

# 保证和声明

软件版本

软件升级可能会增加产品功能，请联系国仪量子（合肥）技术有限公司升级软件，必要时我司会主动与您联系。

声明

•本公司产品受中国及其他国家和地区的专利（包括已取得和正在申请的专利）保护。

•本公司拥有改变产品规格及价格的权利。

•本手册提供的信息取代以往出版的任何资料。

•未经我司事先书面许可，不得影印、复制或改变本手册的任何部分。

•用户一旦使用产品，即视为对本声明的全部内容认可和接受。

联系我们

* 电子邮箱：[sale@ciqtek.com,](mailto:sale@ciqtek.com)[service@ciqtek.com](mailto:service@ciqtek.com)

• 电话： 4000606976,0551-63367168

* 企业官网：[www.ciqtek.com](http://www.ciqtek.com)

目录

[保证和声明 2](#_Toc48138282)

[1、接口程序调用 4](#_Toc48138283)

[1.1C++DLL接口程序 4](#_Toc48138284)

[1.2C++调用事例 10](#_Toc48138285)

[1.3Python调用事例 11](#_Toc48138286)

# 1、接口程序调用

## 1.1C++DLL接口程序

我司为用户提供了 C++ 的动态链接库，各开发语言都能方 便的进行调用。用户可调用接口程序中的函数自己编程生成方波 序列，并将其下载到硬件中然后控制产品播放生成的方波序列。 现将用户所需的函数列出，并作出解释。

**函数 int ASG8x00\_Connect ( )**

函数功能：连接设备。

返回值： int 类型, 0连接成功，其他值连接失败。

参数：无。

**函数 int ASG8x00\_Disconnect()**

函数功能：用来断开软件和仪器的连接。

返回值： int 类型,0断开连接成功,其他断开连接失败。

参数：无

**函数 int ASG8x00\_DeviceMonitor ( )**

函数功能：查看设备连接状态。

返回值： int 类型, 0正常连接，其他值非正常连接。

参数：无。

**函数 int ASG8x00\_StartPlay(int loop = 0)**

函数功能：开始播放asg序列，同时开始采集count,默认无循环次数，一直播放。

返回值： int 类型,0开始播放成功,其他开始播放失败。

参数：int loop，循环次数。如果asg和count功能同时使用，asg和count的段数必须相等，对应每一段的循环次数必须相等。选取asg和count周期最长的作为整体的循环周期，播放loop次，loop为0时一直播放。若不同时使用则无以上限制。参数loop依赖波形下载中的segment的segment\_loop总和即loop = n \*（∑ segment\_loop）。例：asg和count都是2个segment，对应的segment\_loop为[2,3]，循环10次，则loop = 10\*（2+3）=50。

**函数 int ASG8x00\_StopPlay()**

函数功能: 停止播放asg序列和停止count采集。

返回值： int 类型,0停止播放成功,其他停止播放失败。

参数：无。

**函数 int ASG8x00\_SetErrorCallback (DLL\_ERROR\_CALLBACK callback)**

函数功能：用于设置提示信息回调函数，用于返回dll信息和字符串提示。

返回值： int 类型,0设置成功，其他值设置失败。

参数：DLL\_ERROR\_CALLBACK类型的函数指针[typedef void(\*DLL\_ERROR\_CALLBACK)(int type, int length, char\* data);]，type：错误码类型，length：提示信息长度，data：提示信息内容。

**函数 int ASG8x00\_SetCountCallback (DLL\_COUNT\_CALLBACK callback)**

函数功能：用于设置回调函数，用于返回count采集到的数据。

返回值： int 类型,0设置成功,其他设置失败。

参数：DLL\_COUNT\_CALLBACK类型的函数指针[typedef void(\*DLL\_COUNT\_CALLBACK)(int channel, int segment,int length, unsigned int\* data);]，channel：通道号，segment：段编号，length：此段的数据（unsigned int型）的个数，data：unsigned int型数据缓冲区地址。

