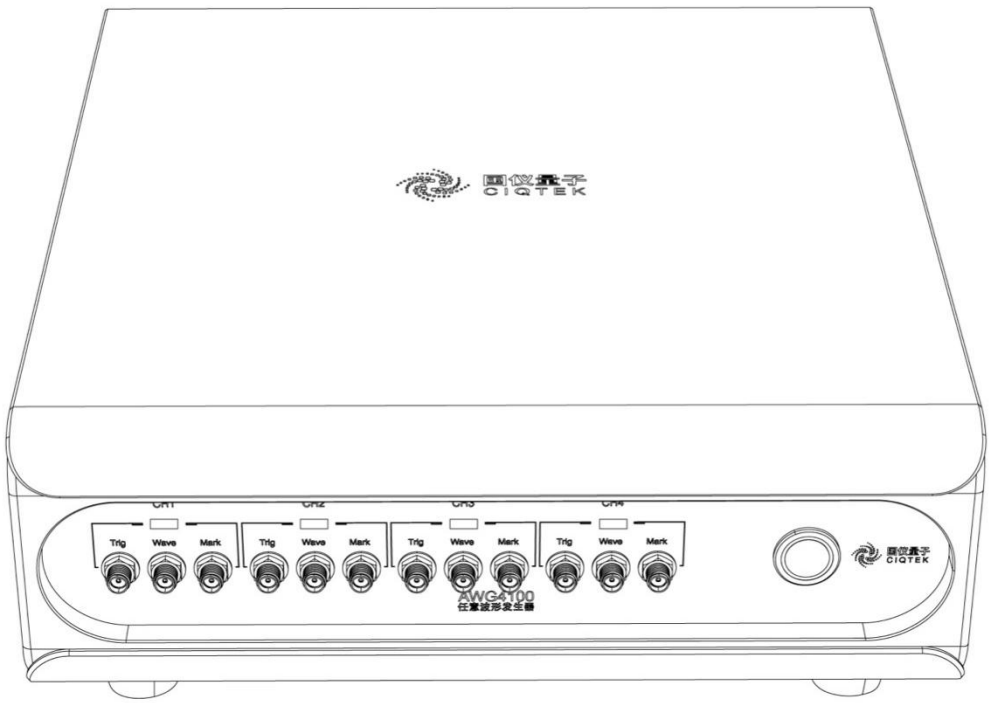
用户手册

AWG4100 任意波形发生器



国仪量子
CIQTEK

保证和声明

软件版本

软件升级可能会修改产品功能，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。

质保

公司货物免费保修一年,时间自最终验收合格并交付使用之日起计算。

联系我们

- 电子邮箱：gylz@ciqtek.com,
- 电　　话：4000606976,0551-63367168
- 企业官网：www.ciqtek.com

安全要求

一般安全概要

为避免可能的危险，以及防止损坏本产品和与本产品连接的任何设备，用户需了解以下安全措施，并按照规定使用本产品。

使用正确的电源线

使用我司所提供的电源线。

确保供电电源正确

为避免对操作人员造成伤害或损坏产品，请在使用产品前仔细阅读本手册，并确保产品供电电源正确。

保持良好的散热条件

为避免因电路板过热而损坏，在使用本产品的过程中请勿堵住通风口。

请勿在潮湿环境下操作仪器

为避免产品内部电路出现短路等危险情况，请勿在潮湿环境下操作仪器。

请勿靠近易燃易爆物品

为避免人身伤害或产品损坏，严禁易燃易爆物靠近本产品。

注意搬运安全

为避免对产品面板上的按键、接口、指示灯等部件造成损坏，请注意搬运安全。

远离高温环境

为避免发生危险，严禁将本产品放置于高温环境中。

严禁不具备操作能力的人使用本产品

为避免造成人身伤害或产品损坏，严禁不具备操作能力的人（如老人、儿童）使用本产品。

保养与清洁

请经常对产品进行清洁，方法如下：先断开电源，再用干抹布轻轻擦拭产品外部。

目录

保证和声明 I

安全要求 II

1、产品介绍 5

1.1 检查运输包装 5
1.2 检查包装内容 5
1.3 检测产品是否合格 5
1.4 检查产品尺寸 5
1.5 连接电源 7

2、产品特征 8

2.1 前面板介绍 8
2.2 后面板介绍 9
2.3 指标参数 10

3、软件安装 12

4、软件使用 14

4.1 打开软件 14
4.1.1 设备和上位机连接 *14*
4.1.2 打开软件 *15*
4.2 软件界面介绍 16
4.2.1 连接界面 *16*
4.2.2 操作主页面 *17*
4.3 基本波形播放 26
4.3.1 线路连接 *26*
4.3.2 DDS 参数设置 *28*
4.3.3 AWG 波形编辑 *29*
4.4 AWG 波形自定义开发手册 31
4.4.1 波形编辑 *31*
4.4.2 编辑和下载 *39*

5、接口程序调用 42

5.1 调用说明 42
5.2 C++ API 接口程序及调用注意事项 43
5.2.1 C++ API 接口 *43*
5.2.2 C++ API 调用注意事项 *51*
5.2.3 C++ API 接口调用示例 *51*
5.3 PYTHON 接口程序及调用注意事项 52
5.3.1 Python 接口程序 *52*
5.3.2 Python 接口示例 *58*
5.3.3 Python 接口调用注意事项 *62*
5.4 LABVIEW 接口调用及注意事项 63
5.4.1 LabVIEW 接口调用流程参考 *63*

6、常见问题 65

6.1 计算机无法 PING 通设备 65
6.2 计算机可以识别设备但软件中设备连接失败 65
6.3 设置正确但无信号输出 65

6.4 连接正常，设置波形后有输出，但是输出异常：..... 65
7、注意事项 68

1、产品介绍

1.1 检查运输包装

用户收到产品后，请先检查包装是否完整，若包装已损坏，请保留被损坏的包装，并及时与我司联系。

1.2 检查包装内容

请根据装箱单检查随机附件，请确认

- 一台 AWG4100 仪器
- 一本 AWG4100 用户手册
- 一根电源线
- 一根网线
- 一张合格证

如有损坏或缺失，请及时与我司联系。

1.3 检测产品是否合格

如产品存在机械损坏，或者产品未通过性能测试，请及时与我司联系。

1.4 检查产品尺寸

AWG4100的尺寸如图1.1所示。

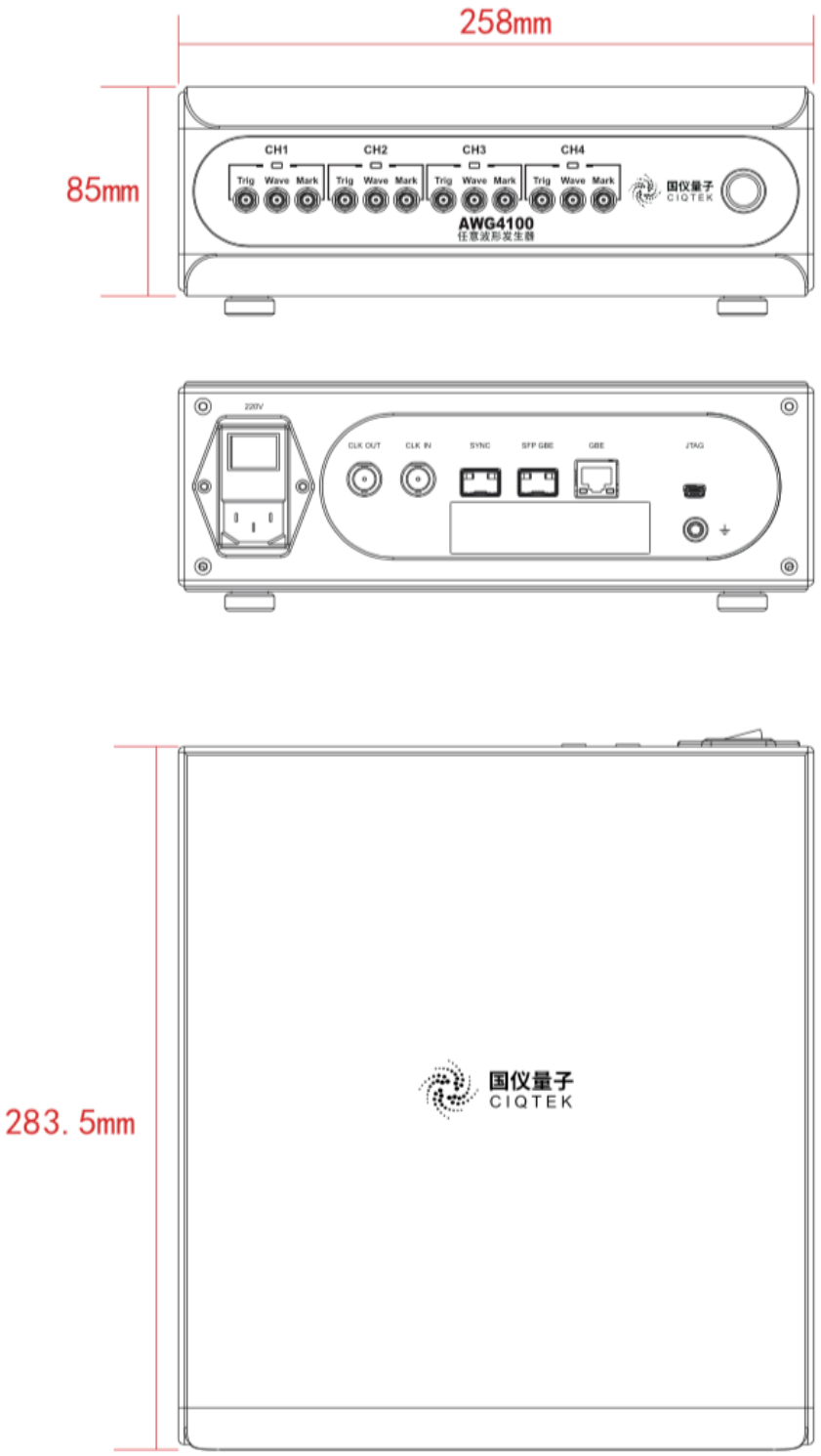


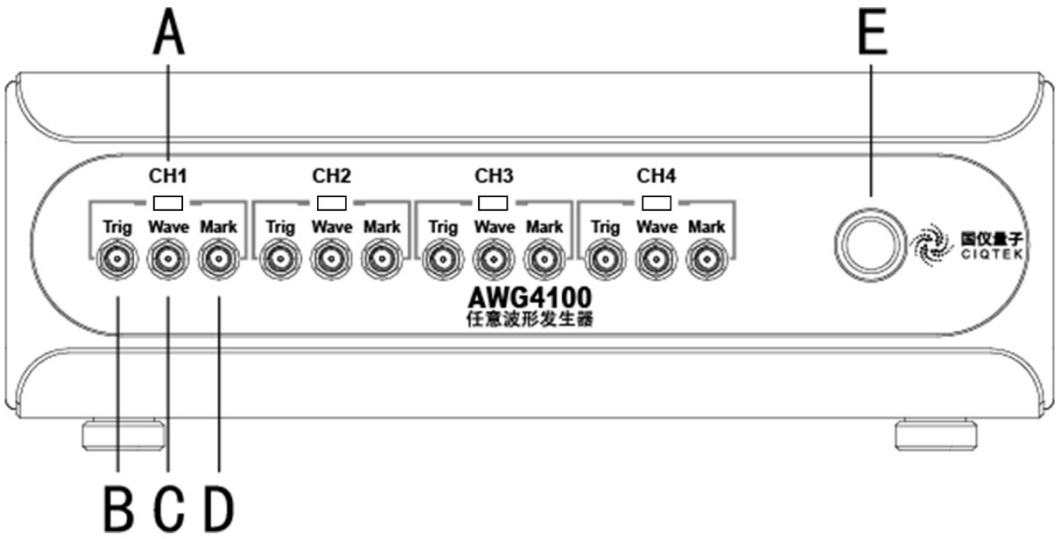
图 1.0-1 AWG4100 外观尺寸

1.5 连接电源

可使用随机附件提供的电源线将AWG4100与220V的交流电进行连接。接通电源后，按下机箱背面电源上方的电源开关，然后按下前面板的设备开关按钮，可以看到指示灯亮起，表示仪器已经处于工作状态，配合上位机就可以进行后续操作。

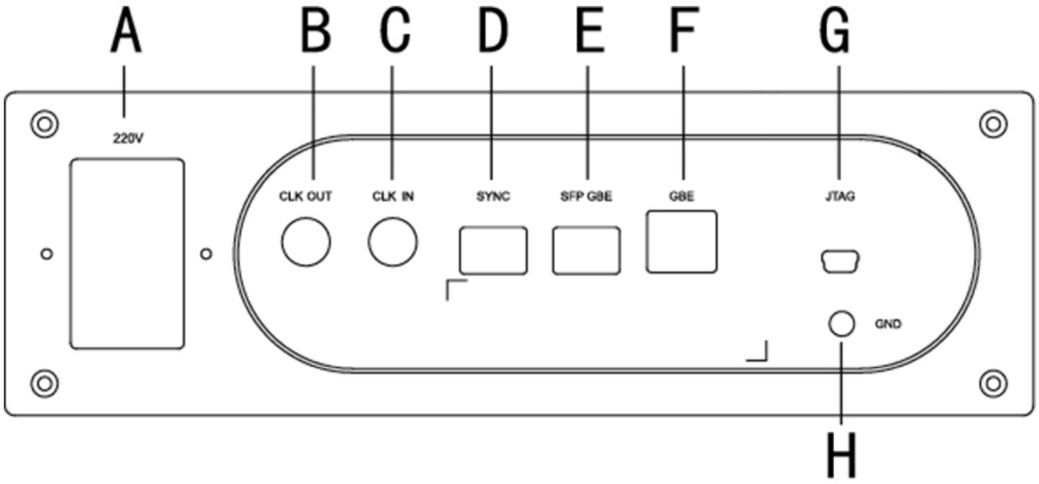
2、产品特征

2.1 前面板介绍



编号	接口名称	描述
A	LED 指示灯	通道播放过程中,该 LED 灯会亮起
B	Trig	TTL 触发输入
C	Wave	单端波形信号输出
D	Mark	TTL Marker 信号输出
E	开关	开关、电源指示灯

