杭州长川科技股份有限公司

Hangzhou Changchuan Technology Co., Ltd.

CTA8290DPlus 测试系统

API 说明书

V3.5

前言

适用对象

说明书适用于设备安装者、使用者和操作员。

符号约定

在文档中可能出现下列标识，代表的含义如下。

图标	定义
 危险	表示有高度潜在危险，如果不能避免，会导致人员伤亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 说明	表示是正文的附加信息，是对正文的强调和补充。
 窍门	表示能帮助您解决某个问题或节省您的时间。

修订记录

版本号	修订内容	软件版本	发布日期
V3.5	新增 HPS100 相关函数	1.0.0.7R5	2024.04
V3.4	在 CFVI2DCVI、CFVI8DCVI、CFVI16DCVI 和 CFHVI1DCVI 中新增 GetMeasResult 函数	1.0.0.7R2	2023.07
V3.3	● 新增 CTM2110_8 相关函数 ● 修改 FVI2-VIKR、FVI8-VIKR、FVI16-VIKR 函数	1.0.0.7R1	2023.06
V3.2	● CRS232：增加 Send 函数，修改 Read 函数 ● DIO：增加 ControlDriverMod 函数	1.0.0.7（S6）	2023.04
V3.1	增加 CFVI16-TMU 相关函数	1.0.0.7（S5）	2023.04
V3.0	软件版本更新	1.0.0.7（S2）	2023.02

法律声明

版权声明

商标声明

- 是杭州长川科技股份有限公司的商标或注册商标。
- 在本文档中可能提及的其他商标或公司的名称，由其各自所有者拥有。
- 本文档中所描述的内容不应被视作未经长川或其他所有者书面许可，以暗示、不反对或其他形式授予使用前述任何商标、注册商标、名称及标志的许可或权利。未经事先书面许可，任何人不得以任何方式使用长川及其他所有者的商标、注册商标、名称及标志。

责任声明

- 在适用法律允许的范围内，在任何情况下，本公司都不对因本文档中相关内容及描述的产品而产生任何特殊的、附随的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉、文档丢失或预期利益的损失进行赔偿。
- 本文档中描述的产品均“按照现状”提供，除非适用法律要求，本公司对文档中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证。

出口管制合规声明

长川遵守适用的出口管制法律法规，并且贯彻执行与硬件、软件、技术的出口、再出口及转让相关的要求。就本文档所描述的产品，请您全面理解并严格遵守国内外适用的出口管制法律法规。长川产品

的用途、销售区域和终端客户由您决定；但如双方拟开展的业务被确认违反相关出口管制法律法规，长川暂停交货和售后服务的行为不视为违约。

关于本文档

- 本文档中的产品外观和功能请以实际为准。
- 如果不按照本文档中的指导进行操作可能会导致财产损失或人员伤害。请务必仔细阅读本文档中的操作说明和指导，并严格按照本文档的指导进行操作。
- 本文档可能包含技术上不准确的地方、或产品功能及操作不相符的地方、或印刷错误。本公司将根据产品功能的增强或者改进而不定期更新本手册的内容以及本手册中涉及的软硬件产品；更新的内容将在本文档的新版本中加入，恕不另行通知。

目录

第一章 TheSoft

1.1 TestParam

1.1.1 LogResult（）

函数功能

给指定参数、指定工位、指定子单元数进行测试结果赋值。

使用说明

无。

函数原型

```
int LogResult(int nIndex, int nSite, int nSubUnit, double dResult)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。
nSite	指定工位，工位号从 0 开始计算，最大到 63。
nSubUnit	指定子单元数，子单元数从 0 开始计算，最大到 7。
dResult	测试结果。

返回值

类型	说明
int	● 0：成功。 ● -1：表示未开始测试 ● -2：设置的工位号小于 0 或者大于最大工位数。 ● -3：nSubUnit 大于最大 Subunit。 ● -4：参数索引大于当前测试函数的参数总数。

示例

```
double dResult = 5.5;
int nFlag = TheSoft.TestParam().LogResult(0,0,0,dResult);
```

1.1.2 LogResult（）

函数功能

给指定参数、指定工位、指定子单元数进行测试结果赋值。

使用说明

无。

函数原型

```
int LogResult(const string& strParamName, int nSite, int nSubUnit, double dResult);
```

参数说明

参数	说明
strParamName	指定参数名称。
nSite	指定工位，工位号从 0 开始计算，最大到 63。
nSubUnit	指定子单元数，子单元数从 0 开始计算，最大到 7。
dResult	指定工位的测试结果。

返回值

类型	说明
int	● 0：成功。 ● -1：表示未开始测试 ● -2：设置的工位号小于 0 或者大于最大工位数。 ● -3：nSubUnit 大于最大 Subunit。 ● -4：参数索引大于当前测试函数的参数总数。

示例

```
double dResult = 5.5;
const string strParamName = "Param";
int nFlag = TheSoft.TestParam().LogResult(strParamName,0,0,dResult);
```

1.1.3 LogResultAll（）

函数功能

给指定参数、指定工位个数、指定子单元数进行测试结果赋值。

使用说明

无。

函数原型

```
int LogResultAll(int nIndex, int nSize, int nSubUnit, double *pResult)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。
nSize	工位数，超过最大工位数的取最大工位数。
nSubUnit	指定子单元数，子单元数从 0 开始计算，最大到 7。
pResult	指定工位数的测试结果。

返回值

类型	说明
int	● 0：成功。 ● - 1：表示未开始测试 ● - 2：设置的工位号小于 0 或者大于最大工位数。 ● - 3：nSubUnit 大于最大 Subunit。 ● - 4：参数索引大于当前测试函数的参数总数。

示例

```
double dResult[2] = {5.5,6.5};
int nFlag = TheSoft.TestParam().LogResultAll(0,2,0,dResult);
```

1.1.4 LogResultAll（）

函数功能

给指定参数、指定工位个数、指定子单元数进行测试结果赋值。

使用说明

无。

函数原型

```
int LogResultAll(const string& strParamName, int nSize, int nSubUnit, double *pResult);
```

参数说明

参数	说明
strParamName	指定参数名称。
nSize	工位数，超过最大工位数的取最大工位数。
nSubUnit	指定子单元数，子单元数从 0 开始计算，最大到 7。
pResult	指定工位数的测试结果。

返回值

类型	说明
int	● 0：成功。 ● -1：表示未开始测试 ● -2：设置的工位号小于 0 或者大于最大工位数。 ● -3：nSubUnit 大于最大 Subunit。 ● -4：参数索引大于当前测试函数的参数总数。

示例

```
const string strParamName = "Param";
double dResult[2] = {5.5,6.5};
int nFlag = TheSoft.TestParam().LogResultAll(strParamName,2,0,dResult);
```

1.1.5 GetConditionIndex（）

函数功能

根据条件名称获取当前测试函数中设置的条件索引。

使用说明

无。

函数原型

```
int GetConditionIndex(const string& strConditionName);
```

参数说明

参数	说明
strConditionName	指设置的条件名称。

返回值

类型	说明
int	● 0：成功。 ● －1：未开始测试。 ● －2：nIndex 超出当前测试函数设置的条件总数或者 nIndex 设置错误，应为大于 0 的整数。

示例

```
int nRet = TheSoft.TestParam().GetConditionIndex("ConditionName")
```

1.1.6 GetConditionName（）

函数功能

获取条件名称的函数。

使用说明

无。

函数原型

```
string GetConditionName(int nIndex);
```

参数说明

参数	说明
nIndex	指定条件的序号，序号含义为该条件是当前函数的第 N 个条件，N 从 0 开始计数。

返回值

类型	说明
string	● 当获取测试函数条件名称成功时，返回条件名称字符串。 ● 当获取测试函数条件名称失败时，返回空字符串。

示例

```
string strConditionName = TheSoft.TestParam().GetConditionName(0);
```

1.1.7 GetConditionUnit（）

函数功能

获取条件单位的函数。

使用说明

无。

函数原型

```
string GetConditionUnit(int nIndex);
```

参数说明

参数	说明
nIndex	指定条件的序号，序号含义为该条件是当前函数的第 N 个条件，N 从 0 开始计数。

返回值

类型	说明
string	● 当获取测试函数条件名称成功时，返回条件名称字符串。 ● 当获取测试函数条件名称失败时，返回空字符串。

示例

```
string strUnit = TheSoft.TestParam().GetConditionUnit(0);
```

1.1.8 GetConditionUnit（）

函数功能

获取条件单位的函数。

使用说明

无。

函数原型

```
string GetConditionUnit(const string& strConditionName);
```

参数说明

参数	说明
strConditionName	条件名称。

返回值

返回值	说明
string	● 当获取测试函数条件名称成功时，返回条件名称字符串。 ● 当获取测试函数条件名称失败时，返回空字符串。

示例

```
string strUnit = TheSoft.TestParam().GetConditionUnit("Condition1");
```

1.1.9 GetConditionValue（）

函数功能

获取条件值的函数。

使用说明

无。

函数原型

```
string GetConditionValue(int nIndex);
```

参数说明

参数	说明
nIndex	指定条件的序号，序号含义为该条件是当前函数的第 N 个条件，N 从 0 开始计数。

返回值

类型	说明
string	● 当获取测试函数条件名称成功时，返回条件名称字符串。 ● 当获取测试函数条件名称失败时，返回空字符串。

示例

```
string strValue = TheSoft.TestParam().GetConditionValue(0);
```

1.1.10 GetConditionValue（）

函数功能

获取条件值的函数。

使用说明

无。

函数原型

```
string GetConditionValue(const string& strConditionName);
```

参数说明

参数	说明
strConditionName	条件名。

返回值

类型	说明
string	● 当获取测试函数条件名称成功时，返回条件名称字符串。 ● 当获取测试函数条件名称失败时，返回空字符串。

示例

```
string strValue = TheSoft.TestParam().GetConditionValue("Condition1");
```

1.1.11 GetCurrentFuncName（）

函数功能

获取当前测试函数的名称。

使用说明

无。

函数原型

```
string GetCurrentFuncName();
```

参数说明

无。

返回值

类型	说明
string	当前测试函数的名称。

示例

```
string strCurFuncName = TheSoft.TestParam().GetCurrentFuncName();
```

1.1.12 GetGPResult（）

函数功能

获取 VI 源 Group 的指定 Pin 名称，指定 Site，测试结果。

使用说明

目前支持 FVI2，FVI8，FVI16。

函数原型

```
int GetGPResult(const string& strPinName, int nSite, double &dResult);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSite	指定工位，工位号从 0 开始。
dResult	返回被测结果。

返回值

类型	说明
int	● 0：正常。 ● －1：Pinname 不存在或者 Pin 对应的板卡类型不支持。

示例

```
double dResult = 0.0;
const string strPinName = "Param";
int nFlag = TheSoft.T
estParam().GetGPResult(strPinName,0,dResult);
```

1.1.13 GetGPResultAll（）

函数功能

获取 VI 源 Group 的指定 Pin 名称，指定工位数的测试结果。

使用说明

目前支持 FVI2，FVI8，FVI16。

函数原型

```
int GetGPResultAll(const string& strPinName, int nSize, double *Result);
```

参数说明

参数	说明
strPinName	Pin 的名字。
nSize	工位数。
Result	返回各工位测试结果。

返回值

类型	说明
int	● 0：正常。 ● －1：Pinname 不存在或者 Pin 对应的板卡类型不支持。

示例

```
double dResultAll[2] = {0};
int nRet = TheSoft.TestParam().GetGPResultAll("pin1", 2, dResultAll);
```

1.1.14 GetParamAverage（）

函数功能

获取参数平均值。

使用说明

无。

函数原型

```
double GetParamAverage(int nFunIndex, int nParamIndex)
```

参数说明

参数	说明
nFunIndex	函数序号。
nParamIndex	参数序号。

返回值

类型	说明
double	DBL_MAX：未获取到参数平均值。

示例

```
double dValue = TheSoft.TestParam().GetParamAverage(0,0);
```

1.1.15 GetParamAverage（）

函数功能

获取参数平均值。

使用说明

无。

函数原型

```
double GetParamAverage(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
double	DBL_MAX：未获取到参数平均值。

示例

```
const string strParamcName = "Param";
double dValue = TheSoft.TestParam().GetParamAverage(strParamcName);
```

1.1.16 GetParamAverage_Site（）

函数功能

获取工位的某一参数的平均值。

使用说明

无。

函数原型

```
double GetParamAverage_Site(int nFunIndex, int nParamIndex, int nSite)
```

参数说明

参数	说明
nFunIndex	函数序号。
nParamIndex	参数序号。
nSite	指定工位号，超过 64 时将按 64 执行。

返回值

类型	说明
double	某一参数的平均值。

示例

```
double dValue = TheSoft.TestParam().GetParamAverage_Site(0,0,0);
```

1.1.17 GetParamAverage_Site（）

函数功能

获取工位的某一参数的平均值。

使用说明

无。

函数原型

```
double GetParamAverage_Site(const string& strParamName, int nSite);
```

参数说明

参数	说明
strParamName	参数名称。
nSite	指定工位数量，超过 64 时将按 64 执行。

返回值

类型	说明
double	某一参数的平均值。

示例

```
double dValue = TheSoft.TestParam().GetParamAverage_Site(Param,0);
```

1.1.18 GetParamComment（）

函数功能

获取设置的 Comment 信息的函数。

使用说明

无。

函数原型

```
string GetParamComment(const string& strParamName)
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
string	返回参数的备注。

示例

```
string strParamName = "Param";
TheSoft.TestParam().GetParamComment(strParamName);
```

1.1.19 GetParamComment（）

函数功能

获取设置的 Comment 信息的函数。

使用说明

无。

函数原型

```
string GetParamComment(int nIndex)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
string	返回参数的备注。

示例

```
string strComment = TheSoft.TestParam().GetParamComment(0);
```

1.1.20 GetParamDispformat（）

函数功能

获取参数显示格式的函数。

使用说明

无。

函数原型

```
string GetParamDispformat(int nIndex)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
string	返回显示格式的字符串。

示例

```
string strformat = TheSoft.TestParam().GetParamDispformat(0);
```

1.1.21 GetParamDispformat（）

函数功能

获取参数结果显示精度。

使用说明

无。

函数原型

```
string GetParamDispformat(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
string	● 当获取参数结果显示精度成功时，返回显示精度字符串。 ● 当获取参数结果显示精度失败时，返回空字符串。

示例

```
string strformat = TheSoft.TestParam().GetParamDispformat("param1");
```

1.1.22 GetParamHbin（）

函数功能

获取参数的 SHFailBin 值。

使用说明

无。

函数原型

```
int GetParamHbin(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
int	获取参数的 SHFailBin 值。

示例

```
int nHbin = TheSoft.TestParam().GetParamHbin("ParamName");
```

1.1.23 GetParamHbin（）

函数功能

获取参数 HighBin 的函数。

使用说明

无。

函数原型

```
int GetParamHbin(int nIndex)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
int	HighBin 号。

示例

```
int nHbin = TheSoft.TestParam().GetParamHbin(0);
```

1.1.24 GetParamIndex（）

函数功能

获取当前函数指定参数的 Index。

使用说明

无。

函数原型

```
int GetParamIndex(const string& strParamName);
```

参数说明

参数	说明
strParamName	指定参数名称。

返回值

类型	说明
int	● 返回参数索引号。 ● 获取失败返回-1。

示例

```
string strParamcName = "Func";
int nIndex = TheSoft.TestParam().GetParamIndex(strParamcName);
```

1.1.25 GetParamLbin（）

函数功能

获取参数 LowBin 的函数。

使用说明

无。

函数原型

```
int GetParamLbin(int nIndex)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
int	LowBin 号。

示例

```
int nLbin = TheSoft.TestParam().GetParamLbin(0);
```

1.1.26 GetParamLbin（）

函数功能

获取参数的 SLFailBin 值。

使用说明

无。

函数原型

```
int GetParamLbin(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
int	参数的 SLFailBin 值。

示例

```
int slFailbin = TheSoft.TestParam().GetParamLbin("param1");
```

1.1.27 GetParamMin（）

函数功能

获取参数 string 类型的 limit 下限的函数。

使用说明

无。

函数原型

```
int GetParamMin(int nIndex, string& strParamMin);
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。
strParamMin	返回对应参数的 Limit 的下限值。

返回值

类型	说明
int	0：获取成功。

示例

```
string paramMinLimit;
TheSoft.TestParam().GetParamMin(0, paramMinLimit);
```

1.1.28 GetParamMin（）

函数功能

获取参数 double 类型的 limit 下限的函数。

使用说明

无。

函数原型

```
double GetParamMin(int index)
```

参数说明

参数	说明
index	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
double	返回对应参数的 Limit 的下限值。

示例

```
double dMin = TheSoft.TestParam().GetParamMin(0);
```

1.1.29 GetParamMin（）

函数功能

获取参数 double 类型的 limit 下限的函数。

使用说明

无。

函数原型

```
double GetParamMin(const string& strParamName)
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
double	返回对应参数的 Limit 的下限值。

示例

```
string strParamName = "Param";
double dMin = TheSoft.TestParam().GetParamMin(strParamName);
```

1.1.30 GetParamMin（）

函数功能

获取参数 limit 下限。

使用说明

无。

函数原型

```
int GetParamMin(const string& strParamName, string& strParamMin);
```

参数说明

参数	说明
strParamName	参数名称。
strParamMin	返回参数的 limit 下限。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示该参数名不存在。

示例

```
string paramMinLimit;
TheSoft.TestParam().GetParamMin("param1", paramMinLimit);
```

1.1.31 GetParamMax（）

函数功能

获取参数名称 string 类型的 limit 上限的函数。

使用说明

无。

函数原型

```
int GetParamMax(int nIndex, string& strParamMax)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。
strParamMax	返回参数的 Limit 上限值。

返回值

类型	说明
int	0：获取成功。

示例

```
string paramMaxLimit;
TheSoft.TestParam().GetParamMax(0, paramMaxLimit);
```

1.1.32 GetParamMax（）

函数功能

获取参数 double 类型的 limit 上限的函数。

使用说明

无。

函数原型

```
double GetParamMax(int index)
```

参数说明

参数	说明
index	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
double	返回对应参数的 Limit 的上限值。

示例

```
double dMax = TheSoft.TestParam().GetParamMax(0);
```

1.1.33 GetParamMax（）

函数功能

获取参数 double 类型的 limit 上限的函数。

使用说明

无。

函数原型

```
double GetParamMax(const string& strParamMax);
```

参数说明

参数	说明
strParamMax	参数名称。

返回值

类型	说明
double	返回对应参数的 Limit 的上限值。

示例

```
string strParamMax = "Param";
double dMax = TheSoft.TestParam().GetParamMax(strParamMax);
```

1.1.34 GetParamMax（）

函数功能

获取参数 limit 上限。

使用说明

无。

函数原型

```
int GetParamMax(const string& strParamName, string& strParamMax);
```

参数说明

参数	说明
strParamName	参数名称。
strParamMax	返回参数的 limit 上限。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示该参数名不存在。

示例

```
string paramMaxLimit;
TheSoft.TestParam().GetParamMax("param1", paramMaxLimit);
```

1.1.35 GetParamName（）

函数功能

获取当前函数指定 Index 的参数名称。

使用说明

无。

函数原型

```
string GetParamName(int nIndex)
```

参数说明

参数	说明
nIndex	参数的 Index。

返回值

类型	说明
string	参数名。

示例

```
string strParamName = TheSoft.TestParam().GetParamName(0);
```

1.1.36 GetParamPassFail（）

函数功能

获取某个参数是否为 Pass（dResult 是否在参数上下限范围内）。

使用说明

无。

函数原型

```
bool GetParamPassFail(const string& strParamName, double dResult);
```

参数说明

参数	说明
strParamName	参数名称。
dResult	参数测试结果。

返回值

类型	说明
bool	• true：在参数 limit 范围内。 • false：不在参数 limit 范围内。

示例

```
const string strParamcName = "Param";
bool bFlag = TheSoft.TestParam().GetParamPassFail(strParamcName,1.0);
```

1.1.37 GetParamSubunit（）

函数功能

获取参数子单元数的函数。

使用说明

无。

函数原型

```
int GetParamSubunit(int nIndex)
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
int	返回参数子单元数。

示例

```
int nSubUnit = TheSoft.TestParam().GetParamSubunit(0);
```

1.1.38 GetParamSubunit（）

函数功能

获取当前测试函数指定参数设置的测试结果个数。

使用说明

无。

函数原型

```
int GetParamSubunit(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
int	-1：未找到该参数名。

示例

```
int nRet = TheSoft.TestParam().GetParamSubunit("ParamName")
```

1.1.39 GetParamSTDEV（）

函数功能

获取参数标准差。

使用说明

无。

函数原型

```
double GetParamSTDEV(int nFunIndex, int nParamIndex);
```

参数说明

参数	说明
nFunIndex	函数序号。
nParamIndex	参数序号。

返回值

类型	说明
double	DBL_MAX：未获取到参数标准差。

示例

```
double dValue = TheSoft.TestParam().GetParamSTDEV(0,0);
```

1.1.40 GetParamSTDEV（）

函数功能

获取参数标准差。

使用说明

无。

函数原型

```
double GetParamSTDEV(const string& strParamName)
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
double	DBL_MAX：未获取到参数标准差。

示例

```
double dValue = TheSoft.TestParam().GetParamSTDEV(Param);
```

1.1.41 GetParamSTDEV_Site（）

函数功能

获取工位的某一参数的标准差。

使用说明

无。

函数原型

```
double GetParamSTDEV_Site(int nFunIndex, int nParamIndex, int nSite);
```

参数说明

参数	说明
nFunIndex	函数序号。
nParamIndex	参数序号。
nSite	指定工位数量，超过 64 时将按 64 执行。

返回值

类型	说明
double	DBL_MAX：未获取到工位的某一参数的标准差。

示例

```
double stdev = TheSoft.TestParam().GetParamSTDEV_Site(0, 0, 0);
```

1.1.42 GetParamSTDEV_Site（）

函数功能

获取工位的某一参数的标准差。

使用说明

无。

函数原型

```
double GetParamSTDEV_Site(const string& strParamName, int nSite);
```

参数说明

参数	说明
strParamName	参数名称。
nSite	指定工位数量，超过 64 时将按 64 执行。

返回值

类型	说明
double	DBL_MAX：未获取到工位的某一参数的标准差。

示例

```
const string strParamcName = "Param";
double dValue = TheSoft.TestParam().GetParamSTDEV_Site(strParamcName,0);
```

1.1.43 GetParamTestNumber（）

函数功能

获取当前测试函数指定参数的测试编号。

使用说明

无。

函数原型

```
uint64_t GetParamTestNumber(int nIndex);
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
uint64_t	参数的测试编号。

示例

```
uint64_t nTestNumber = TheSoft.TestParam().GetParamTestNumber(0);
```

1.1.44 GetParamTestNumber（）

函数功能

获取当前测试函数指定参数的测试编号。

使用说明

无。

函数原型

```
uint64_t GetParamTestNumber(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
uint64_t	参数的测试编号。

示例

```
uint64_t nTestNumber = TheSoft.TestParam().GetParamTestNumber("param1");
```

1.1.45 GetParamUnit（）

函数功能

获取参数单位。

使用说明

无。

函数原型

```
string GetParamUnit(int nIndex);
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。

返回值

类型	说明
string	参数的单位。

示例

```
string strUnit = TheSoft.TestParam().GetParamUnit(0);
```

1.1.46 GetParamUnit（）

函数功能

获取参数的单位。

使用说明

无。

函数原型

```
string GetParamUnit(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名。

返回值

类型	说明
string	参数的单位 mA/V 等。

示例

```
string strUnit = TheSoft.TestParam().GetParamUnit("ParamName");
```

1.1.47 GetResult（）

函数功能

获取指定工位的测试结果函数。

使用说明

无。

函数原型

```
int GetResult(int nSite, double &dResult);
```

参数说明

参数	说明
nSite	指定工位，工位号从 0 开始计算。
dResult	返回被测结果。

返回值

类型	说明
int	● 0：正确返回。 ● -1：参数输入错误（工位数小于 0 或大于 64 时返回 - 1）。

示例

```
double dResult = 0.0;
int nFlag = TheSoft.TestParam().GetResult(0,dResult);
```

1.1.48 GetResultAll（）

函数功能

获取多个工位的测试结果。

使用说明

无。

函数原型

```
int GetResultAll(int nSize, double* dResult)
```

参数说明

参数	说明
nSize	指定工位数量，超过 64 时将按 64 执行。
dResult	返回被测结果的变量指针，申请空间时需确保大于等于 nSize，否则会导致内存越。

返回值

类型	说明
int	● 0：正确。 ● -1：参数输入错误（工位数小于 0 或大于 64 时返回 -1）。

示例

```
double dResult[4] = {0.0};
TheSoft.TestParam().GetResultAll(4,dResult);
```

1.1.49 ParamTestStartTime（）

函数功能

读取当前参数测试开始前的时间。

使用说明

以下情况则不能获得正确的参数测试时间：

- 若接口函数中写入的参数名与放置位置的参数名不符，该参数的 Times(mS)列显示时间，但非该参数的真实测试时间。
- 若没有成对调用，缺失任意一个，则该参数的 Times（ms）列显示为“NULL”。
- 若参数测试前加的是 ParamTestEndTime（），在测试后加的是 ParamTestStartTime（），则该参数的 Times（ms）列显示“ERRO”。
- 如果该对接口中间，除了测试参数外，还跨有其它代码行，则该参数的 Times（ms）列显示的测试时间比实际测试时间大，非该参数的真实测试时间。

函数原型

```
int ParamTestStartTime(const string& strParamName)
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
int nTemp = TheSoft.TestParam().ParamTestStartTime("Param");
```

1.1.50 ParamTestEndTime（）

函数功能

读取当前参数测试结束后的时间。

使用说明

以下情况则不能获得正确的参数测试时间：

Times（ms）列显示参数	说明
NULL	调用函数未成对，缺失任意一个。
ERRO	参数测试前加的是 ParamTestEndTime，在测试后加的是 ParamTestStartTime。
非该参数的真实测试时间	接口函数中的写入的参数名与放置位置的参数名不符。
测试时间大于实际测试时间	该对接口中间，除了测试参数外，还跨有其它代码行。

函数原型

```
int ParamTestEndTime(const string& strParamName)
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
int nTemp = TheSoft.TestParam().ParamTestEndTime(strParamName);
```

1.1.51 SetParamTestNumber（）

函数功能

设置某个参数的 testnumber。

使用说明

无。

函数原型

```
int SetParamTestNumber(const string& strParamName, uint64_t uTestNumber)
```

参数说明

参数	说明
strParamName	参数名称。
uTestNumber	参数的测试编号。

返回值

类型	说明
int	0：设置成功。

示例

```
TheSoft.TestParam().SetParamTestNumber("aa", 1);
```

1.1.52 SetParamMin（）

函数功能

设置参数的下限。

使用说明

该函数仅限在测试程序的 StartLot 和 LoadInit 中调用。

函数原型

```
int SetParamMin(const string& strParamName, double dValue, int iDecimal = 6)
```

参数说明

参数	说明
strParamName	参数名称。
dValue	设置的参数下限值。
iDecimal	参数下限值保存的小数位数，缺省值为 6。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamMin(strParamName,220.929999847,8);
```

1.1.53 SetParamMin（）

函数功能

设置参数的下限。

使用说明

该函数仅限在测试程序的 StartLot 和 LoadInit 中调用。

函数原型

```
int SetParamMin(const string& strParamName, const string& strValue)
```

参数说明

参数	说明
strParamName	参数名称。
strValue	设置的参数下限值。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamMin(strParamName,"220.929999");
```

1.1.54 SetParamMax（）

函数功能

设置参数的上限。

使用说明

该函数仅限在测试程序的 StartLot 和 LoadInit 中调用。

函数原型

```
int SetParamMax(const string& strParamName, double dValue, int iDecimal = 6)
```

参数说明

参数	说明
strParamName	参数名称。
dValue	设置的参数上限值。
iDecimal	参数上限值保存的小数位数，缺省值为 6。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamMax(strParamName,220.929999847,8);
```

1.1.55 SetParamMax（）

函数功能

设置参数的上限。

使用说明

该函数仅限在测试程序的 StartLot 和 LoadInit 中调用。

函数原型

```
int SetParamMax(const string& strParamName, const string& strValue)
```

参数说明

参数	说明
strParamName	参数名称。
strValue	设置的参数上限值。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamMax(strParamName,"3.0");
```

1.1.56 SetParamLBin（）

函数功能

设置参数的 SLFBin。

使用说明

该函数仅限在 StartLot 和 LoadInit 中使用，其他函数中调用无效。

函数原型

```
int SetParamLBin(const string& strParamName, uint32_t uSLFailBin);
```

参数说明

参数	说明
strParamName	参数名称。
uSLFailBin	设置 SLFBin。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamLBin(strParamName,5);
```

1.1.57 SetParamHBin（）

函数功能

设置参数的 SHFBin。

使用说明

该函数仅限在 StartLot 和 LoadInit 中使用，其他函数中调用无效。

函数原型

```
int SetParamHBin(const string& strParamName, uint32_t uSHFailBin);
```

参数说明

参数	说明
strParamName	参数名称。
uSHFailBin	设置 SHFBin。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestParam().SetParamHBin(strParamName,7);
```

1.1.58 SetParamNote（）

函数功能

设置参数的注释信息，可通过 csv 及 excel 文件查看。

使用说明

参数注释不支持中文和英文逗号“,”。

函数原型

```
int SetParamNote(int nSite, const string& strParamName, const string&
strParamNote);
```

参数说明

参数	说明
nSite	工位号。
strParamName	参数名称。
strParamNote	参数注释信息。

返回值

类型	说明
int	● 0：设置成功。 ● 非 0：设置失败。

示例

```
string strParamName = "Param";
string strParamNote = “this is a param”;
TheSoft.TestParam().SetParamNote(1, strParamName, strParamNote);
```

1.2 TestProc

1.2.1 ChgLotID（）

函数功能

改变当前测试 LotID 名称的函数。

使用说明

该函数需在 StartTest 中调用。

函数原型

```
int ChgLotID(const string& strLotID);
```

参数说明

参数	说明
strLotID	存储 LotID 名称的指针。

返回值

类型	说明
int	0：正常。

示例

```
TheSoft.TestProc().ChgLotID("someID");
```

1.2.2 ClrAlarm（）

函数功能

清除参数的钳位报警。

使用说明

无。

函数原型

```
int ClrAlarm(const string& strParamName);
```

参数说明

参数	说明
strParamName	参数名称。

返回值

类型	说明
int	0：正常。

示例

```
string strParamName = "Param";
TheSoft.TestProc().ClrAlarm("Param");
```

1.2.3 CustomerUI_GetTestBinNo（）

函数功能

返回特定用户测试 BinNo 信息。

使用说明

无。

函数原型

```
string CustomerUI_GetTestBinNo();
```

参数说明

无。

返回值

类型	说明
string	返回 TestBinNo。

示例

```
string strTestBinNo = TheSoft.TestProc().CustomerUI_GetTestBinNo();
```

1.2.4 DataNameFindString（）

函数功能

判断保存数据的文件名是否包含 strFindString 中的字符。

使用说明

无。

函数原型

```
int DataNameFindString(const string& strFindString);
```

参数说明

参数	说明
strFindString	要判断的字符串的常量。

返回值

类型	说明
int	0：包含。 - 1：不包含。

示例

```
string strFindString = "Vcc";
int nResult = TheSoft.TestProc().DataNameFindString(strFindString);
```

1.2.5 DelaymS（）

函数功能

设置测试过程中的延时等待。

使用说明

无。

函数原型

```
int DelaymS(double dTmS);
```

参数说明

参数	说明
dTmS	延时时间（ms），延时范围 0.001ms～100000ms，超过 100000ms 按 100000ms 执行。

返回值

类型	说明
int	● 0：设置成功。 ● –1：参数输入错误（延时输入小于 0）。

示例

```
TheSoft.TestProc().DelaymS(3)//延时 3ms
```

1.2.6 EndLot（）

函数功能

结束当前批次。

使用说明

- 调用此函数，在本次测试完成后结束批次，停止测试。
- 在 EndTest 中调用。

函数原型

```
int EndLot();
```

参数说明

无。

返回值

类型	说明
int	0：正常。

示例

```
TheSoft.TestProc().EndLot();
```

1.2.7 ExecutePlot（）

函数功能

获取指定属性的 Plot 结果。

使用说明

无。

函数原型

```
int ExecutePlot(const PlotProperty_T &prop, vector<PlotResult_T> &result, const UserFunctor &before = nullptr, const UserFunctor &after = nullptr);
```

参数说明

参数	说明
prop	配置 Plot 属性。
result	返回 Plot 结果集合。
before	before 过程中单次测试前的回调，回调原型 int（void），返回值为 0 表示函数正常结束。
after	plot 过程中单次测试后的回调，回调原型 int (void)，返回值为 0 表示函数正常结束。

返回值

类型	说明
int	● 0：测试正常结束。 ● -1：当前测试函数没有配置 DIO。 ● -2：属性设置错误。 ● -3：start 和 stop 区间有值校验失败。 ● -4：start 和 stop 的单位和 spec 不一致。 ● -5：resolution 计算出错。 ● -6：spec 值设置出错。 ● -7：pattern 运行错误。 ● -8：patternend 读取错误。

示例

```
PlotProperty_T prop;
prop.bPassStop = false;
prop.eOrder = E_Axis_X_First;
prop.xAxisProperty.eMethod = E_Axis_Linear;
prop.xAxisProperty.strSpecName = "out_OH";
prop.xAxisProperty.strStartValue = "1.5V";
prop.xAxisProperty.strStopValue = "1.9V";
prop.xAxisProperty.uiSteps = 5;
prop.yAxisProperty.eMethod = E_Axis_Linear;
prop.yAxisProperty.strSpecName = "period";
prop.yAxisProperty.strStartValue = "10n";
prop.yAxisProperty.strStopValue = "15n";
prop.yAxisProperty.uiSteps = 5;
vector<PlotResult_T> result;
int ret = TheSoft.TestProc().ExecutePlot(prop, result);
```

1.2.8 ExecuteSearch（）

函数功能

获取指定属性的 Search 结果。

使用说明

无。

函数原型

```
int ExecuteSearch(const SearchProperty_T &prop, vector<SearchResult_T> &result,
const UserFunctor &before = nullptr, const UserFunctor &after = nullptr);
```

参数说明

参数	说明
prop	配置 Search 属性。
result	返回 Search 结果集合。
before	Search 过程中单次测试前的回调，回调原型 int（void），返回值为 0 表示函数正常结束。
after	Search 过程中单次测试后的回调，回调原型 int（void），返回值为 0 表示函数正常结束。

返回值

类型	说明
int	● 0：测试正常结束。 ● -1：当前测试函数没有配置 DIO。 ● -2：属性设置错误。 ● -3：start 和 stop 区间有值校验失败。 ● -4：start 和 stop 的单位和 spec 不一致。 ● -5：resolution 计算出错。 ● -6：spec 值设置出错。 ● -7：pattern 运行错误。 ● -8：patternend 读取错误。

示例

```
SearchProperty_T prop;
prop.bPassStop = false;
prop.eMethod = E_Axis_Linear;
prop.strSpecName = "out_OH";
prop.strStartValue = "1.5V";
prop.strStopValue = "1.9V";
prop.uiSteps = 5;
vector<SearchResult_T> result;
int ret = TheSoft.TestProc().ExecuteSearch(prop, result);
```

1.2.9 GetAlreadyTestParamCnt（）

函数功能

获取已测参数个数。

使用说明

在 StartTest、EndTest 测试函数中调用该函数。

函数原型

```
int GetAlreadyTestParamCnt(int nSite);
```

参数说明

参数	说明
nSite	工位号，取值范围：[0, 63]。

返回值

类型	说明
int	获取失败返回 - 1。

示例

```
int nTestNumber = TheSoft.TestProc().GetAlreadyTestParamCnt(0);
```

1.2.10 GetCurrentPartNo（）

函数功能

获取指定 Site 的器件编号 PartNo。

使用说明

无。

函数原型

```
int GetCurrentPartNo(int nSite);
```

参数说明

参数	说明
nSite	工位号，取值范围：[0, 63]。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示设置的工位号大于最大工位数，获取失败。

示例

```
int DevNumber = TheSoft.TestProc().GetCurrentPartNo(0);
```

1.2.11 Get_CustomizeInput（）

函数功能

获取测试中途用户自定义信息。

使用说明

无。

函数原型

```
int Get_CustomizeInput(const string& strName, const string& strCaption, string& strResult);
```

参数说明

参数	说明
strName	自定义对话框的名字。
strCaption	自定义对话框的标题。
strResult	返回用户输入的信息。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strResult;
TheSoft.TestProc().Get_CustomizeInput("name", "cap", strResult);
```

1.2.12 GetDatalogName（）

函数功能

获取当前测试站 datalog 文件的名称。

使用说明

无。

函数原型

```
int GetDatalogName(string& strDatalogName);
```

参数说明

参数	说明
strDatalogName	datalog 文件名称。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strDatalogName;
TheSoft.TestProc().GetDatalogName(strDatalogName);
```

1.2.13 GetDeviceID（）

函数功能

获取测试站设备的编号。

使用说明

无。

函数原型

```
int GetDeviceID(string& strDeviceID);
```

参数说明

参数	说明
strDeviceID	设备编号。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strDeviceID ;
TheSoft.TestProc().GetDeviceID (strDeviceID );
```

1.2.14 GetFailNum（）

函数功能

获取总 Fail 数。

使用说明

无。

函数原型

```
int GetFailNum();
```

参数说明

无。

返回值

类型	说明
int	总 Fail 数。

示例

```
int nNum = TheSoft.TestProc().GetFailNum();
```

1.2.15 GetFailStop（）

函数功能

获取失效停测状态。

使用说明

无。

函数原型

```
bool GetFailStop();
```

参数说明

无。

返回值

类型	说明
bool	● 0：不进行失效停测。 ● 1：失效停测。

示例

```
bool bstate = TheSoft.TestProc().GetFailStop();
```

1.2.16 GetLotID（）

函数功能

获取当前测试的批次 ID。

使用说明

无。

函数原型

```
int GetLotID(string& strLotID);
```

参数说明

参数	说明
strLotID	当前测试的批次 ID。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strLotID;
TheSoft.TestProc().GetLotID(strLotID);
```

1.2.17 GetPassNum（）

函数功能

获取总 Pass 数。

使用说明

无。

函数原型

```
int GetPassNum();
```

参数说明

无。

返回值

类型	说明
int	总 Pass 数。

示例

```
int nNum;
nNum  = TheSoft.TestProc().GetPassNum();
```

1.2.18 GetProgPath（）

函数功能

获取当前测试程序路径（cpts 的路径）。

使用说明

无。

函数原型

```
int GetProgPath(string& strProgPath);
```

参数说明

参数	说明
strProgPath	当前测试程序路径（cpts 的路径）。

返回值

类型	说明
int	● 0：获取成功。 ● -1：获取失败。

示例

```
string strProgPath;
TheSoft.TestProc().GetProgPath(strProgPath);
```

1.2.19 GetSBinNo（）

函数功能

获取 Site 分到的 SBin 号。

使用说明

该函数在 EndTest 中调用。

函数原型

```
int GetSBinNo(int nSite);
```

参数说明

参数	说明
nSite	需要获取 SBin 的 Site（0，1，2，…，63）

返回值

类型	说明
int	返回 SBin 号。

示例

```
int nSBin = TheSoft.TestProc().GetSBinNo(0);
```

1.2.20 GetSingleSiteStatus（）

函数功能

获取指定工位状态。

使用说明

无。

函数原型

```
int GetSingleSiteStatus(int nSite);
```

参数说明

参数	说明
nSite	指定工位号，工位号从 0 开始，输入 nSite 不能超过当前测试程序的最大工位数。

返回值

类型	说明
int	工位状态。 ● 0：工位无效。 ● 1：工位有效。 ● -1：nSite 不符合条件。

示例

```
int nResult = TheSoft.TestProc().GetSingleSiteStatus(0);
```

1.2.21 GetSiteStatus（）

函数功能

获取当前各工位状态。

使用说明

无。

函数原型

```
unsigned long long GetSiteStatus();
```

参数说明

无。

返回值

类型	说明
unsigned long long	每个二进制位表示一个 Site 的状态，从右起第一位为 Site1，第二位为 Site2，以此类推。 ● 0：工位无效。 ● 1：工位有效。

示例

```
unsigned long long uNum;
uNum = TheSoft.TestProc().GetSiteStatus();
```

1.2.22 GetSiteStatus（）

函数功能

获取当前各工位状态。

使用说明

无。

函数原型

```
int GetSiteStatus(int *nSite);
```

参数说明

参数	说明
nSite	返回当前各工位状态的 int 型指针，其对应的值 0 代表工位无效，1 代表工位有效。

返回值

类型	说明
int	0：获取成功。

示例

```
int nStatus[32] = {0};
TheSoft.TestProc().GetSiteStatus(nStatus);
```

1.2.23 GetSiteCurrentTestResult（）

函数功能

获取指定工位 nSite 的测试 Pass/Fail 结果。

使用说明

- 在测试程序 EndTest 中调用表示获取某一 Site 本次测试的测试结果（Pass or Fail）(出现超能量报警这个值不一定准)。
- 在测试程序 StartTest 中调用表示获取某一 Site 上一次测试的测试结果。
- 建议在 StartTest 中使用。

函数原型

```
int GetSiteCurrentTestResult(int nSite);
```

参数说明

参数	说明
nSite	工位号，取值范围：[0, 63]。

返回值

类型	说明
int	● 0：该工位测试 Fail。 ● 1：该工位测试 Pass。 ● -1：该工位未进行测试。 ● -2：nSite 超出范围。

示例

```
int nResult = TheSoft.TestProc().GetSiteCurrentTestResult(0);
```

1.2.24 GetSiteDistribute（）

函数功能

获取测试站当前设定的针卡排序序号。

使用说明

无。

函数原型

```
int GetSiteDistribute(string& strSiteDistribute);
```

参数说明

参数	说明
strSiteDistribute	针卡排序序号。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strSiteDis;
TheSoft.TestProc().GetSiteDistribute(strSiteDis);
```

1.2.25 GetSiteNum（）

函数功能

获取当前测试程序工位总数。

使用说明

无。

函数原型

```
int GetSiteNum();
```

参数说明

无。

返回值

类型	说明
int	返回工位数量。

示例

```
int nNum;
nNum  = TheSoft.TestProc().GetSiteNum();
```

1.2.26 GetSiteTotalNum（）

函数功能

获取指定 Site 的测试数量，止于上一次测试结果。

使用说明

无。

函数原型

```
int GetSiteTotalNum(int nSite);
```

参数说明

参数	说明
nSite	指定的工位号，取值范围：[0, 63]。

返回值

类型	说明
int	指定 Site 的测试数量。

示例

```
int nNum;
nNum = TheSoft.TestProc().GetSiteTotalNum(0);
```

1.2.27 GetSitePassNum（）

函数功能

获取指定 site 的测试 Pass 数，止于上一次测试结果。

使用说明

无。

函数原型

```
int GetSitePassNum(int nSite);
```

参数说明

参数	说明
nSite	指定的工位号，取值范围：[0, 63]。

返回值

类型	说明
int	对应 Site 的 Pass 数。

示例

```
int nNum;
nNum = TheSoft.TestProc().GetSitePassNum(0);
```

1.2.28 GetSiteFailNum（）

函数功能

获取指定 site 的测试 Fail 数，止于上一次测试结果。

使用说明

无。

函数原型

```
int GetSiteFailNum(int nSite);
```

参数说明

参数	说明
nSite	指定的工位号，取值范围：[0, 63]。

返回值

类型	说明
int	对应 Site 的失效数。

示例

```
int nNum;
nNum = TheSoft.TestProc().GetSiteFailNum(0);
```

1.2.29 GetSiteSBinNum（）

函数功能

获取指定 Site 指定 Bin 的数量，止于上一次测试结果。

使用说明

第二次测试时获取的为第一次测试的结果。

函数原型

```
int GetSiteSBinNum(int nSite, int nBin);
```

参数说明

参数	说明
nSite	指定的工位号，取值范围：[0, 63]。
nBin	需要获取的 Bin 号。

返回值

类型	说明
int	● 0：获取成功。 ● -1：获取失败。

示例

```
int nNum;
nNum = TheSoft.TestProc().GetSiteSBinNum(1,6);
```

1.2.30 GetSoftVersion（）

函数功能

获取当前测试系统软件版本信息的函数。

使用说明

无。

函数原型

```
int GetSoftVersion(string& strSoftVer);
```

参数说明

参数	说明
strSoftVer	软件版本号。

返回值

类型	说明
int	● 0：获取成功。 ● 非 0：获取失败。

示例

```
string strSoftVer;
TheSoft.TestProc().GetSoftVersion (strSoftVer);
```

1.2.31 GetSoftwareName（）

函数功能

获取当前测试系统软件名称的函数。

使用说明

无。

函数原型

```
int GetSoftwareName(string& strSoftwareName);
```

参数说明

参数	说明
strSoftwareName	返回软件名称。

返回值

类型	说明
int	● 0：获取成功。 ● 非 0：获取失败。

示例

```
string strSoftName;
TheSoft.TestProc().GetSoftwareName(strSoftName);
```

1.2.32 GetTestMode（）

函数功能

获取当前测试机测试状态。

使用说明

无。

函数原型

```
int GetTestMode();
```

参数说明

无。

返回值

类型	说明
int	测试机测试状态。 ● 0：Stop 暂停。 ● 1：Single manual test run 单次测试。 ● 2：Continue manual test run 连续测试。 ● 3：Auto test run 联机测试。

示例

```
int nstate = TheSoft.TestProc().GetTestMode();
```

1.2.33 GetTotalNum（）

函数功能

获取测试总数。

使用说明

无。

函数原型

```
int GetTotalNum();
```

参数说明

无。

返回值

类型	说明
int	测试总数。

示例

```
int nNum;
nNum  = TheSoft.TestProc().GetTotalNum();
```

1.2.34 GetTesterSN（）

函数功能

获取测试机台 S/N 号码。

使用说明

无。

函数原型

```
int GetTesterSN(string& strTesterSN);
```

参数说明

参数	说明
strTesterSN	获取机台设置的 S/N 信息。

返回值

类型	说明
int	0：正常。

示例

```
string strTesterSN;
TheSoft.TestProc().GetTesterSN(strTesterSN);
```

1.2.35 GetWaferID（）

函数功能

获取本次测试的 WaferID 号。

使用说明

无。

函数原型

```
int GetWaferID(string& strWaferID);
```

参数说明

参数	说明
strWaferID	WaferID 的值。

返回值

类型	说明
int	● 0：正常。 ● -1：获取失败。

示例

```
string strWaferID;
TheSoft.TestProc().GetWaferID (strWaferID);
```

1.2.36 GetWaferXCoord（）

函数功能

获取指定测试站指定工位的探针台的 x 坐标。

使用说明

无。

函数原型

```
int GetWaferXCoord(int nSite);
```

参数说明

参数	说明
nSite	指定 Site（0，1，2，…，63）。

返回值

类型	说明
int	65535：参数错误，nSite 小于 0 或者大于最大工位数。

示例

```
TheSoft.TestProc().GetWaferXCoord(0);
```

1.2.37 GetWaferYCoord（）

函数功能

获取指定测试站指定工位的探针台的 y 坐标。

使用说明

无。

函数原型

```
int GetWaferYCoord(int nsite);
```

参数说明

参数	说明
nSite	指定 Site（0，1，2，…，63）。

返回值

类型	说明
int	65535：参数错误，nSite 小于 0 或者大于最大工位数。

示例

```
TheSoft.TestProc().GetWaferYCoord(0);
```

1.2.38 GetHandlerBarcode（）

函数功能

获取指定测试站指定工位的二维码信息。。

使用说明

在 Run 测试，且终端为 C6Handler 的情况下才能正确获取二维码信息。其他情况如 single、mtest、终端非 C6Handler 的情况下二维码信息为“None”。。

函数原型

```
string GetHandlerBarcode(int nsite);
```

参数说明

参数	说明
nsite	指定 Site（0，1，2，…，63）。

返回值

类型	说明
string	指定 site 的二维码信息。

示例

```
string info;
info = TheSoft.TestProc().GetHandlerBarcode(0);
```

1.2.39 NewLot（）

函数功能

开始新批次的函数。

使用说明

该函数需在 StartTest 中调用。

函数原型

```
int NewLot(int nStation = 0);
```

参数说明

无

返回值

类型	说明
int	● 0：正常开启新批次。 ● -1：开始新批次失败。

示例

```
TheSoft.TestProc().NewLot();
```

1.2.40 RegUserFunction（）

函数功能

注册用户自定义函数。将指定按键与自定义函数进行绑定。绑定后，可通过快捷键访问该函数。

使用说明

该函数在 LoadInit 中调用（使用 PTSEdit 的 VCProject 打开工程会自动添加）。

函数原型

```
int RegUserFunction(int key, PTR_USERFUN ptrFun);
```

参数说明

参数	说明
key	虚拟按键值。具体含义参见“WinUser.h”。
ptrFun	函数指针 typedefint（*PTR_USERFUN）()。

返回值

类型	说明
int	0：正常。

示例

```
int userFun1()
{
.... //自定义逻辑
return 0;
}
int LoadInit(int iParam = 0)
{
TheSoft.TestProc().RegUserFunction(VK_F1, &userFun1);
return 0;
}
```

1.2.41 ReName_DatalogName（）

函数功能

修改当前测试 datalog 文件的文件名称。

使用说明

无。

函数原型

```
int ReName_DatalogName(const string& strDatalogName);
```

参数说明

参数	说明
strDatalogName	datalog 新的文件名称。

返回值

类型	说明
int	● 0：表示成功。 ● -1：表示失败。

示例

```
string strDatalogName = "datanam";
TheSoft.TestProc().ReName_DatalogName(strDatalogName);
```

1.2.42 SetEachConWaitTime（）

函数功能

设置手动连续测试间隔等待时间，单位为 ms。

使用说明

无。

函数原型

```
int SetEachConWaitTime(unsigned int uWaitTime);
```

参数说明

参数	说明
uWaitTime	等待时间，单位：ms。

返回值

类型	说明
int	● 0：正常。 ● 非 0：设置失败。

示例

```
TheSoft.TestProc().SetEachConWaitTime(100);
```

1.2.43 SetFailStop（）

函数功能

设置是否进行失效停测。

使用说明

无。

函数原型

```
int SetFailStop(const bool bIsFailStop);
```

参数说明

参数	说明
bIsFailStop	是否进行失效停测。 ● True：失效停测。 ● False：失效不停测。

返回值

类型	说明
int	● 0：正常。 ● 非 0：设置失败。

示例

```
TheSoft.TestProc().SetFailStop(true);
```

1.2.44 SetMRepetNum（）

函数功能

设置连续测试的数量，测试完成后停止测试。

使用说明

该函数可在测试程序任意位置调用。

函数原型

```
int SetMRepetNum(int nNum);
```

参数说明

参数	说明
nNum	连续测试的数量。

返回值

类型	说明
int	● 0：正常。 ● -1：设置失败。

示例

```
TheSoft.TestProc().SetMRepetNum(100);
```

1.2.45 SetLowYieldAlarm（）

函数功能

设置 TestControl 页面中的低良率报警，主要有三种 TotalYieldAlarm，SiteYieldDiffAlarm，SiteDiffAlarm。

使用说明

无。

函数原型

```
int SetLowYieldAlarm(bool bTotalYieldAlarm, bool bSiteYieldAlarm, bool bSiteDiffAlarm,int nYieldNum, double dYield, double dSiteDiffYield);
```

参数说明

参数	说明
bTotalYieldAlarm	是否勾选 TotalYieldAlarm。 ● True：勾选 TotalYieldAlarm。 ● False：不勾选 TotalYieldAlarm。
bSiteYieldAlarm	是否勾选勾选 SiteYieldAlarm。 ● True：勾选 SiteYieldAlarm。 ● False：不勾选 SiteYieldAlarm。
bSiteDiffAlarm	是否勾选 SiteYieldDiffAlarm。 ● True：勾选 SiteYieldDiffAlarm。 ● False：不勾选 SiteYieldDiffAlarm。
nYieldNum	设置 YieldAlarmNum。 取值范围：大于 0 的正整数。
dYield	设置 AlarmYield。 取值范围：（0～100）。
dSiteDiffYield	设置 SiteYieldDiffAlarm。 取值范围：（0～100）。

返回值

类型	说明
int	● 0：设置成功。 ● －2：nYieldNum 设置成了小于等于 0。 ● －3：dYield 良率不在 0%～100%内。 ● －4：dSiteDiffYield 良率不在 0%～100%内。

示例

```
int nResult =
TheSoft.TestProc().SetLowYieldAlarm(True,True,True,10,29.999,39.99);
```

1.2.46 SetPartNo（）

函数功能

设置器件编号 PartNo。

使用说明

该函数需要在 EndTest 之前调用，否则会在下次测试生效。

函数原型

```
int SetPartNo(int nPartNo);
```

参数说明

参数	说明
nPartNo	设置当前测试器件的编号。

返回值

类型	说明
int	● 0：正常。 ● –1：设置失败。

示例

```
int nResult = TheSoft.TestProc().SetPartNo(1);
```

1.2.47 SetPassSBin（）

函数功能

给指定参数、指定工位进行 PassBin 赋值的函数。

使用说明

无。

函数原型

```
int SetPassSBin(int nIndex, int nSite, int nPassbin);
```

参数说明

参数	说明
nIndex	指定参数的序号，序号含义为该参数是当前函数的第 N 个参数，N 从 0 开始计数。
nSite	指定工位，工位号从 0 开始计算，最大到 63。
nPassbin	PassBin 的软 Bin 号。

返回值

类型	说明
int	● 0：正确。 ● –1：函数错误。 ● –2：工位错误。 ● –3：nIndex 错误。

示例

```
int nResult = TheSoft.TestProc().SetPassSBin(0, 0, 3);
```

1.2.48 SetPassSBin（）

函数功能

给指定名称参数、指定工位进行 PassBin 赋值的函数。

使用说明

无。

函数原型

```
int SetPassSBin(const string& strParamName, int nSite, int nPassbin);
```

参数说明

参数	说明
strParamName	存储指定参数名称。
nSite	指定工位，工位号从 0 开始计算，最大到 63。
nPassbin	PassBin 的软 Bin 号。

返回值

类型	说明
int	● 0：正确。 ● –1：函数错误。 ● –2：工位错误。

示例

```
int nResult = TheSoft.TestProc().SetPassSBin("Param",0,3);
```

1.2.49 SetSingleSiteStatus（）

函数功能

给指定工位进行有效性赋值的函数。

使用说明

无。

函数原型

```
int SetSingleSiteStatus(int nSite, bool bSiteStatus);
```

参数说明

参数	说明
nSite	指定工位号，工位号从 0 开始，输入 nSite 不能超过当前测试程序的最大工位数。
bSiteStatus	工位有效性。 ● 0 代表无效。 ● 1 代表有效。

返回值

类型	说明
int	● 0：正确。 ● –1：nSite 异常。

示例

```
int nResult = TheSoft.TestProc().SetSingleSiteStatus(0, 1);
```

1.2.50 SetSiteStatus（）

函数功能

给所有工位进行有效性赋值的函数。

使用说明

无。

函数原型

