|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | |
| **北京群源电力科技有限公司**  China QunYuanDianLiCo., Ltd | | | | | | | | | |
| **技术方案**  Technical Solutions | | | | | | | | | |
| 责任部门研发部 | | | | | | | 适用范围：安全显示单元S-VDU软件开发 | | |
| **S-VDU开发环境软件设计说明书** | | | | | | | | | |
| 编码（Code）：QYDL-SVDU-009 | | | | | 版次（Rev**.**）：B | | | | |
| 原编码（Ori. Code）：无 | | | | |
| 正文页数 (Text Pages)： | | | 附录页数 (Appendices)：0 | | | | | | |
|  | | | | | | | | | |
|  | 部门  Dept | 姓名  Name | | 签名  Signature | | 日期  Date | | | 批准  Approved by |
| 编写  Drafted by | 研发 | 李晓虎 | |  | |  | | | 职务（Duty）：总经理  姓名（Name）：支源 |
| 校核  Checked by | 研发 | 黄小平 | |  | |  | | |
| 审查  Reviewed by | 研发 | 刘大智 | |  | |  | | | 签名(Signature)： |
| 审查  Reviewed by |  |  | |  | |  | | |
| 审查  Reviewed by |  |  | |  | |  | | | 日期（Date）： |
| 下次升版时间  （Next ReviewDate） | |  | | | | | | | |
| 分发（Distribution）： | | | | | | | | 原件存（Filing）： | |
| **此文件产权属北京群源电力科技有限公司所有，未经书面许可，不得以任何方式外传。**  This document is the property of China QunYuanDianLi Co., Ltd, no part of this document  shall be transmitted without its written permission. | | | | | | | | | |

**修订页**

1. **文件的修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版次号 | 修订时间 | 修订的章节 | 说明 |
| A | 2016.08.01 | ALL | 初次发布 |
| B |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **本版与旧文件（版本）的关系**

无

1. **文件发放**

**目录**

[1 目的 1](#_Toc474657828)

[2 适用范围 1](#_Toc474657829)

[3 参考文件 1](#_Toc474657830)

[4 专业术语 1](#_Toc474657831)

[5 基础类设计 1](#_Toc474657832)

[5.1 SVBitmap类 1](#_Toc474657833)

[5.2 SVCheckValidException类 1](#_Toc474657834)

[5.3 SVColorButton类 1](#_Toc474657835)

[5.4 SVConfig类 2](#_Toc474657836)

[5.5 SVCrc32类 3](#_Toc474657837)

[5.6 SVNameToObject类 3](#_Toc474657838)

[5.7 SVInterfacePanel接口 4](#_Toc474657839)

[5.8 SVInterfaceBuild接口 5](#_Toc474657840)

[5.9 SVPrinter接口 5](#_Toc474657841)

[5.10 SVPixmapFile类 5](#_Toc474657842)

[5.11 SVProData类 6](#_Toc474657843)

[5.12 SVProject类 7](#_Toc474657844)

[5.13 SVRedoUndo类 10](#_Toc474657845)

[5.14 SVSelectAlignProperty类 12](#_Toc474657846)

[5.15 SVSelectPanelObjs类 12](#_Toc474657847)

[5.16 SVSerialize类 16](#_Toc474657848)

[5.17 SVUniqueID类 16](#_Toc474657849)

[5.18 SVXml类 17](#_Toc474657850)

[5.19 SVPanelNode类 19](#_Toc474657851)

[5.20 SVBasePanel类 19](#_Toc474657852)

[5.21 SVPanel类 19](#_Toc474657853)

[5.22 SVBitmapArray类 21](#_Toc474657854)

[5.23 SVBitmapHead类 21](#_Toc474657855)

[5.24 SVWPFBitmapManagerWindow类 21](#_Toc474657856)

[5.25 SVWpfControl类 21](#_Toc474657857)

[5.26 SVWpfColorDialog类 21](#_Toc474657858)

[5.27 SVWPFLockDialog类 21](#_Toc474657859)

[5.28 SVWPFProgressBar类 21](#_Toc474657860)

[5.29 SVWPFSettingWindow类 21](#_Toc474657861)

[5.30 SVBuildFile类 21](#_Toc474657862)

[5.31 SVLog类 22](#_Toc474657863)

[5.32 SVTextBoxTraceListenter类 22](#_Toc474657864)

[5.33 SVTextTraceListenter类 22](#_Toc474657865)

[5.34 SVMessageBox类 23](#_Toc474657866)

[5.35 SVSqlDataBase类 23](#_Toc474657867)

[6 控件类设计 24](#_Toc474657868)

[6.1 SVListView类 25](#_Toc474657869)

[6.2 SVAnalogProperties类 25](#_Toc474657870)

[6.3 SVAnalog类 26](#_Toc474657871)

[6.4 SVBinaryProperties类 26](#_Toc474657872)

[6.5 SVBinary类 27](#_Toc474657873)

[6.6 SVBinaryTypeConverter类 27](#_Toc474657874)

[6.7 SVButtonProperties类 27](#_Toc474657875)

[6.8 SVButton类 28](#_Toc474657876)

[6.9 SVBitmapTypeEditor类 29](#_Toc474657877)

[6.10 SVBitmapManagerWindow类 29](#_Toc474657878)

[6.11 SVBtnTypeConverter类 29](#_Toc474657879)

[6.12 SVBtnTypeEditor类 29](#_Toc474657880)

[6.13 SVCurveProperties类 29](#_Toc474657881)

[6.14 SVCurve类 30](#_Toc474657882)

[6.15 SVGifProperties类 30](#_Toc474657883)

[6.16 SVGif类 31](#_Toc474657884)

[6.17 SVGifWindow类 31](#_Toc474657885)

[6.18 SVIconProperties类 31](#_Toc474657886)

[6.19 SVIcon类 31](#_Toc474657887)

[6.20 SVLabelProperties类 31](#_Toc474657888)

[6.21 SVLabel类 32](#_Toc474657889)

[6.22 SVLineProperties类 32](#_Toc474657890)

[6.23 SVLine类 32](#_Toc474657891)

[6.24 SVHeartbeatProperties类 32](#_Toc474657892)

[6.25 SVHeartbeat类 32](#_Toc474657893)

[6.26 SVWPFAlignDialog类 32](#_Toc474657894)

[6.27 SVWPFBtnMemoEdit类 33](#_Toc474657895)

[6.28 SVWPFBtnTextEdit类 33](#_Toc474657896)

[6.29 SVWPFFontDialog类 33](#_Toc474657897)

[6.30 SVWPFHeartPic类 33](#_Toc474657898)

[6.31 SVWPfIconPic类 33](#_Toc474657899)

[6.32 SVWPFVariableDialog类 33](#_Toc474657900)

[6.33 WPFWidgetBackGround类 33](#_Toc474657901)

[7 全局数据结构设计 33](#_Toc474657902)

[7.1 数据结构定义 33](#_Toc474657903)

[7.1.1 公共数据结构 33](#_Toc474657904)

[7.1.2 矩形区域数据结构 34](#_Toc474657905)

[7.1.3 线数据结构 34](#_Toc474657906)

[7.1.4 静态文本数据结构 35](#_Toc474657907)

[7.1.5 按钮数据结构 35](#_Toc474657908)

[7.1.6 模拟量显示数据结构 37](#_Toc474657909)

[7.1.7 数字量显示数据结构 38](#_Toc474657910)

[7.1.8 趋势图数据结构 39](#_Toc474657911)

[7.1.9 静态图数据结构 39](#_Toc474657912)

[7.1.10 动态图数据结构 40](#_Toc474657913)

[7.1.11 心跳控件数据结构 40](#_Toc474657914)

[7.2 页面数据结构 41](#_Toc474657915)

[7.3 页面配置文件数据结构 43](#_Toc474657916)

[7.4 图元文件数据结构 43](#_Toc474657917)

[7.5 相关处理函数 44](#_Toc474657918)

[8 总体设计 44](#_Toc474657919)

[8.1 需求概述 44](#_Toc474657920)

[8.2 软件结构 45](#_Toc474657921)

[8.3 界面整体设计 46](#_Toc474657922)

[8.4 模块详述 47](#_Toc474657923)

[8.4.1 工程管理模块 47](#_Toc474657924)

[8.4.2 页面管理模块 50](#_Toc474657925)

[8.4.3 页面编辑 54](#_Toc474657926)

[8.4.4 编译功能 63](#_Toc474657927)

[8.4.5 模板管理 65](#_Toc474657928)

[8.4.6 图元管理 67](#_Toc474657929)

[8.4.7 变量管理 69](#_Toc474657930)

[8.4.8 数据库相关操作设计 70](#_Toc474657931)

[8.4.9 其他功能 70](#_Toc474657932)

