LAUTERBACH
DEVELOPMENT TOOLS

实践脚本语言用户指南

版本09.2021

手册

实践脚本语言用户指南

实践脚本语言用户指南

版本04-2021年11月

为什么使用实践脚本

使用实践脚本（*。在TRACE32将帮助你：

在调试器启动时立即执行命令

根据您的项目需求自定义TRACE32PowerView用户界面

使用目标板的设置设置调试器

标准化重复的和复杂的操作

初始化目标(e。g.要加载应用程序的内存）

加载应用程序和/或符号

添加您自己的功能，并扩展可用的功能

通过自动化加速调试

与其他用户共享调试器方法，并使他们能够更有效地工作

使调试操作具有可重复性，并可用于验证目的和回归测试

相关文件

- **“培训脚本语言实践”(training_practice。**描述了如何运行和创建实践脚本文件(*。cmm).
- **“实践脚本语言参考指南”(practice_ref。实践参考卡(https://www。**劳特巴赫
- 。com/referencecards.html)
- **视频教程(https://www。**劳特巴赫。网站/tut_practice。html)

实践脚本结构

功能

实践是一种面向线的测试语言，可用于解决数字测量工程中的所有常见问题。实践-ii是这种测试语言的增强版本，首次于1984年为电路内模拟器开发。

该测试语言允许交互式脚本开发，并有可能快速删除错误和立即执行脚本。可以随时停止和重新启动实践测试脚本的执行。

实践中包含了一个用于处理脚本变量和命令参数的非常强大的概念。这个宏概念允许在命令中的任何点替换参数。由于实践变量只能作为实践宏出现，因此排除了目标程序名称之间的冲突。

变量与实践宏之间的区别

实践宏是基于一个简单的、类似于C预处理器宏的文本替换机制。然而，与C预处理器宏不同，实践宏可以通过简单地分配一个新值来在其生命周期内更改其内容。

每次实践解释器遇到一个宏时，它都会被替换为相应的字符序列。只有在执行所有文本替换之后，才会解释结果行(就像C编译器只处理完全预处理的文本一样)。

实践宏是使用私有、本地或全局的命令来声明的。

实践宏的可见性与其他脚本语言有明显的不同（除非使用私有命令声明）：当它们处于活动时，可以从所有随后执行的代码中访问它们。g.在

子例程(GOSUB……返回)

子脚本(DO...ENDDO)

子块（IF……、RePeaT、时等）

注：	实践中不知道具有相应类型的变量的概念，如C中的uint32或uint8。

实践脚本元素

实践脚本由标签、命令和注释组成：

样例._____评论从；

//example._____以//开头的注释

开始._____标签

步骤._____命令

要启动._____命令和标签

B::._____用来更改默认设备的命令

B::数据。丢弃._____命令前面有一个设备选择器

标签

标签总是以第一列开始，后面总是有一个冒号。标签区分大小写。

意见

注释以分号开头，或者是两个向前的斜杠。
对Var的内联评论。*命令必须以两个向前的斜杠开始。

```
Var .设置func7（1.5、2.5）//在目标中执行一个函数
```

线路延续字符

要在下一行中继续字符串，将使用反斜杠表示实践脚本中的行延续（*。cmm).反斜杠后不允许有空格。如果在注释行的末尾使用了行延续字符，那么下一行也会被解释为注释行。

```
对话框。好的，“请先打开TRACE32调试器，然后打开”+\
“然后打开目标电路板。”
```

脚本流

有几个命令允许控制脚本流。脚本可以分为几个模块。一个模块中的子例程由GOSUB命令调用，另一个模块由DO命令调用。

STOP	Stop temporarily
END	Terminate script and clear stack
CONTinue	Continue with script execution
DO	Call script module
ENDDO	Terminate script module
RUN	Clear PRACTICE stack and call module
GOSUB	Call subroutine
RETURN	Return from subroutine
GOTO	Branch within module
JUMPTO	Branch to other module

条件脚本流

有条件脚本的执行可由以下几个命令完成：

IF	Conditional block execution
ELSE	Block that is only compiled when the IF condition is false
WHILE	Conditional script loop
RePeaT	Repetitive script loop
ON	Event-controlled PRACTICE script execution
GLOBALON	Global event-controlled PRACTICE script execution
WAIT	Wait for event or delay time

```
如果os。文件（数据.tst）
        打印“文件存在”
其他
        打印“文件不存在”

而注册(pc)==0x1000                    ；步骤，直到pc=1000H
        步骤

RePeaT100。步骤                       ；步骤100次

RePeaT0。步骤                         ；一步无止境

ON错误GOTO错误退欧
```

