

LAI320-PCI 任意波形 发生卡

使用说明书

第一章 概 述

1.1 简介

LAI320 任意波形发生卡，有 2 路波形输出。为我公司 LAI220 产品的升级替代产品，用户可用鼠标编辑所需的任意波形，也可选择正弦波、方波、三角波、锯齿、TTL、白噪声、高斯噪声、梯形、指数、AM、FM、扫频等常规波形，可设置波形的幅度、频率、偏置量等参数。LAI320 采用 DDS 合成技术、全表贴工艺、大规模 FPGA 技术，具有频率精度高、分辨率高、可靠性好、软件支持丰富等优点，可广泛用于个人实验室和自动测试系统。

1.2 技术性能指标

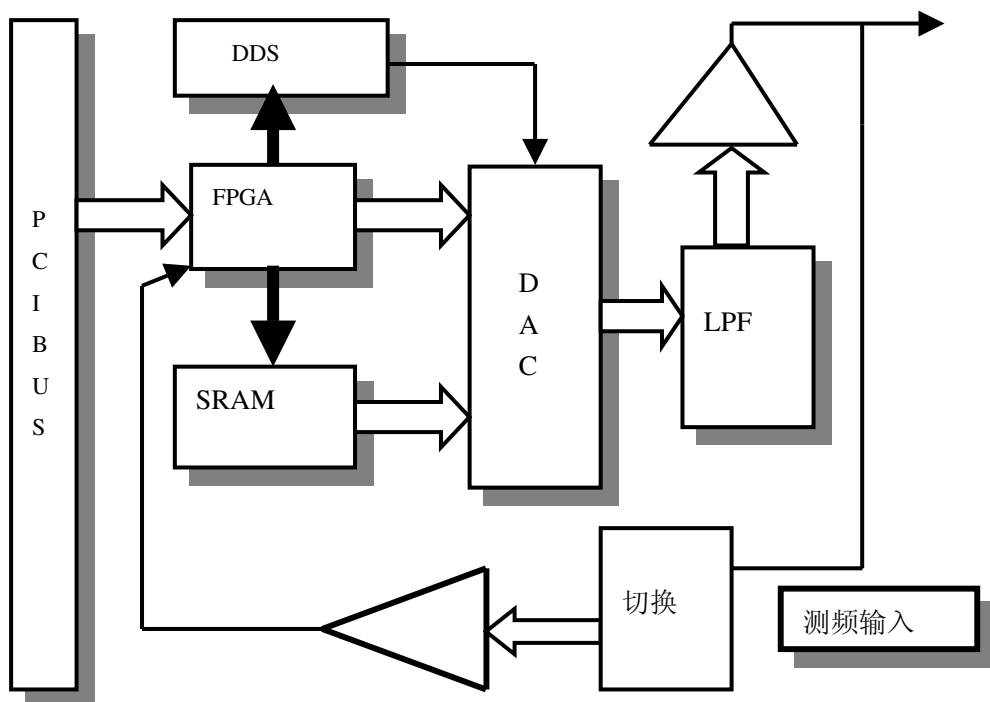
T1 波形输出通道:

波形频率	0.1Hz(DC) ~5MHz
频率分辨率	0.02Hz
DAC 时钟	0~80MHz 连续可调，步进为 0.2Hz
通道数	并行 2CH、可同步
波形长度	标配 256Ksa/CH,可扩到 512Ksa
垂直分辨率	14 位
频率稳定度	<10ppm
波形幅度	0~±10V、0~±1V
波形个数	1~4095 个、循环输出
输出阻抗	100Ω,输出电流 50mA V _{peak} =100mA
低通滤波	5MHz、1MHz、100KHz、10KHz、1KHz 程控
直流精度	±0.1% (FS)
交流精度	±0.2%
谐波失真	LAI320 :-65dBc(<1KHz)、 -58dBc(100KHz)
软件平台	Windows2000/XP
二次开发	提供常用开发平台下的 DEMO 源代码(VC、CVI、VB、LABVIEW、C++ Builder)。

1.3 软件支持

LAI320 任意波形发生卡提供丰富的软件支持，提供集成软件和驱动程序以及编程接口、动态连接库、使用例程等。

1.4 工作原理



图一、LAI320 原理框图

工作原理：

PC 机将波形数据通过 PCI 总线送到存储器中，存储器循环地将波形数据发送到 DAC 电路，由 DDS（数字频率合成器）电路产生相应的 DAC 刷新时钟（0~50MHz, 步进 0.04Hz）。DAC 输出波形经缓冲放大、低通滤波、放大输出。

第二章 硬件安装与设置

2.1 最低配置:

