Prolin XUI Interface

V2.1.7

PAX Computer Technology（Shenzhen）Co., Ltd.

History of Revisions

Date	Version	Note	Author
2013-05-16	V1.0.0	Draft	Xie Lihong
2013-10-18	V2.0.0	Update the document	Huang Lei
2014-01-21	V2.0.1	1. Increase the description of structure XuiSignPoint and XuiSignData; 2. Add a new interface XuiSigBoardSignData().	Huang Lei
2014-03-25	V2.0.2	1. Modify the description of the structure XuiSignPoint; 3. Add three new interfaces XuiCreateCanvasEx(), XuiCanvasMoveToY(), XuiImgLoadFromMem().	Huang Lei
2015-04-28	V2.0.3	1. Added the support of bmp and mbmp in XuiImgSaveToFile(); 2. Modify the ending point of the signature data to 0xffff in XuiSigBoardSignData(); 3. Modify the parameter type of *maxlen* from unsigned to unsigned int in function XuiGetHzString().	Huang Lei
2015-09-02	V2.0.4	1. Added the support of combination key; 2. Added a function XuiBidiStrdup().	Huang Lei
2016-03-01	V2.0.5	1. Added the instruction in XuiCreateSignatureBoard(); 2. Added alpha_key and sharp_key to the structure XuiImeAttr and alpha_key to the structure XuiGetStrAttr; 3. Added XuiShowMode, XuiAnimationType,XuiGestureType,XuiGesture structures in 2.2 Macro Definition; 4. Added XuiCanvasAnimation() and XuiGetGesture().	Huang Lei

2016-03-28	V2.0.6	1. Added XuiSetGestureRect() and XuiClearGesture(); 2. Added XUI_GESTURE_CLICKDOWN and XUI_GESTURE_CLICKUP in XuiGestureType of 2.2 macro definition chapter; 3. Added down_x, down_y, cur_x and cur_y four members in XuiGesture of 2.4 structure chapter.	Huang Lei
2016-04-15	V2.0.7	1. Added the soft key definition to 2.1 definition of key values table; 2. Added two new functions which are XuiImgCompose() and XuiShowSoftKeyboard(); 3. The title of this document changed from “XUI Programming Guide” to “Prolin XUI Interface”.	Huang Lei & Ye Sining
2017-03-01	V2.0.8	1. Updated the instruction of XuiRootCanvas(); 2. Added two new functions which are XuiTextWidthEx() and XuiImgResize(); 3. In section 2.1 definition of key value table, added the key values for hiding softkeyboard and independent key for camera; 4. Modified the default value of *input* device node in XuiOpen().	Huang Lei& Ye Sining
2017-09-05	V2.0.9	Added new function XuiImgToFrameBuffer().	Li Xin
2017-12-14	V2.1.0	1. Added functions XuiCreateCamera(), XuiCameraCapture(), XuiCreateScaner(), XuiScanerDecode() and XuiRgbaToImg(); 2. Added related macro definition of camera; 3. Added new definition for parameter *show* in	Li Xin

		XuiShowSoftKeyboard(); 4. Added new definition for parameter *mode* in XuiGetString().	
2018-04-10	V2.1.1	Added new function XuiCanvasDrawTextEx().	Li Xin
2018-05-14	V2.1.2	Added new function XuiFrameBufferToImg().	Li Xin
2018-11-02	V2.1.3	1. Added a new macor *XUI_CAMERA_IMG_1024_480* in table XuiCameraResolution; 2. Added a new macor *XUI_PREVIEW_ZOOM_25* in table XuiPreviewZoom; 3. Updated the description of parameter *resolution in* XuiCreateCamera().	Li Xin
2019-01-16	V2.1.4	Updated the instruction of XuiOpen().	Li Xin
2019-03-13	V2.1.5	Added new functions XuiImgToGray8(), XuiImgToBgr24() and XuiCameraSetSta().	Li Xin
2019-06-14	V2.1.6	Added support for jepg.	Li Xin
2019-11-14	V2.1.7	Added section “XuiImgLoadFromBase64” and corrected some mistakes.	Li Xin

Table of Contents

Figure & Table List

1 Introduction

1.1 Purpose

In contrast to other GUIs, XUI is relatively easy to understand and use. It adopts imperative programming interfaces, and it is suitable for developing the wizard-style interface for customer-oriented terminals such as POS machine, handhold terminal and ATM etc.

XUI cannot implement a variety of special features as complicated as GUI, but in wizard-style interface, it is simpler and more efficient.

To put it simply, XUI programming is to draw, to write and to wait for keypress.

1.2 Function

The functions of XUI are listed as follows:

- Support black-and-white screen.
- Support monochrome font and gray font.
- Support touch screen.
- Support graphical display.
- Support multi-font display.

- Support bidirectional text display.
- Support translucent. (Alpha Channel)
- Support screenshot.
- Support outputting the screenshot to printer, which means displaying interface and printing interface are unified.
- Support multi-platform, including Linux Framebuffer, X11, SDL, Windows, Android, iOS, platform without operating system etc.
- Support screen rotation.

1.3 Feature

- Imperative programming interface.
- Screen keys and physical buttons are unified.

1.4 XUI Programming Logic

The interface is designed as below.

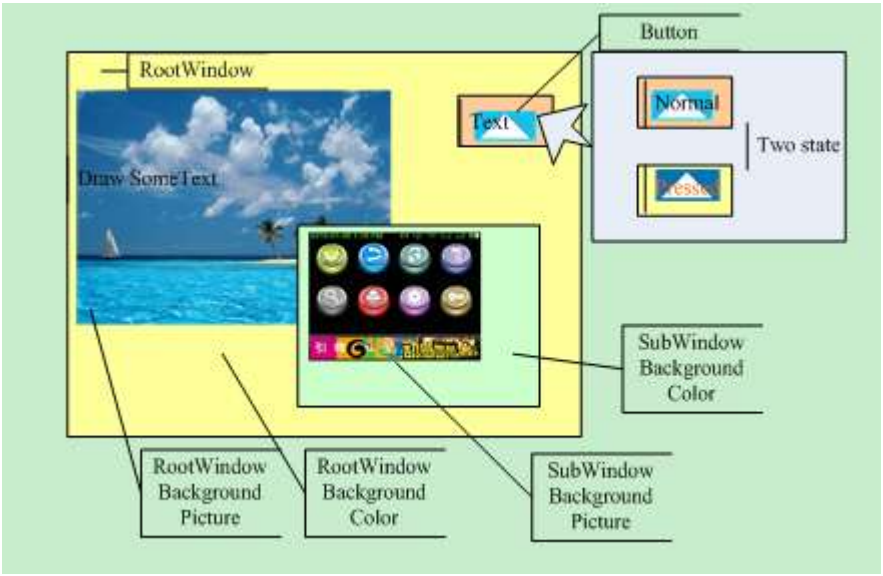

Figure 1.1 UI design plan

The design only includes three elements: canvas, button and key value.

- **Canvas**

1. It must have a RootCanvas, and sub-canvas can be created.
2. Text, picture and buttons can be painted on the canvas.
3. It contains background image and background color, the background will not be cleared when CLS.

- **Button**

1. Buttons contain two states, normal and pressed.
2. Each state includes the following parameters: border, background color, icon, text font, text color and text content.
3. When click on the button, the parameter key takes the value of GetKey(). The key value and physical button value are in the same queue.

- **Key value**

1. The key value and physical button value are in the same queue.
2. All windows have only one queue.
3. Not applicable to multithreading.

When programming, operations are mainly done on the RootCanvas. If dialog boxes are needed, create a sub-canvas and close it after operation.

For printer, user only needs to create a hidden canvas and write on it. After that, cut out the canvas and send it to the printer.

