# 目录

# 第1章 产品介绍

# 1. 什么是 Acute-PG 可程序信号发生器？

皇晶科技股份有限公司研发之 PC-based 可程序信号发生器－Acute PG1000、PG2000 系列及 Acute Pocket PG 系列，是一台可以产生多种数字波形的可程序信号发生器，具有高速，多信道及多用途的功能，可轻易的编辑波形，并输出至测试板上，不需大费周章制作一个集成电路。因此，拥有一台 Acute PG 可节省许多开发成本。

不管是电路仿真、IC 芯片测试、唯读记忆体仿真、总线分析仿真、零件烧录....等等波形，都可以用 Acute PG 来产生。常用的数字传输界面已做成制式功能，让您轻易的产生您要的波形。提供多种波形数据编辑方式：直接画波形、文字文件转入、Altera 波型档转入、Acute LA 波形档转入...。完美的 Window 操作接口，可与笔记本连接，易于携带，完全满足研发者挑剔的需求。

搭配 Acute 逻辑分析仪(以下简称 LA)时，可以用 LA 来撷取量测物的波形，再交由 Acute PG 将波形输出给另一个集成电路。此外，更可组合 Acute LA 与 Acute PG 做成自动测试系统或自动验证系统。皇晶科技将陆续提供各种常用的 PLD 与 FPGA 仿真的波形转换程序，让已经画好的波形或是写好的文字向量档案都能轻易的转换成实际的波形，来验证 PLD 和 FPGA。

**注 1：可程序数据产生器简称 PG**

**注 2：Acute PG1000 系列包括 PG1020、PG1050。(简称 PG1000)**

**注 3：Acute PG2000 系列包括 PG2020、PG2050。(简称 PG2000)**

**注 4：Acute Pocket PG 系列包括 PKPG2016、PKPG2116、PKPG2116+。(简称 PKPG)**

## 2. 产品配备-PG1000 及 PG2000

| | 项目 | PGx020 | PGx050 |
|---|---|---|---|
| 1 | PG1000 or PG2000 主机 | 1 | 1 |
| 2 | 信号隔离放大器 | 2 | 5 |
| 3 | 特殊信号隔离放大器 | 1 | 1 |
| 4 | 信号连接线，每个小组是 1x10 多蕊单芯线， | 2 | 5 |
| 5 | 特殊信号连接线，每组为 1x12 多蕊单芯线 | 1 | 1 |
| 6 | 地线/black 1x2 line with red mark | 3 | 6 |
| 7 | 信号探头(红) | 35 | 68 |
| 8 | PCI 传输适配卡 (内接式配备) | 1 | 1 |
| 9 | 界面卡连接排线 (25pins) (内接式配备) | 1 | 1 |
| 10 | 电源转接电缆线 (内接式配备) | 1 | 1 |
| 11 | 12VDC2A 整流稳压器 (外接式配备) | 1 | 1 |
| 12 | Print Port 传输线 (外接式配备) | 1 | 1 |
| 13 | USB 转换器 (外接式选购配备) | 1 | 1 |
| 14 | 安装光盘片 | 1 | 1 |
| 15 | 说明书 | 1 | 1 |
| 16 | 螺丝 | 1 | 1 |

| 1. | 2. | 3. | 4. |
|---|---|---|---|
|  |  |  |  |
| **5.** | **6.** | **7.** | **8.** |
|  |  |  |  |
| **9.** | **10.** | **11.** | **12.** |
|  |  |  |  |

# 3. 产品配备-Pocket PG

| | 项目 | PKPG 2016/2116/2116+ |
|---|---|---|
| **1.** | Pocket PG 主机 | 1 |
| **2.** | 信号连接线 1x20 color line | 1 |
| **3.** | 地线/black 1x2 line with red mark | 1 |
| **4.** | 信号探头 Probe(Red) | 22 |
| **5.** | USB A-B 接线(1.8m) | 1 |
| **6.** | 安装光盘片 | 1 |

| 1. | 2. | 3. |
|---|---|---|
|  |  |  |
| **4.** | **5.** | **6.** |
|  |  |  |

# 4. 产品技术参数-PG1000 及 PG2000

| 技术参数 | | | PGx020 | PGx050 |
|---|---|---|---|---|
| 电源 (Power) | 电源(Power)(Internal/External) | | PC Power/Adapter (12V) | |
| | 静态消耗功率(Static Power Dissipation) | | 2.4W | |
| | 瞬间最大消耗功率(Max Power Dissipation) | | <12W | |
| 传输界面 (Interface) | 内接式(Internal) | | PCI Card | |
| | 外接式(External) | | Printer Port or USB | |
| 通道(Channels) | | | 20 | 50 |
| 内部频率 (Internal Clock) | | 范围(Range) | 100MHz~1Hz | |
| | | 模式(Mode) | 可微调 | |
| | | 输出通道(Output Channel) | 1 | |
| 外部频率 (External Clock) | | 范围(Range) | <75MHz | |
| | | 模式(Mode) | Logic Not & Logic And | |
| | | 输出通道(Output Channel) | 2 | |
| 数据输出控制方式(Data Flow Control) | | | Loop<br>Jump<br>Wait for Event<br>Branch (If command) | |
| 输出形式(Output Type) | | | POD A-E: 1.5V-5.5V with 255Scakes<br>External POD: UART, I2C | |
| 标准串行信号输出速度(UART Baud Rate) | | | 110-256K | |
| 输出驱动能力(Fan Out) | | | 8 个 TTL | |
| 相位误差(Data Skew) | | | <3ns | |
| 内存(Memory) | 波形存储深度(Storage Depth /Channel) | | 标准规格 64K bits 注 | |
| 内部事件(Internal Event) | | | Hot Key | |

| | | |
|---|---|---|
| 外部事件<br>(External Event) | 通道(Channel) | 3 |
| | 模式(Mode) | Logic Not & Logic Or |
| | 准位(Threshold) | TTL Level |
| 工作温度(Operating Temperature) | | 5oC~45oC (41oF~113oF) |
| 保存温度(Storage Temperature) | | -40oC~75oC (-40oF~167oF) |
| 使用语系(Languages) | | 中文、英文(可切换) |
| 波形存储、读取(Save & Load Waveform) | | 可 |
| 波形打印(Print Waveform) | | Windows 可接受的打印机皆可 |
| 在线辅助功能(Online Help) | | 有 |
| 体积(Dimensions) | 长 x 宽 x 高 (mm) 3 | 197 x 147 x 42 (mm) 3 |

**注：加大容量选项：512K bits**

# 5. 产品技术参数-Pocket PG

| 技术参数 | | PKPG2016/2116/2116+ |
|---|---|---|
| 电源 (Power) | 电源(Power Source) | USB bus-power (+5V) |
| | 静态消耗功率(Static Power Dissipation) | 0.75W |
| | 瞬间最大消耗功率(Max Power Dissipation) | <2.5W |
| 传输界面(Interface) | | USB |
| 内部频率(Internal Clock) | 范围(Range) | 200MHz ~ 1Hz |
| | 模式(Mode) | 可微调 |
| | 输出通道(Output Channel) | 1 |
| 外部频率 (External Clock) | 范围(Range) | <200MHz |
| | 模式(Mode) | Logic Not & Logic And |
| | 输入通道(Input Channel) | 1 |
| 数据输出控制方式(Data Flow Control) | | Loop<br>Jump<br>Wait for Event<br>Branch (If command) |
| 输出形式(Output Type) | | 5V，3.3V，3.0V，<br>2.8V，2.5V，2.1V，<br>1.8V，1.5V，Hi-Z |
| 输出驱动能力(Fan out) | | 8 个 TTL |
| 相位误差(Data Skew) | | <3ns |
| 内存(Memory) | 波形储存深度(Storage Depth/Channel) | 512K bits 注 |
| 内部事件(Internal Event) | | Hot Key |
| 外部事件 (External Event) | 通道(Channel) | 2 |
| | 模式(Mode) | Logic Not & Logic Or |

| | | |
|---|---|---|
| | 准位(Threshold) | TTL 3.3 Level |
| 工作温度(Operating Temperature) | | 5°C ~ 45°C |
| 保存温度(Storage Temperature) | | -40°C ~ 75°C |
| 使用语系(Languages) | | 中文、英文(可切换) |
| 波形储存、读取(Save & Load Waveform) | | 可 |
| 波形打印(Print Waveform) | | 可 |
| 在线辅助功能(Online Help) | | 有 |
| 体积(Dimensions) | 长 x 宽 x 高 (mm) 3 | 117 x 72 x 20 (mm) 3 |

**注：PKPG2016：存储深度为 64K bits / channel，PKPG2116，PKPG2116+：存储深度为 512K bits / channel。PKPG2016, PKPG2116 ：内频范围 100MHz~1Hz，外频范围 < 75MHz，PKPG2116+ ：内频范围 200MHz~10Hz，外频范围 <= 200 MHz。**

# 6. 使用环境

- PC INTEL 486 以上兼容产品，建议使用 Pentium 100 以上等级。
- PC 内存容量 32M Bytes 以上。
- 硬盘可使用空间 10M Bytes 以上。
- 光驱(安装程序用)。
- 显示规格 640x480 VGA 以上，建议使用 800x600 或 1024x768。
- 使用 101 键盘，建议使用 Windows 键盘。
- 二键或三键鼠标。
- 一组 PCI Bus 插槽。注 1
- Printer Port 传输界面。注 1
- USB 传输界面。注 1
- 打印机(非必需)。
- 中文版 Windows 95/98/ME/NT/2000/XP/7 操作系统。注 2

**注 1：使用接口并非必需。Pocket PG 可使用的接口只有 USB； PG1000 及 PG2000 可使用的接口为 PCI、EPP 或 USB(依据选购配件而定)。**
**注 2：使用 USB 接口，至少需要 Win 98se 以上。**

# 第2章 安装

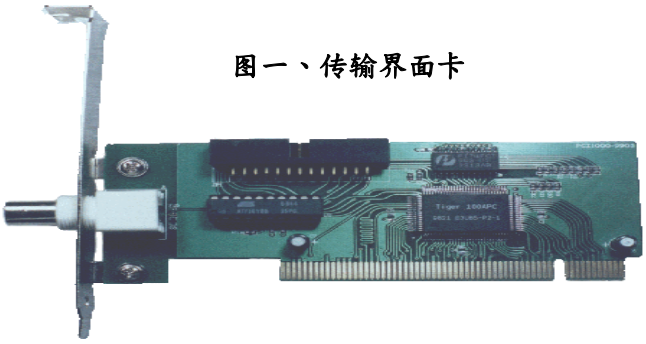
# 1. 安装步骤

## 1.1 硬件安装

### (1) PG1000、PG2000 内接式(使用 PCI 接口)

**a.** 关闭 PC 的电源，并打开 PC 的外壳。

**b.** 将传输界面卡(图一)安装在 PC PCI Bus 的插槽上，并将螺丝锁紧。

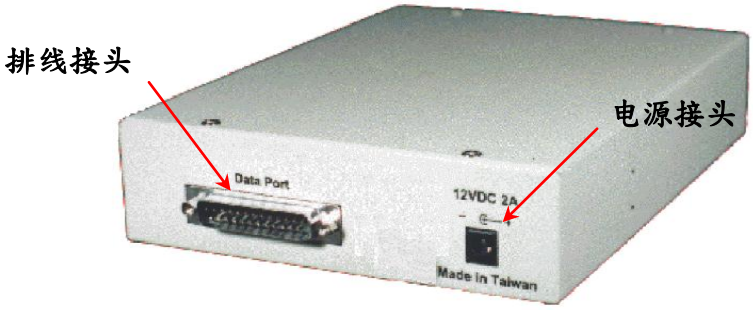


**图一、传输界面卡**

**c.** 将 PG1000/PG2000 主机(图二)安装在 PC 的 CD-ROM 插槽。

**d.** 将界面卡连接排线，从传输界面卡接到 PG1000/PG2000 主机背面排线接头上。

**e.** 将 PC 的磁盘驱动器的电源(5V,12V)透过电源转接器，接到 PG1000/PG2000 主机的背面电源接头。




**图二、PG1000/PG2000 主机**

### (2) PG1000、PG2000 外接式(使用 EPP 或 USB 接口)

**a.** 将 Printer Port 传输线，连接在 Acute PG 与 PC 的 Printer Port 之间。(或是用 USB 传输线连接 PG1000/PG2000 与 USB Port)

**b.** 将电源稳压器(12V)接在 PG1000/PG2000 的电源接头上。USB 界面请跳至

步骤三，Printer Port 界面请跳至步骤四。

**c.** 如果使用 USB 传输界面，进入 Windows 环境时会出现安装 USB 驱动程序的画面，此时只要选择磁盘上的驱动程序即可。

**d.** 打开 PC 电源，并进入 BIOS 设定模式，将您使用的 Printer Port 设定成 EPP 模式，假如没有 EPP 模式，可以设成 EPP+ECP 模式、ECP 模式或是 Bi-direction 模式。但是如果您使用 Windows NT 的话，您只能选用 EPP 模式。注意!您的 PC BIOS 设定如果有 EPP1.7 和 EPP1.9 选项时，请选用 EPP1.9。(请参考故障排除一章说明)

**(3) Pocket PG**

**a.** 将 USB Cable 正方形一端接在 PKPG 上。

**b.** 将 USB Cable 长方形一端接在 PC USB Port 上。

## 1.2 量测套件安装

**(1)** 将信号隔离放大器(请参考第五页的配件图)的排线接到 PG1000/PG2000 主机的面板上，记得要依照面板标示的 A、B、C、D、E 及 Extended Pod 一对一的与信号隔离放大器的标示相符。(PKPG 无隔离放大器，只需将彩虹排线及地线接 上即可)

**(2)** 将信号连接线(10Pins)接在信号隔离放大器上，另一端则接上探头。

**(3)** 将信号连接线地线(2Pins)接在信号隔离放大器上，另一端也接上探头(只用 1Pin 即可)。

**(4)** 将接地探头接(有红色套环)在待测物的地线(Ground)上。

**(5)** 将一般信号探头接在标的物上。

## 1.3 驱动程序安装

驱动程序安装分为三种情况，会因使用不同界面及不同操作系统有所区别。**注意! USB 无法与 Windows95/98SP1/NT 等操作系统搭配。**

**(1) 方式 1** (不需要或不须安装驱动程序)

以下环境组合请直间跳至软件安装。

- Printer Port + Windows 95
- Printer Port + Windows 98
- Printer Port + Windows ME
- Printer Port + Windows NT
- PCI + Windows NT

**(2) 方式 2** (随插即用 Plug & Play)

**以下环境组合请跳至步骤 A1。**

- PCI + 所有操作系统(不包含 Windows NT)
- USB + 所有操作系统

**(3) 方式 3** (非随插即用，但须驱动程序)

以上环境组合请跳至步骤 B1。

- Printer Port + Windows 2000
- Printer Port + Windows XP

**A1.** 硬件安装完成后，打开 PC 电源并进入 Windows 操作系统。

**A2.** 将安装磁盘放入光盘片至光驱中。

**A3.** Windows 在启动时会找到硬件装置的画面，请依照 Windows 的指示操作。

**A4.** 如操作系统要求驱动程序时，请指示 Windows 至光驱上寻找驱动程序。

**A5.** 如果 Windows 无法正确找到驱动程序时，请直接指示 Windows 到光驱的根目录，来寻找适合的驱动程序。

**A6.** 如果仍然无法正常安装驱动程序，请参考故障排除章节或参考本公司网站

提供的 FAQ。仍无法解决时请与本公司联络。

**A7.** 如果正常安装驱动程序时，您可从『装置管理员』中找到一个『Acute PC-Based Instrument』的项目，该项目里面会包括您安装的驱动程序。接下来请安装软件。

**B1.** 硬件安装完成后，打开 PC 电源并进入 Windows 操作系统。

**B2.** 将安装磁盘放入光盘片至光驱中。

**B3.** 从『控制台』中选择『新增/移除硬件』。

**B4.** 选择新增装置。

**B5.** 此时 Windows 会尝试去寻找硬件装置，但是 Windows 会找不到任何已知的硬件装置。

**B6.** Windows 会要求选择一种装置，此时请选择『新增一项装置』。

**B7.** 选择手动选择硬件类型，不要让 Windows 自行搜寻硬件。

**B8.** 在 Windows 2000 时选择『其它装置』，在 Windows XP 时选择『显示所有装置』(Windows 可能会花很长的时间，请耐心等待)。

**B9.** 选择『从磁盘安装』。

**B10.** 指定目录至光驱的根目录。

**B11.** 选择『Acute EPP & ISA Interface Driver』。

**B12.** 如果仍然无法正常安装驱动程序，请参考故障排除章节或参考本公司网站提供的 FAQ。仍无法解决时请与本公司联络。

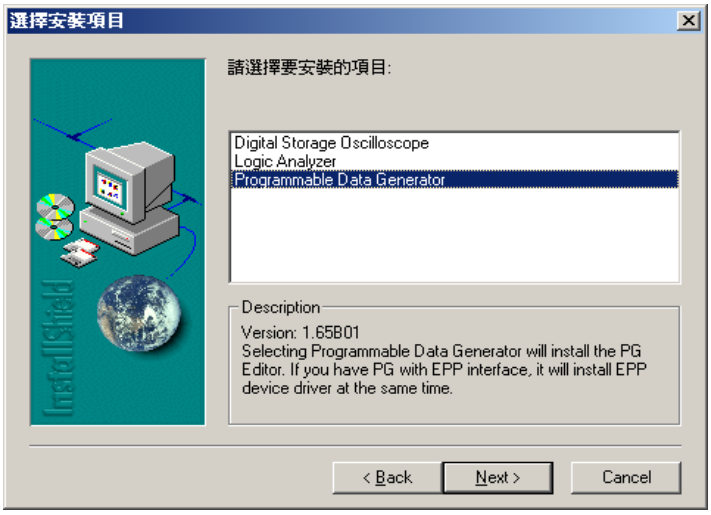
**B13.** 如果正常安装驱动程序时，您可从『装置管理员』中找到一个『Acute PC-Based Instrument』的项目，该项目里面会包括您安装的驱动程序。接下来请安装软件。

## 1.4 软件安装( PG Editor 安装)

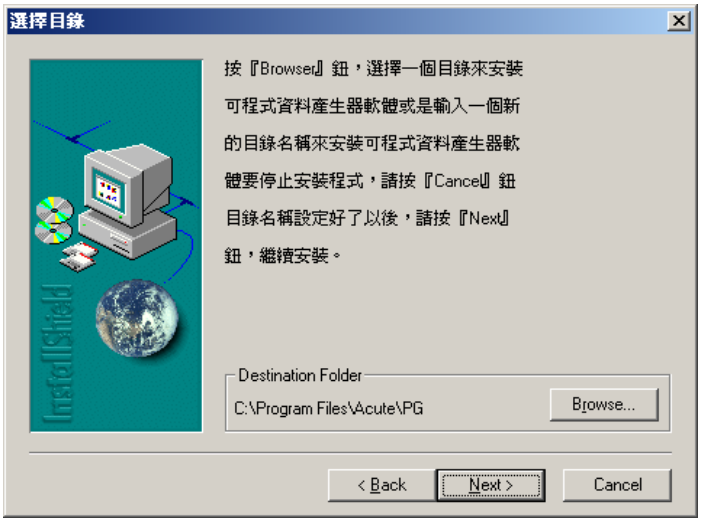
**(1)** 如果您已经安装较旧 PG-Editor 版本时，请先移除旧版本，此移除动作不会将您旧环境设定消除请不用担心。

**(2)** 将安装光盘片放入 CD-ROM 中。

**(3)** 『我的计算机』中找到安置安装光盘的光驱，并点选该光驱，此时应会自动进入安装程序，如果没有自动进入安装程序，请选择光盘中的 Setup.exe 程序并执行它。安装程序会根据操作系统的语系自动安装该语系的版本。(目前只支持中文及英文，非中文的语系皆会安装成英文版) 光盘片中可能含有其它 Acute 相关软件，此时只要选择可程序数据产生器的选项即可。如果安装其它产品软件，该软件因没有硬件配合，所以只会进入 Demo 模式。




**(4)** 依照安装软件的提示输入安装目录。



**(5)** 开始安装软件。

**(6)** 安装结束后，Desktop 与程序集中都有可程序数据产生器的启动图标  ，可以任选一个来启动 PG-Editor。

**(7)** 如果执行 PG-Editor 时，出现 Demo Mode 就代表您的安装上出了问题。请参考故障排除。

## 2. Printer Port 的 BIOS 模式设定

并列端口的规格有很多种，兹将一般 BIOS 有提供的模式与 Acute PG 支持的模式列表于下，请依下表的先后次序找寻您 BIOS 有提供的模式来设定，以达到外接式最佳的效率。

| 优先次序 | BIOS 支持模式 | 备注 |
|---|---|---|
| 1 | EPP 1.9 | 效率最好，建议使用 |
| 2 | ECP + EPP 1.9 | 同上 |
| 3 | EPP | 有可能是 EPP 1.9，也有可能是 EPP 1.7 |
| 4 | EPP 1.7 | 效率较 EPP 1.9 稍差 |
| 5 | SPP + EPP 1.7 | 同上 |
| 6 | ECP + EPP 1.7 | 同上 |
| 7 | ECP + Bi-direction | 效率比 EPP 差 |
| 8 | Bi-direction | 效率比 EPP 差 |
| 9 | ECP | 有些计算机可用，有些不行 |
| 10 | ECP + SPP | 有些计算机可用，有些不行 |
| 11 | SPP | 有些计算机可用，有些不行 |
| 12 | Normal | 不支援 |

# 3. 问题反应

您在安装或使用上如果遇到任何问题，请参考注意事项及故障排除的章节说明，如果仍然无法解决问题，或是对本产品有任何的建议，请与我们联络，我们会尽快的处理您的问题。

我们的网页是 http://www.acute.com.tw，网络上有许多新的讯息，包括新版软件、FAQ、新产品介绍.....等等。如果网页上的信息无法满足您的需求时，也可以用 E-mail 的方式，将您的问题告诉我们，我们的 E-mail 地址是 service@acute.com.tw。

# 第3章 功能说明

# 1. 系统内容及设定


系统内容设定主要是设定程序的使用语系，每一种语系可以随时自由的切换，可以根据个人需求来做调整。另外还有两个讯息字段，一个是使用机种，另一个是使用界面。Acute PG 的机种分成 PG1020、PG1050、PG2020、PG2050、PKPG2016、PKPG2116 以及 PKPG2116+，PG1020 与 PG2020 为 20 个通道的机种，PG1050 与 PG2050 为 50 个通道的机种，PKPG2016、PKPG2116 与 PKPG2116+ 为 16 个通道的机种。而传输界面则分为内接式及外接式两种，内接式为 PCI 界面卡，而外接式则有 Printer Port 及 USB 界面。使用 Printer Port 的界面时，要注意 BIOS 的设定，由于使用 Printer Port 界面会和主机板使用的 I/O Chip Set 以及 BIOS 的设定有关，所以如果设定不对有可能会变成无法与 Acute PG 连接或是使传输速度变慢。(请参考 Printer Port 的 BIOS 模式设定一节。)PG1000、PG2000 及 PKPG 系列提供了输出电压可调的功能。PG1000 及 PG2000 输出准位 1 控制信道 0 到信道 19，输出准位 2 控制信道 20 到信道 49；PKPG 只能调整输出准位 1，输出准位 1 控制信道 0 到信道 15，且准位只有 8 阶。(PKPG 准位一但设定好后，16 个通道皆会变成该准位，因此切换准位前请先确认待测物的技术指标，否则待测物可能会因而损坏。)

# 2. 主机频率设定详述

Acute PG 的内部基频有两大系统，一是固定频率，另一是微调频率。所谓固定频率是包含 100MHz、80MHz、50MHz、40MHz 和 100KHz 等 4 种固定频率。而且 Acute-PG 有四组 10 倍数的除频器，因此以 100MHz 的基频来讲，就可以产生 10MHz、1MHz、100KHz 和 10KHz 等频率。依此类推，80MHz 就可产生 8MHz、800KHz、80KHz 及 8KHz 等频率值。

另一种微调频率系统有两组基频。第一组为 15MHz 到 75MHz，微调的基本单位为 100KHz。也就是可以产生 15MHz、15.1MHz、15.2MHz、…、74.9MHz 及 75MHz 等频率值。第二组的频率范围为 1.25MHz 到 15MHz，但是它的微调基本单位为 50KHz，可以产生的频率为 1.25MHz、1.3MHz、1.35MHz、…、14.95MHz 及 15MHz 等频率值。

当然微调的频率系统下亦可以使用 Acute PG 的内部四组 10 倍数的除频器。因此 15MHz 到 75MHz 且微调值为 100KHz 的环境，一旦除以 10 以后，就会变成 1.5MHz 到 7.5MHz 的范围，而且微调值会变成 10KHz，换句话说可产生的频率值为 1.51MHz、1.52MHz、1.53MHz、…、7.49MHz 及 7.5MHz。

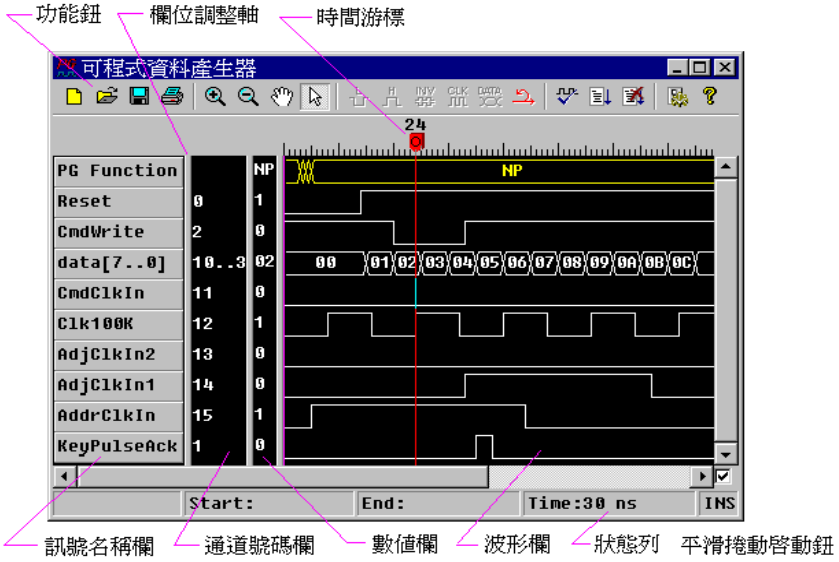
虽然频率值的定义似乎有些复杂，但是在 PG-Editor 的程序环境中，您大可不必太担心这个问题，因为一旦设定错误的频率时，PG-Editor 都会给予最接近该设定的频率值。

# 3. 功能介紹

## 3.1 波形编辑器

波形编辑器提供了一个画波形的环境，透过波形编辑器可以画出您想要输出的波形，同时波形编辑器包含了许多快速画波形的方式，例如 Clock 或 Counter 等的波形产生，并且包括了许多便于修改的使用环境。

### 波形编辑器的快速使用方法




**(1)** 进入波形编辑器。

**(2)** 用『环境参数设定』的功能来设定『工作频率』及『可编辑时间长度』。

**(3)** 启动信号选单 (将鼠标移至信号名称的字段上，按下鼠标的右键，画面将出现选单)。

**(4)** 请选择信号选单上的『新增信号』，会出现一个对话盒。

**(5)** 设定信号名称、信号编号及信号颜色。

**(6)** 重复步骤三、四、五，直到所需要的信号名称都设定好为止。

**(7)** 用鼠标左键在波形区中，利用标记方式来选择某一波形区块，然后再用『低电位』、『高电位』、『波形反相』、『编辑频率』或『编辑信号组数据』等方式来改变波形。

**(8)** 也可以用鼠标的右键在波形区中，进入波形选单(将鼠标移至波形区的字段上，按下鼠标的右键，画面将出现选单)，利用剪贴、复制及还原等功能来修改波形。

**(9)** 将地线接到待测电路上的地线位置。

**(10)** 将探头接到待测电路上，探头编号需与先前设定的信号编号相同。

**(11)** 按下『输出波形』钮。

## 3.2 画面窗口调整

波形编辑画面中的字段(窗口)大小都是可以调整的。包含信号名称字段、信号编号字段、数值字段以及波形窗口。这四个字段(窗口)的中间共有三条垂直的字段调整轴，将鼠标移到字段调整轴时，鼠标图标会由箭头变成一个横移的图示，此时只要按住鼠标左键不放并拖动调整轴即可改变各字段的大小。

注意！当靠左边的调整轴被移动时，右边的字段会被等距离的被拖动，但是左边的字段将不会被改变。因此调整字段大小时，可能将某些字段完全移到窗口之外，此时只要将整个 PG Editor 的窗口变大就可以看到原来看不到的字段或字段调整轴，而顺利的调整字段大小。

每个字段可以用适当调法将字段调整到最适合的位置，只要用鼠标左键在调整轴上按两下，程序就会自动将字段调整到需要显示的字符串都能完全被显示的位置。

## 3.3 名词定义

**(1) 信号名称的定义**

信号名称可以是文数字或是如底线、中括号和句点等符号，也可以是中文字。总长度不能超过 31 个字。(一个中文字相当于两个英文字)

**(2) PG1000、PG2000 信号编号的定义（面板标示由左而右）**

Pod A = 编号 0 ～ 编号 9

Pod B = 编号 10 ～ 编号 19

Pod C = 编号 20 ～ 编号 29 (PGx020 没有 Pod C)

Pod D = 编号 30 ～ 编号 39 (PGx020 没有 Pod D)

Pod E = 编号 40 ～ 编号 49 (PGx020 没有 Pod E)

Extended Pod = EVT1、EVT2、EVT3、RSI1、RSI2、RSO1、RSO2、I2CC、I2CD、CKI1、CKI2、CKO1。

**(3) PKPG 信号编号的定义（面板标示由左而右）**

一般通道 = 编号 0～ 编号 15

扩充通道 = EV1、EV2、CKI 及 CKO

EV1 = Event1

EV2 = Event2

CKI = Clock Input

CKO = Clock Output

## 3.4 光标的使用方法

**波形编辑器的三个光标系统：**

**(1)**  **『指示光标』**代表在波形区时，可用来标记波形区块或是设定波形的时间光标(红色)。标记波形区块的方法是将鼠标移至波形区，然后按下左键不放并移动鼠标，此时就会出现波形区块，直到放开鼠标的左键时，整个标记区才会被确定。当标记了某一个波形区块后，即可利用『波形功能钮』来加以改变标记区的波形，或是用鼠标的右键来剪贴复制标记区的波形。而且在『指示光标』移动的时候画面下方的状态列也会出现目前『指示光标』所在的时间，一旦开始标记时状态列也会出现标记的起始时间及结束时间，标记完成后状态列上也会出现整个标记范围的时间。

标记波形还有另一个方法，就是在波形区中按下鼠标左键两次(Double Click)。此时会出现一个标记区，这个标记区是在鼠标光标位置的信号，从鼠标位置往前及往后延伸，直到该信号值与鼠标位置的信号值不同为止。

至于如何在波形区设定『时间光标』呢？只要将鼠标移至波形区然后单击鼠标左键，就会出现一条红色的『时间光标』。当出现时间光标时，才可以设定『波形指令(PG Function)』。『时间光标』的另一个用途是当作贴上波形时的起始点，『时间光标』的中间会有一小段蓝色的部份，这是代表如果要剪贴波形时的起始通道。

**(2)**  **『拖曳光标』**是代表在波形区时，可用来移动波形的光标。使用时只要将拖曳光标移至波形区中，然后按住鼠标左键并移动鼠标，此时就可以看到波形跟着光标移动。

**(3)**  **『信道标记光标』**只会在数值字段出现，也就是当鼠标移至数值字段时，

光标形状会变成信道标记光标。当信道标记光标出现时，就可以根据当时光标位置所指的信道进行标记。如果要复合标记时只要先按下『Ctrl 键』，再来选择要标记的信道即可。

## 3.5 信号移动和快速组合信号

信号名称栏中的信号名称如果要重新排列，可以用鼠标的左键拖动要移动的信号。信号被移动时有四种模式，这四种模式可以根据拖动信号时的鼠标光标来判断。当鼠标光标变成『信号插入光标』时，就代表某一个信号将要被移动到光标的位置。如果鼠标光标变成『信号快速组合光标』就代表被拖动的信号将要与目前光标位置的信号做组合的动作。前面两种光标模式都是单一信号被移动时所产生的光标。但是如果拖动的信号不是单一信号，而是被选择的多个信号时，就会出现『多信号插入光标』及『多信号快速组合光标』。所以非单一信号被拖动时，就代表将所有被选择的信号一起移动或是被组合。

信号快速组合功能可以将单一信号组合成信号组(Bus)。使用时只要用鼠标左键去拖动某个信号或是信号组，将它拖到另一个信号或是信号组上面，就可将两信号或两信号组合并，组合后的信号名称为被加入的信号名称，信号的顺序安排为：新加入的信号为 High Bits，被加入的旧信号为 Low Bits。例如：

**(1)** 将信号 D1 拖入信号 D0 中，组合后的新信号名称为 D0，而 D0 为 Bit0，D1 为 Bit1。

**(2)** 将信号组 A[2..0]拖入信号组 D[2..0]中，新信号名称会变成 D[2..0]，信号的顺序由 MSB(Most Significant Bit)到 LSB(Least Significant Bit)依序为 A2、A1、A0、D2、D1、D0。

在多信号快速组合时，所有被选择的信号都会被一起加入到新的信号上，其加入的顺序为信号名称栏由上而下排列。但是如果要被移入的新信号也是属于被选择的信号时，那就是代表将所有被选取的信号做快速组合的动作。

 **信号插入光标**

 **信号快速组合光标**

 **多信号插入光标**

 **多信号快速组合光标**

## 3.6 信号选单及信号选择

信号选单的启动方式，是将鼠标移到信号名称栏，按下鼠标的右键就可以启动信号选单。但是启动时的鼠标位置、信号名称增修情形或是通道的选择状况都会影响启动的状态。例如信号名称栏的数据被更改或是新增时，复原的选项就可以选择。反之，如果没有任何信号被更动，复原的选项就会变成灰色而不能选择。选单的动作主要是根据鼠标所在位置的信号来处理，如果所在的位置是被选择的信号(呈蓝色)之一，那就代表选单的动作是对所有被选择的信号来处理。

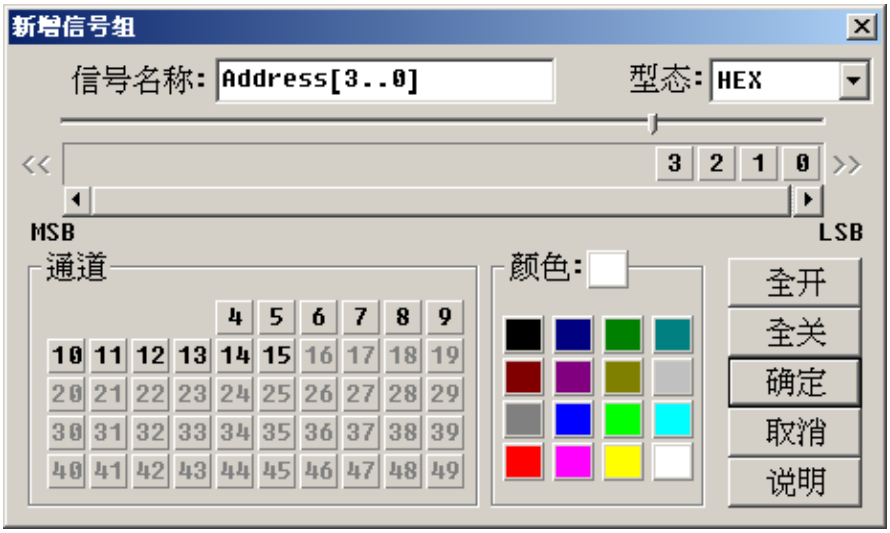
信号的选择方式是 Windows 环境的标准用法，用鼠标左键直接点选任何一个信号就会变成蓝色的信号，也代表该信号是被选择的。一旦用鼠标左键点选了某个信号时，原来被选择的信号就会被还原。如果要复合选择只要按住『Ctrl 键』不放，再用鼠标左键来选择，此时就可以重复选择。或是利用按住『Shift 键』的方式来做区域性的选择。

## 3.7 新增信号

新增信号的方法是从信号选单中，选择新增信号。此时会出现一个新增信号对话盒，这个对话盒可设定三个项目，一是信号名称，二是信号信道，三是信号颜色。设定完成后，按下确定钮，信号名称栏就会出现新增的信号。

或是『新增一组信号』来快速新增 POD 信号组。(PKPG 每个 Pod 为 8 个通道。)

## 3.8 新增信号组




选用新增信号组的功能后，画面会出现一个如上图的对话盒。这个对话盒包括了几个部份：一是信号名称、二是信号型态、三是信道组合区、四是信道区、五是信号颜色区。请依照下列步骤来新增信号组。在信号名称字段中填入信号组的名称。

**(1)** 选择信号型态，有 HEX、DEC、OCT、BIN 和 ASC 等选项。

**(2)** 从信道区中选择要组成信号组的信道号码，并从最低的位开始选择。例如要新增一个信号组，使用的信道为 5、4、3、2 等 4 个信道，5 为最高位，2 为最低位。选择时就必须依序选择 2、3、4、5。如果选择错误时，只要到信道组合区中，将错误的通道再选一次，错误的通道就会被移回通道区。而通道区中灰色的部份，是代表已经被使用的通道，所以不能被选择。