有关逻辑操作的详细信息，请参见“PowerView用户指南”(ide_user.pdf)中的“操作员”章节。

脚本 嵌套

实践脚本可以分层嵌套。第二个脚本可以作为初始脚本中的子例程被调用。这个子例程可能会将第三个脚本调用为一个子例程。这允许结构化的模块化脚本开发。

```
；包含两个脚本调用的脚本

打印“开始”
DO模块1                               ；执行脚本模块1
DO模块2                               ；执行脚本模块2
                                      ；文件扩展名（*。cmm)可以
恩多                                  省略
```

块结构

可以将几个实践命令组合成一个块。块是命令的集合，它们总是同时执行。块通常需要IF、while或RePeaT语句。然而，它们可以在任何地方实现，以标记连接块。方块用圆括号标记。

你可以跳出一个街区，但不能跳进一个街区。

```
；块嵌套
开始
        IF&abc
        (
                打印“函数1”
                DOfunc1
        )
        其他
        (
                打印“功能2”
                DOfunc2
                                                        如果&xyz要开始；跳出街区到“开始”的标签
        )
恩多
```

实践宏

实践宏是最大大小为4kb的字符序列。字符序列将在使用它的上下文中进行解释。对于一个命令，一个实践宏可以被解释为e。g.作为一个数字、布尔表达式、参数，或者仅仅作为一个字符串。练习宏也可以扩展到完成命令。

实践宏是由一个分配来生成的。在脚本文件行中找到的实践宏被它们包含的文本取代(除了特殊命令，例如。**条目）。**如果需要，可以将这些宏放入括号中。如果解析的值仍然包含宏名称，则分配的双超名和形式（‘&&’）将强制进行递归宏解析。可以在交互式命令行内定义和修改宏。

在实践中，宏名称总是以符号和符号（“&”）开头，后面跟着一系列字母（az、a-Z）、数字（0-9）和下划线符号（“_”）。&符号后的第一个字符不能是数字。宏名称区分大小写，因此&a不同于&a。

宏扩展不会在TRACE32命令行内发生！

```
e .g .印刷和杂志或

            数据。列表&my_startaddress
```

宏扩展只在实践脚本中工作(*。cmm):

正常：递归：	*&<macroname>=<expression>* *&&<macroname>=<expression>*

```
&int=1
&int=&int+1                                    ; 增量电流值为&int
&text=“这是一个测试”
&command="Data .转储寄存器(PC)
&float=1 .4e13
&range="0x1000--0x1fff"
&address=func5
&addressrange=P:0x1234 ..0x5555
&boolean=SYStem.在上面

打印和输入
打印“&int”                                    ; 更换后：0x2
打印“&(int)nd”                                 ; 替换后：“第二”
恩多
```

实践宏可以声明为全局、本地或私有宏。

本地的	声明本地宏声明全局宏
全球	声明私有宏
私人的	

全局宏

使用全局声明的实践宏在每个脚本级别上都可访问，并且具有无限的生命周期。

本地宏

使用本地声明的实践宏在其生命周期内所有随后执行的代码中都可见（除非被以后的宏声明隐藏）。特别是，它们可以在：

是的子例程（GOSUB返回）...

是子脚本(doenddo)...

是子块（IF……、RePeaT、时等）

私有宏

使用私有属性声明的实践宏存在于声明块中，并在块结束时被擦除。它们仅在以下内容中可见：

是的，声明的块和所有的子块(e。g. **如果……RePeaT……等等)**

无子例程（go子返回）...

无子脚本(doenddo)...

