赵恒友　著

微机软件二次开发实用技巧

（增订本）

- DEBUG 的改进和功能扩展
- CCDOS 汉字系统的二次开发
- CCBIOS 2.13H 的二次开发

- 汉字编码字典打印程序
- 数据库系统的二次开发
- 汉字输入模块的改进、移值、利用

电子科技大学出版社

微机软件
二次开发实用技巧

（增订本）

赵恒友　著

电子科技大学出版社

内 容 提 要

本书从应用的角度出发，详细介绍了对微机软件进行二次开发的诸多技巧。作者融技巧介绍与实用性于一体，提供了对 DOS 工具软件 DEBUG、COMP、WS 以及五笔字型、CCDOS、2.13H 等汉字系统的功能扩展、纠错、改进、移植等几十个实例；介绍了如何利用汉字系统的资源，实现各种汉字编码字典的打印；提供了西文软件汉化的方法和实例，CDBASE Ⅱ的全解密和空库、宏代换的应用技巧，并特别增加了非常有实用价值的"汉字输入模块的改进、移植和利用"等内容。还提供了 FoxBASE⁺通用动态制表程序，以及彻底解决汉字打印自动换页问题的程序。

本书所有实例都有很强的通用性和实用性，并给出了详细的操作步骤和完整的程序清单。通过此书读者既可学到二次开发的种种技巧，又可解决实际工作中的难题，还可获得二次开发后的软件。

微机软件二次开发实用技巧(增订本)

赵恒友 著

*

电子科技大学出版社出版

(成都建设北路二段四号)邮编 610054

四川郫县唐昌印制厂印刷

新华书店经销

*

开本 787×1092 1/16 印张 26 字数 640 千字

版次 1995 年 7 月第二版 印次 1995 年 7 月第三次印刷

印数 8001—13000 册

ISBN 7-81016-601-8/TP · 48

定价：28.00 元

前　言

当今，微型计算机已在各行各业的生产、科研、教学活动中得到了越来越广泛的应用，功能各异的各种微机实用软件也纷纷涌现。了解这些软件的使用与维护固然重要，而针对用户使用实际对软件进行纠错、改进、功能扩充和环境适应性改造等也是十分必要的。这些工作，对软件的开发者而言，属于软件维护的范围；反之，若以上工作由用户自己来完成，对用户而言，则是对该软件进行二次开发。

如何去发现和纠正软件中的错误，对软件中那些不尽人意的地方，怎样去完善和改进，怎样为软件扩充功能，怎样让手中的软件去适应改变了的软硬件环境，等等一切，都是本书要讨论的问题。

本书通过大量实例，力图使读者掌握软件二次开发中的一些实用技巧，并且通过实际操作以增加感性认识，以便读者能够自己尝试解决现在或今后遇到的一些难题。

融技巧介绍与实用性为一体是本书主要特点。以对调试工具软件 DEBUG 的二次开发为例，即是从最简单的一字节汉化 DEBUG 谈起，到扩充反"反跟踪"功能，由浅入深，循序渐进，直至形成高级调试工具 DEBUGA。在此过程中，又详细地介绍了汉化 DEBUG 和为 DEBUG 扩充各种功能的思路、技巧以及每一步的具体实施过程。这样，读者既可掌握这些开发的思路和技巧，又可按书中的操作步骤，获得一个实用的高级调试工具软件。

本书基本上是以汇编语言为主，由于汇编语言难读、难懂的局限，考虑到读者阅读的方便，故绝大多数汇编语言程序都加了汉字注释。尽管如此，要完全理解作者的意图，还是需要耐心和细致的，这一点特提请读者注意。

由于本书内容涉及面较广，加之成书时间紧迫以及作者本人的水平、精力所限，差错在所难免，恳请读者指正。书中所有二次开发后的实用程序，欢迎读者再优化、再提高，十分高兴与乐于此道的朋友交流切磋。