2.2 后面板介绍



编号	接口名称	描述
A	交流 220V	电源入口+电源开关
B	参考时钟输出	参考时钟输出(10MHz)与其他仪器同步
C	参考时钟输入	参考时钟输入(10/100 MHz)与其他仪器同步
D	同步接口	仪器之间的同步总线连接器
E	SFP GBE	需要插入光模块后，可以使用千兆光网络
F	GBE	千兆网口(网线连接请用此端口)
G	JTAG	用于芯片内部测试以及对系统进行仿真、调试
H	GND	接地端口

2.3 指标参数

核心参数	
通道数	4
垂直分辨率	16
每个通道的波形存储器	512 MSa
波形粒度	4 Sa
最小波形长度	88 Sa
序列编程器时钟频率	300 MHz
标记输出	1 个标记/通道， SMA 连接器（前面板）
参考时钟输入/输出	BNC 连接器 （后面板）
参考时钟输出幅值	3.3Vpp(1MΩ)
参考时钟输出频率	10 MHz
模拟带宽	330MHz
波形输出通道	
连接器类型	SMA
输出阻抗	50 Ω
输出耦合	DC
输出模式	直接输出
输出范围	± 0.5 V （into 50 Ω）
偏置调节范围	± 0.5 V
偏移电压	< 1 mV （into 50 Ω）
相位噪声	< -110 dBc/Hz （100 MHz@10kHz）
电压噪声	< 20 nV/√Hz @100kHz
触发到输出延迟	<300ns
输入	
触发输入	1 路/通道， SMA 连接器（前面板）
触发输入阻抗	50 Ω
触发输入幅值范围	1.2～3.3 V (into 50 Ω)
参考时钟输入	BNC 连接器 （后面板）
参考时钟输入频率	10/100 MHz
时域和频域特性	
采样率	1.2 GSa/s
上升时间（20% 到 80%）	< 1 ns （± 500 mV）
通道间偏差	< 150 ps
SFDR	74dB (@10MHz) 43dB (@300MHz)
谐波失真	HD2 -48db@100 MHz

	HD3 -46db@100 MHz
连接及其他	
主机连接	LAN/以太网，1 Gbps
尺寸	258 mm * 291.88 mm * 93 mm
重量	3 kg

3、软件安装

用户可到我司的官网 https://www.ciqtek.com 产品中心→量子测控系列产品→任意波形发生器（AWG4100）产品介绍界面下方自行下载 AWG4100 的上位机软件安装包到本地后进行安装。具体安装步骤如下：

1、双击 AWG4100.exe 的图标（如图 3.1.1）进入软件安装向导界面，见图 3.1.2。

图 3.1.1 安装包文件图标

2、在安装向导页面可以选择安装的软件语言，选好语言之后可以点击“快速安装”直接进入安装过程，直接安装到默认 D:\Program Files(X86)文件夹；也支持用户自己选择安装目录，点击“自定义选项”进入安装路径选择页面，如图 3.1.3，选择好路径后，点击“快速安装”后进入安装页面，如图 3.1.4。

图 3.1.2 安装向导页面

图 3.1.3 安装路径选择页

3、显示安装的详细过程和信息，完成安装后，进入安装完成页面，可以点击“立即体验”直接进入软件页面。

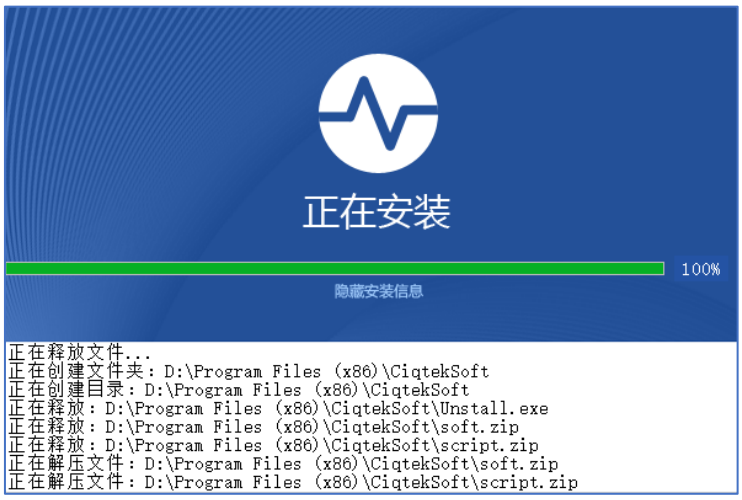


图 3.1.4 安装过程

4、软件使用

4.1 打开软件

4.1.1 设备和上位机连接

如果是将设备和上位机通过网线直连，或者两者是通过交换机进行连接时，都需要先确认自己的电脑 IP 配置情况，确保电脑通过 DHCP 方式或者手动配置方式配置了 IP 地址。通过打开上位机的“网络和 Internet“设置—”更改适配器选项“页面，找到对应的网络连接，鼠标右键打开”属性“—”Internet 协议版本 4（TCP/IPv4）“---”属性“下配置固定 IP 地址。

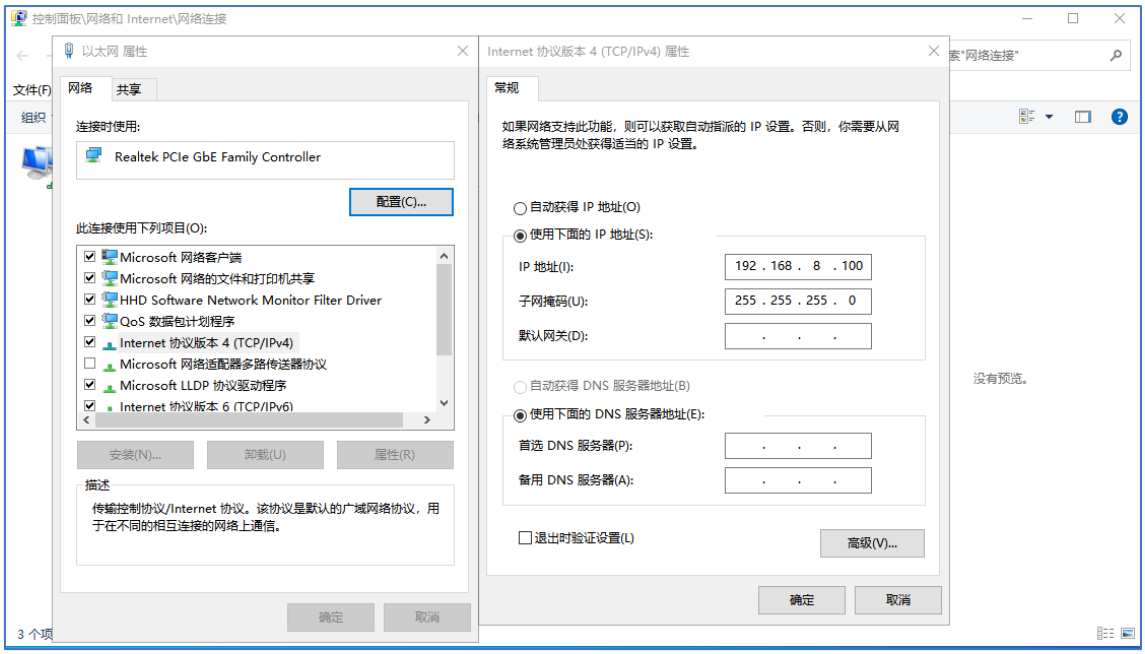


图 4.1.1 上位机配置 IP 地址

由于 AWG4100 和上位机软件之间可以通过跨网段连接，所以可以在保证 IP 地址不冲突的情况下，配置任何 IP，都可以搜索到设备。(设备出厂前会配置固定的 IP 地址，一般不可更改，可以通过上位机软件搜索到)。

如果设备直连的情况下，注意观察下图 4.1.2 设备的网速是否为 1.0Gbps，AWG4100 设备目前仅支持全千兆网连接。

如果通过交换机连接的话，可以注意观察交换机和设备连接口千兆网连接指示灯是否亮起，如图 4.1.3

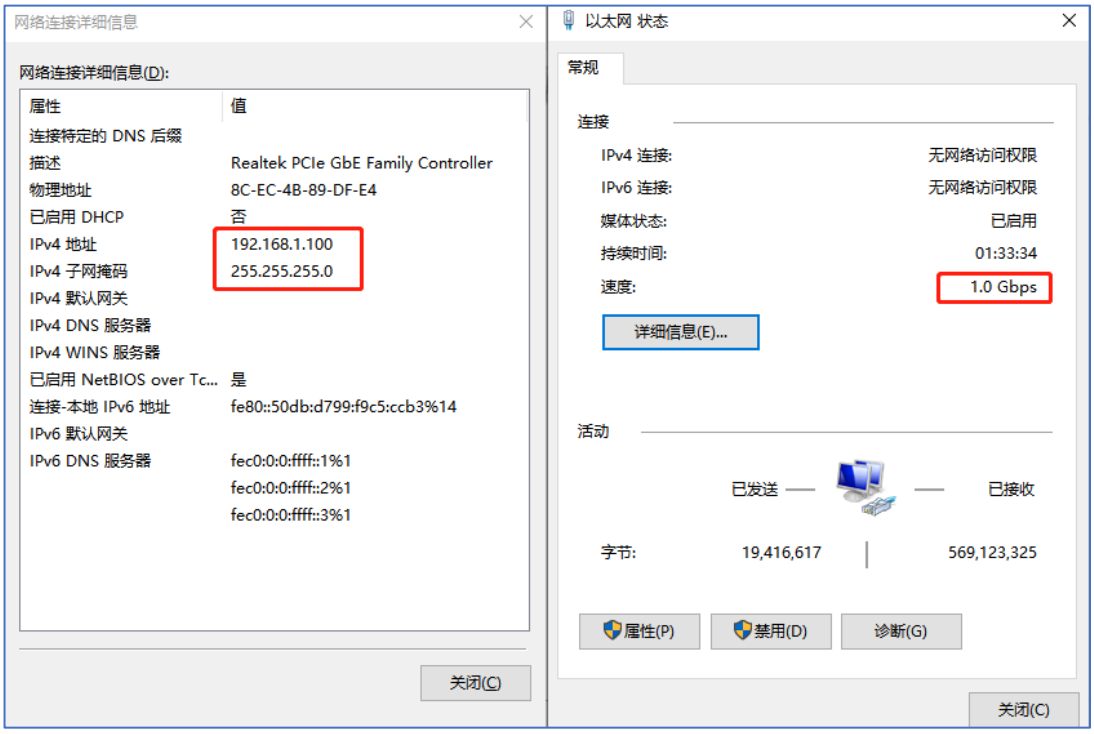


图 4.1.2 上位机软件安装 PC 的 IP 地址配置

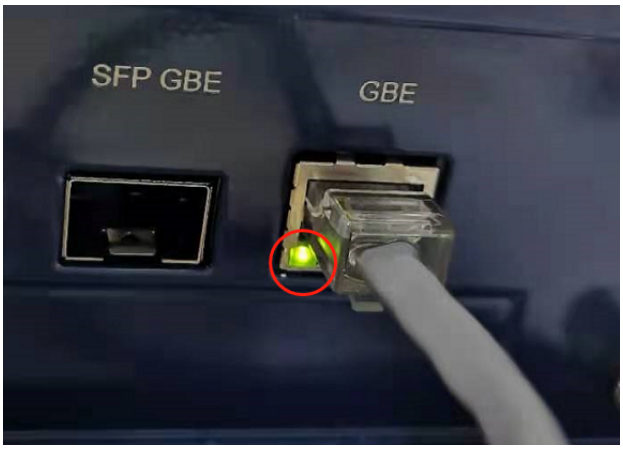


图 4.1.3 设备千兆网连接状态指示

如果上位机和设备之间是通过千兆路由器进行连接，则不需要进行 IP 地址配置，直接连接就可以使用。

4.1.2 打开软件

AWG4100 软件默认安装会在桌面建立快捷方式，可以双击桌面快捷方式打开软件，如下图 4.1.4 所示为 AWG4100 的上位机软件快捷图标。

图 4.1.4 设备快捷图标

此外还可以在开始菜单里面的软件列表里面找到 AWG4100 软件图标打开软件。打开软件后，软件首先进入设备的连接界面，界面如 4.2 章节所示。

4.2 软件界面介绍

4.2.1 连接界面

图 4.2.1 设备连接

1、 设备标识 Logo 区。

2、 窗口最小化和关闭区。

3、 局域网搜索到设备列表，AWG4100 的上位机软件和设备之间可以跨网段方式连接，且需要千兆以太网。

4、“刷新”用户触发该按钮，软件再次在局域网内部进行设备信息的搜索，搜索后更新设备列表页中的信息展示。

5、“连接” 用户触发该按钮，执行软件和对应硬件设备之间的连接，连接成功后，则该设备的状态修改为“已连接” ，同时跳转至实验主页面。

注意：电脑和设备之间连接需要满足千兆网协议，如果未搜索到设备请检查设备端和电脑端是否都满足千兆网协议，如果使用交换机连接，需要使用千兆网交换机。

4.2.2 操作主页面

图 4.2.2 操作主页面

操作主页面主要分为 5 个基本功能区：

1 全局参数设置区：用于设置播放模式，参考时钟信号，参数的设置以及保存，设备设置功能，窗口的最小化、最大化，关闭。

2 DDS 模式和 AWG 模式参数设置区：用于独立组合模式设置，对应模式的参数设置。

3 AWG 模式自定义代码编辑区。

4 自定义代码的编译信息区。

5 AWG 模式自定义代码的预览窗口。

注意：在关闭软件或断开连接后重新来打开连接进入操作页面，软件会自动读取设备参数并显示（自定义代码除外）。

4.2.2.1 全局参数设置区

图 4.2.3 全局参数设置区

1、 设备标识 logo。

2、“断开连接”用户触发该按钮，若断开成功，跳转至设备连接页面。

3、“播放模式”用户可在下拉框中选择“连续模式”、“上升沿触发”、“下降沿触发”和“电平触发”。

上升沿触发：需要外接 TTL 电平的触发信号，以上升沿为触发依据，触发信号范围为 1.2V-3.3V；

下降沿触发：需要外接 TTL 电平的触发信号，以下降沿为触发依据，下降沿信号下降到 1.2V 以后可以触发；

电平触发：需要外接 TTL 电平的触发信号。在 DDS 模式下，在触发信号高电平时播放波形，高电平范围为 1.2V-3.3V，低电平时即停止播放；在 AWG 模式下，在触发信号高电平时播放波形，等该序列播放完成后，再次检测是否还是高电平，如果是的话，继续播放下一个序列，如果不是，则停止播放。