注意：如果将一个值分配给了一个在实践堆栈中还不存在的宏
或者是一个不可访问的私有宏，那么该宏将被隐式地创建为一个本地宏。

切换 实践 宏指令 膨胀 上 或 从…落下

您可以在以下嵌入式脚本块中切换实践宏展开的打开或关闭：

SBUR对话框。 *查看嵌入在实践脚本文件中的块(*。*，如下例所示。

Subar菜单。嵌入在实践脚本文件中的重新编程块(*。cmm)

练习脚本块，并在对话框文件中嵌入(*。dlg)

练习脚本块嵌入到菜单文件(*。人

嵌入式块由打开圆括号“”、“&”、“（&+”或“&-”分隔，并通过关闭圆括号“”）关闭。

上	使用(&或(&+作为打开块分隔符来打开此块及其子块的宏扩展。
从…落下	使用（&-作为打开块分隔符来关闭此块及其子块的宏扩展。

切换宏展开的打开是很有用的，例如，如果你想使一个按钮的文本是可配置的。

示例：要尝试，只需将此脚本复制到test.cmm中，然后在TRACE32中运行它(请参见“Howto”)。...

```
本地&btn

&btn=“按钮”“宏是扩展时，脚本加载“”\
“打印”“”“它工作！“”“”“”“

对话框。视图
(&+；宏观扩展的宏观&btn需要
     ；在此对话框中已打开。视图块，它被嵌入在
     ；一个练习脚本
     POS 0.0.35.1.
     &btn

     按钮“硬编码按钮文本”
     (&-；在此子块中，宏扩展将被关闭
          &btn=“这里没有扩展”
          打印“btn”
     )
)
使停止
```

参数 通过

参数可以传递给具有参数列表的子例程。使用子例程中的输入命令，将参数分配给本地实践宏。

子程序也可以通过参数进行交互式地调用。参数可以通过命令DO、ENDDO、RUN、GOSUB、返回来传递。它们也可以是调用命令扩展或用参数调用主机上的驱动程序程序的结果。

表达式中的运算符前后的空格被解释为连续参数的分隔符。

条目	参数 经过的

系统。在上面

```
GOSUBmyTEST0x0--0xfff
```

恩多

```
; 通过调用子例程myTEST，对地址范围进行内存测试
; 0x0-0xfff可以执行
myTEST:
    入口和范围
    数据。测试和范围
返回
```

输入和输出

有几个输入和输出命令允许与用户进行交互。输入通常是通过一个区域窗口来完成的。所有的打印操作都显示在TRACE32消息行上。默认情况下，会显示AREA窗口A000。输入和输出可以被重新路由到另一个区域窗口。

打印	打印到屏幕
哔哔声	刺激声音发生器窗口基于输
进入	入字符输入
密钥	

```
打印“个人电脑的地址是”        “寄存器(pc)        ；打印消息
哔哔声                                          ；端声信号
密钥                                            ；等待按键
INKEY &char                                     ；等待按键
IF&char==“A”
      GOSUB func_a
IF&char==“B”
       GOSUB func_b
…
地区创建IO区域                                  ；创建一个窗口，名为
                                                ；IO区域
地区选择IO区域                                  ；选择此窗口，为
                                                ；练习i/o
区域。视图IO区域                                ；打开这个窗口
打印“设置PC值”                                  ；打印文本
输入&pc                                         ；得到价值
注册设置电脑和电脑                              ；设置寄存器PC
WINClear顶部                                    ；删除I/O窗口
地区重新设置                                    ；重置区域系统
恩多
```

屏幕更新可以由屏幕命令来控制。

屏幕。陈列	立即更新屏幕
屏幕。无论如何	每个命令行后更新屏幕打印命令上的更新
屏幕。上	屏幕
屏幕。从…落下	只要脚本正在运行，就不要更新屏幕
屏幕。等待	停止练习脚本执行，直到处理完要在窗口中显示的数据。

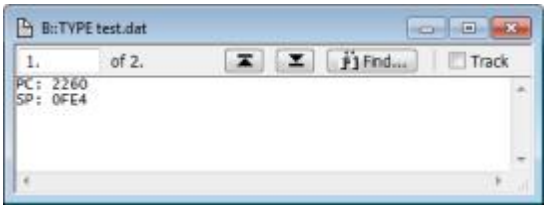
文件操作

测试数据可以写入文件和从文件中读取。在访问文件之前，必须先打开它们。

打开	打开文件
关闭	关闭文件
读取	从文件中读取数据
写	写入数据到文件中
附加	附加数据到文件中

示例1：创建一个名为test.dat的新文件，并将注册信息写入新创建的文件。然后将结果显示在一个类型窗口中。

```
打开#1测试。dat /Create
写入#1“PC：”寄存器(PC)
写入#1“SP：”寄存器(SP)
关闭#1
TYPE测试.dat
恩多
```