**(3)** 选择信号组的颜色。

**(4)** 按下确定钮，即完成设定。

画面右方有两个特殊的功能钮，一为全开，另一为全关。全开代表将通道区的所有可用的信道全移至信道组合区。而全关刚好相反，是将信道组合区的所有信道全移至信道区中。

在点选信道区的信道时，有一个方法可以直接插入信道组合区的某两个通道之间。方法是先将信道组合区上方的指针器先移至要插入的位置，然后再从信道区中来点选通道。

## 3.9 新增所有信号

新增所有信号会将所有的信号依照信道编号来编信号名称，加到信号名称栏(如：CH-00,CH36 等)。PGx050 会加入 CH0-CH49 等 50 个通道，PGx020 会加入 CH0-CH19 等 20 个通道，而 PKPG 只会加入 CH0-CH15 等 16 个通道。

当新增所有信号时，可能会出现一个警告对话盒，内容是『要删除其它信号吗？ 如不删除，已经存在的通道将不会被新增』，这是系统为避免信道重复，而出现的警语；如果要删除原来信号名称栏内的信号，就选择『是』的按钮；如果选择『否』，代表原来已经占用的通道将不会被新增。但是如果名称相同时，该通道也不会被新增。

## 3.10 删除信号、删除所有信号

删除信号的作法有两种。

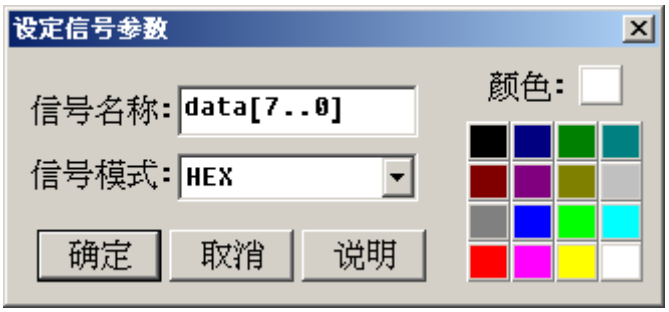
**(1)** 将选择的信号名称删除。将鼠标移到要删除的信号上或是被选择的信号名称上，按下鼠标右键，此时会出现一个信号选单，请选择『删除信号』的项目，即可将该信号删除。

**(2)** 删除所有信号，只要启动信号选单后，选择『删除所有信号』就可以将所有的信号全部删除。

## 3.11 改变信号名称

要改变信号名称方法有二，一是启动信号选单，选择『内容』来改变信号名称。另一种则是启动信号选单，然后选择『改变信号名称』，此时要改变的信号会变成一个可输入的编辑盒，即可轻易的修改信号名称，但是修改名称有几个重点，就是信号名称不能是正在使用的名称，也不能超过 31 个字，信号名称可以区分大小写，所以相同名称的大写及小写是代表不同的信号。

## 3.12 内容及所有内容




设定信号参数的启动有两种方法，一是将鼠标移到需要改变信号参数的信号名称上，并按下鼠标右键，再从信号选单中选择『内容』的选项。另一个方法是将鼠标移到需改变信号参数的信号名称上，并按鼠标左键两下(Double Click)。两种方法都会在画面上出现设定信号参数的对话盒。所以设定信号参数的用途，是在已经存在的信号需要改变设定时使用。信号参数可以设定单一信号或是信号组，设定的部份有三项，一是信号名称，二是信号模式，三是信号颜色。但是如果要设定的是单一信号时，信号模式无法设定。而信号模式主要是设定信号组的表示方式，有 HEX、DEC、OCT、BIN 和 ASC 等选项。

但是如果信号名称栏中有许多信号是被选择的时候，将鼠标移至被选取的信号名称上并按下右键，此时出现的信号选单的『内容』会变成『所有内容』，其意义是代表对所有被选取的信号名称做设定。所以在设定信号参数对话盒的信号名称是无法设定的，因为每个信号名称是不能一样的。只有信号模式及信号颜色是可以被一起改变的，如果原来被选取的信号模式和信号颜色并不是完全一样时，信号模式会变成灰色，但是仍然可以被改变，而颜色部份会出现被选取的第一个信号的颜色。然而在还没有改变设定以前所有的信号都还会维持原来的设定，一旦改变了信号模式或是信号颜色时，所有的信号设定就会被同时设定成一样的。请注意！如果只改变其中一项，只会影响所有被选取信号的该项的设定，其它的设定依然不会被改变。

## 3.13 组合信号

组合信号的方法分为快速组合及一般组合，快速组合的用法请参考『信号移动和快速组合信号』一节说明。而在一般组合的用法与『新增信号组』的用法完全一样，只是通道区里可以被选择的通道并不是全部，而是信号名称栏里面所被选择的信号名称。

## 3.14 组合信号重整

组合信号重整的用法与组合信号及新增信号组的用法一样。组合信号重整主要是对信号组的每个位重新安排顺序的功能。

## 3.15 分解信号

要将组合的信号组(Bus)分解开来的作法是将鼠标移到要分解的信号组(Bus)上，用鼠标右键来打开信号选单，选择分解信号功能来做分解。被分解的信号组会变成一些信号，这些信号会被重新命名，例如有一信号组是由四个信号所组合，名称为 XA，当分解之后，会变成四个信号，名称为 XA-3、XA-2、XA-1、XA-0。

## 3.16 波形檔

透过波形编辑器所编辑出来的波形是可以被储存成档案形式。储存的方式有两种，一个是附属档名为(.PGW)的波形档，另一个是附属档名为(.PGV)的向量檔。

PGW 波形档，它的储存内容包括了信号名称、波形、波形指令及相关参数。它的储存格式是 PG Editor 的内定格式，所以无法用其它未支持 PGW 文件的软件来读取。

PGV 向量文件，是一个文字文件的格式。所以 PGV 文件可以用任何文字编辑程序来编辑它，如 Notepad、PE2 等等软件。至于有关 PGV 的格式请参考『文字(向量)文件转成波形』一节的说明。

由于 Acute PG 所用的波形檔和 Acute LA 所用的波形档有一些差异，所以如果要使用 Acute LA 的波形档必须经过转换。转换方法请参考『Acute LA 波形檔转换』一节。

## 3.17 打印波形


**(1) 打印机设定** 

PG Editor 会呼叫 Windows 的打印机设定，您可以选择适合的打印机，并选择横印或纵印或其它打印机的特殊设定等。

**(2) 打印范围**

波形打印可以根据个人的需求选择任何的范围来打印。打印范围设定方面，可以设定成全部、选择的信道或是标记区。选择的信道是指在信号名称栏里被选择的部份，而这些被选择的信号会呈现成蓝色。而标记区是波形区中被标记的部份。

**(3) 打印比例**

**a.『打印成一张报表纸』**

横向打印范围会自动根据纸张宽度调整成适当的比例，而纵向打印范围也会根据纸张高度自动调整比例。

**b.『打印范围与纸张同宽』**

横向打印范围会自动根据纸张宽度调整成适当的比例，但纵向打印范围必须从垂直比例设定栏来设定，这种设定如果垂直打印部份会超过一张纸的话，就会被打印到下一张纸。

**c.『打印信号与纸张同高』**

纵向打印范围会根据纸张高度自动调整比例，而横向打印范围必须从水平比例设定栏来设定。

**d.『自定比例』**

横向、纵向都必须自行从水平及垂直比例栏来设定。

**(4) 打印网格线**

打印时也可以选择是否要打印网格线，要打印网格线时，可以选择网格线间的距离。

**(5) 彩色打印或灰阶打印**

黑白的打印会将波形的背景色当成白色，而其它的颜色都会设成黑色。但是彩色打印或灰阶打印时，则是保持原来的设定颜色来打印，因此要注意波形的背景色最好先设定成白色，否则打印时整张纸都会变成波形背景色，而那些细细的波形会变得很难辨识。

**(6) 输出至剪贴板**

这个功能是将波形印到剪贴板上，然后在其它的绘图软件或是有提供绘图功能的软件中，利用贴上的功能，就可以轻易的将波形剪贴到另一个软件上。

**(7) 输出至图形文件(.BMP)**

这个功能是将波形变成一个图形文件，当选择这个选项时，画面上会出现一个『Size』的按钮，这个按钮是用来设定要产生的图形文件的长与宽。『Size』的设定就相当于印表时的纸张大小设定。

## 3.18 波形放大

 波形窗口的波形显示是以屏幕的点为单位。在没有任何的放大缩小倍率时，屏幕的一点相当基本频率的倒数(时间)。

ex.假设基本频率为 1MHz，则屏幕的每一点为 1us。所以放大波形代表屏幕每一点为基本频率的倒数除以放大倍数。

ex.假设基本频率为 1MHz，放大倍数为八倍时，则屏幕的每一点为 1us/8 = 125ns。

所以波形放大时，我们会看到波形变宽。要注意的是当波形放大或缩小时，会以时间光标(红色)为参考点来做放大，一旦在没有时间光标的情况下会以波形区的最左边为参考点。所谓的参考点就是放大缩小时的中心点，在放大或是缩小时会以参考点为中心往两边缩放。而放大的倍率有下列的倍数 2、4、8、16、32 或是 64。当按放大钮按到 64 倍时，放大钮将会变成灰色，而不能再放大，直到又按了缩小钮为止。

## 3.19 波形缩小

 波形窗口的波形显示是以屏幕的点为单位。在没有任何的放大缩小倍率时，屏幕的一点相当于基本频率的倒数(时间)。

ex.假设基本频率为 1MHz，则屏幕的每一点为 1us。所以缩小波形代表屏幕每一点为基本频率的倒数乘以缩小倍数。

ex.假设基本频率为 1MHz，缩小倍数为八倍时，则屏幕的每一点为 1us * 8 = 8us。

所以波形放大时，我们会看到波形变窄。要注意的是当波形放大或缩小时，会以时间光标(红色)为参考点来做缩小。一旦在没有时间光标的情况下会以波形区的最左边为参考点。所谓的参考点就是放大缩小时的中心点，在放大或是缩小时会以参考点为中心往两边缩放。而缩小的倍率是没有固定的，只要缩到波形区可完全被显示出来时为止，此时缩小钮也将会变成灰色，而不能再缩小，直到按下放大钮为止。

## 3.20 拖曳模式和指针模式

拖曳模式和指针模式是互斥模式，所以两个模式同时间只有一个模式可以被选择。在波形编辑器中对光标系统的使用，可参考『光标的使用方法』的章节说明。

## 3.21 波形编辑方法

波形的编辑方法主要是用低电位、高电位、波形反相、编辑频率及编辑信号组的方式来修改波形，这些功能可以透过画面上方的功能钮达成。然而除了这些功能以外，也可以运用已经存在的波形来做剪下、复制及贴上等方法来修改波形。这些用法只要在波形区中按下鼠标的右键，就会出现一个选单，选单就包括了以上的功能，而且还包括了一个还原的选项，这个选项是当编辑波形时，如果发现编辑错误，可以立即按下还原选项，让波形恢复成原来的模样。

## 3.22 低电位、高电位和波形反相


低电位、高电位和波形反相都是编辑波形的基本功能，当波形区有部份波形被标记的时候，就可以用这三个功能钮来改变标记区的波形。如果按下『低电位』钮那标记区的波形通通会变成低电位(Low)的信号，如果是信号组的部份就会变成 0。而按下『高电位』钮标记区会变成高电位(High)的信号，如果是信号组会变成该信号组的最大值，例如 8 个 Bits 组合的信号组，就会变成 255(0FFh)。『波形反相』钮就是将信号反相，就是低电位的地方变成高电位，高电位的地方变成低电位。信号组的地方就会变成 1 的补码。

## 3.23 编辑频率

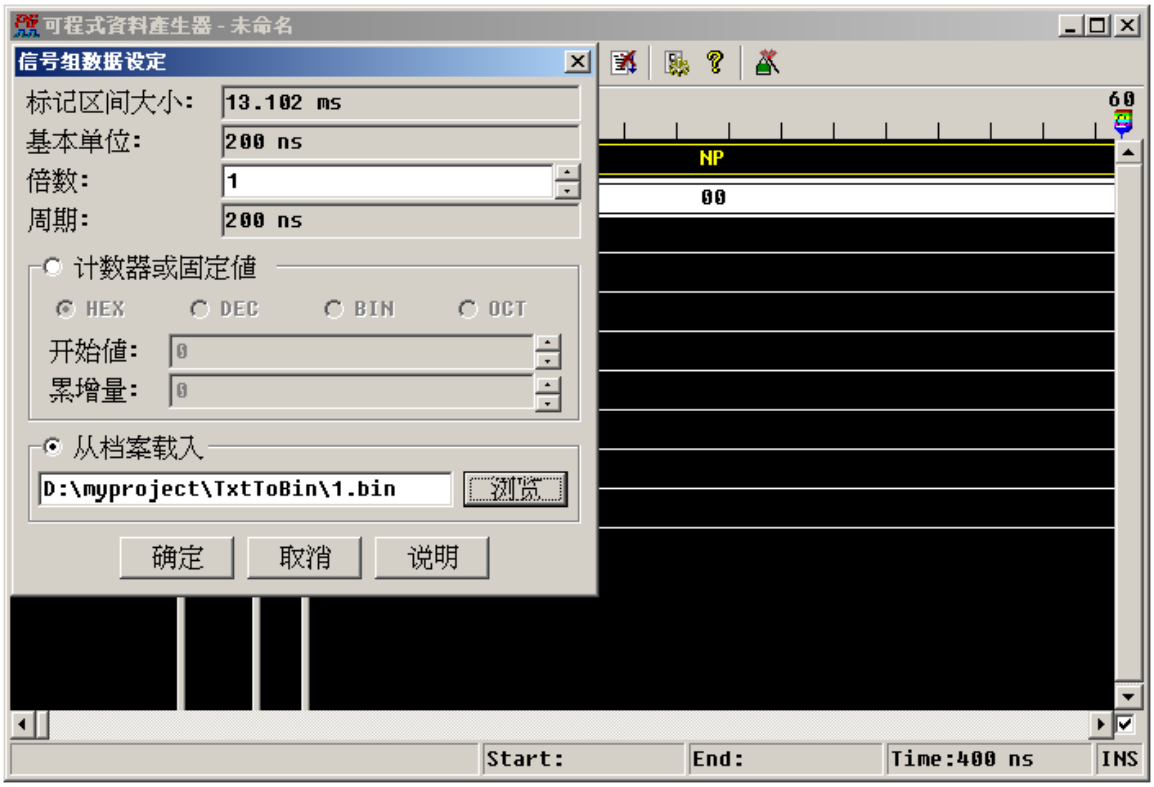

首先必须先在波形区中标记一个区域出来，此时『编辑频率』的按钮才能使用。当按下『编辑频率』钮以后，会出现如上图的对话盒。对话盒会有标记起始位置及标记结束位置，从这里可以得知要产生频率的区域在哪里。对话盒中还有一个重要的讯息就是基本单位。所谓基本单位就是目前使用的频率的倒数，例如目前所设定的频率为10MHz，那基本单位就是 1/10MHz=100ns。这也是波形区的光标每移一格的单位时间，所以称之为基本单位。而在『编辑频率』里，这个基本单位也就代表所能产生的最快频率的高电位周期或是低电位周期。同理，所能编辑出的最快频率刚好是基本频率的一半。当然要设定成最快的频率，也必须将倍数值设成 1。倍数值增加代表频率速度越慢，倍数值乘以基本单位就是频率的半个周期。而为了让使用者可以快速得知目前所设定的频率频率为何，所以在对话盒的最下方会显示目前的设定会产生多少频率的频率。对话盒中还有一个参数值可以设定，那就是开始值，所谓起始值就是从标示区开始画频率波形时，要从高电位开始呢？还是要从低电位开始。如果要从低电位开始的话，那开始值就设定成 0，反之则设定成 1。

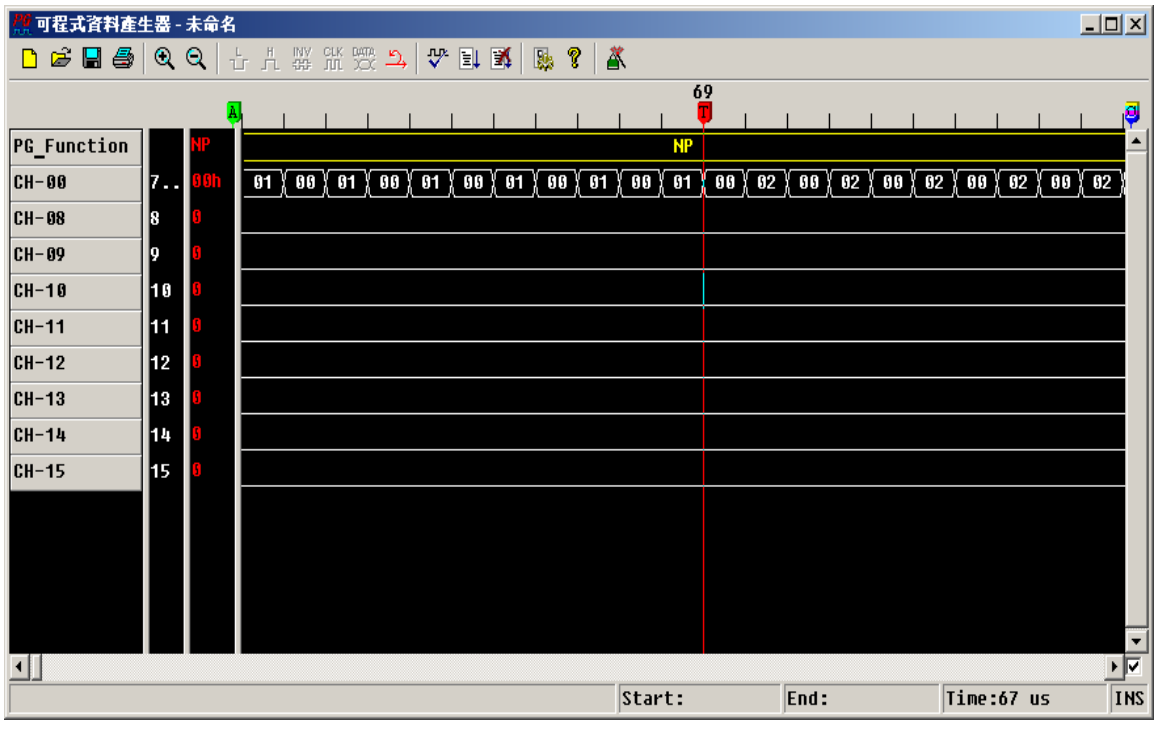
## 3.24 编辑信号组数据


先在波形区中标记一个区域出来，此时『编辑信号组数据』的按钮才能使用。也就是说产生的信号会出现在标记区域内，不会影响非标记区域的信号内容。当按下『编辑信号组数据』钮以后，会出现如上图的对话盒。对话盒中还有一个重要的讯息就是基本单位。所谓基本单位就是目前使用的频率的倒数，例如目前所设定的频率为 10MHz，那基本单位就是 1/10MHz=100ns。这也是波形区的光标每移一格的时间，所以称之为基本单位。还有另一个字段是『周期』，周期就是基本单位乘以倍数的结果，也就是编辑数据时每一笔数据的间隔。

信号组的数据编辑方式有三种：第一是计数器，即是在标记区中填入一组累进的资料。而这个计数器可以设定开始值和累进值，如果开始值为 0，累进值为 3，那就会产生一组 0、3、6・・・的数列。如果开始值和累进值要用其它的进位制来描述，只要选择 HEX、DEC、BIN、OCT 等四个进位选项。第二种编辑数据组的方式是固定值，其实固定值与计数器是一样的用法，只是累进值的设定为 0。因为累进值为 0，所以每一笔数据都会是开始值。

第三种编辑信号组的方式是从档案加载。如果要从档案加载，必须先选择『从档案加载』的选项，并输入一个文件名称。加载的方式是以每个 Byte 为基本单位，也就是从档案的第一个 Byte 开始，每个周期填入一个 Byte 的数据。如果档案的长度太短而不够填入标记区时，程序会自动从档案的开头再读取一次，直到标记区的数据被填满为止。



## 3.25 编辑指令

 在波形区中有一个信号组叫做『PG Function』，它是一个不能被删除的信号组，这个信号组就是波形指令信号组。所谓波形指令顾名思义就是用来控制波形动作的指令，整个波形指令总共有 7 个，包括 NP (No Operation)、JP (Jump)、LP (Loop)、BE(Branch if Event)、LC (Loop Count)、SE (Set Event)和 WE (Wait Event)，如下图的指令列表。

**指令的介绍：**

| 指令 | 指令全名 | 说明 | 周期 |
|---|---|---|---|
| **NP** | No Operation | 没有动作 | 1 |
| **JP** | Jump | 跳跃到设定点继续执行 | 3 |
| **LP** | Loop | 将循环数减一，如果循环数不为 0 就跳跃到设定点，如果为 0 就跳到下一个位置。 | 3 |
| **BE** | Branch if Event | 如果以设定的 Event 信号出现就跳跃到设定点，否则就会跳跃到下一个位置。 | 3 |
| **LC** | Loop Count | 设定循环数，设定值为 2～65536。 | 2 |
| **SE** | Set Event | 选择触发的事件。 | 1 |
| **WE** | Wait Event | 等待事件被启动。 | 1 |
| **OE** | Output Enable | 输出允许(高阻抗模式)注 1 | 3 |

**注 1：OE 指令只在 PKPG 有用。**

**(1)** **NP (No Operation)**在波形执行时，并不会有任何控制流程的改变或是任何内部缓存器改变。用 CPU 的观点来看，那 NP 指令的动作就是 New Address = Address + 1，也就是指针移到下一个位置，其它不做任何改变。

**(2)** **JP (Jump)**是一个改变控制流程的指令。例如『JP 35』，这个指令的动作就是无条件跳至新地址 35(相当于 New Address=35)。

**(3)** **LP (Loop)**和 JP 指令很类似，其最大差异就是 JP 是一个无条件跳跃指令，而 LP 是一个有条件跳跃，它必须与『LC』指令配合使用。Acute PG 有一个内部

缓存器称为循环缓存器(Counter)，这个缓存器就是用来记录循环数目的缓存器。这个缓存器是用 LC (Loop Count)指令来设定，例如『LC 32』就是将循环缓存器设定成 32。循环缓存器可以设定的值为 2～65536，所以不能设定成 0 与 1，这种设定方式与一般的 CPU 或 MCU 的用法有些许的差异，需注意。设定好循环以后就可以使用 LP 指令了，而每次执行 LP 指令，循环缓存器的值就会减 1，而跳跃的新指针就会参考 LP 指令的设定值来决定。

例如指令『LC 32』、『LP 16』，的整个动作步骤如下：

**a.** 将循环缓存器 32 减 1。

**b.** 检查循环缓存器是否为 0。

**c.** 如果循环缓存器为 0，New Address = Next Address。

**d.** 如果循环不为 0，New Address = 16。

有一点要注意的是如果当循环缓存器已经被减成 0，又遇上 LP 指令时，这时候循环缓存器是不会再被减成-1，而是会产生特殊情况。所以循环缓存器被减成 0 以后，LP 指令的工作是不可预期的，所以要使用 LP 指令之前请注意是否已经设定 LC 指令了。

**(4)** **BE (Branch If Event)**指令和 LP 指令又有点类似，因为他们都是一个有条件跳跃的指令。LP 的跳跃条件是循环缓存器，而 BE 的跳跃条件是 Event Bit。当波形执行到 BE 指令时，Acute PG 会立即判断 Event Bit，如果 Event Bit 为 1 就会跳到 BE 所设定的新地址，Event Bit 为 0 的话，则继续执行下一个地址。

**(5)** **SE (Set Event)**是一个选择事件的指令，Acute PG 有 3 个外部事件通道(Event_1、Event_2、Event_3)注 1 及 1 个内部键盘事件(Keyboard Event)，所以总共有 4 个事件来源。Acute PG 将这 4 个事件来源编辑成 16 种事件型态以兹利用，而这个事件型态会被存到一个 Acute PG 的内部缓存器，称之为事件缓存器(Event Register)。这些事件型态包括：

- Keyboard Event
- Event_1

- Event_2
- Event_3 注 2
- Event_1 or Event_2
- Event_1 or Event_3 注 2
- Event_2 or Event_3 注 2
- Event_1 or Event_2 or Event_3 注 2

**注 1：PKPG 只有两个外部事件通道，所以共有 3 个事件来源，编辑成 8 种事件型态。**

**注 2：PKPG 不支持这种事件组合。**

另外 8 种事件型态就是这 8 种型态的反相。例如事件缓存器的设定值为上述 8 种型态之一，此时不管从外部或是内部的事件通道都会被 Acute PG 所监控，Acute PG 会随时比较目前的事件通道的状态是否与事件缓存器的事件型态相同。如果相同就会将 Acute-PG 内部的旗标缓存器(Flag Register)的 Event Bit 设成 1 (True)，如果不同就会将 Event Bit 设成 0 (False)。然而反相型态就是当事件信道的状态与事件缓存器的事件型态相同时，Acute PG 会将 Event Bit 设成 0，不同时则会将 Event Bit 设成 1。

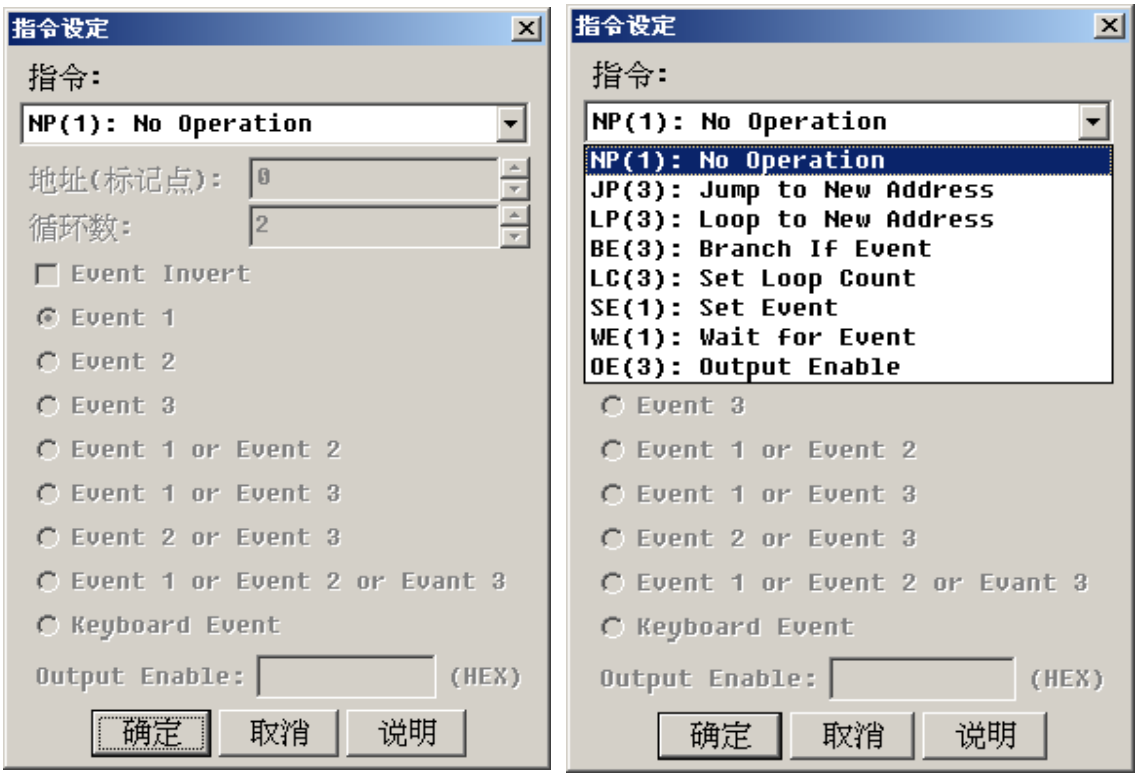
而运用这个 Event Bit 的指令有两个，一是 WE (Wait Event)另一是 BE (Branch If Event)。WE 指令是当波形执行到这个指令时，整个 Acute PG 会因此而暂停，而波形会停留在 WE 指令当时的波形，一直到 Event Bit 为 1 时，才会再继续往下执行。

**(5) OE (Output Enable)**指令可以设定每个信道的输出为高阻抗或是信号输出。OE 指令的参数是一组 4 位数的 16 进位数字，这个值代表 16 个通道的状态。例如 FFF0 代表信道 0 到信道 3 为高阻抗，信道 4 到信道 15 则为正常信号输出。

**指令的编辑方法：**

指令在编辑之前，必须先将光标模式切成指针模式，然后将光标移至波形区需要编

辑指令的位置上，然后按下鼠标左键，这时候会出现一条红色的时间光标。这时编辑指令的按钮就会从灰色变成彩色，也代表编辑指令的功能是可以使用的。在按下编辑指令钮时，就是对时间光标的位置做编辑指令的动作，此时也会出现如下图的对话盒。




左图是原始编辑指令对话盒的外观，右图则是显示指令列表的外观。使用时，首先先选择指令列表里的指令，程序会根据选择的指令，打开相关的控件目。例如指令为 LC 时，循环数的项目才会被启动，也才能改变循环值。设定好指令的项目以后，并按下确定钮，指令就会出现在指令信号组。

如果要将整区的指令清除成 NP (No Operation)时，用法比较特殊。将光标移至波形区的指令信号组上，使用标记区块的方式，将要清除的指令信号组范围标记起来，此时编辑指令钮亦会从灰色变成彩色，但是按下编辑指令钮时，会出现一个对话盒，内容是询问是否要将指令区清除，只要按下确定的按钮就会被清除了。

详细介绍，请参阅：

PG_Function 指令使用方法

## 3.26 波形检查

 由于指令的运用有一些限制，所以在波形输出之前最好先做波形检查，否则输出的波形可能会出现预期以外的情况。波形的检查主要是检查一个指令与另一个指令之间的距离，以及跳跃指令的跳跃点与其它指令的距离。以下是检查方式之详述(指令集请参考『编辑指令』一节)。

下列指令的前后 3 个指令周期(Clocks)之内不能有其它的指令，JP (Jump)、LP (Loop)和 BE (Branch If Event)等 3 个跳跃指令。例如 JP 指令出现在 12 的位置，因为 JP 用了 3 个指令周期，所以 12、13 和 14 都属于 JP 指令周期，而其前后 3 个指令周期就是 9、10、11 和 15、16、17。也就是说在前述的那些位置上不能有其它的指令，否则波形输出时将会出现不可预期的结果。

上述的 3 个跳跃指令的跳跃点前后 3 个周期亦是不能有其它的指令。例如 LP 30 这个指令就是当波形执行到这个指令时，会跳跃到 30 这个周期继续执行，而 30 这个点就是跳跃点。同理，27、28、29 和 31、32、33 就是它的前后 3 个周期，在这区间一样不能有其它的指令。同时跳跃点也不能超出波形长度的设定范围。

其它如 LC (Loop Count)、SE (Set Event)和 WE (Wait Event)等指令，是可以连在一起使用，没有间隔的要求。

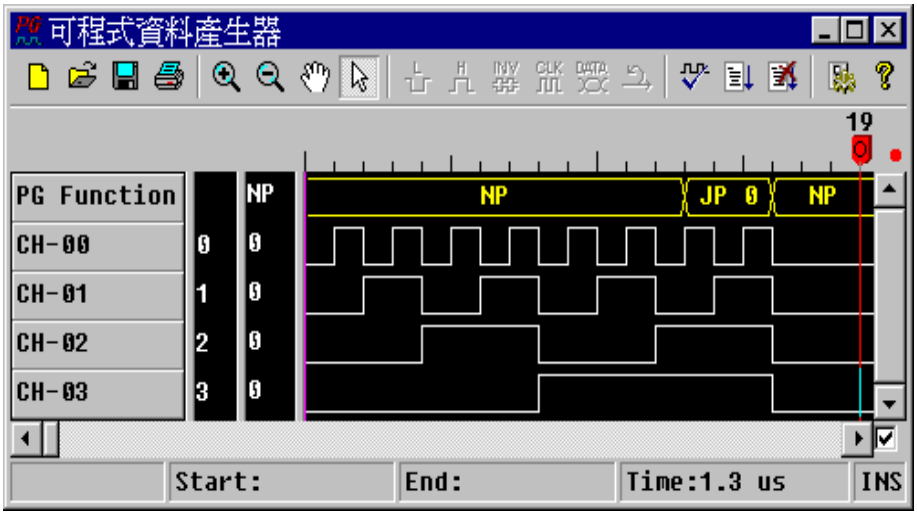
指令检查时还有一项重要的检查，那就是检查是否有不明的指令，不明的指令在波形区的画面上会出现『XX』的字符串。

## 3.27 波形输出与停止输出

当检查波形如果一切正常时，就可以按下『波形输出』钮，此时波形就会从 Acute PG 的输出端将波形输出。由于系统必须将编辑好的波形先传送到 Acute PG 的主机里面，再通知 Acute PG 开始执行波形数据，所以当按下『波形输出』钮时并不是立即输出，大概会延迟 0.5 秒的时间，所以使用时可以运用一些技巧。首先，先将标的物与 Acute- PG 连接，开始编辑波形，波形的最前端最好空出一小段时间，以便加入两个指令，一个是 Set Keyboard Event，另一个是 Wait Event。加上这两个指令后，当按下『波形输出』时，Acute PG 的执行会停留再 Wait Event 上。此时可以操作您的标的物或是电路，以达到开始运作的状态，这时再按下『事件热键』就可以继续输出 Wait Event 之后的波形。

开始输出波形时，状态列会出现如同波形输出的样子，此时代表波形正在输出。如果您想要停止波形继续输出，只要单击『停止输出』钮，就可以让波形输出停下来。然而在按下『波形输出』之前或是在按下『停止输出』之后，Acute PG 的输出端都是呈现高阻抗状态。所以您的电路如果会因为高阻抗而造成工作不正常，此时就必须将上述连接标的物至 Acute PG 的时间，改成按下『输出波形』和按下『事件热键』之间来做连接即可。

下面是一个 4Bits Counter 的例子：



## 3.28 环境参数设定


环境设定主要项目有 2 个，一是基本频率的设定、二是事件热键的设定。

调整基本频率的方法是：先选择『频率形式』，可使用内部频率、外部频率或是混合频率，然后调整频率值即可。

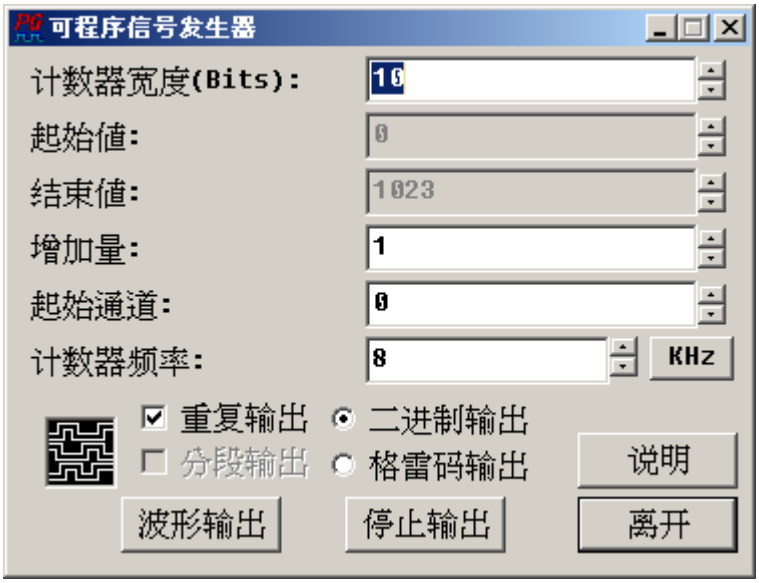
当设定好频率值后，如果按下确定钮时会出现下方的对话盒时，就代表设定的频率值是 Acute PG 所无法产生的频率。此时对话盒会问是否要使用最接近设定值的频率，如果回答『是』的话，Acute PG 就会自动选择一个最接近的频率当作基准频率。


事件热键设定：选择哪一个按键当作『事件键』，有 Space、Return 可供选择。而所谓的『事件键』就是当波形正在输出的状态下，按下这个热键就会产生一个事件，并传送到 Acute PG。

# 4. 同步计数器

同步计数器可以很快制作出一个计数器的信号，可以从任何一个值计数到另一个值，而且让计数器重复不断的计数。由于 Acute PG 的内存深度有限，如果计数的范围超过 Acute PG 的内存深度时，亦可以使用分段输出的方式，来输出计数器的信号。计数器的位数及频率是可以依据使用的需要来做调整。

## 同步计数器使用说明：




### (1) 计数器宽度

计数器的宽度最大设可定范围会依据使用的机型而有所不同。如 PGx020 最多宽度只能设成 20，PGx050 可设成 50，而 PKPG 只能设成 16。当进入同步计数器功能时，起始值和结束值会依计数器宽度自动改变起始值固定为 0，结束值为计数器宽度所能计数的最大值。例如计数器宽度设成 5 时，起始值为 0，结束值为 31($2^5$-1)。直到改变了起始值或结束值后，它们就不会依计数器宽度来改变。

### (2) 起始值

起始值为计数器开始计数的值，如果要更改起始值只要将光标在起始值右边的上下旋转器轻单击就可改变。当起始值大于计数器宽度所能产生的最大值时，起始值自动会被视为该值除以计数器宽度所能产生的最大值加 1 的余数。

例如起始值为 45，而计数器的宽到为 5 时，起始值会被视为 45-$2^5$=45-32=13。

**(3) 结束值**

结束值的设定方法与起始值相同。但是结束值是指当计数器计数到结束值时才算结束，所以当增加量不等于 1 时，计数器可能会超过结束值，此时计数器会根据起始值与增加值来计算下一个计数值。例如，当起始值为 0、结束值为 7、增加值为 3 时，整个计数器所产生的值为 0、3、6、1、4、7。