# 目的

本文为S-VDU开发环境软件设计说明书。设计依据为《S-VDU开发环境软件需求规格书》。主要目的是为了正确指导后续软件编码人员实现具体功能，方便多个开发人员协同工作和软件后期维护中作为维护人员对软件功能修改及完善的参考。本文是对S-VDU开发环境软件中的接口、算法及数据结构等内容进行详细描述。

# 适用范围

本软件设计说明书对应的软件产品为：S-VDU开发环境软件。

# 参考文件

# 专业术语

* 1. 下装配置文件：开发环境软件生成的用于在运行环境软件中使用的接口二进制文件，包含页面和图片数据；
  2. LINQ：语言集成查询，是c#语言中用于提供查询数据能力的一个特性；
  3. 控件：在页面窗口中显示的基础图形单元。

# 基础类设计

## SVBitmap类

定义一个图元类，用来记录当前图元的索引信息。该类具有序列化和自定义属性字段的特点。包含两个字符串类型属性字段ImageFileName和ShowName，分别记录当前图元对应的文件名路径和该图元在窗口中的名称显示。

## SVCheckValidException类

自定义异常处理类，继承自应用程序异常类。包含一个异常处理信息，为字符串类型。主要是用来处理在页面检查过程中对所检查项通不过的时候，记录发生的检查错误信息，并将错误异常以日志的方式显示在界面及记录在日志文件中。

## SVColorButton类

具有显示和选择颜色功能的按钮。通常用来在窗口中提供用户选择颜色的一种方式。

## SVConfig类

一个与当前软件本身相关的配置类，用来记录当前软件的配置信息。在当前软件中主要用来记录用户最近打开工程文件、当前记录工程文件的最大个数、定时保存工程时间间隔。

返回当前配置的一个实例，这里采用单例模式，工程中只存在一个唯一的实例。

public static SVConfig instance()

返回最近打开工程文件的路径名列表

List<String> RectFileItems

参数:：file 表示记录当前打开工程文件名，类型为String

返回值:：void

描述：用来记录名为file的工程文件到最近打开列表中。

public void addRectFilesItem(String file)

用来设置和获取当前可以保存的最大最近打开列表个数，范围（1-10），默认值为5

public Int32 RectCount

保存工程的时间间隔，设置范围为(1-60)，单位分钟，默认值为5

public Int32 SaveInterval

描述：记录当前软件使用的语言版本，方便下次打开软件的时候应用该语言。

分为中文和英文，默认为中文。

public String Language

从配置文件中加载配置信息

public void loadConfig()

保存当前配置信息到配置文件中

public void saveConfig()

## SVCrc32类

计算一段内存的CRC32的值，该类是从C语言版本移植过来。

参数：init 初始值

s 表示内存数组

len 表示内存的长度

返回：32为的crc验证码结果

UInt32 calCrc32Append(UInt32 init, byte[] s, UInt32 len)

参数：buffer表示内存数组

len 表示内存的长度

返回：32为的crc验证码结果

描述：返回一个初始值为0，计算长度为len内存的crc值

public UInt32 calculateCrc32(byte[] buffer, UInt32 len)

## SVNameToObject类

该类是一个全局静态类，主要是通过一个名称及类型来创建相应的对象。利用了C#语言的反射机制特性。主要是用来方便从名字到对象的转换过程，对于加载工程时，从名称到对象的还原提供便利。

参数：value类型Type

返回：void

描述：初始化类型映射表，根据当前输入的类型，记录名称和类型的对应关系。

public static void addInstance(Type value)

参数：name对象名称

返回：object对象，创建失败返回null

描述：与上面函数配合使用，只有当记录了类型，才能够通过名称正确创建对象。

public static object getInstance(String name)

参数：name 对象名称

返回：object对象，创建失败返回null

描述：可以单独使用，相当于上述两个函数共同使用的作用。通过名称可以创建对应的类对象。

public static object createInstance(String name)

## SVInterfacePanel接口

用来定义一个接口，统一所有控件的行为。所有需要使用该接口的类，都需要实现接口中的函数定义。

用来将属性对应的值，应用到控件中，让当前控件的显示与属性中的值相匹配。

void refreshPropertyToPanel();

返回当前对象的属性

object property();

参数：element自定义xml管理对象

isCreate为true新建ID号，false为直接使用文件中的ID

返回：void

描述：从xml文件中加载对象内容到内存对象中

void loadXML(SVXml element, Boolean isCreate = false);

参数：element自定义xml管理对象

返回：void

描述：保存当前对象到xml文件中

void saveXML(SVXml element);

## SVInterfaceBuild接口

定义一个通用接口，所有要生成下装文件内容的类，都需要实现该接口。目的是通过统一的行为将相关类中的内容转换为下装bin文件中的文件格式内容。

## SVPrinter接口

页面打印类，可以打印任意的图片对象。这里主要是用来打印页面控件中的图片，原理是将控件的图片获取出来保存到内存中，然后通过该类执行打印过程。

参数：bitmap 图片对象

返回：void

描述：将图片做为参数传入，最终通过打印机打印出来。

public void printBmp(Bitmap bitmap)

## SVPixmapFile类

作为管理一个图标对象的类，用来在用户导入一个图标后，按一定的格式将图片转换后保存到图标库中。

参数：filename 图标文件名

返回：void

描述：读取图标文件到内存中

public void readPixmapFile(String filename)

参数：filename 图标文件名

返回：void

描述：将图标文件保存到文件中

public void writePixmapFile(String filename)

参数：void

返回：Bitmap图片对象

描述：获取当前的图标图片数据，在读取图片数据的时候，必须保证数据已经从文件中加载

public Bitmap getBitmapFromData()

参数：width图像的宽度

Height图像的高度

Mark 值只能为8或者24，否则无效

返回：根据指定的宽度和高度，返回当前图标的图片对象

描述：返回的图片为8位或者24位bmp图片格式对象。这里的图片已经经过上下反转，目的是下位机不需要再一次转换，节约下位机显示图片的时间。

public Bitmap getBitmapObject (Int32 width, Int32 height, Int32 mark)

参数：inputBitmap输入的图片对象

返回：上下翻转后的图片

描述：将一副图片对象执行上下翻转后，得到新的图片对象。该函数已经可以实现图片的翻转，但是对于大图片的反转效率较低，会造成屏幕显示过程中转换时间花费过多，出现明显的停顿现象，目前已经弃用。

public Bitmap RevPic(Bitmap inputBitmap)

参数：buffer一段大图片内存

offset 偏移量

返回：在指定偏移量的位置来返回图片数据对象

public Bitmap getFromFile(byte[] buffer, UInt32 offset)

## SVProData类

每次工程被打开的时候，记录当前工程的主要路径信息，作为一个全局类。方便在不同的模块中根据确定的文件路径来查找相关文件。

工程名称

static public String ProName

工程路径

static public String ProPath

工程文件完整路径

static public String FullProPath

模板路径

static public String TemplatePath

配置文件

static public String ConfigFile

生成的下装路径

static public String DownLoadFile

图标路径

static public String IconPath

当前工程日志文件保存路径

static public String LogPath

当前工程站号

static public Int32 stationID

当前连接数据库用户名

static public String user

当前连接用户密码

static public String passwd

当前连接数据库ip地址

static public String dbIp

## SVProject类

保存当前工程中的数据。

参数：file工程文件

返回：void

描述：利用c#的linq技术，从工程文件的xml文件中，读取页面相关信息到指定内存结构中

void linqFromFile(String file)

参数：file 工程文件

返回：void

描述：利用c#的linq技术，保存工程内容到xml工程文件中

void linqSaveFile(String file)

参数：path工程路径

Name工程名称

返回：void

描述：通过工程路径以及工程名来新建一个工程。包括了工程文件夹的创建等一系列动作。

public void createProject(String path, String name)

参数：path工程路径

Name工程名称

返回：Bool 失败返回false，成功为true

描述：通过工程路径以及工程名来打开一个工程。

public Boolean openProject(String path, String name)

关闭工程

public void closeProject()

参数：void

返回：void

描述：保存当前工程文件，保存当前工程中未使用的ID号。

public void saveProject()

参数：页面名称

返回：页面相对路径名

描述：在创建和打开工程的时候，通过页面名称来获取页面文件路径。

public String name2Path(String name)

参数：className分类名称

Name当前页面名称

返回：bool 成功-true，失败-false

描述：在打开或者新建页面节点的时候需要调用该函数，页面名称不能重复，否则会导致添加节点失败

public Boolean addPageNode(String className, String name)

参数：className分类名称

file当前页面名称

返回：void

描述：从外部导入一个页面

public void importPageNode(String className, String file)

参数：页面名称

返回：void

描述：通过页面名称来移除对应工程页面信息

public void removePageNode(String name)