示例2：测试。该文件已被打开以供读取。从该文件中读取两行，并存储在两个实践宏中，然后将其打印到TRACE32消息行中。

```
打开#1测试。dat /Read
读取文件的第1%行和第1行
读取文件的第1%行中的第2行
关闭#1
打印“&pc”&sp”
恩多
```

自动 启动 脚本

安装TRACE32软件后，脚本会自动启动。cmm被复制到TRACE32系统目录中。**自动启动。**cmm总是在TRACE32启动后自动执行。它提供了由劳特巴赫定义的各种便利性特性。

建议不要更改自动启动程序。嗯，因为来自劳特巴赫的每个软件更新都将恢复文件的自动启动。cmm到其默认内容。

自动启动。如果存在以下脚本，cmm将调用它们：

- **~~/系统设置。**cmm，其中~~表示TRACE32系统目录。

  建议在这里添加额外的TRACE32设置，以供进行TRACE32安装的所有用户使用。典型的额外设置是菜单/工具栏扩展名或用户定义的对话框。

SUBUAD/用户设置。其中，UAD表示用户特定的应用程序数据目录。TRACE32功能版本。**环境环境(UAD)返回此目录的路径。**

用户可以将所有他们喜欢的额外的TRACE32设置添加到用户设置中。cmm脚本。典型的额外设置是设置命令组和个人菜单/工具栏扩展的所有设置。

Subar./工作设置。嗯，在领先的地方”。”表示TRACE32启动处的工作目录。

> 注意：如果您不使用命令行选项-s<startup_script>和没有
> **文件自动启动。**或者，TRACE32将回到遗留模式并执行脚本t32。从工作目录或从TRACE32系统目录，如果是t32。在工作目录中不存在Cmm。

了解所有其他启动脚本

如果您有一个设置调试环境的脚本，以及如果这个脚本应该在自动启动后自动执行。cmm完成后，您可以将此脚本指定为TRACE32可执行文件的参数。

```
c:\t32\t32arm.exe-sg：\和\arm\start_up。cmm
```

可以将参数直接传递给启动脚本。

```
c:\t32\t32arm.exe-sg：\和\arm\start_up。cmm param1 param2 param3
```

参数可以通过启动脚本读取，如第12页所述。

命令行选项会抑制自动启动的执行。cmm和启动脚本。 其他的

日志记录 那 呼叫 层次结构 的 实践 脚本

实践脚本（*.cmm）可以自动或手动记录。在任何一种情况下，日志机制都是基于日志的。DO命令。

在TRACE32启动期间，练习脚本调用总是自动记录。日志文件的内容被输出到TRACE32的临时目录中的一个自动启动日志文件中。在TRACE32的每次启动时，都将覆盖以前的自动启动日志文件，并生成一个新的日志文件。当前的自动启动日志可以通过TRACE32中的文件菜单进行访问。

此外，您还可以在TRACE32启动后的任何时间手动记录练习脚本调用。在启动日志时，您可以选择文件夹和文件名。要显示日志文件，请使用TRACE32命令行上的TYPE或EDIT命令。

若要启用日志记录，请使用以下选项之一：

- **autostart.嗯**：在TRACE32启动时，自动启动。自动调用Cmm，并且该日志。在自动启动中的DO命令。cmm生成自动启动日志文件。

•--t32-日志自动启动：这个命令行选项开始日志。在内部生成一个自动启动日志文件。

-此选项只需要(a)，如果您没有自动启动。如果脚本块与日志中的cmm或(b)。DO命令已从自动启动中删除。cmm.

-关于命令行选项的描述和——t32-log自动启动的示例，请参考TRACE32安装指南，第61页中的“启动TRACE32的命令行参数”。pdf).