2Macro and Structure

2.1 Definition of Key Values

Table 2.1 Definition of Key Values

Macro	Value	Description
XUI_KEY1	*2*	*/*1*/*
XUI_KEY2	*3*	*/*2*/*
XUI_KEY3	*4*	*/*3*/*
XUI_KEY4	*5*	*/*4*/*
XUI_KEY5	*6*	*/*5*/*
XUI_KEY6	*7*	*/*6*/*
XUI_KEY7	*8*	*/*7*/*
XUI_KEY8	*9*	*/*8*/*
XUI_KEY9	*10*	*/*9*/*
XUI_KEY0	*11*	*/*0*/*
XUI_KEYCANCEL	*223*	*/*Cancel*/*
XUI_KEYCLEAR	*14*	*/*Clear*/*

XUI_KEYENTER	*28*	*/*Enter*/*
XUI_KEYALPHA	*63*	*/*Alpha*/*
XUI_KEYF1	*59*	
XUI_KEYFUNC	*102*	
XUI_KEYUP	*103*	
XUI_KEYDOWN	*108*	
XUI_KEYMENU	*139*	
XUI_KEYENTER1	*30*	/*Enter+1*/ The combination of Enter key and Key1.
XUI_KEYENTER2	*31*	/*Enter+2*/ The combination of Enter key and Key2.
XUI_KEYENTER3	*32*	/*Enter+3*/ The combination of Enter key and Key 3.
XUI_KEYENTER4	*33*	/*Enter+4*/ The combination of Enter key and Key4.
XUI_KEYENTER5	*34*	/*Enter+5*/ The combination of Enter key and Key5.
XUI_KEYENTER6	*35*	/*Enter+6*/ The combination of Enter key and Key6.
XUI_KEYENTER7	*36*	/*Enter+7/ The combination of Enter key and Key7.
XUI_KEYENTER8	*37*	/*Enter+8/ The combination of Enter key and Key8.
XUI_KEYENTER9	*38*	/*Enter+9*/ The combination of Enter key and Key9.
XUI_KEYENTER0	*39*	/*Enter+0*/ The combination of Enter key and Key10.
XUI_SOFTKEYBOARD_KEY BACKSPACE	*0xff+8*	/* backspace key */

XUI_SOFTKEYBOARD_KEY SPACE	*0xff+32*	/* space key*/
XUI_SOFTKEYBOARD_KEY EXCLAM	*0xff+33*	/* ! */
XUI_SOFTKEYBOARD_KEY DOUBLEQUOTE	*0xff+34*	/* " */
XUI_SOFTKEYBOARD_KEY SHARP	*0xff+35*	/* # */
XUI_SOFTKEYBOARD_KEY DOLLAR	*0xff+36*	/* $ */
XUI_SOFTKEYBOARD_KEY PERCENT	*0xff+37*	/* % */
XUI_SOFTKEYBOARD_KEY AMPERSAND	*0xff+38*	/* & */
XUI_SOFTKEYBOARD_KEY SINGLEQUOTE	*0xff+39*	/* ' */
XUI_SOFTKEYBOARD_KEY PARENLEFT	*0xff+40*	/* (*/
XUI_SOFTKEYBOARD_KEY PARENRIGHT	*0xff+41*	/*) */
XUI_SOFTKEYBOARD_KEY ASTERISK	*0xff+42*	/* * */
XUI_SOFTKEYBOARD_KEY PLUS	*0xff+43*	/* + */
XUI_SOFTKEYBOARD_KEY COMMA	*0xff+44*	/* , */
XUI_SOFTKEYBOARD_KEY MINUS	*0xff+45*	/* - */
XUI_SOFTKEYBOARD_KEY PERIOD	*0xff+46*	/* . */

XUI_SOFTKEYBOARD_KEY SLASH	*0xff+47*	/* / */
XUI_SOFTKEYBOARD_KEY0	*0xff+48*	/* 0 */
XUI_SOFTKEYBOARD_KEY1	*0xff+49*	/* 1 */
XUI_SOFTKEYBOARD_KEY2	*0xff+50*	/* 2 */
XUI_SOFTKEYBOARD_KEY3	*0xff+51*	/* 3 */
XUI_SOFTKEYBOARD_KEY4	*0xff+52*	/* 4 */
XUI_SOFTKEYBOARD_KEY5	*0xff+53*	/* 5 */
XUI_SOFTKEYBOARD_KEY6	*0xff+54*	/* 6 */
XUI_SOFTKEYBOARD_KEY7	*0xff+55*	/* 7 */
XUI_SOFTKEYBOARD_KEY8	*0xff+56*	/* 8 */
XUI_SOFTKEYBOARD_KEY9	*0xff+57*	/* 9 */
XUI_SOFTKEYBOARD_KEY COLON	*0xff+58*	/* : */
XUI_SOFTKEYBOARD_KEY SEMICOLON	*0xff+59*	/* ; */
XUI_SOFTKEYBOARD_KEY LESS	*0xff+60*	/* < */
XUI_SOFTKEYBOARD_KEY EQUAL	*0xff+61*	/* = */
XUI_SOFTKEYBOARD_KEY GREATER	*0xff+62*	/* > */
XUI_SOFTKEYBOARD_KEY QUESTION	*0xff+63*	/* ? */
XUI_SOFTKEYBOARD_KEY AT	*0xff+64*	/* @ */
XUI_SOFTKEYBOARD_KEY A	*0xff+65*	/* A */

XUI_SOFTKEYBOARD_KEYB	*0xff+66*	/* B */
XUI_SOFTKEYBOARD_KEYC	*0xff+67*	/* C */
XUI_SOFTKEYBOARD_KEYD	*0xff+68*	/* D */
XUI_SOFTKEYBOARD_KEYE	*0xff+69*	/* E */
XUI_SOFTKEYBOARD_KEYF	*0xff+70*	/* F */
XUI_SOFTKEYBOARD_KEYG	*0xff+71*	/* G */
XUI_SOFTKEYBOARD_KEYH	*0xff+72*	/* H */
XUI_SOFTKEYBOARD_KEYI	*0xff+73*	/* I */
XUI_SOFTKEYBOARD_KEYJ	*0xff+74*	/* J */
XUI_SOFTKEYBOARD_KEYK	*0xff+75*	/* K */
XUI_SOFTKEYBOARD_KEYL	*0xff+76*	/* L */
XUI_SOFTKEYBOARD_KEYM	*0xff+77*	/* M */
XUI_SOFTKEYBOARD_KEYN	*0xff+78*	/* N */
XUI_SOFTKEYBOARD_KEYO	*0xff+79*	/* O */
XUI_SOFTKEYBOARD_KEYP	*0xff+80*	/* P */
XUI_SOFTKEYBOARD_KEYQ	*0xff+81*	/* Q */

XUI_SOFTKEYBOARD_KEY R	*0xff+82*	/* R */
XUI_SOFTKEYBOARD_KEY S	*0xff+83*	/* S */
XUI_SOFTKEYBOARD_KEY T	*0xff+84*	/* T */
XUI_SOFTKEYBOARD_KEY U	*0xff+85*	/* U */
XUI_SOFTKEYBOARD_KEY V	*0xff+86*	/* V */
XUI_SOFTKEYBOARD_KEY W	*0xff+87*	/* W */
XUI_SOFTKEYBOARD_KEY X	*0xff+88*	/* X */
XUI_SOFTKEYBOARD_KEY Y	*0xff+89*	/* Y */
XUI_SOFTKEYBOARD_KEY Z	*0xff+90*	/* Z */
XUI_SOFTKEYBOARD_KEY BRACKETLEFT	*0xff+91*	/* [*/
XUI_SOFTKEYBOARD_KEY BACKSLASH	*0xff+92*	/* \ */
XUI_SOFTKEYBOARD_KEY BRACKETRIGHT	*0xff+93*	/*] */
XUI_SOFTKEYBOARD_KEY CARET	*0xff+94*	/* ^ */
XUI_SOFTKEYBOARD_KEY UNDERSCORE	*0xff+95*	/* _ */
XUI_SOFTKEYBOARD_KEY BACKQUOTE	*0xff+96*	/* ` */

XUI_SOFTKEYBOARD_KEYa	*0xff+97*	/* a */
XUI_SOFTKEYBOARD_KEYb	*0xff+98*	/* b */
XUI_SOFTKEYBOARD_KEYc	*0xff+99*	/* c */
XUI_SOFTKEYBOARD_KEYd	*0xff+100*	/* d */
XUI_SOFTKEYBOARD_KEYe	*0xff+101*	/* e */
XUI_SOFTKEYBOARD_KEYf	*0xff+102*	/* f */
XUI_SOFTKEYBOARD_KEYg	*0xff+103*	/* g */
XUI_SOFTKEYBOARD_KEYh	*0xff+104*	/* h */
XUI_SOFTKEYBOARD_KEYi	*0xff+105*	/* i */
XUI_SOFTKEYBOARD_KEYj	*0xff+106*	/* j */
XUI_SOFTKEYBOARD_KEYk	*0xff+107*	/* k */
XUI_SOFTKEYBOARD_KEYl	*0xff+108*	/* l */
XUI_SOFTKEYBOARD_KEYm	*0xff+109*	/* m */
XUI_SOFTKEYBOARD_KEYn	*0xff+110*	/* n */
XUI_SOFTKEYBOARD_KEYo	*0xff+111*	/* o */
XUI_SOFTKEYBOARD_KEYp	*0xff+112*	/* p */
XUI_SOFTKEYBOARD_KEYq	*0xff+113*	/* q */
XUI_SOFTKEYBOARD_KEYr	*0xff+114*	/* r */
XUI_SOFTKEYBOARD_KEYs	*0xff+115*	/* s */

XUI_SOFTKEYBOARD_KEYt	*0xff+116*	/* t */
XUI_SOFTKEYBOARD_KEYu	*0xff+117*	/* u */
XUI_SOFTKEYBOARD_KEYv	*0xff+118*	/* v */
XUI_SOFTKEYBOARD_KEYw	*0xff+119*	/* w */
XUI_SOFTKEYBOARD_KEYx	*0xff+120*	/* x */
XUI_SOFTKEYBOARD_KEYy	*0xff+121*	/* y */
XUI_SOFTKEYBOARD_KEYz	*0xff+122*	/* z */
XUI_SOFTKEYBOARD_KEYBRACELEFT	*0xff+123*	/* { */
XUI_SOFTKEYBOARD_KEYBAR	*0xff+124*	/* | */
XUI_SOFTKEYBOARD_KEYBRACERIGHT	*0xff+125*	/* } */
XUI_SOFTKEYBOARD_KEYTILDE	*0xff+126*	/* ~ */
XUI_SOFTKEYBOARD_HIDE	*0xff+255*	*/* hide */*
XUI_KEYCAMERA	*212*	*Independent key of camera*

NOTE

1. All the combination keys must be generated through “Enter” key and digital key on the physical keypad, and virtual key cannot generate combination keys. But if the virtual key is bound to the value of a certain combination key, then in this case, this virtual key can also generate this combination key value.
2. D200 (touch-key) doesn’t support combination key.
3. In addition, the value of soft keyboard minus 0xff will be

equal to the key value defined by ASCII.

