Non-0), fail, then return to FALSE(0)

§7.9 Fetch local file to create DOS date and time SCL_GetFileDosDateTime

C Language: `BOOL SCL_GetFileDosDateTime(char *FileName,int *Date,int *Time);`

Delphi Language: `Function SCL_GetFileDosDateTime(FileName:Pchar;Var Date,Time: Integer) :LongBool;`

Function: It is to fetch local file to create DOS date and time.

Entrance: FileName: Filename with its path

Exit: If successfully getting the date and time data, then return to TRUE (Non-0); otherwise, return to FALSE(0)

Date:

d15 d14 d13 d12 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0
Year-1980 Month date

Time:

d15 d14 d13 d12 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0
hour minute Second/10

The below introduces the method to enable SCL_GetDirItem, SCL_SendFile and SCL_ReceiveFile. IF without being especially mentioned, all the functions are followed with SCL_GetPackage function, after exchanging the data with controller, it still need get the response in SCL_CheckAnswer function.

§7.10 SCL_GetDirItem's enablement

- 1、 If using net communication, it need call SCL_LedShow(FALSE)
- 2、 To call SCL_DirItemCount to read the directory in file buffer, to get the number of directory items, the number of data bytes in the file buffer= the number of directory items x 32
- 3、 To call SCL_ReceiveData to retrieve the data from some position in the file buffer back to PC, each time the bytes retrieved back from the buffer cannot exceed 1024 bytes.
- 4、 Revise SCL_ReceiveData function's offset parameter, repeat step 3, until all the data (the number of directory items x 32 bytes) is read back in the PC.
- 5、 If using net communication, then call SCL_LedShow(TRUE)。

§7.11 SCL_SendFile's enablement

- 1、 If using net communication, it need call SCL_LedShow(FALSE)
- 2、 To call SCL_GetFileDosDateTime get the the date and time when the file is created (it doesn't have to call SCL_GetPackage and SCL_CheckAnswer);
- 3、 To call SCL_SendData to send the data from a certain position of the file to one location in controller's buffer, each time the bytes sent cannot exceed 1024 bytes;
- 4、 Revise SCL_SendData's Offset parameter, repeat step 2, until all the data is sent to controller's buffer.
- 5、 To call SCL_SaveFile to save the date and time data which were obtained in the step 2 as a disk file.
- 6、 If it is net communication, then call SCL_LedShow(TRUE)

§7.12 SCL_ReceiveFile enablement

- 1、 If under net communication mode, then call SCL_LedShow(FALSE)
- 2、 To call SCL_LoadFile read the disk file in controller's file buffer, and meanwhile to get the file size in byte.
- 3、 To call SCL_ReceiveData separately retrieve the data from file buffer to PC and write in PC's disk file, each time the bytes retrieved cannot exceed 1024 bytes.
- 4、 Revise SCL_ReceiveData's Offset parameter, repeat Step 3, until all the data are retrieved back to PC.

- 5、 Shutoff PC's disk file;
- 6、 If under communication mode, then call SCL_LedShow(TRUE)。

§7.13 Broadcast transfer

If it was needed to send a large amount of the same data to the same type of controllers (and LED screen should be the same size), then it can use the broadcast. Under the broadcast mode, the controllers under LAN's same segment or RS422 /RS485 net will all receive the data. This way doesn't support to separately send data. In the LAN, only the controllers under the same segment can use broadcast.

The broadcast definition is just for the data sending, not including response. As the data sent is much more than the response data packet, so it can reduce the amount to send under the broadcast mode.

If setup the target IP's last segment as 255 (for example, 10.1.1.255) or setup controller's address code as 255, then DLL will send the data to all of the controllers in the LAN (or in the RS422/ RS485 net). Afterwards, programmer need call SCL_GetLastResult to check if each controller has correctly received the data.