```
int SetSiteStatus(unsigned long long lSiteStatus);
```

参数说明

参数	说明
lSiteStatus	一个 ulonglong 型数字，每个二进制位表示一个 Site 的状态，从右起第一位为 Site1，第二位为 Site2，以此类推。 ● 0：工位无效。 ● 1：工位有效。

返回值

类型	说明
int	0：正常。

示例

```
TheSoft.TestProc().SetSiteStatus(3);
```

1.2.51 SetStopAfterConFail（）

函数功能

设置 TestControl 页面的连续测试失效停测次数。

使用说明

无。

函数原型

```
int SetStopAfterConFail(int nNum);
```

参数说明

参数	说明
nNum	连续测试失效停测次数。

返回值

类型	说明
int	● 0：正常。 ● 非 0：设置失败。

示例

```
TheSoft.TestProc().SetStopAfterConFail(100);
```

1.2.52 SetStopAfterConPass（）

函数功能

设置 TestControl 页面的连续测试 Pass 停测次数。

使用说明

无。

函数原型

```
int SetStopAfterConPass(int nNum);
```

参数说明

参数	说明
nNum	连续测试 Pass 停测次数。

返回值

类型	说明
int	● 0：正常。 ● 非 0：设置失败。

示例

```
TheSoft.TestProc().SetStopAfterConPass(100);
```

1.2.53 SetSamplingStartPoint（）

函数功能

设置每个 Site 在选择抽样保存数据时，前多少次一定保存数据。

使用说明

无。

函数原型

```
int SetSamplingStartPoint(int nPoint, int nSite = -1);
```

参数说明

参数	说明
nPoint	对应 Site 前多少点全部保存数据，超过设置点数后，按照界面设置的采样频率保存数据。
nSite	设置 Site 的序号，从 0 开始，默认为-1，当 nSite 为-1 时，则全部 Site 设置的值一致。

返回值

类型	说明
int	● 0：设置成功。 ● - 1：设置失败。

示例

```
int iRes = TheSoft. TestProc().SetSamplingStartPoint(10, 1);
```

1.2.54 SetTimeDisplay（）

函数功能

设置界面显示（TimeDisplay）的时间类型。

使用说明

无。

函数原型

```
int SetTimeDisplay(int nShowType);
```

参数说明

参数	说明
nShowType	● 0 Test：显示测试时间。 ● 1 Test+Handle：显示测试和换测时间。

返回值

类型	说明
int	● 0：设置成功。 ● - 1：设置失败。

示例

```
TheSoft.TestProc().SetTimeDisplay(0);
```

1.2.55 SetUserDefinedSummary（）

函数功能

设置用户自定义 Summary 信息的函数。

使用说明

无。

函数原型

```
int SetUserDefinedSummary(const string& strUserDefinedSum);
```

参数说明

参数	说明
strUserDefinedSum	设置的自定义的 Summary 信息。

返回值

类型	说明
int	0：正常。

示例

```
TheSoft.TestProc().SetUserDefinedSummary("somesummary");
```

1.2.56 Stdf_GetMirData（）

1.2.56.1 STDF_MIR_TYPE 类型定义

类型	Enumeration

描述	MIR 表字段定义。	
元素	MIR_SETUP_T	设置任务时的日期和时间
	MIR_START_T	测试开始的日期和时间
	MIR_STAT_NUM	测试工位编号（ID）
	MIR_MODE_COD	测试模式代码
	MIR_RTST_COD	重设批次代码
	MIR_PROT_COD	数据保护代码
	MIR_BURN_TIM	老化测试时间（以分钟为单位）
	MIR_CMOD_COD	命令模式代码
	MIR_LOT_ID	批次编号（ID）
	MIR_PART_TYP	元器件类型
	MIR_NODE_NAM	产生数据节点的名称
	MIR_TSTR_TYP	测试机类型
	MIR_JOB_NAM	测试程序名字
	MIR_JOB_REV	测试程序修订记录编号
	MIR_SBLOT_ID	Sub Lot 编号（ID）
	MIR_OPER_NAM	操作员名字或编号（ID）
	MIR_EXEC_TYP	测试机执行软件类型
	MIR_EXEC_VER	测试机执行软件版本编号
	MIR_TEST_COD	测试相位或步进代码
	MIR_TST_TEMP	测试温度
	MIR_USER_TXT	通用用户文本
	MIR_AUX_FILE	辅助数据文件的名称
	MIR_PKG_TYP	封装类型
	MIR_FAMLY_ID	产品系列编号（ID）
	MIR_DATE_COD	日期代码
	MIR_FACIL_ID	测试设备编号（ID）
	MIR_FLOOR_ID	Test Floor 的编号（ID）
	MIR_PROC_ID	量产编号（ID）
	MIR_OPER_FRQ	操作频率或步骤
	MIR_SPEC_NAM	规格说明书名称
	MIR_SPEC_VER	规格说明书版本
	MIR_FLOW_ID	测试流程编号（ID）
	MIR_SETUP_ID	Setup ID
	MIR_DSGN_REV	Design Version
	MIR_ENG_ID	Engineering ID
	MIR_ROM_COD	ROM 代码编号（ID）

	MIR_SERL_NUM	Serial Number
	MIR_SUPR_NAM	监察员编号（ID）
头文件	UserDef.h	

1.2.56.2 函数说明

函数功能

获取 stdf 文件 MIR 表字段值。

使用说明

无。

函数原型

```
string Stdf_GetMirData(const string& strStdfPath, STDF_MIR_TYPE eMirType);
```

参数说明

参数	说明
strStdfPath	文件路径。
eMirType	MIR 表字段对应枚举类型，详细定义请参见“1.2.56.1 STDF_MIR_TYPE 类型定义”。

返回值

类型	说明
string	MIR 表字段值。

示例

```
string strStdfPath = "D:\\CTA8290DPlus\\Data\\222_2.stdf";
string strMirData = TheSoft.TestProc().Stdf_GetMirData(strStdfPath,
MIR_START_T);
```

1.2.57 Stdf_GetMrrData（）

1.2.57.1 STDF_MRR_TYPE 类型定义

类型	Enumeration	
描述	MRR 表字段定义。	
元素	MRR_FINISH_T	测试完成的日期和时间

	MRR_DISP_COD	批次处置代码
	MRR_USR_DESC	用户定义的描述批次测试结果的信息
	MRR_EXC_DESC	描述批次测试结果异常的信息
头文件	UserDef.h	

1.2.57.2 函数说明

函数功能

获取 stdf 文件 MRR 表字段值。

使用说明

无。

函数原型

```
string Stdf_GetMrrData(const string& strStdfPath, STDF_MRR_TYPE eMrrType);
```

参数说明

参数	说明
strStdfPath	文件路径。
eMrrType	MRR 表字段对应枚举类型，详细定义请参见“1.2.57.1 STDF_MRR_TYPE 类型定义”。

返回值

类型	说明
string	MRR 表字段值。

示例

```
string strStdfPath = "D:\\CTA8290DPlus\\Data\\222_2.stdf";
string strMrrData = TheSoft.TestProc().Stdf_GetMrrData(strStdfPath,
MRR_DISP_COD);
```

1.2.58 Stdf_GetSdrData（）

1.2.58.1 STDF_SDR_TYPE 定义

类型	Enumeration
描述	SDR 表字段定义。

元素	SDR_HEAD_NUM	测试头编号
	SDR_SITE_GRP	工位分组编号
	SDR_SITE_CNT	一个工位分组中的测试工位数量
	SDR_SITE_NUM	测试工位数量
	SDR_HAND_TYP	分选机或探针台类型
	SDR_HAND_ID	分选机或探针台编号（ID）
	SDR_CARD_TYP	探针卡类型
	SDR_CARD_ID	探针卡编号（ID）
	SDR_LOAD_TYP	LoadBoard 类型
	SDR_LOAD_ID	LoadBoard 编号（ID）
	SDR_DIB_TYP	DIB 板类型
	SDR_DIB_ID	DIB 板编号（ID）
	SDR_CABL_TYP	线缆类型
	SDR_CABL_ID	线缆编号（ID）
	SDR_CONT_TYP	分选机连接类型
	SDR_CONT_ID	分选机连接器编号（ID）
	SDR_LASR_TYP	激光类型
	SDR_LASR_ID	激光识别编号（ID）
	SDR_EXTR_TYP	外挂设备类型
	SDR_EXTR_ID	外挂设备编号（ID）
头文件	UserDef.h	

1.2.58.2 函数说明

函数功能

获取 stdf 文件 SDR 表字段值。

使用说明

无。

函数原型

```
string Stdf_GetSdrData(const string& strStdfPath, STDF_SDR_TYPE eSdrType);
```

参数说明

参数	说明
strStdfPath	文件路径。
eSdrType	SDR 表字段对应枚举类型，详细定义请参见“1.2.58.1 STDF_SDR_TYPE 定义”。

返回值

类型	说明
string	SDR 表字段值。

示例

```
string strStdfPath = "D:\\CTA8290DPlus\\Data\\222_2.stdf";
string strSdrData = TheSoft.TestProc().Stdf_GetSdrData(strStdfPath,
SDR_SITE_GRP);
```

1.2.59 StopProgTest（）

函数功能

停止测试，注意需要 Stop 按钮有权限时才可以使用。

使用说明

无。

函数原型

```
int StopProgTest();
```

参数说明

无。

返回值

类型	说明
int	● 0：正常。 ● 非 0：停止失败。

示例

```
TheSoft.TestProc().StopProgTest();
```

1.2.60 UnloadProgram（）

函数功能

卸载程序。

使用说明

无。

函数原型

```
int UnloadProgram();
```

参数说明

无。

返回值

类型	说明
int	● 0：正常。 ● -1：失败。

示例

```
TheSoft.TestProc().UnloadProgram();
```

1.2.61 VOSC_SaveWavePic（）

函数功能

测试的电压电流波形保存图片或 vof 文件功能。

使用说明

保存的图片不能通过 VOSC 进行打开，若需要 VOSC 打开建议保存为 vof 格式的文件。

函数原型

```
int VOSC_SaveWavePic(const string& strItemName, const string& strPinName, int nSaveType = 0, const string& strSavePath = "", bool bIsShowWave = false, bool bIsErrorPopMesg = false);
```

参数说明

参数	说明
strItemName	测试项，将作为保存文件名，文件名为 strItemName＋时间戳。
strPinName	需要显示的 Pin 脚波形，要显示多个 Pin 脚波形中间以英文的“;”分隔。
nSaveType	存储文件类型。 ● 0：保存成*.jpg。 ● 1：保存成*.bmp。 ● 2：保存成*.vof，其余值均保存成*jpg。
strSavePath	空：默认保存在测试程序下的 VOSC 文件夹。 非空：保存文件在 cSavePath 路径。
bIsShowWave	显示状态。 ● false：仅保存，不显示在最前面。 ● true：保存并将波形实时显示在虚拟示波器，显示虚拟示波器在前面。
bIsErrorPopMesg	● false：错误不弹框，但记录日志。 ● true：错误弹框并记录日志。

返回值

类型	说明
int	● 0：正常。 ● 非 0：保存失败。

示例

```
TheSoft.TestProc().VOSC_SaveWavePic("VOSC_Pic","FVI8_ADF", 0, "", false, false);
```

1.3 Level

1.3.1 GetCurrentLevelBlockName（）

函数功能

获取 level block 的名称。

使用说明

无。

函数原型

```
string GetCurrentLevelBlockName()
```

参数说明

无。

返回值

类型	说明
string	Level 的 BlockName

示例

```
string strLevelName = TheSoft.Level().GetCurrentLevelBlockName();
```

1.3.2 GetVOH（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 VOH。

使用说明

无。

函数原型

```
double GetVOH(const string& strLevelBlock, const string& strPinName);
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	VOH 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double dvoh = TheSoft.Level().GetVOH(strLevelBlock,strPinName);
```

1.3.3 GetVOL（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 VOL。

使用说明

无。

函数原型

```
double GetVOL(const string& strLevelBlock, const string& strPinName);
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	VOL 的值。

示例

```
double dvol = TheSoft.Level().GetVOL("LevelBlock","Pin1");
```

1.3.4 GetTERM（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 TERM。

使用说明

无。

函数原型

```
LevelTerm_E GetTERM(const string& strLevelBlock, const string& strPinName);
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
LevelTerm_E	• E_TERM_OFF：关闭动态加载模式。 • E_TERM_TERM：端接电压，可设置 VT。 • E_TERM_LOAD：接负载，可设置 VT、IOH、IOL。 • E_TERM_NONE：无。

示例

```
LevelTerm_E term = TheSoft.Level().GetTERM("LevelBlock","Pin1");
```

1.3.5 GetVT（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 VT。

使用说明

无。

函数原型

```
double GetVT(const string& strLevelBlock, const string& strPinName);
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	VT 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double dvt = TheSoft.Level().GetVT(strLevelBlock,strPinName);
```

1.3.6 GetIOL（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 IOL。

使用说明

无。

函数原型

```
double GetIOL(const string& strLevelBlock, const string& strPinName);
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	IOL 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double diol = TheSoft.Level().GetIOL(strLevelBlock,strPinName);
```

1.3.7 GetIOH（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 IOH。

使用说明

无。

函数原型

```
double GetIOH(const string& strLevelBlock, const string& strPinName)
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	IOH 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double dioh = TheSoft.Level().GetIOH(strLevelBlock,strPinName);
```

1.3.8 GetVCL（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 VCL。

使用说明

无。

函数原型

```
double GetVCL(const string& strLevelBlock, const string& strPinName)
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	VCL 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double dvcl = TheSoft.Level().GetVCL(strLevelBlock,strPinName);
```

1.3.9 GetVCH（）

函数功能

获取对应 LevelBlock 中对应 Pin 的 VCH。

使用说明

无。

函数原型

```
double GetVCH(const string& strLevelBlock, const string& strPinName)
```

参数说明

参数	说明
strLevelBlock	LevelBlock 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	VCH 的值。

示例

```
string strPinName = "Pin1";
string strLevelBlock = "LevelBlock";
double dvch = TheSoft.Level().GetVCH(strLevelBlock,strPinName);
```

1.3.10 SetLevel（）

函数功能

设置当前测试项使用的 levelblock 的名称。

使用说明

无。

函数原型

```
int SetLevel(const string& strLevelBlockName);
```

参数说明

参数	说明
strLevelBlockName	LevelBlock 的名称。

返回值

类型	说明
int	● 0：表示设置成功。 ● -1：表示设置失败。

示例

```
TheSoft.Level().SetLevel("LevelBlockName1");
```

1.4 Pattern

1.4.1 AppendPatternBurst（）

函数功能

增加 pattern 文件。

使用说明

无。

函数原型

```
int AppendPatternBurst(const string& strBurstName, const vector<string>
&patternFiles);
```

参数说明

参数	说明
strBurstName	新增的 patternburst 的名称。
patternFiles	增加到 patternburst 中的 pattern 文件的文件名，支持绝对路径和相对路径。

返回值

类型	说明
int	● 0：表示执行成功。 ● 非 0：表示执行失败。

示例

```
string strLotID;
TheSoft.TestProc().GetLotID(strLotID);
if (string::npos != strLotID.find("aa"))
{
  vector<string> vecPatternFiles;
  vecPatternFiles.push_back("E:\\SomeTest\\shmootst.cap");
  TheSoft.Pattern().AppendPatternBurst("newburst", vecPatternFiles);
    }
```

1.4.2 GetBurstPathList（）

函数功能

获取 Pattern 中 Burst 的对应信息。

使用说明

无。

函数原型

```
vector<string> GetBurstPathList(const string& strBurstName);
```

参数说明

参数	说明
strBurstName	patternburst 的名称。

返回值

类型	说明
vector<string>	对应 Burst 中的信息表。

示例

```
string strBurstName = "burstName";
vector<string> vec_stringBursh =
TheSoft.Pattern().GetBurstPathList(strBurstName);
```

1.4.3 GetCurrentPatternBlockName（）

函数功能

获取当前函数对应的 Pattern 的 burstName。

使用说明

无。

函数原型

```
string GetCurrentPatternBlockName();
```

参数说明

无。

返回值

类型	说明
string	Pattern 的 burstName。

示例

```
string s_BlockName = TheSoft.Pattern().GetCurrentPatternBlockName()
```

1.4.4 SetBurstName（）

函数功能

设置 Pattern 的 burstName 对应的信息。

使用说明

无。

函数原型

```
int SetBurstName(vector<string>& vecPatternList, const string& strBurstName)
```

参数说明

参数	说明
vecPatternList	pattern 的列表名称。
strBurstName	burstName 的名称。

返回值

类型	说明
int	● 0：成功。 ● -1：失败。

示例

```
vector<string> vec_Set;
vec_Set.push_back("bb.cbp");
vec_Set.push_back("aa.cbp");
vec_Set.push_back("ff.cbp");
string strBurstName = "burstName";
int burstname = TheSoft.Pattern().SetBurstName(vec_Set,strBurstName);
```

1.4.5 SetPattern（）

函数功能

设置当前函数对应的 Pattern。

使用说明

无。

函数原型

```
int SetPattern(const string& strPatternBlockName)
```

参数说明

参数	说明
strPatternBlockName	Pattern 的 Block 名称。

返回值

类型	说明
int	● 0：成功。 ● -1：失败。

示例

```
string strPatternBlockName = "PatternBlockName";
int result = TheSoft.Pattern().SetPattern(strPatternBlockName);
```

1.5 Timing

1.5.1 GetCompareEdgeTwo（）

函数功能

获取对应 timingblock 对应 pin 的 CompareEdge2。

使用说明

无。

函数原型

```
double GetCompareEdgeTwo(const string& strTimingBlock, const string&
strPinName);
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	CompareEdge2 的数据。

示例

```
string strtimingblock = "timingblock";
```

```
string strPinName = "Pin1";
double dCompareEdge2 =
TheSoft.Timing().GetCompareEdgeTwo(strtimingblock,strPinName);
```

1.5.2 GetCurrentTimingBlockName（）

函数功能

获取当前测试项使用的 TimingBlock 的名称。

使用说明

无。

函数原型

```
string GetCurrentTimingBlockName();
```

参数说明

无。

返回值

类型	说明
string	当前 TimingBlock 的名称。

示例

```
string strTimingBlock = TheSoft.Timing().GetCurrentTimingBlockName();
```

1.5.3 GetCompareEdge（）

函数功能

获取对应 Timingblock 对应 Pin 的 CompareEdge。

使用说明

无。

函数原型

```
double GetCompareEdge(const string& strTimingBlock, const string& strPinName)
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	CompareEdge 的数据。

示例

```
string strtimingblock = "timingblock";
string strPinName = "Pin1";
double dCompareEdge =
TheSoft.Timing().GetCompareEdge(strtimingblock,strPinName);
```

1.5.4 GetDriveOn（）

函数功能

获取对应 timingblock 对应 pin 的 DriveOn。

使用说明

无。

函数原型

```
double GetDriveOn(const string& strTimingBlock, const string& strPinName)
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	DriveOn 的数据。

示例

```
string strtimingblock = "timingblock";
string strPinName = "Pin1";
double ddriveon = TheSoft.Timing().GetDriveOn(strtimingblock,strPinName);
```

1.5.5 GetDriveData（）

函数功能

获取对应 timingblock 对应 pin 的 DriveData。

使用说明

无。

函数原型

```
double GetDriveData(const string& strTimingBlock, const string& strPinName)
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	DriveData 的数据。

示例

```
string strtimingblock = "timingblock";
string strPinName = "Pin1";
double ddrivedata = TheSoft.Timing().GetDriveData(strtimingblock, strPinName);
```

1.5.6 GetDriveReturn（）

函数功能

获取对应 timingblock 对应 Pin 的 DriveReturn。

使用说明

无。

函数原型

```
double GetDriveReturn(const string& strTimingBlock, const string& strPinName)
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	DriveReturn 的数据。

示例

```
string strtimingblock = "timingblock";
string strPinName = "Pin1";
double ddrivereturn = TheSoft.Timing().GetDriveReturn(strtimingblock,
strPinName);
```

1.5.7 GetDriveOff（）

函数功能

获取对应 timingblock 对应 pin 的 DriveOff。

使用说明

无。

函数原型

```
double GetDriveOff(const string& strTimingBlock, const string& strPinName)
```

参数说明

参数	说明
strTimingBlock	Timing Block 的名称。
strPinName	Pin 的名称。

返回值

类型	说明
double	DriveOff 的数据。

示例

```
string strtimingblock = "timingblock";
string strPinName = "Pin1";
double ddriveoff = TheSoft.Timing().GetDriveOff(strtimingblock,strPinName);
```

1.5.8 GetPeriod（）

函数功能

获取 TimingBlock 对应 TimingSet 的 Period 信息。

使用说明

无。

函数原型

```
double GetPeriod(const string& strTimingBlock, const string& strTimingset);
```

参数说明

参数	说明
strtimingblock	Timing 的 Block 的名称。
strtimingset	TimingSet 的名称。

返回值

类型	说明
double	周期，单位 s。

示例

```
string strtimingblock = "timingblock";
string strtimingset = "timingset";
double dperiod = TheSoft.Timing().GetPeriod(strtimingblock,strtimingset);
```

1.5.9 SetTiming（）

函数功能

设置当前测试项使用的 TimingBlock 的名称。

使用说明

无。

函数原型

```
int SetTiming(const string& strTimingBlockName);
```

参数说明

参数	说明
strTimingBlockName	Timing 的 Block 的名称。

返回值

类型	说明
int	● 0：设置成功。 ● - 1：设置失败。

示例

```
TheSoft.Timing().SetTiming("default_timing");
```

1.6 Specs

1.6.1 GetComment（）

函数功能

获取对应表达式的注释信息。

使用说明

无。

函数原型

```
string GetComment(const string& specsName);
```

参数说明

参数	说明
specsName	表达式的名称。

返回值

类型	说明
String	返回表达式的注释信息。

示例

```
string specsName = "DCspecs1";
string strComt = TheSoft.Specs().GetComment(specsName);
```

1.6.2 GetExpression（）

函数功能

获取对应表达式的字符串。

使用说明

无。

函数原型

```
string GetExpression(const string& specsName);
```

参数说明

参数	说明
specsName	表达式的名称。

返回值

类型	说明
String	返回表达式的字符串值。

示例

```
string specsName = "DCspecs1";
string strExp = TheSoft.Specs().GetExpression(specsName);
```

1.6.3 GetValue（）

函数功能

获取对应表达式的数值。

使用说明

无。

函数原型

```
double GetValue(const string& specsName)
```

参数说明

参数	说明
specsName	参数的名称。

返回值

类型	说明
double	表达式计算出来的数值。

示例

```
string specsName = "DCspecs1";
```

```
double strVal = TheSoft.Specs().GetValue(specsName);
```

1.6.4 SetExpression（）

函数功能

更新表达式的值。

使用说明

无。

函数原型

```
bool  SetExpression(const string& specName, const string& specValue,
vector<string>& vecError);
```

参数说明

参数	说明
specName	表达式的名称。
specValue	表达式的值。
vecError	设置时的错误信息。

返回值

类型	说明
bool	● true：成功。 ● false：失败。

示例

```
string strSpecExp = "10n";
vector<string> vecErr;
bool res = TheSoft.Specs().SetExpression("ACSpec1", strSpecExp, vecErr);
```

1.6.5 SetExpression（）

函数功能

按工位设置表达式的值，校验失败可把错误信息发送到 message 进行显示，校验成功设置生效，但原有 spec 值不变。

使用说明

无。

函数原型

```
bool SetExpression(const string& specName, const vector<string>& vecValue,
vector<string>& vecError)
```

参数说明

参数	说明
specsName	表达式的名称。
vecValue	每个工位的 spec 值，spec 值个数与工位数一致。
vecError	错误信息。

返回值

类型	说明
Bool	• true：成功。 • false：失败。

示例

```
vector<string> vecNewSpecExp{"10n", "11n", "10.5n"};
vector<string> vecErr;
bool res = TheSoft.Specs().SetExpression("ACSpec1", vecNewSpecExp, vecErr);
```

第二章 TheInst

2.1 Device

2.1.1 GetBoardID（）

函数功能

获取测试机内所插资源板的板卡 ID 号码。

使用说明

无。

函数原型

```
int GetBoardID(const string& strResource, short iBoardNum, string& strBoardID);
```

参数说明

参数	说明
strResource	资源板类型名称。
iBoardNum	资源板板号，板号从 0 开始。
strBoardID	存储对应板卡 ID 号的参数指针。

返回值

类型	说明
int	返回值 0。

示例

```
string strResource;
TheInst.Device().GetBoardID("FVI2", 0, strResource);
```

2.1.2 GetBoardID（）

函数功能

获取测试机内指定槽位资源板的板卡 ID 号码。

使用说明

无。

函数原型

```
GetBoardID(short iSlot, string& strBoardID);
```

参数说明

参数	说明
iSlot	测试机槽位号，有效值为 0～63。
strBoardID	存储对应板卡 ID 号的参数指针。

返回值

类型	说明
int	返回值 0。

示例

```
string strResource;
TheInst.Device().GetBoardID(15, strBoardID);
```

2.1.3 GetBoardCalDate（）

函数功能

获取板卡校准日期 pDate，格式为 YYYY/MM/DD。

使用说明

无。

函数原型

```
int GetBoardCalDate(const string& strResource, short BoardNum, string& strDate);
```

参数说明

参数	说明
strResource	输入需要获取的源。

参数	说明
BoardNum	板卡序号，从 0 开始。
strDate	pDate 传入的大小。

返回值

类型	说明
int	返回值 0。

示例

```
string strDate;
TheInst.Device().GetBoardCalDate("FVI2", 0, strDate)
```

2.1.4 GetBoardCalTime（）

函数功能

获取校准时间 pTime，格式为 HH:MM:SS。

使用说明

无。

函数原型

```
int GetBoardCalTime(const string& strResource, short BoardNum, string& strTime);
```

参数说明

参数	说明
strResource	输入需要获取的源。
BoardNum	板卡序号，从 0 开始。
strTime	获取校准时间。

返回值

类型	说明
int	返回值 0。

示例

```
string strTime;
TheInst.Device().GetBoardCalTime("FVI2", 0, strTime);
```

2.1.5 GetBoardCalDateTime（）

函数功能

获取校准日期和时间 pDateTime，格式为 YYYY/MM/DDHH:MM:SS。

使用说明

无。

函数原型

```
int GetBoardCalDateTime(const string& strResource, short BoardNum, string&
strDateTime);
```

参数说明

参数	说明
strResource	输入需要获取的源。
BoardNum	第几块板卡，从 0 开始。
strDateTime	输出日期和时间。

返回值

类型	说明
int	返回值 0。

示例

```
string strDateTime;
TheInst.Device().GetBoardCalDateTime("FVI2", 0, strDateTime);
```

2.1.6 GetDIO64BoardNumber（）

函数功能

获取当前机台 DIO64 版卡数量，返回值为 0/1/2。

使用说明

无。

函数原型

```
int GetDIO64BoardNumber();
```

参数说明

无。

返回值

类型	说明
int	当前机台 DIO64 版本数量。

示例

```
int nBoardNumber = TheInst.Device().GetDIO64BoardNumber();
```

2.1.7 GetHandlerGPIBAddr（）

函数功能

返回与 Handler 联机的 GPIB 地址（注意是测试机软件设置的地址）。

使用说明

无。

函数原型

```
int GetHandlerGPIBAddr();
```

参数说明

无。

返回值

类型	说明
int	与 Handler 联机的 GPIB 地址。

示例

```
int GPIBAddr = TheInst.Device().GetHandlerGPIBAddr();
```

2.1.8 GetHandlerType（）

函数功能

获取联机测试终端名称。

使用说明

无。

函数原型

```
int GetHandlerType(string& strGetHandlerType, int nStation = 0);
```

参数说明

参数	说明
strGetHandlerType	测试终端名称。
nStaion	测试终端数量。

返回值

类型	说明
int	与 Handler 联机的 GPIB 地址。

示例

```
string chHandlerType;
TheInst.Device().GetHandlerType(chHandlerType);
```

2.1.9 SetHandlerTemperature（）

函数功能

设置 handler 温度。

使用说明

无。

函数原型

```
int SetHandlerTemperature(double dTemp);
```

参数说明

参数	说明
dTemp	设置温度值。

返回值

类型	说明
int	● 0：设置成功。 ● -1：设置失败。

示例

```
TheInst.Device().SetHandlerTemperature(125);
```

2.1.10 GetHandlerTemperature（）

函数功能

读取 handler 各工位的温度。

使用说明

无。

函数原型

```
void GetHandlerTemperature(double dArryTemp[], int iLength);
```

参数说明

参数	说明
dArryTemp	储存各个 Site 的温度。
iLength	工位个数。

返回值

无。

示例

```
int siteNum = 4;
double temp[4];
TheInst.Device().GetHandlerTemperature(temp, 4);
```

2.1.11 GetUserType（）

函数功能

获取登录用户类型，返回 pChGetUserType–admin、engineer、operator。

使用说明

无。

函数原型

```
int GetUserType(string& strGetUserType);
```

参数说明

参数	说明
strGetUserType	登录用户类型。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
string strUserType;
TheInst.Device().GetUserType(strUserType);
```

2.1.12 IFFVI2_Get_Master_FPGAVer（）

函数功能

获取 IFFVI2 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int IFFVI2_Get_Master_FPGAVer(int nIFFVI2n, int &nFpgaVer, string& strFpgaVer);
```

参数说明

参数	说明
nIFFVI2n	板卡号（0、1）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
int nFpgaVer;

string strFpgaVer;        //注：字符串长度需≥7

int nResult = TheInst.Device().IFFVI2_Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.1.13 IFFVI8_Get_Master_FPGAVer（）

函数功能

获取 IFFVI8 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int IFFVI8_Get_Master_FPGAVer(int nIFFVI8n, int &nFpgaVer, string& strFpgaVer);
```

参数说明

参数	说明
nIFFVI8n	板卡号（0、1）。

参数	说明
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
int nFpgaVer;

string strFpgaVer;        //注：字符串长度需≥7

int nResult = TheInst.Device().IFFVI8_Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.1.14 IFFVI2_GetVer（）

函数功能

获取 IFFVI2 板卡的模块版本信息。

使用说明

返回值类型为以 M 开头的字符串。

函数原型

```
int IFFVI2_GetVer(string& strVer, int nIFFVI2n);
```

参数说明

参数	说明
strVer	返回版本信息的变量指针。
nIFFVI2n	板卡号（0、1）。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
int nFpgaVer;
string strVer;
int nResult = TheInst.Device().IFFVI2_GetVer(strVer, 0);
```

2.1.15 IFFVI8_GetVer（）

函数功能

获取 IFFVI8 板卡的模块版本信息。

使用说明

返回值类型为以 M 开头的字符串。

函数原型

```
int IFFVI8_GetVer(string& strVer, int nIFFVI8n);
```

参数说明

参数	说明
strVer	返回版本信息的变量指针。
nIFFVI8n	板卡号（0、1）。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
string strVer;
int nResult = TheInst.Device().IFFVI8_GetVer(strVer, 0);
```

2.1.16 GPIB

2.1.16.1 Init（）

函数功能

用于 GPIB 的初始化。

使用说明

无。

函数原型

```
int Init();
```

参数说明

无。

返回值

类型	说明
int	● 0：正常。 ● 非 0；不正常。

示例

```
TheInst.Device().GPIB().Init();
```

2.1.16.2 Read（）

函数功能

读取 GPIB 连接设备的信息。

使用说明

无。

函数原型

```
string Read(int nDevNumber);
```

参数说明

参数	说明
nDevNumber	GPIB 地址。

返回值

类型	说明
string	返回值 GPIB 设备信息。

示例

```
string str = TheInst.Device().GPIB().Read(20);
```

2.1.16.3 Send（）

函数功能

需要发送到 GPIB 设备信息。

使用说明

无。

函数原型

```
int Send(int nDevNumber, const string& strSendInfo);
```

参数说明

参数	说明
nDevNumber	GPIB 地址。
strSendInfo	发送 GPIB 的信息内容。

返回值

类型	说明
int	0：该函数正常退出。

示例

```
string strSendInfo = "AAA";
TheInst.Device().GPIB().Send(20, strSendInfo);
```

2.1.17 CRS232

2.1.17.1 Close（）

函数功能

结束后使用 RS232_Close 关闭串口。

使用说明

无。

函数原型

```
bool Close(HANDLE *HCOM);
```

参数说明

参数	说明
HCOM	表示传输指令的设备对象端口句柄。

返回值

类型	说明
bool	● 0：正常。 ● 1：异常。

示例

```
HANDLE hCom;
int FinishLot(int iParam = 0);
{
TheInst.Device().RS232().Close(&hCom);
return 0;
}
```

2.1.17.2 Open（）

函数功能

打开 RS232 串口。

使用说明

使用前先调用 RS232_Open 函数打开串口。

函数原型

```
bool Open(HANDLE &HCOM, int Port, DWORD dwBaud = 9600, Parity parity = NoParity,
BYTE DataBits = 8, StopBits stopBits = OneStopBit, FlowControl fc =
NoFlowControl, bool bOverlapped = false);
```

参数说明

参数	说明
HCOM	串口通信变量，与设备连接后，以该变量来标识设备。
Port	端口号。
dwBaud	波特率。
parity	校验方式。
DataBits	数据位，标准值位 5、7、8。
stopBits	停止位。
fc	控制流位。
bOverlapped	是否为异步通信。 ● True：是异步通讯。 ● False：非异步通讯。

返回值

类型	说明
bool	● 0：正常。 ● 1：异常。

示例

```
HANDLE hCom;
int StartLot(int iParam = 0)
{

//连接设备
```

```
bool bOpen =
TheInst.Device().RS232().Open(hCom,2,9600,NoParity,8,OneStopBit,NoFlowControl,0)
;
return 0;
```

2.1.17.3 Read（）

函数功能

串口通信接收信息。

使用说明

无。

函数原型

```
int  Read(HANDLE *HCOM, char *pGetValue, int nValueSize);
```

参数说明

参数	说明
HCOM	传输指令的设备对象端口句柄。
pGetValue	已读取的数据。
nValueSize	已读取的数据长度。

返回值

类型	说明
int	返回读的字节数，小于 0 表示读失败。

示例

```
HANDLE hCom;
bool bOpen = TheInst.Device().RS232().Open(hCom, 2, 9600, NoParity, 8,
OneStopBit, NoFlowControl, 0);
if (bOpen)
{
    char chVer[256] = { 0 };
    int rdSize = TheInst.Device().RS232().Read(&hCom, chVer, 256);
}
```

2.1.17.4 Send（）

函数功能

串口通信发送。

使用说明

无。

函数原型

```
void Send(HANDLE *HCOM, const char* pSetValue);
```

参数说明

参数	说明
HCOM	传输指令的设备对象端口句柄。
pSetValue	指令内容。RS232 通信的发送指令末尾需要加上"\n"。

返回值

无。

示例

```
HANDLE hCom;
TheInst.Device().RS232().Send(&hCom,"CONFigure:RESistance DEF,DEF\n");
Sleep(200);
TheInst.Device().RS232().Send(&hCom,"CONFigure:FREQuency DEF,DEF\n");
Sleep(200);
TheInst.Device().RS232().Send(&hCom,"CONFigure:FRESistance DEF,DEF\n");
Sleep(200);
TheInst.Device().RS232().Send(&hCom,"CONFigure:PERiod DEF,DEF\n");
Sleep(200);
TheInst.Device().RS232().Send(&hCom,"CONFigure:CURRent:DC\n");
```

2.1.17.5 Send（）

函数功能

串口通信发送信息。

使用说明

无。

函数原型

```
int Send(HANDLE *HCOM, const char* pSetValue, int nValueSize);
```

参数说明

参数	说明
HCOM	传输指令的设备对象端口句柄。
pSetValue	要发送的数据。
nValueSize	要发送的数据长度。

返回值

类型	说明
int	返回写的字节数，小于 0 表示写失败。

示例

```
HANDLE hCom;
bool bOpen = TheInst.Device().RS232().Open(hCom, 2, 9600, NoParity, 8,
OneStopBit, NoFlowControl, 0);
if (bOpen)
{
    char snd[8] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07 };
    TheInst.Device().RS232().Send(&hCom, snd, 8);
}
```

2.2 FVI2

2.2.1 GetPinName（）

函数功能

对齐 FVI12 指定物理通道对应的 Pin。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh);
```

参数说明

参数	说明
nPhyChn	物理通道（0，1，2，……，15）。

返回值

类型	说明
string	Pin 的名字。

示例

```
string strPinName = TheInst.FVI12().GetPinName(1);
```

2.2.2 GetAllSinglePinName（）

函数功能

获取 FVI12 板卡所有的 Pin 的名称。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	FVI12 板卡所有的 Pin 的名称。

示例

```
vector<string> strPinName;
```

```
strPinName = TheInst.FVI12().GetAllSinglePinName(); //获取 FVI12 板卡所有的 Pin 的名字
```

2.2.3 FVI2-DCVI

2.2.3.1 类型定义

emVI_RelayState 类型定义

类型	Enumeration	
描述	继电器状态。	
元素	ON=1	闭合。
	OFF=0	关断。
头文件	UserDef.h	

emVI_CurrLmt 类型定义

类型	Enumeration	
描述	开启或关闭大电流保护功能。	
元素	CurrLmt_ON=0	开启大电流保护功能。
	CurrLmt_OFF=1	关闭大电流保护功能。
头文件	UserDef.h	

emVI_MeasType 类型定义

类型	Enumeration	
描述	测量类型。	
元素	emVI_MV=0	电压。
	emVI_MI=1	电流。
头文件	UserDef.h	

emVI_MeasMode 类型定义

类型	Enumeration	
描述	测量模式。	
元素	emVI_Aver	平均值。
	emVI_Max	最大值。
	emVI_Min	最小值。
头文件	UserDef.h	

emOutMode 类型定义

类型	Enumeration	
描述	工作模式。	
元素	OUTMODE_FV=0	电压源。
	OUTMODE_FI=1	电流源。
	OUTMODE_INIT=2	初始化模式。
头文件	UserDef.h	

emKelvinMode 类型定义

类型	Enumeration	
描述	Kelvin 工作模式。	
元素	Kelvin_H=0	仅测试高端 Kelvin 电阻值（同 Bank 内并测，不会计算低端 Kelvin 电阻）。
	Kelvin_L=1	测试低端 Kelvin 电阻值。
	Kelvin_HL=2	同时测试高端低端 Kelvin 电阻值（同 Bank 内并测，会多次计算低端 Kelvin 电阻）。
	Kelvin_SELF=3	测试内部电阻值。
头文件	UserDef.h	

emVI_KelCalMode 类型定义

类型	Enumeration	
描述	是否使用使用 Kelvin 校准修正。	
元素	KelCal_ON = 0	使用 Kelvin 校准修正，即使用校准后的环路阻抗进行修正。
	KelCal_OFF = 1	不使用 Kelvin 校准修正，即不使用校准后的环路阻抗进行修正。
头文件	UserDef.h	

2.2.3.2 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，从起始值到终点值输出建立时间为 dTms。

使用说明

无。

函数原型

```
int CapLoad(const string& strPinName, double dStartOutVal, double dEndOutVal,
double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dStartOutVal	源输出的起始值。 ● 当 PoutMode=OutMode_FV 时，为电压输出值（–50 – +50）V。 ● 当 PoutMode=OutMode_FI 时，为电流输出值（–10000～+10000）mA。
dEndOutVal	源输出的终点值。 ● PoutMode=OutMode_FV 时，为电压输出值（–50 – +50）V。 ● PoutMode=OutMode_FI 时，为电流输出值（–10000～+10000）mA。
dTms	源输出值变化到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● - 1：Pin 不存在。 ● - 2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";

TheInst.FVI2().DCVI().CapLoad(strPinName,5,2,2);//setmode 模式为 FV 模式时,该函数实现

功能为：FVI2 的 strPinName 通道电压实现从 5V 到 2V,时间跨度为 2ms。
```

2.2.3.3 CapLoad（）

函数功能

根据 FVI2_SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTms。

使用说明

该函数与 FVI2_SetOutValSlow 函数不同的是，此函数无需设置起始输出值（系统内部会自动获取）。

函数原型

```
int CapLoad(const string& strPinName, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dEndOutVal	源输出的终点值。 ● PoutMode=OutMode_FV 时，为电压输出值（–50–+50）V。 ● PoutMode=OutMode_FI 时，为电流输出值（–10000～+10000）mA。
dTms	源输出值变化到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().CapLoad(strPinName, 10, 5);
```

2.2.3.4 Connect（）

函数功能

控制电源通道输出继电器及 Kelvin 继电器的闭合或关断，内部输出值不变。

使用说明

- 该函数使用时应避免带负载切换，以免影响电源继电器的使用寿命。该函数控制的是逻辑通道。

- 调用 Connect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作。

函数原型

```
int Connect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState emRelayState);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().Connect(strPinName,Relay_HF,ON);
TheSoft.TestProc().DelaymS(2);
```

2.2.3.5 Connect（）

函数功能

输出继电器控制。

使用说明

调用该函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 Connect 函数共用。

函数原型

```
int Connect(const string& strPinName, emVI_RelayState isOn);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
isOn	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().Connect(strPinName,ON);
```

2.2.3.6 CurLimitEnable（）

函数功能

大电流保护使能接口。

使用说明

CurLimitEnable 函数设备上电默认使能置 1，开启大电流保护功能。

函数原型

```
int CurLimitEnable(const string& strPinName, emVI_CurrLmt enable);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
enable	开启或关闭大电流保护功能，其类型定义详见 emVI_CurrLmt 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().CurLimitEnable(strPinName, CurrLmt_OFF);
```

2.2.3.7 FastRead（）

函数功能

快速读取 FPGA 存储的电压或电流值。

使用说明

- 测量模式可选平均值、最大值或最小值测量。
- 该函数支持 Group。

函数原型

```
int FastRead(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名字。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().FastRead(strPinName,emVI_MV,emVI_Aver);
```

2.2.3.8 GetCHCondition（）

函数功能

获取通道状态。

使用说明

无。

函数原型

```
int GetCHCondition(int nSite, const string& strPinName, int& nPOutMode, double& dVRange, double& dIRange, double& dIVClampP, double& dIVClampN, double& dOutVal, int& nRelayIsOn);
```

参数说明

参数	说明
nSite	测试工位。
strPinName	Pin 的名称。
nPOutMode	电源工作模式，0（电压源）或者 1（电流源）。
dVRange	电压输出或测量挡位。
dIRange	电流输出或测量挡位。
dIVClampP	电源正钳位设定值。
dIVClampN	电源负钳位设定值。
dOutVal	电源输出值。
nRelayIsOn	输出继电器状态。 ● 0（OFF 关断）。 ● 1（ON 闭合）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
int POutMode, nRelayIsOn;
double dVRange, dIRange, dIVClampP, dIVClampN, dOutVal;
TheInst.FVI2().DCVI().GetCHCondition(0, strPinName, POutMode, dVRange, dIRange,
dIVClampP, dIVClampN, dOutVal, nRelayIsOn);
```

2.2.3.9 GetAverResult（）

函数功能

获取板卡某通道测量结果的平均值。

使用说明

该函数支持 Group。

函数原型

```
int GetAverResult(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI2().DCVI().GetAverResult(strPinName,emVI_MV);// 获取 FVI2  strPinName

通道电压测量平均值。
```

2.2.3.10 GetMeasResult（）

函数功能

测量类型，可选电压或电流。

使用说明

测量模式可选平均值、最大值或最小值测量。

函数原型

```
int GetMeasResult(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode, int nIsShowData);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	参数基本无效，可设置任意值，一般默认为 0，虚拟示波器直接调用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().GetMeasResult(strPinName,emVI_MV,emVI_Aver);
```

2.2.3.11 GetMeasResult（）

函数功能

用于获取测量电流，或电压的所有点数的测量值。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, int nSite, int nSampNum,
emVI_MeasType emType, double *dDataRead) ;
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
nSite	对应工位号。
nSampNum	采样点数，对应设置的采样点数，确定 dDataRead 的大小。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dDataRead	对应通道的电压或电流数据。

返回值

类型	说明
int	● 0：成功。 ● - 1：PinName 不存在或异常。

示例

```
const string strPinName = "Pin1";

double dDataRead[25] = {0};

TheInst.FVI2().DCVI().SetMode(strPinName,OUTMODE_FV,FVI2_10V,FVI2_10mA) ;

TheInst.FVI2().DCVI().SetOutVal(strPinName,5);
TheInst.FVI2().DCVI().MeasureVI(strPinName,25,0.01);

TheInst.FVI2().DCVI().GetMeasResult(strPinName , 0,  25 , emVI, dDataRead);
```

2.2.3.12 KelvinInit（）

函数功能

FVI2 板初始化，断开输出继电器及 Kelvin 继电器并清零。

使用说明

调用 FVI2_KelvinInit 函数后，为确保继电器完全断开，需要至少延时 3ms 再执行其他操作。此延时可与其他通道 KelvinInit 函数共用。

函数原型

```
int KelvinInit(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● - 1：Pin 不存在。 ● - 2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
#define SiteNumb  4
TheInst.FVI2().DCVI().KelvinSetMode(strPinName, Kelvin_H, OUTMODE_FV, FVI2_2V,
FVI2_10mA);
TheSoft.TestProc().DelaymS(2);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 1);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinMeasure(strPinName, 0.01, 10);
TheSoft.TestParam().GetResultAll(SiteNumb, dResult);
TheSoft.TestParam().LogResultAll(0, SiteNumb, 0, dResult);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 0);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinInit(strPinName);
TheSoft.TestProc().DelaymS(3);
```

2.2.3.13 KelvinMeasure（）

函数功能

Kelvin 环路电阻值测量。

使用说明

可设置采样间隔时间，采样点数。

函数原型

```
int KelvinMeasure(const string& strPinName, double dSampleTms, uint32_t
uSampNum, emVI_KelCalMode emMode = KelCal_ON);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
dSampleTms	采样间隔时间，可选返回（0.01ms～30ms）。
uSampNum	采样点数，可选范围（1～2048）
emMode	测量模式设置，其类型定义详见 emVI_KelCalMode 类型定义。 该参数默认值为 KelCal_ON，表示使用校准后的环路阻抗进行修正。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
 TheInst.FVI2().DCVI().KelvinMeasure(strPinName,0.01,100,KelCal_ON);
```

2.2.3.14 KelvinSetMode（）

函数功能

FVI2 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI2_VRange emVRange, emFVI2_IRange emIRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。 emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时为电压输出挡位设定值，可选挡位 FVI2_100mV，FVI2_1V，FVI2_2V，FVI2_5V，FVI2_10V，FVI2_20V，FVI2_50V。 ● POutMode=OUTMODE_FV，最小电压挡位 FVI2_100mV 不能作为输出源使用。
emIRange	电流输出或测量挡位。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FVI2_10uA，FVI2_100uA，FVI2_1mA，FVI2_10mA，FVI2_100mA，FVI2_1000mA，FVI2_10000mA。 ● POutMode=OUTMODE_FI 时，最小电流挡位 FVI2_10uA 不能作为输出源使用。
dIVClampP	电源正钳位设置。 ● POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0.25%IRang～10000.0）mA。 ● POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0.25%VRang－50.0）V。

参数	说明
dIVClampN	电源负钳位设置。 ● POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（－10000.0－（－0.25%IRang））mA。 ● POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（－50.0－（－0.25%VRang））V。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
#define SiteNumb  4
TheInst.FVI2().DCVI().KelvinSetMode(strPinName, Kelvin_H, OUTMODE_FV, FVI2_2V,
FVI2_10mA);
TheSoft.TestProc().DelaymS(2);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 1);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinMeasure(strPinName, 0.01, 10);
TheSoft.TestParam().GetResultAll(SiteNumb, dResult);
TheSoft.TestParam().LogResultAll(0, SiteNumb, 0, dResult);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 0);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinInit(strPinName);
TheSoft.TestProc().DelaymS(3);
```

2.2.3.15 KelvinSetMode（）

函数功能

FVI2 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI2_VRange emVRange, emFVI2_IRange emIRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时为电压输出挡位设定值，可选挡位 FVI2_100mV，FVI2_1V，FVI2_2V，FVI2_5V，FVI2_10V，FVI2_20V，FVI2_50V。 ● POutMode=OUTMODE_FV，最小电压挡位 FVI2_100mV 不能作为输出源使用。
emIRange	电流输出或测量挡位。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FVI2_10uA，FVI2_100uA，FVI2_1mA，FVI2_10mA，FVI2_100mA，FVI2_1000mA，FVI2_10000mA。 ● POutMode=OUTMODE_FI 时，最小电流挡位 FVI2_10uA 不能作为输出源使用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
#define SiteNumb  4
TheInst.FVI2().DCVI().KelvinSetMode(strPinName, Kelvin_H, OUTMODE_FV, FVI2_2V, FVI2_10mA);
TheSoft.TestProc().DelaymS(2);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 1);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinMeasure(strPinName, 0.01, 10);
```

```
TheSoft.TestParam().GetResultAll(SiteNumb, dResult);
TheSoft.TestParam().LogResultAll(0, SiteNumb, 0, dResult);
TheInst.FVI2().DCVI().SetOutVal(strPinName, 0);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().KelvinInit(strPinName);
TheSoft.TestProc().DelaymS(3);
```

2.2.3.16 Init（）

函数功能

FVI2 板初始化，先清零（如果原先是电流源就是电流给到 0，如果是电压源就是电压给到 0），然后再把挡位设置成 10V/10mA 以及断开输出继电器。

使用说明

- 调用 Init 函数后，为确保继电器完全断开，需要至少延时 2ms 再执行其他操作；此延时可与其他通道 Init 函数共用。
- 该函数支持 Group。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().Init(strPinName);
```

2.2.3.17 MeasureI（）

函数功能

测量通道输出电流(mA)，采样次数（25），采样时间间隔（0.01ms）固定，快速测量且返回平均值。

使用说明

该函数支持 Group。

函数原型

```
int MeasureI(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().MeasureI(strPinName);
```

2.2.3.18 MeasureV（）

函数功能

快速测量测量通道输出电压（V），采样次数（25），采样时间间隔（0.01ms）固定并返回平均值，实测值放在 pSite > dRealData[i]中，i 与实际工作的 Site（0～63）相对应。多 Site 并测，并测工作将自动完成，并放到对应的 dRealData 中。

使用说明

- 此函数不同于 MeasureVSamp（）函数，其采样次数（25）、采样时间间隔（0.01ms）为固定

值，且返回平均值。

- MeasureVSamp（）采样次数，采样时间间隔可设置，且可返回最大值，最小值或平均值。
- 该函数支持 Group。

函数原型

```
int MeasureV(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().MeasureV(strPinName);
```

2.2.3.19 MeasureISamp（）

函数功能

测量电流值。

使用说明

- 该函数支持 Group。
- 测量模式可选平均值、最大值或最小值。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureISamp(const string& strPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样时间间隔，可选范围（0.01ms～30ms）。
nSampleN	采样次数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
 TheInst.FVI2().DCVI().MeasureISamp(strPinName,emVI_Aver,0.01,100);
```

2.2.3.20 MeasureVSamp（）

函数功能

测量电压值。

使用说明

- 该函数支持 Group。
- 测量模式可选平均值、最大值或最小值。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureVSamp(const string& strPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样时间间隔，可选范围（0.01ms～30ms）。
nSampleN	采样次数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
 TheInst.FVI2().DCVI().MeasureVSamp(strPinName,emVI_Aver,0.01,100);
```

2.2.3.21 MeasureIRS（）

函数功能

测量通道输出电流（mA）。

使用说明

测试时可重新选择测量范围，确定新的测量量程，以获得更精确的测量值，测量完成后将自动恢复原来的量程。

函数原型

```
int MeasureIRS(const string& strPinName, emFVI2_IRange emIRang, double dTms);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。

参数	说明
emIRang	测量范围（mA），用于选择电流测量量程。 可选挡位 FVI2_10uA、FVI2_100uA、FVI2_1mA、FVI2_10mA、FVI2_100mA、FVI2_1000mA、FVI2_10000mA。
dTms	测量等待时间（ms）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
double dResult; TheInst.FVI2().DCVI().SetMode(strPinName, OUTMODE_FV, FVI2_10V, FVI2_1000mA);
TheSoft.TestProc().DelaymS(3);
TheInst.FVI2().DCVI().SetOutVal(strPinName,5);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().MeasureIRS(strPinName,FVI2_10mA, 5);
TheSoft.TestParam().GetResult(0, dResult);
TheSoft.TestParam().LogResult(0, 0, 0, dResult);
```

2.2.3.22 MeasureVI（）

函数功能

VI 源测量。

使用说明

该函数支持 Group。

函数原型

```
int MeasureVI(const string& strPinName, uint32_t uSampNum, double dSampTms, bool bIsDelay=true);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
uSampNum	采样点数。
dSampTms	采样间隔，单位毫秒。
bIsDelay	是否延时，选择 true 或者 false，此处需要进行缺省处理，默认为 true，客户可选择使用 false。 ● true：延时。 ● false：不延时。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().MeasureVI(strPinName,25,0.01,False);
```

2.2.3.23 RespondSelect（）

函数功能

该函数用于调节板卡响应速度，通过设置不同 mode 切换不同的积分组合，从而调节响应时间。

使用说明

- 当 mode=emVI_Resp_0，代表响应速度最快，mode=emVI_Resp_3 代表响应速度最慢。
- 响应速度随数字变大而变慢。

函数原型

```
int RespondSelect(const string& strPinName, emVI_RespondMode emMode);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
emMode	选择积分电容响应速度模式。 ● emVI_Resp_0，最快。 ● emVI_Resp_1。 ● emVI_Resp_2。 ● emVI_Resp_3，最慢。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().RespondSelect(strPinName,emVI_Resp_0);
```

2.2.3.24 SConnect（）

函数功能

控制电源通道输出继电器开关。

使用说明

调用 SConnect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 SConnect 函数共用

函数原型

```
int SConnect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState emRelayState, int site)
```

参数说明

参数	说明
strPinName	单 Pin 名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。
site	要操作的工位号。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SConnect(strPinName, Relay_HF, ON,0);// 断开 Pin1 的 0 工位 HF 继电器
```

2.2.3.25 SetClamp（）

函数功能

根据 SetMode 的量程设置值，设置通道的钳位值。

使用说明

该函数支持 Group。

函数原型

```
int SetClamp(const string& strPinName, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dIVClampP	量程的百分数（0.25～100），正钳位值设置。 ● 当电源设置为 FV 模式时，对应量程为 IRang。 ● 当电源设置为 FI 模式时，对应量程为 VRang。
dIVClampN	量程的百分数（－100～－0.25），负钳位值设置。 ● 当电源设置为 FV 模式时，对应量程为 IRang， ● 当电源设置为 FI 模式时，对应量程为 VRang。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetMode(strPinName, OUTMODE_FV, FVI2_10V, FVI2_10mA);
TheInst.FVI2().DCVI().SetClamp(strPinName, 10, -10);
```

2.2.3.26 SetMode（）

函数功能

设置 FVI2 电源通道工作模式，闭合通道输出继电器。

使用说明

- dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。
- 调用 SetMode 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 SetMode 函数共用。
- 该函数支持 Group。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI2_VRange
emVRange, emFVI2_IRange emIRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinNam	单 Pin 或 PinGroup 的名称。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位，可选挡位 FVI2_100mV，FVI2_1V，FVI2_2V，FVI2_5V，FVI2_10V，FVI2_20V，FVI2_50V。 ● 当 emPOutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● 当 emPOutMode=OUTMODE_FI 时，为电压测量挡位设定值，当作为电压源输出时（即 POutMode=OUTMODE_FV），最小电压挡位 100mV 不能作为输出源使用，该挡位仅测量功能。
emIRange	电流输出或测量挡位，可选挡位 FVI2_10uA，FVI2_100uA，FVI2_1mA，FVI2_10mA，FVI2_100mA，FVI2_1000mA，FVI2_10000mA。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电流输出挡位设定值。 ● 当作为电流源输出时（即 POutMode=OUTMODE_FI），最小电流挡位 10uA 不能作为输出源使用，该挡位仅测量功能。
dIVClampP	电源正钳位设置（参数可缺省）。 ● 当 POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0.25%IRang～10000.0）mA。 ● 当 POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0.25%VRang－50.0）V。
dIVClampN	电源负钳位设置（参数可缺省）。 ● 当 POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（–10000～（–0.25%IRang））mA。 ● 当 POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（–50–（–0.25%IRang））V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetMode(strPinName,OUTMODE_FV,FVI2_1V,FVI2_100mA,10.0,-10.0);
```

