WP 34S
用户手册

Walter Bonin 著
R. P. N. 翻译组 译

原著	Walter Bonin	
策划	Roy Wan	Tselung Soong
文本翻译	Alias Qli	Beta Vulgaris
	Chicheng Chen	Chienway Tsai
	Geesin	葛炳仑
	滚来滚去的巧胖	刘傲
	Null	N77012
	Roy Wan	Rudal
	Vikill	我的8086电脑
	朱嘉诚	
统稿	Roy Wan	柜里猫
校对	Andy Lithia	Alias Qli
	Chienway Tsai	柜里猫
	MBX008	Roy Wan
	我的8086电脑	温明翰
	武平	无情铁手
技术顾问	Andy Lithia	
封面立绘	无疆	

译者序

WP 34S 是目前最强大的函数计算器之一，它以 *HP-42S* 这款强大而又经典的 RPN 函数计算器为样本进行开发，并且集合了惠普历代计算器的精华并进行了扩充，其功能之多，精度之高，在计算器中罕有。该机自发布以来，在国外爱好者众多，广受好评。

WP 34S 的出现本应是计算器爱好者的一大幸事，但是由于国内缺乏 RPN 用户基础，以及 *WP 34S* 与国内普遍使用的计算器使用方式差别较大，使得部分用户望而却步，甚至出现了一些误解，认为操作界面不友好、反人类等。以上现象，除了 RPN 本身有一定的学习成本之外，*WP 34S* 缺少中文用户手册，使得英文基础较差的用户难于上手也是一个重要原因。

为此，一群自发组成的计算器爱好者，利用业余时间开始了 *WP 34S* 用户手册的翻译。我们旨在将这一强大的计算器介绍给更多国内用户，使这款计算器得到客观正确的评价，以便让广大工程师、科研工作者、学生朋友和计算器爱好者体验到 RPN 函数计算器的魅力。我们将我们的翻译团队命名为 Real Perfect Number（R. P. N.），以致敬 RPN 这一古老而又独特的计算器输入方式。

本手册内容以 3.1 英文版本为主，参考 3.3 英文版本进行了局部更正，适用于 3.3 版本的固件。

本手册可以自由分享，但是请不要篡改，删减本手册内容。请将本手册从封面到末页作为一个整体，保持它的完整性。请尊重我们的劳动成果。

介于译者水平和条件所限，翻译或表述错误在所难免。我们恳切地希望广大使用者能够提出意见，以便继续修订，不断提高质量。

若有问题或欲了解更多信息，可在以下 QQ 群交流：

计算器爱好者交流群：　　435067924

RPN calculator club：　　812224709

R. P. N. 翻译组

2020 年 6 月

Translators' Preface

It is our honor to translate the manual of WP 34S. Misunderstood as outdated technology, never in the past has RPN gained much attention among Chinese users. It is a pity that the value of RPN has only been recognized by a few 12C users even to this day. But we believe WP 34S, the ultimate reincarnation reminiscent of those famous models, shall receive the attention due to the apex of RPN keystroke programming. Such is our primary goal.

However, all of our group members are not WP 34S users. In fact, a large percentage of us have never heard of it before. So, I think we are here to serve a higher purpose: promoting the wide use of all kinds of calculators and wishing them a brighter future.

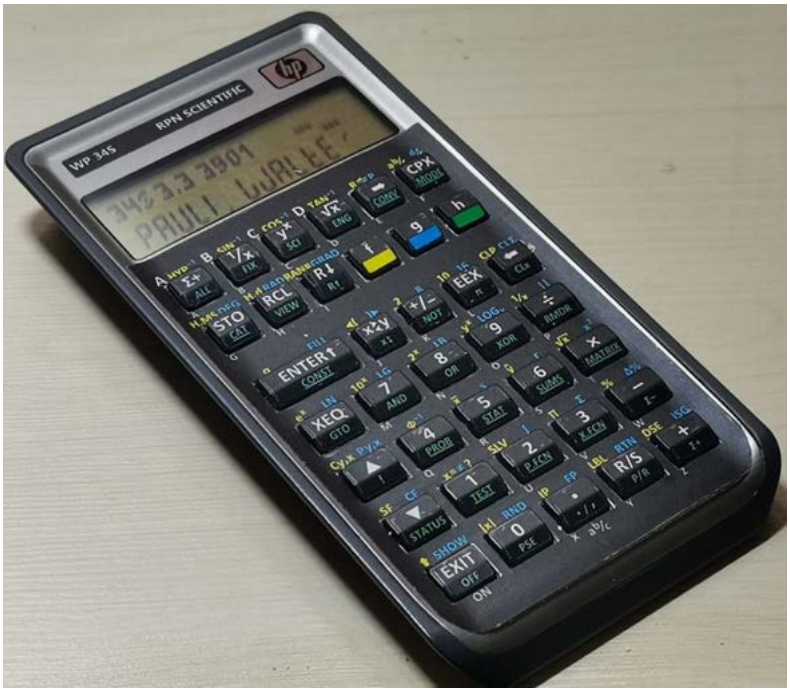
It seems that Chinese people prefer to equip them with the ability to do calculations even when they know calculators are easy at hand. However, calculators, presented to us by the great power of modern technology, are certainly a solution much more efficient to the problems and will definitely take our place to do the calculating work. When the day finally comes, we can say, with pride, that though little, we've made contributions to promoting the popularization of calculators.

The Chinese version of this manual is primarily based on English version 3.1, with additional content from version 3.3 to adapt it to firmware 3.3.

R. P. N. Translation Team

June 2020

WP 34S用户手册



这是 *WP 34S* 的用户手册。*WP 34S* 是一个自由软件。您可以根据自由软件基金会发布的 GNU 通用公共许可证第三版或之后的版本，来重新分发或修改 WP 34S。

我们希望 *WP 34S* 能够物尽其用，但是不提供任何保证，包括任何可销性和特殊用途适应性的保证。请在 *http://www.gnu.org/licenses/*查看 GNU 通用公共许可证以了解详情。

这个手册适用于 3778 版本固件，如果我们（即开发者）修改了 *WP 34S*，这个手册可能会在没有事先通知的情况下改变。我们保留在任何时候这么做的权利。但是，我们开发 *WP 34S* 的初衷会保持不变。要获取最新的更新，请访问：

http://sourceforge.net/p/wp34s/code/HEAD/tree/。

如果不是因为我们对 Classics，Woodstocks，Spices，Nuts，Voyagers，以及 Pioneers 系列计算器的热爱，*WP 34S* 无法达到现在的完成度。因此，我们想引用 HP 在 1980 年前印刷在袖珍计算器手册上的，至今仍不褪色的一句话：

“只有当我们在售前售后为顾客提供了优质的产品，满足真正的需求，持久的提供价值并且提供广泛而有用的服务，我们公司的成功和繁荣才能得到保证。”

企业目标声明
惠普

如果您现在还没有自己的 *WP 34S* 计算器：

WP 34S 运行在 *HP-20b* 商业顾问财务计算器或 *HP-30b* 商务专业计算器上。它们都是袖珍金融计算器。如果您的桌上还有个未修改的以上计算器，请前往附录 A，那里将告诉您怎样自己动手将它转换为高完成度的 *WP 34S*。另一方面，您也可以在互联网上购买一个 *WP 34S* 成品，如：
http://commerce.hpcalc.org/34s.php 或者
http://www.thecalculatorstore.com/epages/eb9376.sf/en_GB/?ObjectPath=/Shops/eb9376/Products/%22wp34s%20Pack%22。在国内，也可以在网上找到 *WP 34S* 刷机服务和贴纸。

第一种方式（自己动手做）将会花费您一些时间，第二种方式则会花费一些钱。如果您选择在我们提到的网站上购买 *WP 34S*，我们（开发者）会得到销售收入中微薄的一部分，以支持我们在 *WP 34S* 项目中无偿的工作。无论使用哪种方法，都由您自己选择。

此外，还有三种可选的硬件改装。它们需要一些精细的焊接，切割，粘合并在金融计算器的塑料外壳上钻一个小孔。这些操作不适合那些手指还不够灵巧，眼神还不太犀利的年轻电子工程师，但可能适合另一些人来做。查看附录 H 来决定是自己制作，还是找别人来帮您。

WP 34S 的软件包内含有一个完整的模拟器，购买机器前您可以在电脑上测试它的功能。在获得机器之后，您也可以在向 *WP 34S* 传输程序之前用它来测试，它们的功能是完全相同的。

接下来，我们认为您已经做好了改造，并且正拿着 *WP 34S*。

目　录

1. 欢迎！

亲爱的用户，现在您已经拿到了属于自己的 *WP 34S*。它使用了 *HP-20b* 或 *HP-30b* 的硬件，因而可以利用其极佳的处理器性能。如果是 *HP-30b*，您还可以使用著名的转轴式按键[1]，体验到复古惠普计算器的按键手感。

而 *WP 34S* 的固件和用户交互界面则是经过我们反复打磨，全新设计的，在可用的内存空间内尽可能塞进各种功能，并经过了反复测试，只为给您一台从未有过的快速而又小巧的科学计算器——可编程，可轻松放入衬衣口袋，还有 RPN[2]。

WP 34S 的功能基于 1988 年发售的 *HP-42S*，这一迄今为止最强大的惠普可编程 RPN 科学计算器[3]。我们还增加了来自 *HP-16C* 的计算机科学需要用到的功能、*HP-32SII* 的分数模式，以及类似于 *HP-21S* 上的概率分布计算功能。

我们还添加了更多关于计算、统计、物理、机械、编程以及输入输出的功能，例如：

- 欧拉 Beta 函数和黎曼 Zeta 函数、伯努利数列和斐波那契数列、朗伯 W 函数、误差函数，以及切比雪夫、埃尔米特，拉盖尔，勒让德正交多项式。
- 各种统计分布和其反函数，如泊松分布、二项分布、几何分布、柯西-洛伦兹分布、指数分布、逻辑斯蒂分布，韦布尔分布，对数正态分布和高斯分布。
- 可编程的求和与求积、求导、解二次方程（包括复数解），质数判断。
- 十五种进制（二进制到十六进制）的整数运算。
- 时间与日期运算，以及基于实时时钟的秒表功能[4]。
- 金融计算，如平均收益率和销售利润率计算。
- 88 种进制转换，主要为国际单位与英制单位之间的互相转换。
- 50 个精度和 NIST 或 PTB 一致的物理常数，以及精选的重要数学常数、天文常数以及测绘用常数。
- 与计算机进行双向通信，以及 HP82240A/B 红外打印功能[5]。
- 备份用内存，提供断电保护。
- 覆盖了绝大多数语言的希腊字母与拉丁字母集（含大小写两种字体），以及数学符号。

WP 34S 是首个突破 4 堆栈的 RPN 计算器，这样就不用再担心计算时出现堆

[1]译者注：原文为 Rotate-and-Click，特指 HP 风格的带有稳定轴的按键。
[2]RPN 是逆波兰表示法（Reverse Polish Notation）的简称。
[3]不支持 42S 的矩阵菜单，*WP 34S* 使用一套基本的矩阵指令和一些库例程来处理矩阵。
[4]需要加焊晶振和两个电容。
[5]打印需要加焊红外二极管和电阻。此外还可选择购买 USB 电路板。

栈溢出问题[6]。*WP 34S* 可以设置为 4 堆栈模式，也可设置为增加了复数 LASTx 寄存器的 8 堆栈模式。传统的 4 堆栈模式与惠普计算器兼容，8 堆栈模式可用于复数计算、高等实数微积分和 4 维向量代数，或任何其它应用。两种模式下均有一组完整的指令用于堆栈的操作和浏览。

WP 34S 的内存还提供多达 107 个通用寄存器、112 全局用户标志位和最多 928 步 *RAM* 编程步数，闪存中则可保存多达 6014 步编程步数。提供 30 字节的 alpha 寄存器用于生成信息，16 个本地标志位以及 144 个本地寄存器用于编程，4 可编程热键用于自定义功能。大部分内存可以由用户自由支配。

WP 34S 项目始于 2008 年，是两个人（一名澳大利亚人和一名德国人）合作的结晶。我们在业余时间开发了 *WP 34S*，所以您可以称之为我们的爱好（尽管一些热心人士有不同的想法）。从一开始，我们就在惠普计算器博物馆的论坛上（*http://www.hpmuseum.org/forum/*）讨论了我们的项目。我们要感谢来自全球的项目组所有成员，他们教会了我们很多东西，贡献他们的想法，并在我们项目的多个阶段给予支持。特别感谢 *Marcus von Cube*（德国），他加入我们后，通过为 v1.14 版设计仿真器，使 *WP 34S* 得以实现，能够广泛地使用和方便地测试。从 v1.17 开始，该软件开始在 *HP-20b* 硬件上运行。自 v1.18 以来，*Neil Hamilton*（加拿大）提供了一个非常有用的编译器/反编译器，甚至在 v2.1 后还添加了一个符号预处理器。对于 v3.0，*Pascal Méheut*（法国）为各种操作系统提供了一个多功能的刷机工具。在 v3.1 中，由于 *Christoph Gießelink*（德国）的大力支持，在 HP82240A/B 打印机上打印成为可能；*Harald Pott*（德国）开发了一套微型 USB 电路板；*Ciaran Brady*（英国）为我们的 *WP 34S* 写了一本入门指南，*Christian Tvergaard* 和 *Peter Murphy*（均为美国）仔细校对了本手册。非常感谢你们的支持！

我们命名我们的宝贝为 *WP 34S*，以纪念 1979 年发布的 *HP-34C*，它是最强大的紧凑 LED 袖珍计算器之一。*WP 34S* 是我们在惠普的硬件限制下，所做的小小的工作。对于未来的 *43S*，我们梦想能成功实现对 *HP-42S* 的超越。愿我们的项目有助于说服那些比我们拥有更多资源的人：为严肃的科学仪器市场提供产品是值得的！

我们已经尽一切所能全面检查了 *WP 34S* 的固件。我们希望 *WP 34S* 没有严重的错误。但是，这是无法保证的。尽管 *WP 34S* 项目已经结束，我们承诺在必要时继续改进 *WP 34S*。如果您发现任何奇怪的运行结果，请向我们报告。如果它们是由内部错误引起的，我们将在新固件发布时为您提供更新。与以前一样，我们将尽快响应。

Enjoy!

Walter Bonin

[6]RPL 计算器的堆栈是无限的（RPL 是 1980 年代由 RPN 发展而来的计算器操作系统），典型的 RPN 计算器只有有限的堆栈。

打印约定和通用缩写

- 在本手册中，标准字体为黑体[7]。强调通过下划线或粗体打印标示；特定术语、标题、商标、名称或缩写以斜体标示；网络链接用斜体标示；文中引用链接用蓝色带下划线标示；粗斜体字母，如 ***n*** 表示变量；常数示例（例如标签、数字或字符）使用粗体标示；计算器指令通常按其名称调用，在文中统一使用大写字母标示。

- 方括号内的字用于引用计算器按键，如**[CPX]**。字母数字和数字显示（如 Hello! 和 1234）使用相应的计算器字体。

- Courier字体用于文件名和数字格式。

- 寄存器地址使用粗体 **Times New Roman** 大写字母，而寄存器内容以小写粗斜体表示。例如：在堆栈寄存器 **Y** 中存在值 ***y***，在通用寄存器 **R45** 中保存有 ***r45***，在 alpha 寄存器中存在 ***alpha***。一般而言堆栈内容按以下顺序引用[***x***, ***y***, ***z***, ***...***]。单位用小写斜体字体表示。

除非另有说明，以上约定在本手册普遍适用。

本手册中不同位置使用以下缩写：

CDF = 累积分布函数（参见第 47 页）。
DP = 双精度（参见第 183 页）。
FM = 闪存（一种特殊的 *RAM*，参见第 76 页）。
HP = 惠普。
IOP = 操作指令目录（参见第 77 页）。
IR = 红外。
LCD = 液晶屏幕。
PDF = 概率密度函数（参见第 47 页）。
QF = 分位点函数（参见第 47 页）。
RAM = 随机存取内存，允许读取和写入操作。
ROM = 只读内存，仅允许读取操作。
RPN = 逆波兰表示法（参见第 15 页）。
SI = 国际单位制。
SP = 单精度（参见第 152 页）。
SRS = 子程序返回堆栈（参见第 149 页）。
USB = 通用串行总线，一种接口。

[7]译者注：本手册没有像大多数手册那样使用宋体作为正文字体。使用黑体主要是为了在移动设备上更清晰的阅读，同时观感也更接近原版英文字体。

XROM = 扩展 ROM（参见第 76 页）。

除以上之外的一些其它的缩写可能在本手册局部使用。

最后：本手册的**警告**字样表示该操作存在极高风险。整本手册中仅有 4 处警告。已知最严重的后果是死机。重启计算器可以解决，但是会丢失所有的数据。

2. 从这里开始

如果知道如何使用一个优秀的老式惠普 RPN 科学计算器，您可以马上开始使用 *WP 34S*。使用此手册可以了解一些基本的设计概念，这些概念使得 *WP 34S* 超越了以前的 RPN 计算器。请继续使用此手册以供参考。

另一方面，如果这是您的第一个 RPN 科学计算器，或者很久没有用过 RPN 计算器了，我们建议先通读本手册的第 1 部分和第 2 部分。

WP 34S 延续了惠普自 1968 年起计算器产品线的传统。惠普于 1972 年推出了世界上第一台科学袖珍计算器。自从惠普生产计算器以来，其就以不断推出品质优良的计算器而闻名。可以在惠普计算器博物馆[8]低价发行的 DVD 上找到这些计算器的信息（参见 *http://www.hpmuseum.org/cd/cddesc.htm*）。本手册中提到的著名的 *PPC ROM* 也可以在上述 DVD 中找到。

此外，还可以下载由我们的一位用户编写的 *WP 34S* 初学者指南：*http://sourceforge.net/projects/wp34s/files/doc/WP_34S_Beg_Guide.pdf*。

WP 34S 上的大多数传统指令将像在 *HP-42S* 上一样工作，所以了解 *HP-42S* 有助于使用 *WP 34S*。不过这个手册已经包含了 *WP 34S* 的所有的功能，包括一些公式和技术解释。然而，它的目的不是取代数学，统计，物理，工程或编程的教科书，也不是一个 RPN 计算入门指南。

接下来的内容首先介绍用户界面，以便了解在哪里可以找到需要的内容。然后介绍了一些基本用法、*WP 34S* 的内存机制和寻址，以及便于了解计算器状态的显示和指示符。再后面是本手册的一个主要部分：包括所有可用操作的索引，访问它们的方法，以及所有菜单内容的完整列表。本手册以涵盖特殊主题的附录结束，例如，*WP 34S* 遇到异常情况没有按预期执行指令，返回的错误消息的列表。在那里还可以找到在出现新的固件修订版本时，更新 *WP 34S* 固件的方法。

WP 34S 旨在帮助您进行各种简单或复杂的计算。但它只是一个工具——虽然是一个非常强大的工具——它不能为您思考，也不能检查您扔给它的问题的合理性。不要因为您的错误而责怪我们或 *WP 34S*。收集信息，在按下按键之前进

[8]这个网上博物馆与惠普没有任何关联。事实上，它是由业余爱好者推动的。我们向您推荐这个网站，因为可以在那里找到数量惊人的计算器相关信息，并且您可以在其论坛上提问。要注意，它的大部分成员是数学或电子爱好者，专注于计算器技术，对他们的爱好极为认真，并生活在一个自豪感很强的国家（即所做的一切都是伟大的和正确的——不要试图告诉他们其他人可能会对此持异议）。

在那里，您会找到聪明的人和能够提供帮助的人——但是要记住，只有一小部分注册成员在该论坛积极活跃。例如，要懂得一些幽默梗的话，需要某种高于平均水平的智力。一个活跃的帖子可能欢迎这样的发言，但另一个成员可能会因为这些东西在帖子的顶部而感到不安并谴责它。

该论坛没有保护少数人的规定。到 2016 年为止，该论坛还在被独裁统治，鼓励所有人抨击其他成员。然而，一旦被抨击，您可能会很容易地被无限期禁言——在这种情况下，管理员不会征求您的意见也不关心指控的真实性。因此，要非常小心！

行思考，并检查结果：这些都是您应该做的。

键盘的基本知识

开始探索 *WP 34S*：按左下角的键（**[ON]**字样印在该键下方）将 *WP 34S* 打开。如果是第一次打开 *WP 34S*，屏幕上显示的内容如下图所示。

要将计算器关闭，请按绿色键**[h]**（按下的时候会有一个小小的ϟ显示在屏幕左上方），然后按**[EXIT]**键（**[OFF]**字样印在该键下方）。由于 *WP 34S* 具有关机保护存储器，因此关机不会导致数据丢失。为了节省电池能量，*WP 34S* 将在闲置 5 分钟后自动关闭——再次打开时，可以从上次中断的地方继续工作。

要调节显示对比度，请按住**[ON]**，同时按**[+]**或者**[-]**来调节。

与 *HP-42S* 的键盘相比，最显著的不同是 *WP 34S* 键盘颜色更丰富。平均每个键有五个功能。白色字符表示按键的主要功能，印在按键上部平面部分。辅助功能由绿色标签标明，印在 34 个按键的倾斜下表面上，金色和蓝色标签印在对应按键的下方。26 个字母用灰色字体标识在对应按键的左下方。

要使用白色标签表示的功能，只需按相应的键（因此称为主要功能）。要使用金色，蓝色或绿色标签表示的功能，分别按前缀**[f]**，**[g]**或**[h]**然后按相应的键[9]。

以键**[5]**为例，按下：

- **[5]**将在显示屏中输入数字 5，
- **[f]**+**[5]**将通过[$\bar{x}$]功能计算统计寄存器中所累积数据的算术平均值，
- **[g]**+**[5]**将通过**[s]**功能计算统计数据的标准差，
- **[h]**+**[5]**将通过**[STAT]**功能打开一个额外的统计函数菜单。键盘上带有下划线

[9]为了在纸上获得更好的可读性，我们用方括号里面的白底深色字表示键盘标签，例如**[EEX]**或**[π]**，由于在彩色打印下显得多余，因此后者省略了前缀**[h]**。

的所有标签均会打开相应菜单，

- 灰色**[R]**会在 alpha 模式下启用，例如文本输入。

使用[f]，[g]和[h]，就可以轻松地执行比 37 种主要功能多几倍的功能。前缀按键被激活时，由显示屏左上角的 f，g 或 h 指示（如果可用）。按住[f]，[g]或[h]可以多次调用相同颜色的多个功能。

现在，让我们假设要围一个长 40 码，宽 30 码[10]的长方形小块土地，已经安装了第一个角柱（A）和第二个角柱（B），相距 40 码，在哪里安装第三个和第四个角柱（C 和 D）可以确保栅栏围形成所需的矩形？此时只需输入：

[3][0]		
[ENTER⇧]		此键在此处将输入中的两个数字分开[11]。
[4][0]		
[⇨P]		注意[⇨P]是通过按键盘右上角的[g]+[⇨]实现的。

因此，只需拿一条 80 码的绳索，将其一端钉在角柱 A 上，另一端钉在角柱 B 上，取下松散的绳圈，然后从 A 开始沿垂直于 AB 的方向走 30 码即可。当绳索的两个部分都拉紧时，停下来将角柱 C 设置在那儿即可。可以用相同的方式设置角柱 D。

此方法适用于任何尺寸的任意矩形。按下[⇨P]，*WP 34S* 就会自动计算对角线长度。

正如在[⇨P]旁边可以找到另外两个有箭头的标签一样，*WP 34S* 上的标签通常根据其用途进行了分组。

其中，四组基本功能如下图所示。

[10]本手册是为国际读者编写的，我们非常了解国际上公认并已被全球几乎所有国家采用的国际单位制（SI）。尽管如此，我们还是在这里使用（老掉牙的）英制单位，以便美利坚合众国的读者可以方便理解。但是该示例也适用于用米做单位的计算。

[11]注意按下按键后数字 40 被移动到了到右侧，并添加了小数点。这表明该数字已经完成输入。通常，为了节省空间，后文我们将只用对应的字体表示屏幕下方显示的数字部分。

通常，函数及其逆函数（如果存在的话）彼此相邻放置。如果使用[f]调用函数，则使用[g]调用其逆函数。以下除外：

- 数字输入键（**[0]**，**[1]**，**[2]**，…，**[8]**，**[9]**，**[.]**，**[+/-]**，**[EEX]**和**[⇦]**），
- 三个前缀（[f]，[g]和[h]），
- 四个基本算术运算（**[+]**，**[-]**，**[×]**和**[/]**）。

按键标签大致区分为 5 个大组，具体如下：

熟悉了这些分组，就能很快在 *WP 34S* 上轻松找到需要的功能。

计算器上面印有 168 个标签，大多数标签的名称都调用与其同名的函数。例如**[ALL]**，调用函数 ALL，而**[FIX]**调用 FIX 函数。但是，在上面的示例中，按下**[⇨P]**调用功能→POL——还有一些类似的特殊标签。让我们从键盘的左上方开始介绍它们：

1. **[A]**、**[B]**、**[C]**和**[D]**被命名为热键——它们可以直接调用带有这些标签的用户程序。通过这些热键，就可以对四个喜欢的功能实现一键访问。如果标签 A、B、C 和 D 没有在当前程序中被指定，则这些键默认分别为**[CLΣ+]**，**[1/x]**，**[y^x]**或**[$\sqrt{x}$]**的功能，和其上方打印的标签一致。

2. [HYP]（即[f]+**[A]**）是双曲函数 SINH，COSH 和 TANH 的前缀；**[HYP^{-1}]**（即[g]+**[A]**）是它们的反函数 ASINH，ACOSH 和 ATANH 的前缀。类似地，**[SIN^{-1}]**代表 ASIN，等等。

3. **[⇨]**是直接转换所显示的值（即 x）的前缀。它的后面可以是[H.MS]，[H.d]，**[DEG]**，**[RAD]**或[GRAD]（相应的函数将被调用，例如→H.MS）。[⇨]后输入[2]，**[8]**或**[16]**将把 x 转换为相应进制的整数。[⇨]也可用于间接寄存器访问（见下文）。[R⇦]将平面中的极坐标转换为直角坐标，**[⇨P]**反之（请参见→REC 和→POL）。

4. **[CPX]**用于调用复数操作（请参阅第 29 页）。[a b/c]和**[b/c]**进入分数模式以显示真分数或假分数（请参阅 PROFRC 和 IMPFRC）。

5. [H.MS]和[H.d]代表两种经典的时间和角度模式，其中后者表示十进制时间和角度，通常也表示十进制浮点数（请参阅 DECM）。

6. [α]设置 alpha 模式（请参阅第 62 页）。而[2]，**[8]**，[10]或**[16]**设置整数模式，分别用于二进制，八进制，十进制或十六进制数进行计算（请参阅第 53 页）。

7. **[LG]**返回以 10 为底的对数，**[LB]**返回以 2 为底的对数（请参见 LOG…）。

8. **[!]**在所有数字模式调用 x!函数。[Cy, x]和**[Py, x]**（在此称为 COMB 和 PERM）的工作方式与 *HP-15C* 相似。**[r]**调用 CORR，[|x|]调用 ABS，**[RND]**调用 ROUND。

9. 有三个切换键：**[./,]**用于小数点切换，**[P/R]**用于编程功能切换（和 *HP-15C* 类似），以及[⇧]表示在 alpha 模式下切换大小写。

以上就是全部的特殊标签。*WP 34S* 上的每个指令的完整列表，调用它的组合键以及必要的个别说明在后面的操作索引（*IOP*）中，供参考。

在进入下一节之前，先说明几个前提：

1. 我们假定您至少从高中，职高，中专毕业或具有同等的学力。因此，我们将在本手册不解释基本的数学规则和概念。

2. 在 40 年的袖珍科学计算史中，不同的作者创造了大量从有趣到复杂的应用程序，并描述了这些应用——比我们自己创造的更多、更好。我们不打算重复它们。相反，我们再次推荐前面提到的 DVD：它包含从在 20 年前，即 1968 年他们的第一个计算器 *HP-9100A* 以来，几乎所有的老式惠普计算器用户指南，简介和说明书。请放心，在 *WP 34S* 上，以上任何计算器涉及的任何计算都会显著的加快，而且大多数计算甚至会以更优雅的方式完成。

3. 无需在计算中输入任何单位。只需在计算过程中保持一致的单位制，就可以获得有意义的结果。但是，如果需要将特定输入值转换到这个单位制，或者希望将结果转换为指定单位或本地单位，请使用**[CONV]**。

4. 即使只以整数输入土地的两个边，*WP 34S* 也会使用浮点数字计算其对角线。它也可以输入和输出小数。如果需要的话，也可以输入分数，例如 $6\,{}^{1}/_{4}$。我们将在下面进一步介绍分数模式和其他模式。

输入数字

输入数字就像打字一样简单。比如输入 12.34，按下**[1][2][.][3][4]**。可以通过按下**[⇦]**立即删除任何错误输入的数字，并随后输入正确的数字。

如果输入负数，以 -5.6 为例，可以按下**[5][+/-][.][6]**或者**[5][.][+/-][6]**或**[5][.][6][+/-]**——在输入数字的同时输入**[+/-]**符号会改变其正负符号。

如果要输入一个很大的数字，比如宇宙的年龄，可以像这样输入：**[1][3][.][8][2][EEX][9]**，输入后显示：

按下**[ENTER⇧]**结束输入[12]，在初始默认显示模式下将会显示：

输入特别小的数字，比如原子的直径（0.0000000001*m*——在 1 前面有 10 个零）用类似的方式输入：**[EEX][+/-][1][0]**[13]会显示：

[12]在完成输入之前，它通常只在数字行中显示。输入由指令（例如这里的**[ENTER⇧]**）结束并释放，以用于后续的操作。

[13]注意这里不需要输入**[1][EEX][+/-][1][0]**，如果**[EEX]**前面不输入数字，默认有效数字部分是 1。**[EEX]**之后输入的**[+/-]**将改变指数部分的符号。如果想改变有效数字部分的符号，请在**[EEX]**之前或输入关闭后按下**[+/-]**。

输入结束后显示:

在其它显示模式下可能会更紧凑地显示：

实数运算

WP 34S 功能的大多数指令都是输入和输出像 1、-2.34、π 或 5.6E-7 这样的实数的运算或函数。请注意，整数（例如 3、10 或-1）只是实数的一个子集。

本机的一些实数函数仅对一个数字进行操作。例如：

输入 **[.][4][9]** 显示 0.49

然后输入 [√x] 会得到 0.7 因为 0.7^2=0.49

通常这类计算功能会用计算结果 **f(*x*)**代替输入值 ***x***（即计算器上显示的值），在本例中， **f(*x*)=**$\sqrt{x}$。这也是大部分科学计算器的工作模式。

WP 34S 的一元函数有**[LN]**和[e^x]，**[LG]**和[10^x]，**[LB]**和[2^x]，倒数**[1/x]**，三角函数及其逆函数，双曲函数及其逆函数，阶乘**[!]**，布尔运算**[NOT]**等。

有一些常用的数学运算发生在两个数字之间，如加法和减法。

示例：小明卡里有 1234 元，花掉 56.7 元后，卡里还剩下多少钱？解决此类问题的一种简单方法如下：

在纸上计算		在 *WP 34S* 上		
写下第一个数字	1234	按出第一个数字	**[1][2][3][4]**	1234
换行		分隔两数	**[ENTER↑]**	
写下第二个数字	56.7	输入第二个数	**[5][6][.][7]**	56.7
两数相减得到	1177.3	两数相减	**[-]**	1177.3

这是 RPN（逆波兰输入法）的精髓：
先提供必要的操作对象，然后执行请求的操作。

与其他计算器的输入方式相比，RPN 的一个主要优点是它始终坚持这一基本规则[14]。

在纸上进行计算时，纸上会留下写下的数字，计算器也一样需要一个地方存储数据。堆栈起到了这样的作用。如果条件允许，也可以用它回顾中间步骤的过程和结果，就像在纸上计算一样。让我们来看看这是如何做到的。

[14]有些人声称这事实上只适用于 RPL。也许他们是对的。然而，在我看来，RPL 其潜在的后缀规则繁复异常，超越了人类的忍耐极限，不仅普通人不胜其烦，许多科学家和工程师同样不能幸免。因此我们决定在 *WP 34S* 上坚持使用 RPN。

堆栈基本机制

可以把堆栈想像成一堆寄存器[15]：从下往上，它们一般被命名为 **X**，**Y**，**Z** 和 **T**，在 *WP 34S* 上，后面还可能跟着 **A**，**B**，**C** 和 **D**。新的输入总是在 **X** 中进行，并在屏幕的底部一行显示。同时 ***y*** 也可以根据需要在屏幕中显示。

堆栈寄存器	内容
D	***d***
C	***c***
B	***b***
A	***a***
T	***t***
Z	***z***
Y	***y***
屏幕显示 **X**	***x***

[ENTER⇧]键用来分隔两个依次输入的数值。它会结束第一个数字 ***x*** 的输入并将其拷贝到 **Y** 中，因此可在不丢失信息的情况下在 **X** 中输入另一个数[16]。

在上面的示例中，输入完第二个数字之后，按**[-]**键从 ***y*** 中减去 ***x*** 并输出结果，**f(*x*, *y*)=*y*-*x*** 的结果会被拷贝到寄存器 **X** 中，并显示在屏幕上。上述操作过程适用于大多数两个数字之间的实数函数运算。

有很大一部分的数学运算是这种二元函数运算，或者这种运算的组合排列，让我们看一例连续计算：

$$\frac{(12.3 - 45.6) \cdot (78.9 + 1.2)}{(3.4 - 5.6)^7}$$

这是六个二元函数的组合，其中包括：两个减法，一个加法，一个乘积，一个幂运算和一个除法。下面显示了从左上角公式入手，*WP 34* 会如何显示，以及计算过程中堆栈内容的变化[17]：

T				
Z				
Y		12.3	12.3	
X	123	123	45.6	-33.3
输入	**[1][2][.][3]**	**[ENTER⇧]**	**[4][5][.][6]**	**[-]**

如上所示，第一个括号内的计算已经完成，现在我们来进行第二个括号的计算：

T					
Z		-33.3	-33.3		
Y	A -33.3	78.9	-78.9	-33.3	
X	78.9	78.9	12	80.1	-2667.33
输入	**[7][8][.][9]**	**[ENTER⇧]**	**[1][.][2]**	**[+]**	**[×]**

通过示例会发现第一个括号的结果（-33.3）被自动（A）提升到 **Y** 栈中，

[15]在后文中您会了解到 *WP 34S* 中寄存器的更多信息。

[16]这里**[ENTER⇧]**沿袭了从 1972 年的 *HP-35* 到 1995 年的 *HP-42S* 上的用法。这种用法通常被称为 ENTER“将 ***x*** 压入堆栈”。在执行这一操作的同时，高一级的堆栈的内容会上移。也就是说在 **X** 栈的内容进入 **Y** 栈之前，**Z** 栈的内容就会升入 **T** 栈，**Y** 栈的内容进入 **Z** 栈。关于堆栈详情的图片请参见 25 页。*HP-30b* 采用了不同的 ENTER 模式——但 *WP 34S* 坚持使用经典的 RPN。

[17]在较高层级的堆栈中可能已经加载了数据，它们来自于之前的操作，与此计算无关，可以搁置不管。

以避免在键入下一个数字时覆盖它。这称为自动堆栈提升[18]，是 RPN 计算器的标准配置。

在完成第二个括号的计算之后，在堆栈上就得到了分子中两个括号内的计算结果。进行乘法以完成分子部分的计算已经准备好，所以按下**[×]**就可以了。

现在继续计算分母：

T						
Z		-2667.33	-2667.33		-2667.33	
Y	-2667.33	3.4	3.4	-2667.33	-2.2	-2667.33
X	3.4	3.4	5.6	-2.2	7	-24943...
输入	**[3][.][4]**	**[ENTER⇧]**	**[5][.][6]**	**[-]**	**[7]**	**[Y^X]**

最后剩下的就是让分子被分母除，只要按一下**[/]**键，就能得到结果：10.693453464 8。

正如上文所见，当计算二元函数时，堆栈寄存器的内容会下降。就像上面提到的自动堆栈提升一样，这个堆栈下降会影响所有级别：***x*** 和 ***y*** 合并，将 **f(*x*, *y*)** 的结果加载到 **X** 中，然后 ***z*** 下降到 **Y**，***t*** 下降到 **Z**。由于再上面没有任何可以用于下降的数，顶部堆栈的内容将会被复制。可以利用这个顶栈复制的特性来实现一些巧妙的计算。请看下面的复利计算：

假设在银行存 15000 美元，银行的利息为 3.25%。那么 2 年、3 年、5 年、8 年后，账户存款数如何？本示例只对货币值感兴趣，因此可将显示格式设置为 **[FIX]**2。它会使输出显示舍入到分（内部数字运算本身保持更高的精度）[19]。

T		1.0325	1.0325	1.0325	1.0325	1.0325	1.0325
Z		1.0325	1.0325	1.0325	1.0325	1.0325	1.0325
Y	10.69...	1.0325	1.0325	1.0325	1.0325	1.0325	1.0325
X	10.325	1.03	15 003	15.9084	16.5105	17.6017	19.3736
输入	1.0325	[FILL]	15**[EEX]**3	**[×][×]**	**[×]**	**[×][×]**	**[×][×][×]**
				2 年后	3 年后	5 年后	8 年后

这里，每个乘法都会消耗 ***x*** 和 ***y***，然后是 ***z*** 下降到 **Y**，***t*** 下降到 **Z**。由于最高位堆栈复制的特性，利率在堆栈上保持为常数，因此累计资本值的计算成为一系列简单的按**[×]**的操作。

[18]对于四级堆栈，全自动堆栈提升过程将 ***z*** 移动到 **T** 栈，然后 ***y*** 移动到到 **Z** 栈，最后移动 ***x*** 到 **Y** 栈。顶层 **T** 的旧内容被覆盖。因此自动堆栈提升会影响所有级别的堆栈。自动堆栈提升后文将不再单独说明。值得一提的是，极个别的情况下自动堆栈提升会被禁用（只有四个指令不自动升栈，其中一个是**[ENTER⇧]**）。

[19]出于篇幅原因，在下文中除非另有说明，本手册将使用纯文本表示数字输入。

WP 34S 还有一些三元实数函数（例如→DATE 和%MRR），它会用函数的结果 **f**(*x*, *y*, *z*)代替 *x*。然后 *t* 下降到 Y，依此类推。并且顶部堆栈的内容会复制两次。

一些实数函数（例如 DECOMP 或 DATE→）输入一个数字，但返回两个或三个数。还有一些其他操作（例如 RCL 或 SUM）根本不消耗任何堆栈输入，但是会返回一个或两个数字。这些额外的数字将被压入堆栈，每个实数需要占用一级堆栈。

除了传统的堆栈和寄存器控制操作**[ENTER⇧]**，**[x⇆y]**，**[R⇩]**，**[STO]**，**[RCL]**，LASTx，CLSTK，**[VIEW]**，**[R⇧]**和**[x⇆]**（主要位于右图的蓝框内），*WP 34S* 还有**[FILL]**，DROP，RCLS，STOS，SSIZE4，SSIZE8，SSIZE?，Y⇆，Z⇆，T⇆和⇆。翻到下一页可以了解**[ENTER⇧]**，**[FILL]**，DROP，**[x⇆y]**，**[R⇩]**，**[R⇧]**，以及在 SSIZE4 或 SSIZE8 设置下 LASTx 对堆栈的影响。有关其他指令的信息，请参阅 *IOP* 表。

计算器有史以来的第一次，*WP 34S* 可以选择四或八个级别的堆栈（参见 SSIZE4 和 SSIZE8）。因此，堆栈内容不仅取决于所执行的特定操作及其范围，还取决于所选择的堆栈大小。几十年来，我们已经熟知了四级堆栈的功能。在 *WP 34S* 的更大的堆栈中，一切都是类似的，只是中间级别更多了而已。翻到下一页以了解详细信息。

	级别	堆栈初始内容	执行操作后堆栈寄存器内容							函数功能		
			[ENTER⇧]	[FILL]	[DROP]	[x⇆y]	[R⇩]	[R⇧]	[RCL][L]	一元运算，比如[x^2]	二元运算，比如[/]	返回两个值的运算，比如[s]
4 级堆栈	T	*t=44.4*	*33.3*	*11.1*	*44.4*	*44.4*	*11.1*	*33.3*	*33.3*	*44.4*	*44.4*	*22.2*
	Z	*z=33.3*	*22.2*	*11.1*	*44.4*	*33.3*	*44.4*	*22.2*	*22.2*	*33.3*	*44.4*	*11.1*
	Y	*y=22.2*	*11.1*	*11.1*	*33.3*	*11.1*	*33.3*	*11.1*	*11.1*	*22.2*	*33.3*	s_y
	X	*x=11.1*	*11.1*	*11.1*	*22.2*	*22.2*	*22.2*	*44.4*	*last x*	*123.21*	*2*	s_x
8 级堆栈	D	*d = 88.8*	*77.7*	*11.1*	*88.8*	*88.8*	*11.1*	*77.7*	*77.7*	*88.8*	*88.8*	*66.6*
	C	*c = 77.7*	*66.6*	*11.1*	*88.8*	*77.7*	*88.8*	*66.6*	*66.6*	*77.7*	*88.8*	*55.5*
	B	*b = 66.6*	*55.5*	*11.1*	*77.7*	*66.6*	*77.7*	*55.5*	*55.5*	*66.6*	*77.7*	*44.4*
	A	*a = 55.5*	*44.4*	*11.1*	*66.6*	*55.5*	*66.6*	*44.4*	*44.4*	*55.5*	*66.6*	*33.3*
	T	*t = 44.4*	*33.3*	*11.1*	*55.5*	*44.4*	*55.5*	*33.3*	*33.3*	*44.4*	*55.5*	*22.2*
	Z	*z = 33.3*	*22.2*	*11.1*	*44.4*	*33.3*	*44.4*	*22.2*	*22.2*	*33.3*	*44.4*	*11.1*
	Y	*y = 22.2*	*11.1*	*11.1*	*33.3*	*11.1*	*33.3*	*11.1*	*11.1*	*22.2*	*33.3*	s_y
	X	*x = 11.1*	*11.1*	*11.1*	*22.2*	*22.2*	*22.2*	*88.8*	*last x*	*123.21*	*2*	s_x

[RCL][L]相当于过去的 LASTx 指令。

在使用堆栈的情况下，RPN 使得所有的括号在计算中完全没有存在的必要。没有运算符优先级。下面是另一个示例，显示稍微复杂的公式和如何按键操作来得到结果（对应操作涉及的堆栈用颜色表示）：

$$\sqrt{\frac{1+\left|(\frac{30}{7}-7.6\times0.8)^4-(\sqrt{5.1}-\frac{6}{5})^2\right|^{0.3}}{\left\{\sin\left[\pi(\frac{7}{4}-\frac{5}{6})\right]+1.7(6.5+5.9)^{\frac{3}{7}}\right\}^2-3.5}}$$

[RAD] 7 [ENTER⇧] 4 [/] 5 [ENTER⇧] 6 [/] [-] [π] [×] [SIN]	$\sin\left[\pi(\frac{7}{4}-\frac{5}{6})\right]$
6.5 [ENTER⇧] 5.9 [+] 3 [ENTER⇧] 7 [/] [y^x] 1.7 [×]	$1.7\times(6.5+5.9)^{\frac{3}{7}}$
[+] [x^2] 3.5 [-]	分母
7.6 [+/-] [ENTER⇧] .8 [×] 30 [ENTER⇧] 7 [/] [+] 4 [y^x]	$(\frac{30}{7}-7.6\times0.8)^4$
6 [ENTER⇧] 5 [/] 5.1 [$\sqrt{x}$] [-] [x^2]	$(\sqrt{5.1}-\frac{6}{5})^2$
[-] [|x|] .3 [y^x] 1 [+]	分子
[x⇆y] [/] [$\sqrt{x}$]	结果（0.35）

如上面的颜色所示，即使计算这样的式子，也仅仅需要四个堆栈。可以注意到，整个运算过程中没有待处理的操作——每个操作单独执行，一次执行一个，可以完美地控制每个中间结果。这是 RPN 的另一个特点和优势。在现实中，中间结果是有价值的，下一步如何计算可能取决于已有的结果——这被称为"探索性数学"，比起计算教科书上的公式来，在专业的工作中更常见。

从里到外计算这样的公式是一个明智的策略。如果从该公式的分子开始算起，那么得出最后的结果至少需要 6 级堆栈。

通过这里提供的八级堆栈，即使在您科学家或工程师的生涯中计算最复杂的公式，也完全没有问题。

飞机的马赫数是其速度和高度的函数，设速度为 350 节[20]，高度为 25500 英尺，用下面的公式来计算马赫数。数十年来，这个式子被用于演示 RPN 的简单性和一致性：

$$\sqrt{5\left(\left[\left\{\left(1+0.2\left[\frac{350}{661.5}\right]^2\right)^{3.5}-1\right\}\left\{1-6.875\times10^{-6}\times25500\right\}^{-5.2656}+1\right]^{0.286}-1\right)}$$

从最内侧的括号开始，如下计算（仅需要 3 个堆栈）：

[20]节是一个古老的英制单位：1 节=1 海里/小时=463/900*m/s*≈0.5144*m/s*。英尺则是一个英制长度单位（见 CONV）。

350 **[ENTER⇧]** 661.5 **[/]** **[x^2]** **.2** **[×]** **1** **[+]** 3.5 **[y^x]** **1** **[-]** 6.875 **[EEX]** **[+/-]** 6 **[ENTER⇧]** 25500 **[×]** **[+/-]** **1** **[+]** 5.2656 **[+/-]** **[y^x]** **[×]** **1** **[+]** .286 **[y^x]** **1** **[-]** 5 **[×]** **[$\sqrt{x}$]**，结果约为 0.84，即音速的 84%。

最后，我们建议：选择 8 个堆栈，在您做数学时，让 *WP 34S* 来做算术[21]！

[21]当然，即使是八个级别的堆栈，构建导致堆栈溢出的示例也不是难事。但这是人为构造的式子，不是现实中的公式。最后，我们假设使用计算器的人是聪明的，知道应该从内到外进行计算，如上文所建议的那样。

修正错误

没有人是完美的——错误不是已经发生，就是在要发生的路上。但是使用 RPN 输入的优势在于，即使计算很长的算式中出现错误，也很容易进行错误的修正，因为 *WP 34S* 在执行函数之前自动将 ***x*** 加载到特殊寄存器 **L**（代表 Last x）[22]。这有哪些帮助呢？

1. 如果在使用函数时收到错误消息，请按[⇦]删除该消息，此时会返回到错误发生前的状态。

2. 如果调用了错误的一元函数功能（或[%]、[Δ%]），只需按[⇦]和**[RCL][L]**[23]来将其撤消。这两个步骤将堆栈完全恢复到错误发生前的状态——然后继续从被中断的位置开始计算。

3. 二元函数计算的错误修正需要三个步骤。

示例：假如在前一页上示例的倒数第二步中无意中按下了**[×]**而不是**[/]**。这个错误很容易撤消，方法如下：

T	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2
Z	旧数据 1	旧数据 2	旧数据 1	旧数据 2	旧数据 1	旧数据 2	旧数据 2
Y	分子	旧数据 1	分子*分母	旧数据 1	分子	旧数据 1	旧数据 1
X	分母	分子*分母	分母	分子	分母	分子/分母	037
输入		**[×]**	**[RCL][L]**	**[/]**	**[RCL][L]**	**[/]**	[√x]
	目前还好	哦不!	1	2	3	恢复	

在步骤 1 中，**[RCL][L]**呼出完整分母，即错误前的最后一个 ***x***。

步骤 2 通过执行反操作来撤消错误操作。

最后，第 3 步完全恢复堆栈，就像发生错误之前一样。现在可以继续计算，获得正确的完整结果。

错误的**[y^x]**可以用相同的方法撤消：

T	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2
Z	旧数据 1	旧数据 2	旧数据 1	旧数据 2	旧数据 1
Y	底数	旧数据 1	底数^{指数}	旧数据 1	底数
X	指数	底数^{指数}	指数	底数	指数
输入		**[y^x]**	**[RCL][L]**	**[$\sqrt[x]{y}$]**	**[RCL][L]**
		哦不!	1	2	3

…但是我们去哪里找**[$\sqrt[x]{y}$]**呢？此问题将在后面解决。

[22]只有少数命令改变 ***x*** 但不加载 **L**。它们在 *IOP* 中已明确指出。

[23]请注意，**[EEX]**键左下方印有灰色 **L** 字样。尽管如此，我们会将这些击键引用为**[RCL][L]**而不是**[RCL][EEX]**，原因在下一节中会明确指出。

更高级的二元函数可能需要更多的精力来修正错误。意外地调用[LOG_X]也可以撤消，但需要一个附加的辅助寄存器：

T	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2	旧数据 2
Z	旧数据 1	旧数据 2	旧数据 1	旧数据 2	旧数据 1	旧数据 2	旧数据 2
Y	n	旧数据 1	LOG_dn	LOG_dn	d	旧数据 1	n
X	d	LOG_dn	d	d	LOG_dn	n	d
输入		[LOG_X]	**[RCL][L]**	**[STO]**00	**[x⇆y]**	[y^X]	**[RCL]**00
		哦不!	1	2	3	4	5

通常，这种修正过程要求提供错误操作的逆操作。

4. 可以通过执行 DROP 来撤消对堆栈的意外推送，对于**[ENTER⇧]**、**[RCL]**、[$\hat{y}$]、[r]等类似的操作执行一次 DROP 或对于[$\bar{x}$]、[s]执行两次 DROP。但是由于此错误，将丢失最高（前两）层堆栈的内容，因此请将堆栈设置为八层以最大程度地减少影响。

LASTx 的小技巧

除了用于错误修正，寄存器 L 还可以有其它用处：

示例：计算$\frac{1.234 + 5.67809}{5.67809}$，YDON 已设置。

1.234	
[ENTER⇧]	
5.67809	
[+]	
[RCL][L]	
[/]	

因此，通过**[RCL][L]**调出 LASTx 可避免冗长数字的重复输入。它还允许再次使用中间结果，而无需单独存储它们。

如果不需要永久显示 y，请输入 YDOFF。

调用键盘上看不见的指令

WP 34S 具有 700 多种不同的指令。键盘上只打印了 168 个功能标签。那么，如何知道还有哪些其他指令？知道了他们的名字后，该如何调用它们？

第一个问题的答案：阅读本手册，*IOP* 包含了所有的指令。

第二个问题的答案："隐藏指令"存储在指令菜单中。请记住，带下划线的按键标签指向此类指令的集合。在 *WP 34S* 的十个按键倾斜的正面可以找到这些标签。例如，**[h]**+**[3]**指向**[X.FCN]**，这是机上最大的一个菜单。

以 17 页提到的 DECOMP 指令为例，这个指令没有在键盘上印出来。从目录找到操作索引（*IOP*），直接跳转到字母 D 处，找到 DECOMP，就可以获得这个指令在什么位置，能够做什么这些必要的信息。OK，输入 0.375 测试一下——调用这样一个指令甚至比查找它还容易：

从 *IOP* 中查到 DECOMP 在**[X.FCN]**菜单中。可如下操作：

输入：	*WP 34S* 显示	
[X.FCN]	↕∛‾x	这是存储的第一个指令
[D]	↕DATE	显而易见
[▼]	↕DATE+	
[▼]	↕DAY	
[▼]	↕DAYS+	
[▼]	↕DECOMP	就是它！现在运行它。
[XEQ]	y/x= 8	
[x⇆y]	3	0. 375=3/8. 这就是我们想要的。
0.46875	0.46875	换一个数字试试。
[X.FCN]	↕DECOMP	很好，这个函数被记住了！
[XEQ]	y/x= 32	
[x⇆y]	15	即 0. 46875=15/32.

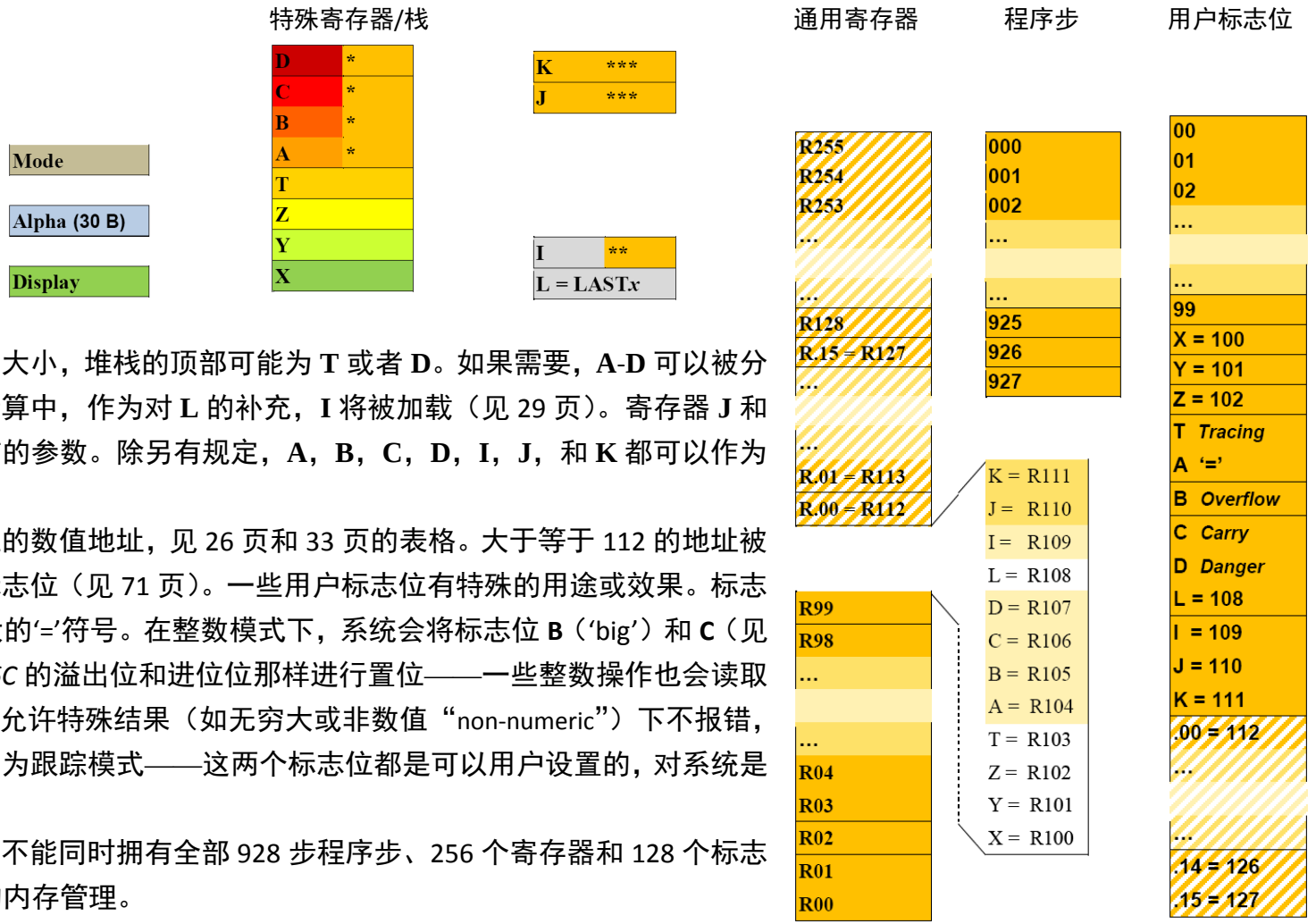
[▼]是向前浏览菜单，**[▲]**是向后浏览菜单。后文将介绍一种更为优雅的调用菜单中的指令的方法。

回答前面在"修正错误"一节提到的问题：

$\sqrt[x]{y}$ 也在**[X.FCN]**中，输入**[X.FCN]**[X]**[▼]**[▼]**[▼]**就可以看到↕ ˣ√‾y 了。

访问 *RAM* 中的对象

这个图给出了 *WP 34S* 完整的地址空间。根据对内存设置的不同，可以访问这些地址的一个子集。



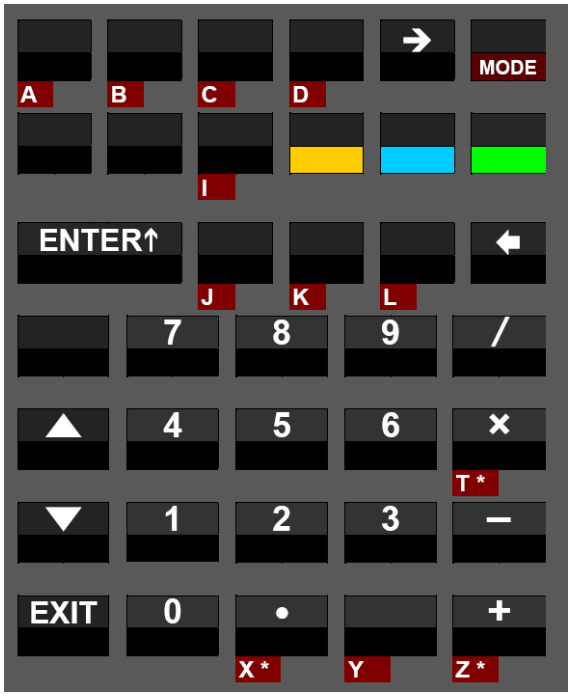
根据两种堆栈的大小，堆栈的顶部可能为 **T** 或者 **D**。如果需要，**A**-**D** 可以被分配为堆栈。在复数计算中，作为对 **L** 的补充，**I** 将被加载（见 29 页）。寄存器 **J** 和 **K** 用于保存统计分布的参数。除另有规定，**A**，**B**，**C**，**D**，**I**，**J**，和 **K** 都可以作为通用寄存器使用。

寄存器和标志位的数值地址，见 26 页和 33 页的表格。大于等于 112 的地址被用作本地寄存器和标志位（见 71 页）。一些用户标志位有特殊的用途或效果。标志位 A 会点亮屏幕中大的'='符号。在整数模式下，系统会将标志位 **B**（'big'）和 **C**（见 53 页）类似于 *HP-16C* 的溢出位和进位位那样进行置位——一些整数操作也会读取标志位 **C**。标志位 **D** 允许特殊结果（如无穷大或非数值"non-numeric"）下不报错，标志位 **T** 将打印设置为跟踪模式——这两个标志位都是可以用户设置的，对系统是只读的。

需要注意的是，不能同时拥有全部 928 步程序步、256 个寄存器和 128 个标志位——参见附录 B 的内存管理。

虚拟键盘——临时 alpha 模式（α_T）

在访问存储器的过程中（例如：输入用于存储、调用、交换、复制、删除或者比较的参数时）进行输入时，只需要 34 个按键标签，这远低于键盘上已有的 168 个标签。仅支持这 34 个按键标签的计算器模式叫做临时 alpha 模式。就像在后两页的示例中写的一样，它可能在寻址时被自动设置。进入临时 alpha 模式后，键盘会像下图所示的一样重新被分配：



左边这个图就是虚拟键盘，它和实际的 *WP 34S* 不相同。在这里，褐色的背景是用来突出显示那些被改变主功能的按键，如按下左上方的按键在临时 alpha 模式模式下会输入 A。也就是说，白色字体的字符表示着这个按键的主功能。

值得注意的是，alpha 模式下所有的按键都是主功能，不需要上档键，这可以快速和简单地输入那些有限的字母。因此所有地址都可以用最小的按键次数来访问。

会有特别的规则约束 **T**，**X**，和 **Z**——请看下面两页的表格。

只要为相应的操作输入了足够的字符，临时 alpha 模式将被终止或退出（返回到之前设置的模式）。可以使用**[⇦]**删除待处理的输入（逐个字符）或者通过**[EXIT]**取消正在输入的指令离开临时 alpha 模式。

注意：所有全局寄存器和分配的标志位的访问是无任何限制的。在 WP 34S 上没有“内存保护”这类机制。因此，需要留意内存的使用情况，以避免宝贵的数据被覆盖或被无意中删除。

下一节介绍如何访问在内存中任意位置的实数。

实数的寻址

1	用户输入 返回	[x=?]，[x≠?]，x< ?，x≤ ?，x≥ ?，x> ?或x≠ ? OP_?（在临时 alpha 模式下） 例如： x>_? x≤? x≥? x≠?			
2	用户输入 返回	**[0]或[1]** **OP *n* ?** 例如：x≤0?	堆栈或者字母寄存器**[Y]，[Z]，…，[J]，[K]** **OP? *x*** 例如：x≥? Y	**[ENTER⇧]**[24] 离开临时 alpha 模式 **OP? _**	**[⇨]** 开启间接寻址 **OP? →_**
3	用户输入 返回	将 *x* 和 0 比较。	将 *x* 和堆栈 Y 中的数字比较。	寄存器号 **[0][0]…[9][9]，[.][0][0]…[.][1][5]**[25] （如果相关寄存器已经被赋值）。 **OP? *nn*** 例如：x≠? 23	关于间接寻址请看下一页。
				将 *x* 和 R23 中的数值比较。	

[24]对于地址>19 的寄存器或本地寄存器可以跳过此步。后者以**[.]**开始他们的地址，请参阅后面有关编程的部分和附录 B。

[25]这里也可以输入**[A]，…，[D]，[I]，…，[L]，[T]，[Y]**或**[Z]**。但是表中绿色背景的输入路径更短。输入**[.][.]**还可以和 *x* 自身进行比较。

<table>
<tr><td>1</td><td>用户输入

返回</td><td colspan="3">[RCL]，[STO]，RCLS，STOS，[αRCL]，[αSTO]，[VIEW]，[VWa+]，[x⇆]，↘↯，↙↯，↖↯，[DSE]，[ISG]，DSL，DSZ，ISE，ISZ，[ALL]，[FIX]，[SCI]，[ENG]，DISP，BASE，位操作或者标志位指令等

OP _（在临时 alpha 模式下）
例如：RCL _[26]</td></tr>
<tr><td>2</td><td>用户输入

返回</td><td>堆栈级别或者字母寄存器[27][X]，[Y]，…，[J]，[K]

OP x
例如：SF K</td><td>寄存器号、标志位编号、进制、位宽或小数位数（范围见下表）

OP nn
例如：SCI 10</td><td>[⇨]
开启间接寻址

OP→ _</td></tr>
<tr><td>3</td><td>用户输入

返回</td><td>将 111 号标志位置位。</td><td></td><td>堆栈或者字母寄存器[X][28]，[Y]，…，[J]，[K]

OP→ x
例如：VIEW→L</td><td>寄存器号
[0][0]…[9][9]，
[.][0][0]…[.][1][5]，
如果相关的寄存器已经赋值。

OP→ nn
例如：STO→45</td></tr>
<tr><td></td><td></td><td></td><td></td><td>显示 L 指向的寄存器的内容。</td><td>将 x 存储到 r45 指向的位置。</td></tr>
</table>

种类	范围[29]	
寄存器号	直接寻址全局寄存器，0 到 99 直接寻址本地寄存器，.0 到.15 间接寻址，0 到 255（在没有本地寄存器时≤111）	上限受所分配的空间限制
标志位编号	直接寻址全局编号的标志位，0 到 99 直接寻址本地标志位（如果被分配），.0 到.15 间接寻址，0 到 127（在没有本地标志位时≤111）	
小数位数	0 到 11	
进制	2 到 16	
位	0 到 63	
字长	1 到 64 位	

[26]对于**[RCL]**和**[STO]**，**[+]**、**[-]**、**[×]**、**[/]**、**[▲]**或**[▼]**中的任何一个都可以在第二步之前输入，但在**[RCL][MODE]**（调用 RCLM）和**[STO][MODE]**（调用 STOM）中不行。连续输入两次将取消输入，例如**[STO][/][/]**等于**[STO]**。请注意，这种所谓的存储和调用算法可以在任何寄存器上运行，也可以在堆栈上，甚至在 L 上操作。

注意：[ENG]**[ENTER⇧]**调出 ENGOVR，[SCI]**[ENTER⇧]**调出 SCIOVR。请参阅操作索引。

[27]例外情形：[ALL]，[FIX]，[SCI]，[ENG]，DISP 和 BASE 间接寻址只接受字母寄存器。另外，指定寄存器 X 以及 RCL T、RCL= T、STO= T、RCL Z、STO Z、RCL+ Z 和 STO= Z 需要在字母前输入一个**[ENTER⇧]**，对最后一个即**[STO][+][ENTER⇧][Z]**。

[28]输入**[.][.]**可访问 *x*。

[29]除此外还有用字母表示的寄存器和标志位。

处理实数矩阵和向量

如果您不知道什么是矩阵，请忽略它——即使不用矩阵功能，也可以很好地使用 *WP 34S*。

但是如果您知道矩阵的话，请悉知 *WP 34S* 具有一组用于对矩阵进行加、乘、求逆、变换以及行操作的函数。通常，相应的指令已经模块化，以便在按键编程上对更多的矩阵函数提供底层的支持。也就是说，这些基本的线性代数子程序构成了 *WP 34S* 对矩阵的支持。同时，*WP 34S* 还具有计算行列式和求解线性方程组的功能。

WP 34S 中的矩阵用 `bb.rrcc` 这个描述符表示，其中
rr 是行数
cc是列数。因此这个矩阵有***rr***×***cc***个元素
这些元素会储存在从|***bb***|开始的基地址中

例：

描述符 7. 0203 表示一个 2×3 的矩阵，我们称其为（***M***），这 6 个元素放在两行三列中，如下所示：

$$(M)=\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{pmatrix}$$

矩阵描述符 **7**. 0203 表示里面的元素是这样子储存的

$$(M)=\begin{pmatrix} r07 & r08 & r09 \\ r10 & r11 & r12 \end{pmatrix}$$

根据寄存器实际的数据，矩阵元素的值可能是这样的：

$$(M)=\begin{pmatrix} -2.3 & 0 & 7.1 \\ 0.4 & 8.5 & -6.9 \end{pmatrix}$$

如果 ***cc*** 在描述符中被省略，那么默认形成一个方矩阵，即 ***cc*** 默认值为 ***rr***。例如：矩阵描述符 13.0400 描述一个 4x4 的矩阵，其元素储存在 **R13** 至 **R28** 中。矩阵元素最多有 100 个，因为用于矩阵功能的全局寄存器总共只有 100 个。

所有的矩阵指令见 *IOP* 和 MATRIX 菜单。

注意：*WP 34S* 不能识别输入的实数是普通数字还是矩阵描述符，使用时请务必注意。

向量可以被视为是一种只有一行或是一列的特殊矩阵。因此矩阵描述符看起来就是 `bb.01cc` 或者 `bb.rr01`。本机提供了可以进行三维向量计算的库程序。

如果只操作二维向量，这里还有一个简单的方法不需要描述符也可以计算：在 **X** 和 **Y** 中输入每一个向量的笛卡尔坐标（例如：在需要的时候用[R⇦]转换极坐标至笛卡尔坐标）然后选择下面的一种方法：

1. 用[Σ+]来相加或者用[Σ-]来相减，计算结果用 SUM 来显示
2. 在复数域计算（参考下一节）。这样的话，通过 ᶜDOT 或者 ᶜCROSS 指令还可以实现向量的乘法。

更多内容可参阅包含线性代数的教材。

复数运算

如果您不知道什么是复数，请忽略它——即使没有复数，也可以很好地使用 *WP 34S*。

如果您知道复数，那么请知悉，*WP 34S* 支持一系列在复数域下使用的指令。按键[CPX]用来调出复变函数[30]。例如：[CPX][f][COS]可以呼出复数下的余弦函数，它显示为ᶜCOS（上标 C 表示这是 *WP 34S* 的复变函数）。

注意 1： *WP 34S* 上的所有工作于复数的函数都使用笛卡尔坐标，每一个复数都占用相邻的两个寄存器，实部在低位寄存器，虚部在高位寄存器。

可以在任何时候使用[⇨P]将 $x_c=x+iy$ 变为极坐标式 $x_c=re^{i\varphi}$——记得在开始复数计算前用[R⇦]恢复它

注意 2： 所有的复变函数都在计算角度的时候使用弧度，在输出角度的时也会输出弧度。作为提示，RAD 会在[CPX]按键按下的时候亮起。要注意提供合适的角度数据，在角度模式转换过程中计算器不会自动换算角度数据。

总的来说，一个任意的实数函数 f 将会按在以下规则运行：

- 如果 f 仅操作一个实数 x，那么复数下的 ᶜf 将操作复数 $x_c=x+iy$。记住 x_c 会使用两级堆栈[31]。
- 如果 f 操作一个寄存器，比如 **R12**，那么 ᶜf 将操作两个寄存器，例如 **R12** 和 **R13**。
- 如果 f 操作 **X** 和 **Y** 两个寄存器，那么 ᶜf 将操作 **X**，**Y**，**Z** 和 **T**。

每当执行复数操作时，无论是 YDON 还是 YDOFF 模式，其结果的虚部都会以 i 开头显示在 LCD 的第一行中（请参阅下面的示例和第 36 页）。

[30]译者注：即使输入是实数，要输出虚部不为零的复数结果，也需要调用复变函数。

[31]请注意，x+iy 的输入类似于 2D 数据点（x, y），即 y[ENTER⇧]x（如果想按正常顺序输入复数，则需要多按一次键：x[ENTER⇧]y[x⇋y]。注意不要弄混实部和虚部）。

一元实数函数将 x 替换为 **f(*x*)**，那么一元复变函数会将 x 和 y 分别替换为结果 ^{C}**f**(x_C)的实部和虚部。而更高级别的堆栈将不会变化。这些函数有 ^{C}1/x，^{C}ABS，^{C}FP，^{C}IP，^{C}ROUND，^{C}x!，^{C}x^2，^{C}√x，^{C}+/-，^{C}Γ；10，2 和 e 为底的对数函数和指数函数，以及双曲函数和三角函数及其逆函数。

示例：

$\sqrt{-1}$的计算如下：

0**[ENTER⇧][-1]**	将-1+0i 作为复数输入，占用两层堆栈	
[CPX][$\sqrt{x}$]		
[CPX][x^2]		即回到了初始值。

二元实数函数会将 x 替换成结果 **f(*x*, *y*)**，类比可知，二元复变函数会 x 和 y 分别替换为结果 ^{C}**f**(x_C)的实部和虚部。相邻级别堆栈会被更高级别堆栈的复数填充，最顶部的两个堆栈中的复数会被复制，详见下一页。这些复变函数有 ^{C}IDIV，^{C}LOG_X，^{C}y^x，^{C}x√y，^{C}β，和 ^{C}||。详见后面的堆栈图表。

和实数一样，复数也由**[ENTER⇧]**进行分隔。

示例：

$(1+i\cdot 2)\cdot(3+i\cdot 4)$按如下步骤输入：

[2][ENTER⇧][1] [ENTER⇧] [4][ENTER⇧][3] [CPX][×]，返回：

以下步骤也是可以的：

[2][ENTER⇧][1] [CPX][ENTER⇧] [4][ENTER⇧][3] [CPX][×]，这样看起看来更有逻辑，但是需要按更多的键。

当复数操作（如：^{C}RCL）没有进行任何的堆栈输入，而只是返回一个复数，这个数字将会被压入两个堆栈。

复数计算使用两个寄存器或者两个级别的堆栈，如下所示：

	级别	赋值给初始堆栈内容	堆栈寄存器复数操作执行后堆栈内容							函数功能	
			^{C}ENTER	^{C}FILL	^{C}DROP	^{C}x⇆y	^{C}R↓	^{C}R↑	LASTx	针对单个数运算，比如 $^{C}x^{2}$	涉及两个数的运算，比如 $^{C}/$
4 级堆栈	T	$Im(y_c)= Im(t_c)$	$Im(x_c)$		$y_c = t_c$	$Im(x_c)$			$Im(x_c)$	$y_c = t_c$	$y_c = t_c$
	Z	$Re(y_c)= Re(t_c)$	$Re(x_c)$			$Re(x_c)$			$Re(x_c)$		
	Y	$Im(x_c)$	$Im(x_c)$		$Im(y_c)$	$Im(y_c)$			*last* x_c	$Im((x_c)^2)$	$Im(y_c/x_c)$
	X	$Re(x_c)$	$Re(x_c)$		$Re(y_c)$	$Re(y_c)$				$Re((x_c)^2)$	$Re(y_c/x_c)$
8 级堆栈	D	$Im(t_c)$	z_c	x_c	t_c	t_c	x_c	z_c	z_c	t_c	t_c
	C	$Re(t_c)$									
	B	$Im(z_c)$	y_c	x_c	t_c	z_c	t_c	y_c	y_c	z_c	t_c
	A	$Re(z_c)$									
	T	$Im(y_c)$	x_c	x_c	z_c	x_c	z_c	x_c	x_c	y_c	z_c
	Z	$Re(y_c)$									
	Y	$Im(x_c)$	x_c	x_c	y_c	y_c	y_c	t_c	*last* x_c	$(x_c)^2$	y_c/x_c
	X	$Re(x_c)$									

由上面的图表可知，复数域的计算中 8 级堆栈和实数计算中所用的 4 级堆栈一样灵活。请参见 *IOP* 查阅复数域下的所有函数，这些函数很多都在复数 XFCN 菜单里

可以在复数域下计算二维向量代数，可用的有 ^{C}ABS，^{C}+，^{C}-，^{C}CROSS，^{C}DOT，和 ^{C}SIGN 函数。

当按下**[CPX]**之后，*WP 34S* 允许使用这个虚拟键盘上的复数操作（但 STOPW 为实数域的程序，参见 137 页）。

记住紧接着按下**[CPX]**后，再次按下**[CPX]**，**[⇦]**，或者**[EXIT]**会退出复数模式，键盘布局随之恢复为默认布局。

在复数域的常量及其计算过程都会和一般的复数一样占用两个寄存器的位置。

复数使用 *虚部* **[ENTER⇧]** *实部* 进行输入。
纯实数，也就是虚部为 0 的数，很容易在复数计算中输入，操作如下：

[0][ENTER⇧]*实部* 或

对 0 到 9 的整数用**[CPX]** ***n*** 来输入。

在编程中，使用**[CPX]**[h][CONST] ***#n***（0≤***n***≤256）可以节约程序步数。

例如： 实数 π 可以用 **[0][ENTER⇧]**π 或者**[CPX]**π 输入，结果都是 y=0 和 x=π。

纯虚数，也就是实部为 0 的数，可以这么输入：

虚部 **[ENTER⇧][0]**。

例如：虚数单位 ***i*** 可以通过 1**[ENTER⇧][0]**或者**[CPX][.]**输入，结果都是 y=1 和 x=0。

参见上页的堆栈的工作机制。

复数的寻址

<table>
<tr><td>1</td><td>用户输入

输出</td><td colspan="4">[CPX][x=?]或[CPX][x≠?]

