6809

汇编语言程序设计

下　册

Lance A. Leventhal

机械工业部重庆工业自动化仪表研究所

一九八二年十月

出　版　说　明

许多微计算机用户程序都使用汇编语言编制程序，大多数工业上微计算机用户程序都需要用汇编语言来提供各种控制功能。几乎所有的微计算机设计者都必须具有一些汇编语言知识。了解汇编语言有助于各种高级语言的分析和评价。本书详尽地介绍了汇编程序和汇编程序设计，列举了在实际应用中的各种典型实例，并且附有代表性的习题。内容丰富具体，深入浅出，通俗易懂。对于从事微计算机应用的工程技术人员是一门很有价值的参考书，也是微型机学习班的好教材。

该书原文为《6809 Assembly Language Programming》，作者是美国 Lance A. Leventhal 博士，(1981年出版的最新板本)。我们组织了从事这方面实际工作、并有一定实践经验的工程技术人员翻译了这本书。原书共22章和一个附录，译本分上下两册出版，上册包括第一章至第十五章；下册从第十六章至第二十二章和一个附录。

本书由下列工程师分章翻译：郑宗汉（第一章至三章），余奔（第四章至第九章），杜芝君（第十章至第十五章），贾永乐（第十六章至第二十章），胡纯阳（第二十二章和附录），校对郑宗汉，编辑李连雨。由于时间短促，加之我们的水平不高，难免在译文中有不妥之处，望读者多加批评指导。

机械工业部重庆工业自动化仪表研究所

1982.4.

第四部分　软件开发

第四部分 软 件 开 发

以上章节叙述了如何书写简短的汇编语言程序。尽管这是一个重要的课题，但它仅是软件开发的一小部份。虽然书写汇编语言程序对于初学者是一个较大的任务，但不久就变得简单了。现在，熟悉了以标准方法编制6809微处理器的汇编语言程序。下面六章将叙述如何制定程序任务，如何组合简短的程序构成一个工作系统。