谨以此书献给有志于微机软件开发的青年朋友们。

赵恒友

一九九三年九月于成都

目　录

第一章　软件的二次开发

第二章　二次开发的工具软件

第三章　DOS 外部命令的二次开发与扩充

第四章 CCDOS 汉字系统的二次开发

第五章 五笔字型汉字输入系统的二次开发

第六章 2.13H 汉字系统的二次开发

第七章 汉字编码字典打印

第八章 西文软件汉化方法和汉化实例

第九章 数据库系统的二次开发

第十章 汉字输入模块的改进、移植和利用

第一章　软件的二次开发

§1.1　二次开发的必要性

软件生命期的前五个阶段为软件的开发期，最后一个阶段称之为软件的维护期。在这个期间，软件已交用户使用。但是，软件开发者仍将继续对该软件作维护性工作，主要包括以下内容：

1. 纠错性维护

任何一个软件，不能说它绝对无错，只能说至今未发现错误。交付用户使用的软件，虽然进行过各种测试，但有可能仍然存在错误，只不过未测试出来罢了。所以，纠正错误是软件维护期中的首要工作。

2. 完善性维护

根据用户对软件使用情况的反馈信息，以及开发者自我感觉到的应予完善的功能，对软件中的一些不尽人意的地方加以改进。完善性维护还包括功能扩充，功能扩充是根据用户在实际使用过程中遇到的问题提出来的。

3. 适应性维护

软件在使用过程中，由于硬件环境的改变，或者支撑该软件的软件有变化，这时必须作适应性的修改，使软件在新的环境下能够正常运行。

软件"维护"是针对软件开发者而言；反之，若上述工作由软件的使用者（用户）来完成，那么，对使用者来说就是对该软件作二次开发了。分析软件维护的三项工作，无一不与用户有关：错误大都是用户发现的，功能不完善是用户首先感觉到的，需要扩充功能是用户提出来的、软硬件环境的改变引起的不适配是用户首先体验到的。如果用户具有二次开发的能力，由他们自己来解决自己发现的问题；自己来实现自己提出的种种功能扩充的设想，这一切由于是用户完全从自己的实际需要提出来的，所以它能更加贴近用户，使软件功能不断完善和提高。而软件的原开发者，完全可以从这些二次开发中吸取精华，集思广益，作用于该软件的高版本中。

二次开发不仅仅包含软件维护的工作，移植和植入也是二次开发不可缺少的重要组成部分，它们往往能达到事半功倍的效果。

软件的二次开发还具有以下意义：

1. 由于软件本身是一种永不磨损的产品，不存在用旧、用坏的概念（软件的载体会用坏，但并不能说是软件用坏了），通过不断的改进、扩充，软件质量将不断提高。

2. 商品化的"通用"软件，由于用户特性的差别，这类通用软件很难满足用户特性的要求。其次，"通用"软件考虑到用户可能出现的各种情况，需用户选择回答操作甚多，降低了使用效率，故有"通用"软件不好用之说。鉴于此，一般情况下都要对这类软件作二次开发、使其紧密结合使用部门的实际工作情况、固定某些操作程式和输出的模式，这一工作是很有必要

的。

3. 国外优秀软件不断进入国内。由于这类软件在硬件环境或编程方法上的差异，可能会出现无法启用或部分功能无法使用。对于这类软件都要作二次开发的工作，使其适应我们的具体情况，这是一种投资少、见效快的软件开发途径。试想，如果没有在西文 DOS 的基础上，对 ROM BIOS 的 INT 05H、INT 10H、INT 16H、INT 17H 等中断处理模块的修改和扩充而形成的汉字操作系统 CCDOS，我国的微机应用绝不可能达到今天这样的规模。

§1.2 二次开发成功的典型

目前，国内微机用户一般都拥有数种汉字操作系统，而其中之一必为 CCBIOS 2.13 系列。2.13 系列从 1986 年问世以来，经过不断的完善和发展，从 2.13A 到 2.13H 已成为国内拥有最多用户的汉字操作系统，其纯软件的最高版本 2.13H 风行大陆，独领风骚。

如果我们对 2.13 系列汉字系统作一番剖析，会发现最初推出的 CCBIOS 2.13A 是在 CCDOS v2.0 的基础上二次开发形成的。2.13A 的推出，主要针对 CCDOS v2.0 存在的两个不足之处：

1. 16 点阵显示字库全部进驻内存

由于字库将近占据 250K 字节内存空间，这对于当时普遍只有 512K 字节内存的 PC 机来说开销似乎太大，影响了一些大型软件的正常运行。

2. 汉字打印功能薄弱、字体单一

打印控制功能仅有字型和行宽选择，行距和字距无法设置控制，这对于中国式报表打印要采取压缩行距的用户来说，尤感头疼，无法实现封闭表格。24 点阵只有宋体一种字模，稍嫌单一。