2.2 Macro Definition

Table 2.2 XuiColor

Macro	Description
b	*Blue channel*
g	*Green channel*
r	*Red channel*
a	*ALPHA channel*

Table 2.3 XuiTransform

Macro	Description
XUI_ROTATE_0	*No rotation*
XUI_ROTATE_90	*Rotate clockwise by 90 degrees*
XUI_ROTATE_180	*Rotate clockwise by 180 degrees*
XUI_ROTATE_270	*Rotate clockwise by 270 degrees*
XUI_FLIP_VERT	*Flip vertically*
XUI_FLIP_HORIZ	*Flip horizontally*

Table 2.4 XuiButtonStatType

Macro	Description
XUI_BTN_NORMAL	*Normal state*
XUI_BTN_PRESSED	*Pressed State*

Table 2.5 XuiBgStyle

Macro	Description

XUI_BG_NORMAL	*Normal, display the picture from the origin x, y.*
XUI_BG_TILE	*Tile*
XUI_BG_CENTER	*Center*
XUI_BG_FOUR_CORNER	*Stretch to four corners*

Table 2.6 XuiFontSet

Macro	Description
XUI_FONT_MONO	*Monochrome font(black and white)*
XUI_FONT_GREY	*Grey font*

Table 2.7 XuiTextStyle

Macro	Description
XUI_TEXT_NORMAL	*Normal*
XUI_BOLD	*Bold*
XUI_ITALIC	*Italic*
XUI_TEXT_BOLD_ITALIC	*Bold and italic*

Table 2.8 XuiSigPenFlat

Macro	Description
XUI_SIG_FLAT	*Signing Board with smooth processing*
XUI_SIG_NORMAL	*The normal Signing Board without smooth processing*

Table 2.9 XuiWindowType

Macro	Description
XUI_WIN_CANVAS	*Canvas window*
XUI_WIN_BUTTON	*Button window*
XUI_WIN_GIF	*GIF window*

XUI_WIN_SIGBOARD	*Signature Board window*

Table 2.10 XuiShowMode

Macro	Description
XUI_SHOW_NORMAL	*Display on the screen normally*
XUI_SHOW_MIRROR	*Display on the mirror*
XUI_SHOW_ALL	*Display on the screen and mirror at the same time.*

Table 2.11 XuiAnimationType

Macro	Description
XUI_TRANSLATION	*Translate right or left.*
XUI_POLL	*Translate up or down*
XUI_SCALE	*Scale*

Table 2.12 XuiGestureType

Macro	Description
XUI_GESTURE_FLINGLEFT	*Slid to the left*
XUI_GESTURE_FLINGRIGHT	*Slid to the right*
XUI_GESTURE_FLINGUP	*Slid up*
XUI_GESTURE_FLINGDOWN	*Slid down*
XUI_GESTURE_FLINGZOOMOUT	*Zoom out with two fingers*
XUI_GESTURE_FLINGZOOMIN	*Zoom in with two fingers.*
XUI_GESTURE_SCROLLLEFT	*Scroll to the left*
XUI_GESTURE_SCROLLRIGHT	*Scroll to the right*
XUI_GESTURE_SCROLLUP	*Scroll up*
XUI_GESTURE_SCROLLDOWN	*Scroll down*
XUI_GESTURE_SCROLLZOOMOUT	*Zoom out with two fingers*

XUI_GESTURE_SCROLLZOOMIN	*Zoom in with two fingers*
XUI_GESTURE_CLICKDOWN	*Click down finger event*
XUI_GESTURE_CLICKUP	*Click up finger event*

Table 2.13 XuiCameraResolution

Macro	Description
XUI_CAMERA_IMG_640_480	*Img Width is 640 and img height is 480*
XUI_CAMERA_IMG_1024_480	*Img Width is 1024 and img height is 480*

Table 2.14 XuiPreviewZoom

Macro	Description
XUI_PREVIEW_ZOOM_100	*The resolution of camera preview window is the same as output's*
XUI_PREVIEW_ZOOM_75	*The resolution of camera preview window is 75% of output's*
XUI_PREVIEW_ZOOM_50	*The resolution of camera preview window is 50% of output's*
XUI_PREVIEW_ZOOM_25	*The resolution of camera preview window is 25% of output's*

2.3 Other Macro Definition

Macro	Description
XUI_RIGHT_X(_x, _width, _extend)	*Get text in the right-most position within _width (text-align right)*
XUI_CENTER_X(_x, _width, _extend)	*Get text in the middle position within _width (text-align horizontal center)*
XUI_CENTER_Y(_y, _height, _extend)	*Get text in the middle position within _height (text-align vertical center)*

2.4 Structure

1. Structure XuiWindow

Table 2.15 Structure XuiWindow

Structure Member	Description
width	*Window width*
height	*Window height*
widget	*Window related canvas pointer*
type	*Window type, refers to XuiWindowType*
key	*Window related key values*

2. Structure XuiImg

Table 2.16 Structure XuiImg

Structure Member	Description
width	*Img width*
height	*Img height*
priy	*Img data pointer*

3. Structure XuiButtonStat

Table 2.17 Structure XuiButtonStat

Structure Member	Description
btn_round	*rounded corner（0 means no rounded corner, 1 means rounded corner, and the default value is 0）*
btn_bg	*background color*
Text	*text*
text_fg	*text color*

text_font	*text font*
text_x	*text position:x*
text_y	*text position:y*
text_height	*text height(font size)*
Img	*Image*
img_x	*Image position:x*
img_y	*Image position:y*
img_style	*Image type*

4. Structure XuiSigBoardStat

Table 2.18 Structure XuiSigBoardStat

Structure Member	Description
btn_round	*rounded corner (0 means has no rounded corner, 1 means has rounded corner, and the default value is 0)*
btn_bg	*Background color (Transparency is not supported)*
text	*text*
text_fg	*text color*
text_font	*text font*
text_x	*Text position: x*
text_y	*Text position: y*
text_height	*Text height(font size)*
img	*Image*
img_x	*Image position: x*
img_y	*Image position: y*
img_style	*image type*

pen_fg	*pen color*
pen_width	*Pen width (ranges from 1 to 10)*
pen_flat	*Pen with smooth processing*

5. Structure XuiCameraStat

Table 2.19 Structure XuiCameraStat

Structure Member	Description
text	*Text*
text_fg	*Text color*
text_font	*Text font*
text_x	*Text position: x*
text_y	*Text position: y*
text_height	*Text height (font size)*
img	*image*
img_x	*Image position: x*
img_y	*Image position: y*
img_style	*image type*

6. Structure XuiImeAttr

Table 2.20 Structure XuiImeAttr

Structure Member	Description
parent	*Parent canvas (valid canvas pointer)*
x	*IME position x (greater than 0)*
y	*IME position y (greater than 0)*
width	*IME width (greater than 0)*
height	*IME height (greater than 4* (text_size+10))*

text_font	*IME text font (pointer of valid font)*
text_size	*IME text size (greater than 12)*
text_fg	*IME text color*
focus_fg	*Switch IME color*
img	*IME background image*
img_bg	*IME background color (transparency is not supported)*
alpha_key	*Customize alpha key value*
sharp_key	*Customize sharp key value*

7. Structure XuiGetStrAttr

Table 2.21 Structure XuiGetStrAttr

Structure Member	Description
parent	*Parent canvas (valid canvas pointer)*
x	*Input position x (greater than 0)*
y	*Input position y (greater than 0)*
font	*Input text font (valid font pointer)*
size	*Input text size (greater than 12)*
fg	*Input text color*
alpha_key	*Customize alpha key value.*

8. Structure XuiSignPoint

Table 2.22 Structure XuiSignPoint

Structure Member	Description
x	*The value of x coordinate of Signature point, the type is unsigned short.*
y	*The value of y coordinate of Signature point, the type is unsigned short.*

9. Structure XuiSignData

Table 2.23 Structure XuiSignData

Structure Member	Description
point_array	*Array of XuiSignPoint structure, which is used to save the coordinates of all the signature track points*
point_len	*Length of Point_array, the number of saved signature track points*

