6914 系列
PXI/PXIe 数字示波器
编程手册



中电科思仪科技股份有限公司

该手册适用下列型号 PXI/PXIe 数字示波器，基于固件版本 Version 1.0 及以上。

- 6914CA/DA/EA 数字示波器
- 6914CX/DX/EX 数字示波器

除标准配件外的选件如下：

- PXI/PXIe 机箱
- 机箱控制器
- 高压探头
- 高压差分探头
- 高压单端探头
- 电流探头
- 阻抗匹配器
- 英文选件：英文菜单、英文面板等，出口使用。

版本：A.1 2021年5月，中电科思仪科技股份有限公司
地　　址：中国山东青岛经济技术开发区香江路98号
免费客服电话：0532-86889847　400-1684191
技术支持：　　0532-86880796
传　　真：　　0532-86889056
网　　址：　　www.ceyear.com
电子信箱：　　techbb@ceyear.com
邮　　编：　　266555

前　言

非常感谢您选择使用中电科思仪科技股份有限公司研制、生产的 6914 系列数字示波器！本所产品集高、精、尖于一体，在同类产品中有较高的性价比。

我们将以最大限度满足您的需求为己任，为您提供高品质的测量仪器，同时带给您一流的售后服务。我们的一贯宗旨是“质量优良，服务周到”，提供满意的产品和服务是我们对用户的承诺。

手册编号

AV2.734.1041SC

版本

A.1　2021.5

中电科思仪科技股份有限公司

手册授权

本手册中的内容如有变更，恕不另行通知。本手册内容及所用术语最终解释权属于中电科思仪科技股份有限公司。

本手册版权属于中电科思仪科技股份有限公司，任何单位或个人非经本所授权，不得对本手册内容进行修改，并且不得以赢利为目的对本手册进行复制、传播，中电科思仪科技股份有限公司保留对侵权者追究法律责任的权利。

产品质保

本产品从出厂之日起保修期为 12 个月。质保期内仪器生产厂家会根据用户要求及实际情况维修或替换损坏部件。具体维修操作事宜以合同为准。

产品质量证明

本产品从出厂之日起确保满足手册中的指标。校准测量由具备国家资质的计量单位予以完成，并提供相关资料以备用户查阅。

质量/环境管理

本产品从研发、制造和测试过程中均遵守质量和环境管理体系。中电科思仪科技股份有限公司已经具备资质并通过 ISO 9001 和 ISO 14001 管理体系。

安全事项

注　意

注意标识代表重要的信息提示，但不会导致危险。它提示用户注意某一操作过程、操作方法或者类似情况。若不能遵守规则或者正确操作，则可能引起的仪器损坏或丢失重要数据。在完全理解和满足所指出的注意条件之后，才可继续下一步。

提　示

提示标识代表信息提示。它提示用户注意仪器或者某一操作过程、操作方法或者类似情况。以引导仪器操作者正确的使用仪器。

目 录

1 手册导航.....1
1.1 关于手册.....1
1.2 关联文档.....1
2 动态链接库使用说明.....3
2.1 准备工作.....3
2.2 动态链接库使用说明.....3
3 函数说明.....3
3.1 系统命令.....3
3.1.1 硬件驱动程序初始化.....4
3.1.2 示波器模块初始化.....4
3.1.3 示波器退出.....4
3.1.4 启动采集.....4
3.1.5 获取采集数据各参数.....5
3.1.6 自动刻度.....5
3.2 水平命令.....7
3.2.1 获取当前时基.....7
3.2.2 设置时基.....8
3.2.3 获取时延.....10
3.2.4 设置时延.....10
3.2.5 调节时延.....10
3.3 垂直命令.....10
3.3.1 获取通道耦合.....11
3.3.2 设置通道耦合.....11
3.3.3 获取通道阻抗.....12
3.3.4 设置通道阻抗.....12
3.3.5 获取通道带宽限制.....12
3.3.6 设置通道带宽限制.....13
3.3.7 获取探头类型.....13
3.3.8 设置探头类型.....13

目录

3.3.9 获取探头系数 14
3.3.10 设置探头系数 14
3.3.11 获取通道反相 15
3.3.12 设置通道反相 16
3.3.13 获取通道垂直灵敏度细调开关 16
3.3.14 设置通道垂直灵敏度细调开关 16
3.3.15 获取外触发通道高频抑制状态 16
3.3.16 设置外触发通道高频抑制状态 17
3.3.17 获取外触发通道衰减倍数 17
3.3.18 设置外触发通道衰减倍数 17
3.3.19 获取通道内部延迟数值 17
3.3.20 设置通道内部延迟数值 18
3.3.21 调节通道内部延迟数值 18
3.3.22 获取通道垂直灵敏度 18
3.3.23 设置通道垂直灵敏度 19
3.3.24 调节通道垂直灵敏度 20
3.3.25 获取垂直偏移 21
3.3.26 设置垂直偏移 21
3.3.27 调节垂直偏移 21
3.4 采集命令 22
3.4.1 获取采集模式 22
3.4.2 设置采集模式 22
3.4.3 获取存储深度索引号 22
3.4.4 设置存储深度索引号 23
3.4.5 获取当前存储分段数 23
3.4.6 设置当前存储分段数 23
3.5 触发命令 24
3.5.1 获取触发类型 24
3.5.2 设置触发类型 24
3.5.3 获取触发源 25
3.5.4 设置触发源 25
3.5.5 设置边沿触发类型 25

3.5.6 设置脉宽触发极性 25
3.5.7 获取脉宽条件类型与当前脉宽数值 26
3.5.8 设置脉宽条件类型 26
3.5.9 设置触发脉宽数值 26
3.5.10 调节触发脉宽数值 27
3.5.11 获取触发模式类型 27
3.5.12 设置触发模式类型 27
3.5.13 获取触发灵敏度数值 27
3.5.14 设置触发灵敏度数值 28
3.5.15 调节触发灵敏度数值 28
3.5.16 获取触发释抑数值 28
3.5.17 设置触发释抑数值 28
3.5.18 调节触发释抑数值 29
3.5.19 获取触发电平数值 29
3.5.20 设置触发电平数值 29
3.5.21 调节触发电平数值 30
3.6 测量 30
参数测量 30
4 返修方法 31
4.1 联系我们 31
4.2 包装与邮寄 31

1 手册导航

本章介绍了 6914 系列数字示波器的编程手册功能、章节构成和主要内容，并介绍了提供给用户使用的仪器关联文档。

- 关于手册……1
- 关联文档……1

1.1 关于手册

本手册介绍了 6914 系列数字示波器的远程控制指令。为方便您熟练使用该仪器，请在操作仪器前，仔细阅读本手册，然后按手册指导正确操作。该手册详细说明了 6914 的编程命令。

编程手册的章节包括：

- **动态链接库使用说明**

  概述了数字示波器进行远程控制编程时需要进行的操作方法，目的使用户可以对远程控制操作快速上手。分两部分介绍：准备工作，介绍与编程有关的软件环境安装；动态链接库使用说明，介绍 6914 系列示波器进行程控前必须的操作步骤，目的使用户可以快速完成仪器的远程控制。

- **函数说明**

  分类介绍了系统命令、水平命令、垂直命令、采集命令、触发命令及测量命令，逐条说明各项命令的功能和参数。

- **返修方法**

  包括联系我们和包装与邮寄。

1.2 关联文档

6914 系列数字示波器的产品文档包括：

- 用户手册
- 编程手册
- 在线帮助

用户手册

本手册详细介绍了仪器的功能和操作使用方法，包括：配置、测量、程控和维护等信息。目的是：指导用户如何全面的理解产品功能特点及掌握常用的仪器测试方法。包含的主要章节是：