软 件 开 发 的 阶 段

软件开发包括几个阶段。图Ⅳ-1是软件开发过程的一个流程图：

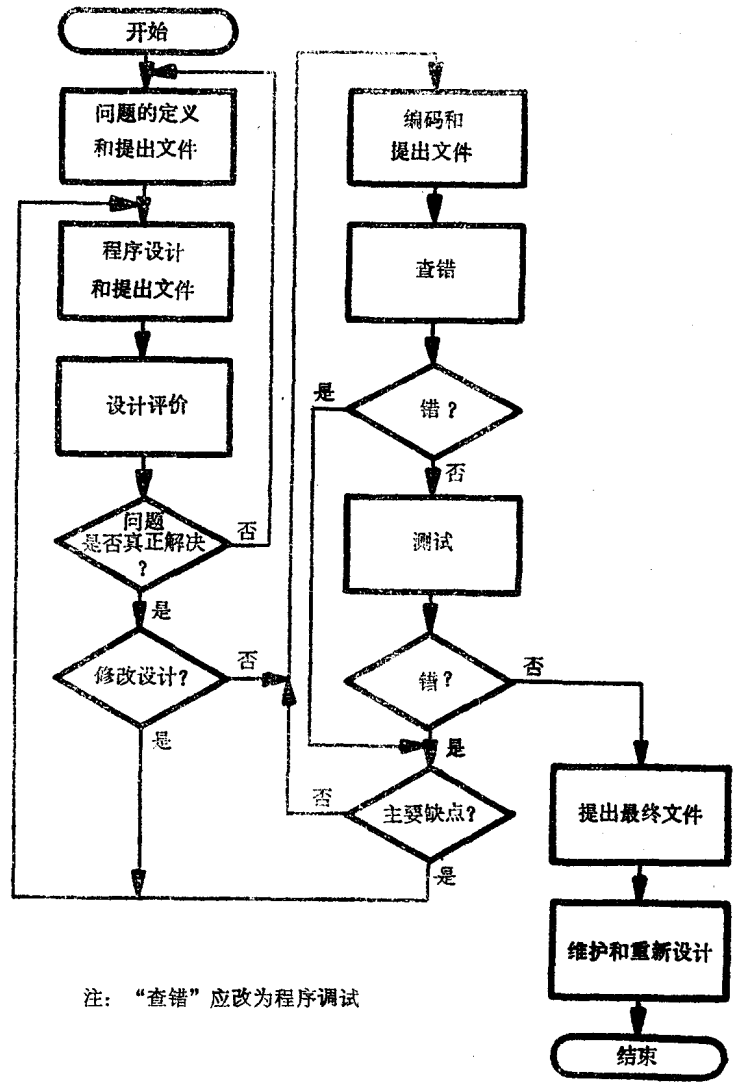
软件开发阶段是

- 问题的定义
- 程序设计
- 编码
- 程序调试
- 测试
- 文件编制
- 维护和再设计

这些阶段的每一个阶段对于构成一个工作系统都是重要的。以计算机理解的形式书写程序仅是漫长过程的一个阶段。

编码的相对重要性

编码对于定义和实现一个工作系统而言，一般是最容易的阶段。书写计算机程序的规则是容易学的。计算机虽不同，但基本技术仍旧是相同的。确实有些软件设计过程因编码带来麻烦，但编码不是软件开发最花时间的部份。专家估计一个程序编制人员每天能写出 1 到10句完全无错的成文的语句。显然区区 1 到10句语句的编



注："查错"应改为程序调试

图四-1 软件开发流程图

码对一整天的劳动而言算不上什么。大多数的软件设计，编码时间占程序编制人员的时间不到25%。

衡量其它阶段的进展

衡量其它阶段的进展是困难的。你可以说程序的一半已经写好，但你几乎不能说错误的一半已剔除或问题的一半已定义。对于诸如程序设计，程序调试和测试这样一些阶段的时间表是很难制作的。好几天或好几周的努力都可能是没有明显的进展。而且某一阶段的一个不完善的工作，对后面产生极其严重的问题。例如，质劣的问题定义对程序设计中的程序调试和测试带来相当大的困难。某一阶段节省的时间是以花费后面阶段好多倍的时间为代价的。

阶 段 的 定 义

问题的定义

问题的定义是阐述对计算机中执行的任务的要求。例如使计算机控制一个工具，必须的是什么，是进行一系列的电气测试还是处理中央控制器和一个远程仪表装置之间的通讯。问题的定义需要你确定输入输出的形式和速率，所需的处理量和速度，可能出错的形式和它们的处理。问题的定义为建立计算机控制系统构成一个笼统的概念和规定计算机的任务和要求。

程序设计

程序设计是制定符合要求的计算机程序的纲要。在设计阶段，任务是以较容易地被换成程序的方式来叙述。本阶段有效的技术有流程图、结构式程序设计、模块程序设计和自上而下设计。

编码

编码是以能被计算机直接理解或者能翻译的形式书写程序。形式可以是机器语言，汇编语言或某种高级语言。

程序调试

程序调试，也称谓程序的检验，是根据设计检查程序的执行。在本阶段使用这样一些工具，如断点、跟踪、模拟、逻辑分析器和联机仿真器。程序调试阶段的结束是很难定的，因为你几乎不知道你已经找到最后一个错误。

测试

测试，也称为程序正确性的校验，是确保程序正确地执行整个系统的任务。设计者使用模拟程序、练习程序和统计技术估量程序的性能。本阶段有点象硬件的质量控制。

文件编制

文件编制是以适合用户或维修人员的形式叙述程序。文件编制还使设计者开发一个程序库，使得以后的任务简单化了。流程图、注释、存贮映象和库等形式是文件编制中使用的一些手段。

维护和再设计

维护和再设计是维修，改进和扩展程序。显然设计者必须准备好处理以计算机为基础的设备的现场问题。需要专门的诊断方法或程序以及其它维护工具。程序的改进或扩展在遇到新的要求或处理新任务时常是必须的。