OP_
（在临时 alpha 模式下）
例如：'x=_?</td></tr>
<tr><td>2</td><td>用户输入

输出</td><td>[0]或[1]

OP n ?
例如：
'x=0?</td><td>堆栈等级或字母寄存器[Z]，[A]，[C]，[L]或[J]

OP? x
例如：
'x≠? Z</td><td>[ENTER⇧][32]
离开临时 alpha 模式

OP? _</td><td>[⇨]
打开间接寻址

OP? →_</td></tr>
<tr><td>3</td><td>用户输入

输出</td><td>将 x+iy 和实数 0 比较</td><td>将 x+iy 和 z+it 比较</td><td>寄存器号[0][0]…[9][9]，[.][0][0]…[.][1][5]，如果相关寄存器已经被赋值

OP? nn
例如：x≠? 26</td><td>间接寻址见下一页</td></tr>
<tr><td></td><td></td><td></td><td></td><td>将 x+iy 和在 r26+i r27 中的数值比较</td><td></td></tr>
</table>

[32]对寄存器号>19 的寄存器或本地寄存器可以跳过此按键。后者以[.]开头。请参阅后文关于编程的部分和附录 B。

<table>
<tr><td>1</td><td>用户输入

输出</td><td colspan="4">[CPX][RCL]，[CPX][STO]，[CPX][VIEW]或[CPX][x⇆]

OP_
（当临时 alpha 模式开启时）
例如：
ʳRCL _[33]</td></tr>
<tr><td>2</td><td>用户输入

输出</td><td>堆栈等级或者字母寄存器[Z][34]，[A]，[C]，[L]或[J]

OP x
例如：
ʳRCL L</td><td>寄存器号[0][0]…[9][9]，[.][0][0]…[.][1][5]，如果相关寄存器已经被赋值

OP nn
例如：
ʳSTO 18</td><td colspan="2">[⇨]
打开间接寻址，和实数的类似

OP →_</td></tr>
<tr><td>3</td><td>用户输入

输出</td><td>这是 ᶜLASTx——实部从寄存器 L 被复制到 X，虚部从 I 到复制 Y。</td><td></td><td>堆栈等级或者字母寄存器[X]，[Y]，…，[K]

OP → x
例如：ʳx⇆→Z</td><td>寄存器号[0][0]…[9][9]，[.][0][0]…[.][1][5]，如果相关寄存器已经被赋值

OP → nn
例如：ʳSTO→45</td></tr>
<tr><td></td><td></td><td></td><td></td><td>交换 x 和 z 所指向寄存器的内容，交换 y 与 z 所指向的后一个寄存器的内容。
假设有 x=4，y=3，z=20，r20=5，r21=6，那么当执行ʳx⇆→Z 后 x=5，y=6，r20=4，r21=3。</td><td>将 x+iy 存储到从寄存器 r45 开始的两个连续的寄存器之中。</td></tr>
</table>

注意：复数操作会影响一对寄存器，包括指定的那个和之后的一个。为了避免混乱，我们强烈建议存储复数的实部到偶数编号寄存器中。这一点可以通过在直接寻址中指定偶数编号寄存器做到。

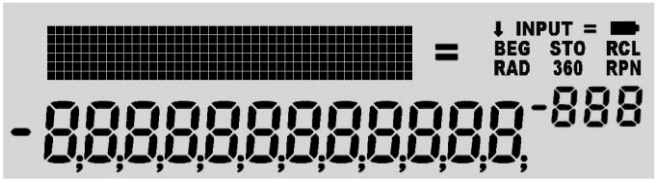
[33]对于**[RCL]**和**[STO]**，**[+]**、**[-]**、**[×]**、**[/]**中的任何一个都可以在第二步之前输入，见 *IOP*。

[34]例外的情况：ᶜRCL Z，ᶜRCL+ Z，ᶜSTO Z，和 ᶜSTO+ Z 需要在**[Z]**之前按下**[ENTER⇧]**键，对于最后一个即：**[CPX][STO][+][ENTER⇧][Z]**。

3. *WP 34S* 的各种模式

屏幕显示

液晶显示屏是 *WP 34S* 的窗口，从这里可以看到计算过程和结果。这个显示屏有 3 个部分：数字、点阵和固定符号，一共 400 个显示单元。数字部分有一个负号和 12 个数字用来显示有效数字部分[35]，以及一个负号和 3 个数字用来显示指数部分。点阵部分高 6 点，宽 43 点，基于字符宽度可以显示 7 个到 12 个字符。右上角的固定符号（大等号'='除外），称为状态指示符。



这十个状态指示符用于指示计算器所处的模式。

液晶显示屏下部的数字部分用于显示不同格式的数字、状态数据或消息的一部分。更多信息请参阅以下示例。

上面的点阵部分用于传递附加信息。

示例：

在指令输入的时候，点阵显示被选择的指令直到输入结束（即所有的后续参数都被输入）。功能键[f]，[g]，[h]，**[CPX]**和**[⇨]**会一直显示，直到他们被解除（默认按下**[⇨]**会调出[g]）。如果错误的按下了任何一个功能键，恢复十分简单，如下所示：

[f][f]=[g][g]=[h][h]=**[CPX][CPX]**=**[⇨][⇨]**=无操作（NOP）

[CPX][⇦]=**[CPX][⇨]**=**[CPX][EXIT]**=无操作

[⇨][⇦]=**[⇨][CPX]**=无操作

[g][f]=[h][f]=[f]

[f][g]=[h][g]=[g]

[f][h]=[g][h]=[h]

寻址中的操作如前文表中所述，可以通过**[⇦]**或者**[EXIT]**取消这些操作。

如果两个或更多显示请求在竞争显示空间，优先顺序如下：

1. 显示输入的指令，
2. 附录 C 里描述的错误信息，

3. 下文描述的特别信息，
4. 整数模式信息，
5. 显示 y，如果 YDON 被设置，
6. 日期模式信息。

识别计算器的模式

大部分模式和系统状态，由指示符和位于点阵或指数部分的特殊字符表示。

符号	在以下情况被设置	在以下情况被清除	备注
↓	[⇧]	[⇧]	Alpha 模式（参见 62 页）下会输入小写字母。
INPUT	[α]，αON	[ENTER⇧]，αOFF，[EXIT]	Alpha 模式。
=	见右	见右	调试模式开启（参见附录 A 和附录 H），或正在进行 I/O 操作（参见附录 D）。 在秒表运行时闪烁。 其他情况不显示。
▬	电池电量低	电压超过 2.5 伏	低电量下 *WP 34S* 会降频运行。如果电池电压低于 2.1 伏，*WP 34S* 将强制自动关机。
☰	[SF][A]	[CF][A]	
BEG	程序指针在步骤 000 或 001 处。	其他情况	无。
STO	[P/R]	[P/R]，[EXIT]	编程模式。
RCL	当某个程序正在运行时闪烁	程序停止时	无。
RAD	[RAD]	[DEG]，[GRAD]	角度的模式（参见 44 页）。
360	[DEG]	[RAD]，[GRAD]	
RPN	几乎所有的指令都会显示	一个临时消息	见 38 页，处理临时消息。
b__	[2]	设定至其他任何进位制 [a b/c]，[d/c]和 FRACT 会进入分数模式 [ALL]，[FIX]，[SCI]，[ENG]，[H.MS]，[⇨][H.MS]，[H.D] 和 TIME 将会进入为默认的浮点十进制小数模式（DECM）。	二进制整数模式。
3__ … 7__	BASE 3 … BASE 7		相应的进制模式。（所有模式见 54 页）。
o__	[8]		八进制模式。
9__	BASE 9		九进制模式。
d__	[10]		十进制模式。
-1__ … -5__	BASE 11 … BASE 15		相应的进位制。
h__	[16]		十六进制。
_c_	进位，[SF][C]	清除进位，[CF][C]	整数模式下对应的位的设置。
__o	溢出，[SF][B]	[CF][B]	
▫	[CPX]	见 35 页	临时显示的前缀符。RAD 也会

[35]即科学计数法中的尾数，为了避免歧义，本手册统一称为有效数字部分。

符号	在以下情况被设置	在以下情况被清除	备注
			随之亮起。
D	DBLON	DBLOFF	见附录 H。
f	[f]	见 35 页	临时显示的前缀符。
g	[g]		
G	[GRAD]	[RAD]，[RAD]	角度的模式（见 44 页）。
g →	[⇨]	见 35 页	临时显示的前缀符。
h	[h]		
i	复数结果	其它情形	见 29 页。
m.dy	M.DY，SETUS	任何其它时间和地区设定	日期模式（见 45 页）。注意默认的 D.MY 模式不会显示。YDON 模式下日期模式不显示。
y.md	Y.MD，SETJPN，SETCHN		
÷	进入任一菜单	退出菜单	见 117 页有关菜单的更多内容。
点阵显示的数字	YDON	YDOFF	*y* 的永久显示可能有助于跟踪堆栈。注意在任何情况下，点阵数字都将在→POL 和→REC 之后暂时出现。

默认的 DECM 模式没有指示符号。小数点和分隔符在数字显示时可见，时间模式（12h/24h）会在时间字符串中显示。分数模式的数字格式也很容易辨认，参见 41 页的示例。

某些模式和显示设置可以由 STOM 和 RCLM 批量保存和恢复，包括堆栈深度、显示对比度、完整的十进制显示设置、角度模式、日期和时间显示的选项、整数和分数模式的参数、曲线拟合模型、舍入模式和选择的舍入精度。STOM 将这些信息存储在指定的寄存器中。RCLM 调出对应的寄存器内容并相应地设置 *WP 34S* 模式。

注意：请确认确实需要调用模式设置，否则 *WP 34S* 可能会变得非常奇怪，可能需要花费大量精力才能恢复，直到发现以前的模式所存储的位置。参见附录 H。

所有的按键输入在输入完成后，根据输入时设定的模式被执行。

返回特定显示的常用指令：STATUS，VERS，ERR 等。

一些常用指令通过 LCD 提供临时消息。临时消息的定义如下：

屏幕显示的与当前模式下的 **X** 的实际内容不同的内容，或者任何附加信息，都是临时消息。此时 RPN 指示符关闭，如第 36 页所述。

如果这些额外数据在菜单或浏览器之外显示，按下任何按键它们都会消失。按**[EXIT]**或**[⇦]**仅清除临时信息并返回正常显示，此外任何其他按键都会被执行。

以下是提供特殊信息的常用指令：

1、**[STATUS]**返回关于当前内存分配和可用空间的信息。它还可显示用户标志位的设置情况。关于这个浏览器的详细说明见 119 页。

2、**[SHOW]**显示所有寄存器的内容，**[CAT]**显示所有的程序。它们都通过浏览器来显示。

3、浏览任意一个菜单，其中包含的指令将作为临时消息显示（有关详细信息，请参阅 117 页）。

4、VERS 会生成一条关于 *WP 34S* 上运行的固件版本和小版本的临时信息。

如果安装了基本固件，将显示版本号及小版本号。

如果安装了计时器固件，版本号后会显示一个“T”。

如果安装了打印固件，版本号后会显示打印机字符。

如何安装不同的固件包见附录 A。

5、ERR 和 MSG 将相应的错误以临时消息显示。参见附录 C。

6、一些影响范围较大的指令（如 CLALL）会在执行前进行确认。这时需要回答 Y（按**[R/S]**）或 N（按**[8]**）。此外，**[EXIT]**或**[⇦]**将被解读为 N，所有其它的输入都将被忽略。

还有一些返回特殊显示和临时消息的指令取决于 *WP 34S* 的特定模式，这将在后文介绍。

浮点模式-1：简介与本地化设置

浮点模式涵盖需要计算的“常用”数字：十进制实数或复数、分数、测量或计数值、时间和日期。DECM 是 *WP 34S* 的初始默认模式。许多指令专门适用于它及其子模式。

可以根据您所在的地区使用下列指令来设置显示模式：

指令	小数点[36]	时间	日期[37]	JG[38]	三位数分隔符	备注
SETCHN	RDX.	24h	Y.MD	1949（新中国成立）	TSOFF	需要四位数分隔符。
SETEUR	RDX,	24h	D.MY	1582（格里高利历出现）	TSON	SETEUR 也适用于南美洲、俄罗斯、越南、印度尼西亚和南非，但并非改历时间。
SETIND	RDX.	24h	D.MY	1752（大英帝国及其殖民地开始使用格里高利历）	TSOFF	超过 10^5 的数字每两位数需要分隔符。也适用于巴基斯坦，孟加拉国和斯里兰卡。
SETJPN	RDX.	24h	Y.MD	1873（明治改历）	TSON	
SETUK	RDX.	12h	D.MY	1752	TSON	也适用于澳大利亚和新西兰。英国已开始接受 24 小时制。
SETUSA	RDX.	12h	M.DY	1752	TSON	

[36]小数点格式的世界分布地图见 *http://upload.wikimedia.org/wikipedia/commons/a/a8/DecimalSeparator.svg*。国际标准 ISO31-0 允许点或逗号作为小数点，并且需要一个狭窄的空白作为数字组的分隔符以避免误解。但是，*HP-20b* 或 *HP-30b* 的数字显示硬件不允许使用窄空白。

[37]日期格式的世界分布地图见 *http://upload.wikimedia.org/wikipedia/commons/0/05/Date_format_by_country.svg*。国际标准 ISO8601：2004 规定时间为 24 小时，年.月日为日期格式。这种组合通常用于东亚。

[38]本栏列出了特定地区引入格里高利历（即公历）的年份，通常取代朱利安历（在东亚，为各自建国年份）。*WP 34S* 支持 1582 和 1752。请参阅 *IOP* 中的 JG1582 和 JG1752。

浮点模式-2：显示十进制小数、分数、时间等

几乎所有用于控制浮点格式的函数都在 *WP 34S* 上的前两行按钮上。

1. 对于浮点十进制小数，只要它们在显示宽度之内，初始默认显示所有数字；它会自动切换到科学计数（即有效数字加指数）以避免溢出显示区域的限制。初始默认的格式为 ALL00，SCIOVR。除了 ALL 和 SCI 之外，还有两种数字显示格式，FIX 和 ENG。使用一个例子可以很容易地展示和区分它们的效果：

格式 / 输入	ALL00	FIX4	SCI4	ENG4
107.12345678	107.12345678	107.1235	1.0712 ^{2}	107.12 ^{0}
[1/x]2**[×]**	1.8670047253 1^{-2}	0.0187	1.8670^{-2}	18.670^{-3}

在 FIX 模式中，小数点将始终保持在定义的固定位置。在其他表示方法中小数点会浮动，例如，1.0712 ^{2}表示 1.0702×10^{2}，而 18.670^{-3} V 表示$18.67\times10^{-3}V$ =18.67mV。在 ENG 中，指数始终是三的倍数，和 SI 中的单位前缀对应。

一旦数字输入完成，有效数字将靠右显示，指数靠左显示。在有效数字内，可以选择点号或逗号作为小数点，并且可以选择以每三位数进行分隔的附加标记。

假设显示设置为 FIX4。键入 12345678**[.]**901**[ENTER⇧]**，在 TSON 生效的情况下[39]，会得到：

或


。

没有这些分隔符（如 TSOFF），同样的数字看起来会是这样的：

或


。

在设置了 ENG3 后按**[+/-]**会看到：

或


。

在默认的 DECM 中可以直接输入$10^{-383}\le|x|\le10^{385}$的浮点十进制数。在此范围

[39]这些分隔符在下面介绍的分数模式中也是有用的。

内，*WP 34S* 以 16 位浮点计算，最多显示 12 位。小于10^{-398}的值将被视为 0（原因见附录 B）。对于$|x| \geq 10^{385}$的结果，除非标志位 D 被置位，否则将出现错误 4 或 5（参见附录 C。）。

复数计算结果如下显示：

其中实部在 LCD 的数码部分显示，虚部在点阵部分以 **i** 开头显示。它们都会根据选择的显示格式进行显示。注意，第一行的数据是个临时消息，且是 **Y** 中存储的虚部的一个舍入显示，其显示的位数由点阵部分的宽度限制。按下**[x⇆y]**可以查看完整的虚部，再次按下**[x⇆y]**将这个复数恢复原样，以便用于后面的计算。

[⇨P]和[R⇦]在屏幕中返回两个数。假设 x=1.2，y=3.4，按下**[⇨P]**将显示：

，

再按下[R⇦]，会显示：

。

注意两个屏幕的第一行都是临时消息，按下任意键都会清除这个附加信息。

2. 按[◀()]和[()▶]来完整地显示某个数 *x*，即标准浮点数有效数字内部的所有十六位数字，以及所有的指数数字。所有这些都显示在一条临时消息中，与 *HP-42S* 中的 SHOW 指令类似。

比如，按**[π][3][y^x]**[◀()]后会返回

，

表示数字为 3.100627668…×10^1

3. 为了使用回归曲线拟合测量和累积的数据点，提供了 *HP-42S* 中的四种数学模型。请参阅 *IOP* 中的指令 EXPF，LINF，LOGF 和 POWERF。指令 BESTF 将自动设置 *WP 34S* 选择绝对相关系数最大的模型（见 CORR）。

在与拟合相关的每个指令之后（即在 CORR，COV，L.R.，s_{XY}，$\hat{x}$，$\hat{y}$ 之后），临时显示所应用的拟合模型，如下例所示：

命令 BESTF 将自动选择模型，以获得最大的绝对相关系数（参见 CORR）。

4. 分数模式与 *HP-35S* 或 *HP-32SII* 中的模式类似。DENMAX 设置最大允许分母（见 *IOP*）。分数模式是通过[a b/c]、[d/c]或在输入数字时输入两次小数点进入的。它通过[ALL]、[FIX]、[SCI]、[ENG]、[H.d]和[H.MS]退出并返回十进制浮点模式。按下**[CPX]**时，分数模式将暂时挂起；只要还在执行复数的操作，它就会保持挂起，但完成后按下第一个键就会返回分数模式。

分数的显示如下面的示例所示（分数已调整到左侧显示）。如果分数完全等于、略小于或略大于被转换的浮点数，则分别指示=，LT 或 GT。在 *WP 34S* 上，分数模式可以处理绝对值在 0.0001 到 100000 之间数字。分母最大为 9999。下溢出和上溢出会在进入分数模式之前显示。

示例：

输入：	然后会看见：	备注：
[MODE][D][▼][▼]	÷DENANY	在模式目录中
[XEQ]		初始化
0[MODE][▼][▼][▼]	÷DENMAX	
[XEQ]		设置最大精度
3[1/x][d/c]	1/3 GT	因为 1/3>0.3333333…。
[MODE][Y][▼]	÷ YDON	
[XEQ]		
[ENTER⇧][1/x]	1/3 > 3/1 =	这是准确的。
50[.]40625	1/3 = 50.40625	
[MODE][Y]	÷ YDOFF	
[XEQ][-]	-15 17/32 =	这也是准确的。
[x²]	230 1289/1024 =	

按下[a b/c]转化成真分数[40]：

L47 36 17 1024 =

显示的第一个数字左边有一个小“钩子”。这表示这个整数未完全显示——至少有两个数字在 47 之前，但没有更多的显示空间。按[◀()]将此真分数的整数部分显示为 2247。

如果输入两个小数符号，第一个被解释为空格，第二个被解释为分数符号：

[40]与以前的 RPN 计算器相比，这是一项新功能。

输入	在真分数模式下会得到	
[1][2][.][3][.][4][ENTER⇧]	12 3/4	
[1][.][2][ENTER⇧]	1 1/2	（小数输入）
[.][1][.][2][ENTER⇧]	1/2	
[.][1][2][ENTER⇧]	3/25	（小数输入）
[1][.][.][2][ENTER⇧]	1 0/1	（=1 $^{0}/_{2}$）[41]

请注意，DECOMP 以与前文相似的方式指示其结果的精度：

输入	显示	
0.33333[H.d]		
[X.FCN][D][E]	÷DECOMP	
[XEQ]	y/x > 3	
[x⇆y]	1	表示 1/3>0.33333。

5. 有三种角度模式：DEG，RAD 和 GRAD，度数（DEG）可以用十进制数字显示，也可以以时，分，秒和百分之一秒（H.MS）显示。本机提供了以下角度转换功能：

从	角度 H.MS	十进制角度	弧度	GRAD	当前角度模式
到角度 H.MS	—	**[⇨][H.MS]**	—	—	—
到十进制角度	**[⇨][H.d]**	—	**[CONV]** rad→°	**[CONV]** G→°	**[⇨][DEG]**
到弧度	—	**[CONV]** °→rad	—	**[CONV]** G→rad	**[⇨][RAD]**
到 GRAD	—	**[CONV]** °→G	**[CONV]** rad→G	—	**[⇨][GRAD]**
到当前角度模式	—	**[X.FCN]** DEG→	**[X.FCN]** RAD→	**[X.FCN]** GRAD→	—

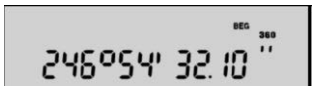
[41]这个整数的显示明确告诉您 *WP 34S* 处于真分数模式。作为对比，请注意 *HP-32SII* 读取**[1][.][.][2]**为 1/2，但这与其在分数模式下的其他输入操作不一致。

示例：

按顺序输入	然后会看到	
[RAD]		在当前的角度模式选择弧度
[π]300[/][FIX]4	0.0105	π/300
[⇨][DEG]	0.6000	等于 0.6 度
[⇨][H.MS]	0.3600	或者 0 度 36 分 0 秒

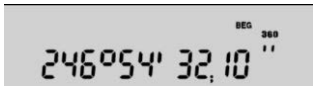
6. H.MS 显示模式通过[H.MS]输入，该模式下十进制数被转换并以hhhh°mm′ss.dd″格式显示为临时信息，小时数或度数的限制为不超过 9000。

角度在 RDX.下显示为：



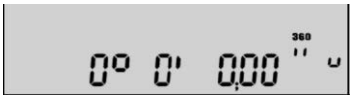
，

在 RDX,下显示为：



。

对于小于 5 毫秒或 0.005 角秒但大于零的角度，指数部分将点亮用于指示下溢的 u：



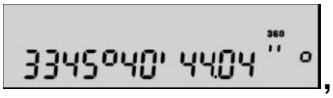
。

请注意，时，分和秒中没有前导零。

对于超过 9000 的时间或角度，在指数部分中显示 o 表示溢出，并且该值以 9000 为模数显示。

例如：

会在按了[H.MS]后变成



，

下一个按键被输入后，显示回到：

。

7. D.MY 模式下，对于类似 13.01201 的输入，WDAY 返回如下所示的显示（相当于 Y.MD 中输入 2010.0113 或 M.DY 中输入 1.13201）：

DAYS+指令也会触发同样的显示[42]。

日期将根据所选的日期模式以数字显示。如果没有选择 YDON 的话，此模式（初始默认的 D.MY 除外）将在点阵中指示，如 36 页后所示。

[42]历史上工作日的计算取决于当时使用的日历，不同国家和地区的实际结果可能不同，具体取决于特定国家和地区引入公历的日期。正式的讲，该日历于 1582 年 10 月在天主教世界生效。世界的大部分地区在花了更多的时间后切换到了公历（见第 41 页）。然而，请注意，在这个星球上还有其他日历，例如在穆斯林世界中的日历。

由于之前闰年规则的应用不一致，在公元 8 年之前的日期可能与当时应该的样子不同。我们指望您能理解，并希望这个缺点不会影响太多的计算。

请注意，8.A.D.最好写成 A.D.8，或者 A.D.VIII，这样更准确。英语中有相当一部分的错误拉丁语。然而，当时没有人这样计算年份——在地中海附近，当时是 DCCLXI A.U.C.（罗马建城 761 年）。另请注意，朱利安日历被引入生效的时间不早于 DCCVIII A.U.C.（罗马建城 708 年）——在这之前，月份是不同的。朱利叶斯·凯撒在 DCCIX A.U.C.（罗马建城 709 年）被谋杀；历法是一个敏感的话题。

浮点模式-3：概率和统计函数

除了基本的[%]和[Δ%]之外，*WP 34S* 中还内置了很多统计指令。它们都集中在图中所示的浅绿色框中。其中许多指令是在首次在 RPN 计算器中实现。

数字[5]下面及其右边的标签囊括了样本统计功能，左边的六个标签为常见的概率函数。

[!]、[Cy, x]和[Py, x]分别是阶乘，组合和排列。

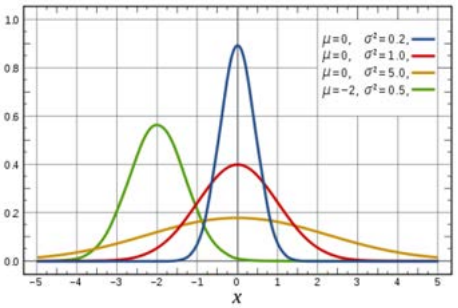
[4]的第二、第三功能为统计分布：标准正态[Φ]及其反函数[Φ^{-1}]，[PROB]菜单里面包含更多连续分布函数及三个离散分布函数。这些函数具有以下特征：

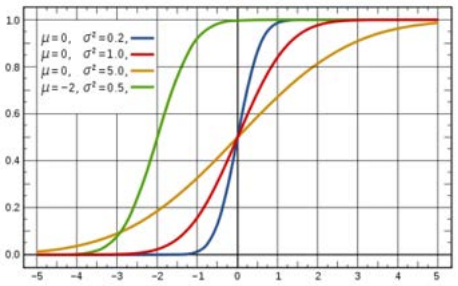
- 离散统计分布只对整数有定义。累积概率质量函数（*pmf*[43]）$p(n)$得到累积分布函数（*cdf*）$F(m)$的计算是从 n=0 开始的，因此：

$$F(m)=\sum_{n=0}^{m} p(n) = P(m)$$

- 当 *WP 34S* 计算积分的时候，从积分下限开始计算。因此对连续概率密度函数（*pdf*）$f(x)$积分得到累积分布函数 $F(x)$是如下计算的：

$$F(x)=\int_{-\infty}^{x} f(\xi)d\xi = P(x)$$





许多经常使用的 *pdf* 和 *pmf* 看起来或多或少类似于右侧上面的图。下面的图用相同的颜色显示了相应的 *cdf*。通常情况下，任何 *cdf* 开始时都是一个很缓的斜率，然后变得更陡，然后随着斜率的减小而逐渐接近 100%。即使实际的 *pdf* 看起来不像这里的示例那么漂亮和对称，这一规律也适用。

[43]离散统计分布用于处理由已知数学模型决定的事件。这类事件可能是：进入商店的人数，原子核的衰变，故障零件的个数等。*pmf* 给出了观察到一定数量（例如 7 次）该事件的概率。*cdf* 给出了观察到 7 次及以下这个事件的概率。

连续变量的统计，如三岁儿童的身高，也是类似的：若数学模型已知，*cdf* 给出他们身高在某个限值之下的概率，比如小于 1 米的概率。而 *pdf* 给出的是这些身高在样本（比如 1000 个儿童中）是如何分布的。

以上只是关于这个概念的粗略解释，准确解释请参阅统计学相关的书籍。

显然，在 *cdf* 的左侧使用 P 可以得到最精确的结果。在其右侧，在 P 接近 1 处，错误概率 Q=1-P 更精确。因此，*WP 34S* 在计算 P 的同时，也为每个分布计算 Q。

在 *WP 34S* 上，*cdf* 用 ***XYZ*** 表示，此外有以下几种名称：

XYZ$_u$：表示误差概率 Q（也称为上侧尾概率）；

XYZ^{-1}：表示 *cdf* 的逆函数（即分位数函数或 *qf*）；

XYZ$_P$：表示 *pdf* 或 *pmf*。

以上命名约定适用于二项、柯西（又称洛伦兹或布雷特-维格纳）、指数、费雪 F、几何、对数正态、逻辑斯蒂、正态、泊松、学生 t 和威布尔分布。考虑到命名传统，卡方分布和标准正态（高斯）分布使用不同的命名规则。请参阅 PROB 菜单和 *IOP* 中的相应条目。如果有兴趣，可以在附录 I 中找到一些数学背景知识。

此外还有大量用于样本和总体统计数据的指令，这些指令可以在一个或两个维度上使用。所有这些都位于键**[4]**右边。用**[CLΣ]**清除求和寄存器后，使用**[Σ+]**（或键盘上的左上角的快捷键**[Σ+]**）输入实验数据——通常为统计或测量值，如同在 *HP-42S* 等机器上一样；加权数据要求权重在 **Y** 中给出，成对数据或数据点坐标必须像往常一样在 **X** 和 **Y** 中提供。用**[Σ-]**可以很容易地进行数据修正。

关于分析功能，有算术平均**[$\bar{x}$]**，标准偏差**[s]**，预测函数**[ŷ]**和相关系数**[r]**（见 CORR）；四个不同的回归模型（线性函数、指数函数、对数函数和幂函数——见 LINF，EXPF，LOGF 和 POWERF 指令）。更多的功能（如加权数据的标准误差、协方差、均值和标准差，总体的标准偏差，曲线拟合参数，几何平均和分布因子）在 STAT 菜单中，所有累积的数据在 SUMS 菜单中——可以在这些菜单中查看这些指令，然后在 *IOP* 中查找相应的条目。

要了解统计功能所能做到的和其固有的一些约束，请参阅下面的示例应用：

应用 1：假设您有一个小工具厂，您想知道生产的零件的质量。您选取了一个零件的典型样本，并精确测量了它们的实际尺寸。您怎么知道您的那批货是合格的?

示例：从一个生产批次中抽取 10 个大头针，测量到的直径如下：12.356、12.362、12.360、12.364、12.340、12.345、12.342、12.344、12.355 和 12.353。从先前的调查已知大头针直径是服从高斯分布的。

您想知道：

1. 后面生产出的针的直径将会是多少?统计数据不能给出全部大头针的直径，但是它会给出几乎所有的（例如 99%）。

示例（续）：重置求和寄存器，并累计 10 个测量值：

[CLΣ]

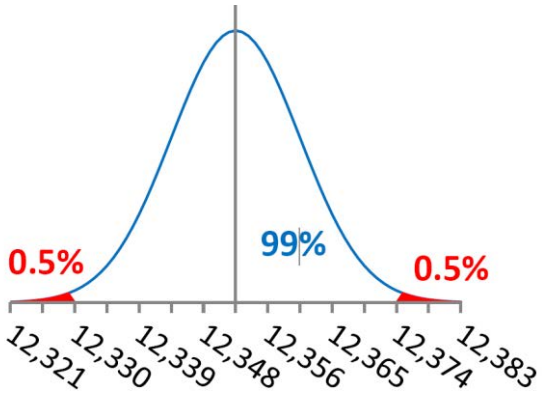
[FIX]03

12.356**[Σ+]**	1.000
12.362**[Σ+]**	2.000
12.360**[Σ+]**	3.000
12.364**[Σ+]**	4.000
12.340**[Σ+]**	5.000
12.345**[Σ+]**	6.000
12.342**[Σ+]**	7.000
12.344**[Σ+]**	8.000
12.355**[Σ+]**	9.000
12.353**[Σ+]**	10.000

这些针的直径服从高斯分布，按下以下按键，可以得到这批产品的均值和标准差的最佳估计：

[x̄][STO][J]	12.352
[s][STO][K]	0.009

存储这两个值为下一步做准备。



根据分析的十个样本，可以预期 0.5%的大头针直径小于

0.005[PROB][O][▲] ÷Norml⁻¹ [XEQ] 12.330

另外还有 0.5%的大头针直径大于

0.995[PROB] ÷Norml⁻¹ [XEQ] 12.375

2. 这批货的平均直径是多少?低于哪个值的概率是 95%[44]?下面先确定可用的样本平均值及其偏差的大小，然后再用它们计算这个限值。

示例（续）：对于高斯分布，算术平均值是适用的。偏差由标准误差给出。限值的计算还需要用到学生 t 分布。

[44] 95%这个值称为这个计算的置信度。99%这个值也经常被使用。

[SUMS]	÷nΣ	[XEQ]	10.000
1[-][STO][J]			9.000
[STAT][S][▼]	÷ŝERR	[XEQ]	0.003
0.95[PROB][U][▲]	÷t-1(p)	[XEQ]	.833
[×]			0.005
[$\bar{x}$]			12.352
[x⇆y][R⇩][45][+]			12.357

因此，有95%的概率产品的平均销直径小于12.357的置信限值。但也有100%-95%=5%的概率大于这个值。

3. 这批大头针平均直径介于哪个上限和下限的概率是95%?因为95%区间之外两侧区间的概率都是 0. 025，所以用 0. 025 和 0. 975 作为随后两次计算中 *qf* 的参数，以同时得到小于和大于限值的采样结果。

示例（续，假设在前面的步骤中加载了 J）：

[STAT][S][▼]	÷ŝERR	[XEQ]	0.003
0.025[PROB][U][▲]	÷t-1(p)	[XEQ]	-2.262
[×][|x|]			0.006
[$\bar{x}$]			12.352
[x⇆y][R⇩][x⇆y]			0.006
[-]			12.346
[RCL][L]			0.006
2[×][+]			12.358

下限
Last x
上限[46]

可以预期这批货物的未来平均直径在 12.346 和 12.358 之间，有 95%的置信度。这意味着它小于下限的几率是 2.5%，大于上限也是同样的几率。

在你只有一个样本（通常是从一个总体中抽取的一小部分有限样本）的信息，但又要知道总体的信息的情况下，这个几率是不可避免[47]的结果。如果不能接受这些几率或者置信度的宽度，不要责怪统计学，而是应该收集更多（或更精确）的数据。

[45] [$\bar{x}$]返回两个数字： $\bar{x}$ 和 $\bar{y}$ 。在这个例子中只有 $\bar{x}$ 是我们感兴趣的。按下**[x⇆y][R⇩]**移除 $\bar{y}$ 。在程序中，**[x⇆y]**后再用 DROP 是一个更好的选择。

[46] 由于 $t^{-1}(p)$是对称的，因此可以用这种简单的方法计算上限。

[47] 统计学家称这些概率为“I 型错误的概率”或“第一类错误的概率。

应用 2：假设在第一天从生产线中抽取了一个样本。然后改变了工艺参数，等待稳定，现在在第二天取了另一个同样大小的样本。严格起见，已经彻底测量并记录了这两天所调查的每个样品的临界值（例如，特征尺寸）。现在两个样本的结果是否有显著差异?按照下面简单的三步测试[48]操作就可以了：

1. 计算两个样本的 $\bar{x}$ 和标准误差 S_E，然后计算它们的归一化距离 $d=\frac{|\bar{x}_1-\bar{x}_2|}{\sqrt{S_{E1}^2+S_{E2}^2}}$。在四堆栈模式下，这个计算可以如下进行：

[STAT][S][▼]	÷ΣERR	
[XEQ]		两个标准误差放入 **X** 和 **Y**。
[x^2][x⇆y][x^2][+][$\sqrt{x}$]		这是完整的分母。
[$\bar{x}$][-][|x|][x⇆y][/][STO][D]		这是 d

接下来两个步骤计算自由度：

[SUMS]	÷nΣ	
[XEQ]		调出样本数量
1[-]		计算自由度
[STO][J]		存起来用于下一步的计算。

2. 让 *WP 34S* 计算概率是为 97.5%，自由度为 f 的学生 t 分布的临界极限 t_{cr}：

0.975	0.975	
[PROB][U][▲]	÷t-1(p)	如前所述，需要的概率函数在 PROB 目录中
[XEQ]		执行这个函数，用存储在 **J** 的自由度得到 t_{cr}。

如果 $d \geq t_{cr}$，那么测试表明两个样本之间的差异仅仅是由于随机偏差引起。祝贺您——您已经得到了一个不受参数影响的稳定的工艺流程。

3. 让 *WP 34S* 计算概率是为 99.5%，自由度为 f 的学生 t 分布的临界极限 t_{cs}：

0.995	0.995	
[PROB]	÷t-1(p)	注意打开菜单后这个函数立即出现
[XEQ]		得到 t_{cs}。

如果 $d \geq t_{cs}$，那么测试表明两个样本之间存在显著差异。参数的更改造成了影响。

[48]这个测试可以追溯到德国 DGQ 公司（Deutsche Gesellschaftfu rQualitat）。它假设数据来自高斯过程，这是现实生活中经常发生的情况（但需要检查）。请注意，“显著”一词在统计学中有明确的定义——这个定义可能偏离了通用语言。

一般来说，标准的置信度限制和水平，也就是那些用来表示显著差异的定义，可能取决于您所在的国

但是，如果 $t_{cr} \le d \le t_{cs}$，则还不能根据所提供的信息做出决定——样本可能包含的数据太少，或者测量不够精确，或者过程分散得太远。

我们强烈建议阅读一本好的统计教科书，以获得关于一般统计方法、使用的术语和数学模型的更多信息。

家或行业。一定要检查适用的有效标准。

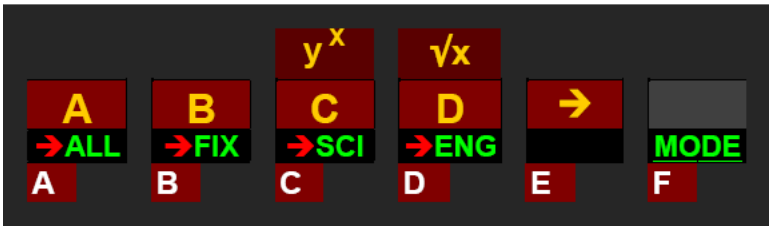
整数模式-1：介绍和虚拟键盘

整数模式的输入如第 36 页所述。整数模式被设计成专门用于输入、输出和计算整数。这对计算机逻辑和系统编程是很有用的，正如 *HP-16C* 所擅长的那样。*WP 34S* 不仅包含了 *HP-16C* 中的所有函数，而且还有所增加，并且支持从二进制到十六进制的十五种进制的整数计算（请参阅下一页）。

在整数模式中，像 SIN 之类的函数显然是没有意义的。因此，对于小于或等于 10 进制的整数，*WP 34S* 的虚拟键盘将会如左图所示（调用红色箭头开头的标签会退出整数模式，一般会返回浮点十进制小数模式）。

另一方面，在 16 进制时，最上方的六个键的主功能将会自动变成直接的数字输入，如下图所示。

任何时候，在按键功能重新调整后，原本默认的主函数不再是主函数时，按下前缀键[f]将允许使用原来默认的那个主函数（例如，在这里[f][D]将会唤出[√x][49]）。为了简化操作，按下任意键（或者一系列前缀和一个键）时，将会在顶部显示它当前的指令以便检查。如果按住最后输入的一个键长于 0.5 秒，显示将会返回到 NULL，并且不会执行任何操作。



在 11 到 15 进制的计算中，那些不需要作为数字输入的按键功能如一张图所示。在任何进制中，输入无效数字（如在二进制中输入 4）将会被阻止。

[49]在这种情况下，按键上印制的金色的指令不能被调用。这里的意思是，例如按键[D]，我们不能在十六进制模式下访问[TAN]——当然，这里并没有什么损失。注意，如果按键[D]已经被自定义了，[f][D]可能会调用一个程序。

整数模式–2：显示整数数字

在整数模式中，屏幕数字部分的有效数字部分显示 **X** 中的整数。这里的数字显示不需要指数部分，它被用来传达其他信息。指数部分的第一个数字及其符号表示进制模式：

进制模式	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
指数部分的第一个字符	b	3	4	5	6	7	o	9	d	-1	-2	-3	-4	-5	h

进位（carry）和溢出（overflow）（如果发生了）分别在指数部分的第二个位置显示为 c 和在第三个位置显示为 o。那么在 12 进制时，指数部分将会看起来像是-2co。一般来说，进位和溢出的显示就和 *HP-16C* 一样。注意它们是标志位——如果想要单独设置、清除或者检查它们，需要使用 *WP 34S* 的 flag 指令。

在点阵屏中，使用格式 xx.ww 来表示字长和补码设置。其中 xx 用 **1c** 和 **2c** 分别代表反码（1's complement，1COMPL）模式和补码（2's complement，2COMPL）模式，**un** 代表无符号（unsigned，UNSIGN）模式，**sm** 代表原码（sign-and-mantissa，SIGNMT）模式。初始的缺省设置是 **2c**。这些模式控制着负数的处理。下例有助于理解上述内容：

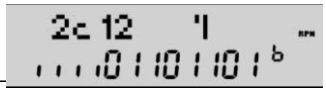
把 *WP 34S* 设置成 WSIZ 12，LZON。此设置允许在 *WP 34S* 中轻松查看所有的 12 位字符。输入 147，然后转到 1COMPL，BASE2。将会看到：

——按下**[+/-]**之后——


这里暂时忽略在右上角的'1'（稍后解释）。为了方便辨识，较低有效位的数字比较高的四个有效位的数字显示的大一些。

显然在 **1c** 中使用**[+/-]**按钮和**[NOT]**按钮是等效的。

现在通过**[+/-]**按钮返回原来的数字，使用 2COMPL 将会看到：

——点击了**[+/-]**之后——


请注意在 **2c** 中负数等于原数取反加 1。

现在通过**[+/-]**按钮返回原始数字，使用 SIGNMT 将会看到：

——点击了**[+/-]**之后——


请注意在 **sm** 中改变一个数的符号，只会让第一个位翻转（0 变成 1 或者 1

变成 0）。

最后，通过**[+/-]**按钮返回原始数字，使用 UNSIGN 将会看到：

——点击了**[+/-]**之后——

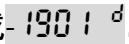

注意，第二个数字看起来和 **2c** 的一样，但这里多了一个溢出。

这里解释一下，根据定义改变符号在 **un** 设置下没有任何意义[50]，最高有效位的变化的是数字的范围，而不是符号。所以，12 位能表示的最大数值是 4095 而不是 2047。因此，在这里使用**[+/-]**本应是不允许的，或者至少不做任何操作。不过，在 *HP-16C* 中，无符号模式（**un** 模式）可以使用并且响应**[+/-]**键。为了向前兼容，我们也遵循这一特性。

因此，在这里按下**[+/-]**并不会返回原来的数；要清除溢出标志，还需要使用 **[CF][B]**（见 24 页）。

正如前文所看到的，在这四种模式中正数都保持不变。另一方面，负数有不同的表示方式。因此，在一种模式下取一个负数并切换到另一种模式会有不同的解释。比如，在固定位数模式下，默认的 **2c** 模式中的 

在 **1c** 模式中会显示成 

，在 **sm** 模式中会显示成 

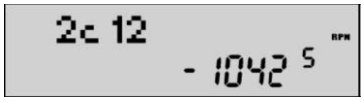
，在 **un** 模式会显示成 

。

保持现有模式（比如 **2c**），改变进制也会使得固定位模式数据的显示发生变化。4、8 和 16 进制下显示将与上面所示类似，显示所有 12 位，而在其他进制下显示一个带符号的有效数字。

作为对比， 

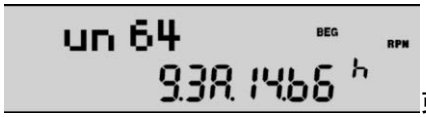
在各种进制下的输出如下：

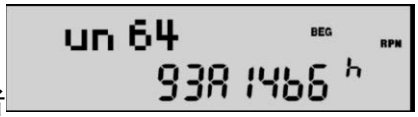


显示的格式考虑到了 2、4、8 和 16 进制对于位和字节操作更为方便，以及和硬件相关的应用最为紧密。另一方面，在 10 进制是共同的标准的情况下，处理不同的数字表达方式及其计算，也可能会对中间的进制感兴趣。

[50]这一点在 1982 年 4 月版的 *HP-16C Computer Scientist Owner's Handbook* 中有明确的陈述。但不幸的是，他们并没有将这个坚持到底。

现在让我们看更大的数：例如，设置 UNSIGN，LZOFF，WSIZE64，BASE16，输入 93A14B6。然后 *WP 34S* 将会根据是否开启分隔符（见 SEPON 和 SEPOFF）显示：



或者


这个数字需要 28 位二进制表示，即 1001.0011.1010.0001.0100.1011.0110。现在返回 SEPON，选择以 BASE 2，12 个最低有效位将会显示出来：

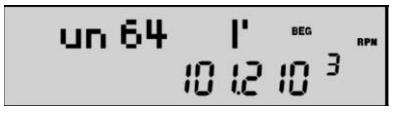
它们与指示符"'"I一起显示，这个指示符表示这个数分四屏显示，现在显示的是最右的那一屏。如上所示，最低有效字节用较大的数字显示。按下[◀()]键，会看到更高的有效字节（注意，总是有 4 位与前面的显示部分重叠）：

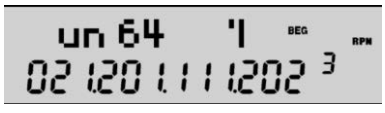
最后一个屏幕显示了这个二进制数字的四个最高有效位，屏幕上显示的I'"也证实了这一点。

如果前导零被打开（参见 LZON），这里将有 8 个显示窗口（对应 8 个字节），其中 4 个“最高有效”字节只包含 0。

注意，在所有的进制模式中，数字输入被限制为 12 位。

以八位数（即按字节）为单元浏览一个大整数仅限二进制模式。在任何其他进制中，每一个显示单元都是完整的屏幕显示宽度，即 12 位数没有任何重叠。例如，同一数字在 3 进制下最高有效部分和最低有效部分：

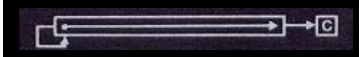




整数模式-3：位操作

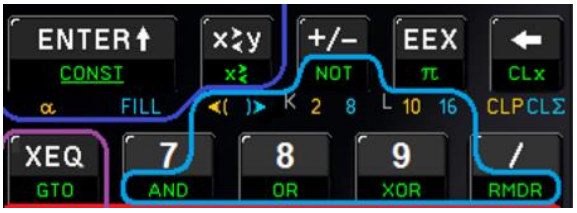
WP 34S 有 *HP-16C* 的几乎所有位操作功能，并且能实现更多的位操作。通常，位从右到左算起，第 1 位为最低有效位。这对于在 BC？，BS？，CB，FB 和 SB 操作中指定位数很重要。

为了便于阅读，下面的例子使用了显示前导零的8位字（WSIZE8，LZON）。对于下面这7个功能（全部在[X.FCN]中），可以在图中找到示意图，它们和*HP-16C*的后面板上打印的一样。方框内的“**C**”代表进位位。*x*初始值为 10110011 b。

操作	示意图	示例	输出
左移位		SL 1 SL 2	01100110 bc 11001100 b
右移位		SR 1 SR 2 SR 3	01011001 bc 00101100 bc 00010110 b
算术右移		ASR3	反码和补码模式 11110110 b 无符号模式 00010110 b 原码模式 10000110 b
循环左移位		RL 1 RL 2	01100111 bc 11001110 b
循环右移位		RR 1 RR 2 RR 3	11011001 bc 11101100 bc 01110110 b
带进位循环左移		RLC 1 RLC 2	01100110 bc 11001101 b
带进位循环右移		RRC 1 RRC 2 RRC 3	01011001 bc 10101100 bc 11010110 b

注意上面算术右移的图片只正确地描述了反码和补码模式时的情况。然而在 *HP-16C* 的所有模式中，ASR3 等于被 2^3 有符号相除，因此后面两种模式的不同结果如上所示。其他位操作不因补码模式设置不同而产生不同结果。有关这些操作的详细信息，请参阅 *IOP*。

现在看看双变量函数。其中，布尔运算符 AND（与），OR（或），XOR（异或）可以在这里画的蓝色框内找到：



我们再一次使用 8 位字符来举例说明：

输入	X	01101011 b
	Y	10111001 b
操作		输出
[AND]		00101001 b
NAND		11010110 b
[OR]		11111011 b
NOR		00000100 b
[XOR]		11010010 b
XNOR		00101101 b

请参阅 *IOP* 以了解这些整数模式下的位操作指令和更高级的指令（NOT，LJ 和 RJ，MASKL 和 MASKR，MIRROR，RAN#，nBITS）。如果不在键盘上，到目前为止提到的指令都可以在整数模式下 X.FCN 菜单中找到。在 TEST 菜单中还有 BS? 和 BC? 。

最后，请注意这些操作不会产生溢出。进位标志位只能通过如上所示的移位或者循环移位函数来置位。而算术右移是唯一对编码模式设置敏感的位操作——算术右移和整数算术运算有关。

整数模式-4：整数算术

四个基本的算术运算（+、-、×和/）的前三个运算在整数模式和在 DECM 中一样，但在二进制模式中最高只有 64 位精度。+/-相当于乘以-1 的乘法，而 y^x 就是重复相乘。根据输入参数和模式设置，操作中将自动设置溢出或进位标志（请参阅本节末尾）。

但是，在整数模式中，必须以不同的方式处理除法，因为这里的结果不能包含小数分。一般地，公式$\frac{a}{b} = (a \operatorname{div} b) + \frac{1}{b} \cdot \operatorname{rmdr}(a; b)$是适用的，横线表示实数除法，div 表示整除，rmdr 表示整数除法的余数部分。虽然正数的余数很简单，但负的被除数或除数的余数有时会引起混淆。不过，上面的公式很容易用于计算这样的余数（也适用于实数——参见本例的第一行）：

示例：$\frac{25}{7} = 3 + \frac{1}{7} \cdot 4$　　（对于实数：$\frac{25}{7.5} = 3 + \frac{1}{7.5} \cdot 2.5$）

$\frac{-25}{7} = -3 + \frac{1}{7} \cdot (-4) \Rightarrow$　　$\operatorname{rmdr}(-25; 7) = -4$

$\frac{25}{-7} = -3 + \frac{1}{-7} \cdot 4 \Rightarrow$　　$\operatorname{rmdr}(25; -7) = 4$

$\frac{-25}{-7} = 3 + \frac{1}{-7} \cdot (-4) \Rightarrow$　　$\operatorname{rmdr}(-25; -7) = -4$

在一般情况下，$a - b \cdot (a \operatorname{div} b) =: \operatorname{rmdr}(a; b)$

还有一个函数做了几乎相同的事情，它被称为 mod。用上面同样的数字，它会返回：

$\operatorname{mod}(25; 7) = 4$
$\operatorname{mod}(-25; 7) = 3$
$\operatorname{mod}(25; -7) = -3$
$\operatorname{mod}(-25; -7) = -4$

因此，如果两个参数有相同的符号，mod 返回的值与 rmdr 相同。mod 的一般公式比上面的公式复杂一点：

$a - b \cdot \operatorname{floor}\left(\frac{a}{b}\right) =: \operatorname{mod}(a; b)$

其中，$\operatorname{floor}\left(\frac{25}{7}\right) = 3$，$\operatorname{floor}\left(-\frac{25}{7}\right) = -4$。

顺带一提，这个公式也适用于实数。因此，它可以直接用于计算，例如：$\operatorname{mod}(25.3; -7.5) = 25.3 - (-7.5) \cdot (-4) = -4.7$。

上面提到的这四个函数在 *WP 34S* 中被称为 IDIV、RMDR、MOD 和 FLOOR。

此外，指数和对数运算，x^2 和$\sqrt{x}$，x^3 和$\sqrt[3]{x}$，COMB 和 PERM 也可以在整数模式下工作。除了以上提到的键盘功能和操作，请参阅 X.FCN 菜单中的更多功能，比如×MOD 和^MOD。请参阅 *IOP* 以获得它们的更多细节。

整数算法中会有涉及溢出或进位的情况。注意，在每个字长和补码设置下，都有一个最大值和最小值——我们称它们为 I_{max} 和 I_{min}。

例如：

- 在 un04 中，I_{max}=15，I_{min}=0
- 在 2c04 中，I_{max}=7，I_{min}=−8
- 在 1c04 和 sm04 中，I_{max}=7，I_{min}=−7

现在从 1 开始，逐个地递增，看看会发生什么：

UNSIGN		2COMP		1COMP		SIGNMT	
0001 b	1 d	0001 b	1 d	0001 b	1 d	0001 b	1 d
0010 b	2 d	0010 b	2 d	0010 b	2 d	0010 b	2 d
…	…	…	…	…	…	…	…
0111 b	7 d	0111 b	7 d	0111 b	7 d	0111 b	7 d
1000 b	8 d	1000 bo	-8 do	1000 bo	-7 do	1000 bco	-0 dco
1001 b	9 d	1001 b	-7 d	1001 b	-6 d	0001 b	1 d
…	…	…	…	…	…		
1110 b	14 d	1110 b	-2 d	1110 b	-1 d		
1111 b	15 d	1111 b	-1 d	1111 b	-0 d		
0000 bco	0 dco	0000 bc	0 dc	0001 bc	1 dc		
0001 b	1 d	0001 b	1 d				

从 1 重新开始，逐个递减：

UNSIGN		2COMP		1COMP		SIGNMT	
0001 b	1 d	0001 b	1 d	0001 b	1 d	0001 b	1 d
0000 b	0 d	0000 b	0 d	0000 b	0 d	0000 b	0 d
1111 bco	15 dco	1111 bc	-1 dc	1110 bc	-1 dc	1001 bc	-1 dc
1110 b	14 d	1110 b	-2 d	1101 b	-2 d	1010 b	-2 d
…	…	…	…	…	…	…	…
		1001 b	-7 d	1001 b	-6 d	1110 b	-6 d
		1000 b	-8 d	1000 b	-7 d	1111 b	-7 d
		0111 bo	7 do	0111 bo	7 do	1000 bo	-0 do
		0110 b	6 d	0110 b	6 d	1001 b	-1 d

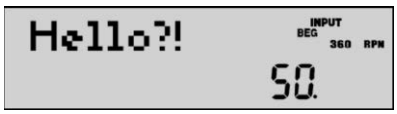
注意在 SIGNMT 中最高有效位是#3，其他模式中是#4，减法实际上是加法。

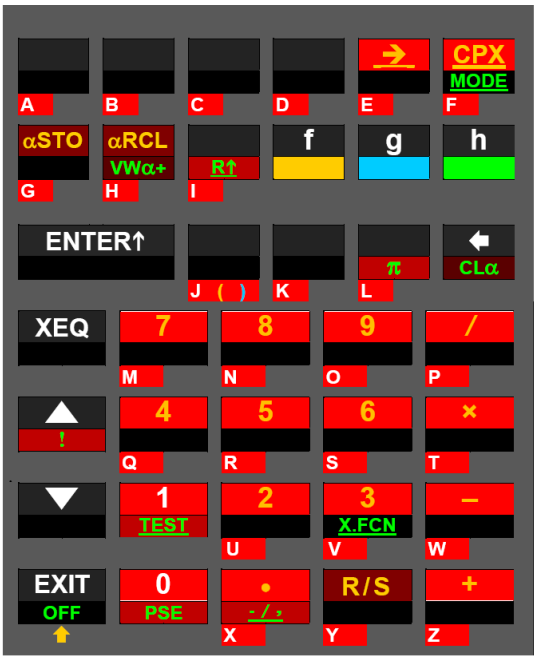
通过 I_{max} 和 I_{min}，设定和清除进位和溢出的一般规则如下：

操作	对进位的影响	对溢出的影响
+，RCL+，STO+，INC 等	如果有一个最高有效位进位，对 **c** 置位，否则清除 **c**。	如果结果超出了[I_{max}; I_{min}]，对 **o** 置位，否则清除 **o**。
-，RCL-，STO-，DEC 等	在减法 m-s 中，以下情况对 **c** 置位： ● 在 1COMP/2COMP 中，二进制减法引起最高有效位借位； ● 在 UNSIGN 中 m 小于 s； ● 在 SIGNMT 中 m 小于 s 且两者符号相同。 其他情形清除 **c**。	如果结果超出了[I_{max}; I_{min}]对 **o** 置位，否则清除 **o**。
×，RCL×，STO×，+/-，(-1)^x，x^2，x^3，LCM，x! 等	无。	如果结果超出了[I_{max}; I_{min}]对 **o** 置位，否则清除 **o**。在 UNSIGN 中，+/-和 x 为奇数的$(-1)^x$总会对 **o** 置位。
DBL×	无。	清除 **o**。
/，RCL/，STO/，DBL/，LB，LG，LN，LOGx，$\sqrt{x}$，$\sqrt[a]{x}$，$\sqrt[x]{y}$	如果余数≠0 对 **c** 置位，否则清除 **c**。	清除 **o**（但是在 2COMP 中的除法 I_{min}/(-1)会置位）。
2^x	如果 ***x***=-1，或者在 UNSIGN 中 ***x***=wsize，或者在其它模式下 ***x***=wsize-1，对 **c** 置位；否则清除 **c**。	如果结果＞I_{max} 对 **o** 置位，否则清除 **o**。
y^x，10^x	如果 ***x***＜0（和 0^0）对 **c** 置位，否则清除 **c**。	
e^x	如果 ***x***≠0 对 **c** 置位，否则清除 **c**。	
ABS	无。	清除 **o**（但是在 2COMP 中 ***x***=I_{min} 时会置位）

完整的 alpha 模式-1：介绍和虚拟键盘

Alpha 模式用于输入字符，比如程序提示的输入或者要求的回答。Alpha 模式通常通过[α]进入。在这种模式下，alpha 寄存器的内容显示在 LCD 的上部。所有的直接输入进入 alpha 寄存器，而数字行（从上次计算中保留下来的）只能通过指令访问。Alpha 模式的显示可能是这样的：





在 alpha 模式下，大多数数学运算既不需要也不适用。所以当进入 alpha 模式时键盘会自动重新分配。

所有印在红色背景上的标签都直接或应通过 alpha 菜单和 ***alpha*** 对应。注意在这个模式下有四个新菜单会被激活。见 *WP 34S* 上的**[⇨]**和**[CPX]**键，[R⇧]和[./,]标志

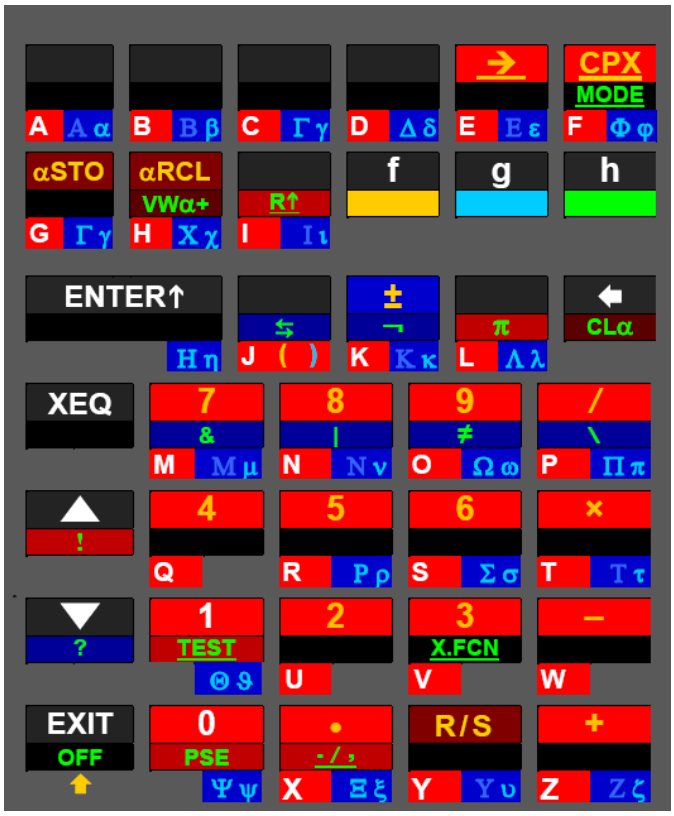
那些印在深红色背景上的标签以其他方式改变了它们的功能。见**[STO]**，**[RCL]**，**[R/S]**键和[VIEW]，[CLx]上的标签。