**函数 int ASG8x00\_SetClockAndWorkMode (int IN1 = 0x80, int IN2 = 0x80)**

函数功能：设置IN1和IN2的功能。

返回值： int 类型,0设置成功,其他设置失败。

参数：int IN1，IN1端口的配置参数，0x80: count模式,0x00: 内部时钟,禁用IN1,0x41: 外部时钟10M,0x42: 外部时钟100M。

int IN2, IN2端口的配置参数, 0x80:count模式, 0x00:禁用IN2, 0x01:asg触发模式。

**函数 int ASG8x00\_AsgSetHightLevel(int iLevel)**

函数功能：设置ASG长高电平。

返回值： int 类型,0设置成功,其他设置失败。

参数： int iLevel，8-15bit依次控制1-8通道，1开启长高，0正常播放。例:0xff00:全部高电平，0x0000:全部正常工作。

**函数 int ASG8x00\_AsgSetChannelEnable(int iEnable)**

函数功能：设置ASG使能开关。

返回值： int 类型,0设置成功,其他设置失败。

参数：int iEnable，0x1ff：开启asg功能，0xff:关闭asg功能。

**函数 int ASG8x00\_AsgDownload(long long \*\*pulses, long long length[][8], unsigned short\* loop, unsigned short segmentNum)**

函数功能：用来下载ASG脉冲序列。

返回值： int 类型， 0下载成功，其他值下载失败。

参数：

1. long long \*\*pulses，8 个通道的序列的电平数据，单位 ns。
2. long long length[][8]，每个通道的segment的结束位置，开始为0,上个segment的结束位置是下个segment的起始位置。
3. unsigned short\* loop，每个 Segment的循环次数。
4. unsigned short segmentNum，Segment个数。

**函数 int ASG8x00\_CounterDownload(long long \*\*counter, long long \*\*length, unsigned short\* loop, unsigned short segmentNum)**

函数功能：用来下载COUNT使能脉冲序列。

返回值： int 类型， 0下载成功，其他值下载失败。

参数：

1. long long \*\*pulses，2 个通道的序列的电平数据，单位 ns。
2. long long length\*\*length，每个通道segment的结束位置，开始为0,上个segment的结束位置是下个segment的起始位置。
3. unsigned short\* loop，每个 Segment的循环次数。
4. unsigned short segmentNum，Segment个数。

**函数 int ASG8x00\_SetCallMode (int mode = 1)**

函数功能：设这dll的数据被动返回给调用者，dll不设置回调函数，设置后数据通讯由调用者主动发起，一般在labview中使用。

返回值： int 类型， 0设置成功，其他值失败。

参数：int 类型，0是正常的模式，可以设置回调函数。1是主动模式，调用者主动接收数据。

**函数 int ASG8x00\_AsgDownloadByFile(char \*file)**

函数功能：下载ASG序列，文件类型与软件里类型格式一致。

返回值： int 类型， 0下载成功，其他值失败。

参数：**char \*** 类型，文件路径。

**函数 int ASG8x00\_CounterDownloadByFile(char \*file)**

函数功能：下载count使能序列，文件类型与软件里类型格式一致。

返回值： int 类型， 0下载成功，其他值失败。

参数：**char \*** 类型，文件路径。

**函数 int ASG8x00\_GetErrorMsgStr(int \*type, int \*length, char\* value)**

函数功能：获取当前dll的状态，代替dll回调调用者，获取到错误的状态后调用者主动发起停止关闭等指令。

返回值： int 类型， 0获取成功，其他值失败。

参数：

1. int \*type，错误类型。
2. int \* length，错误信息长度。
3. char\* value错误信息内容地址。

**函数 int ASG8x00\_GetCountDataByCaller(int \*channel, int \*seg, int \*length, unsigned int\* data)**

函数功能：获取count采集到的数据。

返回值： int 类型， 0获取成功，其他值失败。

参数：

1. int \* channel，通道。
2. int \*seg，段编号。
3. int \*length，数据区的数据点个数，每个点4个字节。
4. unsigned int\* data，数据区地址。

## 1.2C++调用事例

#include<iostream>

#include ASG8xxxAPI.h”

void onASGCallback(int type, int length, char\* info)

{

    printf("type = %d  info = %s",type,data);

}

unsigned int dataList[4096]{};

void onCountCallback(int channel, int segment, int length,  unsigned int\* data)

{

    printf("channel = %d  segment = %d  length = %d ",channel,segment，length);

    for(int len = 0; len<length; len++)

    {

        dataList[len] = data[len];

        //printf(" %d ",dataList[len]);

    }

}

int main()