- 手册导航
- 概述
- 使用入门
- 故障诊断与返修

1.2 关联文档

- 技术指标和测试方法

编程手册

本手册详细介绍了如何通过远程编程实现对数字示波器的控制以及所用到的指令。目的是：指导用户如何快速、全面的掌握仪器的编程命令和编程方法。包含的主要章节是：

- 手册导航
- 动态链接库的使用说明
- 函数说明
- 返修方法

在线帮助

在线帮助集成在仪器产品中，提供快速的文本导航帮助，方便用户本地和远控操作。用户界面工具条有对应的快捷键激活该功能。包含的主要章节同用户手册。

2 动态链接库使用说明

本章介绍了 6914 系列数字示波器进行程控编程时的动态链接库使用说明。具体内容包括：

- 准备工作……3
- 动态链接库使用说明……3

2.1 准备工作

PXI机箱插入6914系列数字示波器，安装Visual Sdudio的开发环境。

2.2 动态链接库使用说明

6914系列数字示波器的动态库包括6914.h、6914.dll、6914.lib等3个文件，另外，包括校准数据文件calidata.dat与驱动动态链接库XilinxPCIeDll.dll（6914CX/DX/EX）或PXI_6914.dll（6914CA/DA/EA）。用户在Visual Sdudio编程环境下，将以上三个文件加入到工程中，即可使用6914.h中的函数对仪器进行控制。

3 函数说明

- 系统命令……3
- 水平命令……7
- 垂直命令……10
- 采集命令……22
- 触发命令……24
- 测量命令……30

3.1 系统命令

- 硬件驱动程序初始化……4
- 示波器模块初始化……4
- 示波器退出……4
- 启动采集……4
- 获取采集数据各参数……5
- 自动刻度……5

3.1.1 硬件驱动程序初始化

SCOPEDLL_API BOOL __stdcall ScopeDeviceInit(void)

函数用途：初始化仪器硬件驱动句柄，该函数是访问仪器时首先要调用的函数。
参数列表：无。
返回值：函数执行情况：真表示成功，假表示失败。

3.1.2 示波器模块初始化

SCOPEDLL_API BOOL __stdcall ScopeInit()

函数用途：示波器各个功能模块初始化。
参数列表：无。
返回值：函数执行情况：真表示成功，假表示失败。

3.1.3 示波器退出

SCOPEDLL_API BOOL __stdcall ScopeExit(void)

函数用途：关闭仪器，在控制仪器结束后需调用该函数关闭仪器。
参数列表：无。
返回值：函数执行情况：真表示成功，假表示失败。

3.1.4 启动采集

SCOPEDLL_API BOOL __stdcall StartAcquire(short nAcqbuff[2][MAX_WAVE_POINTS], short nAcqbuff_High[2][MAX_WAVE_POINTS], DWORD *dwDataLenth, BOOL &bAcqNormal)

函数用途：启动示波器采集。
参数列表： **nAcqbuff：**二维数组，分别存储通道 1 与通道 2 的采集数据，采集数据为 ADC 数字量，最小数字量为 0x00，最大数字量为 0xFF，0x8000 为无效数据点；
nAcqbuff_High：二维数组，分别存储通道 1 与通道 2 的采集数据，采集数据为 ADC 数字量，最小数字量为 0x00，最大数字量为 0xFF，0x8000 为无效数据点；
dwDataLenth：数据长度；
bAcqNormal：真时为正常采集数据，只有 nAcqbuff 数据有效；假时则为峰值、包络等模式采集数据，nAcqbuff 与 nAcqbuff_High 的数据都有效。
MAX_WAVE_POINTS：宏定义，采集最大数据点数目，4000。
返回值：函数执行情况：真表示成功，假表示失败。

3.1.5 获取采集数据各参数

SCOPEDLL_API BOOL __stdcall GetAcquireDataPreamble(int nChan,PPREAMBLE preamble)

函数用途：获取采集数据各参数

参数列表：nCH：通道编号，定义为：

```
#define CH1          0     //通道 1
#define CH2          1     //通道 2
```

preamble：参数结构体，定义为：

```
typedef struct    Preamble
{
   double timeBase;              //时基
   double timeDelay;             //时延
   double timeIncrement;         //采样间隔
   double timeOrigin;            //第一个采样点时间
   double voltBase;              //垂直灵敏度
   double voltOffset;            //偏移电压值
   double voltIncrement;         //每 AD 值电压间隔
   double voltOrigin;            //AD 值 0x00 代表电压值
}PREAMBLE,*PPREAMBLE;
```

返回值：函数执行情况：真表示成功，假表示失败。

3.1.6 自动刻度

SCOPEDLL_API BOOL __stdcall AutoScale(int *nTimeIndex, double &dfTimeBase, int *nVoltIndex, double &dfVolt)

函数用途：自动刻度功能，根据输入的信号，找到合适的时基与垂直灵敏度。

参数列表：nTimeIndex：返回值，自动刻度成功则返回时基索引号，定义为：

```
#define TIMEBASE_100ps      39      //100ps/div
#define TIMEBASE_200ps      38      //200ps/div
#define TIMEBASE_400ps      37      //400ps/div
#define TIMEBASE_1ns        36      //1ns/div
#define TIMEBASE_2ns        35      //2ns/div
#define TIMEBASE_4ns        34      //4ns/div
#define TIMEBASE_10ns       33      //10ns/div
#define TIMEBASE_20ns       32      //20ns/div
#define TIMEBASE_40ns       31      //40ns/div
#define TIMEBASE_100ns      30      //100ns/div
#define TIMEBASE_200ns      29      //200ns/div
#define TIMEBASE_400ns      28      //400ns/div
#define TIMEBASE_1us        27      //1us/div
#define TIMEBASE_2us        26      //2us/div
```

3.1 系统命令

```
#define TIMEBASE_4us        25    //4us/div
#define TIMEBASE_10us       24    //10us/div
#define TIMEBASE_20us       23    //20us/div
#define TIMEBASE_40us       22    //40us/div
#define TIMEBASE_100us      21    //100us/div
#define TIMEBASE_200us      20    //200us/div
#define TIMEBASE_400us      19    //400us/div
#define TIMEBASE_1ms        18    //1ms/div
#define TIMEBASE_2ms        17    //2ms/div
#define TIMEBASE_4ms        16    //4ms/div
#define TIMEBASE_10ms       15    //10ms/div
#define TIMEBASE_20ms       14    //20ms/div
#define TIMEBASE_40ms       13    //40ms/div
#define TIMEBASE_100ms      12    //100ms/div
#define TIMEBASE_200ms      11    //200ms/div
#define TIMEBASE_400ms      10    //400ms/div
#define TIMEBASE_1s         9     //1ps/div
#define TIMEBASE_2s         8     //2s/div
#define TIMEBASE_4s         7     //4s/div
#define TIMEBASE_10s        6     //10s/div
#define TIMEBASE_20s        5     //20s/div
#define TIMEBASE_40s        4     //40s/div
#define TIMEBASE_100s       3     //100s/div
#define TIMEBASE_200s       2     //200s/div
#define TIMEBASE_400s       1     //400s/div
#define TIMEBASE_1000s      0     //1000s/div
```