10. Structure XuiGesture

Table 2.24 Structure XuiGesture

Structure Member	Description
type	*Gesture type, for more information , please refer to XuiGestureType*
velocity	*The velocity of sliding the screen.*
distance	*The sliding distance.*
down_x	*The x-coordinate of where the finger presses down.*
down_y	*The y-coordinate of where the finger presses down.*
cur_x	*The current x-coordinate of gesture.*
cur_y	*The current y-coordinate of gesture.*

3XUI API

3.1 XuiOpen

Prototype	**int XuiOpen(int argc, char **argv);**	
Function	Open XUI and initialize it.	
Parameters	argc【Input】	Number of parameters
	argv【Input】	Parameter list
Return	0	Succeeded
	< 0	Failed
Instruction	The supported formats for *argv* are as below: FB=xxxxx. /*Device node of framebuffer, and the default is "/dev/graphics/fb0".*/ INPUT=xxxx /*Input device nodes, multiple nodes are allowed, and the default is "/dev/keypad" or "/dev/tp ". The input device is loaded by default only if the application does not set the parameter, if the application have set the parameter, only the device node that have been set will be loaded. For example, if this parameter is set to INPUT=/dev/tp, when XUI is initialized, only the default touch screen input will be loaded and the physical keyboard will not be loaded, which is equivalent to blocking the physical key input. */ ROTATE=xxx /*Screen rotation (values can be 0,90,180, the default value is 0, the default value will be used when the value is invalid） */	

	TSDEV=xxxx /*Device node of touch screen, the default is "/dev/input/event2".*/ STATUSBAR=xxx /*Height of the status bar(0-64，the default value is 0, the default value will be used when value is invalid) */ For example： *char *xui_argv[] = {"ROTATE=90","STATUSBAR=18"};* *XuiOpen(sizeof(xui_argv)/sizeof(xui_argv[0]), xui_argv);*

NOTE

1. When calling XuiOpen() for multiple times, only the first time takes effect, the later calls will not work unless XuiClose() is called.
2. When parameter *argc*=0 and *argv*=NULL, default settings will be enabled.
3. XUI does not support multi-process, Calling XuiOpen() between different processes will cause screen robbery during canvas operations.
4. Parameters in *argv* are independent.
5. After setting the ROTATE parameter in *argv*, the left upper corner of the screen will be defined as coordinate origin in the subsequent operations for API.
6. Xuiopen() must be called before calling other related interfaces.

3.2 XuiIsRunning

Prototype	**int XuiIsRunning(void);**	
Function	Check if the XUI is running.	
Parameters	None	
Return	1	Running.
	0	Not running.
Instruction		

3.3 XuiClose

Prototype	**void XuiClose(void);**	
Function	Close the XUI.	
Parameters	None	

Return	None
Instruction	Call this function when the application exits.

3.4 XuiSuspend

Prototype	**int XuiSuspend(void);**	
Function	Suspend the XUI.	
Parameters	None	
Return	0	Succeeded
	-1	Failed
Instruction	1. When the application needs to call another process which occupies *fb* and *event* resource. This function needs to be called suspend the XUI; otherwise, two processes will preempt *fb* and *event* resource at the same time. 2. After suspension, if necessary, call XuiResume() to resume the operation.	

3.5 XuiResume

Prototype	**int XuiResume(void);**	
Function	Resume the running status from suspended state.	
Parameters	None	
Return	0	Succeeded
	-1	Failed
Instruction	Key and touchscreen events will no longer be received after calling XuiSuspend(), so the XUI can't be resumed through those events, it can only be resumed through this function.	

3.6 XuiRootCanvas

Prototype	**XuiWindow *XuiRootCanvas(void);**	
Function	Get root canvas.	
Parameters	None	
Return	NULL	Failed
	Others	Pointer of the root canvas
Instruction	1. Call this function to do the operation on the root canvas:	

	For example: *XuiWindow* root;* *root= XuiRootCanvas();* *XuiCanvasSetBackground(root,XUI_BG_NORMAL,img_bg,color_bg);* 2. When the height of status bar exists, the canvas height is the same as screen height.

3.7 XuiStatusbarCanvas

Prototype	**XuiWindow * XuiStatusbarCanvas(void);**	
Function	Get status bar canvas.	
Parameters	None	
Return	NULL	Failed
	Others	Pointer of the status bar canvas
Instruction	It is similar to XuiRootCanvas().	

3.8 XuiCreateFont

Prototype	**XuiFont *XuiCreateFont(char *fontfile, int index, XuiFontSet fontset);**	
Function	Create font.	
Parameters	fontfile【Input】	Path of the font file.
	index【Input】	Index of the font file.
	Fontset【Input】	Font style, it supports monochrome and grey modes. Details refer to XuiFontSet.
Return	NULL	Failed
	Others	Font pointer
Instruction	Font of displaying text is created by this function. For Example: *XuiFont *font_simsun_0;* *font_simsun_0 = XuiCreateFont("/usr/font/paxfont.ttf", 0, 0);*	

1. Custom font and ttc/ttf vector fonts are supported.
2. The font is matched according to parameter *fontfile*. Firstly, match it with custom font by default, if it doesn't match, then match it with ttf or ttc font. If it doesn't match with all these three font types, NULL will be returned.
3. The parameter *index* is valid for ttc font; it is used to specify a font type of ttc font. It must be 0 for custom font and ttf font since these two only contain one type of font.
4. Users can call *XuiDestroyFont()* to destroy the created fonts which are no longer needed.
5. The custom font is created by *fontextract* tool, which can create highly customized bitmap fonts.

3.9 XuiDestroyFont

Prototype	**void XuiDestroyFont(XuiFont *font);**	
Function	Destroy fonts.	
Parameters	font【Input】	Font pointer
Return	None	
Instruction	Destroy the fonts created by XuiCreateFont().	

3.10 XuiCanvasDrawText

Prototype	**int XuiCanvasDrawText(XuiWindow *window, unsigned int x, unsigned int y, unsigned int height, XuiFont *font, XuiTextStyle textstyle, XuiColor fg, char *text);**	
Function	Display string on canvas window.	
Parameters	window 【Input】	Canvas window
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	height【Input】	Text height.
	font【Input】	Font, created by XuiCreateFont().
	textstyle【Input】	Text style (bold, italic), details refer to the

		XuiTextStyle.
	fg【Input】	Font color.
	text【Input】	Text (UTF-8 code).
Return	0 < 0	Succeeded Failed
Instruction	1. Auto linefeed is not supported. When the displaying length is beyond the canvas, the excess part will not be displayed. 2. Parameter *text* only supports UTF -8 coding; other formats should be converted to UTF-8 code first. 3. Parameter *window* must be a valid canvas pointer, or it will lead to a crash. And this warning applies to all the following interfaces.	

3.11 XuiCanvasDrawTextEx

Prototype	**int XuiCanvasDrawTextEx(XuiWindow *window, unsigned int x, unsigned int y, unsigned int height, XuiFont *font, XuiTextStyle textstyle, XuiColor fg, char *text, int linebreak);**	
Function	Display string on canvas window, and added an auto linefeed function on the basis of XuiCanvasDrawText().	
Parameters	window 【Input】	Canvas window
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	height【Input】	Text height.
	font【Input】	Font, created by XuiCreateFont().
	textstyle【Input】	Text style (bold, italic), details refer to the XuiTextStyle.
	fg【Input】	Font color.
	text【Input】	Text (UTF-8 code).
	linebreak【Input】	Whether to enable the auto linefeed function, 1-enable; 2-disable.

Return	0	Succeeded
	< 0	Failed
Instruction	1. When the *linebreak* value is 0, this function is equivalent to XuiCanvasDrawText(); Only when the *linebreak* value is 1 can the auto linefeed function be enabled. When the displaying length is beyond the canvas, the excess part will not be displayed. 2. Parameter *text* only supports UTF -8 coding; other formats should be converted to UTF-8 code first. 3. Parameter *window* must be a valid canvas pointer, or it will lead to a crash. And this warning applies to all the following interfaces.	

3.12 XuiCanvasDrawImg

Prototype	**int XuiCanvasDrawImg(XuiWindow *window, unsigned int x, unsigned int y, unsigned int width, unsigned int height, XuiBgStyle bgstyle, XuiImg *img);**	
Function	Display images on the canvas window.	
Parameters	window 【Input】	Canvas window
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	width【Input】	Image width.
	height【Input】	Image height.
	bgstyle【Input】	Background style, details refer to the XuiBgStyle.
	img【Input】	Image pointer.
Return	0	Succeeded
	< 0	Failed
Instruction	Parameter *img* must be a valid image pointer created by XuiImgLoadFormFile(); otherwise, the image can't be displayed correctly.	