在 alpha 模式下，大多数按键的主要功能对应的是其左下方的字母——在键盘上是灰色的。[PSE]用于输入空格。当 alpha 超过 30 个字符时，最左边的字符会被丢弃。alpha 模式默认字母大写，并且[⇧]用来切换大写和小写。在整数模式下，[f]能在必要时访问默认的主要功能[51]。

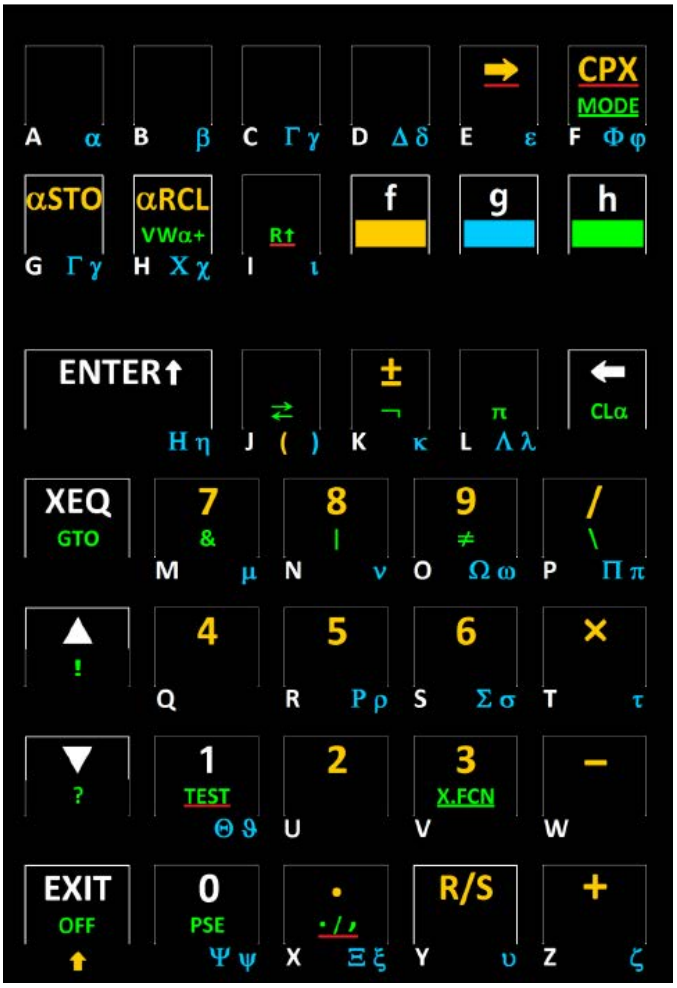
上图仅为键盘的标准标签。在这种模式下，我们还可以安全地提供更多的字符。下一页的虚拟键盘上，所有深蓝色背景上的标签也可以在 alpha 中添加字符。它们与印在 *WP 34S* 键盘上的标签有关，但与它们不同。

前缀[g]一般可调用同音异义的希腊字母。[h]还可以通过布尔操作调用逻辑符号。

[51]数字 0 和 1 可分别用[f]**[0]**和[f]**[1]**输入。



Alpha 菜单可以通过 [f][⇨]，[f][CPX]，[R⇧]，[./,] 唤出。并且[TEST]可以唤出更多字符（见 122 页）。查看 *IOP* 以获得有关 αSTO，αRCL，VWα+ 和更多 alpha 指令的信息。



左图是去掉多余的颜色和冗余的虚拟键盘。

红色下划线的标签调用 alpha 菜单。在绿色下划线的和较粗白色边框的标签不会直接向 alpha 添加字符，但所有其他的标签可以。

如果按照如左边图的大小打印这个虚拟键盘，可以将它贴到 *WP 34S* 的背面，以便随时参考。这对学习希腊字母和它们与拉丁同音异义词的关系也很有帮助。

完整的 alpha 模式-2：显示文本

WP 34S 可以显示大小两种字母数字字体。这两种字体都是基于 Luiz Viera（巴西）2004 年发行的字体。并且添加和修改了一些字母以提高可读性，这也是考虑到了屏幕的点阵只有 6 个像素高的限制。

以下是所有能直接通过虚拟键盘输入的字符：

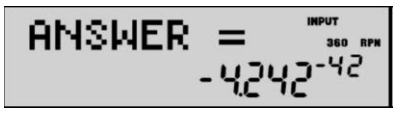
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
αβγδεζηθικλμνξοπρστυφχψω
0123456789 ()+−×/±.!?↳¬\&|≠

如果大字体的字符串超过了屏幕可见范围，*WP 34S* 将自动采取以下小字体以显示尽可能多的字符：

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
αβγδεζηθικλμνξοπρστυφχψω
0123456789 ()+−×/±.!?↳¬\&|≠

可以在 alpha 目录中找到这两种字体的更多字符。

一旦进入 alpha 模式，只要没有退出，alpha 寄存器的内容（简称为 ***alpha***）就会在点阵中显示，并显示出其最右端（即包含的最后一个字符），而数字部分保持上一次的数字操作的结果。显示内容看起来可能是这样的：

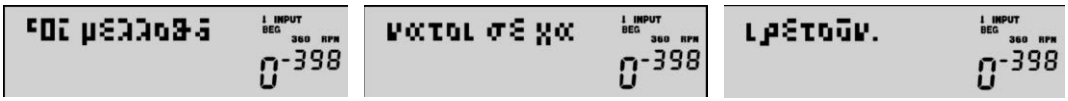


注意，除了点阵外，还可以使用标志位 A 控制的大 "=" 用于消息显示。***alpha*** 中可以添加各种不同的信息。见 *IOP* 中以 α 起始的指令。

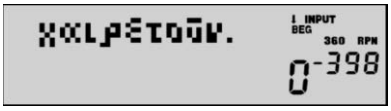
例如，TIME、αTIME 可以在消息中添加时间信息，根据时间模式和实际时刻的不同，生成类似 `It's 11:54:32 PM` 或 `Um 23:54:32 Uhr` 这样的文字。

根据不同的日期格式设置，如果在 2015 年 4 月 16 日被调用，DATE α DATE 将会添加上 `2015−04−16`、`16.04.2015` 或 `04/16/2015` 这样的日期信息。

注意，***alpha*** 最多可以包含 30 个字符。*WP 34S* 具有丰富的特殊字母和其他字符集，支持至少 37 种语言。因此，可以很容易地存储一个希腊文消息，例如：



[▲]和**[▼]**将会六个字符为一组地浏览长消息。**[▲]**在第一个字符显示后将停止，**[▼]**在显示最后一个字符后停止，在此例中即：



即使离开了 alpha 模式，仍然可以显示 alpha 字符：使用 VIEWα 或者 VWα+——它会显示最左端（即，alpha 寄存器里包含的第一个字符）。

不过别忘了，*WP 34S* 主要是作为可编程计算器而设计的。下一章介绍如何编程。

4. *WP 34S* 的编程

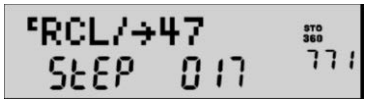
WP 34S 是一台按键编程计算器。如果这句话使您露出了会心的微笑，那么这一部分绝对适合您。如果没有的话，请先打开《*HP-42S* 用户手册》（*HP-42S Owner's Manual*）第二部分，了解用于解决重复性问题的按键式编程。请记住，它有大约 6000 个程序步的空间、700 多个按键编程函数，以及 16 位的精度。它和 HP-65 一样，足以应对巨大的挑战。

程序内存的第一步和最后一步中包含隐藏的 END 语句[52]。在它们中间是作为基本模块的程序。每个程序在上一个 END 之后开始，到下一个（它自己的）END 结束。当前程序是程序指针当前位于的那个程序。当前程序步是程序指针指向的位置。

每个程序都可以包含例程[53]和子程序。一个典型的例程自 LBL 语句开始，并以 RTN 结束。在它们之间，可以放置任意类型的命令（指令，操作，语句）序列以重复使用。除了少数几个指令外，本机任何操作都可用于编程。程序可以使用所有的全局寄存器——几乎没有限制。您是机器内存无可置疑的唯一支配者！

这种自由同样有其代价：一个例程使用的全局寄存器可能会被其它例程修改。所以，最好记下每个例程使用的全局寄存器，并将它们的用途与内容写成文档以备参考。

在编程模式中（即在编辑程序时），屏幕的数字部分中，有效数字部分显示当前程序步数（000-927），指数部分显示剩余步数。点阵部分则显示当前步中包含的指令。例如：



与 *HP-42S* 不同，*WP 34S* 没有每个程序独立的步数计数。

编程专用的指令在这张键盘图中由紫色框出。从左上角开始，依次是执行指令（XEQ），跳转指令（GTO），浏览键（**[▲]**和**[▼]**），标志位指令（SF，CF 和 STATUS），判断指令菜单（TEST），编程函数菜单（P.FCN），额外函数菜单（X.FCN），求解指令（SLV），积分指令，求和指令，退出编程模式键（EXIT），暂停

[52]第一个 END 不可见，但最后一个 END 是可见的。

[53]译者注：例程广义上是程序的一种，指在程序中用来反复调用和使用的部分代码。例程大都以子例程（即子程序）的形式存在。在本手册中不严格区分程序和例程。

执行程序键（PSE），提取部分数的指令（|x|，IP 和 FP），程序运行/暂停键（R/S），切换编程/运行模式键（P/R），标记例程的指令（LBL），从例程返回的指令（RTN）和循环控制指令（DSE 和 ISG）。关于这些指令的更多信息见 *IOP*。

程序标签

给程序添加标签可以结构化程序内存，并使得在其中跳转的工作变得简单。标签可以加在程序的任何一步中。*WP 34S* 有一套完备的字母数字混合的程序标签系统，详见下文。此外，不同的程序可能由 END 语句分隔，整个程序内存可以包含多个 END 语句。

标签寻址见下一节。

查找标签遵循下面的规则。当执行到一条例如 XEQ ***lbl*** 的指令时（其中 ***lbl*** 代表一个 1 到 3 个字符的标签，例如 A，BC，12，Tst，Pg3，x1µ 等），*WP 34S* 将会以如下方式查找这个标签：

1. 如果 ***lbl*** 是一个纯数字标签或是一个热键，*WP 34S* 将从程序指针的当前位置开始向下查找它。当查找到 END 语句处但仍没有发现 ***lbl*** 时，查找将从上一个 END 处继续（这样就只会在当前程序中查找）。这是本地标签的查找过程。这和在 HP-41C 上相同。

2. 如果 ***lbl*** 是一个由不限大小写的三个以内字符组成的 alpha 标签（会自动被单引号括起，像'Ab1'这样），查找将从程序的 000 步开始并以 *RAM*，*FM*，*XROM*（后两者见 76 页）的顺序覆盖整个内存，而与程序指针的位置无关。这是全局标签的查找过程。

示例：

以下是一个 135 步的程序：

```
001 LBL B       例程 B
...             例程 B
012 RTN         例程 B
013 LBL 'C'     例程 'C'
...             例程 'C'
034 LBL 01      例程 'C'
...             例程 'C'
056 RTN         例程 'C'
057 LBL 02      例程 02
...             例程 02
078 RTN         例程 02
079 END         第一个程序结束
080 END         第二个程序结束
081 LBL 'ABC'   例程 'ABC'
...             例程 'ABC'
```

```
123 RTN          例程 'ABC'
124 LBL C        例程 C
...              例程 C
135 RTN          例程 C
136 END          第三个程序结束
```

当程序指针在 001 步和 079 步之间时（也就是 LBL B 的作用范围），热键 B 被指定，因此按下**[B]**不会调用 1/x。当程序指针在 081 步和 136 步之间时（LBL C 的作用范围），热键 C 被指定，因此按下**[C]**不会调用 y^x。当程序指针在 080 步时，没有热键被指定，因此热键会执行对应的函数功能。

[GTO][.][.]会将任意位置的程序指针移动到 136 步（也就是当前内存中所有程序的结尾）。这样就可以开始编写一个新的例程了。

在编程模式之外按下**[RTN]**，会将任意位置的程序指针移动到 001 步。

程序内存中的最后一个 END 语句是不可删除的。它是写保护的，由系统自动生成。

标签寻址

1	用户输入 显示	**[A]，[B]，[C]或[D]** XEQ ***label*** 例：XEQ C	**[XEQ]，[GTO]，[LBL]，**LBL?**，[SLV]，[∫]，[Π]，[Σ]，**αGTO 或 αXEQ OP _ 例：Σ _　GTO _　SLV _　LBL _				
2	用户输入 显示	调用标签为 C 的程序。	**[A]，[B]，[C]或[D]** OP ***label*** 例：Σ B	**[ENTER⇧]** 打开 alpha 模式。 OP '_	**[⇨]**[54] 启用间接寻址并打开临时 alpha 模式。 OP →_		两位数（本地）标签 **[0][0]…[9][9]** OP ***nn*** 例：LBL 07
3	用户输入 显示		对标签为 B 的程序所给出的函数进行求和运算。	字母数字混合（全局）标签（1 到 3 字符[55]） OP'label' 例：SLV'F1μ'	堆栈级别或字符寄存器**[X]，[Y]，[Z]，…，[K]** OP→***x*** 例：∫→T	寄存器序号 **[0][0]…[9][9]，[.][0][0]…[.][1][5]**（如果对应的寄存器已经分配）[56] OP→***nn*** 例：XEQ→44	
				对标签为 F1μ 的程序所给出的函数进行求解运算（用脚注中的方法键入）。	对标签为堆栈 **T** 的值的程序所给出的函数进行积分运算。	执行标签为 **R44** 中的值的程序。	

GTO 指令的特殊情况见 *IOP*。

[54]对除 LBL 外所有操作都有效。

[55]第 3 个字符输入时输入会自动终止并关闭 alpha 模式，而对于较短的标记则必须手动按**[ENTER⇧]**键以关闭 alpha 模式。对于这里的例子，按[f]**[2][ENTER⇧][CPX][1]**[f]**[EXIT]**[g]**[7]**即可。包含超过一个字符的 alpha 标记的语句占用两步的内存。

注意：LBL A 和 LBL'A'是不同的。后者是在 alpha 模式下输入的，前者则是直接用热键输入的。

[56]一些寄存器可能被分配作其它用途。参见 26 页的内存表。

判断指令

像之前的按键编程计算器一样，*WP 34S* 有一整套判断指令。判断指令的名称以“?”结尾。通常，判断指令与 *HP-42S* 中的相同：从键盘调用时，它们在点阵显示中返回 `true` 或 `false`；在程序中被调用时，仅当一条判断指令返回真时才执行程序的下一步，否则跳过下一步。一般的规则都是“假则跳过”，只有 KEY?例外。

正如上文中提及的那样，一个例程一般以一个 RTN 或 END 结束。在程序运行中，这两种语句工作起来十分相似，只有微妙的区别：紧跟在返回 `false` 的判断指令后的 RTN 会被跳过，而 END 不会。更多信息见 *IOP*。所有的二值判断指令都包含在判断指令菜单（TEST）中。

注意还有一些以“?”结尾但返回为数字（如 BASE?）或代码（如 KTP?）而非真或假的指令。这些指令包含在编程函数菜单（P.FCN）中。

本地数据

WP 34S 存储的程序较多时，要记住每个程序所用到的资源就会变得十分困难。大多数现代编程语言通过声明本地变量来解决这个问题，也就是说，从总的数据内存中分配仅供当前程序使用的存储空间；程序结束时，该存储空间即被释放。在 *WP 34S* 上，用于数据存储的是寄存器，所以我们给程序分配本地寄存器以解决此问题。

例：假设有一个标签为 P1 的例程需要 5 个寄存器以完成计算。

可以在 P1 中输入指令 LocR 5 以声明需要 5 个本地寄存器。之后，在整个 P1 中都可以用本地寄存器号.00… .04 来访问这 5 个本地寄存器。

现在，如果在 P1 中调用另一个例程 P2，同样地，P2 可能包含一个 LocR 指令请求它的本地寄存器。它们也会使用从.00 开始的本地寄存器号，但是 P2 的.00 号本地寄存器和 P1 的.00 号本地寄存器是不同的寄存器，所以不会发生冲突。而一旦执行到 return 语句，对应例程中的本地寄存器即被释放，它们所占用的空间返还给空闲内存。

此外，只要在一个程序中请求了本地寄存器，还会同时有 16 个本地标志位可用。

本地数据的保存允许了程序的递归调用，因为每次程序被调用时，都会分配一套新的本地寄存器和本地标志位，与之前被分配的相独立。

关于 LocR，LocR?，MEM?和 POPLR 指令的更多信息以及对于本地数据的限制见 *IOP* 和附录 B。

程序控制输入和输出，用户交互和对话

大量的指令被用来控制程序出入输出。*IOP* 中记载的是它们从按键调用时的行为，而在程序中被调用时，它们的行为会有显著的不同。

当一个程序通过**[XEQ]**或**[R/S]**开始运行时，屏幕先前的显示内容会被 running PrOGrAM 消息代替，并且只在特定事件发生时——而不是每个操作执行之后——获得更新。在手动模式下每一条指令都会在执行时及时地更新显示内容，而在自动模式下只有 PROMPT，PST，STOP，VIEW，VIEWα 和 VWα+会更新显示内容，而在执行到下一条这类指令之前，所显示的内容都将保留。

一旦 running PrOGrAM 被其它程序显示覆盖，只有 ERR 0 或 MSG 0 能让它重新显示。

看下面的例子（其中参数省略）：

VIEW，VIEWα 和 VWα+ 用于简单显示的更新。X 是 VIEW 和 VIEWα+的一个有效参数。VIEW 对复数也是有效的。注意，频繁的显示更新会降低程序运行速度，这是因为机器的防闪烁逻辑会等待显示刷新完毕后才进行下一次显示更新。

下面 4 个代码片段可以显示消息或其它信息。这消息显示的最小时间由 PSE 决定（详见 *IOP*）：

PSE	VIEW PSE	VIEWα PSE	VWα+ PSE
显示纯数字输出		显示更复杂的字母数字混合消息	

用下面 4 个代码片段之一来请求（prompt）数字输入：

STOP	VIEWα STOP	VWα+ STOP	PROMPT	将 VWα+ X 和 STOP 结合在一条指令中

当按下**[R/S]**键继续程序运行时，键入的数字会保存在 **X** 中。如果想保存在其它地方，则请另加注意。

用以下步骤获得字母数字输入：

αON　　打开 alpha 模式并准备显示 alpha 的最后部分。

PROMPT　显示这一部分并等待用户输入，按**[R/S]**终止输入。

αOFF　　返回之前的数字模式。

输入的一切都会被追加到 ***alpha*** 中。按下**[R/S]**键程序将继续运行。

参见 *IOP* 以了解关于这些指令和它们的参数的更多信息。

如果在输入数字数据时或之后按下**[A]**到**[D]**的 4 个热键之一，下一个以该热

键为标签的例程将被调用。

下面的例子展示了这种程序的一个典型结构：

```
001   LBL 'MYP'
002    CLα
003    α 'He1'        创建一条消息…
004    α 'lo!'
005   LBL 00
006    PROMPT         …暂停程序以等待用户输入。
007    GTO 00         按下[R/S]之后简单地返回到输入。
008   LBL A           如果用户在 006 步的输入以热键[A]结束则被调用。
009    ENTRY?         用户有输入任何新的数字数据吗？
010    GTO 03         是则转到 012 步的本地标签 03。
011    XEQ 01         否则调用子程序 01 计算出一个新的数字。
012   LBL 03
013    STO 01         存储输入的或计算出的新数据。
014    RTN            返回 007 步的输入。
015   LBL 01
...    ...            如果用户输入空缺则计算出新数字。
...    ...
...    RTN
...   LBL B           如果用户在 006 步的输入以热键[B]结束则被调用。
...    ...
...    ...
...    RTN
...    ...
...   END
```

TVM 程序就是像这样组织的。

如果 *RAM* 或 *FM* 中有多个程序使用标签 A 到 D，必须移动程序计数器（program counter）到所需的程序并停在该处——前提是程序之间用 END 分隔开。见程序标签页以了解在其它情况下，哪个标签会被搜索到。

按键码和直接键盘访问

有些时候，4 个热键不能满足需求。有另外一个简单的方法来扩展可以直接调用的子程序的量：使用如下图所示的按键码进行数字标签的快捷寻址。可以看到右图中的每个按键都有一个由它在键盘上的行数和列数直接决定的按键码。

[A] 11	[B] 12	[C] 13	[D] 14	[⇨] 15	[CPX] 16
[STO] 21	[RCL] 22	[R⇩] 23	[f] 24	[g] 25	[h] 26
[ENTER⇧] 31		[x⇆y] 32	± 33	[EEX] 34	[⇦] 35
[XEQ] 41	[7] 42	[8] 43	[9] 44	[/] 45	
[▲] 51	[4] 52	[5] 53	[6] 54	[*] 55	
[▼] 61	[1] 62	[2] 63	[3] 64	[-] 65	
[EXIT] 71	[0] 72	[.] 73	[R/S] 74	[+] 75	

无论何时，当使用两位数的程序标签时，可以使用上图中任何一个绿色的按键码用于直接输入，即按下按键码对应的按键将执行此程序。黄色的按键则必须先按[f]键再按下（因为它们已被用作其它用途）。只有[f]键本身不能用于快捷寻址。

举例来说，如果想让一个程序能被**[STO]**键唤起，只需要把标签 21 放在该程序的开始处；之后它就可以被简单地用**[XEQ][STO]**调用。

KEY?指令返回的是按键对应的按键码，这样就可以得到用户键盘输入的“实时”反馈。KEY?接受一个寄存器参数（可以使用 **X** 但并不会提升堆栈）并把程序执行过程中最近按键的按键码储存在该寄存器中。该指令对**[R/S]**和**[EXIT]**不适用，因为它们会立即中断程序运行。在程序运行的过程中键盘输入始终是激活的，但也可以显示一条消息并用 PSE 停下程序等待用户输入。PSE 会被按键结束，所以可以简单地在循环中使用 PSE 99。鉴于 KEY?同样是一条判断语句，一个典型的用户输入循环可能如下例所示：

```
001   LBL 'USR'
002    CLα
003    α 'KEY'      创建一条消息…
004    α ?
005   LBL  00
006    VIEWα        …并显示它。
007    PSE 99       等待用户输入 9.9 秒，除非一个按键被中途按下。
008    KEY? 00      判断用户是否输入并将按键码存储在 R00 中。
009    GTO 00       如果没有输入则返回 005 步。
010    LBL?→00      而如果有一个以该按键码为标签的程序…
011    XEQ→00       …那么调用它，
012    GTO 00       …否则返回 005 步。
```

程序也可以不用空等输入，相反，可以在 PSE 和 KEY?的等待时间里执行

一些计算并更新显示——比如玩月球登陆游戏[57]。

还有更多功能：KTP? ***nn*** 返回按键的类型，其中 ***nn*** 是储有按键码的寄存器。返回值如下所示：

数字键返回相应的 0 到 9；

[.]，**[+/-]**，**[EEX]**返回 10；

[f]，[g]，[h]返回 11；

其它键返回 12。

如果寄存器中的按键码是无效的，则会抛出“invalid Range”错误。

如果不想在程序中处理按键，也可以用 PUTK ***nn*** 指令将它们交由 *WP 34S* 的主处理循环处理。程序将会暂停，之后这个按键会作为按下处理。这在想要得到数字输入却在等待一些特殊按键（如方向键）时特别有用——这样就可以编写一个向量和矩阵编辑器了。在 PUTK 指令执行完毕后需要按下**[R/S]**键或者一个热键以继续程序运行。

打印图形

WP 34S 为设置打印机图形字符串提供了许多命令。这些命令也可以在运行模式下使用，但在例程中更有意义。基本思想如下：

- 存储用于打印图形像素的一组寄存器称为图形块。此类块可能包含 ***w***（最多 166）列和 ***h*** 行像素（默认值为 ***w***=166 和 ***h***=8）。
- 上述图形块的前两个字节保留为参数 ***w*** 和 $\breve{h} = floor(\frac{h+7}{8})$。
- 此类图形块所需的寄存器数量为 $n = floor(\frac{w \cdot \breve{h} + 9}{8})$（参见 FLOOR）。例如，166×8 像素需要 21 个寄存器。
- GDIM 用于初始化这样的图形块。可以在整数模式下通过在第一个寄存器中存储 $256 \cdot \breve{h} + w$ 并清除之后的 n+1 个寄存器精确模拟该命令。

请注意，*WP 34S* 上所有图形指令都依赖于其数据结构。*WP 34S* 无法知道一个寄存器是否包含图形块或其一部分。您必须自己跟踪所使用的图形块。

[57] 译者注：自 HP 9100 以来几乎所有 HP 可编程计算器上都有的一个游戏。

闪存（*FM*）和 *XROM*

除所提供的 *RAM* 之外，*WP 34S* 允许访问 *FM* 中有断电保护的用户程序和数据存储。*FM* 的第一部分是一个 2*kb* 的备份区域，存储着全部用户程序，内存寄存器和 *WP 34S* 状态的镜像，并在每一次完成 SAVE 时刷新。*FM* 的其余部分（大小取决于设置，最多可达约 12*kb*）则只储存程序。和在 *RAM* 中一样，*FM* 中的全局标签可以通过 XEQ 调用。通过这个方法可以在 *FM* 中建立程序库。用 CAT 指令可以查看已定义的全局标签，其中 *FM* 中的标签会被标上 L、b。

FM 用于长期备份和存储数据是理想的，但是不适合像在程序循环中那样被当作临时存储反复使用[58]。相对的，*RAM* 中的寄存器和标准用户程序内存被设计用于频繁改写数据，但在断电时其数据会丢失。两种存储各有其利弊，在使用时需要从程序优化和机器使用寿命两方面综合考虑。关于 *FM* 的更多内容见附录 A。

另外，有一部分称为 *XROM*（扩展 *ROM*）的存储。里面存放有大约 30%的指令例程（例如，f'(x)和 f''(x)，SLV 和 SLVQ，∫，Π 和 Σ，所有统计分布，正交多项式，AGM 函数，Bn 函数，Bn+函数，ERF 函数，ERFC 函数，FIB 函数，gd 函数及其反函数，NEXTP，朗伯 W 函数及其反函数，欧拉 Beta 函数，黎曼 Zeta 函数，以及大部分的复数域函数）。它们虽然由用户代码编写，却是只读的，因此只能被执行或作为指令调用而不能被编辑。一个预编写的 *WP 34S* 程序在 *ROM* 和 *XROM* 中执行没有区别。但是，*XROM* 中的程序易于用户阅读和理解以熟悉 HP 按键式编程——这些程序见 *http://wp34s.svn.sourceforge.net/viewvc/wp34s/trunk/xrom/*，更多信息见附录 H。

[58] *FM* 可能不能写入 1 万次以上。因此，我们禁止写 *FM* 的指令（如 SAVE 和 PSTO）用于编程。

5. 操作指令目录（IOP）

WP 34S 上所有可供使用的指令（超过 700 个）名称及输入方式在下文列出（[CONV]和[CONST]中的指令除外，它们在下一章单另列出）。在表中，凡是使用**粗体**显示名称的，均属于在键盘上可以直接输入的功能；而其他的命令也许来自于菜单。除去某些特殊情况，在菜单和程序列表中命令的名称相同。

索引及菜单中的排序不分大小写，具体顺序如下：

␣ 0…9 A…Z, α…ω ( ) + - × / ± , . ! ? : ;

‘ “ * @ _ ~ → ← ↑ ↓ ⇆ < ≤ = ≈ ≠ ≥ >

% $ € £ ¥ √ ∫ ∞ & \ ^ | [ ] { } 🖶 #

上标和下标在排序中视为正常的字符。上面排序列表末尾处的🖶是代表所有打印命令的打印符号。

除非另有说明，一般函数和按键编程命令和 *HP-42S* 相同，而位操作与整数函数和 *HP-16C* 相同。在索引中提到的某些函数是受其他古老的计算器启发而设计的，它们的手册可能会包含更多有用的信息。以 ^{C} 开头的代表复数操作命令（见 29 页）。**[CPX]**是在列表中名称以*斜体*显示的函数的合法前缀。

在 *WP 34S* 上（也是在 RPN 计算器上）首次出现的，有大约 300 个函数。它们的备注将以黄色背景显示。对于和以往的 RPN 计算器上名称类似但功能上有所差别的函数，它们的备注将以橙色背景显示。打印图形指令用紫红色背景显示。

大多数操作指令以数字开头：

(0) 表示对栈没有影响的函数，

(1) 表示一元函数，

(2) 表示二元函数，

(3) 表示三元函数；

(−1) 表示压入一个实数或复数到堆栈中的函数，

(−2) 表示压入两个实数或复数到堆栈中的函数。

CLx、ENTER、Σ+和 Σ-这四个指令是仅有的禁止自动堆栈提升的指令。此类操作后的数字输入将覆盖 ***x***，而不是将其压入到堆栈上。

参数将会从最底层的栈中获取——除非在索引的第二列中有特别注明。参数紧接在指令之后。有些与统计分布有关的参数必须在寄存器 **J** 和 **K** 中给出。参数记号的例子见下文。

在接下来的三个例子中，假设寄存器 **R12** 包含值 15.67（即 ***r12*** = 15.67）

- ***n*** 表示一个必须由键盘直接输入的任意整数，与此同时，***n*** 表示还能由一个寄存器间接指定的整数（也见 26 和 33 页的表格）；***n*** 表示的是其数值本身。

  示例：RSD **12** 将 ***x*** 四舍五入至 **12** 位有效数字，而 RSD→**12** 则将 ***x*** 四舍五入至 **15** 位有效数字。

- ***s*** 表示一个必须由键盘直接输入的任意寄存器地址，与此同时，***s*** 也表示一个能由寄存器间接指定的地址；并且 ***s*** 表示的是其内容（即在被直接指定的情况下，表示寄存器 ***s*** 的内容，在被间接指定的情况下，表示寄存器 ***s*** 所指向地址的内容）。

  示例：STO **12** 将 ***x*** 存储至寄存器 **R12**，而 STO→**12** 将 ***x*** 存储至寄存器 **R15**。

- ***label*** 表示一个必须由键盘直接输入的任意标签；与此同时，***label*** 也表示一个可由寄存器间接指定的数值（参见 68 页后的表格）；并且 ***label*** 表示的是标签本身。

  示例：GTO **12** 指向本地标签 **12**，而 GTO→**12** 指向本地标签 **15**。

下表中，每一个列出来的函数都说明了它运行时所在的模式，这可以从对应的指示符（见 36 页）或它们名称的缩写中知道。在“模式”这一列中，“integer”代表任意的整数模式，“&”表示布尔“和”，逗号表示“或”，“¬”表示“非”。例如，2^X 可以在除 alpha 模式以外的其他所有模式中使用，甚至是复数域中。除非另有说明，所有的操作也可以在编程模式中输入（但 P.FCN 中的许多函数只在程序中有意义）。模式栏中使用红色字体显示的命令，需要石英晶振才能正常使用（见附录 A 和附录 H）。

名称	按键	模式	备注（通用说明见 77 页）
10^x	[f][10^x]	¬INPUT	(1)返回 10^x。
12h	[h][mode] 12h	All	(0)设定 12 小时制：即 1:23 将会变成 1:23 AM，23:45 将会变成 11:45 PM。这将仅会在 αTIME 中引起变化。
1COMPL	[h][mode]1COMPL	All	(0)为整数设置反码模式。
1/x	[f][1/x]	DECM	(1)返回 x 的倒数。
	[B]	DECM	(1)如果标签 B 未定义，为此功能的快捷键。
24h	[h][mode]24h	All	(0)设定 24 小时制。类似于 12h。
2COMPL	[h][mode]2COMPL	All	(0)为整数设置补码模式。
2^x	[f][2^x]	¬INPUT	(1)返回 2^x。
$\sqrt[3]{x}$	[h][X.FCN]3√x	¬INPUT	(1)返回 x 的立方根。
ABS	[f][|x|]	¬INPUT	(1)返回 x 的绝对值。
	[CPX][f][|X|]	DECM	(1)返回 x_c 的幅值，即将 $r=\sqrt{x^2+y^2}$ 返回 **X**，并清空 **Y**。
ACOS	[g][COS^-1]	DECM	(1)返回 arccos(*x*)[59]。
ACOSH	[g][HYP^-1][COS]	DECM	(1)返回 arcosh(*x*)。注意，这里不需要按[f]。
AGM	[h][X.FCN]AGM	DECM	(2)返回 **X** 和 **Y** 的算术平均数。详见附录 I。
ALL	[h][ALL]*n*	¬INPUT	(0)设置数字显示格式以尽可能显示所有小数。ALL 0 和 *HP-42S* 上的 ALL 几乎相同。 但是对于 $x\geq10^{13}$，显示模式将会切换至 SCI 或 ENG 以显示最大位数（见 SCIOVR 和 ENGOVR）。同样地，$x<10^{-n}$ 且有 12 位以上数字需要显示时，显示模式也会自动转换。 **示例：** 输入 / 显示 700 → 700 ALL 03 → 700.000 1/x → 0.001 428 571 43 10 / → 142 857 142 857^-4
AND	[h][AND]	Integer	(2)位运算，与 *HP-16C* 类似。参见 57 页。
		DECM	(2)与 *HP-28S* 上的 AND 类似，即在运算前先将 x 和 y 进行解析，0 是“false”，任何其他的实数是“true”。
ANGLE	[h][X.FCN]ANGLE	DECM	(2)返回 arctan(*y*/*x*)。
ASIN	[g][SIN^-1]	DECM	返回 arcsin(*x*)[60]。
ASINH	[g][HYP^-1][SIN]	DECM	(1)返回 arsinh(*x*)。注意，这里不需要按[f]。

[59]准确的说，返回的是主值，即在弧度制下实部∈[0, π]，或者在角度值下实部∈[0, 180]，或者在百分制下下实部∈[0, 200]。参见 ISO/IEC 9899。

[60]准确的说，返回的是主值，即在标志位 D 设置的情况下，弧度制下实部∈[-π/2, π/2]，或者在角度值下实部∈[-90, 90]，或者在百分制下实部∈[-100, 100]。其他情况下和 arctan 一样，为(-π/2, π/2)。参见 ISO/IEC 9899。

名称	按键	模式	备注（通用说明见 77 页）
ASR	[h][X.FCN]ASR*n*	Integer	(1)对于 ***n***（≤63），和 *HP-16C* 上的 ASR 命令类似，即被 2^n 除。ASR 0 视为无操作（NOP），但是会改变 L。参见 57 页。
ATAN	[g][TAN^{-1}]	DECM	(1)返回 arctan(x)[61]。
ATANH	[g][HYP^{-1}][TAN]	DECM	(1)返回 artanh(x)。注意，这里不需要按[f]。
BACK	[h][X.FCN]BACK*n*	¬INPUT	(0)跳转到 ***n*** 步之前（0≤***n***≤255）。例如：BACK 1 意味着回到前一个程序步骤。如果 BACK 命令试图越过一个 END，那么就会抛出一个错误。到步骤 000 将会终止执行程序。注意和 SKIP 的区别。 **注意：**如果在程序中交叉使用了 BACK，SKIP，或 CASE 命令，那么很可能需要单独手动维护这些语句。
BASE	[h][MODE]BASE*n*	All	(0)将整数运算设置为 ***n*** 进制，其中 2≤***n***≤16（见 53 页）。常用的进制可在键盘上直接设置。此外，BASE 0 即 DECM，BASE 1 亦即 FRACT。 **注意：**当一个整数模式切换至 DECM 后，堆栈中的内容将会随之被转换；从 DECM 进入其他模式时，堆栈数据会发生截断。其他寄存器中的内容则保持不变（见 152 页）。BASE 10 并不是 DECM。
BASE 10	[f][10]	¬INPUT	
BASE 16	[g][16]		
BASE 2	[f][2]		
BASE 8	[g][8]		
BASE?	[h][P.FCN]BASE?	Integer	(-1)返回整数进制设置。
		DECM	(-1)返回 DECM 之前的整数进制设置。
BATT	[h][X.FCN]BATT	DECM	(0)返回电池电压，测量范围在 1.9*V* 和 3.4*V* 之间。
		Integer	(0)与 DECM 中类似，但是单位为 100*mV*。
BC?	[h][TEST]BC?*n*	Integer	(0)判断 ***x*** 中指定的位是否清零。
BestF	[h][MODE]BestF	All	(0)选择最佳曲线拟合模式，使相关性最大化，与 *HP-42S* 中的 BEST 类似。
Binom	[h][PROB]Binom	DECM	(1)二项分布，成功次数 *g* 由 **X** 给出，成功概率 p_0 由 **J** 给出，样本大小 ***n*** 由 **K** 给出。计算公式见附录 I。 对于给定的 *p*，p_0 和 ***n***，Binom^{-1} 返回最大成功次数 ***m***。
$Binom_p$	[h][PROB]Binom$_p$		
$Binom_u$	[h][PROB]Binom$_u$		
$Binom^{-1}$	[h][PROB]Binom^{-1}		
B_n	[h][X.FCN]B$_n$	DECM	(1)对于 **X** 中的一个整数 ***n***>0，B_n 返回其伯努利数。Bn*则使用旧的定义。详见附录 I。
B_n*	[h][X.FCN]B$_n$*		
BS?	[h][TEST]BS?*n*	Integer	(0)判断 ***x*** 中指定的位是否置位。

[61]准确的说，返回的是主值，即在标志位 D 设置的情况下，弧度制下实部∈(-π/2, π/2)。参见 ISO/IEC 9899。

名称	按键	模式	备注（通用说明见 77 页）
CASE	[h][P.FCN]CASE*s*	¬INPUT	(0)类似于后面的 SKIP 命令，不同之处在于由 ***s*** 指定了跳转的步数。 示例：假设有以下程序部分： ... 100 CASE 12 101 GTO 01 102 GTO 02 103 GTO 07 104 GTO 05 105 LBL 01 ... 132 LBL 02 ... 153 LBL 05 ... 234 LBL 07 ... 在程序的执行过程中，***r12*** 将会在第 100 步被检查：如果 ***r12***≤1 那么程序将进行第 101 步，并跳转到第 105 步；若 ***r12***=2 那么程序将跳转到第 102 步，以此类推。对于 1≤***r12***≤4 都会有相对应的步骤分配。 注意 1：在上面的示例中，若 ***r12***>4，那么 CASE 命令可能会产生意料之外的结果。因此请注意 ***s*** 的数值！ 注意 2：如果在程序中交叉使用了 BACK，SKIP，或 CASE 命令，很可能导致需要单独手动维护所有这些语句。
	[h][CAT]	¬INPUT	浏览器。见 117 页。
Cauch	[h][PROB]Cauch	DECM	(1)柯西-洛伦兹分布（又名洛伦兹分布或布雷特-维格纳分布），位置参数 x_0 由 J 给出，尺度参数 γ 由 K 给出。详见附录 I。 Cauch^{-1} 对于 x 中给定的概率 p，J 中给定的 x_0 和 K 中给定的 γ，返回 x。
Cauch$_P$	[h][PROB]Cauch$_P$		
Cauch$_u$	[h][PROB]Cauch$_u$		
Cauch^{-1}	[h][PROB]Cauch^{-1}		
CB	[h][X.FCN]CB*n*	Integer	(1)清除 ***x*** 中指定的位。
CEIL	[h][X.FCN]CEIL	¬INPUT	(1)返回大于等 ***x*** 的最小整数。
CF	[g][CF]*n*	¬INPUT	(0)清除指定的标志位。
CFALL	[h][P.FCN]CFALL	¬INPUT	(0)清除所有的全局和本地用户标志位。
CLALL	[h][P.PCN]CLALL	¬INPUT	(0)清除 *RAM* 中所有的寄存器（需确认）。计算器的模式保持不变。注意和 RESET 的区别。
CLP	[f][CLP]	All	(0)清除当前程序，即程序指针所在的程序。
CLPALL	[h][P.FCN]CLPALL	¬INPUT	(0)清除 *RAM* 中所有的程序（需确认）。不可用于编程。
CLREGS	[h][P.FCN]CLREGS	¬INPUT	(0)清除所有通用寄存器（参见 REGS 和 LocR）。堆栈和 **L** 中的内容将会被保留。
CLSTK	**[0]**[g][FILL]	¬INPUT	清除当前分配的所有堆栈寄存器，即 **X** 到 **T** 或 **X** 到 **D**。所有其他的寄存器中的内容将会被保留。
	[h][P.FCN]CLSTK		

名称	按键	模式	备注（通用说明见 77 页）
CLx	[h][CLx]	¬INPUT	清除寄存器 **X**，自动堆栈提升被禁用。
CLα	[h][CLx]	INPUT	(0)清除 alpha 寄存器，与 *HP-42S* 上的 CLA 命令类似。
	[h][P.FCN]CLα	¬INPUT	
CLΣ	[g][CLΣ]	DECM	(0)清除求和寄存器并释放分配给它们的内存。
CNST	[h][P.FCN]CNST ***n***	¬INPUT	(-1)返回 CONST 菜单中第 ***n*** 个常数。允许对这些常数进行间接寻址（参见 126 页）。
ᶜCNST	**[CPX]**[h][X.FCN]CNST ***n***	¬INPUT	(-2)与 CNST 类似，但是将常数作为复数调用，见 120 页。
CNVG?	[h][TEST]CNVG? ***s***	DECM	(0)根据 ***s*** 最低的 5 个位，比较 ***x*** 和 ***y*** 来检查收敛性。 a)***s*** 最低的两个位是容差限制设置： 0=10^{-14}， 1=10^{-24}， 2=10^{-32}， 3=根据当前模式设置自动选择最佳值，对于 SP 取 0，对于 DP 取 2（参见附录 H）。 b)接下来的两个位将决定在给定容差限制下的比较模式： 0：相对比较 ***x*** 和 ***y***， 1：绝对比较 ***x*** 和 ***y***， 2：比较 ***x+iy*** 和 ***z+it*** 两个复数的绝对差， 3：和 0 相同。 c)顶部的位表示如何处理特殊数字： 0：NaN 和无穷大视为收敛， 1：NaN 和无穷大视为不收敛。 根据以上可知，***s***=a+4b+16c.
		Integer	(0)判断 ***x=y*** 是否成立。
COMB	[f][Cy, x]	¬INPUT	(2)返回在 ***y*** 个元素的集合中取 ***x*** 个元素的组合数。组合中所有的元素都不重复，且顺序不同的同样 ***x*** 个元素不另行统计。计算公式见附录 I。请注意和 PERM 的区别。
ᶜCONJ	**[CPX]**[h][X.FCN]ᶜCONJ	DECM	(1)翻转 ***y*** 的符号，以返回 x_c 的共轭复数。
	[h][CONST]	¬INPUT	菜单。参见 117 页。
	[h][CONV]	DECM	菜单。参见 117 页。
CORR	[g][r]	DECM	(-1)返回当前统计数据与曲线拟合模型的相关系数。参见附录 I。
COS	[f][cos]	DECM	(1)返回***x***的余弦。
COSH	[f][HYP][COS]	DECM	(1)返回***x***的双曲余弦。
COV	[h][STAT]COV	DECM	(-1)返回两个数据集的总体协方差。这取决于所选的拟合模型。样本协方差见 s_{XY}。详见附录 I。
ᶜCROSS	**[CPX]**[h][X.FCN]ᶜCROSS	DECM	(2)将***x***和***y***作为第一个向量的笛卡尔分量，类似的将***z***和***t***作为第二个向量，返回[***x*** ·***t***-***y*** ·***z***, 0, …]，堆栈下降两个等级。

名称	按键	模式	备注（通用说明见 77 页）
DATE	[h][X.FCN]DATE	DECM	(-1)按照所设定的格式，根据实时时钟调用日期到显示屏的数字部分。参见 D.MY，M.DY，和 Y.MD。此外，DATE 会在点阵中显示星期。*HP-12C* 上的 DATE 命令与 *WP 34S* 上的 DAYS+ 命令一致（参见下文）。
DATE→	[h][X.FCN]DATE→	DECM	(-2)假设*x*包含选定格式的日期，并将其三个部分压入堆栈。
DAY	[h][X.FCN]DAY	DECM	(1)假设*x*包含选定格式的日期，提取其日数。
DAYS+	[h][X.FCN]DAYS+	DECM	(2)假设**Y**中包含选定格式的日期，加上*x*天，并以与WDAY相同的方式显示结果日期和星期。与*HP-12C*上的DATE命令类似。
DBLOFF	[h][MODE]DBLOFF	All	(0)切换 DP 模式。设置仅在 DECM 中有效，在点阵中用 D 表示。参见附录 H。
DBLON	[h][MODE]DBLON		
DBL?	[h][TEST]DBL?	¬INPUT	(0)检查 DP 模式是否开启。
DBLR	[h][X.FCN]DBLR	Integer	双字节长度命令，用于求余，乘法和除法。DBLR 和 DBL/接受放在 **Y** 和 **Z** 中的双倍长度的被除数（最高有效位在 **Y** 中），除数在 **X** 中，结果也会返回到 **X** 中。DBL×将 *x* 和 *y* 作为因子，且它们的乘积返回在 **X** 和 **Y** 中（最高有效位在 **X** 中）。见 *HP-16C* 用户手册，第四部分（52 页）。
DBL×	[h][X.FCN]DBL×		
DBL/	[h][X.FCN]DBL/		
DEC	[h][P.FCN]DEC*s*	¬INPUT	(0)以 1 为步长将 *s* 递减。不会改变 L，即使目标地址为 **X**。
DECM	[f][H.d]	¬INPUT	(0)设置默认的十进制浮点模式。
DECOMP	[h][X.FCN]DECOMP	DECM	(-1)分解 *x*（如果可以变成假分数的话），在顶行显示 *y*/*x*=，堆栈中[分母(*x*), 分子(*y*), ...]。可用除法作为其逆运算。 示例：若 **X** 中的值为 2.25，那么 DECOMP 将会返回 *x*=4 和 *y*=9，并将之前 **Y** 的内容转推到 **Z** 中。
DEG	[g][DEG]	DECM	(0)将角度模式设置为角度制。
DEG→	[h][X.FCN]DEG→	DECM	(1)将 *x* 视为角度制，并将其转换为当前设置的角度模式。
DENANY	[h][MODE]DENANY	All	(0)设置默认分数格式，与 *HP-35S* 类似——分母可以是 DENMAX 设置范围内的任何值。 **示例：**若 DENMAX=5，那么 DENANY 允许的分母有 1，2，3，4，和 5。
DENFAC	[h][MODE]DENFAC	All	(0)设置“最大分母因子”，即分母只能是 DENMAX 的所有整数因子。 **示例：**若 DENMAX=60，那么 DENFAC 允许的分母有 1，2，3，4，5，6，10，12，15，20，30，和 60。
DENFIX	[h][MODE]DENFIX	All	(0)设置固定的分母格式，即唯一允许的分母是 DENMAX 所设置的值。

名称	按键	模式	备注（通用说明见 77 页）
DENMAX	[h][MODE]DENMAX	All	(0 或 1)与 *HP-35S* 上的/c 类似，但可以设置的最大分母是 9999。如果在 DENMAX 执行时 x<1 或者 x>9999 它将被设置为 0。对于 x=1，则将 x 替换为当前的 DENMAX 值。
DET	[h][MATRIX]DET	DECM	(1)读取 **X** 中的方阵的描述符，并返回这个矩阵的行列式。矩阵本身没有被修改。
DISP	[h][MODE]DISP *n*	All	(0)更改显示的小数位数，同时保持基本显示格式不变（FIX、SCI、ENG）。在 ALL 设置下，DISP 将会改变显示模式的切换点（参见 ALL）。
ᶜDOT	**[CPX]**[h][X.FCN]ᶜDOT	DECM	(2)将 x 和 y 作为第一个向量的笛卡尔分量，类似的将 z 和 t 作为第二个向量，返回[$x{\cdot}z+y{\cdot}t$, **0**, …]，堆栈下降两个等级。
dRCL	[h][X.FCN]dRCL *s*	¬INPUT	(-1)假设源数据包含 DP 数据，并以这种方式调用它们。参见附录 H。
DROP	[h][P.FCN]DROP [h][X.FCN]DROP	¬INPUT	丢弃（DROP）x。详见 18 页。
ᶜDROP	**[CPX]**[h][X.FCN]ᶜDROP	DECM	丢弃（DROP）x_c。详见 18 页。
DSE	[f][DSE] *s*	¬INPUT	(0)对于源数据中给定的 ccccc.fffii，DSE 按照步长 ii 来递减 s（s 即为循环控制数），若 ccccccc≤fff 则跳过下一行程序。 若 s 没有小数部分，那么 fff 为 0。 如果 ii=0，则 ccccccc 按 1 递减。 DSE 不会改变 L，即使目标地址为 **X**。 请注意 fff 或 ii 均不能为负，且 DSE 仅当 ccccc>0 时有意义。
DSL	[h][P.FCN]DSL *s*	¬INPUT	(0)与 DSE 类似，但在 ccccccc<fff 时跳过下一行程序。
DSZ	[h][P.FCN]DSZ *s*	¬INPUT	(0)以步长 1 来递减 s，之后若 | s | <1 则跳过下一行程序。不会改变 L，即使目标地址为 **X**。
D.MY	[h][MODE]D.MY	All	(0)设置日期显示的格式为dd.mmyyyy。
D→J	[h][X.FCN]D→J	DECM	(1)根据设定的格式将 x 视为一个日期并根据公历将其转换为儒略日数。
D→R		DECM	(1)角度到弧度的转换，参见 CONV 中的角度转换菜单。
END	[h][P.FCN]END	¬INPUT	(0)例程的最后一个语句和搜索本地标签的终止位置，就像 68 页描述的那样。在其他方面和 RTN 一样。
ENG	[h][ENG] *n*	¬INPUT	(0)设置工程显示格式（参见 41 页）。
ENGOVR	[h][ENG][ENTER⇧]	¬INPUT	(0)定义在 ALL 或 FIX 中超出可显示范围的数字以工程显示格式显示。注意和 SCIOVR 的区别。
ENTER ↑	**[ENTER⇧]**	¬INPUT	(-1)将 x 推至栈中，禁用自动升栈。详见 18 页和 31 页。

名称	按键	模式	备注（通用说明见 77 页）
ENTRY?	[h][TEST]ENTRY?	¬INPUT	(0)检查 entry 标志位。以下两种情况下，这个内部的标志位将会被置位： ● alpha 模式中输入任何字符， ● 用于输入的任何命令（无论是**[ENTER⇧]**，一个功能按键，或部分命令行的**[R/S]**。 可用在 PSE 之后。
erf	[h][X.FCN]erf	DECM	(1)返回误差函数或互补误差函数。详见附录 I。
erfc	[h][X.FCN]erfc		
ERR	[h][P.FCN]ERR***n***	¬INPUT	(0)引发指定的错误，就像是错误真的发生了一样，因此一个运行中的程序可能会被终止。不同的错误代码见附录 C。与 MSG 类似。
EVEN?	[h][TEST]EVEN?	¬INPUT	(0)检查 ***x*** 是否是整数且是偶数。
e^x	[f][e^x]	¬INPUT	(1)返回 e^x。
	[EXIT]	All	见 120 页。
ExpF	[h][mode]ExpF	All	(0)选择指数曲线拟合模型： $R_0 = a_0 e^{a_1 x}$。
Expon	[h][PROB]Expon	DECM	(1)指数分布，率参数 ***λ*** 由 **J** 给出。详见附录 I。Expon^{-1} 对由 **X** 给出的 *p*，**J** 给出的 ***λ*** 返回存活时间 t_s。
Expon$_P$	[h][PROB]Expon$_P$		
Expon$_u$	[h][PROB]Expon$_u$		
Expon^{-1}	[h][PROB]Expon^{-1}		
EXPT	[h][X.FCN]EXPT	DECM	(1)返回以 $x=m\cdot10^h$ 显示的数中的指数 ***h***。注意与 MANT 的区别。
e^X-1	[h][X.FCN]e^x−1	DECM	(1)对于 ***x***≈0，小数部分比 e^X 更精确。
FAST	[h][MODE]FAST	All	(0)将处理器速度设置为“fast”。这是初始默认选项，并且对新电池保持该设置。参见 SLOW。
FB	[h][X.FCN]FB***n***	Integer	(1)翻转 ***x*** 中指定的位。
FC?	[h][TEST]FC?***n***	¬INPUT	(0)判断指定的标志位是否被清除。
FC?C	[h][TEST]FC?C***n***等	¬INPUT	(0)判断指定的标志位是否被清除。在判断之后对其进行清除、翻转或置位等操作。
FC?F			
FC?S			
FF	[h][P.FCN]FF***n***	¬INPUT	(0)翻转指定的标志位。
FIB	[h][X.FCN]FIB	Integer	(1)返回斐波那契数。详见附录 I。
		DECM	(1)返回扩展的斐波那契数。
FILL	[g][FILL]	¬INPUT	(0)将 ***x*** 复制到所有级别的栈中。参见 31 页的 CFILL。
FIX	[h][FIX]***n***	¬INPUT	(0)设置定点显示格式（见 41 页）。
FLASH?	[h][P.FCN]FLASH?	¬INPUT	(-1)返回 *FM* 中剩余空间（以字计）。
FLOOR	[h][X.FCN]FLOOR	¬INPUT	(1)返回≤***x*** 的最大整数。
FP	[g][FP]	¬INPUT	(1)返回 ***x*** 的小数部分。
FP?	[h][TEST]FP?	¬INPUT	(0)判断 ***x*** 是否有一个非零的小数部分。

名称	按键	模式	备注（通用说明见 77 页）
FRACT	[h][MODE]FRACT	All	(0)根据PROFRC或IMPFRC 以及DEN...的设置将 ***x***转换为分数模式。参见41页
FS?	[h][TEST]FS?*n*	¬INPUT	(0)判断指定的标志位是否置位。
FS?C FS?F FS?S	[h][TEST]FS?C*n*等	¬INPUT	(0)判断指定的标志位是否置位。在判断之后对其进行清除、翻转或设置等操作。
$F_P(x)$	[h][PROB]Fₚ(x)	DECM	(1)F 分布。$F_u(x)$等于 *HP-21S* 的 Q(F)。自由度由 **J** 和 **K** 指定。详见附录 I。
$F_u(x)$	[h][PROB]Fᵤ(x)		
F(x)	[h][PROB]F(x)		
$F^{-1}(p)$	[h][PROB]F⁻¹(p)		
f'(x)	[h][P.FCN]f'(x)*label*	DECM	返回 **f(*x*)**在 ***x*** 处的一阶导数。这个 **f(*x*)**必须在以 LBL ***label*** 开始的例程中指定。在返回时，**Y**，**Z** 和 **T** 将会被清除，且 ***x*** 会保存在 **L** 中。 **注意：** f'(x)将会寻找一个标签为 δx（或者依次查找 δX，Δx 或 ΔX）的用户例程，将一个固定步长 ***dx*** 返回给 **X**。如果用户例程未定义，默认设置 ***dx***=0.1（也可以为任意值，但它必须被指定）。 然后，f'(x)将会在 x±5***dx*** 的十个点计算 **f(*x*)**。如果要用其它间隔的话，可以通过改变 ***dx*** 的值来改变它们。
f"(x)	[h][P.FCN]f"(x)*label*	DECM	与 f'(x)类似但是会返回二阶导数。上文的说明在这里同样适用。
GCD	[h][X.FCN]GCD	¬INPUT	(2)返回 ***x*** 和 ***y*** 最大公约数。结果总为正数。
gCLR	[h][P.FCN]gCLR*s*	¬INPUT	(0)清除从寄存器地址 ***s*** 开始的图形块中位置 ***x***，***y*** 处的像素。有效范围为 0≤***x***≤w-1 和 0≤***y***≤h-1。像素点 0，0 位于最左上的位置。≤0 的输入会被设置为最大值。另见 gDIM。
g_d	[h][X.FCN]gₐ	DECM	(1)返回古德曼（Gudermann）函数或其逆函数。详见附录 I。
g_d^{-1}	[h][X.FCN]gₐ⁻¹		
gDIM	[h][P.FCN]gDIM*s*	¬INPUT	(0)从地址 ***s*** 开始初始化一系列寄存器（又称图形块，见 75 页），允许图形数据具有 ***x*** 列像素和 ***y*** 行像素。***x***≤0 时宽度 ***w*** 将会被设置为 166，***y***≤0 时高度 ***h*** 将会被设置为 8。 参见🖶PLOT。
gDIM?	[h][P.FCN]gDIM?*s*	¬INPUT	(-2)假设有一个从地址 ***s*** 开始的图形块，会调用其 ***h*** 和 ***w***。详见 75 页。
Geom	[h][PROB]Geom	DECM	(1)几何分布：cdf 返回 ***m***=***x*** 次伯努利试验后第一次成功的概率。每个实验成功的概率 p_0 必须在 **J** 中指定。详见附录 I。 $Geom^{-1}$ 返回在 **X** 中给定的概率 *p* 和在 **J** 中给定的 p_0 下，第一次成功之前的失败次数。
$Geom_P$	[h][PROB]Geomₚ		
$Geom_u$	[h][PROB]Geomᵤ		
$Geom^{-1}$	[h][PROB]Geom⁻¹		
gFLP	[h][P.FCN]gFLP*s*	¬INPUT	(0)翻转从地址 ***s*** 开始的图形块中的 ***x***，***y*** 处的像素。更多信息请参阅 gCLR。

名称	按键	模式	备注（通用说明见 77 页）
gPIX?	[h][TEST]gPIX?*s*		(0)判断从地址 *s* 开始的图形块中的 *x*，*y* 处是否有像素。更多信息请参阅 gCLR。
gPLOT	[h][P.FCN]gPLOT*s*	¬INPUT	(0)在 LCD 的点阵部分显示图形块的左上角部分（从地址 *s* 开始）。更多信息请参阅 gCLR。
GRAD	[g][GRAD]	DECM	(0)设置角度模式为百分制。
GRAD→	[h][X.FCN]GRAD→	DECM	(1)将 *x* 视为百分制角度，将其转换为当前设置的角度模式。
gSET	[h][P.FCN]gSET*s*	¬INPUT	(0)将从地址 *s* 开始的图形块中位于 *x*，*y* 处的像素置位。更多信息请参阅 gCLR。
GTO	[h][GTO]***label***	STO	(0)插入一个跳转到 ***label*** 的无条件分支。
		¬STO，¬INPUT	(0)将程序指针定位到 ***label***。
	[h][GTO][.]…	¬INPUT	(0)不可用于编程的指令，这些指令详见 112 页。
GTOα	[h][P.FCN]GTOα	¬INPUT	(0)将 ***alpha*** 的前三个字符（或少于三个的全部字符）作为标签，并将程序指针定位到该标签处。
H_n	[h][X.FCN]H.	DECM	(2)概率（H_n）和物理（H_{np}）的埃尔米特多项式。详见附录 I。
H_{np}	[h][X.FCN]H..		
H.MS	[f][H.MS]	DECM	(1)假设 **X** 包含十进制的时刻或角度，将其按 hhhh°mm′ss.dd″的格式转换并临时显示，如第 45 页所示。
H.MS+	[h][X.FCN]H.MS+	DECM	(2)假设 **X** 和 **Y** 含有 60 进制 hhhh.mmssdd 格式的时刻或角度，分别对它们进行加和减。
H.MS-	[h][X.FCN]H.MS−		
IDIV	[h][X.FCN]IDIV*s*	¬INPUT	(2)整除，与 DECM 中的**[/]**[IP]和整数模式中的**[/]**类似。
IMPFRC	[g][d/c]	¬INPUT	(1)设置分数模式，允许显示假分数（例如，$\frac{5}{3}$ 而不是 $1\frac{2}{3}$）。将根据 DEN…的设置转换 *x* 为一个合适假分数（如果可能的话）。*X* 相应的十进制绝对值不得超过 100000。注意与 PROFRC 的区别。
INC	[h][P.FCN]INC*s*	¬INPUT	(0)*s* 增加 1。不改变 L，即使目标地址为 **X**。
INTM?	[h][TEST]INTM?	¬INPUT	(0)判断 *WP 34S* 是否在整数模式中。
INT?	[h][TEST]INT?	¬INPUT	(0)判断 *x* 是否为一个整数，即小数部分等于零。注意与 FP?的区别。
IP	[f][IP]	¬INPUT	(1)返回 *x* 的整数部分。
iRCL	[h][X.FCN]iRCL*s*	¬INPUT	(-1)假设源数据包含整数数据，并以这种方式调用它们。见 152 页。
ISE	[h][P.FCN]ISE*s*	¬INPUT	(0)与 ISG 类似，但当 ccccccc≥fff 时跳过。

名称	按键	模式	备注（通用说明见 77 页）
ISG	[g][ISG]*s*	¬INPUT	(0)对于源数据中指定的 ccccc.fffii，ISG 将循环控制数 *s* 递增 ii，若 ccccccc>fff 则跳过下一行程序。 若 *s* 没有小数部分则 fff=0。 如果 ii=0，则 ccccccc 按 1 递增。 ISG 不会改变 L，即使目标地址为 **X**。 请注意 fff 和 ii 均不可为负，但 ccccc 可以。
ISZ	[h][P.FCN]ISZ*s*	¬INPUT	(0)将 *s* 加 1，若 | *s* | <1 则跳过下一行程序。不会改变 L，即使目标地址为 **X**。和 *HP-16C* 上一样。
I_x	[h][X.FCN]I_x	DECM	返回正则化 Beta 函数。详见附录 I。
JG1582	[h][MODE]JG1582	All	(0)这两个命令反映了格里高利历在世界两个不同地区所引入年份的不同。D→J 和 J→D 将会根据此设置进行计算。参见 40 页。
JG1752	[h][MODE]JG1752		
J→D	J→D	DECM	(1)根据所选的 JG 设置和格式，将 *x* 作为儒略日并将其转换为普通日期。
KEY?	[h][TEST]KEY?*s*	¬INPUT	(0)判断在例程运行或暂停时是否按下了键。如果在此时间内没有按下键，在 KEY?之后的下一行程序将会被执行，否则的话它将被跳过，并且该按键的代码将存储在指定的地址中。按键码反映了该按键在键盘上的行数和列数（参见 74 页）。
KTP?	[h][P.FCN]KTP?*s*	¬INPUT	(-1)假设指定地址中有一个按键码（参见 KEY?）。检查此代码并返回相应的按键类型： ● 0…9，如果为**[0]**…**[9]** ， ● 10，如果为**[.]**，**[EEX]**或**[+/-]**， ● 11，如果为[f]，[g]，或[h]， ● 12，如果为其他按键。 如果寄存器中的按键码是无效的，则会抛出“Invalid Range”错误。 可用于用户与例程的交互（参见 74 页）。
LASTx	**[RCL][L]**	¬INPUT	(-1)详见 18 和 31 页。实际上，该指令在例程中被记录为 RCL L。
LBL	[f][LBL]***label***	STO	(0)标记用于执行和分支的例程。关于指定程序标签见 68 页。
LBL?	[h][TEST]LBL?***label***	¬INPUT	(0)判断程序内存中是否存在指定的标签。详见 LBL。
LCM	[h][X.FCN]LCM	¬INPUT	(2)返回 *x* 和 *y* 的最小公倍数。结果总为正数。
LEAP?	[h][TEST]LEAP?	DECM	(0)以 *x* 作为选定格式的日期，提取年份，并判断其是否为闰年。
		Integer	(0)以 *x* 作为选定格式的日期判断其是否为闰年。
LgNrm	[h][PROB]LgNrm	DECM	(1)以 **J** 中给出的 μ 和 **K** 中给出的 σ 表示的对

名称	按键	模式	备注（通用说明见 77 页）
LgNrmP	[h][PROB]LgNrmP		数正态分布。详见附录 I。 对于 **X** 中给出的 *p*，**J** 中给出的 *μ* 和 **K** 中给出的 *σ*，LgNrm^{-1} 返回 *x*。
LgNrmu	[h][PROB]LgNrmu		
LgNrm^{-1}	[h][PROB]LgNrm-1		
LINEQS	[h][MATRIX]LINEQS	¬INPUT	(3)解一个线性方程组 $(Z)\cdot\vec{x}=\vec{y}$。基准地址在 **X** 中，向量描述符在 **Y** 中，方阵描述符在 **Z** 中。将解向量的描述符返回到 **X** 中。
LinF	[h][MODE]LinF	All	(0)选择线性曲线拟合模型 $R(x)=a_0x+a_1x$。
LJ	[h][X.FCN]LJ	Integer	(-1)在其字长内，左对齐一个位模式，与 *HP-16C* 上的类似：堆栈将上升，将左对齐的字放在 **Y** 中，将移位计数（左对齐这个字所需的移位数）放在 **X** 中。 示例（对于 8 位的字）：10110_2，LJ 结果为 ***x***=3 和 ***y***=10110000_2。
LN	[g][LN]	¬INPUT	(1)返回 ***x*** 的自然对数
L_n	[h][X.FCN]Ln	DECM	(2)拉盖尔多项式。详见附录 I。
LN1+x	[h][X.FCN]LN1+x	DECM	(1)对于 x≈0，这将返回一个比 ln(x)更精确的小数部分结果。
$L_n\alpha$	[h][X.FCN]Lnα	DECM	(3)拉盖尔广义多项式。详见附录 I。
LN β	[h][X.FCN]LNβ	DECM	(2)返回欧拉 Beta 函数的自然对数。参见 β 。
LNΓ	[h][X.FCN]LΓ	DECM	(1)返回 $\Gamma(x)$的自然对数。也可以用来计算非常大的阶乘。 示例：5432!是多少? 请记住 $\Gamma(x+1)=x!$所以，输入 5433 LNΓ 10**[LN][/]** 返回 17931.480374，这是结果的常用对数；然后调用**[FP]**[10^x]返回它的有效数字部分 3.0225535984。因此 $5432!=3.02\cdot10^{17931}$。
LOAD	[h][P.FCN]LOAD	¬INPUT	从 *FM* 恢复全部备份，即执行 LOADP，LOADR，LOADSS，LOADΣ，并返回 Restored。不可用于编程。与 SAVE 相对应。参见下面的指令和附录 A。
LOADP	[h][P.FCN]LOADP	¬INPUT	(0)将备份中的所有程序加载到 *RAM* 中已有的程序之后。只有当 *RAM* 有足够的空间时才能运行，否则将抛出错误信息。不可用于编程。
LOADR	[h][P.FCN]LOADR	¬INPUT	(0)从备份中恢复数字编号通用寄存器（见 SAVE）。字母编号寄存器不会被调出。被复制的寄存器数量，是备份中的和 *RAM* 所分配的二者最小值。
LOADSS	[h][P.FCN]LOADSS	¬INPUT	(0)从备份中恢复系统状态。更多内容见附录 B。
LOADΣ	[h][P.FCN]LOADΣ	¬INPUT	(0)从备份中恢复求和寄存器。如果备份中没有，将抛出错误信息。更多内容见附录 B。
LocR	[h][P.FCN]LocR***n***	¬INPUT	(0)为当前例程分配 ***n*** 个本地寄存器（≤144 个）和 16 个本地标签。参见第 69 页。
LocR?	[h][P.FCN]LocR?	¬INPUT	(-1)返回当前分配的本地寄存器数量。

