3196D 软件使用手册

北京新润泰思特测控技术有限公司

北京东润泰思特测控技术有限公司

版本：2.0

目录

第一章 测试系统界面使用说明

一 综述

TestShell 是 3168/3196D 集成电路测试系统的软件开发平台，它可以开发、运行和管理测试程序；在测试器件的同时可以实现对测试结果的存储、分析和显示。

TestShell 程序主界面如图 1-1 所示。

图 1-1 测试系统主界面

1. 在“管理员设置”中，用户可以进行操作员、程序库的管理；

2. 按“开发” “设置参数” “编制图形”（仅对数字器件） “编辑源文件” “测试”顺序，进行新程序的开发及调试；

3. 按“操作员设置” “选择测试程序” “设置参数”（可选） “编制图形”（仅对数字器件） “编辑源文件”（可选） “BIN 设置”（可选） “测试”顺序，进行器件测试；

4.在“数据分析”中，可以选择测试数据程序库分析测试数据；下面将就这几个方面，分别详细介绍。

二 重要说明

1. 所有的应用程序和库文件应拷贝到 c:\drtest\testshell 下，用户如无特别需要，不应改动本目录下的任何文件；
2. 在 c:\drtest 目录下双击 TestShell.exe，打开运行界面；
3. 管理员初始密码为 123；
4. 所有需要用户输入信息的地方，都应该输入英文字母或数字，以防产生不必要的错误；
5. 同一数据库中所包含的元片个数不要超过 120 个；

三 管理员设置

单击主界面中的“管理员设置”按钮，打开如图 1-2 所示的对话框。

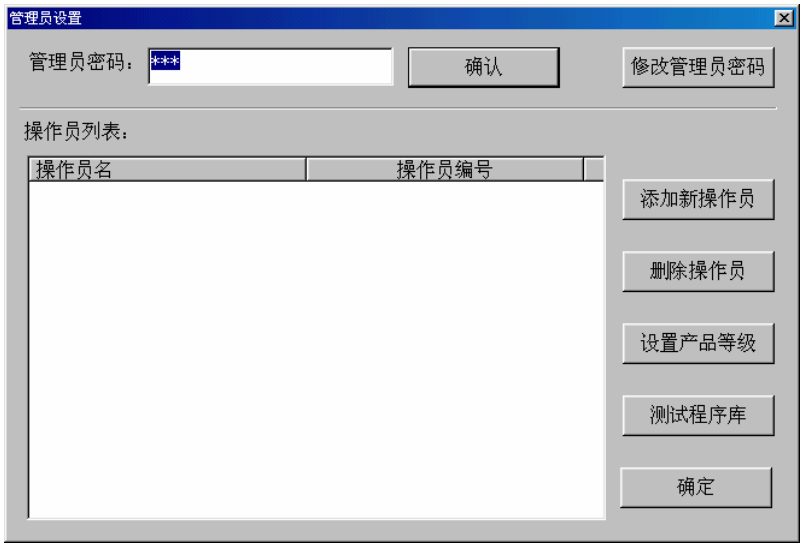


图 1-2 管理员设置

如果管理员没有改变初始密码（123），那么无需密码确认就可以进行以下操作；如果密码已经改变，应在“管理员密码”后面的编辑框中输入管理员密码，然后单击“确认”按钮，这样管理员就可以进行对话框中所示的各项操作了。

1. 修改管理员密码

步骤如下：

(1)单击“修改管理员密码”按钮，打开如图 1-3 所示的对话框。

图 1-3 修改管理员密码

(2)在“原密码”框中输入原来密码，然后在“新密码”中输入新密码，并在“确认密码”中再次输入新密码，以确保密码正确；然后单击“确定”按纽，如果信息正确，则正常退出，否则将弹出错误信息。

2. 添加新操作员

步骤如下：

(1)单击“添加新操作员” 按纽，打开如图 1-4 所示的对话框：

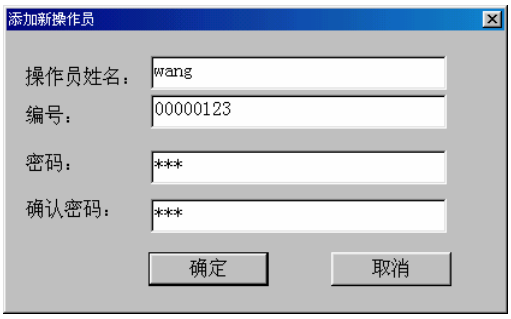


图 1-4 添加新操作员

(2)在“操作员姓名”和“编号”中输入操作员的姓名和编号信息，这里应保证信息的唯一性，不能和其他的操作员重复；

(3)在“密码”中输入操作员密码，并在“确认密码”中再次输入相同密码；

(4)单击“确定”按纽，如果信息中无非法字符，两密码相同，则正常退出，否则将弹出错误信息。

3. 删除操作员

删除操作员的方法是：在用户列表中选中某个操作员，然后单击“删除操作员”按纽，则将该操作员从操作员列表中删除。

4. 维护测试程序库

单击“测试程序库”按钮，打开如图 1-5 所示的对话框，用户可以在此对已有测试程序进行管理，实现如下功能：

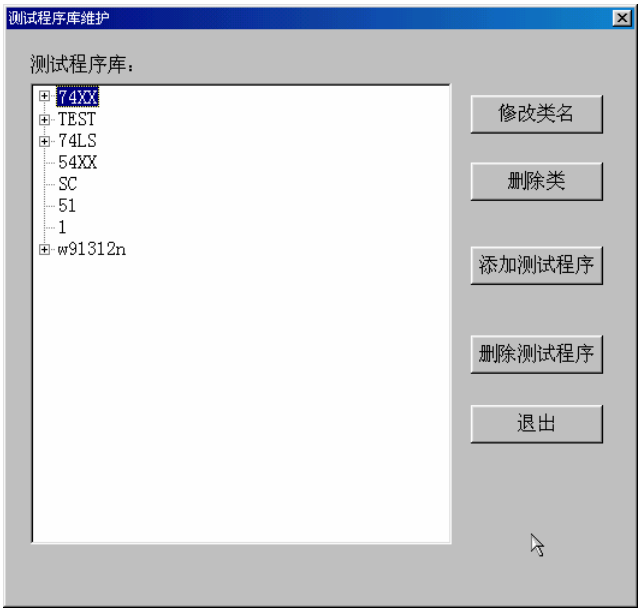


图 1-5 测试程序库

修改类名

修改类名步骤如下：

（1）在列表框中选中一个测试类，单击“修改类名”按钮，打开如图 1-6 所示的对话框。

图 1-6 修改类名

（2）在“请输入新类名”后输入新类名，单击“确定”按钮，完成当前类名的修改。

删除类

在列表框中选中一个测试类，单击“删除类”按钮，即可将当前类及其包含的全部测试程序从中删除；但应注意的是，此处只是将测试程序从登记表中删除，并未将实际的测试程序删除，如果需要删除测试程序，还应在资源管理器中将整个的测试项目从硬盘中删除；

向程序库中添加程序

步骤如下：

（1）单击“添加程序”按钮，打开如图 1-7 所示的对话框，可向程序库中添加已存在的测试程序。

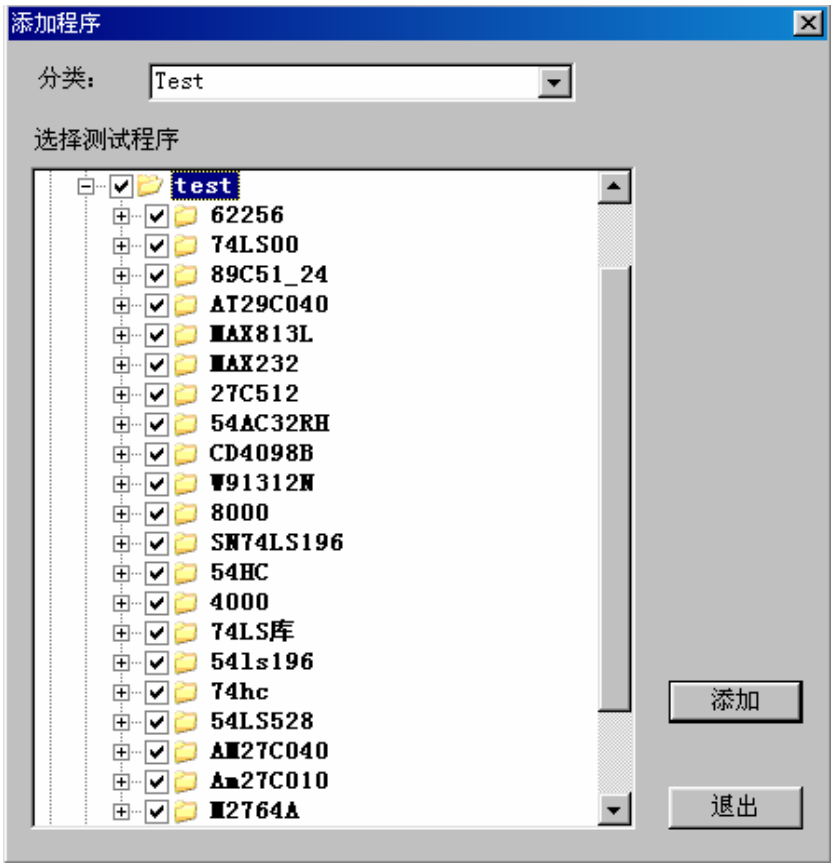


图 1-7 添加程序对话框

（2）在“分类”后面的下拉列表中，选择已存在的分类，或输入新的分类名。

（3）在“选择测试程序”下面的树状列表中选择测试程序，方法是选中测试程序所在的文件夹前面的复选框，如果一个文件夹中包含了多个测试程序工程，则选中该文件夹前面的复选框，则将选择其中的全部测试程序。用户可在该文件夹上单击鼠标右键，将显示该文件夹中选中的所有测试程序。如下图所示。

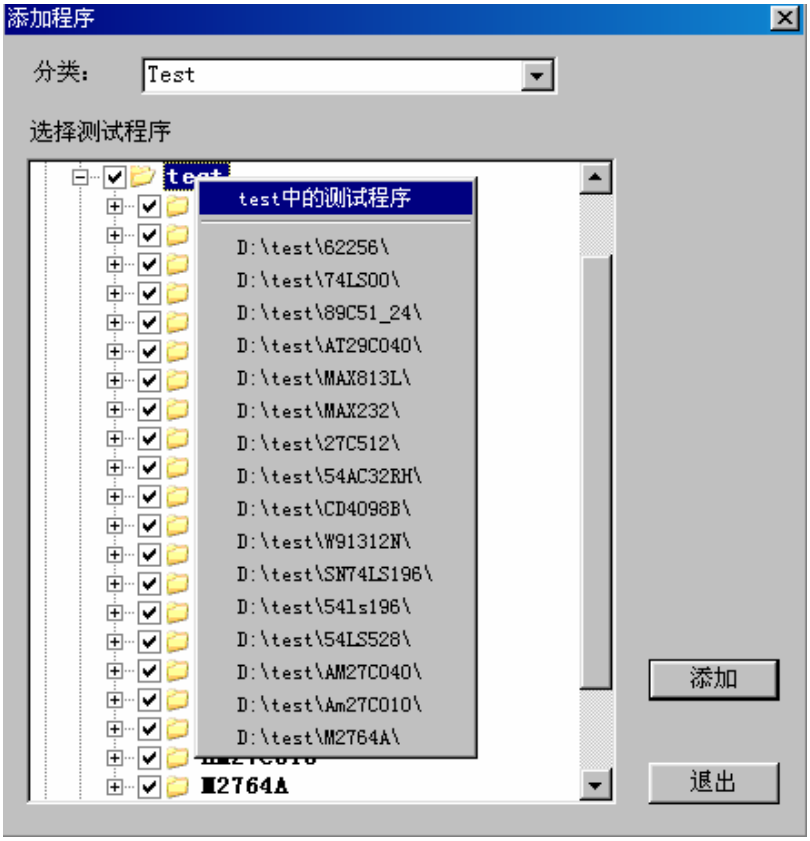


图 1-8 显示所选测试程序

（4）单击“添加”按钮，则将被选择的测试程序添加到所设置的分类中。上述操作可重复进行

（5）单击“退出”按钮，返回测试程序库维护对话框。

删除程序

（1）在列表框中选择某个测试程序名，然后单击“删除程序”按钮，将弹出警告，如图 1-9 所示。

图 1-9 删除警告对话框

（2）单击“是”按钮，将从测试程序库中删除，注意：该操作并不从硬盘中实际删除该测试程序，如果要删除该测试程序，应到资源管理器中自行删除。

（3）单击“否”或“取消”按钮，将取消删除操作。

（4）单击“退出”按钮，退出该对话框。

四 开发测试程序

用户应在主界面下设置程序路径以及各测试项参数、在 Visual C++下编辑源代码、编译成动态链接库后进行调试。具体步骤步骤如下：

1. 初始化

首先设置测试程序的输出路径，器件名称，参数表名以及分类信息，方法是：

（1）在主界面中单击“开发”按钮，打开如图 1-10 所示的对话框。

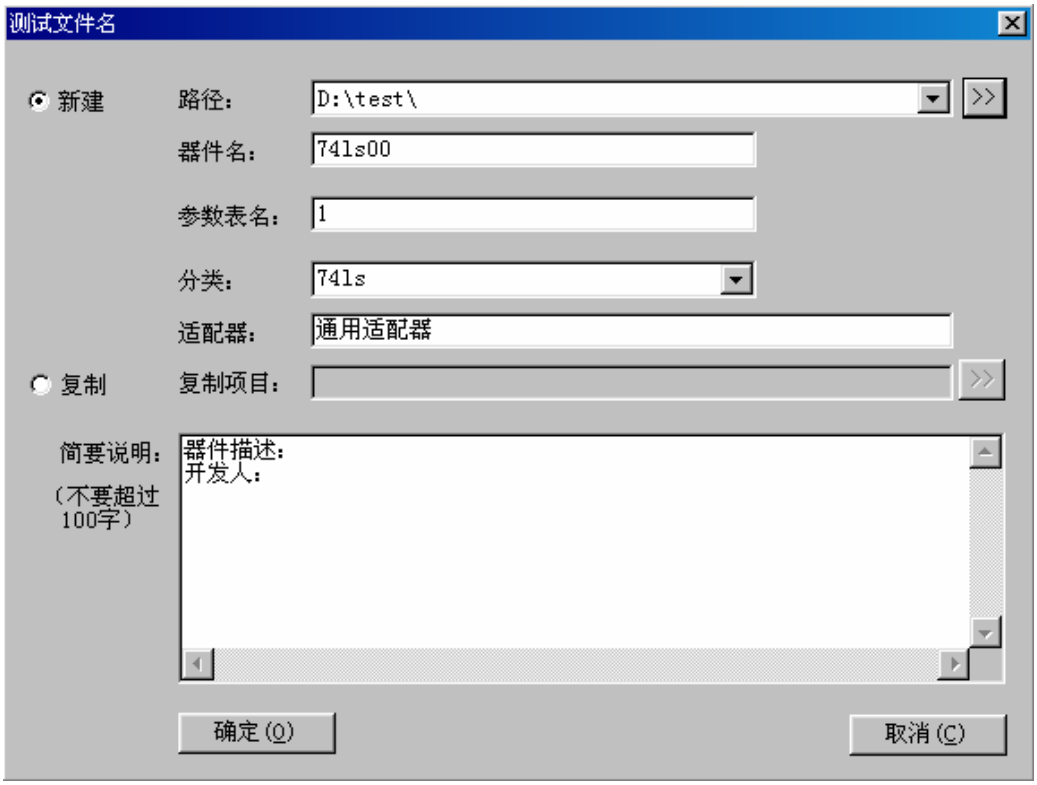


图 1-10 设置测试工程信息

（2）在“路径”框中添入测试程序所在的路径，或从下拉菜单中选择已有路径，或单击后面的 >> 按钮选择路径。

（3）在器件框中添入器件名，这样系统将在所选路径中建立与器件名同名的目录，并在该目录的 DEBUG 目录中建立同名的参数数据库。

（4）在参数表名中添入参数表名，这样将在参数数据库中建立该参数表。

（5）在“分类”列表中选择已存在的类，或输入新的类名。

（6）在“适配器”后面输入该器件所用的适配器。

（7）在“简要说明”后面的编辑框中，输入相关的开发信息，字数不要超过 500 字。

（8）本系统提供两种开发测试程序的方法，一种是新建（默认），选中“新建”按钮，然后单击“确定”按钮退出。

（9）另外，还提供复制方式，选中“复制”按钮，然后击后面的 >> 按钮打开如下的对话框选择源测试程序所在的文件夹。

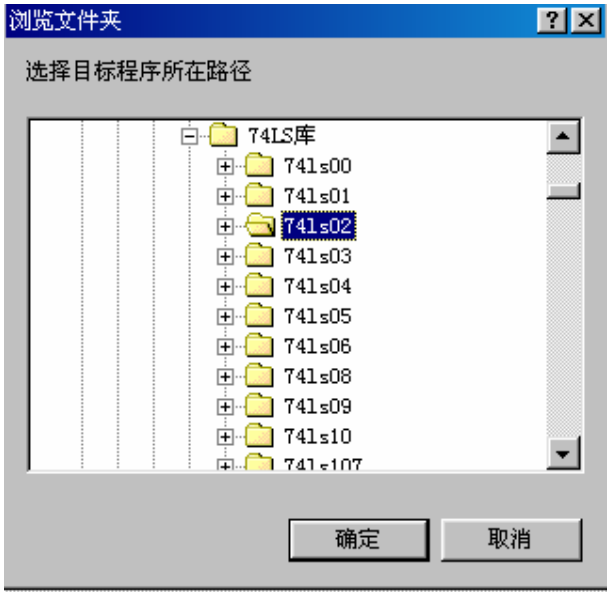


图 1-11 选择复制程序路径

（10）选择好路径以后，单击“确定”按钮退出该对话框。

（11）在图 1-11 所示的对话框中，单击“确定”按钮退出。这样开发的测试程序具有了源测试程序的测试参数、图形文件、.cpp 源文件等，只需稍作修改，便可以编译、测试了。

2. 设置测试项参数

单击“设置参数”按钮，打开如图 1-12 所示的对话框，实现参数设置的功能。

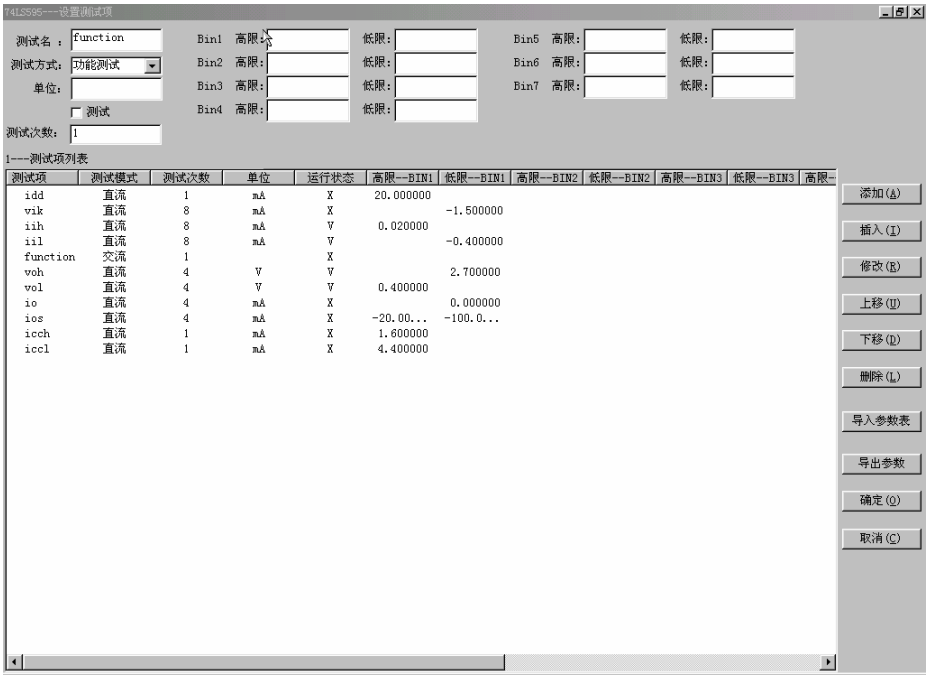


图 1-12 设置测试参数

（1）在末尾添加测试项

在“测试名”框中添入测试名，在“测试方式”中可以选择直流测试或交流测试，如果选择直流测试，则在测试函数中返回测试值，用户可以在“单位”框中输入测试值的单位，

在各 BIN 中输入高低限值；如果选择交流测试，则若测试通过，返回 0，若测试失效，返回非 0 值；若选中“运行状态”，将进行该项测试，若未选中“运行状态”，则忽略该项测试；在“测试次数”框中输入该测试项要返回的测试值数目。最后单击“添加”按钮。

（2）在中间插入测试项

首先选中测试项列表中的一项，然后如上设置测试参数，最后单击“插入”按钮，则该测试项将插入到选中测试项的前面；

（3）修改测试项

首先选中测试项列表中的一项，然后重新设置测试参数，最后单击“修改”按钮，则该测试项的参数将按设置修改；

（4）上移、下移测试项

选中某个测试项后，若单击“上移”按钮，则该测试项将上移一次；若单击“下移”按钮，则该测试项将下移一次；

（5）删除测试项

选中某个测试项后，若单击“删除”按钮，则将该测试项删除；

（6）设置完毕各项，单击“确定”按钮，则将该测试参数表保存到测试参数库中。并退出该对话框，返回主界面；

（7）导入参数表

另外，如果测试程序库已有与该器件测试方式相似的测试程序，建议使用导入参数表，具体使用步骤如下：首先单击“导入参数表”按钮，打开如图 1-13 所示的对话框；然后在列表框中选择测试程序的参数表号，然后单击确认按钮，返回“设置测试项”对话框，测试项列表将被复制，用户可在此基础上如上设置测试参数。

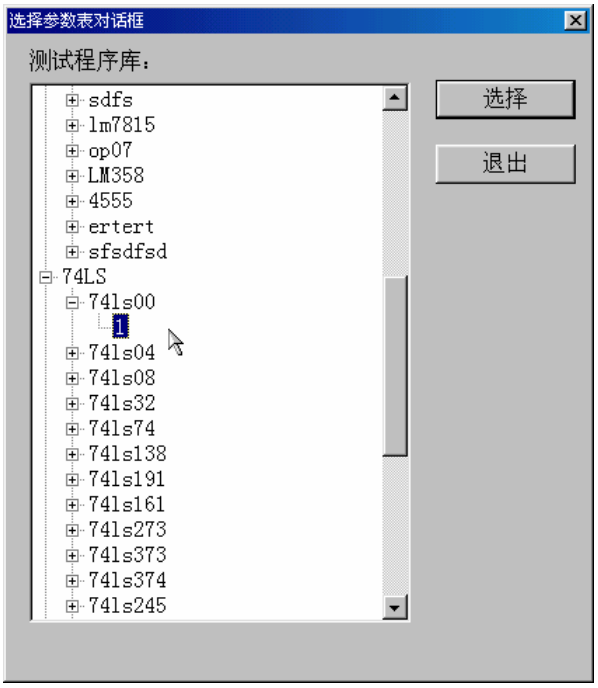


图 1-13 参数表选择对话框

（8）导出参数表

如果用户需要将当前所置的参数以文本的形式导出，那么单击“导出参数表”按钮，打开如图 1-14 所示对话框，输入文件名，然后单击“保存”按钮即可。

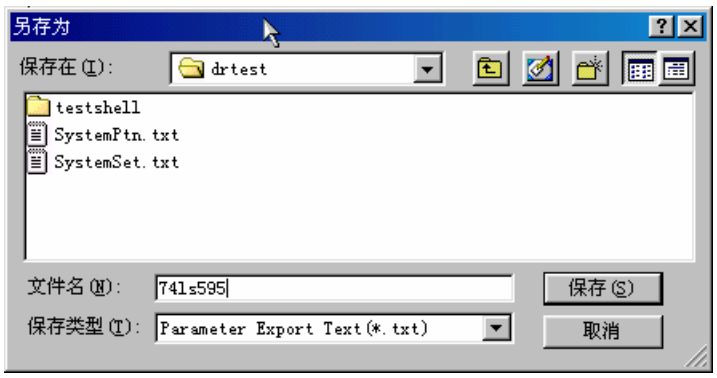


图 1-14 保存参数表

（9）右键功能

在测试项列表框中，用户可单击鼠标选择单行，也可同时按住“Shift”键连续选择多行，或同时按住“Ctrl” 键选择某几行，然后单击鼠标右键，打开如图所示的右键菜单，可实现如下功能：

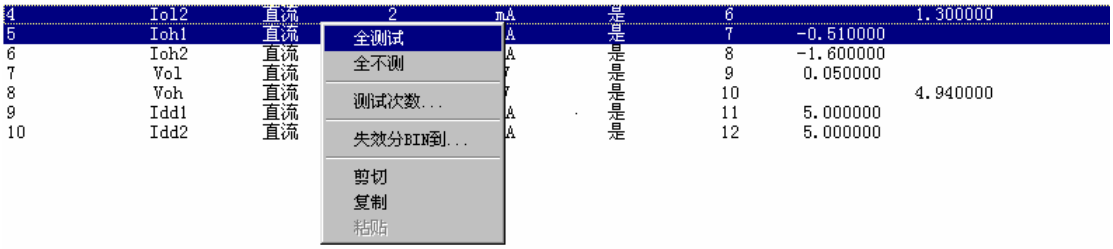


图 1-15 参数设置右键菜单

- 全测试，单击“全测试”菜单项，则将所选测试项的运行状态置为“是”；
- 全不测，单击“全不测”菜单项，则将所选测试项的运行状态置为“否”；
- 设置测试次数，单击“测试次数”菜单项，打开如下对话框，在测试次数后面的编辑框中输入一个整数，然后单击“确定”按钮，退出，这样将修改所选测试项的测试次数。

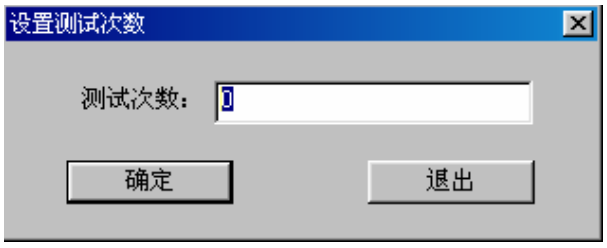


图 1-16 设置测试次数

- 设置失效分 BIN 值，单击“失效分 BIN 到...”菜单项，打开如下对话框，在失效分 BIN 后面的编辑框中输入一个整数，然后单击“确定”按钮，退出，这样将修改所选测试项的失效分 BIN 值。

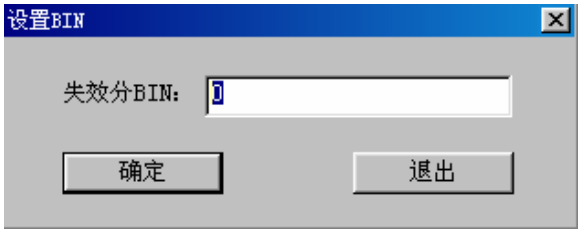


图 1-17 设置失效分 BIN

注：以上功能在未选择任何测试项的情况下，将改变所有测试项的相关信息。

- 剪切，单击“剪切”菜单项，将所选测试项信息从测试项列表中剪切下来，并保存。
- 复制，单击“复制”菜单项，将所选测试项信息保存。
- 粘贴，单击“粘贴”菜单项，将保存下来的测试项信息粘贴在所选测试项的前面，如果未选中任何测试项，则将粘贴在测试项列表末尾。

3. 编辑源代码

在主界面中单击“编辑源代码”按钮，在 Visual C++中打开新建立的测试程序，在.cpp 文件中添加各测试项的函数体。

应注意的是，用户不能在此添加新的测试项，不能更改各测试项函数的函数类型和函数名，若测试次数为 1，则返回值即为实际测试值，若测试次数大于 1，则应将测试值写回到形参数组 1 中，将所对应管脚号写回到形参数组 2 中；对于交流测试，则通过值为 0，失效为任何非 0 值。

编辑完毕，将该测试程序编译成动态链接库，然后在测试界面中单击“测试”按钮，打开运行界面，在其中单击“单次测试”进行调试。

4. 编制测试图形

对于数字类型的器件，需要编辑并装载测试图形，单击“编制图形”按钮，按向导所示进行操作。详细说明请参看本手册第二章“图形编辑界面使用说明”。

五　测试器件

测试器件需要 6 步：

设置输出路径→选择测试程序→修改测试参数（不是必须）→查看或修改源文件（不是必须）→BIN 设置→进入测试界面。

进入测试界面单击“运行”按钮或“单次测试”按钮进行器件测试。

以下将详细说明。

1. 输出路径设置

首先选择操作员及测试数据库输出路径设置。如果用户不关心这些信息，也无需保存测试数据，那么可省略这一步；否则按如下步骤进行设置：

（1）在主界面中点击“操作员设置”按纽，打开如图 1-18 所示的对话框。

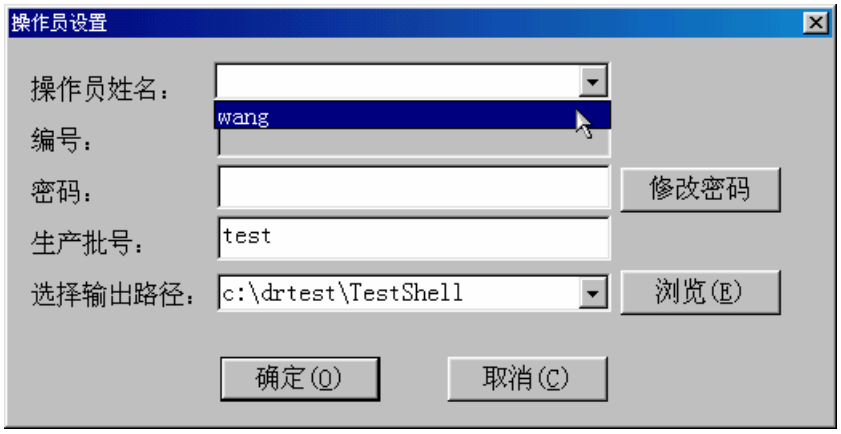


图 1-18 操作员选择对话框

（2）在“操作员姓名”列表中选择操作员，在“编号”后面的编辑框中将显示该操作员的编号。

（3）在“密码”后面的编辑框中输入该操作员的密码。

（4）如果要修改该操作员的密码，则单击“修改密码”按纽，打开如图 1-19 所示的对话框。输入原密码后再输入两次新密码。单击确定按钮返回。

图 1-19 修改用户密码

(5)在“生产批号”后面的编辑框中输入器件的生产批号。

(6)在“选择输出路径”后面的下拉列表中选择数据库所在路径，也可单击后面的“浏览”按纽，打开如图 1-20 所示的对话框，在其中选择输出路径；然后单击“确定”按纽退出该对话框，返回操作员设置对话框。

图 1-20 选择输出路径

（7）单击“确定”退出操作员设置对话框，使设置有效，返回到主界面；这样就在所设输出路径目录中建立以生产批号命名的数据库，测试时将测试数据保存到该数据库中。

2. 选择测试程序

选择测试程序的步骤如下：

（1）在主界面中单击“选择”按钮，打开的如图 1-21 所示的对话框。

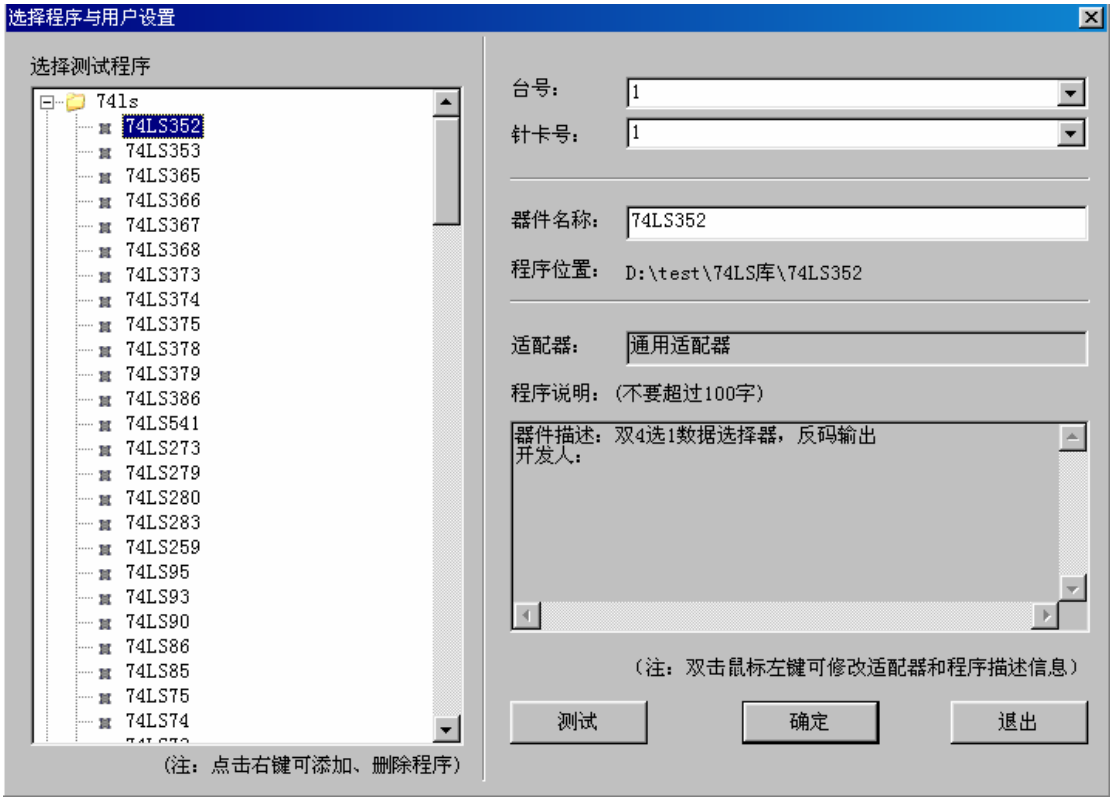


图 1-21 选择测试程序

（2）在“选择测试程序”下面的树形列表中按类显示了程序库中所有的测试程序，单击类所在的结点，选择其中的测试程序。

（3）测试程序维护，在测试程序列表中可通过右键菜单进行测试程序的维护，方法如下：

- 修改类名，在要修改的类名上，单击右键，打开如图所示的菜单，然后单击“修改类名”菜单项，在打开的对话框中添加新的类名，然后按“确定”退出（详见前面测试程序维护部分 P7）