{

    ASG8x00\_Connect ();//连接

    ASG8x00\_SetErrorCallback(onASGCallback);//设置字符串回调函数

    ASG8x00\_SetCountCallback(onCountCallback);//设置count数据采集回调函数

    long long asg\_channelData[]={25,25,25,25};//asg单个channel完整序列

    long long asg\_channellength[]={2,4};//每个segment序列长度,,segment的结束位置，开始为0,上个segment的结束位置是下个segment的起始位置

    long long \*asg\_length[8];//8每个segment序列长度

    long long \*asg\_gData[8];//asg波形序列

    unsigned short asg\_loop[] = {1,2};//每个segment的loop

    unsigned short asg\_segmentNum = 2;//总段数

    for(int i=0;i<8;i++)

    {

        asg\_gData[i] = channelData;

        asg\_length[i] = asg\_channellength;

    }

    ASG8x00\_AsgDownload(asg\_gData,length,asg\_loop,asg\_segmentNum);//下载asg波形

    long long count\_channelData[]={25,25,25,1000000000};//count单个channel完整序列，

    long long count\_channellength[]={2,4};//每个segment序列长度,,segment的结束位置，开始为0,上个segment的结束位置是下个segment的起始位置

    long long \*count\_length[2];//2每个segment序列长度

    long long \*count\_Data[2];//2个通道的count信息

    unsigned short count\_loop[] = {1,2};//每个segment的loop

    unsigned short count\_segmentNum = 2;//总段数

    for(int i=0;i<2;i++)

    {

        count\_Data[i] = count\_channelData;

        count\_length[i] = count\_channellength;

    }

    ASG8x00\_CounterDownload(acount\_Data,count\_length,count\_loop,count\_segmentNum);//count使能下载

    ASG8x00\_AsgSetHightLevel(0);//配置长高电平，高8位对应8个通道，1为长高，0为正常播放， 范围0x00xx - 0xffxx

    ASG8x00\_AsgSetChannelEnable(0x1ff);//配置asg的开关0x1ff打开  0x0ff关闭

    ASG8x00\_SetClockAndWorkMode(0x80,0x80);//in1 和 in2 都开启计数功能

    getchar();//任意键开始播放

    ASG8x00\_StartPlay(0);//开始播放asg，开始采集count,无循环次数，一直播放

    getchar();//任意键停止播放

    ASG8x00\_StopPlay();//停止asg播放，停止count采集

}

## 1.3Python调用事例

import sys

from ASG8x00PySDK import \*

from ctypes import \*

import ctypes

import threading

import time

#本示例asg和count同时使用

#提示信息回调函数，打印提示信息

@CFUNCTYPE(None, c\_int, c\_int, c\_char\_p)

def error\_callback(type, length, data):

    if type == 2:

        print("type = ",type,"length = ",length,"data = ",data)

    if type == 3:

        if data[0] == 1:

            exit("采集速度太快")

        elif data[0] == 2:

            exit("接收段错乱")

    if type == 4:

        exit("设备断开连接")

#count数据回调函数，这里接收到数据直接打印，不保存

@CFUNCTYPE(None, c\_int, c\_int, c\_int, POINTER(c\_uint32))

def count\_callback(channel, seg, length,c\_data):

    print("channel = ", channel, "seg = ", seg, "length = ", length)

    data1 = []

    data2 = []

    for i in range(length):

        if channel == 0:

            data1.append(c\_data[i])

        else:

            data2.append(c\_data[i])

    if channel == 0:

        print("data1", data1)

    else:

        print("data2", data2)

#asg序列

#   asg\_data = [(segment1),(segment2),...,(segmentn)]     asg\_Data由segment列表组成

#   segment = (loop,[(row1),(row2),...,(rown)])           segment由循环和行列表组成

#   row = (time,[ch1,ch2,ch3,ch4,ch5,ch6，ch7，ch8])       row由电平持续时间和各通道在这一行的电平组成

asg\_data = [

    (1, [

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (1000, [0, 0, 0, 0, 0, 0, 0, 0])

         ]),

    (1, [

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 1]),

         (25, [0, 0, 0, 0, 0, 0, 0, 0]),

         (25, [1, 1, 1, 1, 1, 1, 1, 0]),

         (1000, [0, 0, 0, 0, 0, 0, 0, 0])

         ])

]