名称	按键	模式	备注（通用说明见 77 页）
LOG$_{10}$	[g][LG]	¬INPUT	(1)返回 x 以 10 为底的对数。
LOG$_2$	[g][LB]	¬INPUT	(1)返回 x 以 2 为底的对数。
LogF	[h][MODE]LogF	All	(0)选择对数曲线拟合模型： $R(x) = a_0 + a_1 \ln x$。
Logis	[h][PROB]Logis	DECM	(1)以 **J** 中给出的 μ，及 **K** 中给出的 s 的表示的逻辑斯蒂分布。更多内容见附录 I。
Logis$_P$	[h][PROB]Logis.		
LOGIS$_U$	[H][PROB]LOGIS.		
Logis^{-1}	[h][PROB]Logis^{-1}		
LOG$_X$	[g][LOG$_X$]		(2)返回 y 以 x 为底的对数。
	[CPX][g][LOG$_X$]	DECM	(2)返回 z+it 以 x+iy 为底的复对数。
LZOFF	[h][MODE]LZOFF	All	(0)切换前导零，类似 *HP-16C* 的标志位 3。只与 2，4，8 和 16 进制有关。
LZON	[h][MODE]LZON		
L.R.	[h][STAT]L.R.	DECM	(-2)根据拟合类型（参见 LINF，EXPF，POWERF 和 LOGF），返回求和寄存器中数据点的拟合曲线的参数 a_1（X 中）和 a_0（Y 中）。 对于直线（LINF），a_0 是截距，a_1 是斜率。更多内容见附录 I。
MANT	[h][X.FCN]MANT	DECM	(1)返回科学记数法$x = m \cdot 10^k$中的有效数字 m。注意和 EXPT 的区别。
MASKL	[h][X.FCN]MASKL*n*	Integer	(-1)类似 *HP-16C* 的 MASKL 和 MASKR，但在指令后（而不是 **X** 中）读取掩码长度或地址。之后掩码会被推进栈中。 示例：对于 WSIZE8，MASKL 3 返回掩码 11100000_2。之后，就可以使用 AND 从任意字节提取三个最高有效位。
MASKR	[h][X.FCN]MASKR*n*		
	[h][MATRIX]	DECM	矩阵菜单，参见第 117 页。
MAX	[h][X.FCN]MAX	¬INPUT	(2)返回 x 和 y 的最大值。
MEM?	[h][P.FCN]MEM	¬INPUT	(-1)返回程序存储中剩余容量（以字计），被分配的本地寄存器考虑在内。
MIN	[h][X.FCN]MIN	¬INPUT	(2)返回 x 和 y 的最小值。
MIRROR	[h][X.FCN]MIRROR	Integer	(1)翻转 x 的位模式(例如，在字长 8 的情况下，由 00010111_2 生成 11101000_2)。
MOD	[h][X.FCN]MOD	¬INPUT	(2)返回 y mod x（示例见第 59 页）。注意和 RMDR 的不同。 示例： $\mathrm{mod}(25;7)=4$； $\mathrm{mod}(-25;7)=3$； $\mathrm{mod}(25;-7)=-3$； $\mathrm{mod}(-25;-7)=-4$。
	[h][MODE]	All	模式菜单。见第 117 页。
MONTH	[h][X.FCN]MONTH	DECM	(1)假设 x 包含一个选定格式的日期，提取出其中的月份。

名称	按键	模式	备注（通用说明见 77 页）
MROW+×	[h][MATRIX]MROW+×	DECM	(0)读取一个矩阵描述符 x，一个目标行号 y，一个源行号 z，和一个实数 t。将(X)中的元素 m_{zi} 乘以 t，并把它加到 m_{yi} 上。堆栈不会被改变。MROW+×和 *PPC M3* 相似。
MROW×	[h][MATRIX]MROW×	DECM	(0)读取一个矩阵描述符 x，一个行号 y，和一个实数 z。将(X)的元素 m_{yi} 乘以 z。堆栈不会被改变。MROW×和 *PPC M2* 相似。
MROW⇆	[h][MATRIX]MROW⇆	DECM	(0)读取一个矩阵描述符 x，两个行号 y 和 z。将(X)的 y 行和 z 行交换。堆栈不会被改变。MROW+×和 *PPC M1* 相似。
MSG	[h][P.FCN]MSG*n*	¬INPUT	(0)抛出指定的错误消息。这是一个临时信息。对应的错误代码见附录 C。注意和 ERR 的不同。
M+×	[h][MATRIX]M+×	DECM	(3)读取两个矩阵描述符 x 和 y，以及一个实数 z。返回$(X)+(Y)\cdot z$。这样，一个矩阵的数乘被加到另外一个矩阵上。这个乘法和加法，是通过内部的高精度（39 位）完成的。结果会被精确地舍入。
M^{-1}	[h][MATRIX]M-¹	DECM	(0)读取 **X** 中的方阵描述符，在同一位置求出逆矩阵。堆栈不会被改变。
M-ALL	[h][MATRIX]M-ALL	DECM	(1)读取一个矩阵描述符 x[62]，返回一个这个矩阵中适合 ISG 和 DSL 循环的计数值。这个循环能够处理(X)中所有的元素。如果描述符是负数，调用适合 DSL 的循环控制数；反之调用适合 ISG 的循环控制数。
M-COL	[h][MATRIX]M-COL	DECM	(2)读取一个矩阵描述符 x（和 M-ALL 一样可正可负），和一个列号 y。返回一个只处理这个矩阵该列中元素 m_{iy} 的循环控制数。
M-DIAG	[h][MATRIX]M-DIAG	DECM	(1)读取一个矩阵描述符 x（和 M-ALL 类似），返回一个循环控制数。这个循环控制数能够处理这个矩阵(X)对角线中的所有元素 m_{ii}。
M-ROW	[h][MATRIX]M-ROW	DECM	(2)读取一个矩阵描述符 x（和 M-ALL 类似），和一个行号 y。返回一个只处理这个矩阵该行中的元素 m_{yi} 的循环控制数。
M×	[h][MATRIX]M×	DECM	(3)读取两个矩阵描述符 y 和 z。以 x 的整数部分作为结果的基地址，返回$(Z)\cdot(Y)=(X)$。所有的计算是以内部的高精度（39 位）完成的。x 的小数部分被直接覆写以匹配结果矩阵。
M.COPY	[h][MATRIX]M.COPY	DECM	(2)读取一个矩阵描述符 y 和一个基准寄存器号 x。复制矩阵(Y)到以 **R***x* 开始的寄存器中。在 **X** 中返回一个被正确格式化的矩阵描述符。
M.DY	[h][MODE]M.DY	All	(0)设置日期显示的格式为 `mm.ddyyyy`。

[62]描述符将像往常一样保存在 **L** 中。这对其他矩阵操作也适用。

名称	按键	模式	备注（通用说明见 77 页）
M.IJ	[h][MATRIX]M.IJ	DECM	读取一个矩阵描述符 x 和一个寄存器号 y。将该寄存器所在的列号返回 $\mathbf{Y}$，所在的行号返回 $\mathbf{X}$。M.IJ 和 *PPC M4* 相似。注意和 M.REG 的区别。
M.LU	[h][MATRIX]M.LU	DECM	(1)读取 $\mathbf{X}$ 中的方阵描述符。将($\boldsymbol{X}$)原地变换成它的 LU 分解。$\mathbf{X}$ 中的值被替换成一个定义计算分解所需主元的描述符。最高位的数字表示第一个对角线元素对应的主元，第二位数字表示第二个对角线元素对应的主元，以此类推[63]。
M.REG	[h][MATRIX]M.REG	DECM	(3)读取一个矩阵描述符 x，一个行号 y，和一个列号 z。M.REG 返回对应的寄存器号给 $\mathbf{X}$。和 *PPC M5* 相似。注意和 M.IJ 的不同。
M.SQR?	[h][TEST]M.SQR?	DECM	(0)读取一个矩阵描述符 x，如果这个矩阵是方阵，返回 true。
NAND	[h][X.FCN]NAND	¬INPUT	(2)和 AND 属于一类。参见 57 页。
NaN?	[h][TEST]NaN?	¬INPUT	(0)判断 x 是否“不是一个数（NaN）”。
nBITS	[h][X.FCN]nBITS	Integer	(1)统计 x 中置位的位数，类似 *HP-16C* 的#B。
nCOL	[h][MATRIX]nCOL	DECM	(1)读取一个矩阵描述符 x，返回($\boldsymbol{X}$)的列数。
NEIGHB	[h][X.FCN]NEIGHB	DECM	(2)根据设置的模式，返回 x 对 y 方向的最近的机器可表示数[64]。对于 $x<y$（或者 $x>y$），这是 x 的后一个相邻机器数（或是前一个相邻机器数）；对于 $x=y$，就是 y。
		Integer	(2)对于 $x<y$ 返回 $x+1$，$x=y$ 返回 y，$x>y$ 返回 $x-1$。
NEXTP	[h][X.FCN]NEXTP	¬INPUT	(1)返回比 x 大的下一个质数。
NOP	[h][P.FCN]NOP	¬INPUT	(0)空的程序步。
NOR	[h][X.FCN]NOR	¬INPUT	(2)和 AND 属于一类。参见 57 页。
Norml	[h][PROB]Norml	DECM	(1)以 $\mathbf{J}$ 给出的均值 $\boldsymbol{\mu}$ 及 $\mathbf{K}$ 给出的标准差 $\boldsymbol{\sigma}$ 表示的正态分布。更多内容见附录 I。 Norml^{-1} 返回在 $\mathbf{X}$ 给出概率 p，$\mathbf{J}$ 给出 $\boldsymbol{\mu}$，及 $\mathbf{K}$ 给出 $\boldsymbol{\sigma}$ 时，对应的 x。
Norml$_p$	[h][PROB]Norml▸		
Norml$_u$	[h][PROB]Norml↓		
Norml^{-1}	[h][PROB]Norml⁻¹		
NOT	[h][NOT]	Integer	(1)对 x 位取反，与 *HP-16C* 的相同。
		DECM	(1)$x=0$ 则返回 1，$x\neq 0$ 则返回 0。
nROW	[h][MATRIX]nROW	DECM	(1)读取一个矩阵描述符 x，返回($\boldsymbol{X}$)的行数。
nΣ	[h][SUMS]nΣ	DECM	(-1)调出当前累积的数据点的个数。
ODD?	[h][TEST]ODD?	¬INPUT	(0)检验 x 是否是一个整数且是一个奇数。
OFF	[h][OFF]	STO	(0)插入一步程序控制的关机指令。
		¬STO	(0)关闭 *WP 34S*。

[63]译者注：这一部分原文写的较为模糊。根据 *Paul Dale* 在 MoHPC 的帖子答复，这个主元描述符记录了分解过程中行交换的顺序，第一行记为 0。例如：无行交换的 3*3 矩阵分解，描述符为 012，即 12。若描述符为 112，则表示第二行排第一，然后矩阵剩余的行按顺序排列（1→12）。LINEQS 指令也会对矩阵进行 LU 分解但不存储分解矩阵。参见 *https://www.hpmuseum.org/forum/thread-13972.html*。

[64]NEIGHB 对研究数值稳定性非常有用。请参阅 *HP-71* 数学软件包中的"NEIGHBOR"。

名称	按键	模式	备注（通用说明见 77 页）
OR	[h][OR]	¬INPUT	(2)和 AND 属于一类。参见 57 页。
PERM	[g][Py, x]	¬INPUT	(2)返回从 y 个元素中取出 x 个元素的排列数。每个排列中没有重复元素，不同顺序的相同 x 个元素的排列被分别计数。公式见附录 I。注意和 COMB 的区别。
P_n	[h][X.FCN]Pn	DECM	(1)勒让德多项式。更多内容见附录 I。
Poiss	[h][PROB]Poiss	DECM	(1)以 **X** 给出的成功次数 g，**J** 给出的总的错误概率 p_0，**K** 给出的样品数量 n 表示的泊松分布。泊松参数$\lambda = \mathrm{n} \cdot p_0$是自动计算的。见下面的 Pois λ 。 Poiss^{-1} 返回以 **X** 给出概率 p，**J** 给出 p_0，**K** 给出 ***n*** 时对应的最大成功次数 ***m***。
Poiss_P	[h][PROB]Poiss_P		
Poiss_u	[h][PROB]Poiss_u		
Poiss^{-1}	[h][PROB]Poiss^-1		
Pois λ	[h][PROB]Pois λ	DECM	(1)Pois λ 类似 Poiss，但是 λ 由 **J** 给出，不使用 **K**。更多内容见附录 I。 Pois λ $^{-1}$ 返回以 **X** 给出概率 p，**J** 给出 λ 时的最大成功次数 ***m***。
Pois λ_P	[h][PROB]Pois λ_P		
Pois λ_u	[h][PROB]Pois λ_u		
Pois λ $^{-1}$	[h][PROB]Pois λ^-1		
PopLR	[h]P.FCN]PopLR	¬INPUT	(0)不从程序中返回的前提下，释放被分配到当前程序的本地寄存器。参见 LocR 和 RTN。
PowerF	[h][MODE]PowerF	All	(0)选择幂函数曲线拟合模型$R(x) = a_0 x^{a_1}$。
PRCL	[h]P.FCN]PRCL	¬INPUT	(0)从 *FM* 或 *RAM* 复制当前的程序并把它加载到 *RAM*，之后可被编辑（见第 72 页）。允许重复拷贝 *RAM* 中的程序。只有当 *RAM* 有足够的空间才能使用。 通过调用 *FM* 中库的程序，编辑它，然后 PSTO，可以实现对这部分 *FM* 库的编辑。
	[RCL]	CAT 菜单打开	
PRIME?	[h][TEST]PRIME?	¬INPUT	(0)检验 x 的整数部分绝对值是否是一个质数。这个方法对 $9 \cdot 10^{18}$ 以下的整数有效。
	[h][PROB]	DECM	概率菜单。参见第 117 页。
PROFRC	[f][ab/c]	¬INPUT	(0)设置只允许显示真分数或是带分数的分数模式。根据 DEN...的设定，将 x 转换成真分数，例如，将 1.25 或$\frac{5}{4}$转换成$1\frac{1}{4}$。参见 43 页。 注意和 IMPFRC 的区别。
PROMPT	[h]P.FCN]PROMPT	¬INPUT	(0)显示 ***alpha*** 并终止程序执行。更多内容见第 72 页。
PSE	[h][PSE]***n***	¬INPUT	(0)程序运行时，刷新显示并暂停程序运行 ***n*** 个 tick（参见 TICKS），***n*** 的范围是 0≤***n***≤99。按下按键会提前终止暂停。

名称	按键	模式	备注（通用说明见 77 页）
PSTO	[h]P.FCN]PSTO	¬INPUT	(0)复制 *RAM* 中的当前程序，追加到 *FM* 库末尾。不可用于编程。 这个当前程序必须有至少一个字母数字的全局标签 LBL 语句（最好是在开始处）。如果库中的一个程序有相同的标签，会先删除这个程序。*FM* 中的字母数字标签可以在 CAT（参见第 119 页）中浏览并被 XEQ 调用。 可以通过 PRCL 调用 *FM* 中的库程序，编辑它，然后 PSTO 来编辑这部分 *FM* 库。
PUTK	[h]P.FCN]PUTK*s*	¬INPUT	(0)在指定的地址假定有一个按键码。停止程序运行，将这个按键码放进键盘缓冲区，使其立刻被执行。需要**[R/S]**恢复程序运行。可以用于例程中的用户互动。
	[h]P.FCN]	¬INPUT	程序功能菜单。见第 117 页。
RAD	[g][RAD]	DECM	(0)将当前的角度模式设置为弧度制。
RAD→	[h][X.FCN]RAD→	DECM	(1)将 ***x*** 视为弧度制，并转换到当前设置的角度模式。
RAN#	[f][RAN#]	DECM	(-1)类似 *HP-42S* 的 RAN，返回 0 到 1 之间的随机数。
		Integer	(-1)返回一个指定的字长的随机位模式。
RCL	**[RCL]*s***	¬INPUT	(-1)调用寄存器的内容。与 ᶜRCL 相关的内容参见 33 页。
RCLM	**[RCL][MODE]*s***	¬INPUT	(0)调用 STOM 存储的模式。
RCLS	[h]P.FCN]RCLS*s*	¬INPUT	从地址 ***s*** 开始，调用 4 个或 8 个寄存器的值，并将其压入栈。这是 STOS 的反命令。
RCL+	**[RCL][+]*s***	¬INPUT	(1)调用 ***s***，执行相应的操作，并将结果压入堆栈。 例如，RCL-12 从 ***x*** 中减去 ***r12*** 并显示结果（类似**[RCL][1][2][-]**，但不丢失顶层堆栈的内容）。 类似地，ᶜRCL-12 从 ***x*** 中减去 ***r12***，从 ***y*** 中减去 ***r13***。
RCL-	**[RCL][-]*s***		
RCL×	**[RCL][×]*s***		
RCL/	**[RCL][/]*s***		
RCL↑	**[RCL][▲]*s***	¬INPUT	(1)RCL↑(↓)将 ***x*** 替换成 ***s*** 和 ***x*** 中最大的（最小的）数。
RCL↓	**[RCL][▼]*s***		
RDP	[h]X.FCN]RDP*n*	DECM	(1)根据 RM 设置将 ***x*** 保留到小数点后 ***n*** 位（0≤***n***≤99，注意 FIX 设置）。参见 RM 并注意和 RSD 的区别。 示例：1.23456789E-95 RDP **99**将返回 12346⁻⁹⁵
RDX,	[h][MODE]RDX,	All	(0)将小数点设置成逗号。
	[h][./,]	DECM	(0)切换两种小数点。
RDX.	[h][MODE]RDX.	All	(0)将小数点设置成点号。
REALM?	[h][TEST]REALM?	¬INPUT	(0)判断 *WP 34S* 是否在实数模式（DECM）。

名称	按键	模式	备注（通用说明见 77 页）
RECV	[h]P.FCN]RECV	¬INPUT	(0)准备从串口接收数据。更多内容见 SEND…和附录 A。
REGS	[h]MODE]REGS*n*	All	(0)指定全局通用寄存器数量。REGS 100 为默认状态（**R00-R99**），REGS 0 则无寄存器可用。
REGS?	[h]P.FCN]REGS?	¬INPUT	(-1)返回分配的全局通用寄存器数量（0 到 100 之间）。
RESET	[h]P.FCN]RESET	¬INPUT	如果确认，执行 CLALL 并重置所有模式为默认值，即：24h，2COMPL，ALL 0，DBLOFF，DEG，DENANY，DENMAX0，D.MY，TSON，JG1752，LinF，LocR 0，LZOFF，PROFRC，RDX.，REGS 100，RM 0，SCIOVR，SEPON，SSIZE4，WSIZE 64，YDOFF，最后是 DECM。更多内容参见这些指令。不可用于编程。
RJ	[h]X.FCN]RJ	Integer	(-1)右对齐，类似 *HP-16C* 的 LJ。 例如：101100_2RJ 结果是 $y=1011_2$ 和 $x=2$。参见 LJ。
RL	[h]X.FCN]RL*n*	Integer	(1)类似 *n* 个连续的 *HP-16C* 的 RL/RLC，即 RLn/RLCn。对于 RL，0≤n≤63。对于 RLC，0≤n≤64。RL 0/RLC 0 会视为无操作，但是改变 L。更多内容见第 57 页。
RLC	[h]X.FCN]RLC*n*		
RM	[h]MODE]RM*n*	All	(0)设置小数的舍入方式。只有将内部的高精度格式（39 位）转换成紧凑的实际数字才会用到。不会改变显示，也不会改变 ROUND 的行为。支持下列模式： 0：四舍六入五成双：1/2=0.5 舍入到下一个偶数（默认）。 1：四舍五入：≥0.5 进位。 2：五舍六入：≤0.5 舍去。 3：进位：向远离 0 的方向舍入。 4：舍去：向靠近 0 的方向舍入。 5：向上舍入：向正无穷的方向舍入。 6：向下舍入：向负无穷的方向舍入。
RMDR	[h]RMDR]	¬INPUT	(2)返回除法的余数。和 *HP-16C* 的 RMD 相同，但也适用于实数。示例见第 59 页。注意和 MOD 的不同。 示例： $\mathrm{rmdr}(25;7)=4$； $\mathrm{rmdr}(-25;7)=-4$； $\mathrm{rmdr}(25;-7)=4$； $\mathrm{rmdr}(-25;-7)=-4$。
RM?	[h]P.FCN]RM?	¬INPUT	(-1)返回设置的浮点数舍入方式。更多内容见 RM。
ROUND	[g][RND]	¬INPUT	(1)使用当前的显示格式舍入 x，类似 *HP-42S* 的 RND。
		FRC	(1)使用当前的分母舍入 x，类似 *HP-35S* 分数模式下的 RND。

名称	按键	模式	备注（通用说明见 77 页）
ROUNDI	[h]X.FCN]ROUNDI	¬INPUT	(1)将 ***x*** 舍入到最近的整数。1/2 舍入到 1。
RR	[h]X.FCN]RR*n*	Integer	(1)类似 *n* 个连续的 *HP-16C* 的 RRs/RRCs，即 RRn/RRCn。对于 RR，0≤n≤63。对于 RRC，0≤n≤64。RR 0/RRC 0 视为无操作，但是会改变 L。更多内容见第 57 页。
RRC	[h]X.FCN]RRC*n*		
RSD	[h]X.FCN]RSD*n*	DECM	(1)根据 RM 设置将 ***x*** 舍入到 ***n*** 个有效数字（1≤***n***≤34）。参见 RM，并注意和 RDP 的不同。
RTN	[g][RTN]	STO	(0)例程的最后一个可执行指令。
		¬STO	在运行的程序中：释放当前的本地数据（类似 PopLR）并将控制交还给调用例程，即将程序指针移动到调用这个例程的 XEQ 指令的下一步。如果没有找到调用例程（例如程序本身就是最高层级），则停止程序运行，程序指针指向第 000 步，并点亮 **BEG** 指示符。 在运行模式下按下：重设程序指针到 *RAM* 程序的开始，并点亮 **BEG** 指示符。
RTN+1	[h]P.FCN]RTN+1	¬INPUT	(0)类似 RTN，但是在 STO 模式下将程序指针移到调用这个例程的 XEQ 指令的后两步。如果没有指令，则停止程序运行。
R-CLR	[h]P.FCN]R-CLR	DECM	(0)将 ***x*** 解释成 `sss.nn`。清除从地址 ***sss*** 开始的 ***nn*** 个寄存器。 示例：对于 ***x***=34.567，R-CLR 会清除 **R34** 到 **R89**。 **注意：** 对于 ***nn***=0，会清除到最大可用的寄存器。 对于 0 到 99 之间的 ***sss***，会清除到分配的最高全局寄存器。 对于 100 到 111 之间的 ***sss***，会清除到 **K**。 对于大于等于 112 的 ***sss***，会清除到分配的最高本地寄存器。
R-COPY	[h]P.FCN]R-COPY	DECM	(0)将 ***x*** 解释成 `sss.nnddd`。读取从地址 ***sss*** 开始的 ***nn*** 个寄存器，复制它们的值到 ***ddd*** 等。如果 ***nn***=00，会读取到最大可用的寄存器。对于复制矩阵也很有用。 对于 ***x***<0，R-COPY 会从 *FM* 中读取从|***sss***|开始的 ***nn*** 个寄存器，复制目的地永远是 *RAM*。 示例：对于 ***x***=7.03045678，***r07***，***r08***，***r09*** 会被分别复制到 **R45**，**R46**，**R47**。 **注意：**同 R-CLR 的注意事项，只是“清除”改为“复制”。

名称	按键	模式	备注（通用说明见 77 页）
R-SORT	[h][P.FCN]R-SORT	DECM	(0)将 ***x*** 解释成 sss.nn。对从地址 ***sss*** 开始的 ***nn*** 个寄存器进行排序。如果 ***nn***=0，会排序到最大可用的寄存器。 示例：假设 ***x***=49.0369，***r49***=1.2，***r50***=-3.4，***r51***=0；然后 R-SORT 会返回 ***r49***=-3.4，***r50***=0，***r51***=1.2。 **注意：**同 R-CLR 的注意事项，只是“清除”改为“排序”。
R-SWAP	[h][P.FCN]R-SWAP	DECM	(0)类似 R-COPY，但是交换（而不是复制）来源和目的寄存器的内容。
R→D		DECM	(1)见转换菜单。将弧度转换为角度。
R↑	[h][R⇧]	¬INPUT	将堆栈的内容分别向上或向下滚动一层。详见第 18 页、第 31 页。
R↓	[R⇩]		
s	[g][s]	DECM	(-2)读取累积的统计数据，计算出样本标准差 s_x 和 s_y 并将其压入堆栈。公式见附录 I。
SAVE	[h]P.FCN]SAVE	¬INPUT	(0)保存所有用户程序、寄存器内容和系统状态到 *FM*，并返回 Saved。使用 LOAD 调用备份。不可用于编程。更多内容见附录 A。
SB	[h]X.FCN]SB***n***	Integer	(1)将 ***x*** 中指定的位置位。
SCI	[h]SCI]***n***	¬INPUT	(0)设置科学计数法的格式（见第 41 页）。
SCIOVR	[h]SCI][ENTER⇧]	¬INPUT	(0)定义超过 ALL 或 FIX 显示范围的数，会被显示成科学计数法的格式。注意和 ENGOVR 的不同，参见 RESET。
SDL	[h][X.FCN]SDL***n***	DECM	(1)将数字向左（右）移动 n 位数，相当于将 ***x*** 乘以（除以）10^n。对于整数参见 SL 和 SR。
SDR	[h][X.FCN]SDR***n***		
SEED	[h][STAT]SEED	DECM	(0)存储用于生成随机数的种子。
SENDA	[h][P.FCN]SENDA 等.	¬INPUT	(0)SENDA 将所有的 *RAM* 数据，SENDP 将程序内存，SENDR 将全局通用寄存器，SENDΣ 将求和寄存器通过串口发送到和计算器相连接的设备上。更多信息参见 RECV 和附录 A。
SENDP			
SENDR			
SENDΣ			
SEPOFF	[h][MODE]SEPOFF	All	(0)切换整数的位数组分隔符。每隔如下位数显示点号或逗号： 2 和 4 进制：4 位； 16 进制：2 位； 其余进制：3 位。. 整数模式下可以使用[h][./,]。
SEPON	[h][MODE]SEPON		
SERR	[h][STAT]SERR	DECM	(-2)读取累积的统计数据，计算并返回标准误差（即 $\bar{x}$ 和 $\bar{y}$ 的标准差）。
$SERR_w$	[h][STAT]SERRw	DECM	(-1)返回加权数据的标准误差，例如 $\bar{x}_w$ 的标准差。
SETCHN	[h][MODE]SETCHN	All	(0)区域设置（参见 40 页）。
SETDAT	[h][MODE]SETDAT	All	(0)设置实时时钟的日期（模拟器将从 PC 时钟获取）。

名称	按键	模式	备注（通用说明见 77 页）
SETEUR SETIND SETJPN	[h][MODE]SETEUR 等.	All	(0)区域设置（参见 40 页）。
SETTIM	[h][MODE]SETTIM	All	(0)设置实时时钟的时间（模拟器将从 PC 时钟获取）。
SETUK SETUSA	[h][MODE]SETUK 等.	All	(0)区域设置（参见 40 页）。
SF	[f][SF]***n***	All	(0)对指定的标志位置位。
SHOW	[g][SHOW]	All	寄存器浏览器，参见 117 页。
SIGN	[h][X.FCN]SIGN	¬INPUT	(1)x>0 时返回 1，x<0 时返回-1，x=0 或 x 为非数字值时，返回 0。相当于 signum(x)。
CSIGN	[CPX][h][X.FCN]CSIGN	DECM	(1)在 **X**，**Y** 中返回 $x+iy$ 的单位向量。
SIGNMT	[h][MODE]SIGNMT	All	(0)对整数设置原码模式。参见 54 页.
SIN	[f][SIN]	DECM	(1)返回 **X** 中角度的正弦值。
SINC	[h][X.FCN]SINC	DECM	(1)返回 sin(x)/x；x=0 时返回 1。
SINH	[f][HYP][SIN]	DECM	(1)返回角 x 的双曲正弦值。
SKIP	[h][P.FCN]SKIP***n***	¬INPUT	(0)向后跳转 ***n*** 步（0≤n≤255）程序。例如 SKIP2 跳过接下来的两步，比如从第 123 步跳到第 126 步。如果跳转越过了程序的 END 语句，会抛出一个错误消息。 **注意：**如果在一个程序中使用了一个或多个 BACK，SKIP，或 CASE 跳转操作，这可能会导致需要单独手动维护那些语句。
SL	[h][X.FCN]SL***n***	Integer	(1)类似 ***n***(≤63)个连续的 *HP-16C* 上的 SL，SL0 会被视为无操作，但是改变 L。更多内容见第 57 页。
SLOW	[h][MODE]SLOW	All	(0)将处理速度设置为“慢速”，低电量时将自动执行（参见 36 页）。参见 FAST。
SLV	[f][SLV]***label***	DECM	解方程 **f**(x)=0，**f**(x)由指定的例程计算。调用时必须在 **X** 和 **Y** 中提供解的两个初始预测值。用户界面和 *HP-15* 一样。这意味着 x_{root} 返回到 **X** 中，上一个估计的 x 值返回到 **Y** 中，并且 f(x_{root})被返回到 **Z** 中。另外 SLV 也会执行判断，如果 SLV 没有找到根，下一步程序会被跳过。请参考 *HP-15C* 用户手册（13 节和附录 D）以获取自动寻根的更多信息。 注意：SLV 在调用指定的例程之前，会用 x 填充所有的级别的堆栈。

名称	按键	模式	备注（通用说明见 77 页）
SLVQ	[h][X.FCN]SLVQ	DECM	解一元二次方程 $ax^2+bx+c=0$。实系数为输入到堆栈的三个参数[***c***, ***b***, ***a***,...]，并且对结果进行判断： ·若 $r=b^2-4ac\geq 0$，则在 **Y** 和 **X** 中返回 $\frac{-b\pm\sqrt{r}}{2a}$。 在程序中，SLVQ 的下一步程序将被执行。 ·否则，SLVQ 在 **X** 中返回第一个复数根的实部并在 **Y** 中返回其虚部（第二个根是第一个根的共轭复数）。如果直接从键盘上运行，复数指示符 **i** 会点亮。在程序中，SLVQ 后面一步程序会被跳过。 无论那种情况，SLVQ 都会在 **Z** 中返回 ***r***。更高级别的堆栈保持不变。**L** 中为参数 c。
SMODE?	[h][P.FCN]SMODE?	¬INPUT	(-1)返回整数的符号模式： 补码返回 2(true)； 反码返回 1(true)； 无符号数返回 0(false)； 原码返回-1(true)。
SPEC?	[h][TEST]SPEC?	¬INPUT	(0)若 ***x*** 为特殊值，如非数字（non-numeric）或无穷大，返回 True。
SR	[h][X.FCN]SR***n***	Integer	(1)类似 ***n***(≤63)个连续的 *HP-16C* 上的 SR，SR 0 会被视为无操作，但是改变 L。更多内容见第 57 页。
sRCL	[h][X.FCN]s**RCL*s***	¬INPUT	(-1)认为源码中包含 SP 数据并且以该格式调用它们。参见 152 页与 183 页。
SSIZE4	[h][MODE]SSIZE4	All	将堆栈大小设置为 4 级或 8 级。参见 15 页。注意寄存器中的内容不会改变。当栈大小被其他操作定义时其内容也不会改变。（例如 RCLM，参见 37 页）.
SSIZE8	[h][MODE]SSIZE8		
SSIZE?	[h][P.FCN]SSIZE?	¬INPUT	(-1)返回被分配的堆栈级数。
	[h][STAT]	DECM	统计菜单。参见 117 页。
	[h][STATUS]	¬INPUT	标志位浏览器。参见 117 页。
STO	**[STO]*s***	¬INPUT	(0)将***x***存到指定的地址。参见33页的^CSTO。
STOM	**[STO][MODE]*s***	¬INPUT	(0)存储模式设置以供之后使用，如 37 页所述（这里无需按[h]）。用 RCLM 来调用存储的模式设置。
STOP	**[R/S]**	STO	(0)暂停程序，可以用来等待输入。
STOPW	**[CPX][R/S]**	DECM	基于实时时钟的秒表，和 *HP-55* 的一样。详见 137 页。 **注意：**只有在加载了适当的固件之后，这个指令才会成为功能集的一部分。参见附录 A.。
	[h][X.FCN]STOPW		
STOS	[h][P.FCN]STOS***s***	¬INPUT	(0)将当前所有的堆栈存储在从地址 ***s*** 开始的 4 个或 8 个寄存器里。参见 RCLS。
STO+	**[STO][+]*s***	¬INPUT	(0)将指定的操作作用于 ***s***，并且存储至该地址。参见 33 页的 ^CSTO...。
STO-	**[STO][-]*s***		

名称	按键	模式	备注（通用说明见 77 页）
STO×	**[STO][×]*s***		例如：STO-12 将 ***r12*** 减去 ***x***，如同 **[RCL][12][x⇆y][-][STO][12]**一样，但是堆栈保持不变。
STO/	**[STO][/]*s***		
STO↑	**[STO][▲]*s***	¬INPUT	(0)STO↑（↓）获取 ***s*** 与 ***x*** 的最大值（或最小值），并储存至指定地址。
STO↓	**[STO][▼]*s***		
SUM	[h][STAT]SUM	DECM	(-2)调用线性求和 ***Σy*** 与 ***Σx***。可用于 2 维向量的基本计算。
	[h][SUMS]	DECM	菜单。参见 117 页。
s_w	[h][STAT]s_w	DECM	(-1)计算加权数据的标准差（每个数据点 ***x*** 的权重 ***y*** 通过 Σ+输入）。计算公式见附录 I。
s_{XY}	[h][STAT]s_{xy}	DECM	(-1)通过 Σ+计算两个数据集的样本协方差。它和选择的拟合模型有关。计算公式见附录 I。总体协方差参见 COV。
TAN	[f][TAN]	DECM	(1)返回 ***x*** 的正切值[65]。
TANH	[f][HYP][TAN]	DECM	(1)返回 ***x*** 的双曲正切值。
	[h][TEST]	DECM	菜单。参见 117 页。
TICKS	[h][P.FCN]TICKS	¬INPUT	(-1)从实时时钟中返回开机后运行时间的 tick 数。有内置晶振时，1*tick*=0.1*s*。如果没有晶振，会有百分之十左右的误差。所以长期使用时间功能需要晶振。TICKS 也可以无晶振工作。TICKS 可用于测量程序运行时间。
TIME	[h][X.FCN]TIME	DECM	(-1)调用运行时的实时时钟。以标准的 24h 模式，`hh.mmss` 的格式显示。选择 FIX 4 可以获得最准确的结果。
T_n	[h][X.FCN]T_n	DECM	(2)第一类切比雪夫多项式。详见附录 I
TOP?	[h][TEST]TOP?	STO	(0)如果 TOP?在子程序中被调用，返回 `false`；如果程序运行标志位被置位并且子程序返回堆栈指针（SRS）是空的，返回 `true`。
TRANSP	[h][MATRIX]TRANSP	DECM	(1)读取矩阵描述符 ***x*** 并且返回其转置矩阵的描述符。转置是在原矩阵上完成的，不需要任何额外的内存。
TSOFF	[h][MODE]TSOFF	All	(0)切换 DECM 的千分隔符显示（形状取决于 radix 设置）。
TSON	[h][MODE]TSON		
$t_P(x)$	[h][PROB]$t_P(x)$	DECM	(1)学生 t 分布。自由度被保存在 **J** 中。$t_u(x)$等于 *HP-21S* 中的 Q(t)。参见 49 页和附录 I。
$t_u(x)$	[h][PROB]$t_u(x)$		
t(x)	[h][PROB]t(x)		
$t^{-1}(p)$	[h][PROB]$t^{-1}(p)$		
t⇆	[h][P.FCN]t⇆s	¬INPUT	交换 ***t*** 与 ***s*** 的值，和 ***x***⇆类似。

[65]注意如果标志位 D 被置位，在 x=±90°时 TAN 返回“not numeric”。

名称	按键	模式	备注（通用说明见 77 页）
ULP	[h][X.FCN]ULP	¬INPUT	(1)返回在设置的模式中，可在 ***x*** 中加上或减去而使其发生变化的最小值。在整数模式中，会返回 1。
Un	[h][X.FCN]Un	DECM	(2)第二类切比雪夫多项式。详见附录 I。
UNSIGN	[h][MODE]UNSIGN	All	(0)像 UNSGN 在 *HP-16C* 中一样，设置无符号模式。
VERS	[h][X.FCN]VERS	¬INPUT	(0)显示固件版本和小版本号。
VIEW	[h][VIEW]***s***	¬INPUT	(0)显示 ***s***，直到按下一个键为止。复数的显示格式见 42 页，应用见 72 页。
VIEWα	[h][P.FCN]VIEWα	¬INPUT	(0)在顶部一行显示 ***alpha***，在底部一行显示 - - -，直到一个按键被按下为止（类似 *HP-42S* 中的 AVIEW）。更多信息见 72 页。
	[h][VIEW][-]	INPUT	
VWα+	[h][VIEW]***s***	INPUT	(0)在顶部一行显示 ***alpha***，在底部一行显示 ***s***，直到一个按键被按下为止。更多信息见 72 页。
	[h][P.FCN]VWα+	¬INPUT	
WHO	[h][X.FCN]WHO	¬INPUT	(0)显示制作人员清单，向完成这个项目的勇士们致敬。
WDAY	[h][X.FCN]WDAY	DECM	(1)按指定格式将 ***x*** 作为日期读取，在点阵中返回对应星期的名字，并在数字显示区域显示一个对应的整数（Monday=1，Sunday=7）[66]。
W_m	[h][X.FCN]Wm	DECM	(1)W_p 返回朗伯 W 函数的主分支（***x***≥-1/e）。W_m 返回它的负分支。W^{-1} 为 W_p 的反函数（Wp≥-1）。详见附录 I。
W_p	[h][X.FCN]Wp		
W^{-1}	[h][X.FCN]W-1	DECM	
Weibl	[h][PROB]Weibl	DECM	(1)由 **J** 中给出状参数 ***b***，**K** 中给出特征寿命 ***T*** 确定的威布尔分布。见附录 I。 $Weibl^{-1}$ 对于 **X** 中给定的概率 ***p***，**J** 中给定的 ***b***，**K** 中给定的 ***T***，返回生存时间 t_s。
$Weibl_p$	[h][PROB]Weibl_p		
$Weibl_u$	[h][PROB]Weibl_u		
$Weibl^{-1}$	[h][PROB]Weibl-1		
WSIZE	[h][MODE]WSIZE***n***	All（但仅在整数模式下有效）	(0)几乎和 *HP-16C* 一样，但是指令后带一个参数 ***n***(1≤***n***≤64)，而不是从 **X** 中获取这个值。减小字长会截断所分配的堆栈寄存器和 **L** 中的值。所有其他的内存中的值保持不变（参见附录 H）。增加字长会对每个堆栈增加空位。 WSIZE 0 将字长设为最大，也就是 64 位。
WSIZE?	[h][P.FCN]WSIZE?	¬INPUT	(-1)返回所设定的字长。
x^2	[g][x^2]	¬INPUT	(1)返回 ***x*** 的平方。
x^3	[h][X.FCN]x^3	¬INPUT	(1)返回 ***x*** 的立方。
XEQ	**[XEQ]*label***	STO	(0)插入一个对标签指定子程序的调用。
		¬STO，¬INPUT	(0)运行标签指定的例程。

[66]这些数字和中国的第 1 个到第 6 个工作日直接对应。对于葡萄牙工作日（“segunda feira”等），第 1 天到 5 天要加 1。

名称	按键	模式	备注（通用说明见 77 页）
	[A]，[B]，[C]或**[D]**（整数模式进制大于 10 时，需要[f]来访问这几个热键）	STO	(0)插入对相应标签子程序的调用，例如，当**[C]**被按下时会插入 XEQ C。
		¬STO，¬INPUT	(0)如果已定义，运行对应的程序。
XEQα	[h][P.FCN]XEQα	¬INPUT	(0)将 alpha 的前三个字符（不足三个则为所有字符）作为标签，调用或运行相应程序。
XNOR	[h][X.FCN]XNOR	¬INPUT	(2)和 AND 属于一类。参见 57 页。
XOR	[h][XOR]	¬INPUT	(2)和 AND 属于一类。参见 57 页。
XTAL?	[h][TEST]XTAL?	¬INPUT	(0)判断晶振是否已安装并激活。精确的实时时钟需要安装晶振（Crystal 简写为 Xtal 的原因和 Xmas 一样）。
$\bar{x}$	[f][$\bar{x}$]	DECM	(-2)返回累积的 x 数据和 y 数据的算术平均值。参见 s，SERR 和 σ 。
$\bar{x}_g$	[h][STAT]$\bar{x}_g$	DECM	(-2)返回累积的 x 数据和 y 数据的几何平均值。公式见附录 I。参见 ε，ε_m 和 ε_p。
$\bar{x}_w$	[h][STAT]$\bar{x}_w$	DECM	(-1)返回累积数据的加权算术平均值（每个 ***x*** 的权重 ***y*** 通过**[Σ+]**来输入）。公式见附录 I。参见 s_w 和 $SERR_w$。
$\hat{x}$	[h][STAT]$\hat{x}$	DECM	(1)对于给定的 ***y***（保存在 **X** 中），返回所选择拟合模型的预估值 ***x***。参见 L.R.。
	[h][X.FCN]	All	菜单。见 117 页。
x!	[h][!]	DECM	(1)返回 $\Gamma(x+1)$。
		Integer	(1)返回阶乘 $n!$（n>0）。
x→α	[h][X.FCN]x→α	All	(0)将 ***x*** 解释为字符码。将对应的字符添加 ***alpha*** 中，与 *HP-42S* 中的 XTOA 类似。 注意 x→α 使用内部字符码，和 Unicode 不同。使用 α→x 进行检查。另见附录 E。
***x*⇆**	[h][x⇆]***s***	¬INPUT	交换 ***x*** 和 ***s***，和 ***x*⇆*y*** 类似。^{C}x⇆见 33 页。可能出现的形式有 x⇆ J，x⇆.12，x⇆→12 等。
x⇆Y	**[x⇆y]**	¬INPUT	交换堆栈的内容：***x*** 和 ***y***。如果之前刚刚进行了复数操作，则执行 Re⇆Im。
^{C}***x*⇆*Z***	**[CPX][x⇆y]**	¬INPUT	交换复数堆栈的内容：x_c 和 y_c。详见 31 页。
x<?	[h][TEST]x<_?***s***	¬INPUT	(0)比较 ***x*** 与 ***s*** 的大小。例如[h][TEST]x<_?**[K]**比较 ***x*** 与 ***k*** 的大小，在例程中显示为 x<? K。详见 26 页的示例。 x≈?在 ***x*** 与 ***s*** 的舍入值相等时为 true（参见 ROUND）。 带符号的判断 x=+0?和 x=-0 用于整数模式的 1COMPL（反码）和 SIGNMT（原码），如果设置了 flag D，则可用于 DECM 模式。以上模式中，0 除以-7 会显示-0。 **[CPX]**[f][x=?]***s*** 和**[CPX]**[g][x≠?]***s*** 比较复数 *x+iy*，如 33 页所示。
x≤?	[h][TEST]x≤_?***s***		
x=?	[f][x=?]***s***		
x=+0?	[h][TEST]x=+0?		
x=-0?	[h][TEST]x=-0?		
x≈?	[h][TEST]x≈_?***s***		
x≠?	[g][x≠?]***s***		
x≥?	[h][TEST]x≥_?***s***		
x>?	[h][TEST]x>_?***s***		
$\sqrt[x]{y}$	[h][X.FCN]$\sqrt[x]{y}$	¬INPUT	(2)返回 ***y*** 开 ***x*** 次方。

名称	按键	模式	备注（通用说明见 77 页）
YDOFF	[h][MODE]YDOFF		切换点阵中 *y* 的显示。
YDON	[h][MODE]YDON		
YEAR	[h][X.FCN]YEAR	DECM	(1)将 *x* 视为选定格式的日期并返回对应年份。
y^x	[f][y^x]	¬INPUT	(2)返回 *y* 的 *x* 次幂。整数模式下，*x* 必须≥0。
y^x	[C]	¬INPUT	(2)若标签 C 未在当前程序中定义，则运行此功能。
ŷ	[f][ŷ]	DECM	(1)对于给定的 ***x***，返回所选择拟合模型的预估值 ***y***（保存在 **X** 中）。更多信息参见 L.R.。
Y.MD	[h][MODE]Y.MD	All	(0)设置日期的显示格式为 yyyy.mmdd。
y⇆	[h][P.FCN]y⇆*s*	¬INPUT	交换 ***y*** 与 ***s***，与 ***x*⇆**类似。
z⇆	[h][P.FCN]z⇆*s*	¬INPUT	交换 ***z*** 与 ***s***，与 ***x*⇆**类似。
α	[f][α]	STO 和 ¬INPUT	打开 alpha 模式，用于从键盘输入 alpha 常量。**INPUT** 指示符会点亮，程序的前一步将会保持显示直到有字符输入。每一个输入的字符（比如?）都会被存储在一步程序中（比如 α?），并且在程序执行时添加到 ***alpha*** 中。要新建一个字符串，请先用 CLα。
		STO 和 INPUT	打开 alpha 组模式[67]，在一个程序步中直接输入最多三个字符，占用两个字（*4byte*）的空间。*WP 34S* 会在最上方显示 α'。这时可以输入想要追加到 ***alpha*** 中的字符。如果要新建一个字符串，请先用 CLα。 示例：输入 [f][α][T][f][⇧][E][S] [f][α][T][h][PSE][1] 会生成两步程序： α'Tes' α't 1' Test 1会在程序执行时追加到***alpha***中。
		¬STO，¬INPUT	进入 alpha 模式，用于将字符串添加到 ***alpha*** 中。要新建一个字符串，请先用 CLα。
		¬STO 和 INPUT	离开 alpha 模式。
αDATE	[h][X.FCN]αDATE	¬integer	(0)将 ***x*** 视为当前设置格式的日期，将其添加到 ***alpha*** 中。参见 DATE。 想要在 ***alpha*** 中加入日期戳，可调用 DATE αDATE。对于欧洲短日期戳，设置 FIX 2，RDX. 后调用 DATE αRC#X。
αDAY	[h][X.FCN]αDAY	¬integer	(0)将 ***x*** 的对应日期的星期的前三个字母添加到 ***alpha*** 中。

[67]Alpha 组模式在放入三个字符后自动退出，因此如果还需要继续该模式，必须再次调用。某些字符不可在第三个位置输入（参见附录 E）。

名称	按键	模式	备注（通用说明见 77 页）
αGTO	[h][P.FCN]αGTO*s*	¬INPUT	(0)将 s 解释为字符码。将前三个字符（如果少于三个，则是全部字符）作为 alpha 标签，并将程序指针定位到它。
αIP	[h][X.FCN]αIP*s*	All	(0)将 ***s*** 的整数部分添加到 ***alpha*** 中，类似 *HP-42S* 中的 AIP。 **注意：**整数部分是逐位加入的，所以大于 10^{30} 的数字的最高有效位可能会丢失。进制的指示符不添加。
αLENG	[h][X.FCN]αLENG	All	(-1)返回 ***alpha*** 中的字符个数，类似 *HP-42S* 中的 ALENG。
αMONTH	[h][X.FCN]αMONTH	¬integer	(0)将 ***x*** 视为一个日期，将其对应月份名称的前三个字母添加到 ***alpha*** 中。
αOFF	[h][P.FCN]αOFF	¬INPUT	(0)关闭/打开 alpha 模式。类似 *HP-42S* 中的 AOFF 和 AON。
αON	[h][P.FCN]αON		
αRCL	[f][RCL]*s*	INPUT	(0)将 ***s*** 作为字符，添加到 ***alpha*** 中。
	[h][X.FCN]αRCL*s*	¬INPUT	
αRC#	[h][X.FCN]αRC#*s*	All	(0)将 ***s*** 作为一个数字，将其转换为当前设置格式的字符串，并将其添加到 ***alpha*** 中。 示例：如果 ***s*** 为 1234，并且设置了 ENG 2 和 RDX.模式，那么 1.23e3 会被追加到 ***alpha*** 中。
αRL	[h][X.FCN]αRL*n*	All	(0)将 ***alpha*** 向左循环移动 ***n*** 个字符，如同 *HP-42S* 中的 AROT，但是大于零的参数 ***n*** 要尾随这个指令，而不是从 **X** 中获取。aRL 0 为空指令，但是会改变 L。
αRR	[h][X.FCN]αRR*n*	All	(0)和 αRL 类似，但是向右循环移动。
αSL	[h][X.FCN]αSL*n*	All	(0)如 HP-42S 中的 ASHF 一样，将最左端 ***n*** 个字符移出 ***alpha***。αSL 0 为空指令，但是会改变 L。
αSR	[h][X.FCN]αSR*n*	All	(0)和 αSL 类似，但是从最右端开始移出。
αSTO	[f][STO]*s*	INPUT	(0)在指定的地址中存储 ***alpha*** 中最左端的 6 个字符，若少于 6 个，则存储全部字符。
	[h][X.FCN]αSTO*s*	¬INPUT	
αTIME	[h][X.FCN]αTIME	¬integer	(0)将 ***x*** 作为时间并且根据当前选择的时间模式以 `hh:mm:ss` 的形式加入至 ***alpha*** 中。参见 12h，24h 和 TIME。 想要在 ***alpha*** 中加入时间戳，请调用 TIME αTIME。
αXEQ	[h][P.FCN]αXEQ*s*	¬INPUT	(0)将 ***s*** 作为字符码，将其对应字符的前三个（少于 3 个则为全部）作为 alpha 标签，并调用或运行对应的例程。
α→x	[h][X.FCN]α→x	All	(-1)返回 ***alpha*** 中最左端字符的内部字符码，并将其移出 ***alpha***，如 *HP-42S* 中的 ATOX。注意和 x→α 的不同。
β	[h][X.FCN]β	DECM	(2)返回欧拉 Beta 函数。详见附录 I。

名称	按键	模式	备注（通用说明见 77 页）
Γ	[h][X.FCN]Γ	DECM	(1)返回 $\Gamma(x)$。另外，[h][!]调用 $\Gamma(x+1)$。参见 LNΓ。
Γ_p	[h][X.FCN]Γ_p	DECM	返回两种正则化伽玛函数的一种。详见附录 I。
Γ_q	[h][X.FCN]Γ_q		
γ_{XY}	[h][X.FCN]γ_{XY}	DECM	(2)分别返回下不完全伽玛函数与上不完全伽玛函数。详见附录 I。
Γ_{XY}	[h][X.FCN]Γ_{XY}		
ΔDAYS	[h][X.FCN]ΔDAYS	DECM	(2)将 **X** 与 **Y** 视为包含当前格式的日期，并如 *HP-12C* 一样计算它们相隔的天数。
Δ%	[g][Δ%]	DECM	(1)返回 $100\cdot\frac{x-y}{y}$，和 HP-42S 的%CH 一样。
ε	[h][STAT]ε	DECM	(-2)返回对数正态分布的样本数据的分布系数 ε_y 和 ε_x。ε_x 基于几何平均值 $\bar{x}_g$ 进行计算，详见附录 I。
ε_m	[h][STAT]ε_m	DECM	(-2)和 ε 类似，但是计算的是两个几何平均值的分布系数。
ε_p	[h][STAT]ε_p	DECM	(-2)和 ε 类似，但是计算的是两个总体的分布系数。
ζ	[h][X.FCN]ζ	DECM	(1)返回黎曼 Zeta 函数。详见附录 I。
π	[h][π]	DECM	(-1)返回 π。
$^{C}\pi$	[CPX][π]	DECM	(-2)在 **X** 中返回 π 并清空 **Y**。
Π	[h][Π]***label***	DECM	(1)使用指定的程序计算求积。**X** 包含如格式 `ccccc.fffii` 的循环控制数，并将求积初值设置为 1。每次运行一次标签指定的例程都会用其计算一次乘数，将其与之前的乘积相乘；然后将操作数 `ccccc` 减少 `ii`，如果 `ccccc≥fff` 则再运行一遍刚才的例程，否则在 **X** 中返回最终的求积结果。 如果 `ii=0`，`ccccc` 会按 1 递减。 **注意：**在调用指定的例程之前，Π 会用 ***x*** 填充所有级别的堆栈。
σ	[h][STAT]σ	DECM	(-2)类似 s，但返回两个总体的标准差。公式见附录 I。

名称	按键	模式	备注（通用说明见 77 页）
Σ	**[g][Σ]*label***	DECM	(1)使用指定的程序计算求和。**X** 包含如格式 `cccccc.fffii` 的循环控制数，并将求和初值设置为 0。每次运行一次标签指定的程序都会用其计算一次加数。算完这个加数后将其与之前的总和相加；然后将操作数 `cccccc` 减少 `ii`，如果 `cccccc≥fff` 则再运行一遍刚才的程序，否则在 **X** 中返回最终的求和结果。 如果 `ii=0`，`cccccc` 会按 1 递减。 示例： 计算 $\sum_{k=1}^{100}\sqrt{k}$： 1. 写一个小例程 `LBL'S'`，`√`，`RTN`。 2. 输入 100.001**[g][Σ][S]**得到 `671.4629`（输入 100**[g][Σ][S]**可以得到相同的结果）。 **注意：**在调用指定的例程之前，Σ 会用 ***x*** 填充所有级别的堆栈。
$\Sigma\ln^2 x$ $\Sigma\ln^2 y$ $\Sigma\ln x$ $\Sigma\ln xy$ $\Sigma\ln y$	**[h][SUMS]**`Σln²x` 等.	DECM	(-1)计算指定的统计求和。对于线性拟合之外的模型，这些求和是必需的。按名称调用它们可显著提高程序的可读性。请注意，这些统计和存储在 *WP 34S* 的特殊寄存器中。 **注意：**根据输入，全部或部分统计和可能为非数值结果。无论标志位 D 是否置位都不会抛出错误。
σ_w	**[h][STAT]**`σw`	DECM	(-1)和 s_w 类似，但是返回总体的标准差。公式见附录 I。
Σx Σx^2 $\Sigma x^2 y$ $\Sigma x\ln y$ Σxy Σy Σy^2 $\Sigma y\ln x$	**[h][SUMS]**`Σx` 等.	DECM	(-1)计算指定的统计和。对于基本的统计和拟合模型，这些求和是必需的。按名称调用它们可显著提高程序的可读性。请注意，这些统计和存储在 *WP 34S* 的特殊寄存器中。 **注意：**根据输入，全部或部分统计和可能为非数值结果。无论标志位 D 是否置位都不会抛出错误。
Σ+	**[h][Σ+]**	DECM	将一个数据点添加到统计和中。	这两个函数返回 **X** 中的新数据点，禁用自动堆栈提升。可用于二维向量的加减运算（参见 SUM）。
	[A]	DECM	若标签 A 未定义，则运行此功能。	
Σ-	**[h][Σ-]**	DECM	从统计和中减去一个数据点。	
$\Phi_u(x)$	**[h][PROB]**`Φu(x)`	DECM	(1)标准正态误差概率。等于 *HP-32E* 中的 Q 和 *HP-21S* 中的 Q(z)。
$\phi(x)$	**[h][PROB]**`Φ(x)`	DECM	(1)标准正态 *pdf*。公式见附录 I。
Φ(x)	**[f][Φ]**	DECM	(1)标准正态 *cdf*。公式见附录 I。
Φ^{-1}(p)	**[g][Φ^{-1}]**	DECM	(1)标准正态分位数函数。

名称	按键	模式	备注（通用说明见 77 页）
χ^2	[h][PROB]x^2	DECM	(1)卡方分布（自由度在 J 中给出）。在 *HP-21* 中，χ^2_u 等于 $Q(\chi^2)$。详见附录 I。
χ^2INV	[h][PROB]x^2INV		
χ^2_P	[h][PROB]x^2_P		
χ^2_u	[h][PROB]x^2_u		
$(-1)^x$	[h][X.FCN](−1)^x	¬INPUT	(1)如果 x 不是一个整数，那么则返回 $\cos(\pi \cdot x)$。
+	[+]	¬INPUT	(2)返回 $y+x$。
	[CPX][+]	DECM	(2)返回[$x+z$, $y+t$,...]。也可用于二维向量加法。
-	[-]	¬INPUT	(2)返回 $y-x$。
	[CPX][-]	DECM	(2)返回[$x-z$, $y-t$,...]。也可用于二维向量减法。
×	[×]	¬INPUT	(2)返回 $y \cdot x$。
	[CPX][×]	DECM	(2)返回[$x \cdot z-y \cdot t$, $x \cdot t+z \cdot y$,...]。二维向量乘法见 DOT 或 CROSS。
×MOD	[h][X.FCN]×MOD	Integer	(3)返回$(z \cdot y)$ mod x（其中 x>1，y>0，z>0）。
/	[/]	DECM	(2) 返回 y/x。
		Integer	(2) 返回 y div x。注意和 IDIV 的区别。
	[CPX][/]	DECM	(2) 返回[$\frac{x \cdot z+y \cdot t}{z^2+t^2}, \frac{z \cdot y-x \cdot t}{z^2+t^2}$,...]。
+/-	[+/-]	¬INPUT	(1) 取相反数。
→DATE	[h][X.FCN]→DATE	DECM	(3) 假设在堆栈上按顺序存有所选日期格式的三个日期数（年、月、日），并将它们转换为单个日期保存在 x 中。
→DEG	[⇨][DEG]	DECM	(1) 将 x 以当前角度格式转换为角度制。
→GRAD	[⇨][GRAD]	DECM	(1) 类似于→DEG，但是转为百分制。
→H	[⇨][f][H.d]	DECM	(1) 将 x 视为一个 hhhh.mmssdd 格式的时间或角度，转化十进制的时间或角度，以用于算术运算。
→H.MS	[⇨][f][H.MS]	DECM	(1) 将 x 视为一个十进制的时间或角度，转化为 hhhh.mmssdd 格式的时间或角度。要继续进行运算，请使用 H.MS+或 H.MS−。
→POL	[g][⇨P]	DECM	将 X 和 Y 表示的直角坐标或向量分量（x，y）转换为极坐标或分量（r，ϑ），其中半径 r= $\sqrt{x^2+y^2}$。
→RAD	[⇨][RAD]	DECM	(1) 类似于→DEG，但是转为弧度制。
→REC	[f][R⇦]	DECM	将 X 和 Y 表示的极坐标或向量分量（r，ϑ）转换为直角坐标或分量（x，y）。

名称	按键	模式	备注（通用说明见 77 页）
⇆	[h][P.FCN]⇆____	¬INPUT	改变最下方四个堆栈的顺序。 示例： ⇆XXYZ 类似 ENTER↑（但不禁止自动堆栈提升）， ⇆YXZT 类似 x⇆y ⇆YZTX 类似 R↓， ⇆TXYZ 类似 R↑， ⇆ZTXY 类似 ᶜx⇆y， ⇆ZZZX 也是可以的。 **注意：**尽管输入的是字母但是改变的是堆栈内容的顺序。 尽管看起来不太像，但这是一个非常强大的命令。它仅影响最下 4 个级别的堆栈，即便在 8 堆栈模式下。如果您随意使用此命令，则可能会丢失数据并弄乱堆栈。
%	[f][%]	DECM	(1) 返回 $\frac{x\cdot y}{100}$，保持 **Y** 不变。
%MG	[h][X.FCN]%MG	DECM	(2) 以百分比形式返回售价为 x，成本为 y 的销售利润率 $100\cdot\frac{x-y}{x}$。和 *HP-17B* 的%MU-Price 类似。
%MRR	[h][X.FCN]%MRR	DECM	(2) 以百分比形式返回单位周期的平均收益率，即 $100\cdot[(x/y)^{1/z}-1]$，y 为现在的价格，x 为 z 个周期后的价格。 z=1 时，Δ%可以更方便地返回相同结果。
%T	[h][X.FCN]%T	DECM	(2) 返回 $100\cdot\frac{x}{y}$，即总数的百分比。
%Σ	[h][X.FCN]%Σ [h][STAT]%Σ	DECM	(1) 返回 $100\cdot\frac{x}{\sum x}$。
%+MG	[h][X.FCN]%+MG	DECM	(2) 计算销售利润率为 x%，进价为 y 的商品的售价。 公式：$p_{sale}=\frac{y}{1-\frac{x}{100}}$ 也可以使用%+MG 来计算净值。只需在 x 输入一个负百分比。 示例：总费用=221.82€，增值税=19%。净值是多少？ 221.82**[ENTER⇧]**19**[+/−]**[X.FCN]**[▲]**，答案是 186.40。
√	[f][$\sqrt{x}$] **[D]**	¬INPUT	(1) 返回 x 的平方根。 (2) 如果当前程序未定义标签 D，则这个快捷按键有效。

名称	按键	模式	备注（通用说明见 77 页）
∫	[g][∫]***label***	DECM	对指定例程中的函数求定积分，下界在 **Y** 中，上界在 **X** 中。∫在 **X** 中返回积分值，在 **Y** 中返回误差上限。除此之外用户界面和 *HP-15C* 类似。 可以参阅 *HP-15C Owner's Handbook*（14 节和附录 E）获取关于自动积分算法的更多信息以及一些使用注意事项。 **注意：**在调用指定的例程之前，∫会用 x 填充所有级别的堆栈。
∞?	[h][TEST]∞?	¬INPUT	(0) 判断 x 是否无穷大。
^MOD	[h][X.FCN]^MOD	Integer	(3) 返回 (z^y) mod x，其中 x>1，y>0，z>0。 示例： [10]73**[ENTER⇧]**55**[ENTER⇧]**31[X.FCN]**[▲][XEQ]** 返回 26。
||	[g][||]	DECM	(2) 返回 $(\frac{1}{x}+\frac{1}{y})^{-1}$，在电气工程中特别有用[68]。 若 x 或 y 为 0 则返回 0。
🖶ADV	[h][P.FCN]🖶ADV	¬INPUT	(0) 向当前打印缓冲区的内容添加换行符，并打印它。 **注意：**任何打印操作只有在做了相应的硬件修改（见附录 H）后，或者使用模拟器和模拟打印机（见附录 D）才能运行，否则会忽略打印指令。打印机在收到换行指令后才会开始打印。
🖶CHR	[h][P.FCN]🖶CHR***n***	¬INPUT	(0) 向打印机发送单个字符（指定字符码）。字符码 n>127 必须被间接指定。🖶MODE 会生效。参见🖶ADV。
🖶$^{C}r_{XY}$	[h][P.FCN]🖶ᶜrₓᵧS	¬INPUT	(0) 打印指定的寄存器和下一个寄存器，即打印整个复数。分号将把两个部分分开。其余和🖶r 类似。参见🖶ADV。 示例：模式为🖶MODE 1，SCI 1，x=-1.2，并且 y=0. 34。🖶ᶜrₓᵧ X 会如下输出： −1.2e0 ; 3.4e−1。
🖶DLAY	[h][MODE]🖶DLAY***n***	All	(0) 对打印机的每一次换行，设置 ***n*** 个 tick 的延迟（参见 TICKS）。可以用于减轻打印机的负荷。

[68]译者注：可以快速计算并联阻抗。

名称	按键	模式	备注（通用说明见 77 页）
🖷MODE	[h][MODE]🖷MODE*n*	All	(0)设置打印模式，模式有： 0：尽可能使用打印机的字体和字符集（默认的），所有的字符宽度相同（占 5 列，间隔 2 列） 1：使用可变间距字体，虽然会不太整齐，但一行上能够容纳更多字符 2：使用小号打印字体，一行上能还能容纳更多的字符。 3：将输出发送到串行通道，只对纯 ASCII 字符有效——不会翻译任何字符。行设置和串行通道相同：9600 波特，8 位，无奇偶校验。
🖷PLOT	[h][P.FCN]🖷PLOT*s*	¬INPUT	(−1)发送从地址 ***s*** 开始的图形块到打印机，若宽度为 166，则前面会跟着一个换行符。 参见🖷ADV 和 gDIM。
🖷PROG	[h][P.FCN]🖷PROG	¬INPUT	(0)打印当前程序，每一步一行。当前程序是程序指针在执行时所在的程序。参见🖷ADV。
🖷r	[h][P.FCN]🖷r*s*	¬INPUT	(0)打印 ***s***，靠右对齐，不打印标签。若你想要打印标签，先调用🖷α+或者使用🖷REGS。参见🖷ADV。
🖷REGS	[h][P.FCN]🖷REGS	¬INPUT	(1)将 ***x*** 解释成 sss.nn。打印以数字 ***sss*** 开头的 ***nn*** 个寄存器的内容。每个寄存器占用一行，以一个标签开头。参见🖷ADV。 **注意：*nn*=0 时：** 若 ***sss***∈[0；99]，打印将在分配的最高全局寄存器处停止。 若 ***sss***∈[100；111]，打印将在 **K** 停止 若 ***sss***≥112，打印将在分配的最高本地寄存器处停止。
🖷STK	[h][P.FCN]🖷STK	¬INPUT	(0)打印堆栈内容。每一级堆栈占用一行，以一个标签开头。参见🖷ADV。
🖷TAB	[h][P.FCN]🖷TAB*n*	¬INPUT	(0)将打印头定位到 ***n***（0 到 165，其中 ***n***>127 的列只能间接指定）列。对于格式化很有用（尤其在🖷MODE 1 和🖷MODE 2）。也可用于打印图形。如果 ***n*** 小于当前位置的列，则将换行以到达新位置。参见🖷ADV。
🖷WIDTH	[h][P.FCN]🖷WIDTH	¬INPUT	(−1)返回 ***alpha*** 在当前模式打印需要的列数。参见🖷ADV 和🖷MODE。 第二种用法：在🖷MODE 1 或🖷MODE 2 返回 ***alpha*** 在指定字体下的像素宽度（包括一直空着的最后一列）
🖷α	[h][P.FCN]🖷α	¬INPUT	(0)将 ***alpha*** 加入到打印行加上换行符。注意和🖷α+，🖷+α 的不同。参见🖷ADV。
🖷α+	[h][P.FCN]🖷α+	¬INPUT	(0)将 ***alpha*** 加换行符发送到打印机，便于在后面加入更多的信息。可以重复使用。参见🖷ADV，🖷r 和🖷+α。

