XJ380 API 标准文档

XJ380 API Specification C\C++版

2026/5/14
VERSION 1.3

XINGJI INTERACTIVE

目录

CONTENTS

XINGJI INTERACTIVE

CHAPTER 4 – XJ380 API (XAPI GUI 版)

CHAPTER 1

关于、编译和专有类型

ABOUT, COMPILE AND EXCLUSIVE TYPE

本章节将会介绍本篇手册以及 XJ380 API 的相关信息、如何编译为 XJ380 应用程序（EPF）以及本手册内的所有专有类型。

1-1 关于

XJ380 API 由 3（或 4）部分组成：兼容 POSIX 的 XJ380 API、XAPI 版 XJ380 API 和带 GUI 的 XAPI 版 XJ380 API（XJ380 Professional Edition 或更高版本默认集成 BrigeEngine 引擎，可使用 BAPI。详见 BrigeEngine API 标准文档）。**其中兼容 POSIX 的 XJ380 API 和 XAPI 版 XJ380 API 仅可创建控制台程序。带 GUI 的 XAPI 版 XJ380 API 和 BrigeEngine BAPI 可用于创建图形化应用程序**。这 3（或 4）种 API 可以混合使用。

1-2 编译为 XJ380 应用程序

您可以在 **https://www.xingjisoft.top/os/xj380/download** 处下载 XJ380 应用程序编译套件，或使用 SpaceCode 系列编辑器进行编译（须安装额外模块）。**请您仔细阅读套件中的配置指南并配置环境。**您也可以自行链接 XAPI 库。**请务必使用 C++11 或更高标准**，并编译为 ELF/EPF 格式。EPF 格式编译套件已包含在 XJ380 应用程序编译套件中。

您需要引入下列头文件的其中至少一个：

头文件名	用途
x3api.h	**引入所有 XJ380 API（除 BAPI）（建议使用）**
xposix.h	引入 POSIX 版 XJ380 API
xguiapi.h	引入 XAPI（GUI 版）
xtuiapi.h	引入 XAPI（无 GUI 版）
xapi.h	**引入所有 XAPI（建议使用）**
BridgeEngine.h	引入所有 BAPI
x4api.h	引入所有 XJ380 API
krlibc.h	常用函数库
xposix/*.h	部分 C 标准库

除 bapi 外，这些头文件均可以在 XJ380 应用程序编译套件中的 include 文件夹找到。bapi 相关头文件可以在 BridgeEngine 开发套件中找到。

请您务必按照要求编写主函数。主函数格式如下：

主函数格式

```
int  main(int  argc, char** argv, char** envp);
```

其中 argv 为传参数，envp 为当前用户环境变量。

编译为 XJ380 图形化应用程序（EPF 版）

使用 XJ380 开发套件中的 XXCC（XINGJI XJ380 C++ Compiler）进行编译和链接。编译参数与 G++参数相同。由于 XJ380 程序的特殊性，我们只建议您使用-O、-g 两种参数。

```
xxcc  [编译选项]  <源文件>  -o  <输出文件>
```

编译为 XJ380 终端应用程序（EPF 版）

使用 XJ380 开发套件中的 XXCCTe（XINGJI XJ380 C++ Compiler Terminal Edition）进行编译和链接。编译参数与 G++参数相同。由于 XJ380 程序的特殊性，我们只建议您使用-O、-g 两种参数。

```
xxccte  [编译选项]  <源文件>  -o  <输出文件>
```

使用 XJ380 编译套件编译为 XJ380 图形化应用程序（ELF 版）

使用 XJ380 开发套件中的 XXCC-elf（XINGJI XJ380 C++ Compiler）进行编译和链接。编译参数与 G++参数相同。由于 XJ380 程序的特殊性，我们只建议您使用-O、-g 两种参数。

```
xxcc-elf  [编译选项]  <源文件>  -o  <输出文件>
```

使用 XJ380 编译套件编译为 XJ380 终端应用程序（ELF 版）

使用 XJ380 开发套件中的 XXCCTe-elf（XINGJI XJ380 C++ Compiler Terminal Edition）进行编译和链接。编译参数与 G++参数相同。由于 XJ380 程序的特殊性，我们只建议您使用-O、-g 两种参数。

```
xxccte-elf  [编译选项]  <源文件>  -o  <输出文件>
```

使用其他编译器编译为 XJ380 应用程序（ELF 版）

我们将会为您提供参数用于编译和连接。这里拿 clang++和 ld 举例。

```
clang++  -ffreestanding  -fno-builtin  -m64  -std=c++11
-fno-stack-protector -fno-exceptions -fshort-wchar -nostdinc -c <源文
件>  -o  <目标文件>  -D __XJ380_OS__
ld  -Ttext=0x200000  <所有 XAPI TUI/GUI 目标文件+目标文件>  -o
<输出文件>
```