图 1-22 右键菜单

- 删除类，要删除的类名上，单击右键，打开如上图所示的菜单，然后单击“删除类”菜单项，即可将当前类及其包含的全部测试程序从中删除；但应注意的是，此处只是将测试程序从登记表中删除，并未将实际的测试程序删除，如果需要删除测试程序，还应在资源管理器中将整个的测试项目从硬盘中删除。
- 添加测试程序，在列表框的任何位置单击鼠标右键，打开菜单，单击“添加测试程序”菜单项即可实现。（详见前面测试程序维护部分 P7）
- 删除测试程序，在要删除的测试程序上单击单击鼠标右键，打开菜单，单击“删除测试程序”菜单项。（详见前面测试程序维护部分 P8）

（3）在“台号”、“针卡号”以及“器件名称”后面输入相关信息。

（4）“程序位置”显示了所选测试程序在磁盘中的存储位置。

（5）“适配器”和“程序说明”中分别显示了开发测试程序时所输入的相关信息，这些信息可通过双击鼠标键进行修改。

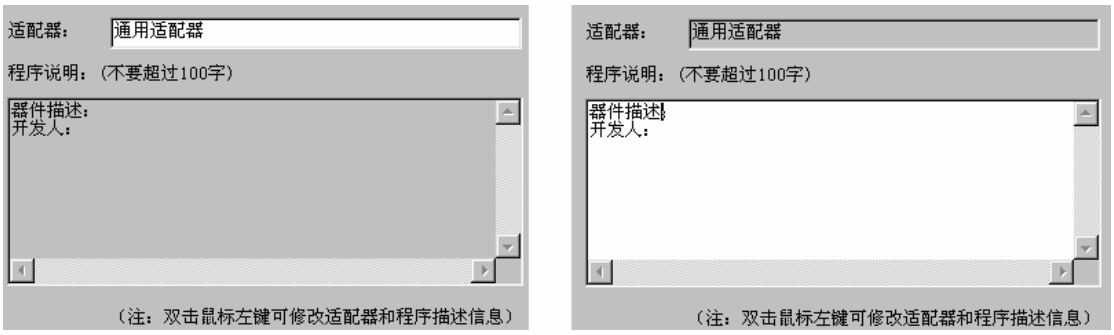


图 1-23 双击鼠标修改“适配器”和“程序说明”信息

（6）单击“确定”后，如果经检查，该程序存在，则选择成功，退出该对话框，返回主界面；如果在所显示位置找不到该程序，则在对话框下方，显示“该测试程序不存在，请重新选择!”提示用户重新选择其它程序，并且所选程序将被删除。

（7）单击“测试”按钮，将直接进入测试界面。

3. 修改或查看测试参数

如果用户具有管理员身份，则有修改测试参数和源码的权限，可以修改或查看测试参数，当然，这一步也可以省略；如果需要则在主界面中单击“设置参数”按钮，打开如图 1-18 所示的对话框，用户可进行下述操作：

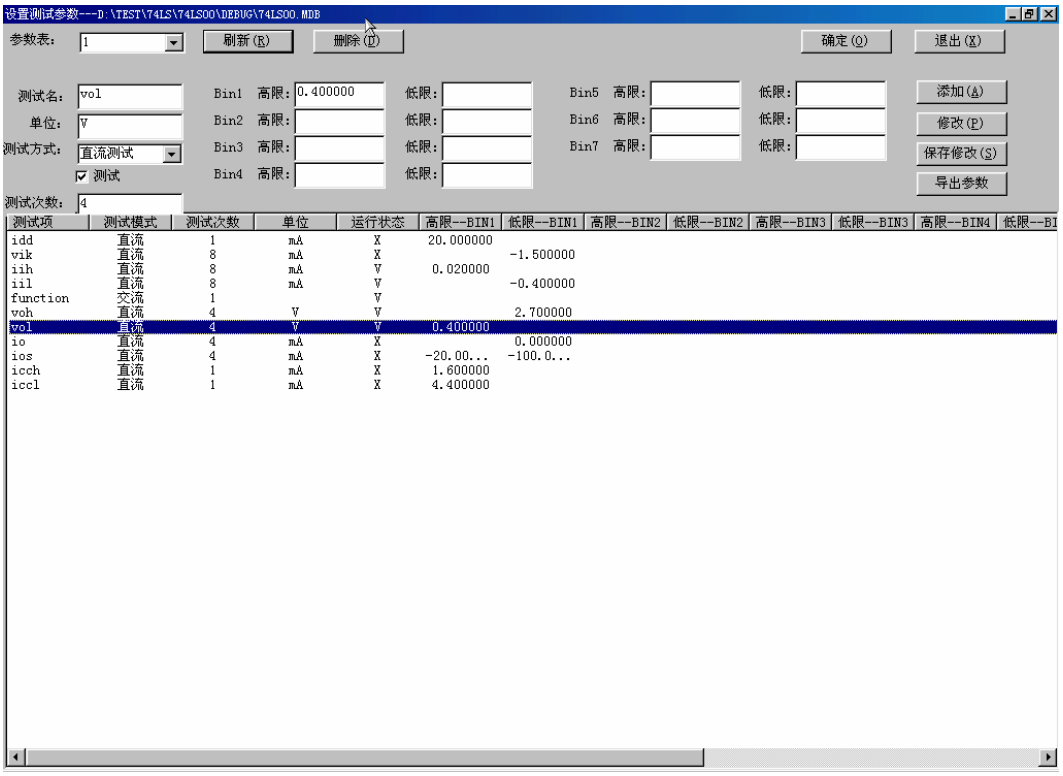


图 1-24 参数修改对话框

（1）选择参数表

如果该器件有多个测试参数表，则可以选择每次测试时所要用的参数表，这些参数表的名字排列在“参数表”后面的列表框中，用户可以从中选择测试要用的参数表，然后单击“刷新”按钮，则测试参数将显示在下面的列表框中。

（2）删除参数表

从“参数表”后面的列表框中选择测试要用的参数表，然后单击“删除”按钮，可将当前参数表删除。

（3）添加测试项

和上述开发测试程序中设置参数相同，在各个编辑框中设置好测试项的各项指标后，单击“添加”按钮，可将测试项添加到测试项列表的末尾；同时在该测试程序的.cpp 文件末尾也加上了该测试项，用户应在设置完测试参数后，去修改源程序，并重新编译。

(4)修改测试项

在测试项列表中选中要修改的测试项，然后修改其各项指标，修改完毕单击“修改”按钮，使修改生效。

(5)保存修改后的参数表

所有参数修改完毕，单击“保存修改”按钮，则将参数保存到“参数表”后面的表名中。

(6)导出参数表

详见“开发程序”的“设置参数”部分

单击“确定”按钮，退出参数设置，返回到主界面中，参数保存到“参数表”后面的表名中，且器件测试参数如表中所设；

(7)在测试项列表中单击鼠标右键，打开右键菜单，可实现一些辅助功能，详见测试程序开发部分(P12)。

4. 修改或查看源代码

如果管理员需要查看或修改源代码，则单击“编辑源代码” 按钮，则可在 Visual C++ 下进行源码编辑；否则可省略这一步。

5. 机械手、探针台以及 BIN 设置

单击“BIN 设置”按钮，打开如下的对话框，

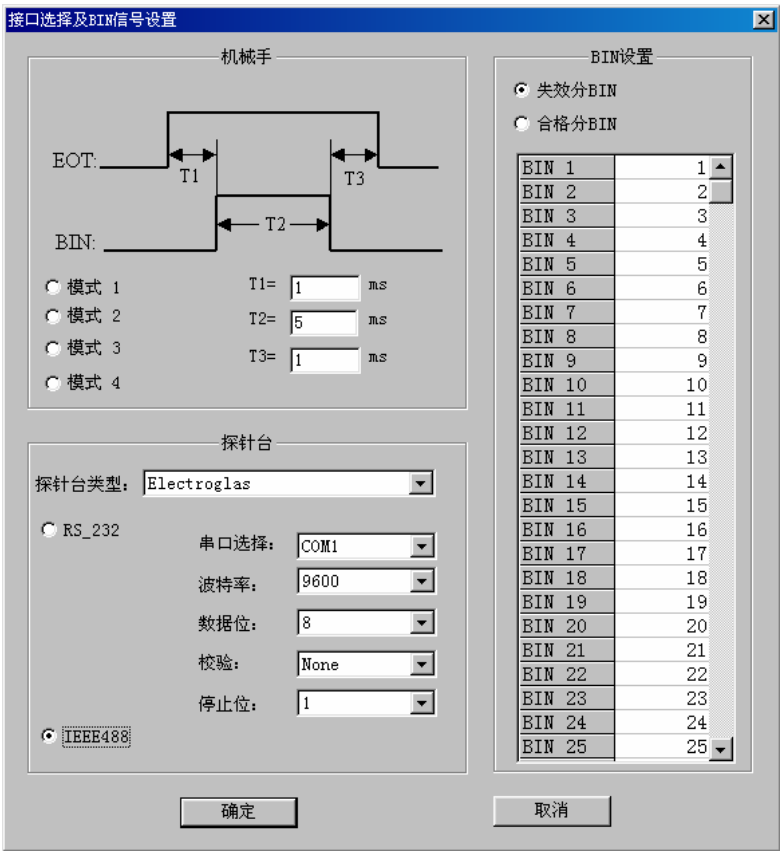


图 1-25 接口选择及 BIN 设置

其中可以设置机械手、探针台以及 BIN 信号，其中机械手有四种模式可供选择，以适应不同的情况。具体模式如下：

（1）模式一：下降沿有效，先发 BIN 信号，再发 EOT 信号，如图所示。

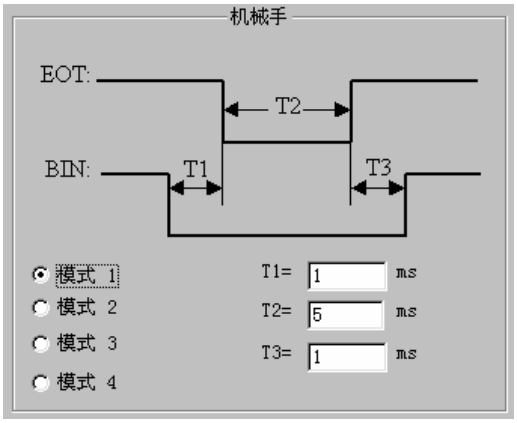


图 1-26 模式一

（2）模式二：下降沿有效，先发 EOT 信号，再发 BIN 信号如图所示。

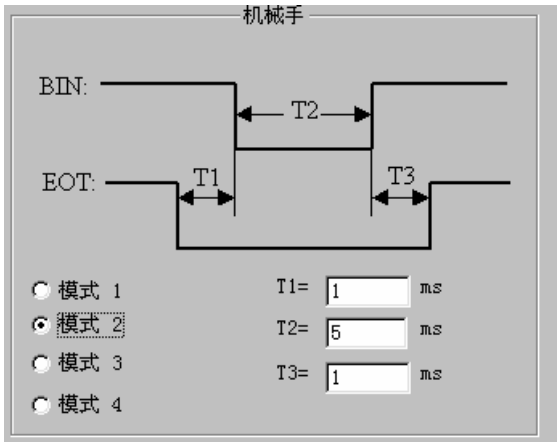


图 1-27 模式二

（3）模式三：上升沿有效，先发 EOT 信号，再发 BIN 信号，如图所示。

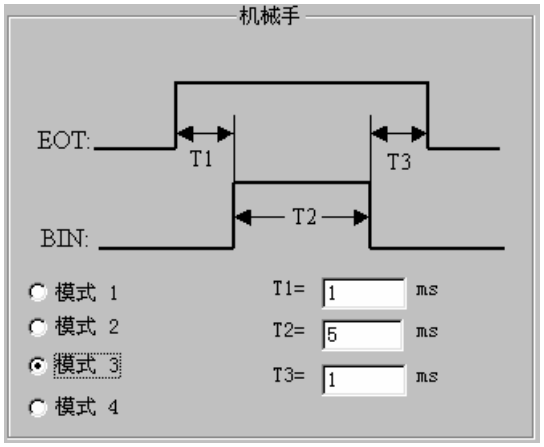


图 1-28 模式三

（4）模式四：上升沿有效，先发 BIN 信号，再发 EOT 信号如图所示。

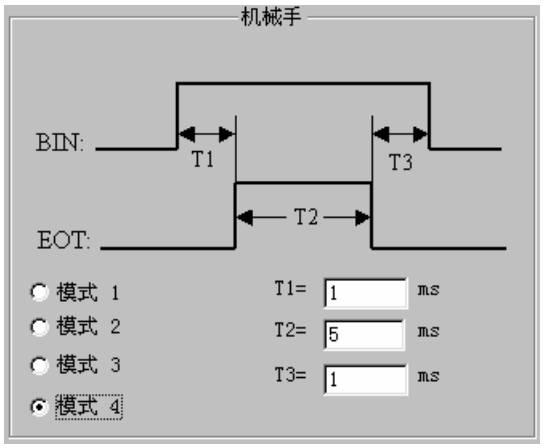


图 1-29 模式四

探针台种类因用户的需求而不同，用户可在在“探针台类型”后面的下拉列表中进行选择，如图所示

图 1-30 选择探针台

接口方式有 RS232C 和 IEEE488 两种选择，当选择 RS232C 方式时，还应做如下设置

（1）选择所用串口，如图所示

图 1-31 选择串口

（2）选择波特率，如图所示

图 1-32 选择波特率

（3）选择数据位个数，如图所示

图 1-33 选择数据位个数

（4）选择奇偶校验位或无校验，如图所示

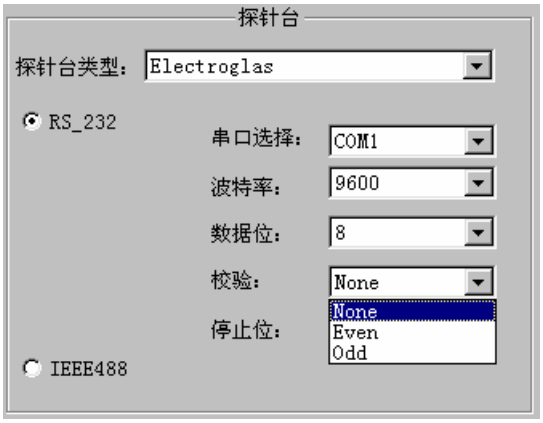


图 1-34 选择校验方式

（5）选择停止位个数，如图所示

图 1-35 选择停止位

设置分 BIN 方式及 BIN 信号值，如图所示

图 1-36 设置 BIN 信号

如选择“失效分 BIN”，则只有 BIN1 是合格的，其它 BIN 值均为失效；如选择“合格分 BIN”，则只有 BIN1~BIN8 均是合格的，其它 BIN 值为失效；

修改 BIN 值时，用鼠标双击表格单元，变成如图所示状态时，就可以输入新的 BIN 值

所有信息设置完毕，单击“确定”按纽，这样每测完一个器件，将按此处定义的信号发送给测试仪。

6. 测试

打开 3168 测试系统电源，单击“运行”按钮，进入测试界面，如图所示。

图 1-37 测试界面

测试界面完成对器件的测试、测试数据的显示、保存、数据分析等功能，以下详细叙述各个功能。

（1）若选中“完全测试”，则测试时将无论 PASS 与否，都要继续进行测试，否则，只要有失效项，就停止对该器件的测试。

（2）如果希望在每次测试失效后要重测几次，那么应在“失效后重测”后面的编辑框中添入次数，这样在测试失效后将重复测试；

（3）显示设置，显示设置的对话框如图所示。

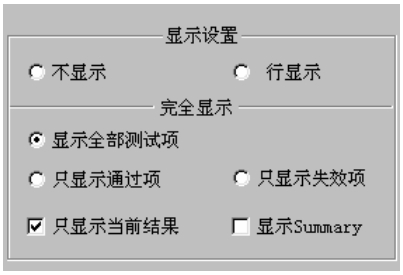


图 1-38 设置显示模式

- 若选中“不显示”，则在测试过程中将不显示测试数据。
- 若选中“行显示”，则在测试过程中将在一行中显示所有的测试数据。

在“完全显示”下面的各选项中，用户将看到完整的测试数据显示

- 用户可以选择显示全部测试项、通过项还是只显示失效项。
- 若选中“只显示当前测试结果”，那么在测试过程中将只显示当前被测器件的测试数据，否则将在原有测试记录后面继续追加显示结果。
- 若选中“显示 Summary”，则每次测试完毕，将在测试结果末尾追加总结信息。

（4）数据存储设置，数据存储设置如图所示。

图 1-39 设置存储信息

- 在“生产批号”后面的编辑框中输入器件的生产批号，它将作为输出数据库的文件名。
- 在“元片号”中输入元片号，它将作为数据库中的数据表名。
- 在“输出路径”后面的编辑框中显示输出数据库所在路径，默认为运行程序（.dll）所在的目录，用户可单击后面的“>>”按钮，打开如图所示的对话框，在其中选择输出路径；单击“确定”按钮退出该对话框。

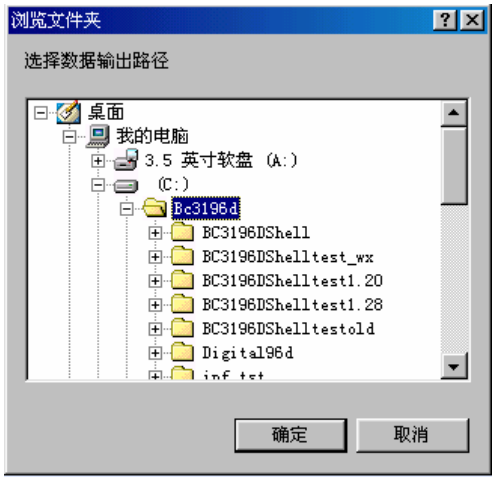


图 1-40 输出路径选择

（5）在调试器件时，可单击“单次测试”按钮，只进行一次测试，该操作不改变测试数量，也不会写数据库，但是会将测试结果以文本形式保存在测试程序所在目录中，文件和程序名相同，后缀为.rep，可以在记事本中打开。

（6）批量测试，单击“运行”按钮，开始进行测试；如果所设置测试数据库已存在，则会弹出如图所示对话框，如果想覆盖原来的则单击“是（Y）”按钮，如果想继续原来的数据库保存，则单击“否(N)”按钮，如果不想继续测试，而去修改其他的设置，则单击“取消”按钮。

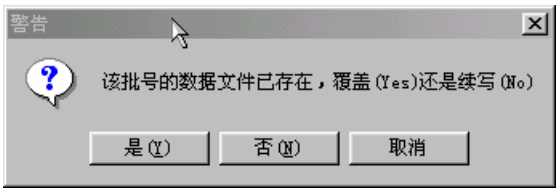


图 1-41 数据库覆盖或续写警告信息

如果没有取消测试，则如果接收到测试信号，就开始测试，并按设置模式在右面显示测试数据，在最上面显示统计数据的结果，在右下方显示每次测试所用的时间和上、下片时间。

（7）在测试过程中，用户可单击“暂停”按钮，暂停测试；或单击“停止”按钮，停止测试。

（8）停止测试后，可单击“数据分析”按钮，在打开的数据分析对话框中对测试数据进行分析。（参看下面的数据分析部分）。

（9）波形分析；如果用户在测试过程中进行了波形采集操作，那么可单击“波形分析”按纽，打开波形分析对话框。（参看下面的波形分析部分）。

（10）单击“清除报告”，将清除右面编辑框中的显示内容。

（11）单击“导出报告”，将把右面编辑框中的显示内容导出到文件中。

（12）单击“退出测试”按纽，退出本次测试。

六 程序调试

用户在开发测试程序时，一般都要经过调试，修改、再调试这样的过程才能完成一个器件的测试程序的编写。用户开发的测试程序为普通的动态链接库程序，不具备独立运行的能力，因此调试和跟踪程序必须选择调用的主程序，调试步骤如下：

（1）编辑好测试源程序，进行编译，编译通过后才可以进行调试。

（2）将光标移至程序中要跟踪的地方，然后按 Ctrl+F10，或者设置断点按 F5 键。对于第一次启动调试的程序，必须指定调用程序的路径。启动调试 VC 调试器弹出如图所示的对话框，单击浏览按钮选择调用程序。如果已经指定过执行程序，将直接启动调用程序。

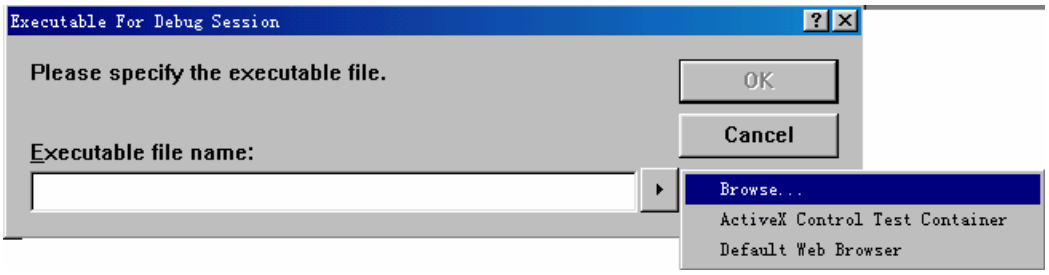


图 1-42 指定调用程序

在  按纽后面的子菜单中单击 Browse 按纽，打开文件选择对话框，如图所示，然后选择 C:\drtest\Testshell.exe 应用程序，单击“打开”按钮。

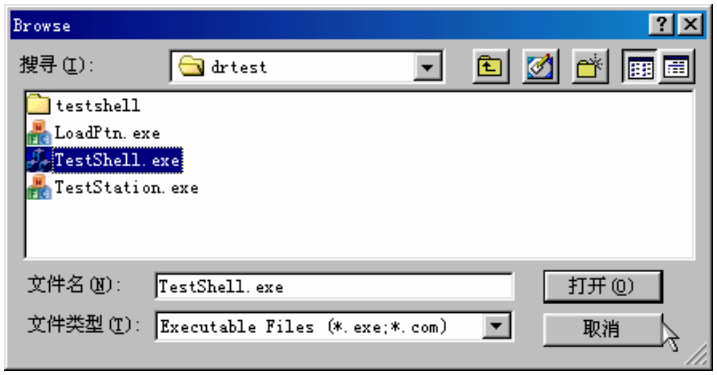


图 1-43 程序选择对话框

单击“打开”按钮后，调用程序启动。

（3）在主界面中单击“管理员设置”按纽，输入管理员密码，然后单击“确定”按纽，退回主界面。

（4）在主界面中单击“选择”按钮，在打开的“选择测试程序”对话框中选择刚刚编译好的测试程序，（注意不要选错），然后单击“确定”按钮，返回主界面。

（5）在主界面中单击“测试”按钮，打开运行界面，单击“单次测试”按钮，即可回到测试程序中，按 F10 将单步执行程序，如果将光标移动到到程序中某处，再按 Ctrl+F10，则将跟踪至该点。

七 数据分析

在主界面中单击“数据分析”按钮，打开选择文件对话框，选择分析数据库，如图所示。

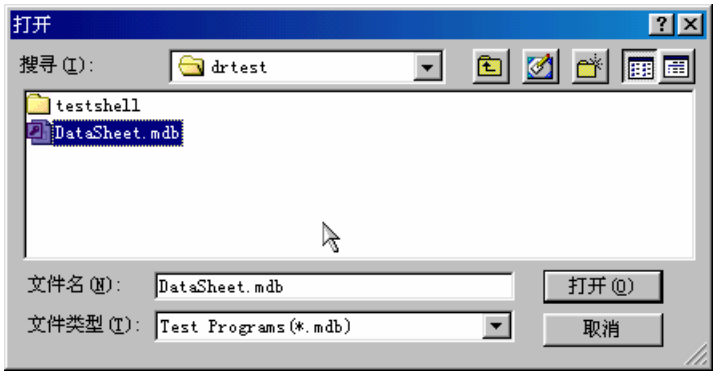


图 1-44 选择分析数据库。

单击“打开”按钮，打开数据分析对话框，如图所示。

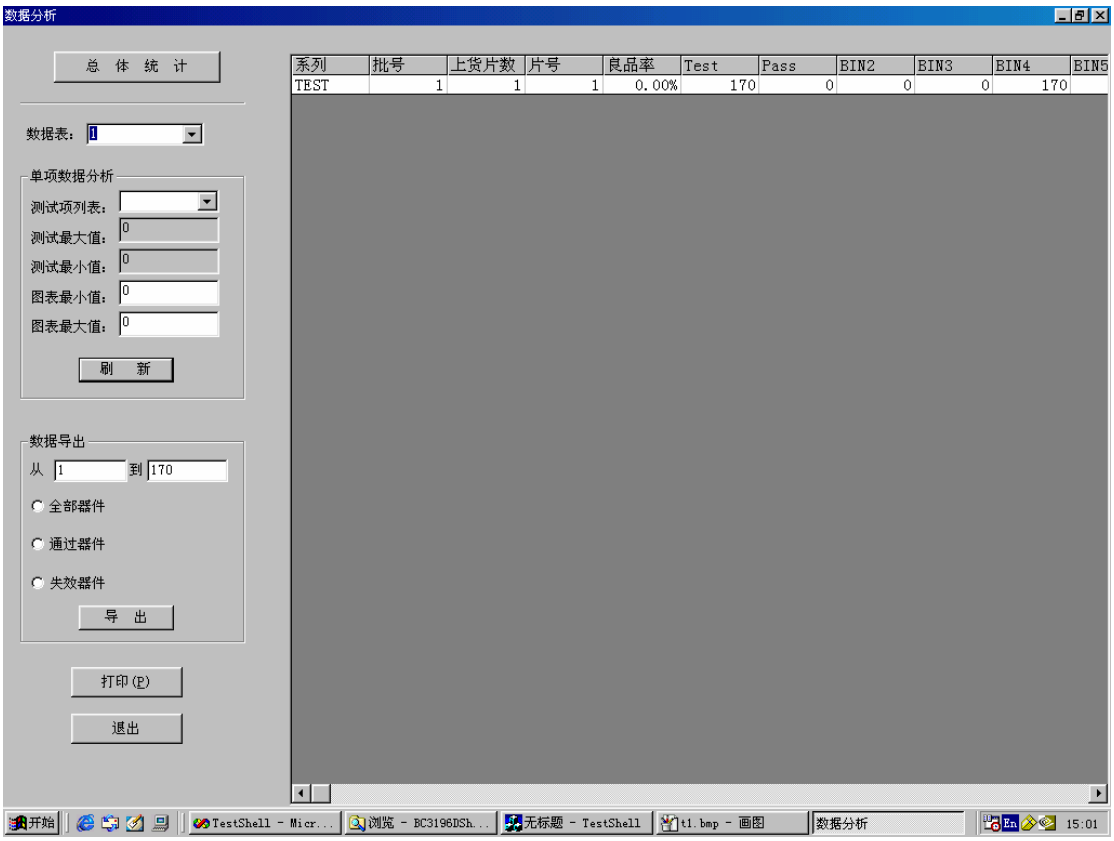


图 1-45 数据分析对话框

（1）单项数据分析。在“数据表”后面的下拉列表中选择不同的数据表，默认打开的是最后一次测试的数据表；按图进行操作。

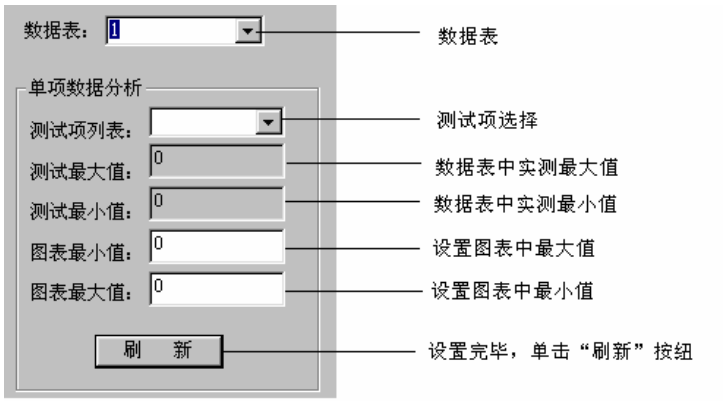


图 1-46 单击数据分析

单击“刷新”按钮后，在右边的图表中则将所设图表最大值和最小值范围分成十份，进行显示，如图所示。

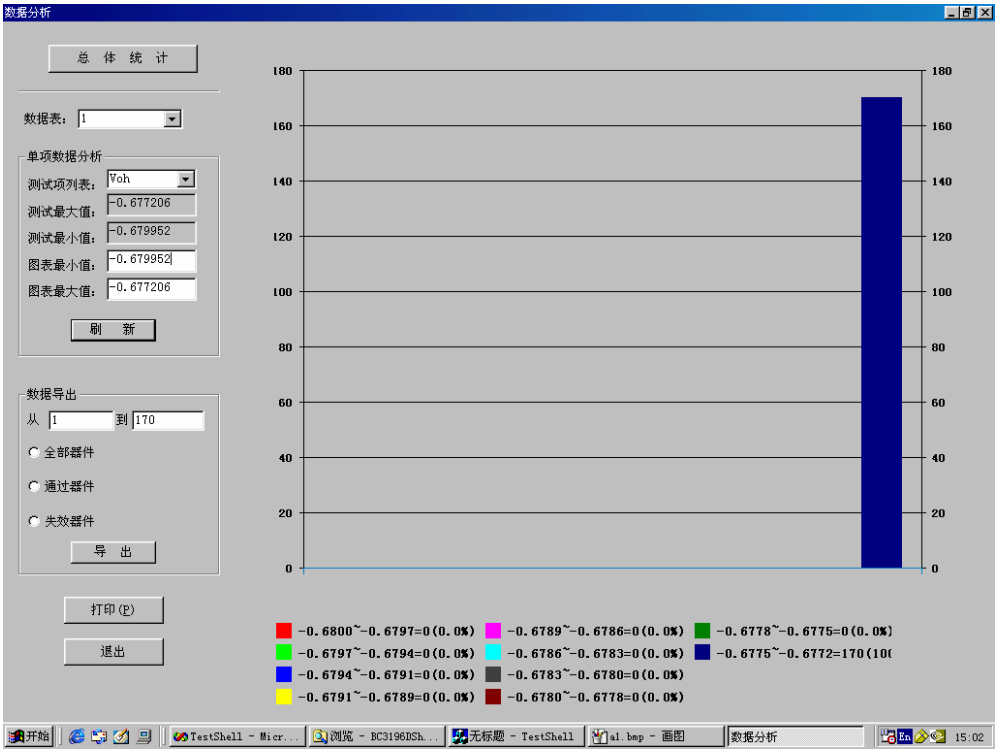


图 1-47 单项数据分析

（2）数据导出。数据分析提供了导出功能，用户可以部分或全部的导出测试数据，供用户查看和分析，导出设置如图所示。

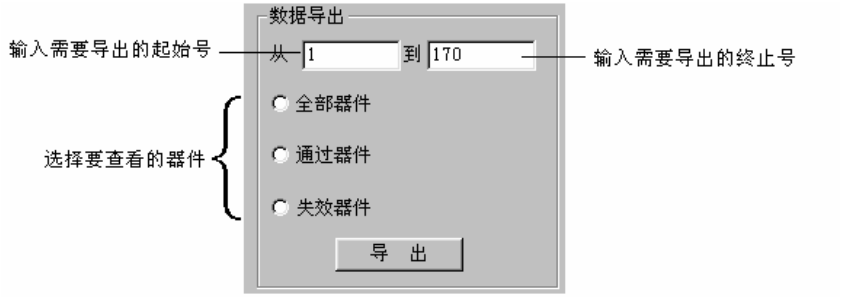


图 1-48 导出设置

设置完毕，单击“导出”按钮，弹出保存文件对话框，如图所示。下所示对话框，选择保存路径，并设置好文件名，然后单击“保存”按纽，就将符合条件的器件的测试数据以文本的形式进行保存。