Pentium 及其兼容机, 128M 内存、1024x768 监视器。

2.2 LAI320 板卡外型



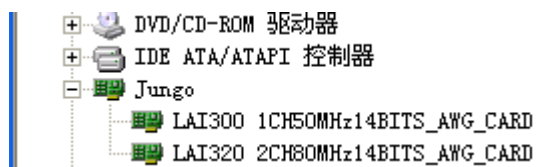
图二、LAI320 板卡外型

2.3、安装 LAI320-PCI

1) 安装驱动程序

插入 PCI 卡, 启动系统, 计算机提示找到新的“PCI Card”, 在界面的提示下将驱动程序指向光盘的: Driver\LAI320.inf

装完后会在“控制面板”的“系统”下看到“GoodInst”下有“LAI320-2CH80MHz_14BITS_AWG_CARD”。



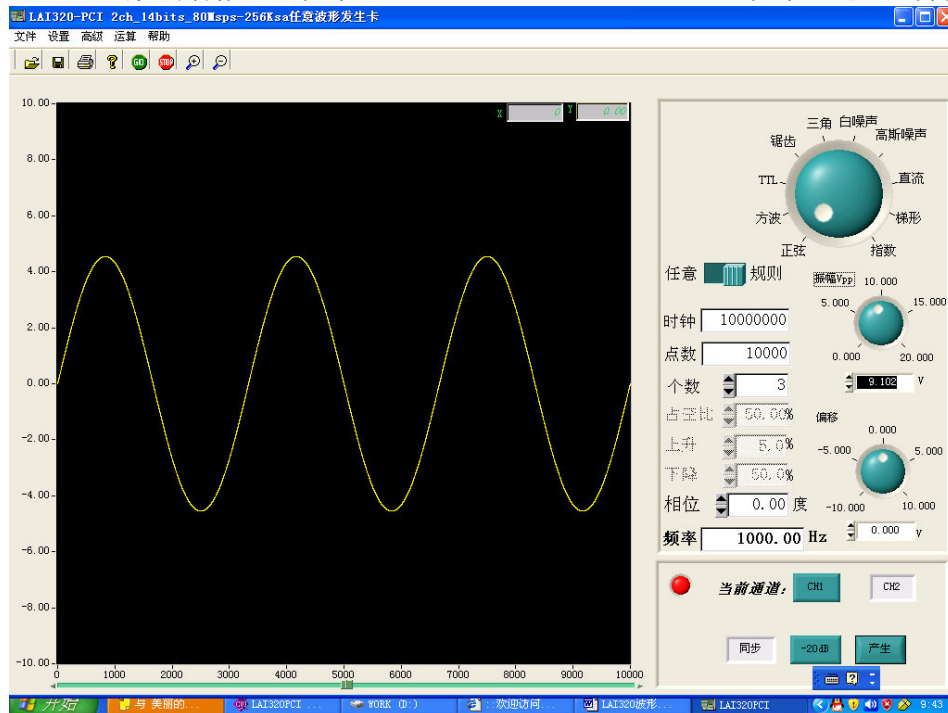
第三章 LAI320 软件使用说明

3.1、安装 LAI320 运行软件

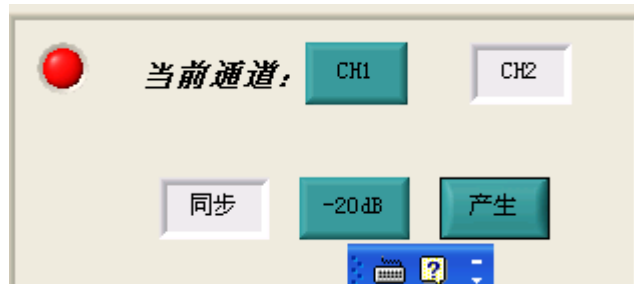
LAI320 安装软件在光盘的 Setup 目录下，运行 Setup.exe 程序，完成安装。

3.2、运行 LAI320

运行“开始”-“程序”-“LAI320PCI”-“LAI320PCI”程序，进入主界面，如下所示：



3.2.1 控制模块



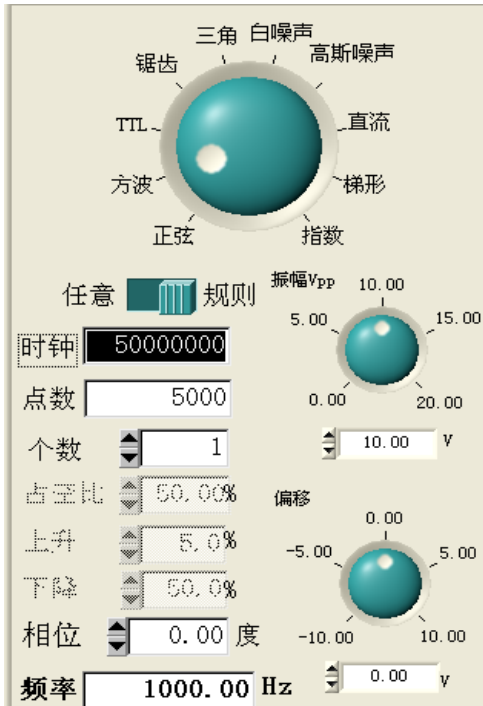
1、选择当前波形输出通道，按下 **CH1**，则所有操作是对本 CH1，按下 **CH2**，则所有操作是对本 CH2。两个通道的操作是相互独立的。