在 C++版本上给前面加上—即可，如 xxcc--。

如果您要编译为终端应用程序，请链接 XAPI TUI 目标文件；反之请链接 XAPI GUI 目标文件。两种 XAPI 目标文件已分别存放在编译套件中。(您可以根据需要对-std 参数进行调整，但必须保证其高于 11 版)

1-3 专有类型概览

类型名称	长度（字节）	概述
INT8	1	8 位整型
UINT8	1	无符号 8 位整型
INT16	2	16 位整型
UINT16	2	无符号 16 位整型
INT32	4	32 位整型
UINT32	4	无符号 32 位整型
INT64	8	64 位整型
UINT64	8	无符号 64 位整型
WSTR	-	char 类型字符串，使用 UTF8 编码
XWINDOW	详见 4-1-1	窗口类型
XFILE	详见 3-2-1	文件类型
XCOLOR	6	包含 3 个无符号 8 位整型 (未对齐)。格式: RGB
XCOLORA	8	包含 4 个无符号 8 位整型。格式: RGBA
HDLE	8	窗口句柄。
UserInfo	详见 1-4	用户信息结构。
XapiTaskList	详见 3-4-4	任务调度列表。

表格 1-3-1

1-4 用户及权限

XJ380 的用户共分为 5 类（见表格 1-4-1)。权限共分为 7 类：读文件（R)、写文件（W)、更改设置（CS)、访问其他目录（NV)、访问系统目录（SV)、读系统文件（SR)、写系统文件（SW)。各类用户所拥有的权限及特权级请参见下表。

用户类别	特权级（x86）	R	W	CS	NV	SV	SR	SW
Root	0	Y	Y	Y	Y	Y	Y	Y
System	3	Y	Y	Y	Y	Y	Y	Y
Admin	3	Y	Y	Y	Y/N	Y	N	N
Visitor	3	Y	Y	N	N	N	N	N
Custo	3	-	-	-	-	-	-	-

表格 1-4-1

（Y 为拥有该权限，N 为无权限，-为由用户自定义，Y/N 代表需要用户手动许可）

XJ380 会默认创建一个 Admin 权限的用户（用户名和密码由用户自行设置），并且用户最高可通过输入密码（详细方法请见官方教程，System 密码默认为 114514，可通过设置更改）获取 System 权限。

没有访问其他目录权限的应用程序仅可访问其所正在运行的程序所在目录及其子目录。其中具有 Admin 权限的应用程序在访问其他目录时，系统会向用户发送一个访问请求，经用户手动输入密码并批准后，该程序才可访问其他目录。

系统目录包括 /system 和 /EFI 目录以及其所有子目录，这里面的所有文件都一定是系统文件。

应用程序启动时，会自动将程序所属用户设置为当前登录用户，如果该线程是由其他进程/线程通过调用 fork 或 execve 生成，将会继承其父线程/进程所属用户。

目前应用程序无法自行提升权限。

1-5 辨别 XJ380 环境

XXCC 2026v2 或更高版本在编译时会自动定义一个名为__XJ380_OS__的宏。

CHAPTER 2

XJ380 API (POSIX 版)

XJ380 API (POSIX EDITION)

XJ380 API（POSIX 版）遵守且兼容一部分 POSIX 标准。

2-1 简介

XJ380 API（POSIX 版）是遵守 POSIX 标准的 API。

CHAPTER 3

XJ380 API (XAPI 版)

XJ380 API (XAPI EDITION)

XJ380 API（XAPI 版）是由 XINGJI Interactive Software 自主设计的 API 标准。所有 XAPI 均向下兼容低版本。3-3 中的类型转换函数严格意义上并不算做 API，但作为 XAPI 提供的一部分仍被写在这里。

3-1 文本输入输出

3-1-1 xapi_Output();

该函数将会向控制台输出一段字符串。

函数参数

```
void    xapi_Output(WSTR    str);
```

str　　将要输出的字符串。
返回值　无

3-1-2 xapi_Input();

该函数将会从控制台读取字符串直到空格。在读取到字符串前不会继续执行。

函数参数

```
void    xapi_Input(WSTR    str)
```

返回值　获取的字符串。

3-1-3 xapi_Getline();

该函数将会从控制台读取字符串直到换行。在读取到字符串前不会继续执行。

函数参数

```
void    xapi_Getline(WSTR    str);
```

str　　用于存储输入数据的字符串。
返回值　无

3-1-4 xapi_Getch();

该函数将会从控制台读取字符。在读取到字符前不会继续执行。

函数参数

```
char    xapi_Getch(void);
```

返回值　获取的字符。

3-1-5 xapi_EndLine();

该函数将会向控制台输出换行。

函数参数

```
void    xapi_EndLine(void);
```

返回值　无

3-1-6 xapi_PrintLine();

该函数将会向控制台输出一段字符串并换行。

函数参数

```
void    xapi_PrintLine(WSTR    str);
```

str　　将要输出的字符串。
返回值　无

3-1-7 xapi_Printf();