3.13 XuiCanvasDrawRect

Prototype	**int XuiCanvasDrawRect(XuiWindow *window, unsigned int x, unsigned int y, unsigned int width, unsigned int height, XuiColor fg, int round, int fill);**	
Function	Display rectangle on the canvas window.	
Parameters	window 【Input】	Canvas window
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	width【Input】	Rectangle width.
	height【Input】	Rectangle height.
	Fg【Input】	Foreground color.
	round【Input】	1: Rounded, 0: Rectangular.
	fill【Input】	1: Filled 0: Hollowed
Return	0 < 0	Succeeded Failed
Instruction		

3.14 XuiClearArea

Prototype	**int XuiClearArea(XuiWindow *window, unsigned int x, unsigned int y, unsigned int width, unsigned int height);**	
Function	Clear the canvas area and cleared area will show the window background color.	
Parameters	window 【Input】	Canvas window
	x【Input】	The position x relative to canvas window

	y【Input】	The position y relative to canvas window
	width 【Input】	Width of clearing area
	height【Input】	Height of clearing area
Return	0 < 0	Succeeded Failed
Instruction	When multiple canvases are overlapped, only the content specified by parameter *window* will be cleared.	

3.15 XuiTextWidth

Prototype	**int XuiTextWidth(XuiFont *font, int size, char *text);**	
Function	Get the text width.	
Parameters	font 【Input】	The specified font created by XuiCreateFont()
	size【Input】	Font size (text height)
	text【Input】	Text string
Return	string width	
Instruction	1. Call this function when setting text alignment to center or right. 2. Parameter *font* must be a valid font created by XuiCreateFont(); otherwise, it will cause program crash. When *font* is NULL, the width of returned string is 0. 3. Parameter *text* must be a valid string pointer. When *text* is NULL, the width of returned string is 0. 4. When *size* <=0, the width of returned string is 0. 5. Only UTF-8 coding is supported; other formats need to be converted to UTF-8 code first.	

3.16 XuiTextWidthEx

Prototype	**int XuiTextWidthEx(XuiFont *font, int size, XuiTextStyle textstyle, char *text);**
Function	Get text width, this is an extended interface which can accurately acquire the width of bold and italic text.

Parameters	font【Input】	The specified font created by XuiCreateFont()
	size【Input】	Font size (text height)
	textstyle【Input】	Text type (bold, italic), details refer to XuiTextStyle.
	text【Input】	Text string
Return	string width	
Instruction	1. Call this function when setting text alignment to center or right. 2. Parameter *font* must be a valid font created by XuiCreateFont(); otherwise, it will cause program crash. When *font* is NULL, the width of returned string is 0. 3. Parameter *text* must be a valid string pointer. When *text* is NULL, the width of returned string is 0. 4. When *size* <=0, the width of returned string is 0. 5. Only UTF-8 coding is supported; other formats need to be converted to UTF-8 code first.	

3.17 XuiCreateCanvas

Prototype	**XuiWindow *XuiCreateCanvas(XuiWindow *parent, unsigned int x, unsigned int y, unsigned int width, unsigned int height);**	
Function	Create canvas.	
Parameters	parent【Input】	Parent canvas
	x【Input】	The position x relative to canvas window
	y【Input】	The position y relative to canvas window
	width【Input】	Canvas width
	height【Input】	Canvas height
Return	NULL Others	Failed Canvas pointer
Instruction	1. Parameter *parent* must be a valid canvas pointer, and this rule also applies to the following interfaces. 2. The new canvas will be displayed on the screen by calling XuiShowWindow(), and the *parent* canvas will be covered.	

3.18 XuiCreateCanvasEx

Prototype	**XuiWindow *XuiCreateCanvasEx(XuiWindow *parent, unsigned int x, unsigned int y, unsigned int width, unsigned int height, unsigned int vh);**	
Function	Create the movable canvas window, and the canvas height can be greater than the window height.	
Parameters	parent【Input】	Parent canvas
	x【Input】	The position x of canvas window relative to the parent canvas
	y【Input】	The position y of canvas window relative to the parent canvas
	width【Input】	width of the canvas window
	height【Input】	height of the canvas window
	vh【Input】	The height of the actual operation area of the canvas
Return	NULL	Failed
	Others	Pointer of the canvas window
Instruction	1. The canvas width cannot be greater than the window width. 2. When the parameter *vh* is not more than height, this function is equivalent to the XuiCreateCanvas().	

3.19 XuiCanvasMoveToY

Prototype	**void XuiCanvasMoveToY(XuiWindow * window, unsigned int my);**	
Function	Move the canvas in the canvas window.	
Parameters	parent 【Input】	Parent canvas created by XuiCreateCanvasEx().
	my【Input】	The moving height of canvas, the height is relative to the original height of canvas window.
Return	None	
Instruction	1. This function takes no effect on the canvas created by XuiCreateCanvas(). It is only valid when the canvas is	

	created by XuiCreateCanvasEx() and actual canvas height is greater than the window height. 2. Canvas can only be moved within the canvas window. 3. When moving the canvas, only the contents drawn by the function of XuiCanvasDraw() series are moveable, but sub-windows such as button, signature board and GIF are unmovable.

3.20 XuiDestroyWindow

Prototype	**void XuiDestroyWindow(XuiWindow *window);**	
Function	Destroy the canvas windows.	
Parameters	window 【Input】	Canvas window
Instruction	1. Destroy the canvas windows created by XuiCreateCanvas(), XuiCreateButton(), XuiCreateSignatureBoard() and XuiCreateGIF(). 2. When destroying the nested canvas windows, user should follow the principle of “the former created canvas windows should be destroyed after the latter created canvas windows”.	

3.21 XuiShowWindow

Prototype	**void XuiShowWindow(XuiWindow *window, int show, int flag);**	
Function	Show or hide the window.	
Parameters	window【Input】	window
	show【Input】	1: Show 0: Hide
	flag【Input】	Reserved for future use, the default value is 0.
Return	None	
Instruction		

3.22 XuiCanvasSetBackground

Prototype	**void XuiCanvasSetBackground(XuiWindow *window, XuiBgStyle bgstyle,**

	XuiImg *img, **XuiColor bg);**	
Function	Set the canvas background.	
Parameters	window【Input】	Canvas
	bgstyle【Input】	Background style. Details refer to the XuiBgStyle.
	img【Input】	Image, NULL indicates no image.
	bg	Background color.
Return	None	
Instruction	1. Screen will be cleared after calling this function. 2. This interface only takes effect on the canvas specified by *window*. Other canvas area will not be affected. 3. It does not support transparency in the background.	

3.23 XuiCreateButton

Prototype	**XuiWindow *XuiCreateButton(XuiWindow *parent,** **unsigned int x,** **unsigned int y,** **unsigned int width,** **unsigned int height);**	
Function	Create button in canvas.	
Parameters	parent【Input】	Parent canvas
	x【Input】	The position x relative to canvas window
	y【Input】	The position y relative to canvas window
	width 【Input】	width
	height【Input】	height
Return	NULL	Failed
	Others	Button pointer
Instruction		

3.24 XuiButtonSetStat

Prototype	**int XuiButtonSetStat(XuiWindow *window,** **XuiButtonStatType type,** **XuiButtonStat *stat);**

Function	Set the button state.	
Parameters	window 【Input】	Button
	type 【Input】	State type, details refer to macro XuiButtonStatType.
	stat【Input】	State variable, details refer to structure XuiButtonStat.
Return	0	Succeeded
	< 0	Failed
Instruction	1. The setting takes effect immediately after calling this function. 2. The parameter *stat* must be a valid state pointer; otherwise, it will lead to crashes. It also applies to the following interfaces. 3. When *stat*'s *text_font* and *text* are NULL, the function can return correctly, but text will not be displayed.	

3.25 XuiButtonSetKey

Prototype	**int XuiButtonSetKey(XuiWindow *window, int key);**	
Function	Set the key value of the button.	
Parameters	window 【Input】	Button
	key【Input】	Key value (key>0)
Return	0	Succeeded
	< 0	Failed
Instruction	1. After releasing the button, key values can be acquired through XuiGetKey(). 2. The *key* value must be greater than 0.	

3.26 XuiCreateSignatureBoard

Prototype	**XuiWindow * XuiCreateSignatureBoard(XuiWindow *parent, unsigned int x, unsigned int y, unsigned int width, unsigned int height);**
Function	Create the signature board.

Parameters	parent【Input】	Parent canvas.
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	width 【Input】	Width.
	height【Input】	Height.
Return	NULL	Failed
	Others	Pointer of the signature board
Instruction	1. When creating signature board, canvases cannot be overlapped. 2. Prolin-2.4 doesn't support multi-touch. 3. Prolin-phoenix-2.5 support multi-touch and it supports up to 3 points.	

3.27 XuiSigBoardSetStat

Prototype	**int XuiSigBoardSetStat (XuiWindow *window, XuiSigBoardStat *stat);**	
Function	Set the state of signature board.	
Parameters	window【Input】	Signature board.
	stat【Input】	State variable, details refer to the structure XuiSigBoardStat.
Return	0	Succeeded
	< 0	Failed
Instruction	1. The setting will take effect immediately after calling this function. 2. When *pen_flat* is XUI_SIG_FLAT in parameter *stat* , pen color and pen width can't be changed. 3. When *text_font* and *text* are NULL in parameter *stat*, the function will return correctly, but the text will not be displayed. 4. The background of signature board does not support semitransparency.	