2.13 的开发者，主要针对这两个问题对 CCDOS v2.0 进行了二次开发，并且突出了重点，在汉字打印驱动程序上狠下功夫，不同凡响地推出了宋、仿宋、黑、楷四体 24 点阵高级打印驱动程序，大受用户欢迎，奠定了 2.13 系列发展的坚实基础。针对 CCDOS v2.0 显示字库全驻内存的问题，2.13 的开发者想出了一种折衷的办法，将二级字库驻留硬盘，只将一级字库驻留内存，为用户节省 100 多 K 字节内存空间，这对当时的用户来说是十分可喜的。

CCBIOS 2.13A 是 1986 年 12 月推出的，在此前后，报刊上关于压缩字库、打印行距和字距的控制等改进文章时有所见，有的还作为交流软件推出，正所谓 CCDOS v2.0 变种丛生，但是大多终未成气候。究其原因，基本上都是特点不突出。2.13 系列的成功，应该首先归功于它的高级打印驱动程序。至今，相当部分用户仍然只用其高级打印驱动程序（并非对 2.13 的其它功能不了解）或者将其移植在其它汉字系统下，足见其强大的生命力。

当然，2.13 系列在其它方面也是颇具特色的，如其最高版本的虚盘字库应用、特殊显示功能以及 CGA 显示器 16 行显示、25 行显示等，但公认其最成功的还是打印驱动部分。

总之，2.13 系列是一个成功的二次开发典型，突出汉字打印功能的完善是其获得成功的关键。

现在我们再来看另外一个获得成功的典型——五笔字型系统。

分析五笔字型的典型版本，流行最广，影响最深的五笔字型 86 年 4.0 版，也是在 CCDOS v2.0 基础上二次开发而来。五笔字型编码的作者主要是研究一种新的汉字输入法，

以解决汉字输入的瓶颈问题，他不可能在当时的情况下重新设计一个汉字系统，只能借用当时的汉字系统 CCDOS，作一些修改，把五笔字型汉字输入法与 CCDOS 有机地连在一起，这是一种明智的办法。而且作者有意识地突出了他的重点——五笔字型。这在版本标志上明显表现出来，其标志为《五笔字型汉字输入技术》。作者没有标明这是某某汉字操作系统，而只是一种汉字输入技术。其实任何一种汉字输入法都要依托于汉字操作系统，不可能在没有汉字系统的情况下实现汉字输入。如果作者当初要标明这是什么汉字操作系统的话，充其量也不过是 CCDOS 2.1x 而也。高明之处是作者反客为主，突出了从属于汉字系统的汉字输入法。

十年沧桑，五笔字型成了当前国内应用最为广泛的汉字输入方法，尽管当初以不计名次榜上无名，尽管至今仍有人对五笔字型提出种种指责，但是，五笔字型普及率之高、用户之多是没有哪一种汉字输入法与之抗衡的。五笔字型系统能有今天的辉煌成就，与当初借用 CCDOS 的二次开发有着密切的关系。

§1.3 如何进行二次开发

二次开发一般要经过选题、理解、实施三个阶段。下面分别叙述：

1. 选题

如果我们确定某软件有二次开发的价值，那么，首先就要确定作哪些方面的工作，有一个“做什么”的大致范围，初拟出一个目标。目标能否实现，还要经过对程序的理解和实施阶段来验证。由于种种困难，不得不放弃最初的目标，这种情况也是有的。选题的范围有以下几种：

(1)纠正错误。如果程序存在错误，大多数情况是可以修正过来的。

(2)寻找薄弱环节，加以改进。

(3)挖掘潜力，增扩功能。

(4)移植其中的部分功能，以作用于其它软件。

(5)将自行开发的软件植入其中；或将两个以上软件合并，构成一个组合式软件。

(6)环境适应性的改造，包括软硬件环境的改变。

(7)西文软件的汉化。

2. 理解

选题和理解并不一定是先和后的顺序，也可能通过理解之后，才作出选题决定。如在理解的过程中发现程序中有可利用于扩充功能的因素存在，或意外地发现了其中的隐式错误等等。理解包含下面三个部分：