名称	按键	模式	备注（通用说明见 77 页）
🖷Σ	[h][P.FCN]🖷Σ	¬INPUT	(0) 打印求和寄存器。每个寄存器占用一行，以一个标签开头。参见🖷ADV。
🖷+α	[h][P.FCN]🖷+α	¬INPUT	(0) 将 ***alpha*** 右对齐添加到打印行，后带一个换行符，注意和🖷α，🖷α+的区别。参见🖷ADV。 **示例：**以下的程序 CLα α'Lef' α t 🖷α+ CLα α'Rig' α'ht' 🖷+α 将会打印（如果设置了🖷MODE 1）： Left Right
🖷?	[h][TEST]🖷?	¬INPUT	(0) 判断是否安装了打印必需的晶振和固件。
🖷#	[h][P.FCN]🖷#*n*	¬INPUT	(0) 发送单个字节（不转义）到打印机（例如一个 CTRL）。*n*>127 只能被间接指定。🖷MODE 设置会被忽视。参见🖷ADV。
#	[h][CONST]#*n*	STO	在一步程序中插入一个数字 *n*，0≤*n*≤255，这样最多可以节省两个程序步和一个 ENTER。
^{C}#	[CPX][h][CONST]#*n* [CPX]*n*	DECM	类似#但是同时清空 *y*。快捷方式只对 1≤*n*≤9 的 *n* 有效。

不可用于编程的控制、清除和消息按键

除了 CLALL、CLPALL、LOAD、LOADP、PSTO、RESET 和 SAVE 指令外，所有的菜单、浏览器、程序调用和以下按键无法在所述条件下用于编程：

按键	模式	备注
[f][B]	CONV	求倒数，用于反向换算（见 131 页）。
[8]	请求确认	回答问题 Sure? 为 N，代表“否”。此外，除了[R/S]代表 Y（是），[EXIT]和[⇦]代表“否”外，任何其他回应都会被忽略。
	在浏览器中	将在相应的章节中进行描述。
[ENTER⇧][69]	CAT，SHOW，STOPW	将在相应的章节中进行描述。
	÷	与[XEQ]类似，选择当前项。
	INPUT	退出 alpha 模式。
	其他	调用可用于编程的 ENTER 指令。
[EXIT]	¬RPN	清空显示的临时消息（见 38 页），返回到输出消息之前的状态。
	请求确认	回答问题 Sure? 为 N，代表“否”。此外，除了[R/S]代表“是”，[8]和[⇦]代表“否”外，其他任何回应都会被忽略。
	STOPW	离开秒表程序（见 137 页）。
	在菜单或浏览器内	离开当前目录或浏览器而不执行任何操作，返回到 *WP 34S* 之前的状态。
	命令输入待处理	取消挂起的操作，返回到 *WP 34S* 之前的状态。
	RCL	像[R/S]一样暂停程序。
	STO	像[P/R]一样离开编程模式。
	INPUT	像[ENTER⇧]一样离开 alpha 模式。
	其他	什么都不做。
[h][GTO][.][A]，[B]，[C]或[D]	¬INPUT	定位程序指针到这些标签，如果定义了的话。
[h][GTO][.]*n*	¬INPUT	定位程序指针到第 ***n*** 步[70]。
[h][GTO][.][▲]	¬INPUT	定位程序指针直接到前一个程序的 END 后，即当前程序的顶部。
[h][GTO][.][▼]	¬INPUT	定位程序指针直接到下一个 END 后，即下一个程序的顶部。

[69]执行此指令时将自上向下进行当前模式检查：
if 打开了 CAT，STOPW 或 STOPW，[ENTER⇧]将按该处所描述的方式工作；
else if 进入了菜单，则选择当前项；
else if 如果设置了 alpha 模式，将会退出；
else 可编程的 ENTER 命令将被执行或插入。
此方法适用于表中有三角形符号的所有指令。

[70]GTO 会在识别输入的地址后立即执行。例如程序空间共有 123 步的程序，GTO.45 会立即跳到第 45 步，不需要输入第三个数。GTO.13 会跳到第 13 步，GTO.129 会跳到最后一步（最后的 END 语句）。

按键	模式	备注
[h][GTO][.][.]	¬INPUT	定位程序指针到程序内存的最后一步，即最后一个 END 语句。
[ON]	关机状态下	打开 *WP 34S*.。
	其他	**[ON]**和其它键的使用组合见附录。最重要的是用**[ON]**+**[+]**或**[ON]**+**[-]**来调整屏幕亮度。
[h][P/R]	¬INPUT	切换编程模式（STO）。
[R/S]	请求确认	回答问题 Sure? 为 Y，代表“是”。此外，除了**[EXIT]**、**[8]**或**[⇦]**代表“否”外，其他任何回应都会被忽略。
	CAT/STOPW	将在相应的章节中进行描述。
	RCL	立刻暂停程序。将会显示 Stopped 直到按下下一个键。
	¬(STO, INPUT)	运行当前程序或从当前步开始恢复其执行。
	INPUT	将 Y 或 Z 加入到 ***alpha***.。
	STO	输入可用于编程的 STOP 指令。
[XEQ]	CAT	将在相应的章节中进行描述。
	÷	选择当前显示的项并退出，执行相应的命令。
	其他	输入可用于编程的 XEQ 指令。
[XEQ][.]	¬INPUT	调用**[GTO][.]**。
[⇦]	¬RPN	清空临时显示内容（见 38 页）返回输出消息前的状态。
	请求确认	回答问题 Sure? 为 N，代表“否”。此外，除了**[R/S]**代表“是”，**[EXIT]**和**[8]**代表“否”外，其他任何回应都会被忽略。
	STOPW	重置计时器（见 137 页）。.
	在菜单或浏览器内	像**[EXIT]**一样退出目录
	命令输入待处理	删除最后一个输入的数字或字符。如果不存在，像**[EXIT]**一样清除被挂起的操作。
	INPUT	删除 ***alpha*** 中最右端的字符。
	STO	删除当前的程序步。
	其他	调用可编程 CLx 指令。
[f][◀()/[g][()▶]	Integer	像 *HP-16C* 那样将显示窗口移动到最左端或最右端。对于较小的进制很有用。参见 54 页。
	DECM	显示完整的有效数字和完整的指数，直到按下下一个键[71]（见 41 页）。注意和之前计算器中的 SHOW 的区别。
[⇨][f][2] **[⇨][8]** **[⇨][f][10]** **[⇨][16]**	DECM	将***x***以2进制、8进制、10进制和16进制整数表示，按下任意键将返回现在的进制模式。前缀**[g]**在这里可以忽略。

[71] 这些按键在 DP 模式中功能不同，参见后面的附录 H。

按键	模式	备注
[f][⇧]	≒	调用字符🖷。
	INPUT	切换大小写（后者由⬇指示）。
	其他	调用可编程命令🖷rX。
[▲]/[▼]	在浏览器或 STOPW	将会在对应章节给出解释。
	≒	转到当前菜单中的上一项/下一项。
	INPUT	如果可能的话，将显示窗口向左/向右滚动 6 个字符。如果少于 6 个字符，则窗口将定位为字符串最左端/最右端，这对于长字符串很有用。
	其他	行为等同于 *HP-42S* 中的 BST/SST。也就是说，按住[▲]/[▼]不放超过 0.5 秒，则将以 5*Hz* 的频率自动重复，用来在编程模式中浏览程序。若不在编程模式中，SST 也将执行当前的程序步骤，但按键不会自动重复。

关于菜单、浏览器和秒表的更多信息，参见后续章节。

字母数字输入

字符	按键	模式	备注
␣	[h][PSE]	INPUT	向 ***alpha*** 中加入一个空格。
°	[,]	DECM	将度或时与分、秒和百分秒分割开，所以输入格式应该为 hhhh.mmssdd。
0...9	[0]...[9]	¬INPUT	标准数字输入。对于小于 10 的整数进制，将阻止非法输入。注意，输入的有效数字不能超过 12 位。
		寻址	输入寄存器号。可输入数字的范围见 26 页。
0...9	[0], [1], [f][2], ..., [f][9]	INPUT	将各位数字添加到 ***alpha*** 中。
A...F	[A]...[F]（灰字）	Integer	输入大于 10 的数字。见 53 页。
A...Z	[A]...[Z]（灰字）	寻址	输入寄存器号。可输入的字符见 25 页。
		INPUT	将对应的拉丁字母加入 ***alpha***。使用[f][⇧]切换大小写。
a...z	[f][CPX]...	INPUT	将对应的字母加入 ***alpha***。使用[f][⇧]切换大小写。
E	[EEX]	DECM&¬FRC	和 Pioneer 系列计算器中的[E]一样。
i	[CPX][.]	DECM&¬FRC	输入复数 ***i***，即 *x*=0，*y*=1 并以 ALL0 格式显示  。
Α...Ω	[g][A]...[g][O]（灰字）	INPUT	将对应的希腊字添加到 ***alpha*** 中。使用[f][⇧]切换大小写。详见 62 页。
(	[f][◀(]	INPUT	将对应的符号添加到 ***alpha*** 中。
)	[g][)▶]		
+	[f][+]		
−	[f][-]		
×	[f][×]		
/	两次[.]	DECM	第二次输入表示分数，其中第一次[.]表示空格，第二次表示分数线。注意不可以在两次按下此键后使用 EEX，但是可以在编辑输入行时删除第二个点号。
		FRC	第一个[.]被当作一个空格，第二个被当作分数线。详见 41 页。
/	[f][/]	INPUT	向 ***alpha*** 中添加一个斜杠。
+/-	[+/-]	¬INPUT	和 Pioneer 系列计算器中的±一样。参见 12 页。
±	[f][+/-]	INPUT	将对应的字符加入到 ***alpha*** 中。
,	[h][./,][XEQ]		
.	[f][.]		
'.'或','	[.]	DECM	根据所选择的样式，插入一个小数点。.
!	[h][!]	INPUT	将对应的字符添加到 ***alpha*** 中。

字符	按键	模式	备注
?	[h][▼]		
⇆	[h][x⇆]		
≠	[h][XOR]		
&	[h][AND]		
\	[h][/]		
|	[h][OR]		
¬	[h][NOT]		
🖷	[f][⇧]	菜单中	输入打印字符以快速访问打印指令（参见 125 页）。

6. 菜单和浏览器

菜单浏览

鉴于 *WP 34S* 具有大量的功能，它们的大部分都放在菜单中，如 23 页所述，

在打开菜单之后，*WP 34S* 将被设置为 alpha 模式，这时可以通过输入字符（串）来快速访问项目。如下图所示，63 页所介绍的全键盘的子集就能够满足菜单浏览的需要：

但是有三个例外：

在 CONV 中，[f][B]（=[1/x]），这是为了更容易进行单位间的反换算（见 131 页），

[f][⇨]仅调出字符➔，

[f][EXIT]调出打印机符号🖷（因为这里不需大小写切换）。

另外：

[▲]和[▼]可以对打开的菜单进行浏览。

按[ENTER⇧]或[XEQ]来选择当前选中的项目，将其调出或执行，并退出菜单界面。

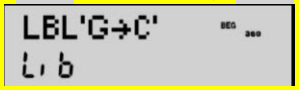
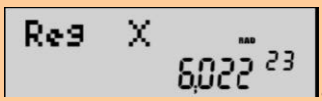
按[EXIT]或[⇦]可离开菜单而不执行任何项目，即结束菜单的调用。

可参见 125 页的实例。

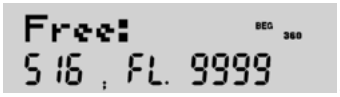
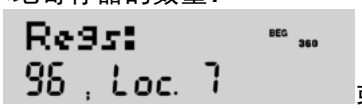
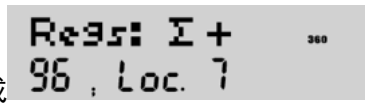
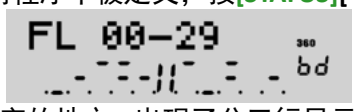
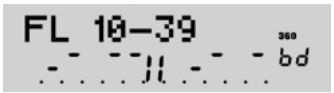
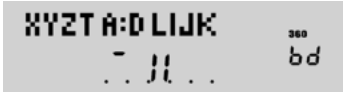
当想要访问当前模式下的其他菜单时，不需要先用**[EXIT]**键退出当前菜单——直接切换即可。

再次打开上次调用的菜单时，之前查看的最后一个项目将被立即显示，以便于重复使用。同一个函数可能在多个菜单中。

还有三个浏览器用来专门检查内存、标志位、程序标签和寄存器（即**[CAT]**、**[SHOW]**和**[STATUS]**）。它们可以在除了 alpha 模式之外的全部模式中被调用。**[EXIT]**和**[⇦]**的功能和在菜单中一样。**[SHOW]**和**[STATUS]**在 α_T 模式也可以使用。在浏览器中可能会有一些特殊按键和特殊规则，将在后文中描述。

浏览器	内容和特殊备注
[h][CAT]	显示已定义的字母数字（即全局）标签。第一个被显示的项目是当前程序[72]的最靠前的全局标签——如果没有这样的标签，则会显示该程序的结尾，如：  。 **[▲]**和**[▼]**浏览全局标签和 END 语句，而所显示标签的位置在下一行中指示（RAM 表示 RAM，Lib 表示 *FM* 库，buP 表示备份区），例如：  。 若标签有重复，还会显示其主地址，比如当它再次在 *RAM* 中被找到时，会显示 CALLS 012，在其他地方再次被找到时，会显示 CALLS Lib。 **[f][▲]**和**[f][▼]**浏览程序，即显示上一个或下一个程序的标签（程序由 END 语句分开）。 **[0]**、**[1]**和**[2]**可以分别快速跳跃到 RAM、Lib 和 buP。 按**[ENTER⇧]**以跳转到所显示的标签，按下**[XEQ]**将执行它。这两个按键都会启动一次对标签的搜索，就像 68 页中所描述的那样。在编程模式下，按**[ENTER⇧]**键将插入一个带有这个标签作为参数的 GTO 语句，而按**[XEQ]**则是插入一个 XEQ 语句。 **[R/S]**键在不进行标签搜索的情况下启动当前所选程序。 **[RCL]**和**[STO]**分别对所选程序执行 PRCL 和 PSTO。后者仅在 RAM 或 buP 中调用有意义。 **[P/R]**则进入 **STO** 模式，允许浏览或编辑所选程序。后者只能在 RAM 中使用。注意，**[⇦]**将会删除一步程序，而**[EXIT]**或再次按下**[P/R]**时，将立即退出 CAT。 **[CLP]**删除所选程序，无论程序是在 RAM 还是 buP 中。
[g][SHOW]	从 **X** 开始，浏览全部已分配堆栈和通用寄存器的内容，初始的显示如下（数字可能不同）：  。

[72]如前文所述，当前程序是指程序指针所指向的程序。

浏览器	内容和特殊备注
	按**[▲]**向上浏览堆栈，先是其它以字母标识的寄存器，之后是 **R00**、**R01**，以此类推。 按**[▼]**则由最高编号的已分配寄存器开始浏览（初始默认是从 **R99** 开始）到 **R00**，之后是 **K**，**J**，以此类推。 如果本地寄存器已分配，按**[.]**会跳到从 **R.00** 开始的本地寄存器。按**[▲]**和**[▼]**分别向上、向下来浏览本地寄存器。再次按**[.]**会回到 **X**。这里的本地寄存器地址可以超过.15。 输入任意的合法字母或二位数字就可以跳转到对应的寄存器。（见 26 页） 按**[ENTER↑]**或**[RCL]**可以调出寄存器所显示的内容。在编程模式中，这样做会输入一行 RCL 语句。 （在 *WP 34S* 上，**[◀(]**和**[)▶]**键和老型号计算器的 SHOW 功能相同，详见 41 页。）
[h][STATUS]	显示内存状态并浏览全部的用户标志位，和 *HP-16C* 的 STATUS 类似。它首先显示 *RAM* 和 *FM* 的剩余容量（以字计）。例如：  。 按**[▼]**可查看求和寄存器是否已被使用，以及被分配的全局编号寄存器和本地寄存器的数量：  或  。 再按一次**[▼]**将简明显示最近三十个用户标志位。 示例： 如果标志位 2、3、5、7、11、13、14、17、19、20、26 和 **X** 被置位，并且标签 B 和 D 在当前程序中被定义，按**[STATUS][▼][▼]**会如下显示：  在通常显示有效数字的地方，出现了分三行显示的水平条。每一行水平条都显示了十个标志位的状态。当一个标志位被置位时，对应的水平条将会亮起。左半第一排的水平条状态表明标志位 0 和 1 为清除，而 2 和 3 已置位，并且 4 为清除。然后，  将前五个标志位与后五个标志位分开。右上的水平条状态表明标志位 5 已置位，6 为清除，7 为置位，8 和 9 为清除。下面的两行以同样形式显示了标志位 10 到 29 的状态。 再次按**[▼]**将起始地址增加 10；屏幕显示将如下图所示：  再按一次**[▼]**就可以看剩 20 到 49 号标志位了。以此类推，继续按就是 70 到 99、80 到 99、90 到 99。最后一次**[▼]**分四行显示最末的十二个全局标志位——注意，标志位 **X** 正如我们预料的，被显示为已置位状态：  指数部分指示了四个热键的状态——如果全部四个热键标签都在程序里被定义，那么将会显示 “  ”。 按**[▲]**则可以往回浏览。 另外，按一个数字（例如**[5]**）将显示从这个数字的十倍开始的最多三十个标志位（这里是从 50 号到 79 号标志位）。按一个类似于**[D]**的合法字符

浏览器	内容和特殊备注
	将显示前十二个标志位。

以下是菜单功能列表：

所按键位	所属模式	内容和特殊备注
[h][CONST]	DECM 和 ¬STO	和像 *HP-35S* 中的常数类似，但比其更为丰富。可以在 126 页（及之后）查阅常数列表。当浏览该菜单时，常数的值将被显示。选择一个常数将会调用它。
	Integer	调用指令#（详见 *IOP*）。
	DECM 和 STO	选择一个常数将插入一个含有被选常数名称的程序步，它带有前缀#。在程序执行到该步的时候会调用所插入常数的值。
[CPX][h] [CONST]	DECM 和 ¬STO	打开和[h][CONST]相同的常数菜单，但选择的常数将作为复数进行调用。所以类似于[*x*, *y*...]的堆栈将在输入后变为[常数, 0, *x*, *y*...]。
	DECM 和 STO	选择一个常数将插入一个含包有被选常数名称的程序步，它带有前缀'#。在程序执行到该步的时候会将所插入常数的值作为复数调用。
[h][CONV]	DECM	换算功能，具体见后文的表。当浏览这个菜单时，对 **X** 进行换算后的数值将被显示。选择被转换内容将返回此值。
[f][CPX]	INPUT	一些语言需要用到的特殊字符（详见 122 页）。可以切换大小写（见前文有关[f][⇧]的内容）。
[h][MATRIX]	DECM	矩阵操作指令库。
[h][MODE]	¬INPUT	更多模式设置。
[h][PROB]	DECM	包括标准正态分布在内的各种概率分布及其逆运算。
[h][P.FCN]	¬INPUT	更多的编程及输入输出函数。
[h][R⇧]	INPUT	上标和下标（见 132 页）。
[h][STAT]	DECM	更多的统计学函数。
[h][SUMS]	DECM	所有的求和寄存器。当浏览这个菜单时，寄存器内容将被显示。选择一个寄存器将调出它的内容。
[h][TEST]	¬INPUT	除了两个可按键直接访问之外的所有条件判断（详见 121 页）。
	INPUT	除了[(]、[)]和[≠]之外的比较符号和括号。(详见 122 页)。
[h][X.FCN]	DECM	更多实数函数。 为了简化编程，这三个菜单在 STO 模式中被合并了。
	Integer	更多整型函数。
	INPUT	更多字符函数。
[CPX][h] [X.FCN]	DECM	更多复变函数。
[h][./,]	INPUT	标点符号和文本符号。 详见 122 页。
[f][⇨]	INPUT	箭头和数学符号。

要了解相关菜单的详细内容，请看下一页。菜单中的项目以字母序排序（排序见 77 页）。可以通过输入指令的第一个字符以快速访问该指令——示例和约束条件请参阅第 125 页。

注意！菜单和浏览器的调用都不可在编程中使用。

菜单的详细内容

[MATRIX]
DET
LINEQS
MROW+×
MROW×
MROW⇆
M+×
M^{-1}
M-ALL
M-COL
M-DIAG
M-ROW
M×
M.COPY
M.IJ
M.LU
M.REG
nCOL
nROW
TRANSP

[CONST]
见 126 页

[CONV]
见 131 页

[MODE]
12h
1COMPL
24h
2COMPL
BASE
BestF
DBLOFF
DBLON
DENANY
DENFAC
DENFIX
DENMAX
DISP
D.MY
ExpF
FAST
FRACT
JG1582
JG1752
LinF
LogF
LZOFF
LZON
M.DY
PowerF
RCLM
RDX,
RDX.
REGS
RM
SEPOFF
SEPON
SETCHN
SETDAT
SETEUR
SETIND
SETJPN
SETTIM
SETUK
SETUSA
SIGNMT
SLOW
SSIZE4

[PROB]
Binom
$Binom_P$
$Binom_u$
$Binom^{-1}$
Cauch
…
Expon
…
$F_P(x)$
$F_u(x)$
F(x)
$F^{-1}(p)$
Geom
…
Lgnrm
…
Logis
…
Norml
…
Poiss
…
Poisλ
…
$t_P(x)$
$t_u(x)$
t(x)
$t^{-1}(p)$
Weibl
…
$\Phi_u(x)$
ϕ(x)
χ^2
χ^2INV
χ^2_p
χ^2_u

[P.FCN]
BACK
BASE?
CASE
CFALL
CLALL
CLPALL
CLREGS
CLSTK
CLa
CNST
DEC
DROP
DSL
DSZ
END
ERR
FF
FLASH?
f'(x)
f"(x)
gCLR
gDIM
gDIM?
gFLP
gPLOT
gSET
GTOα
INC
ISE
ISZ
KTP?
LOAD
LOADP
LOADR
LOADSS
LOADΣ
LocR
LocR?
MEM?
MSG
NOP
PopLR
PRCL
PROMPT
PSTO
PUTK
RCLS
RECV
REGS?
RESET
RM?
RTN+1
R-CLR
R-COPY
R-SORT
R-SWAP
SAVE
SENDA
SENDP
SENDR
SENDΣ
SKIP
SMODE?
SSIZE?
STOS
TICKS
t⇆
VIEWa
VWa+
WSIZE?
XEQα
y⇆
z⇆
αGTO
αOFF
αON
αXEQ
⇆
🖶ADV
🖶CHR
🖶$^{C}r_{XY}$
🖶PLOT
🖶PROG
🖶r
🖶REGS
🖶STK
🖶TAB
🖶WIDTH
🖶α
🖶α+
🖶Σ
🖶+α
🖶#

[STAT]
COV
L.R.
SEED
SERR
$SERR_w$
SUM
s_w
s_{XY}
$\overline{x}_g$
$\overline{x}_w$
$\hat{x}$
ε
ε_m
ε_P
σ
σ_w
%Σ

[SUMS]
nΣ
Σln^2x
Σln^2y
Σlnx
Σlnxy
Σlny
Σx
Σx^2
Σx^2y
Σxlny
Σxy
Σy
Σy^2
Σylnx

[TEST]
BC?
BS?
CNVG?
DBL?
ENTRY?
EVEN?
FC?
FC?C
FC?F
FC?S
FP?
FS?
FS?C
FS?F
FS?S
gPIX?
INTM?
INT?
KEY?
LBL?
LEAP?
M.SQR?
NaN?
ODD?
PRIME?
REALM?
SPEC?
TOP?
XTAL?
x<?
x≤?
x=+0?
x=-0?
x≈?
x≥?
x>?
∞?
🖶?

[X.FCN]内容随模式集而变化，编程中除外[73]。它在各模式中菜单的内容如下：

alpha 模式：
VERS
x→α
αDATE
αDAY
αIP
αLENG
αMONTH
αRC#
αRL
αRR
αSL
αSR
αTIME
α→x

小数模式：		
$\sqrt[3]{x}$	L_n	W^{-1}
AGM	LN1+x	x^3
ANGLE	$L_n\alpha$	XNOR
BATT	LNβ	x→α
B_n	LNΓ	$\sqrt[x]{y}$
B_n*	MANT	YEAR
CEIL	MAX	αDATE
DATE	MIN	αDAY
DATE→	MOD	αIP
DAY	MONTH	αLENG
DAYS+	NAND	αMONTH
DECOMP	NEIGHB	αRCL
DEG→	NEXTP	αRC#
dRCL	NOR	αRL
DROP	P_n	αRR
D→J	RAD→	αSL
erf	RDP	αSR
erfc	ROUNDI	αSTO
EXPT	RSD	αTIME
e^x-1	SDL	α→x
FIB	SDR	β
FLOOR	SIGN	Γ
GCD	SINC	Γ_p
g_d	SLVQ	Γ_q
g_{d-1}	sRCL	γ_{XY}
GRAD→	STOPW	Γ_{XY}
H_n	TIME	ΔDAYS
H_{np}	T_n	ζ
H.MS+	ULP	$(-1)^X$
H.MS-	U_n	→DATE
IDIV	VERS	%MG
iRCL	WDAY	%MRR
I_X	WHO	%T
J→D	W_m	%Σ
LCM	W_p	%+MG

整数模式：	
$\sqrt[3]{x}$	ROUNDI
ASR	RR
BATT	RRC
CB	SB
CEIL	SEED
DBLR	SIGN
DBL×	SL
DBL/	SR
DROP	sRCL
FB	ULP
FIB	VERS
FLOOR	WHO
GCD	x^3
IDIV	XNOR
LCM	x→α
LJ	$\sqrt[x]{y}$
MASKL	αIP
MASKR	αLENG
MAX	αRCL
MIN	αRC#
MIRROR	αRL
MOD	αRR
NAND	αSL
nBITS	αSR
NEIGHB	αSTO
NEXTP	α→x
NOR	Γ
RJ	$(-1)^X$
RL	×MOD
RLC	^MOD

[CPX] [X.FCN]
$^{C3}\sqrt{x}$
^{C}AGM
^{C}CNST
^{C}CONJ
^{C}CROSS
^{C}DOT
^{C}DROP
$^{C}e^x$-1
^{C}FIB
$^{C}g_d$
$^{C}g_d^{-1}$
^{C}IDIV
^{C}LN1+x
^{C}LNβ
^{C}LNΓ
^{C}SIGN
^{C}SINC
$^{C}W_p$
$^{C}W^{-1}$
$^{C}x^3$
$^{Cx}\sqrt{y}$
^{C}β
^{C}Γ
$^{C}(-1)^X$

[73]在编程模式中，这三个模式被合并了。

以下是 alpha 菜单的内容。小号字体被列印在浅灰色背景上。使用[⇧]可切换大小写。大多数情况下，重音字母和普通字母一样宽。

[CPX]											
À			à			Ñ Ň			ñ ň		
Á			á			Ò			ò		
Â Ã Ā Ă			â ã ā ă			Ó			ó		
Ä			ä(ǎ)			Ô Õ Ō Ǒ			ô õ ō ǒ		
Æ			æ			Ö			ö(ǒ)		
Å			å			Ø			ø		
Ć			ć			Ř			ř		
Č			č			Š			š		
Ç			ç						ß		
Đ			đ			Ù			ù		
È			è			Ú			ú		
É			é			Û Ũ Ū Ŭ			û ũ ū ŭ		
Ê Ē Ě Ĕ			ê ē ě ĕ			Ü			ü(ŭ)		
Ë			ë(ě)			Ů			ů		
			ℏ						x̂/x̄		
Ì			ì			Ý			ý		
Í			í						ŷ/ȳ		
Î Ĩ Ī Ĭ			î ĩ ī ĭ			Ÿ			ÿ		
Ï			ï(ǐ)			Ž			ž		

[./,]																		
,	:	;	‘	“	*	@	_	~	%	$	€	£	¥	⊙	⊕	⏏	`	#

[R⇧]																						
0	°	1	2	2	3	A	B	c	c	d	e	k	m	n	p	q	u	w	x	x	Y	µ	-1	*	∞

[TEST]									
<	≤	=	≈	≥	>	[	]	{	}

[⇨]								
→	←	↑	↓	√	∫	∞	^	↕

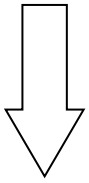
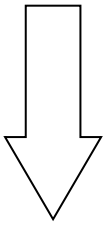
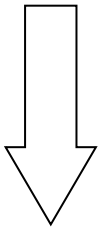
WP 34S 提供的字母可以正确拼写超过 30 亿人使用的希腊字母或拉丁字母简单变体，包含以下语种：

南非语、加泰罗尼亚语、宿务语、捷克语、威尔士语、丹麦语、德语、爱沙尼亚语、英语、西班牙语、巴斯克语、法语、爱尔兰语、加利西亚语、希腊语、克罗地亚语、印尼语、意大利语、爪哇语、斯瓦西里语、海地克里奥尔语、匈牙利语、马来语、荷兰语、挪威语、葡萄牙语、克丘亚语、阿尔巴尼亚语、斯洛伐克语、斯洛文尼亚语、波斯尼亚语、巽他语、芬兰语、瑞典语、他加禄语、瓦赖语，以及中文[74]。这使得 *WP 34S* 成为了目前最通用化、国际化的计算器。如果您知道更多涵盖在内的语种，请通知我们。附录 E中提供了完整的字符集。

[74]汉语普通话有四个音调，一般被转写为例如 mā、má、mǎ 和 mà 的形式。所以我们需要给这里的 ā 和 ǎ 准备不同的字母，并且 e、i、o 和 u 也需要做如上准备。因为字符高度被限制到了六像素，所以在机器的两种字体中很难显示出这些字符，让字母与附加符号能够清晰的辨识。作为一个明确的解决方案，我们建议使用分音符号（没有在汉语拼音方案中被使用）来表示汉语普通话中的上声，就像 Hän–yü pïn–yïn 这样。如果您是汉语拼音使用者，我们希望得到您的理解。

快速访问菜单项

就像在第 117 页所描述的那样，可以仅使用方向键[▲]和[▼]来浏览每个菜单。但是，使用下表中方法，可以更快找到所需要的内容：

<table>
<tr><td rowspan="3">1</td><td>用户输入</td><td>[CONST]、[CONV]、[MATRIX]、[MODE]、[PROB]、[P.FCN]、[STAT]、[TEST]、[SUMS]或[X.FCN]</td><td>在字符模式中的[CPX]或[R⇧]</td><td>[⇨]、[TEST]或[./,]</td></tr>
<tr><td rowspan="2">点阵显示</td><td colspan="3">显示当前菜单的第一个内容</td></tr>
<tr><td>（例如，[TEST]中的 BC?）</td><td>（例如[CPX]中的 ä）</td><td>（例如，[./,]中的;）</td></tr>
<tr><td rowspan="4">2</td><td rowspan="2">用户输入</td><td>欲输入指令的第一个字符</td><td>欲输入的基本字母</td><td rowspan="4">  </td></tr>
<tr><td>（例如，[F]）</td><td>（例如，[U]）</td></tr>
<tr><td rowspan="2">点阵显示</td><td colspan="2">显示以当前字符开头的第一个内容</td></tr>
<tr><td>（例如，FC?）</td><td>（例如，ü）</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">用户输入
点阵显示</td><td>第二个字符（例如，[S]）</td><td rowspan="2">  </td><td rowspan="2">  </td></tr>
<tr><td>以当前字母顺序开头的第一个内容*
（例如，FS?）</td></tr>
<tr><td>…</td><td colspan="4">在找到所想要的内容之前，请用[▼]继续浏览。</td></tr>
<tr><td></td><td></td><td>（例如，FS?C）</td><td>（例如，Ü）</td><td>（例如，€）</td></tr>
<tr><td>n</td><td>用户输入</td><td colspan="2">[XEQ]　或　[ENTER⇧]</td><td></td></tr>
<tr><td rowspan="4"></td><td rowspan="2"></td><td colspan="3">WP 34S 离开菜单并返回到了之前设置的模式。</td></tr>
<tr><td>…并执行或者插入被选中的指令，或者回调被选择的常数。</td><td colspan="2">…并且往字符寄存器中追加所选字符。</td></tr>
<tr><td rowspan="2">点阵显示</td><td>结果</td><td colspan="2">字符寄存器的内容</td></tr>
<tr><td>（例如，true）</td><td colspan="2">（例如，3 Rüben à 0,25€）</td></tr>
</table>

就像是浏览一本词典一样，通过字母序查找比逐一向后浏览更高效。例如，通过按[TEST][F][S][▼]或者[TEST][G][▲][▲][▲] 都可以更方便地输入 FS?C。

可以在搜索中使用前缀[g]来输入希腊字母。例如，通过键入[g]+[A]来输入 α（详见 63 页）。

如果在所选菜单中找不到给出的字符或序列，则将显示以字母顺序排列的第一项。请参阅第 77 页的排序顺序。

可以键入两个以上的字符，但是在三秒之后，或在按[▲]或[▼]之后，搜索字符串将被重置，重新从第一个字符开始。

常数（CONST）

WP 34S 内置 89 个物理、天文和数学常数。第一次输入 CONST 会看到：

所有的常数见下表。天文学和数学的常数被列在彩色背景单元格中。除非单独声明，物理常量的值（包括下面括号中给出的相对标准差[75]）是国际科技数据委员会（CODATA）在 2010 年公布的，并在 2013 年 6 月份被录入进 *WP 34S*。绿色背景的单元格表示精确或极近似的值。而背景越红，相应的常数就越不精确。这些是目前科学界最准确的数值，被全球的国家标准组织广泛采用。

对于单位，记住特斯拉是 $1T=1\frac{Wb}{M^2}=1\frac{V\cdot s}{M^2}$，焦耳是 $1J=1N\cdot m=1\frac{kg\cdot m^2}{s^2}$。另一方面，$1J=1W\cdot s=1V\cdot A\cdot s$，因此，$1\frac{J}{T}=1A\cdot m^2$。

可以使用这里提供的常数来推导出更多实用的等式，例如，用电子伏特来表示焦耳（$1A\cdot s\cdot V=\frac{1}{e}eV\approx 6.24\cdot 10^{18}eV$）。或者经由电磁辐射频率计算出波长（$\lambda=c/f$），或者是您所能想到的任何公式变换。

下面是完整的常数清单：

编号	名称	数值	附注
0	1/2	0.5	微不足道的常量，但它在一些迭代运算中很实用。
1	d	365.242 5 *d*（根据定义）	格里高利年（公历）的天数
2	da	5.291 772 109 2E-11*m* (3.2E-10)	玻尔半径[76]，$a_0=\frac{\alpha}{4\pi\cdot R_\infty}$
3	dm	384.4E6 *m* (1E-3)	月球轨道的半长轴
4	d⊕	1.495 979E11 *m* (1E-6)	地球轨道的半长轴。在上述的不确定性范畴内，它等于 1AU
5	c	2.997 924 58E8 m/s（根据定义）	真空光速
6	c1	3.74177153E-16 $m^2\cdot W$ (4.4E-8)	第一辐射常数 $c_1=2\pi\cdot h\cdot c^2$

[75]括号中的数字的精度，是通过高斯（1777 – 1855）提出的"误差传递"计算方法获得的。这个值在 CONST 菜单中没有给出。如果结果要值得信赖，这个程序是必不可少的，而且不仅仅是在科学上才需要。请查阅合适的参考资料。用尺测量是不能产生精确到四位小数的结果的。

[76]这是氢原子半径的一个很好的估计。另一方面，原子核的典型半径约为 $10^{-15}m$。因此，原子核的体积远远低于原子体积的十亿分之一。因此，原子内几乎全是空的。

编号	名称	数值	附注
7	c2	0.014 387 770 (9.1E-7)	第二辐射常数 $c_2 = hc/k$
8	e	1.602 176 565E-19 C (2.2E-8)	电子电荷 $e = \frac{2}{K_J R_K} = \Phi_0 G_0$
9	eE	2.718 281 828 459 045...	欧拉提出的自然常数 e。注意，本表中其它的 e 均指电子电荷。
10	F	96 485.336 5 $\frac{C}{mol}$ (2.2E-8)	法拉第常数 $F = e \cdot N_A$
11	Fα	2.502 907 875 095 892 8...	费根鲍姆 α 常数和 δ 常数
12	Fδ	4.669 201 609 102 990 6...	
13	g	9.806 65 m/s^2（根据定义）	标准地球重力加速度
14	G	6.673 84E-11 $\frac{m^3}{kg \cdot s^2}$ (1.2E-4)	牛顿万有引力常数，也称为 γ。精确值见下文 GM。
15	G0	7.748 091 734 6E-5/Ω (3.2E-10)	量子电导 $G_0 = 2e^2/h = 2/R_K$
16	Gc	0.915 965 594 177...	卡塔兰常数
17	gc	-2.002 319 304 361 53 (2.6E-13)	朗德 g 因子
18	GM	3.986004418E14 m^3/s^2 (2.0E-9)	计及地球大气的牛顿万有引力常数（根据 WGS84[77]）
19	h	6.626 069 57E-34 $J\,s$ (4.4E-8)	普朗克常数
20	ħ	1.054 571 726E-34 $J\,s$ (4.4E-8)	$= h/2\pi$，即狄拉克常数
21	k	1.380 648 8E-23 J/K (9.1E-7)	玻尔兹曼常数
22	Kj	4.835 978 70E14 Hz/V (2.2E-8)	约瑟夫逊常数 $K_j = 2e/h$
23	lP	1.616 199E-35 m (6.0E-5)	普朗克长度 $l_P = \sqrt{\hbar G/c^3} = t_p c$
24	me	9.109 382 91E-31 kg (4.4E-8)	电子质量
25	Mm	7.349E22 kg (5E-4)	月球质量
26	mn	1.674 927 351E-27 kg	中子质量

[77] WGS84 是一个用于测绘和 GPS 的地球表面模型（参见 *http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html*）。

编号	名称	数值	附注
		(4.4E-8)	
27	m_p	1.672 621 777E-27 *kg* (4.4E-8)	质子质量
28	M_p	2.176 51E-8 *kg* (6.0E-5)	普朗克质量 $Mp=\sqrt{\hbar c/G}\approx 22\mu g$
29	m_u	1.660 538 921E-27 *kg* (4.4E-8)	原子质量单位=$10^{-3}kg/N$
30	m_uc^2	1.492 417 954E-10 *J* (4.4E-8)	原子质量单位的能量当量
31	m_e	1.883 531 475E-28 *kg* (5.1E-8)	介子质量
32	$M_\odot$	1.989 1E30 *kg* (5E-5)	太阳质量
33	$M_\oplus$	5.973 6E24 *kg* (5E-5)	地球质量
34	N_A	6.022 141 29E23 / *mol* (4.4E-8)	阿伏伽德罗常数
35	NaN	not numeric	“不是一个数字”，例如，在 $x\le 0$ 情况下的 $\ln(x)$ 或在非复数定义域下的 $\tan 90^\circ$。所以，NaN 覆盖了极点和函数结果未定义的区域。注意，正负无穷大被视为数字（详见本表结尾）。在标志位 **D** 未被置位的情况下，返回非数字结果将引发错误信息。
36	p_0	101 325 *Pa*（根据定义）	标准大气压
37	q_p	1.875 545 9E-18 A *s* (6.0E-5)	普朗克电荷 $q_p=\sqrt{4\pi\varepsilon_0\hbar c}\approx 11.7e$。这是 CODATA 2006 年的数据，截至 2010 年，还没有更新的数据。
38	R	8.314 462 1 $\frac{J}{mol\cdot K}$ (9.1E-7)	摩尔气体常数
39	r_e	2.817 940 326 7E-15 *m* (9.7E-10)	经典电子半径 $r_e=\alpha^2\cdot a_0$
40	R_K	25 812.807 443 4 Ω (3.2E-10)	克里青（von Klitzing）常数 $R_K=h/e^2$
41	R_m	1.737 530E6 *m* (5E-7)	月球平均半径
42	R_∞	1.097 373 156 853 9E7 / *m* (5.0E-12)	里德伯常量 $R_\infty=\alpha^2 m_e c/2h$
43	$R_\odot$	6.96E8 *m* (5E-3)	太阳平均半径
44	$R_\oplus$	6.371 010E6 *m* (5E-7)	地球平均半径
45	R_a	6.378 137 0E6 *m*（根据定义）	基于 WGS84 模型的地球短半轴，GPS 将其用于定义地球表面以及其它测量需求。
46	R_b	6.356 752 314 2E6 *m* (1.6E-11)	WGS84 定义的地球短半轴
47	R_e^2	6.694 379 990 14E-3	WGS84 定义的地球第一偏心距

编号	名称	数值	附注
		(1.5E-12)	
48	Se'2	6.739 496 742 28E-3 (1.5E-12)	WGS84 定义的地球第二偏心距（抱歉，在标准文件中，它的确被称作 e'^2 ）
49	Sf-1	298.257 223 563（根据定义）	WGS84 定义的平滑参数
50	T₀	273.15 K（根据定义）	$=0^\circ C$，标准温度
51	tP	5.391 06E-44 s (6.0E-5)	普朗克时间 $t_P=\sqrt{\frac{\hbar G}{c^5}}=\frac{l_p}{c}$
52	TP	1.416 833E32 K (6.0E-5)	普朗克温度 $T_P=\frac{c^2}{k}\sqrt{\frac{\hbar c}{G}}=\frac{M_p c^2}{k}=\frac{E_p}{k}$
53	Vm	0.022 413 968 m^3/mol (9.1E-7)	标准条件下理想气体的摩尔体积 $V_m=RT_0/P_0\approx 22.4\frac{l}{mol}$
54	Z₀	376.730 313 461…Ω	自由空间阻抗
55	α	7.297 352 569 8E-3 (3.2E-10)	精细结构常数 $\alpha=\frac{e^2}{4\pi\varepsilon_0\hbar c}\approx\frac{1}{137}$
56	γEM	0.577 215 664 901 532 86…	欧拉-马歇罗尼常数 γ_{EM}
57	γP	2.675 222 005E8 $\frac{1}{s\cdot T}$ (2.4E-8)	质子回旋磁比 $\gamma_p=2\mu_P/\hbar$
58	ε₀	8.854 187 817…E-12 $\frac{As}{Vm}$	绝对介电常数或真空电容率 $\varepsilon_0=\frac{1}{\mu_0 c^2}$
59	λC	2.426 310 238 9E-12 m (6.5E-10)	分别为电子、中子和质子的康普顿波长，对于电子有 $\lambda_C=\frac{h}{m_e c}$ 。
60	λCn	1.319 590 906 8E-15 m (8.2E-10)	
61	λCP	1.321 409 856 23E-15 m (7.1E-10)	
62	μ₀	1.256 637 061 4…E-6 $\frac{As}{Vm}$	真空磁导率 $\mu_0:=4\pi\cdot 10^{-7}\frac{V\cdot s}{A\cdot m}$
63	μB	9.274 009 68E-24 J/T (2.2E-8)	玻尔磁子 $\mu_B=e\hbar/2m_e$
64	μe	-9.28476430E-24 J/T (2.2E-8)	电子磁矩
65	μn	-9.662 364 7E-27 J/T (2.4E-7)	中子磁矩
66	μP	1.410606743E-26 J/T (2.4E-8)	质子磁矩

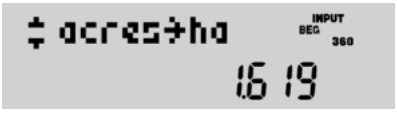
编号	名称	数值	附注
67	μₙ	5.050 783 53E-27 J/T (2.2E-8)	核磁子 $\mu_u = e\hbar/2m_p$
68	μₚ	-4.490 448 07E-26 J/T (3.4E-8)	μ 介子磁矩
69	σ_	5.670 373E-8 $\frac{W}{m^2K^4}$ (3.6E-6)	斯特藩-玻尔兹曼常数 $\sigma_B = \frac{2\pi^5k^4}{15h^3c^2}$
70	Φ	1.618033988749 894…	黄金分割比 $\Phi = \frac{1+\sqrt{5}}{2}$
71	Φ₀	2.067 833 758E-15 Vs (2.2E-8)	磁通量量子 $\Phi_0 = h/2e = 1/K_J$
72	ω	7.292 115E-5 rad/s (2E-8)	WGS84 定义的地球角速度
77	−∞	-1 nFi ni tY	希望数学的主能原谅我们把它们称为“常数”。注意，在 *WP 34S* 中，两者都被视为数值类型。
74	∞	1 nFi ni tY	
75	#		参阅 *IOP* 最后一个指令。
76	(76）	1	额外数值常数[78]
77	(77)	0.367879441171…	1/e（其中高的一半位数）
78	(78)	8.67445811131…·10-34	1/e（其中低的一半位数）
79	(79)	0.4472135955…	1/√5
80	(80)	0.564189583548…	1/√π
81	(81)	0.2214	
82	(82)	0.434294481903…	1/ln(10)
83	(83)	0.69314718056…	ln(2)
84	(84)	1.44269504089…	1/ln(2)
85	(85)	3.14159265359…	π
86	(86)	1.57079632679…	π/2
87	(87)	2.50662827463…	√(2π)
88	(88)	122.134	

[78]76 到 88 号常数只能通过 CNST 指令调用。81 号到 88 号常数仅供内部使用且可能不经通知变更。

单位换算（CONV）

WP 34S 的[CONV]有 88 个单位换算功能。它提供了将地区单位转换到公制单位的方法[79]，反之亦然。本菜单的导航方式与在其它菜单中的一样。然而，这里有一个特例：按[f][B]（即[1/x]）将对屏幕所显示的转换进行反换算并离开 CONV 菜单。

示例：假定当前显示模式为 FIX3。之后输入 [4][h][CONV][A]，则屏幕上显示



这说明 4 英亩等于 1.619 公顷。

现在按[f][B]，屏幕上会显示

表示 9.884 英亩等于 4 公顷。

再按[h][CONV]，屏幕上显示

这证明了上个过程是正确的。

通过按[EXIT]来离开单位换算模式，并将结果  显示到屏幕上。

下表按字母顺序列出了本机所提供的全部换算：

换算	公式	备注	类型
°C→°F	×1.8 + 32		温度
°F→°C	- 32)/ 1.8		温度
°→G	/ 0.9	转换到百分制角度，英文也叫 *grads* 或 *gradians*	角度
°→rad	×π / 180	和 D→R 相同，转换到弧度	角度
acres→ha	×0.404 686	1 公顷（*ha*）=10^4 平方米	面积
acreUS→ha	×0.404 687	acreUS 基于美国测绘英尺（*U.S. Survey foot*），参见后面的 feetUS	面积
ar.→dB	$20\lg(a_1/a_2)$	幅值比	比例
atm→Pa	×1.013 25E5	1 帕=1 牛每平方米	压力
AU→km	×1.495 98E8	天文单位	长度
bar→Pa	×1E5	1 毫巴=100 帕	压力

[79]在国际范围内，公制单位是被普遍认可的。它被世界上的所有国家所采用（除了两个例外）。因此，对于大多数读者来讲，出现在单位换算功能中的大多数单位至少是看似已过时。然而，在这个世界的某些角落（那里也使用英语），这些过时的单位很难被淘汰。

这个表格也能让您意识到在计量单位被统一之前，我们在度量衡的世界中遇到的混乱景象。

换算	公式	备注	类型
Btu→J	×1 055.06	英国热量单位	能量
cal→J	×4.186 8		能量
cft→l	×28.316 9	立方英尺	体积
cm→inches	/ 2.54		长度
cwt→kg	×50.802 4	1 英担=112 磅	质量
dB→ar.	$10^{R_{dB}/20}$	振幅比	比例
dB→pr.	$10^{R_{dB}/20}$	功率比	比例
fathom→m	×1.828 8		长度
feetUS→m	×0.304 801	准确的说，1 美国测绘英尺[80]=$^{1200}/_{1937}$ 米	长度
feet→m	×0.304 8	即 1959 年定义的所谓“国际英尺”，1 英尺=1/3 码	长度
flozUK→ml	×28.413 1	1 毫升=1 立方厘米 1 升=10^{-3} 立方米	体积
flozUS→ml	×29.573 5		
galUK→l	×4.546 09		
galUS→l	×3.785 42		
G→°	×0.9		角度
g→oz	/ 28.349 5	盎司	质量
G→rad	×π / 200		角度
g→tr.oz	/ 31.103 5	金衡盎司	质量
ha→acres	/ 0.404 686	1 公顷（*ha*）=10000 平方米	面积
ha→acreUS	/ 0.404 687	acreUS 基于美国测绘英尺（*U.S. Survey foot*）	面积
hp(E)→W	×746	电马力	功率
hp(I)→W	×745.700	英制马力	功率
hp(M)→W	×735.500	公制马力。1*hp(M)*=1*PS*	
inches→cm	×2.54	1 英寸=1/12 英尺	长度
inHg→Pa	×3 386.39		压力
J→Btu	/ 1 055.06		能量
J→cal	/ 4.186 8		能量
J→kWh	/ 3.6E6		能量
kg→cwt	/ 50.802 4	1 英担=112 磅	质量
kg→lb	/ 0.453 592		质量
kg→stones	/ 6.350 29		质量
kg→s.cwt	/ 45.359 2	1 美担=100 磅	质量
km→AU	/ 1.495 98E8	天文单位	长度
km→l.y.	/ 9.460 73E12	光年	长度
km→miles	/ 1.609 344		长度
km→nmi	/ 1.852	海里	长度
km→pc	/ 3.085 68E16	秒差距	长度

[80]NIST 已宣布美国测绘英尺自 2022 年之后不再在 NSRS 中使用，统一使用改称英尺的原国际英尺。参见：*https://www.nist.gov/pml/us-surveyfoot*。

换算	公式	备注	类型
kWh→J	×3.6E6		能量
lbf→N	×4.448 22		力量
lb→kg	×0.453 592		质量
l.y.→km	×9.460 73E12	光年	长度
l→cft	/ 28.316 9	1 升=10^{-3} 立方米	体积
l→galUK	/ 4.546 09		
l→galUS	/ 3.785 42		
miles→km	×1.609 344		长度
ml→flozUK	/ 28.413 1	1 毫升=1 立方厘米	体积
ml→flozUS	/ 29.573 5		
mmHg→Pa	×133.322		压力
m→fathom	/ 1.828 8		长度
m→feet	/ 0.304 8		长度
m→feetUS	/ 0.304 801		长度
m→yards	/ 0.914 4		长度
nmi→km	×1.852	海里	长度
N→lbf	/ 4.448 22		力量
oz→g	×28.349 5	盎司	质量
Pa→atm	/ 1.013 25E5	1 帕=1 牛每平方米	压力
Pa→bar	/ 1E5		压力
Pa→inHg	/ 3 386.39		压力
Pa→mmHg	/ 133.322		压力
Pa→psi	/ 6 894.76	磅每平方英寸	压力
Pa→torr	/ 133.322	1托=1毫米汞柱	压力
pc→km	×3.085 68E16	秒差距	长度
pr.→dB	$10\lg\left(P_1/P_2\right)$	功率比	比例
psi→Pa	×6 894.76		压力
rad→°	×180 / π	和 R→D 相等，转换为角度制	角度
rad→G	×200 / π	转换为百分制角度	角度
stones→kg	×6.350 29		质量
s.cwt→kg	×45.359 2	1 短担=100 磅	质量
s.tons→t	×0.907 185	短吨	质量
tons→t	×1.016 05	长吨	质量
torr→Pa	×133.322		压力
tr.oz→g	×31.103 5	金衡制盎司	质量
t→s.tons	/ 0.907 185	1 吨=1000 千克	质量
t→tons	/ 1.016 05		
W→hp(E)	/ 746		功率
W→hp(I)	/ 745.700		功率

换算	公式	备注	类型
W→hp(M)	×735.499		功率
yards→m	×0.914 4		长度

常数 T_0 对于温度换算也是有用的，它在常数菜单中。因为温度换算仅仅是简单的加减法而已，所以这里没有收录。

当然也可以根据需要来将换算进行组合。例如：倘若想给轮胎充最多不超过 30 磅每平方英寸的气，那么，下列过程能帮助您在欧洲及其周边的加气站加气。

[3][0][h][CONV][P][S][XEQ]

[h][CONV][P][▼][XEQ] 结果是 2.1 巴。

现在您能安心地打气而不至于担心把轮胎给打爆。

在特定的紧急情况下[81]，记住：贝克勒尔与赫兹相等，格瑞是沉积或吸收能量的单位（$1Gy=1J/kg$），并且希沃特（Sv）是格瑞与一个的剂量转换系数的乘积，用于计量辐射对人体造成损害。

在这个领域，一些过时的单位也会在老文献中出现：《致居里夫人的朋友们》（*Pour les amis de Mme. Curie*），$1Ci=3.7\cdot10^{10}Bq=3.7\cdot10^{10}decays/s$。为了纪念第一个诺贝尔物理学奖获得者的人——发现 X 射线的伦琴先生（他在实验中毁掉了自己的手），物质辐射产生的电荷被下列单位表示：$1R=2.58\cdot10^{-4}As/kg$。在数十年之前，单位雷姆（即人体伦琴当量）曾取代现在的希沃特。$1rem=10mSv$，$1rad=10mGy$。

预定义全局 alpha 标签（CAT）

除标签“δx”是为导数计算时的步长预留的（详见 *IOP* 中的 f'(x)）之外，还有更多的附加标签可能已经为特定任务提供了。当相应的库例程已被加载到闪存（详见下文）中时，它们就会在 CAT 中被列出。因此，它们将不会占用 *RAM* 的用户程序空间。

目前所有库程序都能在 *WP 34S* 的网站 *http://wp34s.svn.sourceforge.net/viewvc/wp34s/library/*[82]目录中找到。按照惯例，它们是扩展名为“.wp34s”的纯文本文件。这些程序包括一套基本的三维向量操作、一个 TVM（货币时间价值）应用程序、包括矩阵编辑器在内的一些矩阵程序等等。可以通过类似于记事本的应用来打开这些纯文本文件，还可以在每个文件的开头部分（有的是在结尾部分）找到必要的用户信息。这些例程的编写类似于用户程序，但还用到了附录 H中所解释的一些扩展指令集。

库文件也包含在以源文件（*.wp34s）发布的 zip 文件和预编译库

[81]译者注：指核辐射环境，例如发生了切尔诺贝利和福岛核电站这样的核泄露事故。

[82]译者注：如果该地址失效，也可以在 *https://sourceforge.net/p/wp34s/code/HEAD/tree/library/*找到。

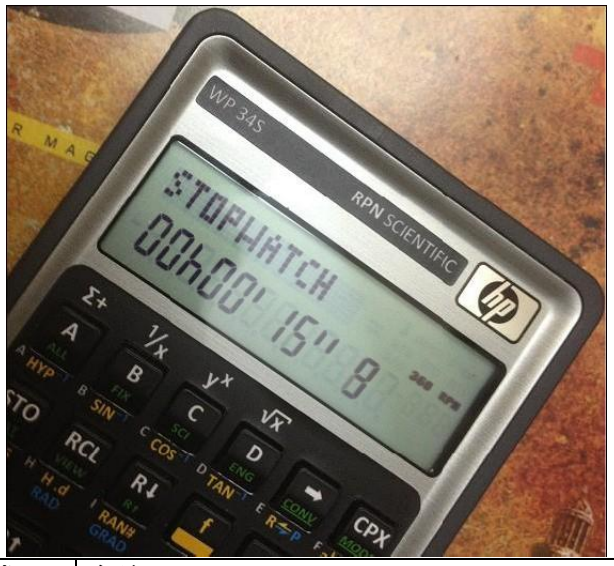
（wp34s-lib.dat -3*kb*）中。这个预编译库是固件文件 calc_full.bin、calc_xtal_full.bin 和 calc_ir_full.bin 中的一部分。因此当将以上任意一个固件文件烧录进 *WP 34S* 时（烧录的方法详见附录 A），就会获取完整的库文件。

将 wp34s-lib.dat 复制进 *WP 34S* 模拟器运行目录中后，可以通过 CAT 来在模拟器中访问这些例程。详见附录 D。

若想自己编写这样的例程，请参阅附录 H。

7. 秒表程序

如 *IOP* 中所述，本机提供了秒表功能。它可以在安装了晶振和相应固件的 *WP 34S* 上使用（请参阅附录 A 和附录 H）。它在模拟器上也可以运行。



命令	按键	所处模式	备注
STOPW	[h][X.FCN]STOPW **[CPX][R/S]**	DECM ¬sto	HP-55 计时器风格的秒表[83]。当启动 STOPW 后，开始计时之前，显示屏将如下所示。 STOPWATCH 00h00' 00'' 0

使用 STOPW 时，以下按键可以使用。

[R/S]	在不改变计时值的情况下启动或停止计时。
[CLx]或**[⇦]**	在不改变计时状态（计时或停止）的情况下复位计时器。
[EEX]	隐藏或显示十分之一秒。默认显示。
[n][n]	设置当前寄存器地址（CRA，初始默认值为 00）。输入将显示在指数部分，如下所示[84]。 STOPWATCH 27h31'55'' 5 01
[ENTER⇧]	将当前计时器值以 `hhh.mmssd` 格式存储在当前寄存器中，而不更改计时器状态或值。然后，它会将 CRA 加 1 并显示，如上所示。
[⇨]	隐藏或显示 CRA。默认值为隐藏。
[▲]或**[▼]**	分别减少或增加 CRA。
[.]	相当于一次按下**[ENTER⇧]**和**[CLx]**，但会在屏幕顶部更新显示自上次按下**[CLx]**或**[⇦]**后的总时间。可用于记录圈时。 01h03'13'' 06 00h02' 29'' 或 00h02'31'' 7 03 00h00' 49'' 5 **注意：**总时间将在按下**[CLx]**或**[⇦]**后消失。
[A]	将当前计时的值累加到求和寄存器中，和**[Σ+]**类似。这样在退出 STOPW

[83]有两个区别：*WP 34S* 不会将 **X** 的内容作为开始时间。开始时间由 RCL 指定。并且 *WP 34S* 会显示十分之一而不是百分之一秒。

[84]尝试指定超出分配的地址范围的 CRA 将会被阻止，并可能导致在指数部分显示 "- -" 或类似的符号。

在 *HP-55* 上，只需输入一个数字就足以存储，因为在那里只有 10 个寄存器用于此目的。此外，没有自动地址递增功能。

	后可以计算算术平均值和标准差。
[+]	相当于一次按下**[A]**和**[.]**。
[RCL]*nn*	在不更改计时器状态的情况下调用 ***rnn***。比如，调用此值作为开始时间用于计时。
[EXIT]	离开应用程序。除非计时已经停止，否则计时器将继续在后台计时（由小'='闪烁指示）。停止计时器需在 STOPW 中按下**[R/S]**或关闭 *WP 34S*[85]。

注意：其他任何键在 STOPW 下无法使用。要增加或减少分段计时，必须离开此应用程序。STOPW 不可用于编程。

秒表的最大显示范围为 99h59' 59'' 9，超过范围后内部继续计时，显示变为计时值对 100 的模数。

[85]固件版本 v3.3 的 *WP 34S* 不会在计时器运行时自动关闭。这意味着定时器运行时，tick 计数器（参见 TICKS）可能会溢出。在 STOPW 内，如果计时器没有运行，自动关机设置为 1:49 小时（即 65536*ticks*）。请注意，无论定时器状态如何，更早期固件版本的 *WP 34S* 将在约 5 分钟后自动关机。

附录 A 制作和升级 *WP 34S*

如何刷机

除非购买了一台已经刷好的 *WP 34S*，您需要自己进行刷机。刷机之前，需要一台未改装的 *HP-20b* 或 *HP-30b* 计算器，一个特制的数据线，一个二进制 ROM 文件，一台计算机和一个传输软件。

传输软件叫 *MySamBa*。可以从 *http://sourceforge.net/projects/wp34s/files/FlashTool/*下载 `MySamBa.zip` 然后解压。

传输到计算器中用来刷机的二进制 ROM 文件叫 `calc.bin`，可以从 *http://sourceforge.net/projects/wp34s/files/* 下载包含它的压缩包。也可以从 *http://wp34s.svn.sourceforge.net/viewvc/wp34s/trunk/realbuild/*单独下载以下文件之一：

`calc.bin`：有最大的 *FM 空间*，但是不支持实时时钟指令（DATE，TIME，SETDAT，和 SETTIM）、STOPW 指令和打印指令。没有 *FM* 程序库。

`calc_full.bin`：和 `calc.bin` 相同，但是包含 *FM* 程序库。

`calc_xtal.bin`：用于安装了晶振的机器，包含实时时钟指令和 STOPW 指令，但是没有打印指令，没有 *FM* 程序库。

`calc_xtal_full.bin`:和 `calc_xtal.bin` 相同,但是包含 FM 程序库。

`calc_ir.bin`：包含 `calc_xtal.bin` 的所有内容，再加上打印指令。需要同时安装晶振和红外二极管。没有 *FM* 程序库。

`calc_ir_full.bin`：和 `calc_ir.bin` 相同，但是包 *FM* 程序库。

STOPW 程序要占用 1.5*kB* 的 *FM* 容量，打印功能需要 3.5*kB* 以上的容量。这个需要您自己权衡取舍。无论用哪个 ROM，一定要检查计算器确保其符合对应的硬件要求（见附录 H），否则计算器很可能死机。

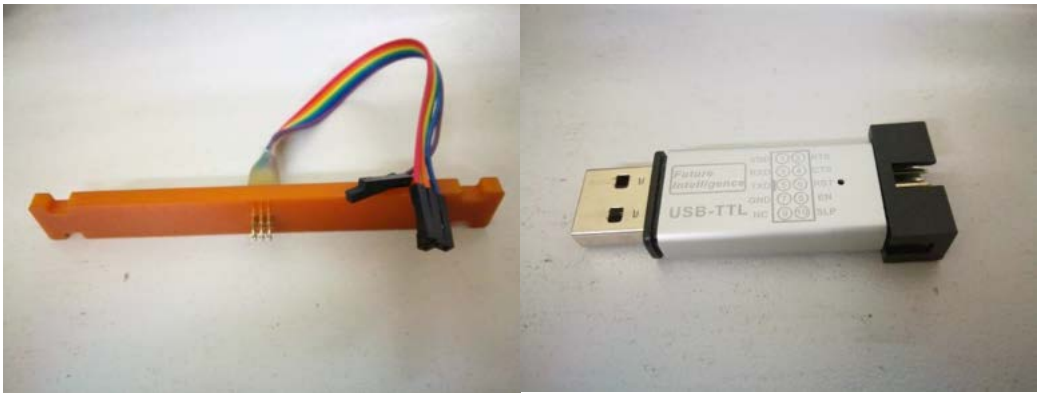
编程数据线目前有三种方案，根据出现的时间顺序如下：

A. 限量生产的惠普原装编程数据线。这种线已经停产且很难买到。如果您有一条，请保存好。还需要一个有 9 针串口的电脑来使用它。如果你的电脑没有这个接口，需要一个 USB—串口转换器将数据线和电脑连接。根据我们的经验，FTDI 芯片的转换器工作良好——其它的可能不行。这个转换器可以在网上购买。

警告：只要这个数据线连在计算器上，它就会从计算器取电，消耗电池电力。如果计算器在刷机过程中碰巧死机了，计算器的处理器可能会在全速运行状态下，在您查找故障原因的时候消耗纽扣电池的电量。因此，请在暂时不需要联机的时候断开数据线。如果刷机比较频繁，用一个外置的 3V 电源性价比更高。连接外置电源务必小心，将'+'连接到外侧电池触点，'-'连接到内侧电池触点。刷机时电压最好在 3V，不过电压不低于 2.5V 亦可。

B. 如果安装了 *Harald Pott* 开发的定制电路板，可以用一根普通的 micro USB 线通过 USB 口将电脑和计算器连接。联机的时候，计算器将从 USB 线取电。不再需要专用编程数据线或硬件串口，同时避免了电池消耗的问题。但是需要改装计算器来安装这个电路板。附录 H 有详细的改装指南。

C. 一个定制的转换线，一头是 6 针编程接头，另一头是杜邦线插头。还需要一个留有杜邦线插针的 USB 转 TTL 小板来使用它。这种线和 TTL 小板都可以在网上买到。



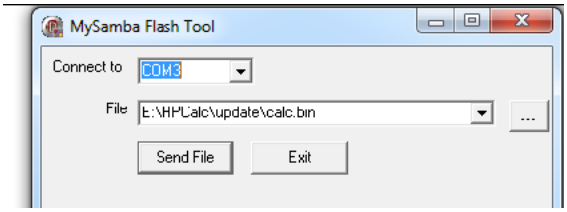
在准备好计算器、电脑、软件、ROM 文件和编程数据线后，根据以下 A、B 或 C 三个步骤之一将 *HP-20b* 或 *HP-30b* 转换为 *WP 34S*。

警告：刷机将完全擦除原有固件，金融计算器将不复存在。原固件将被下载的新固件取代。刷机后您将得到一台 *WP 34S* RPN 科学计算器，功能如同本手册所描述[86]。计算器在步骤 6 和 9 之间不会有任何响应，看起来和死机了一样。请放心，计算器并没有死机。如果刷机操作被中断也不用担心。记住，RESET 总能关闭计算器，按下**[ON]**则会开机[87]。然后从步骤 1 重新开始就可以了。

如果您有一条惠普原装编程数据线：

A1. 打开背部的电池仓盖，用数据编程线将计算器和电脑连接好。

A2. 启动 *MySamBa*。选择通讯端口和要传输的 ROM 文件。窗口显示如下：



先不要开始传输。

A3. 按下**[ON]**开机。

A4. 按下编程数据线上的 ERASE 键（在步骤 A7 之前不要松开）。

A5. 按一下编程数据线上的 RESET 键，计算器将关机。

A6. 按下**[ON]**再次开机。

A7. 松开 ERASE 键，按一下 RESET 键，计算器再次关机。

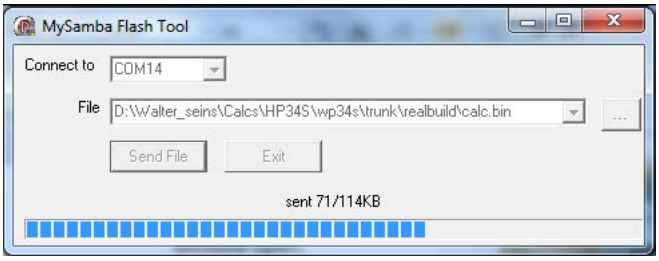
A8. 按下**[ON]**再次开机。此时计算器无响应，这是正常的。

A9. 点击 *MySamBa* 窗口的"Send File"按钮，等待数据传输结束（大约需要 20 秒）。如果有 FTDI USB—串口转换器，会看到蓝色 TX 灯闪烁。如果 *MySamBa* 没有成功刷机[88]，回到 A3 步骤。