3.28 XuiSigBoardImg

Prototype	**XuiImg * XuiSigBoardImg(XuiWindow *window);**
Function	Get the signature image.

Parameters	window【Input】	Signature board。
Return	NULL	Failed
	Others	Image pointer
Instruction	After calling this function, call XuiImgFree() to release the image.	

3.29 XuiSigBoardSignData

Prototype	**XuiSignData* XuiSigBoardSignData(XuiWindow *window);**	
Function	Get the signature data.	
Parameters	window【Input】	Signature board
Return	NULL	Failed
	Others	Data pointer, details refer to structure XuiSignData.
Instruction	1. Record the location of the signature point, the ending point of signature is 0xffff. 2. The obtained signature data pointer does not need to be released.	

3.30 XuiCreateGif

Prototype	**XuiWindow * XuiCreateGif(XuiWindow *parent, unsigned int x, unsigned int y, unsigned int width, unsigned int height , const char* path);**	
Function	Create the GIF animation.	
Parameters	parent【Input】	Parent canvas.
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	width【Input】	width
	height【Input】	height
	path【Input】	path of GIF image
Return	NULL	Failed

	Others	Pointer of GIF window
Instruction		

3.31 XuiHasKey

Prototype	**int XuiHasKey(void);**	
Function	Check whether the key value exists or not.	
Parameters	None	
Return	1 0	Yes No
Instruction		

3.32 XuiGetKey

Prototype	**int XuiGetKey(void);**
Function	Get the key.
Parameters	None
Return	Key value.
Instruction	1. This function won't return until there is a key value. 2. All keypress events are triggered when lifting, except for long keypress events. 3. The method to get long keypress: When a physical key is pressed for more than 3 seconds, the long keypress value XUI_KEYLONGPRESS will be got after calling this function. Then call this function again to get the value of the pressed physical key, but the key value will not be reported again when the key is lifted. For example, if you press *Clear* key for more than 3 seconds, XUI_KEYLONGPRESS will be returned after calling XuiGetKey(), and XUI_KEYCLEAR will be returned after calling XuiGetKey() again, but the XUI_KEYCLEAR will not be reported again when lifting the *Clear* key. 4. "Enter+ number" defines the combination of keys, so long press *Enter* key will not trigger the long keypress value. *Cancel* key is used as a power key on some models, so long press *Cancel* key will not trigger the long keypress value either.

3.33 XuiClearKey

Prototype	**void XuiClearKey(void);**
Function	Clear the key buffer.
Parameters	None
Return	None
Instruction	Clear the key buffer queue, this buffer is a dynamic linked list and the length is not fixed.

3.34 XuiCaptureScreen

Prototype	**XuiImg *XuiCaptureScreen(void);**	
Function	Capture the screen.	
Parameters	None	
Return	NULL	Failed
	Others	Pointer of image
Instruction	After calling this function, call XuiImgFree() to release the screenshot.	

3.35 XuiCaptureCanvas

Prototype	**XuiImg *XuiCaptureCanvas(XuiWindow *window, unsigned int x, unsigned int y, unsigned int width, unsigned int height);**	
Function	Capture the canvas.	
Parameters	window 【Input】	canvas
	x【Input】	The starting position x relative to canvas window
	y【Input】	The starting position y relative to canvas window
	width【Input】	width
	height【Input】	height
Return	NULL	Failed

	Others	Pointer of image
Instruction	1. After using this interface, call XuiImgFree() to release the canvas. 2. It will not capture the button on the canvas when capturing the canvas. 3. It also applies to hidden canvas. 4. Compare the parameter *width (height)* with the width (height) of canvas, and the smaller value will be used as the width (height) of the captured image.	

3.36 XuiImgLoadFromFile

Prototype	**XuiImg *XuiImgLoadFromFile(const char *file);**	
Function	Load the image from a file.	
Parameters	file【Input】	The file path.
Return	NULL Others	Failed Pointer of image
Instruction	Currently it only supports images in bmp, png and jpeg format.	

3.37 XuiImgLoadFromMem

Prototype	**XuiImg *XuiImgLoadFromMem(unsigned char *address, unsigned long length, int type);**	
Function	Load the image from the image data buffer.	
Parameters	address【Input】	Address of the image data buffer
	length【Input】	Length of the image data buffer
	type 【Input】	Image data types. 0 represents bmp data, 1 represents png data. 2 represents JPEG data.
Return	NULL Others	Failed Image pointer
Instruction	Currently it only supports images in bmp, jpeg and png format.	

3.38 XuiImgSaveToFile

Prototype	**int XuiImgSaveToFile(XuiImg *img, const char *file);**	
Function	Save the image to a file.	
Parameters	img【Input】	Image pointer.
	file 【Input】	The file path of the image to be saved. Distinguish the different file types according to suffixes. It supports suffixes of png, bmp (24-bit true color), and mbmp (monochrome bmp).
Return	0	Succeeded
	< 0	Failed
Instruction	Currently it supports png, 24-bit true color bmp and monochrome bmp, the suffix of monochrome bmp is mbmp.	

3.39 XuiImgToRgba

Prototype	**int XuiImgToRgba(XuiImg *img, char *rgba);**	
Function	Save the image to the rgba buffer.	
Parameters	img【Input】	Image pointer
	rgba 【Input】	rgba buffer.
Return	0	Succeeded
	< 0	Failed
Instruction	1. It does not detect the buffer size, please allocate a buffer with size of 4* width * height to save the image. 2. The parameter *img* must be a valid XuiImg pointer; this rule also applies to the following functions.	

3.40 XuiImgToGray8

Prototype	**int XuiImgToGray8(XuiImg *img, const char *gray8);**	
Function	Save the image to the *gray8* buffer, one byte (8bit) for each pixel.	
Parameters	img【Input】	Image pointer

	gray8【Output】	Gray8 buffer
Return	0	Succeeded
	< 0	Failed
Instruction	1. This function does not detect the buffer size, please allocate the buffer size of width * height; 2. The *img* should be a valid *XuiImg* pointer.	

3.41 XuiImgToBgr24

Prototype	**int XuiImgToBgr24(XuiImg *img, char *bgr24);**	
Function	Save the image to the *bgr24* buffer, three byte (24bit) for each pixel.	
Parameters	img【Input】	Image pointer
	bgr24【Output】	bgr24 buffer
Return	0	Succeeded
	< 0	Failed
Instruction	1. This function does not detect the buffer size, please allocate the buffer size of 4 * width * height; 2. The *img* should be a valid *XuiImg* pointer; 3. Each pixel is described with 3 bytes. In every 3 bytes, the values of b, g and r of this point are represented respectively in byte order.	

3.42 XuiImgToFrameBuffer

Prototype	**int XuiImgToFrameBuffer(XuiImg *img, unsigned char* data, unsigned int size);**	
Function	Convert the image content to *framebuffer* data.	
Parameters	img【Input】	Image pointer
	data【Output】	Framebuffer data buffer
	size【Input】	The size of *data* buffer
Return	> 0	The length of the output framebuffer data
	< 0	Failed
Instruction	1. The size of data buffer is [img width]*[img height]*[the bit of display screen]/8. If the screen is 16 bit (RGB565), the *size* value is not less than width * height * 2; if the screen is 24	

	bit (RGB888), the *size* value is not less than width * height * 3; 2. The *img* should be a valid *XuiImg* pointer.

3.43 XuiImgTransform

Prototype	**int XuiImgTransform(XuiImg *img, XuiTransform transform);**	
Function	Transform the image.	
Parameters	img【Input】	Image pointer
	transform【Input】	Transform mode. Details refer to the macro XuiTransform.
Return	0	Succeeded
	< 0	Failed
Instruction		

3.44 XuiImgResize

Prototype	**int XuiImgResize(XuiImg *img, unsigned int width, unsigned int height);**	
Function	Change the image size.	
Parameters	img【Input】	Image pointer
	width【Input】	Image width after change
	height【Input】	Image height after change
Return	0	Succeeded
	< 0	Failed
Instruction		

3.45 XuiImgCompose

Prototype	**XuiImg*XuiImgCompose(XuiImg* img1, XuiImg* img2, unsigned int rate1, unsigned int rate2, int type);**

Function	Combine two XiuImg images.	
Parameters	img1【Input】	Pointer to the buffer of first XuiImg image
	img2【Input】	Pointer to the buffer of second XuiImg image
	rate1【Input】	The ratio of the first image width
	rate2【Input】	The ratio of the second image width
	type【Input】	Reserved for future use, the default value is 0.
Return	NULL	Failed
	Others	Pointer to the newly combined XuiImg image.
Instruction	1. When the combined XuiImg image is no longer in use, call XuiImgFree() to release the memory; otherwise, it will cause memory leak. 2. The width and height of the *img1* and *img2* must be equal; otherwise, combination will fail and NULL will be returned. 3. The sum of *rate1* and *rate2* must be equal to the width of *img1* or *img2*; otherwise, combination will fail and NULL will be returned.	