(1)熟悉软件的使用。对任何一个待开发的软件，开发者都必须熟悉怎样使用它。也许开发者是受人之托要对某个软件作改造，但也必须先熟悉这个软件的使用方法，把各种功能都试试。熟悉使用是为了获得感性理解。明了软件的功能(明显的)和操作方法，才有可能体会到它的长处和短处，从而决定取舍。

(2)观察软件在运行中的情况。仔细观察运行中的情况，可为下步静态分析程序清单作辅证，在读某段程序时，可连想到程序运行的情况，有利于读懂程序和分析程序。有时通过直观观察，也会发现软件存在的问题或者需要改进的地方。

(3)分析程序清单。对程序的理解是比较困难的，特别是他人的程序。在无任何文档资料的情况下，仅靠 DEBUG 反汇编命令获得的程序清单，要读懂也非易事。如果要找到产生错误的相应程序，则更加困难。在静态分析程序的时候，可和动态分析(上机调试)相结合，即读一段程序后，就调试一下这段程序，以印证静态分析的结果。

静态分析程序难度较大，也无什么捷径可言，唯一的途径就是多读程序，见多自然识广，古人云"读书破万卷，下笔如有神"，也是这个道理。读他人程序之难，难在你要顺从他人的思维，别人牵着你的鼻子走，自然感到难于理解。这就需要耐心和毅力，如果说有什么诀窍的话，那就是它。

3. 实施。实施是设想的具体实现步骤。在选题阶段中，我们已确定对软件作哪些修改和功能扩充，通过对程序的理解阶段，可进一步感知设想的修改和功能扩充的可能性。如果确认是可行的，下一步即为具体实施阶段，可用下述方法之一对程序实施修改：

(1)补丁修改法。这是使用最多的一种方法，在要修改的地方使用一条转移指令，指向程序的尾部，在原程序的尾部安排修改的程序块，修改程序执行完后转回原程序。

(2)替换修改法。直接对原程序修改，用新指令替换原指令，被修改的程序长度没有变化。替换修改保留了原程序的结构，是一种理想的修改方法。替换前应对原程序段仔细分析，看能否简化，以便省出空间安排新增的指令。替换修改法只适合于有少量修改的程序段。

(3)嵌入修改法。有相当一部分程序，用补丁法修改是不行的，这部分程序的特点是带有数据参数，在程序执行后数据参数紧跟在程序之后，如 DOS 外部命令 COMP、调试程序 DEBUG 等。对这类程序的修改必须将增加的程序植于原程序体内，否则，程序一旦运行，增加的程序将被数据参数覆盖，出现意想不到的结果。

(4)动态修改法。动态修改是对原程序不作任何修改，是让原程序执行之后，用另一个程序对已经进驻内存的原程序进行修改。这种方法适合于驻留型的原程序，或者需要动态确定某种状态的情况。例如，一些加密的驻留型系统软件，如加密的汉字系统，若解密困难，可考虑用此法。这类软件，不管其加密难度如何，在它自身解密运行之后，总是以明码形式驻留内存的，就可采用另一个程序去修改驻留在内存中的原程序，达到对其某一功能修改的目的。

(5)拦截修改法。指拦截中断处理程序的修改方法，对原中断处理程序不作任何修改。拦截法有时是直接作用于该中断处理，如为了实现汉字系统下自动分页打印，对 INT 17H 打印模块的拦截；有时是间接作用于另外的程序，如为了实现 WS 定时自动存盘，对时钟中断 INT 1CH 和键盘管理模块 INT 16H 进行拦截。

§1.4 二次开发中应当注意的问题

1. 要把实用性摆在第一位

当我们为某一软件增扩功能时，首先应想到是否具有实用性，这是至关重要的。有时实用性和新颖性往往产生矛盾，新颖的东西却不实用，这是不可取的；实用而不新颖，仍有可取之处。假如我们为 DOS 增加一个"键盘按键发声"外部命令，每按一键均发声。这个功能在 PC 机上似乎比较新颖，但从实用性方面来考虑，恐怕就成问题。可以设想，如果每一键都发声，将十分令人心烦。再进一步设想，如果一个机房里有几台、十几台微机，大家都每按一键就发声，其场面是可想而知的。所以，搞这样的功能扩充是没有什么实用价值的。