dfTimeBase：返回值，自动刻度成功则返回当前时基值，单位为 s；

nVoltIndex：返回值，自动刻度成功则返回垂直灵敏度索引，定义为：

```
#define CHSCALE_1mV         0     //1mV/div
#define CHSCALE_2mV         1     //2mV/div
#define CHSCALE_5mV         2     //5mV/div
#define CHSCALE_10mV        3     //10mV/div
#define CHSCALE_20mV        4     //20mV/div
#define CHSCALE_30mV        5     //30mV/div
#define CHSCALE_40mV        6     //40mV/div
#define CHSCALE_50mV        7     //50mV/div
#define CHSCALE_60mV        8     //60mV/div
#define CHSCALE_70mV        9     //70mV/div
#define CHSCALE_80mV        10    //80mV/div
#define CHSCALE_90mV        11    //90mV/div
#define CHSCALE_100mV       12    //100mV/div
#define CHSCALE_200mV       13    //200mV/div
#define CHSCALE_300mV       14    //300mV/div
```

```
#define CHSCALE_400mV    15    //400mV/div
#define CHSCALE_500mV    16    //500mV/div
#define CHSCALE_600mV    17    //600mV/div
#define CHSCALE_700mV    18    //700mV/div
#define CHSCALE_800mV    19    //800mV/div
#define CHSCALE_900mV    20    //900mV/div
#define CHSCALE_1V       21    //1V/div
#define CHSCALE_2V       22    //2V/div
#define CHSCALE_3V       23    //3V/div
#define CHSCALE_4V       24    //4V/div
#define CHSCALE_5V       25    //5V/div
#define CHSCALE_6V       26    //6V/div
#define CHSCALE_7V       27    //7V/div
#define CHSCALE_8V       28    //8V/div
#define CHSCALE_9V       29    //9V/div
#define CHSCALE_10V      30    //10V/div
```

dfVolt：返回值，自动刻度成功则返回当前垂直灵敏度值，单位为 V。

返回值：函数执行情况：真表示成功，假表示失败。

3.2 水平命令

- 获取当前时基……7
- 设置时基……8
- 获取时延……10
- 设置时延……10
- 调节时延……10

3.2.1 获取当前时基

SCOPEDLL_API BOOL __stdcall GetTimeBase(int &nIndex,double &dfTimeBase);

函数用途：获取当前时基序号与当前时基数值。

参数列表：nIndex： 返回值，水平时基序号，定义为：

```
#define TIMEBASE_100ps    39    //100ps/div
#define TIMEBASE_200ps    38    //200ps/div
#define TIMEBASE_400ps    37    //400ps/div
#define TIMEBASE_1ns      36    //1ns/div
#define TIMEBASE_2ns      35    //2ns/div
#define TIMEBASE_4ns      34    //4ns/div
#define TIMEBASE_10ns     33    //10ns/div
#define TIMEBASE_20ns     32    //20ns/div
#define TIMEBASE_40ns     31    //40ns/div
```

```
#define TIMEBASE_100ns      30      //100ns/div
#define TIMEBASE_200ns      29      //200ns/div
#define TIMEBASE_400ns      28      //400ns/div
#define TIMEBASE_1us        27      //1us/div
#define TIMEBASE_2us        26      //2us/div
#define TIMEBASE_4us        25      //4us/div
#define TIMEBASE_10us       24      //10us/div
#define TIMEBASE_20us       23      //20us/div
#define TIMEBASE_40us       22      //40us/div
#define TIMEBASE_100us      21      //100us/div
#define TIMEBASE_200us      20      //200us/div
#define TIMEBASE_400us      19      //400us/div
#define TIMEBASE_1ms        18      //1ms/div
#define TIMEBASE_2ms        17      //2ms/div
#define TIMEBASE_4ms        16      //4ms/div
#define TIMEBASE_10ms       15      //10ms/div
#define TIMEBASE_20ms       14      //20ms/div
#define TIMEBASE_40ms       13      //40ms/div
#define TIMEBASE_100ms      12      //100ms/div
#define TIMEBASE_200ms      11      //200ms/div
#define TIMEBASE_400ms      10      //400ms/div
#define TIMEBASE_1s         9       //1ps/div
#define TIMEBASE_2s         8       //2s/div
#define TIMEBASE_4s         7       //4s/div
#define TIMEBASE_10s        6       //10s/div
#define TIMEBASE_20s        5       //20s/div
#define TIMEBASE_40s        4       //40s/div
#define TIMEBASE_100s       3       //100s/div
#define TIMEBASE_200s       2       //200s/div
#define TIMEBASE_400s       1       //400s/div
#define TIMEBASE_1000s      0       //1000s/div
```

dfTimeBase：返回值，水平时基数值，时间单位 s。

返回值：函数执行情况：真表示成功，假表示失败。

3.2.2 设置时基

SCOPEDLL_API BOOL __stdcall SetTimeBase(int nIndex,double &dfTimeBase);

函数用途：设置水平时基。

参数列表：nIndex：水平时基序号，定义为：

```
#define TIMEBASE_100ps      39      //100ps/div
#define TIMEBASE_200ps      38      //200ps/div
#define TIMEBASE_400ps      37      //400ps/div
```

```
#define TIMEBASE_1ns        36    //1ns/div
#define TIMEBASE_2ns        35    //2ns/div
#define TIMEBASE_4ns        34    //4ns/div
#define TIMEBASE_10ns       33    //10ns/div
#define TIMEBASE_20ns       32    //20ns/div
#define TIMEBASE_40ns       31    //40ns/div
#define TIMEBASE_100ns      30    //100ns/div
#define TIMEBASE_200ns      29    //200ns/div
#define TIMEBASE_400ns      28    //400ns/div
#define TIMEBASE_1us        27    //1us/div
#define TIMEBASE_2us        26    //2us/div
#define TIMEBASE_4us        25    //4us/div
#define TIMEBASE_10us       24    //10us/div
#define TIMEBASE_20us       23    //20us/div
#define TIMEBASE_40us       22    //40us/div
#define TIMEBASE_100us      21    //100us/div
#define TIMEBASE_200us      20    //200us/div
#define TIMEBASE_400us      19    //400us/div
#define TIMEBASE_1ms        18    //1ms/div
#define TIMEBASE_2ms        17    //2ms/div
#define TIMEBASE_4ms        16    //4ms/div
#define TIMEBASE_10ms       15    //10ms/div
#define TIMEBASE_20ms       14    //20ms/div
#define TIMEBASE_40ms       13    //40ms/div
#define TIMEBASE_100ms      12    //100ms/div
#define TIMEBASE_200ms      11    //200ms/div
#define TIMEBASE_400ms      10    //400ms/div
#define TIMEBASE_1s         9     //1ps/div
#define TIMEBASE_2s         8     //2s/div
#define TIMEBASE_4s         7     //4s/div
#define TIMEBASE_10s        6     //10s/div
#define TIMEBASE_20s        5     //20s/div
#define TIMEBASE_40s        4     //40s/div
#define TIMEBASE_100s       3     //100s/div
#define TIMEBASE_200s       2     //200s/div
#define TIMEBASE_400s       1     //400s/div
#define TIMEBASE_1000s      0     //1000s/div
```

dfTimeBase：返回值，水平时基数值，时间单位 s。

返回值：函数执行情况：真表示成功，假表示失败。

3.2.3 获取时延

SCOPEDLL_API BOOL __stdcall GetTimeDelay(double &dfValue)

函数用途：获取水平时延。
参数列表：dfValue：返回值，水平时延当前值。
返回值：函数执行情况：真表示成功，假表示失败。

3.2.4 设置时延

SCOPEDLL_API BOOL __stdcall SetTimeDelay(double dfOriginalValue,double &dfValue)