4、“时钟信号”用户可在下拉框中选择内部时钟或外部时钟。

5、“载入参数”会导入 DDS 模式和 AWG 模式配置参数，除自定义的代码（代码编辑区有单独载入代码功能控件）。

6、“参数保存”会保存 DDS 模式和 AWG 模式配置参数，除自定义的代码（代码编辑区有单独保存代码功能控件）。

7、“设置”显示设备的状态信息

图 4.2.4 状态显示栏

设备的状态显示在设置栏中，绿色表示正常，红色表示异常。

‘PLL1/PLL2’：表示整个设备的时钟状态是否锁定，绿色表示锁定，红色表示未锁定。设备本身或与其他设备做时钟同步的时候，需要 PLL1/PLL2 都锁定才能正常工作。

‘通道 DDR3’：AWG4100 具有 4 片独立编址的 DDR 颗粒，当相应通道 DDR3 正常工作时，显示为绿色，异常为红色。

‘系统初始化’：系统在设备上电时，会加载相应的系统参数，当加载成功时候，显示绿色，失败为红色。

‘DAC 初始化’：当设备上电时，对 DAC 数模转化芯片进行配置，当配置成功，显示绿色，配置失败显示红色。

‘千兆网口初始化’：当网络正常连接时，状态显示绿色，异常连接显示红色。

‘AWG 波形加载’、‘AWG_EN 加载’和‘AWG_Marker 加载’为 AWG 模式下载自定义波形的诊断状态，当所有的状态显示绿色的时候，才能播放正确的波形数据。当诊断自定义的波形状态时候，可以查看这 3 个状态提示。

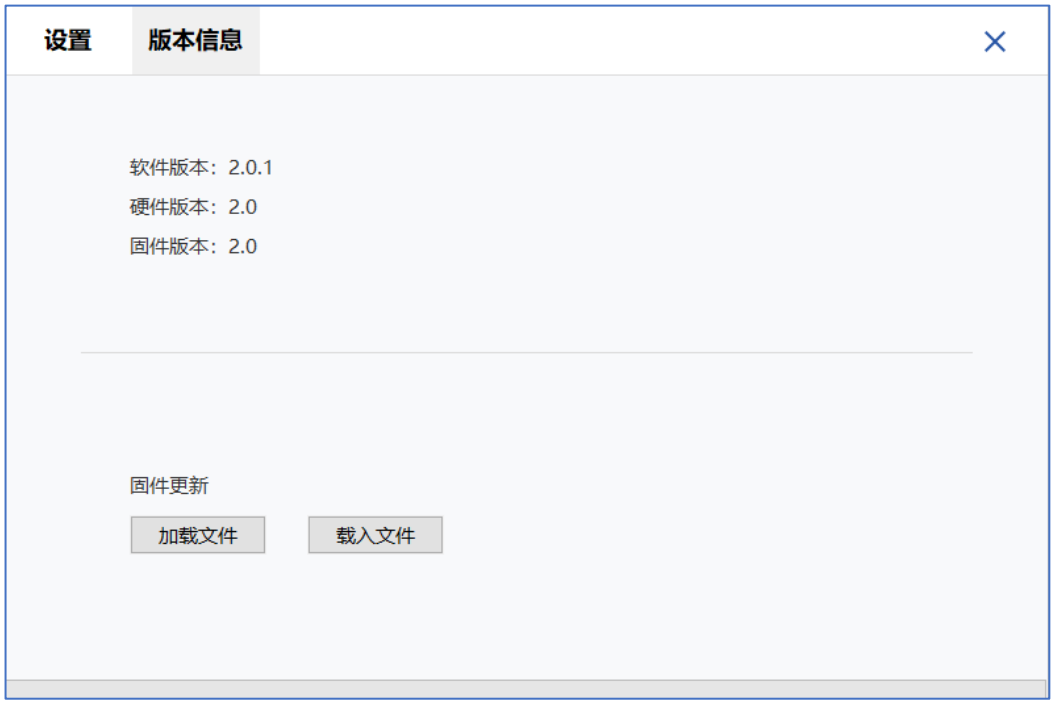


图 4.2.5 版本信息和固件更新

‘版本信息’页面显示，设备的软件、硬件、固件的版本信息，同时本页面可以提供更新设备固件的功能。加载文件为选择需要加载（或下载）的文件，载入文件为将选中的文件下载到设备中。

当提示文件成功下载到设备后，需要断电重启设备，检查固件版本的变化。当下载失败时，可能由于网络等原因导致，可以再次尝试更新，如果仍有问题，请可以通过邮箱（gylz@ciqtek.com）或者电话（4000606976，0551-63367168）联系我司。

固件的升级包文件可以到我司的官网 https://www.ciqtek.com 产品中心→量子测控系列产品→任意波形发生器（AWG4100）产品介绍界面下方，下载 AWG_Firmware******.rar（***为版本号，注意下载时选择最新的版本）的固件升级包文件。

4.2.2.2 DDS 模式和 AWG 模式参数设置区

图 4.2.6 DDS 模式和 AWG 模式参数设置区

1、“下载模式”用户可选择独立下载模式或组合下载模式。独立模式下，用户可独立对四个通道进行操作，需要分别操作通道模块下的下载与播放按钮，控制波形的下载与输出；组合模式下，用户可将四个通道随机组合，可统一下载、播放波形，此时四个通道的波形相位延时小于 150ps。

2、AWG 模式与 DDS 模式下分别对应不同的参数，这两个模式是互斥的，任意一种模式处于工作过程中，必须停止播放，才能切换成另外一种模式。DDS 模式下，输出的波形为正弦波，可以设置的参数为频率（1Hz—330MHz），相位（0-2π），振幅和偏置（振幅和偏置相加的绝对值不超过 500mV）。AWG 模式需要配合自定义代码，才能输出相应的波形序列。AWG 模式支持设置的参数包括：Marker 开关和空闲偏置。Marker 开关：如果需要使用 Marker 功能，需要打开这里的开关；空闲偏置为波形序列播放的间隙输出波形的幅度。

注意：组合模式下，四个通道相互组合，当需要触发模式播放相应的 AWG 波形序列的时候，需要将触发信号接到选择通道中，序号最小的那个通道，例如下图，2,3,4 三个通道在组合模式选择 AWG 模式，如果需要触发播放的时候，只需要将触发信号接到 2 通道的 Trig 口，而无需要接到 1,3,4 通道的 Trig 口。

图 4.2.7 下载模式选择

4.2.2.3 AWG 模式代码编辑区

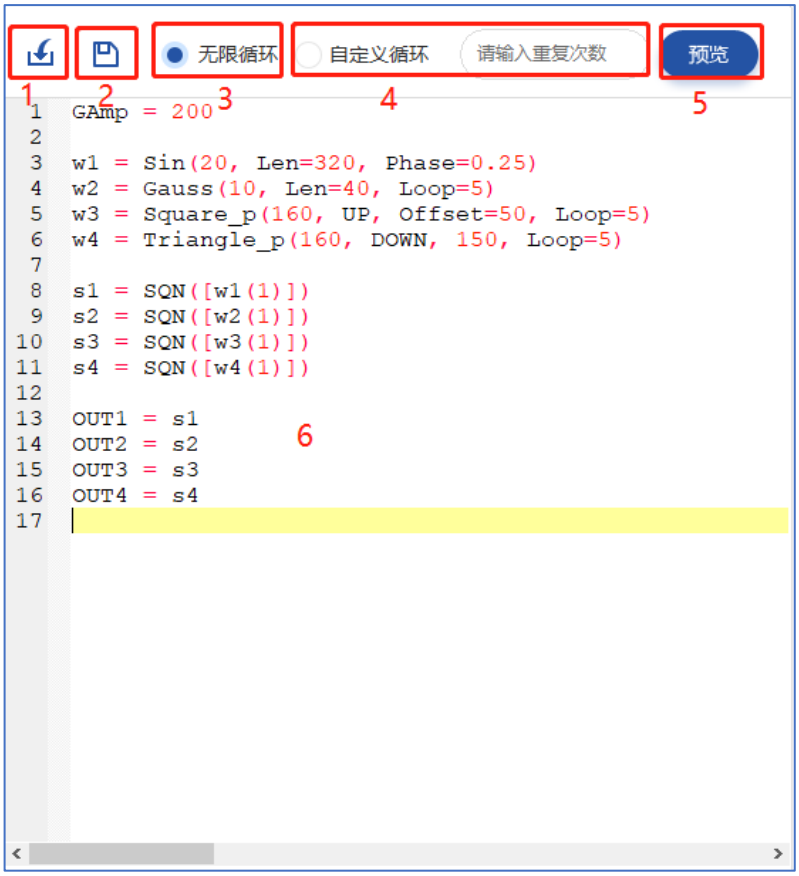


图 4.2.8 AWG 模式参数设置区

1 序号 1 的按钮是载入编辑好的自定义代码（自定义代码的后缀为.cw）

2 序号 2 的按钮是保存编辑好的自定义代码（自定义代码的后缀为.cw）

3 序号 3 的按钮是选中代码中自定义的通道波形无限循环，不限次数，直到点击停止按钮。

4 序号 4 的按钮是选择自定义的通道波形，按指定的次数播放，播放完成后，通道就无波形输出。按钮 3 和按钮 4 只能二选一。

5 序号 5 的按钮是预览代码区域定义好的自定义代码，相应的通道波形在预览区显示波形。

4.2.2.4 AWG 模式自定义代码预览区

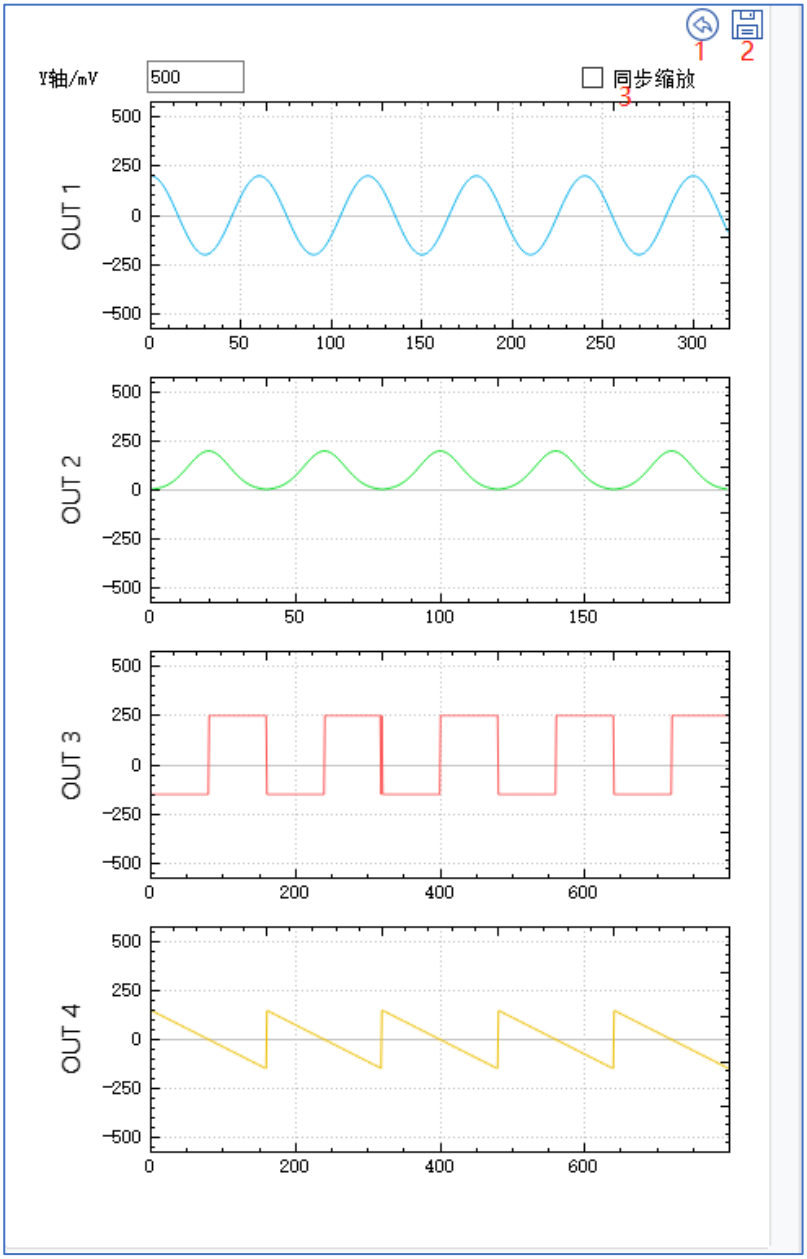


图 4.2.9 预览区

1 序号 1 的按钮是在波形被移动或者缩放之后，可以还原预览波形到最初状态。

2 序号 2 的按钮是保存预览区波形数据，波形数据会按照相应通道生成 out1.txt～out4.txt 和 out1.wave～out4.wave 文件，其中 txt 文件按点表示数据的电压值（单位 mV）,wave 文件保存二进制文件。

3 序号 3 的按钮同步缩放功能，当选中时四个通道的 X 坐标轴按照同样的坐标刻度显示；不选中时，四个通道的 X 坐标轴刻度自适应调整适配波形长度。

4 通道的波形预览窗口中横坐标显示的单位是点（一个点的时间，可以按照 1/1.2Gbps 换算，大约是 0.83ns）。

注意：预览的界面会按照实际的波形显示，但当点数较多时候，采取抽点的方式显示，可以通过鼠标滚轮缩放或者拖动的方式查看细节。当预览点数超过 512M 的时候，出于性能考虑，预览会提示失败，但是如果基础波形的总长度不超过 512M 的情况下，波形仍然可以下载。如下面的示例，OUT1=s1 的波形 w1 虽然重复次数较长（不超过 2 的 32 次方），但是基本波形长度较短，所以虽然不能预览，但是依然可以正常下载和播放。此外注意，单次播放的点数不能超过 2 的 34 次方个点，当超过限制后单次触发，超过的点数无法播放。