2. 要注意新颖性

在实用的前题下，应尽量注意新颖性。新颖性包含扩充功能的新颖和修改技法的新颖以及构思的新颖等。本书中有好些这样的例子，例如在对 2.13H 系统的 CV26 模块的功能扩充中为光标设定颜色、任选显示字符集等；在对 DEBUG 的改造中，增扩的“搜索汇编符号指令”、“反汇编自动分页打印”等；在对五笔字型改造中的“CGA 动态 11 行/16 行切换”等。

3. 要避免重复性开发

这里的重复性是指系统软件中已经具有的功能。如果我们要为 DOS 增扩外部命令，那么要仔细查对一下，DOS 是否已经具有类似命令。必须确信要扩充的命令 DOS 不具备或者不完善。例如，我们要为 DOS 增加一个“若干文件连接为一个文件”的外部命令，其实就没有这个必要，因为拷贝命令 COPY 就具有类似功能，可以将若干个文件合并为一个文件。

4. 要避免盲动性

千万不要把一些特定环境下出现的某种特殊现象作为新发现，盲目地加以应用，甚至当作一种新技术盲目发表。如果确信是新的发现，要反复验证，最好能从原理上得出结论。

5. 要避免顾此失彼的情况产生

由于软件存在连续性和互补性，各个功能之间在互相利用，程序呈犬牙交错状态，当某种错误被纠正或扩充了某种功能之后，可能会因此而产生新的错误，这是二次开发中经常出现的问题。笔者见到好几篇关于对 DOS v3.3 DEBUG 改造的文章，都存在被调试程序将覆盖为 DEBUG 增加的程序中的若干字节问题，为 DEBUG 的正常应用留下了隐患。

6. 不以牺牲原功能为代价来扩充新功能

应该如何理解“扩充”二字？扩充是指保留原功能的基础上，增加的功能。不要片面地认为某某功能没有什么用途，就将它删去。因为这仅仅是你个人的看法，也许你认为没有用的东西，别人却认为非常有用，这是常有的事情。从遵重原作者的角度来说也是不妥当的，不能随意认为别人开发的某某功能没有用。笔者认为还是不删原功能为好，只是为它增加功能。这样作既保留了原软件的风貌，又遵重了别人的劳动成果。

7. 注意用户“先入为主”的心态

从心理学和价值论的观点来看，用户是以应用为目的，只要他现在使用的软件能满足需要，在他完全掌握和熟悉之后，他是不会轻易放弃已经熟悉的软件，而去学习另一个虽然功能更强但完全陌生的软件的。笔者曾见过一个电脑公司的电脑打字员，她一直是用 WS 字处理软件。虽然该公司有比 WS 高级的字处理软件》，但因为她已经非常熟悉 WS 的各种命令，完全能够满足客户打字要求，她又何必花精力去学习其它字处理软件的使用呢？因此，如果那些比 WS 高级的字处理软件（晚于 WS 面世），仍然保留 WS 的所有命令格式（或者最主要的常用命令格式），让 WS 的用户可以不加任何学习即可使用开发出的比 WS 高级的软件，不就能获得更多的用户吗？其实功能键只是一个定义问题，没有必要定义为与已有的同类软件背道而驰。笔者也是用户，对此深有体会。

第二章　二次开发的工具软件

§2.1　调试程序 DEBUG 命令详解

§2.1.1　DEBUG 的功能

调试程序 DEBUG 有下列功能：

·提供一个可控制的调试环境，使操作者能够监视和控制被调试程序的执行。可以在被调试程序中直接设定一些问题，并立即执行，从而确定这些问题是否已经解决，这时不需要重新汇编程序就可以弄清楚更改结果。

·装入(LOAD)，改变(ALTER)或显示(DISPLAY)任意文件。

·执行目标文件(OBJECT FILES)。目标文件是机器语言格式的可执行程序。

§2.1.2　怎样启动 DEBUG 程序

启动 DEBUG 程序，只须打入：

```
DEBUG [d:][path][filename[.ext]][parm1][parm2]
```

如果你输入了文件名(filename)，DEBUG 程序将把你指定的文件装入内存，然后你就可以打入命令去改变、显示或执行指定文件的内容。

任选参数 parm1 和 parm2 指的是文件说明(filespec)里的任选参量。例如：

```
DEBUG COMP.COM A:A1.COM B:
```