参数：页面名称

返回：name 页面名称

描述：根据页面名称来删除对应的磁盘页面文件。如果文件不存在，直接返回。否则执行删除文件操作

public void removePageNodeFile(String name)

参数：pageName页面名称

返回：页面名称对应的页面相对路径

描述：如果存在就返回，否则返回null

public String pageFileFromName(String pageName)

参数：pageName页面名称

返回：bool类型

描述：存在就返回true。判断当前名称为pageName的页面是否存在

public Boolean isExist(String pageName)

参数：oldName 原有的页面分类名称

newName 新的页面分类名称

返回：void

描述：对页面分类进行重命名操作

public void renamePageClassName(String oldName, String newName)

参数：className 当前页面分类名称

oldName 页面名称

newName 新修改的页面名称

描述：对页面名称重命名。

public void renamePageName(String className, String oldName, String newName)

参数：void

返回：字典对象

描述：返回当前工程的页面分类及页面名称对应的页面路径，包含了工程文件中的内容。工程文件中只是记录当前页面名称及对应的文件即可。

public Dictionary<String, Dictionary<String, String>> getDictData()

## SVRedoUndo类

撤销和重做类，主要用来在页面编辑过程中起到撤销编辑过程中的操作以及恢复执行该操作的管理类。实现基于列表模拟的堆栈数据结构，当有数据改变的时候，执行压栈操作。执行撤销的时候，将当前数据向移动栈底并执行对应操作，如果为重做操作，将当前数据向栈顶位置移动并执行对应操作。否则如果为记录当前数据改变，即从当前位置继续压栈，栈顶数据将会丢失。列表中的每一项都有一个对应撤销执行函数和恢复执行函数，只有在当前项发生移动的时候才执行。



图1撤销恢复数据结构

参数：en表示是否记录，true-记录，false-不记录

返回：void

描述：设置当前撤销恢复类的状态，当有数据发生变化的时候，是否进行记录。

public void setEnabled(Boolean en)

参数：item要记录的当前操作项

返回：void

描述：记录当前操作，以便后来执行对应的撤销和恢复操作。该函数与上面函数设置状态有关，如果状态设置为false，该函数的执行将不起作用

public void recordOper(SVRedoUndoItem item)

参数：void

返回：void

描述：执行重做操作

public void Redo()

参数：void

返回：void

描述：执行撤销操作

public void Undo()

参数：void

返回：当前剩余撤销次数，类型int

描述：无

public Int32 getUndoCount()

参数：void

返回：当前剩余恢复次数，类型int

描述：无

public Int32 getRedoCount()

## SVSelectAlignProperty类

为满足现有文本对齐方式，新加的属性装饰类，在属性窗口中限制用户在下拉列表中选择文本的对齐方式，范围设置为左对齐、右对齐、居中对齐、水平和垂直居中对齐。

## SVSelectPanelObjs类

一个全局的选择对象控件的管理类。该类中主要包含了一个选择对象列表，一个需要复制对象列表，全局的Ctrl按键是否按下状态记录。由于目前的设计是基于对象的复制、剪切和粘贴操作是可以跨多个页面进行操作，这里作为全局的类来管理比较合理和方便。当对一个对象执行选中的时候，将该对象添加到对象列表中，如果从选中状态转为未选中，就将该对象从对象列表中删除。只有被选中的对象，即对象列表中的对象才能执行复制的功能。在执行复制前，需要将对象列表中相关对象拷贝一份到复制对象列表中。

参数：无

返回：void

描述：清除所有当前选中控件的状态，使其处于没有选中

public void clearSelectControls ()

参数：control 当前选中控件对象

X 当前鼠标移动在x方向上的偏移

Y当前鼠标移动在Y方向上的偏移

返回：void

描述：移动当前控件的时候，所有被选中的控件也将一并会被移动，这里该函数主要是实现，鼠标拖动批量移动控件功能。

void moveSelectControls(Control control, int x, int y)

参数：无

返回：当前选中控件个数

描述：获取当前用户操作通过鼠标选中的控件数目

public Int32 selectCount()

参数：control当前选中的控件

返回：void

描述：控件选中后添加该控件到对象列表中

public void addControlItem(Control control)

参数：control当前反选的控件

返回：void

描述：将该控件从对象列表中移除

public void removeControlItem(Control control)

参数：void

返回：void

描述：在页面中删除当前选中的控件

public void removeOperator()

参数：void

返回：void

描述：将当前对象列表中的内容移到复制对象列表中

public void copyOperator()

参数：void

返回：void

描述：执行剪切操作

public void cutOperator()

参数：void

返回：void

描述：执行粘贴操作

public void pasteOperator()

参数：void

返回：void

描述：左对齐操作

public void leftAlign()

参数：void

返回：void

描述：右对齐操作

public void rightAlign()

参数：void

返回：void

描述：上对齐操作

public void topAlign()

参数：void

返回：void

描述：下对齐操作

public void bottomAlign()

参数：void

返回：void

描述：垂直居中操作

public void vCenterAlign()

参数：void

返回：void

描述：水平居中操作

public void hCenterAlign()

参数：void

返回：void

描述：垂直等间距

public void vEqual()

参数：void

返回：void

描述：水平等间距

public void hEqual()

参数：void

返回：void

描述：对选中控件执行上移

public void up()

参数：void

返回：void

描述：对选中控件执行下移

public void down()

参数：void

返回：void

描述：对选中控件执行左移

public void left()

参数：void

返回：void

描述：对选中控件执行右移

public void right()

## SVSerialize类

目前下装文件格式是自定义的，为满足组态页面与文件之间的转换对应关系，系统默认的序列化功能不能满足需要，这里自定义序列化类，用来序列化和反序列化数据下装二进制数据。将各个基础数据序列化为以一字节对齐的方式排列，产生与其他接口通用的数据格式。

## SVUniqueID类

在开发环境软件中，页面及控件都有一个唯一的ID标识符，为避免ID的重复性以及随着对控件的不断新建和删除耗尽ID分配，这里采用可回收的方式对ID进行分配。当有页面和控件有新建操作的时候，ID从已有的ID库中进行分配，页面和控件都拥有自己的ID号，当页面或者控件被删除的时候，ID被回收到ID库中以便下次新建时使用。



图 全局ID分配和释放

## SVXml类

新分装的xml操作类，基于c#自身的xml操作类的进一步封装，目的是隐藏部分细节，方便针对单个页面的读取和保存操作，便于程序结构拥有统一的xml接口，方便数据的操作及后续维护。

参数：xmlFileName 表示当前的xml文件

返回：true表示加载成功，false表示加载失败（由于文件不存在或者格式不正确）

描述：在读取xml文件数据的时候执行

public Boolean loadXml(String xmlFileName)

参数：name初始化xml文件格式的根节点名称

返回：true表示初始化成功，否则表示失败

描述：初始化根节点名称，表示判断当前xml文件中是否存在该名称的根节点

public Boolean initRootEle(String name)

参数：name当前根节点名称

返回：void

描述：根据名称创建根节点，用来在保存文件的时候使用

public void createRootEle(String name)

参数：nodeName节点名称

返回：当前节点名称对应的xml对象

描述：根据名称得到对应的xml操作对象

public XmlElement select(String nodeName)

参数：void

返回：得到当前节点的所有子节点对象

描述：无

public List<XmlElement> selectChilds()

参数：nodeName输入的新的节点名称

返回：根据名称新建的xml节点对象

描述：在新建xml节点对象的时候使用

public XmlElement createNode(String nodeName)

参数：key当前节点的属性字段

Value设置当前属性字段的值

返回：void

描述：提供一种方式设置当前属性字段对应值得方式，针对单一属性

public void setAttr(String key, String value)

参数：dictValue以字典的方式提供属性及对应值的设置方式

返回：void

描述：重载设置属性的函数，针对同时设置多个属性值得情况

public void setAttr(Dictionary<String, String> dictValue)

参数：value设置当前节点文本值

返回：void

描述：无

public void setValue(String value)

参数：xmlFileName指定的xml文件路径

返回：void

描述：保存xml文件执行的操作

public void writeXml(String xmlFileName)

## SVPanelNode类

定义一个节点控制类，作用于当前页面中的所有控件。当控件被选中的时候，该类的对象将会在界面上显示出来，位于控件的上、下、左、右、左上角、右上角、左下角、右下角位置，8个对象分别拥有不同的事件处理函数。作为修改控件尺寸的控制点。

## SVBasePanel类

基础控件类，包含SVPanelNode类对象。包含了一个控件锁定、选中、模式切换、控制点的显示方式等基础属性设置及外观显示，形成一个包含控制点的通用基础控件对象。

## SVPanel类

自定义的抽象类，用于统一所有控件的公共行为。

该抽象类继承自基础控件SVBasePanel，目的是利用已有控件的事件属性，如按钮的获取焦点，离开焦点，按下，移动，弹起等事件判断。在此基础上添加额外的控件重绘接口函数和ID分配接口；