[86]如果您想将计算器恢复成原来的惠普金融计算器，您可以将原来的 ROM 刷回去。我们的网站保存了一个 *HP-30b* 的固件。注意这个固件仅仅是我们所知道的最新的，且不再更新。

[87]只要电池有电，即便屏幕上什么都不显示。

[88]如果出现(a)“Error, could not connect to calculator”或者(b)“Unable to connect”，可能的原因有：(a)错误



A10. 按下 RESET 再次关闭计算器。退出 *MySamBa*，断开编程数据线，合上电池仓盖。

A11. 按一下**[ON]**，您的计算器现在是 *WP 34S* 了。

如果已经安装了定制的 USB 电路板，计算器将由 USB 供电。标准的 USB 线上没有专用编程线上的 ERASE 和 RESET 按钮，因此需要按如下步骤刷机：

B1. 打开背部的电池仓盖，用 microUSB 线将计算器的 USB 口和电脑连接好。

B2. 启动 *MySamBa*。和 A2 步骤一样，选择通讯端口和要传输的 ROM 文件（也参见 195 页）。先不要开始传输。

B3. 按下**[ON]**开机。

B4. 临时短接计算器背部 RESET 孔下方六个触点中的右上和左下两个触点，用一个回形针就行，不要断开。

B5. 用合适的针状物捅一下 RESET 孔。计算器将关机。

B6. 保持 B4 步骤的短接，按下**[ON]**再次开机，这会擦除掉旧的固件。

B7. 断开短接，按一下 RESET 键，计算器再次关机。

B8. 按下**[ON]**再次开机。此时计算器无响应，这是正常的。

B9. 点击 *MySamBa* 窗口的"SendFile"按钮，等待数据传输结束（大约需要 20 秒）。如果 *MySamBa* 没有成功刷机 [88]，回到 B3 步骤。

B10. 按下 RESET 再次关闭计算器。退出 *MySamBa*，解除数据线连接，合上电池仓盖。

B11. 按一下**[ON]**，您的计算器现在是 *WP 34S* 了。

如果有一个定制的转接线和 TTL 板，请按如下步骤刷机：

C1. 将 TTL 板和转接线连接好，连接 VCC、GND、TXD 和 RXD 四个针脚即可。TTL 板的 RXD 接转接头的 TXD，TTL 板的 TXD 接转接头的 RXD。刷机接口定义参见下图。

的 COM 端口号或者错误的端口驱动程序，(b)编程数据线没有连接好或者计算器没开机。在编程数据线连接的情况下，电池可能会被耗尽。按下 RESET，断开连接，思考并检查，再重新连接和尝试。如果反复多次不成功，请转到 172 页。

C2. 按下**[ON]**开机，打开背部的电池仓盖。

C3. 临时短接计算器背部 RESET 孔下方六个触点中的右上和左下两个触点，用一个回形针就行，不要断开。

C4. 用合适的针状物捅一下 RESET 孔。计算器将关机。

C5. 保持 C4 步骤的短接，按下**[ON]**再次开机，这会擦除掉旧的固件。

C6. 断开短接，捅一下 RESET 孔，计算器再次关机。

C7. 按下**[ON]**再次开机。此时计算器无响应，这是正常的。

C8. 启动 *MySamBa*。和 A2 步骤一样，选择通讯端口和要传输的 ROM 文件（也参见 195 页）。先不要开始传输。

C9. 将 TTL 板插入电脑的 USB 端口，计算器背部朝上，将转换线的插针插入到计算器背部 RESET 孔下方六个触点，注意插针插入的方向（VCC 在右上角），保持接触良好。

C10. 点击 *MySamBa* 窗口的"SendFile"按钮，等待数据传输结束（大约需要 20 秒）。如果 *MySamBa* 没有成功刷机，跳转到步骤 C13。

C11. 解除转接线和计算器的连接，退出 *MySamBa*。用合适的针状物捅一下背部的 RESET 孔，合上电池仓盖。

C12. 按一下**[ON]**，您的计算器现在是 *WP 34S* 了。

如果没有成功刷机：

C13. 检查 RESET 孔下方六个触点的孔是否偏小，部分机器可能因为外壳注塑精度问题，导致开孔偏小，插针无法深入到底部和触点良好接触。

C14. 如果开孔偏小，用合适的工具扩孔。用一个 1.4mm 的麻花钻就可以。

C15. 跳转到 C10 的步骤。如果还是不行，跳转到 C2。

购买和自己制作键盘贴

刷机成功后，由于大部分按键键位和原来的金融计算器不同，还需要一份键盘贴。

可以从 *Eric Rechlin* 那里买到自粘贴 PVC 键盘贴，地址为：*http://commerce.hpcalc.org/overlay.php*。可以选择的功能键上的 f，g 和 h 文字颜色为白色或者彩色。除号可以选择'/'或者'÷'。

先将键盘贴底板部分揭下来，仔细调整位置，从 ENTER 和 STO 之间的位置开始贴，再向顶部和底部延伸。PVC 贴纸不应和按键有接触。贴好底板后，揭下单独的键帽部分，粘贴到对应的按键上。如果需要调整，先用小刀挑起一个角，再用指甲或镊子揭开。

在国内可以在网上买到 *WP 34S* 自粘贴 PP 材质键盘贴，有黑色和白色两种可选。拿到自粘贴键盘贴后，先将贴膜贴到键盘贴上（这样键盘贴会比较耐用），仔细裁剪键盘底板，将底板粘贴到计算器上，粘贴方法同上。然后裁下键帽部分，仔细贴到按键上。裁剪的工具推荐用钢尺和雕刻刀——这样在底板上开孔要容易一些。如果不想在底板上开孔，也可以直接裁成条状。

如果暂时没有买到自粘贴键盘贴，可以打印 217 页的附图来当作临时键盘贴。按比例将附图打印（图片的宽度为 68mm），然后把它整张剪下来，用透明胶贴在 *WP 34S* 键盘上即可。就算纸面上没有开孔也可以正常操作按键，而且可以一直用到键盘贴到货为止。

升级 *WP 34S*

当新的固件发布后，您可能想自己升级到最新版本。我们建议您每次升级前备份您的工作（用 SAVE 指令，保存成功后会显示 `Saved`）。新的固件启动时会自动恢复这个备份——只要没有按下编程数据线上的 ERASE 按钮。

下载了正确的二进制 ROM 文件（见 139 页）后，按如下步骤升级：

1. 打开电池仓盖
2. 用编程数据线，或者 USB 线将计算器和电脑连接。切记不要触碰编程数据线上的按钮。
3. 启动 *MySamBa*，输入端口号和 ROM 文件信息（见 139 页）。先不要开始

传输。

4. 按下**[ON]**开机，同时按下**[ON]**和**[D]**。屏幕会显示 `DebugON`。松开这两个键[89]。

5. 同时按下**[ON]**和**[S]**。屏幕会显示 `SAM-BA? boot`。仅松开**[S]**并再次按下以确认，然后松开这两个键。这将重启 Samba boot 位并关闭 *WP 34S*。

6. 按下**[ON]**。计算器将没有任何反应，不用担心，这不是死机。

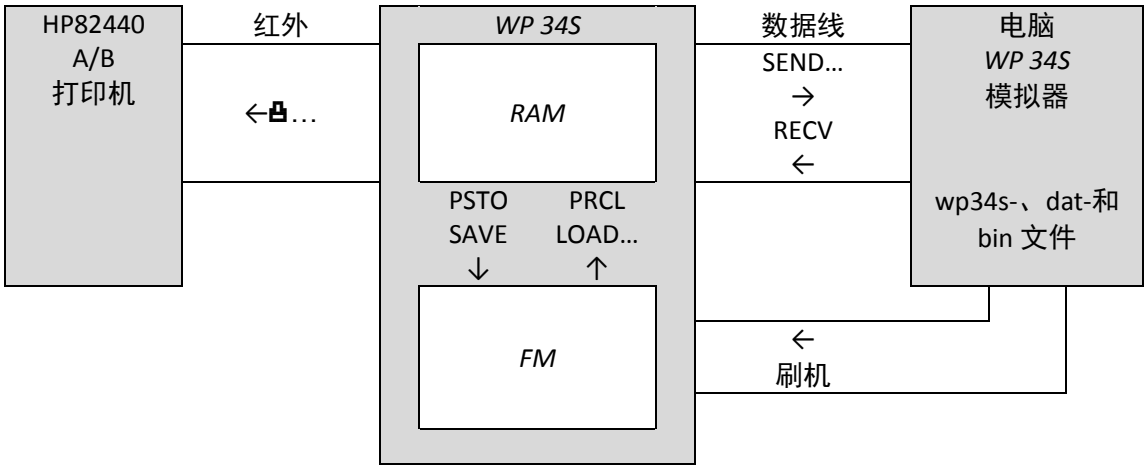
7. 点击 *MySamBa* 窗口的"Send File"等待数据传输结束（大约需要 20 秒）[90]。

8. 用合适的针状物捅一下 *WP 34S* 背部的 RESET 孔。计算器将会关机。退出 *MySamBa*。解除连接，关闭电池仓盖。

9. 按下**[ON]**。*WP 34S* 应该开机，会显示 `Restored`，表明恢复了最近一次的备份且装好了最新的固件。用 VERS 指令检查版本。

I/O 概述

WP 34S 输入输出结构框图如下：



SAVE 用来内部备份，以防止因电池失效丢失数据，参见 76 页和 144 页的描述。LOAD 用来从备份中恢复。除了用 SAVE 和 LOAD 这两个指令，还可以用以下操作来代替：按着**[ON]**，并将以下按键之一按两次：

[STO]用来备份：和 SAVE 一样，在 FM 中创建一个 RAM 的拷贝。

[RCL]用来恢复：和 LOAD 一样，恢复最近一次的备份。

PRCL 将一个单独的程序从 FM 程序库中导入到 RAM 中，PSTO 将一个单独程序的拷贝从 *RAM* 存放到 *FM* 程序库中。请注意，您可以用 PRCL 将一个程序库的程序从 *FM* 导入、编辑然后用 PSTO 回存的方式，修改这一部分的 *FM* 程序库。

[89]下一个步骤仅在 Debug 模式下工作，见附录 H。

[90]如果 *MySamBa* 刷机失效，按下 RESET 并解除编程数据线的连接，花点时间看看脚注 88。如果想重试，请回到步骤 6。

关于这两个指令可以参见操作索引，*WP 34S* 的 *RAM* 和 *FM* 部分的详细介绍见附录 B。

SEND 指令将数据从 *WP 34S* 的 *RAM* 传到电脑上，便于编辑。详见附录 D。

打印指令如果有效，将通过红外通道传输。

更换电池

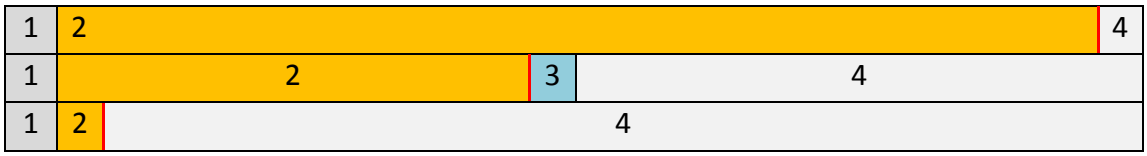
在指示符 ▬ 亮起时（见第 36 页），请更换新电池（CR2032）。请先用 SAVE 保存数据（保存成功将显示 `Saved`）。然后关闭 *WP 34S*，打开电池盖，卸下旧电池，插入新电池（参见第 178 页的图片），盖上电池盖并打开 *WP 34S*。它将在启动时显示 `Restored`，这表明所有保存的数据都已自动调用，可以在停止工作的地方继续工作。

附录 B 内存管理

WP 34S 拥有 6*kB* 的 *RAM*（其中 4*kb* 是易失的，且用户不可访问）和 128*kB* 的 *FM*。固件占据了 *FM* 的大约 90%，具体大小取决于刷入时加载的文件。*FM* 剩余的用户可访问的部分有将近 14*kB*。这之中，2*kB* 被保留作为 *RAM* 中 2*kB* 非易失存储的备份区域；其余部分则可能会被一个各种程序的集合（也叫程序库）填满。

这篇附录的以下部分只讲述 *RAM*。它讨论了可用内存是如何被划分为程序区域，本地数据和全局数据的。这 2*kB*（1024 字）的非易失内存[91]由以下 4 个部分共用：

1. 状态和配置数据；
2. 全局寄存器，也就是通用寄存器和堆栈；
3. 用于累积统计数据的寄存器（可选）；
4. 程序内存，包括子程序返回堆栈（SRS，也包括本地寄存器（若有的话））。



像上图中按比例展示的那样，这 4 个区域是从左到右排布的。中间的横条展示了初始默认 *RAM* 配置（有已分配的统计数据寄存器）。上面的横条展示了所允许的最大寄存器空间（没有统计数据寄存器），下面的横条展示了最大的程序空间。本附录对这些区域边界的各种变化都适用[92]。

状态和配置数据

这个 84 字节的部分被固定在可用存储的最顶端并且对用户是完全透明的。它包括状态和模式数据，30 字节的 alpha 寄存器，和存储了 112 个全局用户标志位的 14 字节空间。

[91]译者注：此处指关机不丢失数据，断电数据仍会丢失。

[92]在配置和加载 *RAM* 时，可以用 SAVE 将其完整副本写入 *FM*。它不仅在 *WP 34S* 关闭时，而且在电池出现故障时，也会安全保存。有关 *FM* 写入和处理的详细信息，请参阅附录 A。

全局寄存器

全局寄存器位于靠近可用内存的末端处。在启用时的默认内存布局中，数字寄存器 **R00** 到 **R99** 位于 12 个堆栈和特殊寄存器 **X**，**Y**，**Z**，**T**，**A**，**B**，**C**，**D**，**L**，**I**，**J** 和 **K** 之前（见 24 页）。这样加起来就是 112 个全局寄存器，即最大的可用寄存器数目。这个数字可以用 REGS 指令减小到仅 12 个字母寄存器（见 *IOP*）。REGS?会返回当前已分配的全局数字寄存器数，它是一个 0 到 100 之间的整数。

REGS 控制着全局寄存器部分的下边界（下文简称为 LBG）。减少寄存器的数量会将 LBG 升到绝对地址的更高处；增加数量则会将其降到更低处。内存中的内容也相应移动以保持存已有寄存器中的数据。字母寄存器则不移动。

示例：注意以下初始默认内存表中的全局寄存器部分。右边的两栏中展示了依次执行 REGS 96 和 REGS 98 后的结果。LBG 用红线标出。

	启用时的默认内存分配		执行 REGS 96 之后		接着执行 REGS 98 之后	
绝对地址	内容	寄存器相对地址	内容	寄存器相对地址	内容	寄存器相对地址
X+11	***k=40.7***	**R111=K**	***40.7***	**K=R111**	***40.7***	**K=R111**
…	**…**	**…**	**…**	**…**	**…**	**…**
X+2	***z=4.5***	**R102=Z**	***4.5***	**Z=R102**	***4.5***	**Z=R102**
X+1	***y=-33.8***	**R101=Y**	***-33.8***	**Y=R101**	***-33.8***	**Y=R101**
X	***x=123.0***	**R100=X**	***123.0***	**X=R100**	***123.0***	**X=R100**
X-1	***r99=-13.6***	**R99**	***23.1***	**R95**	***0.0***	**R97**
X-2	***r98=67.9***	**R98**	***6.4***	**R94**	***0.0***	**R96**
X-3	***r97=-45.2***	**R97**	***4.8***	**R93**	***23.1***	**R95**
X-4	***r96=9.7***	**R96**	**…**	**…**	***6.4***	**R94**
X-5	***r95=23.1***	**R95**	**…**	**…**	***4.8***	**R93**
X-6	***r94=6.4***	**R94**	**…**	**…**	**…**	**…**
X-7	***r93=4.8***	**R93**	**…**	**…**	**…**	**…**
…	**…**	**…**	**…**	**…**	**…**	**…**
X-94	***r06=62.4***	**R06**	***5.7***	**R02**	***29.4***	**R04**
X-95	***r05=-0.6***	**R05**	***-2.4***	**R01**	***81.3***	**R03**
X-96	***r04=29.4***	**R04**	***1.1***	**R00**	***5.7***	**R02**
X-97	***r03=81.3***	**R03**			***-2.4***	**R01**
X-98	***r02=5.7***	**R02**			***1.1***	**R00**
X-99	***r01=-2.4***	**R01**				
X-100	***r00=1.1***	**R00**				
…						

注意在 ***n*** 改变时，**R00** 到 **R*n-1*** 的绝对地址随 REGS ***n*** 的执行而改变，它们的内容也被复制到新的地址。字母寄存器停留在固定的绝对地址处。

在间接寻址中，寄存器中的 0 总是指向 **R00**。如果地址值大于 REGS 设定的

最大值则会抛出一个“out of range”错误，除非这个值落在 100 和 111 之间——这里是字母寄存器的位置。

在下方的内存中跟随着的两个区域（求和寄存器和 SRS）的位置是和 LBG 绑定的，只要 LBG 移动了，它们的内容都会被复制。这使得在一个子程序中可以执行 REGS 指令而不会打乱这个例程。

求和寄存器

累加统计数据所需的内存是单独分配的——这些数据不再使用全局通用寄存器存储。这使得内部精确度更高并防止了这些数据被意外破坏。更新这些统计数据的唯一方式就是通过 Σ+和 Σ-。累加数据由专门的指令计算和读取；不能用 STO 或 RCL 访问它们。

第一次调用 Σ+时会为这 14 个求和寄存器分配 70 字的空间[93]。这段空间是在 LBG 和 SRS 之间插入的，并且将后者在内存中后移。有时候，根据程序和数据对空间竞争的情况，可能需要提前预留空间（见下一页）。

在执行 CLΣ，CLALL 或 RESET 之后，为求和寄存器分配的内存将被释放。所有的指针都将被自动调整，因此内存分配和释放不会打断程序运行。如果数据没有被分配，那么像 Σxy 和 SUM 这样的指令会返回 0；其它的统计操作则会抛出错误。

注意：当一个长程序被加载（从 *FM* 或通过串口通信）后 *RAM* 容纳不下更多寄存器时，求和数据会被自动清除。可以在程序加载之前通过执行 REGS 指令减少数字寄存器的数目来避免这种情况。这样就可以将求和数据移动至可用空间，避免被删除。

子程序返回堆栈（SRS）和程序内存

二者共用内存地址最下方的剩余空间。

SRS 用于返回地址和本地数据的存储。它的上边界由 LBG 或者最后一个求和寄存器（若有）决定。没有设置 SRS 大小的指令——到所存储的程序的第一步之前，所有空间都由它占用。当新的程序步被输入时，SRS 被重置，这不仅仅是为了腾出空间，因为任何存储的地址数据在程序被修改后都可能变得无效。

本地数据是被压入到 SRS 中的。因此它们不会覆盖全局数据；这极大地增加了程序的灵活性。指令 LocR ***n*** 分配 ***n*** 个本地寄存器和固定的 16 个本地标志位。其实现方式是将一个包含着一个标记，一个标志位字，和所请求的寄存器（0 到

[93]这里，Σn 会占用 2 个字的空间，Σx^2，Σy^2，Σxy 和 Σx^2y 占用 4×8 个字的空间，其它的求和计算占用 9×4 个字的空间。如果分配这 70 字的内存失败，将会返回一个错误信息。

144）的数据帧压入 SRS。标记以字为单位记录了数据帧的大小，数值取决于精度模式设置（见附录 H）。内存中一个指向这个数据帧的指针将被初始化。如果这指针值为 0，表示不存在本地寄存器。新分配的寄存器将被清空。

在一个子程序中再次调用 LocR 将会调整本地寄存器的数量。这需要进行数据拷贝，因为这些寄存器将从低地址到高地址被分配，SRS 也同时反向调整大小。LocR?将返回当前例程中已分配的本地寄存器的数目。

后文有本地数据寻址的内容和一个递归程序调用的例子。但是，SRS 必须足够大以容纳这些数据，所以可能需要事先预留空间——参见下一段。

在 SRS 下方，程序内存保存着存储的程序步。一般地，一步程序只占用一个字。每个多字节标签和多字符字符串占用两个字。程序内存的总大小取决于已分配的全局和本地寄存器数目，详细解释见下文。

为需求创造空间

12 个特殊（字母）寄存器是永久性被分配的。SRS 的最小大小是 6 个字，或者 6 个堆栈。在区域 2 到区域 4 中剩下的 982 个字的空间是用户可分配的，如下所示：

982 = ***r*** + ***s*** + ***p***，其中

r = 分配给全局寄存器的字数。每个标准的寄存器占用 4 个字。全局寄存器的数量最少 12 个最多 112 个。所以 ***r*** 的值介于 48 和 896（对这个最大值的解释在附录 H 中）之间；初始默认值是 448。

s = 分配给求和寄存器的字数（若有则为 70；初始默认值是 0）。

p = 程序步和 SRS 可用的空间，以字计。其中有一步已经被最后的 END 声明占用；SRS 最小 6 个字。所以 STSATUS 会表示 *RAM* 中空余字的最大值是 933，代表着最大 927 个空余程序步数。初始默认值是 533 步。子程序嵌套和本地寄存器会扩展 SRS，这样就会减少可用的程序空间。

举个例子，如果需要做统计运算并且使用了 20 个全局数字寄存器，那么 *RAM* 中还剩余最大可容纳 777 程序步的空间。

当需要增加空余空间时，有以下几个选项（参见 147 页的图）：

1. 减少已分配的全局数字寄存器的数目。一般地，减少一个全局寄存器可以增加 4 个程序步。

2. 将程序移到 *FM* 并清除 *RAM* 中相应的程序步。一般地，清除 4 个程序步可以增加一个寄存器。

3. 在不需要求和寄存器的时候将它们释放。腾出的空间可以容纳最多 70 个程序步，或 17 个寄存器，或二者的组合。

哪种解决方案最适合您，取决于您的程序。当然，可以组合以上选项。使用 **[STATUS]**查看可用空间以及分配的全局和本地寄存器数量。

本地数据和子程序

正如在 148 页所说的那样，全局数据占用 0 到 111 的相对地址。所以，本地数据的相对地址从 112 开始，最大到 255——如果分配了 144 个本地寄存器的话。最前面的 16 个本地寄存器和所有的本地标志位也可用点加数字的形式直接寻址——从.00 到.15，对应相对地址的 112 到 127[94]。超过这个界限的任何寄存器都只能间接寻址。间接寻址使用以下模式：

- 通过一个全局指示寄存器寻址一个全局寄存器（例：STO→23，并且 ***r23***<112），
- 通过一个本地指示寄存器寻址一个全局寄存器（例：STO→.15，并且 ***r.15***<112），
- 通过一个全局指示寄存器寻址一个本地寄存器（例：STO→47，并且 ***r47***≥112），
- 通过一个本地指示寄存器寻址一个本地寄存器（例：STO→.06，并且 ***r.06***≥112）。

子程序调用：XEQ（在例程中执行时）在它转移到目标程序之前仅仅把返回地址压入 SRS 中。只要被调用的子程序本身还没有执行 LocR，它将一直能够访问调用方的本地数据。而一旦它执行 LocR，指向本地数据的指针就会被重新设置，这之后被调用的子程序就再也不能访问调用方的本地数据了。

RTN 或 POPLR（在程序中间执行时）会检查当前的 SRS 指针是否指向一个本地数据帧（像在 149 页中解释的那样）。若是，则指针被移到该数据帧之前，然后从该位置向前搜索 SRS 以寻找另一个本地数据帧。如果这样的数据帧被找到，就存储它的指针；否则指向当前激活的本地数据帧的指针将被清除。RTN 将会跳转到所找到的返回地址，而 POPLR 则继续执行程序。这样当前的本地数据帧即被丢弃，下一个更前（或更早）的数据帧将重新激活（若有的话）。

手动执行 RTN，重新开始一个包含 XEQ，SLV 等指令的新程序或编辑程序将以清除指针的方式清除 SRS 并移除全部本地寄存器和标志位。所有这些数据都会丢失！

递归程序：本地寄存器的使用允许创建一个调用自身的子程序。每一次调用都只关系到该次调用的本地数据。当然，RPN 堆栈是全局的，所以小心不要在递归程序中弄乱它。

[94]在操作码中，只有小于等于 127 的参数是可存储的，这就是这个界限的由来。

下面是阶乘函数的递归示例。这仅仅是一个用于演示的例子，因为这个程序既不会正确的设置堆栈也不能在输入值大于几百时正常工作：

```
LBL 'FAC'
IP
x≥1?
GTO 00
1
RTN
LBL 00
LocR 001
STO .00
DEC X
XEQ 'FAC'
RCL× .00
RTN
```

假设当调用 FAC 时 ***x***=4。这时它会分配一个本地寄存器（**R.00**）并在其中存储 4。在 ***x*** 减少后，FAC 会调用自身。

之后 FAC_2 会分配一个本地寄存器（$\mathbf{R.00}_2$）并在其中存储 3。在 ***x*** 减少后 FAC_2 会再一次地调用自身。

之后 FAC_3 会分配一个本地寄存器（$\mathbf{R.00}_3$）并在其中存储 2。在 ***x*** 减少后 FAC_3 会三度地调用自身。

之后 FAC_4 会返回到 FAC_3，此时 ***x***=2。这个 ***x*** 会被乘以 FAC_3 中的 ***r.003*** 并返回到 FAC_2，此时 ***x***=2。这个 ***x*** 会被乘以 FAC_2 中的 $\boldsymbol{r.00}_2$ 并返回到 FAC，此时 ***x***=6。之后 ***x*** 会被乘以 FAC 中的 ***r.00*** 并最终等于 24。

标准实数（SP）模式和整数模式的切换

WP 34S 初始默认处于标准实数模式（DECM）。当然可以使用它进行整数运算——就像上文中多次展示的那样。当从 DECM 切换到任何整数模式时，当前堆栈中的值都会被截断成整数。从整数模式切换到 DECM 时，当前的堆栈内容（全都是整数）也都会被换算成带小数的数。所有其它的内存内容都会保持原样！

看下面例子中经历模式切换的一些寄存器内容的变化，其中将会用访问指令检查 ***j***，***k***，***r00*** 和 ***r01*** 的值：

	X	Y	J	K	R00	R01
初始值，例如	1.1	20.2	300.3	4,00.4	50,0.5	600,00.6
执行 2COMP，WSIZE 32，BASE 10 之后	1 d	20 d	3,07.5 d	4,09.64 d	5 12,00.5 d	6,9 1,46.2 d
用 sRCL 访问寄存器			300 d	4,000 d	50,000	600,000 d
567 STO J，-9 STO 00	-9 d	56.7 d	56.7 d	4,09.64 d	-9 d	6,9 1,46.2 d
DECM	-9.0	56.7.0	5.7^{-396}	4,00.4	3.0^{-389}	600,00.6
用 iRCL 访问寄存器			56.7.0	4,0.4.0	-9.0	6,9 1.2.0

注意在 DECM 和整数模式下，相同的寄存器内容会不同处理，即使是很小的整数也可能变成巨大的数：

	R00	R01	R02
初始值，例如	0	2	10
执行 WSIZE 64，BASE 16 之后	0 h	2238000000000002 h	223C00000000001 h
用 sRCL 访问寄存器	0 h	2 h	A h
2 STO 01，A STO 02	0 h	2 h	A h
DECM	0	2^{-398}	1^{-397}
用 iRCL 访问寄存器	0	2	10

所以使用间接寻址时要小心！

示例：设置 DECM，WSIZE 64，2COMP，0 STO 00，2 STO 01，10 STO 02。这时 RCL→01 应该访问的是 ***r02***。让我们检查一下。键入：

[RCL][⇨][0][1]	10.	仅仅是验证一下：OK
[g][16]	8 h	转换到 16 进制
[RCL][⇨][0][1]	Out of range Error	与上表对比
[2][STO][0][1]	2 h	
[RCL][⇨][0][1]	0000000000001 h	现在它在 16 进制模式下也能工作了
[f][◀()]	223C h	
[f][H.d]	2,46684669589 18	转换到 DECM
[RCL][⇨][0][1]	0.	与上表对比——重定向需要忽略小数部分，否则 ISG/DSE 循环难以工作。

这一切都是由数字的内部表示导致的：整数简单地如字面一般存储（UNSIGN 模式下 $n \le 1.84\times10^{19}$ 或者其它模式下 $|n| \le 9.22\times10^{18}$，占用 64 比特），而标准浮点数用以下格式存储：

- 实数 0 和整数 0 一样存储，也就是说所有比特位都为空。
- 一个实数的有效数字部分（即有效位）分为 5 组三位数进行编码。每一个这样的分组都直接用 10 个比特位（二进制位）表示，比如说 $555_{10}=1000101100_2$，$999_{10}=111100111_2=3E7_{16}$。所以有效数字最右边的 15 个十进制位占用 50 个最低有效位。尾部的 0 会被省略，所以有效数字是右移过的。
- 最高有效位（第 64 位）存储有效数字的符号。
- 剩余的 13 个比特位被用于存储指数部分和有效数字最左边的那位数。这 13 个比特位中，最低的 8 位用于存储指数部分。前面的 5 位则有些复杂[95]：如果它们是…
  - `00ttt`，`01ttt` 或 `10ttt`，那么 `ttt` 存储有效数字最左边的数位（$0–7_{10}$），而最前面的两位会成为指数部分中的最高有效位。
  - `11uut`，那么把 `t` 加上 1000_2，所得的结果（8_{10} 或 9_{10}）就是有效数字最左边的数位。如果 `uu` 是 00，01 或 10，那么这两位会成为指数部分中的最高有效位。而如果 `uu` 是 11，则表示这是编码特殊数字的代码（例如无穷）。

总的来说，我们可以有 16 个十进制位有效数字和略少于 10 个比特位的指数：指数的最大值为 1011111111_2（即 767_{10}）。这个值必须减去 398 以得到所表示数的真正的指数，原因在下文中说明。有效数字的 16 个数位则允许了一个从 1 到几乎 10^{16} 的范围。

作为对您耐心看到这里的回报，我们展示一些关于 *WP 34S* 编码的演示性的例子，而不是灌输给更多理论：

浮点数	存储的 16 进制值	10 个一组的尾部比特位	二进制的前 14 位	存储的指数
1.	2238000000000001		00100010001110	398
-1.	A238000000000001		10100010001110	398
111.	223800000000006F		00100010001110	398
111.111	222C00000001bC6F	06F 06F	00100010001011	395
-123.000123	A220000007b0007b	07b 000 07b	10100010001000	392
$9.99\cdot10^{99}$	23bC0000000003E7		00100011101111	495
$1\cdot10^{-99}$	20AC000000000001		00100000101011	299
$1\cdot10^{-383}$	003C000000000001		00000000001111	15
0.000000 00001·10^{-383}	0004000000000001		00000000000100	4

上表中的最后一个数是可以直接输入的最小的数。用 10^4 除它会得到 1×10^{-398}，存储为 16 进制下的 1。在默认的舍入模式下（以及 RM 1，2，3 和 5。见 RM）用 1.99999999999 去除它，所得的结果仍为 1×10^{-398}。用 2 去除它则结果会变为零。

看看数字上边界的情况：

9.999 999 999 99·10^{384}	77FFE7F9FE7F7800	9 3E7 3E7 3E7 3dE 000	01110111111111	767

这个数（12 个 9）是可以被直接输入的最大数。把它加上 9.999×10^{372} 则会显示 1×10^{385}…

$1\cdot10^{385}$	77FFE7F9FE7F9FE7	9 3E7 3E7 3E7 3E7 3E7	01110111111111	767

…这个数被存储为 $9.999999999999999\times10^{384}$。这是计算器可以表示的最大的数。这一切都遵循 Decimal64 浮点数格式（虽然不完全）。

另外，*WP 34S* 有这些特殊数：

Infinity	7800000000000000		01111000000000	n/a
-Infinity	F800000000000000		11111000000000	n/a
Not a Number	7C00000000000000		01111100000000	n/a

如果标志位 **D** 被置位，这些特殊数在 *WP 34S* 上是合法的结果。

[95]不要怪我们——这一部分遵循 IEEE 754 标准。

附录 C 消息和错误代码

有些指令会在屏幕上的数字或点阵部分生成消息。这些如 DAY，DAYS+，ERR，STATUS，VERS 以及 WDAY 已经在上述屏幕显示的部分被介绍了。其它的还有 PROMPT，αVIEW 等更多 α 指令，还有判断指令（见 71 页）。还有 2 个常数在调用时将会反馈一个特殊信息。

此外，还有更多错误信息。取决于错误状况，下面列出了这些信息及对应的模式：

消息	错误代码	模式	解释以及举例
Bad time or dAtE	2	DECM	错误的日期格式或者数值，例如：月份＞12，日期＞31。
Bad digit Error	9	Integer	输入的整数无效，例如二进制模式的 2 或八进制模式的 9。只要无效的键被按下，就会显示此信息。
Bad mode Error	13	All	该信息是因为调用该模式下未定义的操作引起的，例如在整数模式下调用在 DECM 模式编写的程序中的常量。
Domain Error	1	¬α	参数超出了所调用的数学函数的定义域。可能是负数的根或 x≤0 的对数引起的（两者如果没有先使用**[CPX]**的话），以及 0/0，Γ(0)，tan(90°)及类似的操作，$\|\mathrm{Re}(x)\|\geq 1$ 的 artanh(x)，$\mathrm{Re}(x)<1$ 的 arcosh(x)等[96]。
Flash is FuLL	23	All	闪存中没有空间了，可以删除一个在闪存中的程序来获得空间。
Illegal OPErAti on	7	All	本信息可能在尝试运行老旧程序的时候出现。该程序可能包含了一个在这个程序编写后，不再可用于编程的指令。
Invalid dAtA	18	All	闪存中或串行下载数据出现校验错误时出现。如果 *FM* 因为其他原因不可用，也会出现。
Invalid PArAMEtEr	16	¬α	与错误 1 类似，但为 **J** 或 **K** 中指定的参数超出了被调用函数的有效范围。例如，在 $j<0$ 时调用 LgNrm，
I/O Error	17	¬α	见附录 D。
Matrix MUSt SQUArE	21	Integer	・在一个矩阵应是一个方阵时其非方阵。 ・矩阵维数不匹配。
No crystal MUSt SQUArE	24	All	可能会因 DATE，TIME，SETDAT 或 SETTIM 抛出，解决方法见附录 H。

[96]请注意，如果设置了标志位 D，则 tan(90°)和 0 的对数是合法的。请参见本附录的末尾。

消息	错误代码	模式	解释以及举例
No root Found	20	DECM	求解没有收敛。
No such LAbEL	6	All	尝试访问未定义的标签。
out of range Error	8	All	・数字超出有效范围。这可能是由于指定小数部分位数>11，字长>64，负标志位，整数≥264，小时或角度> 9000，无效时间，分母≥9999 等. ・寄存器或标志地址超出当前分配寄存器的有效范围。也可能发生在间接寻址或调用不存在的本地地址时。 ・R 操作（例如 R-COPY）尝试访问无效的寄存器地址。 ・矩阵描述符超出了可用的寄存器范围或行或列太大。
RAM is FULL	11	All	运行内存不足，也许因尝试编写过大的程序、分配太多寄存器等导致。也可能由于动态分配了太多本地数据，在程序执行中触发。（见 150 页）。
Singular Error	22	DECM	・尝试使用 LU 分解矩阵来求解方程组。 ・当矩阵不是满秩时尝试求逆。
Stack CLASH	12	All	STOS 或 RCLS 尝试使用与堆栈重叠的寄存器。例如 SSIZE = 8 然后 STOS 94。（如果设置了 REGS 100）。
Too few dAtA PoInts	15	DECM	基于太少的数据点开始统计计算，例如，回归或标准偏差小于两个数据点。
Too long Error	10	All	输入内容太长，例如输入多于 12 位的数字。
Undefined OP-CodE	3	All	出现了含有未定义操作代码的指令。这个是本不应该发生的——但谁知道呢？
Word Siz too SMALL	14	Integer ¬STO	寄存器内容对于字长设置来说太大。
Write ProtEctEd	19	All	・尝试在 FM 中修改或删除程序步。 ・尝试删除程序最后一步的 END。
+∞ Error	4	¬ α , ¬STO, ¬flag D	・将>0 的数字除以零。 ・发散的求和，或乘积，或积分。 ・DECM 中的正（或负）溢出（参见第 41 页）。

消息	错误代码	模式	解释以及举例
	5		・将<0 的数字除以零。 ・发散的求和，或乘积，或积分。 ・DECM 中的正（或负）溢出（参见第 41 页）。 ・(+)0 的对数（-0 的对数返回 NaN）。
	25	DECM 与 RCL	这不是错误，只是一条消息。它由 MSG 25 返回，并专门用于程序。

每条错误消息都是临时的（参见第 38 页），因此**[⇦]**或**[EXIT]**将删除它并允许继续操作。按下任何其他键也会将其删除，但会对堆栈内容执行对应的操作。因此，一个简单且安全的返回发生错误之前状态的方法，是按前缀**[h]**两次。

再最后补充关于标记位 **D** 的一些信息：如果该标记位被置位，错误信息 4 和 5（±∞错误）就完全不会出现，错误信息 1（超出定义域）会出现的少一些。造成这种现象的原因是±∞和 NaN 现在被认为是合法的运算结果（更多信息可参见常数（CONST）部分和附录 B 的最末尾）。

附录 D 电脑上的 *WP 34S* 模拟器

在 http://sourceforge.net/projects/wp34s/files/emulator/可以找到 *WP 34S* 模拟器。支持的系统有 Windows、MacOS、Linux 和 Linux64。请根据所用的操作系统下载和使用对应的模拟器。在 Windows 下还可以使用更早的模拟器：*http://sourceforge.net/p/wp34s/code/HEAD/tree/trunk/windows/bin/wp34sgui.exe* 。它和 *WP 34S* 计算器功能一致。虽然与实体计算器相比在触觉反馈，启动速度，便携性和电池寿命这几个方面有所不足，但在有些事情（例如打印（见下文））的操作上却更方便。

通常，模拟器可直接由鼠标操作。单击键盘图案的相应区域而不是按下按键。右键单击按键区是其绿色 h 标签的快捷方式。还有一个重要的额外功能：单击右上角的 WP 标志会直接打开一个菜单，如图所示。

它包括了项目网站的链接和使用手册（包含在 help 中）以及三种可供选择的模拟器皮肤。

“中等”皮肤已经在本手册中多次展示过了；另外两个皮肤较小，显示在下页。这两个皮肤在下方按比例打印显示。

这个菜单还允许输入输出数字，将 ***alpha*** 输出为文本，或者整个 LCD 作为图片复制到剪贴板上，以便其他电脑软件使用。

如果计算机有一个数字键盘，则数字和算术操作键会有模拟器上相应按键的快捷键。上下方向键映射**[▲]**和**[▼]**键，退格和 Del 映射**[⇦]**键，回车和 CR 映射**[ENTER⇳]**键。

WP 34S 与电脑之间的数据传输

模拟器和计算器可以通过数据线相互通信。第 139 页中提到的任何一根数据线都可以，或者需要一个如下所述的修改过的计算器：*http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv020.cgi?read=186826*。

请记住，编程数据线是由 *WP 34S* 的电池供电的，因此如果没有传输需求，请断开与 *WP 34S* 的连接。

在模拟器目录中，必须放置一个纯 ASCII 文本文件 `wp34s.ini`，它只包含一行（由 CR / LF 终止），注明了用于串行通信的计算机端口的名称，例如：

COM2:

以下指令（均存储在 P.FCN 目录中）允许将程序，寄存器或所有 *RAM* 从 *WP 34S* 发送到计算机，反之亦然。使用合适的电缆还可以在两个 *WP 34S* 之间直接传输数据。

在接收数据的机器上，输入 RECV.会显示 `Wait…`

在发送数据的机器上有三个发送方式：

1：SENDP 会从内存中发送现有程序，也会显示 `Wait…`，成功之后发送机会显示 `OK`，接收机显示 `Register`

2：SENDR 会发送所有已分配的全局编号寄存器，显示 `Wait…`，成功之后发送机会显示 `OK`，接收机显示 `Register`

3：SENDA 会将发送完整的两千字节的非易失性 *RAM*，也会显示 **Wait…**，成功之后发送机会显示 **OK**，接收机显示 **All RAM**

如果 *WP 34S* 从计算机模拟器接收数据，则这些数据将直接存储在相应的 *RAM* 部分中。如果模拟器从 *WP 34S* 接收数据，这些数据同样可以立即使用，并且将成为退出模拟器时写入的文件 `wp34s.dat` 的一部分（参见下页）。

串口正在使用时，屏幕上小的**=**指示符会亮起。如有必要，请按**[EXIT]**中止通讯。

四个命令 RECV，SENDP，SENDR 和 SENDA 中的每一个都具有大约 10 秒的固定延时，用于建立连接。在这个时间内没有传输活动，就会抛出 **I/O Error** 错误信息，提示没有发生通信。如果在传输过程中出现此类错误，请再试一次。

使用未安装晶振的 *WP 34S*，由于其波特率偏差较大，可能也会收到 I/O 错误。要确定传输速度，请使用以下循环：

```
CLx
INC X
BANK 001
```

并在其连接到计算机后运行 30 秒。对于 v3.1，标称速度的预期结果约为 123000。在未安装晶振的 *WP 34S* 上，I/O 指令根据 **X** 中的校正因子百分比进行校正。将测量到的数字除以 123000 并将其保留在 **X** 中以用于 RECV 或 SEND。2%已经是一个显著的差异。校正因子接受 80 到 120 之间的整数值——所有其它值会被忽略。在模拟器或安装了晶振并将其初始化的 *WP 34S* 上，*x* 将在这些指令中被忽略。

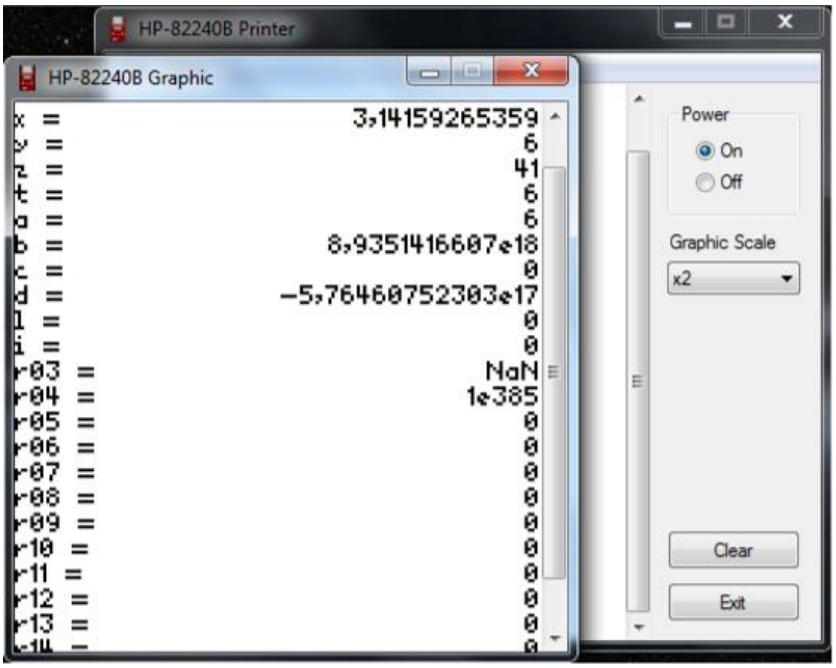
将数据从 *WP 34S* 成功传输到计算机后，请参阅单独的 *WP 34S* 汇编工具套件用户指南的第 4 部分，了解从文件 `wp34s.dat` 中提取 `.wp34s` 文件的方法。可以通过合适的文本文件编辑器打开和编辑 `.wp34s` 文件，例如，写字板。完成编辑和编译例程后（以及在模拟器上成功测试之后）可以恢复传输：在 *WP 34S* 上发出 RECV，并将相应的修改数据从计算机发送回 *WP 34S*。模拟器上的工作原理也适用于计算器（如果安装了必要的硬件）——两者都具有相同的功能。*WP 34S* 是市面上最小的能够进行双向数据交换和备份的计算器之一。

用于 Windows 和 MacOS 的较新的基于 Qt 的模拟器包含串行接口的设置选项。它们将最终取代当前的 Windows 模拟器。指令仍然如上面所述。

内存区域到模拟器状态文件的映射

区域		状态文件	备注
备份	bUP	wp34s-backup.dat	SAVE 创建的文件。
闪存	Lib	wp34s-lib.dat	执行闪存命令时被编写的文件。
内存	rAM	wp34s.dat	备份模拟器 RAM 区域（寄存器，状态和程序）——仅在退出模拟器时写入此文件。

在电脑上模拟打印机

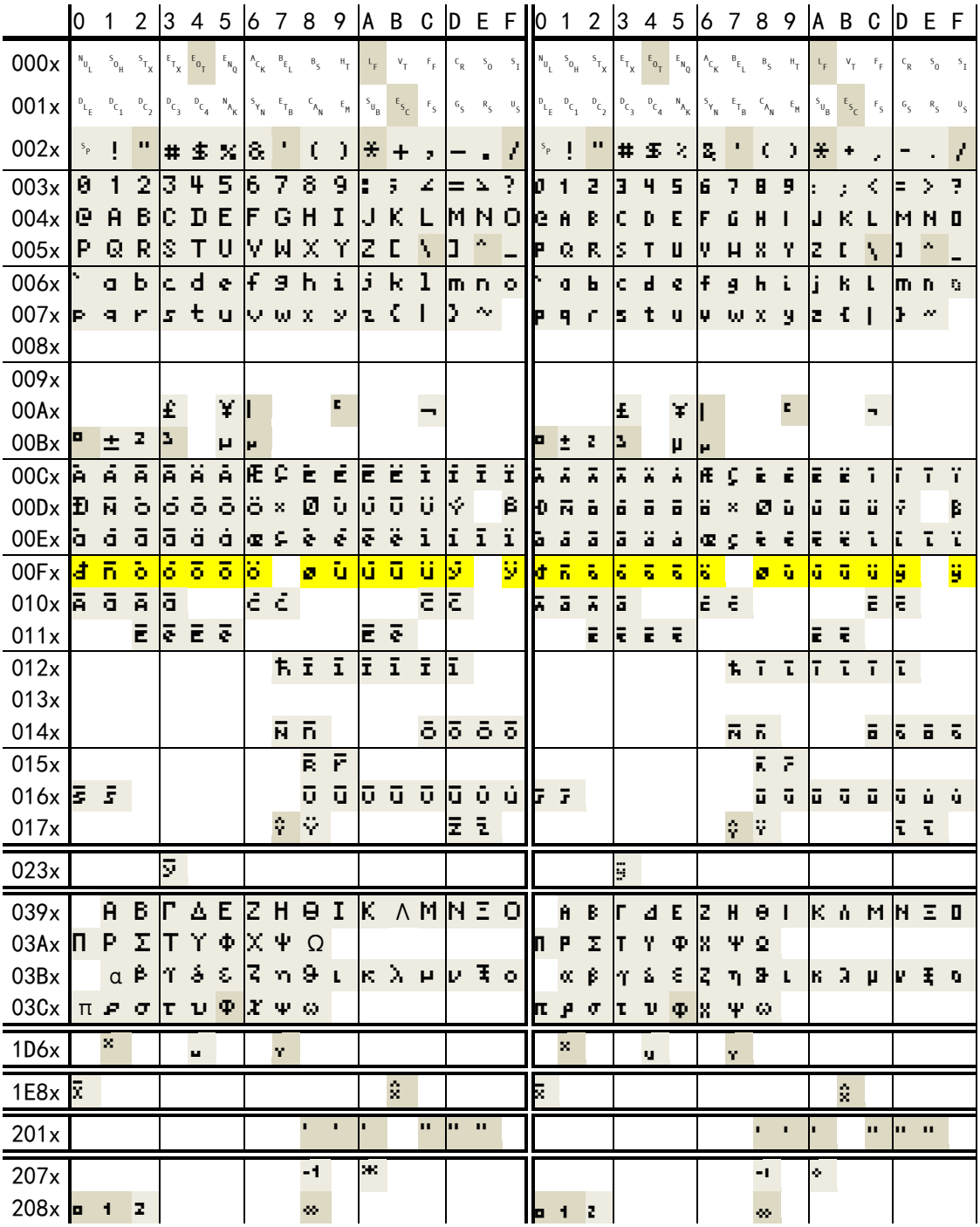


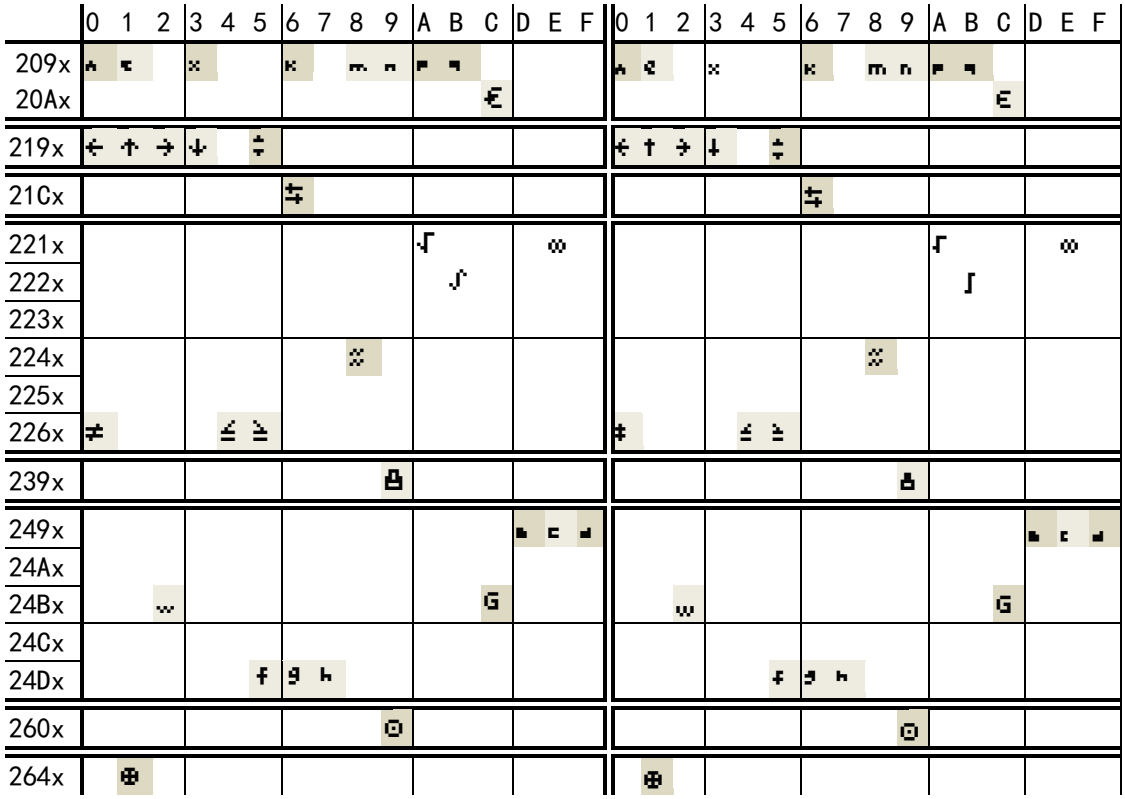
要模拟打印，请使用 *Christoph Gießelink* 的 HP82240B 打印机模拟器，可在这里下载：*http://hp.giesselink.com/hp82240b.htm*。

安装和使用不言自明。它是一个独立的应用程序，所以必须单独启动和退出。这里显示的打印图案是由🖷STK 和🖷REGS 生成的。猜猜设置了多大的堆栈大小？

附录 E 字符集

下表显示了根据十六进制字符代码（Unicode）排序的大小字体实现的完整点阵字符集。代码<20_{16} 的字符用于控制目的——其中一些（4，10_{10}，27_{10}）可能对于 HP82240B 控制有用。深灰色底色的字符在两种字体中显示相同的图案。00F0 到 00FF 的字符（此处为黄色底色）不能是三个一组字符串的最后一个字符（请参阅 LBL 等）。





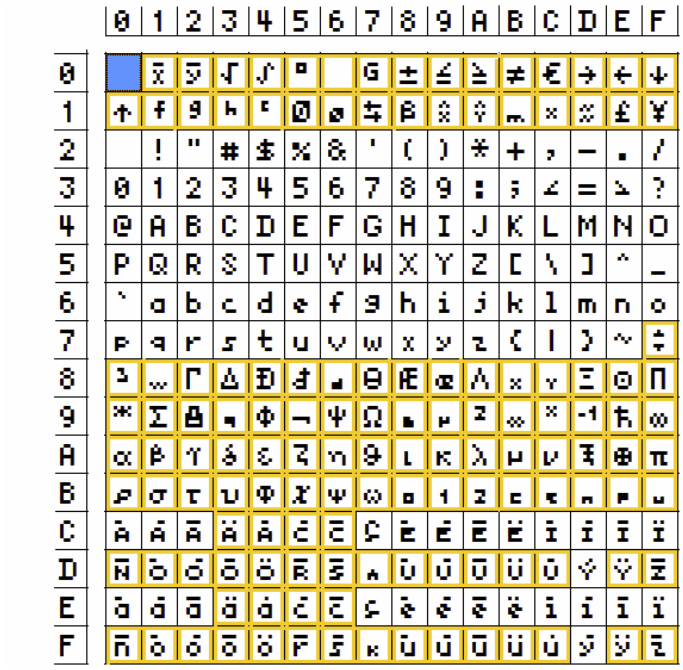
对于数字七段显示和右上角的指示符，提供了另一个字符集和字体：

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	附注
002x			"					'	(	)			,	-	.	'	
003x	0	1	2	3	4	5	6	7	8	9							
004x	°	A	b	C	d	E	F	G	H	I	J		L	M	N	O	注意 M
005x	P	Q	r	S	t	U		V		Y		C		]		_	注意 W
006x		A	b	c	d	E	F	G	h	i	J		L	m	n	o	注意 m
007x	p	q	r	s	t	u		J		y							注意 w
008x																	
009x																	
00Ax		-	-	=	_	¯	=	=	"								用于 STATUS 的指示
00Bx								.	0	'							BASE 2 中的高位码
00Cx																	
00Dx																	
00Ex	↓	INPUT	=	■	BEG	STO	RCL	RAD	360	RPN							状态指示符
00Fx		'	_	'	'	-	'	¯	.							8.	段码

全部字体可以在以下位置下载：

https://sourceforge.net/projects/wp34s/files/wp34s_Fonts.zip/download。

注意：x→α 和 α→x 指令使用 *WP 34S* 的内部字符代码。除了黄色框中字符外，这些代码与 Unicode 对应。字符 06 是一个窄空白。



点阵字体的字符在代码大于 E000 时按内部顺序重复。但是，请勿在文本中使用这些高位代码，因为某些程序（例如 *Preview*）可能无法正确读取它们。

附录 F 与 *HP-42S* 和 *HP-16C* 的功能对比

在 IOP 中，在 *WP 34S* 的相应条目下提到了老式 HP 计算器的相应功能。下表以某种方式对此进行了呼应：第一个表显示了 *HP-42S* 的功能以及 *WP 34S* 的相应功能，除非它们具有相同的名称，并且都是通过键盘访问或存储在菜单中。后面 172 页的表为 *HP-16C* 功能对比表。相同名称指令的差异已在 *IOP* 中说明。

HP-42S

以浅灰色为底色的备注表示指令在 *WP 34S* 上是默认设置或者键盘直接输入的，而 *HP-42S* 上必须使用菜单。

HP-42S	***WP 34S***	**备注**
ABS		按下[|x|]。
ACOSH		按下[HYP^{-1}][COS]。
ADV	ADV	
AGFAPH	n/a	*HP-20b* 和 *HP-30b* 的 LCD 显示屏只有一个非常小的点阵部分（见第 35 页）。想要在其中制作完整的图形将是吃力不讨好的。
AIP	αIP	
ALENG	αLENG	
All		按下[ALL]。
ALLΣ	多余的指令	*WP 34S* 始终在 ALLΣ 模式中运行。
[■][ALPHA]	[α]	请参阅第 62 页的 alpha 模式说明。
AND		按下[AND]。
AOFF	αOFF	
AON	αon	
ARCL	αRCL 或 αRC#	
AROT	αRL 或 αRR	
ASHF	αSL 或 αSR	
ASINH		按下[HYP^{-1}][SIN]。
[■][ASSIGN]	n/a	*HP-20b* 和 *HP-30b* 的显示屏显示不下软键菜单，因此我们无法提供 CUSTOM 菜单这样的功能。
ASTO	αSTO	在 alpha 模式中按下[f]**[STO]**。
ATANH		按下[HYP^{-1}][TAN]。
ATOX	α→X	
AVIEW	αVIEW	
[■][BASE]	多余的指令	HEXM，DECM，OCTM，BINM 以及逻辑操作 AND，OR，XOR 和 NOT，除 alpha 模式之外都在 *WP 34S* 的键盘上。如果适用，A 到 F 在整数模式中提供。对应于 BIT?和 ROTXY 的功能在 X.FCN 中。
BASE+ 等.	多余的指令	*WP3 4S* 会自动以整数模式执行这些算术命令。
BEEP	n/a	*HP-20b* 和 *HP-30b* 没有所需要的硬件。
BINM	BASE 2	按下[2]。
BIT?	BS?	

HP-42S	*WP 34S*	备注
[■][CATALOG]	被替换	在 *WP 34S* 中，分散在不同的菜单中。
CF		按下[CF]。
CLA	CLα	alpha 模式中按下[CLx]。
CLD	多余的指令	按下任意按键都会清除临时信息。
[■][CLEAR]	[P.FCN]	CLΣ，CLP，CLST，CLA 和 CLX 在键盘上。
CLKEYS	n/a	无 CUSTOM 菜单，因此不需要。
CLLCD	n/a	用[α][CLx]来清除点阵屏部分，参见 AGRAPH。
CLMENU	n/a	无 CUSTOM 菜单，因此不需要。
CLP		按下[CLP]。
CLRG	CLREGS	
CLST	CLSTK	按下**[0]**[FILL]。
CLV	n/a	*HP-20b* 和 *HP-30b* 的存储空间太小，难以实现类似 HP-42s 的变量功能。因此，*WP 34S* 上没有变量操作的指令（例如，许多矩阵指令）。
CLx		按下[CLx]。
CLΣ		按下[CLΣ]。
COMB		按下[Cy,x]。
[■][CONVERT]	n/a	*HP-20b* 和 *HP-30b* 存储空间太小，难以实现类似 HP-42s 的多种数据类型。因此，必须在 *WP 34S* 上将复数放入两个寄存器中，如 29 页所述。
[■][CONVERT]	[CONV]	
CORR		按下[r]。
COSH		按下[HYP][COS]。
CPXRES CPX?	n/a	参见 COMPLEX。
[■]CUSTOM	n/a	*HP-20b* 和 *HP-30b* 的显示屏显示不下软键菜单，因此我们无法提供 CUSTOM 菜单这样的功能。
DECM		按下[H.d]。
DEG		按下[DEG]。
DEL	n/a	未提供，在我们看来，这个指令太危险了。
DELAY	🖷DLAY	
DELR，DIM 和 DIM?	n/a	矩阵指令，见 CLV。
[■][DISP]	多余的指令	FIX，SCI，ENG，ALL，RDX.与 RDX,已在键盘上标识。
DSE		按下[DSE]。
[E]	**[EEX]**	
EDIT 和 EDITN	n/a	矩阵指令：参见 CLV。
ENG		按下[ENG]。
EXITALL	n/a	无 CUSTOM 菜单，因此不支持。
FCSTX	x̂	
FCSTY	ŷ	按下[ŷ]。
FIX		按下[FIX]。
[■][DISP]	[TEST]	SF 和 CF 已在键盘上标识。
FNRM	n/a	矩阵指令：参见 CLV。
FP		按下[FP]。
GAMMA	Γ	

HP-42S	*WP 34S*	备注
GETKEY	KEY?	
GETM	n/a	矩阵指令：参见 CLV。
GRAD		按下**[GRAD]**。
GRAD	n/a	矩阵指令：参见 CLV。
HEXM	BASE 16	按下**[16]**。
I+andI- INDEX	n/a	矩阵指令：参见 CLV。
INPUT	PROMPT	
INSR	n/a	矩阵指令：参见 CLV。
INTEG	∫	按下**[∫]**。
INVRT	M^{-1}	
IP		按下**[IP]**。
ISG		按下**[ISG]**。
J+andJ-	n/a	矩阵指令：参见 CLV。
KEYASN KEYG 和 KEYX	n/a	无 CUSTOM 菜单，因此不支持。
[■][LASTx]	RCL L	
LBL		按下**[LBL]**。
LCLBL	n/a	详见 ASSIGN，但是 *WP 34S* 依然提供本地标签，详见 68 页。
LINΣ	多余的指令	*WP 34S* 一直运行在 ALLΣ 模式。
LIST	n/a	用🖷PROG 代替。
LOG	LOG_{10}	按下**[LG]**。
MAN	CF T	此打印模式是 *WP 34S* 的初始默认设置。
[■][DISP] MAT?	n/a	矩阵命令：参见 CLV。
MEAN	$\bar{x}$	按下**[$\bar{x}$]**。
MENU	n/a	无 CUSTOM 菜单，因此不支持。
[■][MODES]	**[MODE]**	
MVAR	n/a	无 CUSTOM 菜单，因此不支持。
N!	x!	按下**[!]**。
NEWMAT	n/a	矩阵命令：参见 CLV。
NORM	n/a	不支持
NOT		按下**[NOT]**。
OCTM	BASE 8	按下**[8]**。
OLD	n/a	矩阵命令：参见 CLV。
ON	n/a	未定义其他功能。
OR		按下**[OR]**。
PERM		按下**[Py, x]**。
[■][PGM.FCN]	**[P.FCN]**	LBL，RTN，VIEW，PSE，ISG 和 DSE 不在键盘上。
PGMINT	包含在∫中	
PGMSLV	包含在 SLV 中	
PI	π	
PIXEL	gSET	参见 AGRAPH。
POLAR	n/a	参见 COMPLEX。
POSA	n/a	alpha 寄存器最多包含 30 个字符。我们认为，使用自动搜索会显得有些浪费。

HP-42S	*WP 34S*	备注
PRA	🖶α	
PRLCD	🖶PLOT	参见 AGRAPH。
[■][PRGM]	[P/R]	
[■][PRINT]	[P.FCN]	PRX 在可以在键盘上找到。
[■][PROB]	被取消	COMB，PERM，N!和 RAN 可以在键盘上找到，SEED 在 STAT 中，GAM(MA)在 X.FCN 中。
PROFF	CF T	按下[CF][T]。
PRON	SF T	按下[SF][T]
PRP	🖶PROG	
PRSTK	🖶STK	
PRUSR	n/a	参见 CLV。
PRV		
PRX	🖶r X	在 alpha 模式外，按下[⇧]。
PRΣ	🖶Σ	
PSE		按下[PSE]。
PUTM	n/a	矩阵命令：参见 CLV。
QUIET	n/a	参见 BEEP。
RAD		按下[RAD]。
RAN	RAN#	按下[RAN#]。
RCLEL 和 RCLIJ	n/a	矩阵命令：参见 CLV。
RDX,		在 alpha 模式外，按下[./,]。
RDX.		
REALRES	n/a	参见 COMPLEX。
REAL?		
RECT	多余的指令	此模式在 *WP 34S* 上是永久性的。参见 COMPLEX。
RND	ROUND	按下[RND]。
RNRM	n/a	矩阵命令：参见 CLV。
ROTXY	RL，RCL，RR 和 RRC	
RSUM	n/a	矩阵命令：参见 CLV。
R<>R		
SDEV	s	按下[s]。
SF		按下[SF]。
[■][SHOW]	[◀()]	
SINH		按下[HYP][SIN]。
SIZE	REGS	
SLOPE	L.R.	
SOLVE	SLV	按下[SLV]。
[■][SOLVER]	包含在 SLV 中	
SQRT	√	
[■][STAT]	[STAT]	MEAN，SDEV，FCSTY 和 CORR 可以在键盘上找到，MODL 的设置在 MODE 菜单中。
STOEL 和 STOIJ	n/a	矩阵命令：参见 CLV。
STR?	n/a	参见 CLV。
TANH		按下[HYP][TAN]。

HP-42S	*WP 34S*	备注
TONE	n/a	参见 BEEP。
[■][TOP.FCN]	n/a	无 CUSTOM 菜单，因此不支持。
TRACE	SF T	按下[SF][T]。
TRANS	TRANSP	
UVEC	^{C}SIGN	
VARMENU	n/a	参见 CLV。
VIEW		按下[VIEW]。
WMEAN	$\overline{x}_w$	
WRAP	n/a	矩阵命令：参见 CLV。
X<>		按下[x⇆]。
X<0?和 X<Y?	x< ?	
X≤0?和 X≤Y?	x≤ ?	
X=0?和 X=Y?	x= ?	按下[x= ?]。
X≠0?和 X≠Y?	x≠ ?	按下[x≠ ?]。
X≥0?和 X≥Y?	x≥ ?	
X>0?和 X>Y?	x> ?	
XOR		按下[XOR]。
XTOA	X→α	
YINT	L.R.	另一种方式是按下[0][ŷ]。
[■]∫f(x)	包含在∫中	
ΣREG 和 ΣREG?	多余的指令	参见附录 B。
→DEC	多余的指令	整数模式中转换自动完成。
→DEG	rad→°	按下[CONV][R][XEQ]。
→H.MS		按下[⇨][H.MS]。
→HR		按下[⇨][H.d]。
→OCT	多余的指令	整数模式中转换自动完成。
→POL		按下[⇨P]。
→RAD	°→rad	按下[CONV][A][▲][XEQ]。
→REC		按下[R⇦]。
%CH	Δ%	按下[Δ%]。

HP-16C

HP-16C 功能表按其按键排序，从左上角开始。和 *HP-42S* 一样，只列出了两个计算器上带有不同名称的功能。*WP 34S* 的键盘上没有或未提及的 *HP-16C* 功能在整数模式下的 X.FCN 菜单中可找到。

HP-16C	***WP 34S***	**备注**
[RL]，[RLn]	RL	在 X.FCN 菜单中。
[RR]，[RRn]	RR	在 X.FCN 菜单中。
[RMD]	[RMDR]	
[RLC]，[RLCn]	RLC	在 X.FCN 菜单中。
[RRC]，[RRCn]	RRC	在 X.FCN 菜单中。
[#B]	nBITS	在 X.FCN 菜单中。
[ABS]	[|x|]	
[DBL÷]	DBL/	在 X.FCN 菜单中。
[x⇆(i)]	多余的指令	每个寄存器都可以用于间接寻址。
[x⇆I]		
[SHOW HEX]	[⇨][16]	
[SHOW DEC]	[⇨][10]	
[SHOW OCT]	[⇨][8]	
[SHOW BIN]	[⇨][2]	
[B?]	BS?	在 TEST 菜单中。
[GSB]	[XEQ]	
[HEX]	[16]	
[DEC]	[10]	
[OCT]	[8]	
[BIN]	[2]	
[SF][3]，[CF][3]	LZON，LZOFF	在 MODE 菜单中。控制前导零的显示。
[SF][4]，[CF][4]	[SF][C]，[CF][C]	进位。
[SF][5]，[CF][5]	[SF][B]，[CF][B]	溢出。
[F?]	FS?	在 TEST 菜单中。
[(i)]	多余的指令	每个寄存器都可以用于间接寻址。
[I]		
[CLEAR PRGM]	[CLP]	注意还有 CLPALL。
[CLEAR REG]	CLREGS	在 P.FCN 中。
[CLEAR PREFIX]	多余的指令	参见 35 页。
[WINDOW]	n/a	
[SET COMPL 1`s]	1COMP	在 MODE 菜单中，请注意那里还有 SIGNMT。
[SET COMPL 2`s]	2COMP	
[SET COMPL UNSGN]	UNSIGN	
[SST]	[▼]	
[BSP]	[⇦]	
[BST]	[▲]	
[x≤y]	x≤ ?	在 TEST 菜单中，那里还有更多的判断指令。
[x<0]	x< ?	
[x>y]	x> ?	