函数用途：设置水平时延。
参数列表：dfOriginalValue：水平时延输入值，单位为 s；
dfValue：返回值，水平时延实际值，系统会根据当前水平时延步进，对输入值适当调整。
返回值：函数执行情况：真表示成功，假表示失败。

3.2.5 调节时延

SCOPEDLL_API BOOL __stdcall ModifyTimeDelay(double &dfTimeDelay,int nDirect, int nNum=1)

函数用途：按照步进，调节水平时延。
参数列表：dfTimeDelay：返回值，水平时延值，单位为 s；
nDirect：调节方向，定义为：
DERICT_TYPE_POS 0 //增加
DERICT_TYPE_NEG 1 //减少
nNum：调节量，默认为 1。
返回值：函数执行情况：真表示成功，假表示失败。

3.3 垂直命令

- 获取通道耦合……11
- 设置通道耦合……11
- 获取通道阻抗……12
- 设置通道阻抗……12
- 获取通道带宽限制……12
- 设置通道带宽限制……13
- 获取探头类型……13
- 设置探头类型……13
- 获取探头系数……14
- 设置探头系数……14

- 获取通道反相…… 15
- 设置通道反相……16
- 获取通道灵敏度细调开关……16
- 设置通道灵敏度细调开关……16
- 获取外触发通道高频抑制状态……16
- 设置外触发通道高频抑制状态……17
- 获取外触发通道衰减倍数……17
- 设置外触发通道衰减倍数……17
- 获取通道内部延迟数值……17
- 设置通道内部延迟数值……18
- 获取通道垂直灵敏度……18
- 设置通道垂直灵敏度……19
- 调节通道垂直灵敏度……20
- 获取垂直偏移……21
- 设置垂直偏移……21
- 调节垂直偏移……21

3.3.1 获取通道耦合

SCOPEDLL_API BOOL __stdcall GetChanCoup(int nChan, int &nCoup)

函数用途：获取通道耦合。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

nCoup：返回值，通道耦合，定义为：

```
CHANCOUP_DC          0   //通道 DC 耦合
CHANCOUP_AC          1   //通道 AC 耦合
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.2 设置通道耦合

SCOPEDLL_API BOOL __stdcall SetChanCoup(int nChan, int nCoup)

函数用途：获取通道耦合。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

nCoup：通道耦合，定义为：

```
CHANCOUP_DC          0   //通道 DC 耦合
CHANCOUP_AC          1   //通道 AC 耦合
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.3 获取通道阻抗

SCOPEDLL_API BOOL __stdcall GetChanImp(int nChan, int &nImp)

函数用途：获取通道阻抗。

参数列表：nChan：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

nImp：返回值，通道阻抗，定义为：

CHANIMP_50 0 //通道阻抗 50 欧姆

CHANIMP_1M 1 //通道阻抗 1M 欧姆

返回值：函数执行情况：真表示成功，假表示失败。

3.3.4 设置通道阻抗

SCOPEDLL_API BOOL __stdcall SetChanImp(int nChan, int nImp)

函数用途：获取通道阻抗。

参数列表：nChan：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

nCoup：通道阻抗，定义为：

CHANIMP_50 0 //通道阻抗 50 欧姆

CHANIMP_1M 1 //通道阻抗 1M 欧姆

返回值：函数执行情况：真表示成功，假表示失败。

3.3.5 获取通道带宽限制

SCOPEDLL_API BOOL __stdcall GetChanBWLimit(int nChan, int &nBWLimit)

函数用途：获取通道带宽限制状态。

参数列表：nChan：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

nBWLimit：返回值，通道带宽限制，定义为：

CHANBWLIMIT_ALL 0 //通道带宽限制：全带宽

CHANBWLIMIT_20M 1 //通道带宽限制：20M

CHANBWLIMIT_250M 2 //通道带宽限制：250M

返回值：函数执行情况：真表示成功，假表示失败。

3.3.6 设置通道带宽限制

SCOPEDLL_API BOOL __stdcall SetChanBWLimit(int nChan, int nBWLimit)

函数用途：设置通道带宽限制状态。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
```

nBWLimit：通道带宽限制，定义为：

```
CHANBWLIMIT_ALL    0   //通道带宽限制：全带宽
CHANBWLIMIT_20M    1   //通道带宽限制：20M
CHANBWLIMIT_250M   2   //通道带宽限制：250M
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.7 获取探头类型

SCOPEDLL_API BOOL __stdcall GetChanProbeType(int nChan, int &nType)

函数用途：获取探头类型。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

nType：返回值，探头类型，定义为：

```
#define PROBE_VOLT   0    //电压
#define PROBE_CURR   1    //电流
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.8 设置探头类型

SCOPEDLL_API BOOL __stdcall SetChanProbeType(int nChan, int nType)

函数用途：设置探头类型。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

nType：返回值，探头类型，定义为：

```
#define PROBE_VOLT   0    //电压
#define PROBE_CURR   1    //电流
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.9 获取探头系数

SCOPEDLL_API BOOL __stdcall GetChanProbe(int nChan, int &nProbe)

函数用途：获取通道探头系数。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

nProbe：返回值，通道探头系数，定义为：

电压探头：

```
CHANPROBE_1P        0   //×1（电压探头）
CHANPROBE_2P        1   //×2（电压探头）
CHANPROBE_5P        2   //×5（电压探头）
CHANPROBE_10P       3   //×10（电压探头）
CHANPROBE_20P       4   //×20（电压探头）
CHANPROBE_50P       5   //×50（电压探头）
CHANPROBE_100P      6   //×100（电压探头）
CHANPROBE_200P      7   //×200（电压探头）
CHANPROBE_500P      8   //×500（电压探头）
CHANPROBE_1000P     9   //×1000（电压探头）
CHANPROBE_2000P     10   //×2000（电压探头）
CHANPROBE_5000P     11   //×5000（电压探头）
```

电流探头：

```
CHANPROBE_1mV       0   //1mV/A（电流探头）
CHANPROBE_2mV       1   //2mV/A（电流探头）
CHANPROBE_5mV       2   //5mV（电流探头）
CHANPROBE_10mV      3   //10mV（电流探头）
CHANPROBE_20mV      4   //20mV（电流探头）
CHANPROBE_50mV      5   //50mV（电流探头）
CHANPROBE_100mV     6   //100mV（电流探头）
CHANPROBE_200mV     7   //200mV（电流探头）
CHANPROBE_500mV     8   //500mV（电流探头）
CHANPROBE_1V         9   //1000mV（电流探头）
```

返回值：函数执行情况：真表示成功，假表示失败。

3.3.10 设置探头系数

SCOPEDLL_API BOOL __stdcall SetChanProbe(int nChan, int nProbe)