代码编译的结果会在代码编译信息区进行展示，包括编译成功、编译错误或者警告等。

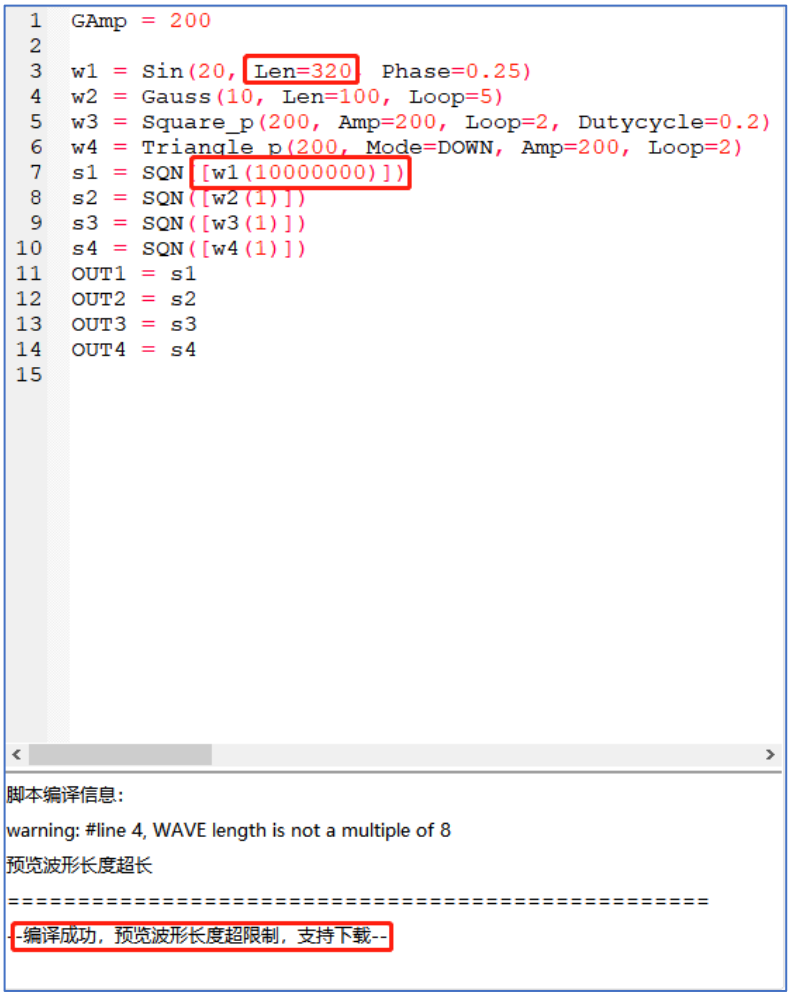


图 4.2.10 预览代码示例

4.3 基本波形播放

本模块的目的是演示 AWG4100 的基本用法。我们将演示独立与组合模式下波形的预览和生成。为了实现多通道信号的可视化，需要一个具有足够带宽和通道数的示波器（建议采用带宽 1G，采样率 4Ga/s 以上的示波器）。

4.3.1 线路连接

连接之前需要准备 AWG4100、安装好上位机软件的电脑和千兆交换机。可直接用网线连接后面板的 GBE 端口到千兆交换机，再用网线将电脑也连接到同一台交换机，具体连接如下图所示：（也可以将 GBE 口和电脑直连，需要电脑有千兆网口）

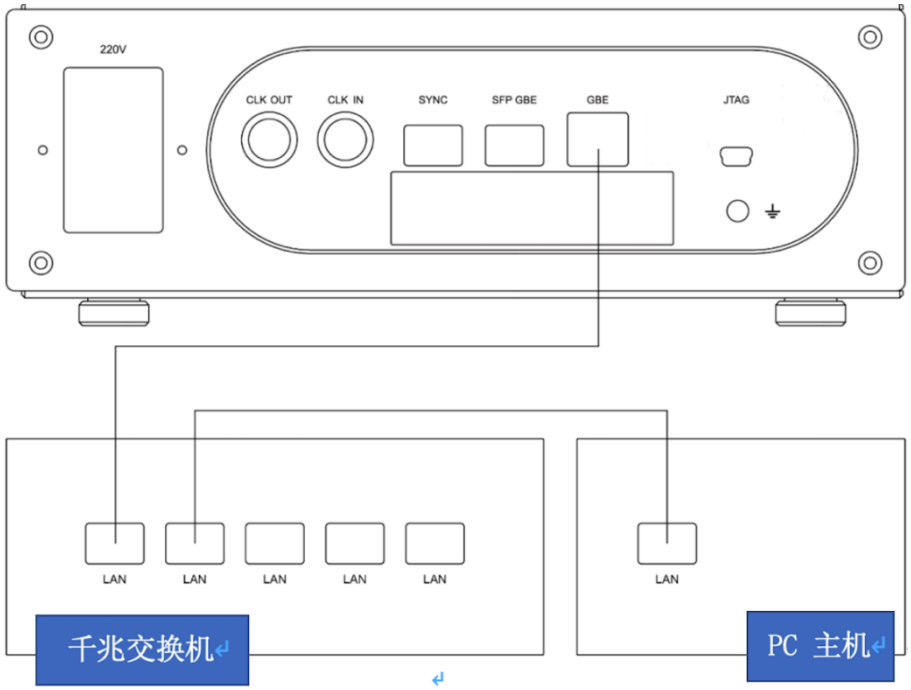


图 4.3.1 后面板连接图示

前面板只需按照如下连接图示将输出端口与示波器连接即可。

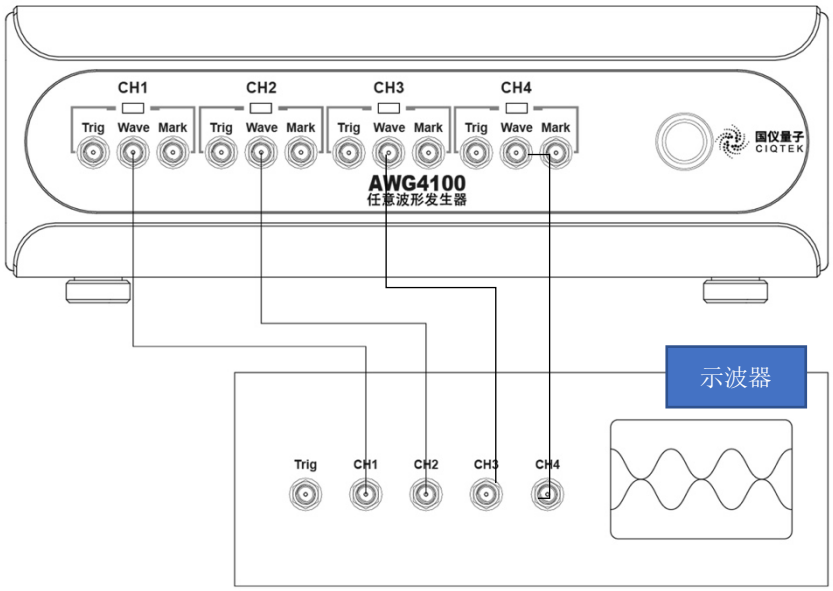


图 4.3.2 前面板连接图示

4.3.2 DDS 参数设置

独立模式下，用户可在 DDS 模式下设置参数，使四个通道输出正弦波，参数设置如图 4.3.3 所示。参数设置完毕后，分别点击每个通道的下载按钮，若下载成功，点击播放即可输出正弦波形，如图 4.3.4 所示。

图 4.3.3 参数设置表

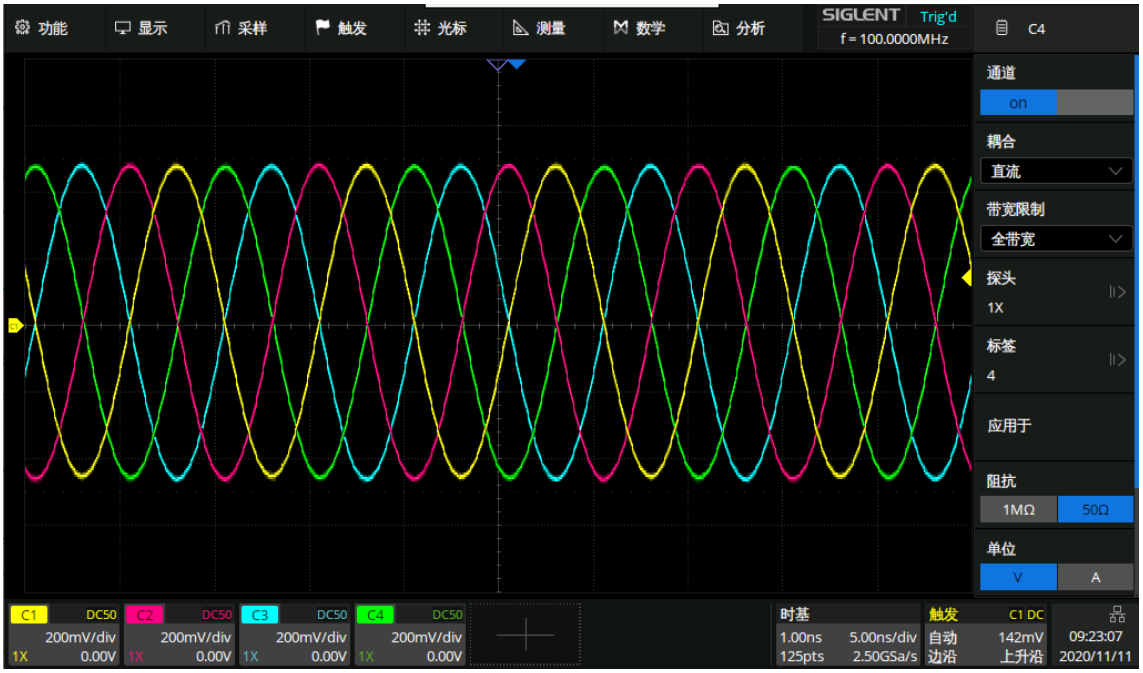


图 4.3.4 DDS 输出波形

4.3.3 AWG 波形编辑

AWG 模式波形的输出，需要在代码编辑区编辑波形序列，波形序列可以通过内置函数来生成或者是自定义的波形数据，然后将 WAVE 类型的波形转化成 SQN 或者 ADVS 类型变量，输出给 OUT1/ OUT2/ OUT3/ OUT4 通道全局变量，就可以在相应的通道输出波形。具体的代码语法编辑规则，见下面的代码编辑章节说明。

关于自定义代码的示例，到我司的官网 https://www.ciqtek.com 产品中心→量子测控系列产品→任意波形发生器（AWG4100）产品介绍界面下方下载。