第十六章　问题的定义

典型的微处理器任务需要很多的定义。例如控制一台秤，或一台现金出纳机或一台信号发生器必须做些什么？显然，我们要对所包括的任务下定义，有较长一段路程。

输　　入

怎样开始定义问题？显而易见是从输入着手。我们首先列出在该应用中计算机接受的所有输入。

输入的例子有：

· 传输线来的数据块

· 外部设备来的状态字

· A/D转换器来的数据

然而，对每一个输入提出下列问题：

1. 输入的形式是什么，亦即计算机实际接受什么信号？

2. 输入什么时候准备好，以及处理器怎样知道它是否准备好？处理器需要以选通信号来接受输入？输入是否备有它本身的时钟信号？

3. 可供使用时间是多久？

4. 它是否经常变化以及处理器如何知道它是否有变化？

5. 输入是由一个序列还是数据块组成？其次序重要吗？

6. 如果数据有错做些什么？这些错可以是传输错，不正确的数据，次序错，额外的数据等。

7. 输入与别的输入或输出有关吗？

输　　出

接下来是定义输出。我们列出计算机必须产生的所有输出。

输出的例子：

· 去传输线的数据块

· 去外部设备的控制字

· 去D/A转换器的数据

接下，我们将对每一个输出提出下列问题：

1. 输出的形式是什么，亦即计算机必须发出什么信号？

2. 什么时候它必须准备好以及外部设备如何知道它是否准备好？

3. 它必须有效多长时间？

4. 它必须怎样变化以及外部设备怎样知道它是否已变化了？

5. 输出是否是一个序列？
6. 为了防止传输错或为了检测和剔除外部故障将需做些什么？
7. 输出与别的输入和输出是什么样的关系？

处 理 部 份

读入数据和输出结果之间是处理部份。我们必须严格确定计算机必须如何处理输入数据。问题是：

1. 输入数据转换为输出结果的基本步骤(算法)是什么？
2. 存在什么时间限制？这些限制可能包括数据速率。
3. 存在什么存贮容量限制？限制是程序存贮量或数据存贮量，还是缓冲区的容量？
4. 必须使用什么标准程序或表？它们的要求是什么？
5. 存在什么特定情况以及程序将如何处理这些情况？
6. 结果必须具什么样的精度？
7. 程序将怎样处理错误或诸如溢出，上溢或有效位丢失等特定状况？

出 错 处 理

在很多应用中，一个重要的因素是出错的处理。很明显，设计人员必须做好剔除一般性错误和诊断故障的准备。在定义阶段，设计人员必须提出的问题有：

1. 将出现什么错误？
2. 多半是哪一种错误？如果由人操作系统，人为错误是最通常的。其次，通讯或传输错误比起机械的、电气的、数学上的或处理器的错误更为常见。
3. 哪一类错误对系统而言将不是立即察觉的？一个特殊的问题是出现系统或操作者不认为其是错的错误。
4. 怎样才能以最低限度的时间和数据使系统补救错误，并且意识到错误已经发生过？
5. 哪一种错误或故障引起相同的系统行为？从诊断角度看怎样才能区别这些错误或故障？
6. 哪一种错误包含了特定的系统过程？例如，解决奇偶错是否需要数据的重新传输？

另一个问题是：在没有专家时，现场技术员如何能系统地找出故障源？内部测试程序，专用诊断程序或特征值分析是有益的。

人为因素/操作者的交互作用

不少的以微处理器为基础的系统包含有人的交互作用。人为因素对这类系统在其整个开发过程中必须考虑。设计人员必须提出的问题有：

1. 什么输入过程对于操作者是最适应的？
2. 怎样使操作者能容易的确定输入操作的开始，继续和结束？
3. 操作者如何知道步骤有错和装置有故障？
4. 什么错是操作者最容易犯的？

5. 怎样使操作者知道数据已正确的打入？

6. 是以操作者易读和易理解的格式显示？

7. 系统的响应对操作者是适当的？

8. 系统对于操作者而言是容易使用？

9. 对于没有经验的操作者有指导性的特征？

10. 对于有经验的操作者是否有捷经和几个合理的选择方案？

11. 在中断或受到干扰之后，操作者总是能判定或重置系统的状态？

建立一个为人们所使用的系统是困难的。微处理器能够制成比较有效，比较灵活的和比较灵敏的系统。然而设计者必须加入人的干予，以大大提高系统的实用性和吸引力以及操作者的操作效率[2]。