3.46 XuiImgFree

Prototype	**void XuiImgFree(XuiImg *img);**	
Function	Destroy the image.	
Parameters	img【Input】	Image pointer
Return	None	
Instruction		

3.47 XuiSetStatusbarIcon

Prototype	**int XuiSetStatusbarIcon(int index, const char* path);**
Function	Set the icon of status bar.

Parameters	index【Input】	The specified icon index is 0-7 from left to right.
	path【Input】	Image path. When it is NULL, the icon will not be displayed.
Return	0	Succeeded
	-1	Failed
Instruction	1. It takes effect after setting STATUSBAR by the parameter *argv* of XuiOpen(). (that is, the height of the status bar has been set) 2. When the *path* is NULL or wrong, the original icon will be hidden.	

3.48 XuiGetHzString

Prototype	**int XuiGetHzString (XuiImeAttr attr, char *outstr, unsigned int maxlen, unsigned int timeout);**	
Function	It is a Chinese inputting interface with the mnemonic function, English letter and numeric character can also be inputted.	
Parameters	attr【Input】	Attributes of the input method, details refer to the structure XuiImeAttr. Parameter specification: ● All the pointers must be valid, such as pointers of font and parent canvas and so on; ● x and y can't be negative; ● 12 < text_size < 40 ; ● height must be greater than 4*(text_size+10); ● The transparent background is not supported.
	outstr【Input】	Store the input string (ending with '\0')
	maxlen【Input】	The maximum length of the input string (the maximum is 1024 bytes)
	timeout【Input】	Timeout value, 0 means no timeout.【unit: second】.
Return	0x00	Succeeded
	0xFE	Invalid parameter.

	0xFD Timeout
Instruction	1. Press key 【Alpha】 to switch input methods among “PinYin-Chinese”, “uppercase”, “lowercase” and “area code”. 2. Input area code. Users can input Chinese character according to the code in the mode of “area code” inputting. 3. Input Chinese. Press the corresponding numeric key in turn in the mode of “PinYin-Chinese” inputting. For example, inputting the Chinese character “中”, users should input “1466” successively, then press 【**Enter**】and key 【1】 to select the “中”. 4. Input alphabet. Press letter in the mode of “Abc” inputting, and it will display on the screen, turn pages by pressing 【Enter】, then select the target character. For example, if you press key【1】twice in succession, character “Q” will be inputted. 5. Input number. Press number in the mode of “123” inputting, then it will display on the screen. 6. Press key【Clear】to clear the inputted characters. 7. After inputting, press【Cancel】to exit the input method, and the inputted character can be obtained from the parameter *OutStr*.

3.49 XuiGetString

Prototype	**int XuiGetString(XuiGetStrAttr attr, char *outstr, unsigned char mode, int minlen, int maxlen);**	
Function	Input the character string and display it on the screen with the specified mode, the character string can be letter, amount or password etc.	
Parameters	attr【Input】	Attributes of inputting string, details refer to structure XuiGetStrAttr.
	outstr【Input】	Store the input string (ending with ‘\0’)
	mode【Input】	• D7 1(0) reserved • D6 1(0) Whether to beep when inputted string exceeds the maximum length • D5 1(0) whether to input number • D4 1(0) whether to input letter

		• D3 1(0) whether to display the ciphertext as '*' • D2 1(0) left(right)-aligned input • D1 1(0) whether the string has a decimal point • D0 1(0) reserved
	Minlen【Input】	The minimum length of the input string.
	maxlen【Input】	The maximum length of the input string (the maximum value is 128 bytes)
Return	0x00	Input successfully
	0xFE	Invalid parameter value (including the mode value is invalid; MaxLen =0; and the initial digital string is invalid.)
	0xFD	Input timeout (120 seconds, and this value can't be modified.)
Instruction		

3.50 XuiBidiStrdup

Prototype	**char * XuiBidiStrdup(const char *str);**	
Function	To do the string conversion for Arabic and Hebrew string characters, and display the Arabic and Hebrew string characters.	
Parameters	str【Input】	The UTF-8 coding string character that needs conversion.
Return	NULL	Conversion failed, parameter *str* is invalid.
	a string	The converted UTF-8 coding string character.
Instruction	When displaying Arabic and Hebrew characters, the contents need to be converted by this interface. Call XuiCanvasDrawText() after conversion to display the string as follows: *char* hebrew_text=NULL;* *hebrew_text = XuiBidiStrdup("אותך אוהב אני"); //I love you.* *XuiCanvasDrawText(XuiRootCanvas(), XUI_RIGHT_X(10, 220, XuiTextWidth(font_simsun_0, 25, hebrew_text)), 260, 25, font_simsun_0,0, color_text, hebrew_text);*	

1. The Arabic and Hebrew string character will be displayed from right to left. Macro XUI_RIGHT_X can be used to display character in right alignment.
2. This function is similar to *strdup*. The return value is stored in the memory assigned by function, and the memory needs to be released after using it; otherwise, it will cause memory leak.

   bidistr= XuiBidiStrdup(str);

   if(bidistr) free(bidistr);

3.51 XuiCanvasAnimation

Prototype	**int XuiCanvasAnimation(XuiWindow *front, XuiWindow *background, unsigned int front_rate, unsigned int background_rate, int type);**	
Function	Create switching animations of two XuiWindows.	
Parameters	front【Input】	XuiWindow before switching
	background【Input】	XuiWindow after switching
	front_rate【Input】	The ratio of front window on the display window during the switch process.
	background_rate【Input】	The ratio of background window on the display window during the switch process.
	type【Input】	The animation type used during the switch process. For more information, please refer to XuiAnimationType
Return	0	Succeeded
	< 0	Failed
Instruction	1. This function is used for switching the windows in the form	

	of animation, currently animation supports up/down/left/right translation and scaling. 2. When switching the two windows in the form of animation, these two windows need to be displayed on the mirror first, that is, calling XuiShowWindow() with XUI_SHOW_MIRROR mode. 3. This function only applies to Prolin-cygnus-2.6.

3.52 XuiGetGesture

Prototype	**int XuiGetGesture(XuiGesture* gesture);**	
Function	Get gesture event.	
Parameters	gesture【Output】	Gesture type, refer to structure XuiGesture.
Return	1	Gesture event exists in current state.
	<= 0	Gesture event doesn't exist in current state.
Instruction	1. The current supported gesture event types are up/down/left/right slide, translation and scaling. 2. This function only applies to Prolin-cygnus-2.6.	

3.53 XuiSetGestureRect

Prototype	**int XuiSetGestureRect(unsigned int x, unsigned int y, unsigned int width, unsigned int height);**	
Function	Set the corresponding area of gesture event.	
Parameters	x【Input】	x coordinate of the gesture corresponding area.
	y【Input】	y coordinate of the gesture corresponding area.
	width【Input】	The width of the corresponding area.

	height【Input】	The height of the corresponding area.
Return	0 < 0	Succeeded Failed.
Instruction	1. This function is called in combination with XuiGetGesture(). 2. This function only applies to Prolin-cygnus-2.6.	

3.54 XuiClearGesture

Prototype	**void XuiClearGesture(void);**
Function	Clear gesture event.
Parameters	None
Return	None
Instruction	This function only applies to Prolin-cygnus-2.6.

3.55 XuiShowSoftKeyboard

Prototype	**int XuiShowSoftKeyboard(int type, int show);**	
Function	Show or hide input/password soft keyboard.	
Parameters	type【Input】	Soft keyboard type: 0 means input soft keyboard; 1 means password soft keyboard.
	show【Input】	• 0 represents hidden; • 1 represents displaying; • 2 represents displaying and resident, that is, soft keyboard will not hide when users click on the “hide” button, but it can be hidden by calling this function.
Return	0 < 0	Succeeded Failed

Instruction	When showing input soft keyboard, the password soft keyboard will be automatically hidden (if it is showing). When showing password soft keyboard, the input soft keyboard will be hidden too (if it is showing). This function only applies to the POS terminal with touch screen.

3.56 XuiCreateCamera

Prototype	**XuiWindow *XuiCreateCamera (XuiWindow *parent, unsigned int x, unsigned int y, int index, int resolution, int zoom);**	
Function	Create camera preview window on canvas for photographing.	
Parameters	parent【Input】	Parent canvas.
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	index【Input】	Camera index, 0 represents the rear camera; 1 represents the front camera.
	resolution【Input】	Image resolution. For more information please refer to XuiCameraResolution.
	zoom【Input】	The zoom ratio of the preview image on preview window. For more information please refer to XuiPreviewZoom.
Return	NULL	Failed
	Others	Pointer of preview window
Instruction	1. Because the taking and previewing picture function and photographing/barcode scanning function use the same camera device, if this function is called after OsScanOpen() or OsCameraOpen(), *NULL* will be returned; if OsScanOpen() or OsCameraOpen() is called after this function, *ERR_DEV_BUSY* will be returned. 2. This function and XuiCreateScaner() use the same camera device, so they cannot be called at the same time.	