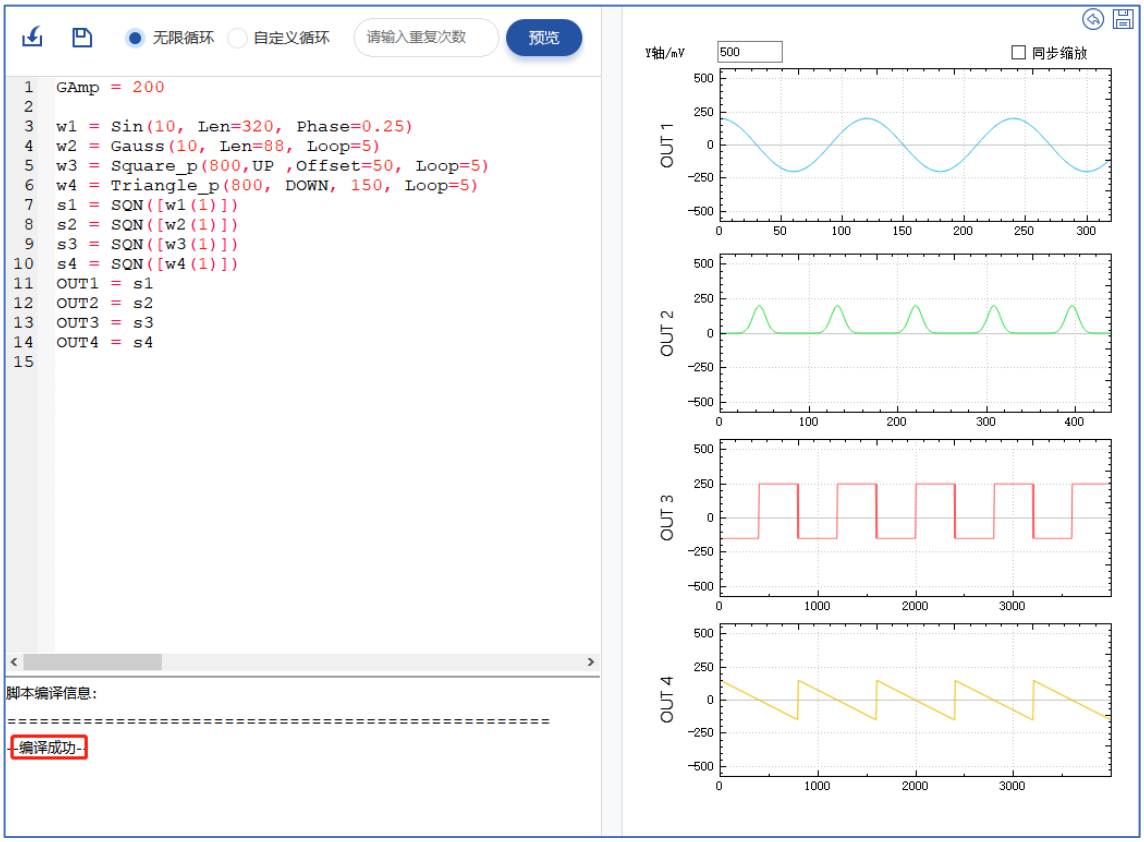


图 4.3.5 AWG 波形编辑

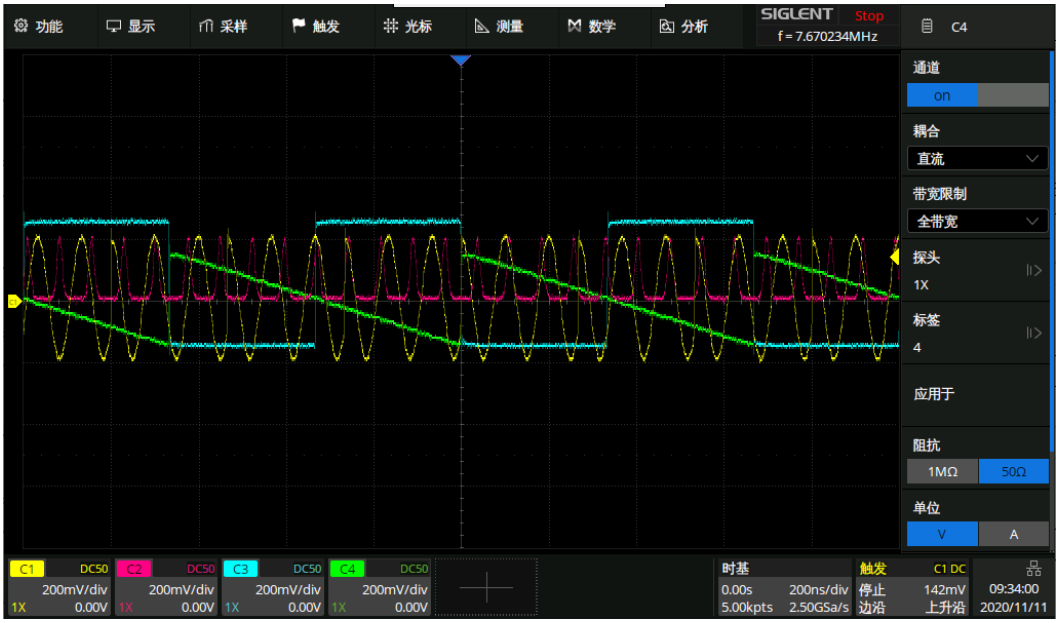


图 4.3.6 AWG 输出波形

4.4 AWG 波形自定义开发手册

自定义波形代码的基本规则如下：

1. 波形相关数据类型有 WAVE、SQN、ADVS；
2. 内置变量 OUT1、OUT2、OUT3、OUT4 表示仪器的四个输出；
3. 内置函数包括 Sin()、Gauss()、Triangle()和 Square()等分别用于生成正弦波序列、高斯脉冲序列、三角波序列和方波序列；
4. 全局变量 GAmp 用于统一指定所有波形的幅度，默认值为 100mV；
5. 注释以#开始；
6. 一条语句以英文分号“;”结尾；
7. 语句需要缩进，可以调用相关 Python 语句；
8. 函数支持可选参数和必选参数，可选参数按顺序依次给出时可以不明确指定参数名，否则必须指定参数名。
9. 内置常量，圆周率 PI；
10. 支持数值变量及其四则运算“+”、“*” ；
11. 字符串仅支持单引号；
12. 基本波形长度需要大于等于 88 点；
13. 设备的输出采用 1.2G 的采样率输出。

4.4.1 波形编辑

任意波形发生器波形编辑主要包括全局常量设置、基本函数库使用和序列的产生。

4.4.1.1 函数类型

基本函数库包含了一些常见波形的内置函数，例如正弦信号、高斯信号、方波、三角波等等，也支持用户使用 Python 语法自定义函数。

1、WAVE

WAVE 类型用来表示一个独立的波形，我们可以通过三种方法生成一个波形，一是直接指定每个采样点的电压值，二是通过文件，三是通过内置函数。

通过列表数据直接指定每个采样点的电压值生成一个波形，电压值的单位是 mV，范围 [-500.0, 500.0]。WAVE 会自动将电压值线性编码为[0, 65535]，其中

0 mV 对应 32768。通过数据列表构造 WAVE 时，mode 参数起作用，有 2 个可选值。当 Mode=’Volt’时（默认值），将数据视为电压值；当 Mode=’Code’时将数据视为已经编码的值。

```
w1 = WAVE([0,1,2,3,4,5,6,7]);      # 电压数据，0mV，1mV...
w1 = WAVE([32768], mode='code');   # 一个 0 电平
```

通过.wave 文件（本质上为 2 进制文件）加载 wave 数据。wave 文件有数据头和数据块组成，数据头和数据块的地址格式为下表所示：

地址	说明
0H00-0H03	’AWGw’波形文件标识
0H04-0H07	’\0u16’ 或 ’\0f32’ 或 ’\0f64’
0H08-0H0B	uint32，波形长度，点数
0H0B-0H0F	（空字段，占位，无用）
0H10 - ...	数据块
u16 表示数据块为编码的，16 位无符号整型	
f32 表示 mV 为单位的电压值，32 位浮点数	
f64 表示 mV 为单位的电压值，64 位浮点数	

可以使用完整的绝对路径也可以指定一个全局的目录然后在该目录下查找文件。注意：不支持相对路径。

```
    wf = WAVE('E:\\data.wave');
    WD = 'E:\\';
    wf = WAVE('data.wave');
```

注意： data.wave 数据格式为数据头+具体的数据，以下是生成数据的 Python 代码示例：

```
import sys
import time
import struct
list_dec = [150, 200, 300, 400, 500,400, 300, 220, 244, 255]
dtype = b'\x00f64'#其中数据类型，还支持 b'\x00f32'，b'\x00u16'
with open('E:\data.wave', 'bw') as wf:
    wf.write(b'AWGw') #文件头标识符
    wf.write(dtype) #数据点的表示格式
    wf.write(struct.pack('I', len(list_dec))) #文件数据的长度
    wf.write(b'\x00\x00\x00\x00')#数据位的补齐，预留
    wf.write(struct.pack('{}d'.format(len(list_dec)), *list_dec))#写入的数据
```

注意：以上是提供的 Python 脚本输出自定义波形数据文件的方法，也可以用 Matlab 等脚本语言生成数据，只要注意数据文件的头部即可。

2、Sin

函数原型为 Sin(Freq, Amp, Len, Phase, Offset, Loop)

其相关参数解释如下：

Freq: 必选，实数，正弦波频率，单位 MHz；

Amp: 可选，实数，正弦波振幅，单位 mV；

Len: 可选，正整数，采样点数；

Phase: 可选，实数，初始相位，单位2 π，默认值为 0；

Offset: 可选，实数，偏置，单位 mV，默认值为 0 mV；

Loop：可选，正整数，循环次数，默认 1；

注意：Len*Loop 长度要在 88 点到 512M 个点之间；Amp+Offset 要在正负 500mV 之间，否则有溢出。

说明：按 Amp*Sin(2*pi*(Freq*t + Phase))+ Offset 计算，生成一个频率为 Freq MHz，振幅为 Amp mV，初始相位为 Phase，具有 Len 个采样点的正弦波序列。当 Amp 的值没有指定时取 GAmp 的值，Len 的值没有指定时自动取 Len=Fs/Freq，向下取整。频率 Freq 必须指定，后面参数可选。

示例：

```
w2 = Sin(10);  #10MHz，振幅为 GAmp m，初始相位为 0 的 1 个周期的正弦波
w3 = Sin(20, 100, 200); # 20MHz，100mV，200 个采样点的正弦波
w4 = Sin(20, Amp=100, Len=200); # 与 w3 相同
w5 = Sin(25, Len=200); # 25MHz, GAmp mV，200 个采样点
```

3、Gauss

函数原型为 Gauss(Var, Amp, Len, Offset, Loop)

其相关参数解释如下：

Var: 必选，实数，高斯函数方差，按点数计算；

Amp: 可选，实数，高斯脉冲强度，即最大值，单位 mV；

Len: 可选，正整数，采样 L 个点，高斯函数的中心在 L/2 处；

Offset: 可选，实数，偏置，单位 mV，默认为 0 mV；

Loop：可选，正整数，循环次数，默认 1；

注意：Len*Loop 长度要在 88 点到 512M 个点之间；Amp+Offset 要在正负 500mv 之间，否则有溢出。

说明：按 Amp*exp(-t*t/(Var*Var))+Offset 计算，生成一个高斯脉冲序列，其中心位置在 Len/2 处。当 A 的值没有指定时取 GAmp 的值，Len 的值没有指定时默认采样范围是[0, 5*Var)。方差 Var 必须指定，后面参数可选。

示例：

```
w6 = Gauss(10); # 方差为 10，强度为 GAmp mV 的高斯脉冲
w7 = Gauss(10, 500, 100); # 方差为 10，强度为 500mV，100 个采样点的高斯脉冲
w8 = Gauss(10, Amp=500, Len=100); # 与 w7 相同
w9 = Gauss(10, Len=200); # 方差为 10，强度为 GAmp mV，200 个采样点的高斯脉冲
```

4、Square_p

函数原型为 Square_p(Len, Mode, Amp, Offset, Loop，Dutycycle)

其相关参数解释如下：

Len: 必选，正整数，方波周期的点数；

Mode: 可选，方波模式，UP 是先低后高，DOWN 是先高后低，默认值是 UP；

Amp: 可选，实数，振幅，单位 mV；

Offset: 可选，实数，偏置，单位 mV，默认值是 0 mV；

Loop: 可选，正整数，方波循环次数，默认 1。

Dutycycle: 可选，占空比，默认 0.5，范围 0-1。

注意：Len*Loop 长度要在 88 点到 512M 个点之间；Amp+Offset 要在正负 500mv 之间，否则有溢出。

示例：

```
w1 = Square_p(160, Mode=UP); #周期为 160 个点数，占空比为 0.5，先低后高的方波
```

5、Square

函数原型：Square(T, mode=UP, A=None, N=1，Dutycycle);

其相关参数解释如下：

T， 必选，正整数，方波周期，单位 ns；

mode, 可选，方波模式，UP 是先低后高，DOWN 是先高后低，默认值为 UP；

A， 可选，实数，同前,范围在 0-500mv，默认值为 GAmp；

N， 可选，正整数，方波循环次数，默认为 1；

Dutycycle: 可选，占空比，默认 0.5，范围 0-1；

注意：1.2*T*N 长度要在 88 点到 512M 个点之间；

```
w1 = Square (100, Mode=UP); #周期为 100ns,一共 120 个点数，占空比为 0.5，先低后高的方波
```

说明：采样率为 1.2GSa/s，方波周期 100ns，则采样的点数为 120 个，占空比为 0.5，所以脉宽为 60 个点数。

6、Triangle_p

函数原型为 Triangle_p(Len, Mode, Amp, Offset，Loop，Symmetry)

其相关参数解释如下：

Len: 必选，正整数，三角波周期的点数；

Mode: 可选，三角波模式，UP 是从低到高，DOWN 是从高到低，默认值为 UP；

Amp: 可选，振幅，单位 mV，默认值为 GAmp；

Offset: 可选，实数，偏置，单位 mV，默认值为 0 mV；

Loop: 可选，正整数，三角波循环次数，默认 1；

Symmetry：对称性，可选参数，实数，参数输入范围为：（0，1）；如果没有指定，则默认值为 0.5；Len*Symmetry 之后的值进行四舍五入取整。

注意：Len*Loop 长度要在 88 点到 512M 个点之间；Amp+Offset 要在正负 500mv 之间，否则有溢出。

示例：

```
w2 = Triangle_p(160, Mode=DOWN); #周期为 160 个点数，振幅为 GAmp mV，先高后低的三角波；
```

7、Triangle

函数原型：Triangle(T, mode=UP, A=None, N=1，Symmetry)；

其相关参数解释如下：

T， 必选，正整数，三角波周期，单位 ns；
mode，可选，三角波模式，UP 是从低到高，DOWN 是从高到低；
A， 可选，振幅，单位 mV，默认值为 GAmp；
N， 可选，正整数，三角波循环次数，默认值为 1；
Symmetry：对称性，可选参数，实数，参数输入范围为：（0，1）；如果没有指定，则默认值为 0.5；Len*Symmetry 之后的值进行四舍五入取整。

```
w4 = Triangle (100, Mode=DOWN); #周期为 100ns,120 个点数，振幅为 GAmp mV，先高后低的三角波；
```

8、Wind

窗函数，其函数原型为 Wind(N1,Val1,N2,……,Nn,Valn)，只参与与其他波形进行乘法运算（不能作为波形进行下载），相关参数解释如下：

Nn: 点数

Valn: 值 0 或 1

注意：不能作为基本波形，需要和别的波形绑定

示例：

```
win = Wind (100,1,200,0,300,1);
w1 = Sin(25, Len=600);
w2 = win*w1;
```

9、Shift

对波形进行移位操作，其函数原型为 Shift(Wave,Offset)，相关参数解释如下：

Wave: 波形

Offset：偏移的点数

示例：

```
w1 = Shift (w1,1000);
```

注意：不能作为基本波形，需要和别的波形绑定或者操作。

10、Delay

对波形进行延迟，其函数原型为 Delay(Cycle)，相关参数解释如下：

Cycle：延迟周期数（采样时钟的周期*4），采样频率为 1.2GHz

示例：

```
Delay (1000);
```

11、Combine

函数原型为 Combine(Wave1, Wave2,..., WaveN)，其功能是可以将多个波形拼接在一起，组成一个新的序列。

示例：

```
W4 = Combine(W1,W2,W3); #将 W1,W2,W3 拼接组成一个新的序列
```

12、Marker

函数原型为 Marker (N1,Val1,N2,Val2)，其相关参数解释如下：

N1,N2：点数（注意 N1 的点数要比 Marker 函数绑定的波形的总点数小 4 个以上）

VAL1,VAL2：值 0 或 1

通过 Marker 函数可以将 Marker 信号与相应的波形进行绑定。

示例：

```
W2_marker = Marker(100,1,200,0)
W2 = w2_wave + W2_marker; #将 Marker 信号与波形进行绑定
```

注意：要输出相关波形的 marker 信号，需要将相关波形通过 Marker 绑定，然后还要将相关通道的 Marker 开关打开。（N1+N2 长度最好和绑定波形一致，如果长度不一致的情况，会截取和波形长度一致）。