在这个命令里，A:A1.COM 和 B:是 DEBUG 程序为 COMP 程序预备的参量。

当启动 DEBUG 程序时，为了调试被调试程序，寄存器和标志位设置如下值：

·段寄存器(CS,DS,ES,SS)设在可用内存的底端，即第一段接在 DEBUG 程序的尾部。

·指令指针(IP) 设置在十六进制的 0100 处。

·堆栈指针(SP)设置在段尾或者在装配程序临时区的底部，两者中取较低的一个。在偏移量 6 处的段长度将减去十六进制的 100 作为堆栈段的长度。

·剩余的寄存器(AX,BX,CX,DX,BP,SI,DI)设置为零。但是，如果启动 DEBUG 程序是带有文件说明(filespec)，则 CX 寄存器将包含有该文件以字节数计数的长度。如果文件大于 64K，其长度就放在寄存器 BX 和 CX(高位放在 BX)。

·标志位的初始状态都是零值：

```
NV UP EI PL NZ NA NA PO NC
```

在代码段中，系统约定驱动器传输地址设为十六进制的 80。

所有的可用内存部分都已分配，因此任何想通过装入程序来分配内存的企图，都会失败。

注意：

(1)如果由 DEBUG 装入的文件带有扩展名.EXE,那么 DEBUG 必须重新定位和设置段寄存器、堆栈指针和指令指针为该文件所定义的值,DS 和 ES 寄存器总是指向最低可用段的程序段的前缀,BX 和 CX 寄存器则会有程序的长度(小于文件长度)。

当连接程序生成该文件时,如果指定了相应的参数,则被装入的程序放在内存高地址一端。

·关于装入.EXE 文件更多的资料,请参考 DOS 技术手册有关章节。

(2)如果由 DEBUG 装入的文件有扩展名.HEX,则认为该文件内部是 INTEL 十六进制的格式,在它被输入时转化为可执行的格式。

§2.1.3 DEBUG 的命令参量

参　量	定　义
address (地址)	按照下面两种格式之一,输入其中一或两部分规定。 ·一个用字母表示的段寄存器标志加上一个偏移量。例如:CS:0100 ·一个段地址,加上一个偏移量。例如:4AB:0100 ·仅有一个偏移量。例如:100 注: (1)前两种格式,用冒号分隔偏移量。 (2)所有的数字都是十六进制的,可打入 1～4 个字符。 (3)地址中规定的内存单元必须是有效的。也就是,它们必须是实际存在的。如果企图访问一个不存在的单元,将发生不可预料的结果。
byte(字节)	输入一个或两个十六进制的数值。
drive (驱动器)	输入一个或者两个数字(例如,对驱动器 A 为 0 对驱动器 B 为 1 来指出从那一个驱动器装入数据或者向那一个驱动器写入数据。 [参阅 LOAD(装入)和 WRITE(写)命令]
filespec (文件说明)	输入一个由一到三部分组成的文件说明。文件说明由驱动器、文件名和文件扩展名组成。三个字段都是任选项。为使 NAME 命令有意义,至少要规定一个驱动器名或者一个文件名(参阅 NAME 命令)。
list (表)	输入一个或多个字节值或字串值,或者同时输入这两种值。例如:F3 ' XYZ' 8D 4" abcd". 在此表中有 5 项(即三个字节项和两个字串项,共有 10 个字节)
portaddress (入口地址)	输入 1～4 个十六进制字符值,以规定一个 8 位或 16 位的接口地址(参阅 input 和 output 命令)
range (范围)	输入下列任意格式以规定内存范围的下限地址和上限地址。 ·address address 例如: CS:100 110 注:在第二个地址中,只允许有一个偏移量。地址之间必须用一个空格或逗点分隔。 address L value 这里,value(值)是由命令处理的以十六进制表示的字节数。 例如: CS:100 L 11 注:(1)范围的极限值是十六进制的 10000,因此,地址的值和偏移部分的总和不能大于 64K 字节。要把 64K 字节的值极限在四个十六进制字符之内,可打入 0000 或(0)。 (2)在范围(range)中规定的内存单元必须是有效的,即它们必须是实际存在的。如果