2.2.3.27 SetMode（）

函数功能

设置 FVI2 电源通道工作模式，闭合通道输出继电器。

使用说明

该函数支持 Group。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI2_VRange emVRange, emFVI2_IRange emIRange);
```

参数说明

参数	说明
strPinNam	单 Pin 或 PinGroup 的名称。
emPOutMode	工作模式选择。 emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位，可选挡位 FVI2_100mV，FVI2_1V，FVI2_2V，FVI2_5V，FVI2_10V，FVI2_20V，FVI2_50V。 ● 当 emPOutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● 当 emPOutMode=OUTMODE_FI 时，为电压测量挡位设定值，当作为电压源输出时（即 POutMode=OUTMODE_FV），最小电压挡位 100mV 不能作为输出源使用，该挡位仅测量功能。

参数	说明
emIRange	电流输出或测量挡位，可选挡位 FVI2_10uA，FVI2_100uA，FVI2_1mA，FVI2_10mA，FVI2_100mA，FVI2_1000mA，FVI2_10000mA。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电流输出挡位设定值。 ● 当作为电流源输出时（即 POutMode=OUTMODE_FI），最小电流挡位 10uA 不能作为输出源使用，该挡位仅测量功能。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetMode(strPinName,OUTMODE_FV, FVI2_1V,FVI2_100mA);
```

2.2.3.28 SetIRang（）

函数功能

用于电源通道在加压/加流模式下，资源板带电切换电流挡位。

使用说明

无。

函数原型

```
int SetIRang(const string& strPinName, emFVI2_IRange emIRang);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emIRang	电流挡位，可选挡位 FVI2_10uA，FVI2_100uA，FVI2_1mA，FVI2_10mA，FVI2_100mA，FVI2_1000mA，FVI2_10000mA。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetIRang(strPinName,FVI2_10mA);
```

2.2.3.29 SetVRang（）

函数功能

用于电源通加压/加流模式下，资源板带电切换电压挡位。

使用说明

无。

函数原型

```
int SetVRang(const string& strPinName, emFVI2_VRange emVRang);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emVRang	电压挡位。 可选挡位 FVI2_100mV、FVI2_1V、FVI2_2V、FVI2_5V、FVI2_10V、FVI2_20V、FVI2_50V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetVRang(strPinName, FVI2_10V);
```

2.2.3.30 SetOutVal（）

函数功能

根据 SetMode 设置电源通道的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。
- 该函数支持 Group。

函数原型

```
int SetOutVal(const string& strPinName, double dOutVal);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	源输出值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetOutVal(strPinName,1.0);
```

2.2.3.31 SetOutValLF（）

函数功能

在加压模式，10μA、100μA 测流挡位情况下，由于此时通常波形建立时间较长，通过并联 1mA 挡位，快速输出函数。

使用说明

- 根据 FVI2 的 SetMode 设置，设置电源通道的输出值，并且设置波形边沿上升下降的提速时间。
- 该函数支持 Group。

函数原型

```
int SetOutValLF(const string& strPinName, double dOutVal, double dTms);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	源输出值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。
dTms	波形边沿上升下降的提速时间。 设置范围为 0ms～0.5ms，推荐使用 0ms～0.5ms 即可快速输出完毕。 当设置为 0.1 时，为波形快速上升下降的持续时间为 0.1ms。 通道电压源先并联 1mA 挡位，然后输出设定值，0.1ms 后断开并联的 1mA 挡位，此时电压源恢复到原电流挡位，继续建立波形至稳定。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetOutValLF(strPinName,1.0,0.1);
```

2.2.3.32 SetOutValSlow（）

函数功能

根据 SetMode 设置,缓慢输出电源通道的输出值，建立时间为 dTms。

使用说明

- 当设置为电压源模式时，StartOutVal 指输出的起始电压，EndOutVal 指输出的终点电压。
- 当设置为电流源模式时，StartOutVal 指输出的起始电流，EndOutVal 指输出的终点电流。
- 该函数支持 Group。

函数原型

```
int SetOutValSlow(const string& strPinName, double dStartOutVal, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dStartOutVal	源输出的起始值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。
dEndOutVal	源输出的终点值。 ● 当 POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● 当 POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。
dTms	从起始值到终点值的建立时间（ms） ● POutMode=OUTMODE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetMode(strPinName, OUTMODE_FV, FVI2_10V,FVI2_1000mA);
TheSoft.TestProc().DelaymS(3);
TheInst.FVI2().DCVI().SetOutVal(strPinName,5);
TheSoft.TestProc().DelaymS(1);
TheInst.FVI2().DCVI().SetOutValSlow(strPinName, 5, 3, 2);
```

2.2.3.33 SetOutValEachSite（）

函数功能

实现同时对不同 site 设置不同的电压/电流输出值。

使用说明

- 根据 FVI2_SetMode 设置，当设置为电压源模式时，*dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

函数原型

```
int SetOutValEachSite(const string& strPinName, const double* dOutVal);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dOutVal	存放各 Site 对应的输出设定值的数组首地址，对应数组长度最长为 64 位（最大支持 64 工位设置），入参传入对应数组的首地址。根据源输出值的赋值数组，实现多 Site 输出同时设置。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetMode(strPinName, OUTMODE_FV, FVI2_10V, FVI2_10mA, 10, -10);
TheSoft.TestProc().DelaymS(5);
double SetVoltage[4] = {2,4,6,8 };
TheInst.FVI2().DCVI().SetOutValEachSite(strPinName, SetVoltage);
TheSoft.TestProc().DelaymS(5);
double SetVoltage1[4] = { 0.0,0.0,0.0,0.0 };
TheInst.FVI2().DCVI().SetOutValEachSite(strPinName, SetVoltage1);
TheSoft.TestProc().DelaymS(5);
```

2.2.3.34 SetOutValSite（）

函数功能

根据 FVI2_SetMode 设置，设置电源通道指定 Site 的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

函数原型

```
int SetOutValSite(const string& strPinName, int nSite, double dOutVal);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
nSite	对应 Site 通道号（0、1、2、3、…、31）。

参数	说明
dOutVal	源输出值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50V–+50.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–10000mA～+10000.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI2().DCVI().SetOutValSite(strPinName, 0, 3);
```

2.2.4 FVI2-VIKR

2.2.4.1 RelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int RelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().RelayOn("KR_BS2_FVI2_H1, KR_HG2_FVI12, KR_LX2_FVI12");
```

2.2.4.2 RelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().RelaySetOn("KR_BS2_FVI2_H1, KR_HG2_FVI12, KR_LX2_FVI12");
```

2.2.4.3 RelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOff(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().RelaySetOff("KR_BS2_FVI2_H1, KR_HG2_FVI12, KR_LX2_FVI12");
```

2.2.4.4 SiteRelayOn（）

函数功能

设置所有 Site 需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int SiteRelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().SiteRelayOn("KR_BS2_FVI2_H1, KR_HG2_FVI12, KR_LX2_FVI12");
```

2.2.4.5 SRelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（物理通道）。

使用说明

无。

函数原型

```
int SRelayOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().SRelayOn(std::vector<int>{0, 1, 2, 3});
```

2.2.4.6 SRelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().SRelaySetOn(std::vector<int>{0, 1, 2, 3});
```

2.2.4.7 SRelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOff(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI2().VIKR().SRelaySetOff(std::vector<int>{0, 1, 2, 3});
```

2.2.5 FVI2-VOSC

2.2.5.1 GetThd（）

函数功能

FVI2 谐波测量模式选择，可选失真度测量或有效值测量测量。

使用说明

测量模式可选择电压测量或电流测量。

函数原型

```
int GetThd(const string& strPinName, emVI_MeasType emType = emVI_MV, int nData);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
emType	选择测量模式，其类型定义详见 emVI_MeasType 类型定义。
nData	数据类型。 ● 0：有效值。 ● 3：失真度（%）。 ● 4：失真度（dB）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().VOSC().GetThd(strPinName, emVI_MV,0);
```

2.2.5.2 SetSampNum（）

函数功能

设置当前通道的采样点数与测量类型。

使用说明

无。

函数原型

```
int SetSampNum(const string& strPinName, int nSampNum, int nType = 2);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSampNum	采样点数。
nType	测量数据类型。 ● 0：电压。 ● 1：电流。 ● 2：初始化模式。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
TheInst.FVI2().VOSC().SetSampNum("Pin1", 100, 0);
```

2.2.6 FVI2-Utility

2.2.6.1 Get_Master_FPGAVer（）

函数功能

获取 FVI2 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int Get_Master_FPGAVer(int nFVI2n, int& nFpgaVer, char* strFpgaVer);
```

参数说明

参数	说明
nFVI2n	板卡号（0、1）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

输出参数

参数	说明
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
string strPinName = "Pin1";
int nFpgaVer;
char strFpgaVer[256]={'\0'};
```

```
//字符串长度需≥7
TheInst.FVI2().Utility().Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.2.6.2 Get_Slave_FPGAVer（）

函数功能

获取 FVI2 板卡从机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int Get_Slave_FPGAVer(int nFVI2n, int& nFpgaVer1, char* strFpgaVer1, int&
nFpgaVer2, char* strFpgaVer2);
```

参数说明

参数	说明
nFVI2n	板卡号（0、1）
nFpgaVer1	返回从机 1FPGA 版本号的变量。
strFpgaVer1	返回从机 1FPGA 版本号的变量指针。
nFpgaVer2	返回从机 2FPGA 版本号的变量。
strFpgaVer2	返回从机 2FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
int nFpgaVer1, nFpgaVer2;
char strFpgaVer1[10]={'\0'};        //注：字符串长度需≥7
char strFpgaVer2[10]={'\0'};        //注：字符串长度需≥7
```

```
TheInst.FVI2().Utility().Get_Slave_FPGAVer(0,nFpgaVer1,strFpgaVer1,
nFpgaVer2,strFpgaVer2);
```

2.2.6.3 GetVer（）

函数功能

获取 FVI2 板卡的模块版本信息。

使用说明

返回值类型为以 M 开头的字符串。

函数原型

```
int GetVer(char* chVer, int nFVI2n);
```

参数说明

参数	说明
chVer	返回版本信息的变量指针。
nFVI2n	板卡号（0、1、2、3……）。

返回值

类型	说明
int	返回值为 0。

示例

```
char chVer [10]={'\0'};
TheInst.FVI2().Utility().GetVer(chVer, 0);
```

2.2.6.4 GetTemperature（）

函数功能

获取指定通道温度。

使用说明

无。

函数原型

```
double GetTemperature(const string& strPinName, int site)
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
site	要操作的工位号。

返回值

类型	说明
double	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI2().Utility().GetTemperature(strPinName, 0);//获取 pin1 对应逻辑通道的 0

工位的温度
```

2.2.6.5 ReadMultimeterModel（）

函数功能

获取 FVI2 板卡的上一次校准校验所使用的万用表型号信息。

使用说明

返回值类型为字符串。

函数原型

```
int ReadMultimeterModel(char* MultimeterModel, int nCard)
```

参数说明

参数	说明
MultimeterModel	返回万用表信息的变量指针。
nCard	板卡号（0、1）。

MultimeterModel 的取值说明如下所示：

参数	说明
3458	3458A 万用表读取到的值
2000	2000 万用表读取到的值
34401	34401 万用表读取到的值
34410	34410 万用表读取到的值
34461	34461 万用表读取到的值
万用表读取到的值 32–bit_ADC	内部 32–bit_ADC
0	其他情况读取到的值

返回值

类型	说明
int	返回值为 0。

示例

```
char MultimeterModel[11]={'\0'};        //注：字符串长度需≥11
TheInst.FVI2().Utility().ReadMultimeterModel(MultimeterModel,0);
```

2.3 FVI8

2.3.1 FVI8_GetPinName（）

函数功能

对齐 FVI8 指定物理通道对应的 Pin。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh)
```

参数说明

参数	说明
nPhyChn	物理通道。

返回值

类型	说明
string	Pin 的名字。

示例

```
string strPinName = TheInst.FVI8().GetPinName(0);
```

2.3.2 FVI8_GetAllSinglePinName（）

函数功能

获取 FVI8 板卡所有的 Pin 的名字。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	FVI8 板卡所有的 Pin 的名字。

示例

```
vector<string> strPinName;
strPinName = TheInst.FVI8().GetAllSinglePinName();
```

2.3.3 FVI8-DCVI

2.3.3.1 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，从起始值到终点值输出建立时间为 dTms。

使用说明

无。

函数原型

```
int CapLoad(const string& strPinName, double dStartOutVal, double dEndOutVal,
double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dStartOutVal	源输出的起始值。 ● 当 PoutMode=OutMode_FV 时，为电压输出值（－50V－＋50V）。 ● 当 PoutMode=OutMode_FI 时，为电流输出值（－10000mA～＋10000mA）。
dEndOutVal	源输出的终点值。 ● 当 PoutMode=OutMode_FV 时，为电压输出值（－50V－＋50V）。 ● 当 PoutMode=OutMode_FI 时，为电流输出值（－10000mA～＋10000mA）。
dTms	源输出值变化到终点值的建立时间（ms）。 ● 当 PoutMode=OutMode_FV 时，为电压输出值（－50V－＋50V）。 ● 当 PoutMode=OutMode_FI 时，为电流输出值（－10000mA～＋10000mA）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetMode(strPinName, OUTMODE_FV, FVI8_10V, FVI8_10mA, 10,-10);//源设置为电压源模式
TheSoft.TestProc().DelaymS(5);
TheInst.FVI8().DCVI().SetOutValSite(strPinName,0,2);  //第一个工位的 0 通道输出 2V 电压
TheInst.FVI8().DCVI().SetOutValSite(strPinName,1,3);  //第二个工位的 0 通道输出 3V 电压
TheSoft.TestProc().DelaymS(5);
TheInst.FVI8().DCVI().CapLoad(strPinName,5,10,2);// 在 2ms 内将各工位上的 strPinName 通道电压输出到 10V
TheSoft.TestProc().DelaymS(5);
```

2.3.3.2 CapLoad（）

函数功能

根据 FVI8_SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTms。

使用说明

该函数与 FVI8_SetOutValSlow 函数不同的是，此函数无需设置起始输出值（系统内部会自动获取）。

函数原型

```
int CapLoad(const string& strPinName, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dEndOutVal	源输出的终点值。 ● 当 PoutMode=OutMode_FV 时，为电压输出值（－50V－＋50V）。 ● 当 PoutMode=OutMode_FI 时，为电流输出值（－10000mA～＋10000mA）。
dTms	源输出值变化到终点值的建立时间（ms）。 ● 当 POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● 当 POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetMode(strPinName, OUTMODE_FV, FVI8_10V, FVI8_10mA, 10,-10);//源设置为电压源模式
 TheSoft.TestProc().DelaymS(5);
TheInst.FVI8().DCVI().SetOutValSite(strPinName,0,2); //第一个工位的 0 通道输出 2V 电压
TheInst.FVI8().DCVI().SetOutValSite(strPinName,1,3);  //第二个工位的 0 通道输出 3V 电压
TheSoft.TestProc().DelaymS(5);
TheInst.FVI8().DCVI().CapLoad(strPinName,5,2); //在 2ms 时间内将各工位上的 0 通道电压变为 5V,此处表明一,二工位 0 通道分别由 2V 和 3V 升到 5V,建立时间为 2ms
```

2.3.3.3 Connect（）

函数功能

控制电源通道输出继电器及 Kelvin 继电器的闭合或关断，内部输出值不变。

使用说明

- 该函数使用时应避免带负载切换，以免影响电源继电器的使用寿命。
- 该函数控制的是逻辑通道。
- 调用 Connect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作。

函数原型

```
int Connect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState emRelayState);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().Connect(strPinName,Relay_HF,ON);
```

2.3.3.4 Connect（）

函数功能

输出继电器控制。

使用说明

调用 FVI8_Connect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 Connect 函数共用。

函数原型

```
int Connect(const string& strPinName, emVI_RelayState isOn);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
isOn	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().Connect(strPinName,ON);
```

2.3.3.5 CurLimitEnable（）

函数功能

大电流保护使能接口。

使用说明

CurLimitEnable 函数设备上电默认使能置 1，开启大电流保护功能。

函数原型

```
int CurLimitEnable(const string& strPinName, emVI_CurrLmt enable);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
enable	开启或关闭大电流保护功能，其类型定义详见 emVI_CurrLmt 类型定义

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().CurLimitEnable(strPinName,CurrLmt_OFF);
```

2.3.3.6 FastRead（）

函数功能

快速读取 FPGA 存储的电压或电流值。

使用说明

- 该函数支持 Group。
- 测量模式可选平均值、最大值或最小值测量。

函数原型

```
int FastRead(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().FastRead(strPinName,emVI_MV,emVI_Aver);
```

2.3.3.7 Init（）

函数功能

FVI8 板初始化，先清 0（如果原先是电流源就是电流给到 0，如果是电压源就是电压给到 0），然后在把挡位设置成 10V/10mA 以及断开输出继电器。

使用说明

- 该函数支持 Group。
- 调用 FVI8_Init 函数后，为确保继电器完全断开，需要至少延时 2ms 再执行其他操作。此延时可与其他通道 Init 函数共用。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().Init(strPinName);
```

2.3.3.8 GetAverResult（）

函数功能

获取测量电压或电流数据的平均值。

使用说明

该函数支持 Group。

函数原型

```
int GetAverResult(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().GetAverResult(strPinName,emVI_MV);
```

2.3.3.9 GetCHCondition（）

函数功能

获取通道状态。

使用说明

无。

函数原型

```
int GetCHCondition(int nSite, const string& strPinName, int& POutMode, double& dVRange, double& dIRange, double& dIVClampP, double& dIVClampN, double& dOutVal, int& nRelayIsOn);
```

参数说明

参数	说明
nSite	测试工位。
strPinName	Pin 的名称。
POutMode	电源工作模式，0（电压源）或者 1（电流源）。
dVRange	电流输出或测量挡位。
dIRange	电流输出或测量挡位。
dIVClampP	电源正钳位设定值。
dIVClampN	电源负钳位设定值。
dOutVal	电源输出值。
nRelayIsOn	输出继电器状态。 ● 0（OFF 关断）。 ● 1（ON 闭合）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
int POutMode, nRelayIsOn;
double dVRange, dIRange, dIVClampP, dIVClampN, dOutVal;
TheInst.FVI8().DCVI().GetCHCondition(0, strPinName, POutMode, dVRange, dIRange,
dIVClampP, dIVClampN, dOutVal, nRelayIsOn);
```

2.3.3.10 GetMeasResult（）

函数功能

电压或电流测量。

使用说明

测量模式可选平均值、最大值或最小值测量。

函数原型

```
int GetMeasResult(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode, int nIsShowData);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	参数基本无效，虚拟示波器直接调用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().GetMeasResult(strPinName,emVI_MV,emVI_Aver);
```

2.3.3.11 GetMeasResult（）

函数功能

用于获取测量电流，或电压的所有点数的测量值。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, int nSite, int nSampNum,
emVI_MeasType emType, double *dDataRead);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
nSite	对应工位号。
nSampNum	采样点数，对应设置的采样点数，确定 dDataRead 的大小。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dDataRead	对应通道的电压或电流数据。

返回值

类型	说明
int	● 0：成功。 ● -1：PinName 不存在或异常。

示例

```
const string strPinName = "Pin1";
double dDataRead[25] = {0};
TheInst.FVI8().DCVI().SetMode(strPinName,OUTMODE_FV,FVI8_10V,FVI8_10mA);
TheInst.FVI8().DCVI().SetOutVal(strPinName,5);
TheInst.FVI8().DCVI().MeasureVI(strPinName,25,0.01);
TheInst.FVI8().DCVI().GetMeasResult(strPinName, 0,  25, emVI, dDataRead);
```

2.3.3.12 KelvinInit（）

函数功能

FVI8 板初始化，断开输出继电器及 Kelvin 继电器并清零。

使用说明

调用该函数后，为确保继电器完全断开，需要至少延时 2ms 再执行其他操作。此延时可与其他通道 KelvinInit 函数共用。

函数原型

```
int KelvinInit(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().KelvinInit(strPinName);
```

2.3.3.13 KelvinMeasure（）

函数功能

Kelvin 环路电阻值测量。

使用说明

可设置采样间隔时间，采样点数。

函数原型

```
int KelvinMeasure(const string& strPinName, double dSampleTms, uint32_t
uSampNum, emVI_KelCalMode emMode = KelCal_ON);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dSampleTms	采样间隔时间，可选返回（0.01ms～30ms）。
uSampNum	采样点数，可选范围（1～2048）
emMode	测量模式设置，该参数默认为 0，其类型定义详见 emVI_KelCalMode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().KelvinMeasure(strPinName,0.01,100, KelCal_ON);
```

2.3.3.14 KelvinSetMode（）

函数功能

FVI8 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI8_VRange emVRange, emFVI8_IRange emIRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。 emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值，可选挡位 FVI8_2V，FVI8_5V，FVI8_10V，FVI8_50V。
emIRange	电流输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电流测量挡位设定值。可选挡位 FVI8_2uA，FVI8_10uA，FVI8_100uA，FVI8_1mA，FVI8_10mA，FVI8_100mA，FVI8_1000mA，FVI8_10000mA。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值，最小电流挡位 FVI8_2uA 不能作为输出源使用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
 string strPinName = "Pin1";

TheInst.FVI8().DCVI().KelvinSetMode(strPinName,Kelvin_H,OUTMODE_FV,FVI8_2V,FVI8_2uA);
```

2.3.3.15 KelvinSetMode（）

函数功能

FVI8 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI8_VRange emVRange, emFVI8_IRange emIRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。

参数	说明
emVRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值，可选挡位 FVI8_2V，FVI8_5V，FVI8_10V，FVI8_50V。
emIRange	电流输出或测量挡位，可选挡位 FVI8_2uA，FVI8_10uA，FVI8_100uA，FVI8_1mA，FVI8_10mA，FVI8_100mA，FVI8_1000mA。 ● POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值，最小电流挡位 FVI8_2uA 不能作为输出源使用。
dIVClampP	电源正钳位设置（可缺省）。 ● POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0.25%IRang–1000.0）mA。 ● POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0.25%VRang–50.0）V。
dIVClampN	电源负钳位设置（可缺省）。 ● POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（–1000.0–（–0.25%Irang））mA。 ● POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（–50.0–（–0.25%Vrang））V。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";

TheInst.FVI8().DCVI().KelvinSetMode(strPinName,Kelvin_H,OUTMODE_FV,FVI8_2V,FVI8_2uA,10,-10);
```

2.3.3.16 MeasureI（）

函数功能

测量通道输出电流（mA），采样次数（25），采样时间间隔（0.01ms）固定，快速测量且返回平均值。

使用说明

该函数支持 Group。

函数原型

```
int MeasureI(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureI(strPinName);
```

2.3.3.17 MeasureISamp（）

函数功能

电流值测量。测量类型选择，可选平均值、最大值或最小值测量。

使用说明

- 该函数支持 Group。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureISamp(const string& strPinName, emVI_MeasMode emMode, double dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，可选范围（0.01ms～30ms）。
nSampleN	采样点数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureISamp(strPinName,emVI_Aver,0.01,100);
```

2.3.3.18 MeasureV（）

函数功能

测量通道输出电压（V）。采样次数（25），采样时间间隔（0.01ms）固定，快速测量且返回平均值。

使用说明

该函数支持 Group。

函数原型

```
int MeasureV(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureV(strPinName);
```

2.3.3.19 MeasureVI（）

函数功能

VI 源测量。

使用说明

该函数支持 Group。

函数原型

```
int MeasureVI(const string& strPinName, uint32_t uSampNum, double dSampleTms,
bool bIsDelay=true);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
uSampNum	采样点数。
dSampleTms	采样间隔，单位毫秒。
bIsDelay	是否延时，选择 true 或者 false，此处需要进行缺省处理，默认为 true，客户可选择使用 false。 ● true：延时。 ● false：不延时。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureVI(strPinName,25,0.01,False);
```

2.3.3.20 MeasureIRS（）

函数功能

测量通道输出电流（mA），测试时可重新选择测量范围，确定新的测量量程，以获得更精确的测量值，测量完成后将自动回复原来的量程。

使用说明

实测值放在“pSite > dRealData[i]”中，i 与实际工作的 Site（0～15）对应，对于多 Site 并测，并测工作将自动完成，并放到对应的 RealData 中。

函数原型

```
int MeasureIRS(const string& strPinName, emFVI8_IRange emIRange, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emIRang	测量范围（mA），用于选择电流测量量程。 可选挡位 FVI8_2uA、FVI8_10uA、FVI8_100uA、FVI8_1mA、FVI8_10mA、FVI8_100mA、FVI8_1000mA。
dTms	测量等待时间（ms）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureIRS(strPinName,FVI8_10uA,5);
```

2.3.3.21 MeasureVSamp（）

函数功能

电压值测量。测量类型选择，可选平均值、最大值或最小值。

使用说明

- 可设置采样间隔时间，采样点数。
- 该函数支持 Group。

函数原型

```
int MeasureVSamp(const string& strPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，可选范围（0.01ms～30ms）。
nSampleN	采样点数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().MeasureVSamp(strPinName,emVI_Aver,0.01,100);
```

2.3.3.22 RespondSelect（）

函数功能

该函数用于调节板卡响应速度，通过设置不同 mode 切换不同的积分组合，从而调节响应时间。

使用说明

- 当 mode=0，代表响应速度最快，mode=5 代表响应速度最慢。
- 响应速度随数字变大而变慢。

函数原型

```
int RespondSelect(const string& strPinName, emVI_RespondMode emMode);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	选择积分电容响应速度模式。 ● emVI_Resp_0，最快。 ● emVI_Resp_1。 ● emVI_Resp_2。 ● emVI_Resp_3。 ● emVI_Resp_4。 ● emVI_Resp_5，最慢。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().RespondSelect(strPinName, emVI_Resp_0);
```

2.3.3.23 SConnect（）

函数功能

控制电源通道输出继电器开关。

使用说明

调用 SConnect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 SConnect 函数共用。

函数原型

```
int SConnect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState
emRelayState, int site)
```

参数说明

参数	说明
strPinName	单 Pin 名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、 Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、 Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。
site	要操作的工位号。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().SConnect(strPinName, Relay_HF, ON,0);// 断开 Pin1 的 0 工位 HF 继电器
```

2.3.3.24 SetClamp（）

函数功能

根据 FVI8_SetMode 的量程设置值，设置通道的钳位值。

使用说明

该函数支持 Group。

函数原型

```
int SetClamp(const string& strPinName, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dIVClampP	量程的百分数（0.25～100），正钳位值设置。 当电源设置为 FV 模式时，对应量程为 IRang，当电源设置为 FI 模式时，对应量程为 VRang。
dIVClampN	量程的百分数（－100～－0.25），负钳位值设置。 当电源设置为 FV 模式时，对应量程为 IRang，当电源设置为 FI 模式时，对应量程为 VRang。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
 TheInst.FVI8().DCVI().SetMode(strPinName, OUTMODE_FV, FVI8_10V, FVI8_100mA);
 TheInst.FVI8().DCVI().SetClamp(strPinName, 10, -10);
```

2.3.3.25 SetIRang（）

函数功能

用于电源通道在加压/加流模式下，资源板带电切换电流挡位。

使用说明

无。

函数原型

```
int SetIRang(const string& strPinName, emFVI8_IRange emIRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emIRange	电流挡位，可选挡位 FVI8_uA，FVI8_10uA，FVI8_100uA，FVI8_1mA，FVI8_10mA，FVI8_100mA，FVI8_1000mA。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetIRang(strPinName, FVI8_10mA);
```

2.3.3.26 SetMode（）

函数功能

设置 FVI8 电源通道工作模式，闭合通道输出继电器。

使用说明

- dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。
- 当 POutMode=OUTMODE_FV 时，SetMode 后由于有漏电流的存在，输出电压值随着漏电流的变化而变化，最大钳位到挡位值。
- 当 POutMode=OUTMODE_FI 时，输出电流为 0mA。调用 FVI8_SetMode 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 SetMode 函数共用。
- 该函数支持 Group。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI8_VRange
emVRange, emFVI8_IRange emIRange, double dIVClampP, double dIVClampN)
```

参数说明

参数	说明
strPinNam	单 Pin 或 PinGroup 的名字。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电压输出挡位设定值，可选挡位 FVI8_2V，FVI8_5V，FVI8_10V，FVI8_50V。 ● 当 POutMode=OUTMODE_FI 时，为电压测量挡位设定值。
emIRange	电流输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FVI8_2uA，FVI8_10uA，FVI8_100uA，FVI8_1mA，FVI8_10mA，FVI8_100mA，FVI8_1000mA。电流源输出时（即 POutMode=OUTMODE_FI），最小电流挡位 FVI8_2uA 不能作为输出源使用。
dIVClampP	电源正钳位设置（参数可缺省）。 ● POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0.25%IRang–1000.0）mA。 ● POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0.25%VRang–50.0）V。
dIVClampN	源负钳位设置（参数可缺省）。 ● POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（–1000.0–（–0.25%Irang））mA。 ● POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（–50.0–（–0.25%Vrang））V

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
```

```
 TheInst.FVI8().DCVI().SetMode(strPinName, OUTMODE_FV, FVI8_2V, FVI8_2uA, 10.0,
-10.0);
```

2.3.3.27 SetMode（）

函数功能

设置 FVI8 电源通道工作模式，闭合通道输出继电器。

使用说明

该函数支持 Group。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI8_VRange
emVRange, emFVI8_IRange emIRange)
```

参数说明

参数	说明
strPinNam	单 Pin 或 PinGroup 的名称。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电压测量挡位设定值，可选挡位 FVI8_2V，FVI8_5V，FVI8_10V，FVI8_50V。
emIRange	电流输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FVI8_2uA，FVI8_10uA，FVI8_100uA，FVI8_1mA，FVI8_10mA，FVI8_100mA，FVI8_1000mA。 ● 电流源输出时（即 POutMode=OUTMODE_FI），最小电流挡位 FVI8_2uA 不能作为输出源使用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetMode(strPinName,OUTMODE_FV, FVI8_2V, FVI8_2uA;
```

2.3.3.28 SetOutVal（）

函数功能

根据 FVI8_SetMode 设置，设置电源通道的输出值。

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

使用说明

该函数支持 Group。

函数原型

```
int SetOutVal(const string& strPinName, double dOutVal);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	源输出值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–50–+50.0）V。 ● POutMode=OUTMODE_FI 时，为电流输出值（–1000～+1000.0）mA。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetOutVal(strPinName, 1.0);
```

2.3.3.29 SetOutValEachSite（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTms。

使用说明

- 该函数与 SetOutValSlow 函数不同的是，此函数无需设置起始输出值（系统内部会自动获取）。
- 当 SetMode 为电压源模式时，EndOutVal 指输出的终点电压。
- 当 SetMode 为电流源模式时，EndOutVal 指输出的终点电流。
- strPinName 代表的电源通道号为逻辑通道号，即在使用此函数时，是对当前所有工位的同一个通道进行相同的操作。

函数原型

```
int SetOutValEachSite(const string& strPinName, const double* dOutVal)
```

参数说明

参数	说明
strPinName	Pin 的名称。
dOutVal	代表存放各 site 对应的输出设定值的数组首地址，对应数组长度最长为 64 位（最大支持 64 工位设置），入参传入对应数组的首地址。根据源输出值的赋值数组，实现多 site 输出同时设置。 ● 当 POutMode=OUTMODE_FV 时，输出电压（－50V－＋50.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（－1000mA～＋1000.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetOutValEachSite (strPinName,1,5);
```

2.3.3.30 SetOutValLF（）

函数功能

在加压模式，2μA、10μA、100μA 测流挡位情况下，由于此时通常波形建立时间较长，通过并联 1mA 挡位，快速输出函数。

使用说明

- 根据 FVI8 的 SetMode 设置电源通道的输出值，并且设置波形边沿上升下降的提速时间。
- 该函数支持 Group。

函数原型

```
int SetOutValLF(const string& strPinName, double dOutVal, double dTms);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	源输出值。 ● 当 POutMode=OUTMODE_FV 时，输出电压（－50V－＋50.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（－1000mA～＋1000.0mA）。
dTms	波形边沿上升下降的提速时间（设置范围为 0ms～5ms，推荐使用 0ms～5ms 即可快速输出完毕）。 ● 当设置为 0.1 时，为波形快速上升下降的持续时间为 0.1ms。 ● 通道电压源先并联 1mA 挡位，然后输出设定值，0.1ms 后断开并联的 1mA 挡位，此时电压源恢复到原电流挡位，继续建立波形至稳定。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetOutValLF(strPinName, 1.0, 0.1);
```

2.3.3.31 SetOutValSlow（）

函数功能

根据 FVI8_SetMode 设置，缓慢输出电源通道的输出值，建立时间为 dTms。

使用说明

- 该函数支持 Group。
- 当设置为电压源模式时，StartOutVal 指输出的起始电压，EndOutVal 指输出的终点电压。
- 当设置为电流源模式时，StartOutVal 指输出的起始电流，EndOutVal 指输出的终点电流。

函数原型

```
int SetOutValSlow(const string& strPinName, double dStartOutVal, double
dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dStartOutVal	源输出的起始值。 ● 当 POutMode=OUTMODE_FV 时，输出电压（－50V－＋50.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（－1000mA～＋1000.0mA）。
dEndOutVal	源输出的终点值。 ● 当 POutMode=OUTMODE_FV 时，输出电压（－50V－＋50.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（－1000mA～＋1000.0mA）。
dTms	从起始值到终点值的建立时间（ms）。 ● 当 POutMode=OUTMODE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● 当 POutMode=OUTMODE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetOutValSlow(strPinName, 5, 3, 2);
```

2.3.3.32 SetOutValSite（）

函数功能

根据 FVI8 的 SetMode 设置，设置电源通道指定 Site 的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

函数原型

```
int SetOutValSite(const string& strPinName, int nSite, double dOutVal)
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSite	对应 Site 通道号（0，1，2，3，…，63）。
dOutVal	源输出值。 ● 当 POutMode=OUTMODE_FV 时，输出电压（－50V－＋50.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（－1000mA～＋1000.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetOutValSite(strPinName,1,5);
```

2.3.3.33 SetVRang（）

函数功能

用于电源通加压/加流模式下，资源板带电切换电压挡位。

使用说明

无。

函数原型

```
int SetVRang(const string& strPinName, emFVI8_VRange emVRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emVRange	电压挡位。 可选挡位 FVI8_2V，FVI8_5V，FVI8_10V，FVI8_50V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI8().DCVI().SetVRang(strPinName,FVI8_10V);
```

2.3.4 FVI8-VIKR

2.3.4.1 RelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int RelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().RelayOn("KR_BS2_FVI8_H1, KR_HG2_FVI8, KR_LX2_FVI8");
```

2.3.4.2 SRelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（物理通道）。

使用说明

无。

函数原型

```
int SRelayOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().SRelayOn(std::vector<int>{0, 1, 2, 3});
```

2.3.4.3 SiteRelayOn（）

函数功能

设置所有 Site 需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int SiteRelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().SiteRelayOn("KR_BS2_FVI8_H1, KR_HG2_FVI8, KR_LX2_FVI8");
```

2.3.4.4 RelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().RelaySetOn("KR_BS2_FVI8_H1, KR_HG2_FVI8, KR_LX2_FVI8");
```

2.3.4.5 RelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOff(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().RelaySetOff("KR_BS2_FVI8_H1, KR_HG2_FVI8, KR_LX2_FVI8");
```

2.3.4.6 SRelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().SRelaySetOn(std::vector<int>{0, 1, 2, 3});
```

2.3.4.7 SRelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOff(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI8().VIKR().SRelaySetOff(std::vector<int>{0, 1, 2, 3});
```

2.3.5 FVI8- VOSC

2.3.5.1 GetThd（）

函数功能

FVI8 谐波测量模式选择，可选失真度测量或有效值测量。

使用说明

测量模式可选电压测量或电流测量。

函数原型

```
int GetThd(const string& strPinName, emVI_MeasType emType = emVI_MV, int nData = 2);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
nData	数据类型。 ● 0：有效值。 ● 3：失真度（%）。 ● 4：失真度（dB）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI8().DCVI().GetThd(strPinName,emVI_MV,0);
```

2.3.5.2 SetSampNum（）

函数功能

设置当前通道的采样点数与测量类型。

使用说明

无。

函数原型

```
int SetSampNum(const string& strPinName, int nSampNum, int nType = 2);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSampNum	采样点数。
nType	测量数据类型。 ● 0：电压。 ● 1：电流。 ● 2：电阻。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
TheInst.FVI8().VOSC().SetSampNum("Pin1", 100, 0);
```

2.3.6 FVI8-Utility

2.3.6.1 Get_Master_FPGAVer（）

函数功能

获取 FVI8 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int Get_Master_FPGAVer(int nFVI8n, int& nFpgaVer, char* strFpgaVer);
```

参数说明

参数	说明
nFVI8n	板卡号（0、1、2、3…8）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	0：正常。

示例

```
int nFpgaVer;
char strFpgaVer[]={0};
int nMasterVer =
TheInst.FVI8().Utility().Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.3.6.2 Get_Slave_FPGAVer（）

函数功能

获取 FVI8 板卡从机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int Get_SLave_FPGAVer(int nFVI8n, int& nFpgaVer1, char* strFpgaVer1, int&
nFpgaVer2, char* strFpgaVer2);
```

参数说明

参数	说明
nFVI8n	板卡号（0、1、2、3…7）。
nFpgaVer1	返回从机 1FPGA 版本号的变量。
strFpgaVer1	返回从机 1FPGA 版本号的变量。
nFpgaVer2	返回从机 2FPGA 版本号的变量，int 型。
strFpgaVer2	返回从机 2FPGA 版本号的变量指针。

返回值

类型	说明
int	0：正常。

示例

```
string strPinName = "Pin1";
int nFpgaVer1, nFpgaVer2;
char strFpgaVer1[256]={0};
char strFpgaVer2[256]={0};
TheInst.FVI8().Utility().Get_Slave_FPGAVer(0,nFpgaVer1,strFpgaVer1,
nFpgaVer2,strFpgaVer2);
```

2.3.6.3 GetVer（）

函数功能

获取 FVI8 板卡的硬件版本信息。

使用说明

返回值类型为以 V 开头的字符串。

函数原型

```
int GetVer(char* chVer, int nFVI8n);
```

参数说明

参数	说明
chVer	返回版本信息的变量指针。
nFVI8n	板卡号（0、1、2、3…7）。

返回值

类型	说明
int	返回值为 0。

示例

```
string strPinName = "Pin1";
char chVer[256]={0};
TheInst.FVI8().Utility().GetVer(chVer, 0);
```

2.3.6.4 GetTemperature（）

函数功能

获取指定通道温度。

使用说明

无。

函数原型

```
double GetTemperature(const string& strPinName, int site)
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
site	要操作的工位号。

返回值

类型	说明
double	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI8().Utility().GetTemperature(strPinName,0);//获取 pin1 对应逻辑通道的 0 工

位的温度
```

2.3.6.5 ReadMultimeterModel（）

函数功能

获取 FVI8 板卡的上一次校准校验所使用的万用表型号信息。

使用说明

返回值类型为字符串。

函数原型

```
int ReadMultimeterModel(char* MultimeterModel, int nCard);
```

参数说明

参数	说明
MultimeterModel	返回万用表信息的变量指针。
nCard	板卡号（0、1）。

返回值

类型	说明
int	● 3458：3458A 万用表读取到的值。 ● 2000：2000 万用表读取到的值。 ● 34401：34401 万用表读取到的值。 ● 34410：34410 万用表读取到的值。 ● 34461：34461 万用表读取到的值。 ● 万用表读取到的值 32–bit_ADC：内部 32–bit_ADC。 ● 0：其他情况读取到的值。

示例

```
string strPinName = "Pin1";
char chVer [256]={0};
TheInst.FVI8().Utility().ReadMultimeterModel(chVer, 0);
```

2.4 FVI16

2.4.1 GetPinName（）

函数功能

对齐 FVI16 指定物理通道对应的 Pin。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh);
```

参数说明

参数	说明
nPhyChn	物理通道（0，1，2，……，15）。

返回值

类型	说明
string	Pin 的名字。

示例

```
string strPinName = TheInst.FVI16().GetPinName(1);
```

2.4.2 GetAllSinglePinName（）

函数功能

获取 FVI16 板卡所有的 Pin 的名称。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	FVI16 板卡所有的 Pin 的名称。

示例

```
vector<string> strPinName;

strPinName = TheInst.FVI16().GetAllSinglePinName(); //获取 FVI16 板卡所有的 Pin 的名

字
```

2.4.3 FVI16-DCVI

2.4.3.1 类型定义

emGangType 类型定义

类型	Enumeration	
描述	Gang 模式类型。	
元素	OneGang=1	两个通道 Gang。
	TwoGang=2	三个通道 Gang。
	ThreeGang=3	四个通道 Gang。
头文件	UserDef.h	

emFVI16_VRange 类型定义

类型	Enumeration	
描述	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值。	
元素	FVI16_1V = 1	1V 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_5V = 5	5V 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_10V = 10	10V 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_30V = 30	30V 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_2V = 2	2V 挡位。 仅 FVI16Plus 支持。
	FVI16_100V = 100	100V 挡位。 仅 FVI16Plus 支持。
	FVI16_150V = 150	150V。 仅 FVI16Plus 支持。
头文件	UserDef.h	

emFVI16_IRange 类型定义

类型	Enumeration

描述	电流输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值。	
元素	FVI16_1uA = -1	1μA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_10uA = -10	10μA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_100uA = -100	100μA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_1mA = 1	1mA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_10mA = 10	10mA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_100mA = 100	100mA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_200mA = 200	200mA 挡位。 FV116 和 FVI16Plus 均支持。
	FVI16_50mA = 50	仅 FVI16Plus 支持。
头文件	UserDef.h	

2.4.3.2 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，从起始值到终点值输出建立时间为 dTms。

使用说明

无。

函数原型

```
int CapLoad(const string& strPinName, double dStartOutVal, double dEndOutVal,
double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dStartOutVal	源输出的起点值。 ● PoutMode=OutMode_FV 时，FVI16 为电压输出值（－30V－＋30.0V）。FVI16plus 为电压输出值（－30V－＋150.0V）。 ● PoutMode=OutMode_FI 时，为电流输出值（－200mA～＋200mA）。
dEndOutVal	源输出的终点值。 ● PoutMode=OutMode_FV 时，FVI16 为电压输出值（－30V－＋30.0V）。FVI16plus 为电压输出值（－30V－＋150.0V）。 ● PoutMode=OutMode_FI 时，为电流输出值（－200mA～＋200mA）。
dTms	源输出值变化到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().CapLoad(strPinName,5,2,2);//setmode 模式为 FV 模式时,该函数实

现功能为：FVI16 的 strPinName 通道电压或电流实现从 5V 到 2V,时间跨度为 2ms。
```

2.4.3.3 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间 dTms。

使用说明

- 该函数与 SetOutValSlow 函数不同的是，此函数无需设置起始输出值（系统内部会自动获取）。
- 当 SetMode 为电压源模式时，EndOutVal 指输出的终点电压。
- 当 SetMode 为电流源模式时，EndOutVal 指输出的终点电流。

函数原型

```
int CapLoad(const string& strPinName, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dEndOutVal	源输出的终点值。 ● PoutMode=OutMode_FV 时，FVI16 为电压输出值（－30V－＋30.0V）。FVI16plus 为电压输出值（－30V－＋150.0V）。 ● PoutMode=OutMode_FI 时，为电流输出值（－200mA～＋200mA）。
dTms	源输出值变化到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetMode(strPinName, OUTMODE_FV, FVI16_10V, FVI16_10mA,
10,-10);//源设置为电压源模式
 TheSoft.TestProc().DelaymS(5);
TheInst.FVI16().DCVI().SetOutValSite(strPinName,0,2); //第一个工位的 0 通道输出 2V 电
压
```

```
TheInst.FVI16().DCVI().SetOutValSite(strPinName,1,3);  //第二个工位的 0 通道输出 3V 电压
TheSoft.TestProc().DelaymS(5);
TheInst.FVI16().DCVI().CapLoad(strPinName,5,2); //在 2ms 时间内将各工位上的 0 通道电压变为 5V,此处表明一,二工位 0 通道分别由 2V 和 3V 升到 5V,建立时间为 2ms
```

2.4.3.4 Connect（）

函数功能

控制电源通道输出继电器开关。

使用说明

无。

函数原型

```
int Connect(const string& strPinName, emVI_RelayState isOn);
```

参数说明

参数	说明
strPinName	单 pin 或 pinGroup 的名称。
isOn	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().Connect(strPinName,  ON);
```

2.4.3.5 Connect（）

函数功能

控制电源通道输出继电器及 Kelvin 继电器的闭合或关断，内部输出值不变。

使用说明

- 该函数使用时应避免带负载切换影响电源继电器的使用寿命。
- 该函数控制的是逻辑通道。

函数原型

```
int Connect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState emRelayState);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
strPinName = "Pin1";
 TheInst.FVI16().DCVI().Connect(strPinName,Relay_HF,ON);
```

2.4.3.6 CurLimitEnable（）

函数功能

控制大电流保护功能使能。

使用说明

无。

函数原型

```
int CurLimitEnable(const string& strPinName, emVI_CurrLmt enable);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
enable	开启或关闭大电流保护功能，其类型定义详见 emVI_CurrLmt 类型定义

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().CurLimitEnable(strPinName, CurrLmt_ON);
```

2.4.3.7 FastRead（）

函数功能

快速读取测量数据。

使用说明

该函数支持 Group。

函数原型

```
int FastRead(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式,，其类型定义详见 emVI_MeasMode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().FastRead(strPinName,emVI_MV,emVI_Aver);
```

2.4.3.8 Gang_Init（）

函数功能

调用此函数后退出 Gang 模式，回到初始化状态，并且输出清零，然后在把挡位设置成 10V/10mA 以及断开输出继电器。

使用说明

- 调用 Gang_Init 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，多个 BANK 使用 Gang 功能时，此延时可与其他通道 Gang_Init 函数共用。
- 该函数仅 FVI16Plus 支持。

函数原型

```
int Gang_Init(const string& pinName, emGangType GangType);
```

参数说明

参数	说明
strPinName	设置 Gang 模式的 Pin 名称。
GangType	设置 Gang 模式类型，其类型定义详见 emGangType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().Gang_Init(strPinName,OneGang);
```

2.4.3.9 GetAverResult（）

函数功能

获取测量数据的平均值。

使用说明

该函数支持 Group。

函数原型

```
int GetAverResult(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().GetAverResult(strPinName,emVI_MV);
```

2.4.3.10 GetCHCondition（）

函数功能

获取通道状态。

使用说明

无。

函数原型

```
int GetCHCondition(int nSite, const string& strPinName, int& nPOutMode, double& dVRange, double& dIRange, double& dIVClampP, double& dIVClampN, double& dOutVal, int& nRelayIsOn);
```

参数说明

参数	说明
nSite	测试工位。
strPinName	Pin 的名称。
POutMode	电源工作模式，0（电压源）或者 1（电流源）。
dVRange	电压输出或测量挡位。
dIRange	电流输出或测量挡位。
dIVClampP	电源正钳位设定值。
dIVClampN	电源负钳位设定值。

参数	说明
dOutVal	电源输出值。
nRelayIsOn	输出继电器状态。 ● 0（OFF 关断）。 ● 1（ON 闭合）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
int nPOutMode, nRelayIsOn;
double dVRange, dIRange, dIVClampP, dIVClampN, dOutVal;
TheInst.FVI16().DCVI().GetCHCondition(0,strPinName, nPOutMode, dVRange, dIRange,
dIVClampP, dIVClampN, dOutVal, nRelayIsOn);
```

2.4.3.11 GetMeasResult（）

函数功能

获取测量数据。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, emVI_MeasType emType, emVI_MeasMode
emMode, int nIsShowData);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

参数	说明
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	该参数无效。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().GetMeasResult(strPinName,emVI_MV,emVI_Aver,0);
```

2.4.3.12 GetMeasResult（）

函数功能

用于获取测量电流，或电压的所有点数的测量值。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, int nSite, int nSampNum,
emVI_MeasType emType, double *dDataRead);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
nSite	对应工位号。
nSampNum	采样点数，对应设置的采样点数，确定 dDataRead 的大小。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。

参数	说明
dDataRead	对应通道的电压或电流数据。

返回值

类型	说明
int	● 0：成功。 ● -1：PinName 不存在或异常。

示例

```
const string strPinName = "Pin1";
double dDataRead[25] = {0};
TheInst.FVI16().DCVI().SetMode(strPinName,OUTMODE_FV,FVI16_10V,FVI16_10mA);
TheInst.FVI16().DCVI().SetOutVal(strPinName,5);
TheInst.FVI16().DCVI().MeasureVI(strPinName,25,0.01);
TheInst.FVI16().DCVI().GetMeasResult(strPinName, 0,  25, emVI, dDataRead);
```

2.4.3.13 KelvinMeasure（）

函数功能

测量通道电阻值（Ω）。

使用说明

无。

函数原型

```
int KelvinMeasure(const string& strPinName, double dSampleTms, uint32_t
uSampNum, emVI_KelCalMode emMode = KelCal_ON);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dSampleTms	采样时间间隔，可选范围（0.01ms～30ms）
uSampNum	采样次数，可选范围（1～2048）
emMode	测量模式设置，该参数默认为 0，其类型定义详见 emVI_KelCalMode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().KelvinMeasure(strPinName,0.01,25, KelCal_ON);
```

2.4.3.14 Kelvin Init（）

函数功能

FVI16 板初始化，断开输出继电器及 Kelvin 继电器并清零。

使用说明

无。

函数原型

```
int KelvinInit(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
TheInst.FVI16().DCVI().KelvinInit(strPinName);
```

2.4.3.15 KelvinSetMode（）

函数功能

FVI16 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI16_VRange emVRange, emFVI16_IRange emIRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值，可选挡位 1V，5V，10V，30V，2V，100V，150V。 说明 其中 2V，100V，150V 仅 FVI16Plus 支持。
emVRange	电压输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电压测量挡位设定值。 ♦ FVI16 可选挡位 FVI16_1V，FVI16_5V，FVI16_10V，FVI16_30V。 ♦ FVI16plus 可选挡位 FVI16_1V，FVI16_2V，FVI16_5V，FVI16_10V，FVI16_30V，FVI16_100V，FVI16_150V。

参数	说明
emIRange	电流输出或测量挡位。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值。 ♦ FVI16 可选挡位 FVI16_1uA，FVI16_10uA，FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA。 ♦ 当 FVI16plus 在-30V～+150V 时，可选挡位 FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_50mA，FVI16_100mA（Gang 模式）。 ♦ 在其他情况下，可选挡位 FVI16_1uA（仅测量），FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA（脉冲）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().KelvinSetMode(strPinName,Kelvin_H,OUTMODE_FV,FVI16_1V,
FVI16_100uA);
```

2.4.3.16 KelvinSetMode（）

函数功能

FVI16 电源通道 Kelvin 工作模式设置，闭合通道输出继电器。但不输出电压或电流（输出 0V 或 0mA），dIVClampP 与 dIVClampN 的缺省值为±VRange 或±IRange。

使用说明

无。

函数原型

```
int KelvinSetMode(const string& strPinName, emKelvinMode emKelvinMode, emOutMode emPOutMode, emFVI16_VRange emVRange, emFVI16_IRange emIRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emKelvinMode	Kelvin 工作模式设置。 emKelvinMode 枚举的详细说明请参见 emKelvinMode 类型定义。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位，其类型定义详见 emFVI16_VRange 类型定义。
emIRange	电流输出或测量挡位，其类型定义详见 emFVI16_IRange 类型定义。
dIVClampP	电源正钳位设置（可缺省）。 ● POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0.25%IRang–1000.0）mA。 ● POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0.25%VRang–50.0）V。
dIVClampN	电源负钳位设置（可缺省）。 ● POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（–1000.0–（–0.25%Irang））mA。 ● POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（–50.0–（–0.25%Vrang））V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
```

```
TheInst.FVI16().DCVI().KelvinSetMode(strPinName, Kelvin_H, OUTMODE_FV, FVI16_2V,
FVI16_10mA,10,-10);
```

2.4.3.17 Init（）

函数功能

对板卡进行初始化，默认设置挡位为 10V/10mA，断开输出继电器。

使用说明

- 该函数支持 Group。
- 该函数用于 FVI16、FVI16Plus 启动或测量结束时，对通道进行初始化设置。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 Group 名字。

返回值

类型	说明
int	● 0：正常 ● -1：Pin 不存在

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().init(strPinName);//将 Pin1 对应通道进行初始化,断开对应输出继电器
```

2.4.3.18 MeasureIRS（）

函数功能

测量通道输出电流（mA），测试时可重新选择测量范围，确定新的测量量程，以获得更精确的测量值，测量完成后将自动恢复原来的量程。

使用说明

实测值放在“pSite > dRealData[i]”中，i 与实际工作的 Site（0～63）对应，对于多 Site 并测，并测工作将自动完成，并放到对应的 RealData 中。

函数原型

```
int MeasureIRS(const string& strPinName, emFVI16_IRange emIRange, double dTms)
```

参数说明

参数	说明
strPinName	单 Pin 或 Group 名字。
emIRange	电流输出或测量挡位，其类型定义详见 emFVI16_IRange 类型定义。
dTms	测量等待时间（ms）。

返回值

类型	说明
int	● 0：正常 ● -1：Pin 不存在

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().MeasureIRS((strPinName,FVI16_10mA,2);//测量 Pin1 对应通道的电流，更改测量时的挡位为 10mA 挡，等待时间 2ms
```

2.4.3.19 MeasureI（）

函数功能

测量通道输出电流（mA），默认采样点数 25，采样间隔 0.01ms，快速测量并返回平均值。

使用说明

- 该函数支持 Group。
- 将 Group 测量下的测量结果放入 g_GPResult[channel][site]中。
- 单 Pin 测量时的测量结果放入 pSite > dRealData[i]，Channel 为逻辑通道（0～255），site 当前工

位号（0～63）。

函数原型

```
int MeasureI(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().MeasureI(strPinName);//测量 Pin1 对应通道的电流
```