图 4.4.1 Marker 开关

基本波形是支持相加和相乘的运算，运算的时候，注意波形的长度（或者点数）需要一致。基本波形是不能够直接赋值给 OUT1-OUT4 的通道变量的。

13、Sweep

函数原型为 Sweep(Fstart,Fstop,Stime)，其相关参数解释如下：

Fstart：必选，正数，扫频开始频率，单位为 MHz；取值范围为：0-330MHz；
Fstop：必选，正数，扫频结束频率，单位为 MHz；取值范围为（0，330MHz]，且值必须大于 Fstart；
Stime：必选，正数加单位，扫频时长，单位可以是 ns、us、ms 和 s，取值范围为(74ns,445ms]。

示例：

```
W1 = Sweep(1,20,20ms); #从 1 到 20MHz 范围内进行扫频，扫频时长为 20ms
```

4.3.1.2 SQN 类型

一个 SQN 包含一个或多个“子序列”，其按时间先后排列，在外部触发模式下，子序列播放受触发信号控制。
SQN([w1(重复次数), …, wn(重复次数)])
其中 w1,…,wn 是 n 个 WAVE，n >= 1；
重复次数为正整数。

示例：

```
# 写出所有波形
s1 = SQN([w2(4), w3(2)]);                    # 包含 2 个子序列
s2 = SQN([w1(5), w2(2), w3(1)])             # 包含 3 个子序列
s3 = SQN([[w1(5), w2(2)], w3(1)])            # 包含 2 个子序列
s4 = SQN([[w1(5),Delay(10), w2(2)], w3(1)])  # 包含 2 个子序列，第 1 个子序列又包含 2 个子序列
```

4.3.1.4 ADVS 类型

通过调用一个 SQN 类型可生成一个 ADVS，ADVS 支持“+”和“+=”运算。SQN(循环次数)
多个 ADVS 可以拼接为一个 ADVS。

示例：

```
a1  = ADVS();
a1 += s1(3) + s2(2);#s1 和 S2 属于 SQN 类型变量
a2 = ADVS(s1(3) + s2(2)); # 与 a1 等效
a3 = s1(3) + s2(2);          # 自动处理类型，与 a1 等效
```

4.3.1.3 OUT 类型

OUT 类型主要有 OUT1、OUT2、OUT3 和 OUT4 四种，分别对应四个通道的输出。

4.4.2 编辑和下载

4.4.2.1 编辑

用户进行波形编辑可以分为以下三个步骤：
A. 使用函数库编辑 WAVE 集合
B. 使用 SQN 定义序列的播放顺序
C. 将 ADVS 和 SQN 定义的序列赋值给相应的通道

示例：

```
####使用函数库编辑 WAVE 集合
GAmp = 200;
WD = 'E:\\'
w0 = WAVE('data.wave')
f = 10;
w1 = Sin(10);
w2_wave = Gauss(10, Len=88);
w3 = Sin(f);
w2_maker = Marker(10,0,20,1);
w2 = w2_wave + w2_maker;
```

```
####使用 SQN 定义序列
s1 = SQN([w0(4),Delay(100), w1(2), w2(2)])
s2 = SQN([w3(4)]);
s3 = s1（1） + s2（1）;
s4 = SQN([[w2(6), w1(2)],[w3(2), w3(2)]]);
####将 SQN 定义的序列赋值给相应的通道
OUT1 = s3;
```

4.4.2.2 解析及下载

软件对用户编辑的语句进行编译的过程为：首先解析出用户定义的 WAVE 集合{Wave1，Wave2，…，WaveN}，然后根据 ADVS 和 SQN 语句，解析出子序列集合{Seq1，Seq2，…，SeqN}和每个序列波形的组成，最后根据 OUT 赋值，将相应的波形集合和序列播放方式下载到硬件进行执行。

1、波形集合解析

使用函数产生波形时，波形的长度如果不满足 8 的倍数，则解析时自动补零使其长度满足 8 的倍数。当两个波形进行运算时，波形短的自动补零，使参与运算的波形长度一样。

Marker 函数是用来产生 Marker 标记信号的，其与波形集合中的波形一一对应，有多少个波形，就有多少个 Marker 信号。使用过程中会出现有些波形定义了 Marker 信号，有些没有定义，这时候波形没有定义的 Marker 信号默认输出 1。

软件解析出用户编辑的所有基本波形后，通过下表定义的格式下发给硬件进行存储，每个波形的开始地址和结束地址，软件建立表进行存储，以方便后续处理调用。

2、ADVS 和 SQN 序列集合解析

使用 ADVS 和 SQN 类型时，需要软件解析出序列的个数、序列的组成结构和序列播放方式，如图所示。

波形示例解析

```
####使用函数库编辑波形集合
GAmp = 200;
freq = 10;
w1 = Sin(freq,Len=40);
w2_wave = Gauss(10, Len=90);
w2_maker = Marker(45,1,45,0);
w2 = w2_wave + w2_maker;
w3_wave = Sin(2*freq,Len=200);
w3_maker = Marker(100,0,100,1);
w3 = w3_wave + w3_maker;
w4 = Square_p(96,Mode=UP);
####使用 SQN 定义序列
####s1 = SQN([[w1,w2],[w3,Delay(100),w2],[w4,w3,w1(3)],w3]);
####注意：以上区域错误，每种波形后面需要加上相应的次数如 w2(1)而不能
####写成为 w2
s1=SQN([[w1(1),w2(1)],[w3(1),Delay(100),w2(1)],[w4(1),w3(1),w1(3)],w3
(1)]);
####将 SQN 定义的序列赋值给相应的通道
OUT1 = s1;
```

编辑的波形时序图如下图所示。

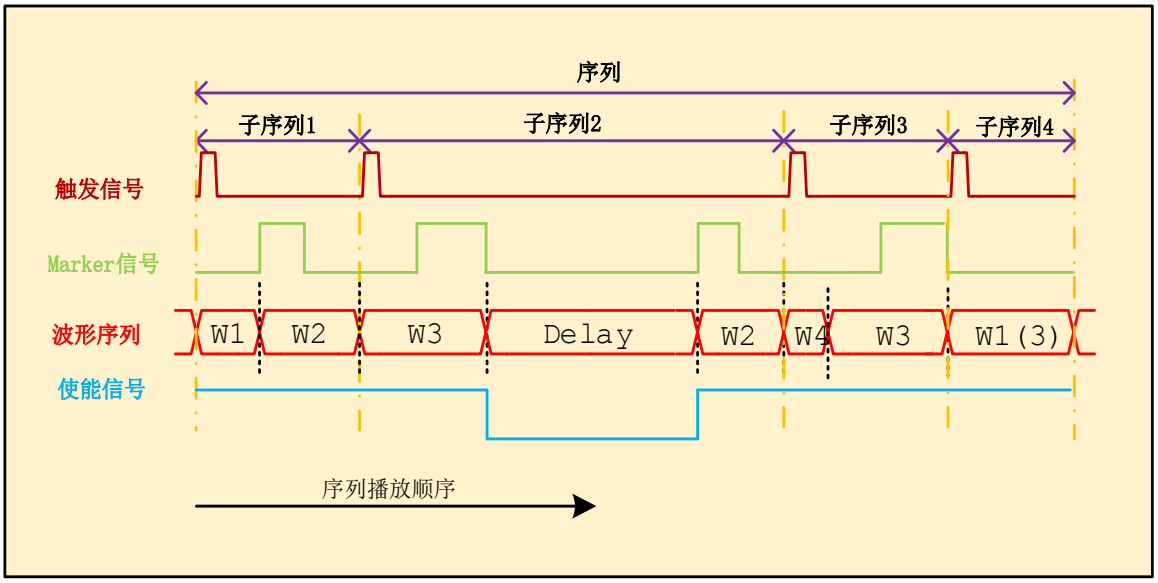


图 4.4.1 编辑波形的时序图

注意：单个基本波形长度必须大于等于 88 个点，如果单个波形的点数小于 88 个点的情况下，会出现序列出错。

5、接口程序调用

5.1 调用说明

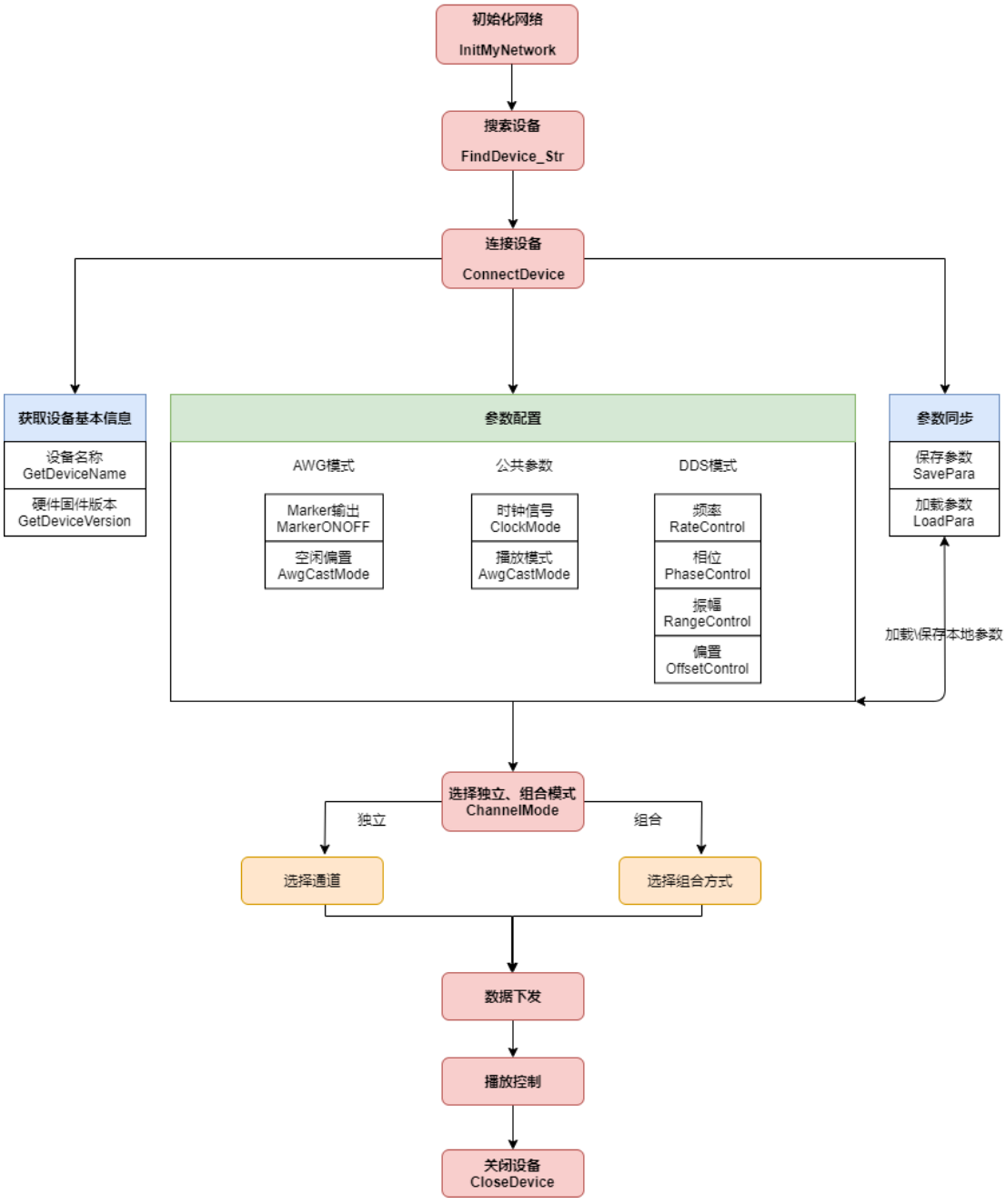


图 5.1.1 API 调用说明

5.2 C++ API 接口程序及调用注意事项

5.2.1 C++ API 接口

1、搜索设备

功能：初始化本机网络
接口函数：int InitMyNetwork(char * localIp);
参数:

localIp	本地 IP

返回值：返回 1 表示成功。

2、搜索设备

功能：搜索当前局域网内所接通的设备
接口函数：int FindDevice_Str(char *RsvIp, char *RsvMac, char *RsvName);
参数:

RsvName	设备名称
RsvIp	IP 地址
RsvMac	Mac 地址

返回值：返回 int 整数表示设备个数。

3、连接设备

功能：连接设备
接口函数：
int ConnectDevice(char *pDestinationIP, char *pDestinationMAC);
参数：

pDestinationIP	设备 IP
pDestinationMAC	设备 MAC

返回值：返回 1 表示成功。

4、关闭设备

功能：断开连接
接口函数：
bool CloseDevice();
返回值：true 表示函数执行成功，false 表示函数执行失败。

5、下载波形

功能：下载波形数据
接口函数：
int LoadWaveData(int channelNumber, const char* code);
参数：

channelNumber	通道号
code	波形代码

返回值：
1 表示成功；
0 表示 AWG SEQ 使能地址加载失败；
-1 表示 AWG 使能信号数据加载失败；
-2 表示 AWG SEQ Marker 地址加载失败；
-3 表示 AWG Marker 信号地址加载失败；
-4 表示 AWG Marker 信号数据加载失败；
-5 表示 AWG SEQ 波形地址加载失败；
-6 表示 AWG 波形地址加载失败；
-7 表示 AWG 波形数据加载失败。