与 C/C++标准内的 printf 用法相同。

3-1-8 xapi_OutputSerial();

该函数将会向该电脑的串行端口输出一段字符串。

函数参数

```
void    xapi_OutputSerial(WSTR    str);
```

str 将要输出的字符串。
返回值 无

3-2 文件操作

3-2-1 XFILE 类型

类型结构

```
typedef struct {
    UINT64 length;
    void* buffer;
} XFILE;
```

length 文件长度（单位字节）。
buffer 文件内容。

3-2-2 xapi_OpenFile();

该函数将会打开文件并读取。

函数参数

```
XFILE * xapi_OpenFile(WSTR path);
```

path 文件路径。
返回值 指向 XFILE 类型结构体的指针。如果执行后为 NULL 则代表读取失败。

3-2-3 xapi_CloseFile();

该函数将会将传入结构体的内容保存至文件后关闭文件并释放内存。

函数参数

```
void xapi_CloseFile(XFILE * fsptr);
```

fsptr 指向 XFILE 类型结构体的指针。执行后会被设置为空指针。
返回值 无

3-2-4 xapi_SearchFile();

该函数将会搜索指定目录下的所有文件/文件名称并返回。最多返回 255 项，超出 255 项时将会把 **count** 的值设置为 256。如果找不到指定路径，将会把 **count** 的值设置为 404。

函数参数

```
void    xapi_SearchFile(
    WSTR            path,
    UINT32          *count,
    Dir_Node        *dir
);
```

path　指定目录的路径。
count　指向一个用于存储该路径下有多少个项目的变量的指针。
dir　需传入一个 256 大小、DIR_NODE 类型的数组，文件信息将会保存在此。
返回值　无

类型结构

```
typedef   struct {
    char        filename[256];
    UINT64      length;
    UINT64      filetype;
}   DirNode;
```

filename　文件名
length　文件长度（单位字节）。
filetype　文件类别（为 0 代表文件，为 1 代表文件夹）。

3-2-5 xapi_Mkdir();

该函数将会在指定目录下创建一个文件夹。

函数参数

```
void    xapi_Mkdir(WSTR    path);
```

path　指定目录的路径（包含要创建的文件夹）。
返回值　无

3-2-6 xapi_CreateFile();

该函数将会在指定目录下创建一个文件。

函数参数

```
void    xapi_CreateFile(WSTR    filename);
```

filename　指定目录的路径（包含要创建的文件）。
返回值　无

3-2-7 xapi_DeleteFile();

该函数将会删除指定文件。

函数参数

```
void    xapi_DeleteFile(WSTR    path)
```

path　　指定文件的路径
返回值　无

3-2-8 xapi_RenameFile();

该函数将会重命名指定文件。

函数参数

```
void    xapi_RenameFile(
    WSTR    old_path,
    WSTR    new_path
);
```

old_path　　指定目录的路径（源文件）。
new_path　　指定目录的路径（目标文件）。
返回值　无

3-2-9 xapi_ReadFile();

该函数将会读取指定文件。

函数参数

```
void    xapi_ReadFile(
    WSTR     filename,
    char     *buffer,
    UINT64   size,
    UINT64   offset
);
```

filename　　文件路径。
buffer　　指向一个用于存储目标内容的变量的指针。
size　　读取长度。
offest　　起始位置相对于文件起始的偏移。
返回值　　无

3-2-10 xapi_WriteFile();

该函数将会写入指定文件。

函数参数

```
void    xapi_WriteFile(
    WSTR     filename,
    char     *buffer,
    UINT64   size,
    UINT64   offset
);
```

filename 文件路径。
buffer 指向一个用于存储源内容的变量的指针。
size 写入长度。
offest 起始位置相对于文件起始的偏移。
返回值 无

3-2-11 xapi_Rmdir();

该函数将会在删除指定文件夹。

函数参数

```
void    xapi_Rmdir(WSTR    path);
```

path 指定目录的路径（包含要创建的文件夹）。
返回值 无

3-3 类型转换

3-3-1 xcr_char2int();

该函数将会把字符串转换为整型。

函数参数

```
UINT64    xcr_char2int(WSTR    str);
```

str 字符串。
返回值 整型。

3-3-2 xcr_int2char();

该函数将会把整型转换为字符串。

函数参数

```
WSTR    xcr_int2char(UINT64    dec);
```

dec 整型。
返回值 字符串

3-3-3 xcr_hex2char();

该函数将会把整型转换为 16 进制格式的字符串。

函数参数

```
WSTR    xcr_hex2char(UINT64    hex);
```

hex 整型。
返回值 字符串

3-3-4 toRGB();

该函数将会将 3 份数据（分别代表 R G B）转换为 32 位整型（RGBA）格式。

函数参数

```
UINT32    toRGB(
    UINT8    r,
    UINT8    g,
    UINT8    b
);
```

R 整型。代表 R 通道。
G 整型。代表 G 通道。
B 整型。代表 B 通道。
返回值 RGBA 格式的整型。

3-3-5 toRGBA();