2.4.3.20 MeasureVI（）

函数功能

VI 源测量。

使用说明

无。

函数原型

```
int MeasureVI(const string& strPinName, uint32_t uSampNum, double
dSampleTms,bool bIsDelay=true)
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称（逻辑通道）。

参数	说明
uSampNum	采样点数。
dSampTms	采样间隔，单位毫秒。
bIsDelay	是否延时。 ● True：延时。 ● False：不延时。 此处需要进行缺省处理，默认为 True，客户可选择使用 False。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().MeasureVI(strPinName,25,0.01,False);
```

2.4.3.21 MeasureV（）

函数功能

测量通道输出电压（V），默认采样点数 25，采样间隔 0.01ms，快速测量并返回平均值。

使用说明

- 该函数支持 Group。
- 将 Group 测量下的测量结果放入 g_GPResult[channel][site]中。
- 单 Pin 测量时的测量结果放入 pSite > dRealData[i]，Channel 为逻辑通道（0～255），Site 当前工位号（0～63）。

函数原型

```
int MeasureV(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().MeasureV(strPinName);//测量 Pin1 对应通道的电压
```

2.4.3.22 MeasureISamp（）

函数功能

测量通道输出电流（mA），可以设置电流采样次数和采样时间间隔。

使用说明

该函数支持 Group。

函数原型

```
int MeasureISamp(const string& strPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样时间间隔。 取值范围：0.01ms～30ms。
nSampleN	采样次数。 取值范围：1～2048。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().MeasureISamp(strPinName,emVI_Aver,0.01,25);
```

2.4.3.23 MeasureVSamp（）

函数功能

测量通道输出电压（V），可以设置电压采样次数和采样时间间隔。

使用说明

该函数支持 Group。

函数原型

```
int MeasureVSamp(const string& strPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样时间间隔。 取值范围：0.01ms～30ms。
nSampleN	采样次数。 取值范围：1～2048。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().MeasureVSamp(strPinName,emVI_Aver,0.01,25,0);
const string strPinName = "Pin1";
```

2.4.3.24 RespondSelect（）

函数功能

选设置响应速度。

使用说明

无。

函数原型

```
int RespondSelect(const string& strPinName, emVI_RespondMode emMode);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emMode	选择积分电容响应速度模式。 ● emVI_Resp_0，最快。 ● emVI_Resp_1。 ● emVI_Resp_2。 ● emVI_Resp_3。 ● emVI_Resp_4。 ● emVI_Resp_5，最慢。 说明 FVI16Plus 不支持 emVI_Resp_5。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().RespondSelect(strPinName, emVI_Resp_0);
```

2.4.3.25 SConnect（）

函数功能

控制电源通道输出继电器开关。

使用说明

调用该函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作，此延时可与其他通道 SConnect 函数共用。

函数原型

```
int SConnect(const string& strPinName, emVI_RealyType emRelay, emVI_RelayState
emRelayState, int site)
```

参数说明

参数	说明
strPinName	单 pin 名称。
emRelay	要操作的继电器类型，共 13 个继电器。 取值如下： Relay_HF、Relay_LF、Relay_HS、Relay_LS、Relay_HFHS_ShortCircuit、 Relay_LFLS_ShortCircuit、Relay_HSLS_ShortCircuit、Relay_HFHS_10KR、 Relay_LFLS_10KR、Relay_H、Relay_L、Relay_F、Relay_S。
emRelayState	继电器状态，其类型定义详见 emVI_RelayState 类型定义。
site	要操作的工位号。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().SConnect(strPinName, Relay_HF, ON,0);//断开 Pin 对应逻辑通道的 0 工位 HF 继电器
```

2.4.3.26 SetClamp（）

函数功能

根据 Setmode 设置的量程，设置 FVI16、FVI16Plus 的钳位值。

使用说明

该函数支持 Group。

函数原型

```
int SetClamp(const string& strPinName, double dIVClampP, double dIVClampN)
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dIVClampP	量程的百分数（0.25～100），正钳位值设置。 ● 当电源设置为 FV 模式时，对应量程为 IRang。 ● 当电源设置为 FI 模式时，对应量程为 VRang。
dIVClampN	量程的百分数（－100～－0.25），负钳位值设置。 ● 当电源设置为 FV 模式时，对应量程为 IRang。 ● 当电源设置为 FI 模式时，对应量程为 VRang。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().SetClamp( strPinName, 100, -100 ) ; //将 Pin1 对应通道的钳位设

置成 100%*Vrang
```

2.4.3.27 SetMode（）

函数功能

设置 FVI16、FVI16plus 电源通道工作模式，闭合通道输出继电器。

使用说明

- 该函数支持 Group。
- 用于 FVI16、FVI16Plus 通道测量时进行测量模式配置。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI16_VRange
emVRange, emFVI16_IRange emIRange, double dIVClampP, double dIVClampN)
```

参数说明

参数	说明
strPinName	单 Pin 或 Group 的名字。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位，其类型定义详见 emFVI16_VRange 类型定义。
emIRange	电流输出或测量挡位，其类型定义详见 emFVI16_IRange 类型定义。
dIVClampP	电源正钳位设置。
dIVClampN	电流负钳位设置。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetMode(strPinName,OUTMODE_FV,FVI16_5V,FVI16_100mA,
FVI16_5V, -1* FVI16_5V);//将 Pin1 对应通道设置为电压模式，5V，10mA 挡位。正负钳位均为 5V
```

2.4.3.28 SetMode（）

设置 FVI16、FVI16plus 电源通道工作模式，闭合通道输出继电器。

使用说明

- 该函数支持 Group。
- 用于 FVI16、FVI16Plus 通道测量时进行测量模式配置。

函数原型

```
int SetMode(const string& strPinName, emOutMode emPOutMode, emFVI16_VRange
emVRange, emFVI16_IRange emIRange)
```

参数说明

参数	说明
strPinName	单 Pin 或 Group 的名字。
emPOutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
emVRange	电压输出或测量挡位，其类型定义详见 emFVI16_VRange 类型定义。
emIRange	电流输出或测量挡位，其类型定义详见 emFVI16_IRange 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().SetMode(strPinName,OUTMODE_FV,FVI16_5V,FVI16_100mA);//将

Pin1 对应通道设置为电压模式,5V,10mA 挡位
```

2.4.3.29 SetOutVal（）

函数功能

根据 Setmode 设置的量程，设置电源通道输出值。

使用说明

该函数支持 Group。

函数原型

```
int SetOutVal(const string& strPinName, double dOutVal)
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	输出值。 ● FV 模式 ♦ FVI16：－30V－＋30V。 ♦ FVI16plus：－30V－＋150V。 ● FI 模式：－200mA～200mA。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().SetOutVal(strPinName, 4);//设置 Pin1 对应通道输出 4V 或 4mA,根

据 setMode 的设定模式而定
```

2.4.3.30 SetOutValLF（）

函数功能

根据 SetMode 设置，设置电源通道的输出值，并且设置波形边沿上升下降的提速时间。

使用说明

该函数支持 Group。

函数原型

```
int SetOutValLF(const string& strPinName, double dOutVal, double dTms)
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dOutVal	源输出起始值。 ● FV 模式 ♦ FVI16：－30V－＋30V。 ♦ FVI16plus：－30V－＋150V。 ● FI 模式：－200mA～200mA。
dTms	从起始值到终点值的建立时间（ms）。 ● 当 POutMode=OUTMODE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● 当 POutMode=OUTMODE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().DCVI().SetOutValLF(strPinName, 4, 3)//设置 Pin1 对应通道,在 3ms 内输出值 4V
```

2.4.3.31 SetOutValSlow（）

函数功能

根据 SetMode 设置，缓慢输出电源通道的输出值，建立时间为 dTms。

使用说明

该函数支持 Group。

函数原型

```
int SetOutValSlow(const string& strPinName, double dStartOutVal, double dEndOutVal, double dTms)
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dStartOutVal	源输出起始值。 ● FV 模式 ♦ FVI16：－30V－＋30V。 ♦ FVI16plus：－30V－＋150V。 ● FI 模式：－200mA～200mA。
dEndOutVal	源输出的终点值。 ● 模式为 FV ♦ FVI16：－30V－＋30V。 ♦ FVI16plus：－30V－＋150V。 ● 模式为 FI：－200mA～200mA。
dTms	从起始值到终点值的建立时间（ms）。 ● 当 POutMode=OUTMODE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● 当 POutMode=OUTMODE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().SetOutValSlow(strPinName, 1,3, 3)//设置 Pin1 对应通道输出在 3ms 内输出从 1~3V
```

2.4.3.32 SetOutValEachSite（）

函数功能

同时对不同 Site 设置不同的电压/电流输出值。

使用说明

- strPinName 代表的电源通道号为逻辑通道号，此函数对各工位同时进行赋值是指对不同工位上的同一个逻辑通道进行迅速赋值，可以设置不同输出值。
- 根据_SetMode 设置。当设置为电压源模式时，*dOutVal 指输出电压。当设置为电流模式时指输出电流，同时设置多 site 的输出值，此函数会调用用户自定的输出赋值数组，要求定义的输出赋值数组的长度必须大于或等于当前调用的最大工作数。（最好输出赋值数组的长度与当前工位数一致）。

函数原型

```
int SetOutValEachSite(const string& strPinName, const double* dOutVal);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dOutVal	存放各 Site 对应的输出设定值的数组首地址，对应数组长度最长为 64 位（最大支持 64 工位设置），入参传入对应数组的首地址。根据源输出值的赋值数组，实现多 Site 输出同时设置。 ● 当 POutMode=OUTMODE_FV 时，FVI16 为电压输出值（–30V－+30V）；FVI16plus 为电压输出值（–30V－+150V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（–200mA～+200.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetMode(strPinName, OUTMODE_FV, FVI2_10V, FVI2_10mA, 10,
-10);
DelaymS(5);
double dOutVal[256] = {2,4,6,8};
TheInst.FVI16().DCVI().SetOutValEachSite(strPinName,dOutVal);
```

2.4.3.33 SetOutValSite（）

函数功能

根据 FVI16/FVI16plus 的 SetMode 设置，设置电源通道指定 Site 的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

函数原型

```
int SetOutValSite(const string& strPinName, int nSite, double dOutVal);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSite	对应 Site 通道号。
dOutVal	存放各 Site 对应的输出设定值的数组首地址，对应数组长度最长为 64 位（最大支持 64 工位设置），入参传入对应数组的首地址。根据源输出值的赋值数组，实现多 Site 输出同时设置。 ● 当 POutMode=OUTMODE_FV 时，输出电压（–30V－+30.0V）。 ● 当 POutMode=OUTMODE_FI 时，输出电流（–200mA～+200.0 mA）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetMode(strPinName, OUTMODE_FV, FVI16_10V,FVI16_200mA);
DelaymS(3);
TheInst.FVI16().DCVI().SetOutValSite(strPinName,1,2);
```

2.4.3.34 SetIRang（）

函数功能

用于电源通道在加压/加流模式下，资源板带电切换电流挡位。

使用说明

无。

函数原型

```
int SetIRang(const string& strPinName, emFVI16_IRange emIRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emIRang	电流挡位。 ● FVI16 可选挡位：FVI16_1uA，FVI16_10uA，FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA。 ● FVI16plus 可选挡位包含如下情况： ♦ 当 FVI16plus 在－30V–＋150V 时，可选挡位：FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_50mA，FVI16_100mA（Gang 模式）。 ♦ 在其他情况下可选挡位：FVI16_1uA（仅测量），FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA（脉冲）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetIRang(strPinName,FVI16_1uA);
```

2.4.3.35 SetGangMode（）

函数功能

设置 Gang 功能工作模式，闭合通道输出继电器。

使用说明

- 调用 SetGangMode 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作。
- 多个 BANK 使用 Gang 功能时，此延时可与其他通道 SetGangMode 函数共用。
- 该函数仅 FVI16Plus 支持。

函数原型

```
int SetGangMode(const string& startPinName,emGangType GangType, emOutMode POutMode, emFVI16_VRange VRange, emFVI16_IRange IRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
strPinName	设置 Gang 模式的 Pin 名称。
GangType	设置 Gang 模式类型，其类型定义详见 emGangType 类型定义。
POutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
VRange	电压输出或测量挡位。可选挡位 1V、2V、5V、10V、30V、100V、150V。 ● 当 POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电压测量挡位设定值。
IRange	电流输出或测量挡位。可选挡位 1μA、10μA、100μA、1mA，10mA、100mA、200mA。 ● 当 POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● 当 POutMode=OUTMODE_FI 时，为电流输出挡位设定值。当作为电流源输出时（即 POutMode=OUTMODE_FI），最小电流挡位 1μA 不能作为输出源使用。
dIVClampP	电源正钳位设置。 ● 当 POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值。 ● 当 POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值。
dIVClampN	输出负钳位设置。 ● 当 POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值。 ● 当 POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值。

返回值

类型	说明
int	● 0：设置成功。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetGangMode(strPinName,OneGang,OUTMODE_FV,FVI16_5V,FVI16_10mA,10,-10);
```

2.4.3.36 SetVRang（）

函数功能

用于电源通加压/加流模式下，资源板带电切换电压挡位。

使用说明

无。

函数原型

```
int SetVRang(const string& strPinName, emFVI16_VRange emVRange);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emVRang	电压挡位。 ● FVI16 可选量程：FVI16_1uA，FVI16_10uA，FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA。 ● FVI16plus 可选挡位存在如下情况： ♦ FVI16plus 在-30～+150V 时，可选量程：FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_50mA，FVI16_100mA（Gang 模式）。 ♦ 在其他情况下，可选挡位 FVI16_1uA（仅测量），FVI16_10uA（仅测量），FVI16_100uA，FVI16_1mA，FVI16_10mA，FVI16_100mA，FVI16_200mA（脉冲）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().DCVI().SetVRang(strPinName, FVI16_1V);
```

2.4.4 FVI16-VIKR

2.4.4.1 RelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int RelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().RelayOn(KName1,KName2,KName3);
```

2.4.4.2 RelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().RelaySetOn("KR_LSS_FVI16, KR_LG1_FVI16, KR_VCC_FVI16,
KR_VIN_FVI16, KR_BS2_FVI2_H1, KR_HG2_FVI16, KR_LX2_FVI16");
```

2.4.4.3 RelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（逻辑通道）。

使用说明

无。

函数原型

```
int RelaySetOff(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().RelaySetOff("KR_LSS_FVI16, KR_LG1_FVI16, KR_VCC_FVI16,
KR_VIN_FVI16, KR_BS2_FVI2_H1, KR_HG2_FVI16, KR_LX2_FVI16");
```

2.4.4.4 SiteRelayOn（）

函数功能

设置所有 Site 需要闭合的用户继电器，对未列出的用户继电器设置为断开（逻辑通道）。

使用说明

无。

函数原型

```
int SiteRelayOn(const string &strPinList);
```

参数说明

参数	说明
strPinList	需要操作的继电器名称组，名称之间用英文逗号隔开。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().SiteRelayOn("KR_LSS_FVI16, KR_LG1_FVI16, KR_VCC_FVI16,
KR_VIN_FVI16, KR_BS2_FVI2_H1, KR_HG2_FVI16, KR_LX2_FVI16");
```

2.4.4.5 SRelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开（物理通道）。

使用说明

无。

函数原型

```
int SRelayOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().SRelayOn(std::vector<int>{0, 1, 2, 3, -1});
```

2.4.4.6 SRelaySetOn（）

函数功能

设置需要闭合的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOn(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().SRelaySetOn(std::vector<int>{0, 1, 2, 3, -1});
```

2.4.4.7 SRelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态（物理通道）。

使用说明

无。

函数原型

```
int SRelaySetOff(const std::vector<int> &vec);
```

参数说明

参数	说明
vec	需要操作的物理通道。

返回值

类型	说明
int	始终为 0，无意义。

示例

```
TheInst.FVI16().VIKR().SRelaySetOff(std::vector<int>{0, 1, 2, 3, -1});
```

2.4.5 FVI16- VOSC

2.4.5.1 GetThd（）

函数功能

FVI16 谐波测量模式选择，可选失真度测量或有效值测量测量。

使用说明

测量模式可选电压测量或电流测量。

函数原型

```
int GetThd(const string& strPinName, emVI_MeasType emType, int nData);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
nData	● 0：有效值。 ● 1：失真度（%）。 ● 2：失真度（dB）。 ● 3：失真度（%）。 ● 4：失真度（dB）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().VOSC().GetThd(strPinName, emVI_MV, 0);
```

2.4.5.2 GetThd（）

函数功能

获取前一次测量得到的采样数据。

函数原型

```
int GetThd(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().VOSC().GetThd(strPinName);
```

2.4.5.3 SetSampNum（）

函数功能

设置当前通道的采样点数与测量类型。

使用说明

无。

函数原型

```
int SetSampNum(const string& strPinName, int nSampNum, int nType = 2);
```

参数说明

参数	说明
strPinName	Pin 的名字。
nSampNum	采样点数。
nType	测量数据类型。 ● 0：电压。 ● 1：电流。 ● 2：预留参数。

返回值

类型	说明
int	设置的结果。 ● 0：设置成功。 ● -1：设置失败。

示例

```
TheInst.FVI16().VOSC().SetSampNum("Pin1", 100, 0);
```

2.4.6 FVI16-Utility

2.4.6.1 Get_Master_FPGAVer（）

函数功能

获取 FVI16 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 类型和 CString 类型。

函数原型

```
int Get_Master_FPGAVer(int nFVI16n, int& nFpgaVer, char* strFpgaVer);
```

参数说明

参数	说明
nFVI16n	板卡号（0、1、2、3…15）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

返回值为 0。

示例

```
int nFpgaVer;
char strFpgaVer[]={0};
int nMasterVer =
TheInst.FVI16().Utility().Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.4.6.2 GetOSDAndGNDAlarmData（）

函数功能

用于查询 SENCE_GND 与 OSD 信号是否报警状态。

使用说明

无。

函数原型

```
int GetOSDAndGNDAlarmData(const string& strPinName)
```

参数说明

参数	说明
strPinName	单 Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().Utility().GetOSDAndGNDAlarmData(strPinName);
```

2.4.6.3 Get_Slave_FPGAVer（）

函数功能

获取 FVI16 板卡从机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 类型和 CString 类型。

函数原型

```
int Get_Slave_FPGAVer(int nFVI16n, int& nFpgaVer1, char* strFpgaVer1, int&
nFpgaVer2, char* strFpgaVer2, int& nFpgaVer3, char* strFpgaVer3, int& nFpgaVer4,
char* strFpgaVer4);
```

参数说明

参数	说明
nFVI16n	板卡号（0、1、2、3…15）。
nFpgaVer1	返回从机 1FPGA 版本号的变量。
strFpgaVer1	返回从机 1FPGA 版本号的变量指针。
nFpgaVer2	返回从机 2FPGA 版本号的变量，int 型。
strFpgaVer2	返回从机 2FPGA 版本号的变量指针。
nFpgaVer3	返回从机 3FPGA 版本号的变量，int 型。
strFpgaVer3	返回从机 3FPGA 版本号的变量指针。
nFpgaVer4	返回从机 4FPGA 版本号的变量，int 型。
strFpgaVer4	返回从机 4FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
int nFpgaVer1,nFpgaVer2,nFpgaVer3,nFpgaVer4;
char strFpgaVer1[256]={'\0'};       //注：字符串长度需≥7
char strFpgaVer2[256]={'\0'};       //注：字符串长度需≥7
char strFpgaVer3[256]={'\0'};       //注：字符串长度需≥7
char strFpgaVer4[256]={'\0'};       //注：字符串长度需≥7
int SlaveVer =
TheInst.FVI16().Utility().Get_Slave_FPGAVer(0,nFpgaVer1,strFpgaVer1,
nFpgaVer2,strFpgaVer2,nFpgaVer3,strFpgaVer3,nFpgaVer4,strFpgaVer4);
```

2.4.6.4 GetTemperature（）

函数功能

获取指定通道温度。

使用说明

无。

函数原型

```
double GetTemperature(const string& strPinName, int site)
```

参数说明

参数	说明
strPinNam	单 Pin 的名称。
site	要操作的工位号。

返回值

类型	说明
double	● 获取成功，返回温度值。 ● 获取失败，返回-1。

示例

```
const string strPinName = "Pin1";

TheInst.FVI16().Utility().GetTemperature(strPinName,0);//获取 pin1 对应逻辑通道的 0

工位的温度
```

2.4.6.5 GetVer（）

函数功能

获取 FVI16 板卡的模块版本信息。

使用说明

返回值类型为以 M 开头的字符串。

函数原型

```
int GetVer(char* chVer, int nFVI16n);
```

参数说明

参数	说明
chVer	返回版本信息的变量指针。
nFVI16n	板卡号（0、1、2、3…15）。

返回值

类型	说明
int	返回值为 0。

示例

```
char chVer [256]={'\0'};
TheInst.FVI16().Utility().GetVer(chVer,0);
```

2.4.6.6 ReadMultimeterModel（）

函数功能

获取 FVI16 板卡的上一次校准校验所使用的万用表型号信息。

使用说明

返回值类型为字符串。

函数原型

```
int ReadMultimeterModel(char* MultimeterModel, int nCard);
```

参数说明

参数	说明
MultimeterModel	返回万用表信息的变量指针。
nCard	板卡号（0、1、2、3、…、15）。

返回值

类型	说明
int	● 3458：3458A 万用表读取到的值。 ● 2000：2000 万用表读取到的值。 ● 34401：34401 万用表读取到的值。 ● 34410：34410 万用表读取到的值。 ● 34461：34461 万用表读取到的值。 ● 万用表读取到的值 32–bit_ADC：内部 32–bit_ADC。 ● 0：其他情况读取到的值。

示例

```
char chMultimeterModel[11]={"\0"};
TheInst.FVI16().Utility().ReadMultimeterModel(chMultimeterModel,0);
```

2.4.7 FVI16-TMU

2.4.7.1 类型定义

FVI16_TMU_Mode 类型定义

类型	Enumeration	
描述	测量模式。	
元素	FVI16_TMU_Frequency=0	测量频率，单位为 kHz。
	FVI16_TMU_Cycle=1	测量周期，单位为 ms。
	FVI16_TMU_HighLevel=2	高电平宽度，单位为 ms。
	FVI16_TMU_LowLevel=3	低电平宽度，单位为 ms。
	FVI16_TMU_Rise=4	上升沿时间，单位为 ms。
	FVI16_TMU_Drop=5	下降沿时间，单位为 ms。
	FVI16_TMU_DutyCycle=11	测量占空比。
	FVI16_TMU_Frequency_Neg=20	测量频率，单位为 kHz，用下降沿测量。
	FVI16_TMU_Cycle_Neg=21	测量周期，单位为 ms，用下降沿测量。
头文件	UserDef.h	

2.4.7.2 Init（）

函数功能

TMU 初始化。

使用说明

无。

函数原型

```
int Init(const string& PinName);
```

参数说明

参数	说明
PinName	单 Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().Init(strPinName) ;
```

2.4.7.3 Measure（）

函数功能

根据设置模式，获取测量结果。

使用说明

无。

函数原型

```
int Measure(const string& PinName);
```

参数说明

参数	说明
PinName	单 Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().Measure(strPinName) ;
```

2.4.7.4 MeasureAver（）

函数功能

设置计算 N 个波的平均值（仅支持周期）。

使用说明

无。

函数原型

```
int MeasureAver(const string& PinName,  int nNumber);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
nNumber	波的个数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nNumber =10;
TheInst.FVI16().TMU().MeasureAver(strPinName,nNumber);
```

2.4.7.5 MeasureWithTms（）

函数功能

在规定时间内获取测量结果。

使用说明

无。

函数原型

```
int MeasureWithTms(const string& PinName,double dSampleTms);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
dSampleTms	制度规定时间，单位（ms）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dSampleTms = 10.0;
TheInst.FVI16().TMU().MeasureWithTms(strPinName,dSampleTms);
```

2.4.7.6 SetMode（）

函数功能

设置 TMU 测量相关参数模式。

使用说明

无。

函数原型

```
int SetMode(const string& PinName, FVI16_TMU_Mode nMode, FVI16_TMU_Filter
nFilter, double dTGV1, double dTGV2);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
nMode	测量模式，其类型定义详见 FVI16_TMU_Mode 类型定义。
nFilter	滤波系数。单位为 ns，范围：0ns～10s，20ns 为间隔，0 为不设置滤波。 ● FVI16_TMU_Filter_0ns=0。 ● FVI16_TMU_Filter_20ns=2。 ● FVI16_TMU_Filter_100ns=10。 ● FVI16_TMU_Filter_200ns=20。 ● FVI16_TMU_Filter_500ns=50。 ● FVI16_TMU_Filter_1μs=100。 ● FVI16_TMU_Filter_2μs=200。 ● FVI16_TMU_Filter_5μs=500。 ● FVI16_TMU_Filter_10μs=1000。 ● FVI16_TMU_Filter_20μs=2000。 ● FVI16_TMU_Filter_50μs=5000。 ● FVI16_TMU_Filter_100μs=10000。 ● FVI16_TMU_Filter_200μs=20000。 ● FVI16_TMU_Filter_500μs=50000。 ● FVI16_TMU_Filter_1ms=100000。 ● FVI16_TMU_Filter_2ms=200000。 ● FVI16_TMU_Filter_5ms=500000。 ● FVI16_TMU_Filter_10ms=1000000。 ● FVI16_TMU_Filter_20ms=2000000。 ● FVI16_TMU_Filter_50ms=5000000。 ● FVI16_TMU_Filter_100ms=10000000。 ● FVI16_TMU_Filter_200ms=20000000。 ● FVI16_TMU_Filter_500ms=50000000。 ● FVI16_TMU_Filter_1s=100000000。 ● FVI16_TMU_Filter_2s=200000000。 ● FVI16_TMU_Filter_5s=500000000。 ● FVI16_TMU8_Filter_10s=1000000000。

参数	说明
dTGV1	触发电压 1。
dTGV2	触发电压 2。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().SetMode(strPinName,FVI16_TMU_Cycle,FVI16_TMU_Filter_100ns,
2,5);
```

2.4.7.7 StartNumber（）

函数功能

设置从第 N 个波开始计算（仅支持周期）。

使用说明

无。

函数原型

```
int StartNumber(const string& PinName, int nNumber);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
nNumber	第 N 个波开始。 取值范围：0～255。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nNumber =10;

TheInst.FVI16().TMU().StartNumber(strPinName , nNumber);
```

2.4.7.8 RamAddClr（）

函数功能

清除当前测量结果。

使用说明

无。

函数原型

```
int RamAddClr(const string& PinName);
```

参数说明

参数	说明
PinName	单 Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().RamAddClr(strPinName);
```

2.4.7.9 RamRead（）

函数功能

抖频获取指定波形测量结果，根据 SetMode 设置的模式获取不同结果。

使用说明

无。

函数原型

```
int RamRead(const string& PinName, int nAdd);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
nAdd	波的序号。 取值范围：0～255。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().RamRead(strPinName,5);
```

2.4.7.10 WaveCountMeasure（）

函数功能

获取波形计数。

使用说明

无。

函数原型

```
int WaveCountMeasure(const string& PinName);
```

参数说明

参数	说明
PinName	单 Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().WaveCountMeasure(strPinName);
```

2.4.7.11 WaveCountStart（）

函数功能

设置波形计数时间。

使用说明

无。

函数原型

```
int WaveCountStart(const string& PinName, double dTimes);
```

参数说明

参数	说明
PinName	单 Pin 的名称。
dTimes	计数时间。 单位：ms。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI16().TMU().WaveCountStart(strPinName, 10);
```

2.5 FHVI1

2.5.1 GetAllSinglePinName（）

函数功能

获取 FHVI1 板卡所有的 Pin 的名字。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

FHVI1 板卡所有的 Pin 的名字。

示例

```
vector<string> strPinName;
strPinName = TheInst.FHVI1().GetAllSinglePinName(); //获取 FHVI1 板卡所有的 Pin 的名字
```

2.5.2 FHVI1-DCVI

2.5.2.1 CapLoad（）

函数功能

根据 FHVI1_SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTms。

使用说明

无。

函数原型

```
int CapLoad(const string& pPinName, double dStartOutVal, double dEndOutVal,
double dTms);
```

参数说明

参数	说明
pPinName	Pin 的名字。
dStartOutVal	源输出的起始值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V－+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20 mA～+20mA）。
dEndOutVal	源输出的终点值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V－+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20mA～+20mA）。
dTms	从起始值到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName;

TheInst.FHVI1().DCVI().CapLoad(strPinName,0,800,10); setmode 模式为 FV 模式时,该函数

实现功能为：FVI2 的 strPinName 通道电压或电流实现从 0V 到 800V,时间跨度为 10ms。
```

2.5.2.2 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTMs。

使用说明

- 当设置为电压源模式时，dStartOutVal 指输出的起始电压，dEndOutVal 指输出的终点电压。
- 当设置为电流源模式时，dStartOutVal 指输出的起始电流，dEndOutVal 指输出的终点电流。

函数原型

```
int CapLoad(const string& strPinName, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dEndOutVal	源输出的终点值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V–+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20mA～+20mA）。
dTms	从起始值到终点值的建立时间（ms）。 ● POutMode=OUTMODE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode=OUTMODE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName;
TheInst.FHVI1().DCVI().CapLoad(strPinName,800,10);
```

2.5.2.3 Connect（）

函数功能

电源通道输出开关，内部输出值不变。

使用说明

- 该函数使用时应避免带负载切换，以免影响电源输出继电器的使用寿命。
- 调用 FHVI1_Connect 函数后，为了使源的内部达到稳定状态，需要至少延时 2ms 再执行其他操作。

函数原型

```
int Connect(const string& pPinName, emVI_RelayState isOn);
```

参数说明

参数	说明
pPinName	Pin 的名字。
isOn	继电器状态，其类型定义详见 emVI_RelayState 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FHVI1().DCVI().Connect(strPinName,  ON);
```

2.5.2.4 GetCHCondition

函数功能

获取通道状态。

使用说明

无。

函数原型

```
int GetCHCondition(int nSite, const string& pPinName, int &POutMode, double &dVRange, double &dIRange, double &dIVClampP, double &dIVClampN, double &dOutVal, int &nRelayIsOn);
```

参数说明

参数	说明
nSite	测试工位。
pPinName	Pin 的名字。
POutMode	电源工作模式。 ● 0：电压源。 ● 1：电流源。
dVRange	电压输出或测量挡位。
dIRange	电流输出或测量挡位。
dIVClampP	电源正钳位设定值。
dIVClampN	电源负钳位设定值。
dOutVal	电源输出值。
nRelayIsOn	输出继电器状态。 ● 0：OFF 关断。 ● 1：ON 闭合。

返回值

类型	说明
int	

示例

```
const string strPinName;
int OutMode, nRelayIsOn;
double dVRange , dIRange, dIVClampP , dIVClampN, dOutVal ;
TheInst.FHVI1().DCVI().GetCHCondition(0,strPinName,OutMode,dVRange,dIRange,
dIVClampP, dIVClampN, dOutVal,nRelayIsOn);
```

2.5.2.5 GetMeasResult（）

函数功能

电压或电流测量。

使用说明

测量模式可选平均值、最大值或最小值测量。

函数原型

```
int GetMeasResult(const string& pPinName, emVI_MeasType emType, emVI_MeasMode
emMode, int nIsShowData);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	选择测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	参数基本无效，虚拟示波器直接调用。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
int nIsShowData;
```

```
const string strPinName = "Pin1";
TheInst.FHVI1().DCVI().GetMeasResult(strPinName, emVI_MV, emVI_Aver,0);
```

2.5.2.6 GetMeasResult（）

函数功能

用于获取测量电流，或电压的所有点数的测量值。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, int nSite, int nSampNum,
emVI_MeasType emType, double *dDataRead) ;
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
nSite	对应工位号。
nSampNum	采样点数，对应设置的采样点数，确定 dDataRead 的大小。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dDataRead	对应通道的电压或电流数据。

返回值

类型	说明
int	● 0：成功。 ● - 1：PinName 不存在或异常。

示例

```
const string strPinName = "Pin1";
double dDataRead[25] = {0};
TheInst.FHVI1().DCVI().SetMode(strPinName,OUTMODE_FV,FHVI1_100V,FHVI1_10mA);
TheInst.FHVI1().DCVI().SetOutVal(strPinName,5);
TheInst.FHVI1().DCVI().MeasureVI(strPinName,25,0.01);
TheInst.FHVI1().DCVI().GetMeasResult(strPinName, 0,  25, emVI, dDataRead);
```

2.5.2.7 Init（）

函数功能

FHVI1 板初始化，先清零，再把挡位设置成 10V/10mA 以及断开输出继电器。

如果原先是电流源就是电流给到 0，如果是电压源就是电压给到 0。

使用说明

无。

函数原型

```
int Init(const string& pPinName);
```

参数说明

参数	说明
pPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常 ● －1：Pin 不存在

示例

```
const string strPinName;
TheInst.FHVI1().DCVI().Init(strPinName);
```

2.5.2.8 MeasureI（）

函数功能

测量通道输出电流（mA），采样次数（25），采样时间间隔（0.01ms）固定，快速测量且返回平均值。

使用说明

实测值放在“pSite > dRealData[i]”中，i 与实际工作的 Site（0～15）相对应，对于多 Site 并测，并测工作将自动完成，并放到对应的 RealData 中。

函数原型

```
int MeasureI(const string& pPinName);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字

返回值

类型	说明
int	● 0：正常 ● -1：Pin 不存在

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().MeasureI(pPinName);
```

2.5.2.9 MeasureISamp（）

函数功能

电流值测量。

使用说明

- 测量模式可选平均值、最大值或最小值测量。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureISamp(const string& pPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，可选范围（0.01ms～30ms）
nSampleN	采样点数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().MeasureISamp(strPinName, emVI_Aver, 0.01, 100);
```

2.5.2.10 MeasureV（）

函数功能

测量通道输出电压（V），采样次数（25），采样时间间隔（0.01ms）固定，且返回平均值，实测值放在“pSite > dRealData[i]”中，i 与实际工作的 Site（0～15）相对应。

多 Site 并测，并测工作将自动完成，并放到对应的 RealData 中。

使用说明

- 此函数不同于 FHVI1_MeasureVSamp（）。FHVI1_MeasureV（）采样次数（25），采样时间间隔（0.01ms）固定，且返回平均值。
- FHVI1_MeasureVSamp（）采样次数，采样时间间隔可设置，且可返回最大值，最小值或平均值。

函数原型

```
int MeasureV(const string& pPinName);
```

参数说明

参数	说明
pPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().MeasureV(pPinName);
```

2.5.2.11 MeasureVI（）

函数功能

VI 源测量。

函数原型

```
int MeasureVI(const string& pPinName, uint32_t uSampNum, double dSampTms, bool bIsDelay=true);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字（逻辑通道）。
uSampNum	采样点数。
dSampTms	采样间隔，单位毫秒。
bIsDelay	是否延时，选择 True 或者 False，此处需要进行缺省处理，默认为 True，客户可选择使用 False。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FVI2().DCVI().MeasureVI(strPinName,25,0.01,False);
```

2.5.2.12 MeasureVSamp（）

函数功能

电压值测量。

使用说明

- 测量类型选择，可选平均值、最大值或最小值测量。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureVSamp(const string& pPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，可选范围（0.01ms～30ms）
nSampleN	采样点数，可选范围（1～2048）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
 TheInst.FHVI1().DCVI().MeasureVSamp(strPinName,emVI_Aver,0.01,100);
```

2.5.2.13 SetIRang（）

函数功能

用于电源通道在加压/加流模式下，资源板带点切换电流挡位。

使用说明

无。

函数原型

```
int SetIRang(const string& pPinName, emFHVI1_IRange IRange);
```

参数说明

参数	说明
pPinName	Pin 的名字。
IRange	电流挡位. 可选挡位 FHVI1_1uA（仅测量），FHVI1_10uA（仅测量），FHVI1_100uA，FHVI1_1mA，FHVI1_10mA，FHVI1_20mA。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
```

```
TheInst.FHVI1().DCVI().SetIRang(pPinName,FHVI1_1uA);
```

2.5.2.14 SetMode（）

函数功能

设置 FHVI1 电源通道工作模式，闭合通道输出继电器。

使用说明

无。

函数原型

```
int SetMode(const string& pPinName, emOutMode POutMode, emFHVI1_VRange VRange,
emFHVI1_IRange IRange, double dIVClampP, double dIVClampN);
```

参数说明

参数	说明
pPinName	Pin 的名字。
POutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。
VRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值。可选挡位 FHVI1_100V，FHVI1_200V，FHVI1_500V，FHVI1_1000V。
IRange	电流输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FHVI1_1uA，FHVI1_10uA，FHVI1_100uA，FHVI1_1mA，FHVI1_10mA，FHVI1_20mA。
dIVClampP	电源正钳位设置。 ● POutMode=OUTMODE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（0～20.0）mA。 ● POutMode=OUTMODE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（0 - 1000.0）V。

参数	说明
dIVClampN	电源负钳位设置。 ● POutMode=OUTMODE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值，可选范围（－20.0～0）mA。 ● POutMode=OUTMODE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值，可选范围（－1000.0－0）V。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().SetMode(pPinName,OUTMODE_FV, FHVI1_100V,FHVI1_10uA,5,-5);
TheSoft.TestProc().DelaymS(2);
```

2.5.2.15 SetMode（）

函数功能

设置 FHVI1 电源通道工作模式，闭合通道输出继电器。

使用说明

无。

函数原型

```
int SetMode(const string& pPinName, emOutMode POutMode, emFHVI1_VRange VRange,
emFHVI1_IRange IRange);
```

参数说明

参数	说明
pPinName	Pin 的名字。
POutMode	工作模式选择。emOutMode 枚举的详细说明请参见 emOutMode 类型定义。

参数	说明
VRange	电压输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电压输出挡位设定值。 ● POutMode=OUTMODE_FI 时，为电压测量挡位设定值。可选挡位 FHVI1_100V，FHVI1_200V，FHVI1_500V，FHVI1_1000V。
IRange	电流输出或测量挡位。 ● POutMode=OUTMODE_FV 时，为电流测量挡位设定值。 ● POutMode=OUTMODE_FI 时，为电流输出挡位设定值。可选挡位 FHVI1_1uA，FHVI1_10uA，FHVI1_100uA，FHVI1_1mA，FHVI1_10mA，FHVI1_20mA。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
TheInst.FHVI1().DCVI().SetMode(pPinName, OUTMODE_FV, FHVI1_100V, FHVI1_10mA);
TheSoft.TestProc().DelaymS(3);
```

2.5.2.16 SetOutVal（）

函数功能

根据 FHVI1_SetMode 设置，设置电源通道的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时指输出电流。

函数原型

```
int SetOutVal(const string& pPinName, double dOutVal);
```

参数说明

参数	说明
pPinName	Pin 的名字。
dOutVal	源输出值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V－+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20mA～+20.0mA）。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().SetOutVal(pPinName,1.0);
```

2.5.2.17 SetOutValSlow（）

函数功能

根据 FHVI1 的 SetMode 设置,缓慢输出电源通道的输出值，建立时间为 dTMs。

使用说明

- 当设置为电压源模式时，dStartOutVal 指输出的起始电压，dEndOutVal 指输出的终点电压。
- 当设置为电流源模式时，dStartOutVal 指输出的起始电流，dEndOutVal 指输出的终点电流。

函数原型

```
int SetOutValSlow(const string& pPinName, double dStartOutVal, double
dEndOutVal, double dTms);
```

参数说明

参数	说明
pPinName	单 Pin 或 PinGroup 的名字
dStartOutVal	源输出的起始值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V－+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20mA～+20.0mA）。
dEndOutVal	源输出的终点值。 ● POutMode=OUTMODE_FV 时，为电压输出值（–1000V－+1000.0V）。 ● POutMode=OUTMODE_FI 时，为电流输出值（–20mA～+20.0mA）。
dTms	从起始值到终点值的建立时间（ms）。 ● 当 POutMode=OUTMODE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● 当 POutMode=OUTMODE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().SetMode(pPinName, OUTMODE_FV, FHVI1_10V, FHVI1_200mA);
DelaymS(3);
TheInst.FHVI1().DCVI(). SetOutVal(pPinName,5);
TheInst.FHVI1().DCVI().SetOutValSlow(0, 5, 3, 2);
```

2.5.2.18 SetVRang（）

函数功能

用于电源通加压/加流模式下，资源板带电切换电压挡位。

使用说明

无。

函数原型

```
int SetVRang(const string& pPinName, emFHVI1_VRange VRange);
```

参数说明

参数	说明
pPinName	Pin 的名字。
VRange	电流挡位，可选挡位 FHVI1_100V，FHVI1_200V，FHVI1_500V，FHVI1_1000V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string pPinName = "Pin1";
TheInst.FHVI1().DCVI().SetVRang(pPinName,FHVI1_1V);
```

2.5.3 FHVI1-VOSC

2.5.3.1 GetThd（）

函数功能

谐波测量类型选择，可选失真度测量或有效值测量。

使用说明

测量类型可选电压测量或电流测量。

函数原型

```
int GetThd(const string& strPinName, emVI_MeasType emType, int nData);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。
nData	数据类型。 ● 0：有效值。 ● 3：失真度（%） ● 4：失真度（dB）

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FHVI1().VOSC().GetThd(strPinName , emVI_MV, 0);
```

2.5.3.2 GetThd（）

函数功能

获取前一次测量得到的采样数据。

使用说明

无。

函数原型

```
int GetThd(const string& strPinName)
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string strPinName = "Pin1";
TheInst.FHVI1().VOSC().GetThd(strPinName);
```

2.5.3.3 GetRms（）

函数功能

谐波有效值测量。

使用说明

测量类型可选电压测量或电流测量。

函数原型

```
int GetRms(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.FHVI1().VOSC().GetRms(strPinName, emVI_MV);
```

2.5.3.4 SetSampNum（）

函数功能

设置当前通道的采样点数与测量类型。

使用说明

无。

函数原型

```
int SetSampNum(const string& strPinName, int nSampNum, int nType = 2);
```

参数说明

参数	说明
strPinName	Pin 的名字。
nSampNum	采样点数。
nType	测量数据类型。 ● 0：电压。 ● 1：电流。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string strPinName = "Pin1";
TheInst.FHVI1().VOSC().SetSampNum("Pin1", 100, 0);
```

2.5.4 FHVI1-Utility

2.5.4.1 Get_Master_FPGAVer（）

函数功能

获取 FHVI1 板卡主机 FPGA 版本号。

使用说明

该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

函数原型

```
int Get_Master_FPGAVer(int nFHVI1n, int &nFpgaVer, char *strFpgaVer);
```

参数说明

参数	说明
nFHVI1n	板卡号（0、1、2、3）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
int nFpgaVer;
char strFpgaVer[256]={'\0'};
TheInst.FHVI1().Utility().Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.5.4.2 GetVer（）

函数功能

获取 FHVI1 板卡的硬件版本信息。

使用说明

返回值类型为以 V 开头的字符串。

函数原型

```
int GetVer(char* chVer, int nFHVI1n);
```

参数说明

参数	说明
chVer	返回版本信息的变量指针。
nFHVI1n	板卡号（0、1、2、3）。

返回值

类型	说明
int	返回值为 0。

示例

```
char chVer[256]={'\0'};
TheInst.FHVI1().Utility().GetVer(chVer,0);
```

2.5.4.3 GetTemperature（）

函数功能

获取指定通道温度。

使用说明

无。

函数原型

```
double GetTemperature(const string& strPinName, int site)
```

参数说明

参数	说明
strPinNam	单 Pin 的名称。
site	要操作的工位号。

返回值

类型	说明
double	相应物理通道的温度测量值。

示例

```
const string strPinName = "Pin1";
double dtemp = TheInst.FHVI1().Utility().GetTemperature(strPinName, 0);
```

2.5.4.4 ReadMultimeterModel（）

函数功能

获取第 nCard 序号的 FHVI1 板卡上一次校准所使用的仪器型号。

使用说明

无。

函数原型

```
int ReadMultimeterModel(char* MultimeterModel, int nCard);
```

参数说明

参数	说明
MultimeterModel	返回万用表信息的变量指针。
nCard	板卡号（0、1、2、3）。

返回值

类型	说明
int	● 3458：3458A 万用表读取到的值。 ● 2000：2000 万用表读取到的值。 ● 34401：34401 万用表读取到的值。 ● 34410：34410 万用表读取到的值。 ● 34461：34461 万用表读取到的值。 ● 万用表读取到的值 32－bit_ADC：内部 32－bit_ADC。 ● 0：其他情况读取到的值。

示例

```
char MultimeterModel[256]={'\0'};
TheInst.FHVI1().Utility().ReadMultimeterModel(MultimeterModel,0);
```

2.6 HAD8

2.6.1 Init（）

函数功能

用于 HAD 通道的初始化，将断开 HAD 板上的接口继电器，AWG 的 DAC 置零，关闭 DIG 和 AWG 的外部触发模式，关闭 DIG 和 AWG 的同步模式。

使用说明

无。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().Init(strPinName);
```

2.6.2 GetAllSinglePinName（）

函数功能

获取 HAD8 板卡所有的 Pin 的名字。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector<string>	strPinName 的字符串。

示例

```
 vector<string> strPinName;
strPinName = TheInst.HAD8().GetAllSinglePinName();
```

2.6.3 GetPinName（）

函数功能

对齐 HAD8 指定物理通道对应的 Pin。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh);
```

参数说明

参数	说明
nPhyCh	物理通道号（0、1、2、3）。

返回值

类型	说明
string	strPinName 的字符串。

示例

```
const std::string pName = TheInst.HAD8().GetPinName(AWGpin);
TheInst.HAD8().GetScanTrigAddr(pName);//获取 AWG 的触发值
TheSoft.TestProc().DelaymS(5);
double dres = g_pSiteData->dRealVal[0];
TheSoft.TestParam().LogResult(0, 0, 0, dres);
```

2.6.4 GetScanTrigAddr（）

函数功能

用于获取电压扫描时，触发点的 AWG 输出电压值。

使用说明

无。

函数原型

```
int GetScanTrigAddr(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().GetScanTrigAddr(strPinName);
```

2.6.5 GetScanTrigVal（）

函数功能

用于获取电压扫描时，触发点的 AWG 输出电压值。

使用说明

无。

函数原型

```
int GetScanTrigVal(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().GetScanTrigVal(strPinName);
```

2.6.6 SetScanParams（）

2.6.6.1 类型定义

Edge 类型定义

类型	Enumeration	
描述	沿触发模式。	
元素	Neg_Edge = 0	上升沿触发。
	Pos_Edge = 1	下降沿触发。
头文件	UserDef.h	

Scan_Type 类型定义

类型	Enumeration	
描述	扫描模式。	
元素	Full_Scan=0	全扫模式。
	Half_Scan=1	半扫模式。 半扫模式下，到达触发点后，保持当前输出电压值。
头文件	UserDef.h	

2.6.6.2 函数说明

函数功能

用于设置电压扫描相关设置。

使用说明

电压扫描功能需要在同一块板卡内占用一个 Digitizer 通道和一个 AWG 通道，通过调用 HAD8 的 SyncRun（）来启动。

函数原型

```
int SetScanParams(const string& strTrigPinName, const string& strOutPinName,
double dTrigVal, Edge emSyncPosOrNeg, Scan_Type emScanType);
```

参数说明

参数	说明
strTrigPinName	指定用于测量的 Digitizer 通道。
strOutPinName	指定用于输出的 AWG 通道。
dTrigVal	测量通道的电压触发值，单位：V。
emSyncPosOrNeg	设置测量通道为上升沿或下降沿触发，其类型定义详见 Edge 类型定义。
emScanType	扫描模式，其类型定义详见 Scan_Type 类型定义。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().SetScanParams(strPinName,0,1,Pos_Edge,Full_Scan);
```

2.6.7 SetRefV（）

函数功能

设置基准电压。

使用说明

无。

函数原型

```
int SetRefV(const string& strPinName, double dValue);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dValue	设置的基准电压范围：±10.0。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().SetRefV(strPinName,2.0);
```

2.6.8 SyncRun（）

函数功能

用于同步模式的启动，包括同步测量，同步输出或者电压扫描功能。

使用说明

在调用本函数前需要通过 HAD8 的 DigSetSync（）、AwgSetSync（）或 SetScanParams（）来指定通道。

函数原型

```
int SyncRun();
```

参数说明

无。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName;
TheInst.HAD8().SyncRun();
```

2.6.9 AWG

2.6.9.1 类型定义

Sync_SW 类型定义

类型	Enumeration	
描述	打开或者关闭同步模式。	
元素	Sync_ON	打开同步模式。
	Sync_OFF	关闭同步模式。
头文件	UserDef.h	

ExtTrig_Model 类型定义

类型	Enumeration	
描述	触发模式。	
元素	Normal_Trig	正常触发模式。 在触发信号到来后，按照 HAD8_AwgSelect（）的设置进行波形输出。
	Multiple_Trig	多次触发模式。 每来一个触发信号，AWG 的输出波形刷新一个点。
头文件	UserDef.h	

HAD8_AWG_BandWidth 类型定义

类型	Enumeration	
描述	滤波器带宽。	
元素	AWG_BandWidth_100kHz	低通 100kHz。
	AWG_BandWidth_10kHz	低通 10kHz。
	AWG_BandWidth_ByPass	直通。
头文件	UserDef.h	

HAD8_AWG_Range 类型定义

类型	Enumeration	
描述	电压挡位。	
元素	HAD_AWG2V5	±2.5V 挡。
	HAD_AWG5V	±5V 挡。
	HAD_AWG10V	±10V 挡。
头文件	UserDef.h	

2.6.9.2 Clear（）

函数功能

用于关闭 AWG 的同步模式、电压扫描模式、外部触发模式，恢复正常模式。

使用说明

无。

函数原型

```
int Clear(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().Clear(strPinName);
```

2.6.9.3 ClearAwgLoader（）

函数功能

清除加载的波形信息。

使用说明

无。

函数原型

```
int ClearAwgLoader(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().ClearAwgLoader(strPinName);
```

2.6.9.4 CreateRampData（）

函数功能

用于生成一个周期的斜波数据。

使用说明

无。

函数原型

```
int CreateRampData(double* awgData, uint32_t uDataNumber, double startValue,
double stopValue);
```

参数说明

参数	说明
awgData	存放波形数据的数组指针。
uDataNumber	1 个周期波形的数据点数。
startValue	起始点电压值，单位：V。
stopValue	最终点电压值，单位：V。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dawgData[256] = {0};
TheInst.HAD8().AWG().CreateRampData(dawgData,1,1.0,3.0);
```

2.6.9.5 CreateSineData（）

函数功能

用于生成一个周期的正弦波数据。

使用说明

无。

函数原型

```
int CreateSineData(double* awgData, uint32_t uDataNumber, double Vpp, double
DCOffset, double phase);
```

参数说明

参数	说明
awgData	存放波形数据的数组指针。
uDataNumber	1 个周期波形的数据点数。
Vpp	波形的峰峰值，单位：V。
DCOffset	波形的偏置电压，单位：V。
phase	起始相位，单位为度。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dawgData[256] = {0};
TheInst.HAD8().AWG().CreateSineData(dawgData,1,1.0,0.1,30);
```

2.6.9.6 CreateSquareData（）

函数功能

用于生成一个周期的方波数据。

使用说明

无。

函数原型

```
int CreateSquareData(double* awgData, uint32_t uDataNumber, double Vpp, double DCOffset, double dutyCycle);
```

参数说明

参数	说明
awgData	存放波形数据的数组指针。
uDataNumber	1 个周期波形的数据点数。
Vpp	波形的峰峰值，单位：V。
DCOffset	波形的偏置电压，单位：V。
dutyCycle	占空比，单位为百分比。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dawgData[256] = {0};
TheInst.HAD8().AWG().CreateSquareData(dawgData,1,1.0,0.1,10);
```

2.6.9.7 CreateTriangleData（）

函数功能

用于生成一个周期的三角波数据。

使用说明

无。

函数原型

```
int CreateTriangleData(double* awgData, uint32_t uDataNumber, double Vpp, double DCOffset, double phase);
```

参数说明

参数	说明
awgData	存放波形数据的数组指针。
uDataNumber	1 个周期波形的数据点数。
Vpp	波形的峰峰值，单位：V。
DCOffset	波形的偏置电压，单位：V。
phase	起始相位，单位为度。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dawgData[256] = {0};
TheInst.HAD8().AWG().CreateTriangleData(dawgData,1,1.0,0.1,30);
```

2.6.9.8 ExtTrigStop（）

函数功能

用于停止 AWG 的外部触发功能。

使用说明

使用外部触发功能后，需要调用本函数来停止，保证后续其它操作能够正常执行。

函数原型

```
int ExtTrigStop(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().ExtTrigStop(strPinName);
```

2.6.9.9 Loader（）

函数功能

用于将波形数据载入对应通道。

使用说明

无。

函数原型

```
int Loader(const string& strPinName, double* dAwgData, int nAwgDataSize);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dAwgData	波形数据数组的指针。
nAwgDataSize	数据的个数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dawgData[256] = {0};
TheInst.HAD8().AWG().Loader(strPinName,strAwgName,dAwgData,256);
```

2.6.9.10 Loader（）

函数功能

通过波形名字将波形信息加载到 FPGA。

使用说明

无。

函数原型

```
int Loader(const string& strPinName, const string& strAwgName, double *awgData,
int awgSize, bool isCover);
```

参数说明

参数	说明
strPinName	Pin 的名字。
strAwgName	定义生成波形的名称，用于区别是否重新载入波形数据。
awgData	存入将要载入的波形数据。
awgSize	载入数组长度。
isCover	是否要重新加载波形。 ● 0：之前加载波形的基础上再进行加载。 ● 1：清除之前加载的数据再重新加载。

返回值

类型	说明
int	● 0：正常。 ● - 1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
const string cAwgName = "Sin0";
double awgData[1000]={0.0};
TheInst.HAD8().AWG().Loader(strPinName, cAwgName, awgData, 1000, 0);
```

2.6.9.11 Select（）

函数功能

用于对 AWG 的输出进行设置。

使用说明

无。

函数原型

```
int Select(const string& strPinName, double dAwgInterVal, int nAwgSize, int
nWaveCycle);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dAwgInterVal	每个点刷新的时间间隔，单位 μs，2 位小数有效，取值范围：1.00～10000000.00。
nAwgSize	1 个周期的波形数据个数，最大 65535。
nWaveCycle	循环的周期数，最大 65535。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().Select(strPinName,1.0,1024,1024);
```

2.6.9.12 Select（）

函数功能

通过波形名字等信息来选择用于输出的波形。

使用说明

无。

函数原型

```
int Select(const string& strPinName, const string& strAwgName, double
awgInterVal, int startAddr, int stopAddr, int waveCycle);
```

参数说明

参数	说明
strPinName	Pin 的名字。
strAwgName	波形的名称。
awgInterVal	各点刷新时间间隔。
startAddr	输出波形的起始地址。
stopAddr	输出波形的终止地址。
waveCycle	循环次数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
const string cAwgName = "Sin0";
TheInst.HAD8().AWG().Select(strPinName, cAwgName, 1.0, 0, 1000, 5);
```

2.6.9.13 SetSync（）

函数功能

设置通道工作在同步输出模式。

使用说明

- 指定通道可以是 1 个或多个，要求在同一块板卡内。
- 不能同时指定共用相同硬件资源的通道（ch0 和 ch4，ch1 和 ch5，ch2 和 ch6，ch3 和 ch7 共用硬件资源）。