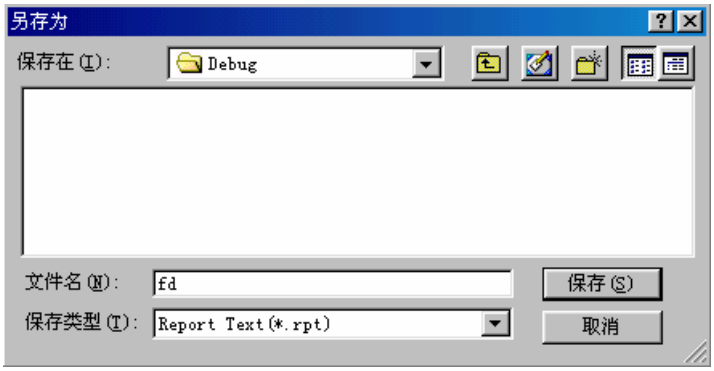


图 1-49 导出文件设定

（3）单击“打印”按纽，可打印整个对话框中显示的分析结果。

单击“退出”按纽，退出数据分析。

第二章 图形编辑界面使用说明

一. 概述

BC3196D 数模混合集成电路测试系统的软件部分包括两个部分：

（1）TestShell 用户软件，主要用于完成开发、测试和管理用户测试程序。

（2）TestStation 图形编辑软件，专门为快速开发数字/数模芯片设计，用于完成数字芯片配置、测试图形的编写和调试。该软件由系统软件 TestShell 自动开启。

本章主要介绍用于东润集成电路测试系统的高性能图形编辑/调试软件系统的功能和使用方法。

该软件系统为数字芯片以及数模混合芯片测试程序的开发和调试提供了最大的便利，为为东润公司所有数模和数字测试系统的通用软件。图形编辑/调试系统具有以下几个主要功能：

- 数字芯片的向导式开发，使用户非常容易的创建数字芯片测试。
- 数字芯片管脚分配、分组，方便直观，很容易管理器件和测试图形。系统采用组管理被测器件管脚和测试图形。测试图形创建后，分组仍可以修改。
- 测试图形可以有多种编辑格式，以便用户在不同情况下查看和编辑图形。图形的编辑格式有：逻辑、二进制、十进制、八进制和十六进制。每个管脚组可以分别设置和任意更改编辑格式。
- 数字芯片测试参数预设功能，包括系统时钟设置、参考电平设置、时序设置和图形运行方式设置等多项参数，使用户可以几乎不编写代码，就可以完成数字芯片的测试。
- 测试图形的编写采用所见即所得的编辑方式。在编辑的同时查找出语法错误，使用户只需关心测试芯片的逻辑和时序即可。
- 文本编辑器提供强大的文本编辑功能，提供单行/多行图形的剪切、复制、粘贴、插入和删除操作；一列/列中部分图形整体修改命令，包括列图形置高、列图形置低、列图形 X/Z 和指定任意值，以及修改组中的任意一位图形。一个图形组称为一列。
- 测试图形调试功能。可以有选择加载用户测试程序的功能测试函数和指定调试的图形段，从而完成器件功能的调试工作。调试界面用红色、蓝色和绿色分别显示期望图形测试失效、通过和未测试。出错图形会自动显示实际测试值。图形调试工具包括：运行图形、单步运行图形、运行到光标处、运行到指定行、停止运行、更新图形等工具，最快速的帮助用户调试测试图形。
- 加载图形和加载配置。使用加载图形按钮可以将测试图形和预设参数加载到测试仪，也可以单独加载预设的参数。
- 导入/导出配置功能，帮助用户快速的新建新的测试程序的图形文件。
- 导入/导出图形功能，提供了与其他 EDA 软件和其他 ATE 的接口。
- 应用程序注册到系统中，可以单独启动查看、修改和创建测试图形。

本章详细介绍如何使用图形编辑/调试界面开发和配置数字芯片以及调试图形。本章包括 4 部分内容：

（1）主界面说明
（2）创建测试图形
（3）编辑测试图形
（4）调试测试图形

二. 主界面说明

打开测试图形文件或创建测试文件完毕后就进入主操作界面，如图 2-1 所示。主控界面由菜单、工具条、窗体和状态栏组成。

图 2-1 主操作界面

主界面功能明晰，操作简单，采用单文档窗体，所有的操作均在一个窗口完成。通过菜单、工具条和主窗口的功能按钮可以迅速完成对测试图形的各种操作。单击编辑图形，即可以打开测试图形编辑窗口，如图 2-2 所示。

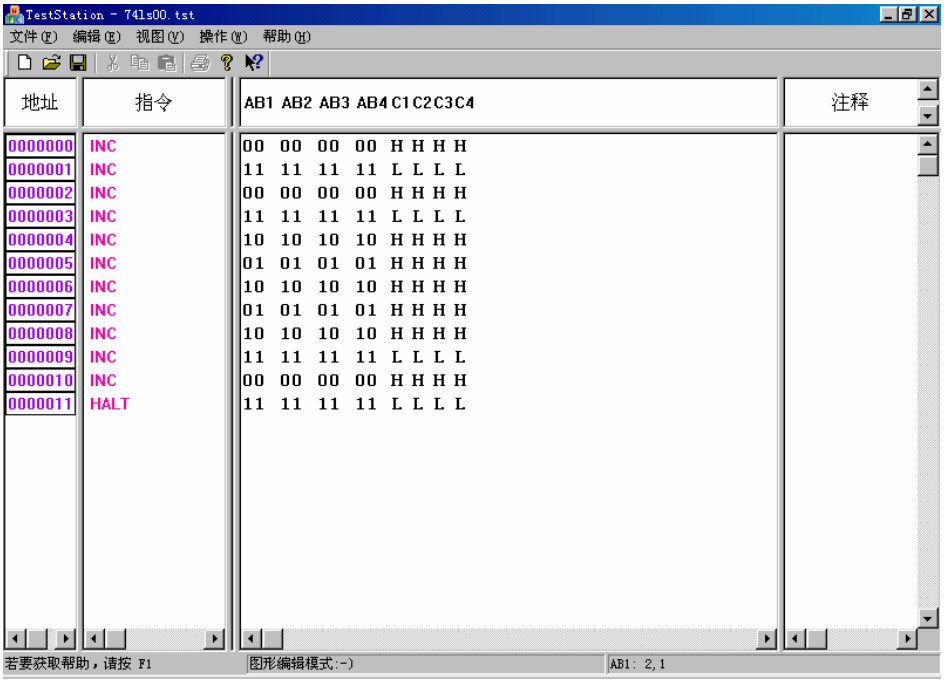


图 2-2 图形编辑窗口

执行菜单“操作”→“主控界面”命令，可以切换到主控窗口，也可以使用 Ctrl+M 快捷键完成操作。

图形编辑窗口用于完成测试图形的编辑、查看和测试图形的调试。

以下详细介绍主界面的各个部分：

1. 菜单

菜单提供了界面操作的各种命令，基本菜单有 5 项，调试菜单只有在调试状态才会显示。

（1）文件菜单

文件菜单提供了对测试文件的 7 个基本操作：

- “新建”命令，用于重新创建一个测试图形文件，扩展名为.tst。
- “打开”命令，用于重新打开一个已存在的测试图形文件扩展名为.tst。
- “保存”命令，用于保存用户对图形文件的修改。
- “另存为”命令，用于将当前测试图形文件保存为新的图形文件。
- “导入/导出图形文本”命令，用于将测试图形保存为普通文本，可供用户在其他机器上查看或打印，以及导入其他 ATE 的测试图形，或 EDA 工具生成的测试图形。本软件目前支持 Tri6010 的图形格式导入。同时可以为用户开发其他的系统图形导入软件。
- “导出配置”命令，用于将芯片的配置信息保存，供新建图形文件使用。
- “退出”命令，用于退出应用程序。

（2）编辑菜单

编辑菜单提供了对图形文件操作的各种命令，共有 9 个命令，这些命令专为图形编辑窗口提供。分为两类：图形的行操作命令和列操作命令。

行操作命令共有 5 个，如下：

- “剪切”命令，用于将选中的一行或多行图形放入剪切板中，并删除所选图形
- “复制”命令，用于将选中的一行或多行图形放入剪切板中。
- “粘贴”命令，用于将剪切板中的图形插入到当前行之前

- “插入”命令，用于插入一行图形模板。新增一行图形多使用该命令。
- “删除”命令，用户删除选中的一行或多行图形。

列操作命令共有 4 个，用于对特定的管脚组的图形或组中某一管脚的图形的操作。

- “列图形置高”命令，用于将选中的一个图形组的全部或部分图形置为高。
- “列图形置低”命令，用于将选中的一个图形组的全部或部分图形置为低。
- “列图形置 Z/X”命令，用于将选中的一个图形组的全部或部分图形置为高阻（输入图形）或不关心（输出图形）。
- “列图形编辑”命令，用于将选中的一个图形组的全部或部分图形置为特定的值，或将组中某一器件管脚的图形修改为特定值。

图形文本的编辑命令的详细说明将在“图形编辑器”一节详细说明。

（3）视图菜单

用于控制是否显示工具栏和状态栏。

（4）操作菜单

用于进行对图形各种操作，包括 5 个命令：

“主控界面”命令，用于切换到主控界面窗口。

“编辑图形”命令，用于打开图形编辑窗口。

“调试图形”命令，用于进入调试图形状态，当进入调试模式，操作菜单无效，只有退出调试时才有效。

“加载图形”命令，用于将测试图形和配置信息加载到测试仪中。

“加载配置”命令，用于将图形的配置信息加载到测试仪中。

（5）帮助菜单

提供系统的帮助。

（6）调试菜单

调试菜单提供调试图形时，所需要的各种命令，详细说明见“调试图形”一节。

2. 工具条

工具条为最常用的命令提供了快捷的操作方式。工具条包括普通工具条和调试工具条两个。在调试状态，调试工具条自动显示。

普通工具条包括新建、打开、保存、剪切、复制、粘贴 6 个常用的命令。

调试工具条包括所有的调试工具。

3. 窗体

窗体上为 6 个常用的命令按钮，用于快速执行最主要的图形操作：

（1）参数设置按钮：用于打开参数设置对话框，完成对芯片的配置和图形显示方式的修改。

（2）编辑图形按钮：用于打开图形编辑窗口。

（3）加载图形按钮：用于向测试仪加载测试图形和配置信息。

（4）调试图形按钮：用于启动图形调试。

（5）加载配置按钮：用于向测试仪加载配置信息

（6）退出界面按钮：用于结束应用程序

4. 状态栏

状态栏，为各种命令操作提供了提示。任务栏分为 6 个部分。

（1）窗口 1 用于显示各种在线操作或解释各种命令。

（2）窗口 2 用于标识当前的操作状态。

（3）窗口 3 用于显示被编辑的管脚组的器件管脚号/通道号。

（4）最后三个用于显示 CAP 键、Num 键和 Scroll 键的状态。

5. 系统操作快捷键:

本系统软件除了提供了菜单、按钮等操作命令，同时也为最常用的操作提供了快捷键，使用户可以不用点击按钮或菜单即可以完成操作。

本系统软件的快捷键分配，保持标准 Windows 快捷方式，同时对系统独有命令重新分配。

（1）继承的 Windows 标准快捷键有：

新建测试图形	Ctrl+N
打开测试图形	Ctrl+O
保存测试图形	Ctrl+S
剪切图形	Ctrl+X
复制图形	Ctrl+C
粘贴图形	Ctrl+P

（2）独有的快捷键

插入一行图形	Insert
删除图形	Ctrl+D
切换到主控界面	Ctrl+M
启动调试	F6
结束调试	Shift+F6
加载图形	F7
加载配置	Ctrl+F7

（3）调试状态的快捷键，基本与 VC 编辑环境相似：

功能测试	Ctrl+F5
运行图形	F5
停止运行图形	Shift+F5
运行到..	F9
运行到光标	Ctrl+F10
单步运行	F10
更新测试图形	F8

三. 创建测试图形

当主测试程序选定一个测试程序，而该测试程序没有测试图形文件时，或主测试程序创建一个新测试程序时，单击“编辑图形”按钮，弹出如图 2-3 所示的对话框，提示用户是新建还是打开已存在的图形文件。

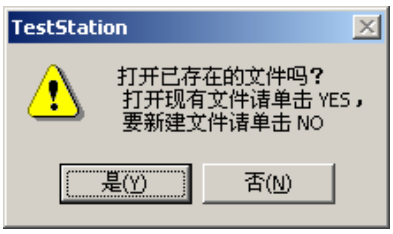


图 2-3 选择打开或新建文件

单击“是”按钮打开一个存在的图形文件，系统自动将打开的图形文件复制到当前测试程序目录中，并重新命名为与测试程序同名的图形文件，文件的扩展名为“.tst”。

当单击“否”按钮时，启动向导，创建一个新的芯片文件，如图 2-2 所示。按照向导的帮助可以很容易的创建一个数字芯片配置和图形文件。

只有完成向导之后，才能主界面。向导主要完成对被测器件进行分配通道、设定 IO 方向、管脚分组、分配算法图形资源和预设测试参数。不完成向导，不能进行图形的编辑。向导完成后，所有的配置均可以修改。

向导一共包含六个对话框，3 个是用于完成器件管脚分配和管理的；三个是用来完成预设参数设置的，这部分内容将作为配置文件，成为测试图形的一部分。用户只需点击鼠标就可以完成全部配置工作。

1. 芯片设置对话框

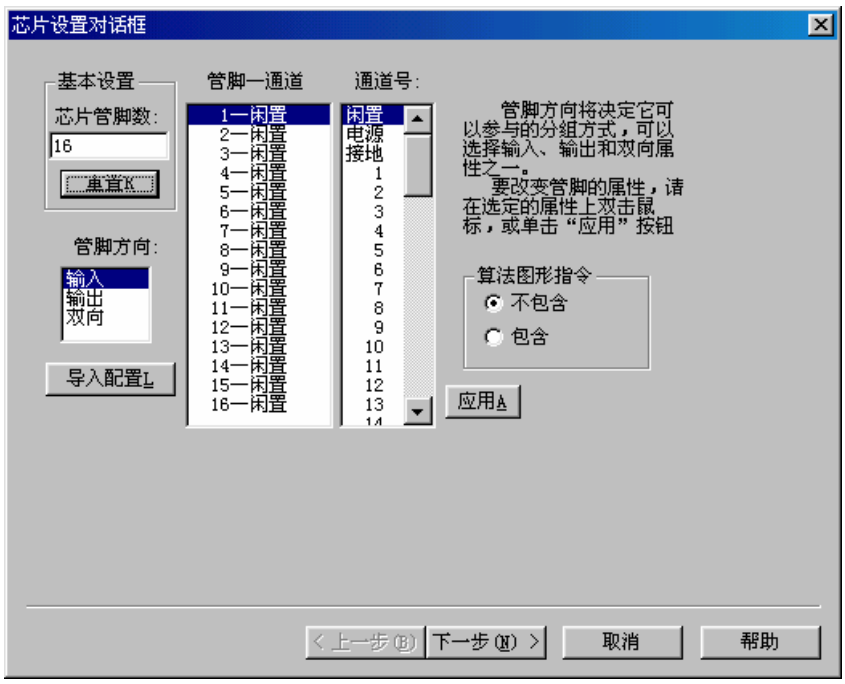


图 2-4 芯片设置对话框

向导的第一个对话框为芯片配置对话框，包括芯片管脚数字选择、芯片管脚输入输出方向确定以及通道到管脚的连接；以及该芯片是否包含算法图形。

芯片的管脚数在 1－1024 内可选，数字通道号在 1－64/128 可选，由系统安装的通道数决定。配置新的芯片首先要配置芯片的管脚数，然后单击“重置”按钮。例如要编辑 74ls00 器件，则输入 14，单击“重置”按钮，器件管脚列表被更新，所有的配置都被清除，如图 2-5 所示。一个芯片的图形文件建立以后，其管脚数不可以被修改，在向导完成之前，器件管脚数可以被修改。

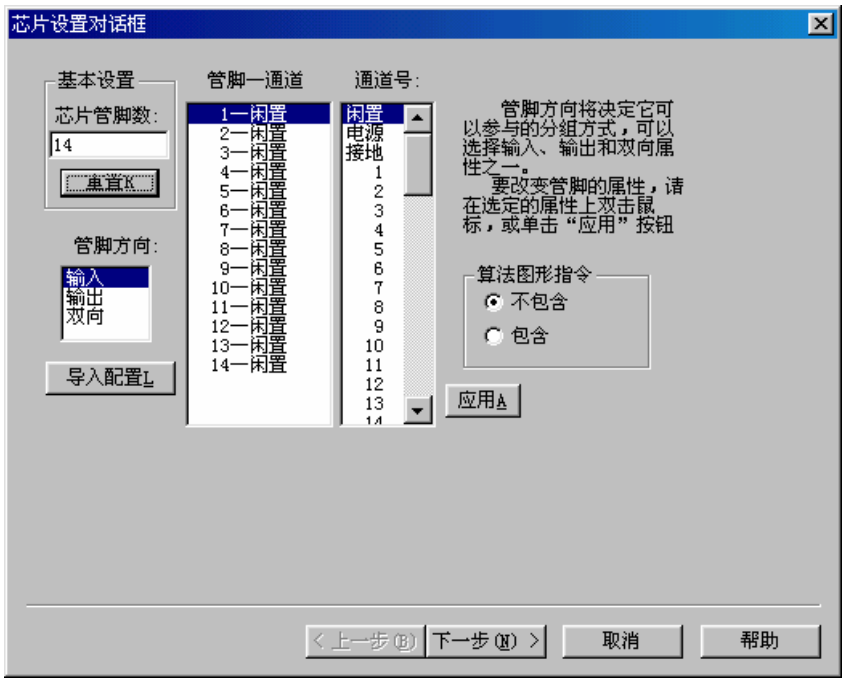


图 2-5 重置器件管脚

管脚方向即被测器件的管脚的输入、输出方向，只有配置了器件的管脚方向，系统才知道如何处理这些管脚。管脚方向配置方法是：首先选定要配置的管脚，在管脚方向列表中双击要设定管脚方向，完成管脚方向配置，如图 2-6 所示为设定一个管脚方向。

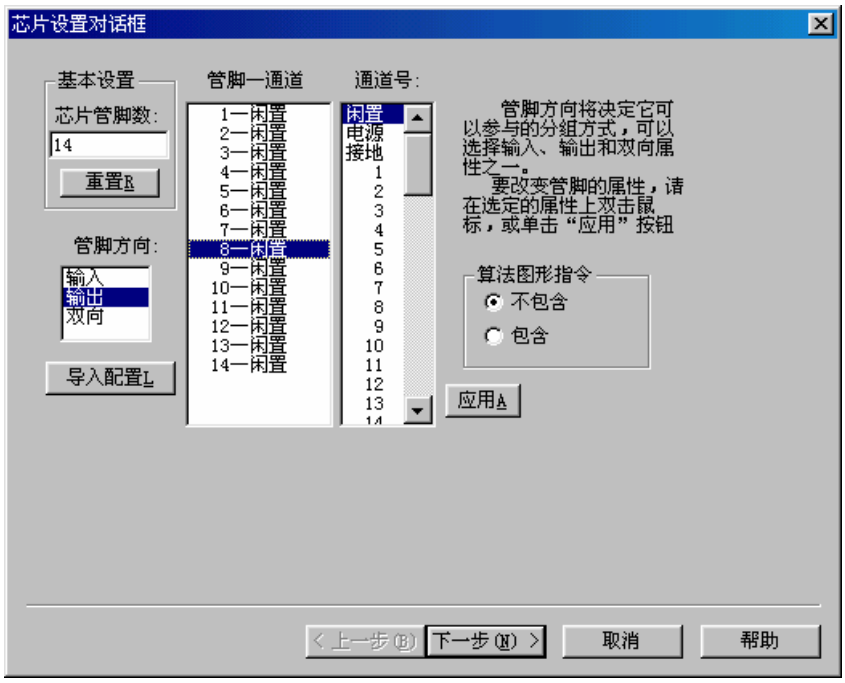


图 2-6 分配器件管脚方向

分配完管脚的方向后，就可以将管脚分配到通道上（按负载板上器件的与通道的连接分配）。管脚到通道的分配方法是：选择要分配的管脚，在通道号列表栏中选择要分配的通道，双击通道号（或单击“应用”按钮），完成通道分配到管脚上，同时该通道自动从通道列表中被删除；器件列表栏显示被分配的管脚，焦点自动移动到下一个管脚。如图 2-7 所示为分配 1 管脚到 10 号通道。

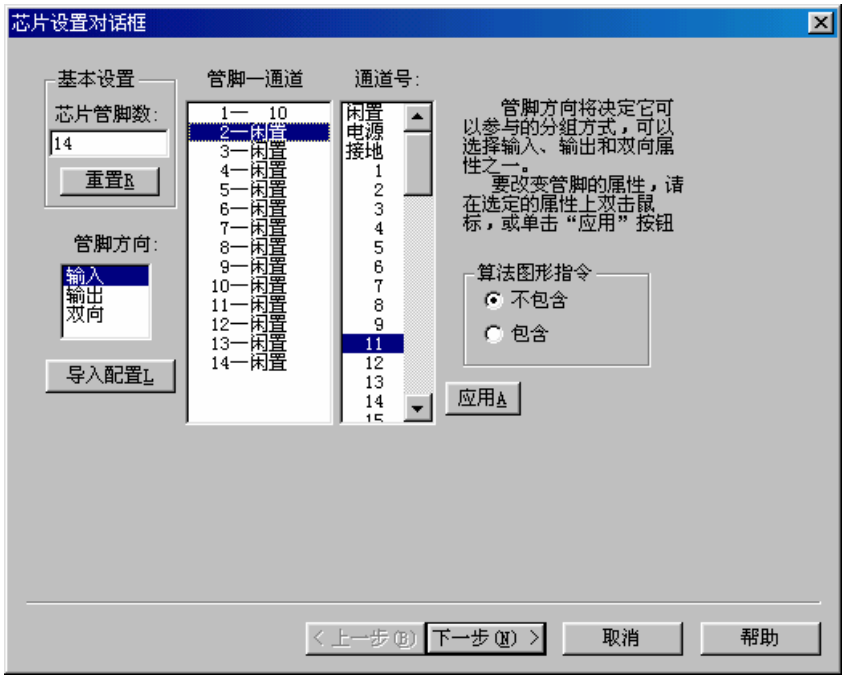


图 2-7 分配管脚到通道

以 74ls00 为例，第 3，6，8，11 号管脚分别对应 10，13，15，18 号通道，且管脚方向为输出，第 1、2、4、5、9、10、12、13 对应通道号为 10、11、13、14、18、19、21、22，管脚方向为输入，分配完毕，如图 2-8 所示。

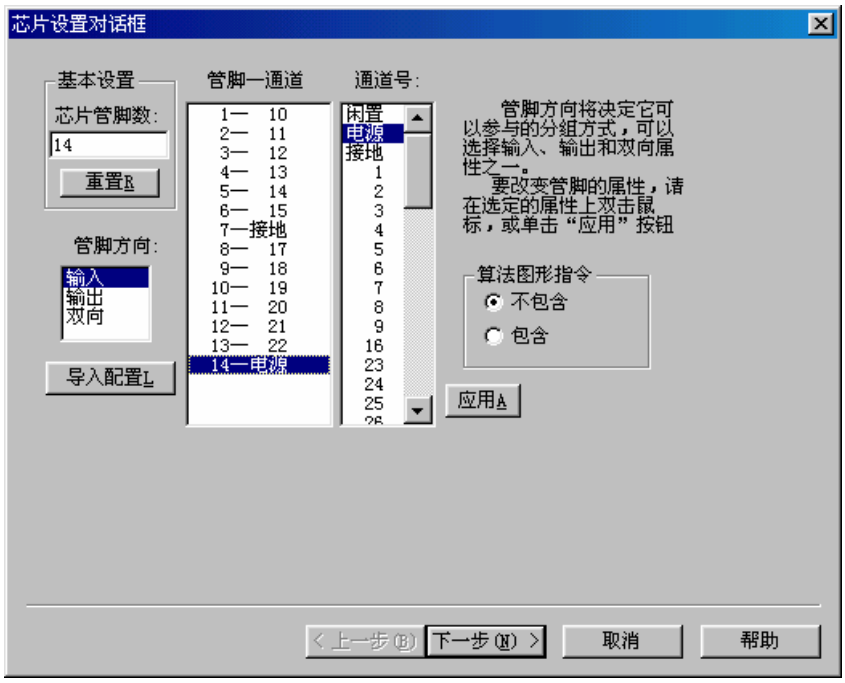


图 2-8 分配 74ls00 的器件管脚

对于不需要连接到通道的管脚可以选择闲置；对于电源或地引脚选择电源或接地。

闲置、电源和接地选项可以保证用户在操作图形管脚时，不会误将数字与电源连接。

对于以配置的管脚，在用户程序中，可以直接对管脚号操作，而不用记忆其具体的通道位置，对于没有连接到数字通道的管脚，必须按负载板上的实际位置操作。

三点说明：

（1）器件管脚的方向分配和通道分配，无先后顺序，用户可以随时修改。

（2）要释放一个以分配管脚的通道，首先选中要释放的管脚，然后在通道列表栏，双击“闲置”，即可释放，可以进行新的分配。

（3）对于已分配通道的管脚，需要重新分配通道，必须先释放通道。

如果该数字芯片不需要使用算法图形资源，选择算法图形栏中的“不包含”选项。该项目可以在参数修改的时候修改。

如果有配置文件，则“导入配置”即可直接进行参数的修改，或相似型号芯片的参数设置，可大大节省时间。

芯片配置完成后，单击“下一步”按钮，进入运行配置向导。单击“取消”退出向导。

小结：

（1）芯片配置对话框配置流程：重置芯片管脚数→设置管脚方向→分配器件管脚到通道→选择是否编辑算法图形→确认进入下一步。

（2）重设已分配的管脚：选中要调整的管脚→双击“闲置”释放管脚→双击新的通道。

（3）单击“重置”按钮，所有分配被取消。

（4）单击“导入配置”按钮，快速完成配置或修改已有配置。

2. 运行设置对话框

单击“下一步”。进行运行配置，如图 2-9 所示。

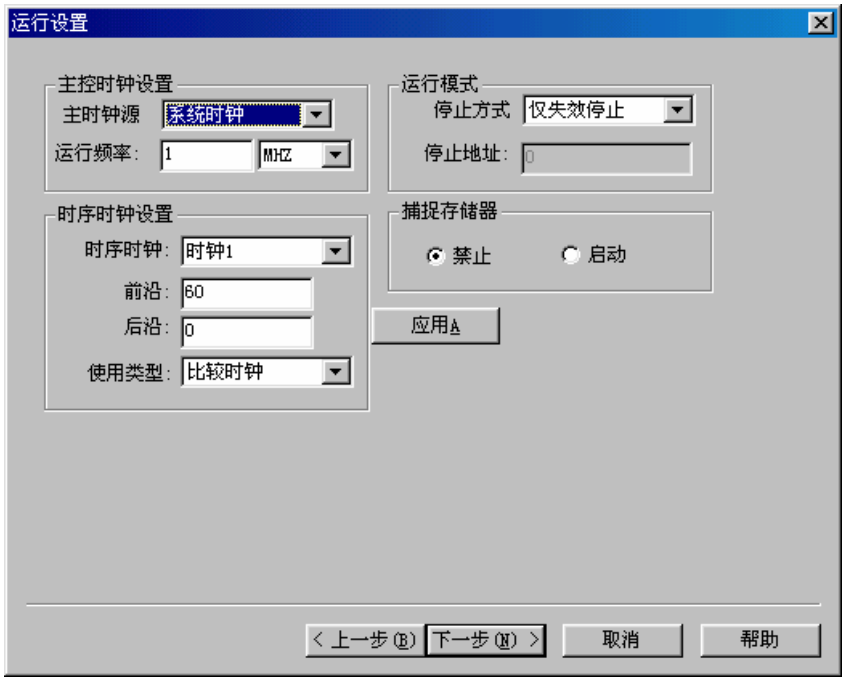
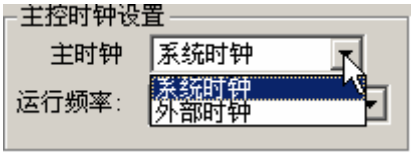


图 2-9 运行设置

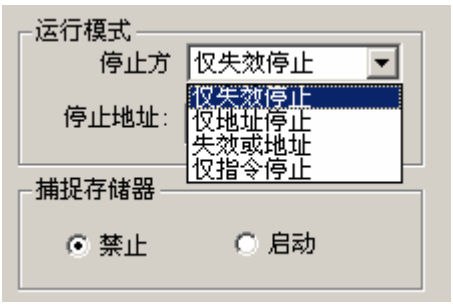
运行设置 4 部分内容：主时钟设置、时序时钟设置、停止方式设置和捕捉存储器设置。

（1）主时钟分为系统时钟和外部时钟。

系统范围：33MHz～1.6KHz，速度分辨率 10ns；时钟精度 1ns。选择外部时钟时，时钟由用户输入，由负载板引入系统。当使用外部时钟时，时序时钟无效，默认使用时钟 1 作为比较时钟，不支持格式化。如图：

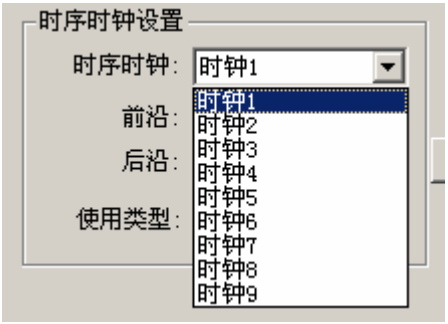


（2）停止方式是指图形运行的停止方式：包括仅失效停止、仅地址停止、地址或失效停止和仅指令停止。



（3）时序时钟设置

系统目前共有 9 路可编程时序时钟供选择，根据需要需分别进行设置，均可设置为比较或格式化时钟。系统一般默认时钟 1 为比较时钟，时钟 9 为格式化时钟。系统至少有一路比较时钟。



用户可以逐一设置每一路时钟的时间参数，如果后沿设置为 0，则默认时钟的占空比为 50。

不使用的时钟，可以不配置。

（4）捕捉存储器：默认为不起动。如果需要启动则设置。

小结

（1）所有对测试程序的参数设置，均可以在测试程序中更改，因而用户如果需要更改设置值，这里可以不配置。所有测试程序中修改的参数将不能被恢复，这点需要注意。参数的配置仅仅是预设。

（2）系统实际共有 16 路时序时钟，时钟 10 到 14 为扩充时钟，目前为恒 0；时钟 15 为空时钟，接地；时钟 16 为外同步时钟，可以通过负载板引入系统。在时序配置对话框用户可以选择全部的时序时钟。