#count序列

#   count\_data = [(segment1),(segment2),...,(segmentn)]   count\_data由segment列表组成

#   segment = (loop,[(row1),(row2),...,(rown)])           segment由循环和行列表组成

#   row = (time,[ch1,ch2])                                row由电平持续时间和各通道在这一行的电平组成

count\_data = [

    (1, [(50, [1,1]),

         (25, [0,0]),

         (80, [1, 1]),

         (25, [0, 0]),

         (500, [1, 1]),

         (900, [0, 0])

         ]),

    (1, [(50, [1,1]),

         (25, [0,0]),

         (80, [1, 1]),

         (25, [0, 0]),

         (100, [1, 1]),

         (1000000000, [0, 0])

         ])

]

#实例化对象

asg8100 = ASG8x00()

#配置提示信息回调函数

ret = asg8100.error\_callback(error\_callback)

if ret != 0: exit(ErrorCode[ret])

#配置count数据回调函数

ret = asg8100.count\_callback(count\_callback)

if ret != 0: exit(ErrorCode[ret])

#连接设备

ret = asg8100.connect()

if ret != 0: exit(ErrorCode[ret])

#配置asg和count使能

'''

高8bit表示ASG输出常高电平控制:   每一bit对应一个通道，为1时，高电平输出，0时，低电平输出。

低8bit表示ASG通道开关:   Bit[0] 表示counter计数功能开关

'''

iLevel = 0

ret = asg8100.AsgSetHightLevel(iLevel)

if ret != 0: exit(ErrorCode[ret])

iEnable = 0x1ff

ret = asg8100.ASG8x00\_AsgSetChannelEnable(iEnable)

if ret != 0: exit(ErrorCode[ret])

#配置IN1、IN2功能

'''

高8bit表示IN1的模式

7       0：时钟模式     1: Counter模式，bit[6:0]无效

6       0: 内部时钟，bit[5:0]无效   1： 外部时钟

5-0     1：外部时钟10M  02为外部时钟100M

低8bit表示IN2的模式

7       0：触发模式   1:  Counter模式，bit[6:0]无效

6-0     00: 正常工作模式，输出只受播放停止命令控制   01：trigger模式，播放后收到trigger信号才输出

'''

IN1 = 0x80

IN2 = 0x80

ret = asg8100.SetClockAndWorkMode(IN1,IN2)

if ret != 0: exit(ErrorCode[ret])

#下载asg数据

ret = asg8100.AsgDownload(asg\_data)

if ret != 0: exit(ErrorCode[ret])

#下载count数据

ret = asg8100.CounterDownload(count\_data)

if ret != 0: exit(ErrorCode[ret])

input("enter to start：")#回车开始播放

#计算segment是否合法

Loop = 10

seg\_num\_count = len(count\_data)

seg\_num\_asg   = len(asg\_data)

#开始播放(播放次数)

rep = seg\_num\_asg\*Loop

ret = asg8100.start()

if ret != 0: exit(ErrorCode[ret])

input("enter to stop：")#回车停止播放

#停止播放

ret = asg8100.stop()

if ret != 0: exit(ErrorCode[ret])

## 1.4错误码

0:"成功",

        1:"设备未连接",

        2:"设备断开连接",

        3:"握手超时",

        4:"握手数据错误",

        5:"关闭等待超时",

        6:"关闭应答数据错误",

        7:"无设备",

        65537:"错误信息回调函数为空",

        65538:"count数据回调函数为空",

        65539:"ASG通道设置参数错误",

        65540:"COUNT使能参数错误",

        65541:"工作模式参数错误",

        65542:"工作模式参数错误",

        65543: "asg和count段信息不相同",

        131073:"ASG脉宽必须是1ns的正整数倍",

        131074:"ASG脉宽必须是2ns的正整数倍",

        131076:"ASG脉宽必须是4ns的正整数倍",

        131077:"ASG段数错误",

        131078:"最大脉宽1.6s",

        131079:"最大总时长8260s",

        131080:"SG循环必须大于0",

        19609:"COUNT脉宽是5的整数倍",

        19610:"COUNT高电平最小10ns",

        19611:"COUNT低电平最小20ns",

        19612:"COUNT最小以25ns低电平结束",

        19613:"COUNT读取线程已存在",

        19614:"count采集速度太快",

        19615:"COUNT最大脉宽5497558138880",

        19616:"COUNT循环必须大于0",

        19617:"COUNT采集停止超时",

        196616:"count接收段数错误"