当然处理器在包括人为特征或修养等选择方面没有固有的基准。处理器既不作出从左至右比从右至左好、顺向比反向好、递增比递减好或十进制数比其它进制数好，也不会识别出操作者对简易性、一致性、以往经验的继承性和操作上逻辑次序的偏爱。处理器也决不会迷惑、迷失方向、混淆或钻牛角尖。设计者必须在设计和开发交互系统时考虑所有这些因素。

举　　例

定义一个开关灯光系统

图16-1表示出一个简单系统，系统的输入来自一个单刀单掷开关，输出是送往一个发光二极管显示。对于开关的每一次闭合，处理器使显示灯亮一秒。该系统是比较容易定义的。

开关输入

让我们首先考察一下输入，是回答前面提出的每一个问题：

1. 输入是一位，它可以是“0”(开关闭合)或者是“1”(开关打开)。

2. 输入总是可获得的，不需要请求。

3. 输入在开关闭合后至多几毫秒内就可使用。

4. 输入很少多于每几秒钟变化一次。处理器仅处理开关弹跳。处理器必须监视开关，确定它什么时候闭合。

5. 不存在输入序列。

6. 明显的输入错误是开关故障，输入线路故障以及操作者在未经过足够多的时间之前再次闭合开关。后面将讨论这些错误的处理。

7. 输入不依赖于任何其它输入或输出

灯光输出

定义系统的下一步工作是考察输出。问题的回答是：

1. 输出是一位，为“0”时显示灯亮，“1”时为关闭。

2. 输出不存在时间限制。外设不需要知道数据是否准备好。

3. 如果显示器是一个发光二极管，在脉冲速率为每秒100次左右，数据仅需存在几个毫秒。观察者将看到一个连续点亮的显示器。

4. 在一秒钟之后，数据必须改变(消失)。

5. 设有输出序列
6. 可能的输出错误是显示器故障和输出线路的故障。
7. 输出仅与开关输入和时间有关。

处　　理

处理部份极其简单。一当开关输入变成一个逻辑“0”，CPU 使灯亮一秒钟。不存在时间或存贮上的限制。

错误处理

现在让我们看看可能的错误和故障。有：

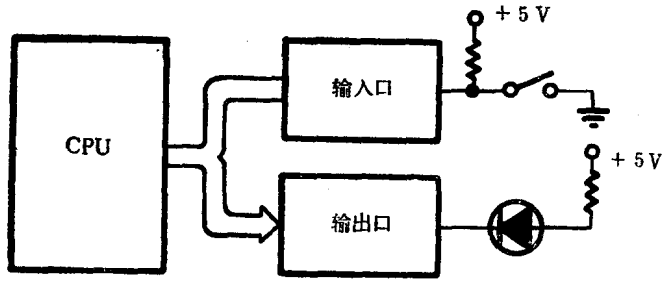
- 一秒钟之内再一次的开关闭合
- 开关故障
- 显示器故障
- 计算机故障

毫无疑问多半是第一种错误。最简单的解决办法是在一秒钟之内处理器不理会开关的闭合。这短暂的不响应周期对于操作者是不显眼的。在这期间里不理会开关意味着防弹跳线路和软件不是必须的，这是因为系统对于弹跳总是不响应的。

显然，后三种故障导致不规则的结果。显示器可以随机地停在亮上，关上或修改状态上。消除这些故障的某些可能的方法是：

- 用灯测试硬件检查显示器，也就是以一个按钮使灯亮，与处理器无关
- 直接接到开关检查它的动作
- 使用诊断程序检查输入和输出线路

如果显示器和开关都在工作，计算机处于事故状态，带有合适设备的现场技术员能确定故障的原因。



如果开关打开，开关输入是“1”，开关闭合是“0”。CPU 将输出加在发光二极管的阴极，“0”使灯亮

图16-1　开关和灯光系统

定义根据开关状态存贮数据的装入程序

图16-2示出允许用户将数据装入微型计算机任一存贮单元的系统，输入口 DPORT从八个扳键开关读数据。另一输入口 CPURT用来读控制信息。还有三个按钮开关：高位地址、低位地址和数据。输出是从数据开关最后打入的完整值，8 个发光二极管用作显示。