**(4) 增加量**

增加量的设定方法与起始值和结束值的设定方法相同。增加量的意义是指每一计数的累进值。增加量亦可以设定成负数，一旦设成负数就代表是一个倒数计数器。

**(5) 起始通道**

起始信道为输出信道的最低位，例如起始信道为 5、计数器的宽度为 6，则输出通道为 10、9、8、7、6、5。信道 10 为最高位，信道 5 为最低位。

**(6) 计数器频率**

计数器的频率是每计数一次的频率。如果频率为 10MHz 就代表每 100ns 计数一次。可以使用的频率请参考频率设定一节的说明。

**(7) 二进制码输出/格雷码输出**

可以选择信号输出方式为二进制码(Binary code)或是格雷码(Gray code)的输出方式。

使用同步计数器可以将一串连续的数值输出，使用『重复输出』的功能让这些数值不断的重复。如果要输出的数值超出 Acute PG 的内存深度时，也可以使用『分段输出』的功能来达到输出数值的目的。因为 Acute PG 有 64K 的记忆深度，也就是说每个通道的深度为 64K。但是 64K 的记忆深度中有 12 个是系统控制用，系统控制用的的长度是根据不同的功能会有所不同，在同步计数器中所需要的长度为 12 个。所以整个同步计数器所能用的记忆深度为 64K-12=65536-12=65524。

因此当同步计数器设定的值会超过 65524 时，程序会询问要不要分段输出，如果不分段输出的话会有两种情形。第一、如果没有设定重复输出时，整个数据只能输出到内存的底部就会被停止。第二、如果有设定重复输出时程序将会出现错误讯息。

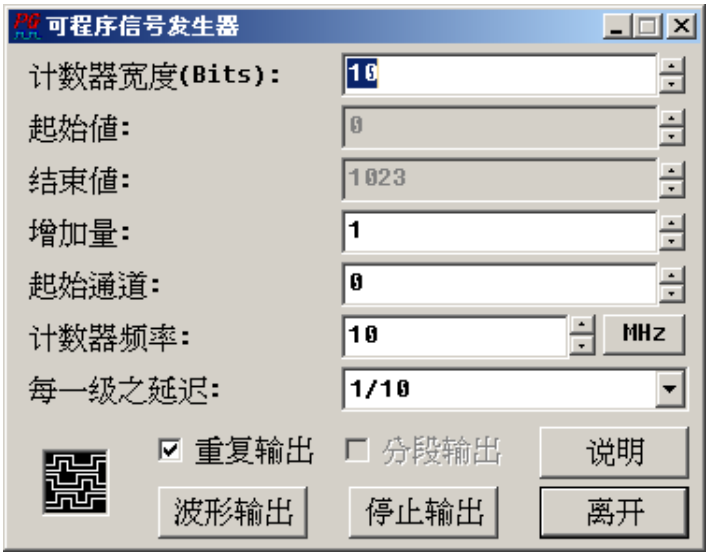
由于 Acute PG 的记忆深度为 64K，所以超过 64K 的数据在使用『分段输出』时，系统的处理方式分为下列几个步骤：第一、将第一部分的数据写入 Acute PG 的内存中。第二、输出数据直到内存的底部。第三、将输出信号暂时停止。第四、再将下一组数据写入 Acute PG 的内存中。重复第二至第四步骤直到资料结束为止。由于输出信号会被暂停来输入下一组的数据，所以整个输出信号将会有所延迟。也就是说当您要的是连续而且中途不能停顿的信号，分段功能将是不能使用的。

当设定结束后，只要按下『波形输出』就可以开始输出波形，此时画面左下角的输出状态图示上会出现波形流动的图示。如要停止波形输出只要按下『停止输出』即可。

# 5. 异步计数器

所谓异步计数器就是一个涟波计数器。Acute PG 能轻易的制作出一个涟波计数器，可以从任何一个值计数到任何一个值，而且可以让计数器重复不断的计数。由于 Acute-PG 的内存深度有限，如果计数的范围超过 Acute PG 的内存深度时，亦可以使用分段输出的方式，来输出计数器的信号。计数器的位数及频率是可以依据使用的需要来做调整。

## 异步计数器使用说明：




### (1) 计数器宽度

计数器的宽度最大可定范围会依据使用的机型而有所不同。如 PGx020 最多宽度只能设成 20，PGx050 可设成 50，而 PKPG 只能设成 16。当进入同步计数器功能时，起始值和结束值会依计数器宽度自动改变起始值固定为 0，结束值为计数器宽度所能计数的最大值。例如计数器宽度设成 5 时，起始值为 0，结束值为 31($2^5$-1)。直到改变了起始值或结束值后，它们就不会依计数器宽度来改变。

### (2) 起始值

起始值为计数器开始计数的值，如果要更改起始值只要将光标在起始值右边的上下旋转器轻单击就可改变。当起始值大于计数器宽度所能产生的最大值

时，起始值自动会被视为该值除以计数器宽度所能产生的最大值加 1 的余数。例如起始值为 45，而计数器的宽到为 5 时，起始值会被视为 45-$2^5$=45-32=13。

**(3) 结束值**

结束值的设定方法与起始值相同。但是结束值是指当计数器计数到结束值时才算结束，所以当增加量不等于 1 时，计数器可能会超过结束值，此时计数器会根据起始值与增加值来计算下一个计数值。例如，当起始值为 0、结束值为 7、增加值为 3 时，整个计数器所产生的值为 0、3、6、1、4、7。

**(4) 增加量**

增加量的设定方法与起始值和结束值的设定方法相同。增加量的意义是指每一计数的累进值。增加量亦可以设定成负数，一旦设成负数就代表是一个倒数计数器。

**(5) 起始通道**

起始信道为输出信道的最低位，例如起始信道为 5，计数器的宽度为 6，则输出通道为 10、9、8、7、6、5。信道 10 为最高位，信道 5 为最低位。

**(6) 计数器频率**

计数器的频率是每计数一次的频率。如果频率为 10MHz 就代表每 100ns 计数一次。但是在异步计数器的频率设定上会比同步计数器低，请参考以下的说明。

**(7) 每一级之延迟**

每一级的延迟是每一步计数的时间乘以延迟设定。例如，计数器频率为 10MHz，则每一步计数的时间为 100ns，而每一级延迟的设定为 1/10，就代表每一级的延迟为 10ns。

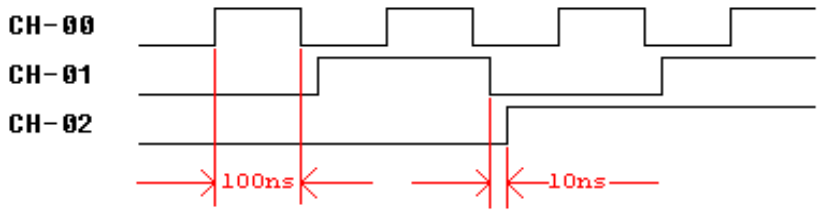
使用异步计数器可以将一串连续的数值输出，使用『重复输出』的功能让要这些数值不断的重复。在异步计数器中是无法使用『分段输出』的功能。因为 Acute PG 有 64K 的记忆深度，也就是说每个通道的深度为 64K。但是 64K 的记忆深度中

有 30 个是系统控制用，系统控制用的的长度是根据不同的功能会有所不同，在异步计数器中所需要的长度为 30 个，而且在使用重复输出时内存深度只有一半可以使用。所以异步计数器所能用的记忆深度分为两种情况。一为在没有设定『重复输出』时，64K-30=65536-30=65506。另一为在有设定『重复输出』时，(64K-30)/2=(65536-30)/2=32753。异步计数器设定的值不能超过可用值，如果超过程序会出现错误讯息。

当设定结束后，只要按下『波形输出』就可以开始输出波形，此时画面左下角的输出状态图示上会出现波形流动的图示。如要停止波形输出只要按下『停止输出』即可。

下面是异步计数器的实例：

计数器宽度=3、起始值=0、结束值=7、增加值=1、频率=10MHz、延迟=1/10、起始通道=0，结果为：



# 6. 并列信号发生器

并列信号发生器可以产生 SPP Mode 的打印机信号，只要将 Acute PG 依设定中的脚位定义接上打印机，就可以将文字数据或是图形数据从打印机打印出来。

## 并列信号发生器使用方法：


并列信号发生器的用法有二。一是，输入一个打印机可以识别的档案，这个档案可以是纯文本文件或是含有打印机句柄的档案。二是，直接在文字编辑区里输入纯文本数据。如果输入档案的是一个纯文本档案，可以按下载入钮，将档案加载编辑区中做修改。数据的尾端也可以加入跳行(LINE FEED)或跳页(FORM FEED)的指令。

数据输出以前必需先设定信道，并且连接到打印机上，记得要接上地线，脚位的连接请参考附注说明。要设定的脚位有 Data[7..0]、nStrobe、nSelectIn、nAutoFeed 和 Busy 等接脚。除了 Busy 接脚是输入信号以外，其他皆为输出信号。所以 Busy 可以设定的通道为 Event_1、Event_2 或是 Event_3，这三个 Event 信道是 Acute PG 的输入信道，他可以通知 Acute PG 是否要继续输出数据。所以并列信号发生器会将产生的信号输出至 Data[7..0]、nStrobe、nSelectIn 和 nAutoFeed 等脚位，但是当打印机送出 Busy 信号时，PG 就会停止送出数据直到 Busy 信号结束有后，才会继续输出数据。

当您数据因太大而超过 Acute PG 的记忆深度时，程序会主动通知是否要分段输出。如果要打印多份时亦可选择『重复输出』的选项。

当设定结束后，只要按下『波形输出』就可以开始输出波形，此时画面左下角的输出状态图示上会出现波形流动的图示。如要停止波形输出只要按下『停止输出』即可。

**附注：(打印机脚位表)**

Data [7.0] = Pin 9.2

nStrobe = Pin 1

NSelectIn = Pin 17

NAutoFeed = Pin 14

Busy = Pin 11

Ground = Pin 25.18

# 7. 串行信号发生器

串行信号发生器可以产生 TTL 准位的 RS232 的信号，而且这个 RS232 信号是标准的信号准位，所以 PG1000/PG2000 不需要额外增加驱动零件就可以直接接到 RS232 的输入埠上 (若要将 PKPG 的 RS232 输出信号与标准的 RS232 连接的话，必须要加一个驱动器〝Drive IC〞，将 TTL 准位变成正负 12 伏特的信号。)

## 串行信号发生器使用方法：


**(1) 传输速率(Baud Rate)**

可以从 110 一直到 256000 等所有标准的速率。

**(2) 同位检查(Parity)**

None, Odd, Even, Mark, Space。

**(3) 数据位(Data Bits)**

4~7

**(4) 停止位(Stop Bits)**

1、1.5、2。

**(5) 输出埠(Output Port)**

Acute-PG 的 Extended Pod 包含了两个 RS232 的输出埠，输出埠的设定可以任选其中一个输出埠或是两个输出埠同时输出，也可以让两个输出端口产生差动信号。(只有 PG1000 和 PG2000 可用)

**(6) 同步输出通道**

PG1000/PG2000 选择任何一个信道来输出未经驱动成 RS232 标准电位的 5V 或 3.3V 信号(根据输出 Pod 而定)。

**(7) 数据格式**

串行信号发生器可以解释 5 种数据格式，分别是 BIN、OCT、DEC、HEX、ASC，这些格式都是利用文本文件的模式来表示。BIN 是二进制的格式，OCT 是八进制的格式，DEC 是十进制的格式，HEX 是十六进制格式。以上 4 种格式必需是一行一笔数据，也就是每一笔数据都必需有 Carriage Return 和 New Line。例如，BIN 格式要输出 1、2、3，数据内容如下：

**00000001**

**00000010**

**00000011**

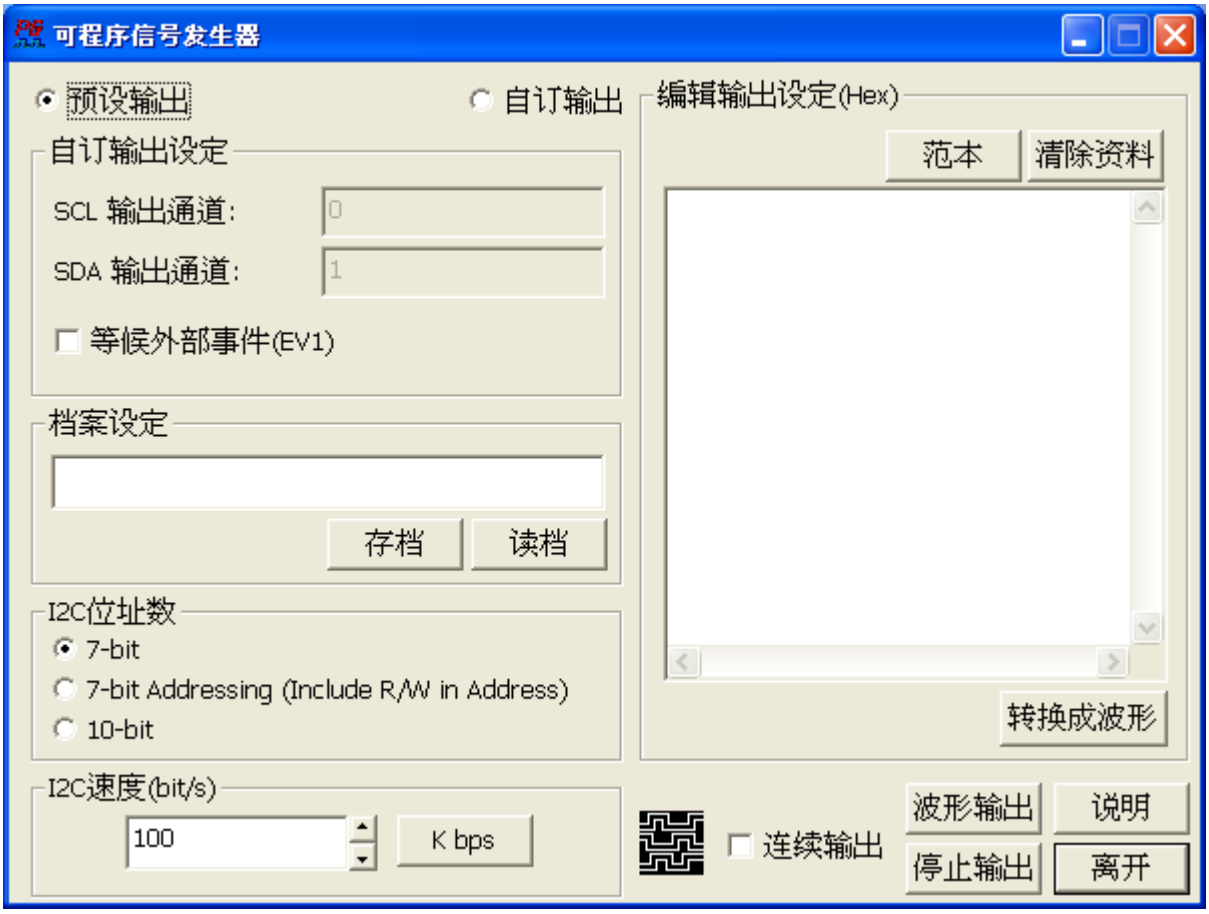
如果格式设定为 ASC 模式，就不须加入 Carriage Return 和 New Line 等字符，所以任何字符都会被输出。

串行信号发生器的数据的输入方式有二。一是，输入一个数据文件。二是，直接在文字编辑区里输入纯文本数据。数据文件如果要修改，可以按下加载钮，将档案加载编辑区中做修改。

由于数据位、停止位及同位位的设定关系，所以使用 Acute PG 的内存情况都不一样。当您数据因太大而超过 Acute PG 的记忆深度时，程序会主动通知是否要分段输出。亦可选择『重复输出』的选项让数据不断的重复输出。

当设定结束后，只要按下『波形输出』就可以开始输出波形，此时画面左下角的输出状态图示上会出现波形流动的图示。如要停止波形输出只要按下『停止输出』即可。

# 8. I2C 信号发生器




I²C 是一种串行通讯的标准，是由 Philips 公司所制定的一种通讯规格。他只用两条信号线，一条是频率线(SCL)和一条资料线(SDA)所构成。其传输的方式是双向的，数据格式分为 7Bits，8Bits 和 10Bits 三种。

I²C 信号发生器的输出模式分为[默认输出]与[自定义输出]，如果您选用的是[自定义输出]，则您可以选择 SDA 与 SCL 的输出通道，但是所输出的波形将不具有 OPEN COLLECTOR 的特性；如果您选用的模式为[默认输出]，并且使用本公司 PG2020, PG2050 以及专门为 I²C 信号发生器所设计的输出脚(PG2020,PG2050 的 Extended Pod 的 I2CC, I2CD 脚)则将具有 OPEN COLLECTOR 的特性，但是如果您使用的是 PKPG 系列(PKPG2016, PKPG2116, PKPG2116+)，只能选择[自定义输出]，而且将不具有 OPEN COLLECTOR 的特性。

I²C 信号发生器可以设定要输出的数据，您可以自行编辑您要输出的资料或是加载文本文件。格式说明如下：

**********

7 bits Address:

Aw=12;

D=10,20,30;

Ar=3f;

D=0;

Aw=46;

D=21,3a;

**********

8 bits Address:

A=25;

D=0,0,0;

A=7f;

D=0;

A=8C;

D=21,3a;

**********

10 bits Address:

Aw=12c;

D=10,20,30;

Aw=23f;

D=4c;

Ar=18a;

D=0,0;

**********

A 表示 I²C 地址(address)，D 表示 I²C 数据(data)。在 I²C 地址数 7 bits 和 10 bits 的命令格式相同；Aw=12; 表示写入地址(write address) 12h, Ar=3f; 表示读取地址(read address) 3fh。而 I²C 地址数 8 bits 和上述 7 bits 的差异主要是将 r/w bits 纳入 address, A=25; 即为 Ar=12,上述的数值皆为十六进制。

每行命令行以[;]作为结尾，若没有使用[;]作为结尾，程序将会忽略该行指令，而在数据 D 中，每一笔数据以[,]间隔，如没有使用[,]作为间隔则会忽略该笔数据。对于 I²C 编辑区的内容，程序会做简易的格式检查，并提示讯息告知编辑的格式是否有错误。

**(1) 自定义输出**

**a. SCL 输出通道**

输入要输出 I2C 频率通道。

**b. SDA 输出通道**

输入要输出 I2C 数据信道。

**c. 等候外部事件(EV1)**

若勾选该项，EV1 脚若有脉波输入，则数据产生器才会输出 I2C 信号。

**(2) 档案设定**

**a. 存档**

将编辑区中的内容储存成文本文件(.txt)

**b. 读档**

将储存的档案读回，并将内容显示于编辑区

**(3) $I^2C$ 地址数**

选择 $I^2C$ 地址(Address)模式

**(4) $I^2C$ 速度(bit/s)**

输入 $I^2C$ 频率

**(5) 编辑输出设定(Hex)**

**a. 范本**

只要按下模板，会根据当 $I^2C$ 地址数，在编辑区中显示每个 $I^2C$ 地址数的命令范例

**b. 清除资料**

清除编辑区的内容

**c. 转换成波形**

会将所编辑 $I^2C$ 信号命令，转换成波形，显示在波形编辑器上，可以让使用者在波形的状态下修改 $I^2C$ 信号。

**d. 连续输出**

可以连续输出 $I^2C$ 信号，不过有一点要注意，当要输出的 $I^2C$ 信号大于 PG 记忆深度，连续输出的功能会 disable

**e. 波形输出和停止输出**

输出 $I^2C$ 信号和停止输出 $I^2C$ 信号

**(6) 说明**

呼叫说明文件

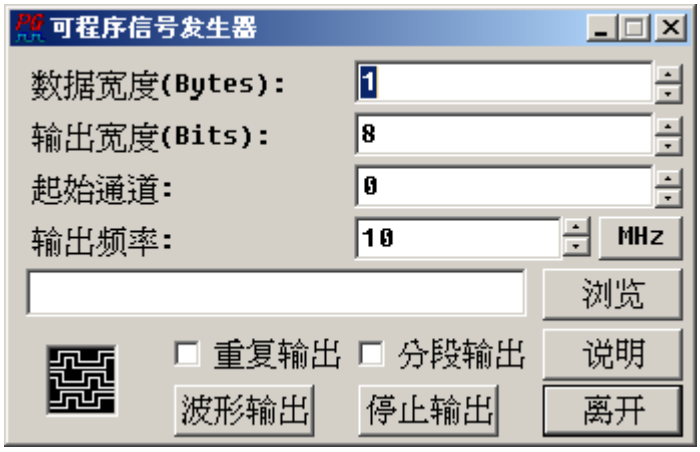
**(7) 离开**

离开 $I^2C$ 信号发生器，回到 PG 主画面

# 9. 资料文件转成波形

任何数据文件可以透过转换变成波形，再藉由 Acute PG 来输出。设定方式如下：

**数据文件转成波形的使用方法：**




**(1) 数据宽度(Bytes)**

数据宽度是以 Byte 为计算单位，和档案的储存单位是一样的。所谓数据宽度就是要将整个数据文件以每次多少个 Bytes 来输出。如果档案长度为 1024Bytes，而数据宽度设定成 2Bytes，那整个数据文件就会被分成 512 组来输出。

**(2) 输出宽度(Bits)**

输出宽度的设定值可以从 1 设到最大输出宽度，最大输出宽度是指数据宽度换算成 Bits 后的值。也就是把数据宽度乘以 8。

**(3) 起始通道**

起始信道为输出信道的最低位，例如起始信道为 5、输出宽度为 6 时，则输出通道为 10,9,8,7,6,5 等 6 个通道。信道 10 为最高位，信道 5 为最低位。

**(4) 输出频率**

输出频率是每输出一笔数据的频率。如果频率为 10MHz 就代表每 100ns 输出一笔数据。可以使用的频率请参考频率设定一节的说明。

当您数据因太大而超过 Acute PG 的记忆深度时 ,程序会主动通知是否要分段输出。如果要将数据不断的重复输出时亦可选择『重复输出』的选项。

当设定结束后，只要按下『波形输出』就可以开始输出波形，此时画面右上角的输出状态图示上会出现波形流动的图示。如要停止波形输出只要按下『停止输出』即可。

# 10. 文字(向量)文件转成波形

 文字(向量)文件是 PG-Editor 专用的向量档，这个档案的附属档名为 PGV。所以您可以用任何的文字编辑程序先将向量文件写好，然后在用这个转换程序将向量文件转成波形。

## PGV (PG Vector File)数据格式说明：

PGV 有下列几个指令(Keyword)，**INPUTS**、**UNIT**、**INTERVAL**、**FREQUENCY**、**PATTERN**、**ASSIGN**、**RADIX**。

**(1) INPUTS**

设定信号名称。每一个名称用空白分隔开来，名称可以为文字或是数字如果为 Bus(Group)时，可以用中括号来指定。例如某个 Bus 为 4 个通道所组成那就可以用 A[3..0]来代替，A[3..0]就代表 A3、A2、A1、A0 等 4 个信号。**注意！『PG_Function』**是一个保留字，如果在 **INPUTS** 指令的定义中有**『PG_Function』**，就代表 PATTERN 中会包含 PG 的专用指令。而且**『PG_Function』**也不能配合 **ASSIGN** 指令使用。

**(2) ASSIGN**

用来指定 **INPUTS** 指令所定义的信号名称之信道值。指定的方式如下：

INPUTS A [3..0]; ASSIGN A [3..0]=8..11;就表示 A3=CH8, A2=CH9, A1=CH10, A0=CH11。INPUTS Reset;ASSIGN Reset=32;就表示 Reset=CH32

**注意！未被指定的信号将会根据信道顺序(由小到大)来安排。ASSIGN 不能指定相同的通道或是超出 Acute PG 所能容许的最大通道。**

**(3) UNIT**

设定 **PATTERN** 区域的单位。可以设定成 **ns**、**us** 和 **ms** 等三种单位。这个设定是配合有 Time Stamp 的 PATTERN 来使用。如果 PATTERN 没有 Time

Stamp 就不需要设定这个指令。

**(4) INTERVAL**

设定每个 PATTERN 之间的时间。如果设定了 INTERVAL 参数后就不能使用 Time Stamp 的用法。

**(5) FREQUENCY**

用途与 INTERVAL 一样。只是 FREQUENCY 所指定的单位是用频率，INTERVAL 所用的单位是时间。

**(6) RADIX**

设定 **PATTERN** 区域的进位值。如果 **PATTERN** 区域的数值有进位识别符号时，此时就要将 RADIX 的值设成 AUTO。例如 RADIX 设定成 AUTO 时，**PATTERN** 区域的某一数值为 35 与 35h 是不一样的，但是当 Radix 设成 HEX(十六进制)时，这两个值就是一样的。**但是当 RADIX 设成 DEC 时，而 PATTERN 区域的值为 35h 时，却是会被当成 35。**

RADIX 共有五种定义：

AUTO：由数值的进位识别符号决定。

HEX：16 进位。

DEC：10 进位。

OCT：8 进位。

BIN：2 进位。

而在 **PATTERN** 区域的进位识别符号定义是『**h**』代表 16 进位、『**o**』代表 8 进位和『**b**』代表 2 进位，而不加任何识别符号就代表 10 进位。

**(7) PATTERN**

是用来定义波形数据的区域。这个区域包含了两个部份，一个是时间部份一

个是波形部份。时间部份称为『Time Stamp』，时间部份和波形部份用『>』(大于符号)来做分隔。时间部份也可以省略，此时波形部份每一行的时间就会由 **INTERVAL** 或是 **FREQUENCY** 来指定，所以 **INTERVAL** 和 **FREQUENCY** 只能选用一种。如果使用 Time Stamp 方式，时间部份的单位就是 **UNIT** 所设定的值。波形部份就是根据 **INPUTS** 的定义顺序来描述波形。(请参考下面例子)

**(8) 注意事项**

**a.** 每个指令后面需用一个『**;**』(分号)来做结尾，PATTERN 后面的所有波形数据都属于 PATTERN 指令，所以不用加『**;**』，一直到所有波形描述结束后才需要加『**;**』。

**b.** 指令没有区分大小写，所以大小写都视为同一个指令。每个指令与参数之间至少要有一个空白。参数与参数之间也是一样需要有一个空白。**但是通道描述不能有空白，因为整个通道描述是当成一个参数。**例如：**3,5,6** 或是 **7..0**。

**c.** 需要一些批注时，可以用『**%**』符号将要说明或是注释的部份框起来，也就是批注的前面及后面各加入一个『**%**』符号即可。如果需要单行的批注时，可以直接加『//』两个斜线符号，该符号一直到换行符号之前都会被认定为批注。这些批注都不会影响任何向量档的结果。

**d.** 使用及指定的通道数不能超过 Acute PG 的使用上限(PGx020P 为 19，PGx050P 为 49，PKPG 为 15)。

**e.** PGV 向量文件的指令部份是不区分大小写，但是信号名称是会区别大小写的例如 Data [7..0]与 data [7..0]是不一样的所以在 **INPUTS** 指令及 **ASSIGN** 指令的指定参数的要一致，否则会发现无法指定通道的情形。

对话盒中有『重复输出』及『按键后才输出波形』的选项，是在转换的过程中自动加入 Acute PG 控制指令。如果向量文件中已经包含控制指令的话，就不需要在加入这些选项。对话盒中还有一个选项是有关频率的选项，这个

选项是给没有 Time Stamp 的 PATTERN 选用的。没有 Time Stamp 的频率是根据 **FREQUENCY** 或 **INTERVAL** 指令来设定，但也可以直接从对话盒来做变更。

请参考下面例子(使用 Time Stamp 的例子)：

**INPUTS Reset KeyPulseAck CmdWrite data[7..0] CmdClkIn Clk100K AdjClkIn2 AdjClkIn1 AddrClkIn;**

**Radix HEX;**

**UNIT ns;**

**PATTERN**

**0.0> 0 0 0 00 0 0 0 0 0**

**40.0> 1 0 0 00 0 0 0 0 0**

**50.0> 1 0 0 01 0 0 0 0 0**

**100.0> 1 0 0 02 0 0 0 0 0**

**150.0> 1 0 0 03 0 0 0 0 0**

**200.0> 1 0 0 04 0 0 0 0 0**

**250.0> 1 0 0 05 0 0 0 0 0**

**300.0> 1 0 0 06 0 0 0 0 0**

**350.0> 1 0 0 07 0 0 0 0 0**

**400.0> 1 0 0 08 0 0 0 0 0**

**450.0> 1 0 0 09 0 0 0 0 0**

**500.0> 1 0 0 0A 0 0 0 0 0**

**550.0> 1 0 0 0B 0 0 0 0 0**

**600.0> 1 0 0 0C 0 0 0 0 0**

**650.0> 1 0 0 0D 0 0 0 0 0**

**700.0> 1 0 0 0E 0 0 0 0 0**

**750.0> 1 0 0 0F 0 0 0 0 0**

**800.0> 1 0 0 10 0 0 0 0 0**

**850.0> 1 0 0 11 0 0 0 0 0**

**900.0> 1 0 0 12 0 0 0 0 0**

**950.0> 1 0 0 13 0 0 0 0 0**

**;**

请参考下面例子(使用 INTERVAL 的例子)：

**INPUTS Reset KeyPulseAck CmdWrite data[7..0] CmdClkIn Clk100K**

**AdjClkIn2 AdjClkIn1 AddrClkIn ;**

**Radix HEX;**

**INTERVAL 12.5ns ;**

**%FREQUENCY 8MHz ;%**

**PATTERN**

**0 0 0 00 0 0 0 0 0**

**1 0 0 00 0 0 0 0 0**

**1 0 0 01 0 0 0 0 0**

**1 0 0 02 0 0 0 0 0**

**1 0 0 03 0 0 0 0 0**

**1 0 0 04 0 0 0 0 0**

**1 0 0 05 0 0 0 0 0**

**1 0 0 06 0 0 0 0 0**

**1 0 0 07 0 0 0 0 0**

**1 0 0 08 0 0 0 0 0**

**1 0 0 09 0 0 0 0 0**

**1 0 0 0A 0 0 0 0 0**

**1 0 0 0B 0 0 0 0 0**

**1 0 0 0C 0 0 0 0 0**

**1 0 0 0D 0 0 0 0 0**

**1 0 0 0E 0 0 0 0 0**

**1 0 0 0F 0 0 0 0 0**

**;**

关于 PGV 档案更细节的部份，请参阅：

使用文字编辑器编辑文字向量文件

# 11. Acute LA 波形档转换

如果您已经使用了本公司的 Acute LA 时，您就可以用 Acute LA 来撷取电路上的波形。您只要取得该波形档，就可以透过转换程序将 Acute LA 的波形档转换成 Acute PG 的波形檔。PG-Editor 程序会在转换结束的时候，将转换结果放入波形编辑器中，此时您可以再经由波形编辑器加以编修，然后由 Acute PG 来输出。也就是说当您没有对方的电路所产生的波形时，也可以藉由 Acute LA 及 Acute PG 让远方的电路信号在您这里重现。

波形转换时要注意目前只能转换 Acute LA 64K 深度模式的波形文件。因为 Acute PG 可用的通道数为 16(PKPG)、20(PGx020)及 50(PGx050)，所以 Acute LA 的波形档，不能设定超过 16、20 或是 50 个通道，否则将出现错误讯息。

**LA Viewer** (Acute LA 系列的操作软件) 1.50 版之后，就支持 PGW 的波形格式，所以您有 Acute LA 系列逻辑分析仪时，请用 LA Viewer 1.5 版之后的版本。在储存波形档时，只要选择存成 PGW 格式即可，详细使用方法请参考 LA Viewer 使用手册。

# 12. Altera 波形档转换

FPGA 与 PLD 可程序逻辑的运用已经越来越广泛，许多的设计师都会用 FPGA 与 PLD 来当作控制电路，所以设计这些可程序逻辑时，零件商都会提供许多设计用的应用软件，通常都会包括仿真程序，让您在软件上就可以轻松的仿真 FPGA 和 PLD 的功能。

但是一旦设计好了以后，您如何去验证 FPGA 或是 PLD 的工作是否正常呢?此时就可以运用 Acute PG 来产生 FPGA 和 PLD 输入脚的信号，并将信号喂入。再用 Acute LA 来撷取输出脚的信号，以此来验证功能的正常与否。这种作法是非常好用的，因为有些仿真程序要将信号完整的仿真必需花费很多时间，而且模拟的结果都是理想化的。运用真实的验证可以真正得到零件的运作情形。同样这种作法也可运用在 IC 的测试或是其他类似的测试及验证环境。

一般仿真程序都必需建立一些波形档案或是向量档案，以仿真设计的电路。一旦建立好这些档案后您就可以利用 PG-Editor 来将这些波形档或是向量档转成 Acute PG 的波形檔，再由 Acute PG 来提供真正的波形给 FPGA 或是 PLD 的电路使用。由于仿真程序相当的繁多，所以本公司会陆续提供这些波形档或是向量文件的转换程序。

而 Altera 波形档转换就是将 Altera 的 Max Plus II 软件所产生的波形档转换 Acute PG 的波形檔。Max Plus II 支持的波形文件格式分成两种，一种是 Binary File(.SCF)，另一种是 Text File(.VEC、.TBL)。而 Text File 是一种向量格式的波形档，PG-Editor 只支持 Text File 格式。所以如果是 Binary File 格式的话，就必需用 Create Table File(请参考 Max Plus II 的用法)的功能来产生 Text File 的格式。然后再用 PG-Editor

的转换程序转成 PG-Editor 的波形。

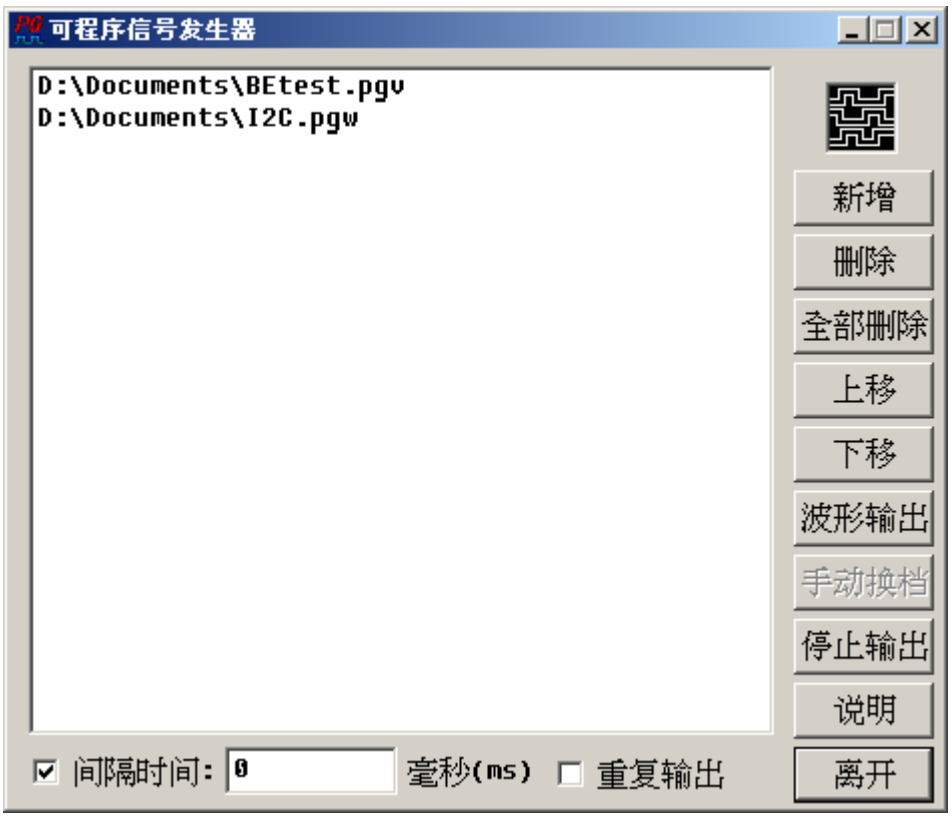
## Altera 波形档转换的使用方法：


首先要先输入一个 Max Plus II 的 Table 档，然后按下加载钮。此时在表列区中就会出现待转换的信号名称和转成 Acute PG 后的信道号码。由于 Acute PG 是一个输出信号的仪器，所以转换程序只会将 Table 档案中的输入脚的信号取出并加以转换。这时『单位设定』会出现经过换算的时间单位，由于 Acute PG 的最小时间单位为 10ns，所以如果 Table 文件所用的时间单位小于 10ns，转换程序就只会设定成 10ns。但是这个时间单位亦可以自行修改，以配合实际的需求。在转换的过程中也可以加入『重复输出』或是『按键后才输出波形』的选项。『重复输出』就是让 Table 档的波形可以不断的重复输出。『按键后才输出波形』的选项是代表当您执行输出波形前会暂时停止，然后再按下启动热键才会真正输出波形。一旦两个选项同时选择时，可以达到重复测试待测电路以外，而且每测试完一次都可停下来，等待下次按键再行测试。

# 13. 批次输出

批次输出可以让您一次输出大量的档案，它所接受的格式为使用波形编辑器所编辑的波形档(扩展名为.PGW 或是.PGV)。而且它提供一个很方便的功能，就是您能够设定每个输出档案之间的间隔时间以及使用重复输出功能，让您所要输出的档案就像循环一样依序输出，可以让您的测试工作自动化，让您的测试工作更简单。