该函数将会将 32 位整型格式（ARGB）转换为 32 位整型（RGBA）格式。
（接下页）

函数参数

```
UINT32    toRGBA(
    UINT32    color
);
```

color 整型，格式为 ARGB。
返回值 RGBA 格式的整型。

3-3-6 CC();

该宏将会将 32 位整型格式（ARGB）转换为 32 位整型（RGBA）格式。
（接下页）

函数参数

```
UINT32  CC(
    UINT32  color
);
```

color　　整型，格式为 ARGB。
返回值　RGBA 格式的整型。

3-4 进程

3-4-1 xapi_Fork();

该函数会复制当前进程映像至新进程，子进程将从 fork()后开始运行。

函数参数

```
UINT64  xapi_Fork(void);
```

返回值　父进程：子进程 PID
　　子进程：0。

3-4-2 xapi_Execve()

该函数会使目的可执行文件替换当前进程映像。

函数参数

```
UINT64  xapi_Execve(
    WSTR  filename,
    WSTR  argv[],
    WSTR  envp[]
);
```

filename　可执行文件路径。
argv[]　传给新程序的命令行参数字符串数组。
envp[]　传给新程序的环境变量字符串数组。
返回值　无：执行成功
　　-1：执行失败。

3-4-3 xapi_Exit()

该函数会退出当前线程。

函数参数

```
UINT64    xapi_Exit(UINT64    value);
```

value　返回值。
返回值　无。

3-4-4 xapi_GetTaskList()

该函数会获取操作系统当前任务调度列表。

函数参数

```
UINT64    xapi_GetTaskList(XapiTaskInfo *buffer, UINT64 max_count);
```

buffer　指向用于存储进程信息的指针。
max_count　最大条目数。
返回值　无。

3-4-5 xapi_KillProcess()

该函数会关闭指定进程（对于内核进程无效）。

函数参数

```
UINT64    xapi_KillProcess(UINT64    pid);
```

pid　进程 ID。
返回值　无。

3-5 获取当前信息

3-5-1 xapi_GetSystemVersion();

该函数会返回一串字符串作为当前操作系统版本号。

函数参数

```
void    xapi_GetSystemVersion(WSTR    version);
```

version　指向一个空字符串的指针，版本号将会储存于此。
返回值　无。

3-5-2 xapi_GetTime()

该函数会返回当前时间（单位：秒（从 1980 年开始））。

函数参数

```
UINT64    xapi_GetTime(void);
```

返回值　当前时间（单位：秒（从 1980 年开始））。

3-5-3 xapi_GetCurrentUser()

该函数会返回当前线程的用户（权限）信息。

函数参数

```
void    xapi_GetCurrentUser(UserInfo    *user_info);
```

user_info　指向一个用户结构体的指针，返回的信息将会储存于此。
返回值　无。

3-5-4 xapi_GetTimeX()

该函数会返回当前计算后的时间信息。

函数参数

```
void    xapi_GetTimeX(TimeType    *tm);
```

tm　指向一个时间结构体的指针，返回的信息将会储存于此。详细变量对应数据请参考注释。
返回值　无。

3-5-5 xapi_GetCpuModel();

该函数会返回一串字符串作为当前客户机的中央处理器型号。

函数参数

```
void    xapi_GetCpuModel(WSTR    version);
```

version　指向一个空字符串的指针，版本号将会储存于此。
返回值　无。

3-5-6 xapi_GetMemorySize();

该函数会返回当前客户机的内存大小（单位 MB）。

函数参数

```
UINT64    xapi_GetMemorySize();
```

返回值 无。

3-6 系统消息及服务

3-6-1 xapi_Broken()

该函数会结束当前程序并弹出 *应用程序崩溃* 对话框。

函数参数

```
void xapi_Broken(WSTR broken_info);
```

broken_info	自定义需要显示的崩溃信息。可以为 NULL。
返回值	无。

3-6-2 xapi_SendAppMessage()

该函数会发送一个消息通知。

函数参数

```
void xapi_SendAppMessage(WSTR title, WSTR text);
```

title	需要显示的消息标题。
text	需要显示的消息文本。
返回值	无。

3-6-3 xapi_Sleep()

该函数会进行休眠。单位为毫秒（不保证绝对准确，有一定误差）。

函数参数

```
void xapi_Sleep(UINT64 ms);
```

ms	休眠时长（单位：毫秒）。
返回值	无。

3-6-4 xapi_Run()

该函数会使用默认方式打开指定的程序或文件。

函数参数

```
void xapi_Run(WSTR path);
```

path 程序或文件的路径。
返回值 无。

3-6-5 xapi_FlushTime()

该函数会刷新当前的时间偏移量。

函数参数

```
void    xapi_FlushTime(void);
```

返回值 无。

3-7 内存

3-7-1 xapi_AllocateMemory()