3. 算法图形配置对话框

单击“下一步”，进入算法图形配置。若在芯片配置时选择“不包含算法图形”，则该对话框不可以设置，如图 2-10 所示。

图 2-10 算法图形配置对话框

算法图形对话框可以分配器件管脚到算法图形资源；可以设置算法地址的使用模式和编辑模式。

本芯片不包含算法图形，因而不需要设置。

4. 管脚分组对话框

单击下一步，进行管脚分组，如图 2-11 所示。通过管脚分组，可以将相同和相似管脚分成组统一管理，同时也方便编辑图形。

因芯片管脚包括三类：输入、输出和双向管脚组，因此分组时，系统已将三类管脚进行分类，选择组方向，在管脚号列表框中自动列出相同方向的管脚。

不同 IO 方向的管脚，不可以分配到一组。

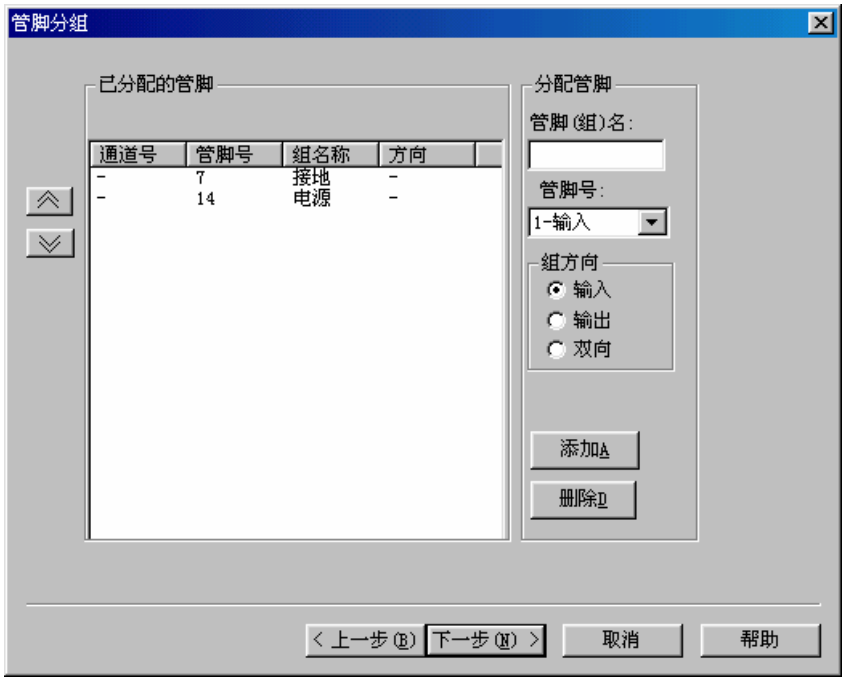


图 2-11 管脚分组对话框

管脚分组对话框，自动将电源、地的管脚以及算法图形分配的管脚按组列出，闲置的管脚不显示。

先选择“输入”方向，然后在“管脚（组）名”处输入管脚组名，如 ab1，单击“添加”按钮，管脚就被分配到组中，如图 2-12 所示。组名不区分大小写。

图 2-12 分配管脚分组

管脚组中管脚的排列顺序，由列表框中显示的位置决定，依次是从最低有效位到最高有效位。即管脚 2 为 AB1 组的高位。同一管脚组中管脚的位置，可通过向上或向下箭头改变管脚的在组中的顺序。

如果要添加新组，在管脚组名中填入新的名称，即可。如图 2-13 所示为分配全部输入管脚组。

图 2-13 分配输入管脚组

按同样的方法分配输出管脚组，如图 2-14 所示为，为输出分配 4 个管脚组。

图 2-14 管脚分配完毕

管脚组列表中显示的组的顺序，就是图形编辑窗口中显示图形组的顺序。如果需要调整顺序，可以使用删除按钮，和添加按钮重新分配组。

双向组在图形编辑窗口中将以两个组显示，一个输出组，一个输入组，组名后由系统自动添加“I”或“O”尾缀进行区分。例如 DA 为双向组，则图形编辑界面将显示为 DAI 和 DAO 两组。只有当 DAI 中的数据为 Z，即高阻，才可以编辑 DAO 中的图形值，否则只能为 X。DAI 中的图形编辑不受限制。

所有管脚必须分组，图形编辑程序对所有管脚都按组管理。不分组的管脚不能编写对应的图形，也不能设置参数。

5. 时序设置对话框

管脚分组完毕后，单击“下一步”。开始时序设置，如图 2-15 所示。

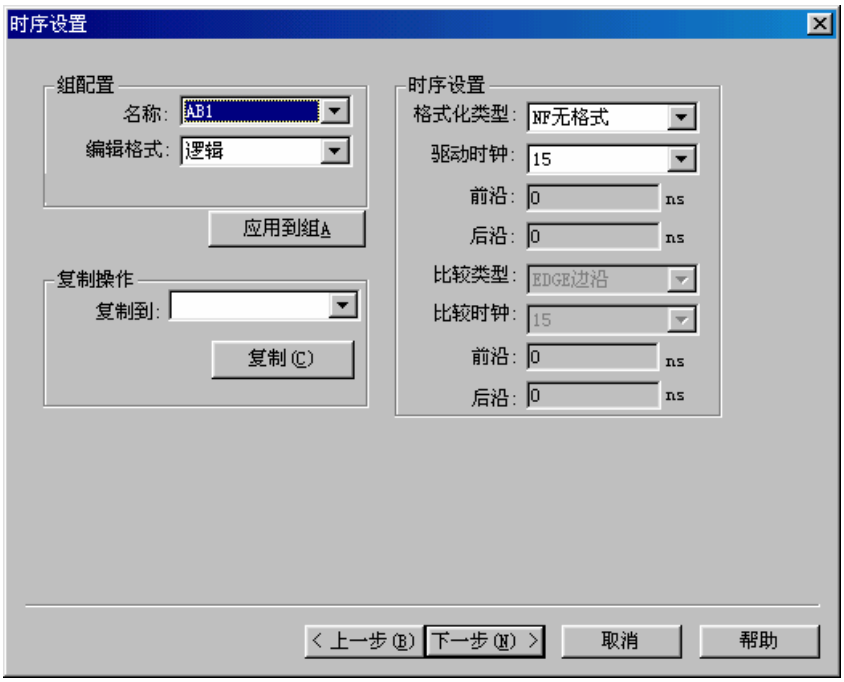


图 2-15 时序设置

组配置栏中包含全部已分配的管脚组，用户可以设置组的显示模式和时序。

管脚组在图形编辑界面有 5 种显示格式：

- 逻辑：即按位显示每管脚的图形，输入管脚为 1、0 和 Z；输出管脚为 L、H 和 X。
- 二进制：以二进制数显示图形，合法的显示值为 1、0、Z 和 X
- 十进制：以十进制数显示图形
- 八进制：以八进制数显示图形
- 十六禁止：以十六进制显示图形

所有的管脚组必须进行时序配置。对于输入管脚有：格式化方式和格式化时钟。系统默认格式化方式为 NF 不格式化，格式化时钟为 15。对于输出管脚默认比较方式为边沿比较，比较时钟为 15；双向管脚必须设置分别设置比较和驱动模式。

单击组配置名称对话框的下拉按钮，可以看到全部可供配置的组，如图所示。

此处最为独特的是设计了相同参数的复制操作。

以 74ls00 为例，因为 C1、C2、C3 和 C4 四个组的属性完全相同，因此只需设置好 C1 组的时序后，选择“复制到－全部输出组”即可完成其余 3 个组的时序设置。不再需要逐个设置。

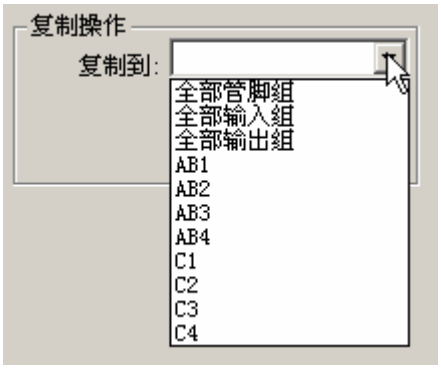


图 2-16 为配置 C1 的时序，选择比较时钟为 1。单击应用按钮，应用配置。

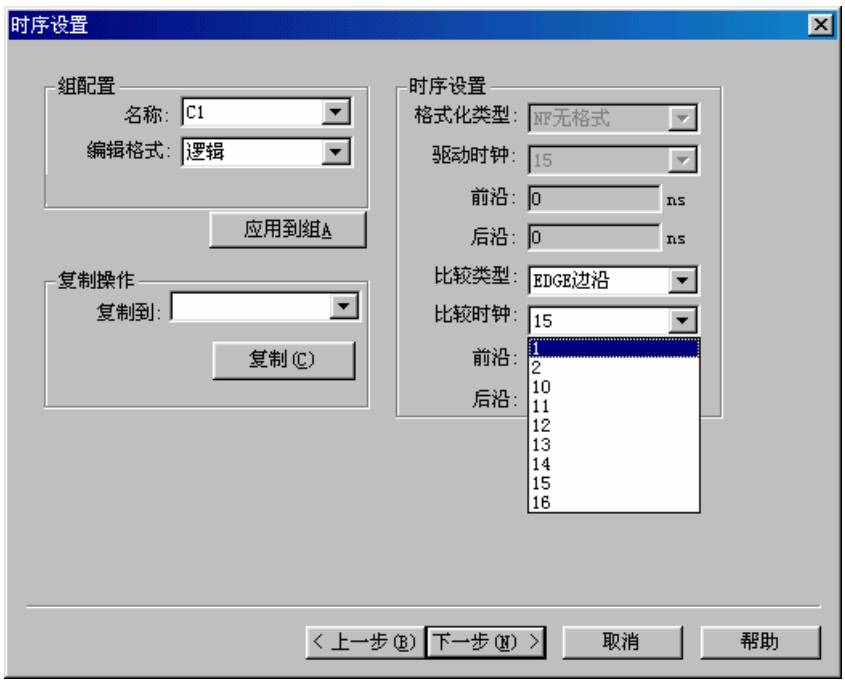


图 2-16 C1 组时序配置

应用后，打开“复制到:”下拉列表框，选择全部输出组，如图 2-17 所示。

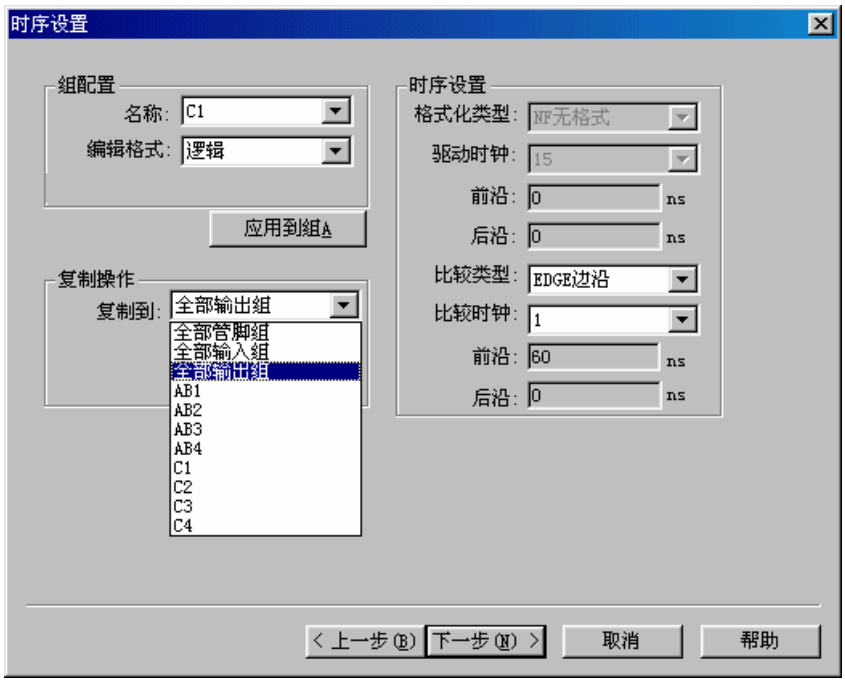


图 2-17 选择复制目标

单击“复制”按钮，完成全部输出组的时序和显示格式的配置，图 2-18 显示了，C2 组的时序配置，与 C1 完全相同。

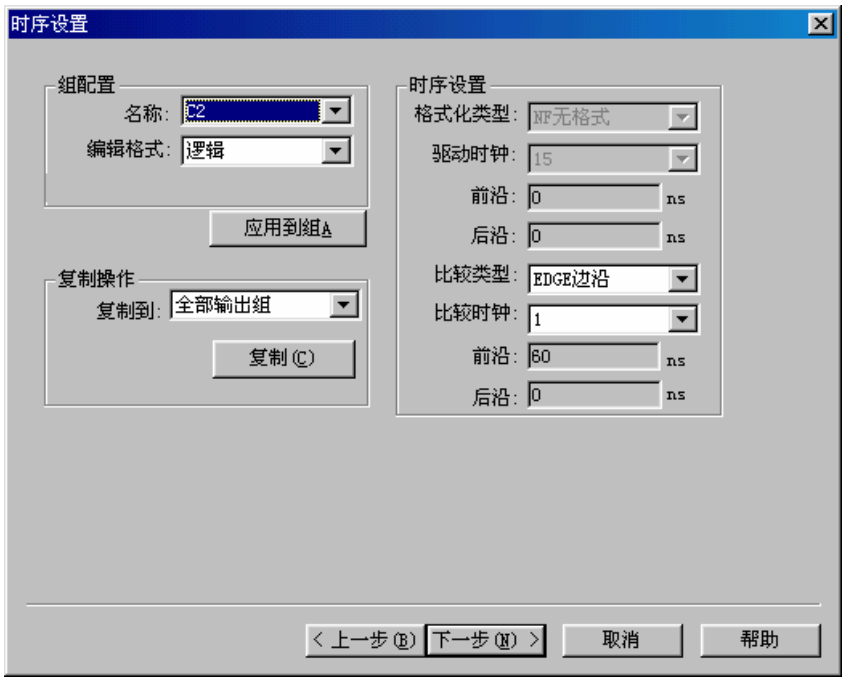


图 2-18 查看 C2 组的时序配置

6. 参考电平设置对话框

时序设置完毕，单击“下一步”进行管脚组参考电平设置，如图 2-19 所示。

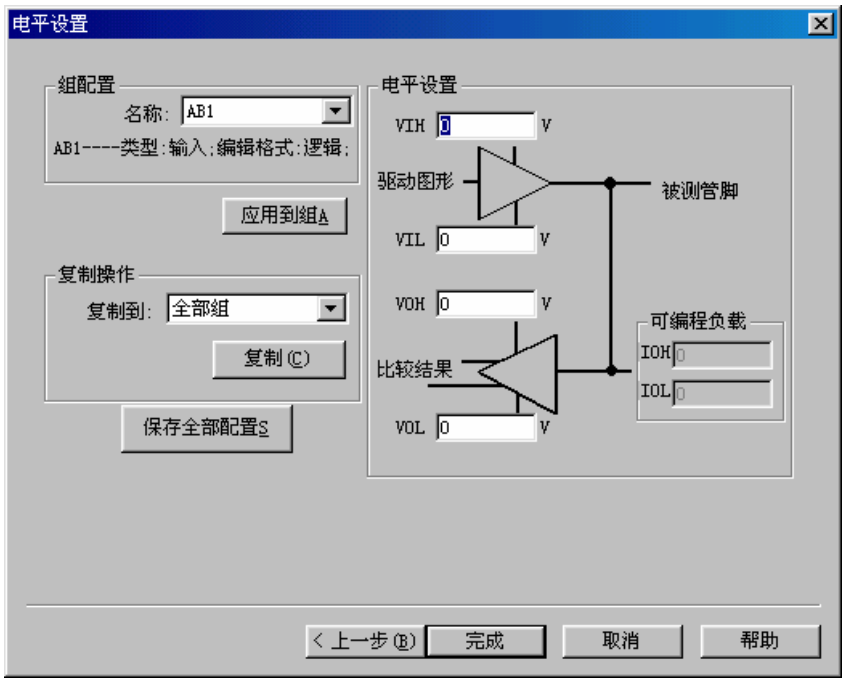


图 2-19 管脚组的参考电平设置

系统为每个管脚提供独自的参考电平，因此可以为任何的管脚或组配置电平。对于输入管脚组，VOH 和 VOL 将被忽略；同样对于输出管脚组 VIH 和 VIL 将被忽略。驱动电平的范围是－9～＋9V；比较电平为－7～＋7V。

组设置完毕后，要单击“应用到组”按钮，确认修改。通过复制操作可以快速完成相同属性组的电平配置。

以 74ls00 为例，设置完毕如图 2-20 所示：

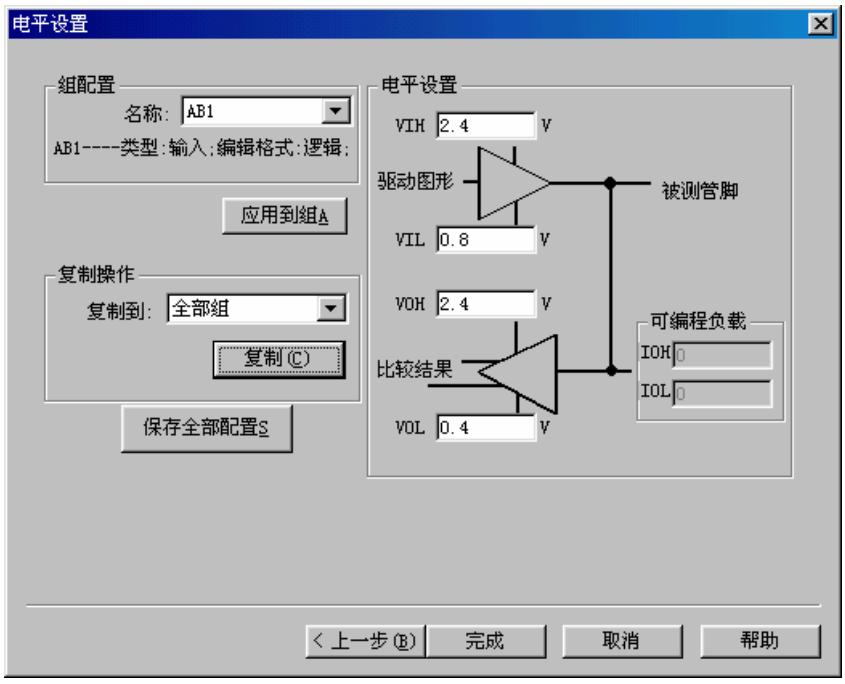


图 2-20 电平配置对话框

配置完毕，单击“保存全部配置”，即可将之前的全部设置保存下来，遇到相似类型芯片直接导入配置即可。（如 74ls00,74ls04,74ls08，74ls32 等等）

单击“完成”进入主控界面，完成向导，如图 2-21 所示。

图 2-21 主控界面

执行菜单“文件”→“保存配置”配置向导中的“保存全部配置”一样，可把配置保存为“.chp 文件”，方便以后使用。

单击“参数设置”按钮，打开参数设置对话框，如 2-22 所示，由 6 个标签页组成，包括全部向导中的 6 个对话框。可以设置各个组的时序、电平等参数以及进行运行设置。管脚分组对话框和算法图形对话框可以查看，也可以修改器件的管脚分组信息。新建的分组必须重新设置其时序和电平。

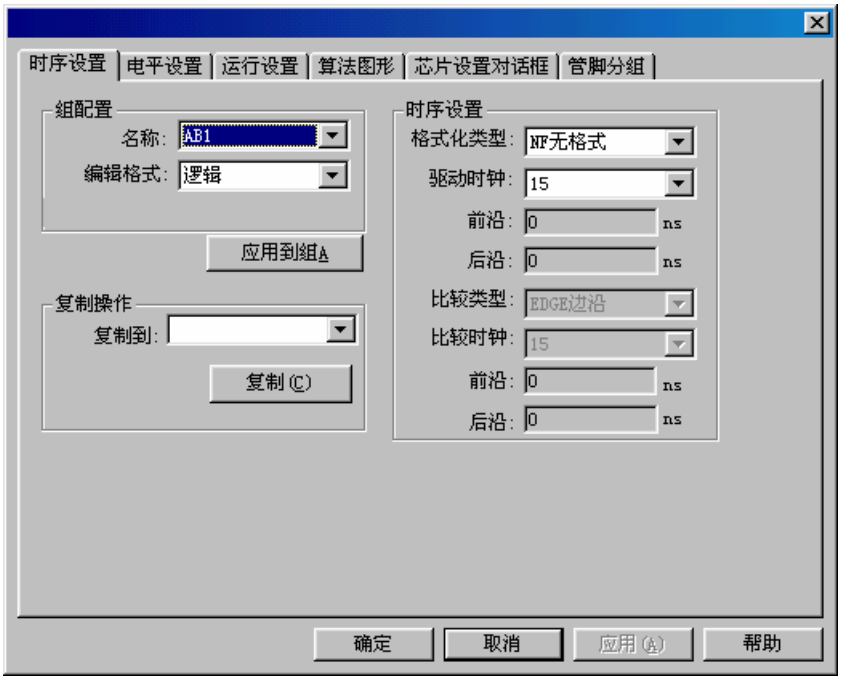


图 2-22 参数设置选项对话框

单击“芯片设置”标签，打开芯片设置标签，如图 2-23 所示。除了芯片管脚数不可以配

置之外，所有的选项都可以配置。

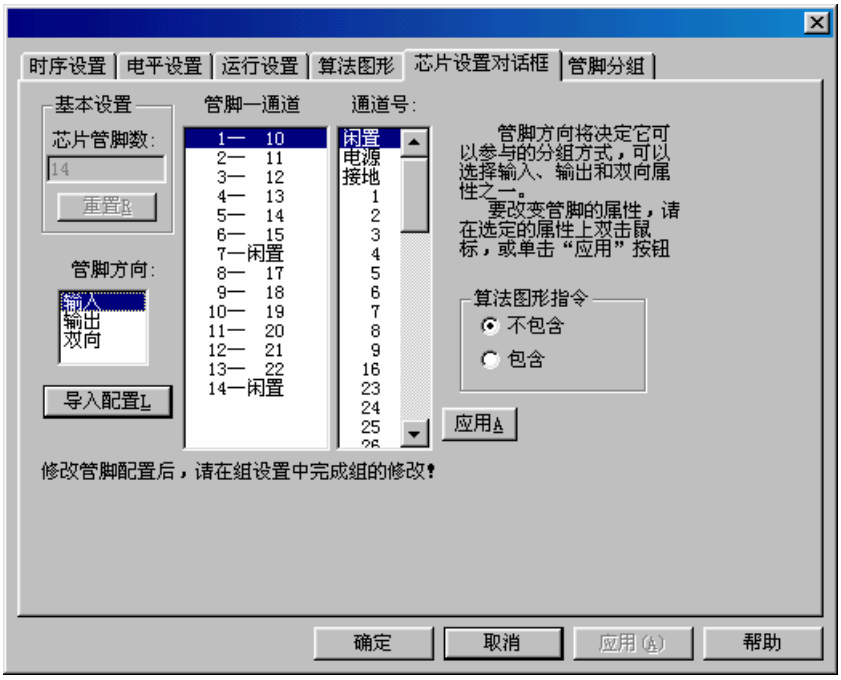


图 2-23

至此完成了新建测试图形的工作，用户只需要单击“编辑图形”按钮，就可以进行图形编辑工作了。

新建器件测试图形时，如果在芯片配置对话框中，选择“导入配置”，则一步完成上述工作，直接可以进行图形编辑。

四.测试图形编辑

在主控界面，单击“编辑图形”按钮或执行菜单“操作”→“图形编辑”命令，就可以打开图形编辑界面，进行图形编辑，如图 2-24 所示。在此界面完成芯片的图形文件的编辑（以上述创建的 74ls00 为例）。

图形编辑器采用可见即所得的文本控制技术，在线纠正用户的语法错误，用户只要集中精力完成时序或逻辑真值的编写。

图形编辑器包括五部份：行号显示窗口，指令编辑窗口、算法图形编辑窗口、普通图形编辑窗口和注释窗口。每个窗口由两部分组成：标题子窗口和内容子窗口组成。各个窗口的功能如下：

- 行号窗口：用于显示测试图形的行号，即图形地址，自动由系统控制生成。用鼠标单击行号窗口，可以完成图形的行选择和行操作。行操作包括：多行/单行图形的选择、剪切、复制、粘贴、插入和删除操作。用鼠标双击行号可以将一行图形注释/还原。
- 指令窗口：用于编写图形的控制命令，如 HALT、INC 等，本系统支持顺序、循环、条件跳转和无条件跳转、匹配等多种指令，详见系统编程使用手册。
- 算法图形窗口：如果开发的图形包含算法图形，则在指令和图形窗口之间将显示算法图形窗口，算法图形窗口用于完成算法指令的编辑。
- 普通图形窗口：用于完成普通图形编辑。用鼠标单击图形窗口的标题子窗口可以完成对一列图形的各种操作。也可以在图形内容编辑窗口中对一个管脚组的部分图形进行指定的编辑操作。
- 注释窗口：用户可以为每行图形添加注释，注释的长度要小于 255 个字符。

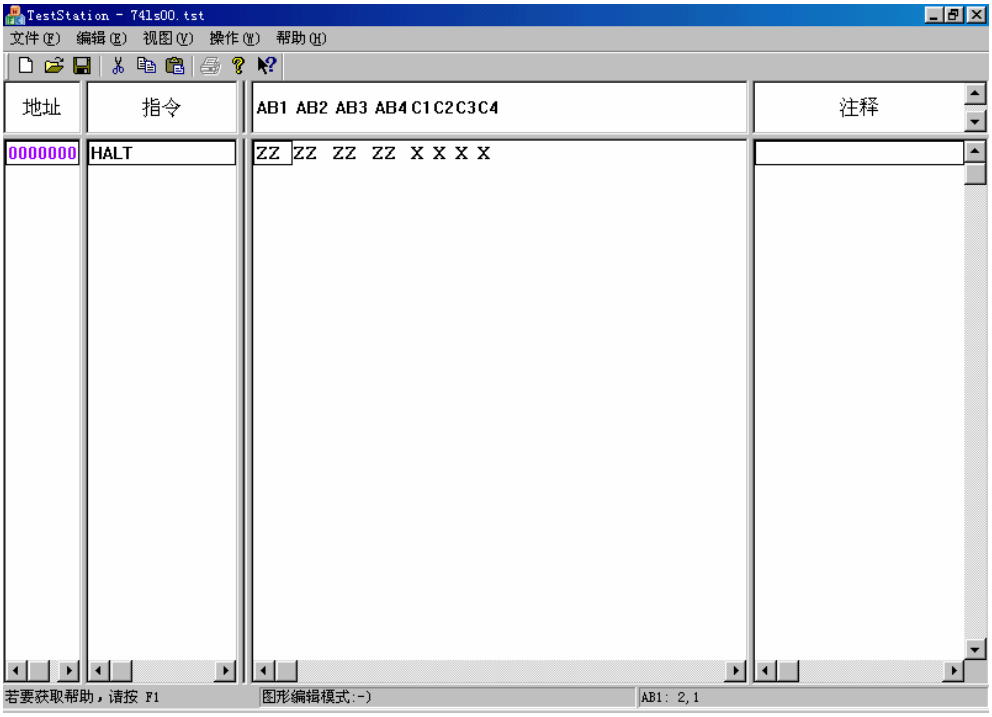


图 2-24 图形编辑界面

以下详细说明如何使用图形编辑器创建测试图形。

1. 使用插入键（Insert）插入图形模板，创建图形

使用一次插入（Insert）键，或在行号窗口右击鼠标，弹出菜单上选择“插入”命令，如图 6-25 所示，可以迅速插入一行图形模板，然后由用户修改为所需的图形。图 2-26 为插入 5 行图形模板。

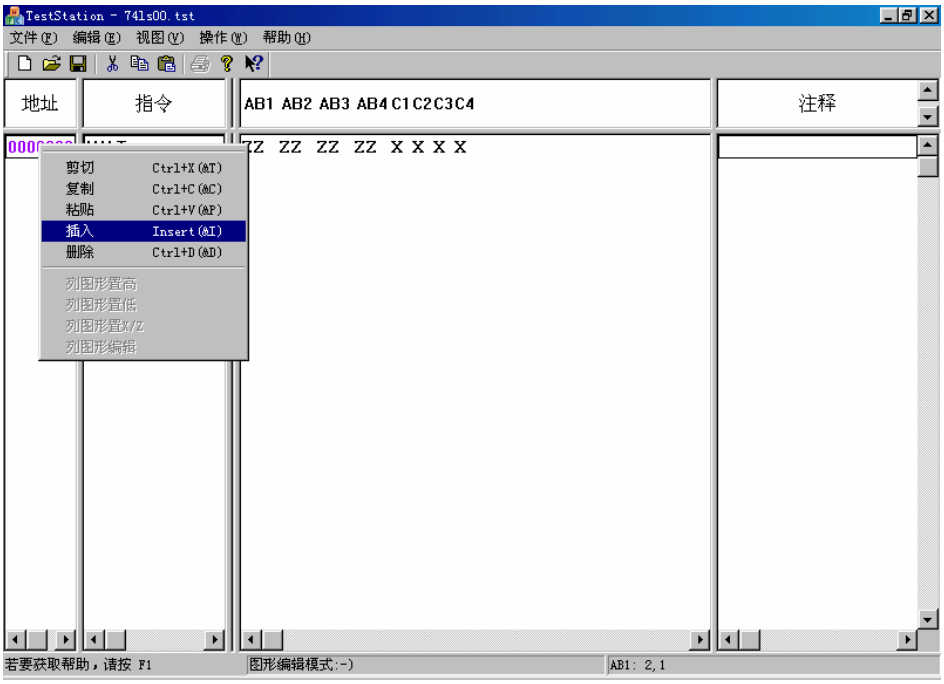


图 2-25 插入图形模板

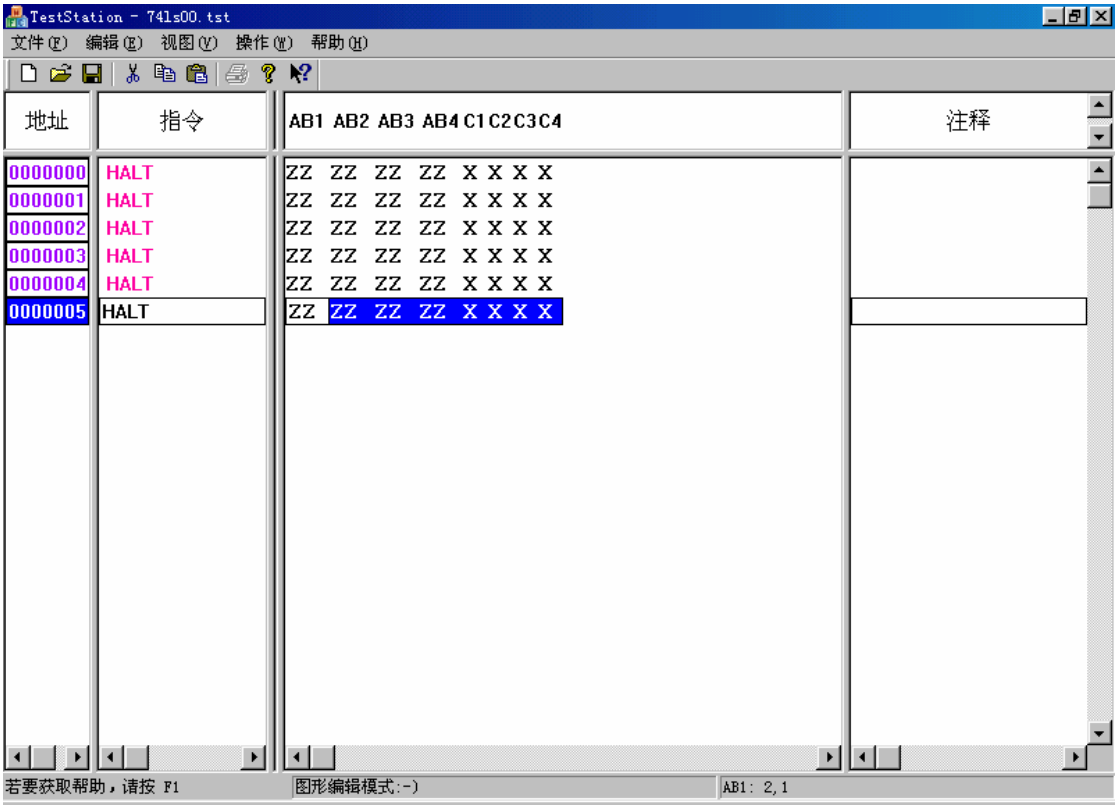


图 2-26

单击行号，即可选中一行图形。在图形编辑其中每个指令或每个图形组的图形都是一个独立的单位，用鼠标左键单击即可进入编辑模式，可以进行编辑。选中第一行指令，输入 INC，如图 2-27 所示。然后在新的位置单击鼠标左键，如果用户输入正确，则命令被输入，否则自动恢复为默认值。