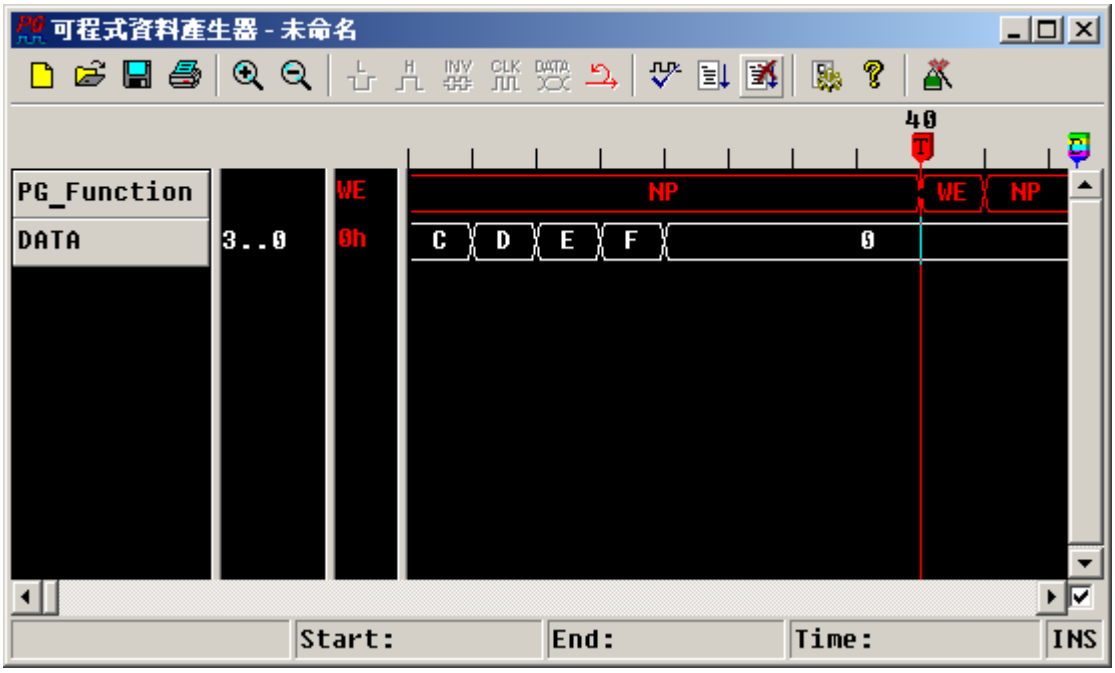
**批次输出的使用方法：**




首先，按下『新增』选择您要输出的档案，假如你要输出多个档案，您可以使用『上移』或是『下移』来决定您所要输出档案的顺序。『删除』是删除单一档案;而『全部删除』顾名思义就是删除你在窗体中的所有档案。您可以选择勾选『间隔时间』并填入数字来决定你每个输出档案之间的间隔时间，或是不勾选而选择『手动换档』，而勾选『重复输出』可以让您的档案循环般的输出，直到您按下『停止输出』

才会停止，而倘若您没有勾选『重复输出』那只会输出一遍就停止输出了。

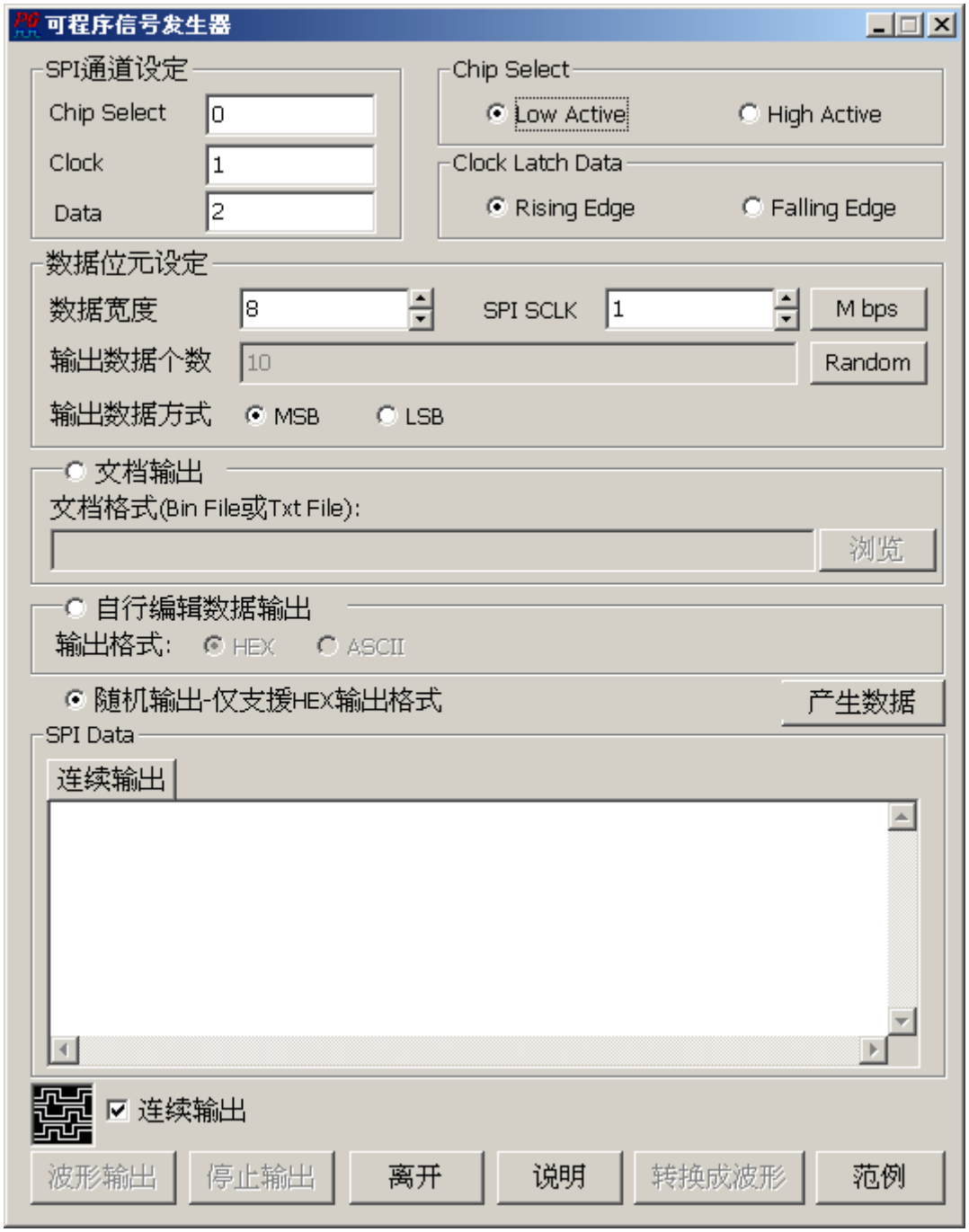
注意：多个档案输出时，您必须在每一个波形档(.PGW 或是.PGV)的波形结尾部分加上 Wait for Event(WE)的指令，来让程序辨别档案的结束处并输出下一个档案，否则将无法产生批次输出的效果。您可以使用波形编辑器来下该指令，以下为使用范例。



# 14. SPI 信号发生器

SPI 信号发生器可以产生 SPI 的信号，它可以读取扩展名为.bin 或.txt 的档案来产生或是自行编辑 SPI 的数据，也可以使用程序来随机产生数据。SPI 信号发生器可以让您的测试工作更有效率。

**SPI 信号发生器的使用方法：**




首先，是输入您要产生 SPI 信号的参数部份。在通道设定方面：您可以选择 PG 要输出 SPI 信号的信道，默认为 CH0 为 SPI 信号的 Chip Select，CH1 为 Clcckt，CH2 为 Data。 而在 Chip Select 和 Clock Latch Data 的参数设定上就根据您的需求来做设定，在此不多做叙述。

在 Data Bits Setup 的参数设定里，输出频率为 SPI Clock 的输出频率；而 MSB 和 LSB 则是在 Data Bits 设定完成之后选择高低位的输出顺序，例如：1ah (选择 8 Data

Bits, MSB)那如果您选择 8 Data Bits, LSB，将会变成 58h。

输出数据个数则是会显示您自行编辑的数据个数或是使用随机的方式产生数据，使用随机产生数据的方式其数据个数可以手动输入数据个数或是随机产生数据个数，以下为使用随机输入数据个数步骤：勾选『随机输出』→『波形输出』。以下为手动输入数据个数步骤：勾选『随机输出』→『随机』→输入要输出的数据个数。勾选『连续输出』可以连续输出该数据，若不勾选，该数据只会输出一次，随机输出只能产生十六进制数据。

使用档案输出的步骤：按下『浏览』→选择 Bin 或是 Txt 档案→『波形输出』，Bin 档案只能以 Hex 的格式输出，而 Txt 档案只能以 ASCII 格式输出。

Data Bits 项目必须详细做说明，Data Bits 可以输入的范围为 8~24 Bits。倘若您是使用读取 bin 档案的方式来产生 SPI 的数据，如果此时您选择的 Data Bits 的个数为 8Bits 则程序会以 8 Bits(1 byte)的方式读取档案数据并输出，但是如果您选择的 Data Bits 个数为 9~15，则程序会以 16 Bits(2 Byte)来读取数据，不足 16 Bits 的部份将会被屏蔽掉，也就是说，如果您是选择 9 Bits 来读取档案数据，读取的 16 Bits 中只保留低位 9 Bits 的数据,而其他的部分将被屏蔽掉，其他依此类推。而倘若你是选择随机产生 SPI 数据值的方式，而这种方式所能产生的数据值最大为 7fff (16 Bits)，如果您选择的 Data Bits 为 16 bits 以上将无意义。

而如果您是选择自行编辑 SPI 数据的方式，有一点必须要注意的是；如果您编辑的数据为 16 Bits，但是您却选择 Data Bits 为 8 Bits，则高字节的部份将会被屏蔽掉，例如：您输入 162ah，这个数字必须使用 16 Bits 才足以表达该数字，但是您若是选择 8 Bits，则 SPI 数据产生器只会输出 2ah，其他依此类推。而自行编辑数据的格式如下(Hex)：

输入：

2a;↵(Enter 键)

45;↵(Enter 键)

1c;↵(Enter 键)

67;↵(Enter 键)

52;↵(Enter 键)

如果您是要以 Hex 的方式输出，您必须遵照上述格式，也就是输入一个数字，然后输入分号，再按下『Enter』键。但是倘若您是要以 ASCII 的方式输出，就不需要遵照上述格式。清除数据则是可以清除数据编辑区的数据，而范例可以自动输出一些范例的格式介绍。

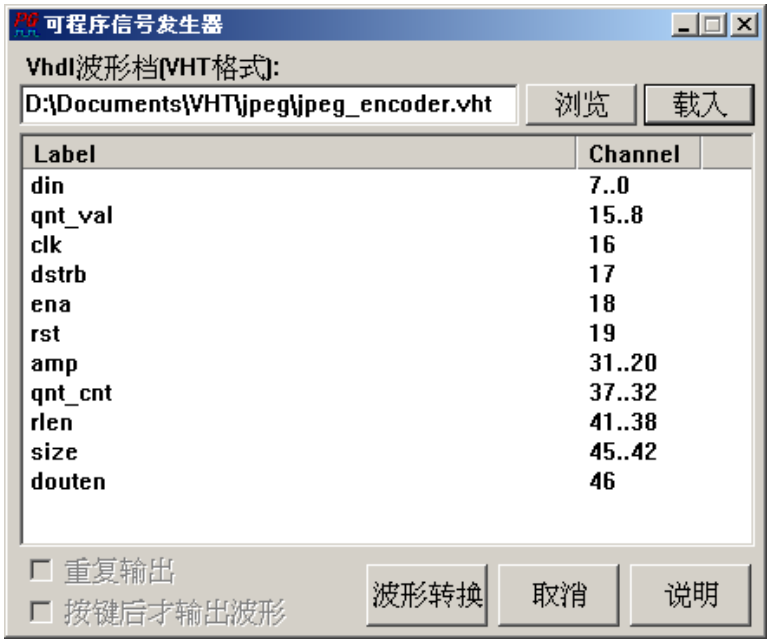
# 15. VHDL 波形档转换


FPGA 与 PLD 可程序逻辑的运用已经越来越广泛，许多的设计师都会用 FPGA 与 PLD 来当作控制电路，所以设计这些可程序逻辑时，零件商都会提供许多设计用的应用软件，通常都会包括仿真程序，让您在软件上就可以轻松的仿真 FPGA 和 PLD 的功能。

但是一旦设计好了以后，您如何去验证 FPGA 或是 PLD 的工作是否正常呢?此时就可以运用 PG 来产生 FPGA 和 PLD 输入脚的信号，并将信号喂入。再用 LA 来撷取输出脚的信号，以此来验证功能的正常与否。这种作法是非常好用的，因为有些仿真程序要将信号完整的仿真必需花费很多时间，而且模拟的结果都是理想化的。运用真实的验证可以真正得到零件的运作情形。同样这种作法也可运用在 IC 的测试或是其他类似的测试及验证环境。

而 VHDL 波形档转换就是将 Altera 的 Ouartus II 或是 Xilinx 软件所产生的波形档转换 Acute PG 的波形。然后再用 PG-Editor 的转换程序转成 PG-Editor 的波形。

## VHDL 波形檔的使用方法：



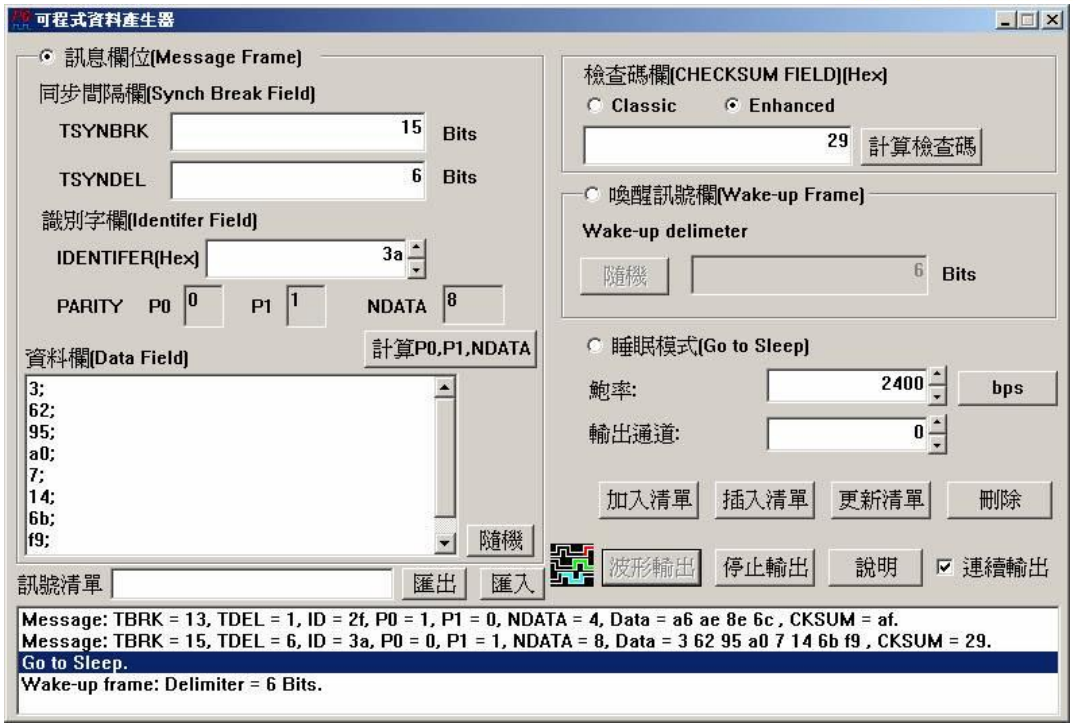

选择要转换的 VHT 档案，然后根据您的需求，选择是否要重复输出或是要按键后才输出波形，当然也可以同时勾选或不勾选。再按下加载键，再按下波形转换键。转换程序会将 VHT 档案中的信号依输入信号组、输入单一信号、输出信号组以及输出单一信号的顺序作转换。

**注：信号信道数目如果超过 50 个，将不会输出。**

# 16. LIN 信号发生器

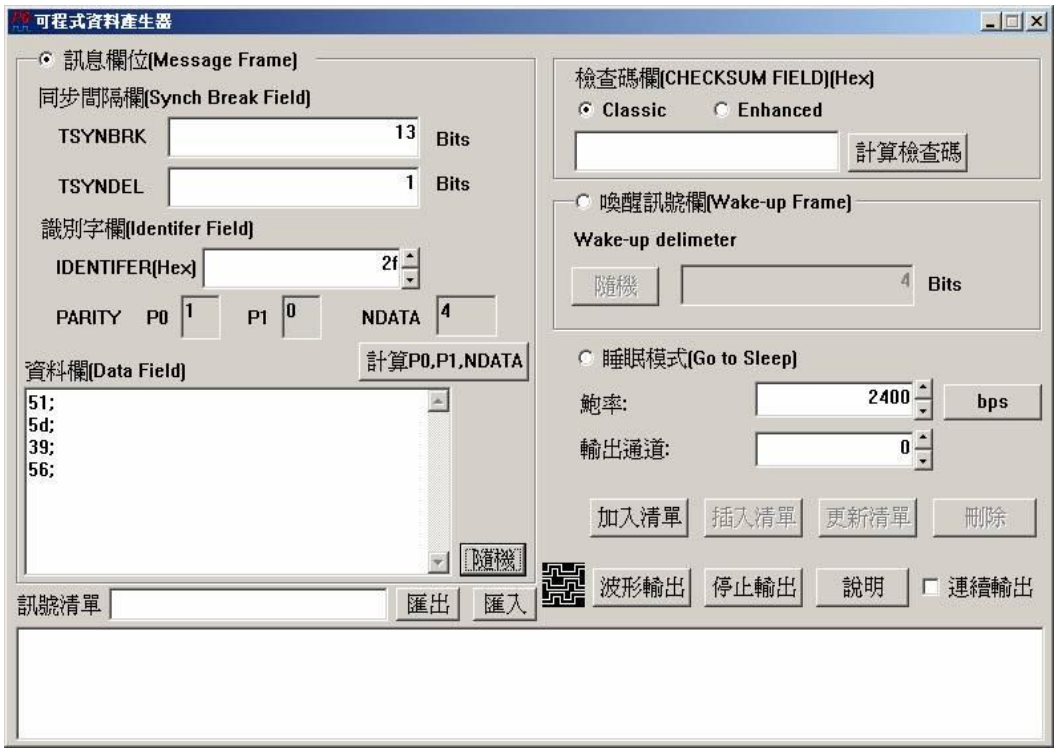
随着汽车市场的蓬勃发展，车用电子的传输控制也越来越重要；CAN 和 LIN 都是车用电子里常见的传控接口。而 LIN BUS 是车用电子中为因应低成本趋势而产生的一种传控接口，主要是使用在低速的周边装置，如车门控制、车窗控制等。

**LIN 信号发生器的使用方法：**



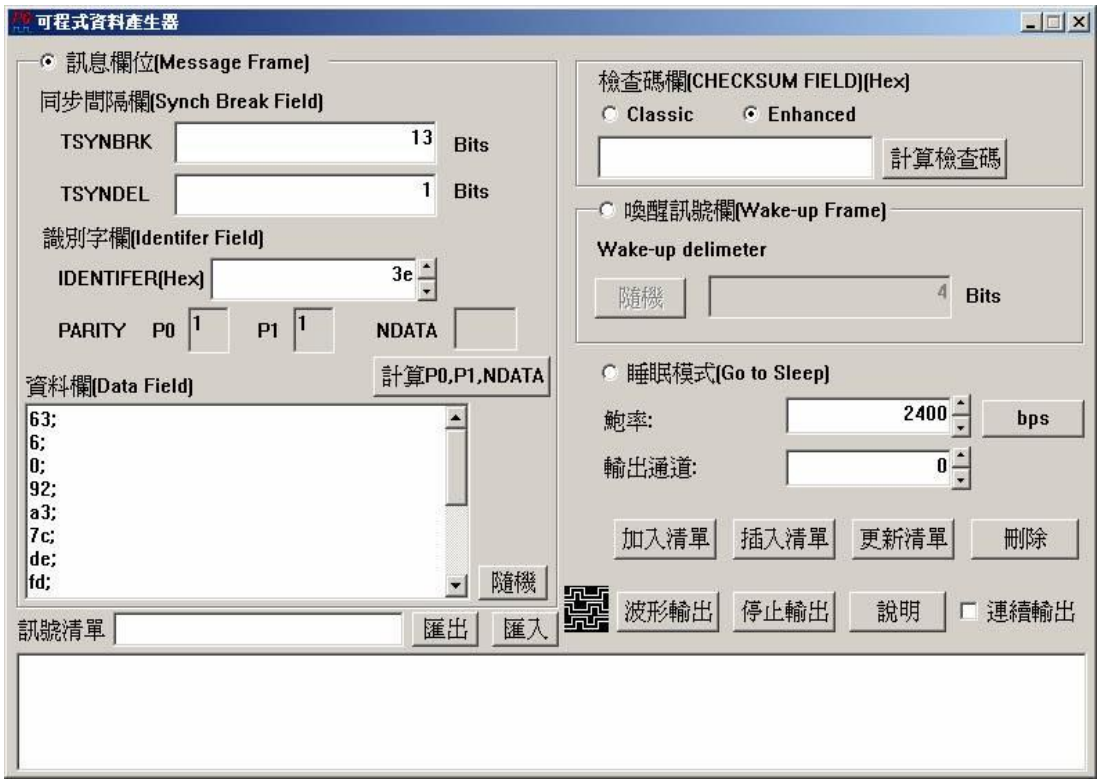

LIN 信号中一个完整的 MESSAGE FRAME 是由 HEADER 和 RESPONSE 所组成。而 HEADER 可以依序区分成三个部份，分别为 SYNCH BREAK FIELD(同步间隔栏)、SYNCH FIELD(同步栏)和 IDENTIFER FIELD(识别栏)。而 RESPONSE 依序分为两部分，分别为 DATA FIELD(数据域)和 CHECKSUM FIELD(校验和栏)。SYNCH BREAK FIELD(同步间隔栏)可以区分为两部分，分别是 TSYNBRK 和 TSYNDEL(synchronization delimiter)。按照 LIN 通讯协议的规定，TSYNBRK 必须大于或等于 13 bits; TSYNDEL 必须大于或等于 1 个 bits。您可以自行输入该值，如果您输入的值超过限制，该程序会以讯息通知您，请您不用担心。

SYNCH FIELD(同步栏)固定为十六进制值 0x55，不需要做任何变动，所以程序中并没有显示这方面的信息。再来为 IDENTIFER FIELD(识别栏)，它依序区分为 6 个 bits 的标识符(IDENTIFER)和 2 个 bits 奇偶校验位。您只要按下[计算 P0,P1,NDATA]按钮，程序会自动帮您计算奇偶校验位和 N-DATA，NDATA 是决定您所要传送数据的个数。以程序来举例做说明：




输入 IDENTIFER 的值为 0x2F，接下来按下[计算 P0,P1,NDATA]按钮，经过程序计算 P0(奇校验)为 1，P1(偶校验)为 0，NDATA 为 4。也就是说您必须自行输入 4 个 8bits 的数据或是使用随机来产生(该数据为十六进制值)。

这里有一点必须要注意 ,如果您输入的数据个数超过 4 个 ,那程序只会按照 NDATA 之值，也就是 4 去选取您所输入的前 4 个的数据。如果您不想局限于资料个数，您可以输入 IDENTIFER 为 0x3E 或是 0x3F，此时就可以不用理会 NDATA 之值，程序会根据您所输入的数据个数去选择所有的数据。




这里我们输入 IDENTIFER 为 0x3E，并按下[随机]，结果产生 14 个数据值，当然您也可以自行输入想要传送的数据值。

**注：这里随机产生的数据个数为 0~20 个。**

MESSAGE FRAME 的最后一个字段为 CHECKSUM FIELD(校验和栏)，您可以选择勾选 Classic(Ver 2.0 之前的计算检查码的方式)或是 Enhanced(Ver 2.0 之后)。最后输入鲍率以及决定使用 PG 的哪一个通道输出，如果选择 0 就是从 PG 的通道 0 输出；1 就是从通道 1 输出，依此类推。

然后按下[加入列表]，程序会帮您将该 MESSAGE FRAME 加入到 LIN 的信号列表，PG 就是根据这份列表来输出 LIN 的信号；而[插入列表]、[更新列表]以及[删除]可以帮助您来修改该信号列表。接下来，就针对这 4 个修改列表的按钮来做说明：

[加入清单]会依序将您想要的输出字段依序加入到输出清单，举例来说：我想让 PG 送出一个 LIN 的 MESSAGE FRAME，然后让 LIN 进入睡眠模式，然后再送一个唤

醒信号，最后再紧接着一个 MESSAGE FRAME。做法如下：

**(1)** 首先选取[讯息字段]

**(2)** 输入 IDENTIFER，假设为 0x2F

**(3)** 然后按下[计算 P0,P1,NDATA]，结果 NDATA = 4

**(4)** 然后我按下[随机]产生数据，也可以自行输入，注意自行输入数据有其格式，请遵照以下方式：

5d;↵

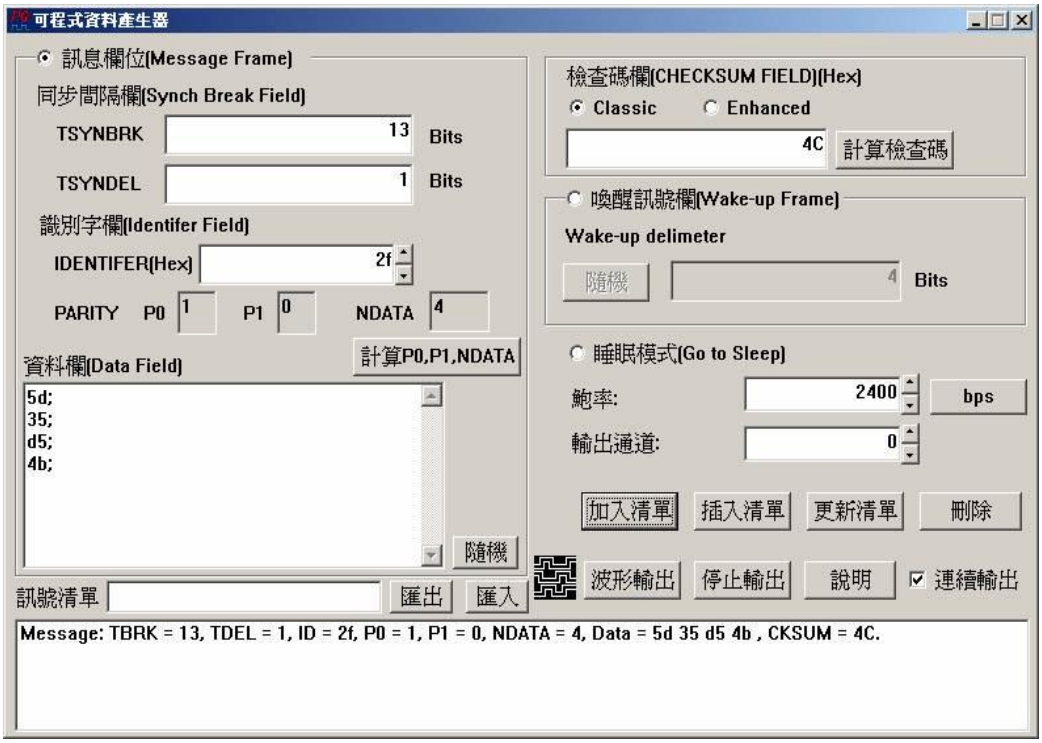
35;↵

d5;↵

4b;↵

**(5)** 按下计算检查码，我勾选 Classic 的计算方式

**(6)** 按下[加入清单]，结果如下：

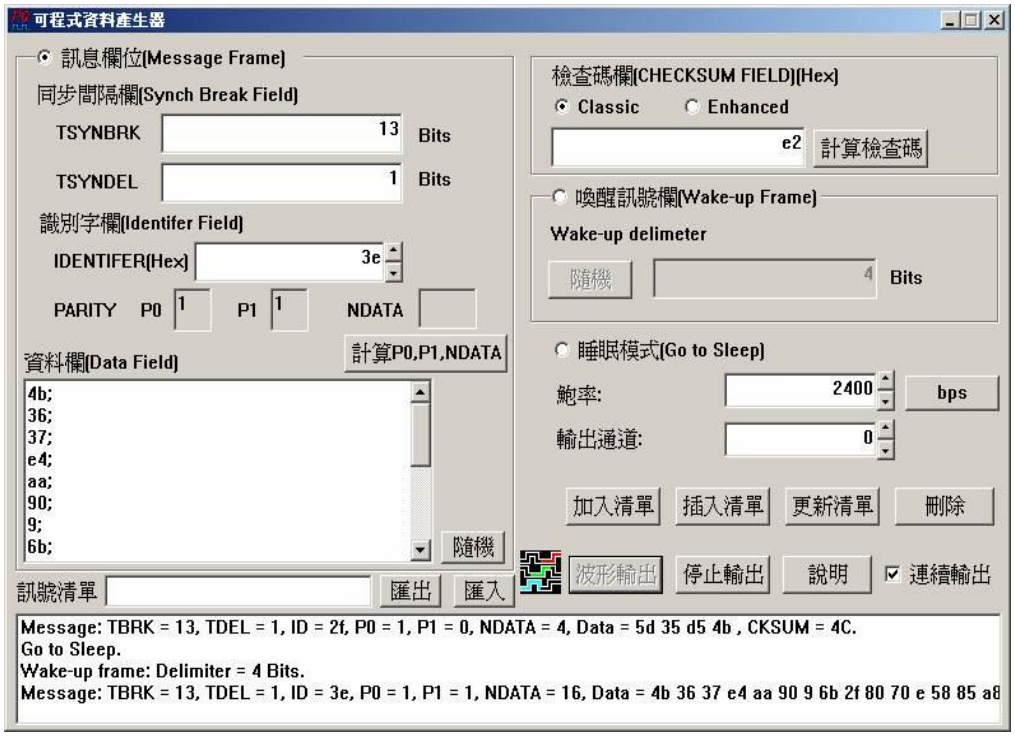



**(7)** 然后选取[睡眠模式]，再按下[加入清单]

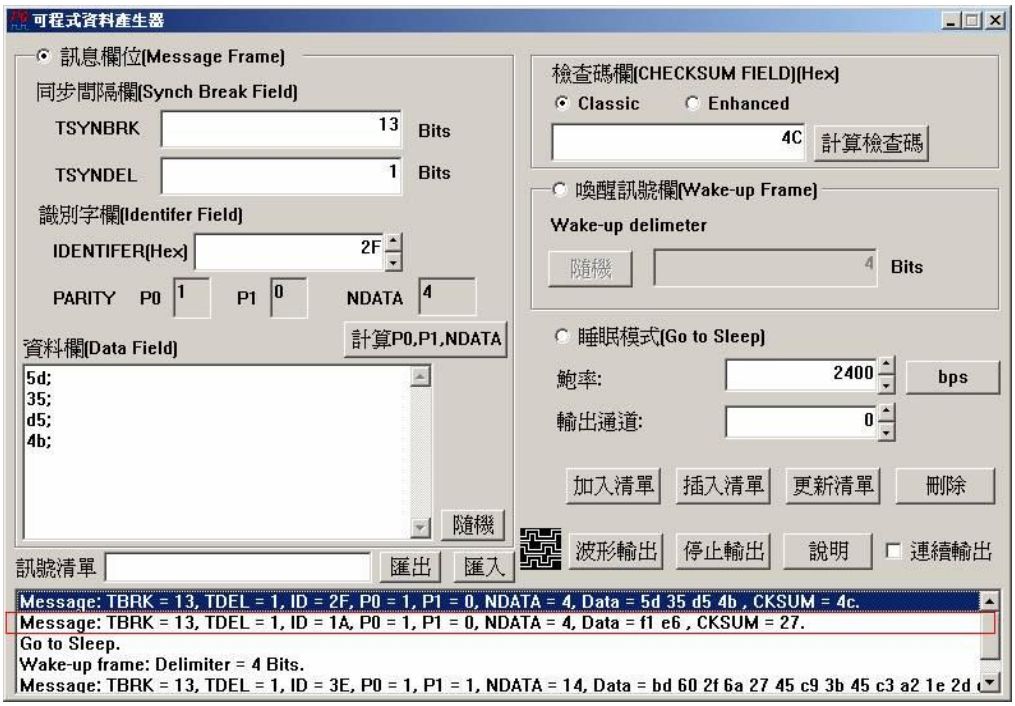
**(8)** 选取[唤醒信号栏]，再按下[加入清单]

**(9)** 再选取[讯息字段]，此时输入的 IDENTIFER 为 0x3E，接下来的步骤同 1~4，

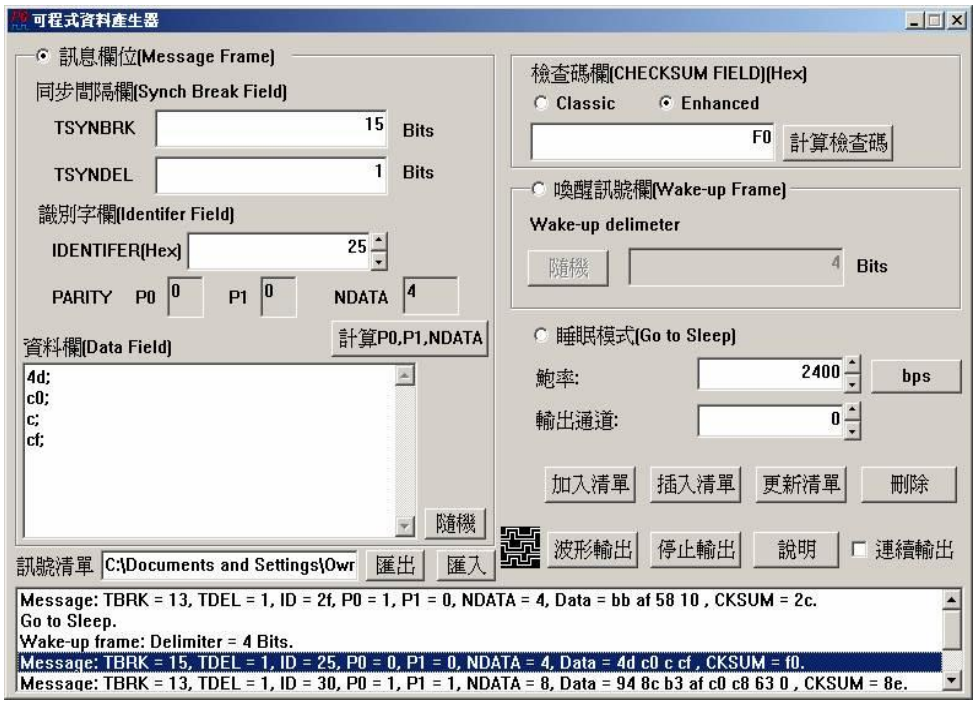
最后再按下[加入清单]，结果如下：




[插入清单]的使用方式：您可能会在上一例中，想在第一个 MESSAGE FRAME 之后再插入一个 MESSAGE FRAME，此时您就可以使用[插入列表]的功能，首先，使用鼠标选择您想要在哪一个项目之后插入新项目。我们选择在第一项之后插入，也就是成为清单的第二项。所以使用鼠标点选第一项，然后我们加入一个 IDENTIFER 为 0x1A 的 MESSAGE FRAME，然后遵照 1~4 的步骤，最后按下[插入清单]，您可以看到在点选的第一项之后多出了我们刚刚插入的项目。



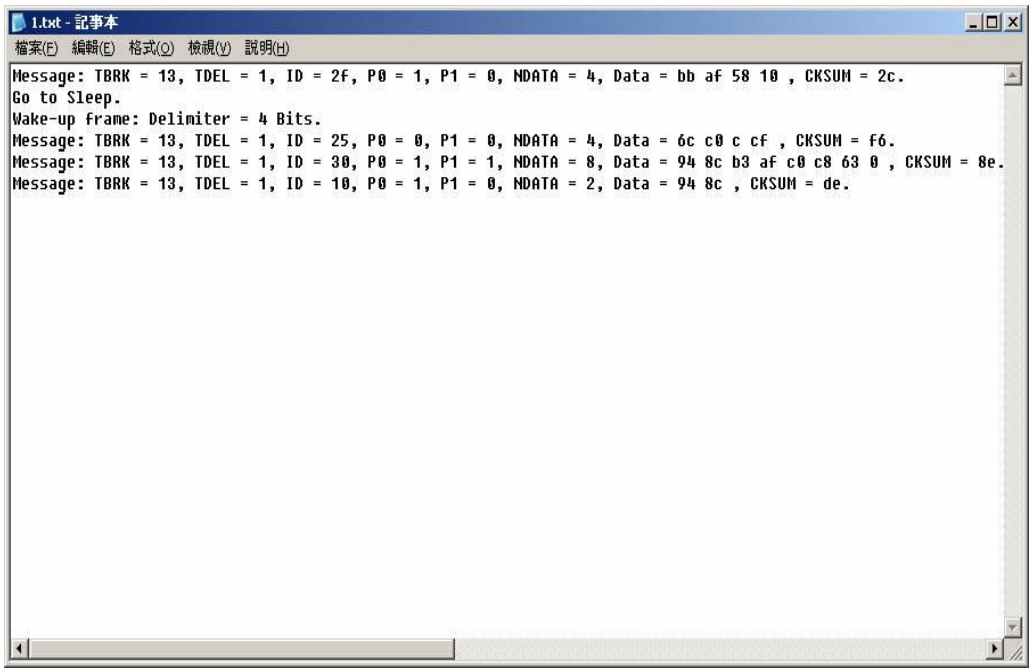

[更新清单]的使用方式：您可能想要在一长串的信号中修改某项目里的某一个字段的内容，此时您就要使用[更新列表]的功能，假设您想要修改第四项的 TSYNBRK 以及将第一个数据从 0x6C 改为 0x4D，此时将鼠标点选第四项，然后将 TSYNBRK 的 13 bits 改为 15 bits，将数据域的第一个 0x6C 改为 0x4D，然后按下[更新清单]，就完成修改的动作。注：此时的 CHECKSUM 也必须做修改。




[删除]的使用方式：您可以使用鼠标点选项目，然后按下[删除]键，则程序会

将该项目从信号列表删除。

[汇出]和[汇入]是将您的信号列表存成文本文件或是将该文本文件加载，您也可以藉由修改该文本文件来更改您所要输出的 LIN 信号列表，不过，请依照该文本文件的格式，否则将会发生错误。