提示：要在启动时显式地禁用所有练习脚本调用，请使用——t32-safestart。

TRACE32命令行：使用日志。DO<file>命令以生成日志文件。

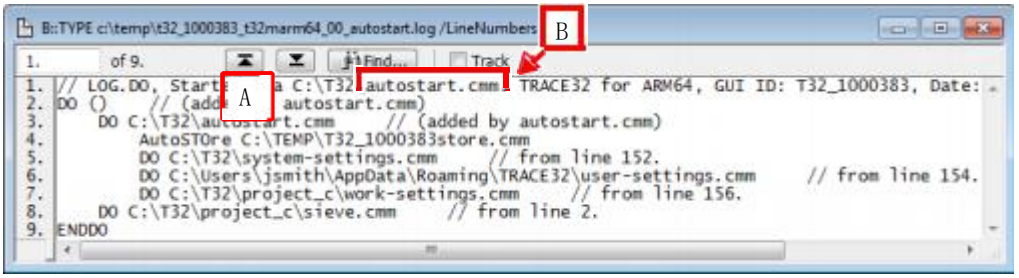
要访问TRACE32中的自动启动日志文件：

1. 通过T32启动启动TRACE32。

自动启动。cmm会自动生成自动启动日志文件。

2. **在启动>查看自动启动日志上选择文件菜单>自动脚本。**

该文件将在“类型”窗口中打开。屏幕截图显示了一个自动启动日志文件的示例：



A自动启动日志的文件名约定如下所述。

B日志文件头告诉您是如何生成自动启动日志的。

自动启动日志文件的文件名约定：~~~/<id>_t32m<arch>_<xx>_autostart。记录

~~	TRACE32的临时目录的路径前缀。参见操作系统。**当前时间目录()。**
<id>	已启动的PowerViewGUI的ID。参见操作系统。**身份证**
t32m<arch>	PowerView可执行文件的名称（没有文件扩展名），e。g."t32marm"
<xx>	PowerView可执行文件的实例号。

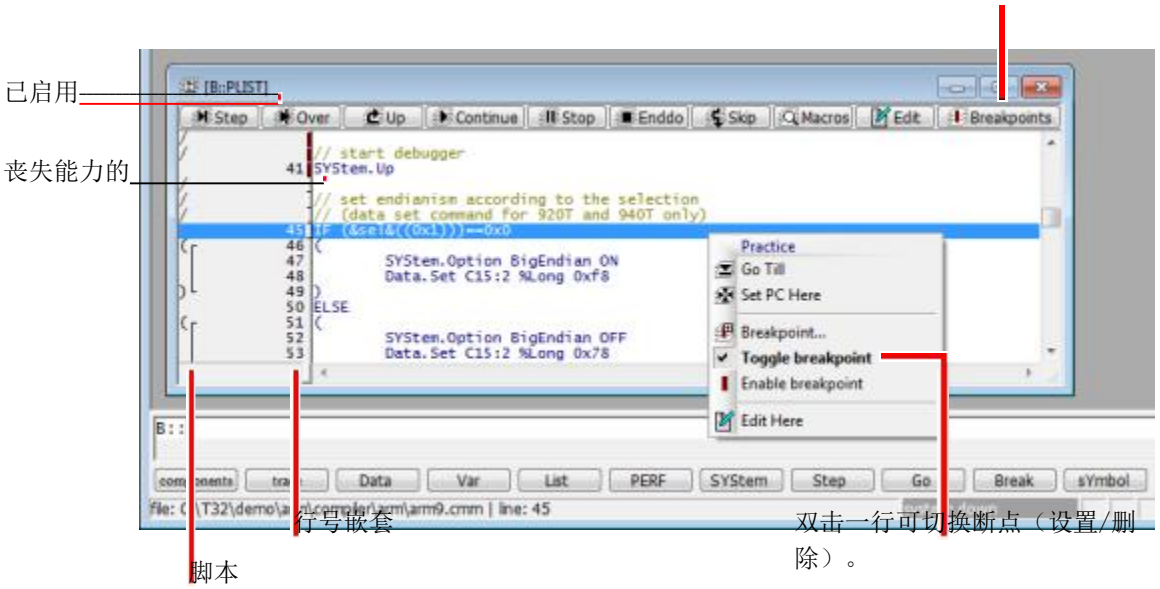
调试 实践 脚本

TRACE32支持针对实践脚本的广泛调试特性。PEDIT命令允许您创建和编辑实践脚本。两个基本窗口显示了脚本、内部堆栈和实践宏。

多段线编辑	编辑实践脚本
参数表	列表实践脚本
pmacro。清单	列出实践脚本嵌套和实践宏

在PLIST窗口中，您可以设置无限数量的程序断点来调试实践脚本。双击整行将设置断点；再次双击同一行将删除断点。也就是说，您可以通过双击某一行来切换断点。
或者，右键单击某一行，然后从弹出式菜单中选择切换断点。