函数用途：设置通道探头系数。

参数列表：nChan：通道编号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
```

#define CHEXT 2 //外通道

nProbe：通道探头系数，定义为：

电压探头：

CHANPROBE_1P 0 //×1（电压探头）
CHANPROBE_2P 1 //×2（电压探头）
CHANPROBE_5P 2 //×5（电压探头）
CHANPROBE_10P 3 //×10（电压探头）
CHANPROBE_20P 4 //×20（电压探头）
CHANPROBE_50P 5 //×50（电压探头）
CHANPROBE_100P 6 //×100（电压探头）
CHANPROBE_200P 7 //×200（电压探头）
CHANPROBE_500P 8 //×500（电压探头）
CHANPROBE_1000P 9 //×1000（电压探头）
CHANPROBE_2000P 10 //×2000（电压探头）
CHANPROBE_5000P 11 //×5000（电压探头）

电流探头：

CHANPROBE_1mV 0 //1mV/A（电流探头）
CHANPROBE_2mV 1 //2mV/A（电流探头）
CHANPROBE_5mV 2 //5mV（电流探头）
CHANPROBE_10mV 3 //10mV（电流探头）
CHANPROBE_20mV 4 //20mV（电流探头）
CHANPROBE_50mV 5 //50mV（电流探头）
CHANPROBE_100mV 6 //100mV（电流探头）
CHANPROBE_200mV 7 //200mV（电流探头）
CHANPROBE_500mV 8 //500mV（电流探头）
CHANPROBE_1V 9 //1000mV（电流探头）

返回值：函数执行情况：真表示成功，假表示失败。

3. 3. 11 获取通道反相

SCOPEDLL_API BOOL __stdcall GetChanInv(int nChan, int &nInv)

函数用途：获取通道反相状态。

参数列表：nChan：通道编号，定义为：

#define CH1 0 //通道 1
#define CH2 1 //通道 2

nInv：返回值，通道反相状态，定义为：

STATE_TYPE_OFF 0 //反相/细调：关
STATE_TYPE_ON 1 //反相/细调：开

返回值：函数执行情况：真表示成功，假表示失败。

3.3.12 设置通道反相

SCOPEDLL_API BOOL __stdcall SetChanInv(int nChan, int nInv)

函数用途：设置通道反相状态。
参数列表：nChan：通道编号，定义为：
#define CH1 0 //通道 1
#define CH2 1 //通道 2
nInv：通道反相状态，定义为：
STATE_TYPE_OFF 0 //反相/细调：关
STATE_TYPE_ON 1 //反相/细调：开
返回值：函数执行情况：真表示成功，假表示失败。

3.3.13 获取通道垂直灵敏度细调开关

SCOPEDLL_API BOOL __stdcall GetChanScaleFine(int nChan, int &nFine)

函数用途：获取通道垂直灵敏度细调开关状态。
参数列表：nChan：通道编号，定义为：
#define CH1 0 //通道 1
#define CH2 1 //通道 2
nFine：返回值，通道细调开关状态，定义为：
STATE_TYPE_OFF 0 //反相/细调：关
STATE_TYPE_ON 1 //反相/细调：开
返回值：函数执行情况：真表示成功，假表示失败。

3.3.14 设置通道垂直灵敏度细调开关

SCOPEDLL_API BOOL __stdcall SetChanScaleFine(int nChan, int nFine)

函数用途：设置通道垂直灵敏度细调开关。
参数列表：nChan：通道编号，定义为：
#define CH1 0 //通道 1
#define CH2 1 //通道 2
nFine：通道细调开关，定义为：
STATE_TYPE_OFF 0 //反相/细调：关
STATE_TYPE_ON 1 //反相/细调：开
返回值：函数执行情况：真表示成功，假表示失败。

3.3.15 获取外触发通道高频抑制状态

SCOPEDLL_API BOOL __stdcall GetChanExtTrigHF(int &nTrigHF)

函数用途：获取外触发通道高频抑制状态。
参数列表：nTrigHF：返回值，高频抑制状态，定义为：

STATE_TYPE_OFF　　0　//反相/细调：关
STATE_TYPE_ON　　1　//反相/细调：开

返回值：函数执行情况：真表示成功，假表示失败。

3.3.16 设置外触发通道高频抑制状态

SCOPEDLL_API BOOL __stdcall SetChanExtTrigHF(int nTrigHF)

函数用途：设置外触发通道高频抑制状态。
参数列表：nTrigHF：高频抑制状态，定义为：

STATE_TYPE_OFF　　0　//反相/细调：关
STATE_TYPE_ON　　1　//反相/细调：开

返回值：函数执行情况：真表示成功，假表示失败。

3.3.17 获取外触发通道衰减倍数

SCOPEDLL_API BOOL __stdcall GetChanExtAtten(int &nAtten)

函数用途：获取外触发通道衰减倍数。
参数列表：nAtten：返回值，衰减倍数，定义为：

CHANATTEN_DIV1　　0　//÷1 衰减
CHANATTEN_DIV10　　1　//÷10 衰减

返回值：函数执行情况：真表示成功，假表示失败。

3.3.18 设置外触发通道衰减倍数

SCOPEDLL_API BOOL __stdcall SetChanExtAtten(int nAtten)

函数用途：设置外触发通道衰减倍数。
参数列表：nTrigHF：衰减倍数，定义为：

CHANATTEN_DIV1　　0　//÷1 衰减
CHANATTEN_DIV10　　1　//÷10 衰减

返回值：函数执行情况：真表示成功，假表示失败。

3.3.19 获取通道内部延迟数值

SCOPEDLL_API BOOL __stdcall GetCHInternalPhase(int nChan, double &dfValue)

函数用途：获取通道内部延迟数值。
参数列表：nCH：通道编号，定义为：

#define CH1　　0　//通道 1
#define CH2　　1　//通道 2

dfValue：返回值，通道内部延迟数值，单位 s。

返回值：函数执行情况：真表示成功，假表示失败。

3.3.20 设置通道内部延迟数值

SCOPEDLL_API BOOL __stdcall SetCHInternalPhase(int nChan,double dfOrignalValue, double &dfRetValue)

函数用途：设置通道内部延迟数值。

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

dfOriginalValue：输入值，通道内部延迟值，单位 s，最小值为-150.4ns，最大值为 150.4ns；

dfValue：返回值，通道内部延迟实际值，系统会根据步进，对输入值适当调整。

返回值：函数执行情况：真表示成功，假表示失败。

3.3.21 调节通道内部延迟数值

SCOPEDLL_API BOOL __stdcall ModifyCHInternalPhase(int nChan,double &dfValue,int nDirect, int nNum=1)

函数用途：按照步进，调节通道内部延迟数值。

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

dfValue：返回值，通道内部延迟数值，单位 s，最小值为-150.4ns，最大值为 150.4ns；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.3.22 获取通道垂直灵敏度

SCOPEDLL_API BOOL __stdcall GetCHVerticalRange(int nChan,int &nVoltScale,double &dfValue)

函数用途：获取垂直灵敏度；

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

nVoltScale：返回值，垂直通道灵敏度索引值，定义为：

#define CHSCALE_1mV 0 //1mV/div

#define CHSCALE_2mV 1 //2mV/div

#define CHSCALE_5mV 2 //5mV/div

#define CHSCALE_10mV 3 //10mV/div

#define CHSCALE_20mV 4 //20mV/div

```
#define CHSCALE_30mV      5      //30mV/div
#define CHSCALE_40mV      6      //40mV/div
#define CHSCALE_50mV      7      //50mV/div
#define CHSCALE_60mV      8      //60mV/div
#define CHSCALE_70mV      9      //70mV/div
#define CHSCALE_80mV      10     //80mV/div
#define CHSCALE_90mV      11     //90mV/div
#define CHSCALE_100mV     12     //100mV/div
#define CHSCALE_200mV     13     //200mV/div
#define CHSCALE_300mV     14     //300mV/div
#define CHSCALE_400mV     15     //400mV/div
#define CHSCALE_500mV     16     //500mV/div
#define CHSCALE_600mV     17     //600mV/div
#define CHSCALE_700mV     18     //700mV/div
#define CHSCALE_800mV     19     //800mV/div
#define CHSCALE_900mV     20     //900mV/div
#define CHSCALE_1V        21     //1V/div
#define CHSCALE_2V        22     //2V/div
#define CHSCALE_3V        23     //3V/div
#define CHSCALE_4V        24     //4V/div
#define CHSCALE_5V        25     //5V/div
#define CHSCALE_6V        26     //6V/div
#define CHSCALE_7V        27     //7V/div
#define CHSCALE_8V        28     //8V/div
#define CHSCALE_9V        29     //9V/div
#define CHSCALE_10V       30     //10V/div
```

dfValue：返回值，垂直灵敏度数值，单位 V。

返回值：函数执行情况：真表示成功，假表示失败。

3. 3. 23 设置通道垂直灵敏度

SCOPEDLL_API BOOL __stdcall SetCHVerticalRange(int nChan,int &nVoltScale,double &dfValue)

函数用途：设置垂直灵敏度；

参数列表：nCH：通道编号，定义为：