函数原型

```
int SetSync(Sync_SW bIsSyncOn, const vector<string>& vecPinName);
```

参数说明

参数	说明
bIsSyncOn	打开或者关闭同步模式，其类型定义详见 Sync_SW 类型定义。
vecPinName	Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
vector<string> vecPinName;
vecPinName.push_back("Pin1");
const string strPinName = "Pin1";
TheInst.HAD8().AWG().SetSync(Sync_ON,vecPinName);
```

2.6.9.14 SetMode（）

函数功能

用于设置高精度 AWG 的模式设置，此处主要是挡位和滤波器设置。

使用说明

无。

函数原型

```
int SetMode(const string& strPinName, HAD8_AWG_Range emRange, HAD8_AWG_BandWidth
emBandWidth);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emRange	电压挡位，其类型的定义详见 HAD8_AWG_Range 类型定义。
emBandWidth	滤波器带宽，其类型的定义详见 HAD8_AWG_BandWidth 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().SetMode(strPinName,HAD_AWG2V5,AWG_BandWidth_10kHz);
```

2.6.9.15 SetVal（）

函数功能

用于设置 AWG 输出的电压值，在直流输出时或者交流输出前设置初始电压使用。

使用说明

无。

函数原型

```
int SetVal(const string& strPinName, double dVal);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dVal	设置输出的电压值。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().SetVal(strPinName,1.0);
```

2.6.9.16 SetExtTrig（）

函数功能

用于 AWG 的外部触发功能设置。

使用说明

设置后不需要通过 HAD8 的 AwgRun（）或者 HAD8 的 SyncRun（）来启动，只需等待外部触发信号。

函数原型

```
int SetExtTrig(const string& strPinName, Edge emPosOrNeg, ExtTrig_Model
emTrigType = Normal_Trig);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emPosOrNeg	沿触发方式，其类型定义详见 Edge 类型定义。
emTrigType	选择正常触发模式或多次触发模式，其类型定义详见 ExtTrig_Model 类型定义。

返回值

类型	说明
int	● 0：正常。 ● - 1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().SetExtTrig(strPinName,Pos_Edge,Normal_Trig);
```

2.6.9.17 Run（）

函数功能

用于启动 AWG 输出波形（在正常模式下）。

使用说明

无。

函数原型

```
int Run(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().AWG().Run(strPinName);
```

2.6.10 DIG

2.6.10.1 类型定义

emMeasure_Mode 类型定义

类型	Enumeration	
描述	测试结果类型。	
元素	Aver	采样结果平均值。
	Max	采样结果最大值。
	Min	采样结果最小值。
	Delt	采样结果最大值减最小值。
头文件	UserDef.h	

HAD8_HAccuracy_Range 类型定义

类型	Enumeration	
描述	电压量程挡位。	
元素	HAccuracy_100mV	±100mV 挡。
	HAccuracy_200mV	±200mV 挡。
	HAccuracy_500mV	±500mV 挡。
	HAccuracy_1V	±1V 挡。
	HAccuracy_2V	±2V 挡。
	HAccuracy_5V	±5V 挡。
	HAccuracy_10V	±10V 挡。
	HAccuracy_20V	±20V 挡。
	HAccuracy_50V	±50V 挡。
	HAccuracy_100V	±100V 挡。
	HAccuracy_200V	±200V 挡。

头文件	UserDef.h

HAD8_BandWidth 类型定义

类型	Enumeration	
描述	滤波器带宽。	
元素	BandWidth_100kHz	低通 100kHz。
	BandWidth_10kHz	低通 10kHz。
	BandWidth_ByPass	直通。
头文件	UserDef.h	

2.6.10.2 Clear（）

函数功能

用于同时关闭 Digitizer 的同步模式和外部触发模式，恢复正常模式。

使用说明

无。

函数原型

```
int Clear(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().Clear(strPinName);
```

2.6.10.3 ChkSetMode（）

函数功能

检查本次 SetMode 和上次 SetMode 参数是否一致，如果一致，则不用切换继电器。

使用说明

无。

函数原型

```
int ChkSetMode(const string& strPinName, HAD8_Mode mode, HAD8_HAccuracy_Range range, HAD8_BandWidth bandwidth, double freq, unsigned int num, bool offsetenable = false, double OffSetVal = 0.0);
```

参数说明

参数	说明
strPinName	Pin 的名字。
mode	保留参数。
range	电压挡位，其类型定义详见 HAD8_HAccuracy_Range 类型定义。
bandwidth	滤波器带宽，其类型的定义详见 HAD8_BandWidth 类型定义。
freq	采样间隔。
num	采样点数。
offsetenable	OffSet 功能使能。
OffSetVal	offSetVal OffSet 的值。

返回值

类型	说明
int	0：成功。

示例

```
const string HAD8_VIN = "PIN0";
TheInst.HAD8().DIG().ChkSetMode(HAD8_VIN, HAD8_High_Accuracy, HAccuracy_50V, BandWidth_10kHz, 0.01, 100, 0, 0);
```

2.6.10.4 ChkSetMode_and_DigMeasureV（）

函数功能

将 SetMode 与 Measure 绑定操作的接口函数。

使用说明

无。

函数原型

```
int ChkSetMode_and_DigMeasureV(const string& strPinName, HAD8_Mode mode,
HAD8_HAccuracy_Range range, HAD8_BandWidth bandwidth, double freq, unsigned int
num, bool offsetenable, double dOffSetVal, bool bFoundFile);
```

参数说明

参数	说明
strPinName	Pin 的名字。
mode	保留参数。
range	电压挡位，其类型定义详见 HAD8_HAccuracy_Range 类型定义。
bandwidth	滤波器带宽，其类型定义详见 HAD8_BandWidth 类型定义。
freq	采样间隔。
num	采样点数。
offsetenable	OffSet 功能使能。
dOffSetVal	offSetVal OffSet 的值。
bFoundFile	是否记录文件。 ● true：记录文件。 ● false：不记录文件。

返回值

类型	说明
int	0：成功。

示例

```
const string HAD8_VIN = "PIN0";
TheInst.HAD8().DIG().ChkSetMode_and_DigMeasureV(HAD8_VIN,HAD8_High_Accuracy,
HAccuracy_50V, BandWidth_10kHz, 0.01, 100, 0, 0, false);
```

2.6.10.5 ChkSetMode_and_DigFastMeasureV（）

函数功能

将 SetMode 与 DigFastMeasureV 绑定操作的接口函数。

使用说明

无。

函数原型

```
int ChkSetMode_and_DigFastMeasureV(const string& strPinName, HAD8_Mode mode,
HAD8_HAccuracy_Range range, HAD8_BandWidth bandwidth, double freq, unsigned int
num, bool offsetenable, double dOffSetVal, emMeasure_Mode emType);
```

参数说明

参数	说明
strPinName	Pin 的名字。
mode	保留参数。
range	电压挡位，其类型定义详见 HAD8_HAccuracy_Range 类型定义。
bandwidth	滤波器带宽，其类型定义详见 HAD8_BandWidth 类型定义。
freq	采样间隔。
num	采样点数。
offsetenable	OffSet 功能使能。
dOffSetVal	offSetVal OffSet 的值。
emType	获取测试结果的类型，其类型定义详见 emMeasure_Mode 类型定义。

返回值

类型	说明
int	0：成功。

示例

```
const string HAD8_VIN = "PIN0";
ChkSetMode_and_DigFastMeasureV(HAD8_VIN, HAD8_High_Accuracy, HAccuracy_50V,
BandWidth_10kHz, 0.01, 100, ture, 0, emVI_Aver);
```

2.6.10.6 ExtTrigStop（）

函数功能

用于停止 Digitizer 的外部触发功能。

使用说明

使用外部触发功能后，需要调用本函数来停止，保证后续其它操作能够正常执行。

函数原型

```
int ExtTrigStop(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● - 1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().ExtTrigStop(strPinName);
```

2.6.10.7 FastMeasureV（）

函数功能

该函数用于直流测量，启动直流快速测量，获取当前采样点数的最大值、最小值、平均值以及差值。

使用说明

无。

函数原型

```
int FastMeasureV(const string& strPinName, emMeasure_Mode emType);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emType	获取结果类型，其类型定义详见 emMeasure_Mode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nMeasureV = TheInst.HAD8().DIG().FastMeasureV(strPinName,Aver);
```

2.6.10.8 FastGetTHDAndSNR（）

函数功能

用于快速获取 THD、SNR 以及 SINAD 参数。

使用说明

该函数无需用户获取 ADC 采样数据，驱动会通过从 FPGA 获取基波数据（H1）、2～5 次谐波数据（H2～H5）以及噪声和数据（Noise），再根据这些数据计算 THD、SNR、SINAD，计算公式如下所示：

- $\mathrm{THD} = 20 * \log_{10} \frac{\sqrt{H2^2+H3^2+H4^2+H5^2}}{H1}$
- $\mathrm{SNR} = 20 * \log_{10} \frac{H1}{\sqrt{Noise}}$
- $\mathrm{SINAD} = 20 * \log_{10} \frac{H1}{\sqrt{Noise+H2^2+H3^2+H4^2+H5^2}}$

FPGA 内部采用如下算法计算基波、谐波以及噪声和：

算法	说明	备注
自动模式	与函数 U_HAD8_GetThd（）计算结果相近（因快速计算精度原因产生的差异，差异<1dB）。	● 通常情况下，自动模式和手动模式计算结果相近，但是在信号直流分量较大的情况下，计算结果可能存在较大偏差。考虑到手动模式需要调整 signal bin 的范围，具体值受信号频率、采样率、FFT 点数影响，使用手动模式前请先分析频谱，或者向厂家技术人员咨询。 ● 关于直流分量处理策略 在 FFT 之后，FPGA 会将[0，3]区间的值置 0（完整区间为 0～N/2，N 表示 FFT 点数），对应的频率范围根据采样率以及 FFT 点数可计算得到。
手动模式	在计算基波和谐波值时，FPGA 根据 SetSignalBin（）函数设置的值在基波、2～5 次谐波的中心频点左右分别选取设定值范围内的数据点计算入基波和谐波，除此之外的数据点当成噪声。 ● 该模式为非标准通用算法。 ● 通过设置合适的入参，该模式计算结果与标准算法趋向一致。	

函数原型

```
int FastGetTHDAndSNR(int nSite, const string& strPinName, double &dTHD, double &dSNR, double &dSINAD);
```

参数说明

参数	说明
nSite	测试工位
strPinName	Pin 的名称。
dTHD	用于输出 THD 参数。
dSNR	用于输出 SNR 参数。
dSINAD	用于输出 SINAD 参数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
double dThd, dSNR, dSINAD;
const string strPinName = "Pin1";
TheInst.HAD8().DIG().FastGetTHDAndSNR(0, strPinName,dThd, dSNR,dSINAD);
```

2.6.10.9 FastGetRms（）

函数功能

用来快速获取 RMS 参数。

使用说明

该接口无需用户获取 ADC 采集数据，RMS 数据由 PFGA 内部计算，节约 RMS 参数获取时间。

函数原型

```
int FastGetRms(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().FastGetRms(strPinName);
```

2.6.10.10 FftWindowTypeSet（）

函数功能

用于 FFT 窗函数设置。

使用说明

不调用该接口进行设置时，窗默认为 R-V（III）窗。

函数原型

```
int FftWindowTypeSet(const string& strPinName, emFftWindowType windowType);
```

参数说明

参数	说明
strPinName	Pin 的名称。
windowType	FFT 窗类型。 emFftWindowType 枚举类型定义如下： ● Fft_RVIII：R-V（III）窗。 ● Fft_Hamming：Hamming 窗。 ● Fft_Hanning：Hanning 窗。 ● Fft_Kaiser：Kaiser 窗。 ● Fft_Blackman_Harris：Blackman_Harris 窗。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().FftWindowTypeSet(strPinName,Fft_RVIII );
```

2.6.10.11 GetAverResult（）

函数功能

用于获取测量数据的平均值。

使用说明

在调用本函数前需先调用 HAD8_DigMeasureV（）将测量数据获取到上位机。

函数原型

```
int GetAverResult(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nAver = TheInst.HAD8().DIG().GetAverResult(strPinName);
```

2.6.10.12 GetMaxAndMin（）

函数功能

用于获取测量数据中的最大值和最小值。

使用说明

在调用本函数前需先调用 HAD8_DigMeasureV（）将测量数据获取到上位机。

函数原型

```
int GetMaxAndMin(int nSite, const string& strPinName, double& dMax, double&
dMin);
```

参数说明

参数	说明
nSite	Site 号（0，1，2，…，63）。
strPinName	Pin 的名称。
dMax	用于存放最大值的变量。
dMin	用于存放最小值的变量。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dMax;
double dMin;
int nMaxMin = TheInst.HAD8().DIG().GetMaxAndMin(0,strPinName,dMax,dMin);
```

2.6.10.13 GetFastMeasureV（）

函数功能

该函数主要用于除获取平均值以外有获取其他类型数据的需求，例如最大值，最小值等。

使用说明

- 与函数 HAD8_DigFastMeasureV 依附使用，根据设置的返回值枚举类型获取函数 HAD8_DigFastMeasureV 测量到的对应数值并保存至“pSite > dRealData[i]”中，i 与实际工作的 Site 相对应。
- 若单独调用该函数，而没有在该函数之前调用 HAD8_DigFastMeasureV，则获取的数据无效，均为 0。

函数原型

```
int GetFastMeasureV(const string& strPinName, emMeasure_Mode emType);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emType	测试结果类型，其类型定义详见 emMeasure_Mode 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nFastMeasureV = TheInst.HAD8().DIG().GetFastMeasureV(strPinName,emVI_Aver);
```

2.6.10.14 GetRms（）

函数功能

用于获取测量数据的有效值（仅限正弦波）。

使用说明

调用本函数前需先调用 HAD8_DigMeasureV（）将测量数据获取到上位机。

函数原型

```
int GetRms(const string& strPinName, int nCycleNum);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nCycleNum	1 个周期内的采样点数，需要用户根据信号周期和采样间隔去确定。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nAver = TheInst.HAD8().DIG().GetRms(strPinName, nCycleNum);
```

2.6.10.15 GetSpecificLocateV（）

函数功能

通过形参 uIndex 获取采样任意点的值。

使用说明

无。

函数原型

```
int GetSpecificLocateV(const string& strPinName, unsigned int uIndex);
```

参数说明

参数	说明
strPinName	Pin 的名称。
uIndex	获取指定某个采样点的数据。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().GetSpecificLocateV(strPinName,100);
```

2.6.10.16 GetSTDEVal（）

函数功能

用于用户获取各个 Site 测量数据标准差。

使用说明

无。

函数原型

```
int GetSTDEVal(const string& strPinName, double* dSTDE, int nSTDESize =
MAXSITE_NUM);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dSTDE	用于存储标准差的数组首地址。
nSTDESize	每个 Site 的标准差。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
double dStde[MAXSITE_NUM] = {0};
const string strPinName = "Pin1";
int nAver = TheInst.HAD8().DIG().GetSTDEResult(strPinName,dStde);
```

2.6.10.17 GetSTDEResult（）

函数功能

用于用户获取测量数据标准差，结果存储在 RealData 数组内。

使用说明

无。

函数原型

```
int GetSTDEResult(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nAver = TheInst.HAD8().DIG().GetSTDEResult(strPinName);
```

2.6.10.18 GetTHDAndSNR（）

函数功能

该函数用于获取 DIG 的 THD 和 SNR 的数据。

使用说明

使用该函数前在设置 HAD8_DigSetMode（）时确保采样频率至少是被测信号的频率的 10 倍，因为根据 Nyquest 定理，并且如果 THD 计算的是 5 次谐波，所以采样频率至少是被测信号的频率的 10 倍。

函数原型

```
int GetTHDAndSNR(int nSite, const string& strPinName, double& dTHD, double&
dSNR, const int nHSize = 5);
```

参数说明

参数	说明
strPinName	通道。
nSite	Site。
dTHD	获取的失真度。
dSNR	获取的信噪比。
nHSize	设置测量 THD 的谐波次数，默认是 5 次谐波，用户可以根据需要更改谐波次数，如果选择默认则 nHSize 可以不赋值。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dTHD;
double dSNR;
int nThdSnr = TheInst.HAD8().DIG().GetTHDAndSNR(0,strPinName,dTHD,dSNR);
```

2.6.10.19 GetValEachSite（）

函数功能

用于获取测量数据到指定数组。

使用说明

在调用本函数前需先调用 HAD8_DigMeasureV（）将测量数据获取到上位机。

函数原型

```
int GetValEachSite(int nSite, const string& strPinName, double* dEachData, int nEachDataSize);
```

参数说明

参数	说明
nSite	Site 号（0，1，2，…，63）。
strPinName	Pin 的名称。
dEachData	返回测量结果数组首地址。
nEachDataSize	获取结果长度。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().GetValEachSite(0, strPinName, dEachData, 10);
```

2.6.10.20 MeasureV（）

函数功能

用于软件启动电压测量，并将测量值获取到上位机。

使用说明

无。

函数原型

```
int MeasureV(const string& strPinName, bool bCreateFile = false);
```

参数说明

参数	说明
strPinName	Pin 的名称。
bCreateFile	是否创建文件，默认情况下不生成 CSV 文件。 当 bFoundFile 为 1 时，将获得的数据自动保存到相应 CSV 文件中，产生的 CSV 文件路径为“D:\CTA8290DPlus\Data\HAD 数据”，文件名为 HAD_CH 通道号时间。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().MeasureV(strPinName);
```

2.6.10.21 MeasureVSampleData（）

函数功能

获取所有采样点数据。

使用说明

无。

函数原型

```
int MeasureVSampleData(int nSite, const string& strPinName, double *dDigdata,
bool bFoundFile);
```

参数说明

参数	说明
nSite	Site 号（0，1，2，…，63）。
strPinName	Pin 的名称。
dDigdata	用于存储获取到的采样点。
bFoundFile	● 1：采样点保存到文件中。 ● 0：不保存。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dDigdata[1000]={0.0};
TheInst.HAD8().DIG().MeasureVSampleData(0,strPinName,dDigdata,0);
```

2.6.10.22 SampleSynDelay（）

函数功能

设定 Dig 触发延时时间。

使用说明

无。

函数原型

```
int SampleSynDelay(const string& strPinName, double dTuS);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dTuS	延时时间设置，最小 0.01μs。 单位：μs

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SampleSynDelay(strPinName,0.2);
```

2.6.10.23 SetExtTrig（）

函数功能

用于 Digitizer 的外部触发功能设置。

使用说明

设置完毕需要等待外部触发信号到来后，再通过 HAD8_DigMeasureV（）将测量数据获取到上位机。

函数原型

```
int SetExtTrig(const string& strPinName, Edge emPosOrNeg, ExtTrig_Model
emTrigType = Normal_Trig, int nTrigNum = 0);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emPosOrNeg	选择沿触发方式，其类型定义详见 Edge 类型定义。
emTrigType	选择正常触发模式或多次触发模式，其类型定义详见 ExtTrig_Model 类型定义。
nTrigNum	触发次数，在多次触发模式下需要设置，最大不超过 4M。 HAD8_DigMeasureV（）在获取数据时按照设置的触发次数去获取相应个数的数据。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SetExtTrig(strPinName,Pos_Edge,Normal_Trig);
```

2.6.10.24 SetFastRmsCfg（）

函数功能

用于设置计算 RMS 值的 DIG 数据的区间范围。

使用说明

- 一般情况下无需调用该函数，FPGA 会自动根据 Setmode（）获取计算范围，起始点为零点，数据长度为 SetMode（）设置的采样点数。
- 如需选取 DIG 数据的部分区间，可以使用该函数指定。

  例如使用 SetMode（）函数设置 4096 个采样点，若不使用该函数，则 FPGA 计算 DIG 数据 1～4096 区间的 RMS 值；若使用该函数，将 DataOffset 设置为 10，DataNum 设置为 1024，则 FPGA 计算 DIG 数据 11～1024 区间的 RMS 值，而其他的点不计入计算。
- 该函数调用顺序为：位于 SetMode（）函数后，DigStartWork（）函数前。

函数原型

```
int SetFastRmsCfg(const string& strPinName, unsigned int DataOffset, unsigned int DataNum, bool bIsCalFundWave);
```

参数说明

参数	说明
strPinName	Pin 的名称。
DataOffset	用来设置计算数据的偏移位置（相较于起始零点）。
DataNum	用来设置计算数据的长度。
bIsCalFundWave	用来设置是否计算直流分量。

返回值

类型	说明
int	● 0：正常。 ● -1：参数设置异常。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SetFastRmsCfg(strPinName,0, 4096,TRUE);
```

2.6.10.25 SetMode（）

函数功能

用于高精度测量功能的模式设置，包括量程挡位选择，滤波器选择，采样间隔、采样点数设置等。

使用说明

无。

函数原型

```
int SetMode(const string& strPinName, HAD8_HAccuracy_Range emRange, HAD8_BandWidth emBandWidth, double dInterval, int nDataNum, double dDCoffSet = 0.0);
```

参数说明

参数	说明
strPinName	Pin 的名称。
emRange	电压量程挡位。其类型定义详见 HAD8_HAccuracy_Range 类型定义。
emBandWidth	滤波器带宽，其类型定义详见 HAD8_BandWidth 类型定义。
dInterval	采样间隔，单位为 μs，2 位小数有效，范围 1.00～10000000.00。
nDataNum	采样点数，范围 1～64K。
dDCoffSet	保留参数。

返回值

类型	说明
int	● 0：正常。 ● -1：参数设置异常。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SetMode(strPinName,HAccuracy_100mV,BandWidth_100kHz,10,2);
```

2.6.10.26 SetSignalBin（）

函数功能

该函数用于配置算法采用手动模式时设置 signal bin 值。

使用说明

- 若调用 SetMode，噪声和默认为自动模式。
- 若在 SetMode 后调用该函数来设置 signal bin 范围，则切换为手动模式。
- 手动模式为非标准通用算法，使用前建议先分析频谱或向厂家技术人员咨询。

函数原型

```
int SetSignalBin(const string& strPinName, unsigned int value);
```

参数说明

参数	说明
strPinName	Pin 的名称。
value	signal bin 范围，取值为[0，10]。

返回值

类型	说明
int	● 0：正常。 ● - 1：参数设置异常。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SetSignalBin(strPinName,2);
```

2.6.10.27 SetSync（）

函数功能

用于同步测量模式相关设置，指定通道可以是 1 个或多个，要求在同一块板卡内，且共用相同硬件资源的通道不能被同时指定。

使用说明

ch0 和 ch4，ch1 和 ch5，ch2 和 ch6，ch3 和 ch7 共用硬件资源。

函数原型

```
int SetSync(Sync_SW bIsSyncOn, const vector<string>& vecPinName);
```

参数说明

参数	说明
bIsSyncOn	打开或者关闭同步模式，其类型定义详见 Sync_SW 类型定义。
vecPinName	Pin 的名称。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
vector<string> vecPinName;
vecPinName.push_back("Pin1");
const string strPinName = "Pin1";
TheInst.HAD8().DIG().SetSync(Sync_ON,vecPinName);
```

2.6.10.28 StartWork（）

函数功能

启动 ADC 测量。

使用说明

为提高执行效率该接口不进行各个采样点数据的获取，配合快速获取 RMS、THD、SNR 和 SINAD 接口使用

函数原型

```
int StartWork(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HAD8().DIG().StartWork(strPinName);
```

2.7 VIAWG

2.7.1 类型定义

emVI_MeasTrigType 类型定义

类型	Enumeration	
描述	触发模式。	
元素	emVI_Before_V=0	电压（触发前的点）。
	emVI_Before_I=1	电流（触发前的点）。
	emVI_After_V=2	电压（触发后的点）。
	emVI_After_I=3	电流（触发后的点）。
头文件	UserDef.h	

2.7.2 Clear（）

函数功能

清除 AWG 设置，结束 AWG 波形输出，输出值保持在 Clear 发送命令的时刻。

使用说明

无。

函数原型

```
int Clear(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Clear(strPinName);
```

2.7.3 ClearAwgLoader（）

函数功能

重置 AWG 参数，信号复位。

使用说明

无。

函数原型

```
int ClearAwgLoader(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
ClearAwgLoader(strPinName);
```

2.7.4 CreateSineData（）

函数功能

生成正弦波数据，存储在 dAwgData 数组中。

使用说明

无。

函数原型

```
int CreateSineData(double *dAwgData, int nAwgSize, double dVpp, double
dDcOffset, double dPhase);
```

参数说明

参数	说明
dAwgData	数组指针，用于保存生成的波形数据。
nAwgSize	数组长度，即波形中输出点的个数，最终载入数据不超过 VI 源限制。
dVpp	正弦波形峰峰值大小。
dDcOffset	设置直流偏置大小。
dPhase	相位偏移大小。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dAwgData[256] = {0};
TheInst.VIAWG().CreateSineData (dAwgData, 256, 5, 0, 0);
```

2.7.5 CreateTriangleData（）

函数功能

生成三角波数据，存储在 dAwgData 数组中。

使用说明

无。

函数原型

```
int CreateTriangleData(double *dAwgData, int nAwgSize, double dVpp, double
dDcOffset, double dPhase);
```

参数说明

参数	说明
dAwgData	数组指针，用于保存生成的波形数据。
nAwgSize	数组长度，即波形中输出点的个数。
dVpp	三角波形峰峰值大小。
dDcOffset	设置直流偏置大小。
dPhase	相位偏移大小。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dAwgData[256] = {0};
TheInst.VIAWG().CreateTriangleData (dAwgData, 256, 5, 0, 0);
```

2.7.6 CreateSquareData（）

函数功能

生成方波数据，存储在 dAwgData 数组中。

使用说明

无。

函数原型

```
int CreateSquareData(double *dAwgData, int nAwgSize, double dVpp, double
dDcOffset, double dDutyCycle);
```

参数说明

参数	说明
dAwgData	数组指针，用于保存生成的波形数据。
nAwgSize	数组长度，即波形中输出点的个数。
dVpp	方波峰峰值大小。
dDcOffset	设置直流偏置大小。
dDutyCycle	占空比。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dAwgData[256] = {0};
TheInst.VIAWG().CreateSquareData(dAwgData, 256, 5, 2.5, 50);
```

2.7.7 CreateRampData（）

函数功能

生成斜波数据，存储在 dAwgData 数组中。

使用说明

无。

函数原型

```
int CreateRampData(double *dAwgData, int nAwgSize, double dStartValue, double
dStopValue);
```

参数说明

参数	说明
dAwgData	数组指针，用于保存生成的波形数据。
nAwgSize	数组长度，即波形中输出点的个数。
dStartValue	初始值大小。
dStopValue	结束值大小。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
double dAwgData[256] = {0};
TheInst.VIAWG().CreateRampData (dAwgData, 256, 2.0, 3.0);
```

2.7.8 Disable（）

函数功能

用于同步测试（目前只针对 FVI16），取消该资源板通道对 TIF 的同步信号响应。

使用说明

该函数与 Enable 配套使用。

函数原型

```
int Disable(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Disable(strPinName);
```

2.7.9 Enable（）

函数功能

用于同步测试，设置该资源板通道对 TIF 的同步信号响应，可以通过连接到 TIF 上的同步总线，控制 FVI 的资源板同步输出或测量。

使用说明

无。

函数原型

```
int Enable(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Enable(strPinName);
```

2.7.10 GetTrigData（）

函数功能

返回值为触发点对应的理论值，返回值存储在“pSite > dRealVal”中。

使用说明

该函数用于 RunTrigXXX 之后。

- 若 SetTrigVal 设置的 nMode 为 0、1，返回值为触发前一个点的值。
- 若 SetTrigVal 设置的 nMode 为 2、3，返回值为触发时的值。

函数原型

```
int GetTrigData(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().GetTrigData(strPinName);
```

2.7.11 GetTrigVal（）

函数功能

返回输出通道的设定值大小，返回值存储在“pSite > dRealVal”中。

使用说明

- 该函数用于 RunTrigXXX 之后。
- 若 SetTrigVal 设置的 nMode 为 0、1，返回值为触发前一个点的值。
- 若 SetTrigVal 设置的 nMode 为 2、3，返回值为触发时的值。

- 与 AwgGetTrigData 类似，差别在于返回的是理论值。由于这 2 个函数都需要以当前存储在 pSite->dRealVal 中的触发点位置为输入再返回相应结果存入“pSite > dRealVal”，2 者不能同时使用，且都只能用于 RunTrigXXX 之后。

函数原型

```
int GetTrigVal(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为输出通道的设定值大小。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().GetTrigVal(strPinName);
```

2.7.12 Loader（）

函数功能

载入输出通道的 AWG 波形数据，将 dAwgData 数组中的数据载入到对应的 VI 源及通道。

使用说明

对于 VI 源，属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间，使用时需合理分配通道与波形点数。

相同 Slave FPGA 判断依据：物理通道号/Slave FPGA 通道数（除法计算），商值相同。

- 对于 FVI2，nAwgSize∈[1,4096]。
- 对于 FVI8/FVI16/FVI16Plus，单通道使用 nAwgSize∈[1,4096]，4 通道同时使用 nAwgSize∈[1,1024]。
  - ♦ 若当前通道 nAwgSize∈[1025,2048]，后面 1 通道不能同时使用。
  - ♦ 若当前通道 nAwgSize∈[2049,3072]，后面 2 通道不能同时使用。

- ♦ 若当前通道 nAwgSize∈[3073,4096]，只能单通道使用。
- ♦ 通道顺序为 ch0、ch1、ch2、ch3，环形结构。

函数原型

```
int Loader(const string& strPinName, const string& strAwgName, double* dAwgData,
int nAwgSize);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。
strAwgName	原定义生成波形的名称，用于区别是否重新载入波形数据，目前无实际意义，可设置任意字符串类型。
dAwgData	存储了需要载入到 VI 源中的波形数据的数组。
nAwgSize	载入数组长度，波形中输出点的个数，即 dAwgData 中需要载入的数据个数，详细说明请参见使用说明。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Loader(strPinName, “RAMP”, dAwgData, 100);
```

2.7.13 Loader（）

函数功能

载入 AWG 波形数据。

使用说明

- 对于 VI 源，属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间，使用时需合理分配通道与波形点数。

  相同 Slave FPGA 判断依据：物理通道号/Slave FPGA 通道数（除法计算），商值相同。

- 对于 FVI2，nAwgSize∈[1,4096]。
- 对于 FVI8/FVI16/FVI16Plus，单通道使用 nAwgSize∈[1,4096]，4 通道同时使用 nAwgSize∈[1,1024]。
  - 若当前通道 nAwgSize∈[1025,2048]，后面 1 通道不能同时使用。
  - 若当前通道 nAwgSize∈[2049,3072]，后面 2 通道不能同时使用。
  - 若当前通道 nAwgSize∈[3073,4096]，只能单通道使用。
  - 通道顺序为 ch0、ch1、ch2、ch3，环形结构。

- 对于 VI 源，如果参数 isCover 为 false，则当前通道各次加载波形点数之和需遵循前述 4096 存储空间分配原则。

函数原型

```
int Loader(const string& strPinName, const string& strAwgName, double *dAwgData, int nAwgSize,bool isCover);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
strAwgName	定义生产波形的名称，用于区别是否重新载入波形数据。
dAwgData	需要载入的波形数据的数组。
nAwgSize	载入数组长度，即波形输出点的个数，详细说明请参见使用说明（Loader）。
isCover	是否覆盖已加载的 AWG 波形数据。 HPS100 该参数无效。

返回值

类型	说明
int	• 0：正常。 • -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
```

```
Loader(strPinName,AwgName1,AwgData1,n1,false);//pin1 加载波形 AwgName1，波形 n1 点
Loader(strPinName,AwgName2,AwgData2,n2,false);//pin1 加载波形 AwgName2，波形 n2 点
//pin1 当前共加载(n1+n2)点，(n1+n2)需遵循前述 4096 存储空间分配原则
```

2.7.14 MeasureComp（）

函数功能

对测量结果进行补偿。

使用说明

无。

函数原型

```
vector<double> MeasureComp(const string& strPinName, double* dMeasData, unsigned int uMeasNum, double dCompCoef);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
dMeasData	测量点的数组。
uMeasNum	测量点的个数。
dCompCoef	补偿系数,，可取范围（0, 1）。

返回值

类型	说明
vector<double>	返回补偿结果。

示例

```
vector<double> Comp_data;
const string strPinName = "Pin1";
double* dMeasData = new double[1024];
Comp_data = TheInst.CVIAWG().MeasureComp(strPinName, dMeasData, 10, 0.5);
```

2.7.15 MeasResult（）

函数功能

将相应 Site 和通道的 AWG 测试存储的波形数据，返回到数组 dMeasData 当中。

使用说明

无。

函数原型

```
int MeasResult(int nSite, const string& strPinName, emVI_MeasType emType,
double* dMeasData);
```

参数说明

参数	说明
nSite	测试工位。
strPinName	单 Pin 或 PinGroup 的名称
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dMeasData	用于存储 AWG 波形数据数组。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double * dMeasData = new double[1024];
TheInst.FVI16().DCVI().MeasResult(0, strPinName, 5 , 0);
delete []dMeasData;
```

2.7.16 Run（）

函数功能

启动相应通道的 AWG 进行输出，此时输出不区分是否设置同步。

使用说明

该函数目前只用于单通道的 AWG 波形输出，无同步设置。当 AwgSelect 设置循环次数不等于 1 时，使用 RunTrigXXX 时仍只能输出一个周期，需使用 AwgRun 进行多周期输出。停止输出时使用 AwgClear。

函数原型

```
int Run(const string& strPinName);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Run(strPinName);
```

2.7.17 RunTrigSync（）

函数功能

以 TrigChannel 作为触发通道，输出各个 OutChannel 通道的 AWG 波形，以按照设定波形输出直到结束，返回 TrigChannel 的触发地址存储在“pSite > RealData”中。

使用说明

- vecOutPinName 最大可以设置 16 个通道。大于 31 的逻辑通道建议调整逻辑通道顺序。
- emVISOURCE 类型的数据可设置 FVI_CH0，FVI_CH1，FVI_CH2……FVI_CH31，对应 FVI 相应的逻辑通道。测量通道有且只有一个，输出通道也可以同时作为测量通道。
- 运行结束后，“pSite > RealData/dRealVal”中存储了最后一个触发点的位置，当整个过程无触发时，返回地址为最后一个点。

函数原型

```
int RunTrigSync(const string& strTrigPinName, const vector<string>&
vecOutPinName);
```

参数说明

参数	说明
strTrigPinName	触发 Pin 的名字。
vecOutPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
vector<string> vec_OutPinName;
vec_OutPinName.push_back("Pin1");
TheInst.VIAWG().RunTrigSync(strPinName, vec_OutPinName);
```

2.7.18 RunTrigSync（）

函数功能

函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 AwgLoader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型 Pin 的名字。

使用说明

bKeepCfg 表示此次 AWG 运行完成后，是否保持此次配置，用于下一次的 AWG 输出，默认为否。

正反向输出操作步骤：

1. 完整设置正向输出，并在 AWG_RunTrigSync 时，设置 bKeepCfg=1。
2. 设置各通道 AwgSelect 为反向，运行 AWG_RunTrigSync 时则以前一次配置的通道进行输出与测量。

函数原型

```
int RunTrigSync(bool bKeepCfg = false);
```

参数说明

参数	说明
bKeepCfg	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.VIAWG().RunTrigSync(false);
```

2.7.19 RunTrigStop（）

函数功能

输出各个 OutChannel 通道的 AWG 波形，当 TrigChannel 通道触发时，在相应的 SITE 触发后保持输出并返回此时的触发地址，返回地址存储在“pSite > RealData”中。

使用说明

- vecOutPinName 最大可以设置 16 个通道，同一小板内的通道不能同时作为输出 Pin 的名字。
- 同类源逻辑通道可设置为 0～31。
- emVISOURCE 类型的数据可设置 FVI_CH0，FVI_CH1，FVI_CH2……FVI_CH31，对应 FVI 相应的逻辑通道。
- 运行结束后，“pSite > RealData/dRealVal”中存储了第一个触发点的位置，当整个过程无触发时，返回地址为最后一个点。
- 简化函数中，函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 AwgLoader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型 Pin 的名字。

函数原型

```
int RunTrigStop(const string& strTrigPinName, const vector<string>&
vecOutPinName);
```

参数说明

参数	说明
strTrigPinName	触发 Pin 的名字。
vecOutPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
 string strPinName = "Pin1";
vector<string> vec_OutPinName;
vec_OutPinName.push_back("Pin1");
TheInst.VIAWG().RunTrigStop(strPinName,vec_OutPinName);
```

2.7.20 RunTrigStop（）

函数功能

输出各个 OutChannel 通道的 AWG 波形，当 TrigChannel 通道触发时，在相应的 SITE 触发后保持输出并返回此时的触发地址，返回地址存储在“pSite > RealData”中。

使用说明

- vecOutPinName 最大可以设置 16 个通道，同一小板内的通道不能同时作为输出 Pin 的名字。
- 同类源逻辑通道可设置为 0～31。
- emVISOURCE 类型的数据可设置 FVI_CH0，FVI_CH1，FVI_CH2……FVI_CH31，对应 FVI 相应的逻辑通道。
- 运行结束后，“pSite > RealData/dRealVal”中存储了第一个触发点的位置，当整个过程无触发时，返回地址为最后一个点。
- 简化函数中，函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 AwgLoader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型 Pin 的名字。

函数原型

```
int RunTrigStop(bool bKeepCfg = false);
```

参数说明

参数	说明
bKeepCfg	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.VIAWG().RunTrigStop(false);
```

2.7.21 RunTrigStopVal（）

函数功能

输出各个 OutChannel 通道的 AWG 波形，当 TrigChannel 通道触发时，在相应的 SITE 触发后 OutChannel 输出设定值并返回此时的触发地址，返回地址存储在“pSite > RealData”中。

使用说明

- vecOutPinName 最大可以设置 16 个通道，同一小板内的通道不能同时作为输出 Pin 的名字。
- 同类源逻辑通道可设置为 0～31。
- emVISOURCE 类型的数据可设置 FVI_CH0，FVI_CH1，FVI_CH2……FVI_CH31，对应 FVI 相应的逻辑通道。
- 简化函数中，函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 AwgLoader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型 Pin 的名字。

函数原型

```
int RunTrigStopVal(double dOutVal, const string& strTrigPinName, const vector<string>& vecOutPinName);
```

参数说明

参数	说明
dOutVal	触发停止后输出通道的输出值。

参数	说明
strTrigPinName	触发 Pin 的名字。
vecOutPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
vector<string> vec_OutPinName;
vec_OutPinName.push_back("Pin1");
TheInst.VIAWG().RunTrigStopVal(0.0, strPinName, vecOutPinName);
```

2.7.22 RunTrigStopVal（）

函数功能

输出各个 OutChannel 通道的 AWG 波形，当 TrigChannel 通道触发时，在相应的 SITE 触发后 OutChannel 输出设定值并返回此时的触发地址，返回地址存储在 pSite－＞RealData 中。

使用说明

- vecOutPinName 最大可以设置 16 个通道，同一小板内的通道不能同时作为输出 Pin 的名字。
- 同类源逻辑通道可设置为 0～31。
- emVISOURCE 类型的数据可设置 FVI_CH0，FVI_CH1，FVI_CH2……FVI_CH31，对应 FVI 相应的逻辑通道。
- 简化函数中，函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 AwgLoader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型 Pin 的名字。

函数原型

```
int RunTrigStopVal(double dOutVal, bool bKeepCfg = false);
```

参数说明

参数	说明
dOutVal	触发停止后输出通道的输出值。
bKeepCfg	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.VIAWG().RunTrigStopVal(0.0);
```

2.7.23 RunTrigStopNum（）

函数功能

触发停止后输出 Pin 的输出值。

使用说明

无。

函数原型

```
int RunTrigStopNum(double dOutVal, const string& strTrigPinName, const
vector<string>& vecOutPinName);
```

参数说明

参数	说明
dOutVal	触发停止后输出通道的输出相应地址的输出值。
strTrigPinName	触发 Pin 的名字。
vecOutPinName	输出 Pin 的名字。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strTrigPinName = "Pin1";
vector<string> vec_OutPinName;
vec_OutPinName.push_back("Pin1");
TheInst.VIAWG().RunTrigStopNum(10,strTrigPinName,vec_OutPinName);
```

2.7.24 RunTrigStopNum（）

函数功能

函数自动通过函数前面设置的 SetTrigVal 锁定触发通道，通过 CVIAWG::Loader 锁定输出通道，然后自动设置这些通道进行 AWG 输出与测量，无需另外写入枚举类型通道号。

使用说明

无。

函数原型

```
int RunTrigStopNum(double dOutVal, bool bKeepCfg = false);
```

参数说明

参数	说明
dOutVal	输出 Pin 的名字。
bKeepCfg	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.VIAWG().RunTrigStopNum(10 ,false);
```

2.7.25 Select（）

函数功能

设置波形的输出与测量参数，设置起始地址，终止地址，循环次数及输出间隔等。

AWG 每个输出点的时间间隔为 dAwgInterVal，在当前输出点的时间间隔内需要完成采样的过程才能进行下一个点的输出，因此，时间参数上要求如下：

- 对于 FVI2/FVI8/FVI16，$dAwgInterVal \geq dAwgMeasDelay + nSampNum * dSampTms$。
- 对于 FVI16Plus，$dAwgInterVal \geq dAwgMeasDelay + dSampTms * (nSampNum + 1)$且 $dAwgMeasDelay + dSampTms * 2 \geq 7\mu s$。

同步输出与测量的所有通道均需按照相同的参数进行 AwgSelect 设置，保证输出与测量同步性。

使用说明

nSampNum 取值不同，存在如下差异：

条件	说明	示例
nSampNum !=0，启动 DIG 采样测量	若当前通道同时 AWG 输出时，存在如下情况： ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096，且不大于当前与同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：(|nStopAddr-nStartAddr| +1)最大为 2048，且不大于当前与同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 loader 波形 500 点；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则(|nStopAddr-nStartAddr| +1)应不大于 500（不大于当前与同步通道波形加载点数）。
	若当前通道仅测量时，存在如下情况： ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096，且不大于同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：(|nStopAddr-nStartAddr| +1)最大为 2048，且不大于同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 仅测量；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则(|nStopAddr-nStartAddr| +1)应不大于 1000（不大于同步通道波形加载点数）。

条件	说明	示例
nSampNum =0，仅 AWG 输出	属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间。 ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096。 ● FVI8/FVI16/FVI16Plus：单通道使用(|nStopAddr-nStartAddr| +1)最大为 4096；4 通道同时使用(|nStopAddr-nStartAddr| +1)最大为 1024。 ♦ 若当前通道(|nStopAddr-nStartAddr| +1)∈[1025,2048]，后面 1 通道不能同时使用。 ♦ 若当前通道(|nStopAddr-nStartAddr| +1)∈[2049,3072]，后面 2 通道不能同时使用。 ♦ 若当前通道(|nStopAddr-nStartAddr| +1)∈[3073,4096]，只能单通道使用。 ♦ 通道顺序 ch0、ch1、ch2、ch3，环形结构。	

函数原型

```
int Select(const string& strPinName, double dAwgInterVal, double dAwgMeasDelay,
int nSampNum, double dSampTms, int nStartAddr, int nStopAddr, int nWaveCycle);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。
dAwgInterVal	输出点之间的时间间隔，单位 ms，最小 10us，最大 50ms。
dAwgMeasDelay	输出后等待多长时间开始采样，单位 ms，最小 0，最大 50ms。
nSampNum	每次输出后采样点的个数，设置为 0 时不启动采样测量，最大 200。
dSampTms	采样间隔，单位 ms，最大 30ms。 FVI8/FVI16 最小 10μs，FVI2/FVI16Plus 最小 2μs。
nStartAddr	输出波形的起始地址。
nStopAddr	输出波形的终止地址，波形统一由 nStartAddr 向 nStopAddr 输出，当 nStartAddr＞nStopAddr 时，波形与载入的顺序相反，每个周期输出的总点个数为 nStopAddr-nStartAddr+1。
nWaveCycle	循环次数。 ● 当 nWaveCycle 不为 1 的情况下，运行 AWG 只能使用 AwgRun 函数。 ● 当循环次数设置>1000 或设置为 0 时，波形输出无限循环。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Select(strPinName, 0.5, 0.1, 20, 0.01, 0, 99, 0);
```

2.7.26 Select（）

函数功能

设置波形的输出与测量参数，设置起始地址，终止地址，循环次数及输出间隔等。

使用说明

AWG 每个输出点的时间间隔为 dAwgInterVal，在当前输出点的时间间隔内需要完成采样的过程才能进行下一个点的输出，因此，时间参数上要求如下：

- 对于 FVI2/8/16，$dAwgInterVal \geq dAwgMeasDelay + nSampNum * dSampTms$。
- 对于 FVI16Plus，$dAwgInterVal \geq dAwgMeasDelay + dSampTms * (nSampNum + 1)$且 $dAwgMeasDelay + dSampTms * 2 >= 7\mu s$。
- 其中内部 dAwgMeasDelay 设置为 dAwgInterVal 的 1/2，但当$0.5 * double\ dAwgInterVal < 0.005$时，$dAwgMeasDelay = dAwgInterVal - 0.005$。采样间隔 dSampTms 设置为 10μs，循环次数 nWaveCycle 设置为 1，适用于一般的同步触发点测量。
- 当 int nAwgSize＞0 时，起始地址为 0，终止地址为$nAwgSize - 1$。
- 当 nAwgSize＜0 时，起始地址为$-nAwgSize-1$，终止地址为 0。
- nSampNum 取值不同，存在如下差异：

条件	说明	示例
nSampNum !=0，启动 DIG 采样测量	若当前通道同时 AWG 输出时，存在如下情况： ● FVI2：|nAwgSize|最大为 4096，且不大于当前与同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：|nAwgSize|最大为 2048，且不大于当前与同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 loader 波形 500 点；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则|nAwgSize|应不大于 500（不大于当前与同步通道波形加载点数）。

条件	说明	示例
	若当前通道仅测量时，存在如下情况： ● FVI2：|nAwgSize|最大为 4096，且不大于同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：|nAwgSize|最大为 2048，且不大于同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 仅测量；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则|nAwgSize|应不大于 1000（不大于同步通道波形加载点数）。
nSampNum =0，仅 AWG 输出	属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间。 ● FVI2：|nAwgSize|最大为 4096。 ● FVI8/FVI16/FVI16Plus：单通道使用|nAwgSize|最大为 4096，4 通道同时使用|nAwgSize|最大为 1024。 ♦ 若当前通道|nAwgSize|∈[1025,2048]，后面 1 通道不能同时使用。 ♦ 若当前通道|nAwgSize|∈[2049,3072]，后面 2 通道不能同时使用。 ♦ 若当前通道|nAwgSize|∈[3073,4096]，只能单通道使用。 ♦ 通道顺序 ch0、ch1、ch2、ch3，环形结构。	

- 同步输出与测量的所有通道均需按照相同的参数进行 AwgSelect 设置，保证输出与测量同步性。

函数原型

```
int Select(const string& strPinName, double dAwgInterVal, int nSampNum, int nAwgSize);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。
dAwgInterVal	输出点之间的时间间隔，单位：ms，最小 10μs，最大 50ms。
nSampNum	每次输出后采样点的个数，设置为 0 时不启动采样测量，最大 200。
nAwgSize	Awg 长度。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().Select(strPinName, 0.2, 1, 100);
```

2.7.27 Select（）

函数功能

通过波形名字等信息来选择用于输出的波形。

AWG 每个输出点的时间间隔为 dAwgInterVal，在当前输出点的时间间隔内需要完成采样的过程才能进行下一个点的输出，因此，时间参数上要求如下：

- 对于 FVI2/8/16，$dAwgInterVal \geq dAwgMeasDelay + nSampNum * dSampTms$。
- 对于 FVI16Plus，$dAwgInterVal \geq dAwgMeasDelay + dSampTms * (nSampNum + 1)$且 $dAwgMeasDelay + dSampTms * 2 \geq 7\mu s$。

同步输出与测量的所有通道均需按照相同的参数进行 AwgSelect 设置，保证输出与测量同步性。

使用说明

nSampNum 取值不同，存在如下差异：

条件	说明	示例
nSampNum !=0，启动 DIG 采样测量	若当前通道同时 AWG 输出时，存在如下情况： ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096，且不大于当前与同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：(|nStopAddr-nStartAddr| +1)最大为 2048，且不大于当前与同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 loader 波形 500 点；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则(|nStopAddr-nStartAddr| +1)应不大于 500（不大于当前与同步通道波形加载点数）。
	若当前通道仅测量时，存在如下情况： ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096，且不大于同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：(|nStopAddr-nStartAddr| +1)最大为 2048，且不大于同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 仅测量；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则(|nStopAddr-nStartAddr| +1)应不大于 1000（不大于同步通道波形加载点数）。

条件	说明	示例
nSampNum =0，仅 AWG 输出	属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间。 ● FVI2：(|nStopAddr-nStartAddr| +1)最大为 4096。 ● FVI8/FVI16/FVI16Plus：单通道使用(|nStopAddr-nStartAddr| +1)最大为 4096；4 通道同时使用(|nStopAddr-nStartAddr| +1)最大为 1024。 ♦ 若当前通道(|nStopAddr-nStartAddr| +1)∈[1025,2048]，后面 1 通道不能同时使用。 ♦ (|nStopAddr-nStartAddr| +1)∈[2049,3072]，后面 2 通道不能同时使用。 ♦ (|nStopAddr-nStartAddr| +1)∈[3073,4096]，只能单通道使用。 ♦ 通道顺序 ch0、ch1、ch2、ch3，环形结构。	

函数原型

```
int Select(const string& strPinName, const string& strAwgName, double
dAwgInterVal, double dAwgMeasDelay, int nSampNum, double dSampTms, int
nStartAddr, int nStopAddr, int nWaveCycle);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
strAwgName	定义生产波形的名称，用于区别是否重新载入波形数据。
dAwgInterVal	AWG 输出点间隔，单位 ms。
dAwgMeasDelay	DA 输出后 AD 测量前的等待时间，单位 ms。
nSampNum	AD 采样个数。
dSampTms	AD 采样间隔，单位 ms，最大为 30ms。 FVI8/FVI16 最小 10μs，FVI2/FVI16Plus 最小 2μs。
nStartAddr	起始地址。
nStopAddr	终止地址。
nWaveCycle	循环次数。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
Const string strWaveName = " WAVE1";
TheInst.VIAWG().Select(strPinName, trWaveNam, 0.5, 0.1, 20, 0.01, 0, 99, 0);
```

2.7.28 Select（）

函数功能

通过波形名字等信息来选择用于输出的波形。

AWG 每个输出点的时间间隔为 dAwgInterVal，在当前输出点的时间间隔内需要完成采样的过程才能进行下一个点的输出，因此，时间参数上要求如下：

- 对于 FVI2/8/16，$dAwgInterVal \geq dAwgMeasDelay + nSampNum * dSampTms$。
- 对于 FVI16Plus，$dAwgInterVal \geq dAwgMeasDelay + dSampTms * (nSampNum + 1)$且 $dAwgMeasDelay + dSampTms * 2 >= 7\mu s$。

同步输出与测量的所有通道均需按照相同的参数进行 AwgSelect 设置，保证输出与测量同步性。

使用说明

- 当 int nAwgSize＞0 时，起始地址为 0，终止地址为$nAwgSize - 1$。
- 当 nAwgSize＜0 时，起始地址为$-nAwgSize-1$，终止地址为 0。
- nSampNum 取值不同，存在如下差异：

条件	说明	示例
nSampNum !=0，启动 DIG 采样测量	若当前通道同时 AWG 输出时，存在如下情况： ● FVI2：|nAwgSize|最大为 4096，且不大于当前与同步通道波形加载点数。 ● FVI8/FVI16/FVI16Plus：|nAwgSize|最大为 2048，且不大于当前与同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 loader 波形 500 点；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则|nAwgSize|应不大于 500（不大于当前与同步通道波形加载点数）。

条件	说明	示例
	若当前通道仅测量时，存在如下情况： ● FVI2：|nAwgSize|最大为 4096，且不大于同步通道波形加载点数； ● FVI8/FVI16/FVI16Plus：|nAwgSize|最大为 2048，且不大于同步通道波形加载点数。	FVI2 与 FVI16 同步。FVI2 仅测量；FVI16 loader 1000 点。 Select（）函数中 FVI2、FVI16 参数设置应相同，则|nAwgSize|应不大于 1000（不大于同步通道波形加载点数）。
nSampNum =0，仅 AWG 输出	属于相同 Slave FPGA 的通道共用 4096 波形数据存储空间。 ● FVI2：|nAwgSize|最大为 4096。 ● FVI8/FVI16/FVI16Plus：单通道使用|nAwgSize|最大为 4096，4 通道同时使用|nAwgSize|最大为 1024。 ♦ 若当前通道|nAwgSize|∈[1025,2048]，后面 1 通道不能同时使用。 ♦ 若当前通道|nAwgSize|∈[2049,3072]，后面 2 通道不能同时使用。 ♦ 若当前通道|nAwgSize|∈[3073,4096]，只能单通道使用。 ♦ 通道顺序 ch0、ch1、ch2、ch3，环形结构。	

- DA 输出后 AD 测量前的等待时间$dAwgMeasDelay = 0.5 * double\ dAwgInterVal$，但当$0.5 * double\ dAwgInterVal < 0.005$时，$dAwgMeasDelay = dAwgInterVal - 0.005$。
- 循环次数默认为 1，不可设置。
- AD 采样间隔默认为 0.01ms，不可设置。

函数原型

```
int Select(const string& strPinName, const string& strAwgName, double dAwgInterVal, int nSampNum, int nAwgSize);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
strAwgName	定义生产波形的名称，用于区别是否重新载入波形数据。
dAwgInterVal	AWG 输出点间隔，单位 ms。
nSampNum	AD 采样个数。
nAwgSize	波形点数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
const string strWaveName = " WAVE1";
TheInst.VIAWG().Select(strPinName, trWaveNam, 0.1 , 20, 100);
```

2.7.29 SetExtSync（）

函数功能

设置 FVI2、FVI8、FVI16 外部触发扫描使能。

使用说明

无。

函数原型

```
int SetExtSync(const string& strPinName, bool bIsSyncOn);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。
bIsSyncOn	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
Const string strPinName = "PIN1";
TheInst.VIAWG().SetExtSync(strPinName,false);
```

2.7.30 SetSync（）

函数功能

设置 FVI2、FVI8、FVI16 外部触发扫描使能。

使用说明

无。

函数原型

```
int SetSync(const string& strPinName, bool bIsSyncOn);
```

参数说明

参数	说明
strPinName	输出 Pin 的名字。
bIsSyncOn	是否保留此次通道配置信息，用于下一次的 AWG 输出，默认为否。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
string strPinName = "Pin1";
TheInst.VIAWG().SetSync(strPinName ,false);
```

2.7.31 SetTrigVal（）

函数功能

设置对应同步测量通道的触发值。

使用说明

无。

函数原型

```
int SetTrigVal(const string& strPinName, emVI_MeasTrigType type, double
dTrigVal, bool bPosOrNeg);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
type	触发模式，其类型定义详见 emVI_MeasTrigType 类型定义。
dTrigVal	触发值。
bPosOrNeg	设置上升沿或下降沿。 ● 0：上升沿。 ● 1：下降沿。

返回值

类型	说明
int	● 0：设置的上升沿触发。 ● -1：设置的下降沿触发。

示例

```
TheInst.CVIAWG().SetTrigVal(strPinName, emVI_Before_V , 5, 0);
```

2.7.32 SetDigMeasureType（）

函数功能

设置 DIG 测量类型

使用说明

仅 HPS100 支持。

函数原型