C Language: `BOOL SCL_GetLastResult(WORD nDevID);`
 Delphi Language: `Function SCL_GetLastResult(nDevID:Word):LongBool;`

If last operation is correct, then SCL_GetLastResult return to TRUE(Non-0), otherwise return to FALSE(0)。

For example, supposing the LAN has three LED screens "10.1.1.127", "10.1.1.128" and "10.1.1.129", now it needs broadcast to send SCL_Replay(nDevID,1,3), the calling procedure under net communication mode as below:

- 1、 SCL_NetInitial;
- 2、 SCL_SetRemoteIP, setup target IP as broadcast address
- 3、 SCL_Replay, (or other data sending command, for example: SCL_SendData)
- 4、 SCL_SetRemoteIP, setup the target as "10.1.1.127"; call SCL_GetLastResult to check the result
- 5、 SCL_SetRemoteIP, setup the target as "10.1.1.128"; call SCL_GetLastResult to check the result
- 6、 SCL_SetRemoteIP, setup the target as "10.1.1.129"; call SCL_GetLastResult to check the result
- 7、 SCL_Close;

If under serial communication mode, in the above 4, 5, 6 steps, SCL_SetRemoteIP need be changed to SCL_SetLEDNum.

When calling SCL_GetLastResult, if the return value is FALSE (0), which means the controller hasn't receive the correct data, in this situation, it is necessary to resend the data to this individual controller.

Supposing when calling SCL_GetLastResult command to #128 LED screen, the return value is FALSE, the operation to resend the data is: Between step 5 and step 6, insert SCL_Replay function (or other data sending command), as before the function SCL_GetLastResult, it has already correctly pointed the target to LED screen #128, so from the returned value of SCL_Replay, it can directly tell if #128 LED screen has correctly received the resent data.

Chapter 8 File format convert

§8.1 Image file convert to XMP file SCL_PictToXMPFile

C Language: `BOOL SCL_PictToXMPFile(int colortype,int width,int height,BOOL bstretch, char* Pictfilename,char* XMPfilename);`

Delphi Language: `Function SCL_PictToXMPFile(colortype:integer;width:integer;height:integer; bstretch:LongBool; Pictfilename:Pchar; XMPfilename:Pchar): LongBool;`

Function: Convert BMP or JPEG file to an including image XMP file, don't support other format of image file.

Entrance: colortype: Color types , 0 indicates double color 256 grayscales(for SuperComm), 1 indicates 3 colors 64grayscales(for SuperComm) , 2indicates double colors without grayscale(for SCL2008);

width: Target image width, unit: pixel;

height: Target image height, unit: pixle;

bstretch: automatic adjust to fit;

Pictfilename: BMP or JPEG image filename with its path ;

XMPfilename: XMP file name with its path

Exit: If successful, return to TRUE (Non-0); otherwise, return to FALSE (0).
False is probably because color types are illeagle, or displaying range exceeds the controller's Max ontrolling range, or fail in reading in BMP, JPEG image, or fail in outputting XMP file.

§8.2 XMP file size SCL_GetMaxFileSize

C Language: `int SCL_GetMaxFileSize(int ScreenCount, BOOL bSmallest, BOOL bSCL2008);`

Delphi Language: `Function SCL_GetMaxFileSize(ScreenCount:Integer;bSmallest:LongBool;bSCL2008: LongBool):Integer;`

Function: It is to calculate a XMP file's Max bytes quantity based on the the number of individual program's areas and based on wheather the playing area is the smallest one or not. As the controller will automatically arrange the file preread buffer according to the the number of the current program's areas (it will read next file into the memory while it is playing), so the XMP file size cannot exceed buffer's size.

This function is to get a XMP file's Max length according to controller's buffer allocation algorithm.

Entrance: ScreenCount: program's areas number, 1-4;

bSmallest: the current playing area is the smallest one or not;
bSCI2008: Controller type, FALSE:SuperComm, TRUE: SCL2008
Exit: XMP file size, unit: byte

§8.3 Combine XMP files SCL_AddXMPToXMP

C Language: `int SCL_AddXMPToXMP(char* InFilename, char* OutFilename,int
MaxSize);`

Delphi Language: `Function
SCL_AddXMPToXMP(InFilename:Pchar;OutFilename:Pchar;
MaxSize:integer): Integer;`

Function: to combine 2 XMP image file into a XMP image file.