```
#define CH1       0    //通道 1
#define CH2       1    //通道 2
```

nVoltScale：垂直通道灵敏度索引值，系统会根据通道垂直灵敏度细调开关是否打开进行调整，定义为：

```
#define CHSCALE_1mV       0      //1mV/div
#define CHSCALE_2mV       1      //2mV/div
#define CHSCALE_5mV       2      //5mV/div
#define CHSCALE_10mV      3      //10mV/div
```

```
#define CHSCALE_20mV      4      //20mV/div
#define CHSCALE_30mV      5      //30mV/div
#define CHSCALE_40mV      6      //40mV/div
#define CHSCALE_50mV      7      //50mV/div
#define CHSCALE_60mV      8      //60mV/div
#define CHSCALE_70mV      9      //70mV/div
#define CHSCALE_80mV      10     //80mV/div
#define CHSCALE_90mV      11     //90mV/div
#define CHSCALE_100mV     12     //100mV/div
#define CHSCALE_200mV     13     //200mV/div
#define CHSCALE_300mV     14     //300mV/div
#define CHSCALE_400mV     15     //400mV/div
#define CHSCALE_500mV     16     //500mV/div
#define CHSCALE_600mV     17     //600mV/div
#define CHSCALE_700mV     18     //700mV/div
#define CHSCALE_800mV     19     //800mV/div
#define CHSCALE_900mV     20     //900mV/div
#define CHSCALE_1V        21     //1V/div
#define CHSCALE_2V        22     //2V/div
#define CHSCALE_3V        23     //3V/div
#define CHSCALE_4V        24     //4V/div
#define CHSCALE_5V        25     //5V/div
#define CHSCALE_6V        26     //6V/div
#define CHSCALE_7V        27     //7V/div
#define CHSCALE_8V        28     //8V/div
#define CHSCALE_9V        29     //9V/div
#define CHSCALE_10V       30     //10V/div
```

dfValue：返回值，垂直灵敏度数值，单位 V。

返回值：函数执行情况：真表示成功，假表示失败。

3. 3. 24 调节通道垂直灵敏度

SCOPEDLL_API BOOL __stdcall ModifyCHVerticalRange(int nChan, double &dfValue,int nDirect, int nNum=1)

函数用途：调节垂直灵敏度。

参数列表：nCH：通道编号，定义为：

```
#define CH1          0     //通道 1
#define CH2          1     //通道 2
```

dfValue：返回值，垂直灵敏度数值，单位 V；

nDirect：调节方向，定义为：

```
DERICT_TYPE_POS     0     //增加
DERICT_TYPE_NEG     1     //减少
```

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，小于 0 表示失败。

3.3.25 获取垂直偏移

SCOPEDLL_API BOOL __stdcall GetCHVerticalOffset(int nChan,double &dfValue)

函数用途：获取垂直偏移。

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

dfValue：返回值，垂直偏移值，单位 V。

返回值：函数执行情况：真表示成功，假表示失败。

3.3.26 设置垂直偏移

SCOPEDLL_API BOOL __stdcall SetCHVerticalOffset(int nChan, double dfOriginalValue, double &dfValue)

函数用途：设置垂直偏移。

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

dfOriginalValue：输入值，垂直偏移值，单位 V；

dfValue：返回值，垂直偏移实际值，系统会根据垂直偏移步进，对输入值适当调整。

返回值：函数执行情况：真表示成功，假表示失败。

3.3.27 调节垂直偏移

SCOPEDLL_API BOOL __stdcall ModifyCHVerticalOffset(int nChan, double &dfValue,int nDirect, int nNum=1)

函数用途：按照步进，调节垂直偏移。

参数列表：nCH：通道编号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

dfValue：返回值，垂直偏移值，单位 V；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.4 采集命令

- 获取采集模式……22
- 设置采集模式……22
- 获取存储深度索引号……22
- 设置存储深度索引号……23
- 获取当前存储分段数……23
- 设置当前存储分段数……23

3.4.1 获取采集模式

SCOPEDLL_API BOOL __stdcall GetAcqType(int &nAcqType)

函数用途：获取当前采集模式。

参数列表：nAcqType：采集模式类型，定义为：

```
#define ACQMODE_NORMAL0     //正常
#define ACQMODE_PEAK    1   //峰值
#define ACQMODE_AVE     2   //平均
#define ACQMODE_HIGH    3   //高分辨率
#define ACQMODE_CLOSEIN 4   //包络
```

返回值：函数执行情况：真表示成功，假表示失败。

3.4.2 设置采集模式

SCOPEDLL_API BOOL __stdcall SetAcqType(int nAcqType,int nData=0)

函数用途：设置采集模式。

参数列表：nAcqType：采集模式类型，定义为：

```
#define ACQMODE_NORMAL0     //正常
#define ACQMODE_PEAK    1   //峰值
#define ACQMODE_AVE     2   //平均
#define ACQMODE_HIGH    3   //高分辨率
#define ACQMODE_CLOSEIN 4   //包络
```

nData：当采集模式为平均、包络时，需要设置数值。平均采集模式，最小值 2，最大值为 512，且为 2 的幂次方；包络采集模式时，最小值为 1，最大值为 2000。

返回值：函数执行情况：真表示成功，假表示失败。

3.4.3 获取存储深度索引号

SCOPEDLL_API BOOL __stdcall GetAcqDepth(int &nAcqDepth)

函数用途：获取当前存储深度序号。

参数列表：nAcqDepth：存储深度序号，定义为：

```
#define ACQDEPTH_1K     0   //1k
```

```
#define ACQDEPTH_10K      1    //10k
#define ACQDEPTH_100K     2    //100k
#define ACQDEPTH_1M       3    //1M
#define ACQDEPTH_10M      4    //10M
#define ACQDEPTH_100M     5    //100M
#define ACQDEPTH_200M     6    //200M
```

返回值：函数执行情况：真表示成功，假表示失败。

3.4.4 设置存储深度索引号

SCOPEDLL_API BOOL __stdcall SetAcqDepth(int nAcqDepth)

函数用途：设置当前存储深度序号。

参数列表：nAcqDepth：存储深度序号，定义为：