```
void SetDigMeasureType(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	单 Pin 或 PinGroup 的名称。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
void	返回为空。

示例

```
TheInst.CVIAWG().SetDigMeasureType();
```

2.8 CBIT128

2.8.1 GetAllSinglePinName（）

函数功能

获取 CBIT128 板卡所有的 Pin 的名字。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	CBIT128 板卡所有的 Pin 的名字。

示例

```
vector<string> strPinName;
TheInst.CBIT128().GetAllSinglePinName();
TheInst.CBIT128().RelayOn(strPinName);
```

2.8.2 RelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开。

使用说明

无。

函数原型

```
int RelayOn(const string& strPinList);
```

参数说明

参数	说明
strPinList	Pin Name 的列表。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string   strPinList;
TheInst.CBIT128().RelayOn("strPinList");
```

2.8.3 RelaySetOn（）

函数功能

设置所有 Site 需要闭合的用户继电器，未列出的用户继电器保持原来状态，未开启的 Site 通道用户继电器状态保持不变。

使用说明

无。

函数原型

```
int RelaySetOn(const string& strPinList);
```

参数说明

参数	说明
strPinList	Pin Name 的列表。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string  strPinList = "Pin1";
TheInst.CBIT128().RelaySetOn(strPinList);
```

2.8.4 RelaySetOff（）

函数功能

设置需要断开的用户继电器，未列出的用户继电器保持原来状态。其中未开启的 Site 通道用户继电器状态保持不变。

使用说明

无。

函数原型

```
int RelaySetOff(const string& strPinList);
```

参数说明

参数	说明
strPinList	Pin Name 的列表。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string  strPinList = "Pin1";
TheInst.CBIT128().RelaySetOff(strPinList);
```

2.8.5 SiteRelayOn（）

函数功能

设置所有 Site 需要闭合的用户继电器，未列出的用户继电器设置为断开，未开启的 Site 通道用户继电器状态保持不变。

使用说明

无。

函数原型

```
int SiteRelayOn(const string& strPinList);
```

参数说明

参数	说明
strPinList	Pin Name 的列表。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
string  strPinList;
TheInst.CBIT128().SiteRelayOn("strPinList");
```

2.8.6 SRelayOn（）

函数功能

设置需要闭合的用户继电器，对未列出的用户继电器设置为断开。

使用说明

与 CBIT128_SiteRelayOn（）函数格式完全相同，但意义不同，使用该函数，并行测试调整参数，便于使用物理通道控制继电器。

- CBIT128_SiteRelayOn（）函数控制的是继电器的逻辑通道。
- CBIT128_SRelayOn（）控制的是继电器的物理通道。

函数原型

```
int SRelayOn(const std::vector<int>& vec);
```

参数说明

参数	说明
vec	逻辑通道继电器编号列表，最大输入个数为 768 个，可无输入参数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
int Channel;
TheInst.CBIT128().SRelayOn(std::vector<int>{ Channel, Channel+2});
```

2.9 CTM2110_8

2.9.1 Get_R（）

函数功能

下位机将工位电阻测试结果返回给上位机。

使用说明

无。

函数原型

```
int GetR(double& dRes);
```

参数说明

参数	说明
dRes	电阻测试结果。

返回值

类型	说明
int	工位电阻测试结果。

示例

```
if(CTM2110_8_SITE_R() == 0)
   {
    if(CTM2110_8_GET_R(iData) == 0)
    {
     dResult = iData;
    }
    else
    {
     dResult = -1;
    }
```

```
}
```

2.9.2 GetResult_K（）

函数功能

获取 Kelvin 测试的结果。

使用说明

无。

函数原型

```
int GetResult_K(int &iResult)
```

参数说明

参数	说明
iResult	获取测试结果。

返回值

类型	说明
int	● 0：Kelvin Pass。 ● 255：Kelvin Fail。

示例

```
if(CTM2110_8_Kelvin() == 0)
{
   if(CTM2110_8_GetResult_K(iData) == 0)
    {
        dResult = iData;
    }
}
```

2.9.3 GetResult_O（）

函数功能

获取 OS 测试的结果。

使用说明

无。

函数原型

```
int GetResult_O(int &iResult1, int &iResult2, int &iResult3);
```

参数说明

参数	说明
iResult1	获取 DS-short 测试结果。
iResult2	获取 GS-short 测试结果。
iResult3	获取 DS-open 测试结果。

返回值

类型	说明
int	● 0：表示 Pass。 ● 1：表示 Fail。

示例

```
double dResult1 = -100,dResult2 = -100,dResult3 = -100;
 double iOpenlow=-1,iOpenHigh=-1;
  if(CTM2110_8_OS(0x00,iOpenlow,iOpenHigh) == 0)
    {
     if(CTM2110_8_GetResult_O(iData1,iData2,iData3) == 0)
     {
        dResult1 = iData1;
        dResult2 = iData2;
        dResult3 = iData3;
     }
}
```

2.9.4 GetResult_U（）

函数功能

获取雪崩测量结果。

使用说明

无。

函数原型

```
int GetResult_U(double &dData_Ids,double &dData_Vds,
     double &dData_MVDD, double &dData_MVG, double &dData_MVGR,
     short &dData_MT1,short &dData_MT2);
```

参数说明

参数	说明
dData_Ids	获取 dData_Ids 结果。
dData_Vds	获取 dData_Vds 结果。
dData_MVDD	获取 Data_MVDD 结果。
dData_MVG	获取 Data_MVG 结果。
dData_MVGR	获取 dData_MVGR 结果。
dData_MT1	获取 Data_MT1 结果。
dData_MT2	获取 Data_MT2 结果。

返回值

类型	说明
int	● dData_Ids：雪崩电流测回结果，单位 A。 ● dData_Vds：雪崩电压测回结果，单位 V。 ● dData_MVDD：漏极电压测回结果，单位 V。 ● dData_MVG：栅极正向开启电压测回结果，单位 V。 ● dData_MVGR：栅极负向关断电压测回结果，单位 V。 ● dData_MT1：电流上升时间结果，单位 μs。 ● dData_MT2：电流下降时间结果，单位 μs。

示例

```
if(CTM2110_8_UIS(dVDD,dVG,dVGR,dIDSCon,dVDSCon,dL,dVGM,dVGRM,0x01,dCTO) == 0)
{
if(CTM2110_8_GetResult_U(dIDS,dVDS,dMVDD,dMVG,dMVGR,dMT1,dMT2) != 0)
```

```
{
   dIDS = -1;
     dVDS = -2;
     dMVDD = -3;
     dMVG = -4;
     dMVGR = -5;
     dMT1 = -6;
     dMT2 = -7;
}
  else
  {
     LogResult(0,i-1,0,dIDS);
     LogResult(1,i-1,0,dVDS);
     LogResult(2,i-1,0,dMVDD);
     LogResult(3,i-1,0,dMVG);
     LogResult(4,i-1,0,dMVGR);
     LogResult(5,i-1,0,dMT1);
     LogResult(6,i-1,0,dMT2);
  }
}
```

2.9.5 GetResult_U（）

函数功能

获取雪崩测量结果。

使用说明

无。

函数原型

```
int GetResult_U(double &dData_Ids,double &dData_Vds,
     double &dData_MVDD, double &dData_MVG, double &dData_MVGR,
     short &dData_MT1,short &dData_MT2,short &dData_MT3,short &dData_MT4);
```

参数说明

参数	说明
dData_Ids	获取 dData_Ids 结果。
dData_Vds	获取 dData_Vds 结果。
dData_MVDD	获取 Data_MVDD 结果。
dData_MVG	获取 Data_MVG 结果。
dData_MVGR	获取 dData_MVGR 结果。
dData_MT1	获取 Data_MT1 结果。
dData_MT2	获取 Data_MT2 结果。
dData_MT3	获取 Data_MT3 结果。
dData_MT4	获取 Data_MT4 结果。

返回值

类型	说明
int	● dData_Ids：雪崩电流测回结果，单位 A。 ● dData_Vds：雪崩电压测回结果，单位 V。 ● dData_MVDD：漏极电压测回结果，单位 V。 ● dData_MVG：栅极正向开启电压测回结果，单位 V。 ● dData_MVGR：栅极负向关断电压测回结果，单位 V。 ● dData_MT1：电流上升时间结果，单位 μs。 ● dData_MT2：电流下降时间结果，单位 μs。 ● dData_MT3：电流上升时间结果，单位 μs。 ● dData_MT4：电流下降时间结果，单位 μs。

示例

```
if(CTM2110_8_UIS(dVDD,dVG,dVGR,dIDSCon,dVDSCon,dL,dVGM,dVGRM,0x01,dCTO) == 0)
{
if(CTM2110_8_GetResult_U(dIDS,dVDS,dMVDD,dMVG,dMVGR,dMT1,dMT2) != 0)
{
   dIDS = -1;
     dVDS = -2;
     dMVDD = -3;
```

```
    dMVG = -4;
    dMVGR = -5;
    dMT1 = -6;
    dMT2 = -7;
}
  else
  {
    LogResult(0,i-1,0,dIDS);
    LogResult(1,i-1,0,dVDS);
    LogResult(2,i-1,0,dMVDD);
    LogResult(3,i-1,0,dMVG);
    LogResult(4,i-1,0,dMVGR);
    LogResult(5,i-1,0,dMT1);
    LogResult(6,i-1,0,dMT2);
  }
}
```

2.9.6 Kelvin（）

函数功能

向下位机发送开始进行 Kelvin 测试的指令。

使用说明

无。

函数原型

```
int Kelvin()
```

参数说明

无。

返回值

类型	说明
int	Kelvin 测试的指令。

示例

```
if(CTM2110_8_Kelvin() == 0)
{
   if(CTM2110_8_GetResult_K(iData) == 0)
    {
```

```
        dResult = iData;
    }
}
```

2.9.7 NP（）

函数功能

设置 NMOS 或者 PMOS。

使用说明

无。

函数原型

```
int NP(char DataNP)
```

参数说明

参数	说明
DataNP	MOS 类型（0，1）。 ● 1 表示 NMOS。 ● 2 表示 PMOS。

返回值

类型	说明
int	设置 NMOS 或者 PMOS。

示例

```
CTM2110_8_NP(0x00);
```

2.9.8 OS（）

函数功能

向下位机发送开始进行 OS 测试的指令。

使用说明

无。

函数原型

```
int OS(char Mode,double dOpenlow, double dOpenHigh);
```

参数说明

参数	说明
Mode	OS 测试模式（0，1，2，3，4，5，6）。 ● 0 代表测试 DS-short、GS-short 和 DS-open。 ● 1 代表测试 DS-short。 ● 2 代表测试 GS-short。 ● 3 代表测试 DS-open。 ● 4 代表测试 DS-short、GS-short。 ● 5 代表测试 DS-short、DS-open。 ● 6 代表测试 GS-short、DS-open。
dOpenlow	DOpen 测试结果下限。
dOpenHigh	DOpen 测试结果上限。

返回值

类型	说明
int	OS 测试的指令。

示例

```
double dResult1 = -100,dResult2 = -100,dResult3 = -100;
 double iOpenlow=-1,iOpenHigh=-1;
  if(CTM2110_8_OS(0x00,iOpenlow,iOpenHigh) == 0)
    {
     if(CTM2110_8_GetResult_O(iData1,iData2,iData3) == 0)
     {
        dResult1 = iData1;
        dResult2 = iData2;
        dResult3 = iData3;
     }
}
```

2.9.9 OS_SetDOpen（）

函数功能

设置 DOpen 测试结果上下限、驱动电流值、加电流时间。

使用说明

无。

函数原型

```
int OS_SetDOpen(double dDOpenI = 10,double dDOpenDelay = 1)
```

参数说明

参数	说明
dDOpenI	DOpen 测试驱动电流，默认 10mA。
dDOpenDelay	DOpen 加电流时间，默认 1ms。

返回值

无。

示例

```
CTM2110_8_OS_SetDOpen(0, 6, 10, 1);
```

2.9.10 OS_SetDShort（）

函数功能

设置 DShort 测试的判断阈值、驱动电流值、加电流时间。

使用说明

无。

函数原型

```
int OS_SetDShort(double dDShortLimit = 5, double dDShortI = 1, double
dDShortDelay = 2,char TestRSel = 0x1)
```

参数说明

参数	说明
dDShortLimit	DShort 测试上限电压，默认 5V。
dDShortI	DShort 测试驱动电流，默认 1mA。
dDShortDelay	DShort 加电流时间，默认 2ms。
TestRSel	测试回路中是否接入 470R 电阻，默认 1，无需修改。

返回值

无。

示例

```
CTM2110_8_OS_SetDShort(5,1, 2, 0x1);
```

2.9.11 OS_SetGShort（）

函数功能

设置 GShort 测试的判断阈值、驱动电流值、加电流时间。

使用说明

无。

函数原型

```
int OS_SetGShort(double dGShortLimit = 5, double dGShortI = 0.099, double
dGShortDelay = 1,char TestRSel = 0x1)
```

参数说明

参数	说明
dGShortLimit	GShort 测试上限电压，默认 5V。
dGShortI	GShort 测试驱动电流，默认 0.099mA。
dGShortDelay	GShort 加电流时间，默认 1ms。
TestRSel	测试回路中是否接入 470R 电阻，默认 1，无需修改。

返回值

无。

示例

```
CTM2110_8_OS_SetGShort(5, 0.099, 1, 0x1);
```

2.9.12 RelayOn（）

函数功能

向下位机发送开始进行闭合工位继电器的指令。

使用说明

无。

函数原型

```
int RelayOn(int nSite)
```

参数说明

参数	说明
nSite	工位。

返回值

类型	说明
int	闭合工位继电器的指令。

示例

```
int i;
CTM2110_8_RelayOn(i);
```

2.9.13 RelayOff（）

函数功能

向下位机发送开始进行断开工位继电器的指令。

使用说明

无。

函数原型

```
int RelayOff(int nSite);
```

参数说明

参数	说明
nSite	工位。

返回值

类型	说明
int	断开工位继电器的指令。

示例

```
int i;
CTM2110_8_RelayOff(i)
```

2.9.14 RelayOnByIndex（）

函数功能

通过索引向下位机发送开始进行闭合工位继电器的指令。

使用说明

无。

函数原型

```
int RelayOnByIndex(int nSite);
```

参数说明

参数	说明
nSite	工位。

返回值

无。

示例

```
int i;
CTM2110_8_RelayOnByIndex(i);
```

2.9.15 RelayOffByIndex（）

函数功能

通过索引向下位机发送开始进行断开工位继电器的指令。

使用说明

无。

函数原型

```
int RelayOffByIndex(int nSite);
```

参数说明

参数	说明
nSite	工位。

返回值

无。

示例

```
int i;
CTM2110_8_RelayOffByIndex(i);
```

2.9.16 RPF（）

函数说明

向下位机发送开始进行 RPF 测试的指令。

使用说明

无。

函数原型

```
int RPF(double dVDD,double dVG,double dVGR,double dVDS,double dL, double dBIDS,
double dEIDS, double dDeI, double dDeT, double dCTO, int nki);
```

参数说明

参数	说明
dVDD	漏极电压。
dVG	正向开启电压。
dVGR	栅极关断电压。
dVDS	雪崩电流设定值。
dL	电感值。
dBIDS	雪崩电流开始值。
dEIDS	雪崩电流结束值。
dDeI	步进电流值。
dDeT	每次测试等待时间。
dCTO	雪崩超时时间。
nki	测试工位。

返回值

无。

示例

```
double dIDS = -1, dVDS = -1, dMVDD = -1, dMVG = -1, dMVGR = -1, dEAS = -1;
 short dMT1 = -1, dMT2 = -1;
 double test_ids;
 double dVDD, dVG, dVGR, dIDSCon_BIDS, dIDSCon_EIDS, dID_STEP, dVDSCon, dL,
dVGM, dVGRM, dCTO, delay_ms;

while (test_ids <= dIDSCon_EIDS)
   {
    if (CTM2120_UIS(dVDD, dVG, dVGR, test_ids, dVDSCon, dL, dVGM, dVGRM, 0x01,
dCTO) == 0)
```

```
    {
     CTM2120_GetResult_U(dIDS, dVDS, dMVDD, dMVG, dMVGR, dMT1, dMT2);
    }
   }
```

2.9.17 SetWaitTime（）

函数功能

向下位机发送等待延时。

使用说明

无。

函数原型

```
int SetWaitTime(int nTime);
```

参数说明

参数	说明
nTime	等待延时。

返回值

无。

示例

```
int nWTime;
CTM2110_8_SetWaitTime(nWTime);
```

2.9.18 Site_R（）

函数功能

向下位机发送测试工位电阻的指令。

使用说明

无。

函数原型

```
int Site_R();
```

参数说明

无。

返回值

类型	说明
int	● 0：成功。 ● -1：发送失败。 ● -2：重新连接失败。

示例

```
TheInst.CTM2110_8().Site_R();
double dResult = 0.0;
TheInst.CTM2110_8().Get_R(dResult);
```

2.9.19 UIS（）

函数功能

向下位机发送开始进行 UIS 测试的指令。

使用说明

无。

函数原型

```
int UIS(double dVDD, double dVG, double dVGR, double dIDS, double dVDS, double dL, double dVGM, double dVGRM, char chMode, double dCTO);
```

参数说明

参数	说明
dVDD	漏极电压。
dVG	正向开启电压。
dVGR	栅极关断电压。
dIDS	雪崩电流设定值。
dVDS	栅极检测电压上限。
dL	电感值。
dVGM	栅极检测电压上限。
dVGRM	栅极检测电压下限。
chMode	数据模式（1，2）。 ● 1 代表发送雪崩数据（UIS，默认为 1。 ● 2 代表雪崩波形。
dCTO	雪崩超时时间。

返回值

无。

示例

```
if(CTM2110_8_UIS(dVDD,dVG,dVGR,dIDSCon,dVDSCon,dL,dVGM,dVGRM,0x01,dCTO) == 0)
{
if(CTM2110_8_GetResult_U(dIDS,dVDS,dMVDD,dMVG,dMVGR,dMT1,dMT2) != 0)
{
   dIDS = -1;
     dVDS = -2;
     dMVDD = -3;
     dMVG = -4;
     dMVGR = -5;
     dMT1 = -6;
     dMT2 = -7;
}
  else
  {
```

```
    LogResult(0,i-1,0,dIDS);
    LogResult(1,i-1,0,dVDS);
    LogResult(2,i-1,0,dMVDD);
    LogResult(3,i-1,0,dMVG);
    LogResult(4,i-1,0,dMVGR);
    LogResult(5,i-1,0,dMT1);
    LogResult(6,i-1,0,dMT2);
  }
}
```

2.9.20 GetARMVer（）

函数功能

获取 CTMARM 版本号。

使用说明

无。

函数原型

```
int GetARMVer(char * szVer, int nBufLen);
```

参数说明

参数	说明
szVer	获取的版本号字符串。
nBufLen	缓存区大小。

返回值

类型	说明
int	● 0：成功。 ● 非 0：失败。

示例

```
Char* szVer = nullptr;
int nBufLen = 128;
CTM2110_8_GetARMVer(szVer, nBufLen);
```

2.9.21 GetARMVer（）

函数功能

获取 CTMARM 版本号。

使用说明

无。

函数原型

```
int GetARMVer(int &nVer);
```

参数说明

参数	说明
nVer	获取的版本号整数值

返回值

类型	说明
int	● 0：成功 ● 非 0：失败

示例

```
int nVer = 0;
CTM2110_8_GetARMVer(nVer);
```

2.10 TIF

2.10.1 AwgSetExtSync（）

2.10.1.1 类型定义

TIF_SyncMode 类型定义

类型	Enumeration	
描述	边沿敏感模式（0～3）。	
元素	TIF_Sync_Direct（0）	直通模式。

	TIF_Sync_Rise（1）	上升沿触发。
	TIF_Sync_Drop（2）	下降沿触发。
	TIF_Sync_RiseDrop（3）	上升&下降沿触发。
头文件	UserDef.h	

2.10.1.2 函数说明

函数功能

设置外部触发功能是否启用，配置外部触发信号的消抖、沿敏感模式。

使用说明

只有使能 TIF 的外部触发功能，同时相应 VI 源板卡也设置为外部同步触发使能。此时由外部激励发出外部触发信号，才会启动 AD 或 AWG 通道的 DA。

函数原型

```
int AwgSetExtSync(int nPhyCh, bool bIsOn, int nFilterNum = 1, TIF_SyncMode emSyncMode = TIF_Sync_Direct);
```

参数说明

参数	说明
nPhyCh	TIF 板卡序号，0。
bIsOn	外部同步触发输入通道是否开启。 ● 1：开启。 ● 0：关闭。
nFilterNum	对外部同步触发信号的毛刺消抖系数（1～255），信号宽度在 20nS×nFilterNum 以上时，才能被认为是有效输入信号，其他信号会被当做毛刺处理。
syncMode	边沿敏感模式，其类型定义详见 TIF_SyncMode 类型定义。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().AwgSetExtSync(0, 1, 5, TIF_Sync_Rise);//外部同步触发功能开启,滤波系数100ns,上升沿有效。
```

2.10.2 CHKBConnect（）

函数功能

CHK_HFB、CHK_LFB 接入 TIF 内部测量模块。

使用说明

无。

函数原型

```
int CHKBConnect(int nPhyCh, emConnState emConnect);
```

参数说明

参数	说明
nPhyCh	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。
emConnect	连接状态，其类型定义详见 emConnState 类型定义。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().CHKBConnect(0, Conn_ON);//TIF 板卡上 CHK_HFB、CHK_LFB 接入 TIF 内部负载模块,闭合状态使用结束后需要断开。
```

2.10.3 Init（）

函数功能

TIF 模块初始化。

使用说明

无。

函数原型

```
int Init(int nChannel);
```

参数说明

参数	说明
nChannel	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().Init(0);
```

2.10.4 I2CConfig（）

函数功能

配置 I2C 总线输出时的频率。

使用说明

该函数在 I2CWrite（）函数前使用。

函数原型

```
int I2CConfig(const unsigned int uFreq);
```

参数说明

参数	说明
uFreq	总线频率，设置范围（100～1000），单位为 kHz。

返回值

类型	说明
int	● 0：正常 ● - 1：总线频率设置范围异常。

示例

```
unsigned int Freq = 400;
TheInst.TIF().I2CConfig(Freq);
```

2.10.5 I2CWrite（）

函数功能

通过 I2C 总线往从机写入单个 8bits 数据。

使用说明

使用该函数前需调用 I2CConfig（）函数设置频率。

函数原型

```
int I2CWrite(const unsigned int uRegAddrLen, const unsigned int uslaveAddr,
const unsigned int uRegAddr, const int nWrData);
```

参数说明

参数	说明
uRegAddrLen	寄存器地址字节数，可设范围（0 代表无无地址、1 代表地址长度为 1 个字节、2 代表地址长度为 2 个字节）。
uslaveAddr	从机地址，最后 1bit 为读写位。
uRegAddr	寄存器地址，当 nRegAddrLen=0 时，此参数无效。
nWrData	8bits 数据。

返回值

类型	说明
int	● 0：正常 ● - 1：总参数设置异常。

示例

```
unsigned int Freq = 400;
TheInst.TIF().I2CConfig(Freq);
TheInst.TIF().I2CWrite((1,0x40,0x20,0x11);
```

2.10.6 I2CWrite（）

函数功能

通过 I2C 总线往从机写入 1 个或多个 8bits 数据。

使用说明

无。

函数原型

```
int I2CWrite(const unsigned int uRegAddrLen,
const unsigned int uslaveAddr,
const unsigned int uRegAddr,
const unsigned int uDataLen,
const int* pWrData);
```

参数说明

参数	说明
uRegAddrLen	寄存器地址字节数，可设范围（0 代表无无地址、1 代表地址长度为 1 个字节、2 代表地址长度为 2 个字节）。
uslaveAddr	从机地址，最后 1bit 为读写位。
uRegAddr	寄存器地址，当 nRegAddrLen=0 时，此参数无效。
uDataLen	要写入的 8bits 数据的个数，可设范围（1～8）。
pWrData	数据指针，按顺序存放要写入的数据。

返回值

类型	说明
int	● 0：正常 ● - 1：总参数设置异常。

示例

```
    int WriteData[8];
    WriteData[0]=0x11;
    WriteData[1]=0x22;
    WriteData[2]=0x33;
    WriteData[3]=0x44;
    WriteData[4]=0x55;
    WriteData[5]=0x66;
    WriteData[6]=0x77;
    WriteData[7]=0x88;
TheInst.TIF().I2CWrite(2,0x40, 0x0220,4,WriteData);
```

2.10.7 I2CRead（）

函数功能

通过 I2C 总线从从机读回 1 个或多个 8bits 数据。

使用说明

无。

函数原型

```
int I2CRead(const unsigned int uRegAddrLen,
const unsigned int uslaveAddr,
const unsigned int uRegAddr,
 const unsigned int uDataLen,
 int* pRdData);
```

参数说明

参数	说明
uRegAddrLen	寄存器地址字节数，可设范围（0 代表无地址、1 代表地址长度为 1 个字节、2 代表地址长度为 2 个字节）。
uslaveAddr	从机地址，最后 1bit 为读写位。
uRegAddr	寄存器地址，当 nRegAddrLen=0 时，此参数无效。
uDataLen	要写入的 8bits 数据的个数，可设范围（1～8）。
pRdData	数据指针，按顺序存放读回的数据。

返回值

类型	说明
int	● 0：正常 ● - 1：总参数设置异常。

示例

```
int ReadData[8];
TheInst.TIF().I2CRead(2,0x40, 0x0220, 4, ReadData);
```

2.10.8 GetAdcVal（）

2.10.8.1 类型定义

TIF_ADCMode 类型定义

类型	Enumeration	
描述	设置 ADC 的采样模式。	
元素	TIF_ADC_MV = 1	MV 测量电压。
	TIF_ADC_MI = 2	MI 测量电流。
	TIF_ADC_MT_IN = 3	AD 内部温度。
	TIF_ADC_MT_OUT = 4	AD 外部温度。
头文件	UserDef.h	

2.10.8.2 函数说明

函数功能

获取 TIF 上 ADC 的采样值，返回 double 类型数据。

使用说明

无。

函数原型

```
double GetAdcVal(int nPhyCh, TIF_ADCMode emMode);
```

参数说明

参数	说明
nPhyCh	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。
emMode	设置 ADC 的采样模式，其类型定义详见 TIF_ADCMode 类型定义。

返回值

类型	说明
double	返回值为 ADC 的测量结果。

示例

```
double dAdcVal = TheInst.TIF().GetAdcVal(0, TIF_ADC_MV);//获取 ADC 采样电压值。
```

2.10.9 GetMasterFPGAVer（）

函数功能

获取 TIF 板卡主机 FPGA 版本号，该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

使用说明

无。

函数原型

```
int Get_Master_FPGAVer(int nTIFn, int &nFpgaVer, char* strFpgaVer);
```

参数说明

参数	说明
nTIFn	板卡号（0）。
nFpgaVer	返回主机 FPGA 版本号的变量，int 型。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
int nFpgaVer;
char strFpgaVer[256]={'\0'};
TheInst.TIF().Get_Master_FPGAVer(0,nFpgaVer,strFpgaVer);
```

2.10.10 GetVer（）

函数功能

获取 TIF 板卡的模块版本信息。

使用说明

无。

函数原型

```
int GetVer(char* chVer, int nTIFn);
```

参数说明

参数	说明
chVer	返回版本信息的变量指针。
nTIFn	板卡号（0）。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
char chVer [256]={'\0'};
Ver = TheInst.TIF().GetVer(chVer, 0);
```

2.10.11 SetAdcMode（）

函数功能

设置 TIF 上 32-Bit_ADC 的参数。

使用说明

无。

函数原型

```
int SetAdcMode(int nPhyCh, int nGain, int nDataRate, TIF_FilterMode
emFilterMode);
```

参数说明

参数	说明
nPhyCh	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。
nGain	● 0：1V/V。 ● 1：2V/V。 ● 2：4V/V。 ● 3：8V/V。 ● 4：16V/V。 ● 5：32V/V。 ● OTHER：BYPASS。

参数	说明
nDataRate	● 0：2.5SPS。 ● 1：5SPS。 ● 2：10SPS。 ● 3：16.6SPS。 ● 4：20SPS（default）。 ● 5：50SPS。 ● 6：60SPS。 ● 7：100SPS。 ● 8：400SPS。 ● 9：1200SPS。 ● 10：2400SPS。 ● 11：4800SPS。 ● 12：7200SPS。 ● 13：14400SPS。 ● 14：19200SPS。 ● 15：38400SPS。
emFilterMode	● 0（TIF_Filter_Sinc1）：Sinc1。 ● 1（TIF_Filter_Sinc2）：Sinc2。 ● 2（TIF_Filter_Sinc3）：Sinc3。 ● 3（TIF_Filter_Sinc4）：Sinc4。 ● 4（TIF_Filter_FIR）：FIR（default）。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().SetAdcMode(0, 1, 1, TIF_Filter_Sinc1);
```

2.10.12 SetCalMode（）

2.10.12.1 类型定义

TIF_CalMode 类型定义

类型	Enumeration

描述	外接源表的连接方式。	
元素	TIF_Cal_OFF = 0	全断开。
	TIF_Cal_CV = 1	连接电压表。
	TIF_Cal_CI = 2	连接电流表。
	TIF_Cal_CF = 3	连接频率计。
头文件	UserDef.h	

2.10.12.2 函数说明

函数功能

设置校准校验，切换电阻，切换测量模式。

使用说明

无。

函数原型

```
int SetCalMode(int nPhyCh, int nResNum, TIF_CalMode emCalMode);
```

参数说明

参数	说明
nPhyCh	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。
nResNum	设置所切换的负载大小。 ● 0：全断开。 ● 1：0.1R。 ● 2：1R。 ● 3：10R。 ● 4：100R。 ● 5：1K。 ● 6：10K。 ● 7：100K。 ● 8：1M。 ● 9：10M。 ● 10：100M。 ● 11：无穷。
emCalMode	设置外接源表的连接方式，其类型定义详见 TIF_CalMode 类型定义。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().SetCalMode(0,5, TIF_Cal_CV);//选择 0 通道和 1K 电阻负载,以测量电压的方式连接。
```

2.10.13 SetMeasRange（）

函数功能

设置 TIF 上 32Bit_ADC 的量程。

使用说明

无。

函数原型

```
int SetMeasRange(int nPhyCh, int nMeasRange);
```

参数说明

参数	说明
nPhyCh	测试通道号（目前统一设置为 0），一个 TIF 资源板卡只有 1 个通道。
nMeasRange	● 0：断开。 ● 1：1V。 ● 2：10V。 ● 3：100V。 ● 4：1000V。 ● 5：2000V。 ● 6：内温。 ● 7：外温。

返回值

类型	说明
int	返回值为 0，无意义。

示例

```
TheInst.TIF().SetMeasRange(0,0);
```

2.11 TMU8

2.11.1 CheckConnect（）

2.11.1.1 类型定义

emConnState 类型定义

类型	Enumeration	
描述	断开或闭合继电器。	
元素	Conn_OFF = 0	断开。
	Conn_ON = 1	闭合。
头文件	UserDef.h	

2.11.1.2 函数说明

函数功能

对指定工位且指定自检继电器进行控制。

使用说明

无。

函数原型

```
int CheckConnect(const string& strPinName, emConnState isOn, int site);
```

参数说明

参数	说明
strPinName	单 Pin 的名称。
isOn	是否闭合 TMU8 自检总线继电器。emConnState 详细定义请参见 emConnState 类型定义。
site	要操作的工位号。

返回值

类型	说明
int	● 0：正常 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().CheckConnect(strPinName, Conn_OFF,0);
```

2.11.2 Init（）

函数功能

TMU8 通道初始化，断开通道输入继电器。

使用说明

无。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	单 Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().Init(strPinName);
```

2.11.3 GetAllSinglePinName（）

函数功能

获取 TMU8 板卡的模块版本信息。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	所有的 Pin 的名字。

示例

```
vector<string> strPinName;
strPinName = TheInst.TMU8().GetAllSinglePinName();
```

2.11.4 Get_Master_FPGAVer（）

函数功能

获取 TMU8 板卡主机 FPGA 版本号，该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

使用说明

无。

函数原型

```
int Get_Master_FPGAVer(int nTMU8n, int &nFpgaVer, char *strFpgaVer);
```

参数说明

参数	说明
nTMU8n	板卡号（0～1）。
nFpgaVer	返回主机 FPGA 版本号的变量。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
int nFpgaVer;

char strFpgaVer[256]={'\0'};        //注：字符串长度需≥7

TheInst.TMU8().Get_Master_FPGAVer(0,nFpgaVer, strFpgaVer);
```

2.11.5 GetPinName（）

函数功能

获取 TMU8 板卡的模块版本信息。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh);
```

参数说明

参数	说明
nPhyCh	通道号。

返回值

类型	说明
string	Pin 的名字。

示例

```
string strPinName = TheInst.TMU8().GetPinName(0);
```

2.11.6 GetVer（）

函数功能

获取 TMU8 板卡的模块版本信息。

使用说明

无。

函数原型

```
int GetVer(char*pchVer, int nTMU8n);
```

参数说明

参数	说明
pchVer	返回版本信息的变量指针。
nTMU8n	板卡号（0～1）。

返回值

类型	说明
int	返回值无意义。

示例

```
char chVer [256]={'\0'};
TheInst.TMU8().GetVer(chVer, 0);
```

2.11.7 Measure（）

函数功能

获取通道测量值。

- nMode=0 时返回实测频率（kHz）,其余模式时返回时间（ms）。实测值放在 g_pSite–> dRealData[i]中，i 与实际工作的 nSite 相对应。
- 多 nSite 并测，并测工作将自动完成，并放到对应的 dRealData 中。

使用说明

该函数不适用于精测模式下测量单个波形参数，可以用于测量连续波形，精测模式下为实现更高的测试效率，不推荐使用该函数，推荐使用 TMU_TDC_Init、TMU_TDC_Measure 函数。

函数原型

```
int Measure(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().SetMode(strPinName,TMU8_Cycle,
TMU8_Impedance_1MOhm,TMU8_VRange_10V,TMU8_Filter_10ns,5.0,5.0,0.1);
TheSoft.TestProc().DelaymS(3);
SignalInput;//信号输入。
TheInst.TMU8().Measure(strPinName);
```

2.11.8 MeasureAver（）

函数功能

在设定模式下，nSampleTms 时间内，测试 nSampleN 次，求平均值并放到对应的 dRealData 中。

- nMode0：频率。
- nMode1：周期模式下。
- nMode2：高电平。
- nMode3：低电平。

使用说明

只适用于低精度测量。

函数原型

```
int MeasureAver(const string& strPinName, double dSampleTms, int nSampleN);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dSampleTms	采样等待时间（ms）。
nSampleN	采样次数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().MeasureAver(strPinName,0.5,10);
```

2.11.9 MeasureWithTms（）

函数功能

在限定时间内，获取通道测量值，当 nMode=0 时返回实测频率（KHz），其余模式时返回时间（ms）。

使用说明

无。

函数原型

```
int MeasureWithTms(const string& strPinName, double dSampleTms = 10.0);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dSampleTms	限时测量时间，单位为 ms，缺省默认设置为 10ms。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
TheInst.TMU8().MeasureWithTms(strPinName, 5);//限时 5ms 返回测量结果
```

2.11.10 SetMode（）

2.11.10.1 类型定义

TMU8_Mode 类型定义

类型	Enumeration	
描述	测量模式。	
元素	TMU8_Frequency=0	测量频率（KHZ），触发与 TGV1 有关，用上升沿测量。

	TMU8_Cycle=1	测量周期（ms），触发与 TGV1 有关，用上升沿测量。
	TMU8_HighLevel=2	电平宽度（ms），触发与 TGV1 有关。
	TMU8_LowLevel=3	电平宽度（ms），触发与 TGV1 有关。
	TMU8_Rise=4	升沿时间（ms），触发与 TGV1 —>TGV2 有关，设置时 TGV1<TGV2。
	TMU8_Drop=5	降沿时间（ms），触发与 TGV1 —> TGV2 有关，设置时 TGV1>TGV2。通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
	TMU8_RiseToRise=6	离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个上升沿 PP。
	TMU8_RiseToDrop=7	离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个下降沿 PN。
	TPS8_DropToRise=8	距离 Stop 第一个下降沿最近 Start 下降沿到 Stop 第一个上升沿 NP。
	TPS8_DropToDrop=9	距离 Stop 第一个下降沿最近 Start 下降沿到 Stop 第一个下降沿 NP。
	TMU8_DutyCycle=11	测量占空比。
	TMU8_RiseToRise_FF=16	Start 第一个上升沿到 Stop 第一个上升沿 PP。
	TMU8_RiseToDrop_FF=17	Start 第一个上升沿到 Stop 第一个下降沿 PN。
	TMU8_DropToRise_FF=18	Start 第一个下降沿到 Stop 第一个上升沿 NP。
	TMU8_DropToDrop_FF=19	Start 第一个下降沿到 Stop 第一个下降沿 NN。
	TMU8_Frequency_Neg=20	测量频率（kHz），触发与 TGV1 有关，用下降沿测量。
	TMU8_Cycle_Neg=21	测量周期（ms），触发与 TGV1 有关，用下降沿测量。
头文件	UserDef.h	

TMU8_VRang 类型定义

类型	Enumeration	
描述	输入电压类型。	
元素	TMU8_VRange_10V=0	低输入电压，<=10V。
	TMU8_VRange_50V=1	高输入电压，>10V。
	TMU8_VRange_2V=2	高速测量时，输入电压<2.0V 时选择该参数，该挡位仅适用于精测模式。

头文件	UserDef.h

2.11.10.2 函数说明

函数功能

设置 TMU8 通道的测量模式。

使用说明

调用 TMU8_SetMode 函数后，为了使源的内部达到稳定状态，需要至少延时 3ms 再执行其他操作。

函数原型

```
int SetMode(const string& strPinName, TMU8_Mode emMode, TMU8_Impedance isHImped, TMU8_VRang emRang, TMU8_Filter emFilterEn, double dTGV1, double dTGV2, double dValLimit);
```

参数说明

参数	说明
strPinName	通道号（0～15）。
emMode	测量模式（0、1、2、3、4、5、6、7、8、9、11、16、17、18、19、20、21）。详细说明请参见 TMU8_Mode 类型定义。
isHImped	输入阻抗匹配。 取值范围： ● 0（TMU8_Impedance_50Ohm）：50Ω，低输入阻抗。 ● 1（TMU8_Impedance_1MOhm）：1MΩ，高输入阻抗。 通常选 1。
emRang	输入电压类型，其类型定义详见 TMU8_VRang 类型定义。

参数	说明
emFilterEn	选择触发释抑时间。 低速测量时，设置范围：0～65535，释抑时间为（所设参数*10）ns。 ● 低速测试： ♦ 当信号的压摆率（slewrare）大于 1V/μs 时，该参数可以设置为 0。 ♦ 当信号的压摆率小于 1V/μs 时，该参数不小于 10。 ♦ 当信号的压摆率小于 0.01V/μs 时，该参数不小于 1000。 ● 高速测量时，测量模式设置为 0。 枚举说明： TMU8_Filter_0ns、TMU8_Filter_1ns、TMU8_Filter_20ns、TMU8_Filter_50ns、TMU8_Filter_100ns、TMU8_Filter_200ns、TMU8_Filter_500ns、TMU8_Filter_1us、TMU8_Filter_2us、TMU8_Filter_5us、TMU8_Filter_10us、TMU8_Filter_20us、TMU8_Filter_50us、TMU8_Filter_100us、TMU8_Filter_200us、TMU8_Filter_500us。
dTGV1	设置触发电压。 ● （–10–+10）V（IsHV=0）。 ● （–50–+50）V（IsHV=1）。
dTGV2	设置触发电压，未使用时，建议与 TGV1 相同。 ● （–10–+10）V（IsHV=0）。 ● （–50–+50）V（IsHV=1）。
dValLimit	设置测量范围，默认选择低速模式，该值一般要设置成小于实际频率值或大于被测信号的最长时间。基于常规应用，频率模式下，高速测量，该参数可直接设置为 10000，低速测量时，该参数设置为 1。其它测试模式下，高速测量模式设置为 0.0001，低速模式设置为 1。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().SetMode(strPinName, TMU8_Cycle, TMU8_Impedance_1MOhm,
TMU8_VRange_10V, TMU8_Filter_10ns, 5.0, 5.0, 0.1);//1 通道测量周期,低压粗测,触发电压
5V,信号约 0.1ms 左右。
```

2.11.11 StartNumber（）

函数功能

设置 Start 后第 Number 个周期的时间参数。注意该函数只适用于低精度测量，从 0 开始计算波形。

使用说明

该函数只适用于低精度测量，从 0 开始计算波形。

函数原型

```
int StartNumber(const string& strPinName, uint32_t uNumber)
```

参数说明

参数	说明
strPinName	Pin 的名字
uNumber	周期数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().SetMode(strPinName,TMU8_Cycle,TMU8_Impedance_1MOhm,TMU8_VRange_10
V,TMU8_Filter_10ns,5.0,5.0,0.1);
TheSoft.TestProc().DelaymS(3);
TheInst.TMU8().StartNumber(strPinName,1);

SignalInput;//信号输入。

TheInst.TMU8().Measure(strPinName);
```

2.11.12 Start（）

函数功能

开始启动测量。

使用说明

该函数只适用于低精度测量，从 0 开始计算波形。

函数原型

```
int Start(const string& strpPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().Start(strPinName);
```

2.11.13 TDC_Init（）

函数功能

可用于精测测量模式下测量单个波形参数，也可以测量连续波形，待测时间长度一般不大于 1μs。

使用说明

- 此函数需要放在波形到来之前。
- 该函数只适用于高精度测量。

函数原型

```
int TDC_Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().TDC_Init(strPinName);
```

2.11.14 TDC_Measure（）

函数功能

用于返回精测测量模式测量结果,和 TMU8_TDC_Init 配合使用，待测时间长度一般不大于 1μs。

使用说明

需要放在单个波形到来之后，只适用于高精度测量，测量单个或连续波形，实现更高的测试效率。

函数原型

```
int TDC_Measure(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TMU8().SetMode(strPinName,TMU8_Cycle,TMU8_Impedance_1Mohm,TMU8_VRange_10
V,TMU8_Filter_10ns,5.0,5.0,0.0001);
TheSoft.TestProc().DelaymS(3);
TheInst.TMU8().TDC_Init(strPinName);

SignalInput;//信号输入。

TheInst.TMU8().TDC_Measure(strPinName);
```

2.12 DIO

2.12.1 GetAllSinglePinName（）

函数功能

获取 DIO64 板卡所有的 Pin 的名称。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

类型	说明
vector	DIO6Plus 板卡所有 Pin 的名称。

示例

```
vector<string> strPinName;
strPinName = TheInst.DIO().GetAllSinglePinName();
```

2.12.2 GetTemperature（）

2.12.2.1 类型定义

DIOArea_E 类型定义

类型	Enumeration	
描述	温度区域。	
元素	E_FE_PE_ZONE=0	FE 中 PE 芯片区域温度。
	E_FE_FPGA	FE 中 FPGA 芯片区域温度。
	E_BE_ZONE	BE 板卡温度。
	E_BE_FPGA	BE 中 FPGA 芯片区域温度。
头文件	UserDef.h	

2.12.2.2 函数说明

函数功能

获取 DIO 板卡区域温度。

使用说明

无。

函数原型

```
double GetTemperature(int nBoardNum, DIOArea_E emArea);
```

参数说明

参数	说明
nBoardNum	板卡号，0～1。
emArea	温度区域，其类型定义详见 DIOArea_E 类型定义。

返回值

类型	说明
double	返回 DIO 板卡区域温度。

示例

```
uint32_t uiSlot = 0;
```

```
double dTemperature = TheInst.DIO().GetTemperature(uiSlot, E_FE_PE_ZONE);
cout << "Get Slot " << uiSlot << " E_FE_PE_ZONE Temperature:" << dTemperature <<
endl;
dTemperature = TheInst.DIO().GetTemperature(uiSlot, E_FE_FPGA);
cout << "Get Slot " << uiSlot << " E_FE_FPGA Temperature:" << dTemperature <<
endl;
dTemperature = TheInst.DIO().GetTemperature(uiSlot, E_BE_ZONE);
cout << "Get Slot " << uiSlot << " E_BE_ZONE Temperature:" << dTemperature <<
endl;
dTemperature = TheInst.DIO().GetTemperature(uiSlot, E_BE_FPGA);
cout << "Get Slot " << uiSlot << " E_BE_FPGA Temperature:" << dTemperature <<
endl;
```

2.12.3 Level

2.12.3.1 类型定义

LevelDrive_E 类型定义

类型	Enumeration	
描述	驱动模式。	
元素	E_DRIVE_HIGH=0	高电平。
	E_DRIVE_LOW	低电平。
	E_DRIVE_HIZ	高阻态。
	E_DRIVE_NONE	/
头文件	UserDef.h	

LevelTerm_E 类型定义

类型	Enumeration	
描述	端接模式。	
元素	E_TERM_OFF = 0	无端接模式。
	E_TERM_TERM	端接阈值电压模式。
	E_TERM_LOAD	动态负载模式。
	E_TERM_NONE	/
头文件	UserDef.h	

2.12.3.2 Apply（）

函数功能

参数配置下发。

使用说明

无。

函数原型

```
void Apply();
```

参数说明

无。

返回值

无。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).Apply();
```

2.12.3.3 CancelVhhPin（）

函数功能

取消高压模式。

使用说明

无。

函数原型

```
CDIOLevel_Pin& CancelVhhPin(string strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().CancelVhhPin(strPinName);
```

2.12.3.4 ControlDriverMode（）

函数功能

控制通道驱动模式。

使用说明

该函数立即生效，无需调用 Apply 函数。

函数原型

```
CDIOLevel_Pin& ControlDriverMode(LevelDrive_E enDriveMode, double dVinValue);
```

参数说明

参数	说明
enDriveMode	驱动模式，其类型定义详见 LevelDrive_E 类型定义。
dVinValue	电压值。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string SigName7 = "DIO_P007";
TheInst.DIO().Level().Pins(SigName7).ControlDriverMode(E_DRIVE_HIGH, 4);
```

2.12.3.5 ControlDriverMode（）

函数功能

控制通道驱动模式，针对板卡实际物理通道。

使用说明

该函数立即生效，无需调用 Apply 函数。

函数原型

```
void ControlDriverMode(const string& strChannel, LevelDrive_E enDriveMode,
double dVinValue);
```

参数说明

参数	说明
strChannel	板卡物理通道名称，格式为“DIOx.x”。第一个 x 表示板卡槽位号，范围（1～2），第二个 x 表示通道号，范围（0～63）。
enDriveMode	驱动模式，其类型定义详见 LevelDrive_E 类型定义。
dVinValue	电压值，单位：V。

返回值

无。

示例

```
TheInst.DIO().Level().ControlDriverMode("DIO1.1", E_DRIVE_LOW, 1);
```

2.12.3.6 GetDriverMode（）

函数功能

获取通道驱动模式。

使用说明

无。

函数原型

```
map<string, map<uint32_t, LevelDrive_E>> GetDriverMode();
```

参数说明

无。

返回值

参数	说明
map	对应 Pin 对应 Site 的通道的驱动模式。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, LevelDrive_E>> mapPin_Site_VtData;
mapPin_Site_VtData = TheInst.DIO().Level().Pins(strPinName).GetDriverMode();
```

2.12.3.7 GetTerminationMode（）

函数功能

获取通道端接模式。

使用说明

无。

函数原型

```
map<string, map<uint32_t, LevelTerm_E>> GetTerminationMode();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的通道的端接模式。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, LevelTerm_E>> mapPin_Site_VtData;
mapPin_Site_VtData =
TheInst.DIO().Level().Pins(strPinName).GetTerminationMode();
```

2.12.3.8 GetVih（）

函数功能

获取输入的电压上限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVih();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的输入电压上限，单位：V。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_VihData;
mapPin_Site_VihData = TheInst.DIO().Level().Pins(strPinName).GetVih();
```

2.12.3.9 GetVil（）

函数功能

获取输入的电压下限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVil();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的输入电压下限，单位：V。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_VilData;
mapPin_Site_VilData = TheInst.DIO().Level().Pins(strPinName).GetVil();
```

2.12.3.10 GetVoh（）

函数功能

获取输出的电压上限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVoh();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的输出的电压上限，单位：V。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_VohData;
mapPin_Site_VohData = TheInst.DIO().Level().Pins(strPinName).GetVoh();
```

2.12.3.11 GetVol（）

函数功能

获取输出电压下限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVol();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的输入电压下限，单位：V。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_VolData;
mapPin_Site_VolData = TheInst.DIO().Level().Pins(strPinName).GetVol();
```

2.12.3.12 GetIoh（）

函数功能

获取通道输出电流上限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetIoh();
```

参数说明

无。

返回值

类型	说明
map	输入引脚所允许的最大灌电流，单位：A。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_IohData;
mapPin_Site_IohData = TheInst.DIO().Level().Pins(strPinName).GetIoh();
```

2.12.3.13 GetIol（）

函数功能

获取通道输出电流下限。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetIol();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的输出电流下限，电流值，单位：A。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_IolData;
mapPin_Site_IolData = TheInst.DIO().Level().Pins(strPinName).GetIol();
```

2.12.3.14 GetVt（）

函数功能

获取通道端接电压。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVt();
```

参数说明

无。

返回值

类型	说明
map	对应 Pin 对应 Site 的通道的端接电压，电压值，单位：V。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> mapPin_Site_VtData;
mapPin_Site_VtData = TheInst.DIO().Level().Pins(strPinName).GetVt();
```

2.12.3.15 Set（）

函数功能

设置通道所有状态值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& Set(double dVilVal, double stVihVal, double dVolVal, double
dVohVal, double dIolVal, double dIohVal, double dVtVal);
```

参数说明

参数	说明
dVilVal	输入低电压。
stVihVal	输入高电压。

参数	说明
dVolVal	输出低电压。
dVohVal	输出高电压。
dIolVal	输出低电流。
dIohVal	输出高电流。
dVtVal	端接电压。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).Set(1.1,1.1,1.3,2.2,1.55,1.62,1.7);
```

2.12.3.16 SetIoh（）

函数功能

设置输入引脚所允许的最大灌电流的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetIoh(double dIohVal);
```

参数说明

参数	说明
dIohVal	输入引脚所允许的最大灌电流，单位：A。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetIoh(1.41*mA);
```

2.12.3.17 SetIol（）

函数功能

设置输入引脚所允许的最大漏电流的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetIol(double dIolVal);
```

参数说明

参数	说明
dIolVal	输入引脚所允许的最大漏电流，单位：A。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetIol(1.31*mA);
```

2.12.3.18 SetVih（）

函数功能

设置输入高电平的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVih(double dVihVal);
```

参数说明

参数	说明
dVihVal	输入高电平的值，VIH：-1.5V－6.5V。默认单位为 V。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetVih(1.0*V);
```

2.12.3.19 SetVil（）

函数功能

设置输入低电平的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVil(double dVilVal);
```

参数说明

参数	说明
dVilVal	输入低电平，VIL：﹣1.8V﹣4.5V。默认单位为 V。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetVil(1.0*V);
```

2.12.3.20 SetVoh（）

函数功能

设置输出高电平的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVoh(double dVohVal);
```

参数说明

参数	说明
dVohVal	输出高电平的值，VOH：﹣1V﹣6V。默认单位为 V。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
```

```
TheInst.DIO().Level().Pins(strPinName).SetVoh(1.2*V);
```

2.12.3.21 SetVol（）

函数功能

设置输出低电平的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVol(double dVolVal);
```

参数说明

参数	说明
dVolVal	输出电压下限，VOL：﹣1V﹣6V。默认单位为 V。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetVol(1.11*V);
```

2.12.3.22 SetVt（）

函数功能

设置端接电压的值。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVt(double dVtVal);
```

参数说明

参数	说明
dVtVal	端接电压，单位：V。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetVt(1.234*V);
```

2.12.3.23 SetVhh（）

函数功能

设置高压模式。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetVhhPin(string strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名称。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
```

```
TheInst.DIO().Level().SetVhhPin(strPinName);
```

2.12.3.24 SetTerminationMode（）

函数功能

设置端接模式。

使用说明

无。

函数原型

```
CDIOLevel_Pin& SetTerminationMode(LevelTerm_E eTermMode);
```

参数说明

参数	说明
eTermMode	端接模式，其类型定义详见 LevelTerm_E 类型定义。

返回值

类型	说明
CDIOLevel_Pin	返回 CDIOLevel_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().Level().Pins(strPinName).SetTerminationMode(E_TERM_OFF);
```

2.12.4 Timing

2.12.4.1 Apply（）

函数功能

参数配置下发。

使用说明

调用 SetVih、SetVil 等参数后需要在调用此函数，将参数配置下发生效。

函数原型

```
void Apply();
```

参数说明

无。

返回值

无。

示例

```
TheInst.DIO().Timing().Apply();
```

2.12.4.2 GetFreeRunningClock（）

函数功能

获取 FreeRunningClock。

使用说明

无。

函数原型

```
double GetFreeRunningClock(const string& strSinglePinName);
```

参数说明

参数	说明
strSinglePinName	Pin 的名称。

返回值

参数	说明
double	返回设定的固定时钟频率的值。

示例

```
string strSinglePinName = "Pin1";
double dClock = TheInst.DIO().Timing().GetFreeRunningClock(strSinglePinName);
```

2.12.4.3 SetDriveEdge（）

函数功能

设置驱动边沿。

同一个周期可设置 3 个 timing 值，如 period=40ns，则三个 timing 值可设置为 0ns，15ns，39ns。

使用说明

无。

函数原型

```
bool SetDriveEdge(const string& strPinList, const string& strTimingsetName,
const double dDriveOn, const double dDriveData, const double dDriveReturn);
```

参数说明

参数	说明
strPinList	PinList 的名称。
strTimingsetName	Timingset 的名称。
dDriveOn	开始输入时间，单位：s。
dDriveData	输入数据时间，单位：s。
dDriveReturn	输入返回时间，单位：s。

返回值

类型	说明
bool	• succeed：配置成功。 • fail：配置失败。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
string strTimingSetName = "timeplate_1_0";
bool result = TheInst.DIO().Timing().SetDriveEdge(strPinList, strTimingSetName,
1.0e-9, 2.0e-9, 3.0e-9);
```

2.12.4.4 SetFreeRunningClock（）

函数功能

在配置当前 TimingBlock 界面中，以 UserCode 启动方式的 FreeRunningClock 输出，FreeRunning 用于为 DUT 提供一个固定频率的时钟信号，频率可调。

使用说明

无。

函数原型

```
bool SetFreeRunningClock(const string& strPinList, const double dPeriod);
```

参数说明

参数	说明
strPinList	引脚的名字。
dPeriod	周期的数值，范围为 5ns～200ns，默认单位为 s，若周期不在此范围内，则默认以 5ns 输出。

返回值

类型	说明
bool	● succeed：配置成功。 ● fail：配置失败。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
bool result = TheInst.DIO().Timing().SetFreeRunningClock(strPinList,
0.000000005);
```

2.12.4.5 SetReceiveEdge（）

函数功能

设置比较边沿。

使用说明

无。

函数原型

```
bool SetReceiveEdge(const string& strPinList, const string& strTimingSetName,
const double  dDriveOff, const double  dRecieve1, const double dRecieve2)
```