当然，系统还需要电阻，缓冲寄存器和驱动器。

输　　入

各开关的特性与上例相同。为了简化去抖动过程和迫使操作者释放按钮，系统仅在按钮释放之后响应。这是降低开关磨损的通用技术，因为操作者总想重复的按按钮。在该系统中，有一个下列的明确的输入序列：

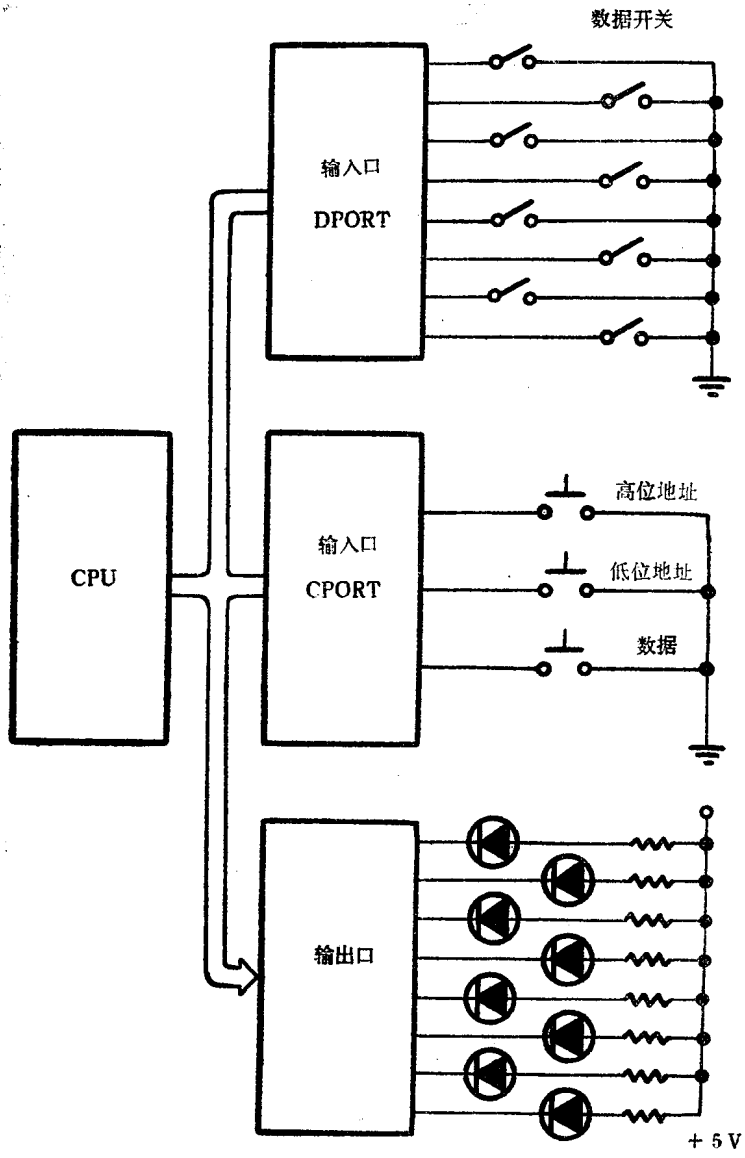


图 16-2　根据开关壮态存贮数据的装入电路

1．操作者必须根据地址的高 8 位设置数据开关状态，然后

2．按和放开高位地址按钮。高位地址由指示灯显示出来，程序将数据认作地址的高位字节。

3．然后，操作者根据地址的低位字节值设置数据开关，并且

4．按和放开低位地址按钮。低位地址由指示灯显示出来，程序将数据认作地址的低位字节。

5．最后，操作者按要求的数据设置数据开关，以及

6．按和放开数据按钮。现在显示器显示数据，程序将数据存到前面打入的地址单元里。

操作者可以重复此过程输入一个完整的程序。显然即使在这样简单的情况下，仍有较多的可能序列要考虑的。如何解决有错的序列，怎样才使系统使用方便？

输　出

输出没有问题。在每一个输入之后，程序送输入的反码到显示器（因为显示器是低有效）。直至下一次输入操作之前，输出数据保持不变。

处　理

处理部份仍是相当简单的。不存在时间或存贮量的限制。程序通过等待几个毫秒去掉开关的抖动，程序必须为显示器提供数据的反码。

出错处理

最可能的错误是操作者误操作。包括有

· 输入不正确

· 次序不正确

· 输入不完整，例如掉了数据。

系统必须能以合适的方式处理这些问题，因为这些错误必定出现于实际操作中。

设计人员还必须考虑设备故障带来的影响。正如前述，可能的问题是：

· 开关故障

· 显示器故障

· 计算机故障

然而，在这系统中，必须更加留心这些故障是怎样影响系统的。计算机故障使整个系统失灵，因此容易检测，显示故障可能不是马上被察觉的，显示灯故障测试用来检查显示灯的操作。注意，希望分别的测试每一个发光二极管，以便诊断输出短路在一起的情况。另外，操作者也不可能立即查出开关的故障，但操作者应马上注意它，通过淘汰确定哪个开关出了故障。

操作者错误的纠正

让我们看一下操作者可能出的错误，典型的错误有：

· 数据有错

· 输入的次序或开关的次序有错