该函数会分配一块指定大小的内存。

函数参数

```
void    *xapi_AllocateMemory(UINT64    size);
```

size 申请的内存大小（单位：字节）。
返回值 指向分配的内存，如为 NULL 则代表分配失败。

3-7-2 xapi_FreeMemory()

该函数会释放一块指定大小的内存。

函数参数

```
void    xapi_FreeMemory(void    *ptr);
```

ptr 指向要释放的内存的指针。
返回值 无。

3-7-3 xapi_MapMemory()

该函数会映射一块指定大小的内存到指定位置。

函数参数

```
void    *xapi_MapMemory(void    *addr, UINT64    size, UINT32    flags);
```

addr 映射起始地址。为 NULL 时则由内核选择。
size 映射范围（单位：字节）。

flags 映射标志位。

返回值 映射区域的起始地址。

PTE_PRESENT
PTE_WRITEABLE
PTE_USER
PTE_FLAG_U
PTE_HUGE
PTE_NO_EXECUTE

表格 3-7-3-1 标志位宏

CHAPTER 4

XJ380 API (XAPI GUI 版)

XJ380 API (XAPI GUI EDITION)

本章节将会介绍 XAPI 的 GUI 版本。坐标系以窗口左上角（不含标题栏）为(0, 0)。

4-1 创建图形化应用程序

4-1-1 XWINDOW 类型

类型结构

```
typedef  struct {
    UINT32   width;
    UINT32   heigh;
    WSTR     title;
    UINT8    sets;
} XWINDOW;
```

width　窗口宽度。
height　窗口高度（不包含标题栏）。
title　标题。
sets　窗口参数。如表格 4-1-1-1。不可使用多个不同类型的参数。

参数	说明
XWIN_NORMAL 或未设置	使用默认设定。
XWIN_FRAME_OFF	创建无边框窗口。
XWIN_FULL_SCR	全屏。（无边框）
XWIN_SUPPORT_RESIZEABLE	可调整窗口大小。（需搭配其他参数使用）

表格 4-1-1-1

4-1-2 xapi_CreateWindow();

该函数将会创建一个窗口。窗口被用户关闭后对应线程也将被结束。

函数参数

```
void   xapi_CreateWindow(
    HDLE*      handle,
    XWINDOW*   xwin
);
```

handle **指向**窗口句柄的指针。即指向 HDLE 类型的变量的指针。
xwin 一个指向存储了窗口参数的 XWINDOW 类型的指针。
返回值 无

4-1-3 xapi_SetWindowTitle();

该函数将会设置窗口的标题。

函数参数

```
void    xapi_SetWindowTitle(
    HDLE    handle,
    WSTR    str
);
```

handle 窗口句柄。
str 标题。
返回值 无

4-1-4 xapi_CloseWindow();

该函数将会关闭窗口，但不会结束程序。

函数参数

```
void    xapi_CloseWindow(HDLE    handle);
```

handle 窗口句柄。
返回值 无

4-1-5 xapi_SetIcon();

该函数将会设置窗口在任务栏的图标。

函数参数

```
void    xapi_SetIcon(
    HDLE    handle,
    WSTR    path
);
```

handle 窗口句柄。
path 图标路径。需为 BMP/PNG/JPEG/GIF 格式，大小 16*16。
返回值 无

4-1-6 xapi_GetWindowSize();

该函数将会获取窗口的长和宽。

函数参数

```
void    xapi_SetIcon(
    HDLE        handle,
    UINT64      *width,
    UINT64      *height
);
```

handle 窗口句柄。
width 窗口宽度。
height 窗口高度。
返回值 无

4-2 绘图

4-2-1 xapi_DrawPoint();

该函数将会在窗口上绘制点。

函数参数

```
void    xapi_DrawPoint (
    HDLE    handle,
    UINT32  x,
    UINT32  y,
    UINT32  color
);
```

handle 窗口句柄。
x 横坐标。
y 竖坐标。
color 图形的颜色。采用 RGBA（32 位色）格式。
返回值 无

4-2-2 xapi_DrawLine();

该函数将会在窗口上绘制线。

函数参数

```
void    xapi_DrawLine(
        HDLE        handle,
        UINT32          x1,
        UINT32          y1,
        UINT32          x2,
        UINT32          y2,
        UINT32          color
);
```

handle 窗口句柄。
x1 起始点横坐标。
y1 起始点竖坐标。
x2 中止点横坐标。
y2 中止点竖坐标。
color 图形的颜色。采用 RGBA（32 位色）格式。
返回值 无

4-2-3 xapi_DrawRect();

该函数将会在窗口上绘制矩形。

函数参数

```
void    xapi_DrawRect(
        HDLE        handle,
        UINT32          x1,
        UINT32          y1,
        UINT32          x2,
        UINT32          y2,
        UINT32          color,
        bool            fill
);
```

handle 窗口句柄。
x1 起始点横坐标。
y1 起始点竖坐标。
x2 中止点横坐标。
y2 中止点竖坐标。
color 图形的颜色。采用 RGBA（32 位色）格式。
fill 是否为实心。
返回值 无