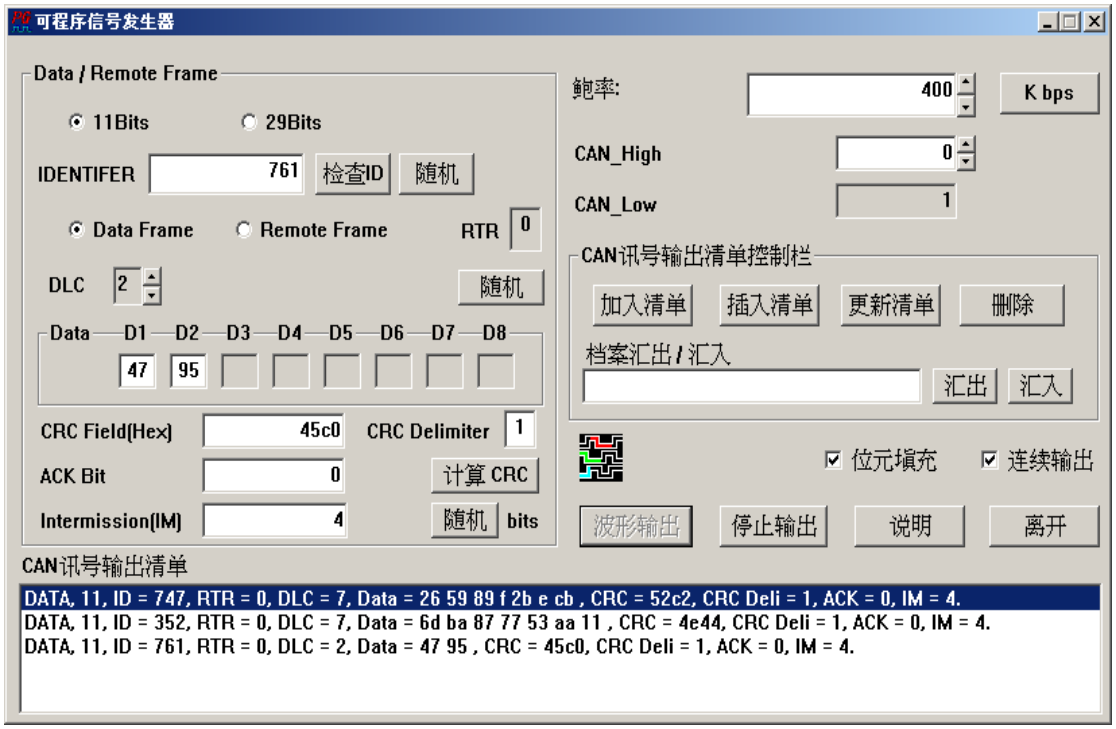



## 17. CAN 信号发生器

CAN BUS(Controller Area Network)协议技术的使用在汽车工业上已有相当一段时间，其目的为因应汽车日益进步，电子系统及配备也越来越多且复杂，需要有许许多多线路来连结，形成庞大而复杂的线束系统，直接影响造车成本提高、车体重量等问题。

CAN Bus 的原理是将计算机网络技术概念，运用在汽车上，每个汽车电子零件或作动器就像是网络用户一样，只需简单一条线即可达成接收传送沟通等功能，再由控制器整合所有信号，传递至各个系统使用，而各个感知器、作动器、控制单元如有任何问题或故障，可以透过信号传递，诊断问题所在。

## CAN 信号发生器的使用方法：




CAN 信号的 MESSAGE FRAME 可以分成 4 种，分别为 DATA FRAME、REMOTE FRAME、ERROR FRAME、OVERLOAD FRAME 等。DATA FRAME 是 CAN 中最为普遍的类型，数据都是从一个节点传送到所有节点；REMOTE FRAME 与 DATA FRAME 极为相似(只有 RTR bit 被设为 1)，主要的用途是用来向其他的 CAN 的节点要求数据；ERROR FRAME 是用来表示在 CAN BUS 上有错误状况被侦测到，而 OVERLOAD FRAME 用来告知其他的 CAN 的节点，必须有更多的时间来处理收到的数据，主要功能是延迟下一个封包被传送的时间。

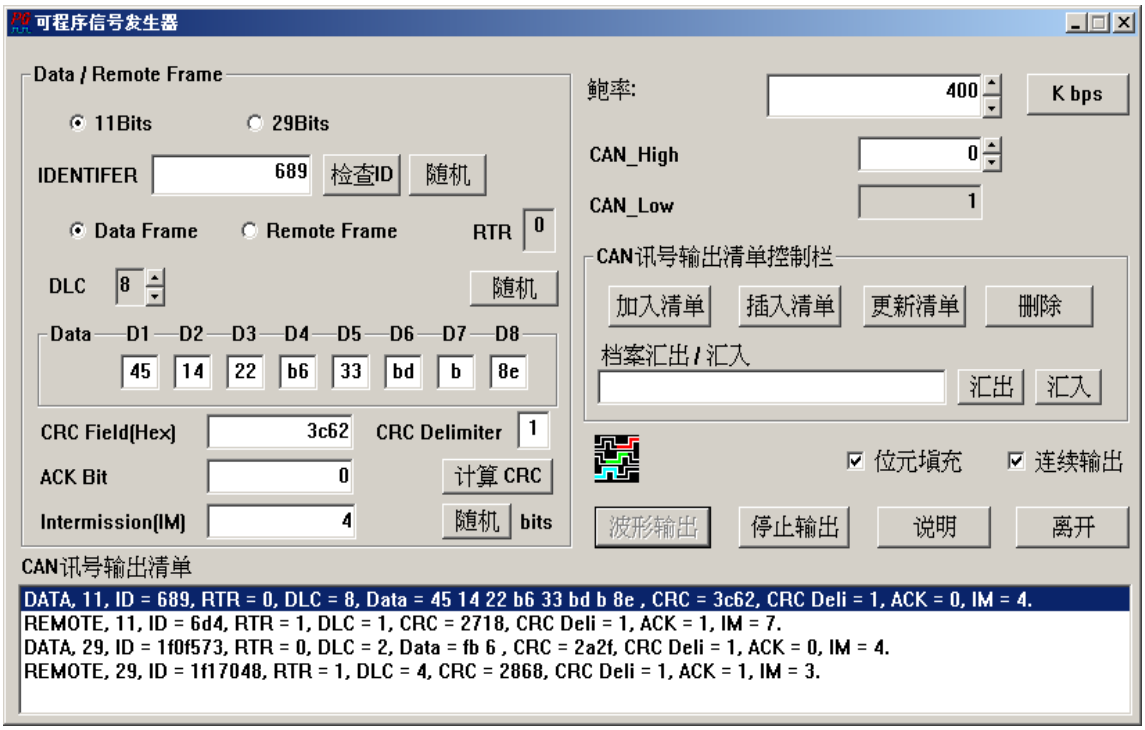
使用该程序时，首先，决定您要使用 STANDAND FRAME 还是 EXTENDED FRAME，如果您选择 11 bits；那就表示您是使用 STANDARD FRAME 的模式，反之，您就是选择 EXTENDED FRAME 模式。11 bits 和 29 bits 是表示您是使用哪一个 IDENTIFER 的表示方式，以实例来说，如果您输入的 IDENTIFER 为 0x5A9，那就要请您选择 11 bits，倘若您输入的 IDENTIFER 为 0x1F1518D，那就要选择 29 bits。这里有一点必须加以说明：当您选择 29 bits 时，它是将您输入的 IDENTIFER 分为 11 个 bits MSB 和 18 个 bits LSB，所以是 29bits，然后，中间插入 SRR、IDE 2

个为 1 的 bit 以及最后加入 1 个 RTR bit 结合成 Arbitration Field。如果您担心输入的 IDENTIFER 是否正确，您可以按下[随机]由程序来帮您产生正确的 IDENTIFER。

然后，RTR 会根据您是要使用 DATA FRAME 或是 REMOTE FRAME，如果是 DATA FRAME，RTR 为 0，REMOTE FRAME RTR 为 1。再者根据您的需求填入 DLC(Data Length Code)而 Data 字段 D1~D8 会根据您输入的 DLC 个数来运作。

在您填妥 IDENTIFER、DLC、DATA 之后，接下来请按下[计算 CRC]，程序会自动帮您算出该段输出数据之 CRC 值，再来根据您的需求输入 CRC Delimiter、ACKnowledge 和 Interframe Space (间隔位)，Interframe Space 可以随机产生，而 ACKnowledge 和 CRC Delimiter 请输入 0 或 1 和 1。

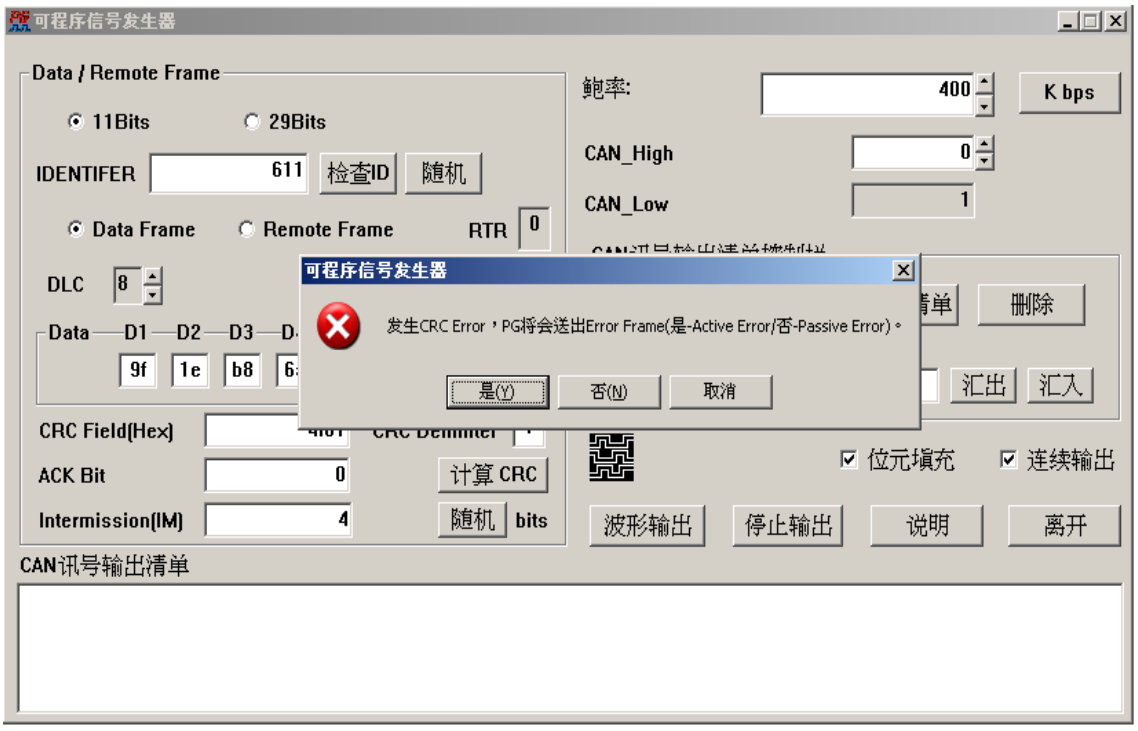
然后决定鲍率和 CAN_HIGH、CAN_LOW 的输出通道，然后按下[加入清单]，将您要输出的 CAN 信号填入输出列表，以下以实例说明：



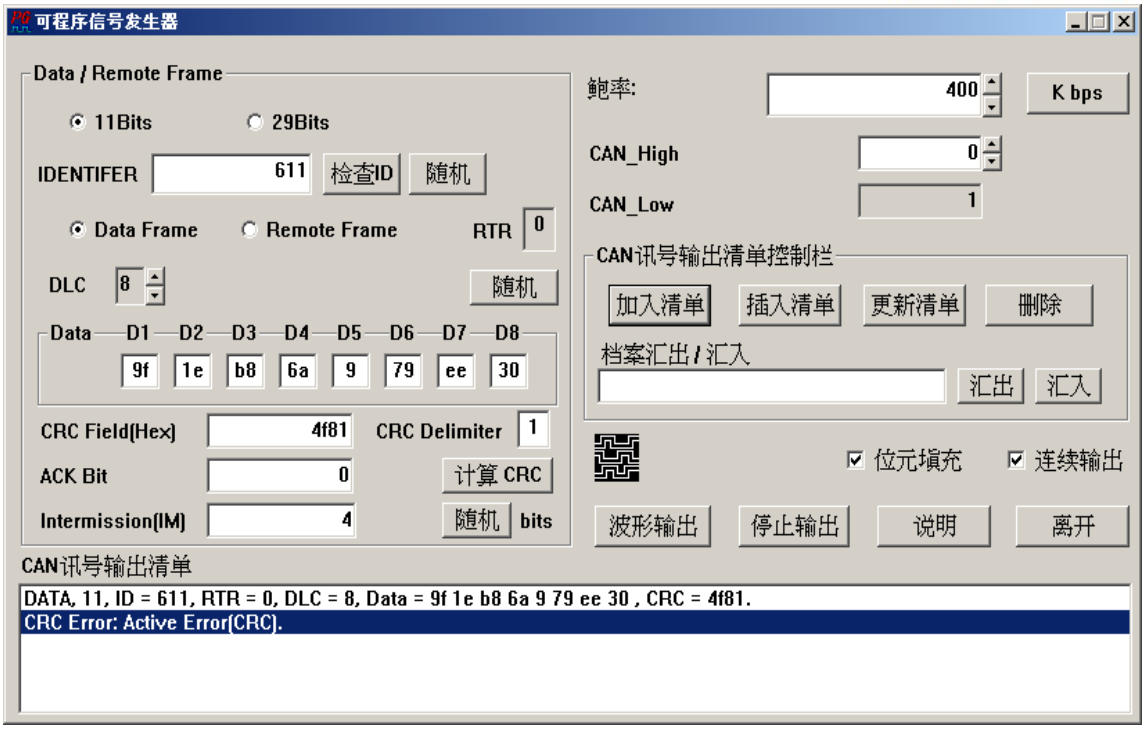

假设我要产生 4 个 FRAME，2 个 DATA FRAME 和 2 个 REMOTE FRAME，分别为 STANDARD FRAME 和 EXTENDED FRAME，请参考上图的信号输出列表。

接下来为 4 个新增和修改 CAN 输出列表的按钮，分别为[加入列表]、[插入列表]、[更新列表]、和[删除]。使用方式和 LIN 信号发生器的方式一样，请参考 LIN 信号发生器的说明文件。

这里有一点要说明，CAN 信号发生器在您输入错误的 CRC、CRC Delimiter 会产生相对应的 ERROR FRAME，以 CRC ERROR 做实例说明：假设我使用随机方式产生 IDENTIFER 为 0x611，选择 DATA FRAME，STANDARD FRAME，随机产生 DLC 和 DATA 分别为 8，0x9F、0x1E、0xB8、0x6A、0x09、0x79、0xEE、0x30，然后计算过后，正确的 CRC 值为 0x4f80 它改成错误的 CRC 值为 0x4f81 受默认值，然后按下[加入清单]，您可以看到以下图例的部分：

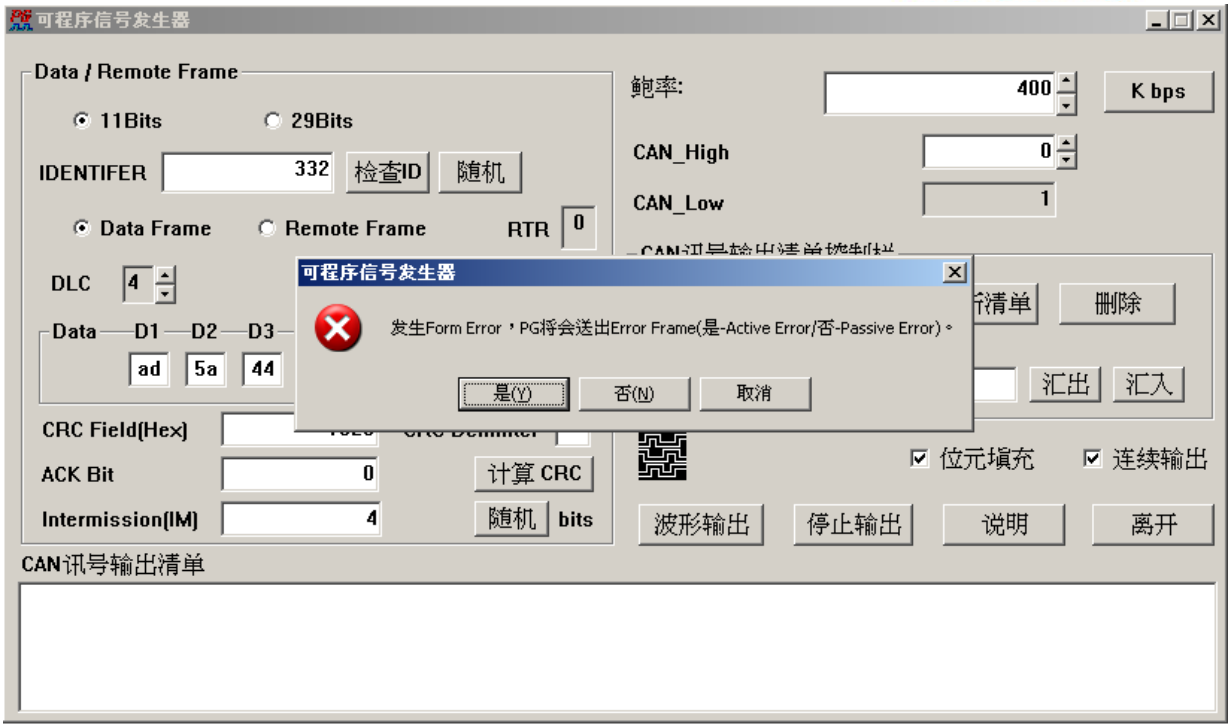


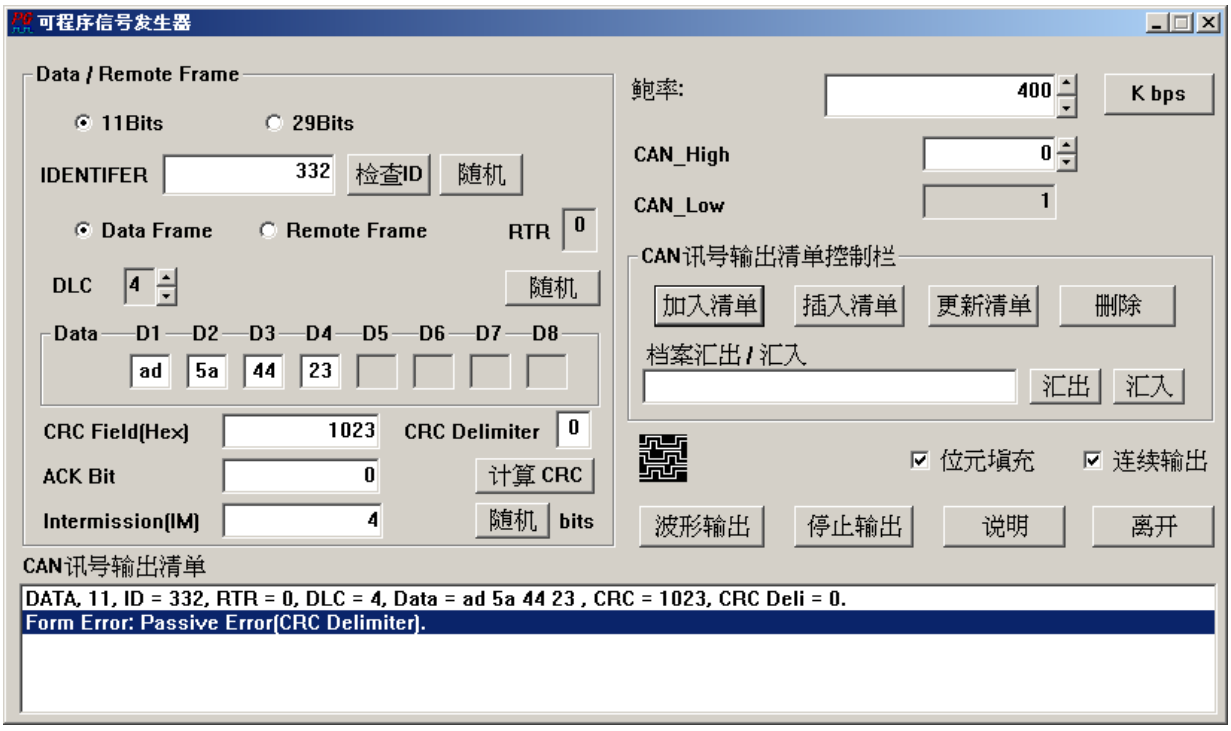

程序会出现发生 CRC ERROR，会请您选择产生 Active Error 或是 Passive Error 或者是取消这个 FRAME，此例我是选择产生 Active Error，可以参考底下图例的部份：




这个 CAN 信号列表将会产生 DATA FRAME 11 bits IDENTIFER，1 个 RTR bit，DLC = 8，Data = 0x9F, 0x1E, 0xB8, 0x6A, 0x09, 0x79, 0xEE, 0x30，CRC = 0x4f81。但是因为这个 FRAME 是 CRC ERROR，所以这个 FRAME 后面的部份就不再传送而紧接着由使用者所选择的 Active Error Frame。

再者，说明产生 Form Error 的情形：将原本正确的 IDENTIFER 为 0x332，RTR = 0，DLC = 4，Data = 0xAD, 0x5A, 0x44, 0x23, CRC = 0x1023, CRC delimiter = 1, ACK = 0, Intermission = 4 的 11bits DATA FRAME 中将原本正确的 DATA FRAME 中的 CRC Delimiter 改为 0，即会产生 Form Error，如下图：



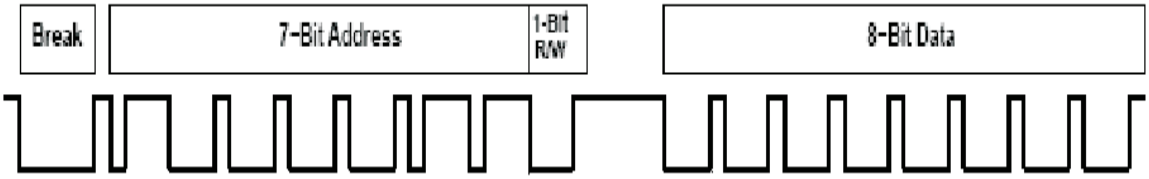




**备注：每个发生错误的原因都有( )注明在 ERROR FRAME 之后。**

而[导出]和[汇入]的功能则和 LIN 信号发生器的用法相同，请参考 LIN 信号发生器的说明。

# 18. HDQ 信号发生器

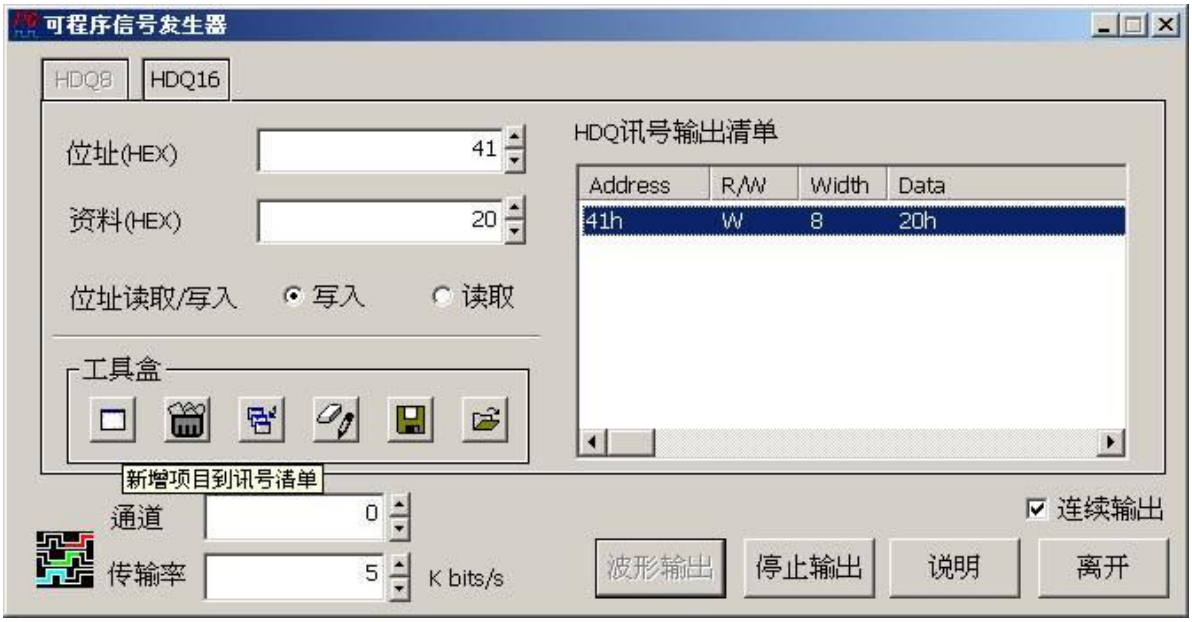
HDQ 协议是由德州仪器(TEXAS INSTRUMENTS)所制定，使用于电池管理的显示应用，主要是运用在消费性电子产品方面。




HDQ 分为 8 位与 16 位两种数据宽度格式，地址固定为 7 位。一个 HDQ 的封包主要由 **Break**、**7-Bit Address**、**1-Bit R/W** 和 **8-Bit Data** 或是 **16-Bit Data** 所组成。传输的方式为 LSB (Least-significant bit)到 MSB (Most-significant bit)，最大传输率为 5Kbits/s。

## HDQ 信号发生器的使用方法


**HDQ** 信号发生器的使用方法主要是根据您的信号需求，输出列表里的每一列就代表一个封包。举例来说：假设要产生一个地址为 41h、地址写入、8 位数据宽度然后数据为 20h 的 HDQ 封包。操作步骤如下：1.按下 HDQ8 按钮(选择 8 位数据宽度，此为程序默认项目)2.分别在地址与数据填入 41h 和 20h，勾选写入 3.按下工具盒中的『新增项目到信号列表』请参考下图。

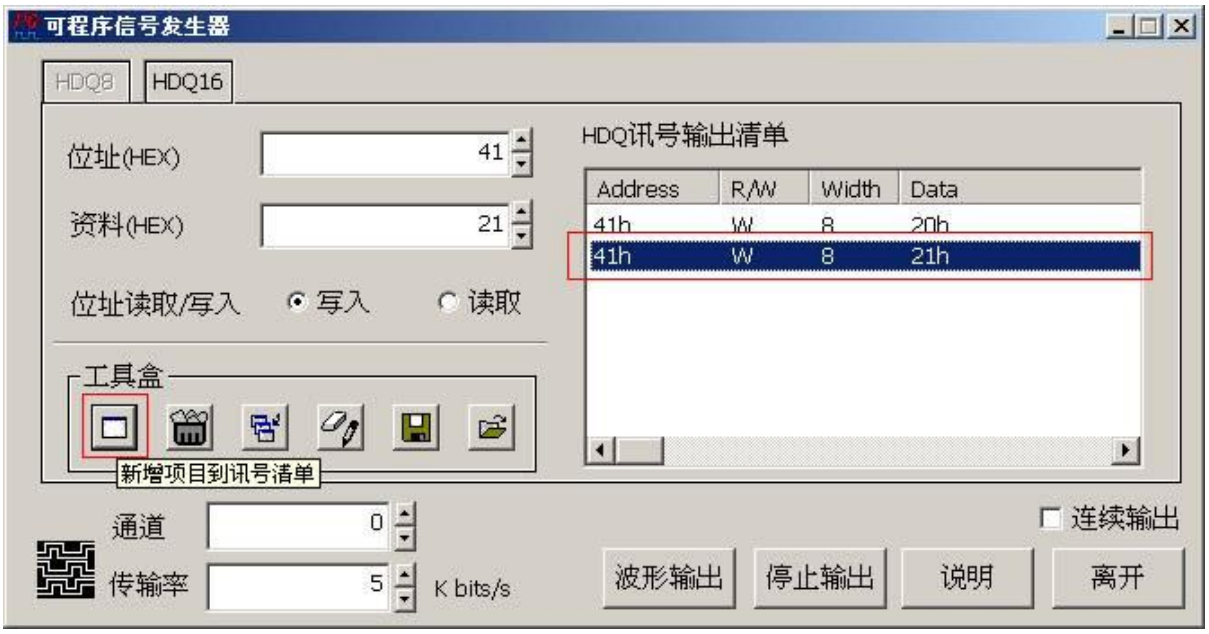



『新增项目到信号列表』会将您的信号封包增加在现有封包之后.。但是如果想在相同地址之下，增加第二个信号呢？

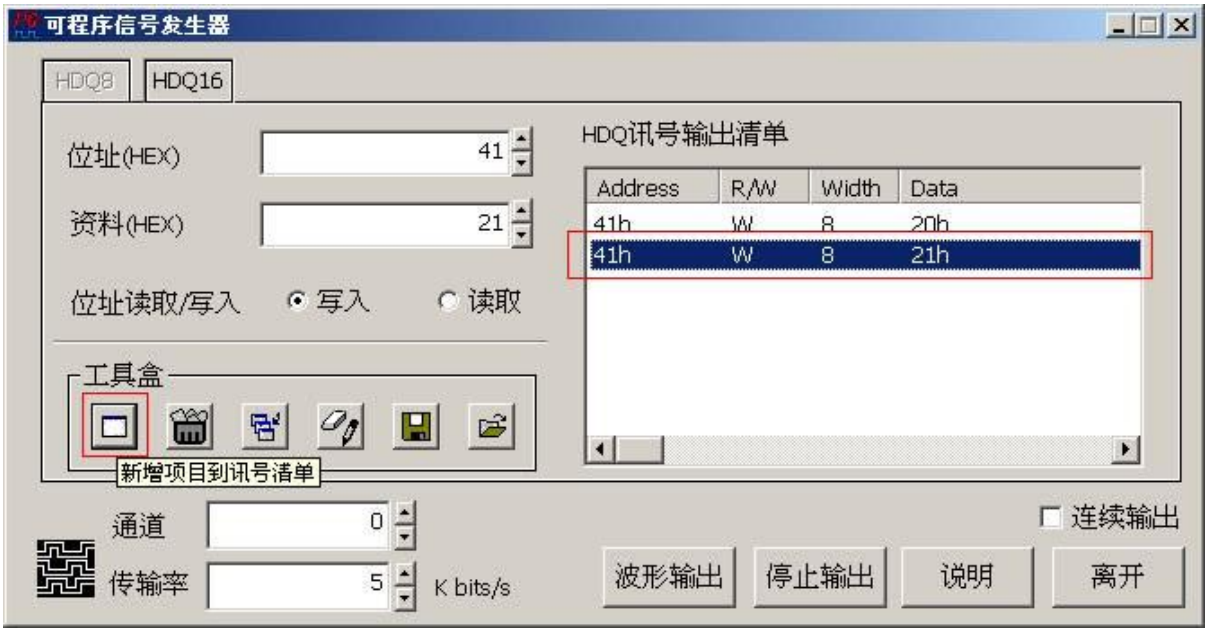
『新增项目到信号列表』功能会根据上一个封包的地址、R/W、和数据宽度来判断是否将该资料附加至上一个封包之后。只有在地址、R/W、和数据宽度和上一个封包相同时，程序会询问用户是否在该数据附加之上一个封包之后？

如果选择“是”则会在数据附加至上一个封包之后，若选择“否”则会增加一个新封包。请参考下图：


选择“**是**”的情形：

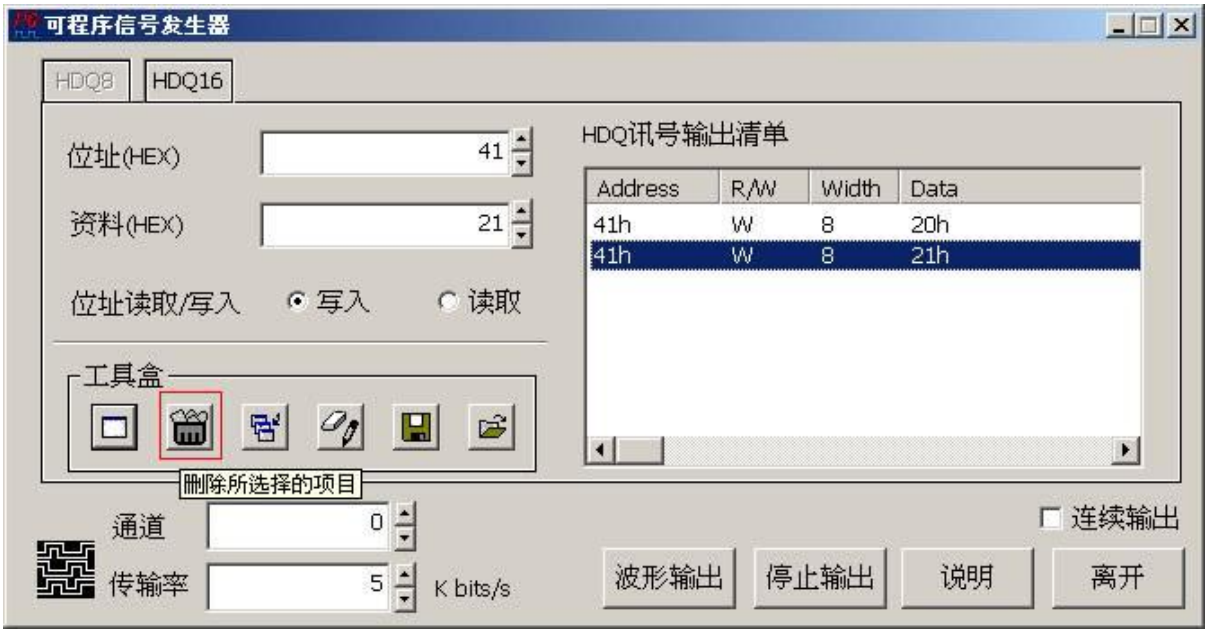



选择“否”的情形：




『删除所选择的项目』功能可以将在『**HDQ** 信号输出列表』所选择的项目删除掉，

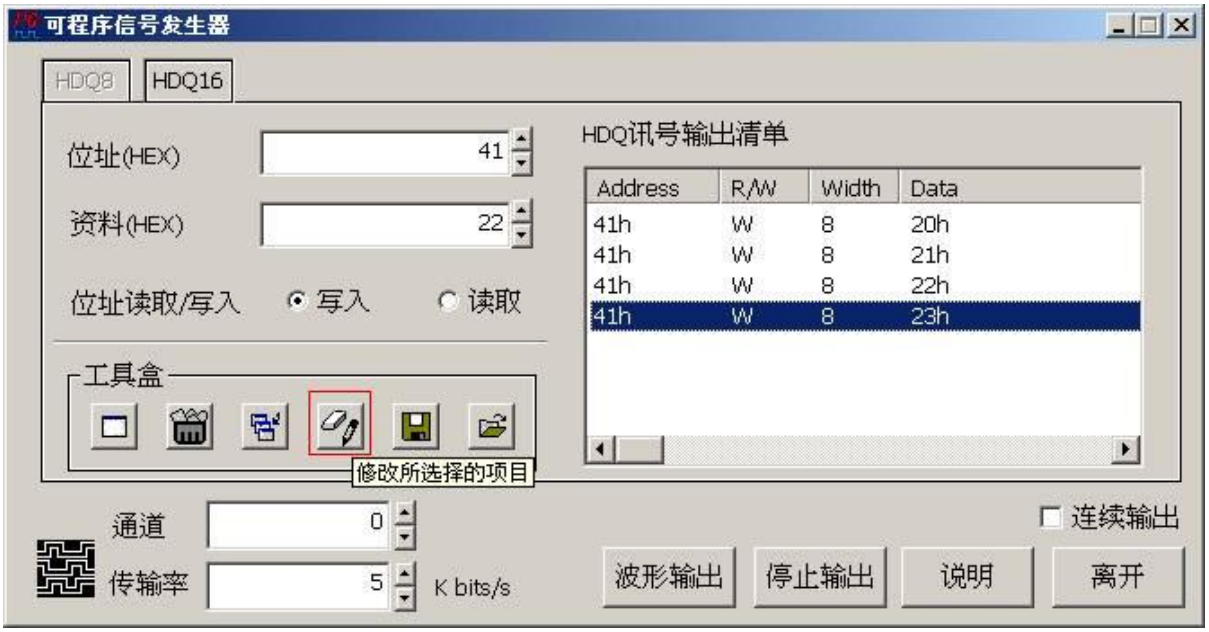
只要按下该按钮，就可以将目前选择的该封包删除。请参考下图：




『插入新项目到所选择的项目之后』功能以将在『**HDQ** 信号输出列表』所选择的项目之后插入一个新封包。请参考下图：


按下该按钮之后，会将地址 41h、地址写入、8 位数据宽度然后数据为 22h 的 **HDQ** 封包插入至所选择封包的下一个项目。请参考下图：


请注意:『插入新项目到所选择的项目之后』功能并无根据上一个封包的地址、R/W、和数据宽度来判断是否将该数据附加至上一个封包之后的功能。『修改所选择的项目』功能可以修改所选择封包的地址、R/W、数据位宽度和数据值，请参考下图：