· 现有的尚未输完就试图进行下一个。

操作者可以通过显示器马上注意到数据出错。那么可行的补救办法是什么？几种选择方案是：

1. 操作者必须完成整个输入过程，也就是如果出现在高位地址，仍输入低位地址的数据。显然，这样的过程是浪费而麻烦的。

2. 操作者可通过返回高位地址的输入步骤，重新开始输入过程。如果错误在高位地址，这样的做法是有效的；如果错误在低位地址或数据上，就迫使操作者重新输入先前的数据。

3. 操作者可通过设置数据开关为要求的数据和按相应的按钮，随时输入整个序列的任一部份。这过程使得操作者可在序列的任一点上作出修正。

这种类型的过程总是比不允许立即修正错误而又有各种各样结果的，或输入数据到系统

中而不许操作者最后检查的过程为好。以是否提高操作者的效率来证明硬件或软件的任何复杂性的增加。总是让微型计算机做枯燥无味的工作及识别任意的序列，它决不会感到疲倦，也决不会忘记操作步骤。

更有用的特征是设置状态灯，它指出显示器显示的含意。三个状态灯各表示“高位地址”、“低位地址”和“数据”，使操作者无需记已按了什么按钮就知道输入了什么内容。处理器监视该序列，这是以增加软件的复杂性减轻操作者的任务。显然，三个独立的显示器再加上检查存贮单元的能力，将更有助于操作者。

应指出，虽然强调了人机的相互作用，但机器或系统的相互作用具有很多相同的特性。微处理器将做这方面工作。如果复杂了微处理器的任务，使得错误纠正简单以及使故障原因明显，那么整个系统将工作得更好以及维护也容易。请注意，不要等软件完成之后再考虑系统的使用和维护，而应该在问题定义阶段就包括进这些因素。

定义终端校验

图16-3是一个简单的信用卡校验终端的方框图。一个输入口从键盘接收数据（见图16-4），另一个输入口接受来自传输线的校验数据。输出口将数据发送到一组显示器去(见图16-5)，另一输出口将信用卡号码发送到中央计算机。第三个输出口用来当终端准备接收询问时点亮一个指示灯，当操作者发送信息时另一个指示灯就亮。当终端接受一个回答，“忙碌”指示灯灭，显然，数据的输入和输出比上述情况要复杂得多，虽然处理仍是简单的。

附加的显示灯可能有助于突出回答的意义。大多数终端以绿灯表示“是”，红灯表示“否”黄灯表示“查询存贮管理”。注意，还应给这些灯简明地注以它们的意义，方便色盲操作者。