该类继承自接口SVInterfacePanel；

该类继承自接口ICustomTypeDescriptor，用于在设置属性窗口的时候，可以通过直接赋值的方式将当前对象的属性生效，并且能够屏蔽原有控件属性显示。

以下为抽象类接口，需要子类在继承的时候实现。

描述：设置当前对象在显示窗口中的起始位置。

abstract public void setStartPos(Point pos);

描述：获取当前对象的属性

abstract public object property();

描述：将当前属性中与显示有关的内容应用到控件显示中

abstract public void refreshPropertyToPanel();

描述：从文件中加载该控件的数据，isCreate为真表示新建ID号，否则为读取文件中的ID号使用。

abstract public void loadXML(SVXml element, Boolean isCreate = false);

描述：保存当前控件数据到xml文件中。

abstract public void saveXML(SVXml element);

描述：克隆当前对象，使当前对象产生一个副本而不是引用。

abstract public object cloneObject();

描述：从ID库中获取一个新的ID号做为当前控件ID

abstract public void createID();

描述：回收当前控件的ID

abstract public void delID();

描述：设置撤销和恢复相关的初始化操作

abstract public void initalizeRedoUndo();

描述：只针对当前控件的数据进行的合法性检查，如果不合法将抛出指定异常信息。

abstract public void checkValid();

描述：将当前控件置顶显示

public void showOnTheTop()

描述：重置系统绘制函数，自定义绘制方式

protected override void OnPaint(PaintEventArgs e)

描述：接口虚函数，用来在子类中自定义绘制方式

virtual protected void panelOnPaint(PaintEventArgs e)

描述：绘制控件背景颜色

void drawBackGroundColor(PaintEventArgs e)

描述：绘制背景文本

void drawText(PaintEventArgs e)

描述：绘制背景图片

void drawBackGroundImage(PaintEventArgs e)

## SVBitmapArray类

针对SVBitmap自定义图片类的进一步分装，形成的图片数组类。目的是方便对于多个图片数据的保存和操作，并且在属性编辑图片数组的时候对用户的操作及显示方式进行限制。

## SVBitmapHead类

将图元文件数据转换为内存中的二进制数据。

## SVWPFBitmapManagerWindow类

图元管理窗口。包含图元的分类的添加、删除、重命名。图元的添加、删除、预览显示。

## SVWpfControl类

为了在自定义的控件中使用wpf类型的控件，该类作为一个中间层，在使用自定义的控件的时候内嵌wpf控件，使得winform和wpf控件可以一起在对话框中被使用。

## SVWpfColorDialog类

自定义的颜色选择对话框类，通过wpf生成颜色选择方框并设置外观样式。

## SVWPFLockDialog类

针对属性窗口中编辑锁定属性字段弹出的自定义对话框，通过锁图片的切换来改变锁定状态的值。

## SVWPFProgressBar类

在编译工程页面的时候，页面较多的情况会造成编译时间等待过长并且页面出现卡死的现象，这里采用多线程的方式弹出自定义的进度条来显示编译进度，解决界面卡死等现象。

## SVWPFSettingWindow类

使用wpf重写设置窗口界面。

## SVBuildFile类

对下装文件原始数据文件头的包装，用来生成和读取该文件头及文件内容信息。文件头中包含文件标识字符串，文件的版本号，文件的长度，文件数据CRC校验，图片数据的起始和偏移。便于生成和验证下装文件的合法性。

## SVLog类

全局的日志记录类，用来记录当前用户操作记录，错误提示，软件异常等日志内容。日志类型分为：消息、警告、错误、严重崩溃四个等级。日志的记录方式包括：界面输出窗口显示及文本文件。包含TraceLog类，该类的具体操作如下：

以消息的方式记录日志

Message 表示当前日志信息

public void Info(String message)

以警告等级记录日志

Message 表示当前日志信息

public void Warning(String message)

以错误等级记录日志

Message 表示当前日志信息

public void Error(String message)

以程序崩溃记录日志，当发生记录该日志的时候会抛投一个异常

Message 表示当前日志信息

public void Critical(String message)

对当前的异常信息进行记录，记录异常的详细信息

Ex 异常对象

public void Exception(Exception ex)

## SVTextBoxTraceListenter类

继承自TraceListener自定义类，定义当前日志在界面窗口中显示的格式。

## SVTextTraceListenter类

继承自TraceListener自定义类，定义当前日志信息在文本文件中存储的格式，如添加日志的时候自动添加当前时间、当前日志等级信息、日志内容保存、日志记录方式（防止日志文件过大，以单天一个日志文件）等。

## SVMessageBox类

自定义的MessageBox类，由于系统需要进行中英文切换，及弹出的对话框中的语言也要进行切换。由于系统自带的MessageBox类，不能随着语言环境进行变化。重新定义一个来代替以实现语言切换功能。

## SVSqlDataBase类

数据库交互类，该类只针对Mysql数据库操作进行封装。封装了与数据库中指定业务数据交互的接口。

参数：

Ip 数据连接地址

Database数据库连接名称

User 连接用户名

Passwd 连接密码

Charset 设置字符集，默认为utf-8

返回：成功返回True，失败返回False

描述：数据库连接函数，连接成功才能继续操作

public Boolean connect(String ip, String database, String user, String passwd, String charset)

参数：无

返回：当前生成svdu下装文件目录

描述：从数据库表格中获取当前下装文件生成目录，如果数据库没有打开，或者数据库表中存在错误，将返回null

public String getProPath()

参数：stationID 站号

返回：返回当前站点中的变量信息

DataTable c#语言中一个表格对象，用来保存变量表信息。

描述：根据站号来获取当前表格中变量信息。包括变量名称、变量的发送和接收偏移地址、变量的类型。

public DataTable getVarDataList(Int32 stationID)

参数：无。

返回：void

描述：关闭数据库连接。数据库类似文件操作，不能由系统自动回收。需要在关闭当前软件的时候释放资源，以避免系统资源泄漏。

public void Close()

# 控件类设计

由于页面中的控件具有相同的行为，比如拖拽，移动，改变尺寸，事件选择等。又存在不同行为，这里将控件采用继承的方式。将相同行为的内容放置在一个基类中处理，然后分别继承以实现不同控件的行为模式。由于每一种控件都有保存xml和读取xml，生成下装配置文件的需求，这里需要分别继承自接口SVInterfacePanel和SVInterfaceBuild。



## SVListView类

继承自ListView控件，打开列表项的拖拽功能，在拖拽过程中添加附加数据，使其满足创建自定义控件。每一项都有单独可识别的图片标识及单独的拖拽事件。

参数：void

返回：void

描述：初始化控件功能，设置列表项的图片、默认值、根据每一项的索引号来注册拖拽过程中要执行的函数委托(即不同的索引，拖拽后将会产生不同的控件对象并进行传递)。

void initControl()

描述：鼠标执行拖拽后事件的触发函数，如果当前索引在已经注册的合法范围内，将执行对应的操作，否则将不会发生响应。

void slotItemDrag(object sender, ItemDragEventArgs e)

## SVAnalogProperties类

模拟量控件的属性类，该类记录了当前模拟量控件的所有属性字段。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前模拟量的ID号 |
| 2 | \_max | Single | 当前模拟量的最大值 |
| 3 | \_min | Single | 当前模拟量的最小值 |
| 4 | \_font | Font | 当前模拟量显示的字体 |
| 5 | \_normalColor | Color | 正常显示颜色 |
| 6 | \_overMaxClr | Color | 超出最大值范围的颜色显示 |
| 7 | \_overMinClr | Color | 低于最小值范围的颜色显示 |
| 8 | \_normalBgColor | Color | 正常显示背景颜色 |
| 9 | \_overMaxBgClr | Color | 超出最大值范围的背景颜色显示 |
| 10 | \_overMinBgClr | Color | 低于最小值范围的背景颜色显示 |
| 11 | \_decNum | Byte | 小数点的显示位数 |
| 12 | \_rect | Rectangle | 当前模拟量的位置尺寸 |
| 13 | \_var | String | 模拟量关联的变量名称 |
| 14 | \_isExponent | Boolean | 是否以指数显示 |
| 15 | \_controlType | String | 控件显示属性 |
| 16 | \_exceptionColor | Color | 出现异常后的字体显示颜色 |
| 17 | \_exceptionBgColor | Color | 出现异常后背景显示颜色 |
| 18 | \_isLock | Bool | 当前控件是否锁定，不能移动 |