然后按下该钮，程序会出现一个可以让您修改的对话盒，该对话盒会显示目前您所选封包之参数值，请参考下图：


假设要将所选择的封包修改为地址 5Ah，数据为 24h、25h、26h，改为地址读取然后将数据宽度改成勾选 HDQ16(16 位数据宽度)，请参考下图：

请注意：数据和数据之间请输入一个空格键(space)。结果如下，请参考下图：


『导出信号列表』功能可以将您编辑好的信号列表存成档案，之后可以使用『汇入信号列表』功能将该档案读回，就不用再重新编辑信号列表了。请参考下图：

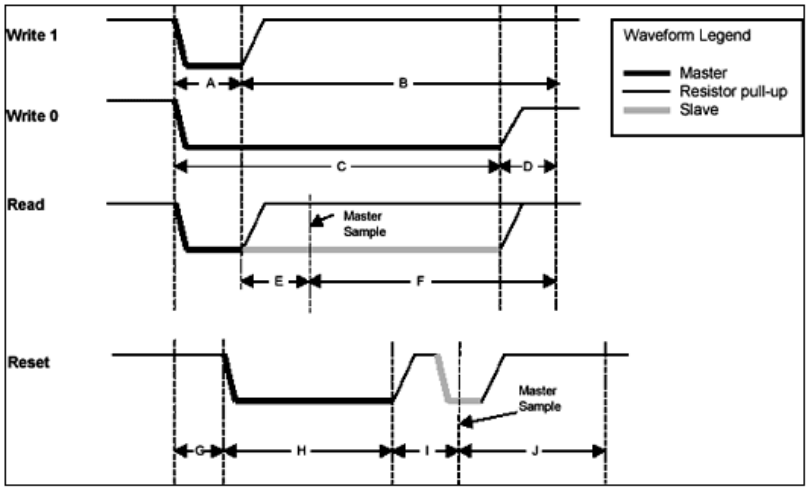

『信道』可以让您选择数据产生器的输出信道，0 表示选择信道 0 来输出 **HDQ** 信号，1 表示选择信道 1 来输出，依此类推。

『传输率』根据 **HDQ** 协议，输入范围为 3.5K bits/s ~ 5.5K bits/s 之间，预设为 5K bits/s。

『连续输出』则是选择信号是输出一次，还是重复输出。

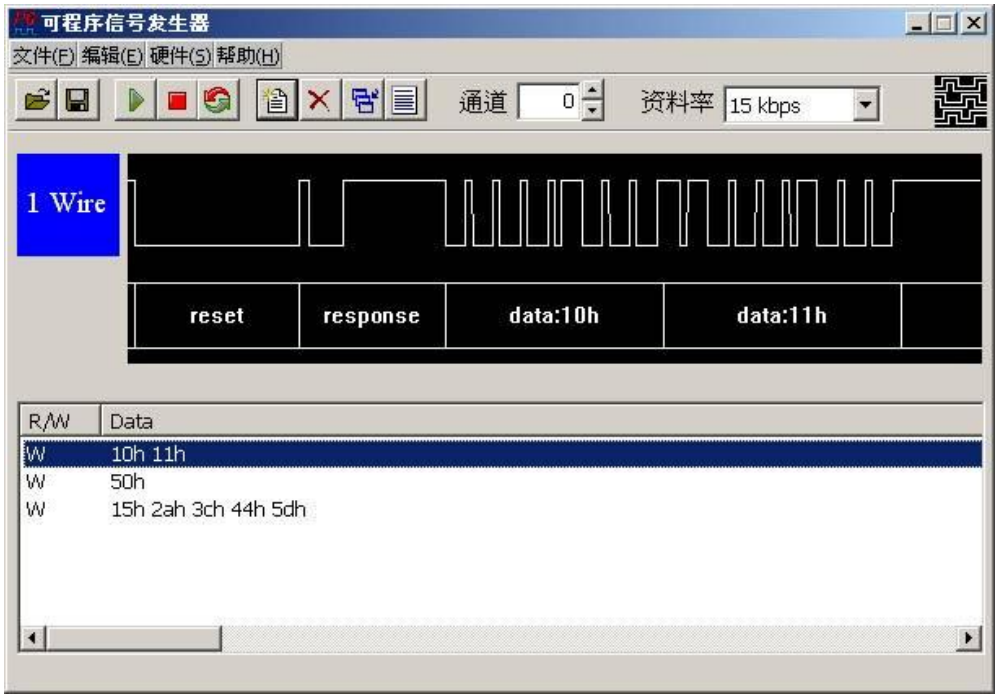
# 19. 1-Wire 信号发生器

1-Wire 协议是由美国达拉斯公司(Dallas Semiconductor)所制定。1-Wire 协议定义了 Reset Pulse, Presence Pulse, Write 1, Write 0, Read 1, Read 0 等几种信号类型并由这些信号类型组合成命令序列。通常应用于 EEPROM 的通信方面。




传输的方式为 LSB (Least-significant bit)到 MSB (Most-significant bit)，传输的速度分为高速(Overdrive speed)和低速(Standard speed)。

## 1-Wire 信号发生器的使用方法




1-Wire 信号发生器的使用方法主要是根据您的信号需求，来编辑『1-Wire 输出列表』

让数据产生器依照您的信号列表来输出该信号，输出列表里的每一列就代表一个封包。

**(1) 档案**

**a. 清单导出**

将『**1-Wire** 输出清单』储存成文本文件。

**b. 清单汇入**

读取所储存的文本文件。

**(2) 编辑**

**a. 新增封包**

新增一个 **1-Wire** 的封包，它会附加至清单内现有封包之后。

**b. 删除所选取封包**

删除在清单内被选取的封包。

**c. 插入新封包到所选择的封包之后**

插入一个新封包在清单内被选取的封包之后。

**d. 修改所选择封包**

可以修改在清单内所选择的封包。

**e. 删除所有封包**

可以删除在清单内所有的封包。


注意：在编辑封包时，请留意数据的编辑格式，每一个数据之间必须有一个空白(space)间隔，接受的数据为十六进制值，请参考上图。

**(3) 硬件**

**a. 波形输出**

将清单内的封包输出一次。

**b. 停止输出**

停止 **1-Wire** 封包输出。

**c. 重复输出**

将清单内的封包重复输出。

**(4) 帮助**

**a. 内容**

开启关于 **1-Wire** 信号发生器的在线使用说明。

**b. 通道**

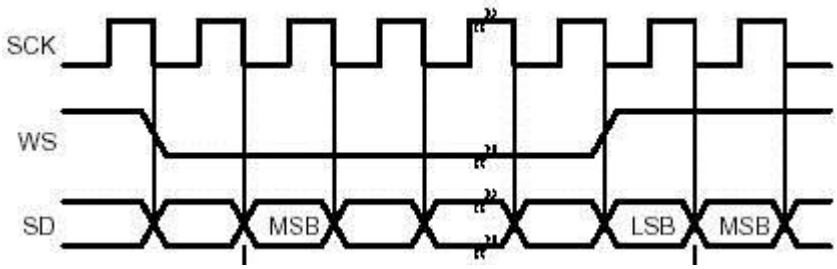
选择 **1-Wire** 信号输出的信道。

**c. 资料率**

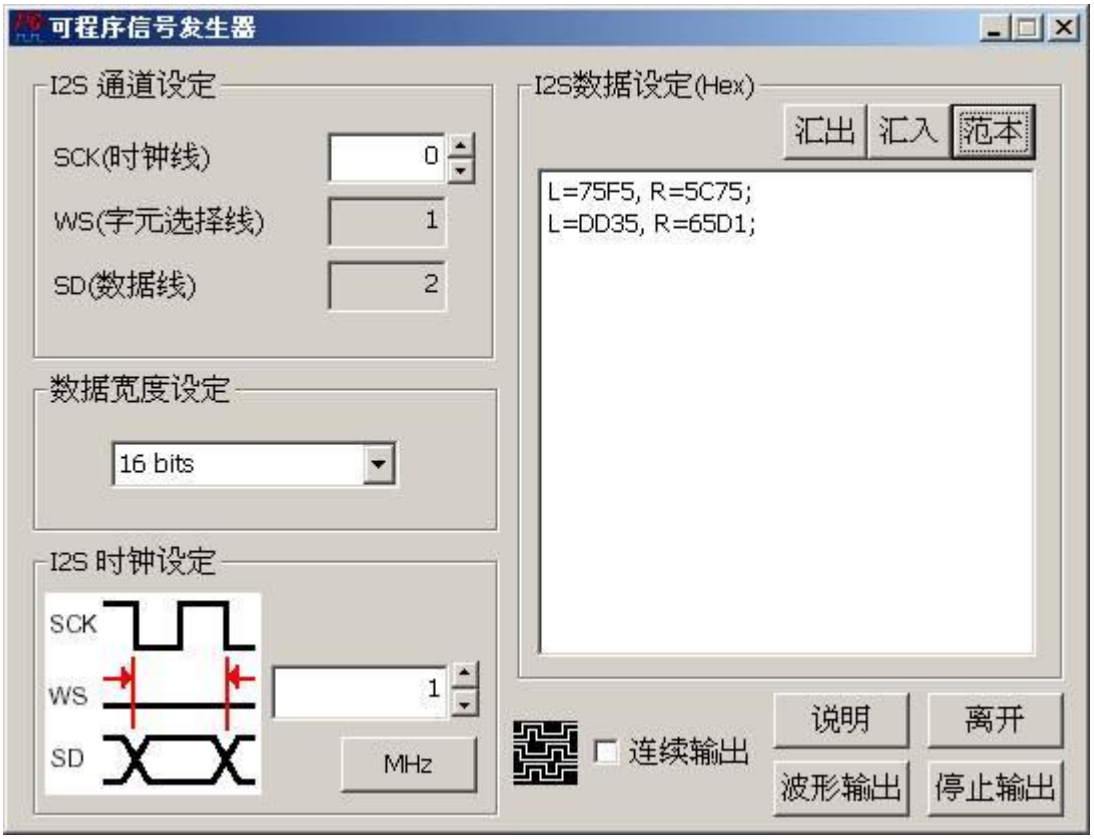
本程序支持低速 15kbps 和高速 111kbps。

# 20. I²S 信号发生器

I²S(Inter-IC Sound 或 Integrated Interchip Sound)是 IC 间传输数字音频数据的一种接口标准，是飞利浦公司为数字音频设备之间的音频数据传输而制定的一种总线标准，常被使用在传送 CD 的 PCM 音讯到 CD 播放器中的 DAC 上。在 I²S 的标准中，规定了硬件接口规范及数字音频数据的格式，采用序列的方式传输 2 组(左右声道)数据。由三条传输线组成，一条是频率线(SCK)、另一条是字符选择线(WS)以及数据线(SD)。数据格式最大到 20Bits。




## I²S 信号发生器的使用方法




I²S 信号发生器的使用方法主要是根据您的信号需求，来编辑 I²S 数据设定让数据产

生器依照您的信号列表来输出该信号，输出列表里的每一列就代表一个封包。

**(1) I²S 通道设定**

设定数据产生器信号输出的信道，默认 SCK 为信道 0，WS 为信道 1，SD 为信道 2。

**(2) 数据宽度设定**

设定 I²S 数据宽度，有 16bits、20bits、24bits 可选择。

**(3) I²S 频率设定**

设定 I²S 频率。

**(4) I²S 数据设定**

编辑 I²S 信号数据内容 L = 75F5, R=5C75;表示第一笔的左信道数据为 0x75F5，右信道数据为 0x5C75。

L = DD35, R=65D1;表示第二笔的左信道数据为 0xDD35，右信道数据为 0x65D1。

编辑 I²S 信号数据时请按照上述编辑格式以”,”分隔左右信道数据，然后以”;”作为结尾。

**a. 汇出**

将编辑的数据储存成文本文件(.txt)。

**b. 汇入**

开启储存的文本文件。

**c. 范本**

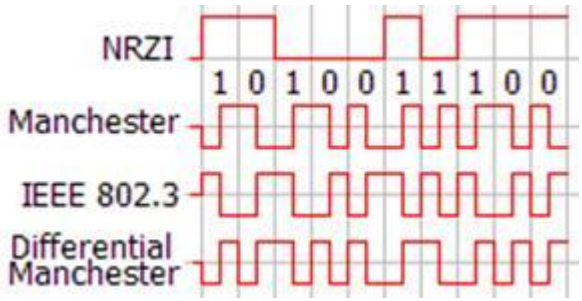
产生 I²S 信号数据的模板。

**(5) 连续输出**

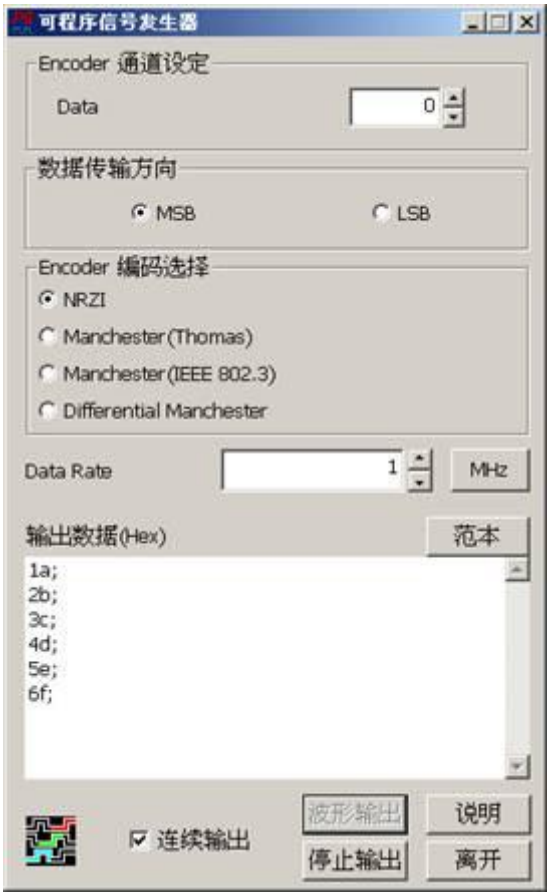
连续输出所编辑的 I²S 信号。注：I²S 信号发生器输出的 I²S 信号为 I²S 格式。

# 21. 编码器

Encode Encode Encode 编码器有四种编码选择，分别是 NRZI, Manchester(Thomas), Manchester(IEEE 802.3), Differential Manchester。




## 编码器的使用方法




### (1) Encoder 通道设定

选择输出的通道。

### (2) 资料方向

选择所要输出的数据方向为 MSB 或是 LSB。例如 16 进制值 1a，MSB =>

00011010, LSB => 01011000。

**(3) Encoder 编码选择**

可以选择 NRZI、Manchester(Thomas)、Manchester(IEEE 802.3)、Differential Manchester。

**(4) Data Rate**

产生 1 bit 数据的时间。

**(5) 范本**

产生范例数据，此为输出数据之正确格式(为 16 进制数值，一列一个并以分号隔开)。

# 22. USB1.1 信号发生器

USB (Universal Serial Bus)称为”万用串行总线” ,起初由7家公司所制订的规格：Intel, Microsoft, National Semiconductor, Compaq, Nortel, NEC and AT&T。

### 标准 USB 接口

| Pin | Name | Cable color | Description |
|---|---|---|---|
| 1 | VCC | Red | +5V |
| 2 | D– | White | Data – |
| 3 | D+ | Green | Data + |
| 4 | GND | Black | Ground |

### USB1.1 信号发生器的使用方法


USB1.1 信号发生器的使用方法主要是根据您的信号需求，来编辑 USB1.1 输出封包让数据产生器依照您的信号列表来输出该信号，输出列表里的每一列就代表一个封包。

**(1) USB1.1 通道设定**

主要是选定数据产生器的信道来代表 USB1.1 的 D+和 D-接脚。

**(2) USB1.1 频率设定**

设定 USB1.1 频率，预设为 12MHz。

**(3) USB1.1 输出封包**

为 USB1.1 信号封包列表，输出的顺序为由上而下。透过底下的窗体(在列表区按下鼠标右键)来编辑该输出封包清单：


**a. 新增封包**

新增一个封包到 USB1.1 输出封包清单。

**b. 删除封包**

删除所选择的封包。

**c. 插入封包**

新增一个封包到所选择的封包之后。

**d. 修改封包**

修改所选择的封包内容。

**e. 范例**

显示一个 USB1.1 输出封包列表范例。

**f. 删除所有封包**

删除 USB1.1 清单中所有封包。

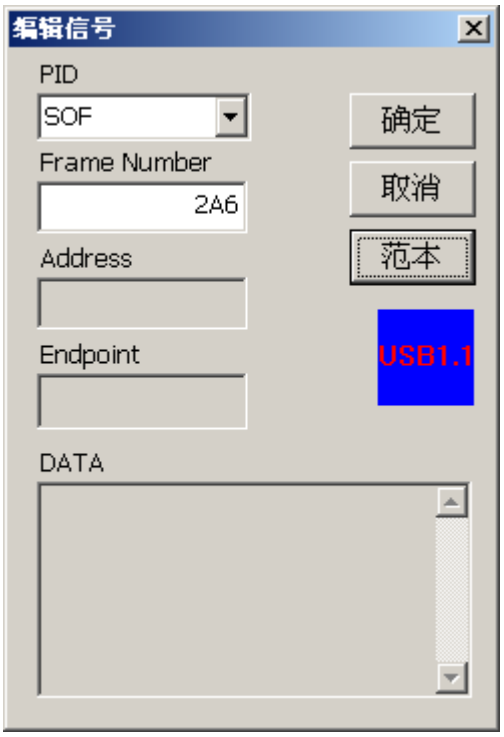
**g. 储存档案**

将清单内容储存成一个文本文件。

**h. 开启档案**

开启所储存的清单内容的档案。

- 新增封包、插入封包和修改封包都会出现底下的编辑封包的对话盒：




- 输入封包的内容值，程序会自动计算出 CRC5 或 CRC16 的值并且根据所选择的 PID 来组成 USB 的 Token, Data, Handshake 封包。

**(4) 连续输出**

会重复输出 USB1.1 封包清单的内容。

**注:所有输入的数值都以十六进制表示。**

# 23. SMBus 信号发生器

SMBus SMBus SMBus SMBus 全名系统管理总线（System Management Bus）源自于 $I^2C$ 总线，是一种两条信号所组成的一种总线。SMBus 由 Intel 于 1995 年所定义。包含有 clock, data, 以及基于 Philips' I²C serial bus 协议的指令。其时钟频率范围在 10 kHz 到 100 kHz。

## SMBus 信号发生器的使用方法


SMBus 信号发生器的使用方法主要是根据您的信号需求，来编辑 SMBus 输出封包让数据产生器依照您的信号列表来输出该信号，输出列表里的每一列就代表一个封包。

### (1) SMBus 封包内容设定

SMBus 的封包种类共有 21 种，分别是 Quick command protocol, Send byte protocol, Send byte protocol with PEC, Receive byte protocol, Receive byte protocol with PEC, Write byte protocol, Write Word Protocol, Write byte protocol with PEC, Write Word Protocol with PEC, Read Byte Protocol, Read

byte protocol with PEC, Read word protocol, Read word protocol with PEC, Process Call, Process Call with PEC, Block Write, Block Write with PEC, Block Read, Block Read with PEC, Block Write - Block Read Process Call 和 Block Write - Block Read Process Call with PEC 封包种类的说明请自行参考 SMBus 规格(version 2.0)。首先选择封包种类，程序会根据所选择的封包种类对 Address, Command, Data 和 Sr (Repeated Start)做相对应的回应。假设选择的封包种类是 Process Call with PEC，该种类的封包内容如下：

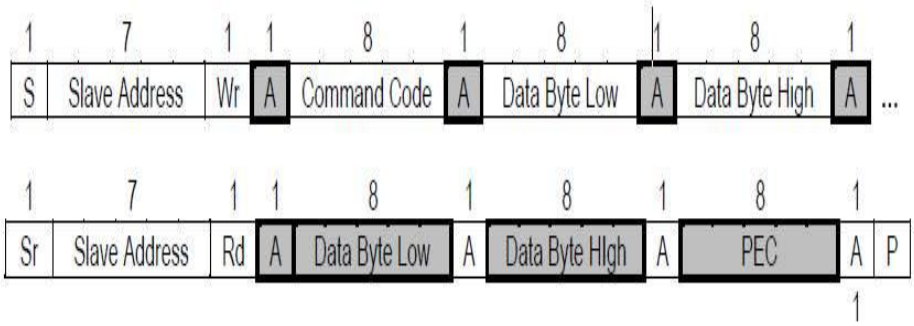
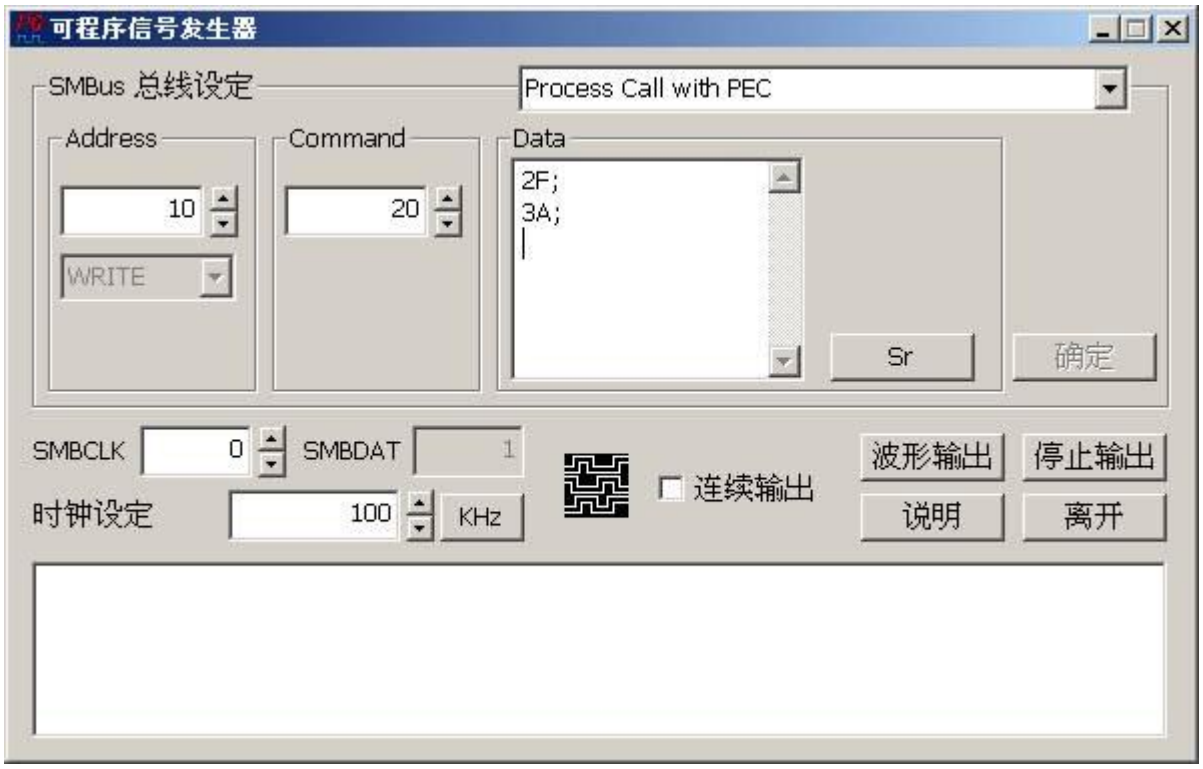
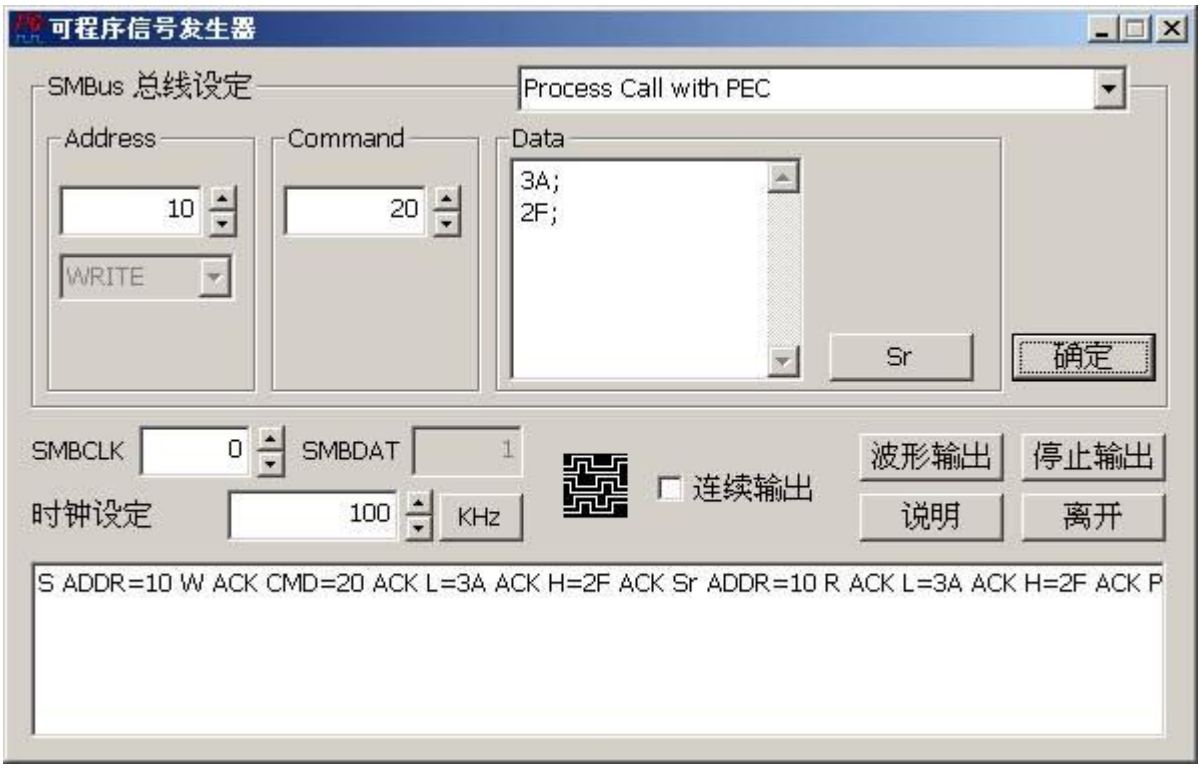



Figure 5-16: Process Call with PEC

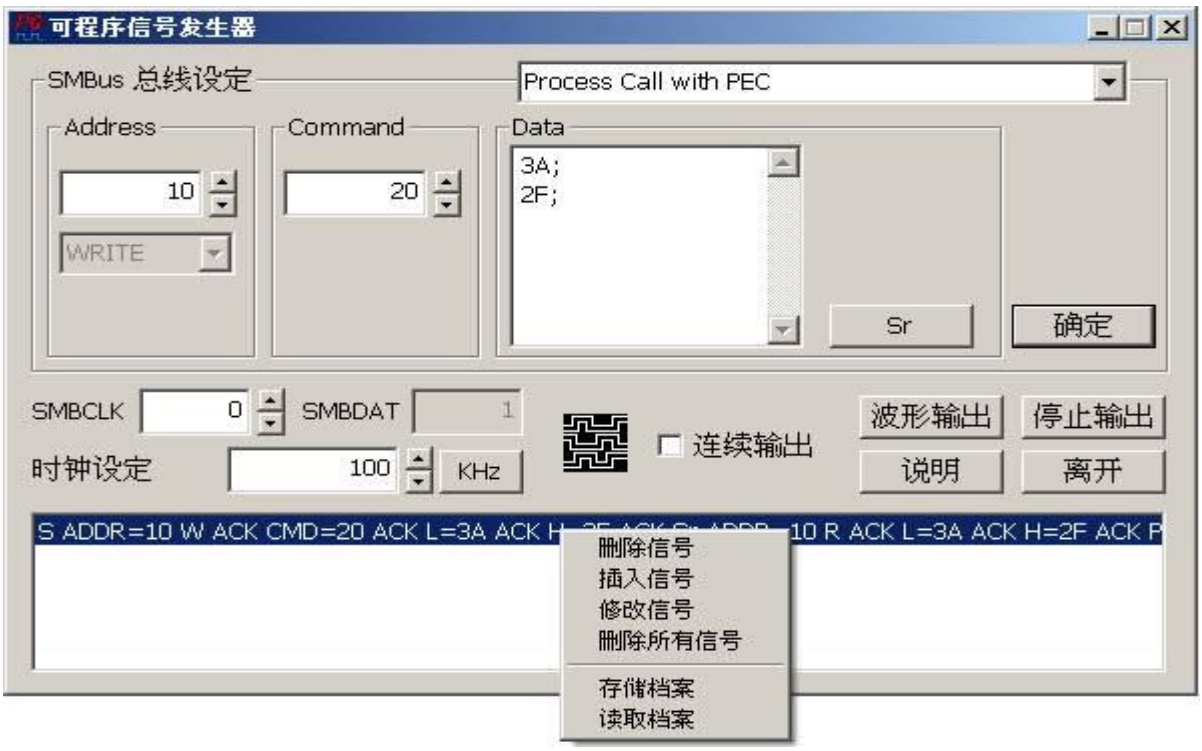
**(2)** 根据封包内容输入 Address, Command Code, Data 和 Sr 然后按下确定。做法如下：



Address 和 Command 使用默认值 0x10 以及 0x20，Data 输入 2F 和 3A(皆为十六进制值)。2F 为 Data Byte Low 的数值，3A 为 Data Byte High 的数值，数值之后必须接一个分号";"如上图所示。然后根据 Process Call with PEC 格式，接着按下 Sr 按键产生一个 Repeated Start，然后输入 slave 响应 master 的数据(例如：Data Byte Low: 3A, Data Byte High: 2F)，再按下确定即完成一个 Process Call with PEC 种类的封包，而封包中的 ACK 或是 NACK，READ/WRITE 或是 PEC，程序会自动加入。结果如下：




上图列表中的 S 表示 Start, ADDR 表示 Address, CMD 表示 Command, W 表示 WRITE，R 表示 READ，L 表示 Data Byte Low, H 表示 Data Byte High。该封包后面看不见的部份，您可以按下鼠标右键，选择修改封包即可。





**a. 删除封包**

将选择的封包删除。

**b. 插入封包**

插入一个新的封包到所选择封包的下一个(顺序由上到下)。

**c. 删除所有封包**

删除列表中所有封包。

**d. 储存档案**

将列表中的内容存成文本文件。

**e. 开启档案**

开启所储存的封包内容文本文件。

**(3) 波形输出**

根据列表中的内容输出波形。

**(4) 停止输出**

停止输出波形。

**(5) 说明**

显示该工具程序的说明文件。

**(6) 离开**

离开该程序。

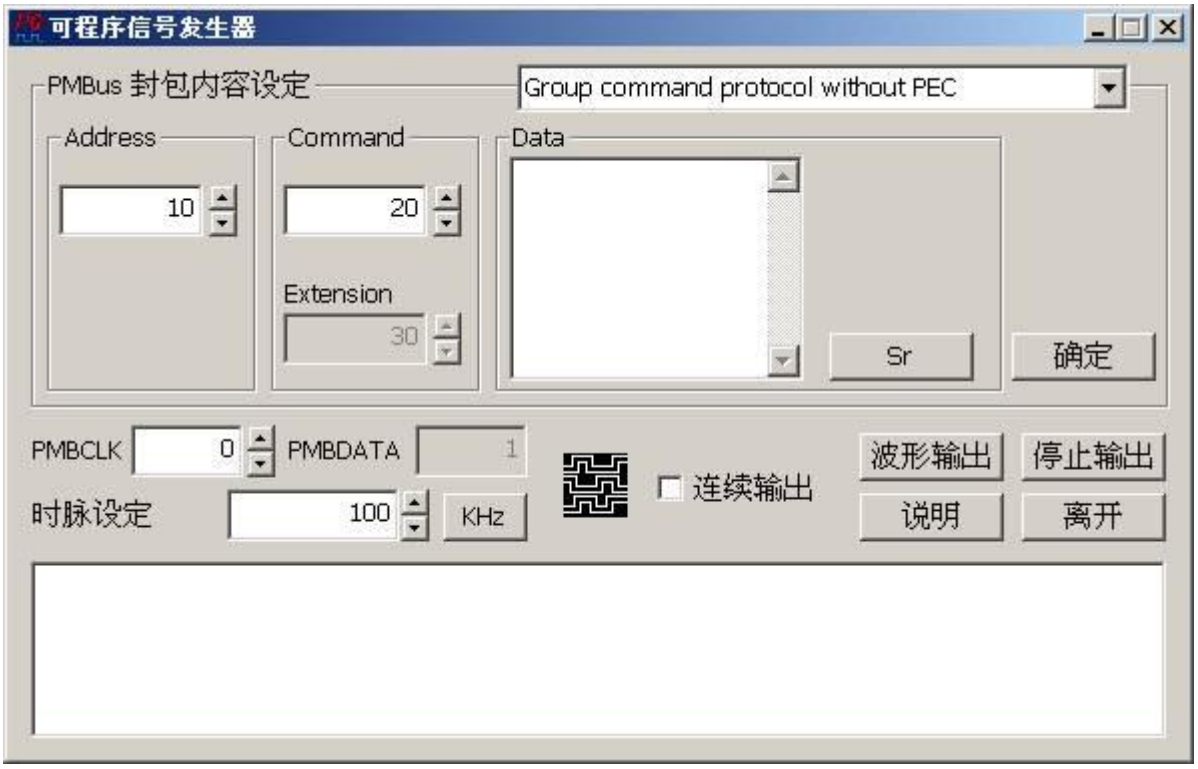
**(7) 连续输出**

连续输出列表中的封包内容。

# 24. PMBus 信号发生器

PMBus PMBus 全名电源管理总线（Power Management Bus）是一种两条信号所组成的一种总线。包含有 clock, data。和 SMBus 相同皆源自于 $I^2C$ 总线，而 PMBus 最大传输速度为 400KHz 高于 SMBus 的 100KHz。

## PMBus 信号发生器的使用方法




PMBus 信号发生器的使用方法主要是根据您的信号需求，来编辑 PMBus 输出封包让数据产生器依照您的信号列表来输出该信号，输出列表里的每一列就代表一个封包。

### (1) PMBus 封包内容设定

PMBus 的封包种类共有 10 种，分别是 Group command protocol without PEC, Group command protocol with PEC, Extended command read byte protocol, Extended command read byte protocol with PEC, Extended command write byte protocol, Extended command write byte protocol with PEC, Extended command

read word protocol, Extended command read word protocol with PEC, Extended command write word protocol 和 Extended command write word protocol with PEC。封包种类的说明请自行参考 PMBus 规格(Revision 1.1)。首先选择封包种类，程序会根据所选择的封包种类对 Address, Command, Data 和 Sr (Repeated Start)做相对应的回应。假设选择的封包种类是 Group command protocol with PEC，该种类的封包内容如下：

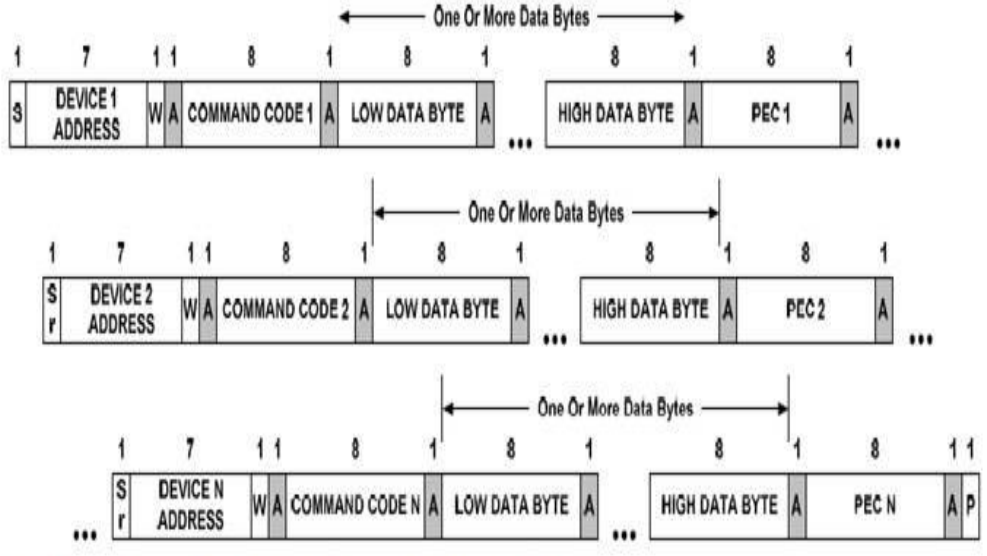
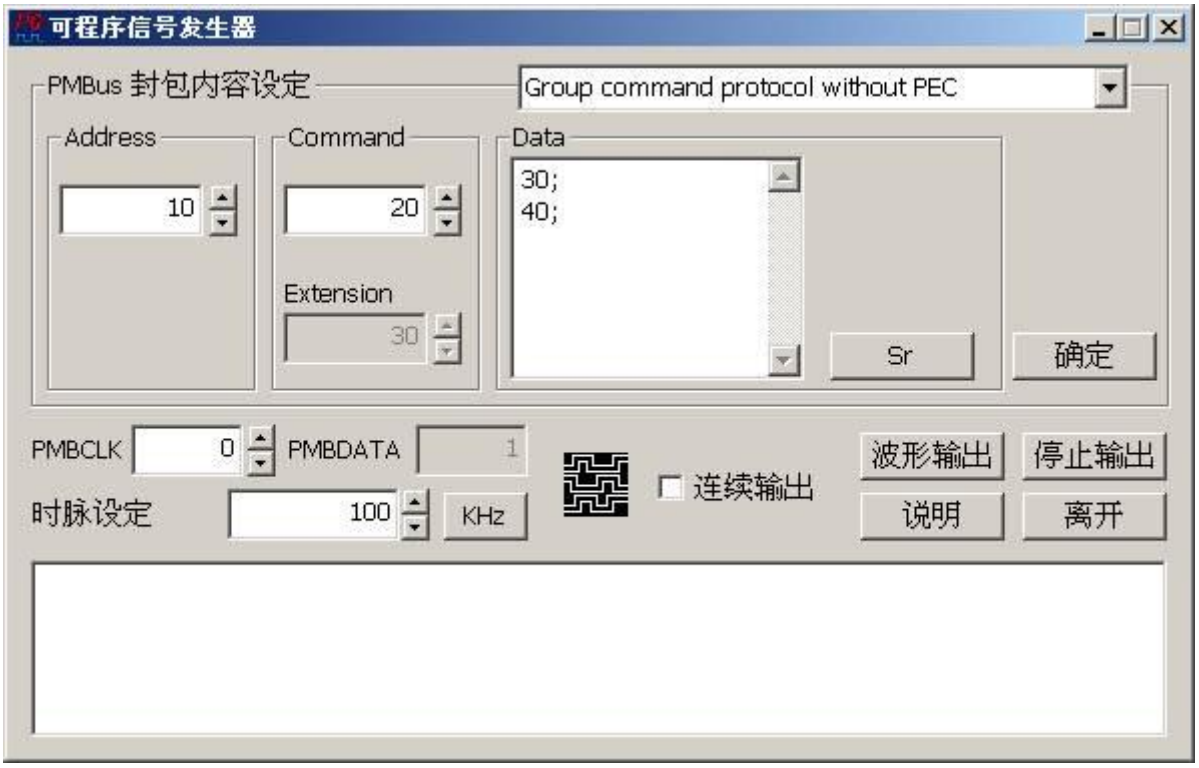