2、按 **产生**，产生波形，再按变为 **暂停**，无波形输出。

3、按 **-20dB**，波形输出衰减为 10:1，对小信号输出保持波形失真度很好。

4、按 **同步**，控制 CH2 的波形与 CH1 保持同步，波形相位差可设定。CH1 与 CH2 只能是同频率信号。

3.2.2 波形参数设置模块





1) **规则波形**为正弦、方波、TTL、锯齿、三角、白噪声、高斯噪声，直流等，由程序自动生成波形数据参数，用户可修改幅度、偏移、波形个数、频率，并在显示区内显示。

实际波形输出频率=频率*个数

规则波模式下时钟和点数由程序按优化原则自动生成，用户无法修改。波形个数、占空比、上升时间、下降时间等参数根据您所选择的规则波形的情况确定有效或无效，无效相应输入框变灰。否则您可以输入相应的设置参数。

2) **任意波**模式为用户鼠标画，可在规则波的基础上修改，可修改时钟和点数，为保证绘制的细

度，每屏为 128 点，可拖动水平滑块左右移动，可以  先将波形展开，画好细节在移动  到下一个细节再画，将整个屏幕的波形作为一个周期：

实际波形输出频率=时钟/点数

任意波模式下频率、幅度、偏移由用户绘制决定、更多的情况用户获的任意波形的方式由运算功能或文本编辑功能实现，这在附件“任意的实现”将详细介绍。

3.2.3 ARB 模式扫频率控制



- 1) 设置好扫频参数后，按确认后开始扫频
- 2) 扫频结束后，将提示。
- 3) 用户需要更精确的扫频，需要浏览“**扫频的实现**”

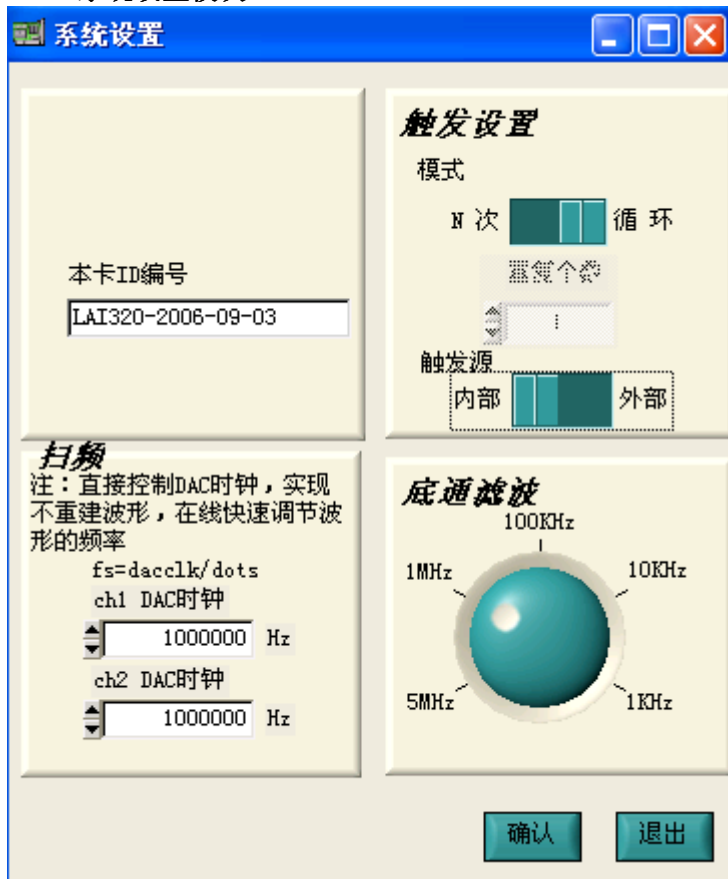
3.2.5 波形运算



- 1)本软件提供波形运算功能，以获得复杂的波形，如调幅波。
- 2)先选择波形 A
- 3)再选择波形 B
- 4)按“计算”按钮，+、-、*等计算结果并显示
- 5)重新保存为新文件。