当该属性类中的字段发生改变，UpdateControl委托将会被执行。该委托的作用是将当前改变发生进行记录和更新控件显示。

参数：pageArrayBin当前下装文件格式结构

Serialize当前下装格式图片对象

返回：void

描述：作用是将当前模拟量属性转换为下装配置文件格式中的数据对应值。

public void make(ref PageArrayBin pageArrayBin, ref SVSerialize serialize)

## SVAnalog类

模拟量控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVBinaryProperties类

开关量控件的属性字段。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前开关量的ID号 |
| 2 | \_trueColor | Color | 值为真时的颜色显示 |
| 3 | \_falseColor | Color | 值为假时的颜色显示 |
| 4 | \_font | Font | 当前模拟量显示的字体 |
| 5 | \_trueBgColor | Color | 值为真时的背景颜色显示 |
| 6 | \_falseBgColor | Color | 值为假时的背景颜色显示 |
| 7 | \_rect | Rectangle | 开关量的控件位置尺寸 |
| 8 | \_type | String | 格式显示设置 |
| 9 | \_controlType | String | 控件背景显示类型 |
| 10 | \_exceptionColor | Color | 出现异常显示颜色 |
| 11 | \_exceptionBgColor | Color | 出现异常背景显示颜色 |
| 12 | \_customTrueText | String | 为真自定义文本 |
| 13 | \_customFlaseText | String | 为假自定义文本 |
| 14 | \_isLock | Bool | 是否锁定当前控件不被移动 |
| 15 | \_truePicture | SVBitmap | 值为真的背景图片 |
| 16 | \_flasePicture | SVBitmap | 值为假的背景图片 |
| 17 | \_exPicture | SVBitmap | 真异常背景图片 |

参数：pageArrayBin当前下装文件格式结构

Serialize当前下装格式图片对象

返回：void

描述：作用是将当前开关量属性转换为下装配置文件格式中的数据对应值。

public void make(ref PageArrayBin pageArrayBin, ref SVSerialize serialize)

## SVBinary类

模拟量控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVBinaryTypeConverter类

开关量中自定义类型属性选择器，作用于\_type属性字段，限定用户界面开关量的类型选择必须在合法的范围内。目前开关量的类型选择范围如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 类型表示 | 值 |
| 1 | 文本显示 | 0 |
| 2 | 图片显示 | 1 |

## SVButtonProperties类

Button属性类，记录当前Button所有相关属性数据。是与用户界面交互、生成下装文件格式的一个中间层。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前按钮的ID号 |
| 2 | \_bgcolor | Color | 背景颜色 |
| 3 | \_bgDownColor | Color | 按下去的背景显示颜色 |
| 4 | \_font | Font | 文本字体 |
| 5 | \_fgcolor | Color | 按钮文本颜色 |
| 6 | \_text | Color | 按钮文本内容 |
| 7 | \_rect | Rectangle | 按钮位置尺寸 |
| 8 | \_isConfirm | Boolean | 该按钮是否有二次确认对话框 |
| 9 | \_controlType | String | 控件显示类型 |
| 10 | \_btnType | SVBtnTypeConverter | 按钮的类型选择，主要分为跳转页面和操作变量 |
| 11 | \_isShowPic | Boolean | True表示显示背景图片，false表示显示背景颜色值 |
| 12 | \_btnDownPic | SVBitmap | 按下去后显示的背景图片 |
| 13 | \_btnUpPic | SVBitmap | 按钮弹起后显示的背景图片 |
| 14 | \_isLock | Bool | 是否锁定当前控件不被移动 |
| 15 | \_btnType | SVBtnTypeConverter | 按钮类型，包括页面跳转和设备控制变量 |
| 16 | \_fText | String | 值为假的时候文本内容 |
| 17 | \_fMemo | String | 按钮的备注信息 |

## SVButton类

按钮控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

按钮控件进行了重新绘制，以下是单独的绘制接口函数：

描述：用来绘制按钮在没有任何操作的情况下，按钮外观显示。

protected void drawButtonNormal(PaintEventArgs e)

描述：用来绘制按钮处于按下状态时的外观显示。

protected void drawButtonDown(PaintEventArgs e)

描述：当控件处于自绘制的时候，文本也需要自己重新绘制，该函数为通过自定义的方式显示文本。

protected void drawText(PaintEventArgs e)

描述：编译之前按钮相关内容的合法性检查

public override void checkValid()

## SVBitmapTypeEditor类

统一图标选择方式。便于在属性窗口中编辑相关的图片属性，重载UITypeEditor基类，以窗口弹出SVBitmapManagerWindow类型的对话框，用于选择图标库中的相关图标与当前属性关联。

## SVBitmapManagerWindow类

自定义图标选择窗口类，首先是统一图标的用户选择方式；其次是添加部分图标管理功能，方便用户在选择图标库中的图标时，可以方便的添加新增加图标，增强用户的交互方式，减少操作过程。

该窗口的功能主要包含：

* 管理图标分类，包括图标分类的创建、修改名称、删除（分类删除对应分类中的图标也会一并被删除）；
* 图标分类的重命名功能；
* 用户选择图片添加或者批量选取图片进行添加；
* 图标项的重命名及删除功能；
* 对图标项名称的重复性检查；
* 显示分类列表中图标项列表和缩略图显示，显示当前选中图标的详细信息及图标预览。

## SVBtnTypeConverter类

按钮的类型申明类，一种自定义类型，用来记录当前按钮的状态及对应的值，区分按钮处于不同类型的时候，记录当前按钮关联的页面相关信息。用于提供生成下装文件之前必要的信息来源。

## SVBtnTypeEditor类

用于在属性窗口与按钮动作属性相关联中，对于按钮的类型字段的选择，采用自定义的方式弹出对话框。

## SVCurveProperties类

趋势图属性类，包含了当前趋势图控件拥有的所有属性字段。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前趋势图的ID号 |
| 2 | \_bgcolor | Color | 背景颜色 |
| 3 | \_fgcolor | Color | 趋势图显示颜色 |
| 4 | \_font | Font | 文本字体 |
| 5 | \_max | Single | 纵坐标最大值 |
| 6 | \_min | Single | 纵坐标最小值 |
| 7 | \_rect | Rectangle | 按钮位置尺寸 |
| 8 | \_interval | UInt16 | 趋势图的横坐标范围 |
| 9 | \_controlType | String | 控件显示类型 |
| 10 | \_varArray | String[] | 趋势图关联变量数组 |
| 11 | \_varColorArray | Color[] | 趋势图线条颜色数组 |
| 12 | \_lineEnabled | Byte[] | 趋势图变量数组使能标志 |
| 13 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVCurve类

趋势图控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

对趋势图控件进行了重新绘制，以下是单独的绘制接口函数：

描述：为保持与下位机运行结果一致，这里对趋势图控件的外观进行了重绘。

protected void drawCurve(PaintEventArgs e)

## SVGifProperties类

动态图属性类。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前动态图的ID号 |
| 2 | \_rect | Rectangle | 位置尺寸 |
| 3 | \_var | String[] | 关联的变量数组 |
| 4 | \_pic | SVBitmapArray | 图标数组 |
| 5 | \_picError | SVBitmap | 发生错误时候对应的图标 |
| 6 | \_controlType | String | 显示控件类型 |
| 7 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVGif类

动态图控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVGifWindow类

用来编辑当前动态图数组的对话框。

## SVIconProperties类

静态图属性字段如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前模拟量的ID号 |
| 2 | \_rect | Rectangle | 位置尺寸 |
| 3 | \_iconData | SVBitmap | 关联的变量数组 |
| 4 | \_controlType | String | 显示图标 |
| 5 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVIcon类

静态图控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVLabelProperties类

文本控件对应属性字段，在属性窗口中查看或者编辑的所有值。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前文本控件的ID号 |
| 2 | \_rect | Rectangle | 位置尺寸 |
| 3 | \_controlType | String | 显示图标 |
| 4 | \_bgcolor | Color | 文本背景显示颜色 |
| 5 | \_fgcolor | Color | 文本显示颜色 |
| 6 | \_font | Font | 文本字体 |
| 7 | \_align | String | 文本的对齐方式 |
| 8 | \_transparent | Boolean | 文本背景是否透明 |
| 9 | \_text | String | 文本内容 |
| 10 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVLabel类

文本控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVLineProperties类

线条属性，在属性窗口中仅显示当前当前线条属性中的字段。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前线条控件的ID号 |
| 2 | \_lineColor | Color | 记录当前线条的颜色 |
| 3 | \_lineWidth | Byte | 当前线条的宽度（默认值为3） |
| 4 | \_lineLen | UInt16 | 线条的长度 |
| 5 | \_showType | Boolean | 线条显示类型，true-横向，false-纵向 |
| 6 | \_startPos | Point | 线条的起始位置 |
| 7 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVLine类