Figure 3. Group Command Protocol With PEC

**(2)** 根据封包内容输入 Address, Command , Data 和 Sr 然后按下确定。做法如下：




Address 和 Command 使用默认值 0x10 以及 0x20，Data 输入 30 和 40(皆为十六进制值)。30 为 Data Byte Low 的数值，40 为 Data Byte High 的数值，数值之后必须接一个分号";" 如上图所示。然后根据 Group command protocol with PEC 格式，接着按下 Sr 按键产生一个 Repeated Start，然后输入 slave 响应 master 的数据(例如：Data Byte Low: 50, Data Byte High: 60)，再按下确定即完成一个 Group command protocol with PEC 种类的封包，而封包中的 ACK 或是 NACK，READ/WRITE 或是 PEC，程序会自动加入。结果如下：


上图列表中的 S 表示 Start, ADDR 表示 Address, CMD 表示 Command, W 表示 WRITE，R 表示 READ，L 表示 Data Byte Low, H 表示 Data Byte High。

该封包后面看不见的部份，您可以按下鼠标右键，选择修改封包即可。


**a. 删除封包**

将选择的封包删除。

**b. 插入封包**

插入一个新的封包到所选择封包的下一个(顺序由上到下)。

**c. 删除所有封包**

删除列表中所有封包。

**d. 储存档案**

将列表中的内容存成文本文件。

**e. 开启档案**

开启所储存的封包内容文本文件。

**(3) 波形输出**

根据列表中的内容输出波形。

**(4) 停止输出**

停止输出波形。

**(5) 说明**

显示该工具程序的说明文件。

**(6) 离开**

离开该程序。

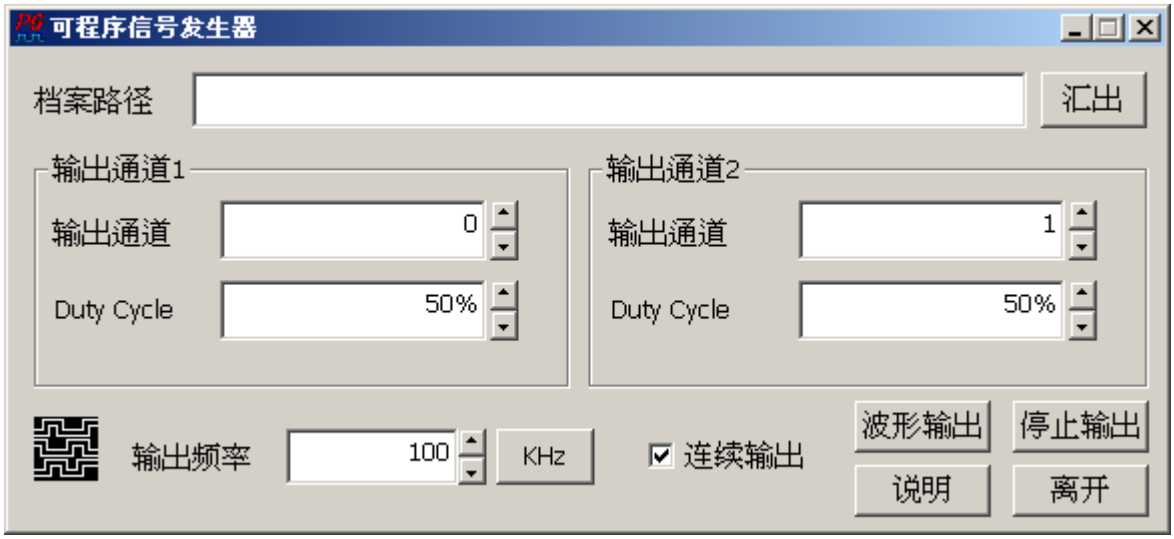
**(7) 连续输出**

连续输出列表中的封包内容。

# 25. PWM 信号发生器

PWM (Pulse Width Modulation)，称为脉宽调变，它不是一种总线分析协议。主要是利用脉冲宽度之周期对模拟电路进行控制的一种非常有效的技术，广泛应用在一些转速控制、亮度控制和温度控制等。

## PWM 信号发生器的使用方法




PWM 信号发生器可以同时输出 2 个不同 Duty Cycle 的 PWM 信号，还可以将设定好的 PWM 信号存储成文字向量文档 (*.PGV)。

**(1) PWM 通道设定**

主要是设定 PWM 的输出通道 1/2 以及该 PWM 信号的 Duty Cycle。

**(2) PWM 输出频率**

设定 PWM 频率，预设为 100KHz。

**(3) 汇出档案**

将设定的 PWM 波形汇出为文字向量文档(.*PGV)。

**(4) 波形输出**

根据列表中的内容输出波形。

**(5) 停止输出**

停止输出波形。

**(6) 说明**

显示该工具程序的说明文件。

**(7) 离开**

离开该程序。

**(8) 连续输出**

连续输出列表中的封包内容。

# 26. VCD 波形档转换

Value change dump，简称 VCD。用于记录由 EDA 仿真工具所产生的信号信息。


## VCD 波形档转换的使用方法

VCD 波形档转换可以将 VCD 档案的内容转换成 PG 的波形。

**(1) 浏览**

选择 VCD 档案。

**(2) 载入**

载入 VCD 档案。

**(3) 取消载入**

取消载入 VCD 档案。

**(4) 重复输出**

连续输出列表中的封包内容。

**(5) 按键后才输出波形**

在转换成 PG 波形的前端加入 Keyboard event (space)。

**(6) 波形转换**

转换成 PG 波形。

**(7) 取消**

离开该程序。

**(8) 说明**

显示该工具程序的说明文件。

# 第4章 注意事項

# 1. 硬件注意事项

**(1)** 将 Acute PG 与标的物连接时，请注意不要把输出端接到 Ground 或 VCC 上。因为 Acute PG 是一种输出装置，如果将输出端与标的物的 Ground 或 VCC 接在一起，会造成 Acute PG 的前端隔离放大器的损坏。

**(2)** 请不要使用非本公司提供的整流稳压器(Power Adapter)，来连接 PG1000/PG2000 的电源接头。本公司提供的整流稳压器是一个符合 PG1000/PG2000 电源需求的电源供应器，他的稳压与功率都是经过计算的。如果使用自行购买的整流稳压器，可能造成 PG1000/PG2000 工作不正常或是零件破坏。

**(3)** Pocket PG 是可以调整输出电压值所以调整电压时，请注意标的物电压容许值，否则将影响或破坏该标的物。

**(4)** 每个隔离放大器及 Pocket PG 的地线至少要接一支到标的物上，如量测出的波形信号干扰太大时，就多接一支地线到标的物上以改善量测的质量。

**(5)** 隔离放大器虽然有标示 A、B、C、D、E，但是每个隔离放大器都是一样的，所以任何一个隔离放大器都可以对调。可是建议不要乱接，避免通道编号与 PC 端的对应，产生不必要的错误。但是 Extended Pod 是特殊的隔离放大器，请不要将它接到 A、B、C、D、E 等 Pod 的接头上，同样的 A、B、C、D、E 的 Pod 也不能接到 Extended Pod 的接头上，否则会对 PG1000/PG2000 主机造成伤害。

**(6)** PG1000/PG2000 的主机与界面卡的接头和 Printer 所使用的 Cable 相同，但是不要将 PG1000/PG2000 与 Printer 同时并联接在一起，否则将有不可预期的后果。而且请注意，不要使用自行购买的 Printer Cable 来连接 PG1000/PG2000 和 PC，请使用本公司提供的连接 Cable，如此我们才能保证 PG1000/PG2000 的工作正常。

**(7)** Pocket PG 使用的是 USB 传输界面，所以必需用 USB Cable 连接 Pocket PG 与 PC 主机，使用 USB Cable 时请不要让接头部份摇晃，否则连接器一旦松动，将造成 Pocket PG 无法正常工作。

**(8)** 将 PG1000/PG2000 主机安装在 PC 的磁盘驱动器插槽时，尽量放在下方的位置，因为 PG1000/PG2000 的面板会接上排线，所以可能会挡住放在下方的磁盘驱动器或光驱，造成使用上的不便。

**(9)** 标的物的测试脚，如果用 Acute PG 所提供的探头，仍很难正常的接在标的物上时，可以向本公司选购较细的探头，或是测试夹等配备，或是自行购买合适的探头或测试夹。

# 2. 软件注意事项

**(1)** 软件程序或是驱动程序安装的顺序不会影响正常工作，但是要执行软件之前，必须已经装好硬件驱动程序才能正常工作。

**(2)** 安装新版软件时，请先将旧版软件移除。

**(3)** 请至网站上查询新软件版本，自行更新。http://www.acute.com.tw

# 第5章 其它

# 1. 故障排除

**(1)** 如果执行 PG Editor 时出现『Demo Mode』，就代表您的安装有问题，请按下列步骤处理：

- **PCI 与 EPP 界面**

  **a.** 检查电源接头是否接好。

  **b.** 检查传输界面的接头是否接好。

  **c.** 如果您使用内接式，请至控制台-系统-装置管理员-系统装置中寻找是否有一个 LP1000 的装置。如果没有就代表界面卡没有接好，请关闭 PC 电源并将界面卡重新插入 PCI 插槽或换另一个插槽。

  **d.** 如果您使用外接式 ,请检查 PC BIOS 的 Printer 设定是否已经设成 EPP 模式(请参考『Printer Port 的 BIOS 模式设定』)。

  **e.** 做完上述检查后，请重新新启动 PG-Editor 程序。

  **f.** 如果仍然无法正常使用本产品，请与我们联络，让我们为您服务。

- **USB 界面**

  **a.** 至装置管理员中，检查驱动程序是否存在。(参考硬件安装一节)

  **b.** 如果驱动程序存在的话，请确认是否已安装最新版本的 PG-Editor。

  **c.** 如果已安装最新版本的 PG-Editor，请重新启动 Windows 操作系统。

  **d.** 如果未安装最新版本的 PG-Editor，请至 Acute 网站 http://www.acute.com.tw 上 Download 并重新安装。

  **e.** 如果驱动程序不存在时，请重新插拔 USB Cable，并检查驱动程序是否已出现。

  **f.** 驱动程序如果仍然不存在时，请重新开机，并再次检查驱动程序是否存在。

  **g.** 七、经过第六步骤后，驱动程序还是不存在，此时请注意 USB Cable 插拔时，是否新增一个其它的装置，而并非是 Acute PG 的驱动程序，如果是这样，请与本公司联络。

**(2)** 输出波形时，如果信号未正常输出，请按下列步骤处理：

**a.** 检查探头是否正常接在信号连接在线。

**b.** 检查信号连接线是否正常接在隔离放大器或 Pocket PG 上。

**c.** 检查探头是否与标的物正常连接。

**d.** 检查标的物是否有信号产生(可将有信号产生的信道与有问题的信道对调来确定问题所在)。

**e.** 再重新输出波形一次。

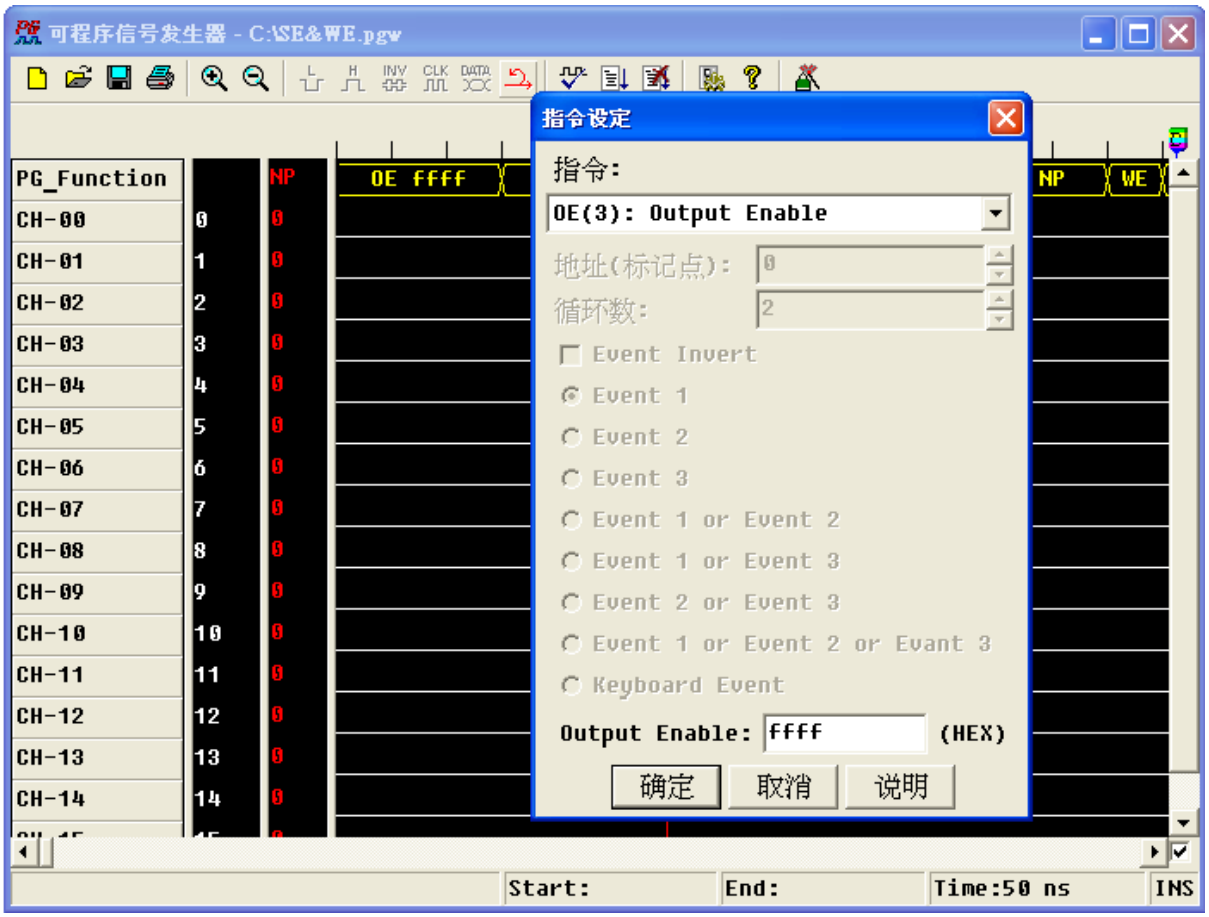
**(3)** 当使用波形编辑器时，如果波形无法如预期的结果时，请按波形检查钮，看看编辑波形有没有错误。或是按环境设定钮，检查基本频率的设定有没有设错。

# 2. PG_Function 指令使用方法

**(1)** Output Enable OE(3):

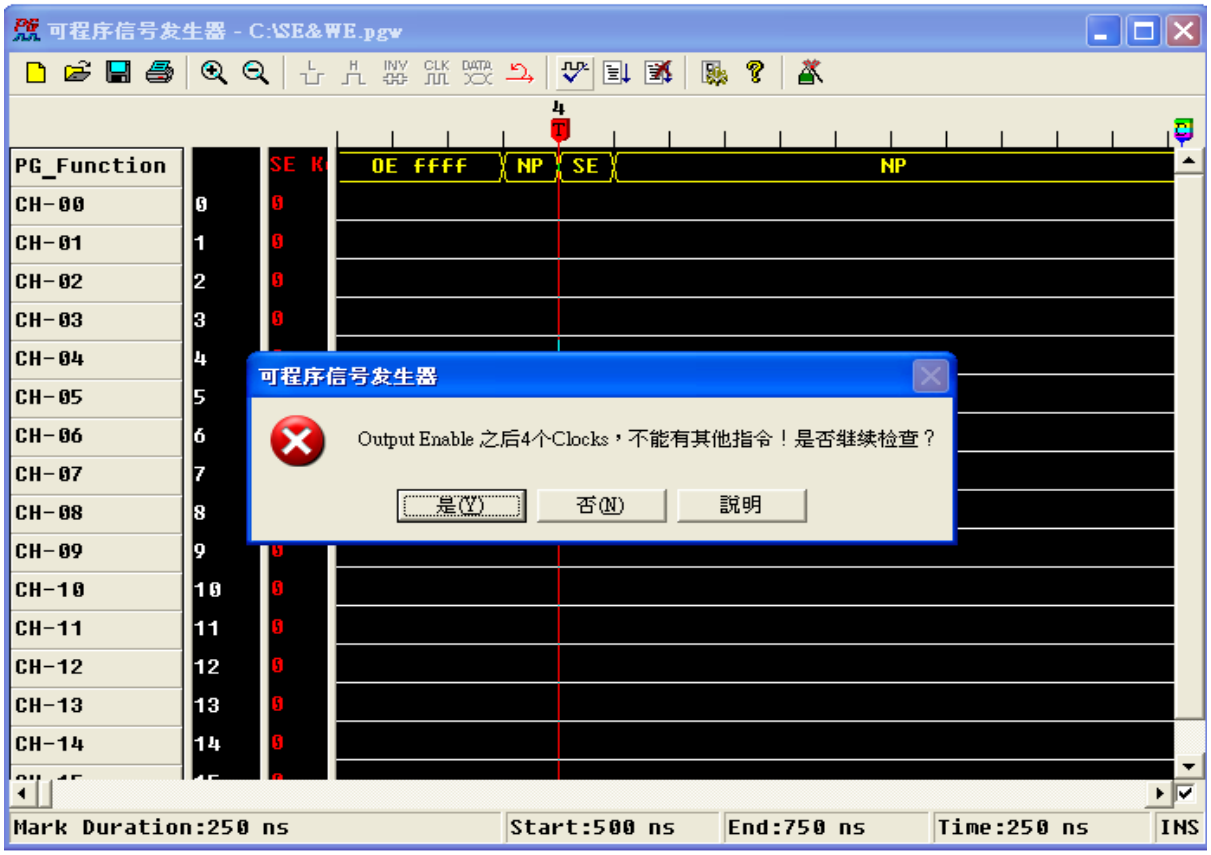
OE 该指令是 PKPG[註]系列才支持的指令而 PG[註]系列并不支持。这里输入 0xFFFF 是表示 PKPG 的 16 个数据输出脚输出致能，如果是输入 0x000F 则表示 PKPG 的 CH-00~CH-03 4 个数据输出脚为输出致能，其它数据输出脚为高阻抗；如果您输入为 0xF000，则表示 PKPG 的 CH-12~CH-15 4 个资输出脚为输出致能，其它为高阻抗。

OE(3) 括号里的数字 3 表示该指令使用 3 个 Clock。而使用该指令之后的 4 个 Clock 不能在使用其它指令。




[註] PKPG 系列包含 PKPG2016/2116/2016+/2116+

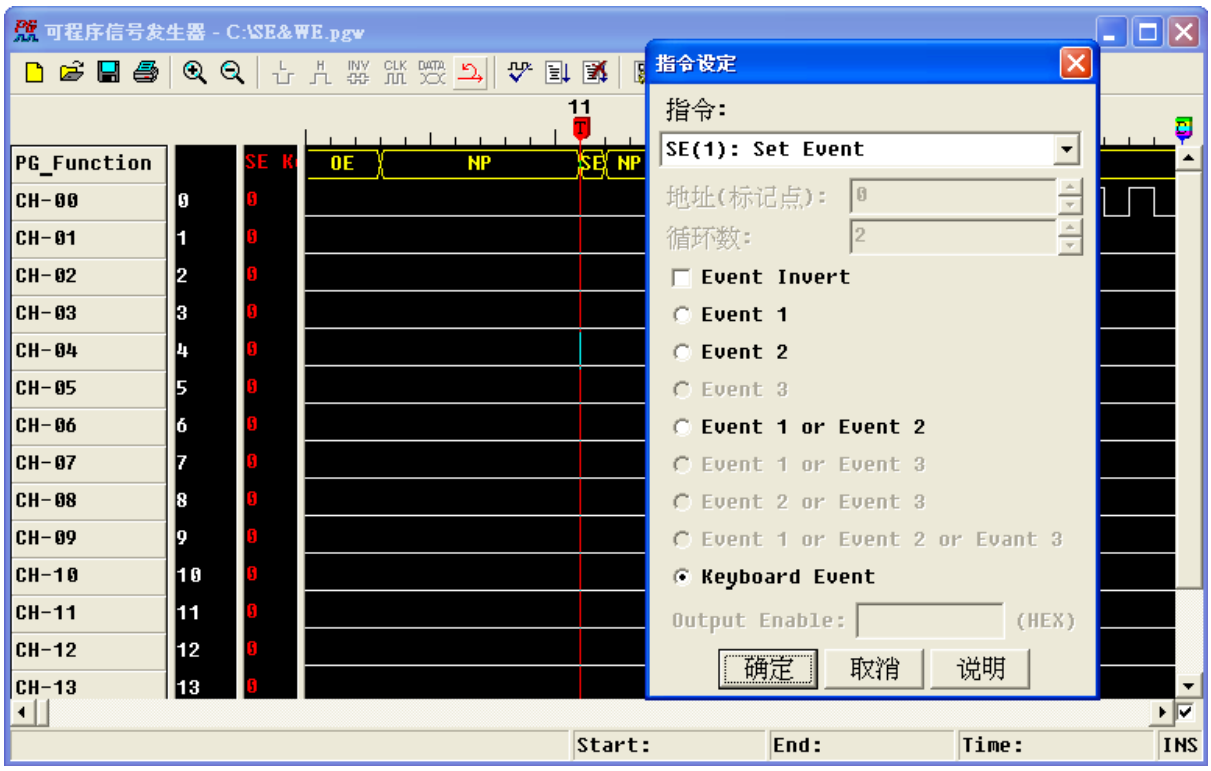
[註] PG 系列包含 PG1020/2020/1050/2050

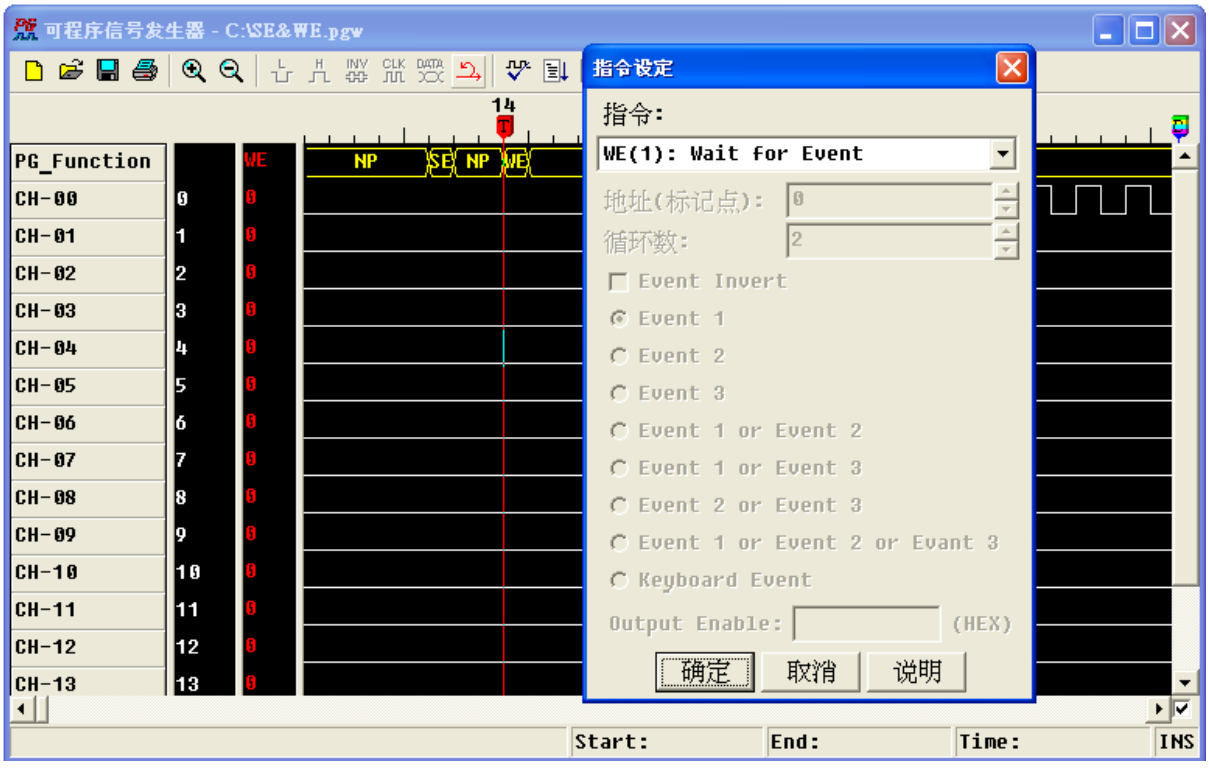



上图即是在使用 OE 该指令后未间隔 4 个 Clock 而接着使用 SE 该指令导致错误发生。再每次输入指令时，如果您不确定每个指令的间隔是否正确，都可以按下[波形检查]按钮来检查每个指令的间隔时间是否正确。

**(2)** Set Event SE(1) 和 Wait Event WE(1):

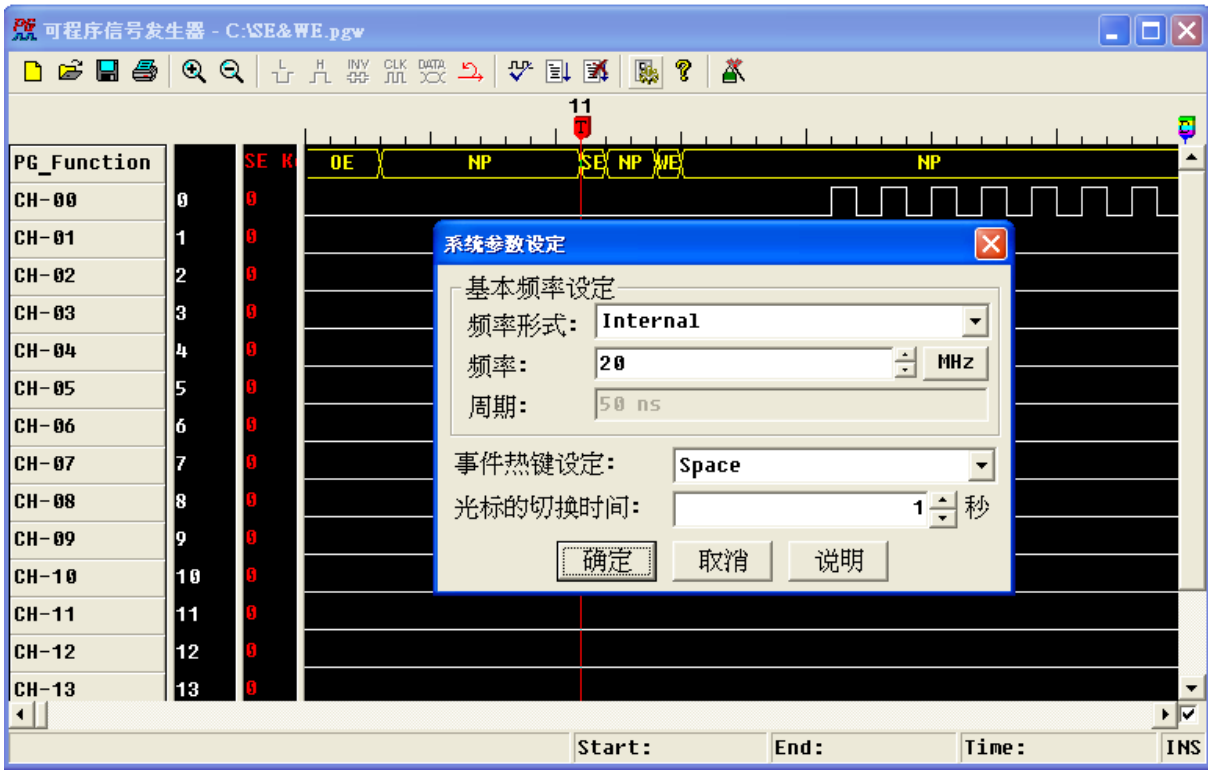
SE(1) 和 WE(1) 是设定 PG 在碰到某事件发生时，接下来要执行的动作。



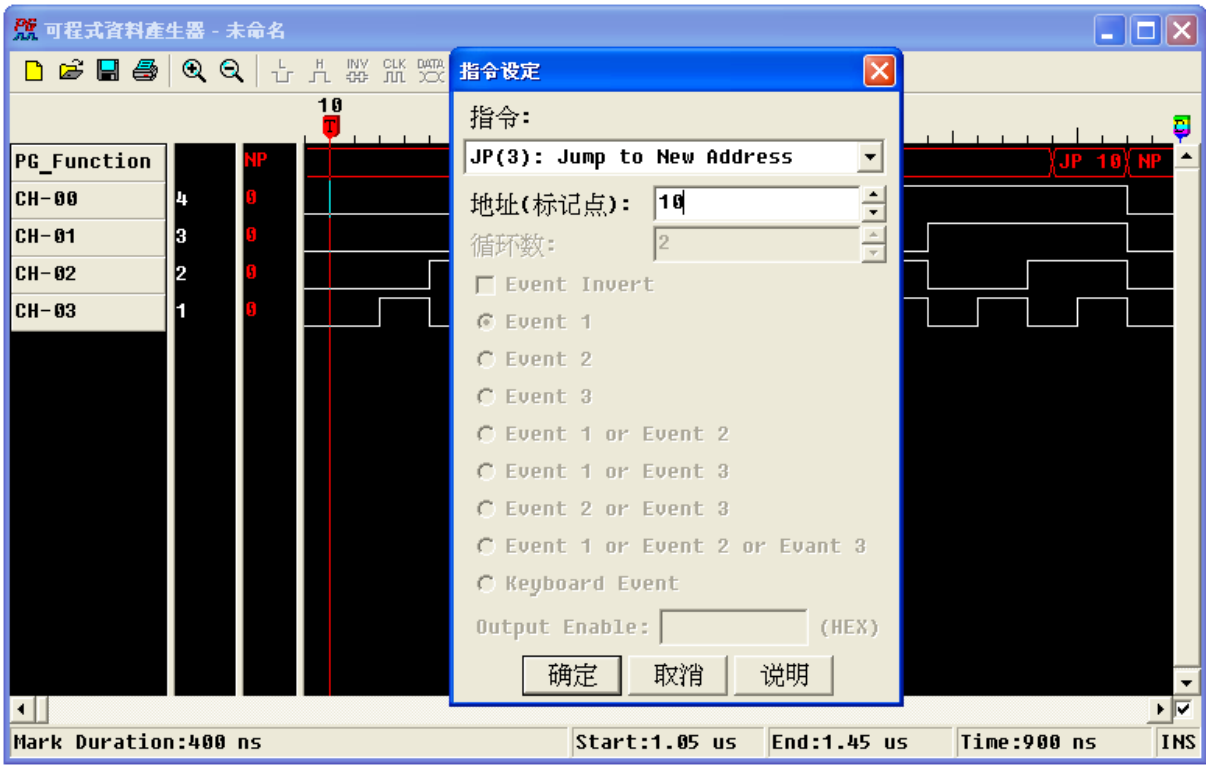

这一组 SE 和 WE 指令的输出结果: 在按下[波形输出]按钮时 ,PG 并未输出波形，直到按下 PC 键盘的 <space> 键，才会输出连续的波形。这是因为设定 PG 要等到 Keyboard 事件的发生，PG 才会动作，否则 PG 只会一直等待事件

的发生。而 Keyboard 可以选择 <space> 或 <enter>。




**(3)** Jump to New Address JP(3):

表示 PG 无条件跳跃至新地址。

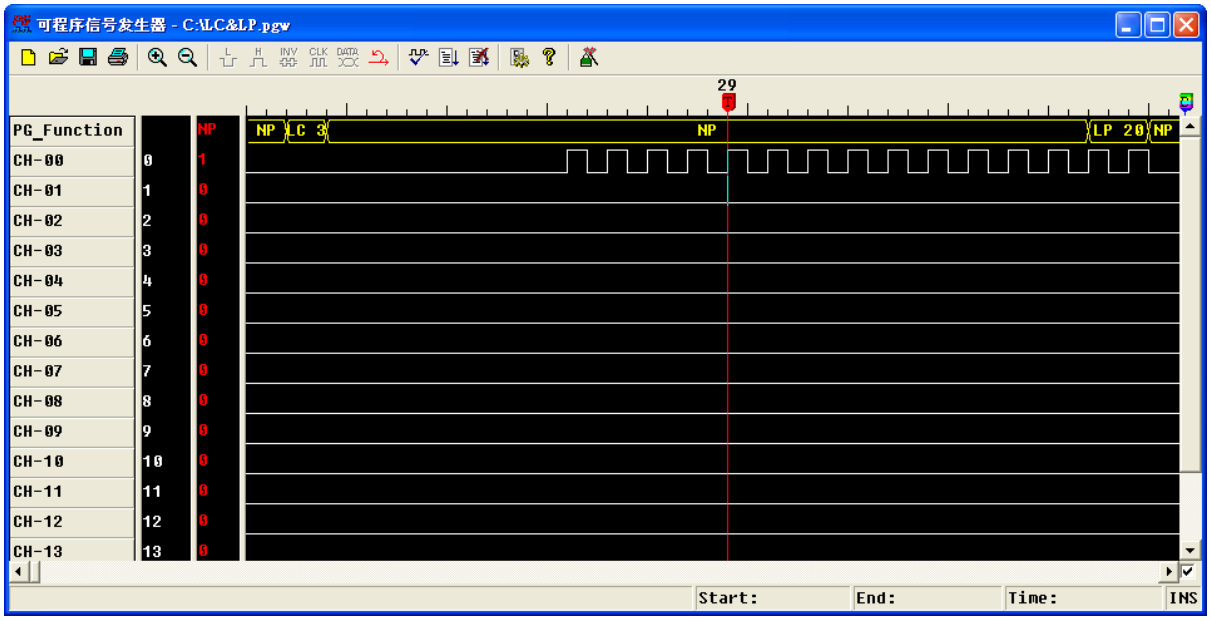



上图表示 PG 会无限次的输出地址 10~地址 26 的波形，直到按下[停止输出波

形]。

**(4)** Set Loop Count LC(3) 和 Loop to New Address LP(3):

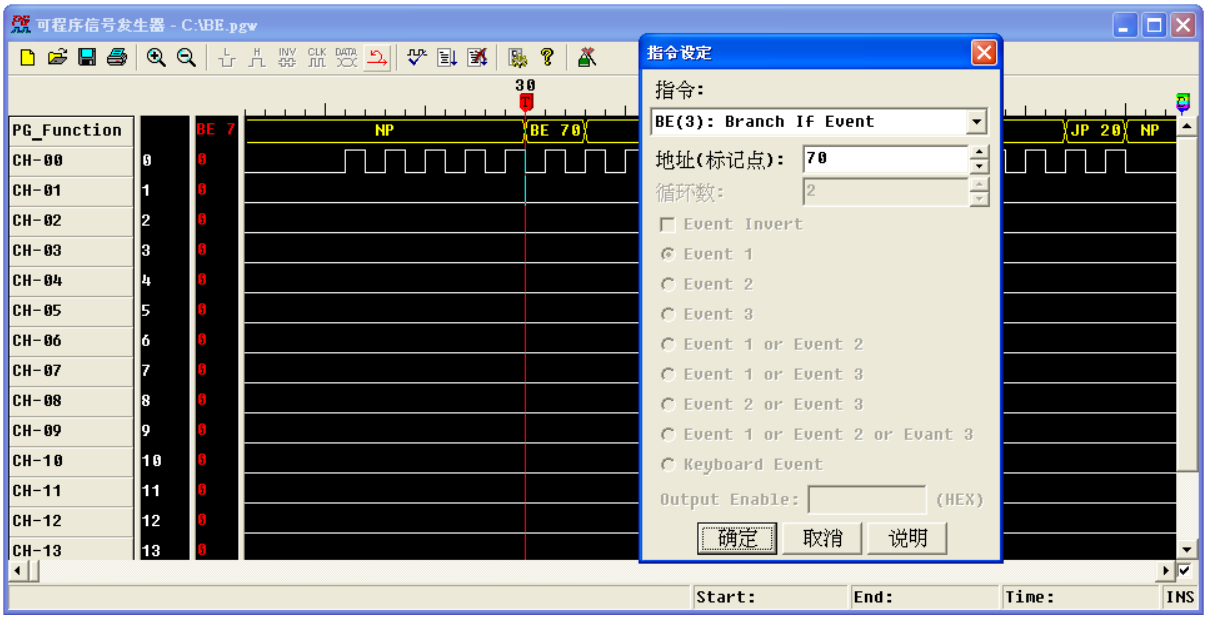
表示设定 PG 有次数限制的输出波形。LC 是设定输出次数；LP 是设定输出波形的范围，请参考撷图说明的部份。