参数说明

参数	说明
strPinList	PinList 的名称。
strTimingsetName	Timingset 的名称。
dDriveOff	结束输入时间，单位：s。
dRecieve1	输出边沿 1 时间，单位：s。
dRecieve2	输出边沿 2 时间，单位：s。

返回值

类型	说明
bool	• succeed：配置成功。 • fail：配置失败。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
string strTimingSetName = "timeplate_1_0";
bool result = TheInst.DIO().Timing().SetReceiveEdge(strPinList,
strTimingSetName, 1.0e-9, 2.0e-9, 3.0e-9);
```

2.12.4.6 StartFreeRunningClock（）

函数功能

开始机台所有的 free running clock 输出。

使用说明

无。

函数原型

```
uint32_t StartFreeRunningClock()
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：配置成功。 ● 其他：配置失败。

示例

```
uint32_t start_clock = TheInst.DIO().Timing().StartFreeRunningClock();
```

2.12.4.7 StopFreeRunningClock（）

函数功能

停止机台所有的 free running clock 输出。

使用说明

无。

函数原型

```
uint32_t StopFreeRunningClock()
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：配置成功。 ● 其他：配置失败。

示例

```
uint32_t end_clock =TheInst.DIO().Timing().StopFreeRunningClock();
```

2.12.4.8 StopFreeRunningClock（Pinlist）

函数功能

停止指定的某个或某些 Pin 的 free running clock 输出。

使用说明

无。

函数原型

```
uint32_t  StopFreeRunningClock(const string strPinList);
```

参数说明

参数	说明
strPinList	引脚名称。

返回值

类型	说明
uint32_t	● 0：配置成功。 ● 其他：配置失败。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
uint32_t pin_end_clock =
TheInst.DIO().Timing().StopFreeRunningClock(strPinList);
```

2.12.5 TimingBlock

2.12.5.1 Apply（）

函数功能

参数配置下发。

使用说明

无。

函数原型

```
void Apply();
```

参数说明

无。

返回值

无。

示例

```
TheInst.DIO().TimingBlock().Apply();
```

2.12.6 Pattern

2.12.6.1 类型定义

Opcode_E 类型定义

类型	Enumeration	
描述	微指令	
元素	E_NORM = 1	Noop 指令。
	E_STOP	Stop 指令，暂不支持。
	E_REPEAT	Repeat 指令。
	E_LOOP	Loop 指令。
	E_ENDLOOP	ENDLOOP 指令。需配合 Loop 指令一起使用。
	E_MATCH	MATCH 指令，暂不支持。
	E_MATCHLOOP	MATCHLOOP 指令，暂不支持。
	E_ENDMATCH	ENDMATCH 指令，暂不支持。
	E_TRIG	TRIG 指令，暂不支持。
	E_CLEV	CLEV 指令，暂不支持。
	E_SCAN	SCAN 指令，暂不支持。
头文件	UserDef.h	

2.12.6.2 Continue（）

函数功能

继续运行 Pattern。

使用说明

无。

函数原型

```
uint32_t Continue()
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：配置成功。 ● 其他：配置失败。

示例

```
RunPatStatus_E enStatus;
 TheInst.DIO().Pattern().NonBlockingStart ();
  do {
    enStatus = TheInst.DIO().Pattern().GetRunningStatus();
    if (enStatus == E_PAT_INFINITE)
     {
  PMU_Test();
       TheInst.DIO().Pattern().Continue ();
   }
 }
  while ( (enStatus != E_PAT_COMPLETE) && (enStatus != E_PAT_ERROR));
```

2.12.6.3 GetLabelVector（）

函数功能

获取 Label 索引。

使用说明

无。

函数原型

```
uint64_t GetLabelVector(string& strPatternName, string& strLabel);
```

参数说明

参数	说明
strPatternName	Pattern 名称（cbp 全路径）。
strLabel	Label 的名称。

返回值

类型	说明
uint64_t	● ≥0：正常索引值。 ● -1：获取索引失败。

示例

```
string strPatPath;
TheSoft.TestProc().GetProgPath(strPatPath);
if (strPatPath.find(".cpts") >= 0)
{
strPatPath = strPatPath.substr(0, strPatPath.rfind("\\"));
}
if (strPatPath[strPatPath.size() - 1] != '/' && strPatPath[strPatPath.size() -
1] != '\\')
{
strPatPath += "\\";
}
strPatPath += "Patterns\\";
string strPatternName = "FailCycle-32Ch-4EC-1SITE_Label.cbp";
string strLabel = "test1";
int64_t i64Index = TheInst.DIO().Pattern().GetLabelVector(strPatPath +
strPatternName, strLabel);
```

2.12.6.4 GetTimeOut（）

函数功能

获取设置超时的值。

使用说明

无。

函数原型

```
uint32_t GetTimeOut();
```

参数说明

无。

返回值

类型	说明
uint32_t	超时值。

示例

```
uint32_t timeout = TheInst.DIO().Pattern().GetTimeOut();
```

2.12.6.5 GetRunningStatus（）

函数功能

获取 pattern 运行状态。

使用说明

无。

函数原型

```
RunPatStatus_E GetRunningStatus();
```

参数说明

无。

返回值

类型	说明
RunPatStatus_E	pattern 运行状态。 ● E_PAT_COMPLETE（0）：已完成。 ● E_PAT_RUNNING（1）：正在运行。 ● E_PAT_INFINITE（2）：执行 INFINITE 指令。 ● E_PAT_ERROR（0xFF）：执行出错。

示例

```
RunPatStatus_E enStatus = TheInst.DIO().Pattern().GetRunningStatus();
```

2.12.6.6 NonBlockingStart（）

函数功能

非阻塞式运行 pattern。

使用说明

无。

函数原型

```
uint32_t NonBlockingStart();
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：成功。 ● 其他：失败。

示例

```
RunPatStatus_E enStatus;
 TheInst.DIO().Pattern().NonBlockingStart ();
  do {     enStatus = TheInst.DIO().Pattern().GetRunningStatus();
    if (enStatus == E_PAT_INFINITE)
     {
  PMU_Test();
       TheInst.DIO().Pattern().Continue ();
   }
 }
  while ( (enStatus != E_PAT_COMPLETE) && (enStatus != E_PAT_ERROR));
```

2.12.6.7 Start（）（软件选择）

函数功能

运行当前选中的 flow 中对应的 PatternBlockName。

使用说明

无。

函数原型

```
bool Start();
```

参数说明

无。

返回值

参数	说明
bool	Pattern block 运行结果。 ● true：运行 Pattern block 成功。 ● false：运行 Pattern block 异常。

示例

```
bool PatternBlock = TheInst.DIO().Pattern().Start();
```

2.12.6.8 Start（）（程序写入）

函数功能

运行用户指定的 patternBlockName。

使用说明

无。

函数原型

```
bool Start(const string& strPatternBlockName);
```

参数说明

参数	说明
strPatternBlockName	所选的 patternBlock。

返回值

参数	说明
bool	Pattern block 运行结果。 ● true：运行 Pattern block 成功。 ● false：运行 Pattern block 异常。

示例

```
string strPatternBlockName = "PatternBlockName";
bool PatternBlock = TheInst.DIO().Pattern().Start(strPatternBlockName);
```

2.12.6.9 SetTimeOutEnable（）

函数功能

设置超时的使能。

使用说明

该函数默认开启了超时功能，若通过 SetTimeOutEnable(false)配置关闭超时功能时会不生效，需使用 SetTimeOut（）接口配置等待超时时间。

函数原型

```
void SetTimeOutEnable(bool bEnable);
```

参数说明

参数	说明
bEnable	超时使能。 取值如下所示。 ● true：使能超时。 ● false：关闭超时。

返回值

无。

示例

```
TheInst.DIO().Pattern().SetTimeOutEnable(true);
```

2.12.6.10 SetTimeOut（）

函数功能

设置超时的值。

使用说明

无。

函数原型

```
void SetTimeOut(uint32_t uiTimeout);
```

参数说明

参数	说明
uiTimeout	设置超时时间值，单位：ms。

返回值

无。

示例

```
TheInst.DIO().Pattern().SetTimeOut(1000);
```

2.12.6.11 SetVectorDataArray（）

函数功能

对齐所有动态 Pattern 函数的返回值。

使用说明

无。

函数原型

```
uint32_t SetVectorDataArray(const string& strPatternBlockName, const string&
strPatternName, const string& strPinList, uint32_t iLabel, uint32_t iOffset,
vector<uint32_t> vecDataArray);
```

参数说明

参数	说明
strPatternBlockName	要修改的 Pattern Block 的名称。
strPatternName	Pattern 文件名称。
strPinList	要修改的 PinsList。
iLabel	要修改的起始 vector 数。
iOffset	要修改的数据个数。
vecDataArray	要修改的具体数据。

返回值

类型	说明
uint32_t	● 0：成功。 ● 其他：错误码。

示例

```
TheInst.DIO().Pattern().SetVectorDataArray(strPatternBlockName, strPatternName,
"DIO_P000", 0, 9, vecDriveData);
```

2.12.6.12 SetVectorDataArray（）

函数功能

设置动态修改 Pattern。

使用说明

无。

函数原型

```
uint32_t SetVectorDataArray(const string& strPatternBlockName, const string&
strPatternName, uint32_t uiLabel, map<string, vector<uint32_t>>
&mapPinNameData);
```

参数说明

参数	说明
strPatternBlockName	要修改的 Pattern Block 的名称。
strPatternName	Pattern 名称（cbp 全路径）。
uiLabel	修改 pattern 的起始位置。
mapPinNameData	对应 mapPin 对应 Site 的数据。

返回值

类型	说明
uint32_t	● 0：设置成功。 ● 其他：设置失败。

示例

```
String strPatternBlock=”patterblock”;
String strPatternName=”patternname.cbp”;
map<string, vector<uint32_t>> mapPinNameData;
for (int i = 0; i < uiVecSize; i++)
{
    mapPinNameData[strDrivePin].push_back((i+1) % 2);
}
TheInst.DIO().Pattern().SetVectorDataArray(strPatternBlock, strPatternName, 0,
mapPinNameData);
```

2.12.6.13 SetVectorDataArray（）

函数功能

为提升动态 Pattern 效率，将要更改的起始向量参数和具体修改的向量数参数一次性写入，在需要更改几个起始位置时能够避免重复调用。

使用说明

无。

函数原型

```
uint32_t SetVectorDataArray(const string& strPatternBlockName, const string&
strPatternName, map<string, map<uint32_t, vector<uint32_t>>>&
mapPinNameIndexData);
```

参数说明

参数	说明
strPatternBlockName	Pattern Block 名称。
strPatternName	Pattern 文件名称。
mapPinNameIndexData	map<PinName，map<需要修改的起始 vector，vector<具体修改的数据>>>。

返回值

类型	说明
uint32_t	● 0：成功。 ● 其他：错误码。

示例

```
String strPatternBlock=”patterblock”;
String strPatternName=”patternname.cbp”;
string strDrivePin = "DIO_P001";
map<string, map<uint32_t, vector<uint32_t>>> mapPinNameIndexData;
for (int i = 0; i < uiVecSize; i++)
{
    mapPinNameIndexData[strDrivePin][0].push_back(i % 2);
}
TheInst.DIO().Pattern().SetVectorDataArray(strPatternBlock, strPatternName,
mapPinNameIndexData);
```

2.12.6.14 SetVectorSitesDataArray（）

函数功能

为提升动态 Pattern 效率，将要更改的起始向量参数和具体修改的向量数参数一次性写入，在需要更改几个起始位置时能够避免重复调用。

使用说明

无。

函数原型

```
uint32_t SetVectorSitesDataArray(const string& strPatternBlockName, const
string& strPatternName, map<string, map<uint32_t, map<uint32_t,
vector<uint32_t>>>>& mapPinNamesSitesIndexData);
```

参数说明

参数	说明
strPatternBlockName	Pattern Block 名称。
strPatternName	Pattern 文件名称（cbp 全路径）。
mapPinNamesSitesIndexData	map<PinName, map<需要修改的 Site，map<需要修改的起始 vector，vector<具体修改的数据>>>>。

返回值

类型	说明
uint32_t	● 0：成功。 ● 其他：错误码。

示例

```
string strPatPath;
TheSoft.TestProc().GetProgPath(strPatPath);
if (strPatPath.find(".cpts") >= 0)
{
strPatPath = strPatPath.substr(0, strPatPath.rfind("\\"));
}
if (strPatPath[strPatPath.size() - 1] != '/' && strPatPath[strPatPath.size() -
1] != '\\')
{
strPatPath += "\\";
}
strPatPath += "Patterns\\";
map<uint32_t, vector<uint32_t>> mapIndexData;
mapIndexData.insert(make_pair(5, vecDriveData1));
mapIndexData.insert(make_pair(15, vecDriveData2));
mapIndexData.insert(make_pair(25, vecDriveData3));
map<uint32_t, map<uint32_t, vector<uint32_t>>> mapSitesIndexData;
mapSitesIndexData.insert(make_pair(0, mapIndexData));
mapSitesIndexData.insert(make_pair(1, mapIndexData));
mapSitesIndexData.insert(make_pair(2, mapIndexData));
```

```
mapSitesIndexData.insert(make_pair(3, mapIndexData));
map<string, map<uint32_t, map<uint32_t, vector<uint32_t>>>>
mapPinNamesSitesIndexData;
mapPinNamesSitesIndexData.insert(make_pair("DIO_P000", mapSitesIndexData));
TheInst.DIO().Pattern().SetVectorSitesDataArray("EP", strPatPath + "FailCycle-
32Ch-4EC-1SITE.cbp", mapPinNamesSitesIndexData);
```

2.12.6.15 SetVectorMicroInstruction（）

函数功能

动态修改微指令。

使用说明

微指令 E_LOOP 和 E_ENDLOOP 需配合使用，否则会导致异常。

函数原型

```
uint32_t SetVectorMicroInstruction(const string& strPatternBlockName, const
string& strPatternName, uint32_t uiVectorNum, Opcode_E emOpcode, uint32_t
uiOperand);
```

参数说明

参数	说明
strPatternBlockName	准备修改的 Pattern Block 的名称。
strPatternName	对应 Pattern 文件名称（cbp 全路径）。
uiVectorNum	修改微指令索引值。
emOpcode	微指令，其类型定义详见 Opcode_E 类型定义。
uiOperand	微指令参数。

返回值

类型	说明
uint32_t	● 0：设置成功。 ● 其他：设置失败。

示例

```
String strPatternBlock=”patterblock”;
String strPatternName=”patternname.cbp”;
Opcode_E eOpcodeSet = E_REPEAT;
TheInst.DIO().Pattern().SetVectorMicroInstruction(strPatternBlock,
strPatternName, 10, eOpcodeSet, 10);
```

2.12.6.16 SetVectorSitesDataArray（）

函数功能

动态修改 pattern，多 Site 修改为不同向量。

使用说明

无。

函数原型

```
uint32_t SetVectorSitesDataArray(const string& strPatternBlockName, const
string& strPatternName, uint32_t uiFirstVector, map<string, map<uint32_t,
vector<uint32_t>>> mapPinNamesSitesData);
```

参数说明

参数	说明
strPatternBlockName	准备修改的 Pattern Block 的名称。
strPatternName	Pattern 文件名称（cbp 全路径）。
uiFirstVector	第一个向量的位置。
mapPinNamesSitesData	添加 Pattern 有效数据说明。

返回值

类型	说明
uint32_t	● 0：设置成功。 ● 其他：设置失败。

示例

```
string strPatternBlockName = "PatternBlockName";
string strPatternName = "writecpb\SD6221_DR2W_template_test.cbp";
```

```
map<string, map<uint32_t, vector<uint32_t>>> mapPinNamesSitesData;
vector<uint32_t> vecData = {1, 2};
vector<uint32_t> vecData1 = {3, 4};
map<uint32_t, vector<uint32_t>> mapData ;
mapData[0] = vecData;
mapData[1] = vecData1;
mapPinNamesSitesData["TX_ANA0_P"] = mapData;
uint32_t site_vector =
TheInst.DIO().Pattern().SetVectorSitesDataArray(strPatternBlockName,strPatternNa
me,0, mapPinNamesSitesData);
```

2.12.6.17 StopAllPattern（）

函数功能

停止运行 Pattern。

函数原型

```
bool StopAllPattern();
```

参数说明

无。

返回值

参数	说明
bool	● true：pattern 正在运行。 ● false：完成运行 Pattern。

示例

```
bool pattern = TheInst.DIO().Pattern().StopAllPattern();
```

2.12.7 PatEng

2.12.7.1 CaptureDataCount（）

函数功能

获取 capture 数据的总数。

使用说明

无。

函数原型

```
uint32_t CaptureDataCount(map< uint8_t, uint64_t >& mapCaptureCount);
```

参数说明

参数	说明
mapCaptureCount	总的 Capture 数据。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
string strPinList = "DIO_P000";
map<uint8_t, map<string, uint64_t>> mapCaptureCount;
uint32_t uiCap = TheInst.DIO().PatEng().CaptureDataCount(strPinList,
mapCaptureCount);
```

2.12.7.2 GetExecutedCyclesCount（）

函数功能

获取当前 pattern 执行的 device cycle 数。

使用说明

无。

函数原型

```
uint64_t GetExecutedCyclesCount();
```

参数说明

无。

返回值

类型	说明
uint64_t	当前 Pattern 执行的 Device cycle 数。

示例

```
uint64_t pattern_cycle = TheInst.DIO().PatEng().GetExecutedCyclesCount();
```

2.12.7.3 GetFailCycles（Pinlist）

函数功能

获取当前 Pattern 指定 Pinlist 对应 Site 的 Device cycles 数。

使用说明

无。

函数原型

```
map<uint32_t, vector<uint64_t>> GetFailCycles(uint64_t iStartNum, const string&
strPinList);
```

参数说明

类型	说明
iStartNum	指定开始测量位置。
strPinList	指定的 Pin 列表。

返回值

类型	说明
map	返回对象对应关系。

示例

```
map<uint32_t, vector<uint64_t>> mapFailCycle;
string strPinList = "GPIO_1,GPIO_2";
mapFailCycle = TheInst.DIO().PatEng().GetFailCycles(0,strPinList);
```

2.12.7.4 GetFailCycles（Site）

函数功能

获取对应 Site 运行 Pattern 文件 Fail 的 Cycle 号。

使用说明

无。

函数原型

```
map<uint32_t, vector<uint64_t>> GetFailCycles(uint64_t iStartNum);
```

参数说明

类型	说明
iStartNum	指定开始测量位置。

返回值

类型	说明
map	返回对象对应关系。

示例

```
map<uint32_t, vector<uint64_t>> map_site_FailCycle;
map_site_FailCycle = TheInst.DIO().PatEng().GetFailCycles(0);
```

2.12.7.5 GetFailSize（）

函数功能

获取 Fail 结果的存储深度。

使用说明

无。

函数原型

```
uint32_t GetFailSize();
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：获取存储 Fail 结果的深度成功。 ● 其他值：获取 Fail 结果的深度失败。

示例

```
uint32_t uiRet = TheInst.DIO().PatEng().GetFailSize();
```

2.12.7.6 GetCaptureData（Site）

函数功能

获取当前 Site Capture 测量的结果。

使用说明

无。

函数原型

```
map<uint32_t, map<string, vector<GetCaptureData_T>>> GetCaptureData()
```

参数说明

无。

返回值

类型	说明
map	当前 Site Capture 测量的结果。

示例

```
map<uint32_t, map<string, vector<GetCaptureData_T>>> PatCaptureData;
PatCaptureData = TheInst.DIO().PatEng().GetCaptureData();
```

2.12.7.7 GetCaptureData（Pinlist）

函数功能

获取指定 PinList Capture 测量的结果。

使用说明

无。

函数原型

```
map<uint32_t, map<string, vector<GetCaptureData_T>>> GetCaptureData(const string& strPinList);
```

参数说明

参数	说明
strPinList	Pinlist 的名称。

返回值

类型	说明
map	指定 PinList Capture 测量的结果。

示例

```
string strPinList = "GPIO_1,GPIO_2";
map<uint32_t, map<string, vector<GetCaptureData_T>>> PatCaptureData;
PatCaptureData = TheInst.DIO().PatEng().GetCaptureData(strPinList);
```

2.12.7.8 GetCapturDataByBlock（）

函数功能

分包读取 capture 数据。

使用说明

无。

函数原型

```
uint32_t GetCapturDataByBlock(uint32_t uiStart, uint32_t uiLength, map<uint8_t,
map<string, vector<GetCaptureData_T>>>& mapCaptureDataR, map<uint8_t,
map<string, vector<uint8_t>>>& mapCaptureRawDataR);
```

参数说明

参数	说明
uiStart	串行数据的起始 cycle。
uiLength	串行数据取数据的长度。

mapCaptureDataR	map<site 号, map<PInName，vector<获取的原始数据>>>。
	其中 GetCaptureData_T 结构体定义如下：
	`struct GetCaptureData_T;`
	`{`
	`ucECResult Timing;//配置 EC 时获取的数据`
	`ucECCResult Timing;//配置 ECC 时获取的数据`

mapCaptureRawDataR	map<site 号, map<PInName，vector<获取的原始数据>>>。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint8_t, map<string, vector<GetCaptureData_T>>> mapCaptureDataR;
map<uint8_t, map<string, vector<uint8_t>>> mapCaptureRawDataR;
uint32_t usize = TheInst.DIO().PatEng().GetCapturDataByBlock(1,3,
mapCaptureDataR,mapCaptureRawDataR);
```

2.12.7.9 GetCaptureRawData（Site）

函数功能

获取当前 Site capture 测量结果的原始数据。

使用说明

无。

函数原型

```
map<uint32_t, map<string, vector<uint8_t>>> GetCaptureRawData();
```

参数说明

无。

返回值

类型	说明
map	当前 Site capture 测量结果的原始数据。

示例

```
map<uint32_t, map<string, vector<uint8_t> > > captureRawData;
captureRawData = TheInst.DIO().PatEng().GetCaptureRawData();
```

2.12.7.10 GetCaptureRawData（PinList）

函数功能

获取指定 pinList 的 capture 测量结果的原始数据。

使用说明

无。

函数原型

```
map<uint32_t, map<string, vector<uint8_t>>> GetCaptureRawData(const string&
strPinList);
```

参数说明

参数	说明
strPinList	PinList 的名称。

返回值

类型	说明
map	map < site, map< inName, vector < capture data > >

示例

```
captureRawData = TheInst.Digital().PatEng().GetCaptureRawData(strPinList);
```

2.12.7.11 GetFailHILData（Pin）

函数功能

获取指定 Pin 的 HIL 模式 Capture 数据的接口。

使用说明

该函数一定有 Fail 的情况下才会有返回数据，全 Pass 无数据。

函数原型

```
uint32_t GetFailHILData(const uint64_t uiStartNum, const string& strPinList,
map<uint8_t, map<string, map<uint64_t, uint8_t>>>& mapSitePinCycleData);
```

参数说明

参数	说明
uiStartNum	需要获取数据的 start cycle（fail cycle）。
strPinList	Pin list。
mapSitePinCycleData	对应 SitemapSitePinCycleData 是返回的数据，第一层是 Site 号，第二层是 Pin name，第三层的 key 值是 cycle 号，value 是 Capture 数据。

返回值

类型	说明
uint32_t	按照传入的 Pin 列表返回数据，无 Pinlist 是按照 Pattern 文件中的 Pin 列表获取全部数据。

示例

```
string strPinList = "PinName1, PinName2,PinGroup1";
```

```
map<uint8_t, map<string, map<uint64_t, uint8_t>>> mapSitePinCycleData;
uint32_t usize = TheInst.DIO().PatEng().GetFailHILData(1,strPinList,
mapSitePinCycleData);
```

2.12.7.12 GetFailHILData（）

函数功能

获取所有失败的 HIL 数据。

使用说明

该函数一定有 Fail 的情况下才会有返回数据，全 Pass 无数据。

使用说明

无。

函数原型

```
uint32_t GetFailHILData(const uint64_t uiStartNum, map<uint8_t, map<string,
map<uint64_t, uint8_t>>>& mapSitePinCycleData);
```

参数说明

参数	说明
uiStartNum	是要获取数据的 Start cycle（Fail cycle）。
mapSitePinCycleData	对应 sitemapSitePinCycleData 是返回的数据，第一层是 Site 号，第二层是 Pin name，第三层的 key 值是 Cycle 号，Value 是 Capture 数据。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint8_t, map<string, map<uint64_t, uint8_t>>> mapSitePinCycleData;
uint32_t usize = TheInst.DIO().PatEng().GetFailHILData(1, mapSitePinCycleData);
```

2.12.7.13 GetFailCount（Site）

函数功能

获取指定 Pin 的失败 cycle 数。

使用说明

无。

函数原型

```
uint32_t GetFailCount(map<uint32_t, uint32_t>& mapFailCount);
```

参数说明

参数	说明
mapFailCount	是要获取数据的 Start cycle（Fail cycle）。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint32_t, uint32_t> mapFailCount;
uint32_t uiRet= TheInst.DIO().PatEng().GetFailCount(mapFailCount);
```

2.12.7.14 GetFailCount（Pinlist）

函数功能

获取指定 Pin 的失败 cycle 数。

使用说明

无。

函数原型

```
uint32_t GetFailCount(const string& strPinList, map<uint32_t, map<string,
uint32_t>>& mapFailPinCount);
```

参数说明

参数	说明
strPinList	Pin 的名称。
mapFailPinCount	map < site number, map<PinName, Counter >>。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint32_t, map<string, uint32_t>> mapFailPinCount;
string strPinName = "DIO_P001,DIO_P002"
uint32_t uiRet=
TheInst.DIO().PatEng().GetFailCount(strPinName ,mapFailPinCount);
```

2.12.7.15 GetParallelCaptureData（）

函数功能

并行获取 capture data 数据接口。

使用说明

无。

函数原型

```
uint32_t GetParallelCaptureData(const std::string& strPinNames,
std::map<uint8_t, std::vector<uint64_t>>& mapParaData, uint32_t uiStart,
uint32_t uiLength);
```

参数说明

参数	说明
strPinNames	pinName 列表，最大 64 个 SinglePin。
uiStart	并行数据的起始 cycle。

参数	说明
uiLength	并行数据取数据的长度。
mapParaData	并行数据返回值。string：pinName，vector 中每个成员代表一个 cycle。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint8_t, vector<uint64_t>> mapParaData;
string strPinList = "PinName1,PinName2,PinGroup1";
uint32_t usize = TheInst.DIO().PatEng().GetParallelCaptureData(strPinList, 0,
10, mapParaData);
```

2.12.7.16 GetParallelMatchWaitCycle（）

函数功能

获取 Match 匹配循环次数。

使用说明

无。

函数原型

```
uint32_t GetParallelMatchWaitCycle(map<uint8_t, map<string, vector<uint32_t>>>
&mapSitePinCycle)
```

参数说明

参数	说明
mapSitePinStartIndex	<Site 号，<PinName，<该 PinName 从开执行 Match 到 Match 成功所执行的循环次数>>>。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint8_t, map<string, vector<uint32_t>>> mapSitePinCycle;
uint32_t uiRet = GetParallelMatchWaitCycle(mapSitePinCycle);
```

2.12.7.17 GetParallelMatchWaitFail（）

函数功能

获取 Match 失败结果。

使用说明

无。

函数原型

```
uint32_t GetParallelMatchWaitFail(map<uint8_t, map<string, map<uint32_t,
uint64_t>>> &mapSitePinFailVector);
```

参数说明

参数	说明
mapSitePinFailVector	<Site 号，<PinName，<对应 Site 及 PinName 下的 MactchWait fail 的微指令索引，对应的 fail cycle>>>。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
map<uint8_t, map<string, map<uint32_t, uint64_t>>> mapSitePinFailVector;
Uint32_t uiRet =
TheInst.DIO().PatEng().GetParallelMatchWaitFail(mapSitePinFailVector);
```

2.12.7.18 GetPassFail（）

函数功能

获取执行 Pattern 所有 Site 的 Pass，Fail 结果。

使用说明

无。

函数原型

```
map<uint32_t, bool> GetPassFail();
```

参数说明

无。

返回值

类型	说明
map	获取对应 Site，Pass/Fail 结果。

示例

```
map<uint32_t, bool> map_pattern;
map_pattern = TheInst.DIO().PatEng().GetPassFail();
```

2.12.7.19 GetSerialCaptureData（）

函数功能

串行获取 capture data 数据接口。

使用说明

无。

函数原型

```
uint32_t GetSerialCaptureData(const std::string& strPinNames, std::map<uint32_t,
std::map<std::string, std::vector<uint64_t>>>& mapSeriData, uint32_t uiStart,
uint32_t uiLength, uint32_t uiBits, uint32_t uiDelta, uint32_t uiMLsb);
```

参数说明

参数	说明
strPinNames	pinName 列表，最大 64 个 SinglePin。
mapSeriData	串行数据返回值，Vector 中每个成员代表一组数据。
uiStart	串行数据的起始 Cycle。
uiLength	串行数据取数据的长度。
uiBits	串行数据多少个 Cycle 为一组，最大值为 64。
uiDelta	串行数据一组之后跳过的 Cycle 数。
uiMLsb	串行数据一组之中大小端顺序。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
map<uint32_t, map<string, vector<uint64_t> > > mapSeriData;
uint32_t uiRet = TheInst.DIO().PatEng().GetSerialCaptureData(strPinList, 1, 10,
2, 1,7,mapSeriData);
```

2.12.7.20 SetFailMode（）

函数功能

设置抓取模式 only fail or all，只能设置连续模式或非连续模式。

使用说明

- run Pattern 之前调用。
- 默认值修改为 E_SET_NORMAL_MODE。

函数原型

```
uint32_t SetFailMode(SetFailModeType_E emMode);
```

参数说明

参数	说明
emMode	● E_SET_SEQ_PF_MODE：pass/fail 模式。 ● E_SET_SEQ_HIL_MODE：高阻态模式。 ● E_SET_SEQ_BLOCK_MODE：BlocK 模式。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
uint32_t uiRet = TheInst.DIO().PatEng().SetFailMode(E_SET_SEQ_PF_MODE);
```

2.12.7.21 SetFailSize（）

函数功能

设置存储 Fail 结果的深度。

使用说明

无。

函数原型

```
uint32_t SetFailSize(const uint32_t uiSize);
```

参数说明

参数	说明
uiSize	存储深度，最大为 128M。

返回值

类型	说明
uint32_t	● 0：设置存储 Fail 结果的深度成功。 ● 其他：设置 Fail 结果的深度失败。

示例

```
uint32_t uiRet = TheInst.DIO().PatEng().SetFailSize(10000);
```

2.12.7.22 SetFailStartVector（）

函数功能

设置从指定 device cycle 开始存储 fail cycle 信息。

使用说明

run Pattern 之前调用。

函数原型

```
uint32_t SetFailStartVector(const uint64_t  uiStartNum);
```

参数说明

参数	说明
uiStartNum	开始的数。

返回值

类型	说明
uint32_t	指定 Device cycle 开始存储 Fail cycle 信息。

示例

```
uint32_t uiRet = TheInst.DIO().PatEng().SetFailStartVector(10);
```

2.12.7.23 SetParallelMatchWaitPin（）

函数功能

Match Pin 使能。

使用说明

无。

函数原型

```
uint32_t SetParallelMatchWaitPin(const string &strPatternBlockName, const vector<string> &vecPinList)
```

参数说明

参数	说明
strPatternBlockName	PatternBlockName 的名称。
vecPinList	PinName 的列表。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
string strPatternBlockName = strPatternBlock1;
vector<string> vecPinList;
vecPinList.pushback(“Pinname1”);
uint32_t uiRet = SetParallelMatchWaitPin(strPatternBlockName, vecPinList);
```

2.12.7.24 SetParallelMatchWaitStartIndex（）

函数功能

Match Pin 获取结果起始索引配置。

使用说明

无。

函数原型

```
uint32_t SetParallelMatchWaitStartIndex(const string &strPatternBlockName, const map<uint8_t, map<string, uint32_t>> &mapSitePinStartIndex)
```

参数说明

参数	说明
strPatternBlockName	PatternBlockName 的名称。
mapSitePinStartIndex	<Site 号，<PinName，索引号>>。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
string strPatternBlockName = strPatternBlock1;
string Pin = "DIO_P007";
map<uint8_t, map<string, uint32_t>> mapSitePinStartIndex;
mapSitePinStartIndex[0][Pin] = 1;
mapSitePinStartIndex[1][Pin] = 1;
mapSitePinStartIndex[2][Pin] = 3;
mapSitePinStartIndex[3][Pin] = 4;
TheInst.DIO().PatEng().SetParallelMatchWaitStartIndex(strPatternBlock,
mapSitePinStartIndex);
```

2.12.7.25 SetupMCFData（）

函数功能

配置 MCF 数据。

- 设置的当前 patternblockname。
- 需要在 run pattern 之前调用。

使用说明

无。

函数原型

```
uint32_t SetupMCFData();
```

参数说明

无。

返回值

类型	说明
uint32_t	● 0：获取成功。 ● 其他：获取失败。

示例

```
uint32_t usize = TheInst.DIO().PatEng().SetupMCFData();
```

2.12.8 PPMU

2.12.8.1 类型定义

PmuDcMeasureMode_E 类型定义

类型	Enumeration	
描述	测量模式。	
元素	E_DC_FV_MI	加压测流。
	E_DC_FV_MV	加压测压。
	E_DC_FV_MOhm	测量负载。
	E_DC_FI_MI	加流测流。
	E_DC_FI_MV	加流测压。
	E_DC_FI_MOhm	加流/测量负载。
	E_DC_VOL	Search the min VOL。
	E_DC_VOH	Search the max VOH。
	E_DC_VOX	Measurement of the common mode of diff pins。
	E_DC_MAX	无效值。
头文件	UserDef.h	

DcMeasureType_E 类型定义

类型	Enumeration	
描述	测量类型。	
元素	E_GONOGO	GONOGO 模式。
	E_MEASURE	测量模式。
	E_MEASURE_RAW	测量数据未处理模式。

	E_MAX	无效值。
头文件	UserDef.h	

2.12.8.2 Apply（）

函数功能

参数配置下发至硬件。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& Apply();
```

参数说明

无。

返回值

类型	说明
CDIOPPMU_Pin	返回当前 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).Apply();
```

2.12.8.3 Connect（）

函数功能

连接 F_S 引脚。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& Connect(bool bConnect);
```

参数说明

参数	说明
bConnect	是否连接。 ● 0：不连接。 ● 1：连接。

返回值

类型	说明
CDIOPPMU_Pin	返回当前 CDIOPPMU_Pin 自身对象。

示例

```
string strPinList="GPIO_9,PinGroup1";
TheInst.DIO().PPMU().Pins(strPinList).Connect(true);
TheInst.PPMU().Pins(strPinList).Connect(true)
```

2.12.8.4 Measure（）

函数功能

开始测试。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& Measure();
```

参数说明

无。

返回值

类型	说明
CDIOPPMU_Pin	返回当前 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).Measure();
```

2.12.8.5 MeasWithRCompensate（）

函数功能

获取线阻补偿结果。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& MeasWithRCompensate(uint32_t uicount);
```

参数说明

参数	说明
uicount	补偿次数（1 或 2）。

返回值

类型	说明
CDIOPPMU_Pin	● 0：成功。 ● 其他：错误码。

示例

```
TheInst.DIO().PPMU().Pins(pcSigName).MeasWithRCompensate(1);
```

2.12.8.6 GetIForce（）

函数功能

获取输出电流的值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetIForce();
```

参数说明

无。

返回值

类型	说明
map	输入电流的值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> IForce;
IForce = TheInst.DIO().PPMU().Pins(strPinName).GetIForce();
```

2.12.8.7 GetVForce（）

函数功能

获取输出电压的值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVForce();
```

参数说明

无。

返回值

类型	说明
map	输入电压的值。

示例

```
string strPinName = "Pin1";
```

```
map<string, map<uint32_t, double>> VForce;
VForce = TheInst.DIO().PPMU().Pins(strPinName).GetVForce();
```

2.12.8.8 GetFIRange（）

函数功能

获取输出电流的范围。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetFIRange();
```

参数说明

无。

返回值

类型	说明
map	输出电流的范围。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> FIRange;
FIRange = TheInst.DIO().PPMU().Pins(strPinName).GetFIRange();
```

2.12.8.9 GetMIRange（）

函数功能

获取测量电流的范围。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetMIRange();
```

参数说明

无。

返回值

类型	说明
map	测量电流的范围。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> MIRange;
MIRange = TheInst.DIO().PPMU().Pins(strPinName).GetMIRange();
```

2.12.8.10 GetVClampL（）

函数功能

获取钳位电压下限值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVClampL();
```

参数说明

无。

返回值

类型	说明
map	钳位电压下限值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> VClampL;
VClampL = TheInst.DIO().PPMU().Pins(strPinName).GetVClampL();
```

2.12.8.11 GetVClampH（）

函数功能

获取钳位电压上限值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetVClampH();
```

参数说明

无。

返回值

类型	说明
map	钳位电压上限值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> VClampH;
VClampH = TheInst.DIO().PPMU().Pins(strPinName).GetVClampH();
```

2.12.8.12 GetIClampL（）

函数功能

获取钳位电流下限值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetIClampL();
```

参数说明

无。

返回值

类型	说明
map	钳位电流下限值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> IClampL;
IClampL = TheInst.DIO().PPMU().Pins(strPinName).GetIClampL();
```

2.12.8.13 GetIClampH（）

函数功能

获取钳位电流上限值。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetIClampH();
```

参数说明

无。

返回值

类型	说明
map	钳位电流上限值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> IClampH;
IClampH = TheInst.DIO().PPMU().Pins(strPinName).GetIClampH();
```

2.12.8.14 GetWaitTime（）

函数功能

获取设置的等待时间。

使用说明

无。

函数原型

```
double GetWaitTime();
```

参数说明

无。

返回值

类型	说明
double	等待时间。

示例

```
double dwait_time;
string strPinName = "Pin1";
dwait_time = TheInst.DIO().PPMU().Pins(strPinName).GetWaitTime();
```

2.12.8.15 GetSampleSize（）

函数功能

获取设置的采样次数。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetSampleSize();
```

参数说明

无。

返回值

类型	说明
map	采样点数值。

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> SampleSize;
SampleSize = TheInst.DIO().PPMU().Pins(strPinName).GetSampleSize();
```

2.12.8.16 GetMeasureMode（）

函数功能

获取测量模式。

使用说明

无。

函数原型

```
PmuDcMeasureMode_E  GetMeasureMode();
```

参数说明

无。

返回值

类型	说明
PmuDcMeasureMode_E	测量模式，其类型定义详见 PmuDcMeasureMode_E 类型定义。

示例

```
string strPinName = "Pin1";
PmuDcMeasureMode_E MeasureMode =
TheInst.DIO().PPMU().Pins(strPinName).GetMeasureMode();
```

2.12.8.17 GetMeasureType（）

函数功能

获取测量类型。

使用说明

无。

函数原型

```
DcMeasureType_E GetMeasureType();
```

参数说明

无。

返回值

类型	说明
DcMeasureType_E	测量类型，其类型定义详见 DcMeasureType_E 类型定义。

示例

```
string strPinName = "Pin1";
DcMeasureType_E MeasureType =
TheInst.DIO().PPMU().Pins(strPinName).GetMeasureType()
```

2.12.8.18 GetMeasureResults（）

函数功能

获取测量结果。

使用说明

无。

函数原型

```
map<string, map<uint32_t, double>> GetMeasureResults();
```

参数说明

无。

返回值

类型	说明
map	map<string, map<uint32_t, double>>

示例

```
string strPinName = "Pin1";
map<string, map<uint32_t, double>> MeasureResults;
MeasureResults = TheInst.DIO().PPMU().Pins(strPinName).GetMeasureResults();
```

2.12.8.19 Pins（）

函数功能

获取 Pin 的名字。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& Pins(const string& PinList)
```

参数说明

参数	说明
PinList	Pin 的名字。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName);
```

2.12.8.20 SetClear（）

函数功能

清除所有配置信息。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetClear();
```

参数说明

无。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetClear();
```

2.12.8.21 SetFIRange（）

函数功能

设置输出电流范围。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetFIRange(const double dForceCurrentRange);
```

参数说明

参数	说明
dForceCurrentRange	输出电流范围，可选挡位±5μA、±50μA、±500μA、±5mA、±50mA，默认单位为mA。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetFIRange(0.05*mA);
```

2.12.8.22 SetIClampH（）

函数功能

设置钳位电流上限值。

使用说明

钳位电流下限值需小于上限值。

函数原型

```
CDIOPPMU_Pin& SetIClampH(const double dHighClampCurrent);
```

参数说明

参数	说明
dHighClampCurrent	钳位电流上限值，可选挡位±5μA，±50μA，±500μA，±5mA，±50mA，单位为A。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetIClampH(0.05*mA);
```

2.12.8.23 SetIClampL（）

函数功能

设置钳位电流下限值。

使用说明

钳位电流下限值需小于上限值。

函数原型

```
CDIOPPMU_Pin& SetIClampL(const double dLowClampCurrent);
```

参数说明

参数	说明
dLowClampCurrent	钳位电流下限值，可选挡位±5μA，±50μA，±500μA，±5mA，±50mA，单位为A。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetIClampL(1.1*mA);
```

2.12.8.24 SetIForce（）

函数功能

设置输出电流值。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetIForce(const double dForceCurrent);
```

参数说明

参数	说明
dForceCurrent	输出电流值，可选挡位±5μA、±50μA、±500μA、±5mA、±50mA，默认单位为A。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetIForce(0.05*mA);
```

2.12.8.25 SetIRange（）

函数功能

设置输出电流的挡位。

使用说明

- 该接口包含了 SetFIRange 及 SetMIRange 函数功能，故不能和 SetFIRange 及 SetMIRange 一起使用。
- 不支持多 Site 处理。

对应关系如下：

参数	para1	para2	para3	para4	para5
FIRange	IR0	IR1	IR2	IR3	IR4
MIRange	MI0	MI1	MI2	MI3	MI4
IMax	±5μA	±50μA	±500μA	±5mA	±50mA

函数原型

```
CDIOPPMU_Pin& SetIRange(const double dIRange);
```

参数说明

参数	说明
dIRange	电流范围，单位：A。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetIRange(5*mA);
```

2.12.8.26 SetMeasureMode（）

函数功能

设置测量模式。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetMeasureMode(PmuDcMeasureMode_E  emMeasureMode);
```

参数说明

参数	说明
emMeasureMode	测量模式，其类型定义详见 PmuDcMeasureMode_E 类型定义。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetMeasureMode(E_DC_FI_MV);
```

2.12.8.27 SetMeasureType（）

函数功能

设置测量类型。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetMeasureType(DcMeasureType_E emMeasureType);
```

参数说明

参数	说明
emMeasureType	测量类型，其类型定义详见 DcMeasureType_E 类型定义。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetMeasureType(E_MEASURE);
```

2.12.8.28 SetMIRange（）

函数功能

设置被测电流的范围。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetMIRange(const double dMeasureCurrentRange);
```

参数说明

参数	说明
dMeasureCurrentRange	测量电流范围。 可选挡位±5μA、±50μA、±500μA、±5mA、±50mA，默认单位为 mA。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetMIRange(0.05*mA);
```

2.12.8.29 SetSampleSize（）

函数功能

设置采样点数。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetSampleSize(const double dSampleSize)
```

参数说明

参数	说明
dSampleSize	采样点数，取值为 2 的 0～12 次方。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetSampleSize(8);
```

2.12.8.30 SetVClampL（）

函数功能

设置钳位电压下限值。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetVClampL(const double dLowClampVoltage);
```

参数说明

参数	说明
dLowClampVoltage	钳位电压下限值，范围为 - 4V－10V，单位为 V。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
```

```
TheInst.DIO().PPMU().Pins(strPinName).SetVClampL(-2*V);
```

2.12.8.31 SetVClampH（）

函数功能

设置钳位电压上限值。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetVClampH(const double dHighClampVoltage);
```

参数说明

参数	说明
dHighClampVoltage	钳位电压上限值，范围为 - 4V－10V，单位为 V。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetVClampH(3.0*V);
```

2.12.8.32 SetVForce（）

函数功能

设置输出电压值。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetVForce(const double dForceVoltage);
```

参数说明

参数	说明
dForceVoltage	输出电压值，范围 - 2V－7V，默认单位为 V。

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetVForce(1.1*V)
```

2.12.8.33 SetWaitTime（）

函数功能

设置读取结果的等待时间。

使用说明

无。

函数原型

```
CDIOPPMU_Pin& SetWaitTime(const double dWaitTimes);
```

参数说明

参数	说明
dWaitTimes	等待时间，单位 s。

各挡位分别对应的最短时间如下：

参数	para1	para2	para3	para4	para5
FIRange	IR0	IR1	IR2	IR3	IR4
MIRange	MI0	MI1	MI2	MI3	MI4
IMax	±5μA	±50μA	±500μA	±5mA	±50mA

参数	para1	para2	para3	para4	para5
等待时间	5ms	500μs	50μs	10μs	10μs

返回值

类型	说明
CDIOPPMU_Pin	返回 CDIOPPMU_Pin 自身对象。

示例

```
string strPinName = "Pin1";
TheInst.DIO().PPMU().Pins(strPinName).SetWaitTime(0.005);
```

2.12.9 TMU

2.12.9.1 类型定义

FreqCountersource_E 类型定义

类型	Enumeration	
描述	测量源通道。	
元素	E_FC_SLOPE_VOH=1	VOH 测量。
	E_FC_SLOPE_VOL	VOL 测量。
头文件	UserDef.h	

2.12.9.2 Apply（）

函数功能

设置通道执行输出动作。

使用说明

无。

函数原型

```
CDIOTMU& Apply()
```

参数说明

无。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().Apply();
```

2.12.9.3 EnablePins（）

函数功能

选择测试通道。

使用说明

无。

函数原型

```
CDIOTMU&EnablePins(const string& strPinList)
```

参数说明

参数	说明
strPinList	Pinlist 的名称。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
string strPinList = "PinName1,PinName2,PinGroup1";
TheInst.DIO().TMU().EnablePins(strPinList);
```

2.12.9.4 Clear（）

函数功能

清零测试逻辑。

使用说明

无。

函数原型

```
CDIOTMU& Clear()
```

参数说明

无。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().clear();
```

2.12.9.5 GetFrequencyCounterTriggerResults（）

函数功能

获取 TriggerFrequencycounter 的测试结果。

使用说明

无。

函数原型

```
uint32_t GetFrequencyCounterTriggerResults(uint32_t uiCount, string strPinname,
map<int, double> & mapResult);
```

参数说明

参数	说明
uiCount	测试次数
strPinname	指定 Pin
mapResult	返回工位号和对应的结果

返回值

参数	说明
uint32_t	0：成功获取

示例

```
map<int, double>  mapResult;
const string pcPinlist = "DIO_P001";
TheInst.DIO().Level().Apply();
TheInst.DIO().TimingBlock().Apply();
    TheInst.DIO().TMU().EnablePins(pcPinlist);
TheInst.DIO().TMU().Clear();
TheInst.DIO().TMU()  .SetInterValMode(E_FC_IVM_TRIG)
                  .SetSource(E_FC_SOURCE_VOH)
                     .SetSlope(E_FC_SLOPE_RISE)
                     .SetFilterTime(0.000)
                  .SetInterValTime(0.001)
                     .Apply();
    TheInst.DIO().Pattern().Start();
    uint32_t uiCounts = 0;
    string strPinme = "DIO_P001";
    uint32_t iRet =
TheInst.DIO().TMU().GetFrequencyCounterTriggerResults(uiCounts, strPinme,
mapResult);
```

2.12.9.6 GetTrigNmuber（）

函数功能

获取 InterValMode 模式为 E_FC_IVM_TRIG 的测量总次数。

使用说明

无。

函数原型

```
uint32_t GetTrigNmuber();
```

返回值

类型	说明
uint32_t	测量总次数。

示例

```
uint32_t uiTrigNmuber = TheInst.DIO().TMU().GetTrigNmuber();
```

2.12.9.7 GetResults（）

函数功能

获取测试结果。

使用说明

无。

函数原型

```
uint32_t GetResults(map<string, map<int, double> > &mFreqCountResult);
```

参数说明

参数	说明
mFreqCountResult	频率的计数。

返回值

参数	说明
uint32_t	测试结果。

示例

```
map<string, map<uint32_t, double> > mFreqCountResult;
uint32_t usize = TheInst.DIO().TMU().GetResults(mFreqCountResult);
```

2.12.9.8 SetMeasureMode（）

函数功能

设置测量模式（高精度/高速）。

使用说明

无。

函数原型

```
CDIOTMU& SetMeasureMode(FreqCounterMeaseureMode_E enMeasureMode);
```

参数说明

参数	说明
enMeasureMode	E_FC_MM_HA=1：高精度。 E_FC_MM_HF：高速，可测 200M 以上。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetMeasureMode(E_FC_MM_HA);
```

2.12.9.9 SetInterValMode（）

函数功能

设置测量窗口模式。

使用说明

无。

函数原型

```
CDIOTMU& SetInterValMode(const FreqCounterInterValMode_E emMode)
```

参数说明

参数	说明
emMode	测量窗口模式。 ● E_FC_IVM_FIXED=1：程序设置测量模式。 ● E_FC_IVM =2：Pattern Trig 微指令测量模式

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetInterValMode(E_FC_IVM_FIXED);
```

2.12.9.10 SetInterValTime（）

函数功能

设置测量窗口宽度。

使用说明

无。

函数原型

```
CDIOTMU& SetInterValTime(const double dInterValTime)
```

参数说明

参数	说明
dInterValTime	窗口宽度，单位：S。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetInterValTime(1);
```

2.12.9.11 SetFilterTime（）

函数功能

设置毛刺滤波窗口宽度参数。

使用说明

无。

函数原型

```
CDIOTMU& SetFilterTime(const double dFilterTime)
```

参数说明

参数	说明
dFilterTime	窗口宽度，单位：S。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetFilterTime(1);
```

2.12.9.12 SetSlope（）

函数功能

设置边沿类型（上升沿/下降沿）。

使用说明

无。

函数原型

```
CDIOTMU& SetSlope(const FreqCounterSlope_E emSlope)
```

参数说明

参数	说明
emSlope	E_FC_SLOPE_RISE=1：上升沿。 E_FC_SLOPE_FALL：下降沿。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetSlope(E_FC_SLOPE_RISE);
```

2.12.9.13 SetSource（）

函数功能

设置测量源通道（VOH 测量/VOL 测量）。

函数原型

```
CDIOTMU& SetSource(const FreqCountersource_E emSource)
```

参数说明

参数	说明
emSource	测量源通道，其类型定义详见 FreqCountersource_E 类型定义。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().SetSource(E_FC_SLOPE_VOH);
```

2.12.9.14 Start（）

函数功能

开始测试。

使用说明

无。

函数原型

```
CDIOTMU& Start()
```

参数说明

无。

返回值

类型	说明
CDIOTMU	返回 CDIOTMU 自身对象。

示例

```
TheInst.DIO().TMU().Start();
```

2.13 TPS8

2.13.1 类型定义

TPS8_EdgeTypeA 类型定义

类型	Enumeration	
描述	start 路触发边沿类型。	
元素	RisingEdgeA = 0	上升沿。
	FallingEdgeA = 1	下降沿。
头文件	UserDef.h	

TPS8_EdgeTypeB 类型定义

类型	Enumeration	
描述	stop 路触发边沿类型。	
元素	RisingEdgeB = 0	上升沿。
	FallingEdgeB = 1	下降沿。
头文件	UserDef.h	

TPS8_TypeAB 类型定义

类型	Enumeration	
描述	选择 A 或 B 通道进行测量。	
元素	GetFrom_A = 0	A 通道。
	GetFrom_B = 1	B 通道。

	GetFrom_AB = 2	A–B 通道。
头文件	UserDef.h	

TPS8_AbsType 类型定义

类型	Enumeration	
描述	TPS8 测量结果的最大、最小值。	
元素	TPS8_Min=0	最小值。
	TPS8_Max=1	最大值。
头文件	UserDef.h	

TPS8_EdgeType 类型定义

类型	Enumeration	
描述	测量边沿类型。	
元素	RisingEdge = 0	返回测到的上升沿边沿数量。
	FallingEdge = 1	返回测到的下降沿边沿数量。
	RisingOrFallingEdge = 2	返回测到的上和下降沿边沿数量。
头文件	UserDef.h	

TPS8_CHANNEL 类型定义

类型	Enumeration	
描述	选择通道内 A、B 信号。	
元素	TPS8_ChannelA = 0	选择 A 通道。
	TPS8_ChannelB = 1	选择 B 通道
头文件	UserDef.h	

TPS8_Impedance 类型定义

类型	Enumeration	
描述	输入阻抗匹配	
元素	Impedance_50Ohm = 0	低输入阻抗，Impedance_50Ohm：50Ω，只适用于高压精测模式。
	Impedance_1MOhm = 1	高输入阻抗，Impedance_1MOhm：1MΩ。
头文件	UserDef.h	

2.13.2 GetAllSinglePinName（）

函数功能

应为获取 TPS8 板卡所有的 Pin 的名字。

使用说明

无。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无

返回值

类型	说明
vector	TPS8 板卡所有的 Pin 的名字。

示例

```
vector<string> strPinName;
strPinName = TheInst.TPS8().GetAllSinglePinName();
```

2.13.3 GetEdgeTms（）

函数功能

获取所有 Site 指定通道指定边沿序号（N1～N2）的时间值，支持 A、B 通道单独测试。

使用说明

此函数需要放在 TPS8_SetMode 之后，建议 SetMode 模式设置为（TPS8_RiseToRise）。

函数原型

```
int GetEdgeTms(const string& strPinName, TPS8_EdgeTypeA emEdgeTypeA, int
nStartNum, TPS8_EdgeTypeB emEdgeTypeB, int nStopNum, TPS8_TypeAB emTypeAB);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emEdgeTypeA	start 路触发边沿类型，其类型定义详见 TPS8_EdgeTypeA 类型定义。
nStartNum	需要获取的起始边沿序号。 ● A 通道：0～32766。 ● B 通道：0～32766。
emEdgeTypeB	stop 路触发边沿类型，其类型定义详见 TPS8_EdgeTypeB 类型定义。
nStopNum	需要获取的结束边沿序号。 ● A 通道：1～32767。 ● B 通道：1～32767。
emTypeAB	选择 A 或 B 通道进行测量，其类型定义详见 TPS8_TypeAB 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nEdgeTms =
TheInst.TPS8().GetEdgeTms(strPinName,RisingEdgeA,1024,RisingEdgeB,2048,GetFrom_A
); //测量 A 通道第 1024 个上升沿到第 2048 个上升沿的时
```

2.13.4 Get_Master_FPGAVer（）

函数功能

获取 TPS8 板卡主机 FPGA 版本号。

使用说明

无。

函数原型

```
int Get_Master_FPGAVer(int nTPS8n, int &nFpgaVer, char *strFpgaVer);
```

参数说明

参数	说明
nTPS8n	板卡号（0）。
nFpgaVer	返回主机 FPGA 版本号的变量。
strFpgaVer	返回主机 FPGA 版本号的变量指针。

返回值

类型	说明
int	该函数可返回板卡主机 FPGA 版本号。

示例

```
int nFpgaVer;
char strFpgaVer[256] = { '\0' };
TheInst.TPS8().Get_Master_FPGAVer(0, nFpgaVer, strFpgaVer);
```