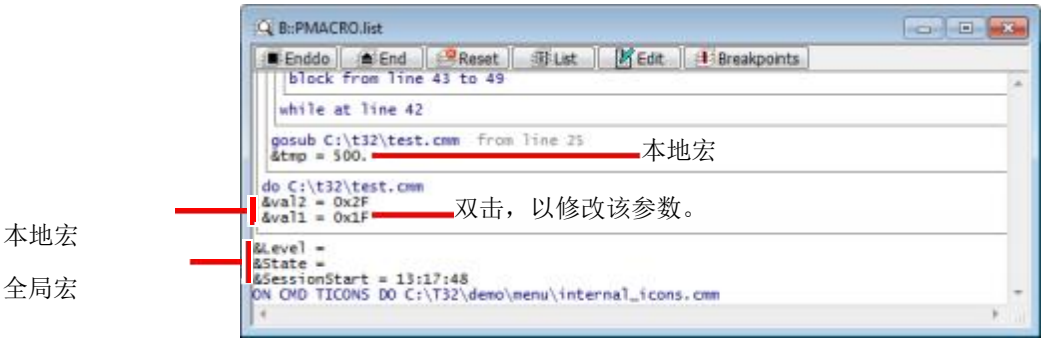
列出了练习断点
在PBREAK。列表窗口。



启用的断点用一个小的红色条标记，禁用的断点在PLIST窗口中用一个小的灰色条标记。

pbreak。设置	在实践脚本中设置断点
pbreak。删除	删除断点
pbreak。清单	显示断点列表
pbreak。启用	启用断点
pbreak。DISable	禁用断点

PMACRO。列表窗口显示脚本嵌套、本地和全局实践宏以及ON和全局化定义：



双击一个练习宏。&val11)将其插入到TRACE32命令行中，在那里您可以修改练习宏的参数。

可以使用PSTEP命令逐步执行脚本。TRACE32主工具栏中的“停止”按钮将停止任何正在运行的练习脚本。

PSTEP <script>	以单步模式启动要调试的脚本

```
PBREAK .设置4。测试.cmm          置断点
DO测试。cmm                      ；一直运行到第4行
普斯特尔
普斯特尔                         单步执行
```

附录A

本附录：

以下是一些如何运行您从pdf手册中复制的演示脚本的建议：

-动态地创建练习脚本。

-为一个测试创建一个永久的工具栏按钮。cmm更多。

-将复制的脚本作为嵌入式脚本运行。

从TRACE32~~\demo文件夹中运行一个演示脚本[更多]。

怎样 向 运行 演示物 脚本 已复制 从… 那 弹头信管 手册

pdf手册提供了一些可读到运行的演示脚本（除了实践脚本片段之外）。尝试这些演示脚本的一种方法是创建一个测试。在PEDIT窗口中的cmm中，复制并粘贴演示脚本到*中。计算机和控制文件，然后执行它。

有关如何操作的信息，请参见下面的具体步骤。

创建 a 实践 脚本 在…上 那 飞

1. 在TRACE32命令行中，键入：

```
PEDIT ~~~/test .cmm
```

实践脚本编辑器PEDIT打开，显示test.cmm。路径前缀~~~将扩展到TRACE32的临时目录。

2. 将已从手册中复制的演示脚本粘贴到test.cmm中。

3. 在演示脚本的末尾追加ENDDO（如果演示脚本中缺少ENDDO）。

4. 单击“保存”。

5. 要逐行完成，请在PEDIT窗口中单击“调试”，然后单击PLIST窗口中的“步骤”。

6. 要执行实践脚本文件，请单击PEDIT窗口中的“执行”。

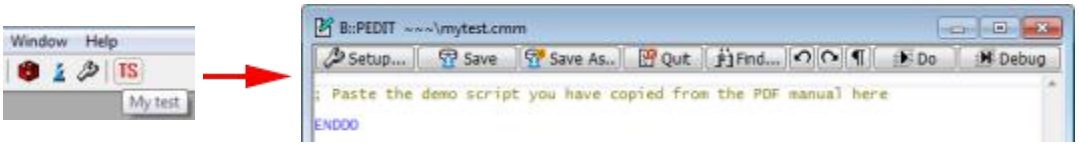
7. 可选地，单击“宏”以查看实践堆栈框架。

1. 将以下设置添加到系统设置中。cmm（如果该文件还不存在，则创建该文件）：

```
菜单。添加工具“MyTest”“TS，R”“PEDIT~~~/mytest”。cmm″
```