3.57 XuiCameraSetStat

Prototype	**int XuiCameraSetStat(XuiWindow *window, XuiCameraStat *stat);**	
Function	Set the preview window state and insert images and text.	
Parameters	window【Input】	The preview window
	start【Input】	State variable, please refer to structure XuiCameraStat for more details.
Return	0	Failed
	< 0	Image pointer
Instruction	1. This function will take effect immediately after setting; 2. when *stat* is NULL, illegal argument will be returned, but when *stat* is not NULL, it must be a valid pointer to the XuiCameraStat structure or the program will crash; 3. When the text_font and text in *stat* are NULL, the function returns correctly, but does not display *text*; when *img* is NULL, the function returns correctly, but does not display the image.	

3.58 XuiCameraCapture

Prototype	**XuiImg *XuiCameraCapture(XuiWindow *window);**	
Function	Acquire a frame image captured by camera.	
Parameters	window【Input】	Camera preview window
Return	NULL	Failed
	Others	Image pointer
Instruction	XuiImgFree() must be called to release memory after calling this function; otherwise, there will be a memory leak.	

3.59 XuiCreateScaner

Prototype	**XuiWindow *XuiCreateScaner (XuiWindow *parent, unsigned int x,**

	unsigned int y, **int index,** **int zoom);**	
Function	Create camera preview window on canvas for scanning barcode.	
Parameters	parent【Input】	Parent canvas.
	x【Input】	The position x relative to canvas window.
	y【Input】	The position y relative to canvas window.
	index【Input】	Camera index, 0 represents the rear camera; 1 represents the front camera.
	zoom【Input】	The zoom ratio of the preview image on preview window. For more information please refer to XuiPreviewZoom.
Return	NULL	Failed
	Others	Pointer of preview window
Instruction	1. Because the taking and previewing picture function and photographing/barcode scanning function use the same camera device,, if this function is called after OsScanOpen() or OsCameraOpen(), *NULL* will be returned; if OsScanOpen() or OsCameraOpen() is called after this function, *ERR_DEV_BUSY* will be returned. 2. This function and XuiCreateCamera() use the same camera device, so they cannot be called at the same time.	

3.60 XuiScanerDecode

Prototype	**int XuiScanerDecode(XuiWindow *window** **unsigned char *outdata** **int *datalen** **int maxlen** **int timeout);**	
Function	Decode the image captured by camera.	
Parameters	window【Input】	Preview scanning window.

	outdata【Output】	Buffer which stores decoded data, the length of one-dimensional code is recommended to be greater than 512 bytes, and the length of two-dimensional code is recommended to be greater than 3072 bytes.
	datalen【Output】	The actual length of decoded data.
	maxlen【Input】	The size of parameter *outdata* in buffer.
	timeout【Input】	Timeout period of decoding, unit: ms.
Return	0	Succeeded
	<0	Failed
Instruction		

3.61 XuiRgbaToImg

Prototype	**XuiImg *XuiRgbaToImg(char *rgba, unsigned int len, unsigned int width, unsigned int height);**	
Function	Covert rgba buffer data to XuiImg data.	
Parameters	rgba【Input】	rgba buffer data.
	len【Input】	Length of buffer data.
	width【Input】	Image width.
	height【Input】	Image height.
Return	NULL	Failed
	Others	Image pointer
Instruction	1. The parameters should meet the condition: *len* = *width* * *height* * 4; 2. XuiImgFree() must be called to release memory after calling this function; otherwise, there will be a memory leak.	

3.62 XuiFrameBufferToImg

Prototype	**XuiImg *XuiFrameBufferToImg(char *data, unsigned int len, unsigned int width, unsigned int height);**	
Function	Covert the framebuffer data to XuiImg data.	
Parameters	data【Input】	framebuffer data.
	len【Input】	Data length.
	width【Input】	Image width.
	height【Input】	Image height.
Return	NULL Others	Failed Image pointer
Instruction	1. The value of *len* can be obtained while obtaining the framebuffer data pointer. In general, the parameters should meet the condition: *len* >= *width* * *height* * *bpp*, the *bpp* is the number of bytes occupied by each pixel data in the framebuffer. It is recommended to obtain whole screenful of the framebuffer data as much as possible, the actual width and height of the screen should be set to the values of parameters *width* and *height*, and the value of the *bpp* can be ignored; 2. XuiImgFree() must be called to release memory after calling this function; otherwise, there will be a memory leak.	

3.63 XuiImgLoadFromBase64

Prototype	**XuiImg *XuiImgLoadFromBase64(const char *base64string, int *errcode);**	
Function	Load the image data which is transcoded into a string through base64.	
Parameters	base64string 【Input】	String buffer to store the image data which is transcoded through base64, the image data before transcoding can be bmp, png or jpeg format.

	errcode【Output】	When the function returns NULL, the error code is recorded. If you do not care about error messages, you can set this parameter to NULL.
Return	NULL Others	Failed Image pointer
Instruction	When transferring data between devices, it is sometimes necessary to transcode image data into a string for easy transmission, and the most common transcoding method is base64 transcoding. Based on this application scenario, this function directly receives image data (the original image format can be bmp, png or jpeg) after base64 transcoding and loads it as XUI image pointer.	

4 Note

4.1 Multi-process

Currently, XUI does not support multi-process, because they will preempt screen when running at the same time. (Multiple processes will respond to key pressing and touch duration at the same time, and showing the windows on the screen inconsistently.)

If multiple processes need to run at the same time, users can implement it by remote calling. Use a main process to manage the screen and create a canvas for each process to implement screen switches of the multiple processes.

4.2 XuiDestroyWindow

Note that when calling XuiDestroyWindow() to destroy the window, other resources used by the window have not been destroyed . So destroy window firstly and then resource (such as image, font) followed.

Please abide to this principle: the former created canvas windows should be destroyed after the latter created canvas windows. For example:

The right way to destroy:
*/*Create*/* *font_simsun_0 = XuiCreateFont ("./res/fallback.ttf", 0, 0);* *img_bg = XuiImgLoadFromFile ("./res/bg.png");* *btn = XuiCreateButton(XuiRootCanvas(), 10, 50, 220, 30);*

```
/*Destroy*/
XuiDestroyWindow(btn);
XuiImgFree(img_bg);

XuiDestroyFont(font_simsun_0);
```

The wrong way to destroy:

```
/*Create*/
font_simsun_0 = XuiCreateFont ("./res/fallback.ttf", 0, 0);
img_bg = XuiImgLoadFromFile ("./res/bg.png");
btn = XuiCreateButton(XuiRootCanvas(), 10, 50, 220, 30);
/*Destroy*/
XuiImgFree(img_bg);

XuiDestroyFont(font_simsun_0);

XuiDestroyWindow(btn);
```

5FAQ

1. The root canvas exists after opening the XUI, so can the root canvas be gotten by calling XuiRootCanvas()? Can the root canvas be destroyed?

Answer: Users can call XuiRootCanvas() to get the root canvas which cannot be destroyed. In addition, if the status bar has been set in XuiOpen(), and the status bar canvas exists after calling the XuiOpen(), users can directly get the status bar canvas by XuiStatusbarCanvas(), and the canvas cannot be destroyed.

2. Does XUI support canvas nesting? For example, Root canvas-> sub-canvas 1 -> sub-canvas 2-> sub-canvas 3-> ... -> sub-canvas N? Is there a limit to N?

Answer: Yes, it supports nesting and there is no limit to N. But users need to manage XuiWondow pointer of each canvas and not to mix them up. Follow the principle to destroy windows: the former created canvas windows should be destroyed after the latter created canvas windows.

3. Does the canvas support using the ShowWindow to display?

Answer: Yes, it does.

4. Does DestoryWindow() need to be called to release the signature board?

Answer: Yes, it does. All the returning type of XuiWindow* need to be destroyed except XuiRootCanvas() and XuiStatusbarCanvas(), since they will be destroyed automatically.

5. When displaying images, how to adjust the image size? Stretch or fill?

Answer: Do not stretch. If the image size is larger than the display area, it only displays the part which is in the area. If the image size is smaller than the display area, the blank space will be filled with the background color.

6. When calling ClearArea(), does it only clear contents in the upmost layer or all the layers? Or it is just a form of covering the area with background color?

Answer: It depends on the parameter *XuiWindow *window*, user can specify the canvas pointer of the layer that needs to be cleared, and the canvas background color will be displayed when clearing the canvas.

Prolin XUI Interface




PAX

Your Payment Partner of Choice

www.pax.com.cn

Shenzhen

4/F,No.3 Building,Software Park,Second Central Science-Tech Road,

High-Tech Industrial Park,Shenzhen

Tel:+86-755-88169630

Fax:+86-755-86169634

Hong Kong

Room 2416,24/F,Sun Hung Kai Center,30 Harbour Road,

Wanchai,Hong Kong

Tel:+852-25888800

Fax:+852-28023300 / 22951800