4-2-4 xapi_DrawCircle();

该函数将会在窗口上绘制正圆。由于算法问题该函数被暂时移除。

4-2-5 xapi_DrawText();

该函数将会在窗口上绘制字符串。

函数参数

```
void    xapi_DrawText(
        HDLE        handle,
        UINT32      x,
        UINT32      y,
        WSTR        str,
        UINT32      size,
        UINT32      color
);
```

handle　窗口句柄。
x　横坐标。
y　竖坐标。
str　字符串。
size　字号（请注意单位不是像素）。
color　图形的颜色。采用 RGBA（32 位色）格式。
返回值　无

4-2-6 xapi_DrawTextl();

该函数将会在窗口上绘制字符串，并返回字符串绘制出来的宽度。

函数参数

```
void    xapi_DrawTextl(
        HDLE        handle,
        UINT32      x,
        UINT32      y,
        WSTR        str,
        UINT32      size,
        UINT32      color,
        UINT32      *width

);
```

handle　窗口句柄。

x 横坐标。
y 竖坐标。
str 字符串。
size 字号（请注意单位不是像素）。
color 图形的颜色。采用 RGBA（32 位色）格式。
width 字符串绘制出来的宽度（单位：像素），需传入一个指向 uint32_t 类型变量的指针。
返回值 无

4-2-7 xapi_DrawSWText();

该函数将会在窗口上绘制等宽字符串。字符宽度约为 9 像素（含两字间距）。该字体大小恒为 9 像素宽、16 像素高。（中文为 18 像素宽）

函数参数

```
void    xapi_DrawSWText(
    HDLE        handle,
    UINT32      x,
    UINT32      y,
    WSTR        str,
    UINT32      color
);
```

handle 窗口句柄。
x 横坐标。
y 竖坐标。
str 字符串。
color 图形的颜色。采用 RGBA（32 位色）格式。
返回值 无

4-2-8 xapi_CalcTextWidth();

该函数将会计算指定字符串在指定大小下的宽度。

函数参数

```
UINT64    xapi_CalcTextWidth(WSTR    str, UINT32    size);
```

str 字符串。
size 字号（请注意单位不是像素）。
返回值 宽度（单位：像素）

4-2-9 xapi_DrawSVG();

该函数将会绘制矢量图形。

函数参数

```
INT32    xapi_DrawSvg(
        HDLE handle,
        UINT32 x,
        UINT32 y,
        UINT32 width,
        WSTR svgText,
        bool enableTrans
);
```

handle 窗口句柄。
x 横坐标。
y 纵坐标。
width 宽度。
svgText SVG 字符串。
enableTrans 是否反转颜色。
返回值 成功返回高度，失败返回-1。

4-2-10 xapi_DrawFA();

该函数将会绘制指定文件中的矢量图形。

函数参数

```
INT32    xapi_DrawSvg(
        HDLE handle,
        UINT32 x,
        UINT32 y,
        UINT32 width,
        WSTR name,
        bool enableTrans
);
```

handle 窗口句柄。
x 横坐标。
y 纵坐标。
width 宽度。
svgText SVG 字符串。
enableTrans 是否反转颜色。
返回值 成功返回高度，失败返回-1。

4-3 插入图片

4-3-1 xapi_DrawBMP();

该函数将会在窗口上绘制 BMP 格式的图片。

函数参数

```
void    xapi_DrawBMP (
        HDLE      handle,
        UINT32    x,
        UINT32    y,
        UINT32    width,
        UINT32    height,
        WSTR      path
);
```

handle 窗口句柄。
x 横坐标。
y 竖坐标。
width 图片宽度。会进行拉伸。
height 图片高度。会进行拉伸。
path 图片路径。
返回值 无

4-3-2 xapi_DrawPNG();

该函数将会在窗口上绘制 PNG 格式的图片。

函数参数

```
void    xapi_DrawPNG(
        HDLE      handle,
        UINT32    x,
        UINT32    y,
        UINT32    width,
        UINT32    height,
        WSTR      path
);
```

handle 窗口句柄。
width 图片宽度。会进行拉伸。
height 图片高度。会进行拉伸。
x 横坐标。
y 竖坐标。
path 图片路径。
返回值 无

4-3-3 xapi_DrawPicture();

该函数将会在窗口上绘制 PNG\BMP\JPEG\GIF\HDR 格式的图片。

函数参数

```
void   xapi_DrawPicture(
    HDLE     handle,
    UINT32   x,
    UINT32   y,
    UINT32   width,
    UINT32   height,
    WSTR     path
);
```

handle 窗口句柄。
width 图片宽度。会进行拉伸。
height 图片高度。会进行拉伸。
x 横坐标。
y 竖坐标。
path 图片路径。
返回值 无

4-3-4 xapi_GetPicSize();