注：参与运算的两波形点数要一致，否则要弹出出错信息。

3.2.7 系统设置模块



触发设置:N 次/循环,单次为每触发一次，只产生 n(1~4095)个波形,循环为一但触发后循环产生波形。

触发源: 内部触发：由软件启动。

外部触发：由外部信号启动。

滤波设置: 设置低通滤波档位，为 5 挡。

3.3 菜单结构

文件	
打开	打开一个波形
打开 MATLAB	打开 MATLAB 格式波形
保存	保存一个波形
打印	打印一个波形
退出	退出软件
设置	
系统设置	设置系统控制参数
高级	
ARB 扫频	ARB 模式正弦波扫频
运算	
Y-反向	波形 Y 轴反向
X-反向	波形 X 轴反向
A+B	两波形相加
A-B	两波形相减
调幅 AM	调幅波
调幅 FM	调频波
拼接	两波形相拼接
文本编辑	调用文本编辑器编辑波形
帮助	
关于	产品在线帮助

第四章 二次开发接口

4.1 编程接口

LAI320 提供 LAI320.DLL、LAI320.LIB、LAI320.H 文件供用户二次开发。在光盘 Example 下的 VcDemo、VbDemo、CviDemo、LabView 目录下相应的例程并用详细的注释，用户可在此基础上二次开发，应用到自己的测试系统。

4.2 LAI320.DLL 函数简介：

4.2.1 板卡自检函数

```
int LAI320_AutoCheck(unsigned char *Numbers,
                    unsigned int *CardAddress);
```

功能描述：初始化板卡，必须调用。

出口参数：CardAddress: LAI320-PCI 卡地址列表
: Numbers LAI320-PCI 卡总数。

函数返回：1，自检成功
2，自检失败

4.2.2 计算规则波形数据函数

```
void LAI320_CacuWavePara(
    int WaveType,
    int Cycles,
    double Frequency,
    double Amplitude,
    double Offset,
    int Duty,
    unsigned int *mDots,
    double *mDacclk,
    double *mWaveData,
    double kr,
    double kf);
```

功能描述：计算规则波形数据函数

入口参数：Wavetype: 波形类型 1、正弦 2、方波 3、TTL 4、锯齿 5、三角 6、白噪声 7、高斯噪声、8 直流。

Frequency: 波形频率

Amplitude: 幅度, 0-20Vpp

Offset: 波形偏置量 0-2.0V

Duty: 方波的占空比 0.1~99.9%

对三角波: kr 为上升斜率, kf 保留不用

对梯形波: kr 为上升斜率, kf 为下降斜率

对指数波: kr 为上升系数, kf 为下降系数

指数波表达:

上升段: $V = \text{Amp} * \text{kr} + \text{Offset}$

下降段 $V = \text{Amp} * e^{(-t * \text{kf} * 20/T)} + \text{Offset};$

出口参数: mDots : 波形时钟

mDacclk : 波形时钟

mWaveData : 波形数据 范围: -10.0V---+10.0V

4.2.3 产生波形

```
int LAI320_GenWave(
    int CardAddressess,
    int ch,
    int loopmode,
    unsigned int Dots,
    double dacclk,
```

```

double *mWaveData,
int sysoff,
int fcidx );

```

功能描述：产生波形

入口参数： CardAddress: 板卡的基地址。
 Ch: 0->ch0 1->ch1
 Loopmode: 0->循环产生
 1->N次
 Dots: 波形的点数
 Dacclk: 波形的时钟
 mWaveData: 波形数据
 sysoff: 系统的零点偏移
 fcidx: 滤波挡位

4.2.4 ARB 正弦扫频控制函数

LAI320-PCI 卡函数原型为：

```

int LAI320_ArbScanfreq(
    int CardAddressess,
    int ch,
    double delaytime,
    unsigned int dots,
    double Amplitude,
    double startfreq,
    double endfreq,
    double deltafreq,
    int cycles,
    int fcidx );

```

入口参数： CardAddressess: 板卡的基地址。

Ch: 通道
 delaytime: 扫频时间间隔
 dots: 点数
 Amplitude: 幅度
 Startfreq: 起始频率
 Endfreq: 结束频率
 Cycles: 扫频次数
 Fcidx: 滤波挡位

4.2.5 设置触发源

```

void LAI320_SetTrigSource(int CardAddress,
    int ch,
    unsigned char TrigSource);

```

入口参数： CardAddress: 板卡的基地址。
 TrigSource: 0->软件内触发
 1->外部上边沿触发
 ch: 通道

4.2.6 读 LAI320 卡系统参数

```

void LAI320_ReadConfig(unsigned int CardAddress,
    int ch,
    int length,
    unsigned char *CardID,

```

```
int *CardOffset,
unsigned char *Type);
```

功能描述：读 LAI320 卡的系统参数

入口参数：CardAddress：板卡的基地址。
 Length：固定为 30
 CardID：LAI320-PCI 卡的编号
 CardOffset：卡 CH1 的零点偏置
 Type：卡 CH2 的零点偏置

4.2.7 设置低通滤波

```
void LAI320_SetFilter(int CardAddress,
int ch,
int fcidx);
```

功能描述：设置低通滤波档位

入口参数：CardAddress：板卡的基地址。
 Fcidx：低通滤波，
 0=>1MHz
 1=>100KHz
 2=>10KHz
 3=>1KHz,
 4=>5MHz