2.13.5 GetMeasResult（）

函数功能

获取 TPS8 测量结果的最大、最小值。

使用说明

- 模式通过 SetMode 函数设置，支持频率（0）、周期（1）、高电平（2）以及低电平测量模式（3）。
- 若无波形输入则返回值为 0。

函数原型

```
int GetMeasResult(const string& strPinName, TPS8_AbsType emType);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emType	TPS8 测量结果的最大、最小值，其类型定义详见 TPS8_AbsType 类型定义。

返回值

类型	说明
int	● 0：无波形输入。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TPS8().SetMode(strPinName, TPS8_Frequency, Impedance_1MOhm,
TPS8_LLVrange, Filter_5ns, 2.5,2.5, HighAccuracy, 0, 1000);
TheSoft.TestProc().DelaymS(3);//加延时确保采到足够信号
TheInst.TPS8().GetMeasResult(strPinName, TPS8_Min);//返回 1000 个波形最小值
```

2.13.6 GetMeasResult（）

函数功能

一次性获取特定测试工位的最大与最小值。

使用说明

无。

函数原型

```
int GetMeasResult(const string& strPinName, int nSite, double &dMax, double
&dMin);
```

参数说明

参数	说明
strPinName	Pin 的名字。
nSite	测试工位。

参数	说明
dMax	用于获取最大值。
dMin	用于获取最小值。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
double dMax = 0.0;
double dMin = 0.0;
const string strPinName = "Pin1";
TheInst.TPS8().SetMode(strPinName, TPS8_Frequency, Impedance_1MOhm,
TPS8_LLVrange, Filter_5ns, 2.5,2.5, HighAccuracy, 0, 1000);
TheSoft.TestProc().DelaymS(3);//加延时确保采到足够信号
TheInst.TPS8().GetMeasResult(strPinName, 1,dMax,dMin);//返回 site1 10000 个波形频率最大和最小值
```

2.13.7 GetPinName（）

函数功能

获取 TPS8 板卡的模块版本信息。

使用说明

无。

函数原型

```
string GetPinName(int nPhyCh);
```

参数说明

参数	说明
nPhyCh	通道号。

返回值

类型	说明
string	Pin 的名字。

示例

```
string strPinName = TheInst.TPS8().GetPinName(0);
```

2.13.8 GetRamResult（）

函数功能

获取 TPS8 单个测量通道的所有采样结果。

使用说明

无。

函数原型

```
int GetRamResult(const string& strPinName, int nSite, int nSampNum,
TPS8_ParaMeasure emMode, double *dDataRead);
```

参数说明

参数	说明
strPinName	测试的次数。
nSite	测试工位。
nSampNum	采样点数。
emMode	测量模式选择，其类型定义详见“0 TPS8_ParaMeasure 类型定义”。 说明 不支持 16、17、18、19 模式。
dDataRead	用于存储采样结果。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
double dDataRead[100] = { 0 };
TheInst.TPS8().GetRamResult(strPinName, 0, 100,TPS8_Frequency,dDataRead);
```

2.13.9 Get_Slave_FPGAVer（）

函数功能

获取 TPS8 板卡从机 FPGA 版本号，该函数可返回两种类型的 FPGA 版本号，int 型和 CString 型。

使用说明

无。

函数原型

```
int Get_Slave_FPGAVer(int nTPS8n, int &nFpgaVer1, char* pcFpgaVer1, int
&nFpgaVer2, char* pcFpgaVer2, int &nFpgaVer3, char* pcFpgaVer3, int &nFpgaVer4,
char* pcFpgaVer4, int &nFpgaVer5, char* pcFpgaVer5, int &nFpgaVer6, char*
pcFpgaVer6, int &nFpgaVer7, char* pcFpgaVer7, int &nFpgaVer8, char* pcFpgaVer8);
```

参数说明

参数	说明
nTPS8n	板卡号（0、1）。
nFpgaVer1	返回从机 1FPGA 版本号的变量。
pcFpgaVer1	返回从机 1FPGA 版本号的变量指针。
nFpgaVer2	返回从机 2FPGA 版本号的变量。
pcFpgaVer2	返回从机 2FPGA 版本号变量指针。
nFpgaVer3	返回从机 3FPGA 版本号的变量。

参数	说明
pcFpgaVer3	返回从机 3FPGA 版本号变量指针。
nFpgaVer4	返回从机 4FPGA 版本号的变量。
pcFpgaVer4	返回从机 4FPGA 版本号变量指针。
nFpgaVer5	返回从机 5FPGA 版本号的变量。
pcFpgaVer5	返回从机 5FPGA 版本号变量指针。
nFpgaVer6	返回从机 6FPGA 版本号的变量。
pcFpgaVer6	返回从机 6FPGA 版本号变量指针。
nFpgaVer7	返回从机 7FPGA 版本号的变量。
pcFpgaVer7	返回从机 7FPGA 版本号变量指针。
nFpgaVer8	返回从机 8FPGA 版本号的变量。
pcFpgaVer8	返回从机 8FPGA 版本号变量指针。

返回值

类型	说明
int	返回值为 0。

示例

```
int
nFpgaVer1,nFpgaVer2,nFpgaVer3,nFpgaVer4,nFpgaVer5,nFpgaVer6,nFpgaVer7,nFpgaVer8;
char strFpgaVer1[]={0};
char strFpgaVer2[]={0};
char strFpgaVer3[]={0};
char strFpgaVer4[]={0};
char strFpgaVer5[]={0};
char strFpgaVer6[]={0};
char strFpgaVer7[]={0};
char strFpgaVer8[]={0};
TheInst.TPS8().Get_Slave_FPGAVer(0,nFpgaVer1,strFpgaVer1,nFpgaVer2,strFpgaVer2,n
FpgaVer3,strFpgaVer3,nFpgaVer4,strFpgaVer4,nFpgaVer5,strFpgaVer5,nFpgaVer6,strFp
gaVer6,nFpgaVer7,strFpgaVer7,nFpgaVer8,strFpgaVer8);
```

2.13.10 GetVer（）

函数功能

获取 TPS8 板卡的模块版本信息。

使用说明

无。

函数原型

```
int GetVer(char*pchVer, int nTPS8n);
```

参数说明

参数	说明
pchVer	返回版本信息的变量指针。
nTPS8n	板卡号（0～1）。

返回值

类型	说明
int	返回值无意义。

示例

```
char chVer []={0};
TheInst.TPS8().GetVer(chVer, 0);
```

2.13.11 Init（）

函数功能

初始化 TPS8 通道。

使用说明

- 断开通道输入继电器，恢复默认状态。
- DAC 输出电平置 0 和 PE 比较电平置 0。

函数原型

```
int Init(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TPS8().Init(strPinName);
```

2.13.12 Measure（）

函数功能

在限定时间内，获取通道测量值，当 nMode=0 时返回实测频率（KHz），其余模式时返回时间（ms）。dTimeout 设置限时测量时间，可缺省，默认不进行限时测量。

使用说明

测量返回值异常基于以下几种情况。

- FPGA 捕获到的波形无 Start 有 Stop，返回 0。
- FPGA 捕获到的波形有 Start，无 Stop，返回极大值。
- FPGA 捕获到的波形无 Start，无 Stop，返回 0。

函数原型

```
int Measure(const string& strPinName, double dTimeout = 0.0);
```

参数说明

参数	说明
strPinName	Pin 的名字。
dTimeout	限时测量时间（ms）。

返回值

类型	说明
int	● FPGA 捕获到的波形无 Start 有 Stop，返回 0。 ● FPGA 捕获到的波形有 Start，无 Stop，返回极大值。 ● FPGA 捕获到的波形无 Start，无 Stop，返回 0。

示例

```
const string strPinName = "Pin1";
TheInst.TPS8().SetMode(strPinName, TPS8_Frequency, Impedance_1MOhm,
TPS8_LLVrange, Filter_5ns, 2.5, 2.5, LowAccuracy);
int Measure = TheInst.TPS8().Measure(strPinName, 5);//限时 5ms 获取测量结果
```

2.13.13 MeasStart（）

函数功能

开始测量函数。

使用说明

该函数应用于 TPS8_SetMode 函数后连续测量相同模式使用，无需重新设置测量模式和继电器切换。

函数原型

```
int MeasStart(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TPS8().SetMode(strPinName,TPS8_Frequency,Impedance_1Mohm,TPS8_LLVrange,pFilter_5ns,2.5,2.5,LowAccuracy);

int Measure1=TheInst.TPS8().Measure(strPinName);//第一次获取测量结果

TheInst.TPS8().MeasStart(strPinName);//开始第二次测量

int Measure2=TheInst.TPS8().Measure(strPinName);//第二次获取测量结果
```

2.13.14 RamRead（）

函数功能

抖频测量功能，支持 A，B 通道单独测试。

使用说明

- 当测试不到值时默认返回 0。
- 此函数需要放在 TPS8_SetMode 之后。使用抖频功能测量频率、周期、高电平、低电平时建议 setmode 模式设置为周期（TPS8_Cycle）。测量沿到沿时建议 setmode 模式设置为（TPS8_RiseToRise）。

函数原型

```
int RamRead(const string& strPinName, TPS8_ParaMeasure emMode, int nOffset);
```

参数说明

参数	说明
strPinName	Pin 的名字。

参数	说明
emMode	测量模式，测量模式类型和枚举变量设置可以参考“0 TPS8_ParaMeasure 类型定义”。 说明 不支持 16、17、18、19 模式。
nOffset	待测量波形的位置。 ● A 通道：0～32767。 ● B 通道：0～32767。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.TPS8().SetMode(strPinName, TPS8_Cycle, Impedance_1MOhm, TPS8_LLVrange,
Filter_5ns, 2.5,2.5, HighAccuracy, 0, 2000);
TheSoft.TestProc().DelaymS(3);//加延时确保采到足够信号
TheInst.TPS8().RamRead(strPinName, TPS8_Frequency, 1000);//获取第 1000 个频率结果
TheInst.TPS8().RamRead(strPinName, TPS8_Cycle, 1024);//获取第 1024 个周期结果
```

2.13.15 SetMode_Simul（）

函数功能

设置 TPS8 通道的同时测量模式。

使用说明

- 仅支持精测。
- 需配合 TPS8_RamRead_Simul 使用

函数原型

```
int SetMode_Simul(const string& strPinName, TPS8_Impedance emIsHImped,
TPS8_Vrange emIsHV, TPS8_Filter emFilterEn, double dTGV1, double dTGV2, double
dTGV3, double dTGV4);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emIsHImped	输入阻抗匹配，通常设置为 Impedance_1Mohm，其类型定义详见 TPS8_Impedance 类型定义。
emIsHV	输入电压类型。 ● 低输入电压：－1V－＋6V。 ● 高输入电压：－5V－＋30V。
emFilterEn	选择滤波系数。固定频率波形滤波系数建议设置为测试频率的 1%。 支持的枚举量如下： Filter _0ns、Filter_5ns、Filter_10ns、Filter_25ns、Filter_50ns、Filter_100ns、Filter_250ns、Filter_500ns、Filter_1us、Filter_2_5us、Filter_5us、Filter_10us、Filter_25us、Filter_50us、Filter_100us、Filter_250us、Filter_500us、Filter_1ms、Filter_2_5ms、Filter_5ms、Filter_10ms、Filter_25ms、Filter_50ms、Filter_100ms。
dTGV1	设置 A 路触发电平 TGV1。 ● TPS8_LLVrange：－1.0V－＋6.0V。 ● TPS8_HHVrange：－5V－＋30V。
dTGV2	设置 A 路触发电平 TGV2。 ● TPS8_LLVrange：－1.0V－＋6.0V。 ● TPS8_HHVrange：－5V－＋30V。
dTGV3	设置 B 路触发电平 TGV1。 ● TPS8_LLVrange：－1.0V－＋6.0V。 ● TPS8_HHVrange：－5V－＋30V。
dTGV4	设置 B 路触发电平 TGV2。 ● TPS8_LLVrange：－1.0V－＋6.0V。 ● TPS8_HHVrange：－5V－＋30V。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode_Simul(strPinName,Impedance_1MOhm,TPS8_LLVrange,Filter_0ns
,2.5,2.5,2.5,5);

//设置 TPS8 通道 0 为精测低压模式,A 路触发电平 TGV1 设为 2.5V,触发电平 TGV2 设为 2.5V,B 路触发电

平 TGV3 设为 2.5V,触发电平 TGV4 设为 5V。
```

2.13.16 RamRead_Simul（）

函数功能

抖频同时测量功能。当测试不到值时默认返回 0。

使用说明

- 此函数需要放在 TPS8_SetMode_Simul 之后。
- 此函数与 TPS8_RamRead 的使用上的区别是 TPS8_RamRead 当为精测时仅支持与 TPS8_SetMode 中设定的 Mode 一致，TPS8_SetMode_Simul 支持任意模式连续获取。

函数原型

```
int RamRead_Simul(const string& strPinName, TPS8_ParaMeasure emMode, int
nOffset, TPS8_CHANNEL emABChannel);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emMode	测量模式，测量模式类型和枚举变量设置可以参考“0 TPS8_ParaMeasure 类型定义”。 **说明** 不支持 16、17、18、19 模式。
nOffset	待测量波形的位置。 ● A 通道：0～4095。 ● B 通道：0～4095。
emABChannel	选择通道内 A、B 信号，缺省时默认值为 A 通道，其类型定义详见 TPS8_CHANNEL 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode_Simul(strPinName,Impedance_1MOhm,TPS8_LLVrange,Filter_0ns
,2.5,2.5,2.5,5);

//设置 TPS8 通道 0 为精测低压模式，A 路触发电平 TGV1 设为 2.5V，触发电平 TGV2 设为 2.5V，B 路触发电

平 TGV3 设为 2.5V，触发电平 TGV4 设为 5V。

Signal in  //波形输入
TheInst.TPS8().RamRead_Simul（strPinName, TPS8_HighLevel, 1000, TPS8_ChannelA）//获取
0 通道 A 路第 1000 个高电平时间。
TTheInst.TPS8().RamRead_Simul（strPinName, TPS8_HighLevel, 1000, TPS8_ChannelB）//获
取 0 通道 B 路第 1000 个高电平时间。
TheInst.TPS8().RamRead_Simul（strPinName, TPS8_Rise, 1000, TPS8_ChannelB）//获取 0 通
道 B 路第 1000 个上升沿时间。
TheInst.TPS8().RamRead_Simul（strPinName, TPS8_RiseToRise, 500, TPS8_ChannelA）//获取
0 通道 A 路第 500 个上升沿到 B 路下一个上升沿的时间差。
```

2.13.17 SetMode（）

TPS8_ParaMeasure 类型定义

类型	Enumeration	
描述	测量模式。	
元素	TPS8_Frequency = 0	测量频率（kHz）。
	TPS8_Cycle = 1	测量周期（ms）。
	TPS8_HighLevel = 2	高电平宽度（ms）。
	TPS8_LowLevel = 3	低电平宽度（ms）。
	TPS8_Rise = 4	上升沿时间（ms）。
	TPS8_Drop = 5	下降沿时间（ms）。
	TPS8_RiseToRise = 6	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_RiseToDrop = 7	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。

	TPS8_DropToRise = 8	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_DropToDrop = 9	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_DutyCycle = 11	测量占空比。
	TPS8_RiseToRise_FF = 16	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_RiseToDrop_FF = 17	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_DropToRise_FF = 18	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_DropToDrop_FF = 19	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_Frequency_Neg = 20	下降沿触发测量频率（kHz）。
	TPS8_Cycle_Neg = 21	下降沿触发测量周期（ms）。
头文件	UserDef.h	

TPS8_ParaMeasure 类型对应的测量模式及其触发模式如下表所示：

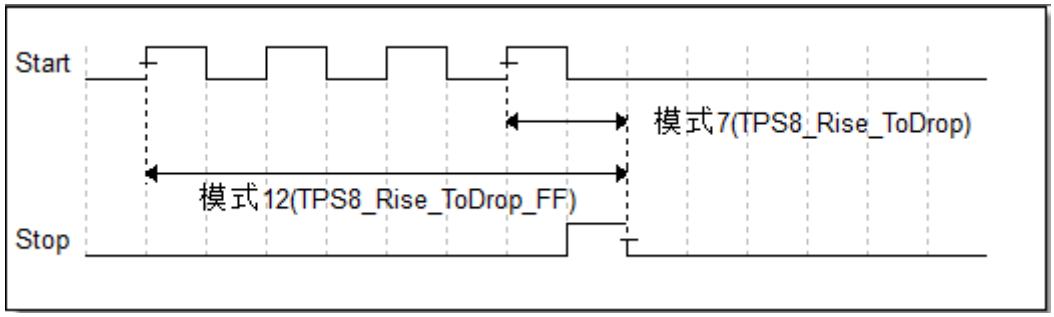
序号	模式	测量内容	触发模式
0	TPS8_Frequency	测量频率（kHz）。	触发与 TGV1 有关，用上升沿测量。
1	TPS8_Cycle	测量周期（ms）。	触发与 TGV1 有关，用上升沿测量。
2	TPS8_HighLevel	高电平宽度（ms）。	触发与 TGV1—>TGV2 有关。
3	TPS8_LowLevel	低电平宽度（ms）。	触发与 TGV1—>TGV2 有关。
4	TPS8_Rise	上升沿时间（ms）。	触发与 TGV1—>TGV2 有关，设置时 TGV1<TGV2。
5	TPS8_Drop	下降沿时间（ms）。	触发与 TGV1—>TGV2 有关，设置时 TGV1>TGV2。

序号	模式	测量内容	触发模式
6	TPS8_RiseToRise	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个上升沿 PP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
7	TPS8_RiseToDrop	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个下降沿 PN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
8	TPS8_DropToRise	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个下降沿最近 Start 下降沿到 Stop 第一个上升沿 NP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号
9	TPS8_DropToDrop	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个下降沿最近的 Start 下降沿到 Stop 第一个下降沿 NN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
10	TPS8_DutyCycle	测量占空比。	/

序号	模式	测量内容	触发模式
11	TPS8_RiseToRise_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个上升沿到 Stop 第一个上升沿 PP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
12	TPS8_RiseToDrop_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个上升沿到 Stop 第一个下降沿 PN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
13	TPS8_DropToRise_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个下降沿到 Stop 第一个上升沿 NP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
14	TPS8_DropToDrop_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个下降沿到 Stop 第一个下降沿 NN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
15	TPS8_Frequency_Neg	测量频率（kHz）。	测量频率（kHz），触发与 TGV1 有关，用下降沿测量。 触发与 TGV1 有关，用下降沿测量。
16	TPS8_Cycle_Neg	测量周期（ms）。	测量周期（ms），触发与 TGV1 有关，用下降沿测量。 触发与 TGV1 有关，用下降沿测量。

序号 6～9 模式和 11～14 模式区别参考下图模式 7 和 12 的时序图。

图2-1 时序图



2.13.17.1 TPS8_ParaMeasure 类型定义

类型	Enumeration	
描述	测量模式。	
元素	TPS8_Frequency = 0	上升沿触发测量频率（kHz）。
	TPS8_Cycle = 1	上升沿触发测量周期（ms）。
	TPS8_HighLevel = 2	高电平宽度（ms）。
	TPS8_LowLevel = 3	低电平宽度（ms）。
	TPS8_Rise = 4	上升沿时间（ms）。
	TPS8_Drop = 5	下降沿时间（ms）。
	TPS8_RiseToRise = 6	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_RiseToDrop = 7	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_DropToRise = 8	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_DropToDrop = 9	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。
	TPS8_DutyCycle = 11	测量占空比。
	TPS8_RiseToRise_FF = 16	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_RiseToDrop_FF = 17	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_DropToRise_FF = 18	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。

	TPS8_DropToDrop_FF = 19	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。
	TPS8_Frequency_Neg = 20	下降沿触发测量频率（kHz）。
	TPS8_Cycle_Neg = 21	下降沿触发测量周期（ms）。
头文件	UserDef.h	

TPS8_ParaMeasure 类型对应的测量模式及其触发模式如下表所示：

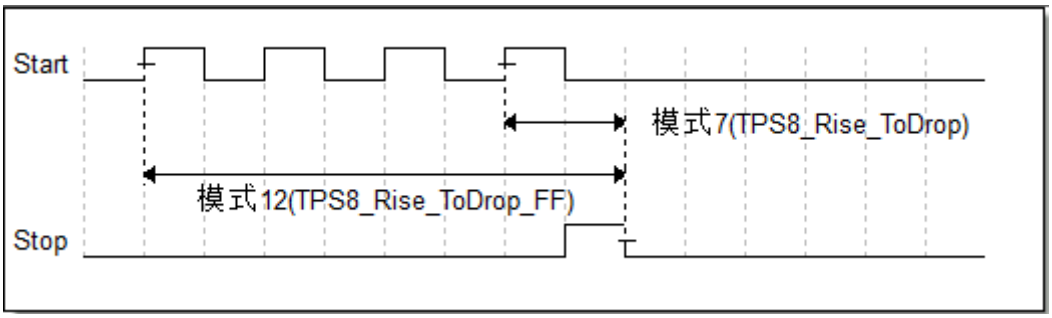
序号	模式	测量内容	触发模式
0	TPS8_Frequency	测量频率（kHz）。	触发与 TGV1 有关，用上升沿测量。
1	TPS8_Cycle	测量周期（ms）。	触发与 TGV1 有关，用上升沿测量。
2	TPS8_HighLevel	高电平宽度（ms）。	触发与 TGV1—>TGV2 有关。
3	TPS8_LowLevel	低电平宽度（ms）。	触发与 TGV1—>TGV2 有关。
4	TPS8_Rise	上升沿时间（ms）。	触发与 TGV1—>TGV2 有关，设置时 TGV1<TGV2。
5	TPS8_Drop	下降沿时间（ms）。	触发与 TGV1—>TGV2 有关，设置时 TGV1>TGV2。
6	TPS8_RiseToRise	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个上升沿 PP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。

序号	模式	测量内容	触发模式
7	TPS8_RiseToDrop	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个上升沿最近的 Start 上升沿到 Stop 第一个下降沿 PN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
8	TPS8_DropToRise	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个下降沿最近 Start 下降沿到 Stop 第一个上升沿 NP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号
9	TPS8_DropToDrop	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间）。	距离 Stop 第一个下降沿最近的 Start 下降沿到 Stop 第一个下降沿 NN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
10	TPS8_DutyCycle	测量占空比。	/
11	TPS8_RiseToRise_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个上升沿到 Stop 第一个上升沿 PP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
12	TPS8_RiseToDrop_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个上升沿到 Stop 第一个下降沿 PN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。

序号	模式	测量内容	触发模式
13	TPS8_DropToRise_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个下降沿到 Stop 第一个上升沿 NP。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
14	TPS8_DropToDrop_FF	通道内 Start 信号、Stop 信号触发时间差（测两个不同事件触发延时时间，Start 第一个信号沿到 Stop 第一个信号沿）。	Start 第一个下降沿到 Stop 第一个下降沿 NN。 触发与 TGV1—>TGV2 有关，TGV1 设置通道 Start 信号，TGV2 设置通道 Stop 信号。
15	TPS8_Frequency_Neg	测量频率（kHz）。	测量频率（kHz），触发与 TGV1 有关，用下降沿测量。 触发与 TGV1 有关，用下降沿测量。
16	TPS8_Cycle_Neg	测量周期（ms）。	测量周期（ms），触发与 TGV1 有关，用下降沿测量。 触发与 TGV1 有关，用下降沿测量。

序号 6～9 模式和 11～14 模式区别参考下图模式 7 和 12 的时序图。

图2-2 时序图



2.13.17.2 函数说明

函数功能

设置 TPS8 通道的测量模式。

使用说明

非规则波形测试详细函数使用说明参见“2.13.20 不规则波形测量函数使用实例”。

函数原型

```
int SetMode(const string& strPinName, TPS8_ParaMeasure emMode, TPS8_Impedance emIsHImped, TPS8_Vrange emIsHV, TPS8_Filter emFilterEn,double dTGV1, double dTGV2, TPS8_MeasType emValLimit, int nStartNum = 0, int nSampleN = 1, TPS8_CHANNEL emABChannel = TPS8_ChannelA);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emMode	测量模式，TPS8_ParaMeasure 详细说明请参见“0 TPS8_ParaMeasure 类型定义”。
emIsHImped	输入阻抗匹配，通常设置为 Impedance_1Mohm，其类型定义详见 TPS8_Impedance 类型定义。
emIsHV	输入电压类型。 ● 粗测低输入电压，－10V–+10V。高输入电压，－50V–+50V。 ● 精测低输入电压：－1V–+6V。高输入电压：－5V–+30V。 ● TPS8_LLVrange：低压（A 通道）低压（B 通道）。 ● TPS8_HHVrange：高压（A 通道）高压（B 通道）。 ● TPS8_LHVrange：低压（A 通道）高压（B 通道）。 ● TPS8_HLVrange：高压（A 通道）低压（B 通道）。 **说明** TPS8_LHVrange，TPS8_HLVrange 只支持沿到沿（“0 TPS8_ParaMeasure 类型定义”章节中序号 6～9，11～14）模式测量。
emFilterEn	选择滤波系数。固定周期波形滤波系数建议设置为测试周期的 1%。 支持的枚举量如下： Filter _0ns、Filter_5ns、Filter_10ns、Filter_25ns、Filter_50ns、Filter_100ns、Filter_250ns、Filter_500ns、Filter_1us、Filter_2_5us、Filter_5us、Filter_10us、Filter_25us、Filter_50us、Filter_100us、Filter_250us、Filter_500us、Filter_1ms、Filter_2_5ms、Filter_5ms、Filter_10ms、Filter_25ms、Filter_50ms、Filter_100ms。

参数	说明
dTGV1	● 粗测 ♦ －10V–+10V（TPS8_LLVrange）。 ♦ －50V–+50V（TPS8_HHVrange）。 ● 精测 ♦ －1.0V–+6.0V（TPS8_LLVrange）。 ♦ －5V–+30V（TPS8_HHVrange）。
dTGV2	● 粗测 ♦ –10V– +10V（TPS8_LLVrange）。 ♦ –50V– +50V（TPS8_HHVrange）。 ● 精测 ♦ －1.0V–+6.0V（TPS8_LLVrange） ♦ –5V – +30V（TPS8_HHVrange）。 未使用时，建议设置与 TGV1 相同 ValLimit。
emValLimit	设置精测、粗测。 ● LowAccuracy=0：选择粗测。 ● HighAccuracy=1：选择精测。
nStartNum	设置 Start 后第 StartNum 个周期的时间参数，设置数值参考如下。 ● A 通道：0～32767。 ● B 通道：0～32767。 可缺省，默认为 0。 说明 不支持沿到沿模式。
nSampleN	设置均值个数，均值个数＋StartNumber＜32768 且需要小于待测波形总数。 可缺省，默认为 1。 ● A 通道：1～32767。 ● B 通道：1～32767。 说明 不支持沿到沿模式。
emABChannel	通道内 A、B 信号选择。缺省时默认值为 A 通道。 TPS8_CHANNE 的枚举定义如下： ● TPS8_ChannelA=0：选择 A 通道。 ● TPS8_ChannelB=1：选择 B 通道。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName, TPS8_Frequency, Impedance_1MOhm,
TPS8_LLVrange, Filter_5ns, 2.5,2.5, LowAccuracy, 1, 100, TPS8_ChannelA);//A 通道测量 100 个波形频率均值,以第一个周期为起始点,粗测低压,触发电压 2.5V。
```

2.13.18 WaveCountStart（）

函数功能

设置边沿计数的总计数时长和边沿类型，清除之前的波形数据。

使用说明

- 此函数需要放在波形到来之前，TPS8_SetMode 之后。
- 调用函数后等待波形的延时需不小于设置的采样时间（dMeasTime），以确保波形均被抓取。

函数原型

```
int WaveCountStart(const string& strPinName, double dMeasTime, TPS8_EdgeType emEdgeType = RisingEdge);
```

参数说明

参数	说明
strPinName	通道号（0、1、2、3…15）。
dMeasTime	总计数时长。单位为 ms，数值为 0.01～2000，时间为 0.01ms～2s。
emEdgeType	测量边沿类型，其类型定义详见 TPS8_EdgeType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";

TheInst.TPS8().WaveCountStart(strPinName,0.1,RisingEdge);//测量 0.1ms 内上升沿边沿数

量
```

2.13.19 WaveCountMeasure（）

函数功能

返回计数边沿的数量。

使用说明

无。

函数原型

```
int WaveCountMeasure(const string& strPinName);
```

参数说明

参数	说明
strPinName	Pin 的名字。

返回值

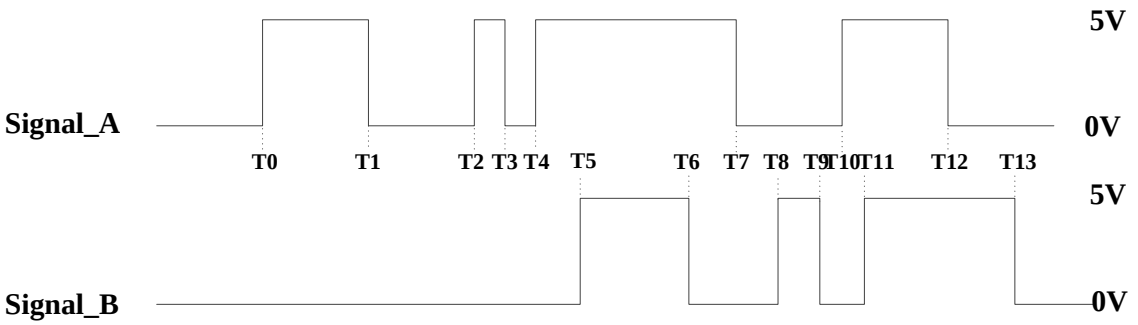
类型	说明
int	● 0：正常。 ● －1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
int nWave = TheInst.TPS8().WaveCountMeasure(strPinName);
```

2.13.20 不规则波形测量函数使用实例

图2-3 不规则图形波形



2.13.20.2 测量非规则波形高电平时间

测量 Signal_A T4→T7 的时间（即第 2 个高电平时间）。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_HighLevel,Impedance_1MOhm,TPS8_LLVrange,
Filter_0ns,1,1,HighAccuracy,2,1);
//（测量第 2 个高电平，TGV1 触发电压 1V，TGV2 触发电平 1V，低压通道高精度测量，只获取 1 个测量结果求
平均值）
Delay_ms();
//（待测信号输入）
TheInst.TPS8().Measure(strPinName );
//（获取测量结果）
```

2.13.20.3 测量非规则波形周期的均值

测量 Signal_A T0→T10 之间波形的平均周期时间。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_Cycle,Impedance_1MOhm,TPS8_LLVrange,Filt
er_0ns,2,2,HighAccuracy,0,3);
//（测量第 0 个周期，TGV1 触发电压 2V，TGV2 触发电平 2V，低压通道高精度测量，3 个周期求取平均值）
Delay_ms();//（待测信号输入）
TheInst.TPS8().Measure(strPinName );
//（获取测量结果）
```

2.13.20.4 测量 A 信号的第一个上升沿到 B 信号第一个上升沿的时间差

测量 Signal_A、Siganl_B T0→T5 的上升沿到上升沿的时间差。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_RiseToRise_FF,Impedance_1MOhm,TPS8_LLVra
nge,Filter_0ns,2,2,HighAccuracy,0,1);

//(测量 A、B 信号的 PP,A 信号触发电压 2V,B 信号触发电平 2V,,低压通道高精度测量,只获取 1 个测量结

果求平均值)

Delay_ms();

//(待测信号输入)

TheInst.TPS8().Measure(strPinName );

//(获取测量结果)
```

2.13.20.5 测量 A 信号距离 B 信号第一个上升沿最近的上升沿的时间差

测量 Signal_A、Siganl_B T4→T5 的上升沿到上升沿时间差。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_RiseToRise,Impedance_1MOhm,TPS8_LLVrange
,Filter_0ns,2,4,HighAccuracy,0,1);

//(测量 A、B 信号的 PP,A 信号触发电压 2V,B 信号触发电平 4V,,低压通道高精度测量,只获取 1 个测量结

果求平均值)

Delay_ms();

//(待测信号输入)

TheInst.TPS8().Measure(strPinName);

//(获取测量结果)
```

2.13.20.6 测量 A 信号距离 B 信号第一个下降沿最近的下降沿的时间差

测量 Signal_A、Siganl_B T3→T6 的下降沿到下降沿时间差。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_DropToDrop,Impedance_1MOhm,TPS8_LLVrange
,Filter_0ns,4,1,nHighAccuracy,0,1);

//(测量 A、B 信号的 NN,A 信号触发电压 4V,B 信号触发电平 1V,低压通道高精度测量,只获取 1 个测量结

果求平均值)

Delay_ms();

//(待测信号输入)

TheInst.TPS8().Measure(strPinName);

//(获取测量结果)
```

2.13.20.7 波形计数功能

测量 Signal_A 下降沿的个数。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_Cycle,Impedance_1MOhm,TPS8_LLVrange,Filter_0ns,2,2,HighAccuracy,0,1);
//（测量 A 信号的下降沿，TGV1 触发电压 2V，TGV2 触发电平 2V，低压通道高精度测量，只获取 1 个测量结果求平均值）
TheInst.TPS8().WaveCountStart（strPinName ,T13,FallingEdge）;
//（测量 T13 时间内下降沿个数）
Delay_ms();
//（待测信号输入）
TheInst.TPS8().WaveCountMeasure(strPinName);
//（获取下降沿个数）
```

2.13.20.8 波形存储测量功能

连续获取 Signal_A 不同周期时间参数。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_Cycle,Impedance_1MOhm,TPS8_LLVrange,Filter_0ns,,2,2,HighAccuracy,0,4);
//（测量 A 信号的周期，TGV1 触发电压 2V，TGV2 触发电平 2V，低压通道高精度测量，波形最大测量量 4 个（波形存储测量模式））
Delay_ms();
//（待测信号输入）
TheInst.TPS8().RamRead(strPinName , TPS8_Cycle, 0);
//（获取第 0 个周期时间，T0→T2）
TheInst.TPS8().RamRead(strPinName , TPS8_Cycle, 1);
//（获取第 1 个周期时间，T2→T4）
TheInst.TPS8().RamRead(strPinName , TPS8_Cycle, 2);
//（获取第 2 个周期时间，T4→T10）
```

2.13.20.9 波形边沿测量功能

测量 Signal_A T2→T10 的时间差。

```
const string strPinName = "Pin0";
TheInst.TPS8().SetMode(strPinName ,TPS8_HighLevel,Impedance_1MOhm,TPS8_LLVrange,
Filter_0ns,2,2,HighAccuracy,0,0);

//（测量 A 信号的高电平时间，TGV1 触发电压 2V，TGV2 触发电平 2V，低压通道高精度测量）

Signal Input

//（待测信号输入）

TheInst.TPS8().GetEdgeTms(strPinName ,RisingEdgeA,1,RisingEdgeB,3,GetFrom_A);

//（获取 A 路第 1 个上升沿 T2 到第 3 个上升沿 T10 的时间）

极值测量功能

TheInst.TPS8().SetMode(strPinName ,TPS8_HighLevel,TPS8_HImped,TPS8_LLVrange,Filt
er_0ns,2,2,HighAccuracy,0,1000);

//（测量 A 信号的高电平时间，TGV1 触发电压 2V，TGV2 触发电平 2V，低压通道高精度测量）

Signal Input;

//（待测信号输入）

TheInst.TPS8().GetMeasResult(strPinName ,TPS8_Max);

//（获取 A 路测量高电平 1000 次的极大值）

TheInst.TPS8().GetMeasResult(strPinName ,TPS8_Min);

//（获取 A 路测量高电平 1000 次的极小值）
```

2.14 HPS100

2.14.1 GetAllSinglePinName（）

函数功能

获取 HPS100 板卡所有的 Pin 的名字。

函数原型

```
vector<string> GetAllSinglePinName();
```

参数说明

无。

返回值

HPS100 板卡所有的 Pin 的名字。

示例

```
vector<string> strPinName;
strPinName = TheInst.HPS100().GetAllSinglePinName();
```

2.14.2 HPS100-DCVI

2.14.2.1 类型定义

emHPS100OutMode 类型定义

类型	Enumeration	
描述	工作模式	
元素	OUTMODE_DC_FV = 0	直流输出电压。
	OUTMODE_DC_FI = 1	直流输出电流。
	OUTMODE_PULSE_FV = 2	脉冲输出电压。
	OUTMODE_PULSE_FI = 3	脉冲输出电流。
	OUTMODE_HPS100_INIT = 4	初始化。
头文件	UserDef.h	

emHPS100_Quadrant 类型定义

类型	Enumeration	
描述	输出或测量象限。	
元素	HPS100_Quadrant1 = 1	在 1 象限进行输出或测量。
	HPS100_Quadrant2	在 2 象限进行输出或测量。
	HPS100_Quadrant3	在 3 象限进行输出或测量。
	HPS100_Quadrant4	在 4 象限进行输出或测量。
头文件	UserDef.h	

emHPS100_VRange 类型定义

类型	Enumeration	
描述	电压输出或测量挡位。	
元素	HPS100_3V = 3	3V 挡位。
	HPS100_10V = 10	10V 挡位。
	HPS100_30V = 30	30V 挡位。
	HPS100_60V = 60	60V 挡位。
	HPS100_120V = 120	120V 挡位。
头文件	UserDef.h	

emHPS100_IRange 类型定义

类型	Enumeration	
描述	电流输出或测量挡位。	
元素	HPS100_30uA = -30	30μA 挡位。
	HPS100_3mA = 3	3mA 挡位。
	HPS100_300mA = 300	300mA 挡位。
	HPS100_2A = 2000	2A 挡位。
	HPS100_30A = 30000	30A 挡位。
	HPS100_100A = 100000	100A 挡位。
头文件	UserDef.h	

emVI_ResType 类型定义

类型	Enumeration	
描述	负载阻值。	
元素	emRes_open = 0	开路。
	emRes_100K	1000000 欧姆。
	emRes_1K	1000 欧姆。
	emRes_10	10 欧姆。
	emRes_1	1 欧姆。
	emRes_0_1	0.1 欧姆。
	emRes_0_02	0.02 欧姆。
头文件	UserDef.h	

2.14.2.2 Init（）

函数功能

对板卡进行初始化，默认设置挡位为 3V/3mA，断开输出继电器。

使用说明

用于 HPS100 启动或测量结束时，对通道进行初始化设置。

函数原型

```
int Init(const string& pPinName);
```

参数说明

参数	说明
pPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
const string strPinName;
TheInst.HPS100().DCVI().Init(strPinName);
```

2.14.2.3 CapLoad（）

函数功能

根据 SetMode 的输出设置，缓慢输出电源通道的输出值，输出建立时间为 dTms。

使用说明

该函数与 SetOutValSlow 函数不同的是，此函数无需设置起始输出值（系统内部会自动获取）。

函数原型

```
int CapLoad(const string& strPinName, double dEndOutVal, double dTms);
```

参数说明

参数	说明
strPinName	Pin 的名称。
dEndOutVal	源输出的终点值。 ● PoutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压输出值（－50－＋50）V。 ● PoutMode= OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电流输出值（-10000～＋10000）mA。

参数	说明
dTms	源输出值变化到终点值的建立时间（ms）。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压值变化到 EndOutVal 所需的时间。 ● POutMode= OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电流值变化到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string pPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(pPinName, OUTMODE_DC_FV, HPS100_3V, HPS100_3mA,
10.0, -10.0, HPS100_Quadrant1);
TheInst.HPS100().DCVI().CapLoad(pPinName, 5, 3);
```

2.14.2.4 ConnectDcc2Ext（）

函数功能

连接校准版。

使用说明

无。

函数原型

```
int ConnectDcc2Ext(const int& nPhyCh);
```

参数说明

参数	说明
nPhyCh	物理通道。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
TheInst.HPS100().DCVI().ConnectDcc2Ext(0);
```

2.14.2.5 DisConnectDcc2Ext（）

函数功能

取消连接校准版。

使用说明

无。

函数原型

```
int DisConnectDcc2Ext(const int& nPhyCh);
```

参数说明

参数	说明
nPhyCh	物理通道。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
TheInst.HPS100().DCVI().DisConnectDcc2Ext(0);
```

2.14.2.6 GetAverResult（）

函数功能

获取板卡通道测量结果的平均值。

使用说明

无。

函数原型

```
int GetAverResult(const string& strPinName, emVI_MeasType emType);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emType	选择测量类型，其类型定义详见 emVI_MeasType 类型定义。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().MeasureVI(strPinName, emVI_MV, 100, 0.01, false);
TheInst.HPS100().DCVI().GetAverResult(strPinName,emVI_MV);
```

2.14.2.7 GetCHCondition（）

函数功能

获取通道状态和信息。

使用说明

无。

函数原型

```
int GetCHCondition(int nSite, const string& pPinName, int &POutMode, double &dVRange, double &dIRange, double &dIVClampP, double &dIVClampN, double &dOutVal, int &nRelayIsOn);
```

参数说明

参数	说明
nSite	测试工位。
pPinName	Pin 的名字。
POutMode	获取电源工作模式。
dVRange	获取电压输出或测量挡位。
dIRange	获取电流输出或测量挡位。
dIVClampP	获取电源正钳位设定值。
dIVClampN	获取电源负钳位设定值。
dOutVal	获取电源输出值。
nRelayIsOn	获取输出继电器状态。 ● 0（OFF 关断）。 ● 1（ON 闭合）。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
int POutMode, nRelayIsOn;
double dVRange, dIRange, dIVClampP, dIVClampN, dOutVal;
TheInst.HPS100().DCVI().GetCHCondition(0, strPinName, POutMode, dVRange, dIRange, dIVClampP, dIVClampN, dOutVal, nRelayIsOn);
```

2.14.2.8 GetMeasResult（）

函数功能

获取电压或电流的测量结果。

使用说明

测量模式可选平均值、最大值或最小值测量。

函数原型

```
int GetMeasResult(const string& pPinName, emVI_MeasType emType, emVI_MeasMode emMode, int nIsShowData);
```

参数说明

参数	说明
pPinName	Pin 的名称。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	预留参数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().MeasureVI(strPinName, emVI_MI, 100, 0.01);
TheInst.HPS100().DCVI().GetMeasResult(strPinName, emVI_MI, emVI_Aver, 0);
```

2.14.2.9 GetMeasResult（）

函数功能

用于获取测量电流或电压的所有点数的测量值。

使用说明

无。

函数原型

```
int GetMeasResult(const string &strPinName, int nSite, emVI_MeasType emType,
double *dDataRead);
```

参数说明

参数	说明
strPinName	Pin 的名称。
nSite	对应工位号。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dDataRead	对应通道的电压或电流数据。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
double dDataRead[25] = {0};
TheInst.HPS100().DCVI().SetMode(strPinName,
OUTMODE_DC_FV,HPS100_10V,HPS100_10mA);
TheInst.HPS100().DCVI().SetOutVal(strPinName,5);
TheInst.HPS100().DCVI().MeasureVI(strPinName,25,0.01);
TheInst.HPS100().DCVI().GetMeasResult(strPinName, 0, emVI_MV, dDataRead);
```

2.14.2.10 GetPluseMeasResult（）

函数功能

获取脉冲测量的结果。

使用说明

无。

函数原型

```
int GetPluseMeasResult(const string& strPinName, emVI_MeasType emType,
emVI_MeasMode emMode, int nIsShowData);
```

参数说明

参数	说明
strPinName	Pin 的名字。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
nIsShowData	是否显示数据。 ● 0：不显示数据。 ● 1：显示数据。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().MeasureVI(strPinName, emVI_MV, 100, 0.01, false);
TheInst.HPS100().DCVI().GetPluseMeasResult(strPinName, emVI_MV, emVI_Aver,0);
```

2.14.2.11 MeasureI（）

函数功能

测量通道输出电流（mA），采样次数（25），采样时间间隔（0.01ms）固定，快速测量且返回平均值。

使用说明

无。

函数原型

```
int MeasureI(const string& pPinName);
```

参数说明

参数	说明
pPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string pPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(strPinName, OUTMODE_DC_FV, HPS100_3V,
HPS100_3mA, 10.0, -10.0, HPS100_Quadrant1);
TheInst.HPS100().DCVI().SetOutVal(strPinName, 5);
TheInst.HPS100().DCVI().MeasureI(pPinName);
```

2.14.2.12 MeasureISamp（）

函数功能

电流值测量。

使用说明

- 测量模式可选平均值、最大值或最小值测量。
- 可设置采样间隔时间，采样点数。

函数原型

```
int MeasureISamp(const string& pPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
pPinName	Pin 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，取值范围：0.01ms～20ms。
nSampleN	采样点数，取值范围：1～2048。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().MeasureISamp(strPinName, emVI_Aver, 0.01, 100);
```

2.14.2.13 MeasureV（）

函数功能

测量通道输出电压（V），采样次数（25），采样时间间隔（0.01ms）固定，且返回平均值。

使用说明

无。

函数原型

```
int MeasureV(const string& pPinName);
```

参数说明

参数	说明
pPinName	Pin 的名称。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string pPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(strPinName, OUTMODE_DC_FI, HPS100_3V,
HPS100_3mA, 10.0, -10.0, HPS100_Quadrant1);
TheInst.HPS100().DCVI().SetOutVal(strPinName, 5);
TheInst.HPS100().DCVI().MeasureV(pPinName);
```

2.14.2.14 MeasureVI（）

函数功能

进行 VI 源测量。

使用说明

无。

函数原型

```
int MeasureVI(const string& pPinName, emVI_MeasType emType, uint32_t uSampNum,
double dSampleTms, bool bIsDelay = true);
```

参数说明

参数	说明
pPinName	Pin 的名称。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
uSampNum	采样点数。
dSampTms	采样间隔，单位为 ms。
bIsDelay	是否延时，选择 true 或者 false，此处需要进行缺省处理，默认为 true，客户可选择使用 false。 ● true：延时。 ● false：不延时。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(strPinName, OUTMODE_DC_FV, HPS100_3V,
HPS100_3mA, 10.0, -10.0, HPS100_Quadrant1);
TheInst.HPS100().DCVI().SetOutVal(strPinName, 5);
TheInst.HPS100().DCVI().MeasureVI(strPinName, emVI_MI, 100, 0.01, false);
```

2.14.2.15 MeasureVIPulse（）

函数功能

进行 VI 源脉冲测量。

使用说明

需要在 SetOutValue()之前使用。

函数原型

```
int MeasureVIPulse(const string& pPinName, emVI_MeasType emType, double
dMeasDelay, uint32_t nSampNum, double dSampTms);
```

参数说明

参数	说明
pPinName	Pin 的名称。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
dMeasDelay	采样延时，指设置开始脉冲到开始采样的延时，单位为 ms。
nSampNum	采样点数。
dSampTms	采样间隔，单位为 ms。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetPulseCnt(strPinName,1);
TheInst.HPS100().DCVI().SetPulseCycleTime(strPinName, 20000, 2000000);
TheInst.HPS100().DCVI().MeasureVIPulse(strPinName, emVI_MV, 10, 100, 0.1);
```

2.14.2.16 MeasureVSamp（）

函数功能

电压值测量。

使用说明

- 测量模式可选平均值、最大值或最小值。
- 可设置采样间隔时间、采样点数。

函数原型

```
int MeasureVSamp(const string& pPinName, emVI_MeasMode emMode, double
dSampleTms, int nSampleN);
```

参数说明

参数	说明
pPinName	Pin 的名称。
emMode	测量模式，其类型定义详见 emVI_MeasMode 类型定义。
dSampleTms	采样间隔时间，取值范围：0.01ms～20ms。
nSampleN	采样点数，取值范围：1～2048。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
const string strPinName = "Pin1";
TheInst.HPS100().DCVI().MeasureVSamp(strPinName, emVI_Aver, 0.01, 100);
```

2.14.2.17 SetMeasAvg（）

函数功能

设置脉冲测量平均滤波器的参数。

使用说明

无。

函数原型

```
int SetMeasAvg(const string& pPinName, int nNSample);
```

参数说明

参数	说明
pPinName	Pin 的名称。
nNSample	点个数。可设置为 0～4，实际设置值 2^N，如 nNSample=2，抽取次数=4。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetMeasAvg(strPinName, 2);
```

2.14.2.18 SetMode（）

函数功能

设置 HPS100 电源通道工作模式，闭合通道输出继电器。

使用说明

无。

函数原型

```
int SetMode(const string& pPinName, emHPS100OutMode POutMode, emHPS100_VRange
VRange, emHPS100_IRange IRange, double dIVClampP, double dIVClampN,
emHPS100_Quadrant Quadrant = emHPS100_Quadrant::HPS100_Quadrant1);
```

参数说明

参数	说明
pPinName	Pin 的名称。
POutMode	选择工作模式，其类型定义详见 emHPS100OutMode 类型定义。

参数	说明
VRange	电压输出或测量挡位。 当 POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压输出挡位设定值，可选挡位 HPS100_3V，HPS100_10V，HPS100_30V，HPS100_60V，HPS100_120V。
IRange	电流输出或测量挡位。 当 POutMode=OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电流输出挡位设定值。 ● DC 模式下可选挡位 HPS100_30uA、HPS100_3mA、HPS100_300mA、HPS100_2A； ● 脉冲模式下可选挡位 HPS100_30uA、HPS100_3mA、HPS100_300mA、HPS100_2A、HPS100_30A、HPS100_100A。
dIVClampP	电源正钳位设置。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电流正钳位设定值，可选择当前电流测量挡位内的任意值。 ● POutMode= OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电压正钳位设定值，可选择当前电压测量挡位内的任意值。
dIVClampN	电源负钳位设置。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电流负钳位设定值，可选择当前电流测量挡位内的任意值。 ● POutMode= OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电压负钳位设定值，可选择当前电压测量挡位内的任意值。
Quadrant	输出或测量象限（缺省时使用默认值 HPS100_Quadrant1），其类型定义详见 emHPS100_Quadrant 类型定义。

返回值

类型	说明
int	● 0：正常。 ● －1：Pin 不存在。 ● －2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
```

```
 TheInst.HPS100().DCVI().SetMode(strPinName, OUTMODE_DC_FV, HPS100_3V,
HPS100_3mA, 10.0, -10.0, HPS100_Quadrant1);
```

2.14.2.19 SetOutVal（）

函数功能

根据 SetMode 设置，设置电源通道的输出值。

使用说明

- 当设置为电压源模式时，dOutVal 指输出电压。
- 当设置为电流模式时，dOutVal 指输出电流。
- 若 SetMode 设置为 OUTMODE_PULSE_FV 或 OUTMODE_PULSE_FI 时，使用该函数输出 0，不会启动 DIG，读取值为 0。

函数原型

```
int SetOutVal(const string& pPinName, double dOutVal);
```

参数说明

参数	说明
pPinName	Pin 的名称。
dOutVal	输出值。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
 string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(strPinName, OUTMODE_DC_FV, HPS100_3V,
HPS100_3mA, 10.0, -10.0, HPS100_Quadrant1);
 TheInst.HPS100().DCVI().SetOutVal(strPinName, 5);
```

2.14.2.20 SetOutValSlow（）

函数功能

根据 SetMode 设置，缓慢输出电源通道的输出值，建立时间为 dTms。

使用说明

- 当设置为电压源模式时，StartOutVal 指输出的起始电压，EndOutVal 指输出的终点电压。
- 当设置为电流源模式时，StartOutVal 指输出的起始电流，EndOutVal 指输出的终点电流。

函数原型

```
int SetOutValSlow(const string& pPinName, double dStartOutVal, double
dEndOutVal, double dTms);
```

参数说明

参数	说明
pPinName	Pin 的名称。
dStartOutVal	源输出的起始值。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压输出值（－60V－＋120.0V）。 ● POutMode= OUTMODE_DC_FI，为电流输出值（－2A～＋2A）。 ● POutMode= OUTMODE_PULSE_FI 时，为电流输出值（–100A～＋100A）。
dEndOutVal	源输出的终点值。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压输出值（－60V－＋120.0V）。 ● POutMode= OUTMODE_DC_FI，为电流输出值（－2A～＋2A）。 ● POutMode= OUTMODE_PULSE_FI 时，为电流输出值（－100A～＋100A）。
dTms	从起始值到终点值的建立时间（ms）。 ● POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压值从 StartOutVal 到 EndOutVal 所需的时间。 ● POutMode= OUTMODE_DC_FI 或 OUTMODE_PULSE_FI 时，为电流值从 StartOutVal 到 EndOutVal 所需的时间。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetMode(pPinName, OUTMODE_DC_FV, HPS100_3V,HPS100_3mA,
10.0, -10.0, HPS100_Quadrant1);
TheInst.HPS100().DCVI().SetOutValSlow(strPinName, 0, 5, 3);
```

2.14.2.21 SetPIDParameter（）

函数功能

设置 PID 参数，可调整 FV 或 FI 的波形参数，如上升斜率的快慢。

使用说明

无。

函数原型

```
int SetPIDParameter(const string& pPinName, emVI_MeasType emType, uint32_t
uDataKP, uint32_t uDataKI, uint32_t uDataKD);
```

参数说明

参数	说明
pPinName	Pin 的名字。
emType	测量类型，其类型定义详见 emVI_MeasType 类型定义。
uDataKP	KP 参数，取值范围：40～100。
uDataKI	KI 数据，推荐设置为 KP 的 0.25，不超过 KP 的 0.3。
uDataKD	KD 数据，推荐设置为 KP 的 0.25，不超过 KP 的 0.3。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetPIDParameter(strPinName, emVI_MV, 40, 10, 10);
```

2.14.2.22 SetPulseCnt（）

函数功能

设置脉冲的输出个数。

使用说明

无。

函数原型

```
int SetPulseCnt(const string& pPinName, int nConut);
```

参数说明

参数	说明
pPinName	Pin 的名称。
nCount	脉冲个数。 取值范围：1～1000。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetPulseCnt(strPinName, 100);
```

2.14.2.23 SetPulseCycleTime（）

函数功能

设置脉冲的周期时间。

使用说明

无。

函数原型

```
int SetPulseCycleTime(const string& pPinName, int nHTime, int nLTime);
```

参数说明

参数	说明
pPinName	Pin 的名称。
nHTime	高电平持续时间。 ● 单位为 μs。 ● 取值范围：0.5ms～20ms，可设置 10ns 的倍数。
nLTime	低电平持续时间。 ● 单位为 μs。 ● 取值范围：6s～10s，可设置 10ns 的倍数。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
string strPinName = "Pin1";
TheInst.HPS100().DCVI().SetPulseCnt(strPinName, 1000, 5000);
```

2.14.2.24 SetRes（）

函数功能

切负载，靠切电流挡位决定负载大小。

使用说明

无。

函数原型

```
int SetRes(const int& nPhyCh, emVI_ResType emType, emHPS100_VRange VRang);
```

参数说明

参数	说明
nPhyCh	物理通道。
emType	负载阻值，其类型定义详见 emVI_ResType 类型定义。
VRang	电压输出或测量挡位。 当 POutMode= OUTMODE_DC_FV 或 OUTMODE_PULSE_FV 时，为电压输出挡位设定值，可选挡位 HPS100_3V、HPS100_10V、HPS100_30V、HPS100_60V、HPS100_120V。

返回值

类型	说明
int	● 0：正常。 ● -1：Pin 不存在。 ● -2：板卡类型不匹配。

示例

```
TheInst.HPS100().DCVI().SetRes(0, 10, HPS100_3V);
```