Entrance: InFilename: XMP file name (with path) including one or several images.

OutFilename: XMP file name (with path) including one or more images, if OutFilename is not existing, then excute the copying operation from InFilename to OutFilename

MaxSize: Max length of XMP file using SCL_GetMaxFileSize. To combine the 2 imgages, it needs both images of InFilename and OutFilename have the same image color types, basic width and height.

Exit: if successful, return to 0, otherwise return the wrong code:

- 1: InFilename is not existing
- 2: InFilename open failure
- 3: OutFilename open failure
- 4: 255 the images number exceed 255 after combination
- 5: the versions of the 2 files are different (each pixel is different color)
- 6: the width and height of the 2 images are different.
- 7: InFilename file, the size of InFilename file size doesn't match the defined images' type and size (file is damaged)
- 8: OutFilename file, the size of OutFilename doesn't match the defined image type and size(file damaged)
- 9: The combined file exceeds the limit of the Max file length.

Chapter 9 Font

§9.1 Download font

SuperComm/SCL2008 controllers support multiple fonts.

In the chapter 2, we already introduced Config.LY font installation.

The controllers can provide Max 8M for storage of fonts, it can contain Max 8 different fonts .

The developing packet provides a program MakeFontLib.exe to automatically create Chinese and western characters. Programmer can use it to create the fonts they want, after the programs executed, there will show CharCount, width and height info, which need be written in Config.LY file when the font is installed.

MakeFontLib can create a font which can rotate 90 degree. It can be used for displaying words in the vertical direction, while the words are input in vertical direction, the effect will be as below



§9.2 Words message

The first parameter of ShowString is pointing at “[Words message](#)” pointer. “word message” is a record type, the format in C language as below:

```
struct TextOutInfo
{
    WORD    Left;
    WORD    Top;
    WORD    Width;
    WORD    Height;
    LONG    Color;
    WORD    ASCFont;
    WORD    HZFont;
    WORD    XPos;
    WORD    YPos;
};
```

```
typedef struct TextOutInfo    TEXTINFO;
```

The format in C language:

TextOutInfo = Packed Record

```
Left    : Word;
Top     : Word;
Width   : Word;
Height  : Word;
Color   : Cardinal;
ASCFont : Word;
HZFont  : Word;
Xpos    : Word;
Ypos    : Word;
```

End;

In the above:

Left, Top, width, Height are respectively the area's topleft coordinate and the area's width and height, Left and Top coordinate is the absolute coordinate based on the Max controlling range. The controller will automatically typeset and output. namely, when the display reaches the most right side of the range, and the remaining space is less than the width of the character to input, it will automatically skip to the most left side of a new line; when the display reaches the bottom of the range, and the remaining space is less than the height of the character to input, then the controller will stop displaying in the current range. This area has to be one part of the schedule's areas, it cannot display across areas.

Color is the words' color, which meet the standard 24-bit color definition, namely, in a 32-bit long integer, D23-D16 are blue, D15-D8 are green, D7-D0 are red;

ASCFont is the chosen font index while outputting western characters, this index is the sequence of installation of the fonts in the Config.LY file, they are numbered from 1.

HZFont is the chosen font index while outputting Chinese characters or other local language characters, this index is the sequence of installation of the fonts in the Config.IY file, they are numbered from 1

XPos and YPos is the starting coordinate of the displaying; this coordinate is a relative coordinate to the 'Left, Top' defined area, whose topleft corner is the origin.

When the controller is powered on, it will install the fonts to memory according to Config.LY designated sequence. ASCII code characters and Chinese characters (or other local language fonts) are consolidated to create index.

Font assignation rule: Before the controller outputs fonts, it will use the smallest ASCII fonts of the installed index as the default ASCII font; and use the smallest 'C' type font of the installed index as the default 'C' font. While assigning one type of font as an index 'x', and if the corresponding font is not this type of font, then controller will neglect this command.