2. 重新启动TRACE32。

3. 单击新的工具栏按钮以打开实践脚本编辑器。



将演示脚本作为嵌入式脚本运行

要逐步复制以下步骤，我们建议您复制此演示脚本：

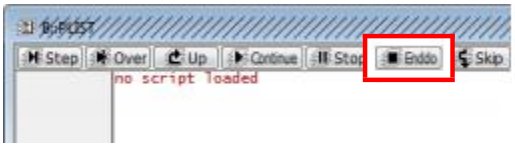
```
；为TRACE32的虚拟内存设置一个测试模式
数据。设置VM：0--0x4f%字节1 0 0 0
数据。转储VM：0x0；打开数据。转储窗口
；将TRACE32虚拟内存的内容可视化为一个图形
数据。DRAWFFT %Decimal .字节VM：0++0x4f2.0 512。
```

要以嵌入式脚本的形式运行演示脚本：

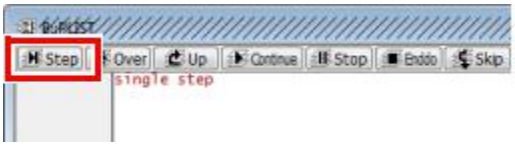
1. 从pdf手册中复制演示脚本。

2. 在TRACE32命令行中，键入PL工ST以打开一个PLIST窗口。

3. 单击Enddo，直到PLIST窗口显示没有加载任何脚本。

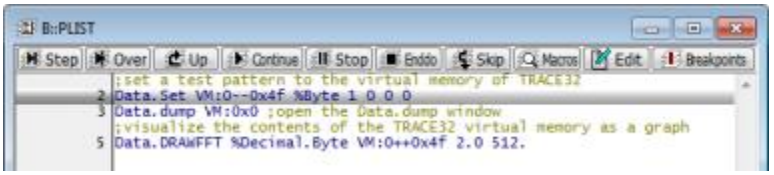


4. 单击“步骤”。PLIST窗口将显示单步操作。



5. 将演示脚本粘贴到TRACE32命令行中。

结果：



6. 请执行以下操作之一：

-单击“继续”以运行嵌入式脚本。

-单击此步骤，一步一步地逐行完成任务。

TRACE32演示文件夹中的演示脚本

使用TRACE32中的搜索脚本窗口，您可以在本地的TRACE32演示文件夹中搜索实践演示脚本(*.cmm)，以及劳特巴赫网站上的最新脚本。

“搜索脚本”窗口将显示您所选定的每个脚本的简要说明。描述从脚本头中的元数据中提取。双击脚本可以在运行脚本之前预览脚本的源代码。