6、播放控制

功能：控制播放开关
接口函数：
int AwgBroadcast(int channelNumber, int mode, char * recvbuf);
参数：

channelNumber	1-4 表示 AWG，5 表示组合模式开关
mode	1 表示播放，0 表示停止
recvbuf	返回信息

返回值：
非 0 执行成功。

7、播放次数

功能：设置播放次数
接口函数：

int AwgCastNumber(int mode, char * recvbuf);
参数：

mode	次数
recvbuf	返回信息

返回值：
非 0 执行成功。

8、 播放模式

功能：设置播放模式
接口函数：
int AwgCastMode(int mode, char * recvbuf);
参数：

mode	1 表示 trigger 模式播放 0 表示连续模式
recvbuf	返回信息

返回值：
非 0 执行成功。

9、AWG 通道重置

功能：设置通道偏置
接口函数：
int AwgOffset(int channelNumber, char * Offset, char * recvbuf);
参数：

channelNumber	通道号 1-4
Offset	偏置
recvbuf	返回信息

返回值：
非 0 执行成功。

10、时钟模式

功能：设置时钟模式
接口函数：
int ClockMode(int mode, char * recvbuf);
参数：

mode	0 代表内部时钟 1 代表 10MHZ 2 代表 100MHZ
recvbuf	返回信息

返回值：
非 0 执行成功。

11、独立组合模式

功能：选择独立或者组合模式
接口函数：
int ChannelMode(int mode, char * recvbuf);
参数：

mode	0 代表独立　1 代表组合
recvbuf	返回信息

返回值：
非 0 执行成功。

12、组合模式通道开关

功能：组合模式下通道开关控制
接口函数：
int ChannelONOFF(int channelNumber, int mode, char * recvbuf);
参数：

channelNumber	1-4 表示 AWG，　5-8 表示 DDS
mode	0 表示关闭 1 表示打开
recvbuf	返回信息

返回值：
非 0 执行成功。

13、 Marker 开关

功能：设置 Marker 模式
接口函数：
int MarkerONOFF(int channelNumber, int mode, char * recvbuf);
参数：

channelNumber	通道号 1-4
mode	0 表示关闭 1 表示打开
recvbuf	返回信息

返回值：
非 0 执行成功。

14、DDS 播放开关

功能：独立模式 DDS 播放控制
接口函数：
int DDSCast(int channelNumber, int mode, char * recvbuf);
参数：

channelNumber	通道号 1-4

mode	0 代表关闭　1 代表打开
recvbuf	返回信息

返回值：
非 0 执行成功。

15、 频率控制字

功能：设置频率
接口函数：
int RateControl(int channelNumber, char * frequencyChannel, char * recvbuf);
参数：

channelNumber	通道号 1-4
frequencyChannel	频率
recvbuf	返回信息

返回值：
非 0 执行成功。

16 、相位控制字

功能：设置相位
接口函数：
int PhaseControl(int channelNumber, char * phaseChannel, char * recvbuf);
参数：

channelNumber	通道号 1-4
phaseChannel	相位
recvbuf	返回信息

返回值：
非 0 执行成功。

17、 幅度控制字

功能：设置幅度
接口函数：
int RangeControl(int channelNumber, char * amplitudeValue, char * recvbuf);
参数：

channelNumber	通道号 1-4
frequencyChannel	频率
recvbuf	返回信息

返回值：

非 0 执行成功。

18、偏置控制字

功能：设置偏置
接口函数：
int OffsetControl(int channelNumber, char * offsetChannel, char * recvbuf);
参数：

channelNumber	通道号 1-4
offsetChannel	偏置
recvbuf	返回信息

返回值：
非 0 执行成功。

19 、参数同步

功能：同步设备参数
接口函数：
int LoadPara(char * filename, char * recvbuf);
参数：

filename	文件名
recvbuf	返回信息

返回值：
1 执行成功。

20、 保存参数

功能：保存参数到本地
接口函数：
int SavePara(char * filename, char * recvbuf);
参数：

filename	文件名
recvbuf	返回信息

返回值：
1 执行成功。

21、 下载数据

功能：下载数据
接口函数：

int LoadData(int channelNumber, const char* code, char * s, char * err);

参数：

channelNumber	通道号
code	波形代码
s	基础波形数据 示例：SQN([w1(1),w2(1)])
err	错误信息

返回值：

1 表示成功；

0 表示获取下载状态失败；

-7 表示下载 AWG 波形数据失败。

22、 下载地址

功能：下载地址

接口函数：

int LoadAddr(int channelNumber, const char* code, char *s,char *err);

参数：

channelNumber	通道号
code	波形代码
s	基础波形数据 示例：SQN([w1(1),w2(1)])
err	错误信息

返回值：

1 表示成功；

0 表示 AWG SEQ 使能地址加载失败；

-1 表示 AWG 使能信号数据加载失败；

-2 表示 AWG SEQ Marker 地址加载失败；

-3 表示 AWG Marker 信号地址加载失败；

-4 表示 AWG Marker 信号数据加载失败；

-5 表示 AWG SEQ 波形地址加载失败；

-6 表示 AWG 波形地址加载失败。

23、 获取通道播放状态

功能：获取通道播放状态

接口函数：

int PlayStatus(int channelNumber, char * playMode, char * expermentMode);

参数：

channelNumber	通道号
playMode	播放模式　0-3 0：连续模式 1：上升沿触发模式 2：下降沿触发模式 3：电平触发模式
expermentMode	实验模式 DDS/AWG

返回值：

1 表示正在播放。

24、 获取参数

功能：获取参数，采用 | 分割

接口函数：

```
int GetParas(char * pData);
```

参数：

序号	pData
0	播放模式
1	时钟信号
2	循环次数
3	AWG 通道 1Marker 开关
4	AWG 通道 2Marker 开关
5	AWG 通道 3Marker 开关
6	AWG 通道 4Marker 开关
7	AWG 通道 1 偏置
8	AWG 通道 2 偏置
9	AWG 通道 3 偏置
10	AWG 通道 4 偏置
11	DDS 通道 1 频率
12	DDS 通道 2 频率
13	DDS 通道 3 频率
14	DDS 通道 4 频率
15	DDS 通道 1 相位
16	DDS 通道 2 相位
17	DDS 通道 3 相位
18	DDS 通道 4 相位
19	DDS 通道 1 振幅
20	DDS 通道 2 振幅
21	DDS 通道 3 振幅
22	DDS 通道 4 振幅

23	DDS 通道 1 偏置
24	DDS 通道 2 偏置
25	DDS 通道 3 偏置
26	DDS 通道 4 偏置

返回值：

1 表示成功。

25、错误码

代码	含义	备注
1	操作正常	操作成功
0	Bool 类型表示失败	检查调用是否合法
996	读取寄存器失败	检查连接是否正常设备是否正常工作
997	写寄存器失败	检查连接是否正常设备是否正常工作

5.2.2 C++ API 调用注意事项

1. C++调用版本要求：建议 VS2008 及以上版本。
2. 32 位与 64 位动态库应与所建程序保持对应。

5.2.3 C++ API 接口调用示例

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include "AWGDLL.h"
using namespace std;

#define FAIL        0
#define RECV_SIZE   64
int main()
{    char recvbuf[RECV_SIZE];
     int rtn = FAIL;

     /****************** STEP1 连接设备
*********************/
     char localip[16] = "192.168.1.100";
     int port = 32000;
     rtn = InitMyNetwork(localip);
     char devIP[256];
     char devMAC[256];
     char devNAME[256];
     /********************* STEP3 下载波形
*********************/
     //定义波形代码
     const char* code = " w1 = Sin(20,
Len=300, Phase=0.25)\
s1 = SEQ([w1(1)])\
OUT1 = s1";\

     rtn = LoadWaveData(1, code);//通道一
     if (0 == rtn){
          printf("run failed ");//函数执行
失败
          return FAIL;
     }

     /********************* STEP4 播放控制
*********************/
     rtn = AwgBroadcast(1, 1, recvbuf);
//播放通道一
     if (rtn){
          //do
     }
```

```
    int devnum = FindDevices_Str(devIP, devMAC, devNAME);
    if (0 == devnum){
        return FAIL;
    }
    rtn = ConnectDevice(devIP, devMAC);

    if (1 != rtn){
        printf("Connection failed ");
        return FAIL;
    }

    /******************** STEP2 参数配置 ********************/
    rtn = ChannelMode(0, recvbuf);
//选择独立模式
    rtn = AwgCastMode(0, recvbuf);
//播放模式，连续
    rtn = AwgOffset(1, "50", recvbuf);
//通道 1，AWG 空闲偏置
    rtn = MarkerONOFF(1, 1, recvbuf);
//通道 1，Marker 打开
    rtn = ClockMode(0, recvbuf);
//内部时钟
    rtn = AwgBroadcast(1, 0, recvbuf);
//停止播放

    /******************** STEP5 关闭设备 ********************/
    result = CloseDevice();
    if (0 == result){
        return FAIL;
    }

    free(devIP);
    free(devMAC);
    free(devMAC);

    system("pause");
    return rtn;
}
```

5.3 Python 接口程序及调用注意事项

5.3.1 Python 接口程序

1. init_network

功能：功能初始化网络。C API 中 InitMyNetWork 的封装
函数：init_network(localIp)
参数：

localIp	本机 IP

返回值：0 表示失败，1 表示成功。

2. find_device

功能：搜索设备。C API 中 FindDevice_Str 的封装
函数：find_device()
返回值：IP 地址、mac 地址和设备名元组的列表

3. connect

功能：连接设备。C API 中 ConnectDevice 的封装
函数：connect(ip, mac)
参数：

ip	目标设备 ip 字符串
mac	目标设备 mac 字符串

返回值：1 表示成功。

4. close_device

功能：断开设备。C API 中 CloseDevice 的封装
函数：close_device()
返回值：True 表示函数执行成功，False 表示函数执行失败。

5. system_init

功能：系统复位，重置所有设置。C API 中 SystemInit 的封装
函数：system_init()
返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息。

6. load_wave_data

功能：按照通道号下载数据。C API 中 LoadWaveData 的封装
函数：load_wave_data(channel, code)
参数：

channel	道号
code	波形代码

返回值：
1 表示成功；
0 表示 AWG SEQ 使能地址加载失败；
-1 表示 AWG 使能信号数据加载失败；
-2 表示 AWG SEQ Marker 地址加载失败；
-3 表示 AWG Marker 信号地址加载失败；
-4 表示 AWG Marker 信号数据加载失败；
-5 表示 AWG SEQ 波形地址加载失败；
-6 表示 AWG 波形地址加载失败；
-7 表示 AWG 波形数据加载失败。

7. awg_broadcast

功能：按照通道号播放波形。C API 中 AwgBroadcast 的封装
函数：awg_broadcast(channel, mode)
参数：

channel	通道号，1-4 表示 AWG，5-8 表示 DDS；

mode	1 表示播放，0 表示停止

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

8. awg_cast_number

功能：AWG 播放次数。C API 中 AwgCastNumber 的封装
函数：awg_cast_number(repeat)
参数：

repeat	播放次数

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

9. awg_cast_mode

功能：AWG 播放模式。C API 中 AwgCastMode 的封装
函数：awg_cast_mode(mode)
参数：

mode	播放次数

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

10. awg_offset

功能：AWG 通道偏置。C API 中 AwgOffset 的封装
函数：awg_offset(channel, offset)
参数：

channel	通道号，1-4；
offset	偏置

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

11. clock_mode

功能：时钟模式。C API 中 ClockMode 的封装
函数：clock_mode(mode)
参数：

mode	0 代表内部时钟，1 代表 10MHZ，2 代表 100MHZ；

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

12. channel_mode

功能：独立组合模式。C API 中 ChannelMode 的封装
函数：channel_mode(mode)

参数：

mode	0 代表独立,1 代表组合；

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

13. channel_switch

功能：组合模式通道开关。C API 中 ChannelONOFF 的封装
函数：channel_switch(channel, mode)
参数：

channel	1-4 表示 AWG，5-8 表示 DDS；
mode	0 表示关闭，1 表示打开

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

14. marker_switch

功能：Marker 开关。C API 中 MarkerONOFF 的封装
函数：marker_switch(channel, mode)
参数：

channel	通道号 1-4；
mode	0 表示关闭，1 表示打开

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

15. DDS_cast

功能：DDS 播放。C API 中 DDSCast 的封装
函数：DDS_cast(channel, mode)
参数：

channel	通道号 1-4；
mode	0 表示独立，1 表示组合

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

16. rate_control

功能：频率控制字。C API 中 RateControl 的封装
函数：rate_control(channel, freq)
参数：

channel	通道号 1-4；
freq	频率字符串值

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

17. phase_control

功能：相位控制字。C API 中 PhaseControl 的封装
函数：phase_control(channel, phase)
参数：

channel	通道号 1-4；
phase	相位字符串值

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

18. range_control

功能：幅度控制字。C API 中 RangeControl 的封装
函数：range_control(channel, ampl)
参数：

channel	通道号 1-4；
ampl	频率字符串值

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

19. offset_control

功能：偏置控制字。C API 中 OffsetControl 的封装
函数：offset_control(channel, offset)
参数：

channel	通道号 1-4；
offset	偏置字符串值

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

20. load_params

功能：加载参数。C API 中 LoadPara 的封装
函数：load_params(filename)
参数：

filename	文件名

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

21. save_params

功能：保存参数。C API 中 SavePara 的封装
函数：save_params(filename)
参数：

filename	文件名

返回值：(res, msg)，res=1 表示执行成功，msg 是返回信息

22. load_wave

功能：下载波形数据
函数：**load_wave** (channel, code, s, err)
参数：

channelNumber	通道号
code	波形代码
s	基础波形数据 示例：SQN([w1(1),w2(1)])
err	错误信息

返回值：1 表示成功。

23. load_wave_addr

功能：下载波形地址
函数：**load_wave_addr**(channel, code, s, err)

参数：

channelNumber	通道号
code	波形代码
s	基础波形数据 示例：SQN([w1(1),w2(1)])
err	错误信息

返回值：1 表示成功。

24. awg_PlayStatus

功能：获取状态
函数：awg_PlayStatus(channel)
参数：

channel	通道号

返回值：(res, msg1, msg2)，res=1 表示正在播放，msg1 是播放模式，msg2 是实验模式。

25. awg_GetParas

功能：获取参数，采用|分割
函数：awg_GetParas()

参数：

序号	返回

0	播放模式
1	时钟信号
2	循环次数
3	AWG 通道 1Marker 开关
4	AWG 通道 2Marker 开关
5	AWG 通道 3Marker 开关
6	AWG 通道 4Marker 开关
7	AWG 通道 1 偏置
8	AWG 通道 2 偏置
9	AWG 通道 3 偏置
10	AWG 通道 4 偏置
11	DDS 通道 1 频率
12	DDS 通道 2 频率
13	DDS 通道 3 频率
14	DDS 通道 4 频率
15	DDS 通道 1 相位
16	DDS 通道 2 相位
17	DDS 通道 3 相位
18	DDS 通道 4 相位
19	DDS 通道 1 振幅
20	DDS 通道 2 振幅
21	DDS 通道 3 振幅
22	DDS 通道 4 振幅
23	DDS 通道 1 偏置
24	DDS 通道 2 偏置
25	DDS 通道 3 偏置
26	DDS 通道 4 偏置

返回值：1 表示成功。

5.3.2 Python 接口示例