4.2.8 直接控制 DAC 时钟函数

```
int LAI320_SetDacClk(int CardAddressess,
int ch,
double DacFrequency)
```

函数说明：直接控制 DAC 时钟

入口参数：CardAddressess：卡的地址
 ch：通道
 DacFrequency：DAC 输出时钟
 范围（0~80MHz）

4.2.9 设置输出衰减函数

```
void LAI320_SetAttr(int CardAddressess,
int ch,
int AttrStatus);
```

功能描述：设置输出衰减

入口参数：CardAddressess：卡的地址
 ch：通道号
 AttrStatus：0：不衰减
 1：-20Db(10:1) 衰减

出口参数：无

4.2.10 设置波形个数

```
int LAI320_SetWaveNumbers(unsigned int CardAddress,
int ch,
int Numbers);
```

功能描述：设置低通滤波档位

入口参数：CardAddress：板卡的基地址。
 ch：通道号
 Numbers：1~4095

5.5 波形文件数据结构

波形文件采用便于阅读和编辑的 txt 文件，第一个值波形点数，以后为数据流电压值，例如：

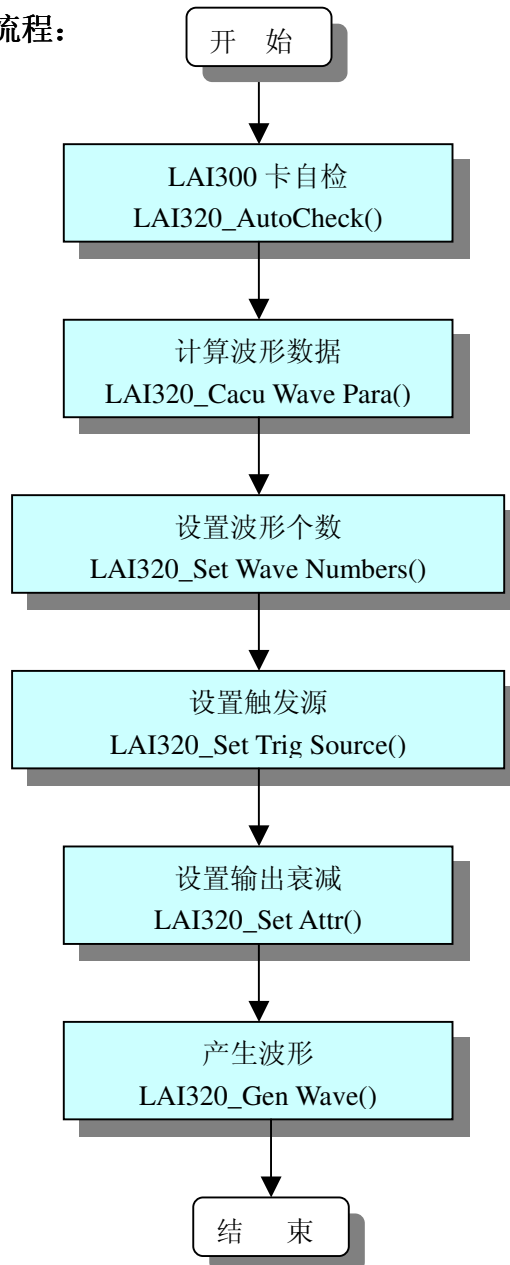
138261

0.039318

0. 023693

0. 017833

.....

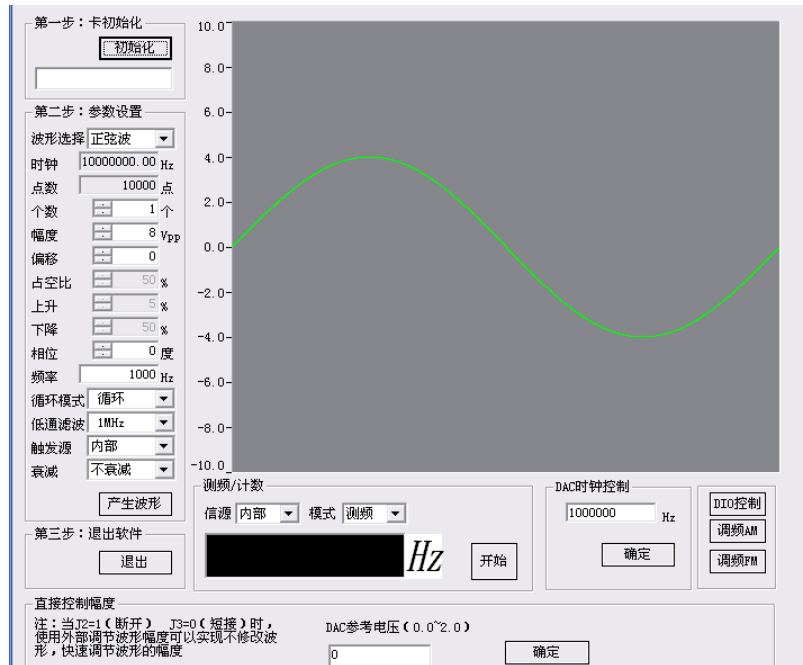
5.6 函数一般调用流程:

第五章：二次开发例程序

本卡提供在 VC、VB、LABVIEW、CVI、C++BUILDER 语言下的例程，用户可利用这些例程为蓝本，快速地开发自己的应用程序。

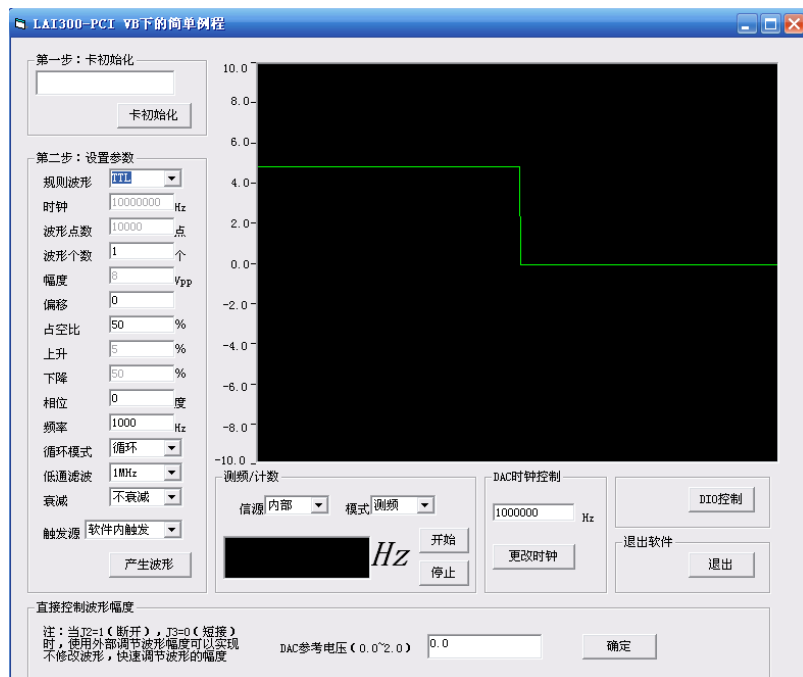
5.1 VC++ 例程

在光盘\任意波形发生卡\PCI 任意波形发生卡\LAI320-PCI\Examples\VC 目录下的 Example.dsw。



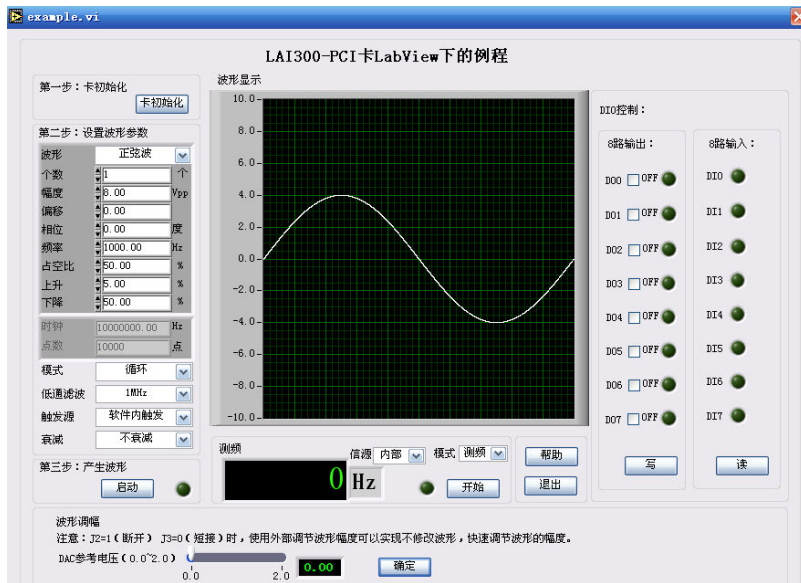
5.2 VB 例程

在光盘\任意波形发生卡\PCI 任意波形发生卡\LAI320-PCI\Examples\VB 目录下的 LAI320VB。



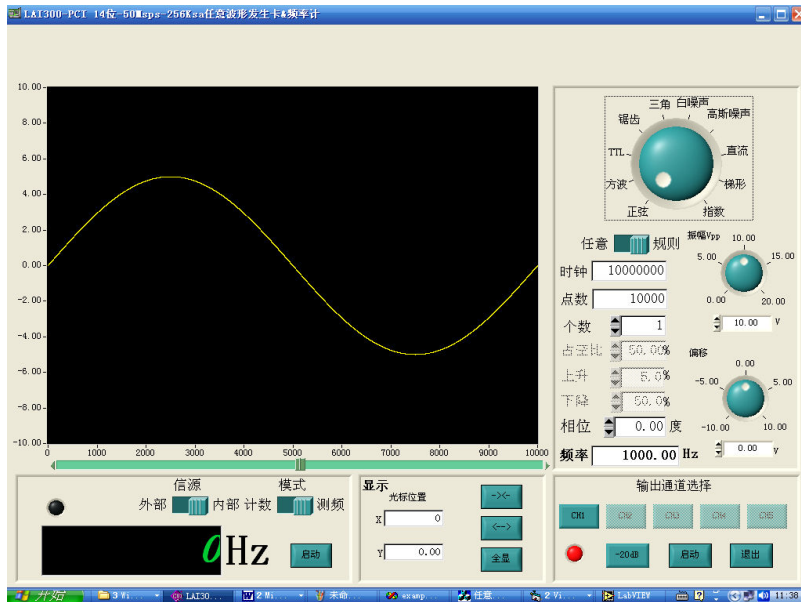
5.3 LABVIEW 例程

在光盘\任意波形发生卡\PCI 任意波形发生卡\LAI320-PCI\Examples\LABVIEW 目录下的 EXAMPLE。



5.4 CVI 例程

在光盘\任意波形发生卡\PCI 任意波形发生卡\LAI320-PCIEamples\CVI 目录下的 LAI320PCI。