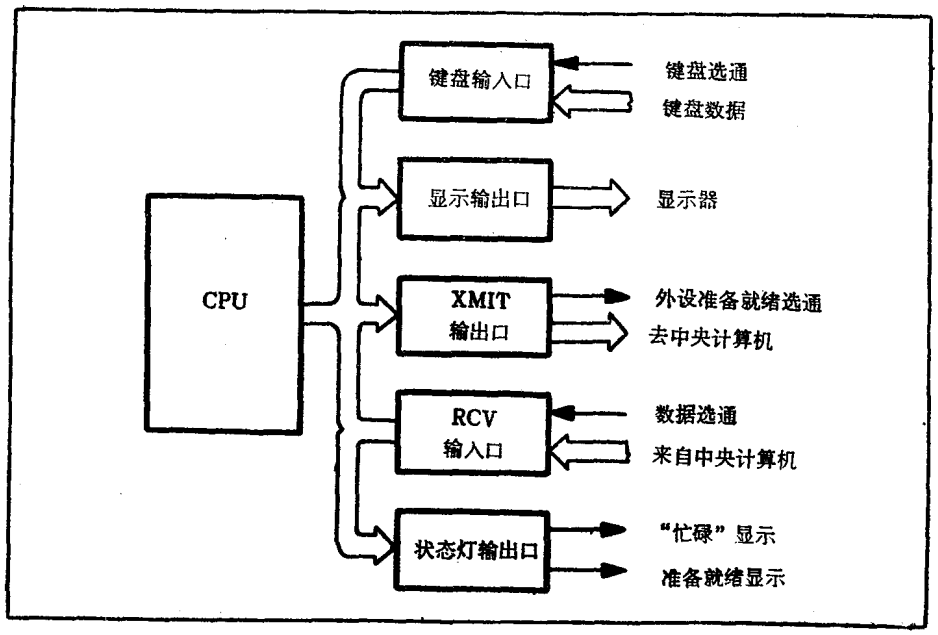


图 16-3　校验终端的方块图

输　入

先看键盘输入。当然这些输入与开关是不同的，它要求 CPU 必须具有识别新数据的一

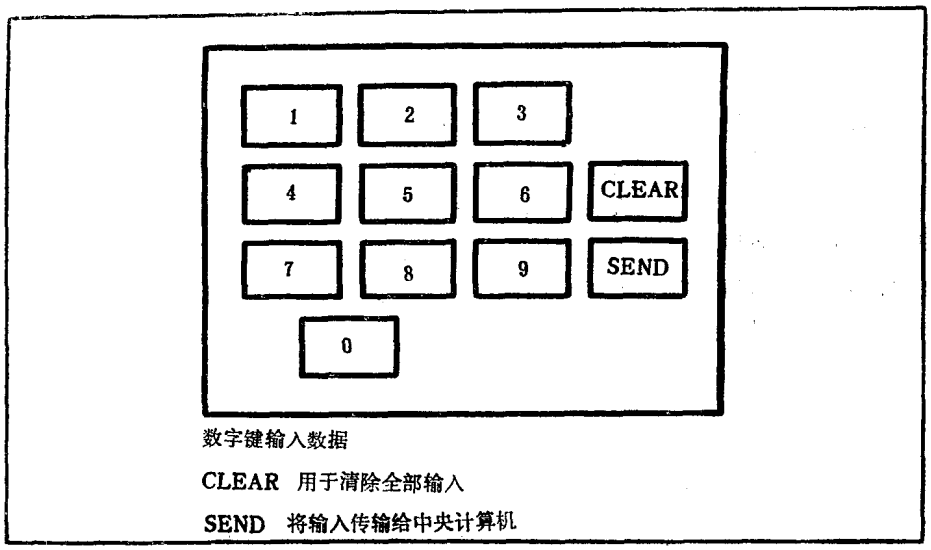


图 16-4 校验终端的键盘

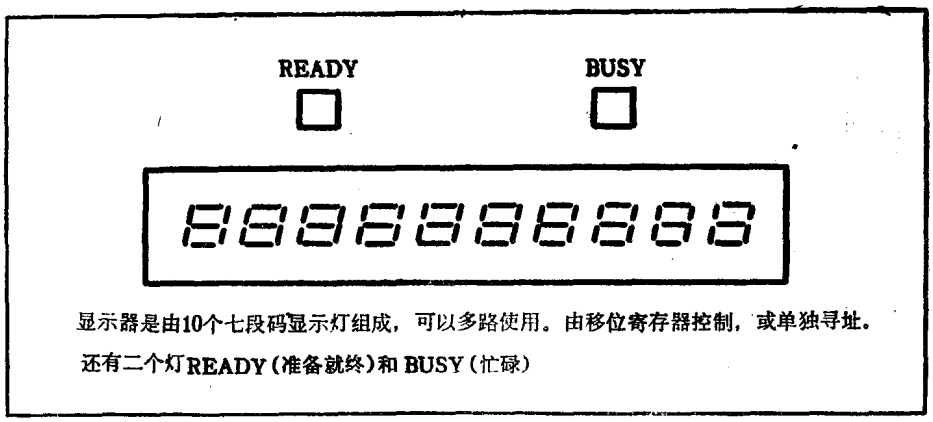


图 16-5 校验终端显示器

些方法。假设，每一个键闭合提供一个唯一的十六进制码（可以以一位数字编码12个键的每一个）和选通。程序将认出选通以及取得识别该键的十六进制数。由于程序不能漏掉任一数据或选通，因而有时间限制。由于键输入之间至少间隔好几毫秒，因此限制并不是严厉的。

类似地，传输输入由一串字符所组成，通过选通（大多来自于UART）识别字符。程序应识别每一个选通以及检出字符。通过传输线发送的数据，一般是组织成信息。可能的信息格式是：

- 引导字符或标题
- 终端目的地址
- 编码的“是”或“否”
- 结束字符或结尾

终端将检查标题，读目的地址，以及检查信息是否是给它的。如果信息是给它的，那么终端接受数据。地址可以是（常常是）通过硬接线连到终端，使得终端只接受发给它的信息。此法是以某种灵活性的代价换取软件的简化。

输出

输出也比前面几例复杂。如果显示器是多路的，处理器不仅仅将数据发送给显示器，而且还必须直接将数据送到某一特定显示灯上。这就需要一个独立的控制口，或要一个计数器和译码器来处理这工作。注意硬件的空白控制可以使多位数第一位数前面的0是空白而不是0。时间约束包括为使操作者得到一个连续显示所需的脉冲宽度和频率。