线条控件类，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。

## SVHeartbeatProperties类

心跳包控件，以下为其包含属性：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 描述 |
| 1 | \_id | UInt16 | 当前线条控件的ID号 |
| 2 | \_bitMapArray | SVBitmapArray | 心跳包控件包含的图标数组 |
| 3 | \_rect | Rectangle | 位置尺寸 |
| 4 | \_showType | Boolean | 线条显示类型，true-横向，false-纵向 |
| 5 | \_isLock | Bool | 是否锁定当前控件不被移动 |

## SVHeartbeat类

心跳包控件，继承自SVPanel。对SVPanel抽象类的具体实现，实现抽象类中定义的抽象接口，具体抽象接口详见SVPanel类说明。用于展示当前系统的运行状态。

## SVWPFAlignDialog类

自定义的文本对齐属性选择对话框，采用wpf技术编写，布局了多个按钮来反映文本对齐方式。

## SVWPFBtnMemoEdit类

自定义的按钮备注信息修改对话框，采用wpf技术编写。

## SVWPFBtnTextEdit类

自定义文本修改对话框，采用wpf技术编写。

## SVWPFFontDialog类

属性窗口中，字体属性选择自定义对话框类。

## SVWPFHeartPic类

属性窗口中心跳控件图片选择对话框，有添加和删除功能。

## SVWPfIconPic类

属性窗口中静态图图片选择窗口，采用wpf技术编写。

## SVWPFVariableDialog类

变量选择自定义窗口。

## WPFWidgetBackGround类

页面背景设置对话框，有颜色背景和图片背景两种选择方式。

# 全局数据结构设计

## 数据结构定义

SVDU开发环境与下位机的交互主要是体现在一个公共的数据结构，双方通过统一的格式及格式中内容的范围限定，开发环境通过对结构中相关数据的填充，下位机获取该数据结构后按相同的规则进行解析显示及关联相关操作。该数据结构SVDU开发环境和运行环境之间通信接口桥梁，起着非常重要的作用。以下是对该结构在开发环境端展现形式的详细描述。所有关于该结构的设计都体现在SVBinData.cs文件中。

### 公共数据结构

对最大页面以及各个类型控件的数量控制。

public const Int32 MAX\_PAGE\_NUMBER = 800;

public const Int32 PAGE\_TITILE\_MAXLEN = 128 \* 2;

public const Int32 PAGE\_BTN\_MAXNUM = 80;

public const Int32 PAGE\_AREA\_MAXNUM = 150;

public const Int32 PAGE\_ICON\_MAXNUM = 150;

public const Int32 PAGE\_LINE\_MAXNUM = 80;

public const Int32 PAGE\_GIF\_MAXNUM = 50;

public const Int32 PAGE\_BOOL\_MAXNUM = 40;

public const Int32 PAGE\_ANA\_MAXNUM = 40;

public const Int32 PAGE\_TCHART\_MAXNUM = 6;

public const Int32 PAGE\_TICKGIF\_MAXNUM = 1;

public const Int32 TEXT\_MAX\_LEN = 64 \* 2;

public const Int32 BTN\_MAX\_LEN = 64;

public const Int32 BINARY\_MAX\_LEN = 64;

public const Int32 AREA\_MAX\_LEN = 64 \* 2;

public const Int32 TREND\_LINE\_MAX\_NUM = 4;

### 矩形区域数据结构

矩形区域是指在显示器区域中的位置区域。以下结构中描述了矩形区域的左上角和右下角的坐标点。

public struct RectBin

{

public UInt16 sX;

public UInt16 sY;

public UInt16 eX;

public UInt16 eY;

}

### 线数据结构

U开发环境支持水平和垂直直线的绘制。对应的线段数据结构为，采用默认4字节：

public struct LineBin

{

public UInt32 color; //线条颜色值

public UInt16 id; //id号

public UInt16 x1; //起始点x坐标

public UInt16 y1; //起始点y坐标

public UInt16 x2; //结束点x坐标

public UInt16 y2; //结束点y坐标

public Byte width; //线条宽度

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 1)]

public Byte[] rsv;

}

### 静态文本数据结构

S-VDU开发环境中文本控件，拥有对文本位置、文本大小、文本背景颜色、文本背景颜色是否透明、文本颜色、对齐方式及字体选择的输入和编辑功能。

public struct AreaBin

{

public RectBin rect;

public UInt32 bgClr;

public UInt32 fontClr;

public UInt16 id;

public Byte[] text;

public Byte font;

public Byte align;

public Byte transparent;

public Byte[] rsv;

}

### 按钮数据结构

S-VDU开发环境软件中，具有对按钮位置、按钮大小、按钮文本、按钮的背景颜色、按钮的按下颜色、按钮的类型选择、按钮的按下显示背景图片、按钮弹起时显示背景图片等编辑功能。

[Serializable]

[StructLayout(LayoutKind.Explicit,Pack = 4)]

public struct btnParaBin

{

[FieldOffset(0)]

public UInt32 pageId;

[FieldOffset(0)]

public UInt32 addrOffset;

}

public struct ButtonBin

{

public RectBin rect;

public UInt32 fontClr;

public UInt32 bgUpColor;

public UInt32 bgDownColor;

public btnParaBin param;

public UInt32 enableAddrOffset; //使能变量地址

public UInt16 id;

[MarshalAs(UnmanagedType.ByValArray,SizeConst = SVLimit.TEXT\_MAX\_LEN)]

public Byte[] text;

public Byte[] fText; //为假

public Byte[] fMemo; //备注信息

public Byte type;

public Byte confirm;

public Byte font;

public Byte bgUpFlag; //0：使用颜色填充 1：图片填充

public Byte bgDownFlag; //0：使用颜色填充 1：图片填充

public Byte enable; //是否使能变量，0：否，1：是

public Byte varTypeEn; //使能变量类型

public Byte varTypeBtn; //关联变量的类型

}

### 模拟量显示数据结构

S-VDU开发环境软件支持对模拟量控件位置、大小、最大最小值、字体、精度、设置不同条件下的背景显示颜色进行编辑。

public struct AnalogBin

{

public RectBin rect;

public UInt32 addrOffset;

public UInt32 normalClr;

public UInt32 overMaxClr;

public UInt32 overMinClr;

public UInt32 normalBgClr;

public UInt32 overMaxBgClr;

public UInt32 overMinBgClr;

public UInt32 vinfoInvalid;

public UInt32 vinfoInvalidBg;

public Single vMax;

public Single vMin;

public UInt16 id;

public Byte font;

public Byte nDecimalNum;

public Byte enExponent;

public Byte varType; //变量类型

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]

public Byte []rsv;

}

### 数字量显示数据结构

S-VDU开发环境对数字量控件位置、大小、字体、类型、不同条件下背景颜色的显示编辑。

public struct BinaryBin

{

public RectBin rect;

public UInt32 addrOffset;

public UInt32 trueClr;

public UInt32 trueBgClr;

public UInt32 falseClr;

public UInt32 falseBgClr;

public UInt32 vinfoInvalid;

public UInt32 vinfoInvalidBg;

public Byte[] trueText; //值为真的时候文本

public Byte[] falseText; //值为假的时候文本

public UInt16 id;

public Byte font; //文本显示字体

public Byte type; //0：自定义文本，1：自定义图片

public Byte varType; //关联的变量类型

}

### 趋势图数据结构

S-VDU开发环境软件中支持对趋势图位置、大小、横纵坐标范围限制、趋势图背景及前景颜色的设置、趋势图要显示的变量及线条颜色等的编辑功能。

public struct TrendChartBin

{

public RectBin rect;

public UInt32 bgClr;

public UInt32 scaleClr;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.TREND\_LINE\_MAX\_NUM)]

public UInt32[] addrOffset; //关联的变量

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.TREND\_LINE\_MAX\_NUM)]

public UInt32[] lineClr; //每个变量对应的颜色显示

public Single yMin; //纵轴最小值

public Single yMax; //纵轴最大值

public UInt16 maxTime; //趋势图的最大时间，单位：秒，范围：60-3600

public UInt16 stepTime; //时间的步进值

public UInt16 id;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.TREND\_LINE\_MAX\_NUM)]

public Byte[] lineWidth; //线条的使能标志，0：禁能，1：使能

public Byte[] varType; //关联变量的类型

public Byte font; //趋势图相关字体设置

}

### 静态图数据结构

S-VDU开发环境可以编辑静态图的位置、大小和要显示的图片。

public struct IconBin

{

public RectBin rect; //显示范围

public UInt32 imageOffset; //图片地址

public UInt16 id; //图片控件ID号

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]

public Byte[] rsv;

}

### 动态图数据结构

S-VDU开发环境可以支持动态图位置、大小、类型、变量设置及对应图片的显示等。

public struct GifBin

{

public RectBin rect; //显示范围

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]

public UInt32[] addOffset; //变量数组地址

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]

public UInt32[] imageOffset; //图片数组地址

public UInt32 iamgeOffsetErr; //异常图片地址

public UInt16 id;

public Byte type;

public Byte[] varType; //变量数组类型

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 1)]

public Byte[] rsv;