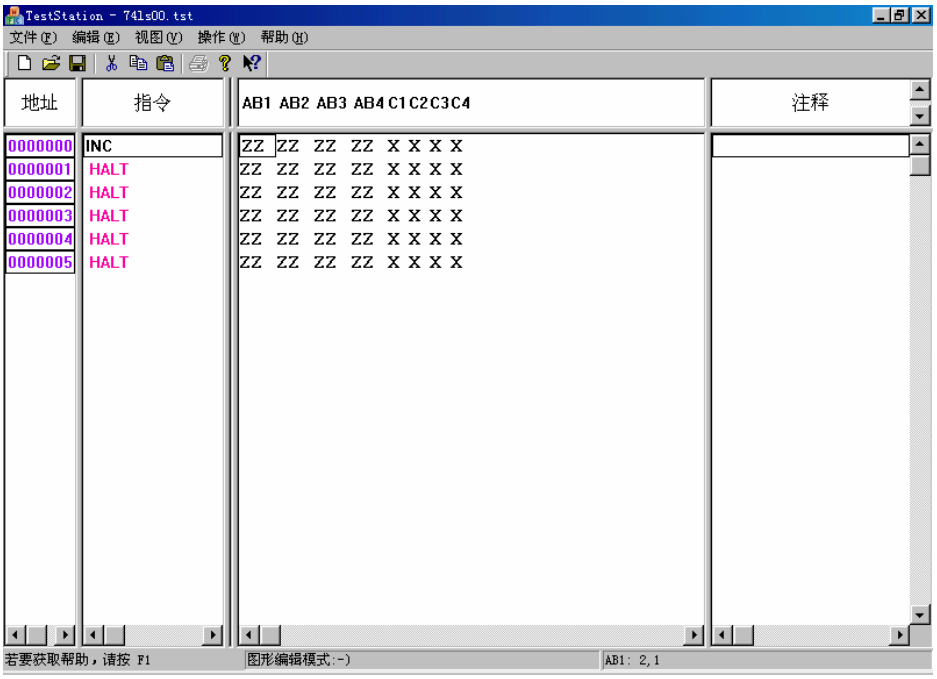


图 2-27

图 2-28 所示为编辑一行图形。对于用户输入的图形，系统会自动转换为标准格式，例

如以逻辑形式显示图形，如果用户为输入管脚写入图形 X，则系统会转换为 Z，输出管脚输入图形 0，则系统会转换为 L。如果用户输入的格式错误，新值不能被输入。

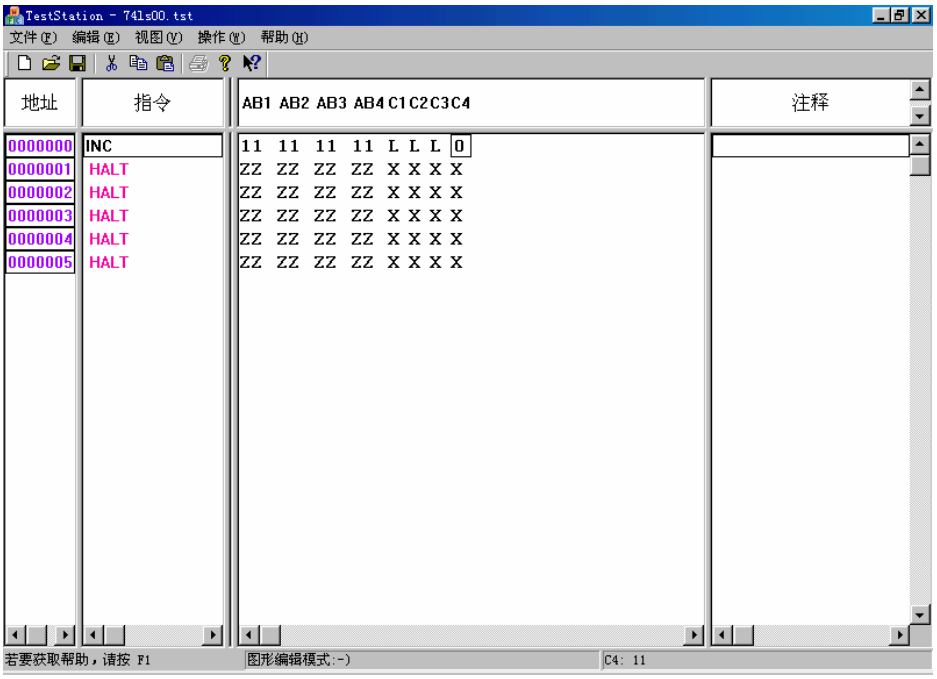


图 2-28

2. 使用行操作命令复制图形

多行图形选择的操作如下：

- 选择一行图形后，再按住 Ctrl 键，再单击要选择的行号，既可以增加该行到选择的序列中。
- 如果希望连续选择多行，首先选择起始行，单击鼠标左键在指定的行号处，则按住 Shift 键，单击结束行，即可连续选择多行。
- 在图形编辑器中其他位置单击鼠标，并且不按下 Ctrl 键或 Shift 键，即可以取消选择。
- 选择多行图形后，单击鼠标右键，既可以弹出操作菜单，可以对选择的多行图形进行操作。

单击鼠标左键，选中 0 行图形，再按住 Ctrl 键，选择第二行图形，如图 2-29 所示。

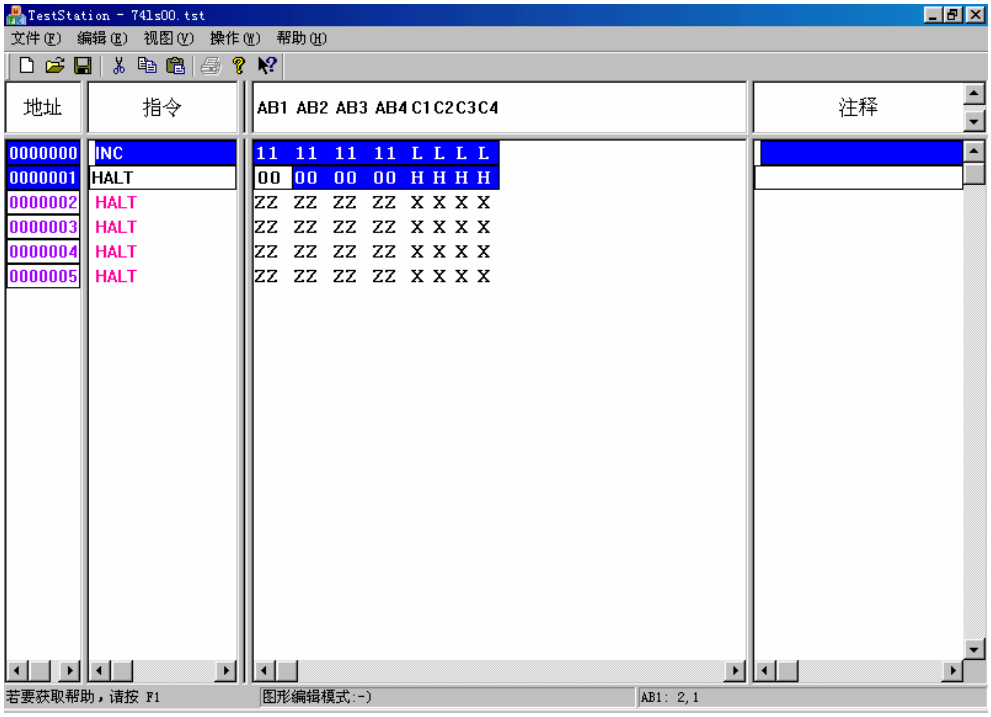


图 2-29 选择两行图形

右击鼠标，弹出菜单，如图 2-30 所示，选择“复制”命令。

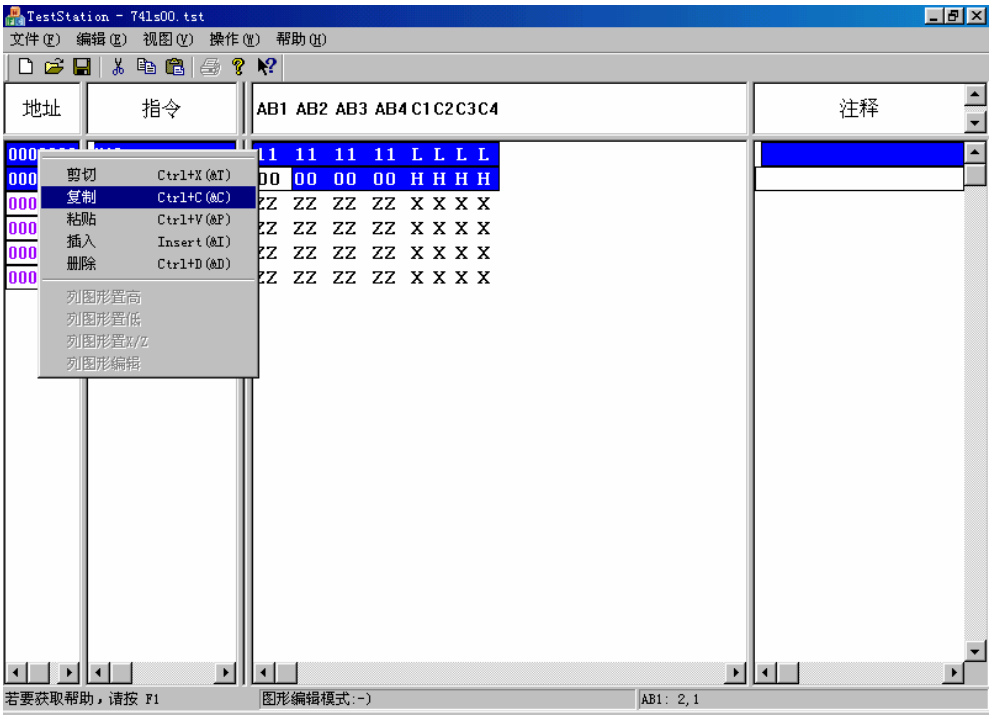


图 2-30 单击鼠标右键弹出操作命令

选择复制命令，然后在选中要插入图形的行，如第 4 行，单击鼠标右键，选择粘贴命令，如 2-31 所示。

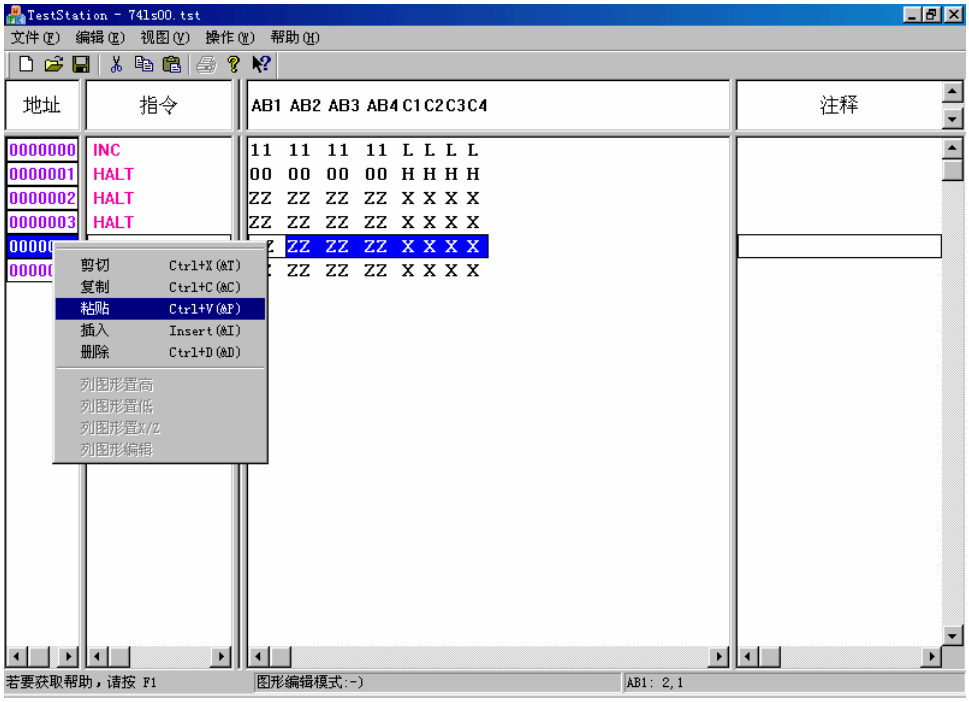


图 2-31 执行“粘贴”图形命令。

执行结果，如图 2-32 所示。

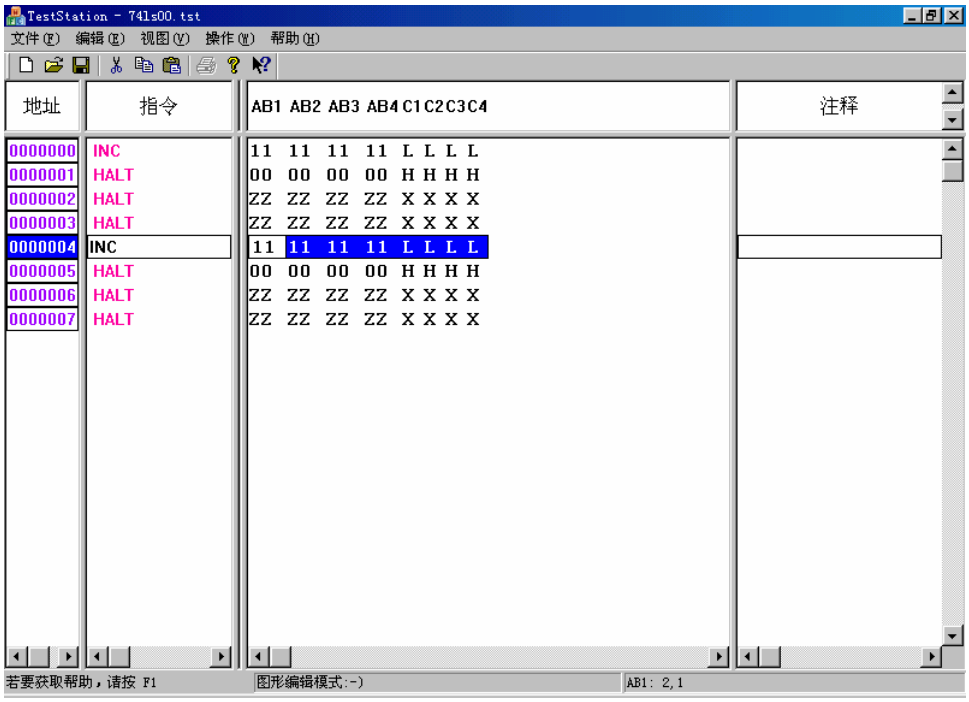


图 2-32 完成粘贴图形

3. 使用列图形命令可以完成对一组图形的操作

列图形操作可以选择一列图形，也可以选择该列中多个图形进行操作，方法是在图形窗口的标题子窗口中单击鼠标左键选择一列图形或在内容子窗口中使用 Ctrl 键或 Shift 键选择多个图形。

在组名 AB1 上，单击鼠标左键，选中改组的全部图形，如图 2-33 所示。

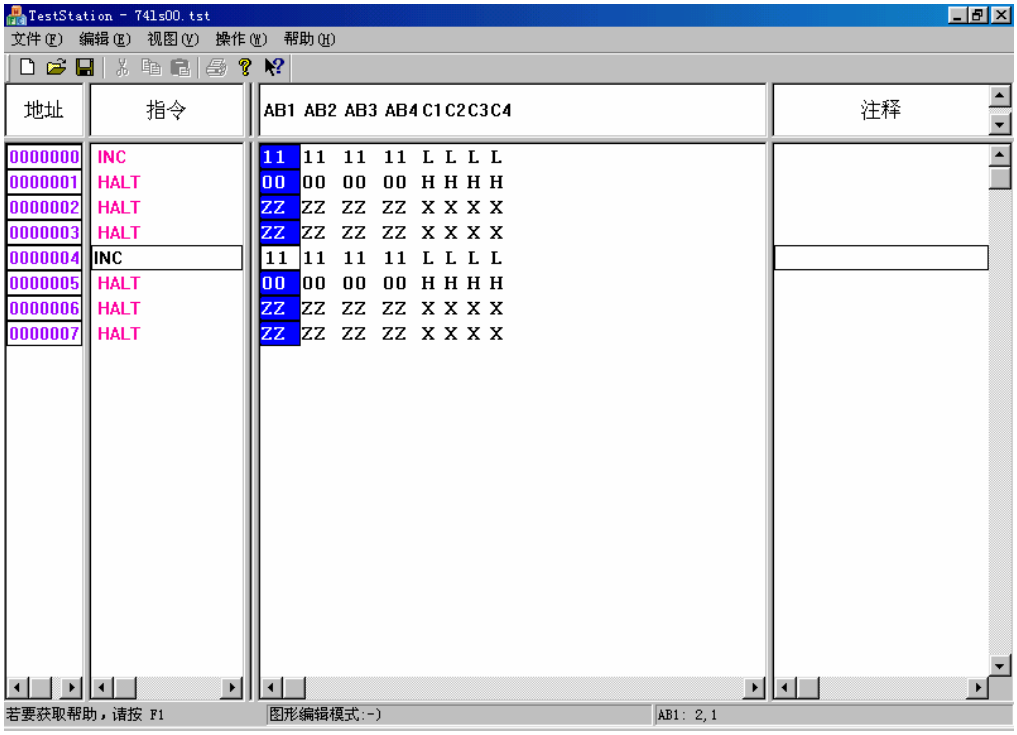


图 2-33 选择 AB1 组的全部图形

右击鼠标，弹出菜单，选择图形置高命令，操作结果如图 2-34 所示，管脚组 AB1 的全部图形都被置为 1。

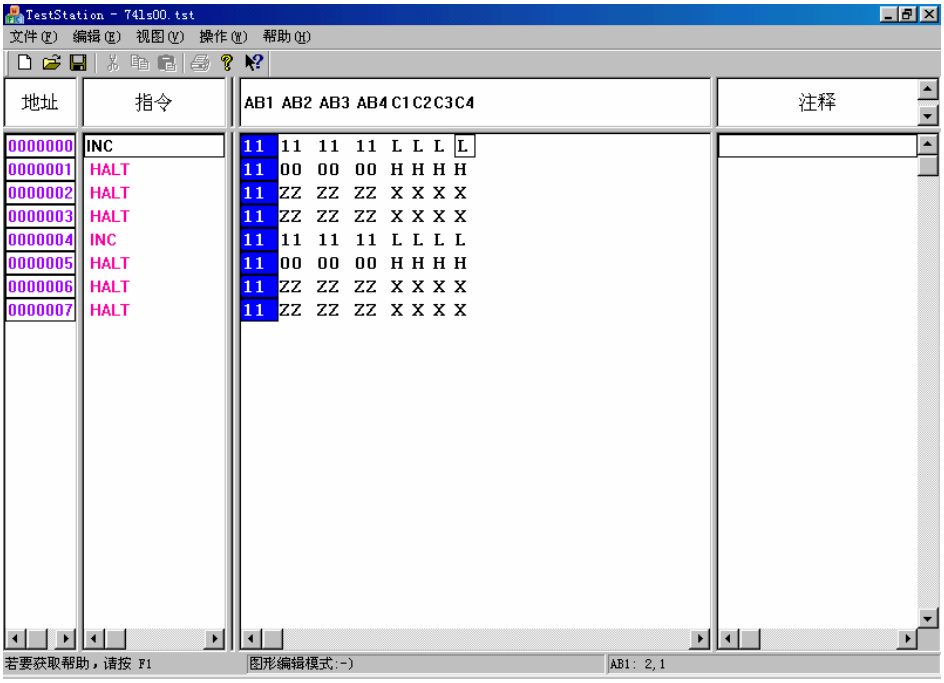


图 2-34 执行“列图形置高”命令

选择 AB2 组图形，右击鼠标，在弹出菜单上选择“列图形编辑”命令，如图 2-35 所示。

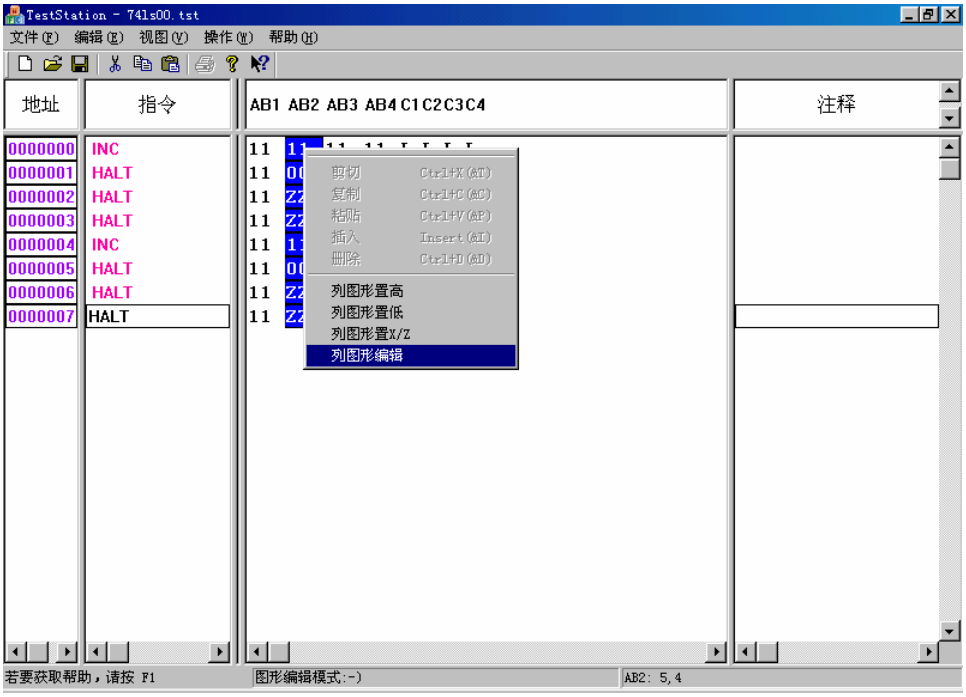


图 2-35 列图形编辑命令

执行命令后，弹出如图 2-36 所示的对话框，由用户选择图形编辑选项。

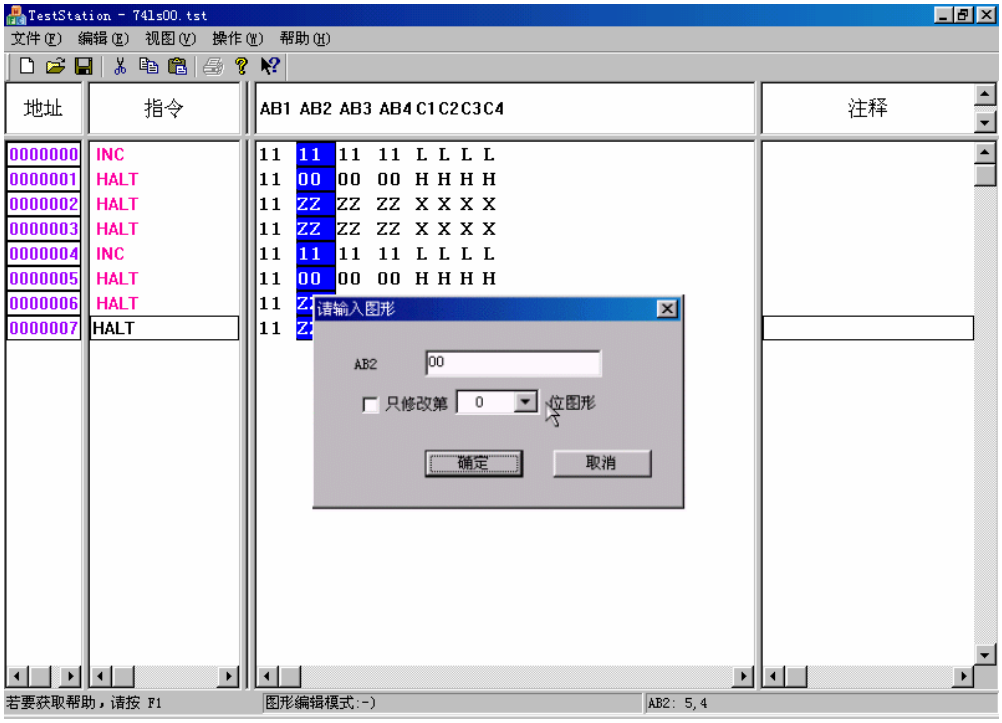


图 2-36 列图形编辑选项对话框

对话框中显示了被选中组的名子，和处于被编辑状态的图形。用户可以在图形框中输入要修改的图形，将该组全部图形设置为指定值。也可以选中复选框，修改指定列的图形。列表框中列出了该管脚组可选的全部位。

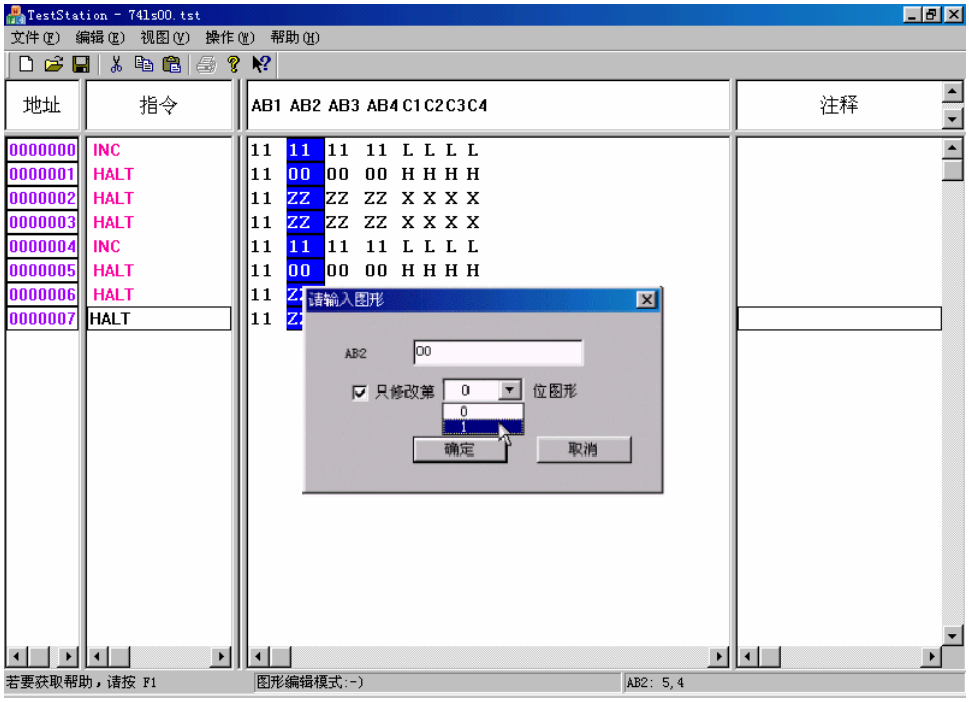


图 2-37

选中复选框，在下拉列表中选择第 1 位，如图 2-37 所示。单击确定按钮，完成列图形的编辑，如图 2-38 所示。

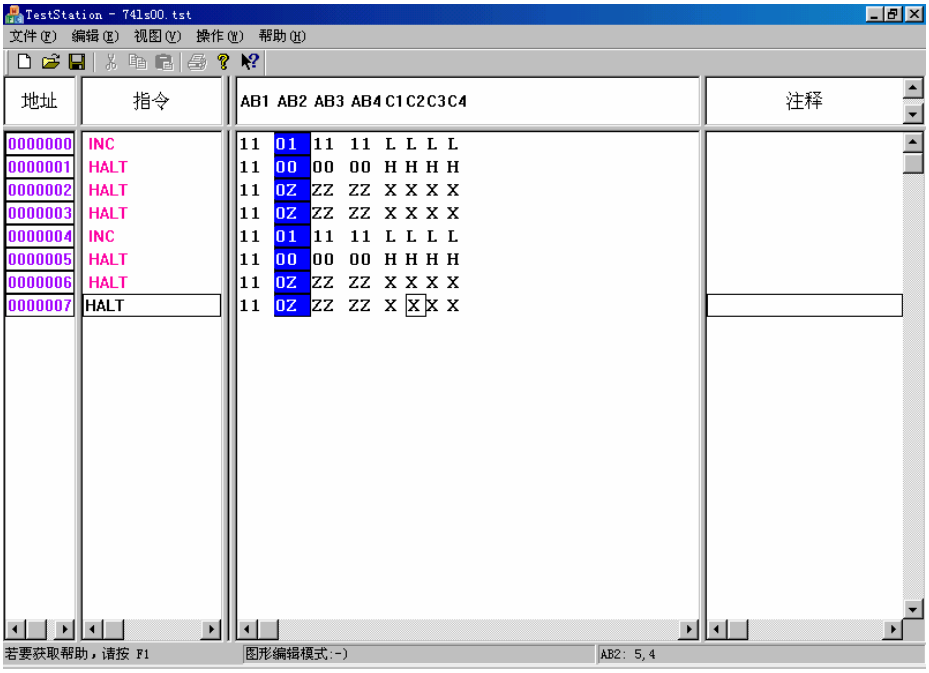


图 2-38

也可以使用 Ctrl 键或 Shift 键选择某一组图形的任意多行图形，然后进行编辑，如图 6-39 为选择组 AB4 的部分图形。

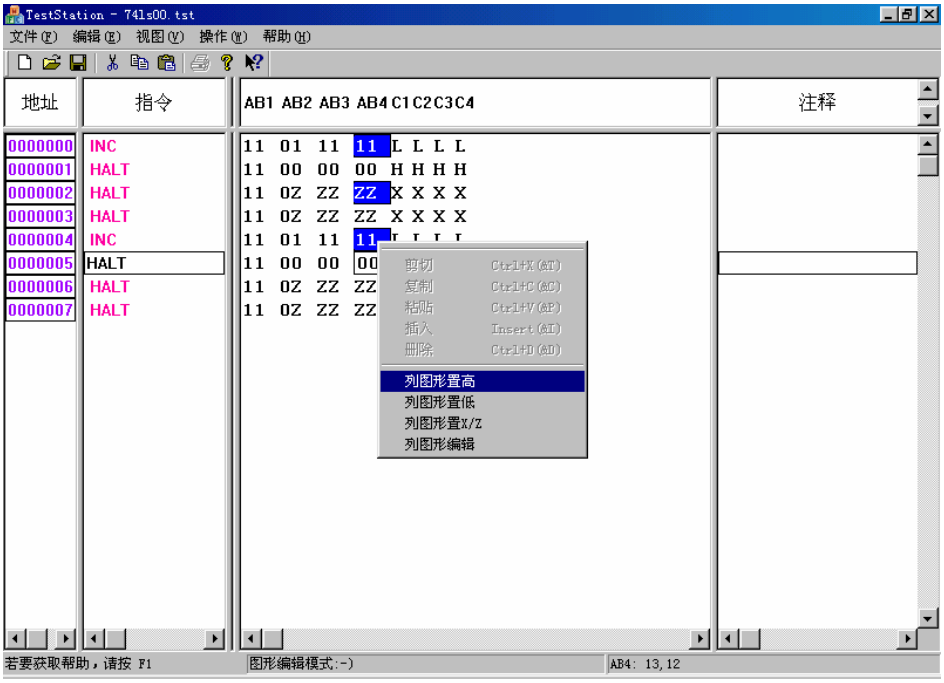


图 2-39

4. 使用参数设置，更改图形显示

执行菜单“操作”→主控界面命令，切换到主控界面。单击“参数设置”按钮，打开参数设置对话框，选择 AB1 组的编辑格式为十六进制，如图 2-40 所示。

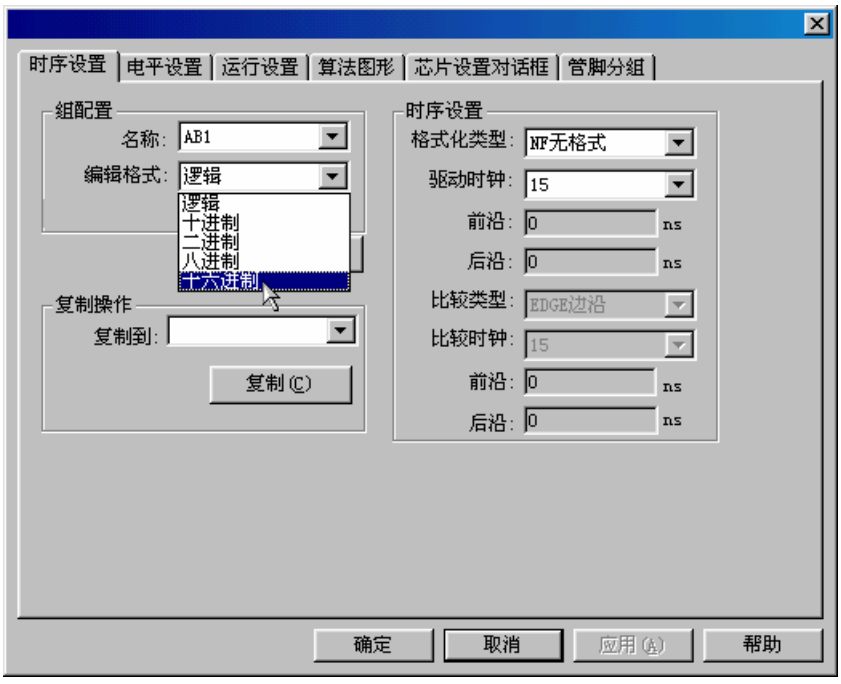


图 2-40

单击确定，回读主控界面，单击编辑图形按钮，打开图形界面，显示如图 2-41 所示。AB1 管脚以 16 进制格式显示。

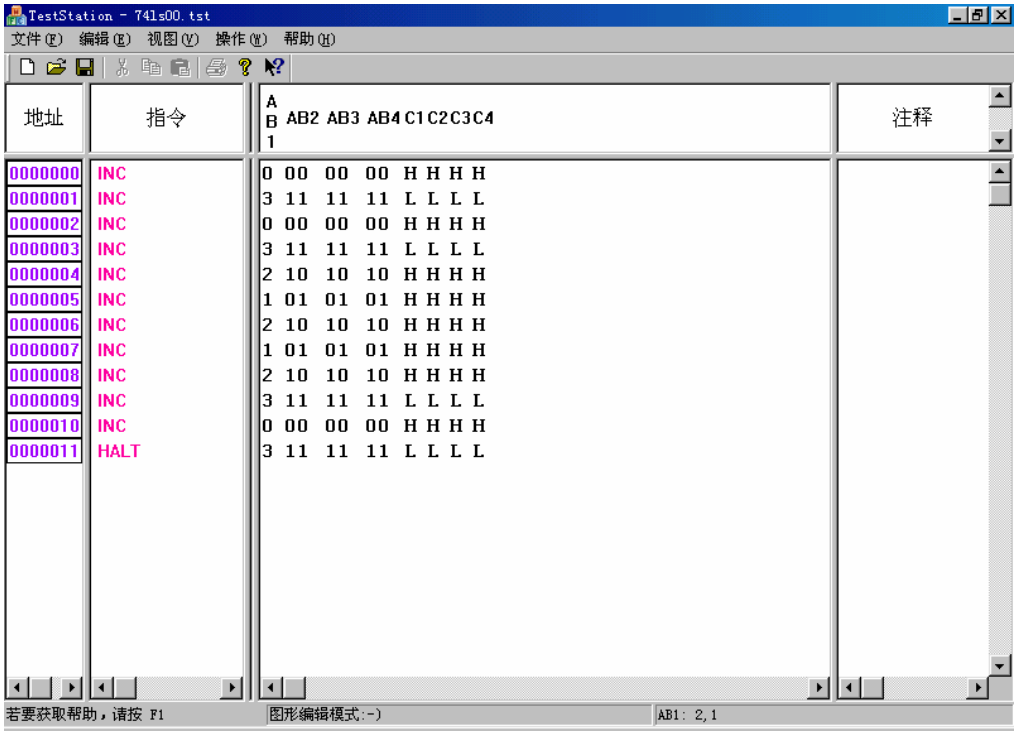


图 2-41

在图形编辑过程中，可以随时更改编辑模式，以利于调试或图形的书写。

5. 重新分组

图形分组后，如果认为，分组不合适，可以随时调整，管脚的图形会随管脚一起被调整到新的组中。执行菜单“操作”→主控界面命令，切换到主控界面。单击“参数设置”按钮，打开参数设置对话框。选择标签“管脚分组”，删除 AB2 管脚组，然后使用添加命令，将管脚 13、14 添加到 AB1 组中，如图 2-42 所示。