```
import sys
import platform
from awg4100 import AwgDevice
import time
import struct
local_ip = "172.16.4.221"  # 本机 IP
out_ch = 1
```

```
# 定义波形代码
wave_code = """
w1 = Sin(10, A=10, L=160)
w2 = Sin(20, A=50, L=160, ph0=0.25)
w3 = 2 * w1
w4 = w1 * w2
s1 = SQN([w3(1)])
s2 = SQN([w4(1)])
OUT1 = s1
OUT2 = s2
"""
dev = AwgDevice()
result = dev.init_network(local_ip)
if result == 0:
    print("Init network failed.")
    sys.exit()
dev_info = dev.find_device()
dev_num = len(dev_info)
if dev_num == 0:
    print("Cannot found device")
    sys.exit()
for idx in range(dev_num):
    print("[{}] IP={}, MAC={}, Name={}".format(idx, \
        dev_info[idx][0], dev_info[idx][1], dev_info[idx][2]))
trgt = int(input("choice: "))

ip = dev_info[trgt][0]
mac = dev_info[trgt][1]
# 1. 连接设备
result = dev.connect(ip, mac)
if result != 1:
    print("Connect failed.")
    sys.exit()
```

```
def check_ret(rtn, msg=None):
    if rtn == 0:
        print(msg)
        sys.exit()
rtn, msg = dev.system_init()
check_ret(rtn, "System Reset failed.")
# 2. 参数配置
rtn, msg = dev.channel_mode(0)       # 选择独立模式
check_ret(rtn, "set mode failed: {}".format(msg))

rtn, msg = dev.awg_cast_mode(0)      # 播放模式，连续
check_ret(rtn, "set awg cast mode failed: {}".format(msg))

rtn, msg = dev.awg_offset(out_ch, "10")  # 通道 1，AWG 空闲偏置
check_ret(rtn, "set offset failed: {}".format(msg))

rtn, msg = dev.marker_switch(out_ch, 1)  # 通道 1，Marker 打开
check_ret(rtn, "set marker failed: {}".format(msg))

rtn, msg = dev.clock_mode(0)         # 内部时钟
check_ret(rtn, "set clock failed: {}".format(msg))

# 3. 下载波形
result = dev.load_wave_data(out_ch, wave_code) # 通道 1
if result == 0:
    print("wave download failed: {}".format(result))
    sys.exit()

# 4. 设置播放次数
rtn, info = dev.awg_cast_number(0)
check_ret(rtn, "set awg cast number failed: {}".format(info))

# 5. 播放控制
rtn, info = dev.awg_broadcast(out_ch, 1)    # 播放通道 1
```

```
check_ret(rtn, "start failed: {}".format(info))
input("enter any to stop")

# 6. 停止播放
rtn, info = dev.awg_broadcast(out_ch, 0)
check_ret(rtn, "stop failed: {}".format(info))

# 7. 关闭设备
result = dev.close_device()
if not result:
    sys.exit()
```

5.3.3 Python 接口调用注意事项

1、使用说明

方法一：脚本运行环境为 Python3(建议版本 > 3.6) 。使用时需要与模块 `awg4100` 在同一目录下，如下所示，目录`awg4100`中的内容缺一不可。**使用前请确保 AWG4100 能够正常运行**。

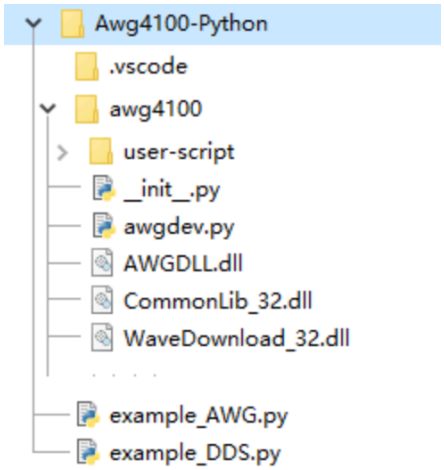


图 5.4.1 工程目录

方法二:将 awg4100 模块直接放到 Python 的安装目录下的 /Lib/site-packages 目录下面。

2、异常处理

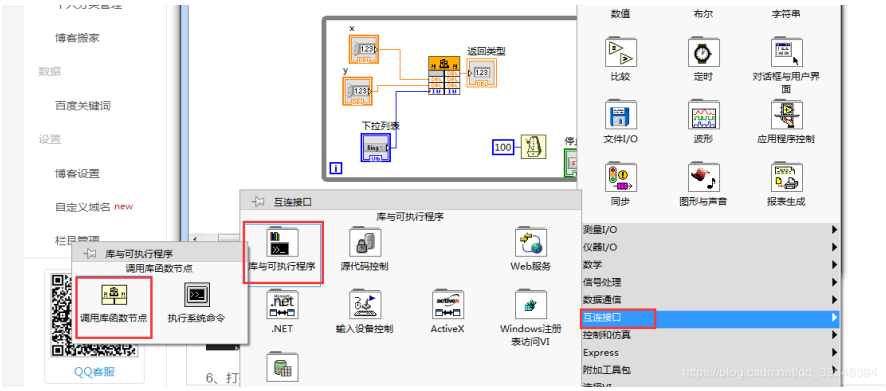
若出现 OSError:[WinError126] ，可能是由于缺失 VC 运行时造成的，请安装 msvc2013 运行时环境（https://www.microsoft.com/zhcn/download/details.aspx?id=40784）和 msvc2015 运行时环境（https://www.microsoft.com/en-us/download/details.aspx?id=48145），建议 x86 版本与 x64 版本都要安装。

如果安装之后依旧报错。请将 msvcp120.dll、msvcr120.dll、msvcp140.dll vcruntime140.dll 放到 `awg4100` 目录下。请注意这几个 dll 是 32 位的还是 64 位的，如果使用 32 位 Python 则复制 32 位的，如果使用 64 位 Python 则复制 64 位的。

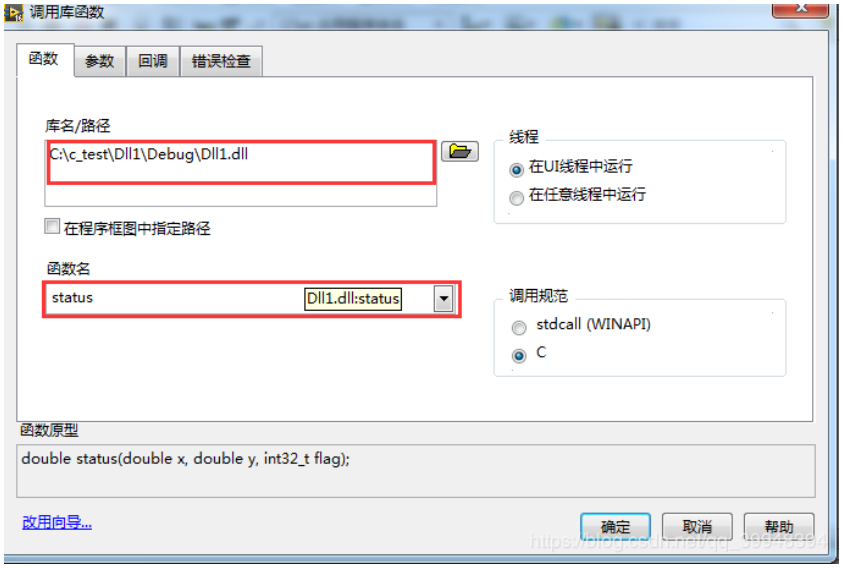
5.4 LabVIEW 接口调用及注意事项

5.4.1 LabVIEW 接口调用流程参考

打开【程序框图】窗口
在空白处鼠标右键，在出现的【函数选板】中选择【互连接口】-【库与可执行程序】-【调用库函数节点】

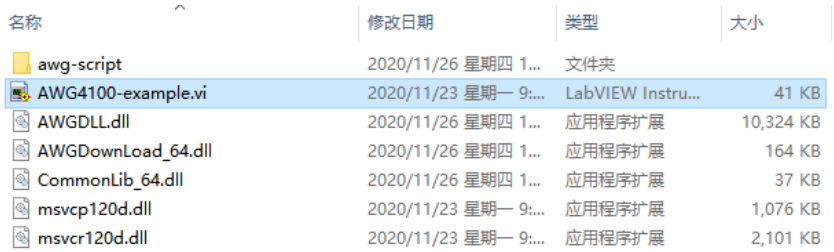


双击函数，找到 dll 文件，选择函数。



参数里面添加返回值和参数，注意顺序和类型

注意：dll 的目录中，需要包含如下图所示文件（其中 AWG4100-example.vi 为 LabVIEW 文件，其他文件为 AWGDLL 的依赖库，必须包含同一目录下）



名称	修改日期	类型	大小
awg-script	2020/11/26 星期四 1...	文件夹	
AWG4100-example.vi	2020/11/23 星期一 9:...	LabVIEW Instru...	41 KB
AWGDLL.dll	2020/11/26 星期四 1...	应用程序扩展	10,324 KB
AWGDownLoad_64.dll	2020/11/26 星期四 1...	应用程序扩展	164 KB
CommonLib_64.dll	2020/11/26 星期四 1...	应用程序扩展	37 KB
msvcp120d.dll	2020/11/23 星期一 9:...	应用程序扩展	1,076 KB
msvcr120d.dll	2020/11/23 星期一 9:...	应用程序扩展	2,101 KB

相关的二次开发库，可以到官网到我司的官网 https://www.ciqtek.com 产品中心→量子测控系列产品→任意波形发生器（AWG4100）产品介绍界面下方下载。

6、常见问题

6.1 计算机无法 ping 通设备

在支持千兆网卡的计算机上连接仪器，ping 命令没有响应，可进行下列尝试。

1. 检查计算机是否是千兆网卡。
2. 检查网线是否支持千兆网。
3. 检查交换机是否为千兆交换机。
4. 检查电脑 IP 地址是否与板卡是同网段(ping 协议需要同网段，但是连接不需要)。
5. 重启设备。

6.2 计算机可以识别设备但软件中设备连接失败

计算机的设备管理器显示正确识别设备但是软件中设备连接失败，则重启软件进行尝试，如果还是显示无法连接，重启设备。

6.3 设置正确但无信号输出

1. 检查 SMA 线缆是否接紧。
2. 检查波形及其设置是否正确：模块开启的按钮是否正确；开启的模块，是否波形都已进行下载。
3. 检查示波器等设置是否正确：示波器的触发模式是否设置正确（特别是对于 sequence 播放模式，如果设置的等待时间过长，使用示波器的自动(Auto)模式一般无法看到波形，需要切换到正常（Normal）模式或者单次触发（Trigger 模式）才可以看到波形。
4. 断开连接，看是否能够重新连接设备，如果发现连接报错，检查网络的状态，看是否网络连接已经断开。
5. 重启设备。

若以上操作完成后仍无法解决问题，请及时与我司联系。

6.4 连接正常，设置波形后有输出，但是输出异常

1. 检查 SMA 线缆是否接紧。
2. 检查波形及其设置是否正确。

3. 检查示波器设置是否正确：示波器采集时，需要设置直流 50 欧姆的阻抗匹配（Coupling: DC 50Ω）。
4. 断开连接，看是否能够重新连接设备，如果发现连接报错，检查网口的状态，看是否网络连接已经断开。
5. 断开连接后重连成功，但是波形仍然不对，可尝试点击设置中的‘切换’按钮（可以不改变时钟输出源，直接点击切换），然后再进行测试。
6. 重启设备。

若以上操作完成后仍无法解决问题，请及时与我司联系。

6.5 编写的波形函数输出的波形不完整，比如正弦波每个重复内最后的正弦波都不完整

这是因为波形长度必须是 8 的整数倍，如何保证输出的波形完整，详细计算方法如下：

计算公式：

Len*1/1.2G = 1000/f *N（f 在这里是 MHz）

Len=1000/f*N*1.2G

其中 Len：输出的波形长度；

f：设置的波形频率(单位 MHz)

N：正整数；

限制条件：

Len*Loop >=88

Len*Loop 必须位 8 的正整数倍。（Loop 为循环次数）

示例：

示例 1：M=10MHz，Loop=1

则：Len=1000/10*N*1.2=120*N

Len*Loop=120*N，则可以同时满足两个限制条件，则可以设置 Len 为 120、240……，都是可以的。

示例 2：M=200MHz，Loop=1

则：Len=1000/200*N*1.2=6*N

Len*Loop=6*N，则为了满足两个限制条件，则 N 可以是 16、20……，即 Len=96、120 等。

7、注意事项

1. 任意波形发生器的外界工作环境温度必须在-10 ℃ 至 +50 ℃ 以内。
2. 搬运仪器时请轻拿轻放，切勿将重物放置在机箱上。
3. 如果您怀疑本产品出现故障，请立即联络我司授权的维修人员进行检测。任何由未经我司允许的维护、调整或零件更换而造成损失，我司概不承担任何责任。
4. 如果第一次运行时提示缺少 xx.dll，由于系统缺少该 dll，可以自行下载放到安装目录或者联系我们提供。
5. 运行环境 win7 及以上版本，该程序是 32 位程序，在 32 位及 64 位电脑均可使用。
6. 直连环境下建议电脑手动配置固定 IP，子网掩码 255.255.255.0。

国仪量子公众号

国仪量子售后服务小程序

用量子技术感知世界

FEEL THE WORLD IN A QUANTUM WAY

国仪量子(合肥)技术有限公司
地址：合肥市高新区创新产业园二期E2楼

无锡量子感知研究所
地址：无锡市惠山区惠山站区站前路2号

国仪量子(上海)测量技术有限公司
地址：上海市虹口区北外滩峨眉路315号8405室

4000606976-602

www.ciqtek.com

service@ciqtek.com