}

### 心跳控件数据结构

心跳控件用来表示当前系统的运行状态，每一个页面最多只能存放一个心跳控件对象。

public struct TickBin

{

public RectBin rect;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]

public UInt32[] imageOffsetList; //图片地址数组

public UInt16 id; //控件ID号

public Byte count; //图片个数

}

## 页面数据结构

当前每个页面的数据结构描述，包含了页面中每种控件元素的个数及控件属性内容，拥有绘制和操作一个页面的所有信息。

public struct PageBin

{

public UInt32 bgClr;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_AREA\_MAXNUM)]

public AreaBin[] m\_area;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_BTN\_MAXNUM)]

public ButtonBin[] m\_btn;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_LINE\_MAXNUM)]

public LineBin[] m\_line;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_ANA\_MAXNUM)]

public AnalogBin[] m\_analog;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_BOOL\_MAXNUM)]

public BinaryBin[] m\_binary;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_TCHART\_MAXNUM)]

public TrendChartBin[] m\_trendChart;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_ICON\_MAXNUM)]

public IconBin[] m\_icon;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = SVLimit.PAGE\_GIF\_MAXNUM)]

public GifBin[] m\_gif; //动态图数据

public TickBin[] m\_tick; //心跳控件数据

public UInt16 id;

public UInt16 index;

public Byte pointAlign; //小数点对齐, 范围：0-6

public Byte btnNum; //按钮个数

public Byte areaNum; //文本个数

public Byte iconNum; //静态图个数

public Byte lineNum; //线条个数

public Byte gif\_num; //动态图个数

public Byte analog\_num; //模拟量个数

public Byte binaryNum; //开关量个数

public Byte trendChartNum; //趋势图个数

public Byte tickNum; //心跳控件个数

public Byte bgSet; //0：表示背景按照颜色解析，1：背景按照图片地址解析

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]

public Byte[] rsv;

}

## 页面配置文件数据结构

下装文件对应的数据结构描述，是对所有页面的整合，也是上位机和下位机进行的交互手段，目前为节省存储空间，文件的生成依赖于当前的页面个数，根据当前页面个数的大小动态的调整生成的文件大小，便于该文件发往下位机的时候减少文件存储体积。

public struct PageArrayBin

{

public static readonly PageArrayBin Empty;

public UInt32 pageCount; //有效页面个数

public PageBin[] pageArray; //有效页面数据

}

## 图元文件数据结构

目前图元的存储格式采用标准的BMP位图文件格式，位数为8、无压缩的存储方式。在实际使用过程中可能会涉及多个图元文件的绘制显示，所以这里采用多个位图文件紧密排列存储，只需将偏移地址计算保存到与图元显示相关的控件中。默认的BMP位图文件，数据是按照倒着的方式存储，显示需要将数据翻转180°后才能显示正常，为节约下位机的处理过程，目前在生成图元文件的时候，所有的图片数据已经经过一次翻转处理。



图元文件结构

## 相关处理函数

参数:： in bin当前配置文件数据结构内存块

out result 转换后的内存结构

返回：void

描述：将当前配置文件的结构体转换为可以保存为文件的内存数据

static public void structToByteArray(PageArrayBin bin, ref byte[] result)

参数:： in result文件内存结构

out bin页面数组可识别的结构

返回：void

描述：读取文件数据后，转换为可是别的结构体数据

static public void byteArrayToStruct(byte[] result, ref PageArrayBin bin)

# 总体设计

## 需求概述

S-VDU是核电厂安全级显示单元的简称，是核电站安全级平台的一个组成部分。开发环境软件组态软件是针对S-VDU运行的配套软件，用来在下装模式中，用户根据需要配置S-VDU的输入命令及显示参数，是用户管理和配置S-VDU安全显示单元功能的入口点。



图SVDU开发环境软件所在位置



软件主要功能

## 软件结构

S-VDU开发环境软件按代码组织结构分为SVCore、SVPro、SVControl、SVSimulation四个主要模块，分别对应一个dll或者exe文件。

SVCore为基础模块，包含软件中涉及到的一些通用的类，目的是为上层功能模块服务，该模块中包含了通用基础数据结构、模型及界面。

SVPro为软件中的主模块，包含主程序的入口点以及整合所有相关子功能模块类，根据功能需求控制整个操作和数据的流程。

SVControl为页面以及控件的创建、管理模块。包含了控件自身功能、控件显示和控件数据管理，提供控件数据修改的相关辅助类。使数据操作和显示分离，降低耦合度，方便将来扩展与维护。

SVSimulation为仿真模块，主要是通过反解析程序生成的下装文件，在窗口中重新绘制页面相关元素并按预期展示其操作。提供对下装文件的验证的一种方式，方便用户对所生成的下装文件功能有直观了解。



图各模块之间的依赖关系

## 界面整体设计

S-VDU开发环境软件主界面设计分为以下几个部分：菜单栏、工具栏、工程管理窗口、控件窗口、工作区窗口、属性窗口、对象窗口、输出窗口、查找窗口、状态栏窗口。软件中主界面显示的所有窗口采用Dock的方式进行管理，采用类似VS的管理方式，方便用户通过鼠标拖拽就可以很容易的布局每一个子窗口的显示方式，提高用户操作体验。

菜单栏按照分类包含了软件中所有操作功能，对常用功能提供快捷键的操作方式；

工具栏中包含常用操作的快捷入口，通过分隔符进行归类并对每个不同的功能添加图标显示；

工程树窗口中的树节点展示的是当前工程名相关节点以及管理的页面节点，提供右键快捷菜单来快速执行页面的部分功能，包括新建页面、删除页面、移除页面、导入页面、打印页面、页面检查等；

控件窗口包含了所有的基础控件，提供拖拽的方式将这些控件添加到页面窗口中；

属性窗口显示当前被选中的控件及页面属性内容，该窗口中的内容是用来显示当前属性值，并提供对部分属性的编辑功能，属性编辑成功后，控件要及时刷新显示状态；

对象窗口目的是解决在页面中对多个控件对象进行编辑的时候，可能会出现多个控件发生重叠情况，鼠标难以选中被覆盖的控件，这个时候可以通过对象窗口中列出的对象节点执行双击以使该控件在页面中置于顶层显示；

输出窗口用来显示操作过程中的重要信息的提示；

查找窗口提供用户的查找结果输出，并且有双击查找结果显示的相关信息定位到相关页面的功能；

状态栏中显示一些常用命令的信息状态。

## 模块详述

### 工程管理模块

工程管理模块包含新建工程、打开工程、保存工程和关闭工程。涉及的模块包括SVCore和SVPro。由于工程管理部分是由其他程序调用的功能，软件本身并不会直接在界面中提供操作入口，菜单中无需对该功能进行体现。

#### 新建工程

* 概述

新建工程表示一个项目的开始，主要工作是为后续操作做一些初始化的工作。将执行以下步骤，工程文件夹的创建、模板文件夹创建、图标管理文件夹创建、工程文件创建、工程文件路径记录、工程配置文件初始化创建。

* 输入数据

工程文件名。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 打开工程

* 概述

打开工程主要是加载工程文件信息和配置信息，并在导航树中添加工程管理节点。

* 输入数据

工程文件。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 保存工程

* 概述

将当前工程文件内容及对应的页面数据以xml的方式保存到指定目录的磁盘文件上。对于模板文件以及图元管理数据，目前是采用在操作过程中实时保存文件的方式，所以在保存工程的时候不在执行这些操作。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 关闭工程

* 概述

主要执行的内容为保存数据及释放界面相关资源。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程

执行保存工程操作；

清理界面相关资源。

### 页面管理模块

#### 新建页面

* 概述

新建页面以提供空白和模板方式创建页面。

* 输入数据

新建的页面名称；

对应的模板文件。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 打开页面

* 概述

将页面数据在工作区中显示。

* 输入数据

选中的页面节点。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 保存页面

* 概述

将当前正在编辑的页面执行保存。页面中都有一个是否有修改的标志，如果该标志为true的时候，则进行页面的保存操作。

* 输入数据

当前选中的页面对象。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 关闭页面

* 概述

将当前显示的页面，从界面中移除，并将该页面的属性从属性窗口中移除。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