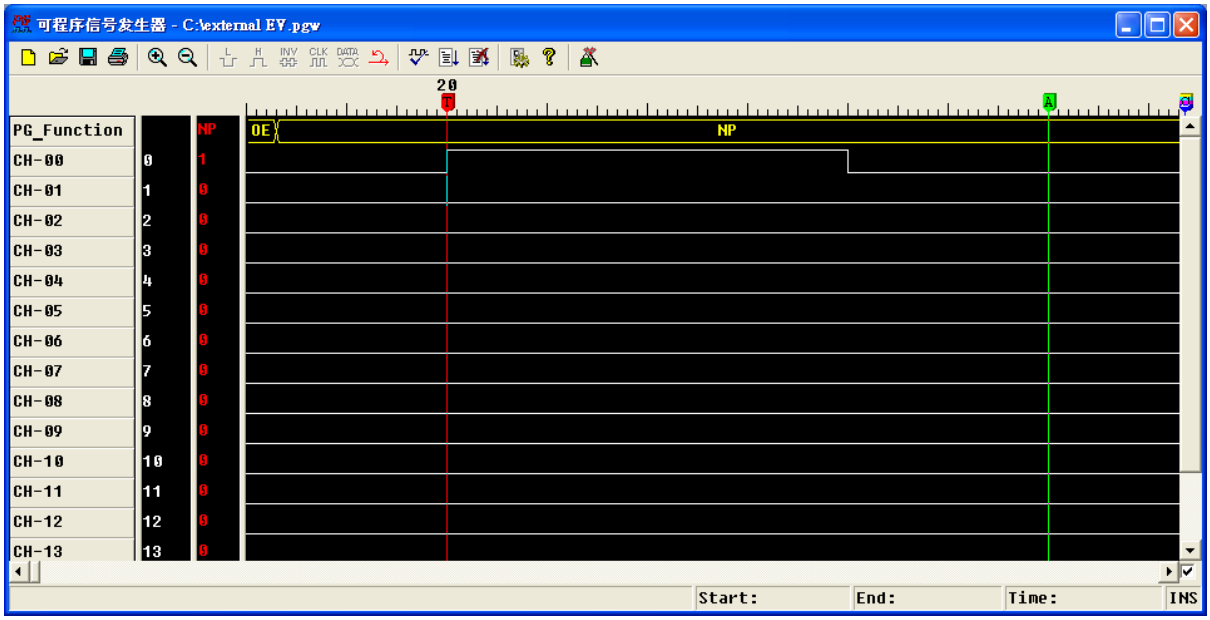

上图表示 PG 会输出图上的波形 3 次。

**(5)** Branch If Event BE(3):

表示当事件发生时，PG 跳跃至指定的新地址。请参考撷图说明:



图中的 SE 设为外部事件 Event1。这个例子是使用另一部 PG 当作外部事件的讯号源，将它简称为 PG2。而原先那一个则简称为 PG1。接法如下:接 PG2 的 CH-00 连接到 PG1 的 EV1 接脚，然后 PG2 送出如下图之讯号:




由上图可知，若 Event1 一直没发生，则 PG1 会重复送出地址 20~地址 60 的波形。若 Event1 事件发生，则 PG1 跳跃至地址 70，输出地址 70 后的波形。

# 3. 使用文字编辑器编辑文字向量文件

可程序化数据产生器除了可读取波形档(*.PGW)之外,还可以读取以文字编辑器所编辑的文字文件,而该文字文件.***PGV (PG Vector File)**的内容主要是您想要观察的一些数据以及给 PG 的指令等。

而该文字文件有一定的格式,以下的内容就该文字文件的格式及使用方法做说明,首先以下为撷取某文字向量文件的部分来做为范例并透过它来说明 **PGV** 檔的格式部分。

文字向量文件还有根据您所下的指令(Keyword)的不同而有不一样的档案内容,在定义波形数据的区域,可以区分为使用『Time Stamp』与否。除了介绍基本指令,还有许多在应用上会使用到的 PG_Function 讯号组的波形指令，整个波形指令总共有 7 个，包括 **NP(No Operation)、JP(Jump)、LP(Loop)、BE(Branch if Event)、LC(Loop Count)、SE(Set Event)**和 **WE(Wait Event)**等。以下会针对这些指令来做说明:

```
INPUTS PG_Function DATA;
ASSIGN DATA 3..0;
RADIX AUTO;
FREQUENCY 1000 Hz;
%INTERVAL 1ms;%                    『%..%』:註解符號
PATTERN
8FFh      0h   //  0              ( MOV RL, 255 )
2FFh      0h   //  1              ( MOV RH, 255 )
900h      0h   //  2               OE 65535
000h      0h   //  3
000h      0h   //  4
000h      0h   //  5            『//』:註解符號
000h      0h   //  6
000h      0h   //  7
000h      0h   //  8
000h      0h   //  9
000h      0h   // 10             START PATTERN
000h      1h   // 11
000h      2h   // 12
000h      3h   // 13
000h      4h   // 14
000h      5h   // 15
000h      6h   //  16
000h      7h   //  17
```

```
000h      8h   //  18
000h      9h   //  19
000h      Ah   //  20
000h      Bh   //  21
000h      Ch   //  22
816h      Dh   //  23          ( MOV RL, 22 )
200h      Eh   //  24          ( MOV RH, 0 )
100h      Fh   //  25           JP 10
000h      0h   //  26
;
```

上述的范例为一个 4Bits、1KHz 计数器的应用。以上述范例先来说明 PGV 的指令:

**INPUT PG_Function DATA**

设定讯号名称。每一个名称用空白分隔开来，名称可以为文字或是数字如果为 Bus(Group)时，可以用中括号来指定。例如: 某个 Bus 为 4 个通道所组成那就可以用 A[3..0]来代替，A[3..0]就代表 A3、A2、A1、A0 等 4 个讯号。

注意！『PG_Function』是一个保留字，如果在 INPUTS 指令的定义中有『PG_Function』，就代表 PATTERN 中会包含 PG 的专用指令。而且『PG_Function』也不能配合 ASSIGN 指令使用。

**ASSIGN DATA 3..0**

用来指定 INPUTS 指令所定义的讯号名称之信道值。结果如下

DATA0 = CH0 , DATA1 = CH1……DATA3= CH3 等。

**RADIX AUTO**

RADIX 为区域的进位值。设定 PATTERN 区域的进位值。如果 PATTERN 区域的数值有进位识别符号时，此时就要将 RADIX 的值设成 AUTO。例如 RADIX 设定成 AUTO 时，PATTERN 区域的某一数值为 35 与 35h 是不一样的，但是当 RADIX 设成 HEX(十六进制)时，这两个值就是一样的。但是当 RADIX 设成 DEC 时，而 PATTERN 区域的值为 35h 时，却是会被当成 35。

RADIX 共有五种定义:

AUTO : 由数值的进位识别符号决定。

HEX : 16 进位。

DEC : 10 进位。

OCT : 8 进位。

BIN : 2 进位。

而在 PATTERN 区域的进位识别符号定义是『h』代表 16 进位、『o』代表 8 进位和『b』代表 2 进位，而不加任何识别符号就代表 10 进位。

**FREQUENCY 1000 Hz**

这里是指定使用 PG 主频为 1000Hz,主要为了设定计数器的频率 1KHz。

**PATTERN**

是用来定义波形数据的区域。这个区域包含了两个部份，一个是时间部份一个是波形部份。时间部份称为『Time Stamp』，时间部份和波形部份用『>』(大

于符号)来做分隔。时间部份也可以省略，此时波形部份每一行的时间就会由 INTERVAL 或是 FREQUENCY 来指定，所以 INTERVAL 和 FREQUENCY 只能选用一种。如果使用 Time Stamp 方式，时间部份的单位就是 UNIT 所设定的值。波形部份就是根据 INPUTS 的定义顺序来描述波形。上述 PGV 的档案中实际的数据为 10~25 共 16 笔,您可以观察到从第 10 笔开始一直到第 25 笔结束,数据值为 0h,1h,2h,3h……4Fh,上述为十六进制值,如果以十进制表示,则 0,1,2,3……15。

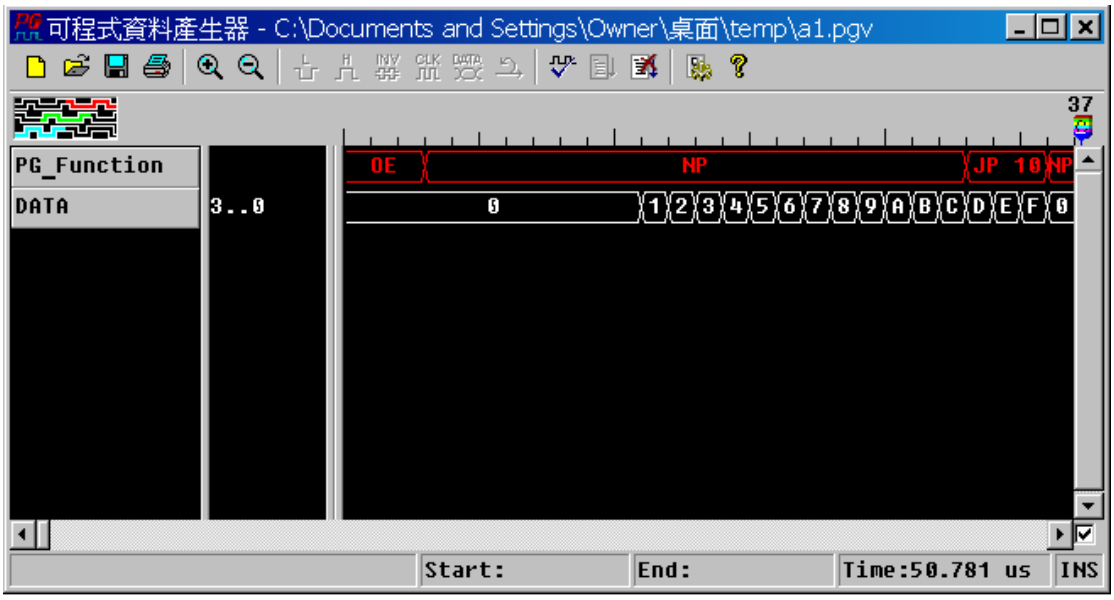
PS:在第 25 笔的批注中的 JP 10 是表示在输出第 25 笔的数据值之后,在跳回第十笔来重复的输出数据。

我们使用 PG EDIT  来观察上述计数器的文字向量文件所产生的结果,使用

步骤如下:1.执行 PG 程序 

2. 执行**文字(向量)文件转成波形**该程序  接口如下,并且选取编辑好的文字向量文件并加载该档案


然后再按下『**波形转换**』会出现以下画面,该画面为 PG EDIT 的程序接口:



上述为文字向量文件 PGV 格式中的其中一种,主要的分别为以上的 PATTERN 的波形数据部分使采用 FREQUENCY 的例子,您可以看到 PGV 档档头的部分:

```
FREQUENCY 1000 Hz;        %INTERVAL 1ms;%
```

而另一种格式为上述 PATTERN 说明中关于 Time Stamp 的部分,底下为使用 Time Stamp 的例子:

```
INPUTS PG_Function DATA;
ASSIGN DATA 3..0;
RADIX AUTO;
UNIT   ms;
PATTERN
0.0> 8FFh     0h
1.0> 2FFh     0h
2.0> 900h     0h
10.0>000h     0h
11.0>000h     1h
12.0>000h     2h
13.0>000h     3h
14.0>000h     4h
15.0>000h     5h
16.0>000h     6h
17.0>000h     7h
18.0>000h     8h
```

以上为 Time stamp 格式的简单范例,您可以看到与 FREQUENCY 的不同之处在于档头与 PATTERN 内容开头的部分:

UNIT ms

1.0> 1h

如上所说明的,UNIT ms 是指定数据的时间单位 ms(毫秒),而 1.0> 1h 是表示在 1.0ms 这个时间点有 PG_Function 指令的改变或是数据的改变,在 Time Stamp 的例子中都是记录状态有改变的时间点。『>』是区隔时间部分以及波形数据部分。而关于每笔数据采用的时间单位为 1.0> 主要是要配合所设定的计数器频率 1KHz,您可以注意到只要是 PG_Function 的指令或是 DATA 改变,Time Stamp 才会加 1 ms,例如说时间 3.0>~9.0> ms 之间并没有任何的 PG_Function 或是 DATA 改变,所以没有记录在档案内,您可以跟使用 FREQUENCY 的 PGV 档案内容做比较。

注 1:8FFh、2FFh、900h、816h、200h、100h 为 PG_Function 指令,该指令的功能都有在该行末的批注说明,关于 PG_Function 指令的完整说明请参考底下注 2

注 2:关于 PG_Function 指令的说明:

表一

| 指令 | 指令全名 | 说 明 | 周期 |
|---|---|---|---|
| NP | No Operation | 没有动作 | 1 |
| JP | Jump | 跳跃到设定点继续执行 | 3 |
| LP | Loop | 将循环数减一,如果循环数不为 0 就跳跃到设定点,如果为 0 就跳到下一个位置。 | 3 |

| | | | |
|---|---|---|---|
| BE | Branch if Event | 如果以设定的 Event 讯号出现就跳跃到设定点，否则就会跳跃到下一个位置。 | 3 |
| LC | Loop Count | 设定循环数，设定值为 2～65536。 | 2 |
| SE | Set Event | 选择触发的事件。 | 1 |
| WE | Wait Event | 等待事件被启动。 | 1 |

PG 内部有下列几个缓存器: RT, REX, RC, ROE。使用时都是透过 PG_FUNCTION 的指令控制。而 PG_FUNCTION 是一个 12Bits 的指令集。

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 8 | XX | MOV RL,XX | 将 PG_Function 的 LSB 8Bits 值移入 RL 缓存器中。 |
| 2 | XX | MOV RH,XX | 将 PG_Function 的 LSB 8Bits 值移入 RH 缓存器中。 |
| 1 | XX | JP RT | 跳至 RT-12 的新地址。 |

RT 有 16 个 Bits，可拆成两个 8 Bits 的缓存器来用分别是 RL 及 RH。RT 的用途主要是用来传递数据用的。填入 RT 值的方法是藉由对 RL 以及 RH 这两个 8Bits 的缓存器填值,例如:

```
MOV     RL     16h          //对缓存器 RL 填入 16h
MOV     RH     00h          //对缓存器 RH 填入 0h
会等同于: MOV     RT     016h
```

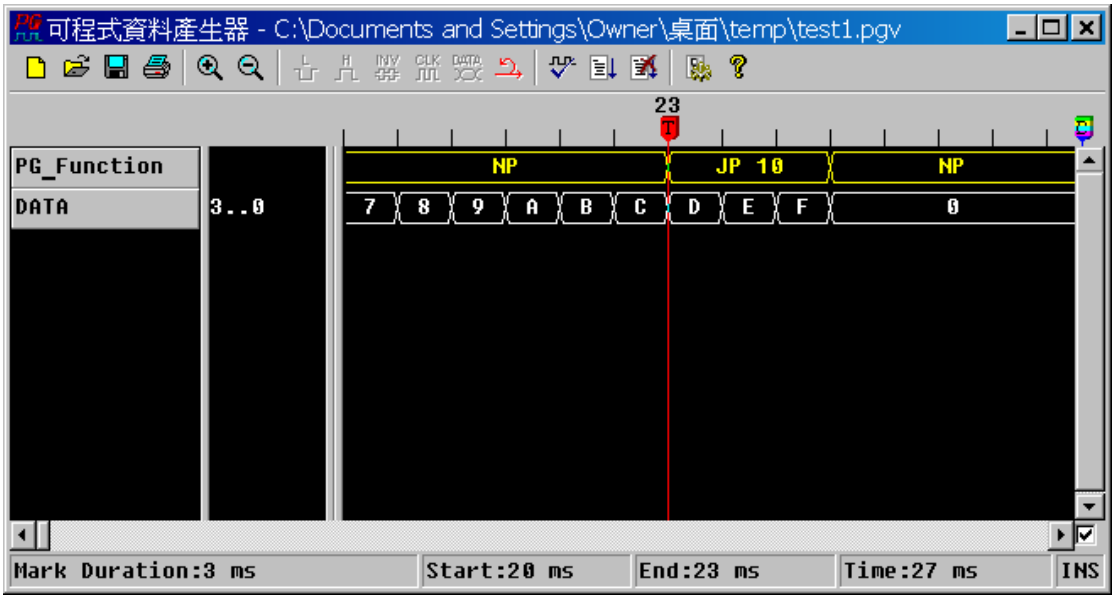
PG_Function 的指令主要是告知 PG 内部的缓存器指针做什么对应的动作,然后由内存取出数据传送出去。以下就 PG_Function 内部的指令说明如下:

**NP(No Operation):**在波形执行时，并不会有任何控制流程的改变或是任何内部缓存器改变。用 CPU 的观点来看，那 NP 指令的动作就是 New Address = Address + 1，也就是指针移到下一个位置，其它不做任何改变。

**JP(Jump):**是一个改变控制流程的指令。例如『JP 35』，这个指令的动作就是无条件跳至新地址 35(相当于 New Address=35)。而在文字向量文件里(*.PGV)该如何使用这个指令,以下皆使用前述 1KHz 的计数器的 PGV 档案内容做说明:

```
000h Bh    // 00021:
000h Ch    // 00022:
816h Dh    // 00023:        (MOV RL, 22)
200h Eh    // 00024:        (MOV RH, 0)
100h Fh    // 00025:        JP 10
000h 0h    // 00026:
;
```

以上三行红色字体的十六进制的 PG_Function 的指令, 816h:对缓存器 RL 填入数值 22(MOV RL, 22),200h: 对缓存器 RH 填入数值 0 (MOV RH, 0), 100h: 跳至 RT-12 的新地址( 22 – 12 = 10 ),也就是说送出第 25 点数据后,跳跃到第十点后再继续传送数据。以下是透过 PG EDIT  来观察文字向量文件(*.PGV)内的数据与 PG_Function 的指令:




**LP(Loop):** LP 和 JP 指令很类似，其最大差异就是 JP 是一个无条件跳跃指令，而 LP 是一个有条件跳跃，它必须与『LC』指令配合使用。PG 有一个内部缓存器称为循环缓存器(Counter)，这个缓存器就是用来记录循环数目的缓存器。这个缓存器是用 LC(Loop Count)指令来设定，例如『LC 32』就是将循环缓存器设定成 32。循环缓存器可以设定的值为 2～65536，所以不能设定成 0 与 1，这种设定方式与一般的 CPU 或 MCU 的用法有些许的差异，需注意。设定好循环以后就可以使用 LP 指令了，而每次执行 LP 指令，循环缓存器的值就会减 1，而跳跃的新指针就会参考 LP 指令的设定值来决定。

例如指令『LC 32』、『LP 16』，的整个动作步骤如下：

一、将循环缓存器 32 减 1。

二、检查循环缓存器是否为 0。

三、如果循环缓存器为 0，New Address = Next Address。

四、如果循环不为 0，New Address = 16。

```
200h 0h   // 00007: (MOV RH, 0)
401h 0h   // 00008: LC 3
000h 0h   // 00009:
000h 0h   // 00010:

============================

820h Dh   // 00033: (MOV RL, 32)
200h Eh   // 00034: (MOV RH, 0)
300h Fh   // 00035: LP 20
```

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 3 | XX | LP RT | 跳至 RT-12 的新地址。 |

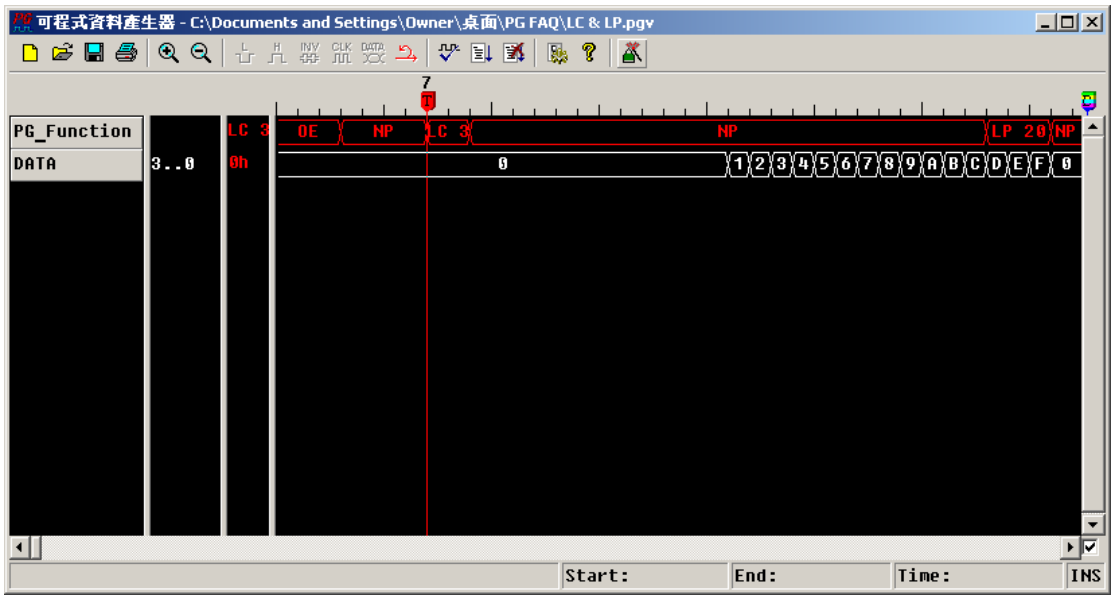
先看到前三行红色字体的十六进制的 PG_Function 的指令, 820h: 对缓存器 RL 填入数值 32(MOV RL, 32), 200h: 对缓存器 RH 填入数值 0 (MOV RH, 0), 300h: 跳至 RT-12 的新地址( 32 – 12 = 20 )做循环的动作。

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 4 | XX | LC RC | 设定 RC 缓存器的循环数 |

接下来看到后两行的部分: 200h: 对缓存器 RH 填入数值 0(MOV RH, 0), 401h:对循环缓存器 RC 填入数值 3(1 + 2 = 3,因为循环数从 2 开始),也就是说缓存器 RC 里的值为 1,因为 RC 也是 16Bits 的缓存器,所以这两行所代表的意义为 :

| RC | 00000000 (00h) | 00000001(01h) |
|---|---|---|

以下为透过 PG EDIT  来观察文字向量文件(*.PGV)内的数据与 PG_Function 的指令:




有一点要注意的是如果当循环缓存器已经被减成 0，又遇上 LP 指令时，这时候循环缓存器是不会再被减成-1，而是会产生特殊情况。所以循环缓存器被减成 0 以后，LP 指令的工作是不可预期的，所以要使用 LP 指令之前请注意是否已经设定 LC 指令了。

**SE(Set Event):**是一个选择事件的指令，PG 有 3 个外部事件通道(Event_1、Event_2、Event_3)及 1 个内部键盘事件(Keyboard Event)，所以总共有 4 个事件来源。PG 将这 4 个事件来源编辑成 16 种事件型态以兹利用，而这个事件型态会被存到一个 PG 的内部缓存器，称之为事件缓存器(Event Register)。这些事件型态包括:

**1. Keyboard Event**

**2. Event_1**

**3. Event_2**

**4. Event_3**

**5. Event_1 or Event_2**

**6. Event_1 or Event_3**

**7. Event_2 or Event_3**

**8. Event_1 or Event_2 or Event_3**

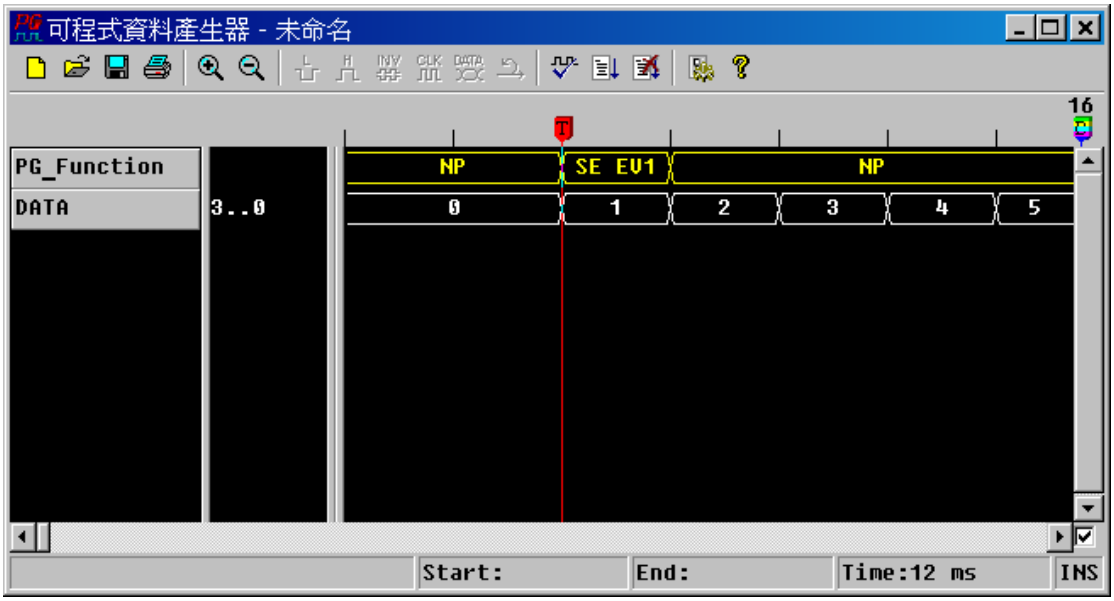
另外 8 种事件型态就是这 8 种型态的反相。例如事件缓存器的设定值为上述 8 种型态之一，此时不管从外部或是内部的事件通道都会被 PG 所监控，PG 会随时比较目前的事件通道的状态是否与事件缓存器的事件型态相同。如果相同就会将 PG 内部的旗标缓存器(Flag Register)的 Event Bit 设成 1(True)，如果不同就会将 Event Bit 设成 0(False)。然而反相型态就是当事件信道的状态与事件缓存器的事件型态相同时，PG 会将 Event Bit 设成 0，不同时则会将 Event Bit 设成 1。

```
000h 0h    // 00010:
609h 1h    // 00011:                    SE EV1
000h 2h    // 00012:
```

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 6 | XX | SE EV | 设定 REX 缓存器事件值 |

8Bits 的 LSB 中:00h 代表内部键盘事件(Keyboard Event)、01h:代表外部事件信道 1 的反相型态(!EV1)、02h: 代表外部事件通道 2 的反相(!EV2)型态、03h: 代表外部事件信道 1 的反相型态与外部事件信道 2 反相型态的交集( !EV1 & !EV2 ) 、04h:代表外部事件信道 3 的反相型态(!EV3)、05h:代表外部事件通道 1 的反相与外部事件通道 3 的反相的交集(!EV1 & !EV3 )、 06h:代表外部事件通道 2 的反相与外部事件通道 3 的反相交集( !EV2 & !EV3 ) 、07h:代表设定外部事件通道 1 的反相与外部事件通道 2 的反相与外部事件通道 3 的反相的交集(!EV1 & !EV2 & !EV3 )、08h:代表内部键盘事件的反相( !KEY)、09h:代表外部事件 1(EV1)、0Ah:代表外部事件 2(EV2)、0Bh:表示 Event_1 or Event_2,也就是!( !EV1 & !EV2 )……其它的依此类推,总共会有 16 种不同的事件。

只要在 8Bits 的 LSB 中输入想要的情况即可。以下为上述例子之图示:

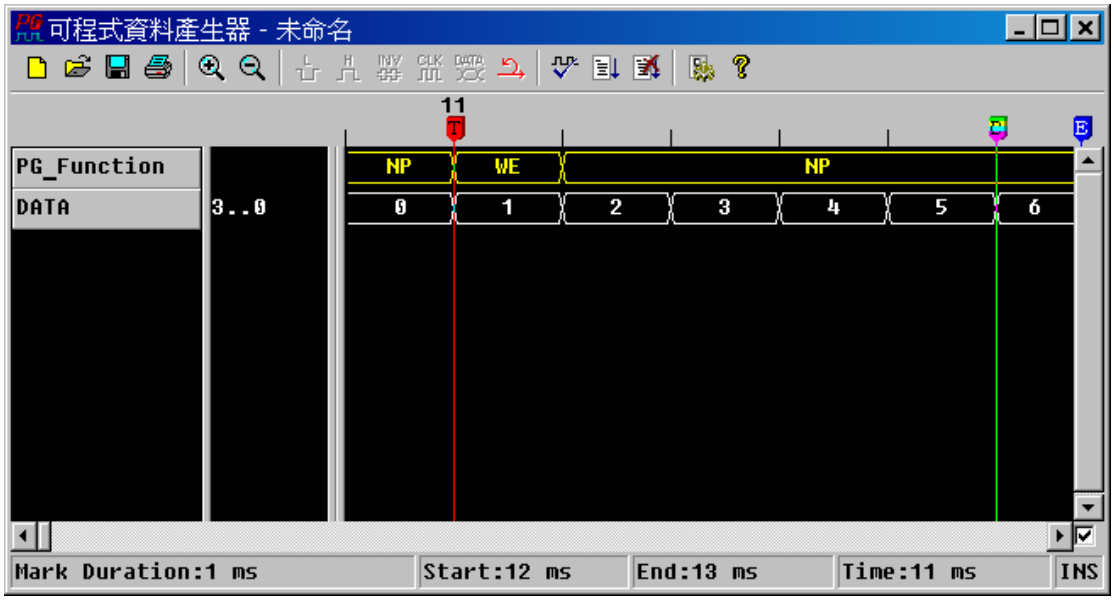


而运用 Event Bit 的指令有两个，一是 WE(Wait Event)另一是 BE(Branch If Event)。

**WE(Wait Event):**指令是当波形执行到这个指令时，整个 PG 会因此而暂停，而波形会停留在 WE 指令当时的波形，一直到 Event Bit 为 1 时，才会再继续往下执行。

```
000h 0h    // 00010:
700h 1h    // 00011:              WE
000h 2h    // 00012:
```

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 7 | XX | WE | PG 暂停,等待事件发生 |




**BE(Branch If Event):**指令和 LP 指令又有点类似，因为他们都是一个有条件跳跃的指令。LP 的跳跃条件是循环缓存器，而 BE 的跳跃条件是 Event Bit。当波形执行

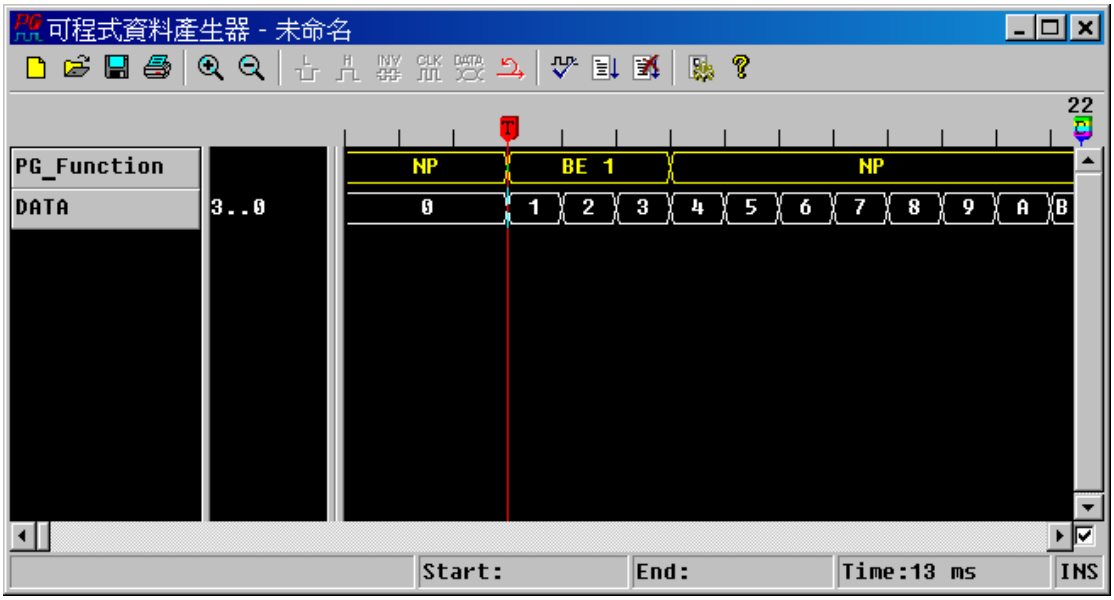
到 BE 指令时，PG 会立即判断 Event Bit，如果 Event Bit 为 1 就会跳到 BE 所设定的新地址，Event Bit 为 0 的话，则继续执行下一个地址。

```
000h 0h   // 00010:
80Dh 1h   // 00011:                    (MOV RL, 13)
200h 2h   // 00012:                    (MOV RH, 0 )
500h 3h   // 00013:                     BE 13
000h 4h   // 00014:
```

| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 5 | XX | BE | 跳至 REX 的新地址。 |

而 80Dh: 对缓存器 RL 填入数值 13(MOV RL, 13), 200h: 对缓存器 RH 填入数值 0(MOV RH, 0),500h: 跳至 REX 的新地址。

以下为上述例子之图示:




在 PKPG 系列中,PG_Function 的指令还有一个 Output Enable 『OE』

,说明如下:

```
8FFh 0h    // 00000:          (MOV RL, 255)
2FFh 0h    // 00001:          (MOV RH, 255)
900h 0h    // 00002:          0E   65535
000h 0h    // 00003:
000h 0h    // 00004:
```

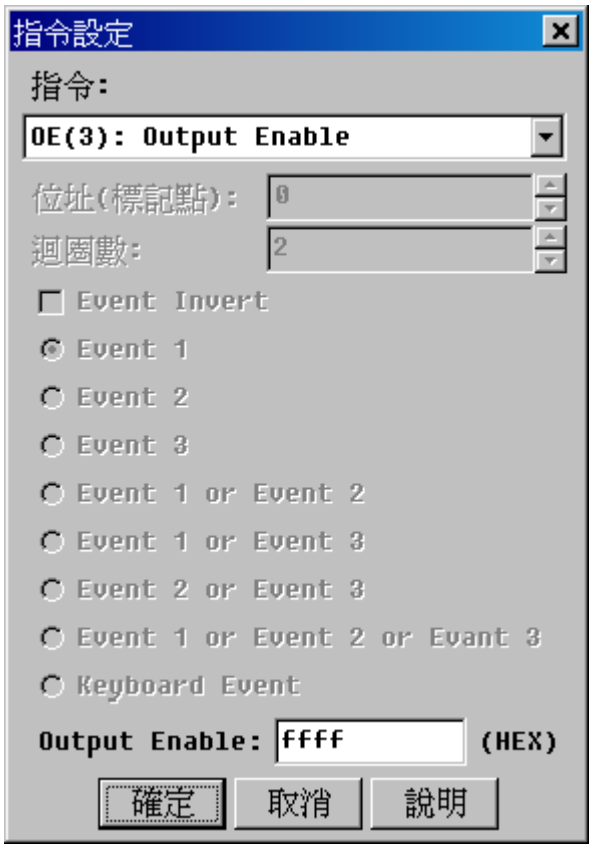
| PG_Function (12Bits) | | | |
|---|---|---|---|
| 4Bits(MSB) | 8Bits(LSB) | | |
| 9 | XX | OE | 将通道输出致能 |

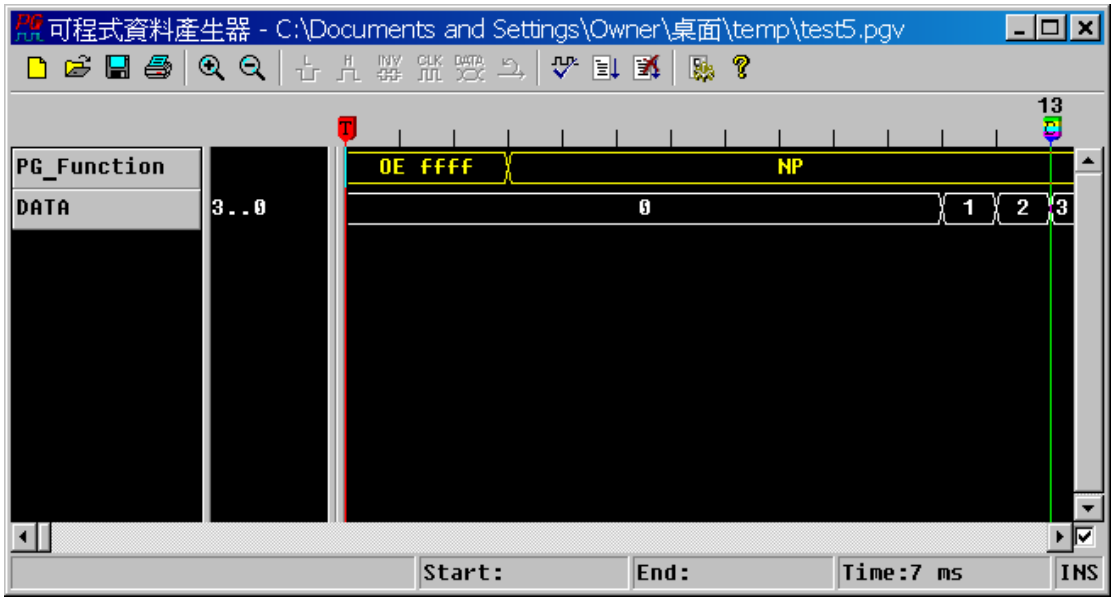
8FFh: 对缓存器 RL 填入数值 255(MOV RL, 255),2FFh:对缓存器 RH 填入数值 255(MOV RH, 255), 900h: 输出致能。

缓存器 ROE 就如同缓存器 RT 一样,都是由 8Bits 的 RL 与 RH 组成,所以整个 ROE 缓存器也可以看成如下所示:

| **缓存器 ROE** | 11111111(FFh) | 11111111(FFh) |
|---|---|---|

底下为图示:



# PC-Based Pattern Generator Manual





Acute
PC-based T&M Instruments

皇晶科技股份有限公司

www.acute.com.tw

地址：新北市三重区 24159 重新路五段 609 巷 12 号 6 楼之 7 (汤城园区)

连络电话：(02)2999-3275

传真：(02)2999-3276

E-mail: mailto:service@acute.com.tw