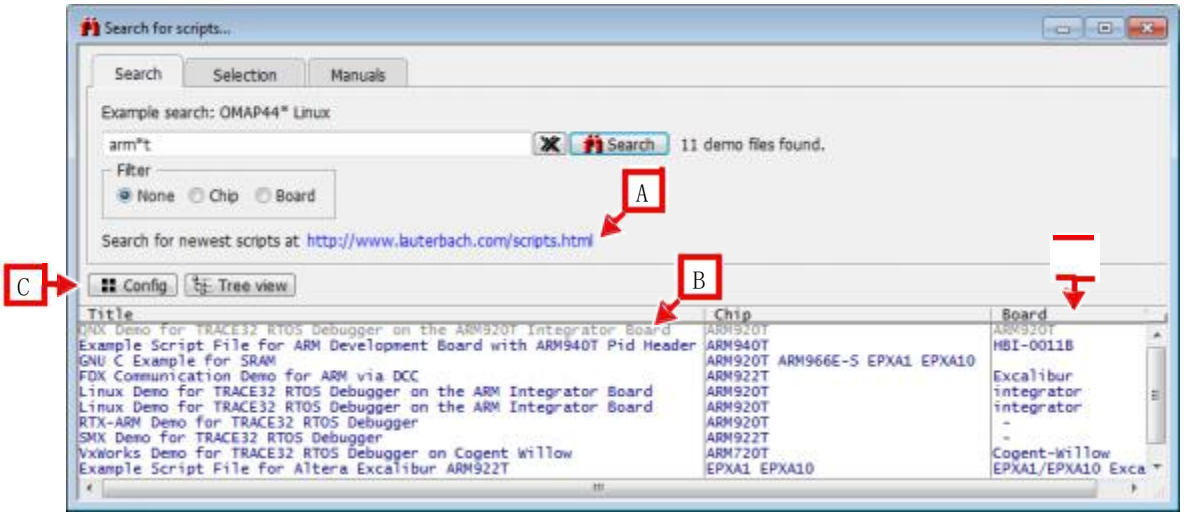
要在本地的TRACE32演示文件夹中搜索演示脚本：

1. 选择文件菜单>搜索脚本，以打开相同名称的窗口，或在TRACE32命令行类型：欢迎。**脚本**

2. **在示例搜索下，输入您要寻找的内容，e。**g.一个芯片或电路板的名称。系统支持通配符（*）。

3. 可选择将过滤器设置为“芯片”或“板”。

4. **单击“搜索”。**
   满足搜索条件的演示脚本列在窗口的底部。



点击这个超链接可以继续你在www上的搜索。**劳特巴赫。com/scripts.html.**

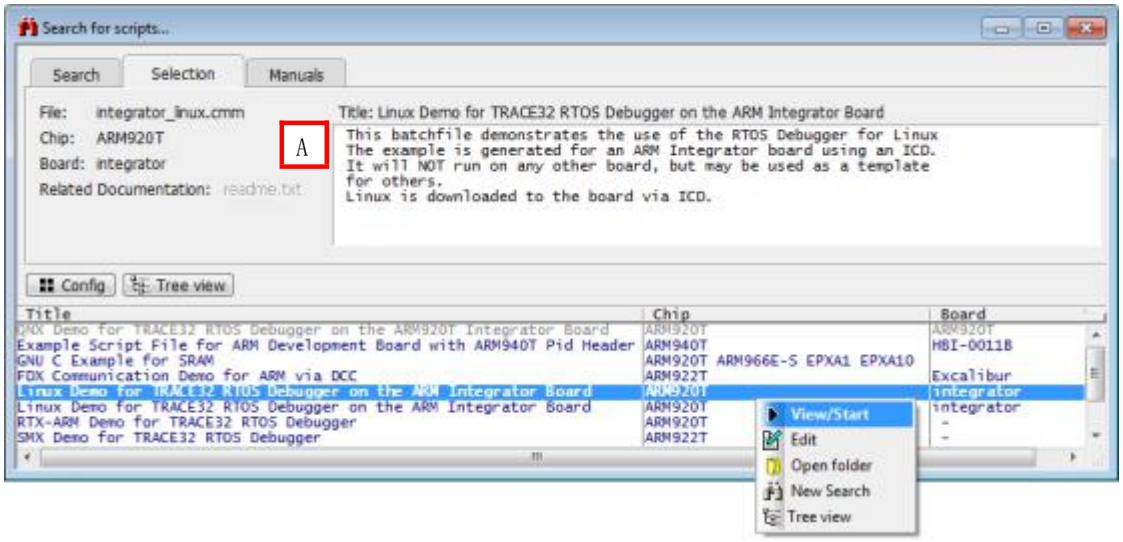
B实践脚本模板用灰色突出显示，准备运行的脚本用中突出显示蓝色

C允许您向演示脚本搜索中添加您自己的搜索路径。

D单击列标题可更改排序顺序。

要预览并运行演示脚本，请执行以下操作：

1. 单击一个脚本以自动切换到“选择”选项卡，在那里您可以查看所选脚本[A].的简要说明



2. 要在PSTEP窗口中查看脚本的源代码，请双击该脚本或右键单击，然后从弹出菜单中单击“查看/开始”。

3. **若要运行该脚本，请在PSTEP窗口中单击“继续”。**

弹出菜单-选项	描述
查看/开始	在PSTEP窗口中打开该脚本。
编辑	在PEDIT窗口中打开该脚本。
打开文件夹	打开文件资源管理器并选择该脚本文件。
新搜索	切换到“搜索”选项卡，您可以开始新的搜索。
树形视图	显示文件夹和演示脚本的类似文件资源管理器的树状视图。