#### 导出模板

* 概述

主要功能是将当前的页面以模板的形式进行保存。

* 输入数据

当前的页面数据；

新建的模板名称。

* 输出数据

模板文件及缩略图文件。

* 算法

无。

* 逻辑流程

无。

### 页面编辑

#### 左对齐

* 概述

将当前页面中选中的所有控件对象，按位置执行左对齐操作。

* 输入数据

当前选中控件。

* 输出数据

无。

* 算法

在所有选中控件中，遍历获取x轴方向最小值，在y轴坐标保持不变的情况下，将所选控件坐标全部设定为当前最小x值。

* 逻辑流程

无。

#### 右对齐

* 概述

对所有选中控件执行右对齐操作。

* 输入数据

无。

* 输出数据

无。

* 算法

在所有选中控件中，遍历获取x和宽度之和的最大值，在y轴坐标保持不变的情况下，将所选控件x坐标加自身宽度设为该值。

* 逻辑流程

无。

#### 上对齐

* 概述

将当前页中所选控件执行上对齐操作。

* 输入数据

无。

* 输出数据

无。

* 算法

遍历当前所有选中控件中的y轴最小值，将所有控件当前坐标y轴设定为该值，重新刷新界面。

* 逻辑流程

无。

#### 下对齐

* 概述

将当前页面中选中的控件执行下对齐操作。

* 输入数据

无。

* 输出数据

无。

* 算法

遍历当前页面中选中控件的y轴坐标和高度之和的最大值，将这些控件y轴和高度之和全部设置为此值，刷新界面即可。

* 逻辑流程

无。

#### 水平居中

* 概述

将多个控件执行水平居中操作。

* 输入数据

无。

* 输出数据

无。

* 算法

取当前所有控件y轴坐标的最大和最小值求平均值，在保持x轴不变的情况下，将所有控件的y坐标全部设置为此平局值。

* 逻辑流程

无。

#### 垂直居中

* 概述

将多个控件执行垂直居中操作。

* 输入数据

无。

* 输出数据

无。

* 算法

取当前所有控件x轴坐标的最大和最小值求平均值，在保持y轴不变的情况下，将所有控件的x坐标全部设置为此平均值。

* 逻辑流程

无。

#### 水平等间距

* 概述

将当前页面中所选控件执行水平等间距。

* 输入数据

无。

* 输出数据

无。

* 算法

将当前所有控件取x坐标轴的最小值及控件右侧最大值求差，将该差值与所有控件宽度之和再次求差值，如果为负数，就将所有控件从左到右紧密排列。反之计算它们之间的平均距离后排列。

* 逻辑流程

无。

#### 垂直等间距

* 概述

将当前页面中所选控件执行垂直等间距功能。

* 输入数据

无。

* 输出数据

无。

* 算法

将当前所有控件取y坐标轴的最小值及控件底端最大值求差，将该差值与所有控件高度之和再次求差值，如果为负数，就将所有控件按照从上到下的顺序紧密排列。反之计算它们之间的平均距离后排列。

* 逻辑流程

无。

#### 复制功能

* 概述

将当前页面中所选控件执行复制操作。

* 输入数据

无。

* 输出数据

无。

* 算法

主要是将当前选中的控件列表中的元素，复制一个副本到拷贝列表中。

* 逻辑流程

无。

#### 剪切功能

* 概述

将当前页面中所选控件执行剪切操作。

* 输入数据

无。

* 输出数据

无。

* 算法

主要是将当前选中的控件列表中的元素，复制一个副本到拷贝列表中，并在当前页面窗口中将这些控件执行删除。

* 逻辑流程

无。

#### 粘贴功能

* 概述

将当前页面中所选控件执行剪切操作。

* 输入数据

无。

* 输出数据

无。

* 算法

遍历当前拷贝列表中的所有控件元素，将粘贴信号发出，只要在对应的页面窗口中关联该信号执行相应的操作即可，目前支持跨页面粘贴操作，多次执行粘贴。

* 逻辑流程

无。

#### 撤销功能

* 概述

对页面中的控件元素的操作进行记录，并执行撤销。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 恢复功能

* 概述

对页面中的控件元素的操作进行记录，并在执行撤销操作后可以恢复。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 控件选择

* 概述

对页面中的控件操作之前需要选中，这里具体描述控件的选取过程。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 控件移动

* 概述

控件移动，通过鼠标和键盘移动。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 控件属性修改

* 概述

针对不同控件有着不同的属性字段，控件被选中的同时在属性窗口中显示属性。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



### 编译功能

#### 页面检查

* 概述

检查当前页面中的内容是否合法。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 编译

* 概述

将当前工程中的所有页面及页面中控件元素编译为可以下装的二进制文件，最终生成一个包含有当前工程中所有页面数据及图元数据的文件，生成的文件与当前实际的页面文件个数有关，随着工程中页面的增加该文件也会随之变大。

* 输入数据

工程页面数据。

* 输出数据

配置文件。

* 算法

设置当前工程中的启动页面作为页面数组元素的第一位置；

循环遍历工程中的所有页面对象，执行页面对象的编译函数接口；

页面对象包含所有控件的编辑函数接口，依次调用即可，生成对应内存结构数据；

指定目录，将内存数据以文件方式进行保存。

* 逻辑流程

无。

#### 仿真

* 概述

解析编译生成的下装配置文件，将文件中的数据反解析后在窗口中进行绘制并设置相关事件关联函数，模拟下位机上的部分显示功能过程。

* 输入数据

配置文件。

* 输出数据

界面显示。

* 算法

反解析二进制下装配置文件；

根据文件中的信息创建相关页面及控件窗口，并设置相应外观；

添加定时器事件及鼠标操作事件；

读取相关范围信息，并设置定时器随机产生数据，模拟在控件中显示；

增加控制流程，在仿真过程中可以自动播放，也可暂停后手动输入数据查看控件反馈情况。

* 逻辑流程

无。

### 模板管理

#### 新建模板

* 概述

新建模板是根据当前页面文件中控件元素来保存相关信息。

* 输入数据

页面文件。

* 输出数据

模板文件和页面缩略图。

* 算法

将当前页面文件与页面缩略图保存到当前工程中模板文件夹中。

* 逻辑流程

无。

#### 模板改名

* 概述

对模板文件夹中的模板文件及缩略图进行改名操作。

* 输入数据

模板文件。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

#### 删除模板

* 概述

从磁盘中删除模板文件，更新界面显示。

* 输入数据

模板文件和缩略图文件。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

### 图元管理

#### 添加图元

* 概述

软件中所有用到的图片文件，需要首先导入到软件中才能够继续使用。图元文件在磁盘上名称以当前系统时间转换后的字符串命名。为了解决批量导入多个图元文件命名冲突的为问题，在图元文件名称生成的时候自动加入序号。最终图元名称为：系统时间+序号。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程



#### 重命名图元

* 概述

图元项名称的修改，只是针对图元管理文件中配置项进行更改，图元文件本身并不发生变化。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

#### 删除图元

* 概述

删除图元管理文件中的图元项，并删除对应的图元文件。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

#### 图元分类

* 概述

图元分类只是属于逻辑抽象，在图元管理文件中对当前所有图元进行分类。

* 输入数据

无。

* 输出数据

无。

* 算法

无。

* 逻辑流程

无。

### 变量管理

* 概述

目前的变量只是在软件中关联控件，即只有只读功能，不能进行修改和删除。所有用到的变量都是来源于数据库。最终将变量地址与控件进行关联并设置到下装文件中。从数据库中获取的变量字段包括：变量名、变量地址、发送区地址、接收区地址、类型、备注等显示信息。

* 输入数据

无。

* 输出数据

无。

* 算法

读取数据库中的表结构内存，缓冲到内存中表格结构中。方便根据指定条件过滤和查询符合条件的变量。

* 逻辑流程



### 数据库相关操作设计

* 概述

封装基础mysql数据库操作类，通过拼接字符串查询的方式，通过一个接口来获取多个表中字段信息。

* 输入数据

无。

* 输出数据

内存数据库表。

* 算法

无。

* 逻辑流程

无。

### 其他功能

#### 页面打印

* 概述

界面打印就是将当前页面控件以图片的形式发往打印机类执行。

* 输入数据

页面控件对象。

* 输出数据

控件的内存图片。

* 算法

无。

* 逻辑流程

无。

#### 日志记录

* 概述

日志以界面和文本的方式记录。界面上显示日志信息，能够及时提醒用户当前操作过程中及必要的信息提示。文本日志主要是记录整个操作流程以及错误发生记录异常信息。日志根据显示类别分为：消息、警告、错误、严重错误四个等级。严重错误即表示当前进程发生奔溃，只记录在文本文件中。

* 输入数据

无。

* 输出数据

日志文本文件。

* 算法

无。

* 逻辑流程

无。

#### 语言切换

* 概述

支持界面的中英文语言切换，需要中文和英文两套语言资源文件。

* 输入数据

无。

* 输出数据

无。

* 算法

程序启动加载配置文件，并根据配置文件来加载对应的语言环境；

当切换软件语言，更新配置文件内容。重启软件，以便重新加载语言包来实现语言切换功能。

* 逻辑流程

无。