For example, #1 and #2 fonts are type A, #3 and #4 are type C, and ASCFont designate to use #3 font, HZFont designate to use #2 font, then the controller will use #1 font as the default font of Type A, use #3 font as the default font of Type C.

§9.3 Display extension code

The second parameter of ShowString function is the pointer pointing at the characters string. In the character string, except for the common ASCII code and standard Chinese code, SuperComm/ SCL2008 also provide some extension codes, these codes are lead by "" (the key beside '1' key, the ASCII code is 0x60) or ESC(invisible, ASCII code 0x1B), followed with

some letter:

- '': directly output '', for example: ''';
- 'A': Followed with a digit from '1' to '8', corresponding to ASCFont in "word message", it indicates all the ASCII new fonts after this definition. For example: 'A3' indicates to use the 3rd font; this assignation conforms the "Font assignation rule" in the previous section.
- 'B': It is followed with 6 hexadecimal characters, which indicates the background colors of the message. For example: 'B203fff' indicates the background colors are blue 0x20, green 0x3f, red 0xff.
- 'C': It is followed with 6 hexadecimal characters, which indicates the foreground colors, they are corresponding to the colors in "word message"
- 'D': It is followed with a decimal digit, which indicates the words outputting direction, '0': horizontal output; '1': vertical direction
- 'H': It is followed with a digit character from '1' to '8', corresponding to HZFont in "word message", which indicates to use the new font for all Chinese character output. For example: 'H1' indicates to use the first font; this assignation conform "font assignation rule" in the previous section.
- 'M': It is followed with a digit character from '1' to '3', which indicates the words outputting mode. '1': coverage output, '2' superimposed output, '3': reverse output. **SCL2008 is not supportive to Superimposed output.**
- 'R': It is followed with 4 groups (total 12 bits) of decimal digits, they respectively indicate the one area's 'x' coordinate, 'y' coordinate, width and height. This command is to clear one sub area in the display. This area is a relative coordinate whose origin is topleft corner. For example: 'R008010096032' means to clear the content in the area from 8th pixel to 8+96th pixel in horizontal direction, and from 10th pixel to 10+32th pixel;
- 'W': It is followed with a digit character, which can be '0' or '1', they respectively indicate the fixed width and changing width output (SCL2008 is not supportive to changing width output). In some characters, some characters are very wide, and some are narrow, if they are to output based on the defined width, the characters may not look good, in this situation, it can use "W1" to let the controller have changing width output. Under changing width method, when output one character, the next output character will start from the second dot after the last dot of last character.
- 'X': It is followed with 3 decimal digits, which indicates the new output x coordinate of next character; it can add '+' or '-' before the 3 digits, it means to move the coordinate from the current place. This area is a relative coordinate based on the current output coordinate.
- 'Y': It is followed with 3 digits, which indicates the new output y coordinate of next character; it can add '+' or '-' before the 3 digits, for example, Y-001, which means to move coordinate from the current coordinate. This coordinate is a relative one based on the current output coordinate.
- 'Z': It is followed with 4 groups (12bit) of decimal digits, which respectively indicate

one area's x coordinate, y coordinate, width and height. 'Z' command is to redefine the output area (namely, refresh Left, Top, Width and Height of the message range to a new value). This coordinate is an absolute coordinate according to the controller's Max controlling range.

It can display some special effects to use the above extension code.

For example: supposing the first font's width and height both are 32, the second font's width and height both are 16, then character string "H1 will have special display effect, H2's first row of small size words 'x128'Y16



For another example, "X002Y002`C00FFFF 镶边汉字`X001Y003`C0000FF 镶边汉字 " will output as the below effect (SCL2008 is not supportive for the shadow mode)





深圳市齐普光电子有限公司

SHENZHEN CHIP OPTECH CO.,LTD.

Tel: +86-755-83419012 83893997 83894037 83419034 88350045

Fax: +86-755-82975893

Website: www.chipshow.com Email: manager@chipshow.com

Msn: led_signs@hotmail.com Skype: led_screen

Add: Yufeng Industrial Park, 82 Zone, Bao'an District, Shenzhen,
Guangdong, China