```
#define ACQDEPTH_1K       0    //1k
#define ACQDEPTH_10K      1    //10k
#define ACQDEPTH_100K     2    //100k
#define ACQDEPTH_1M       3    //1M
#define ACQDEPTH_10M      4    //10M
#define ACQDEPTH_100M     5    //100M
#define ACQDEPTH_200M     6    //200M
```

返回值：函数执行情况：真表示成功，假表示失败。

3.4.5 获取当前存储分段数

BOOL __stdcall GetAcqDepthSeg(int &nAcqSegExp, DWORD &nAcqSegNum)

函数用途：获取当前存储分段数参数与数目。

参数列表：nAcqSegExp：返回值，返回分段参数；

nAcqSegNum：返回值，返回分段数目；

nAcqSegNum = pow(2,nAcqSegExp)；nAcqSegNum 最大数目与存储深度相关，nAcqSegNum*存储深度< 256MB。当 nAcqSegExp 输入值不合理时，则系统会更改为合理的数值，并返回。

返回值：函数执行情况：真表示成功，假表示失败。

3.4.6 设置当前存储分段数

SCOPEDLL_API BOOL __stdcall SetAcqDepthSeg(int &nAcqSegExp, DWORD &nAcqSegNum)

函数用途：设置当前存储分段数参数与数目。

参数列表：nAcqSegExp：输入值，输入分段参数；

nAcqSegNum：返回值，返回分段数目；

nAcqSegNum = pow(2,nAcqSegExp)；nAcqSegNum 最大数目与存储深度相关，nAcqSegNum*存储深度< 256MB。当 nAcqSegExp 输入值不合理时，则系统会更

改为合理的数值，并返回。

返回值：函数执行情况：真表示成功，假表示失败。

3.5 触发命令

- 获取触发类型……24
- 设置触发类型……24
- 获取触发源……25
- 设置触发源……25
- 设置边沿触发类型……25
- 设置脉宽触发极性……25
- 获取脉宽条件类型与当前脉宽数值……26
- 设置脉宽条件类型……26
- 设置触发脉宽数值……26
- 调节触发脉宽数值……27
- 获取触发模式类型……27
- 设置触发模式类型……27
- 获取触发灵敏度数值……27
- 设置触发灵敏度数值……28
- 调节触发灵敏度数值……28
- 获取触发释抑数值……28
- 设置触发释抑数值……28
- 调节触发释抑数值……29
- 获取触发电平数值……29
- 设置触发电平数值……29
- 调节触发电平数值……30

3.5.1 获取触发类型

SCOPEDLL_API BOOL __stdcall GetTriggerType(int &nType)

函数用途：获取当前触发类型。

参数列表：nType：当前触发类型，定义为：

```
#define TRIGTYPE_EDGE        0        //边沿
#define TRIGTYPE_PULSE       1        //脉宽
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.2 设置触发类型

SCOPEDLL_API BOOL __stdcall SetTriggerType(int nType)

函数用途：设置触发类型。

参数列表：nType：当前触发类型，定义为：

```
#define TRIGTYPE_EDGE        0        //边沿
#define TRIGTYPE_PULSE       1        //脉宽
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.3 获取触发源

SCOPEDLL_API BOOL __stdcall GetTriggerSource(int &nSource)

函数用途：获取当前触发源。

参数列表：nSource：返回值，通道序号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.4 设置触发源

SCOPEDLL_API BOOL __stdcall SetTriggerSource(int nSource)

函数用途：设置触发源。

参数列表：nSource：通道序号，定义为：

```
#define CH1          0    //通道 1
#define CH2          1    //通道 2
#define CHEXT        2    //外通道
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.5 设置边沿触发类型

SCOPEDLL_API BOOL __stdcall SetTriggerEdgeType(int nType)

函数用途：设置边沿触发类型。

参数列表：nType：边沿触发类型，定义为：

```
#define TRIGEDGE_POS  0    //触发上升沿
#define TRIGEDGE_NEG  1    //触发下降沿
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.6 设置脉宽触发极性

SCOPEDLL_API BOOL __stdcall SetTriggerPulsePolarityType(int nType)

函数用途：设置脉宽触发极性。

参数列表：nType：边沿触发类型，定义为：

```
#define TRIGEDGE_POS  0    //触发上升沿
#define TRIGEDGE_NEG  1    //触发下降沿
```

返回值：函数执行情况：真表示成功，假表示失败

3.5.7 获取脉宽条件类型与当前脉宽数值

SCOPEDLL_API BOOL __stdcall GetTriggerPulseConditionType(int &nType,double &dfWidth1,double &dfWidth2)

函数用途：设置脉宽条件类型。

参数列表：nType：脉宽条件类型，定义为：

```
#define TRIGPULSECOND_SMALL     0    //脉宽小于
#define TRIGPULSECOND_BIG       1    //脉宽大于
#define TRIGPULSECOND_EQU       2    //脉宽等于
#define TRIGPULSECOND_NOEQU     3    //脉宽不等于
#define TRIGPULSECOND_IN        4    //范围内
#define TRIGPULSECOND_OUT       5    //范围外
```

dfWidth1：脉宽 1 设置的数值，单位为 s；

dfWidth2：脉宽 2 设置的数值，单位为 s，仅触发类型为范围内与范围外有效；

返回值：函数执行情况：真表示成功，假表示失败。

3.5.8 设置脉宽条件类型

SCOPEDLL_API BOOL __stdcall SetTriggerPulseConditionType(int nType)

函数用途：设置脉宽条件类型。

参数列表：nType：边沿触发类型，定义为：

```
#define TRIGPULSECOND_SMALL     0    //脉宽小于
#define TRIGPULSECOND_BIG       1    //脉宽大于
#define TRIGPULSECOND_EQU       2    //脉宽等于
#define TRIGPULSECOND_NOEQU     3    //脉宽不等于
#define TRIGPULSECOND_IN        4    //范围内
#define TRIGPULSECOND_OUT       5    //范围外
```

返回值：函数执行情况：真表示成功，假表示失败。

3.5.9 设置触发脉宽数值

SCOPEDLL_API BOOL __stdcall SetTriggerPulseWidth(double &dfValue,double dfRoughValue,int nWidthIdx, double &dfStep)

函数用途：设置通道内部延迟数值。

参数列表：dfValue：返回值，脉宽实际数值，单位为 s，系统会根据步进，对输入值适当调整，最小值 800ps，最大值为 12.8s。

dfRoughValue：输入值，脉宽设置输入值，单位 s；

nWidthIdx：脉宽序号，定义为：