图 2-42

单击确定，返回主控界面，单击编辑图形按钮，打开图形编辑界面，如图 2-43 所示，为调整后的图形。

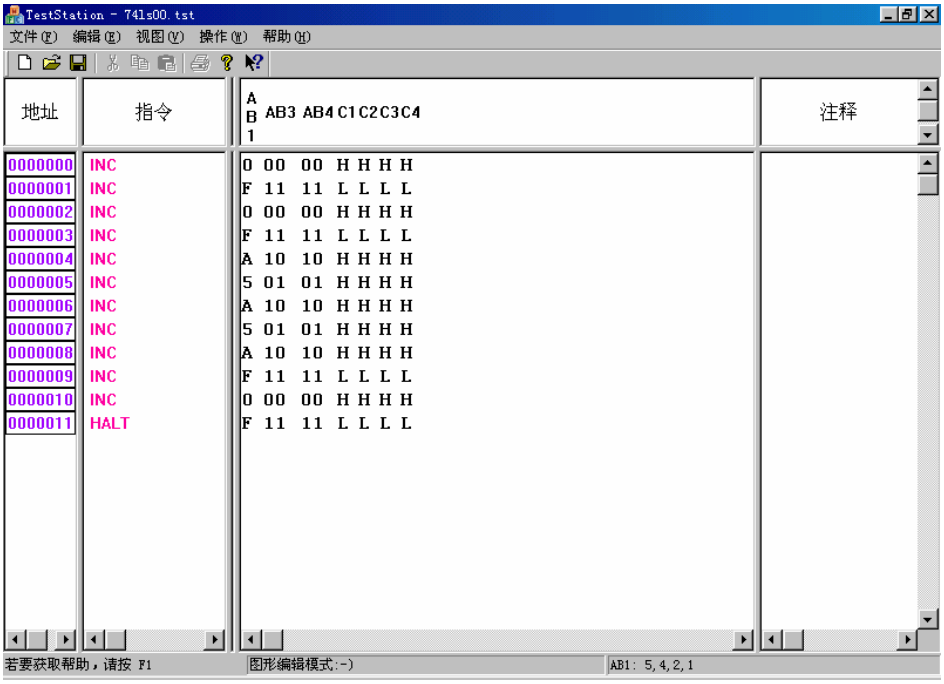


图 2-43 AB1 重新分组

对于组调整的几点说明：

（1）如果将管脚调整到新的组中，其预设参数与新组相同。

（2）如果新增组，则该组中的预设参数为空，需要用户重新设定，主要包括编辑格式、时序设置和参考电平。

（3）若删除管脚，则管脚的图形仍然被保留，恢复时图形自动恢复。但如果图形长度发生变化，其保存的值也会发生变化。图形减少，该管脚图形减少，增加时，该管脚图形被自动设置为 X 或 Z。

五.图形调试

对于数字芯片的测试，大量的工作是在编辑图形和调试图形，因此方便的图形编辑调试器，将大大缩短用户开发数字芯片的时间。

本软件可以实时调试用户的测试图形，使用户在最短的时间内完成图形的调试。

1. 启动调试

操作步骤如下：

（1）打开测试仪，装上被测芯片（如 74ls00），并打开测试仪的电源。

（2）运行 TestShell，选择 74ls00 测试程。

（3）单击“加载图形”按钮，将测试图形加载到测试仪中。

（4）单击图形编辑按钮，启动图形编辑界面，如图 2-44 所示。也可以直接单击“调试图形按钮”启动图形调试。

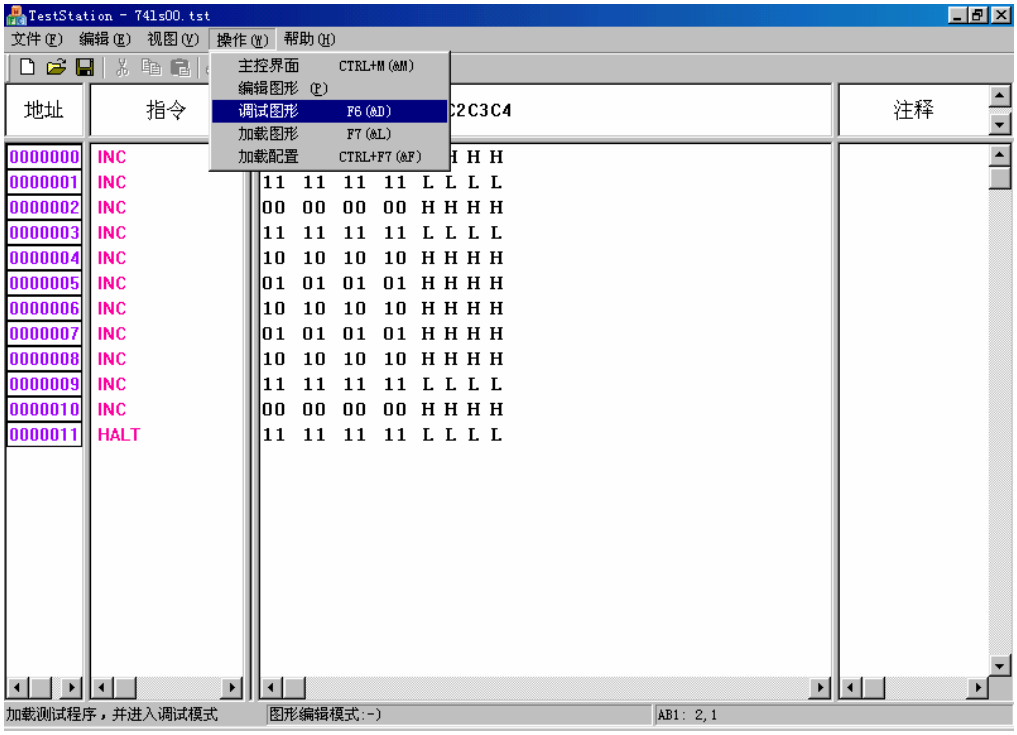


图 2-44 进入图形编辑界面

执行菜单操作→调试图形，启动调试，弹出如图 2-45 所示的调试选项对话框。

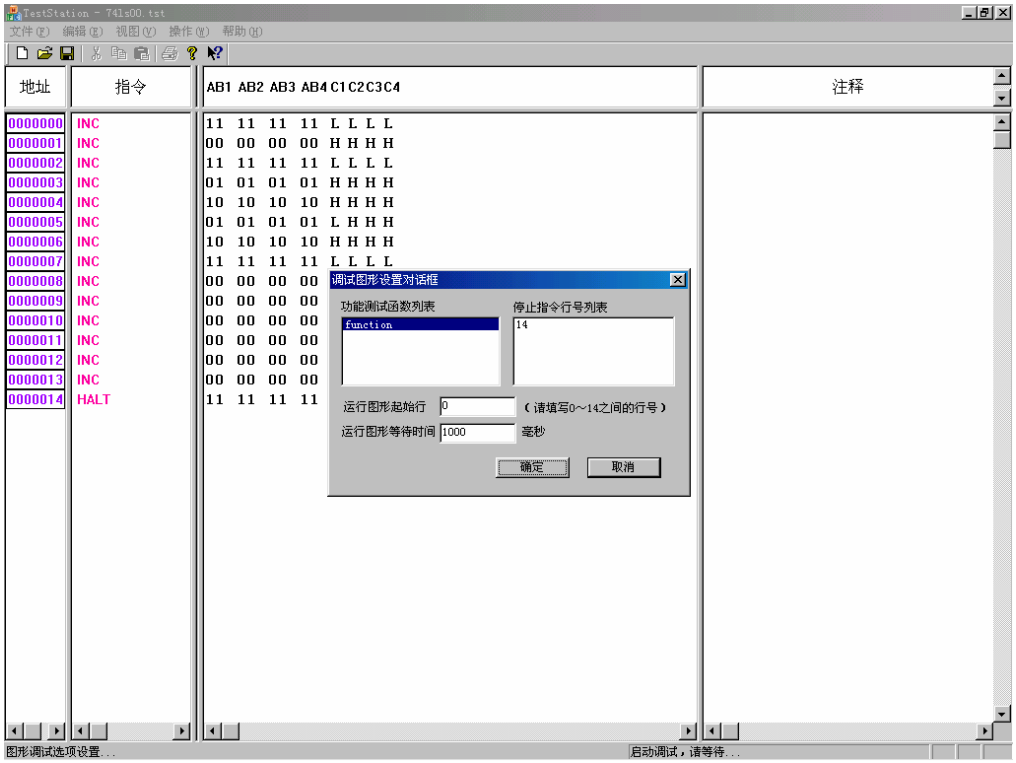


图 2-45

对话框包括 3 个部分：

- 功能测试函数列表框：列出用户测试程序中，所有功能测试的函数名，可以选择一个函数作为图形调试的运行函数。
- 停止指令列表框：列出测试图形中的所有指令的位置，可以帮助用户选择从测试的哪行

开始启动调试。对于多个功能测试编写在一个图形文件中的图形调试是非常有用的。单击停止指令的行号，运行起始行的数字自动填写为停止指令行＋1；如果是最后一行，启动行为最后一行。

- 运行起始行：由用户填写从哪行测试图形开始调试。文本框列出了可以填写的行号的范围。范围由测试图形的长度决定

选择起始行为 2，单击确定，进入调试状态，如图 2-46 所示。

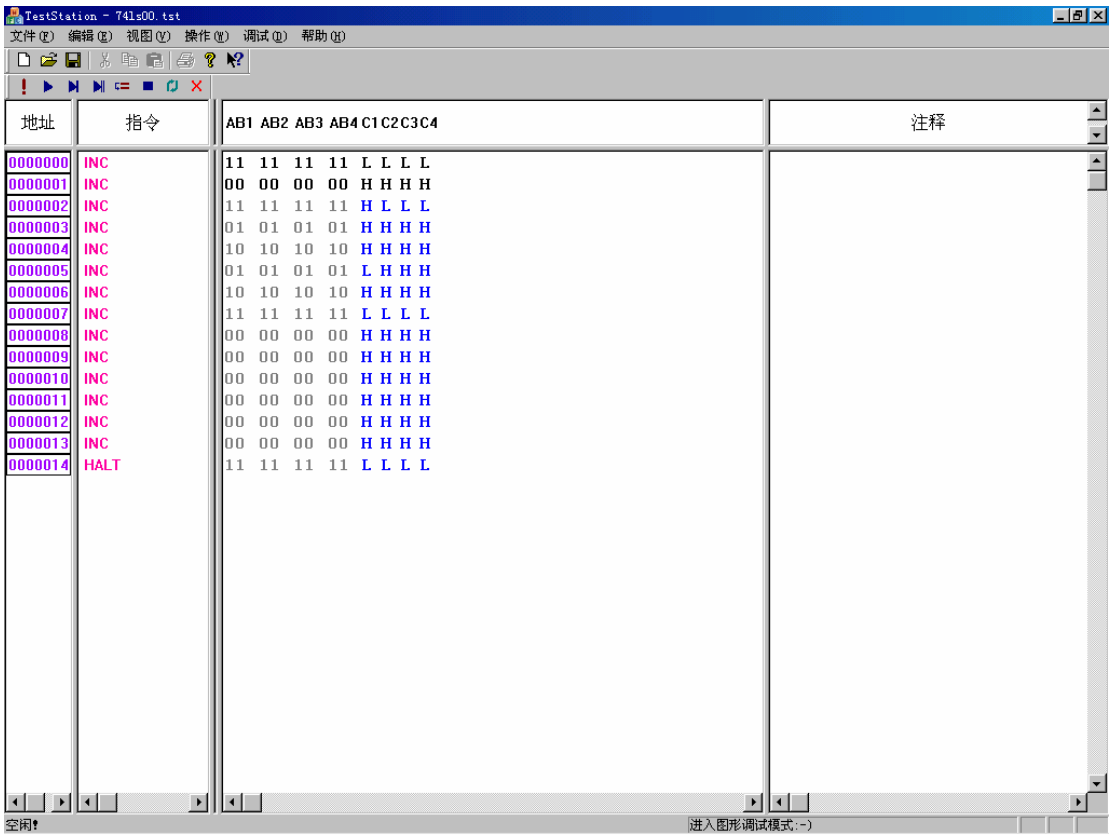


图 2-46 启动调试

进入调试状态，用户图形按输入输出组用不同的颜色显示。输入管脚组图形为灰色；输出管脚组图形：未运行时为蓝色；运行时，通过的图形为绿色，不通过的图形为红色。不调试的图形仍为黑色。

同时，系统自动添加调试菜单和工具栏，如图 2-46 所示。调试工具栏共 7 个按钮。每个按钮均有快捷键，与 VC 的快捷键设置基本相同。

2. 调试命令说明

：进行功能测试，测试通过，显示 PASS，测试失败显示 FAIL，并显示出错的图形。

：运行图形按钮，快捷键 F5。单击该按钮，从停止行运行图形到最后一行。并更新图形。如图 2-3 所示。

：停止按钮，快捷键 Shift+F5。单击停止按钮，退出图形运行，并返回到起始状态。

：运行到当前行按钮，快捷键 Ctrl+F10。单击该按钮，图形将运行到当前行

：运行到指定行按钮，快捷键 F9。单击该按钮，弹出输入对话框，用户填入要停

止的函数，确定后图形将运行到指定的行，并更新界面。

：单步运行图形按钮，快捷键 F10。单击该按钮，图形运行一步。

：更新图形按钮，快捷键 F8。调试时，可以修改图形，这时需要单击该按钮，向测试仪更新图形。

：退出调试按钮，快捷键 Shift+F6。单击该按钮，退出调试模式，进入图形编辑模式。

在正常模式下，启动调试，快捷键为 F6。

3. 调试图形示例：

单击“运行按钮”，启动图形，运行结果如图 2-47 所示。

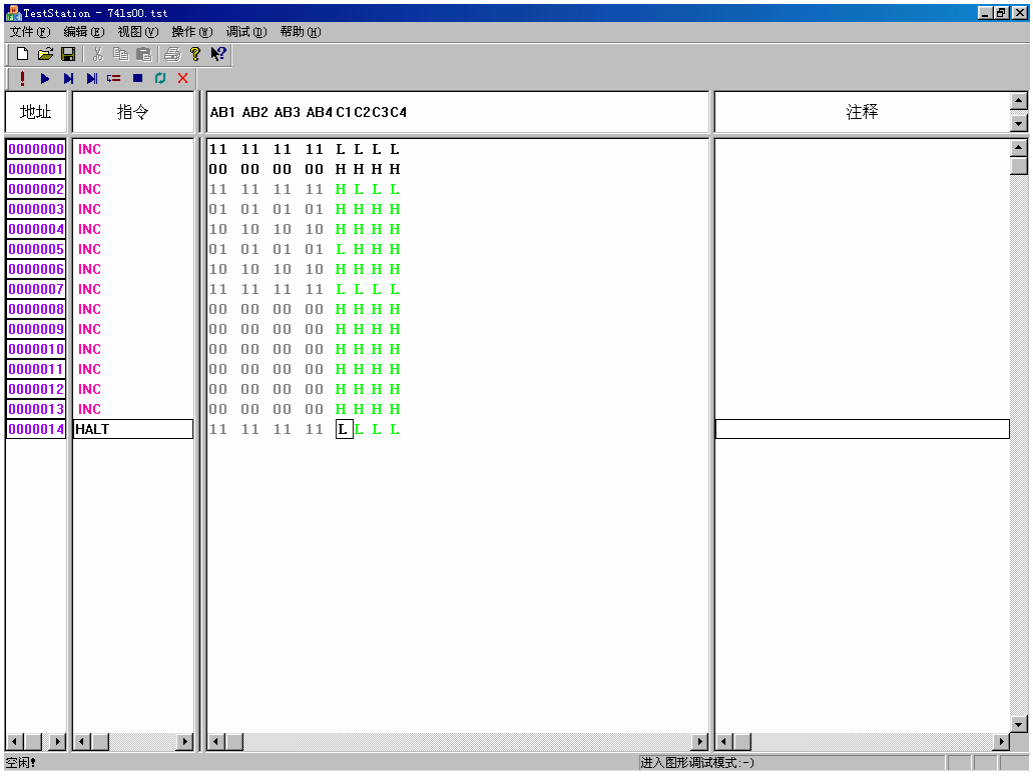


图 2-47 运行图形

单击“停止按钮”，停止运行，则图形恢复为运行前的状态，如图 2-48 所示。

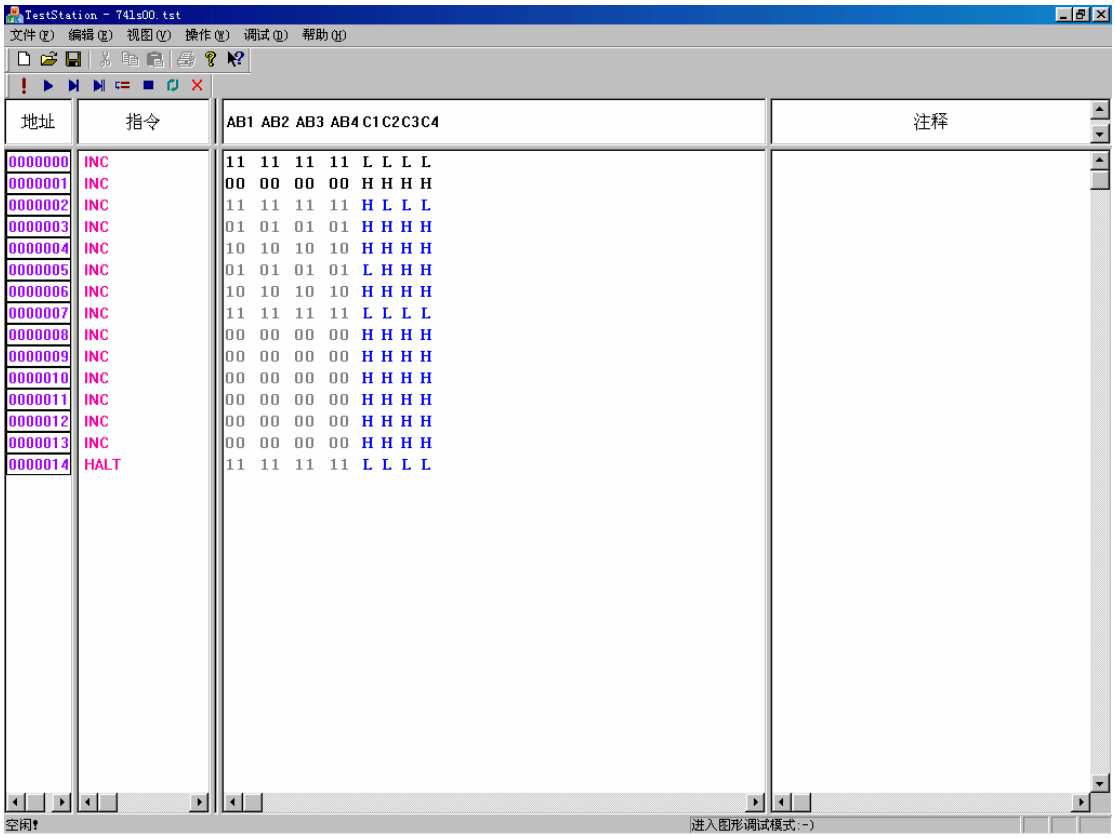


图 2-48 停止运行

单击单步运行按钮，图形就会从当前行运行一步，再单击一次，再运行一步，如图 2-49 所示为执行两次单步运行。

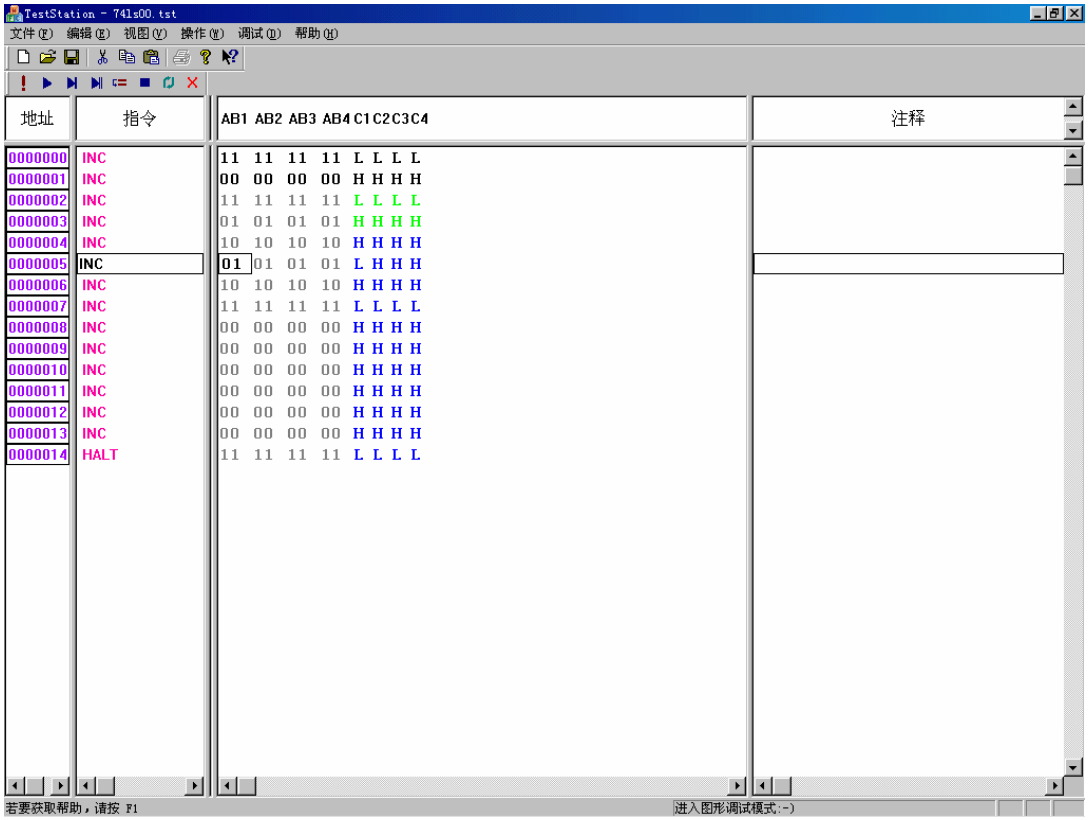


图 2-49 单步运行图形

在行号窗口，单击鼠标左键，选中行号 7，单击运行到当前行按钮，图形运行到第 7 行，如图 2-50 所示。

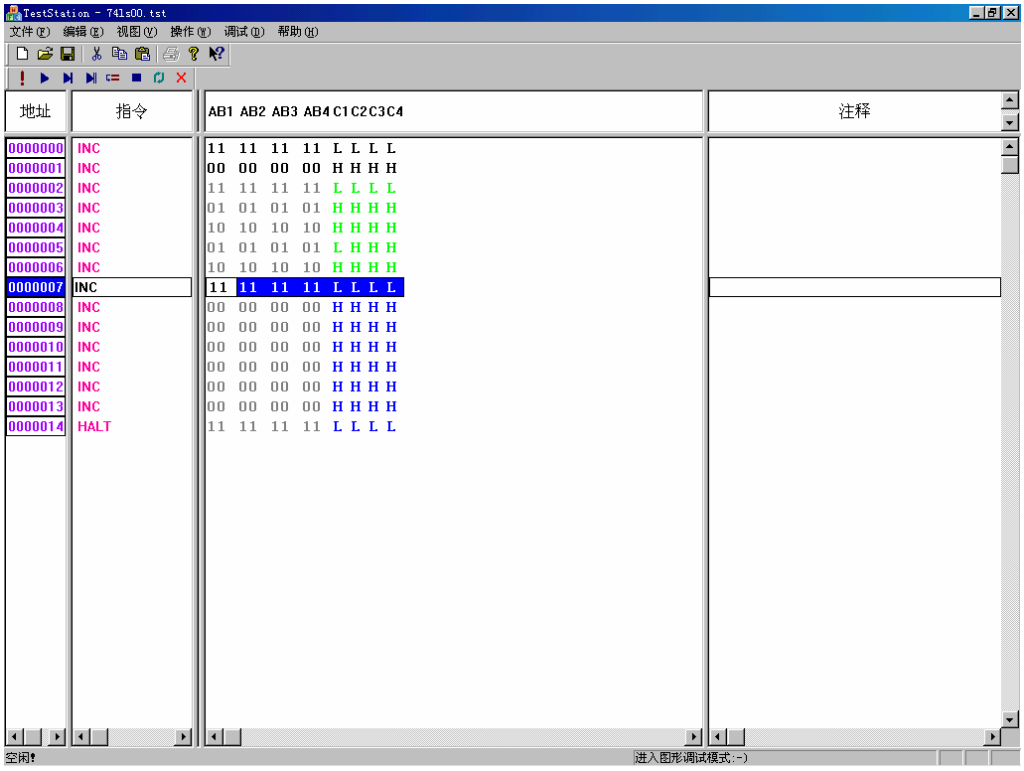


图 2-50 运行到指定行

将图形文件的第 3 行的 C1 图形改为 H，同时单击按钮，更新测试图形,如图 2-51 所示。

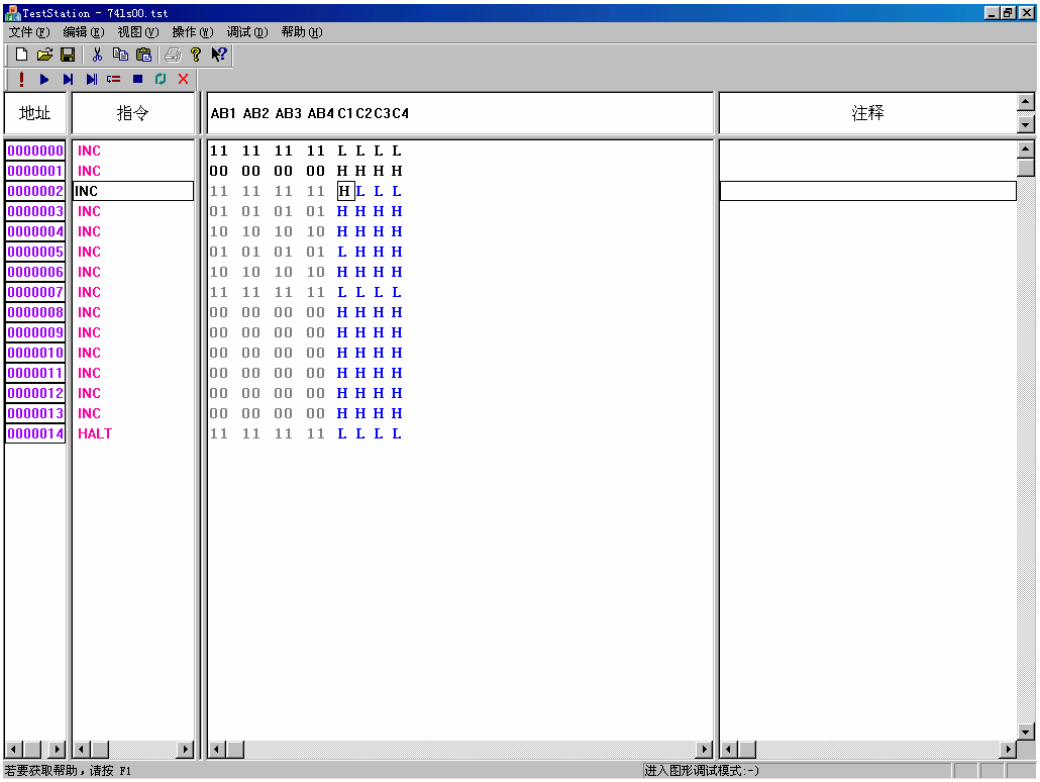


图 2-51 更改并刷新测试图形

单击运行图形按钮，可以看到如图 2-52 的运行结果。错误的图形被标出，同时显示出期望值。

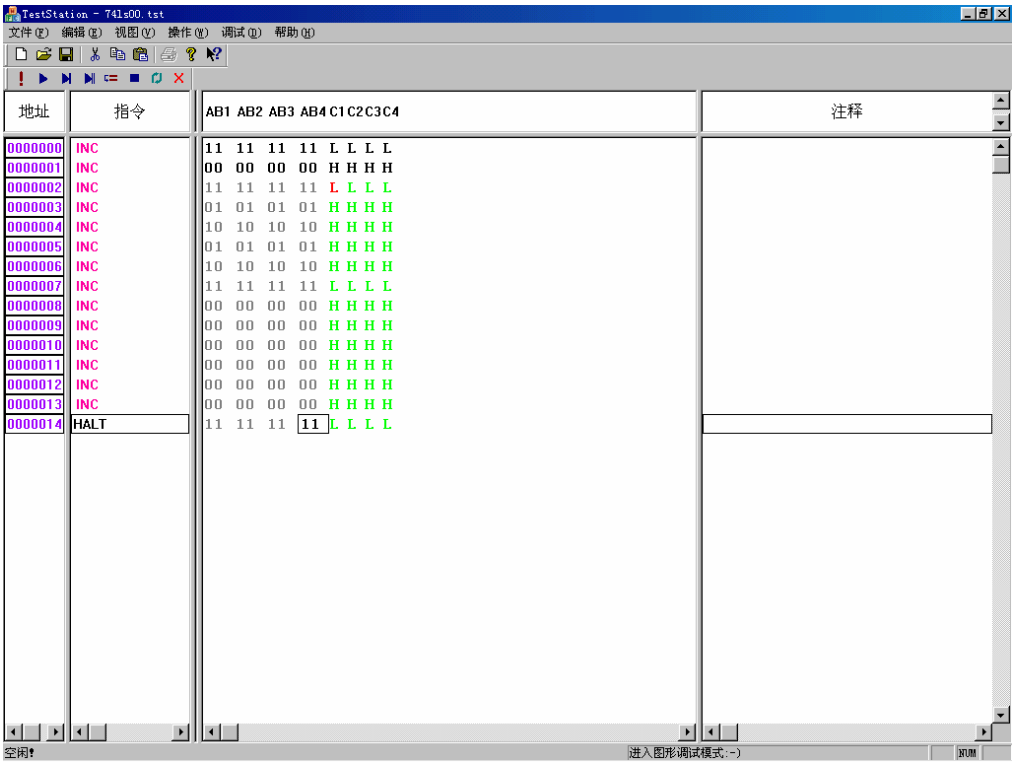


图 2-52 重新运行图形

单击第 2 行的 C1 的图形，可以显示出，实际的图形，如图 2-53 所示。

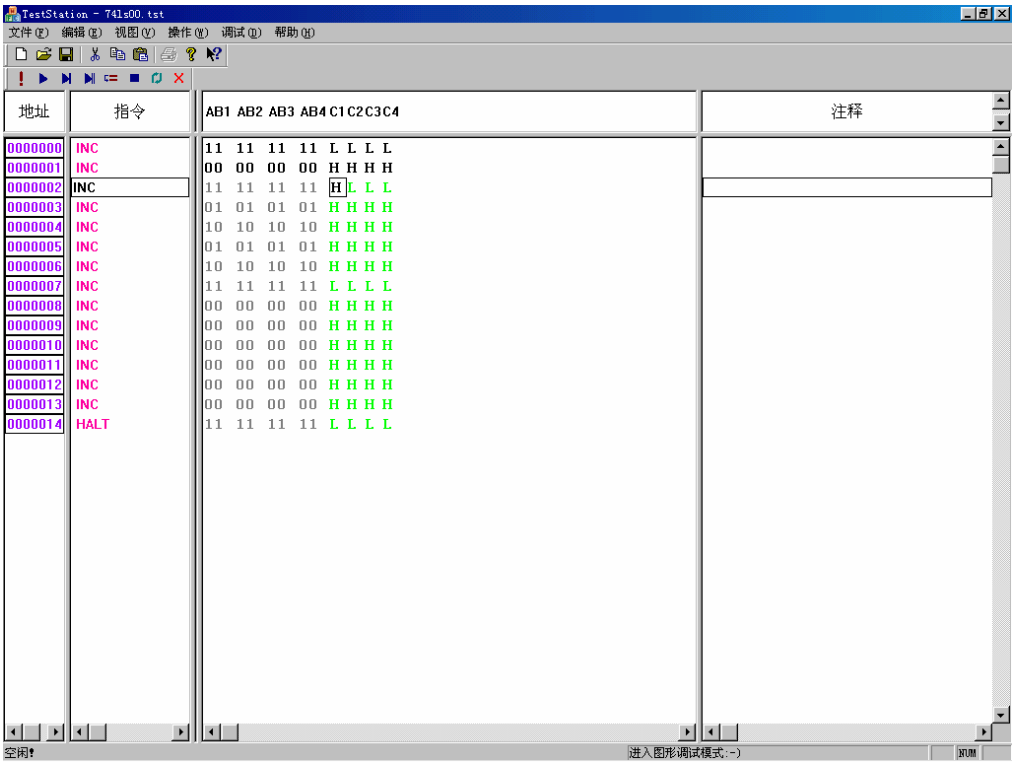


图 2-53 显示 C1 的图形

调试图形，仅仅在用户界面显示错误或正确的图形，并不修改源图形文件。停止运行后，

图形会自动恢复。

4. 图形调试的几点说明

- 运行调试功能，必须从 TestShell 应用程序启动图形编辑界面。
- 必须打开测试机，否则不能进入调试界面。启动调试按钮，将对系统开机状态进行检测，不开机，不会进入到调试状态。
- 启动调试前，请先加载图形。不加载图形，图形将是随机值。
- 对于测试程序，要求被调试的功能函数，必须包括完整的施加条件，即只运行功能测试函数时，就可以完成相关的测试。否则调试不能正确的进行。
- 当图形全部通过时，功能测试通过。
- 单步调试通过的测试图形，连续运行后必然通过。
- 目前功能测试和单步调试已经完成，调试图形使用这两个工具可以完成。
- 对于只有 INC 和和 HALT 的图形，系统提供学习功能。学习功能能正确的前提是用户必须保证有一行到两行以上的图形可以被正确执行。