该函数将会读取 PNG\BMP\JPEG\GIF\HDR 格式的图片宽度和高度。

函数参数

```
void   xapi_GetPicSize(
    UINT32   *width,
    UINT32   *height,
    WSTR      path
);
```

width 图片宽度。
height 图片高度。
path 图片路径。
返回值 无

4-4 消息处理

4-4-1 消息处理函数

为了进行人机交互，**应用程序需要自行设定一个消息处理函数用于处理除窗口操作外的其他操作。消息处理函数必须为固定格式，并通过 SetMsgPrcor() 进行设置。**XJ380 回向处理函数传入消息类型码以及 128 位的数据。**仅在选中该窗口时才会将消息传入该**

窗口。此外，消息处理程序是作为该程序所属进程底下的一个子线程运行的。

消息处理函数固定格式：

```
void Function(UINT64 Type, UINT64 hData, UINT64 lData) {
    /* 代码 Code*/
}
```

Type 消息类型。用于识别消息。
hData 数据（高 64 位）。不同消息的内容分布不同。
lData 数据（低 64 位）。不同消息的内容分布不同。

4-4-2 键盘消息

4-4-2-1 CHAR 消息

代表有字符输入。特殊按键（如 F12、ESC、BACKSPACE 等，具体请参考表格 4-4-2-2-1）不会发送该消息。

消息识别码（宏）：**MSG_CHAR**

数据分布结构：

63 **0**

保留

hData

63 **0**

UTF-8 格式的字符。

lData

4-4-2-2 SP_CHAR 消息

代表有特殊按键被按下。普通字符不会发送此消息。

消息识别码（宏）：**MSG_SPCHAR**

数据分布结构：

63 **0**

保留

hData

63 **0**

特殊按键编号

lData

按键	编号
Esc	128
Backspace	同 \b

TAB	130
Enter	同 \n
Caps Lock	132
Shift	133
Ctrl	134
Alt	135
F1~F12	136~148
Num Lock	149
Scroll Lock	150

表格 4-4-2-2-1

4-4-3 鼠标消息

4-4-3-1 MOVE 消息

代表鼠标进行了移动。（传入的 X、Y 为相对（于窗口）坐标）

消息识别码（宏）：**MSG_MOVE**

数据分布结构:

63 **0**

鼠标的 X 坐标。

hData

63 **0**

鼠标的 Y 坐标。

lData

4-4-3-2 LBUTTON 消息

代表左键被按下后释放。

消息识别码（宏）：**MSG_LBUTTON**

数据分布结构:

63 **0**

鼠标的 X 坐标。

hData

63 **0**

鼠标的 Y 坐标。

lData

4-4-3-3 RBUTTON 消息

代表右键被按下后释放。

消息识别码（宏）：**MSG_RBUTTON**

数据分布结构：

63	0
鼠标的 X 坐标。	

hData

63	0
鼠标的 Y 坐标。	

lData

4-4-3-4 MBUTTON 消息

代表中键被按下后释放。

消息识别码（宏）：**MSG_MBUTTON**

数据分布结构：

63	0
鼠标的 X 坐标。	

hData

63	0
鼠标的 Y 坐标。	

lData

4-4-3-5 ROLLER 消息

代表滚轮被滚动。

消息识别码（宏）：**MSG_ROLLER**

数据分布结构：

63	31	0
鼠标的 X 坐标。	鼠标的 Y 坐标。	

hData

63	0
鼠标的 Z 轴坐标偏移量。（有符号）	

lData

4-4-4 控件消息

用于接收控件消息。例如按钮被按下。

消息识别码（宏）：**MSG_CRL**

数据分布结构：

63	0

控件识别码。

hData

63 **0**

控件数据。（不同控件数据不同）

lData

4-4-5 刷新消息

当窗口大小变动（如最大化或调整窗口大小），需要重绘窗口时将会发送此消息。

消息识别码（宏）：**MSG_RESIZE**

数据分布结构：

63 **0**

保留

hData

63 **0**

保留

lData

4-5 对 framebuffer 进行操作

用于直接操作窗口的 framebuffer。

4-5-1 xapi_ReadBuffer();

该函数将会获取 framebuffer 中的指定区域。

函数参数

```
void   xapi_ReadBuffer(
    HDLE        handle,
    UINT32      x,
    UINT32      y,
    UINT32      width,
    UINT32      height,
    XCOLOR*     buffer
);
```

handle 窗口句柄。

x 起始横坐标。

y 起始竖坐标。
width 宽度。
height 高度。
buffer 图像。
返回值 无

4-5-2 xapi_WriteBuffer();

该函数将会写入 framebuffer 中的指定区域。

函数参数

```
void   xapi_WriteBuffer(
    HDLE        handle,
    UINT32      x,
    UINT32      y,
    UINT32      width,
    UINT32      height,
    XCOLOR*     buffer
);
```

handle 窗口句柄。
x 起始横坐标。
y 起始竖坐标。
width 宽度。
height 高度。
buffer 图像。
返回值 无

4-5-3 xapi_ReadBufferA();