HP-16C	*WP 34S*	备注
[x>0]		
[FLOAT]	[FIX]	
[MEM]	[STATUS]	
[STATUS]		
[CHS]	[+/-]	
[<]	[◀()]	
[>]	[)▶]	
[LSTX]	[RCL][L]	
[x≠y]	[x≠ ?]	
[x≠0]		
[x=y]	[x= ?]	
[X=0]		

涉及到的其他计算器

如上文所述，*WP 34S* 的某些高级功能是从 *HP-42S* 和 *HP-16C* 以外的惠普计算器中继承的。建议使用以下惠普计算器资料作为有关主题的更多信息（就计算和应用而言）来源。它也包含在前文提到的 DVD 中（见第 5 页）。

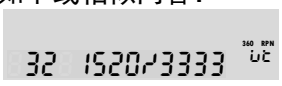
主题	文献
求根，数值积分	*HP-34C Owner's Handbook and Programming Guide* *HP-15C Owner's Handbook* *HP-15C Advanced Functions Handbook*
统计分布及其应用	*HP-21S Owner's Manual*
金融财务计算	*HP-17BII+ User's Guide*

附录 G 排除故障

排除计算中的故障

通常，当 *WP 34S* 无法满足请求时，它会抛出一条错误消息（占用显示屏的两行）。对于这类消息，请查阅附录 C。

指示符已在第 36 页解释过了，命令 RM？、SSIZE？、XTAL？、🖶？以及它们的输出已在前文中说明。此处介绍其它（可能令人困惑）的情况：

症状	可能的原因	推荐的解决方法
输入十进制数，例如 32.4567 后，显示屏显示如下或相似内容： 	意外进入了整数模式。	按[f][H.d]回到默认的浮点模式。
输入十进制数，例如 32.4567 后，显示屏显示如下或相似内容： 	意外进入了分数模式	按[f][H.d]回到默认的浮点模式。
输入十进制数，例如 32.4567 后，显示屏显示如下或相似内容： 	意外进入了编程模式。	按**[EXIT]**回到运行模式。
在二进制整数模式下，屏幕显示如下或类似内容，右上角出现了奇怪的符号： 	显示了一个长整数。	见 54 页，了解如何处理长整数。
显示屏显示如下或相似内容，运算指令没有执行： 	意外进入了编程模式。	按**[EXIT]**回到运行模式。
所有的输入都跑到第一行去了。	意外进入了字符模式。	按**[EXIT]**回到之前的模式。
显示很奇怪，但不是以上的任何一种情况。	意外进入了浏览器或菜单。	按**[EXIT]**回到之前的模式和显示界面。
输入**[1][.][2][3]**这一类的小数后出现了一个逗号小数点。怎么才能出现正常的小数点？	错误的小数点设置。	按[h][./,]或者从 40 页的表中选择一个正确的区域设置。
输入**[1][.][2][3]**这一类的小数后出现了一个小数点，但我要用逗号表示小数点。		
输入小数如 3.4567，屏幕仅显示 3.或者 3.4（有小数点但小数丢失）。	错误的输出格式设置。	设置[h][ALL]0 以显示输入的小数位。
WP 34S 不能识别输入的日期和时间。	错误的日期或时	从 40 页的表中选择一个正确的

症状	可能的原因	推荐的解决方法
	间设置。	区域设置。
输入小时、分钟和秒后，只有小时和分钟显示出来了。	错误的输出格式设置。	设置**[h][FIX]**4 将秒也显示出来。
输入日期如 3.021951，却显示 3.0220。 输入日期如 11.112012，年份被截短了。	错误的输出格式设置。	按**[h][FIX]**6 以显示完整的年份。
长数字中，每 3、4 或 2 个数字位有一个不需要的点或逗号。	TSON 或 SEPON 被设置。	如果是在整数模式，按下**[h][MODE][S][XEQ]**设置 SEPOFF；如果是在浮点模式，按下**[h][MODE][T][XEQ]**设置 TSOFF。
安装了晶振和电容，但是时间漂移很大。	晶振未激活。	参见 182 页
转换如 0.46875 的小数位为分数，得到如下结果：  尽管是对的，但这结果没有我想要的那么精确。	DENMAX 值太小。	按 0**[h][MODE][D][I][▲][XEQ]**以获得分数模式下的最大精度。
转换一个数字为分数，如 64**[1/x]**，得到的不是 1/64。	DENFAC 或 DENFIX 被设置。	按**[h][MODE][D][E][XEQ]**回到 DENANY。
打印一组数据（例如一堆寄存器）时，看到了这样的消息：  （数值可能不同）。	由于处理器供电不足而打印失败（由于红外二极管要消耗一些电流，因此会在打印时出现）。	临时的解决方法： 按**[h][MODE][f][EXIT][XEQ]**5（或类似数值）可增加打印延迟，给电池一点恢复时间。 根治此问题的办法显而易见。
打开 *WP 34S*，发现内存丢失（如第 7 页所示的 `Erased`），或者看到 `Restored`，尽管最近既没有刷机也没有重置它。	电压不足。	保证供电良好。

很多因素会导致检测到低电压[97]。

[97]电压可能由于不同原因下降，甚至可能仅仅是因为电池松动。我的第一个 *HP-20b*（后壳 ID HSTNJ-IN05）每个电池都配有一个小安全螺栓，它们可靠地固定了电池。这两个螺栓显然成为成本削减的受害者——您可以看到他们的痕迹仍然在新的后盖 HSTNJ-IN07 的塑料模具里，但是螺纹孔不见了。每个电池槽下方的圆形浅凹陷就是拧入安全螺栓的地方。我猜惠普相信电池盖 BTDR-MJ920 足以永久压住电池。

在低温下或从冷处到另一个环境（当湿气在计算器冷的部件上凝结时）电压也可能下降。它甚至可能只是零件规格的问题。请注意，我们迄今不知道真正的根本原因——我们只是注定要承担后果。以下建议将有助于排除或至少减少一些负面的影响：

1. 如果您多次看到 `Restored`，请定期使用 SAVE（例如，在关闭计算器之前），因为只要电压从零恢复到常规电平，*WP 34S* 将自动恢复上次备份。仅当没有可用的备份时，才会显示 `Erased`。
2. 增加接触压力：拆下两节电池。小心地将两个中心电池触点的三条腿向上弯曲约 1mm（参见后面的图片）。然后再次插入电池。
3. 永久克服电池接触问题：以下替代方法完全由您自行承担风险：

a）获取旧的 *HP-20b*——可能它的主板坏了。检查安全螺栓是否存在。拆解它并和 *WP 34S* 交换后盖（参

如果遇到了以上情形之外的意外情况，我们建议检查屏幕上显示的状态指示符，见 36 页。如果更改计算器设置无法解决，请用 SAVE 保存数据，调 RESET 用重置机器，然后重试。如果问题仍然存在，无论是否解决了问题，请向我们报告您的观察结果，我们将不胜感激。您可能发现了一个隐藏至今的 Bug。我们提前在此感谢您。

见附录 H—*WP 34S* 的硬件改造—步骤 **a** 和 **b**）。然后转向步骤 **j** 进行重新组装。

b）取下两个安全螺栓。它们看起来像 M1.7×3.5，但有一个高 0.5 毫米，直径 5 毫米的扁平圆柱头。如果您有这样的螺栓，您可以根据附录 H（*WP 34S* 的硬件改造）的步骤 **a**，**b**，**i2** 和 **j** 继续操作。结果见下页中的图。

安装安全螺栓后，电池至少得到了紧固。就我个人的经验而言，单靠这一点并不足以完全解决间歇性电压问题。

排除刷机故障

有时候计算器看起来怎么都不想被刷机。您可能已经按照附录 A 的步骤反复尝试过，但是屏幕就是什么都不显示。不要灰心[98]，以下步骤可能有所帮助：

1. 如果还有数据线将计算器和电脑连接，立即解除连接。

2. 检查电池。
   a. 打开电池仓门，取出全部电池。
   b. 小心地将两个中心电池触点的三条腿向上弯曲约 1mm
   c. 检查电池电压。两个电池电压都应在 2.8V-3.2V 之间。如果电压不足，将其全部更换为新的 CR2032 纽扣电池。

   d. 正确装入电池（正极朝上），如图所示。现在计算器已经准备好了。先不要盖上电池仓盖。

3. 检查电脑接口和连接线。
   a. 您可能有：
      (i)一根编程线，可能还需要用一个合适的转换器（USB—串口）；
      (ii)一个 micro USB 线，这需要在计算器安装一个定制的电路板——如果这个改装是按照附录 H 的步骤完成的并且已经检查无误，那可以放心使用；
      (iii)一个转换接头，和一个 USB 转 TTL 小板。二者已正确连接好。
   b. 检查线缆两端，有无弯曲或者脏的针脚？如果没有并且线缆没有任何割断或拉伤的迹象，那应该没问题。如果有问题，换一根同型号线缆。
   c. 将线缆连接到电脑。如果是情况(i)，在使用 USB—串口转换器的时候，应该在控制面板—设备管理器里面看到一个“USB Serial Converter”（记住它的端口号）；如果是情况(iii)，会在控制面板—设备管理器看到“USB Serial PORT”（记住它的端口号）。如果没有出现以上显示，必须更换一个转换器。
   d. 检查计算器的触点是否干净。将线缆连接到计算器。如果是情况(ii)，会在控制面板—设备管理器看到“FT230X USB Half UART”（记住它的端口号）

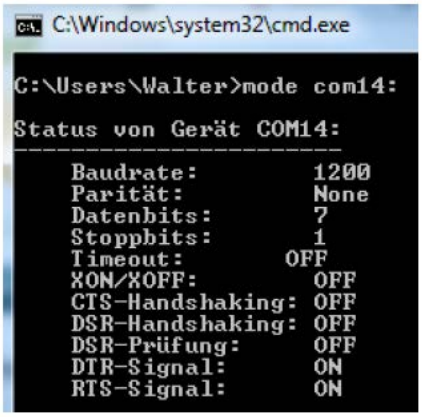
[98]目前还没有计算器因为正常刷机变砖的情况（但是可以肯定的是，由于编程线连接时间太长或者计算器后台全速运行，会有电池电量用尽的情况）。

——如果没有，那么 USB 线或者定制的电路板可能有问题（见 195 页）。

4. 所有的硬件已经准备好。启动电脑上的*MySamBa*程序，输入端口号（例如步骤3. c或3. d所确定的端口号）和刷机文件信息（见139页）。暂时不要开始传输，保持程序窗口开启！

5. 回到计算器这边——它已经和计算机连上。按下**[ON]**保持两秒钟。
计算器屏幕有反应吗？有的话，如果是第一次刷机，且是情况(i)，继续回到附录 A 步骤 A4，如果是情况(ii)回到步骤 B，如果是情况(iii)回到步骤 C，如果是升级 *WP 34S*，回到 144 页。
如果屏幕还是什么都不显示，按下 RESET 然后继续以下步骤。

6 确定端口可被操作系统正确识别。在电脑上，打开开始窗口，输入 CMD——这将打开一个古老的 DOS 窗口；在里面输入 MODE COMx，其中 x 是由步骤 3.c 或 3.d 所确定的端口号。这个 MODE 指令将返回类似这里所显示的数据，或者是以您的语言显示的同样的内容。不用关心具体数值，因为参数是由 *MySamBa* 来控制的。
如果得到了类似右图的信息，如果是情况(i)，附录 A 步骤的步骤 A3，如果是情况(ii)，回到步骤 B，如果是情况(iii)，回到步骤 C。
如果这个端口无法识别，回到上面的步骤 3.c，尝试另外一个端口。



我们两个早期的基于 *HP-20b*（无内置 USB 电路板）的计算器使用中出现了串口输入线路损坏。这些线路无法轻易修复。这是我们目前发现的仅有的两例因无法刷机导致屏幕显示空白，无法使用的 *WP 34S*[99]。

[99]译者注：如果在写入 ROM 的过程中出现中断或意外错误，计算器可能会永久变砖，无法开机也无法刷机。

附录 H 适合高级用户的额外功能

当要进行高级计算（尤其是编程）时，此处提供的信息可以简化工作。本节还介绍了可选的硬件修改（参见第 189 页）。这里收集的所有主题都需要特别小心，并更深入地了解 *WP 34S* 相应的“机制”——否则您可能会对结果感到惊讶。**使用本附录中提供的信息需要您自担风险！**

在整数模式下更改字长

在整数模式下增大或减小字长只会影响当前的堆栈内容（请参阅 IOP 中的 WSIZE 说明）。

WP 34S 的所有其他存储内容将保持原样。这与 *HP-16C* 中不同，在 *HP-16C* 中，所有寄存器都根据字长的变化进行了重映射。

请参阅以下示例，*WP 34S* 初始状态为 BASE 16，WSIZE 16，SSIZE 4，并使用 F12E 填充堆栈：

I	F12E	F12E	F12E	F12E	F12E	F12E	F12E
L	旧数据	旧数据	E61	61	61	61	2E
T	F12E	F12E	F12E	2E	2E	2E	2E
Z	F12E	F12E	F12E	2E	2E	2E	2E
Y	F12E	F12E	F12E	2E	8F	8F	2E
X	F12E	E61	FF8F	8F	2E	2E	1962
输入	**[STO][I]**	E61	**[+]**	WSIZE 8	**[RCL][I]**	WSIZE 16	**[×]**

注意：增加字长只会将空位添加到高位侧。因此，负整数将立即变为正数。没有自动扩展符号位！ 如果想要保留它，得自己小心留意。

模式存储和调用

命令 STOM 将模式数据存储在一个寄存器中。下表显示了更多详细信息。类型编码为 G 表示通用，F 表示分数，D 表示小数，I 表示整数。

位	类型	内容
0…3	G	LCD 对比度设置（0…7，在电脑上模拟器总是 4）
4，5	F	分数分母模式（DENANY = 0，DENFAC = 1，DENFIX = 2）
6…19	F	DENMAX（14 位，0…9999）
20	F	清除为 PROFRC，置位为 IMPFRC
21	F	如果分数模式打开，则置位
22，23	D	小数显示模式（ALL = 0，FIX = 1，SCI = 2，ENG = 3）

位	类型	内容
24 … 27	D	小数位数（0 …11 位）
28	D	清除为 SCIOVR，置位为 ENGOVR
29	D	清除为 RDX.，置位为 RDX,
30	D	如果是 TSOFF，则置位
31	I	如果是 SEPOFF，则置位
32	I	如果整数模式打开，则置位
33	I	LZON 打开则置位
34，35	I	整数符号模式（1COMPL = 1，2COMPL = 0，UNSIGN = 2，SIGNMT = 3）
36 … 39	I	整数进制（2 …16 的编码为 1 …15）
40 … 45	I	WSIZE（1 …64 位编码为 0 …63）
46	D	DBLON 则置位
47	D	时间格式（24h = 0，12h = 1）
48，49	G	打印模式（0 …3，参见🖷）
50		尚未使用
51	G	堆栈大小（SSIZE4 = 0，SSIZE8 = 1）
52，53	D	日期格式（Y.MD = 1，M.DY = 2，D.MY = 3）
54，55	D	角度模式（DEG = 0，RAD = 1，GRAD = 2）
56 … 58	D	曲线拟合模型（LINF = 0，EXPF = 1，POWERF = 2，LOGF = 3，BESTF = 4）
59	G	清除为 FAST，置位为 SLOW（在模拟器上始终为清除）
60 … 62	D	舍入模式（0…7，见 RM）
63	D	清除为 JG1782，置位为 JG1582

高级用户命令

有三个 ON 组合在前文未提到：

[ON]+[C]：	告诉系统安装了晶振以获得准确的实时时钟。有关硬件修改的更多信息，请参见第 189 页。晶振是时钟在长时间范围可用的必备条件（参见 DATE，TIME，STOPW，SETDAT，SETTIM），也是打印操作所必需的。系统会询问 `Crystal? InStALLEd`——保持**[ON]**按住并第二次按下**[C]**激活晶振。**[ON]+[C]**只有在晶振尚未激活的情况下才能工作。 **[ON]+[C]**只有在加载了 `calc.bin` 或 `calc_full.bin` 文件后才需要，而其他固件文件会自动启用晶振。 **警告：**如果未安装晶振，输入**[ON]+[C]**或启动了需要晶振的固件，系统将死机。此时需要硬复位（按下 RESET 按钮）或拔电池！这将会删除 *RAM* 中所有的数据。 由于无法自动检查，因此缺少红外二极管不会导致系统死机。晶振和红外二极管安装指南见 189 页。
[ON]+[D]：	切换调试模式，显示 `Debug ON` 或者 `Debug OFF`。小的=符号亮起，

	表示进入调试模式。请注意，在此模式下，*WP 34S* 不会自动关闭。**[ON]+[D]** 用于 144 页的机器升级。
[ON]+[S]：	表示 *MySamBa*：系统会询问 SAM-BA? Boot——保持**[ON]**按住并再次按**[S]**对 GPNVM1 位清零并关闭计算器。这将仅用于调试模式下（见上文），以防止意外访问这个有潜在危险的功能。它在 144 页的机器升级中会用到。 **警告：**使用此组合键后，*WP 34S* 只能在 *MySamBa* 启动模式下启动！如果没有 *MySamBa* 软件和正确的电缆（如 139 页所述），就无法继续下一步了！

双精度（DP）计算和模式切换

默认情况下，*WP 34S* 以 SP 模式启动，其中所有计算都为 16 位精度。在 152 页已经讨论了 SP 和整数模式之间的切换。此外，可以在 DP 模式下使用 *WP 34S*。每个 DP 寄存器将包含 16 个字节而不是 8 个字节，允许 34 位数字而不是 16 位（见下文）。注意矩阵指令在 DP 中不起作用。

DP 允许更精确的计算。虽然一些计算将达到高精度，但在 DP 模式下不保证所有的计算中都具有 34 位精度[100]。

下图说明了 SP 和 DP 模式之间转换时内存中发生的情况，假设在 SP 模式下设置了 REGS 16。*RRA* 表示“相对寄存器地址”

绝对地址	起始内存地址		执行 DBLON 后		再次执行 DBLOFF 后	
	内容	*RRA*	内容	*RRA*	内容	*RRA*
X+11	***k=1.40E-397***	**R111=K**	***1.40E-397***	**K=R111**	***1.40E-397***	**K=R111**
X+10	***…***	**…**	***…***	**…**		
…	***…***	**…**	***…***	**…**	***…***	**…**
X+1	***y=-33.8***	**R101=Y**		**C=R106**	***-33.8***	**Y=R101**
X	***x=123.0***	**R100=X**			***123.0***	**X=R100**

[100] *WP 34S* 的软件基于支持任意精度 BCD 数的 decNumber 库。正如 *IOP* 中某些地方提到的，内部计算是用 39 位数字进行的。其实这已经是最低限度了，有些模数的计算是用几百位数字来进行的，以避免运算抵消。更为复杂的算法则被编码为 DP 按键程序，以节省闪存空间（代价是执行速度和在 DP 模式下损失几位数的精度）。WP 34S 中用于存储数字的内部格式（SP 参见 152 页，DP 参见后文）需要从 decNumber 格式来回转换。这是个很大的开销，而且在执行速度方面也不是没有代价的。

有一个准标准可以在一定程度上找出处理器和测试计算器的准确性，计算 $arcsin\{arccos[arctan(tan\{cos[sin(9°)]\})]\}$。理想的计算器应该准确地返回 9。实际的的计算器（都是用有限的数字计算）则是不同的结果。*WP 34S* 会在 SP 模式下返回

9.000 000 000 029 361=9+3·10^{-11}

以及 DP 模式下返回

8.999 999 999 999 999 999 999 999 999 937 535=9-6·10^{-29}。

如果你对其他计算器在该测试中的结果也感兴趣的话，请参阅 http://www.rskey.org/~mwsebastian/miscprj/results.htm。

最近在网上讨论到的另一个简单的测试是：输入 1. 000 000 1，然后执行 x^2 27 次。你的 *WP 34S* 在 SP 模式下返回

674 530.470 539 687 4

以及 DP 模式下返回

674 530.470 741 084 559 382 689 184 727 772 2。

后者是目前已知的袖珍计算器最精确的结果。尽管如此，在这里最后 9 位数还是偏离了真实值。在评估微小的系统偏差或涉及到多位小数时，要考虑到这些结果。

绝对地址	起始内存地址		执行 DBLON 后		再次执行 DBLOFF 后	
X-1	***r15=-43.6***	**R15**	***…***	**B**	***0.0***	**R15**
X-2	***r14=167.9***	**R14**			***0.0***	**R14**
X-3	***…***	**R13**	***…***	**A**	***0.0***	**R13**
X-4	***…***	**R12**			***0.0***	**R12**
X-5	***…***	**R11**	***…***	**T**	***0.0***	**R11**
X-6	***…***	**R10**			***0.0***	**R10**
X-7	***…***	**R09**	***…***	**Z**	***0.0***	**R09**
X-8	***…***	**R08**			***0.0***	**R08**
X-9	***…***	**R07**	***-33.8***	**m**	***0.0***	**R07**
X-10	***…***	**R06**			***0.0***	**R06**
X-11	***…***	**R05**	***123.0***	**X=R100**	***0.0***	**R05**
X-12	***r04=-12.9***	**R04**			***0.0***	**R04**
X-13	***r03=-1234.89***	**R03**	***-1.95E184***	**R01**	***-1234.89***	**R03**
X-14	***r02=5.43E-396***	**R02**			***5.43E-396***	**R02**
X-15	***r01=6.6***	**R01**	***1.03E182***	**R00**	***6.6***	**R01**
X-16	***r00=0.54***	**R00**			***0.54***	**R00**
X-17						

通过 DBLON 从 SP 切换到 DP 模式，十二个寄存器 **X...K** 的内容被复制，将 48 个字节剪切到以前的 SP 编号寄存器区。因此，SP 模式下的前面十二个编号的寄存器数据将在这样的转换中丢失。所有其他内存内容都原地原样保持——只是每个 DP *RRA* 涵盖了之前两个 SP 寄存器。在这种转换中，为求和寄存器分配的空间不会被更改。

在默认内存配置下，执行 DBLON，会留下 44 个 DP 寄存器。在 DP 中使用参数>44 的 REGS 是合法的，但全局编号寄存器区域将侵入原程序区域。

从 DP 返回到 SP 模式，将再次复制有字母的寄存器。如果使用≤44 个 DP 寄存器，其他所有内容都保持原状，仅每个 SP RRA 指向前 DP 寄存器的一半，减少的寄存器释放的内存允许在顶部添加（或返回）12 个编号寄存器，每个寄存器数值为零。

使用>44 个 DP 寄存器时，对应性变得更加复杂——但是全局寄存器的数量不会超过 112 个。

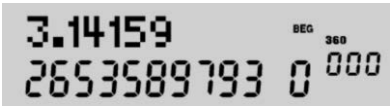
对于下表，假设在 BASE10、WSIZE32、REGS16 中启动。现在，请查看 **J**、**K** 和编号最低的寄存器的内容，通过将其调到 **X** 来检查：

	J	**K**	**R00**	**R01**	**R02**	**R03**
初始数据	3.504 d	14 d	54 d	66 d	543 d	126.441 d
DECM	3.43^{-395}	1.40^{-397}	5.40^{-397}	6.60^{-397}	5.43^{-396}	1.23^{-393}

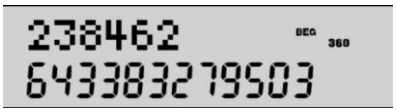
	J	K	R00	R01	R02	R03
DBLON[101]	3.43^{-395}	1.40^{-397}	1.03$^{-HI\ G}$	1.95$^{-HI\ G}$	n/a	n/a
由 sRCL 调出	1.40^{-397}	1.28^{-23}	5.40^{-397}	6.60^{-397}	5.43^{-396}	1.23^{-393}
…由 iRCL 调出	14.00	0.00	54.00	66.00	543.00	126.4 100
DBLOFF	3.43^{-395}	1.40^{-397}	5.40^{-397}	6.60^{-397}	5.43^{-396}	1.23^{-393}
由 dRCL 调出	Out of range Error	Out of range Error	0.00	0.00	n/a	n/a
DBLON	3.43^{-395}	1.40^{-397}	1.03$^{-HI\ G}$	1.95$^{-HI\ G}$	n/a	n/a
RCL J，STO J，123E456 STO 00，然后由 sRCL 调出	1.40^{-397}	1.28^{-23}	1.23^{-396}	5.12^{-30}	5.43^{-396}	1.23^{-393}
…由 iRCL 调出	14.00	0.00	123.00	0.00	543.00	126.4 100
DBLOFF	3.43^{-395}	1.40^{-397}	1.23^{-396}	5.12^{-30}	5.43^{-396}	1.23^{-393}
由 dRCL 调出	Out of range Error	Out of range Error	+∞ Error	0.00	n/a	n/a

请注意 iRCL 和 sRCL 正常工作，如第 152 页所述。

在 DP 模式下，[◀()]显示 *x* 的 34 位有效数字中的前（最高有效位）16 位数字及其四位指数，而[()▶]显示 18 个后面的有效数字，两者均为临时消息。例如，[π][◀()]显示：



[()▶]会显示：



[101]DP 模式实数大致按照 decimal128 格式编码后存储，但有一些区别。最低的 110 位存储有效数字的最右侧 33 位。其左边是一个 12 位的指数字段，然后是 5 位和 SP 完全一致的编码，最后是符号位。存储指数的最大值为 $10111111111111_2=12287_{10}$。和在 SP 中所解释的原因一样，必须从此值中减去 6176，以获得表示浮点数的真实指数。因此，DP 支持 10^{-6143} 到 10^{+6145} 内的 34 位浮点数。编码的工作原理与附录 B 中描述 SP 的方式完全类似。

使用十进制有效数字甚至可以输入更小的数，但每减小一个数量级就会丢失一个有效数字。如果将 10^{-6143} 多次除以 10，也会发生同样的情况。减少到 10^{-6176} 时将仅剩一位有效数字，这个数字作为十六进制的 1 存储。将其除以 1.9 999 999 999，结果仍为 10^{+6176}。将其除以 2，结果将变为零。

超出 10^{-999} 到 10^{+1000} 范围的数字将分别在其指数中显示 -HI G 或 HI G。只有[◀()]键可以显示它们的真实指数。用此方式可以查看 ***r00***=1.032 000 000 000 000 054 E-6155 和 ***r01***=1.951 656 000 000 000 000 000 543 E-6152 这样的数字。

字母寄存器的数字返回 SP 模式时，如果数值超过 SP 的范围，会显示 0 或 InFinitY。

请记住，不是每一次调用都能和上例一样精确。误差如 CONST 的脚注所述会累加。

注意：舍入模式（参见 RM）的差别可能会影响 DP 计算的结果！

库和 *XROM* 程序中使用的其他命令

下面列出的操作供编程的程序员使用。这些指令使得编写程序时比使用原指令集更舒适，使得编程工作更轻松。这里解释了这些指令和伪指令便于对这些程序的理解，但是它们不能通过任何菜单访问。编译器（通常包含它的预处理程序）将大多数汇编程序转换为前文提到的指令，从而生成正确的程序步。更多的信息请参阅这个用户指南：

http://sourceforge.net/p/wp34s/code/HEAD/tree/doc/WP34s_Assembler_Tool_Suite.pdf。

（伪）指令	*XROM* 或 *LIB*	功能
A...D→	*XROM*	将 ***a***，***b***，***c*** 和 ***d*** 保存在易失性临时存储中。易失性意味着数据会在输入后约 500*ms* 丢失。
GSB ***longlabel***	均可	此伪指令允许调用长字母标签。如果距离太远，汇编程序将 GSB 转换为 BSRB 或 BSRF（参见第 188 页）——如果距离过远会转换为 XEQ（通过 XROM 中的跳转表或通过 LIB 中的标签）。
JMP ***longlabel***	均可	此伪指令允许分支到长字母标签。汇编程序将 JMP 转换为 BACK 或 SKIP——如果距离太远会转换为 GTO（通过 XROM 中的跳转表或通过 LIB 中的标签）。
		注意： *WP 34S* 上的 STOP 和 PROMPT 不适用于 XROM。并且 XROM 代码中没有 LBL 和 END 语句。 长字母标签（看起来像 ***longlabel*****::**）可以超过 6 个字符。它们可能会出现在为汇编程序编写的代码中，但会被解析。
Num ***sn***	*XROM*	当 ***sn***=0，π，√2π，…等数值时，插入相应的常数，就像从 CONST 中调用它一样。注意和_INT 的区别。
POPUSR	*XROM*	必须紧跟在 XEQUSR 之后，才能正确地恢复 XROM 的执行状态（寄存器、标志位和返回地址）。
XEQ ***label***	*XROM*	调用带有指定全局标签的用户程序（XROM 中没有 LBL 语句）。另见 xIN。
XEQUSR ***label***	*XROM*	将堆栈用 ***x*** 填满，并调用包含用于求解，积分，求和或求积的函数的程序。该功能的标签作为上述相应指令（例如 SLV）的参数发送。 XEQUSR 后必须立即跟随 POPUSR 才能正确恢复 XROM 执行状态（寄存器，标志位和返回地址）。.
xIN ***type***	*XROM*	使用 ***type*** = NILADIC，MONADIC，DYADIC，TRIADIC 或 . . . _COMPLEX 来定义函数输入需要多少层级的堆栈。此外，它还进行了一些初始化工作（例如 SSIZE8 和 DBLON）。 推荐 XROM 程序以 xIN 开始。此外，SSIZE4 也是可以的，但是 DBLOFF 不行。此处要注意 xIN 是无法嵌套的，使用 xIN 的 XROM 程序无法调用用户代码。.
XLBL"***label***"	*XROM*	定义这个程序的外部标签。XLBL 不生成任何代码，它提供了可以利用 C 函数表的入口点。通常，***label*** 代表相应的指令名称，可以超过 6 个字符。
xOUT ***way***	*XROM*	通常，***way***=xOUT_NORMAL。xOUT 清除并恢复 xIN 的设置，请小心保证合适的返回值，包括 ***l*** 和堆栈正确设置。通常，xOUT 应该是 XROM 程序的最后一个指令。
_INT ***n***	*XROM*	插入相应的整数，如同#那样。也可以是编译时计算的数。注意和 NUM 的区别。
"***text***"	均可	双引号允许方便地输入文本字符串。汇编程序将字符串转换为所需数量的 α 语句。
→A...D	*XROM*	从它们的临时存储中调出 ***a***，***b***，***c*** 和 ***d***（临时存储保存时间非常短，见上文）。.
# ***name***	*LIB*	其中***name***= □ …，插入和名称对应的常数，如同从CONST中调用一样。是*IOP*中#指令的扩充。

（伪）指令	*XROM* 或 *LIB*	功能
#define ***name value***	*XROM*	允许在此类例程中使用命名变量，以便于阅读。名称可能超过六个字符。每个#define 都必须用#undef 关闭。
#undef ***name***	*XROM*	

此外，有几种“别名”拼写可以简化英语计算机键盘上某些指令甚至单个非标准字符的输入。例如：希腊字母“α”经常被拉丁字母“a”取代，打印字符“🖷”被拉丁字母“P.”取代等。编译器将负责翻译别名。

有关示例，请参阅文件 `8queens_alias.wp34s`（或程序库中名为…`_alias.wp34s` 的文件）。技术文档中的 `Command-Aliases.pdf` 包含所有别名的列表（大约 50 页）。

汇编输出

编译器将`.wp34s` 文本文件作为输入，并为 *WP 34S* 生成优化代码。它不是为了程序的进一步编辑和维护而优化。“优化”意味着它将尽可能少地使用 *WP 34S* 中有限的资源，尤其是更少地使用标签。因此，在编译的程序中尽量使用 BACK 和 SKIP 而不是 GTO，因为在编译时足以获得所有转换所需的信息。

此外，有两个特殊指令经常在汇编代码中替换 GSB/XEQ：

1. BSRB ***n*** 调用向后 ***n*** 步的子程序，0≤***n***≤255。它将程序计数器压入子程序返回堆栈上，然后执行 BACK ***n***。

2. BSRF ***n*** 调用向前 ***n*** 步的子程序。它将程序计数器压入子程序返回堆栈上，然后执行 SKIP ***n***。

以这种方式调用的子程序不需要起始标签。因此，如果缺少本地标签，BSRB 和 BSRF 非常有用——就和 BACK，SKIP 和 CASE 一样。另一方面，如果编辑由一个或多个 BSRB，BSRF，BACK，SKIP 或 CASE 跳过的程序的一部分，这可能导致需要单独手动维护所有这些语句。为了程序的可读性，可维护性和可靠性，我们建议编辑未编译的代码并将无标签分支的生成留给汇编程序。

HP-20b 和 *HP-30b* 的基本硬件规格

电源：3V（2 个 CR2032 纽扣电池）

处理器：Atmel AT91SAM7L128

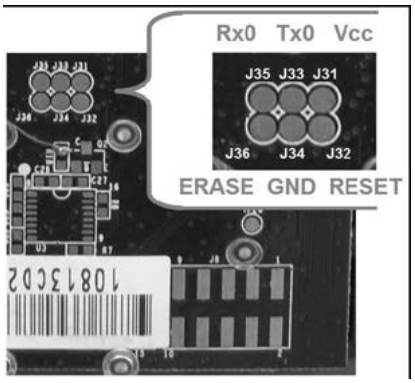
—带 thumb 模式的 ARM 7 内核（32kHz 至 40MHz）

—RAM：2kB 可备份，4kB 易失性

—ROM：128kB 闪存，*SAM-BA* 引导加载程序
—I/O：并口，用于刷写固件的串口（请参见下图中的引脚排列）
—LCD 控制器
—完善的电源管理
—多个时钟源（RC，石英，PLL），实时时钟

400 段液晶显示屏：
—12 x 9 段（有效数字），3 x 7 段（指数），2 个符号，
—6 x 43 像素的点阵，
—10 个指示符
—1 个固定符号（参见第 35 页）。

37 个键的键盘，分 5 到 6 列，7 行排列。



计算器的六个串行端触点于电池盖下方。它们的间距是 2 毫米，这里给出了引脚定义。两种计算器型号（HP-20b 和 HP-30b）都是相同的。电压电平为 0 到 3V。图片是打开机器后盖后看到印刷电路板，RESET 按钮位于六个触点的上方（按下此按钮相当于短路 RESET 和 GND 触点）。

主板下方有一个额外的 JTAG 接口（从外部无法访问），在图中被标签覆盖了一部分。

WP 34S 的硬件改造

正如手册开头所述，这里有三个可选的硬件改造。请阅读各段简述并选择感兴趣的方法。在进行实际改装之前，请阅读这些方法的详细介绍，以防 *WP 34S* 变砖。下面是三种改装：

1. 在 *WP 34S* 的主板上安装额外的一个晶振和两个电容，使其具备准确的实时时钟功能。

由于电容尺寸很小，这个改装需要精巧的焊接技能。这些 SMD 元件比一粒谷子还小。除了将这两个电容安装在正确的位置之外，这个改装是一个容易的工作。在 *wp34s.svn.sourceforge.net/viewvc/wp34s/doc/How_to_install_crystal_and_IR_diode.pdf* 可以看到 *Alexander Oestert* 写的很棒的介绍文档[102]。

[102]本手册后附有此文档的中文翻译。

2. 在改造1完成后，通过安装一个红外二极管和电阻，以实现与HP-82240A/B打印机之间的红外线连接。电路的改造要比上一个改造简单的多,这里有一些额外的机械改装：在外壳上打一个小孔，这需要精准的手工和正确的钻头，上面的文档有提到如何完成这项工作。

3. 可以通过安装一个 *Harald Pott*[103] 开发的定制电路板，它将 IR 打印机和用于数据交换和提供电源的 USB 接口结合起来，它在附录 A中被提到过。

这里不仅需要焊接技能（并不需要像改造 1 那样精细，但也有一定要求），也需要机械技能，因为除了安装红外二极管所需的小孔外，计算器外壳上还需要一个用于 Micro USB 插座的开孔。

接下来，我们会使用一台 *HP-30b* 以及由 *Alexander* 和 *Harald* 提供的材料，基于上图所示的定制电路板介绍如何改造。

a. 拆解计算器（简单的部分）。

需要的工具：小型十字螺丝刀，木制牙签。

步骤：

移除电池盖。

用牙签拆除两个纽扣电池。

卸下后盖的两颗螺丝。

b. 拆解计算器（困难的部分）。

需要的工具：瑞士军刀

有用的信息：*HP-30b* 亮银色前面板是卡在灰色框架中的。在计算器的两侧各有三个卡扣，顶部两个，底部一个。参见拆开的框架照片，

[103]除了这款，*Harald Pott* 还为 *WP 34S* 开发了多款定制板，我们选择被他叫做 PCB Version2 的定制电路板。我们认为它是最有用的，因为它结合了上述功能，并且如下文一样易于安装。阅读 *http://wp34s.svn.sourceforge.net/viewvc/wp34s/doc/WP_34S_USB_installation.pdf* 获取有关其他定制板的信息和 *Harald* 的联系方式。

步骤：

在计算器一侧的中央，小心的将军刀的大刀插入前面板和框架之间。,用力向上撬起前面板。

在另一侧重复此做法。

将刀片向计算器的顶部移动。轻微用力来分开两侧外壳。此处请小心操作，以免损坏液晶屏。

将刀插入计算器顶部的框架之间，轻微用力分开前面板。现在前面板顶部的部分应该会松动。

最后用同样的方式使底部松动。

将前面板翻转过来（印刷电路板向上）放在桌面上。将灰色的框架放在一旁。

c. 安装 USB 板之前的准备。

需要的工具：细小而锋利的小刀

步骤：

切去顶部右侧螺丝柱和屏幕框架的塑料支架。当完成时，计算器外观应该和下图一致。

d. 安装 USB 板。

需要的材料：任何塑料胶水，例如 UHU。

步骤：

在螺丝柱周围，“h.pott”的左侧，USB 接口的下端涂上胶水，将电路板固定在计算器上。将它们放在一边晾干。

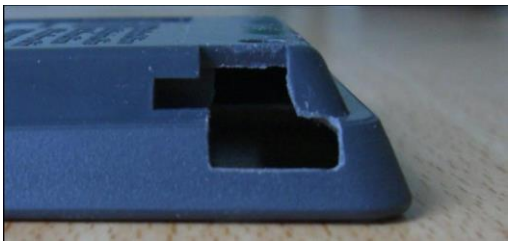
e. 为 USB 接口开孔。

需要的工具：小方锉。

步骤：

拿起灰色框架底面向上放在桌面上，橡皮脚垫面向自己。然后扩大顶部右侧的开口。

开口的位置和大小视 USB 板而定。当完成时，框架的情况如下图所示。记住，电池盒盖在上面。



f. 将红外二极管与 USB 板相连。

需要的工具：烙铁（烙铁头不应比焊点大太多，推荐使用小于 30 瓦的烙铁）。

需要的材料：焊锡。

步骤：

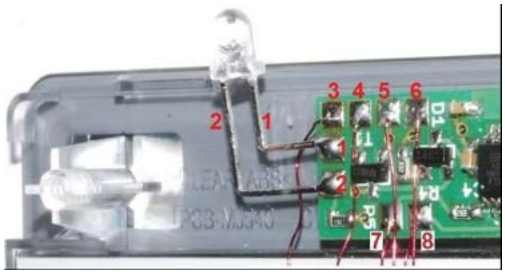
预热烙铁。

在灰色框在上寻找一个可供二极管穿过的地方。安装位置应该远离芯片。

转动二极管，使二极管的长脚在右侧。弯曲长引脚使其到达 1 号焊点（如图所示）。弯曲较短的引脚使其达到 2 号焊点。修剪引脚使引脚长度合适。两引脚之间不能相互接触。

在焊点 1 挂上少许焊锡。在焊点 2 和二极管两引脚末端也这样做。

二极管原来的长引脚焊接到焊点 1。二极管原来的短引脚焊接到焊点 2。



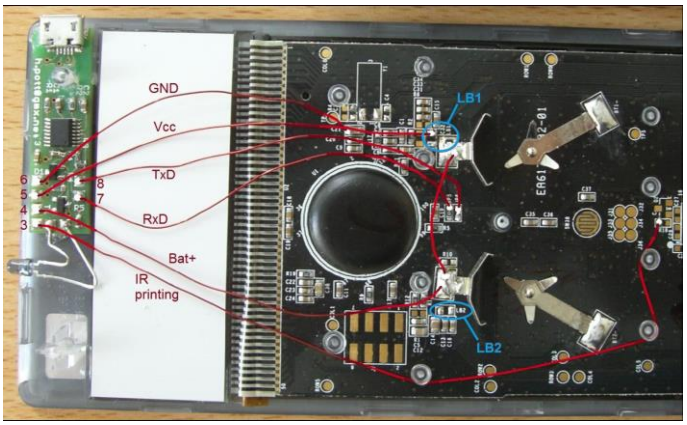
g. 将 USB 板与计算器主板相连。

需要的工具：与步骤 f 相同的烙铁。

需要的材料：漆包线（直径约 0.3mm，这样便于布线），焊锡。

步骤：

USB 板和计算器主板之间需要六条连线，计算器主板上还需要一条连线。



首先必须从计算器主板上拆除两个元件。

拆除 LB1。因为之后不会用到这个元件，所以将加热的烙铁头放在 LB1 的两侧一段时间来加热它。过一段时间，LB1 应该会融化变得容易移除。它甚至会黏在烙铁头上并因此离开电路板。

用相同的方式移除 LB2。

按以下方式连接焊点：

- 3（红外二极管）连接到 R18 上边的焊点；
- 4（电池正极）连接到电池上边的大焊点；
- 5（电源正）连接到 LB1 的右侧焊点；
- 6（公共地）连接到 C21 顶部的焊点；
- 7（数据接收）连接到 JP5 的任意一端[104]；
- 8（数据发送）连接到 JP6 的任意一端；
- 连接两个电池上边的大焊点。

对于每个连接，切下一段约 150mm 的电线。剥出最前端 2mm 的芯线，在线头和焊点上挂锡，并将一端焊在 USB 板的焊点上。然后确定连接需要的长度，裁剪电线，剥出另一端的 2mm 芯线，挂锡。之后焊接在计算器主板的焊点上。

[104]JP5 和 JP6 是跨接电阻器，即 0 欧电阻。

最后，用欧姆表检查每一个焊点是否良好连接（测试电压约 1.5V）[105]。

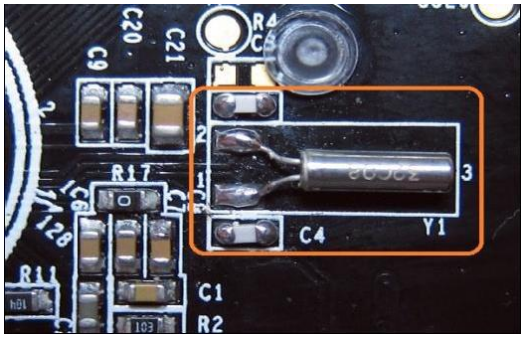
h. 可选改装：安装额外的晶振以获得准确的实时时钟。

需要的材料：与步骤 **f** 相同的烙铁，用于固定晶振的木制牙签。

需要的材料：两个 18pF 0603 电容，一个 32.768KHz 石英晶振，焊锡。

步骤：

检查计算器主板的右上角。如果石英晶振 Y1 和附近的两个电容 C3、C4 还没有被安装，现在可以将它们焊上。这不是必须的，但是有利于通讯。但如果要使用所有打印、时间、日期指令以及秒表则必须要安装它们。有关部件规格和工作说明，请参见此处的图片以及 189 页第 1 点中提到的文档。



对于一般人来说，焊接这些电容是一个挑战，因为它们真的非常的小。也可以使用直插电容，尽管看起来非常不专业，但是可以更简单的完成这项工作。记得在改装完成后激活晶振（参见 182 页）。

i. 为红外二极管钻孔。

需要的工具：与二极管尺寸相同的钻头，如果不要求快速完成，可使用任何可以制造合适尺寸圆孔的工具都行（甚至是瑞士军刀）。

步骤：

将红外二极管按在灰色框架上以确认钻孔位置。在它的中心距离框架边缘约 5mm 处开始钻孔。

j. 重新组装。

需要的工具：小号十字螺丝刀。

步骤：

检查前面板是否可以安装上。然后将其装入灰色框架中，将二极管和钻好的孔对齐。压入卡扣（最好从顶部和底部开始，不要按压液晶屏或过度用力）

拧上五个螺丝。

装回纽扣电池。

盖上电池盖，计算器装好了。

[105]因为电子元件很小并且密集地安装在电路板上，这很容易产生连焊。请彻底检查，并在发生连焊时移除它们。还要注意不要将任何装好的元件加热到可移动的程度。在主板上焊接需要温和的操作，良好的视力和一些经验。

k. 将计算器通过 USB 与电脑连接。

需要的材料：Micro USB 电缆。

步骤：

将电缆插入计算器。根据插头的形状，可能需要切去插头上的部分塑料使插头适合插入。

打开计算器。

将它与电脑相连。如果操作正确，在设备连接时电脑的操作系统将自动开始寻找驱动。它也可能找不到。可从 FTDI 获取对应操作系统的最新驱动（访问 *http://www.ftdichip.com/Drivers/VCP.htm*）。安装驱动后，连接设备时设备管理器将会显示计算器为“FT230X USB Half UART”并会分配一个 COM 端口。需要通过这个端口刷新固件

现在，可以像附录 A中提到的那样，很容易的刷新和升级固件了。

附录 I 高等数学函数

WP 34S 有一些操作涉及高等数学。他们是首次用于 RPN 计算器上；它们都工作在 DECM 模式下。这里将这些函数集中起来，进行更为详细地描述。此外还有一些符合本章主题的传统的袖珍计算器功能。

基于本手册第一部分中解释的原因，我们假设您能够阅读并了解实数域函数的数学公式。无论复数是否是有效的输入或输出，请确保了解相应的基本数学概念（至少如第 29 页所述）。否则，不用理会这些功能。根据经验，使用既不知其然也不知其所以然的东西是没有益处的——它甚至可能对您和您的同事是危险的。

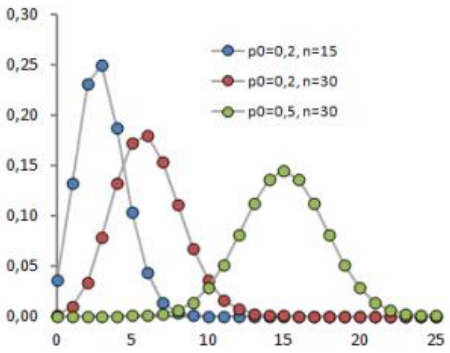
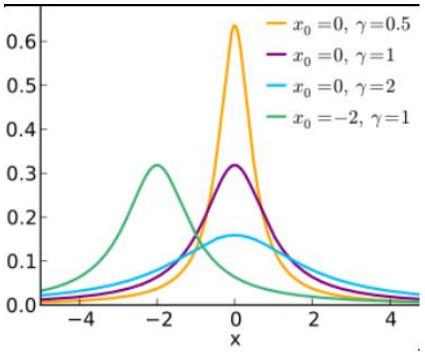
数字

除非另有说明，以下都是一元函数。

名称	说明（通用信息参见 77 页）
Bn Bn*	返回对 **X** 中给定的大于 0 的整数 ***n*** 的伯努利数： $B_n = (-1)^{n+1} n \cdot \zeta(1-n)$。 Bn*为旧定义的伯努利数： $B_n^* = \frac{2 \cdot (2n)!}{(2\pi)^{2n}} \cdot \zeta(2n)$。 参见 206 页的 ζ(x)。
COMB	(2)对于 ***y***≥***x***≥0 的实数，计算组合数： $C_{y,x} = \binom{y}{x} = \frac{y!}{x! \cdot (y-x)!} = \frac{\Gamma(y+1)}{\Gamma(x+1) \cdot \Gamma(y-x+1)}$。含有 Γ 函数的公式也适用于非整数。 $C_{y,x}$ 和二项分布有关：在一个有 ***y*** 行钉子的高尔顿钉板中投入 2^y 个小球，$C_{y,x}$ 为落在 x 列中的小球数的期望值（第 1 列计为 0）。
FIB	在整数下对 ***n***=***x*** 返回斐波那契数 f_n。这个数列定义为 f_0=0，f_1=1；对于 n≥2，f_n=f_{n-1}+f_{n-2}。在 UNSIGN 模式下，f_{93} 是不溢出的最大值。
	在 DECM 模式下，对任意实数或复述 ***x*** 返回扩展斐波那契数： $F_x = \frac{1}{\sqrt{5}} \left\lfloor \left(\frac{1+\sqrt{5}}{x} \right)^x - \left(\frac{2}{1+\sqrt{5}} \right)^x \cos(x\pi) \right\rfloor = \frac{1}{\sqrt{5}} \left[\Phi^x - \Phi^{-x} \cos(x\pi) \right]$，其中 Φ 为黄金分割比例。
PERM	(2) 对于 ***y***≥***x***≥0 的实数，计算排列数： $P_{y,x} = \frac{y!}{(y-x)!} = \frac{\Gamma(y+1)}{\Gamma(y-x+1)}$。含有 Γ 函数的公式也适用于非整数。

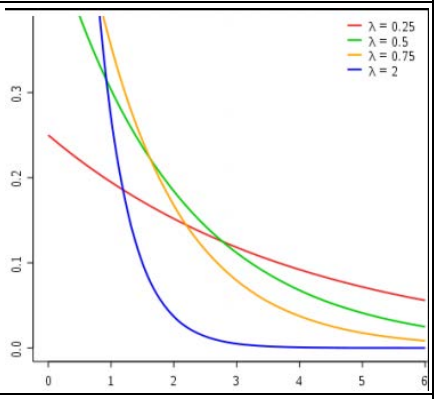
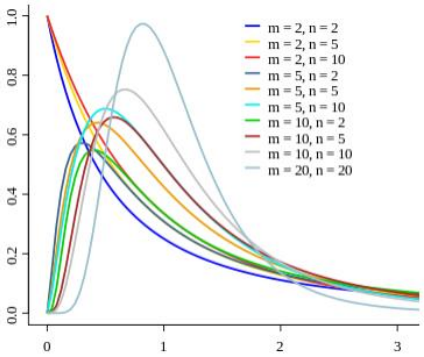
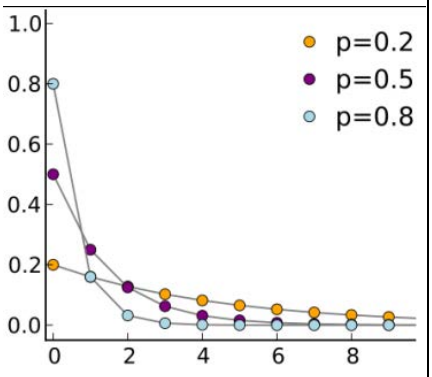
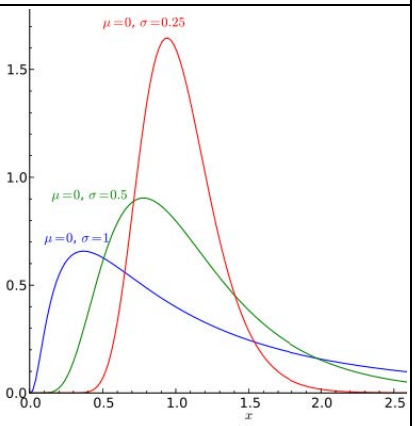
统计分布

从堆栈的使用方式看，下列均为一元函数，储存于 PROB 中。下表中，所列前三者为离散分布，而后者均为连续分布函数。下表分别列出了典型的 *pmf* 及 *pdf* 图。

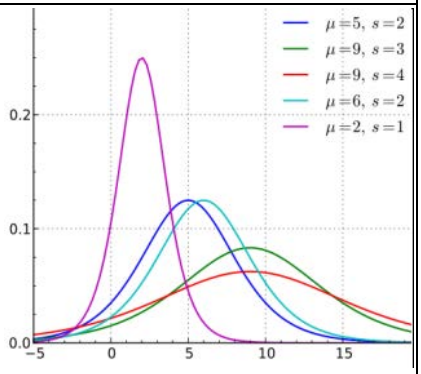
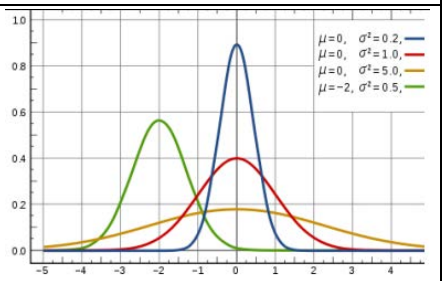
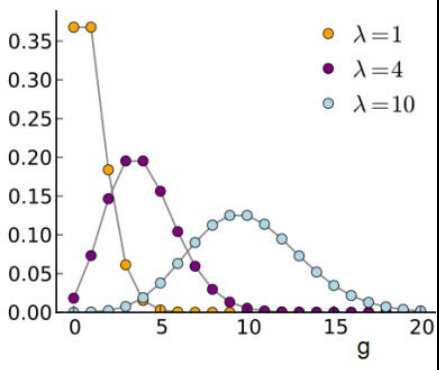
名称	说明（通用信息参见 77 页）
Binom	成功次数 *g* 由 **X** 给出，成功概率由 p_0 由 **J** 给出，样本 ***n*** 由 **K** 给出的二项分布。  Binom$_P$ 返回： $P_B(g;n;p_0)=\binom{n}{g}\cdot p_0^g\cdot(1-p_0)^{n-g}$ $=C_{n,g}\cdot p_0^g\cdot(1-p_0)^{n-g}$。 （参见上文 COMB） Binom 返回 $F_B(m;n;p_0)=\sum_{g=0}^{m}p_B(g;n;p_0)$，其中最大成功次数 ***m*** 由 **X** 给出[106]。 二项分布在工业上的采样误差统计中起重要作用。如，设计测试计划。如果想知道，在 300 个同一批次产品中所抽取到的 15 个样品中没有出现缺陷产品的概率，已知有 3%的产品缺陷率，请如下操作： 0.03**[STO][J]**15**[STO][K]**0**Binom** 返回 0.633——抽取到无任何缺陷的 15 个样品的概率约为 2/3[107]。 可从此获取更多信息： *https://www.itl.nist.gov/div898/handbook/eda/section3/eda366i.htm*。
Cauch	柯西-洛伦兹分布（又名洛伦兹分布或布雷特-维格纳分布），其中位置参数 x_0 由 **J** 给出，尺度参数 γ 由 **K** 给出。  Cauch$_P$ 返回 $f_{Ca}(x)=\left\{\pi\gamma\cdot\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]\right\}^{-1}$， Cauch 返回 $f_{Ca}(x)=\frac{1}{2}+\frac{1}{\pi}\arctan\left(\frac{x-x_0}{\gamma}\right)$， Cauch^{-1} 返回 $F_{Ca}^{-1}(p)=x_0+\gamma\tan\left[\pi\cdot\left(p-\frac{1}{2}\right)\right]$。 此类分布流行于物理学中。它是学生 t 分布的一个特例。 可从此获取更多信息：*https://en.wikipedia.org/wiki/Cauchy_distribution*。

[106]在 Microsoft Excel 中 Binom$_P$ 当于 BINOMDIST(***g***; ***n***; ***p*****0**; **0**)，而 Binom 则相当于 BINOMDIST(***m***; ***n***; ***p*****0**; **1**)。

[107]准确数值为 0.626，这是使用了数学上更为准确的模型的结果。此类结果很好地展示了两个有效数字是典型的统计误差——没什么是比通常使用的（简化）模型更符合现实的了。

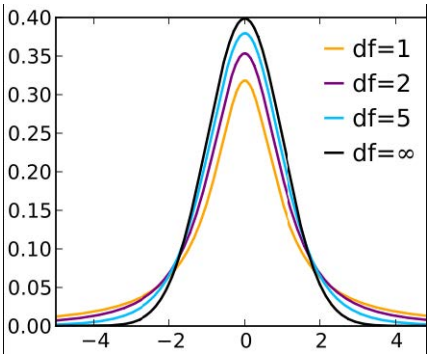
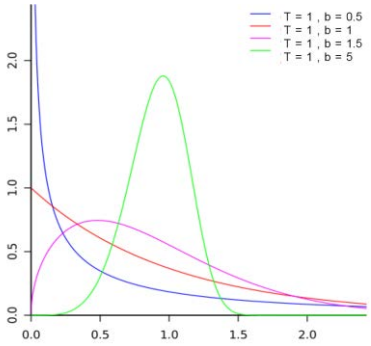
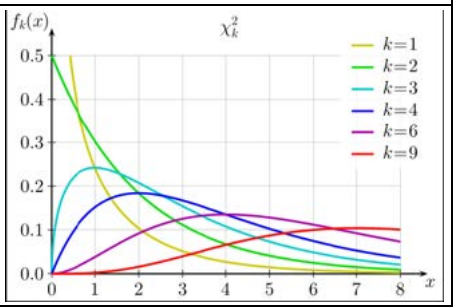
名称	说明（通用信息参见 77 页）
Expon	指数分布，率参数 ***λ*** 由 **J** 给出。 Expon$_P$[108]返回 $f_{Ex}(x)=\lambda\cdot e^{-\lambda x}$，参见右图所绘曲线。 Expon 返回 $F_{Ex}(x)=1-e^{-\lambda x}$。 可从此获取更多信息：*http://www.itl.nist.gov/div898/handbook/eda/section3/eda3667.htm*。 
F(x)	费舍尔 F 分布，自由度由 **J** 和 **K** 指定。它被用于方差分析（ANOVA）这样的场合。 图片展示了对于对应于 ***j*** 和 ***k*** 的不同自由度 ***m*** 和 ***n*** 所绘制的 pdf 图像。 可从此获取更多信息：*http://www.itl.nist.gov/div898/handbook/eda/section3/eda3665.htm*。 
Geom	几何分布： Geom$_P$ 返回 $f_{Ge}(n)=p_0(1-p_0)^n$， Geom 返回 $F_{Ge}(m)=1-(1-p_0)^{m+1}$， 为 ***m*** 次伯努利实验后的首次成功率，其中 ***m*** 由 **X** 给出。必须在 **J** 中指定每个此类实验的成功率 $\boldsymbol{p}_0$。可从此获取更多信息：*https://en.wikipedia.org/wiki/Geometric_distribution*。 
LgNrm	对数正态分布，$\mu=\ln\bar{x}_g$ 由 **J** 给出，$\sigma=\ln\varepsilon$ 由 **K** 给出。*Pdf* 参见右图所绘图像。 LgNrm$_P$ 返回 $f_{Ln}(x)=\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{[\ln(x)-\mu]^2}{2\sigma^2}}$， LgNrm 返回 $F_{Ln}(x)=\Phi\left(\frac{\ln(x)-\mu}{\sigma}\right)$，其中 $\Phi(z)$ 表示标准正态 *cdf*，如本表最后所示。 右图展示了指定参数的对数正态 *pdf*。可从此获取更多信息：*https://www.itl.nist.gov/div898/handbook/eda/section3/eda3669.htm*。 
Logis	由 **J** 给出的任意平均值 ***μ***，及 **K** 给出的尺度参数 ***s*** 表示的逻辑斯蒂分布。

[108]在 Microsoft Excel 中 pdf 相当于 EXPONDIST(***x***; ***λ***; **0**)，而 cdf 则相当于 EXPONDIST(***x***; ***λ***; **1**)。

名称	说明（通用信息参见 77 页）
	$\xi = \frac{x-\mu}{s}$。 Logis$_P$ 返回 $f_{Lg}(x) = \frac{e^{-\xi}}{s\cdot\left(1+e^{-\xi}\right)^2}$，Logis 返回 $F_{Lg}(x) = \frac{1}{1+e^{-\xi}}$，Logis^{-1} 返回 $F_{Lg}^{-1}(p) = \mu + s\cdot\ln\left(\frac{p}{1-p}\right)$。  可从此获取更多信息：*http://en.wikipedia.org/wiki/Logistic_distribution*。
Norml	由 **J** 给出的任意平均值 ***μ***，及 **K** 给出的任意标准差 ***σ*** 表示的正态（或高斯）分布。 Norml$_P$[109]返回 $f_N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$， Norml 返回 $F_N(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$，其中 $\Phi(z)$ 表示标准正态 *cdf*，如本表最后所示。一般来说，将 0 存储在 **J** 中，将 0 存储在 **K** 中，并调用 Norml 对应的指令会得到标准正态分布。  可从此获取更多信息： *http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm*。
Poiss, Poisλ	以 **X** 给出的成功次数 ***g***，**J** 给出的泊松参数 ***λ***（Poisλ）或者 **J** 给出的总的错误概率 p_0 和 **K** 给出的样品数量 ***n***（Poiss）给定的泊松分布。在后一种情况下，会根据 ***λ*** **=*n*·p_0** 自动算出 ***λ***。 Poisλ$_P$[110]计算 $P_P(g;\lambda) = \frac{\lambda^g}{g!}e^{-\lambda}a$，而 Poisλ 返回 **X** 中的最大成功数 ***m*** 对应的 *cdf*。 泊松分布为工业采样测试提供数学上最简单的统计模型。重复上述二项分布中的示例，这里得到 0.638。  可从此处取更多信息： *https://www.itl.nist.gov/div898/handbook/eda/section3/eda366j.htm*。

[109]在 Microsoft Excel 中 Norml$_P$ 相当于 NORMDIST(***x***; ***μ***; ***σ***; **0**)，而 Norml 则相当于 NORMDIST(***x***; ***μ***; ***σ***; **1**)，而 Norml^{-1} 则相当于 NORMINV(***F*$_N$**; ***μ***; ***σ***)。

[110]在 Microsoft Excel 中 Poisλ$_P$ 相当于 POISSON(***g***; ***λ***; **0**)，而 Poisλ 则相当于 POISSON(***g***; ***λ***; **1**)。

名称	说明（通用信息参见 77 页）
t(x)	由 **J** 给出自由度确定的标准化学生 t 分布，用于假设检验及计算诸如平均值的置信区间。右图显示了以不同自由度所绘制的 *pdf* 图像。对于 ***df***→∞，函数肩部缩小，且 *pdf* 图像接近标准正态分布图像（图中红色曲线）。 可从此获取更多信息：*http://www.itl.nist.gov/div898/handbook/eda/section3/eda3664.htm*。 
Weibl	由 **J** 给出状参数 ***b***，**K** 给出特征寿命 ***T*** 确定的威布尔分布。 t≥0，Weibl$_p$[111]返回 $f_W(t)=\frac{b}{T}\left(\frac{t}{T}\right)^{b-1}e^{-\left(\frac{t}{T}\right)^b}$，否则返回 0。这是一种非常灵活的函数——参见右图。 Weibl 返回 $F_W(t)=1-e^{-\left(\frac{t}{T}\right)^b}$。 可从此获取更多信息：*http://www.itl.nist.gov/div898/handbook/eda/section3/eda3668.htm*，更多应用场景见 *http://en.wikipedia.org/wiki/Weibull_distribution#Uses*。 
$\varphi(x)$，$\Phi(x)$	$\varphi(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ 是标准正态 *pdf*（知名的钟形曲线，参见正态分布节配图的红线），而 $\Phi(x)=\int_{-\infty}^{x}\varphi(\tau)d\tau$ 是对应的 *cdf*（类似 206 页所展示的误差函数）。参见前文 NORML 给出的链接。
χ^2	卡方分布，用于计算标准差、方差等的置信区间。图像显示了以不同自由度所绘制的 *pdf* 图像。 可从此获取更多信息：*http://www.itl.nist.gov/div898/handbook/eda/section3/eda3666.htm*。 

[111]在 Microsoft Excel 中 *pdf* 相当于 WEIBULL(*x*; ***b***; ***T***; **0**)，*cdf* 则相当于 WEIBULL(*x*; ***b***; ***T***; **1**)。

更多统计公式

注意，测量抽样的完整结果必须包含期望值及其不确定性信息。

- 对于以（可加）高斯过程进行的采样，期望值是其算数平均值，而其不确定性由其标准误差给出（参见 $\bar{x}$ 节及 SERR 节）。
- 对于以对数正态（几何）过程进行的采样，期望值是其几何平均值，而其不确定性由其分布系数给出（参见 $\bar{x}_g$ 及 ε_m ）。
- 对于以其他过程进行的采样，由其他方法来确定。

可以肯定的是，现实世界上并非所有事物都遵循高斯分布[112]！过程的特征可以通过合适的检验来确定（并且在计算统计数据前进行检查）——使用正确的参考文件也是必要的。

下表左侧名称的函数均可在 STAT 中找到（除了 $\bar{x}$ 和 **s**，它们在这里仅作比较用途）。

名称	说明（通用信息参见 77 页）
CORR	于线性拟合模型，其相关系数为 $r=\frac{s_{xy}}{s_x\cdot s_y}$ 。参见下文 S_{XY} 及 s。 对于任意拟合模型 ***R(x)***，值 $r^2=1-\frac{\sum\left[R(x_i)-y_i\right]^2}{\sum\left(\bar{y}-y_i\right)^2}$ 是可决系数；它表示由独立数据 x 的变化所确定的非独立数据 y 的变化比例。对于 r^2=1，y 完全由 x 来决定；对于 r^2=0，y 完全独立于 x；对于 r^2=0.93， y 的总变化的 93%都取决于 x 。 若 $\lvert r\rvert\cdot\sqrt{\frac{n-2}{1-r^2}}>t_{n-2}^{-1}(0.99)$，则回归是显著的，其中式子右侧为自由度为 n-2，置信度为 99%的逆 t 分布。
COV	对于线性拟合模型，其总体协方差为 $COV_{xy}=\left(n\sum x_iy_i-\sum x_i\sum y_i\right)/n^2$ 。参见下文的 S_{XY}。