5.5 C++builder 例程

在光盘\任意波形发生卡\PCI 任意波形发生卡\LAI320-PCIEamples\C++builder 目录下的 example。

附件一、 LAI320 任意波形发生卡的典型应用

一、任意波形的编辑

1、鼠标画方式

先用规则波形方式产生一个波形蓝本，选择到任意波形模式，
，可在此规则波的基础上修改，可修改时钟和点数，为保证绘制的细度，每屏为 128 点，可拖动水平滑块左右移动，可以先将波形展开，画好细节在移动到下一个细节再画，将整个屏幕的波形作为一个周期：

实际波形输出频率=时钟/点数

将您所绘制好的波形保存起来，方便下次直接调用。

2、文本编辑

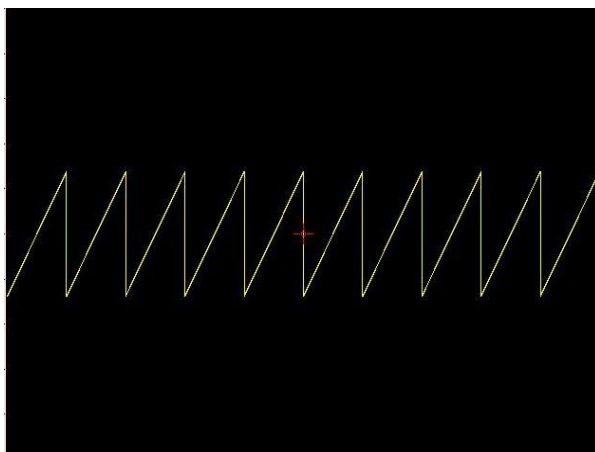
在菜单下，“运算”下的“文本编辑”下，直接打开波形文件，您可以直接修改所对应的波形数据，修改好后，直接存盘，下次调用此波形。

3、运算产生

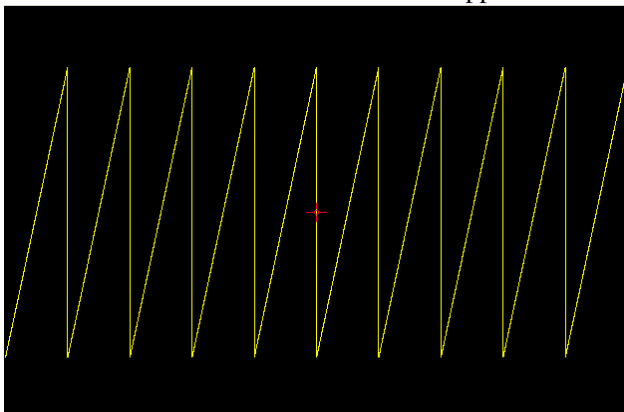
对一些特殊的波形，如分段波形等，可采用两个规则波形通过运算生成一个新的波形，直接存盘，下次调用此波形。下面举一个例子：