六.图形编辑器的辅助功能

图形编辑器除了提供测试图形编辑、在线调试功能外，还包括一些辅助功能，这些辅助功能可以帮助用户进行快速开发测试图形、保存图形设置以及与其他 ATE 或 EDA 工具接口等。辅助功能主要有：保存图形配置、导入图形配置、导入图形文本、导入测试图形以及导入测试图形文本。以下将详细叙述每个辅助功能的使用。

1. 保存图形配置和导入配置

测试图形文件*.tst 为二进制文件，包括两部分内容：一是对被测芯片的预设参数，包括器件管脚分配、分组信息、测试图形运行设置、算法图形设置、管脚格式和管脚电平设置；二是测试图形矢量。

保存配置功能可以将对芯片的全部配置信息保存为单独的文件，当开发相类似芯片时，可以快速应用，节约用户的开发时间。在 TestStation 中，执行文件→保存配置命令，就可以将当前芯片的配置信息保存到硬盘，文件的扩展名*.chp。

在开发新图形文件时，若使用向导，芯片设置对话框中有导入配置的按钮，如图 2-54 所示。

图 2-54 芯片配置对话框

单击“导入配置按钮”，打开文件选择对话框，选择配置文件，如图 2-55 所示，单击打开，向导的各个对话框全部自动完成设置，用户只需更改部分不相同的配置，或直接完成向导。

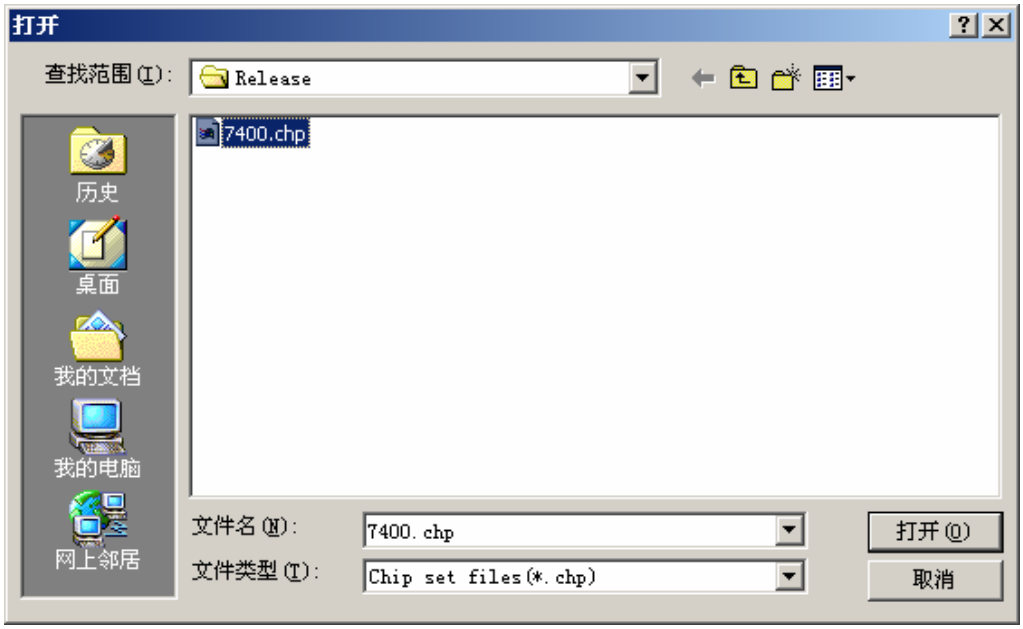


图 2-54 选择配置文件

图 2-56 完成配置导入

2. 导入测试图形

执行文件→导入测试图形命令，可以打开导入测试图形对话框，如图 2-57 所示。

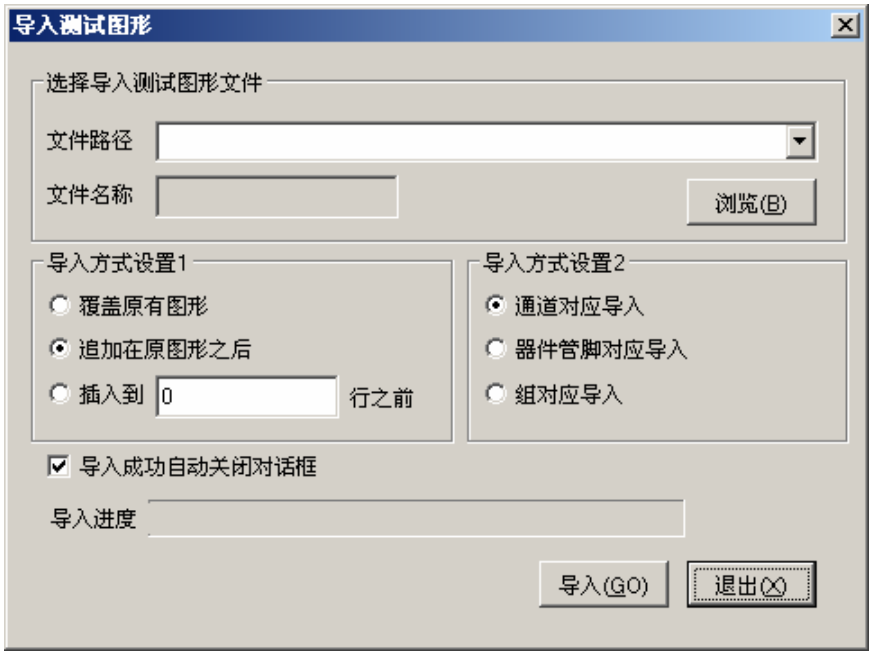


图 2-57 导入测试图形对话框

在导入图形对话框完成测试图形的导入。首先选择被导入图形文件，单击浏览按钮，选择*.tst。然后设置导入方式。导入方式设置 1 用于设置导入图形与原图形矢量的合并方式，有三种：

（1）覆盖原有图形

（2）追加在原有图形之后

（3）插入到指定行图形之前

通过三种方式选择，可以使导入图形更灵活。

导入方式设置 2 用于设置导入图形与原有图形的对应关系，有三种方式：

（1）通道对应导入。

选择该模式，导入图形将和原有图形按通道进行图形匹配，即按通道号一一对应图形。若原有图形的某一通道未分配器件管脚，而导入图形分配了器件管脚，则在导入图形该通道的图形时，自动转换该图形为 Z；若原有图形的某通道分配了器件管脚，而导入图形的通道未分配管脚，则该通道的导入图形为 Z。

（2）器件管脚对应导入

选择该模式，导入图形将和原有图形按器件管脚进行匹配图形，即原有图形的管脚 1 接收导入图形管脚 1 的图形。若原有图形的器件管脚多于导入图形的器件管脚数，则多于部分被忽略；相反，则导入图形为 Z。

（3）组对应导入

选择该模式，导入图形和原有图形按分组顺序中器件管脚排列顺序进行匹配图形。例如原有图形的分组为 AB1、AB2 和 AB3；AB1 中包含管脚 14，6；AB2 中包含 7，15；AB3 中包含 8，1，2；而导入图形组为 CC1 包含 5，7；CC2 包含 9，8，6；CC3 包含 1，2，3，4，则图形的匹配方式为：

原有图形管脚	导入图形管脚
14	5
6	7
7	9
15	8
8	6
1	1
2	2
	3 （被忽略）

通过三种导入模式可以很快完成类似芯片的开发，例如可以很容易从 AT29C010 的测试图形得到 AT27c040 的测试图形；例如可以将 62256 的不同测试图形合并到一个图形文件等。

3. 导入图形文本

执行文件→导入图形文本命令，可以打开导入测试图形对话框，如图 2-58 所示。

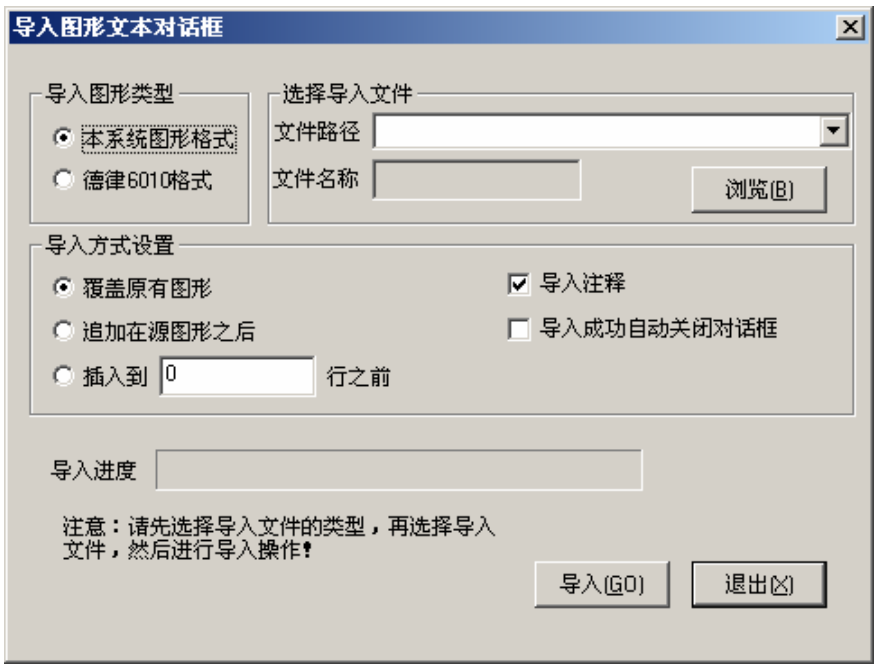


图 2-58 导入图形文本

用户可以不在图形编辑器内编写测试图形，可以使用任何文本编辑器，只要按照指定的格式即可。编辑的测试图形矢量可以通道图形文本导入对话框导入到 TestStation 中。同时文本也是本系统与其他 ATE 和 EDA 软件的一个接口。只要被连接的程序可以按指定格式输出文本，本系统就可以接收。目前系统接受按本系统格式导出的文本格式和德律 6010 的测试图形矢量格式。

4. 导出图形文本

执行文件→导出图形文本命令，可以打开导出测试图形对话框，如图 2-59 所示。该对话框可以实现将测试图形矢量保存成纯文本形式，可供用户查看和修改。

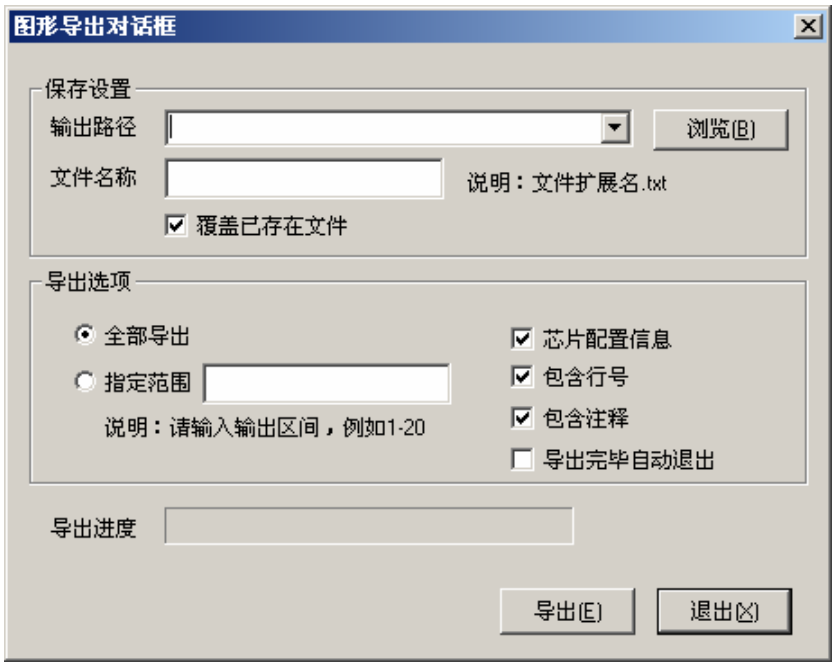


图 2-59 导出图形文本对话框

用户可以自定义导出的内容。

第三章. 波形产生和波形分析

3196D 测试系统为音频测试提供了菜单式的操作方式。使用波形产生器产生波形，只需在选择测试程序或新建测试程序后，单击波形产生按钮，打开波形产生对话框就可以进行各种波形的设置。在进入测试后，系统会自动将用户定义的波形加载到 WG 存储器中。用户只需在测试程序中控制音频源的输出和停止即可。

波形产生对话框具体操作如下：

一. 波形产生对话框操作步骤

使用音频源必须首先选择测试程序。操作步骤如下：

（1）打开 3196D 测试系统主用户界面 TESTSHELL 。

（2）选择测试程序，如 m2822 。

（3）单击“波形产生”按钮，打开波形产生对话框，如图 3-1 所示。

（4）设置波形。

（5）保存波形文件，退出。

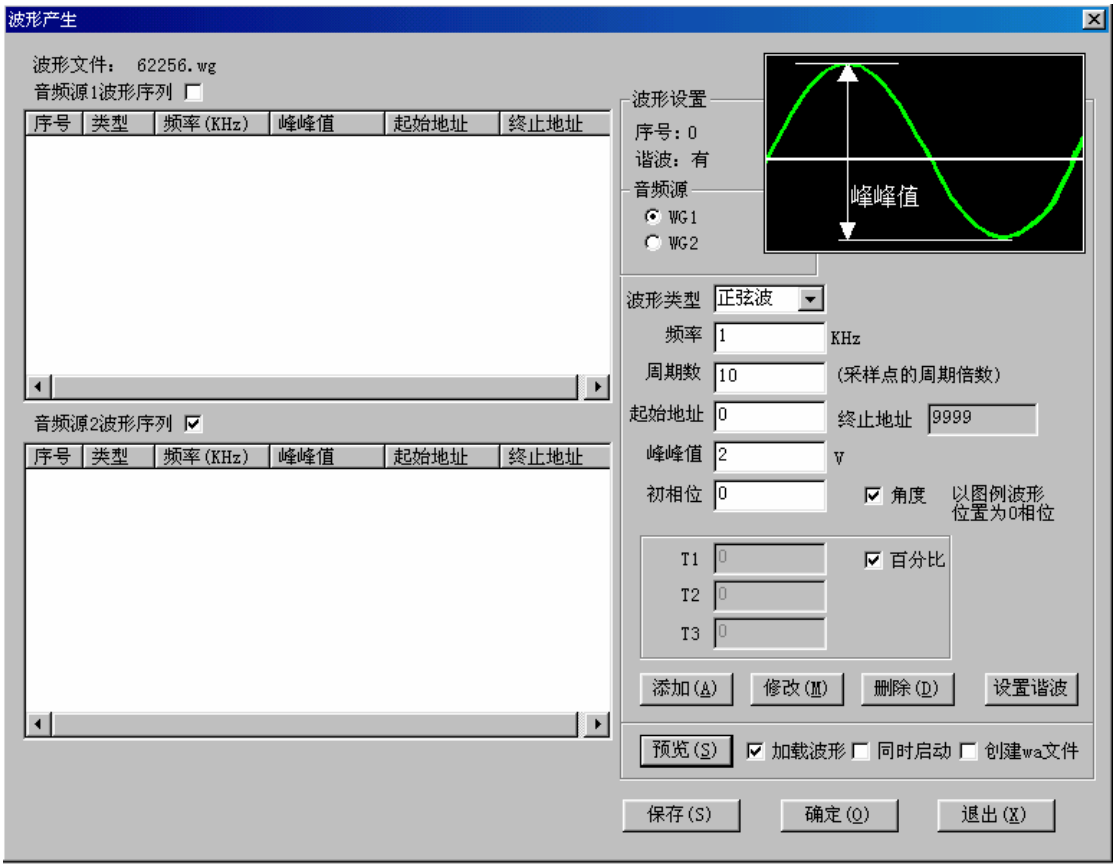


图 3-1 波形产生对话框

二. 波形产生对话框的组成

波形产生对话框分为四部分，如图 3-1 所示：

（1）对话框的左上角显示波形文件的名称。如果波形文件还没有创建则在文件名后显示“[-未创建]”。

（2）左侧两个列表框依次为音频源 1 和音频源 2 中已产生的波形列表。列表框中显示了波形的类型、频率、峰值等信息。选则波形序号，右侧波形设置对话框就会显示波形的设置信息，用户可以通过单击“修改”按钮修改波形，或单击“删除”删除所选择的波形。

（3）右侧为波形设置部分，用户可以通过填写参数既可以产生所需的波形。同时也可以显示已存在的波形信息。

（4）对话框的右下角为对话框的控制按钮。单击保存按钮保存当前的波形设置，单击确定按钮，保存当前设置，并关闭对话框；单击退出按钮，不保存修改，关闭对话框。

三.创建波形

波形设置部分用于完成波形的创建、修改和参数显示。波形设置部分要设置的参数有：

（1）音频源选择

选择创建的波形由哪个音频源发出。

（2）波形类型选择

波形设置部分右上角为波形示例图片框，图中示例了测试仪可以产生的波形类型和该波形需要设置的参数。单击图片框可以自动切换波形类型，也可以通过波形类型的列表框选择波形的类型。如下图所示为单击图片框，显示梯形波。

图 3-2 创建波形

选择不同的波形类型，需要设定不同的时间参数，时间参数的定义由图例中给出。例如对于梯形波需要用户设置 T1、T2 和 T3 三个参数，方波只需要设置 T1 一个参数，而正弦波不需要设置任何的时间参数。

（3）频率和周期数

设置所产生信号的频率，以及产生几个周期的采样点，默认值为一个周期的采样点。任意波形的频率为 10Hz～20KHz 之间。

（4）起始地址和终止地址。

波形是存储在波形存储器中，用户可以任意指定存储器中的位置，存储器为 128K，因而最大地址为 131072；起始地址为偶地址；终止地址不需要用户设置，由系统自动计算。

（5）峰峰值和相位

用户可以任意设置所产生波形的峰峰值和相位角。峰峰值为 0～20V，相位角为 0～359 度或小于一个周期的时间。其中相位是以图例的位置为 0 相位的。这点需要用户注意。

（6）时间参数

时间参数为 T1、T2 和 T3。时间参数的意义由图片框给出，不同的波形需要填写不同的时间参数。时间参数可以是百分比也可以是以毫秒为单位的时间参数。

（7）操作按钮

添加按钮：向列表框中添加一个波形，其参数为参数设置部分所设置的参数。

修改按钮：修改列表框中选中的波形的参数，其参数为参数设置部分所设置的参数

删除按钮：删除列表框中被选中的波形，可以多选。

预览按钮：单击预览按钮，可以将产生的波形发到测试仪的测试头上，用户可以用示波器等工具观察，波形是否正确。由于每次只能有一个波形被选中，因而预览每次只显示当前操作的波形。

参数设置完毕，单击添加按钮，波形就被添加到指定的音频源列表框中。如图 3-3 所示为添加一个峰峰值 2V 的正弦波。

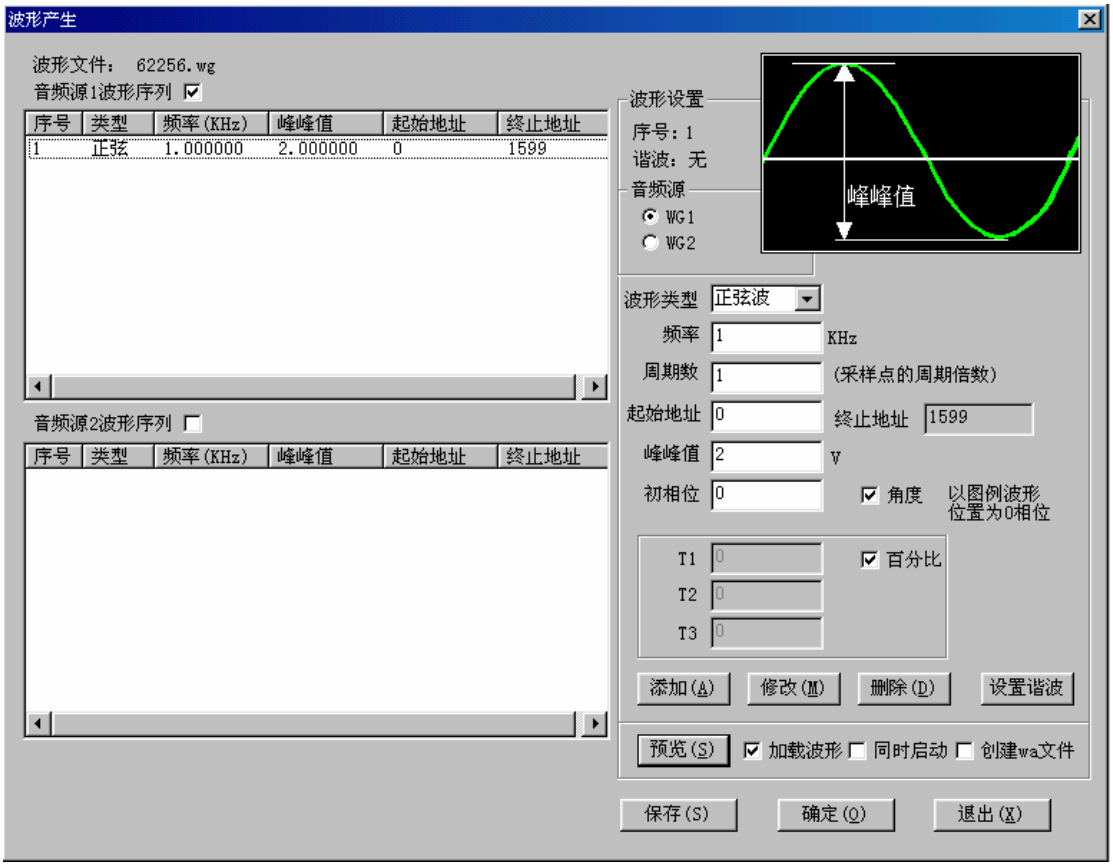


图 3-3 添加一个正弦波

改变音频源，单击添加按钮，该参数的正弦波就被添加到音频源 2 的波形列表中。如图 3－4 所示。

图 3-4 添加正弦波到音频源 2

如果需要修改已经产生的波形，先选中指定列表中的波形序号，波形设置部分就显示出该波形的全部参数，直接修改参数，单击修改按钮即可。例如：将图 3—4 中音频源 1 的波形修改为 2KHz , 4V 正弦波。如图 3—5 所示。

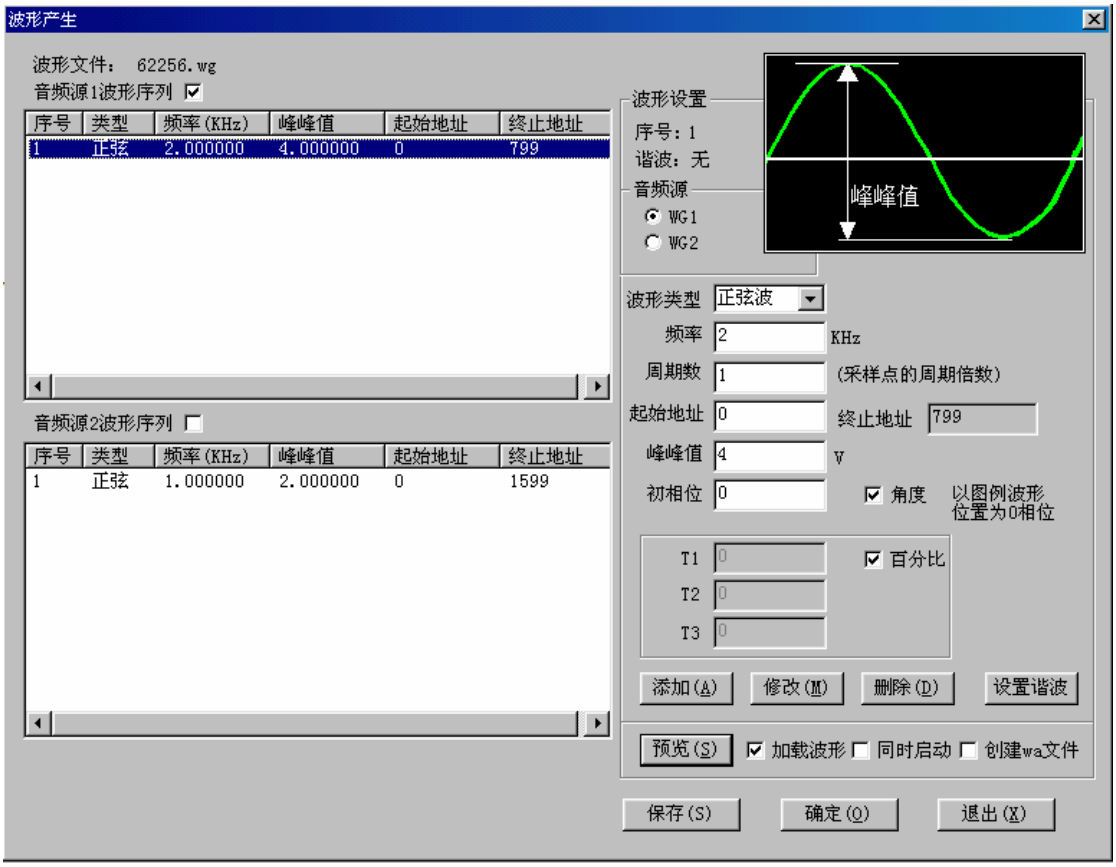


图 3-5 修改波形参数

添加谐波：

如果需要产生一个多次谐波，需要先设定基波的参数，然后选定需要添加谐波的波形，再单击设置谐波按钮，调出谐波设置对话框，图 3－6 所示为以音频源 1 的 1KHz，2V 正弦波为基波，设置谐波参数的对话框：

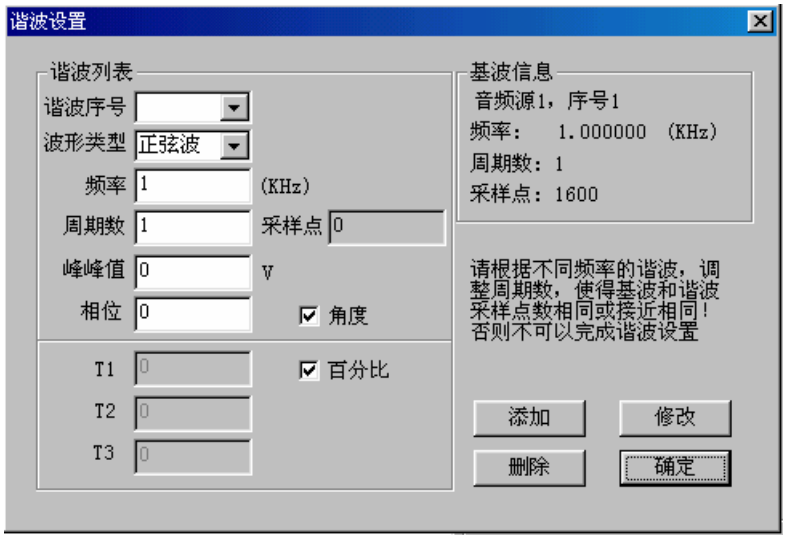


图 3-6 谐波设置对话框

谐波设置对话框分为四部分：

左侧为谐波的参数设置区，右上方为用户所选基波的信息，基波信息下方为提示区，提示区下方为功能按钮。

设置方法：

同单一波形设置方法一样。采样点数是根据谐波的周期数自动机计算出的。需注意的是，计算出的采样点数应和基波的采样点数接近或者相同。例如：添加一个 3KHz，0.05V 正弦波为谐波。见图 3—7 所示（注意周期数的设置，以及采样点的计算所得值）：