该函数将会获取 framebuffer 中的指定区域（包含透明色）。

函数参数

```
void   xapi_ReadBuffer(
    HDLE        handle,
    UINT32      x,
    UINT32      y,
    UINT32      width,
    UINT32      height,
    XCOLORA*    buffer
);
```

handle 窗口句柄。
x 起始横坐标。

y 起始竖坐标。
width 宽度。
height 高度。
buffer 图像。
返回值 无

4-5-4 xapi_WriteBufferA();

该函数将会写入 framebuffer 中的指定区域（包含透明色）。

函数参数

```
void   xapi_WriteBufferA(
    HDLE        handle,
    UINT32      x,
    UINT32      y,
    UINT32      width,
    UINT32      height,
    XCOLORA*    buffer
);
```

handle 窗口句柄。
x 起始横坐标。
y 起始竖坐标。
width 宽度。
height 高度。
buffer 图像（含 Alpha 通道）。
返回值 无

4-5-5 xapi_RefreshWindow();

该函数将会刷新整个窗口。

函数参数

```
void   xapi_RefreshWindow(
    HDLE      handle
);
```

handle 窗口句柄。

4-5-6 xapi_RefreshPartWindow();

该函数将会刷新部分窗口。

函数参数

```
void    xapi_RefreshPartWindow(
    HDLE      handle,
    UINT32        x1,
    UINT32        y1,
    UINT32        x2,
    UINT32        y2,
);
```

handle 窗口句柄。
x1 起始横坐标。
y1 起始竖坐标。
x2 结束横坐标。
y2 结束竖坐标。

4-6 控件

用于更简易的创建用户界面。

4-6-1 按钮控件

4-6-1-1 xapi_Button()

该函数将会在窗口上创建一个按钮控件。高度为 24 像素，宽度为文本宽度+22 像素，文本居中放置。

函数参数

```
void    xapi_Button(
    HDLE      handle,
    UINT64    CRLid,
    UINT64    x,
    UINT64    y,
    WSTR      text
);
```

handle 窗口句柄。
CRLid 控件识别码。
text 文本。
x 按钮左上角 X 坐标。
y 按钮左上角 Y 坐标。
返回值 无
控件数据 无

4-6-1-2 xapi_EmpButton()

该函数将会在窗口上创建一个蓝色按钮控件（可用于突出强调选项）。高度为 24 像素，宽度为文本宽度+22 像素，文本居中放置。

函数参数

```
void    xapi_EmpButton(
    HDLE      handle,
    UINT64    CRLid,
    UINT64    x,
    UINT64    y,
    WSTR      text
);
```

handle	窗口句柄。
CRLid	控件识别码。
text	文本。
x	按钮左上角 X 坐标。
y	按钮左上角 Y 坐标。
返回值	无
控件数据	无

4-6-1-3 xapi_DeleteButton()

该函数将删除指定按钮（如果有多个对应识别码的按钮则删除最先创建的），**但不会对按钮所在区域进行擦除**。

函数参数

```
void    xapi_DeleteButton(
    HDLE      handle,
    UINT64    CRLid
);
```

handle	窗口句柄。
CRLid	控件识别码
返回值	无
控件数据	无

4-7-1 右键菜单控件

4-6-1-1 RightMenuItem 类型

该类型用于存储右键菜单的其中一个目录项。

类型结构

```
typedef  struct {
    UINT64   CRLid;
    WSTR     text;
} RightMenuItem;
```

CRLid 控件识别码。
text 文本。
返回值 无
控件数据 无

4-6-1-2 xapi_RegisterRightButtonMenu()

该函数将会登记一个右键菜单。(不适用于全屏应用程序)

函数参数

```
void   xapi_RegisterRightButtonMenu(
    HDLE              handle,
    RightMenuItem   *items,
    UINT64            count
);
```

handle 窗口句柄。
items 指向一个或多个右键菜单目录项开头的指针。
count 目录项数量。
返回值 无
控件数据 无

4-6-1-3 xapi_DeleteRightButtonMenu()

该函数将会删除登记的右键菜单。

函数参数

```
void   xapi_DeleteRightButtonMenu(
    HDLE     handle
);
```

handle 窗口句柄。
返回值 无
控件数据 无

CHAPTER 5

BridgeEngine API

BridgeEngine API

本章节将会介绍鹊桥引擎的 BAPI。

5-1 简介

BridgeEngine（鹊桥引擎）是由 XINGJI 工作室开发的一款跨平台图形库及 2.5D 游戏引擎。**请移步 BridgeEngine BAPI 标准文档。**

CHAPTER 6

Stardust UI

Stardust UI

本章节将会介绍 XINGJI 星尘跨平台 UI 框架。

6-1 简介

Stardust UI（星尘 UI）是由 XINGJI 工作室开发的一款跨平台的 UI 框架。请移步 StardustUI 文档（https://github.com/xingji-studio/StardustUI）。