要产生一个锯齿波形，前 10 个波形 V_{pp} 为 5V，后 10 个波形 V_{pp} 为 15V，总的重复周期为 400ms。

3.1，先选择锯齿波，输入波形个数为 10 个， V_{pp} 调节到 5V，频率为 100Hz，得到如下波形



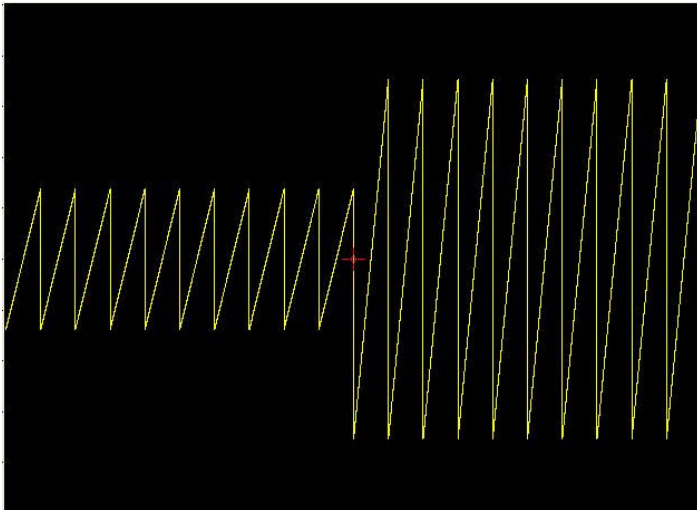
3.2，先选择锯齿波，输入波形个数为 10 个， V_{pp} 调节到 15V，频率为 100Hz，得到如下波形



3.3，先选择运算功能下的拼接功能



得到如下波形

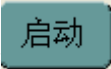


将此文件保存起来 SAW3.TXT，以后可直接调用。

3.4. 产生波形

打到任意波模式，用文件方式打开 SAW3.TXT,本波形长度为 40000 个点，根据 “实际波形输出频率=时钟/点数”

$$\text{时钟} = 40000 * 2.5(400\text{ms}) = 100000$$

将时钟设置为 100000,按  产生波形。

4、二次开发

用户可二次开发，通过计算得出波形数据，控制产生特殊的波形。

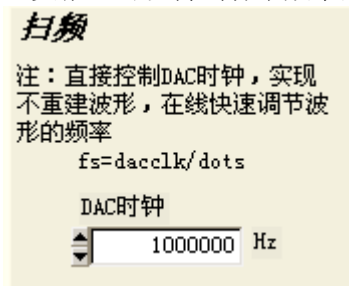
二、扫频的实现

本卡的设计是将 DAC 时钟和波形生产电路完全独立开来，所以不需要重波形，很轻松地实现扫频的功能：

假设：一正弦波 从 20Hz~20KHz ，间隔为 100Hz

- 1、在软件上选择正弦，20KHz
得到的时钟是 50M，波形点数 2500。
生成此波形。
- 2、10KHz 时，DAC 时钟=2500*10000=25MHz
按“设置”下的，修改 DAC 时钟即可得到 10KHz 波形。
- 3、100Hz 时，DAC 时钟=2500*200=500KHz
按“设置”下的，修改 DAC 时钟即可得到 100Hz 波形。

以此类推，可以得到各个频率的信号，实现扫频功能。

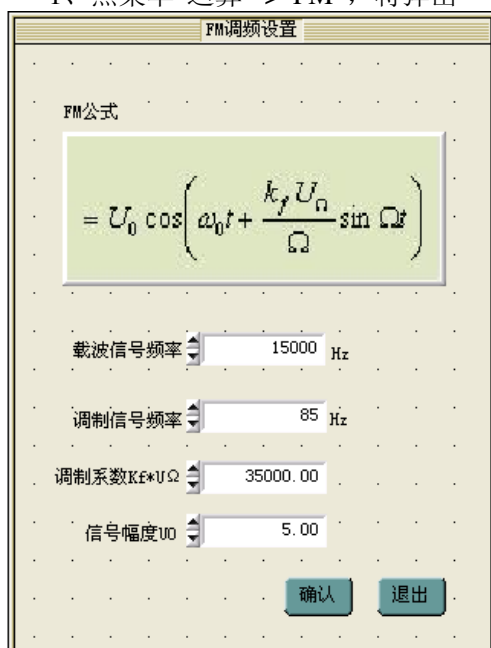


4、若要精确的确定各扫描间隔：

- 1、可以调用扫频功能，计算机将只完成此功能，对其他程序不响应，直到扫描结束。时间间隔与系统配置有关。
- 2、可以二次开发：按一定的时间间隔修改 DACCLK。

四、调频FM

1、点菜单“运算”->“FM”，将弹出



- 1、根据公式，设置好各项参数
- 2、产生波形，是将整个波形作为一个大周期循环产生。

五、调频FM

1、点菜单“运算”->“AM”，将弹出

AM调幅设置

FM公式

若载波信号为

$$c(t) = K_0 \cos(\omega_c t + \varphi_0)$$

则调制后的信号为

$$S_{AM}(t) = K_0 m(t) \cos(\omega_c t + \varphi_0)$$

取: $m(t) = \sin(\omega_m t)$, $\varphi_0 = 0$

载波信号频率 ω_c Hz

调制信号 $m(t)$ 频率 Hz

信号幅度 K_0

- 2、按公式，设置好各项参数
- 3、生波形，是将整个波形作为一个大周期循环产生。