图 3-7 添加一个谐波

四. 波形分析

用户可以借助系统提供的波形分析工具分析波形分析器所采集的波形数据，操作步骤如下：

（1）在主界面中单击“波形分析”按钮，打开如图 3-6 对话框

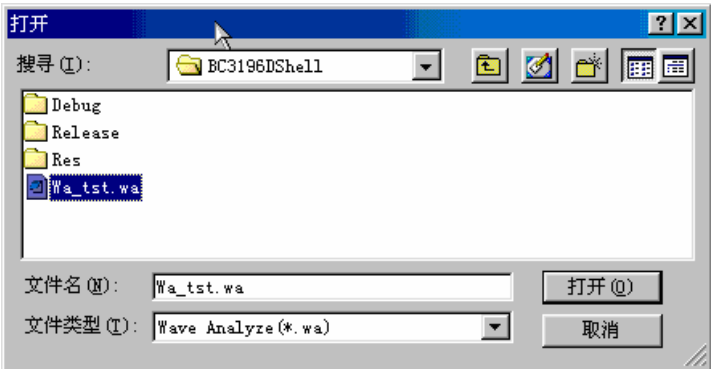


图 3-8 选择波形分析文件

（2）选择所要分析的测试波形文件，然后单击“打开”按钮，打开如图 3-7 所示的对话框。

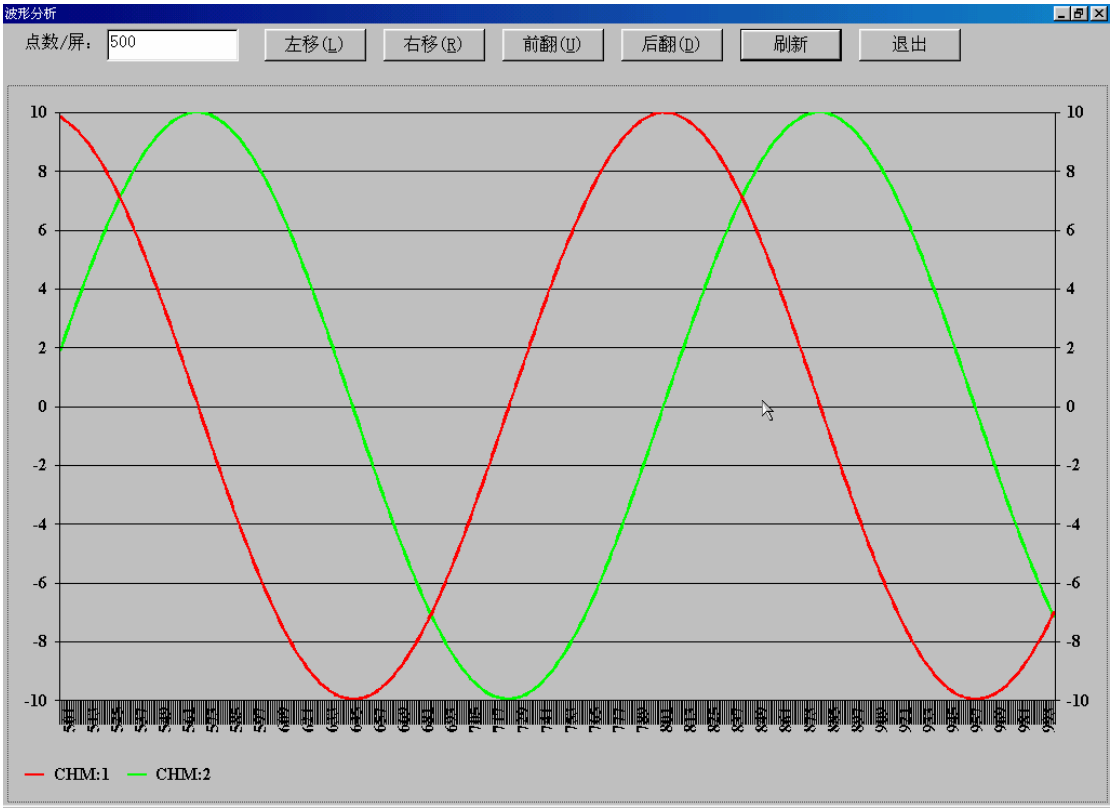


图 3-9 波形分析

- 设置每屏显示的点数；在“点数/屏”后面的编辑框中键入一个整数值，然后单击“刷新”按钮，这样在下面的图表中每屏将按所设点数显示；
- 向左移一个点，单击“左移”按钮即可；
- 向右移一个点，单击“右移”按钮即可；
- 向前翻一页，单击“前翻”按钮即可；
- 向后翻一页，单击“后翻”按钮即可；
- 退出，单击“退出”按钮退出该对话框；

第四章　测试程序的开发

一. 如何测试数字芯片

如何测试数字芯片是用户关心的一个重要问题。通过了解数字芯片的测试过程，可以更快的熟悉数字子系统的各个部分及其功能和使用。图 4-1 为数字芯片测试的基本原理。

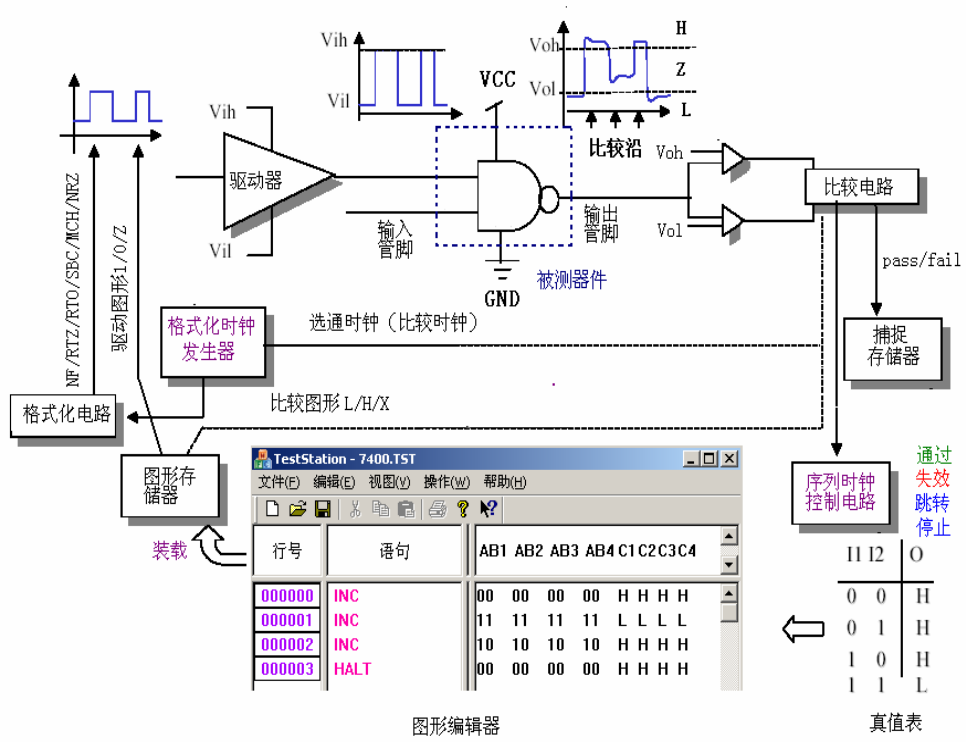


图 4-1 数字芯片测试流程

如图 4-1 所示，要测试数字芯片必须具有：图形存储器、时钟发生电路、格式化电路、管脚驱动器、电平比较器和序列时钟控制电路。

捕捉存储器可以方便的帮助用户分析测试图形，对于复杂芯片的测试和分析非常的有用。

整个系统在序列时钟的控制下按照一定的时序运行，完成器件的测试。

数字器件测试需要如下步骤：

（1）首先编写测试图形，即数字芯片的真值表

（2）将测试图形装载到图形存储器中

（3）设置驱动管脚的格式化、起始、结束时间

（4）设置比较器的选通时间

（5）设置驱动电平（VIH、VIL）和比较电平（VOH、VOL）

（6）启动测试，筛选 Pass 或 Fail 芯片或者分析测试图形

二. 数字芯片测试范例

（一）74LS245 八总线接收器、发送器

第一步：开发新程序

输入路径、器件名、参数表名及分类

第二步：设置测试参数　如图 4-2 所示：

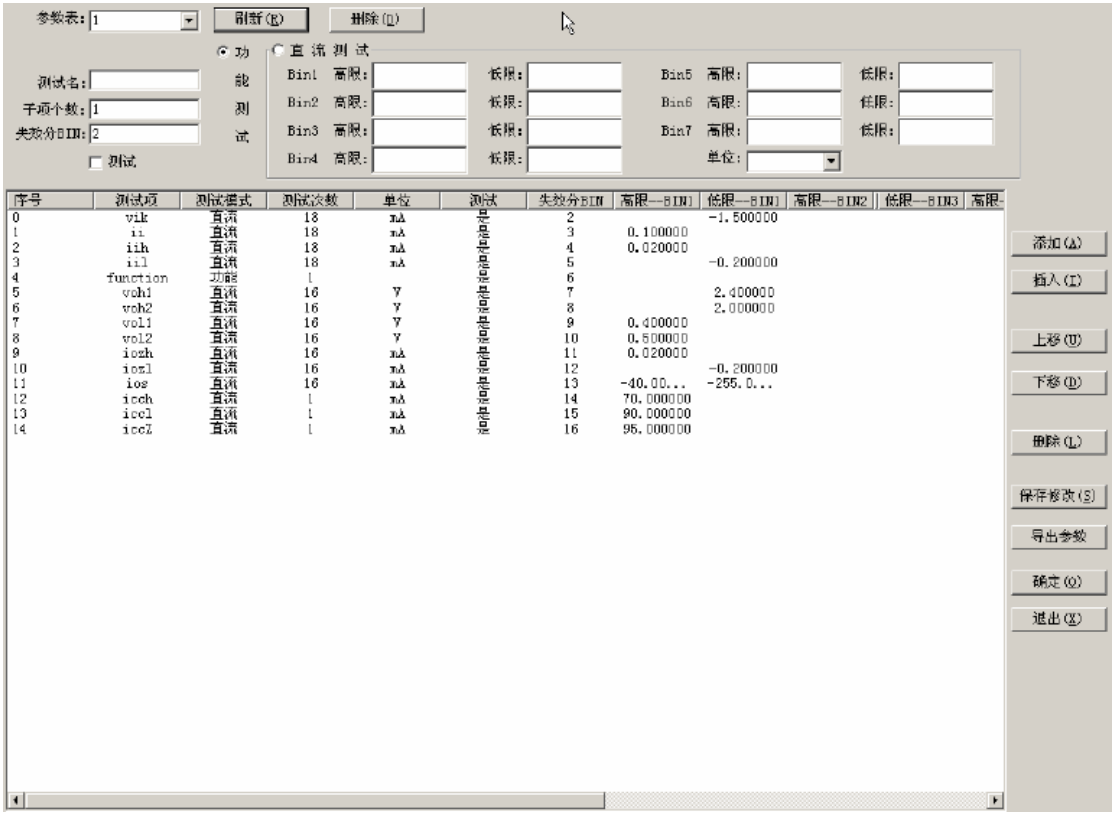


图 4-2 设置测试参数

第三步：编辑源文件

74LS245 源文件见附录

第四步：编辑图形

1．创建测试图形，单击编辑图形按钮，弹出对话框，选择否。

2．芯片设置对话框，如图 4-3 所示。

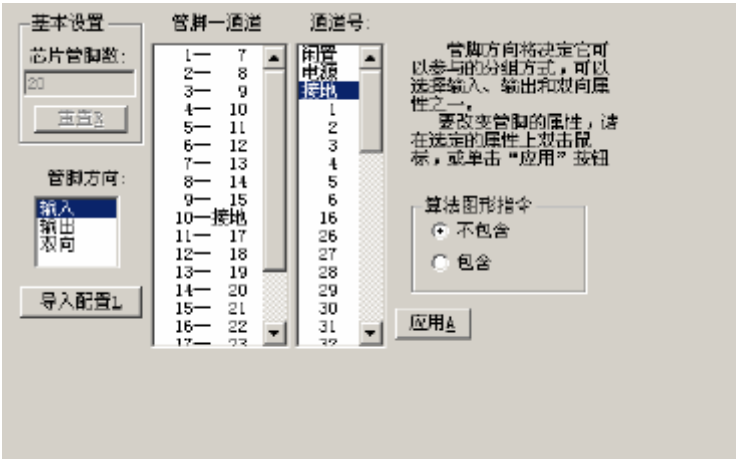


图 4-3 芯片设置对话框

分配管脚通道，指定输入输出脚，默认 16 通道为接地

3．运行设置对话框

设置主时钟频率 1MHz，比较时钟 1 前沿为 60ns

4，算法图形配置对话框

74LS245 不包含算法图形，故这里不需设置

5．管脚分组对话框

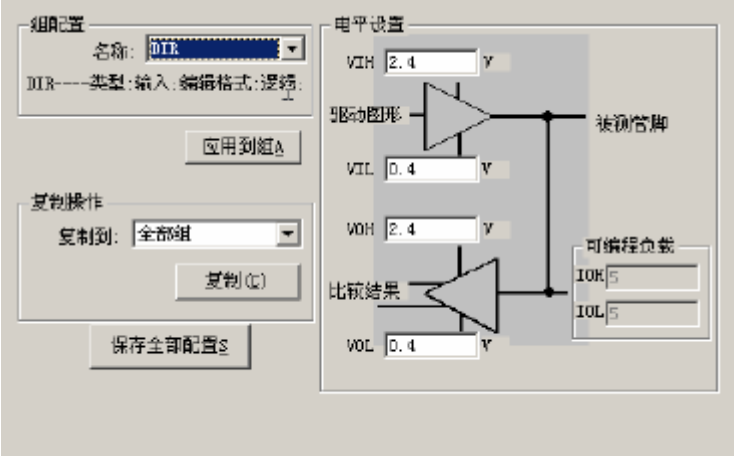
对器件管脚分组，把功能相同的管脚分为一组，分配好了如下图所示：

6．时序设置对话框

此芯片不需要格式化时钟，所有输入管脚均为 NF 无格式，且驱动时钟为 15；所有输出脚则 比较类型为 EDGE 边沿比较，比较时钟为 1。双向管脚则即要设置驱动时钟也要设置比较时钟。

7．参考电平设置对话框

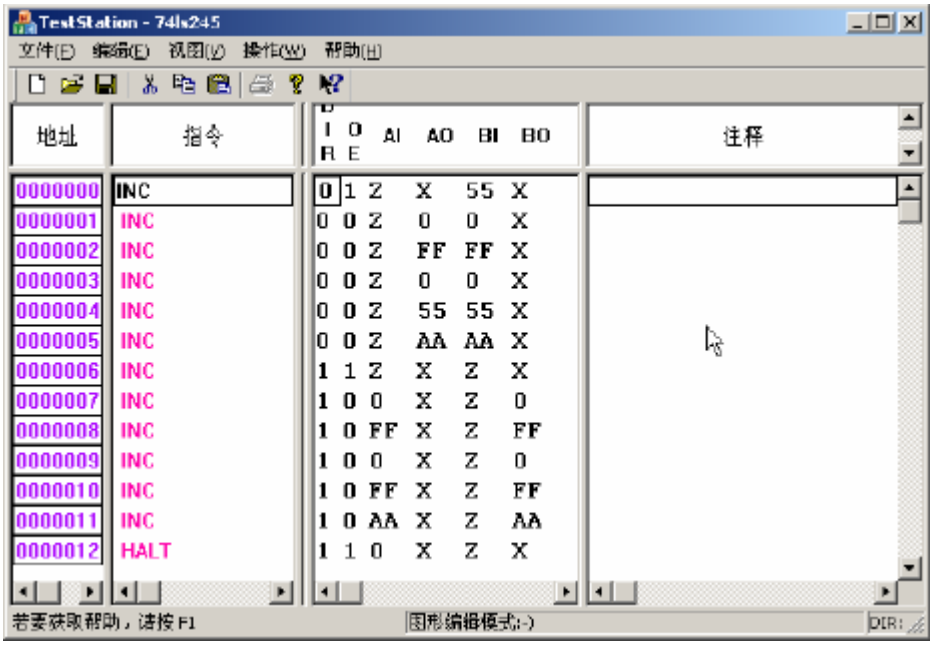
根据器件手册输入相应 VIH，VIL，VOH，VOL。（调试功能的时候，可先放宽驱动电平，待功能调试通过后，再把线设回规定值） 如下图所示：



8．图形编辑

按照器件手册，把器件真值表输入到对应项。

74LS245 功能表如下图所示：



至此 74LS245 开发完毕。

（二）测试程序源文件

测试源文件，74ls245.cpp

```
#include "74ls245.h"
double   IIKF=-18;              //测量 VIK 时所加的电流
double   VIF[]={5.5,7};         //测量 II 时所加的电压
double   VIHF=2.7;               //测量 IIH 时所加的电压
double   VILF=0.4;               //测量 IIL 时所加的电压
double   IOHF[]={-3,-15};       //测量 VOH 时所加的电流
double   IOLF[]={12,24};         //测量 VOL 时所加的电流
double   VOZHF=2.7;              //测量 IOZH 时所加的电压
double   VOZLF=0.4;              //测量 IOZL 时所加的电压
double   VOSF=0;                 //测量 IOS 时所加的电压
```

```
//测试项名：vik       测试第一项  VIK
MYDLLAPI double test0(double* grpTestValue,short* pingrp)
{
      SYSRESULT result;
      int j;
      hvforce_v(3,4.75,50);                    //通过 3 号 PMU，设定施加电压值 4.75V
      result=dut_fimv("1，19",IIKF,5,5);
      //先对 DIR“1”和 E“19”脚分别加流（IIKF）测压，嵌压值为 5mA，延时 5ms
      for (j=0;j<2;j++)
      {
            grpTestValue[j]=result.measval[j];
            pingrp[j]=result.index[j];
      }
      ptn_to_dut("1,19");                      //连接图形通道到管脚“1，19”
      run_pattern(0,ST_ADDR,8);
      //运行图形函数，停止在第 8 行，即指定方向 A→B，再测量输入管脚 A
      result=dut_fimv("2-9",IIKF,5,5);
      //对 8 个输入管脚 A 分别加流（IIKF）测压，嵌压 5V，延时 5ms
      for (j=2;j<10;j++)
      {
            grpTestValue[j]=result.measval[j-2];
            pingrp[j]=result.index[j-2];
      }
      run_pattern(0,ST_ADDR,2);
      //运行图形函数，停止在第 2 行，即指定方向 B→A，再测量输入管脚 B
      result=dut_fimv("11-18",IIKF,5,5);
      //对 8 个输入脚分 B 别加流（IIKF）测压，嵌压 5V，延时 5ms
      for (j=10;j<18;j++) //系统自动生成，用于显示和回传数据
      {
            grpTestValue[j]=result.measval[j-10];
            pingrp[j]=result.index[j-10];
      }
      hvdisconn_pmu(3);
      return 0;
}

//测试项名：ii       测试第二项  II
MYDLLAPI double test1(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int j;
    hvforce_v(3,5.25,50);
    ptn_to_dut("1,19",PS_OFF);
```

```
    result=dut_fvmi("1,19",VIF[0],0.35, 5);
     //对输入脚“1，19”分别加压（VIF0）测流，嵌流 0.35mA，选定量程 4UA，延时 5ms
    for (j=0;j<2;j++)
     {
          grpTestValue[j]=result.measval[j];
          pingrp[j]=result.index[j];
     }
    ptn_to_dut("1,19");                    //连接图形通道到管脚“1，19”
    run_pattern(0,ST_ADDR,8);
    //运行图形，从第 0 行起，停止在第 8 行，既选定方向由 A→B，测输入脚 A
    result=dut_fvmi("2-9",VIF[1],0.35,MI_4UA,5);
    //对输入脚 A“2-9”分别加压（VIF1）测流，嵌流 0.35mA，量程 4UA，延时 5ms
    for (j=2;j<10;j++)
     {
          grpTestValue[j]=result.measval[j-2];
          pingrp[j]=result.index[j-2];
     }
    run_pattern(0,ST_ADDR,2);
    result=dut_fvmi("11-18",VIF[1],0.35, 5);
    for (j=10;j<18;j++)
     {
          grpTestValue[j]=result.measval[j-10];
          pingrp[j]=result.index[j-10];
          }
     hvdisconn_pmu(3);
     return 0;
}

//测试项名：iih      测试第三项 IIH
MYDLLAPI double test2(double* grpTestValue,short* pingrp)
{
     SYSRESULT result;
     int j;
     hvforce_v(3,5.25,50);
     ptn_to_dut("1，19",PS_OFF);
     result=dut_fvmi("1,19",VIHF,0.03, 5);
          for (j=0;j<2;j++)
     {
          grpTestValue[j]=result.measval[j];
          pingrp[j]=result.index[j];
     }
     ptn_to_dut("1,19");
     run_pattern(0,ST_ADDR,8);
     result=dut_fvmi("2-9",VIHF,0.03, 5);
```

```
    for (j=2;j<10;j++)
    {
        grpTestValue[j]=result.measval[j-2];
        pingrp[j]=result.index[j-2];
    }
    run_pattern(0,ST_ADDR,2);
    result=dut_fvmi("11-18",VIHF,0.03, 5);
    for (j=10;j<18;j++)
    {
        grpTestValue[j]=result.measval[j-10];
        pingrp[j]=result.index[j-10];
    }
    hvdisconn_pmu(3);
    return 0;
}

//测试项名：iil      测试第四项  IIL
MYDLLAPI double test3(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int i;
    hvforce_v(3,5.25,50);
    ptn_to_dut("1,19",PS_OFF);
    result=dut_fvmi("1,19",VILF,0.5, 5);
    for ( i=0;i<2;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
    ptn_to_dut("1,19");
    run_pattern(0,ST_ADDR,8);
    result=dut_fvmi("2-9",VILF,0.5, 5);
    for (i=2;i<10;i++)
    {
        grpTestValue[i]=result.measval[i-2];
        pingrp[i]=result.index[i-2];
    }
    run_pattern(0,ST_ADDR,2);
    result=dut_fvmi("11-18",VILF,0.5, 5);
    for (i=10;i<18;i++)
    {
        grpTestValue[i]=result.measval[i-10];
        pingrp[i]=result.index[i-10];
    }
```

```
    hvdisconn_pmu(3);
    return 0;
}

//测试项名：function      测试第 5 项  功能  FUNCTION
MYDLLAPI double test4(double* grpTestValue,short* pingrp)
{
    short fail=0;
    hvforce_v(3,5,250);
    ptn_to_dut("1-9,11-19");
    fail=run_pattern(0,ST_FAIL);
    hvdisconn_pmu(3);
    return fail;
}

//测试项名：voh1      测试第 6 项  VOH（IOHF=-3mA 时）
MYDLLAPI double test5(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int i;
    hvforce_v(3,4.75,100);
    ptn_to_dut("1-9,11-19");
    ptn_to_dut("2-9",PS_OFF);
    run_pattern(0,ST_ADDR,2);
    result=dut_fimv("2-9",IOHF[0],5,5);
    for (i=0;i<8;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
    ptn_to_dut("1-9,11-19");
    ptn_to_dut("11-18",PS_OFF);
    run_pattern(0,ST_ADDR,8);
    result=dut_fimv("11-18",IOHF[0],5,5);
    for (i=8;i<16;i++)
    {
        grpTestValue[i]=result.measval[i-8];
        pingrp[i]=result.index[i-8];
    }
    hvdisconn_pmu(3);
    return 0;
}

//测试项名：voh2      测试第 7 项  VOH（IOHF=-15mA 时）
```

```
MYDLLAPI double test6(double* grpTestValue,short* pingrp)
{
	SYSRESULT result;
	int i;
	hvforce_v(3,4.75,100);
	ptn_to_dut("1-9,11-19");
	ptn_to_dut("2-9",PS_OFF);
	run_pattern(0,ST_ADDR,2);
	result=dut_fimv("2-9",IOHF[1],5,5);
	for (i=0;i<8;i++)
	{
		grpTestValue[i]=result.measval[i];
		pingrp[i]=result.index[i];
	}

	ptn_to_dut("1-9,11-19");
	ptn_to_dut("11-18",PS_OFF);
	run_pattern(0,ST_ADDR,8);
	result=dut_fimv("11-18",IOHF[1],5,5);
	for (i=8;i<16;i++)
	{
		grpTestValue[i]=result.measval[i-8];
		pingrp[i]=result.index[i-8];
	}
	hvdisconn_pmu(3);
	return 0;
}

//测试项名：vol1	测试第 8 项 VOL（IOLF=12mA 时）
MYDLLAPI double test7(double* grpTestValue,short* pingrp)
{
	SYSRESULT result;
	int i;
	hvforce_v(3,4.75,100);
	ptn_to_dut("1-9,11-19");
	ptn_to_dut("2-9",PS_OFF);
	run_pattern(0,ST_ADDR,1);
	result=dut_fimv("2-9",IOLF[0],5,5);
	for (i=0;i<8;i++)
	{
		grpTestValue[i]=result.measval[i];
		pingrp[i]=result.index[i];
	}
```

```
    ptn_to_dut("1-9,11-19");
    ptn_to_dut("11-18",PS_OFF);
    run_pattern(0,ST_ADDR,7);
    result=dut_fimv("11-18",IOLF[0],5,5);
    for (i=8;i<16;i++)
    {
        grpTestValue[i]=result.measval[i-8];
        pingrp[i]=result.index[i-8];
    }
    hvdisconn_pmu(3);
    return 0;
}


//测试项名：vol2      测试第 9 项  VOL（IOLF=12mA 时）
MYDLLAPI double test8(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int i;
    hvforce_v(3,4.75,100);
    ptn_to_dut("1-9，11-19");
    ptn_to_dut("2-9",PS_OFF);
    run_pattern(0,ST_ADDR,3);
    result=dut_fimv("2-9",IOLF[1],5,5);
    for (i=0;i<8;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
    ptn_to_dut("1-9，11-19");
    ptn_to_dut("11-18",PS_OFF);
    run_pattern(0,ST_ADDR,9);
    result=dut_fimv("11-18",IOLF[1],5,5);
    for (i=8;i<16;i++)
    {
        grpTestValue[i]=result.measval[i-8];
        pingrp[i]=result.index[i-8];
    }
    hvdisconn_pmu(3);
    return 0;
}
//测试项名：iozh      测试第 10 项  IOZH
MYDLLAPI double test9(double* grpTestValue,short* pingrp)
{
```

```
    SYSRESULT result;
    int i;
    hvforce_v(3,5.25,250);
    ptn_to_dut("1-9，11-19");
    ptn_to_dut("2-9,11-18",PS_OFF);
    run_pattern(0,ST_ADDR,6);
    result=dut_fvmi("2-9,11-18",VOZHF,0.035, 5);
    for (i=0;i<16;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
    hvdisconn_pmu(3);
    return 0;
}

//测试项名：iozl      测试第 11 项  IOZL
MYDLLAPI double test10(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int i;
    hvforce_v(3,5.25,250);
    ptn_to_dut("1-9，11-19");
    ptn_to_dut("2-9,11-18",PS_OFF);
    run_pattern(0,ST_ADDR,6);
    result=dut_fvmi("2-9,11-18",VOZLF,0.5,5);
    for (i=0;i<16;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
    hvdisconn_pmu(3);
    return 0;
}

//测试项名：ios      测试第 12 项  IOS
MYDLLAPI double test11(double* grpTestValue,short* pingrp)
{
    SYSRESULT result;
    int i;
    hvforce_v(3,5.25,250);
    ptn_to_dut("1-9，11-19");
    ptn_to_dut("2-9",PS_OFF);
    run_pattern(0,ST_ADDR,2);
```

```
result=dut_fvmi("2",VOSF,250, 5);
for ( i=0;i<1;i++)
{
	grpTestValue[i]=result.measval[i];
	pingrp[i]=result.index[i];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("3",VOSF,250, 5);
for ( i=1;i<2;i++)
{
	grpTestValue[i]=result.measval[i-1];
	pingrp[i]=result.index[i-1];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("4",VOSF,250, 5);
for ( i=2;i<3;i++)
{
	grpTestValue[i]=result.measval[i-2];
	pingrp[i]=result.index[i-2];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("5",VOSF,250, 5);
for ( i=3;i<4;i++)
{
	grpTestValue[i]=result.measval[i-3];
	pingrp[i]=result.index[i-3];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("6",VOSF,250, 5);
for ( i=4;i<5;i++)
{
	grpTestValue[i]=result.measval[i-4];
	pingrp[i]=result.index[i-4];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("7",VOSF,250, 5);
for ( i=5;i<6;i++)
{
	grpTestValue[i]=result.measval[i-5];
	pingrp[i]=result.index[i-5];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("8",VOSF,250, 5);
for ( i=6;i<7;i++)
{
	grpTestValue[i]=result.measval[i-6];
	pingrp[i]=result.index[i-6];
}run_pattern(0,ST_ADDR,2);
result=dut_fvmi("9",VOSF,250, 5);
for ( i=7;i<8;i++)
```

```
{
    grpTestValue[i]=result.measval[i-7];
    pingrp[i]=result.index[i-7];
}

ptn_to_dut("1-9，11-19");
ptn_to_dut("11-18",PS_OFF);
run_pattern(0,ST_ADDR,8);
result=dut_fvmi("18",VOSF,250, 5);
for ( i=8;i<9;i++)
{
    grpTestValue[i]=result.measval[i-8];
    pingrp[i]=result.index[i-8];
}
result=dut_fvmi("17",VOSF,250, 5);
for ( i=9;i<10;i++)
{
    grpTestValue[i]=result.measval[i-9];
    pingrp[i]=result.index[i-9];
}
result=dut_fvmi("16",VOSF,250, 5);
for ( i=10;i<11;i++)
{
    grpTestValue[i]=result.measval[i-10];
    pingrp[i]=result.index[i-10];
}
result=dut_fvmi("15",VOSF,250, 5);
for ( i=11;i<12;i++)
{
    grpTestValue[i]=result.measval[i-11];
    pingrp[i]=result.index[i-11];
}
result=dut_fvmi("14",VOSF,250, 5);
for ( i=12;i<13;i++)
{
    grpTestValue[i]=result.measval[i-12];
    pingrp[i]=result.index[i-12];
}
result=dut_fvmi("13",VOSF,250, 5);
for ( i=13;i<14;i++)
{
    grpTestValue[i]=result.measval[i-13];
    pingrp[i]=result.index[i-13];
}
```

```
    result=dut_fvmi("12",VOSF,250, 5);
    for ( i=14;i<15;i++)
    {
        grpTestValue[i]=result.measval[i-14];
        pingrp[i]=result.index[i-14];
    }
    result=dut_fvmi("11",VOSF,250, 5);
    for ( i=15;i<16;i++)
    {
        grpTestValue[i]=result.measval[i-15];
        pingrp[i]=result.index[i-15];
    }
    hvdisconn_pmu(3);
    return 0;
}
//测试项名：icch      测试第 13 项  ICCH
MYDLLAPI double test12(double* grpTestValue,short* pingrp)
{
    double testvalue=0;
    int i=0;
    hvforce_v(3,5.25,250);
    ptn_to_dut("1-9,11-19",PS_OFF);
    ptn_to_dut("1,19,11-18");
    run_pattern(0,ST_ADDR,2);
    testvalue=hvfvmi(3,5.25,100);
    hvdisconn_pmu(3);
    return testvalue;
}
//测试项名：  iccl
MYDLLAPI double test13(double* grpTestValue,short* pingrp)
{
    double testvalue=0;
    int i=0;
    hvforce_v(3,5.25,250);
    ptn_to_dut("1-9,11-19",PS_OFF);
    ptn_to_dut("1,19,11-18");
    run_pattern(0,ST_ADDR,3);
    testvalue=hvfvmi(3,5.25,100);
    hvdisconn_pmu(3);
    return testvalue;
}
//测试项名：  iccZ
MYDLLAPI double test14(double* grpTestValue,short* pingrp)
{
```

```
double testvalue=0;
int i=0;
hvforce_v(3,5.25,250);
ptn_to_dut("1-9,11-19",PS_OFF);
ptn_to_dut("1,19,11-18");
run_pattern(0,ST_ADDR,6);
testvalue=hvfvmi(3,5.25,100);
hvdisconn_pmu(3);
return testvalue;
```

（三）存储器 62256 测试程序示例

```
#include "62256.h"
double    IOHF=-1;              //测量 VOH 时所加的电流
double    IOLF=2.1;             //测量 VOL 时所加的电流

//测试项名：function       测试第一项  功能 function
MYDLLAPI double test0(double* grpTestValue,short* pingrp)
{
      short fail=0;
      hvforce_v(3,5,20);                        //通过 3 号 PMU，设定施加电压值 5V，嵌流 20mA
      ptn_to_dut("1-13,15-27");                                 //连接图形通道到全部输入输出管脚
      fail=run_pattern(0,ST_FAIL);          //运行图形函数，从第 0 行起，仅失效（halt）停止
      hvdisconn_pmu(3);
      return fail;
}

//测试项名：  Voh       测试第二项  VOH
MYDLLAPI double test1(double* grpTestValue,short* pingrp)
{
      SYSRESULT result;
      hvforce_v(3,5,20);                                  //通过 3 号 PMU，设定施加电压值 5V
      ptn_to_dut("1-13,15-27");                           //连接图形通道到所有输入输出管脚
      ptn_to_dut("11-13,15-19",PS_OFF);                   //断开图形通道与全部输出管脚的连接
      run_pattern(0,ST_ADDR,1);                 //运行图形，停止在第 1 行，即输出全为高电平
      result=dut_fimv("11-13,15-19",IOHF,5,5);            //对全部输出管脚加流（IOHF）测压
      for (int i=0;i<10;i++)
      {
            grpTestValue[i]=result.measval[i];
            pingrp[i]=result.index[i];
      }
       hvdisconn_pmu(3);                                  //断开该路 PMU
       return 0;
}

//测试项名：  Vol
MYDLLAPI double test2(double* grpTestValue,short* pingrp)
{
      SYSRESULT result;
      hvforce_v(3,5,20);
      ptn_to_dut("1-13,15-27");
      ptn_to_dut("11-13,15-19",PS_OFF);
      run_pattern(0,ST_ADDR,3);                 //运行图形，停止在第 3 行，即输出全为低电平
      result=dut_fimv("11-13,15-19",IOLF,5,5);            //对全部输出管脚加流（IOLF）测压
```

```
    for (int i=0;i<8;i++)
    {
        grpTestValue[i]=result.measval[i];
        pingrp[i]=result.index[i];
    }
     hvdisconn_pmu(3);
     return 0;
}
```

（四） 62256 存储器测试图形说明（棋盘格）

在存储器阵列中所有单元中交替写入 1 和 0，然后依次读出，然后再将原来位置取反（1 位置写 0，0 位置写 1）