```
PULSE_WIDTH_IDX1        0    //脉宽 1
PULSE_WIDTH_IDX2        1    //脉宽 2
```

dfStep：返回值，当前脉宽的步进值，与时基相关。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.10 调节触发脉宽数值

SCOPEDLL_API BOOL __stdcall ModifyTriggerPulseWidth (double &dfValue,int nDirect,int nWidthIdx, int nNum=1)

函数用途：按照步进，调节通道内部延迟数值。

参数列表：dfValue：返回值，脉宽实际数值，单位为 s，系统会根据步进，对输入值适当调整，最小值 800ps，最大值为 12.8s；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nWidthIdx：脉宽序号，定义为：

PULSE_WIDTH_IDX1 0 //脉宽 1

PULSE_WIDTH_IDX2 1 //脉宽 2

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.11 获取触发模式类型

BOOL __stdcall GetTriggerModeType(int &nType)

函数用途：获取当前触发模式类型。

参数列表：nType：返回值，当前触发模式类型，定义为：

#define TRIGMODE_AUTO 0 //自动

#define TRIGMODE_WAIT 1 //等待

返回值：函数执行情况：真表示成功，假表示失败。

3.5.12 设置触发模式类型

SCOPEDLL_API BOOL __stdcall SetTriggerModeType(int nType)

函数用途：设置触发类型。

参数列表：nType：当前触发模式类型，定义为：

#define TRIGMODE_AUTO 0 //自动

#define TRIGMODE_WAIT 1 //等待

返回值：函数执行情况：真表示成功，假表示失败。

3.5.13 获取触发灵敏度数值

SCOPEDLL_API BOOL __stdcall GetTriggerSensitivity(double &dfValue)

函数用途：获取触发灵敏度数值。

参数列表：dfValue：返回值，触发灵敏度数值，单位 div。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.14 设置触发灵敏度数值

SCOPEDLL_API BOOL __stdcall SetTriggerSensitivity(double &dfValue,double dfRoughValue, double &dfStep)

函数用途：设置触发灵敏度数值。

参数列表：dfValue：返回值，触发灵敏度实际数值，单位 div；系统会根据步进，对输入值适当调整。

dfRoughValue：输入值，触发灵敏度输入数值，单位 div，最小值 0div，最大值为 8div；

dfStep：返回值，触发灵敏度的步进值。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.15 调节触发灵敏度数值

SCOPEDLL_API BOOL __stdcall ModifyTriggerSensitivity(double &dfValue,int nDirect,int nNum=1)

函数用途：按照步进，调节触发灵敏度数值。

参数列表：dfValue：返回值，触发灵敏度实际数值，单位 div，最小值 0div，最大值为 8div；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.16 获取触发释抑数值

SCOPEDLL_API BOOL __stdcall GetTriggerHoldoffTime(double &dfValue)

函数用途：获取触发灵敏度数值

参数列表：dfValue：返回值，触发释抑数值，单位 s；

返回值：函数执行情况：真表示成功，假表示失败。

3.5.17 设置触发释抑数值

SCOPEDLL_API BOOL __stdcall SetTriggerHoldoffTime(double &dfValue,double dfRoughValue, double &dfStep)

函数用途：设置触发释抑数值。

参数列表：dfValue：返回值，触发释抑实际数值，单位 s，系统会根据步进，对输入值适当调整。

dfRoughValue：输入值，触发释抑输入数值，单位 s，最小值 3.2ns，最大值为 12.8s；

dfStep：返回值，触发释抑的步进值。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.18 调节触发释抑数值

SCOPEDLL_API BOOL __stdcall ModifyTriggerHoldoffTime(double &dfValue,int nDirect, int nNum=1)

函数用途：按照步进，调节触发释抑数值。

参数列表：dfValue：返回值，触发释抑实际数值，单位 s，最小值 3.2ns，最大值为 12.8s；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.19 获取触发电平数值

SCOPEDLL_API BOOL __stdcall GetTriggerLevel(int nChan,double &dfValue)

函数用途：获取触发电平数值。

参数列表：nChan：通道序号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

#define CHEXT 2 //外通道

dfValue：返回值，触发电平数值，单位 V。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.20 设置触发电平数值

SCOPEDLL_API BOOL __stdcall SetTriggerLevel(double &dfValue,double dfRoughValue, double &dfStep)

函数用途：设置触发电平数值。

参数列表：dfValue：返回值，触发电平实际数值，单位 V，系统会根据步进，对输入值适当调整。

dfRoughValue：输入值，触发电平输入数值，单位 V；

dfStep：返回值，触发电平的步进值，与垂直灵敏度相关。

返回值：函数执行情况：真表示成功，假表示失败。

3.5.21 调节触发电平数值

SCOPEDLL_API BOOL __stdcall ModifyTriggerLevel(double &dfValue,int nDirect, int nNum=1)

函数用途：按照步进，调节触发电平数值；

参数列表：dfValue：返回值，触发电平实际数值，单位 s；

nDirect：调节方向，定义为：

DERICT_TYPE_POS 0 //增加

DERICT_TYPE_NEG 1 //减少

nNum：调节量，默认为 1。

返回值：函数执行情况：真表示成功，假表示失败。

3.6 测量

- 参数测量 …… 30

参数测量

SCOPEDLL_API BOOL __stdcall Measure(double *dfResult, int nChan, int nMearType)

函数用途：测量信号参数。

参数列表：dfResult：返回值，测量结果，时间单位 s，电压单位 V，频率单位 Hz；

nChan：通道序号，定义为：

#define CH1 0 //通道 1

#define CH2 1 //通道 2

nMearType：测量类型，定义为：

MEAR_TYPE_PERiod 0 //周期

MEAR_TYPE_FREQuency 1 //频率

MEAR_TYPE_RTIMe 2 //上升时间

MEAR_TYPE_FTIMe 3 //下降时间

MEAR_TYPE_PDUTy 4 //正占空比

MEAR_TYPE_NDUTy 5 //负占空比

MEAR_TYPE_PWIDth 6 //正脉宽

MEAR_TYPE_NWIDth 7 //负脉宽

MEAR_TYPE_AMPLitude 8 //幅度

MEAR_TYPE_HIGH 9 //高电平

MEAR_TYPE_LOW 10 //低电平

MEAR_TYPE_MAXimum 11 //最大值

MEAR_TYPE_MINimum 12 //最小值

MEAR_TYPE_PTPeak 13 //峰峰值

MEAR_TYPE_PShOot 14 //预冲

MEAR_TYPE_NSHOot 15 //过冲

MEAR_TYPE_TOPFALL 16 //顶降
MEAR_TYPE_AVER 17 //平均值
MEAR_TYPE_CAVEr 18 //周期平均
MEAR_TYPE_RMS 19 //均方根
MEAR_TYPE_CRMS 20 //周期均方根
MEAR_TYPE_BURStwidth 21 //突发脉宽
MEAR_TYPE_PPULsenum 22 //正脉冲数
MEAR_TYPE_NPULsenum 23 //负脉冲数
MEAR_TYPE_REDGenum 24 //上升沿数
MEAR_TYPE_FEDGenum 25 //下降沿数
MEAR_TYPE_ACREage 26 //面积
MEAR_TYPE_CACReage 27 //周期面积

返回值：函数执行情况：真表示成功，假表示失败。

4 返修方法

- 联系我们……31
- 包装与邮寄……31

4.1 联系我们

若6914系列数字示波器出现问题，首先观察错误信息并保存，然后根据下面的联系方式与我公司服务咨询中心联系并提供收集的错误信息，我们将以最快的速度协助您解决问题。

联系方式：

免费客服电话：**0532-86889847 400-1684191**

技术支持：**0532-86880796**

传　　真：**0532-86889056**

网　　址：**www.ceyear.com**

电子信箱：**techbb@ceyear.com**

邮　　编：**266555**

地　　址：**中国山东青岛经济技术开发区香江路98号**

4.2 包装与邮寄

当您的数字示波器出现难以解决的问题时，可通过电话或传真与我们联系。如果经联系确认是数字示波器需要返修时，请您用原包装材料和包装箱包装数字示波器，并按下面的步骤进行包装：

1) 写一份有关数字示波器故障现象的详细说明，与数字示波器一同放入包装箱；
2) 用原包装材料将数字示波器包装好，以减少可能的损坏；
3) 在外包装纸箱四角摆放好衬垫，将仪器放入外包装箱；
4) 用胶带密封好包装箱口，并用尼龙带加固包装箱；

5) 在箱体上标明“易碎！勿碰！小心轻放！”字样；
6) 请按精密仪器进行托运；
7) 保留所有运输单据的副本。

注 意

包装信号发生器需注意

使用其它材料包装数字示波器，可能会损坏仪器。禁止使用聚苯乙烯小球作为包装材料，它们一方面不能充分保护仪器，另一方面会被产生的静电吸入仪器中，对仪器造成损坏。

提 示

仪器的包装和运输

运输或者搬运本仪器（例如，因发货期间的损坏）时，请严格遵守用户手册章节“3.1.1.1 开箱”中描述的注意事项。