[112]通常，应选择与观察结果最匹配的统计模型。但是在现实生活中，大多数事物都与模型分布存在显着差别——因此，您不能指望计算出的结果能够更匹配于现实。

顺便说一句：由于高斯分布的 *pdf* 永远不会达到零，因此该统计模型告诉您，当样本变得足够大时，您可以期望单个个体的值远远偏离平均值。但是这是与现实不符的。因此，我们必须承认在现实世界中没有完全符合高斯分布的事物。然而，高斯分布是描述许多现实世界观察结果的非常成功的模型。只是永远也不要忘记这种模型的局限性。

名称	说明（通用信息参见 77 页）
L.R.	对于线性拟合模型，线性参数为 $a_0=\frac{\sum x_i^2\cdot\sum y_i-\sum x_i\cdot\sum x_iy_i}{n\cdot(n-1)\cdot s_x^2}$ 及 $a_1=\frac{s_{xy}}{s_x^2}=r\cdot\frac{s_y}{s_x}$（参见上文 CORR 及下文 s、$S_{XY}$）。二者标准差可使用 $s_E(a_1)=\frac{s_y}{s_x}\sqrt{\frac{1-r^2}{n-2}}$ 和 $s_E(a_0)=s_E(a_1)\cdot\sqrt{\frac{n-1}{n}s_x^2+\bar{x}^2}$ 计算。 通常，回归系数在 $\frac{\lvert a_i\rvert}{s_v(a_i)}>t_{n-2}^{-1}(0.995)$ 时是显著的，其中式子右侧为自由度为 n-2，置信度为 99%的逆 t 分布。
S_{XY}	对于线性拟合模型，其样本协方差为 $s_{xy}=\left(n\sum x_iy_i-\sum x_i\sum y_i\right)/[n\cdot(n-1)]$。参见上文的 COV。
s，SERR	样本方差为 $s_x^2=\frac{n\sum x_i^2-\left(\sum x_i\right)^2}{n(n-1)}=\frac{\sum x_i^2-n\bar{x}^2}{n-1}$。 样本标准偏差（SD）为 $s_x=+\sqrt{s_x^2}$。 标准误差（既平均值 $\bar{x}$ 的样本标准差）为 $s_{Ex}=\frac{s_x}{\sqrt{n}}$。
s_w，$SERR_w$	有加权的数据的样本标准差（对于每个数据点 x_i 的权重 y_i，通过**[Σ+]**输入）为 $s_w=+\sqrt{\frac{\sum y_i\cdot\sum\left(y_i\cdot x_i^2\right)-\left[\sum\left(y_i\cdot x_i\right)\right]^2}{\left(\sum y_i-1\right)\cdot\sum y_i}}$。 对应的标准误差（平均数 $\bar{x}_w$ 的标准差）为 $s_{Ew}=+\frac{1}{\sum y_i}\cdot\sqrt{\frac{\sum y_i\cdot\sum\left(y_i\cdot x_i^2\right)-\left[\sum\left(y_i\cdot x_i\right)\right]^2}{\sum y_i-1}}$。
$\bar{x}$	算术平均数（或平均数）：$\bar{x}=\frac{1}{n}\sum x_i$。
$\bar{x}_g$	几何平均数：$\bar{x}_g=\sqrt[n]{\prod x}=e^{\frac{1}{n}\sum\ln x}$。
$\bar{x}_w$	有加权数据（见 s_w）的算术平均数：$\bar{x}=\frac{\sum x_iy_i}{\sum y_i}$。
ε	对数正态分布的样本分布系数：$\ln(\varepsilon_x)=\sqrt{\frac{\sum\ln^2(x_i)-2n\cdot\ln(\bar{x}_g)}{n-1}}$。参见上文 s。
ε_m	几何平均数的分布系数为 $\varepsilon_m=\varepsilon^{\frac{1}{\sqrt{n}}}$。参见上文 SERR。
ε_p	对数正态分布的总体分布系数：$\ln(\varepsilon_p)=\sqrt{\frac{n-1}{n}}\cdot\ln(\varepsilon_x)$。参见下文 σ。
σ	正态分布的总体标准差为 $\sigma_x=\frac{1}{n}\cdot\sqrt{\sum\left(x_i-\bar{x}\right)^2}=s_x\cdot\sqrt{\frac{n-1}{n}}$。

名称	说明（通用信息参见 77 页）
σ_w	有加权数据（参见上文 s_w）总体标准差为 $\sigma_w = +\sqrt{\frac{\sum y_i \left(x_i - \overline{x}_w\right)^2}{\sum y_i}}$ 。

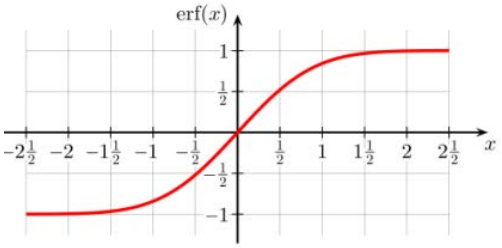
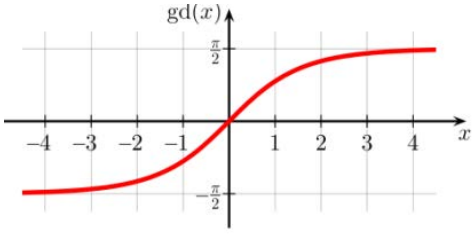
正交多项式

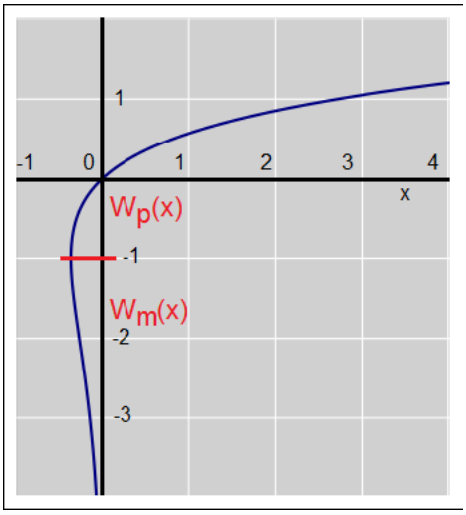
下表列出了 X.FCN 菜单中所包含的全部多项式。

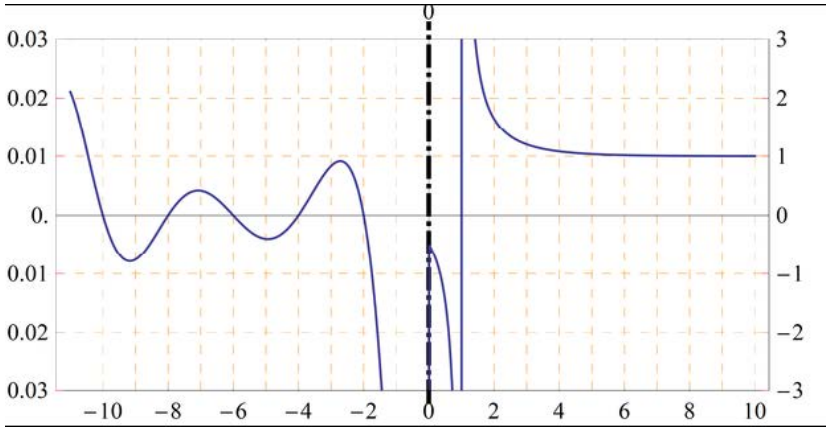
名称	说明（通用信息参见 77 页）
H_n	(2)概率论中的埃尔米特多项式：$H_n(x)=(-1)^n\cdot e^{\frac{x^2}{2}}\cdot\frac{d^n}{dx^n}\left(e^{-\frac{x^2}{2}}\right)$，***n*** 由 **Y** 给出。它是微分方程 $f''(x)-2x\cdot f'(x)+2n\cdot f(x)=0$ 的解。
H_{np}	(2)物理学中的埃尔米特多项式：$H_{np}(x)=(-1)^n\cdot e^{x^2}\cdot\frac{d^n}{dx^n}\left(e^{-x^2}\right)$，***n*** 由 **Y** 给出。它也是上面同一个微分方程的解。
L_n	(2)拉盖尔多项式（参见 $L_n\alpha$）：$L_n(x)=\frac{e^x}{n!}\cdot\frac{d^n}{dx^n}\left(x^n e^{-x}\right)=L_n^{(0)}(x)$，***n*** 由 **Y** 给出。它是微分方程 $x\cdot f''(x)+(1-x)f'(x)+n\cdot f(x)=0$ 的解。
$L_n\alpha$	(3)拉盖尔广义多项式（参见上文 L_n）：$L_n^{(\alpha)}(x)=\frac{x^{-\alpha}e^x}{n!}\cdot\frac{d^n}{dx^n}\left(x^{n+\alpha}e^{-x}\right)$，***n*** 由 **Y** 给出。
P_n	(1)勒让德多项式：$P_n(x)=\frac{1}{2^n n!}\cdot\frac{d^n}{dx^n}\left[\left(x^2-1\right)^n\right]$，***n*** 由 **Y** 给出。它是微分方程 $\frac{d}{dx}\left[\left(1-x^2\right)\cdot\frac{d}{dx}f(x)\right]+n(n+1)f(x)=0$ 的解。
T_n	(2)切比雪夫（也叫 Chebychev，Čebyšev，Tschebyschow，Tschebyscheff）第一多项式 $T_n(x)$，***n*** 由 **Y** 给出。它是微分方程 $\left(1-x^2\right)f''(x)-x\cdot f'(x)+n^2\cdot f(x)=0$ 的解。
U_n	(2)切比雪夫第二多项式 $U_n(x)$，***n*** 由 **Y** 给出。它是微分方程 $\left(1-x^2\right)f''(x)-3x\cdot f'(x)+n(n+2)f(x)=0$ 的解。

更多数学函数

下表所列函数均可在 X.FCN 菜单中找到，它们中的一些是纯数学理论上才用到的，但它们在 *WP 34S* 开发的一些阶段中也很实用，所以我们使其成为可访问函数。

名称	说明（通用信息参见 77 页）
AGM	(2)返回算术几何平均数。可从此获取更多信息：*http://mathworld.wolfram.com/Arithmetic-GeometricMean.html*。
Erf	(1)返回误差函数。 $erf\left(x\right)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-\tau^2}d\tau$。 注意，$erf\left(\frac{x}{\sqrt{2}}\right)=2\cdot\Phi\left(x\right)-1$，其中 $\Phi\left(x\right)$ 在 201 页中已描述过，表示标准正态 *cdf*。 
Erfc	(1)返回互补误差函数 $erfc\left(x\right)=1-erf\left(x\right)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-\tau^2}d\tau$。此函数同标准正态分布的错误概率相关。
g_d	(1)返回古德曼函数 $g_d(x)=\int_0^x\frac{d\xi}{\cosh\xi}$。 它连接了双曲函数及三角函数。右图绘制了其实部的值。反古德曼函数为 $g_d^{-1}(x)=\int_0^x\frac{d\xi}{\cos\xi}$。可从此获取更多信息：*https://en.wikipedia.org/wiki/Gudermannian_function*。 
I_X	(3)返回正则化 Beta 函数 $\frac{\beta_x\left(x,y,z\right)}{\beta\left(y,z\right)}$，其中 $\beta\left(y,z\right)$ 为欧拉 Beta 函数（参见 104 页），而 $\beta_x\left(x,y,z\right)=\int_0^x t^{y-1}\left(1-t\right)^{z-1}dt$ 是不完全 Beta 函数。

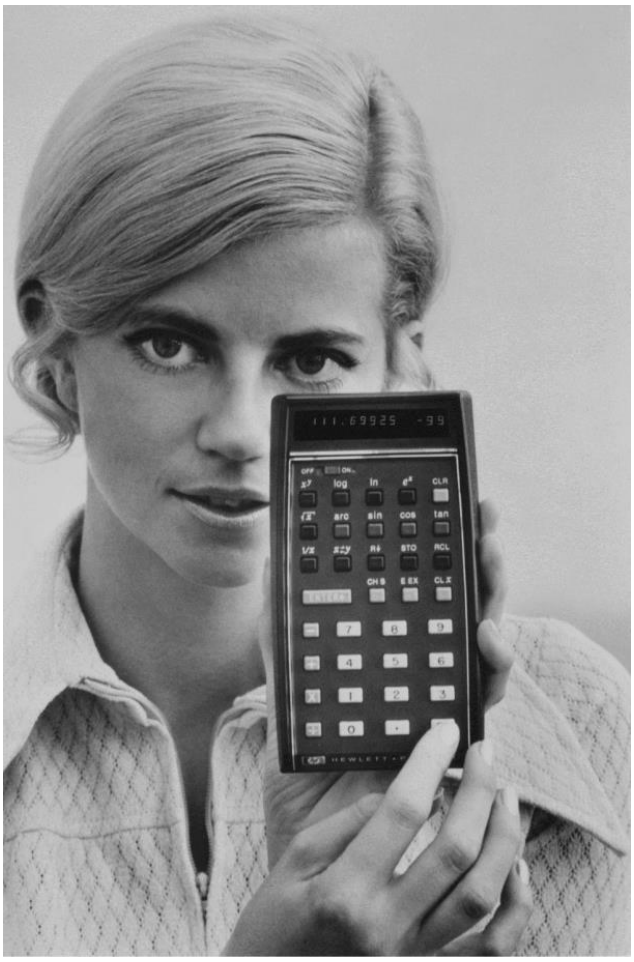
名称	说明（通用信息参见 77 页）
W_p W_m	(1)返回朗伯 W 函数的主分支（此处称其为 W_p）及其负分支（此处称其为 W_m，m 代表负数（minus））。二者分界点位于 $\left(-\frac{1}{e},-1\right)$。右图展示了两组分支的实部值。 可从此获取更多信息：*https://en.wikipedia.org/wiki/Lambert_W_function* 及 *http://mathworld.wolfram.com/LambertW-Function.html*。 
β	(2)返回欧拉 Beta 函数 $B(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$，其中 Re($x$)>0，Re($y$)>0。为了避免混淆，这里称为 β 函数。
Γ	(1)返回 Γ 函数： $\Gamma(x)=\int_0^{+\infty}t^{x-1}e^{-t}dt$，其中 Re($x$)>0。
Γ_p	(2)返回正则化伽马函数 $P(x,y)=\frac{\gamma(x,y)}{\Gamma(x)}$。对于 $\gamma(x,y)$，参见 γ_{XY}。
Γ_q	(2)返回正则化伽马函数 $Q(x,y)=\frac{\Gamma_u(x,y)}{\Gamma(x)}$。对于 $\Gamma_u(x,y)$，参见 Γ_{XY}。
γ_{XY}	(2)返回下不完全伽马函数 $\gamma(x,y)=\int_0^{y}t^{x-1}e^{-t}dt$。$\Gamma_p$ 的计算需要用到此函数。
Γ_{XY}	(2)返回上不完全伽马函数 $\Gamma_u(x,y)=\int_y^{\infty}t^{x-1}e^{-t}dt$。$\Gamma_q$ 的计算需要用到此函数。

名称	说明（通用信息参见 77 页）
ζ	(1)对实数参数返回黎曼 Zeta 函数。对于 *x*>1 的情况，$\zeta\left(x\right)=\sum_{n=1}^{\infty}\frac{1}{n^{x}}$，而对于 *x*<1 的情况，其解析延拓为：$\zeta\left(x\right)=2^{x}\pi^{x-1}\sin\left(\frac{\pi}{2}x\right)\cdot\Gamma\left(1-x\right)\cdot\zeta\left(1-x\right)$。  可从此获取更多信息：*http://mathworld.wolfram.com/RiemannZetaFunction.html*。

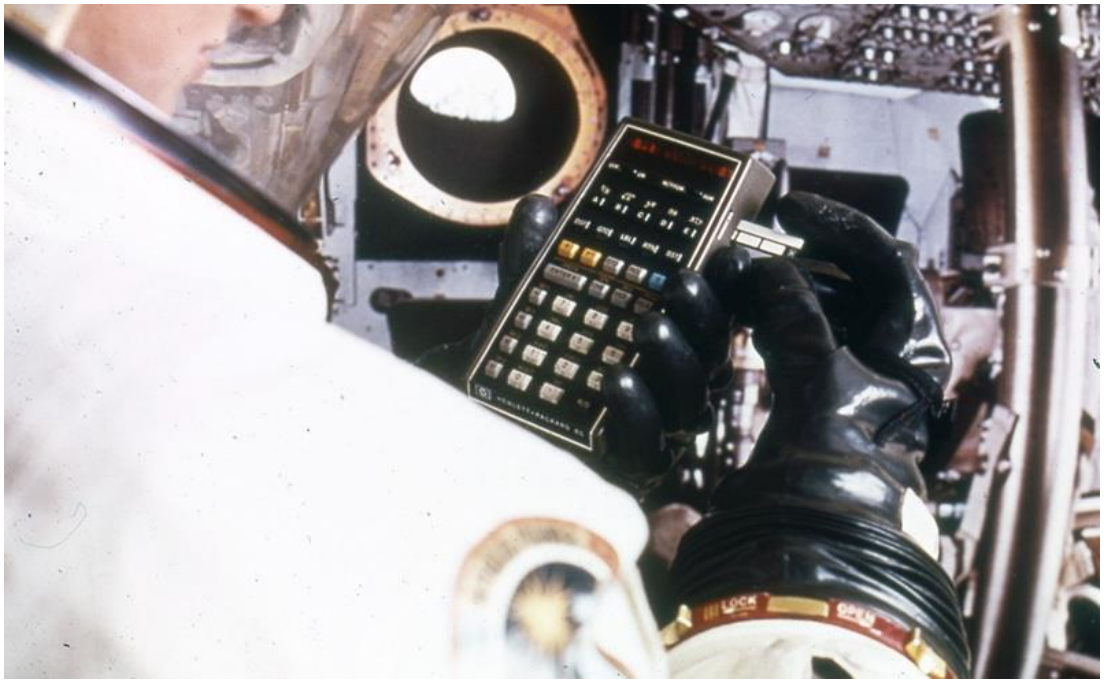
此外，还能在互联网上找到有关 *WP 34S* 所实现的特殊功能的大量相关信息。通常，对于入门者来说，使用维基百科是个良好的开始——建议阅读不同语言版本的文章，因为它们可能包含不同的资料并使用不同的叙述方式。Mathworld 网内有更多资料。对于应用统计类知识，上文所引用的 NIST Sematech 在线手册是有效的资料来源。在这些网站还可以找到更多参考资料。

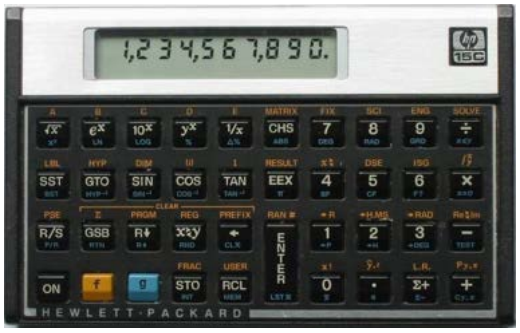
附录 J RPN 袖珍科学计算简史

右图显示的是 HP-35，它是世界上第一台可以计算超越函数的袖珍计算器，在 1972 年推出后，使得计算尺成为过时产品。下图是 1974 年推出的世界上第一台可编程袖珍计算器 HP-65。

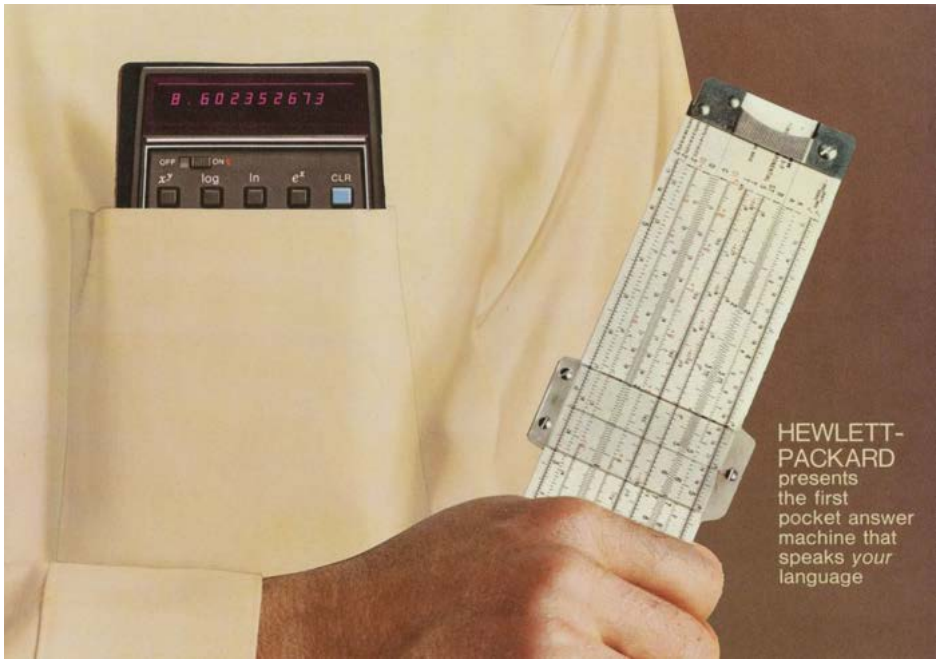


其他的经典型号有 1975 年的 HP-25C（第一个带有 CMOS 存储器），1979 年的 HP-34C（第一个带求解和积分）和 HP-41C（第一个开放系统，图见下页），1982 年的 HP-15C（第一个支持矩阵，图见下页）和 1988 年的 HP-42S。HP-42S 的生产于 1995 年停止。





下图和下一页的图，展示了两则惠普公司为第一台电子科学袖珍计算器所做的广告。要知道，即使是登月任务也还在使用计算尺——电影《阿波罗 13 号》中就有很好的反映。第一台被带入近太空的电子袖珍计算器是 1975 年的 HP-65（也见上页图）。



Careful human engineering assures convenient, reliable operation

The HP-35 features a bright, easy-to-read light-emitting diode display specially developed for this application by Hewlett-Packard's own opto-electronic design group. Since the displays are made from semi-conductor materials, they—like the large-scale integrated circuits used inside the HP-35—do not wear out with time.

The unique HP-35 keyboard has been carefully laid out for convenient fingertip operation. Because the keys are small and widely separated, there is less chance of misentering data by striking two keys simultaneously. Moreover, each key has a "breakaway" or "overcenter" touch similar to the key action for a high-quality electric typewriter to give a positive indication that contact has been made.

Finally, the compact, contoured HP-35 case represents a breakthrough in modern packaging techniques. Small enough to fit in a shirt pocket, rugged enough to withstand substantial punishment in field use, the case gives the HP-35 a comfortable "feel" and makes the calculator even more convenient to use.

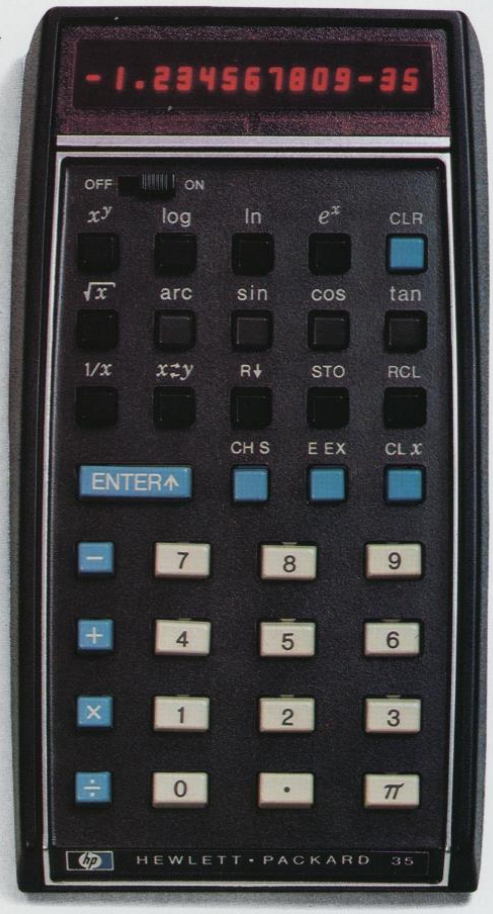
HP-35 The first pocket calculator designed to fit the needs of today's engineering/scientific world.

Easy-to-read LED display gives answers with accuracy of up to 10 significant digits. Decimal point is automatically positioned. Numbers smaller than 10^{-2} and larger than 10^{10} are displayed in scientific notation, with the exponent of 10 (plus or minus) shown at the extreme right.

Pre-programmed functions let you perform log, trig and exponential calculations 10 times faster than with a slide rule.

Operates on battery or AC-line power.
The HP-35 comes with a rechargeable nickel-cadmium battery pack which provides enough power for three to five hours of continuous operation. Impending battery depletion is indicated by the lighting of all decimal points in the display.

AC-line operation (115 or 230 volts) and simultaneous battery charging are accomplished by simply plugging in the battery charger supplied with the HP-35.

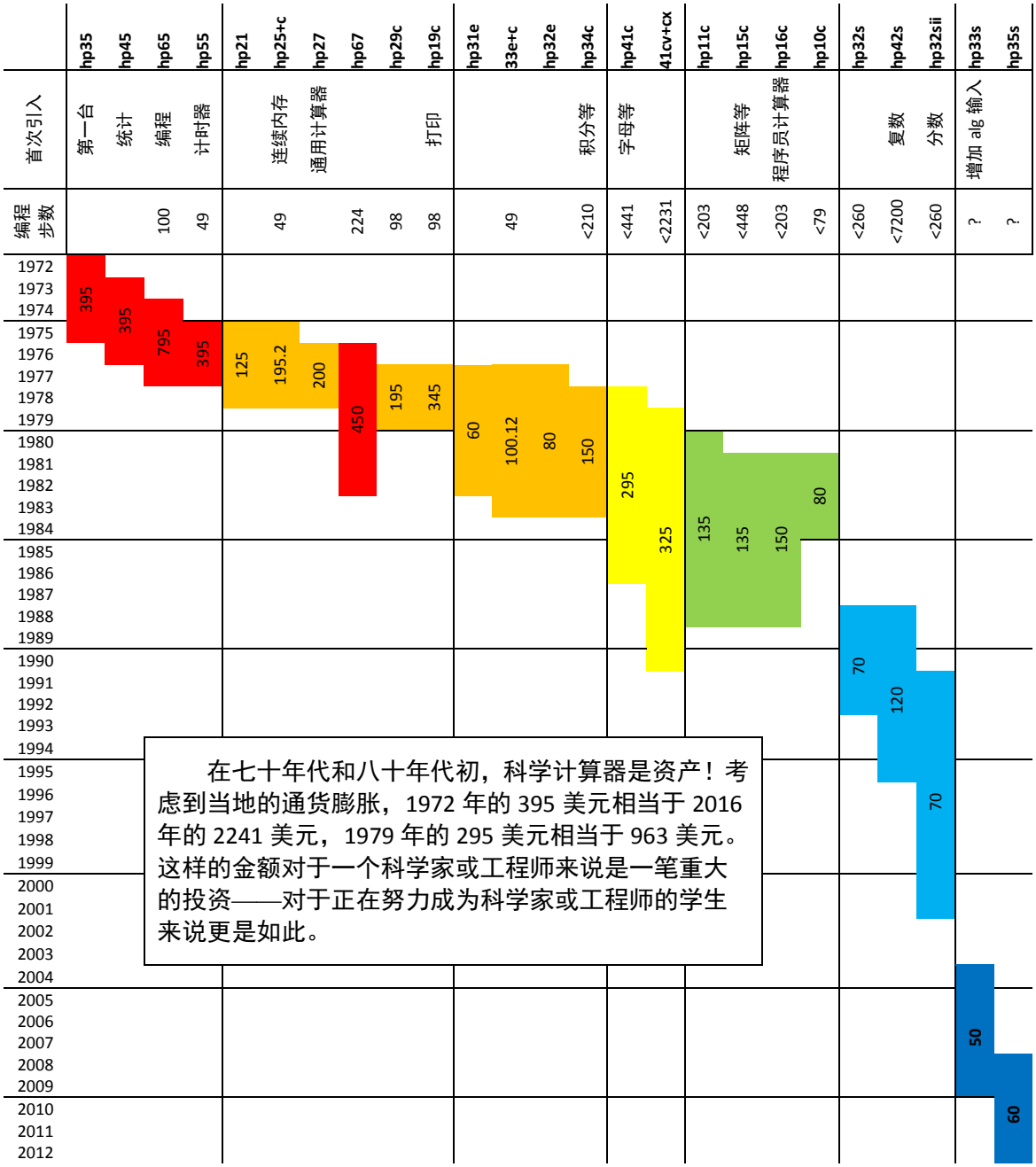


Compact, rugged case fits easily into shirt pocket or briefcase.

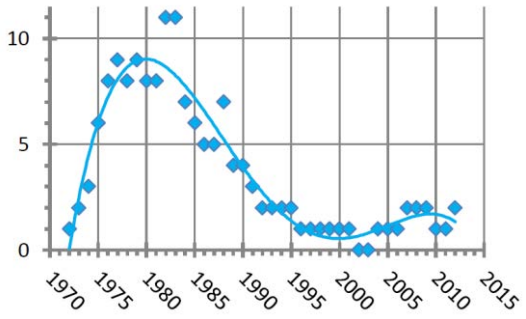
Four-register, operational memory-stack automatically stores and retrieves intermediate solutions during complex calculations, eliminates need for scratch notes and re-entry of data. Data manipulation keys allow contents of any register to be displayed for review. A separate, addressable memory register is also provided for storage and retrieval of constants.

Try it for 15 days
Shirt-pocket test the powerful HP-35 scientific pocket calculator for 15 days to see how much time and money it can save you. To take advantage of this offer, or obtain information on the complete line of HP pocket and desktop calculators, contact your local HP sales office. Or, write us at the address shown on the back page of this brochure.

惠普公司生产的所有 RPN 科学计算器的时间轴请看下页。彩条内的数字表示进入市场时的价格（美元）。垂直线将产品线分开：Classic 系列，Woodstock 系列，Spice 系列，Nut 系列，Voyager 系列和 Pioneer 系列。



在七十年代和八十年代初，科学计算器是资产！考虑到当地的通货膨胀，1972 年的 395 美元相当于 2016 年的 2241 美元，1979 年的 295 美元相当于 963 美元。这样的金额对于一个科学家或工程师来说是一笔重大的投资——对于正在努力成为科学家或工程师的学生来说更是如此。



左图是 1972 年以来，惠普公司生产和为惠普公司生产的不同型号的 RPN 科学袖珍计算器的数量。线条用于观察趋势[113]。

在七十年代和八十年代早期，也有个别其他公司生产了一些 RPN 袖珍计算器（如 Commodore，Novus，Omron 和 Sinclair）。

[113]惠普在 1987 年推出了 RPL 计算器。从那时起直到今天，它们的体积都明显大于所有的 RPN 型号，从而使“袖珍计算器”一词变得很牵强。这张图中不包括它们。

为什么堆栈和回车键的工作方式是这样的？

惠普公司的第一台（台式）电子计算器，1968 年的 9100A 型，只有 3 级堆栈——但所有 3 级堆栈都显示在一个小阴极射线管的屏幕上：键盘（输入）寄存器 **X**，累加器 **Y** 和临时寄存器 **Z**。

以前的机械式计算器只有一个键盘寄存器和一个累加器。要让计算器能够处理稍微高级的数学表达式，还需要第三个寄存器。

为了解决像 123/(45+67)这样的问题，必须执行以下步骤（从清除堆栈开始），如这里每一栏所显示：

Z	0	0	0	123	123	123	123	123
Y	0	123	123	45	45	112	123	1.0982…
X	123	123	45	45	67	67	112	112
输入	123	**[⇧]**[114]	45	**[⇧]**	67	**[+]**	**[⇩]**[115]	**[/]**

但由于空间和功率的原因，像 HP-35 这样的早期袖珍计算器只能显示一个寄存器，而这个寄存器必须是 **X**，以便在键盘输入时提供反馈。因此，为了显示在累加器 **Y** 中提供的计算结果，它的内容 ***y*** 必须被移到 **X** 中，因此**[+]**和**[⇩]**必须紧接着执行，最好是在一个操作中（同样适用于其他二元操作）。

另一方面，在 **X** 中这样的中间结果必须在输入下一个数字之前被（再次）移位到累加器 **Y** 中。因此，**[⇧]**是自动执行的。当然，如果**[⇧]**已被按下，这是不需要的。这就是为什么在按下**[ENTER⇧]**后自动堆栈提升被禁用的原因。

T	0	0	0	0	0	0	0
Z	0	0	0	123	123		0
Y	0	123	123	45	45	123	0
X	123	123	45	45	67	112	1.0982…
输入	123	**[ENTER⇧]**	45	**[ENTER⇧]]**	67	**[+]**	**[/]**

自动堆栈提升在**[CLx]**后也被禁用，因为输入的新数字将会简单地取代刚刚删除的数字。还有**[Σ+]**以及**[Σ-]**是输入统计数据点的专用命令——它们不是用来进行堆栈计算的。

[114]这个键的功能在后来计算器上的**[ENTER⇧]**一样。

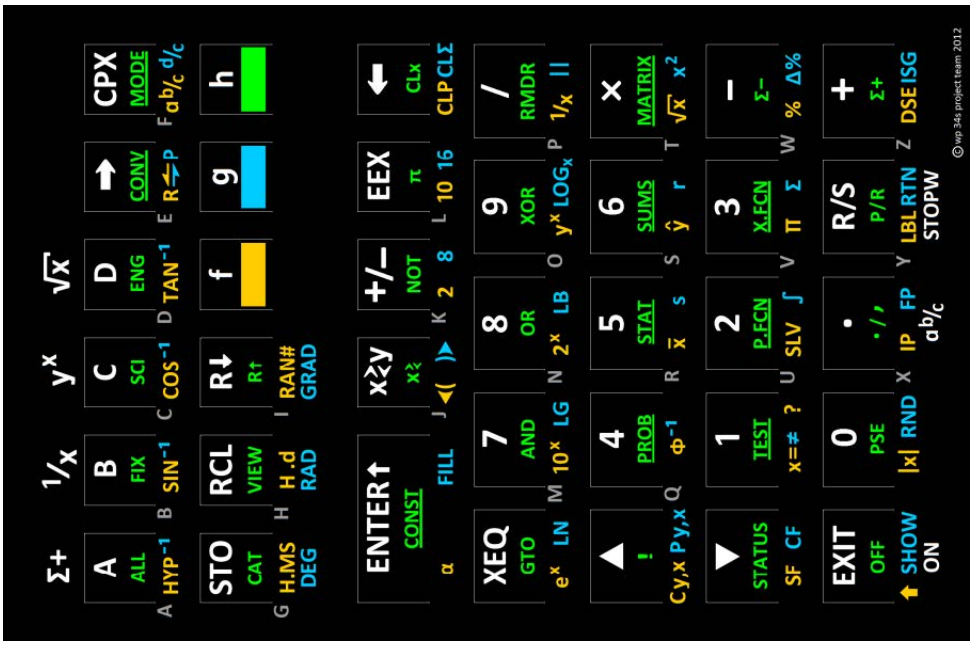
[115]事实上，**[⇩]**做了和你的 *WP 34S* 上的 DROP 一样的功能。由于键盘空间的原因，这个功能在惠普的袖珍计算器上消失了。**[R⇩]**保留了下来——它的功能几乎是一样的。

附录 K 3.3 版本的更新

- 增加了 YDOFF 和 YDON；
- 延长秒表运行时间；
- 将 E3ON/OFF 重命名为 TSON/OFF，Iβ 命名为 I_X，IΓ$_p$ 命名为 Γ$_p$，IΓ$_q$ 命名为 Γ$_q$，并重命名三个功率转换；
- 添加了 ^{C}VIEW；
- 添加了 XEQ。

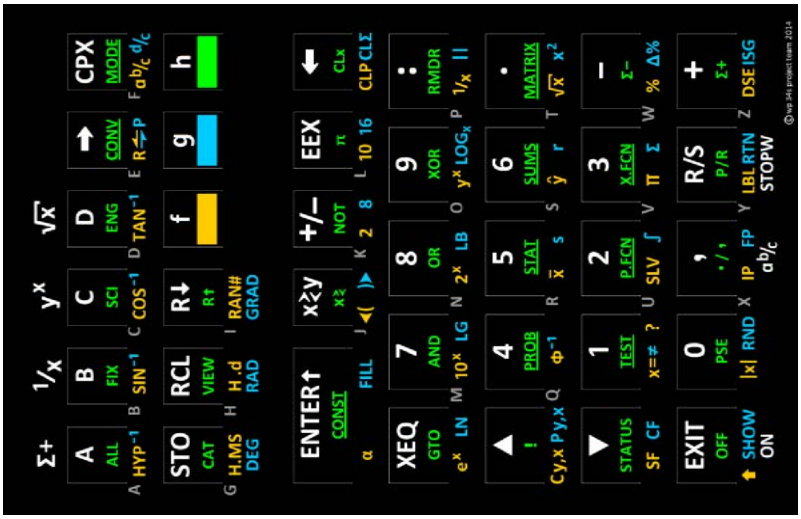
附录 L 自制临时键盘贴

可以用下面的图片自制键盘贴。靠下的一张图是供 v2.2 固件用的。

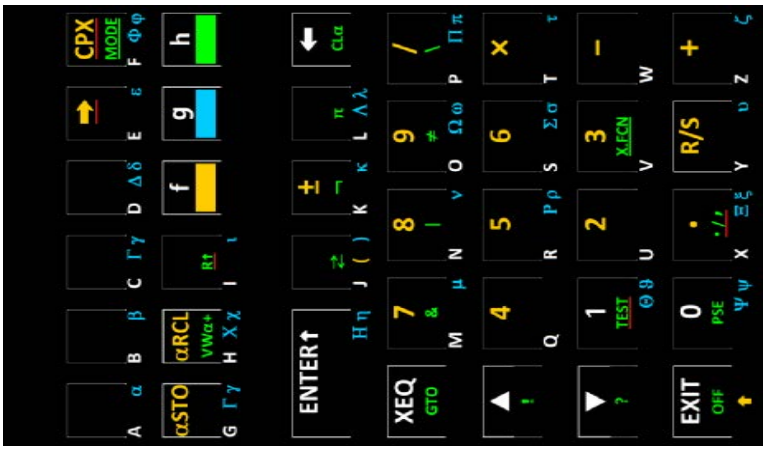


请注意，v2.2 中的 SHOW 的工作方式与老式惠普计算器类似，即它显示完整的有效数字。有关不同版本固件之间的功能差异的详细信息，请参阅英文版手册的更新说明。

以下键盘贴纸比上面的更适合在世界大部分地区使用，因为用的是逗号小数点而不是点号小数点。请注意，这个贴纸也没有使用乘号×：



下图是虚拟 alpha 键盘的贴纸，可将这个图片剪下来，并将其贴到计算器背面，用于熟悉 alpha 模式下的键位。



快速指南

WP 34S 大部分功能和 *HP-42S* 类似，但有几处不同。

一般情况：

•**[STATUS]**用于检查可用内存并查看所有标志位。内存可以自由分配。

•切换显示模式的键包括：

实数（TSOFF/ON，DBLOFF/ON）	[a b/c]，[d/c]⇨ ⇦[H.d]	分数（DENANY、DENFAC、DENFIX、DENMAX）
	[2]，[8]，[10]，[16]⇨ ⇦[H.d]，[ALL]，[FIX]，[SCI]，[ENG]	整数（WSIZE，1COMP，2COMP，SIGNMT，UNSIGN，SEPOFF/ON，LZOFF/ON）
数字模式	[α]⇨ ⇦**[EXIT]**，**[ENTER⇧]**	Alpha 模式
运行模式	⇦ [P/R] ⇨ ⇦**[EXIT]**	程序编辑模式
逗号小数点	⇦ [./,] ⇨	点号小数点
灵活使用点阵	YDOFF⇨ ⇦YDON	始终显示 *y*

在数字模式下：

• 通过 SSIZE4 或 SSIZE8 设置堆栈大小。最多的级别为 **X**、**Y**、**Z**、**T**、**A**、**B**、**C**、**D**。**[FILL]**用 *x* 填充所有级别的堆栈

• **[◀()]**可以显示 *x* 的完整有效数字和指数。

• **[RCL][L]**调出 LASTx

• 最多 100 个通用全局编号寄存器（00...99），没有变量，没有数据类型。堆栈和后面的字母寄存器从寄存器#100 开始。

• 通过**[STO][⇨]**间接寻址。

• **[SHOW]**：查看所有寄存器内容。

• **[CONV]**：查看所有单位换算。

• **[PROB]**：查看所有概率分布。

• **[SUMS]**：访问求和寄存器。

• 矩阵是 *WP 34S* 上的一组寄存器。因此，矩阵处理与 *HP-42S* 有很大不同。

• **[CPX]+[函数]**调用复变函数。每个复数需要 2 个寄存器。

在 alpha 模式下：

- Alpha 寄存器有 30 字节（***alpha***）。
- 大多数键的主要功能是将相应的灰色拉丁字母添加到 ***alpha*** 上。
- [g]+**[字母]**在适用的情况下调出希腊字母。
- 使用[f]+**[CPX]**调用更多字母。
- 使用[⇧]切换大小写。
- [f]+**[数字]**在适用的情况下将数字追加到 alpha。

在编程模式下：

- **[CAT]**查看 *RAM* 和闪存中的所有全局标签。
- **[STATUS]**查看所有全局标志位。
- 剩余的程序步数和所用的程序语句无关。
- 通过 LocR 指令，可为每个程序分配最多 144 个本地寄存器和 16 个本地标志位（取决于内存设置）。通过如**[STO][.]**的操作对本地对象寻址。

在 I/O 模式下：

- 使用 SAVE 进行断电安全备份。使用 LOAD 恢复备份。
- 使用 SEND...将来自 *WP 34S* 的数据发送到计算机上的模拟器；
  使用 RECV 从模拟器接收数据。
- 打印指令包含在**[P.FCN]**中。

如何安装一个时钟晶体和一个打印需要的红外二极管[116]

1 除了 WP-34s，我还用了什么？

我用了一个 16W 的电烙铁，一把热胶枪，一把小小的飞利浦螺丝刀，镊子和一根木牙签。

2 我从 www.conrad.de 买了什么？

1）晶振 MH32768C Bestell-Nr. 156007 0.79EUR。

2）IR-LED 3mm Typ L-934F3C Bestell-Nr. 154394 0.37EUR。

3）KERKO 芯片 0603 NP0 18PF 5% 50V Bestell-Nr. 445644 0.10EUR。

4）Widerstand Kohle 0.25W 5% 390R BF 0207 Bestell-Nr. 403202 0.10EUR。

通过这种配置，我实现了大约 25 厘米的打印范围。使用不同的部件，可以实现更远的范围。凯蒂-瓦瑟曼推荐使用红外二极管 Vishay TSAL4400，范围可达 50cm。这个问题请问电子专家，而不是我。

总的来说，虽然电容非常微小，处理起来相当有难度，但安装是一个焊接技术不高的人也能完成的任务。

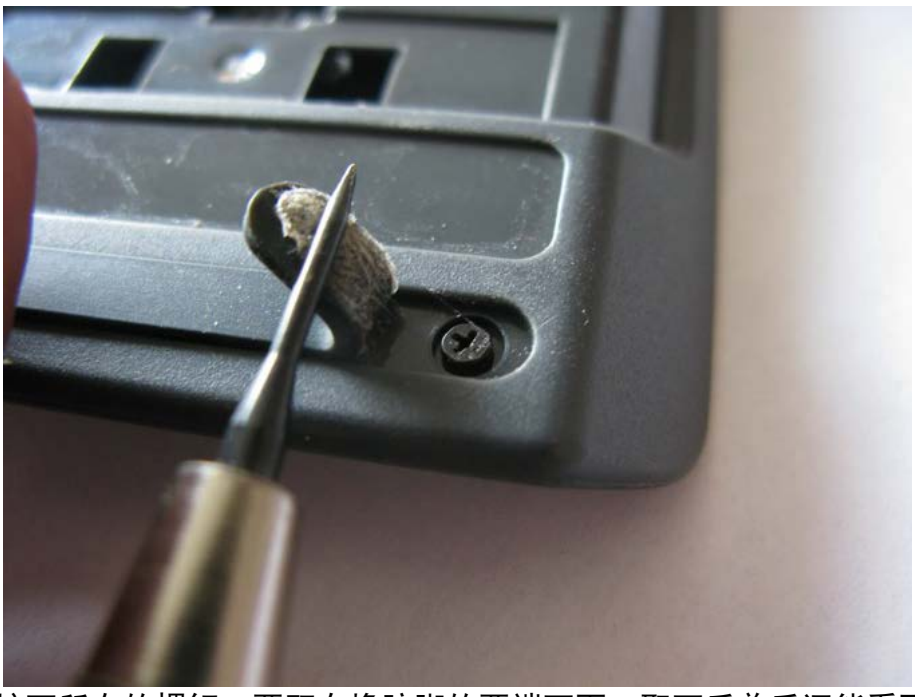


图 1：拧下所有的螺钉。两颗在橡胶脚的两端下面，取下后盖后还能看到三颗。

[116] *Alexander Oestert*，德国，2012 年 5 月 5 日星期六，11:08

图 2：将工具插入银色正面和灰色框架之间，打开；先是侧面，再是上下。我用镊子来完成这个任务。

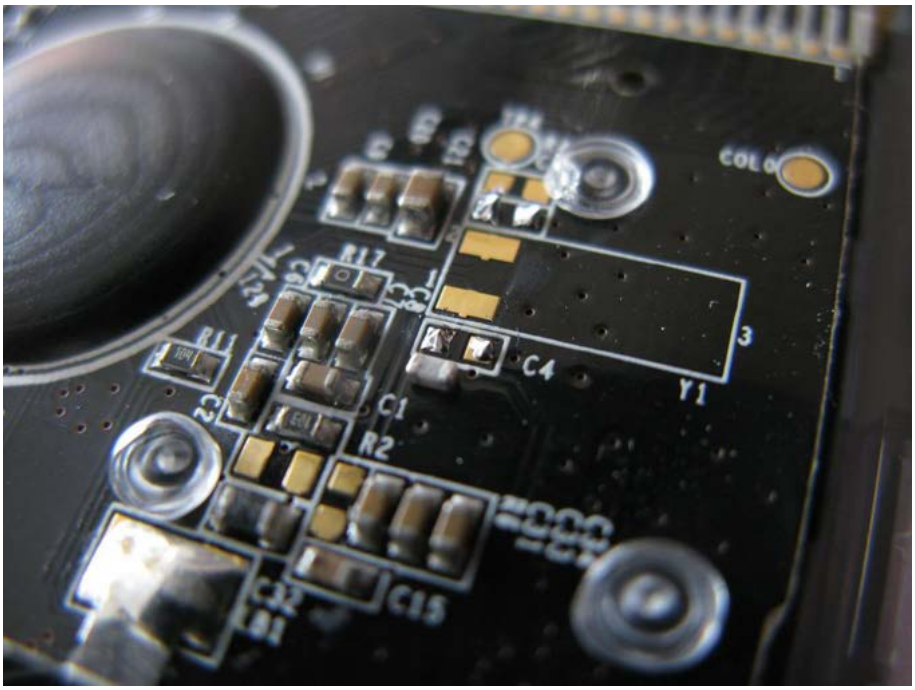


图 3：确定 C3 和 C4 的触点。给它们上焊料。

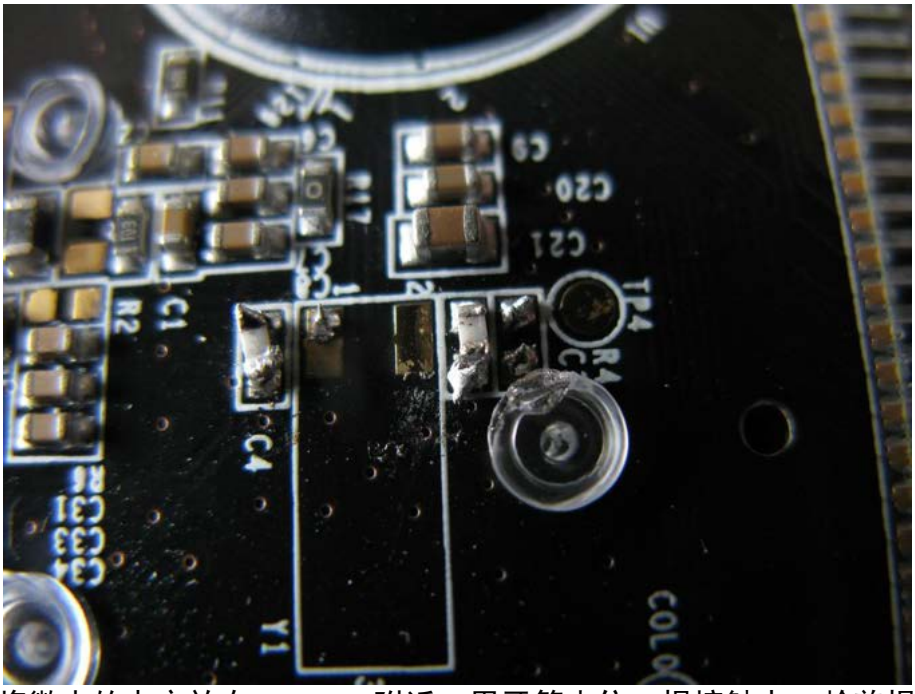


图 4：将微小的电容放在 C3、C4 附近，用牙签夹住，焊接触点，趁着焊锡是液体，用牙签将电容放好。我这个显得外行了。如果你能做得更完美的话，那就更好了!

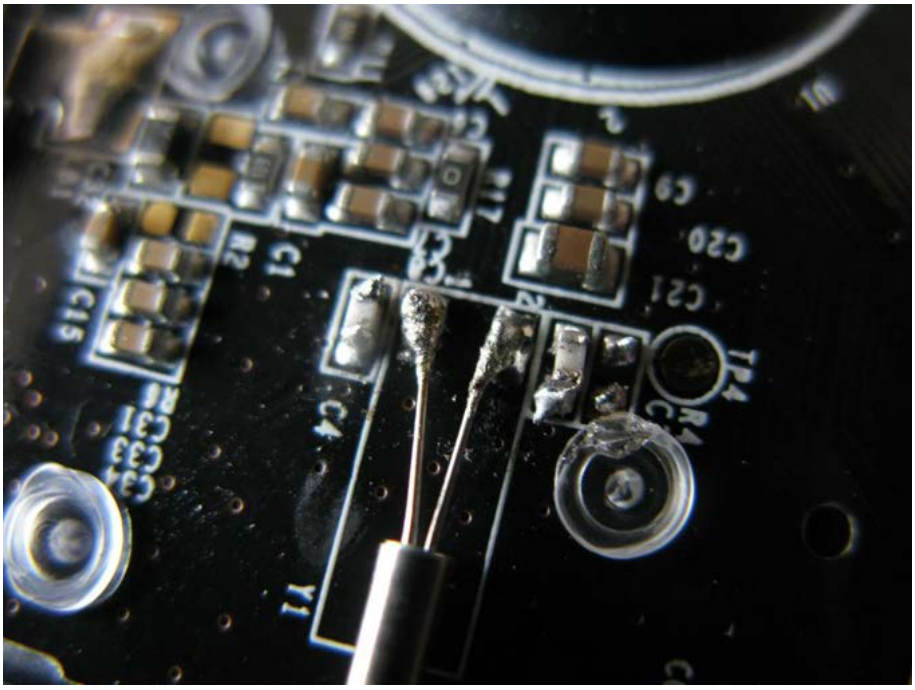


图 5：剪短晶振的管脚，使其处于矩形 Y1 内。将焊料涂在晶振的脚和 PCB 的触点上。在焊接之前，先将晶振的脚适当弯曲，使其能够整体放平。在用镊子或您的手指夹住晶振的同时，将其焊入。

图 6：通过脚和长脚来识别二极管的阳极（长引线）和阴极（短引线）。尽量缩短红外二极管和电阻的管脚，可以比图上的更短，使整体尽可能的紧凑。将电阻焊接到红外二极管原来的长管脚。

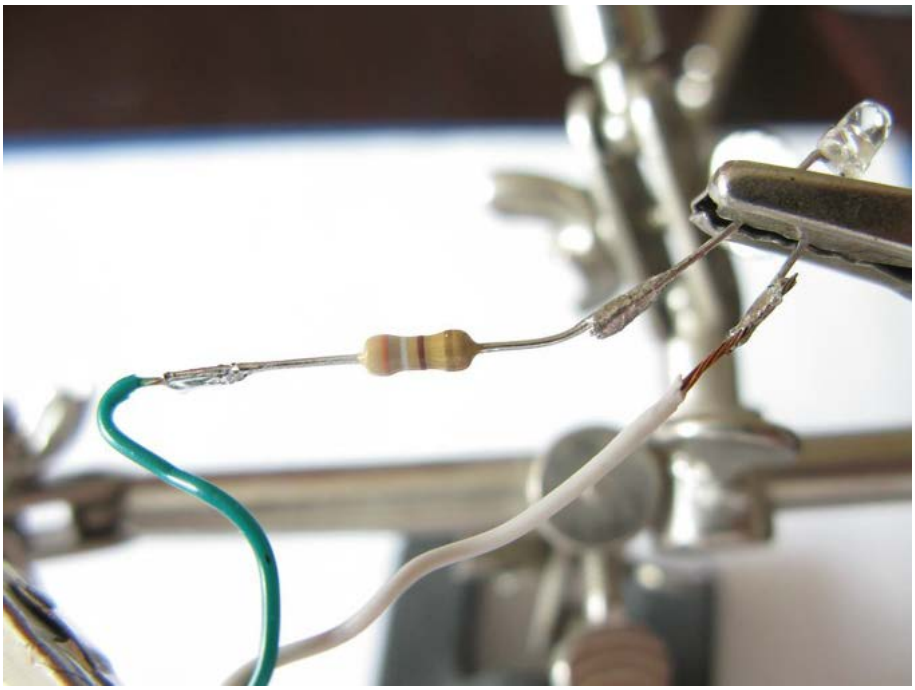


图 7：将两根 10cm 的导线焊接到电阻和短管脚上。建议使用比我细的导线。线越细，它越容易被塞进机身里。如果你喜欢让二极管从侧面伸出来，你还可以用更短的线。

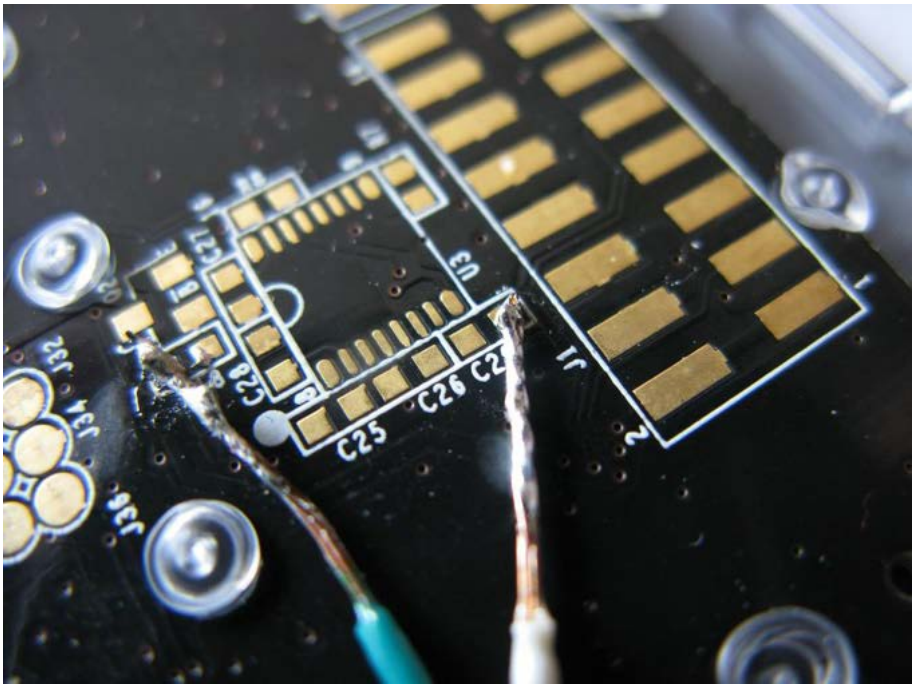


图 8：将长脚的导线焊接到 R18 上，另一根导线焊接到 C29（U3 的角脚）上。在重新组装之前，我把电线用热胶粘在 PCB 和后壳上。

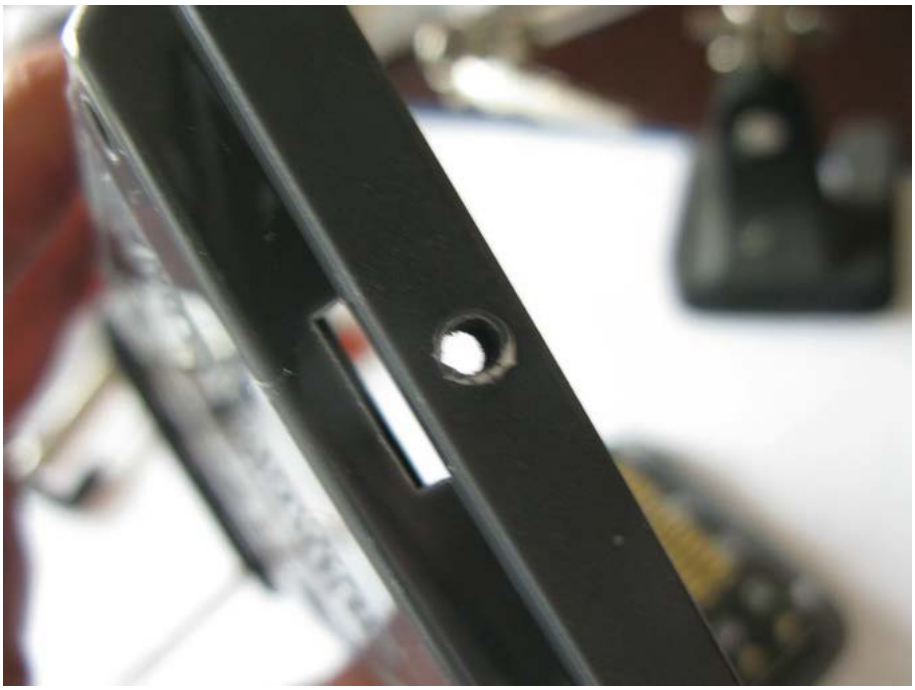


图 9：这是我用电烙铁在后壳上侧熔化的孔——不是后盖。您也可以钻孔。

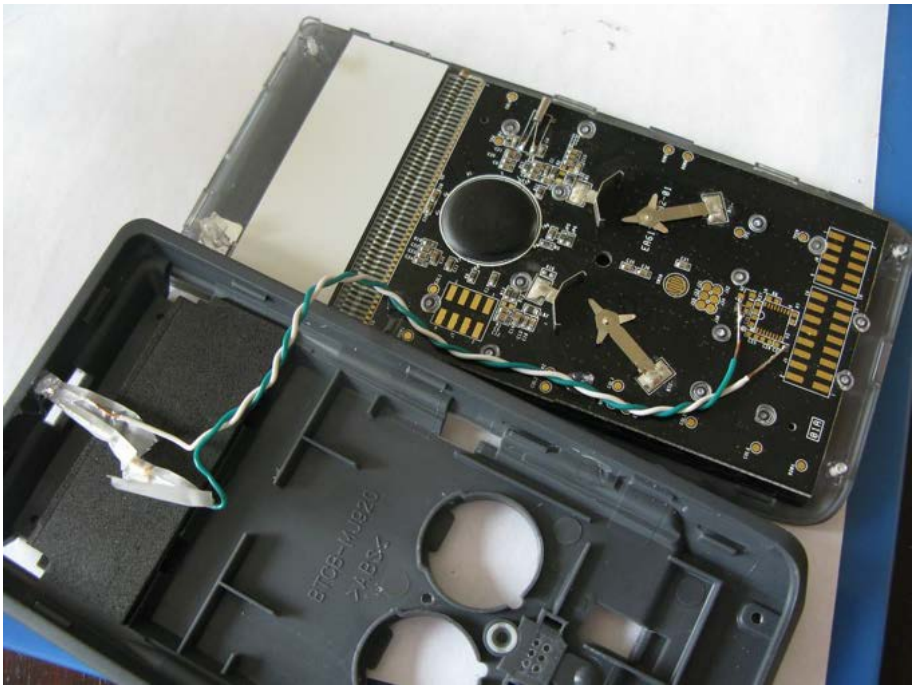


图 10：红外二极管和连接线就位。将它们做得更细或者更短，可以更容易地将它们放在箱子里。如果你有漂亮的收缩管或电工胶带，请务必使用。我只有透明胶带来做这项工作……

图 11：这是我第一台机器损坏的 LCD。我在重装回去时按压了液晶屏，它的后面有电线，这可能是致命的。注意再次组装时不要按压液晶玻璃，尽量按压外框即可。

图 12：我的另一台 WP-34s 中安装了全部组件。在焊接完成后，我才成功地刷入红外固件。这之前刷机不成功，计算器无响应。

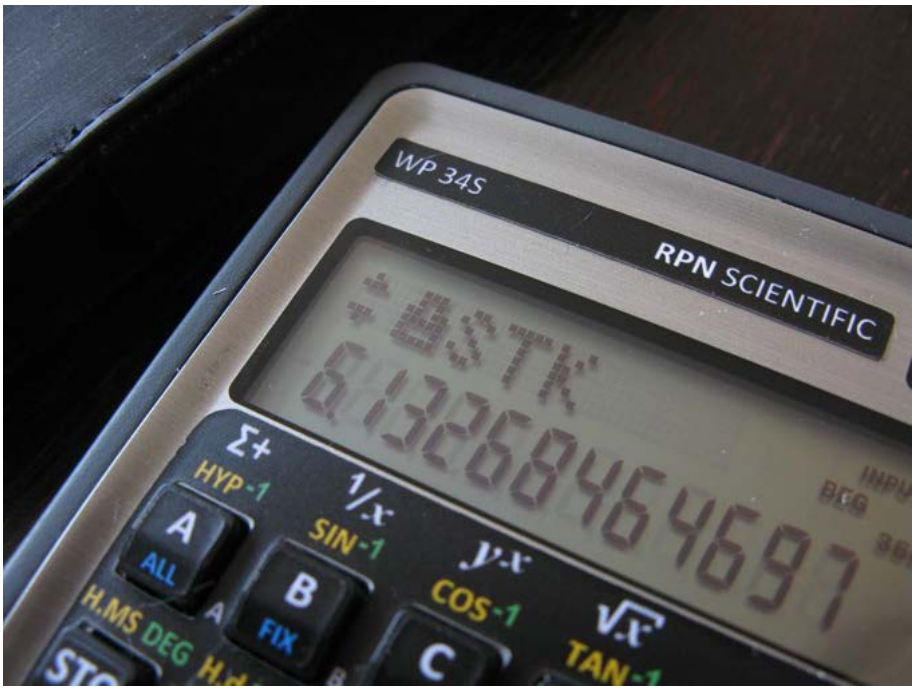


图 13：打印命令位于 P.FCN 菜单的末尾，都以这个漂亮的打印机符号开头，而不是以 P 开头，所以搜索时不要混淆。P.FCN 中的打印机条目可以通过按 f EXIT（小箭头）到达。在菜单之外，这个小箭头可以作为[print]r X 命令的键盘快捷键——便于只打印显示屏上的数字。

图 14：我的红外二极管的最终位置。

译 后 记

WP 34S 翻译项目不知不觉已经过去两年了，咕咕咕…

计算器作为人类发明的计算工具，是人类文明的结晶，爱好者众多。曾几何时，拥有一台计算器就是最酷的事。随着社会发展，昔日昂贵的生产力机器现在也逐渐成为了日常消费品。初中生就摆弄各种计算器，甚至旗舰图形机已经司空见惯，拥有几十台上百台各种计算器的也大有人在。

可惜的是，随着科学技术的发展，在科研和工程市场计算器已经被电脑和各种专业工具取代。计算器教具化，单一化已是大势所趋，小巧灵活又强大的计算器屈指可数。WP 34S 的出现，就像在平静的水面掀起了波澜，受到了各国爱好者的广泛欢迎。在国内也有一些 WP 34S 玩家，其中有人还尝试过翻译用户手册，但未能完成，甚为遗憾。

2018 年 6 月的某天，计算器爱好者交流群提议翻译用户手册，项目就这样开始了。参与翻译项目的成员有中学生，大学生和已经工作的上班族。手册涉及面极广，除了初等代数、线性代数、概率统计和高等数学等数学领域之外，还有计算机科学、物理、天文、历史、地理人文等方面，翻译难度超乎想象，项目进度也经过了几次延期。不过大家还是克服重重困难，一起努力完成了这个项目。大家都有共同的愿景，就是把这款强大的机器介绍给更多的人！感谢大家的辛苦付出!特别感谢项目技术顾问 Andy Lithia 为翻译的科学性和严谨性作出的贡献！

感谢我的妻子，奉献出自己的时间带娃做家务，为我专心参与翻译项目腾出了时间。感谢我的小宝贝添添给我带来的欢乐。

“一支独放不是春，百花齐放春满园。”

Roy Wan

2020 年 6 月于安徽天峡山庄

Discipulī Pȳthagorae in disputātiōnibus saepe dīcēbant: "Ipse dīxit!"

——Alias Qli

我们抛弃了梦想与功名，只留下回忆相伴。

——葛炳仑

不仅仅是 WP 34S 非凡的性能，一群爱好者素未谋面而奇迹般地创造出它的过程同样令人振奋。

——武平

不仅是熟悉的味道，更是经典的重生。

——我的 8086 电脑

祝你的 WP 34S 使用愉快。

——温明翰

做了一些微小的工作，谢谢大家。

——刘傲

You are your time.

——Rudal

虽然不是很习惯逆波兰表示，但是休利特-帕卡德还是很经典。

——Beta Vulgaris

工欲善其事，必先利其器。

——柜里猫

你的下一台计算器，还是计算器。

——滚来滚去的巧胖

与其给我爱，金钱和名誉，不如给我 WP 34S。

——Tselung Soong

Index 到此一游。

——N77012

金属光泽的外壳和锋利的小刀。

——Chienway Tsai

年方十六，失恋时译，祝君幸福。

——朱嘉诚

计算器爱好者交流群

群号：435067924

RPN calculator club

群号：812224709