通讯输出将由一串特定格式的字符所组成。程序还必须考虑字符之间所需的时间。信息输出的一种可能格式是：

- 标题
- 终端地址
- 信用卡号码
- 结尾

中央通讯计算机可以询问各个终端，以检查数据是否已准备发送。

处理

该系统的处理部份包括不少新任务，诸如：

- 通过数识别控制键，执行相应的动作。
- 对发送的信息加上标题，终端地址和结尾。
- 识别返回信息中的标题和结尾。
- 检查进入的终端地址。

注意，没有一项任务涉及任何复杂的算法或任何严厉的时间或存贮量的约束。

出错处理

显然，该系统中可能的出错数比前面的几个例子多得多。首先考虑一下操作者可能的错误。这些是：

- 打入的信用卡号码不对
- 试图发送一个不完全的信用卡号码
- 当中央计算机正在处理前一个号码时，试图发送另一个号码，
- 清除非现行的输入

这些错误中的一些，可以通过正确地组织程序较方便的处理。例如，在信用卡号码没有完全输入之前不接受发送键，以及在中央计算机返回回答之前不理睬任何附加的键盘输入。注意，因为忙碌灯还没亮，操作者会知道输入尚未发送。操作者也可以知道键盘何时被锁住了，因为输入没有在显示器上出现以及忙碌灯是关闭的。

校正键盘错

不正确的输入是一个明显的问题。如果操作者识别出错误，那么可以使用清除键进行校正。操作者多半认为有二个清除键更合适，一个清除键清除刚按的键，另一个清除整个输入。这样将修正三种情况，一种是操作者立即识别出错，另一种是在过程的后期识别出错。操作者应有立即修正错误的能力，以尽量少地重复按键。然而，操作者总有一定量的错且没有识别出。大多数信用卡码包括一个自校数字，在将号码发送给中央计算机之前，终端

先检查号码，这一步将节省中央计算机时间，不至浪费检查号码的处理时间。

然而，这就要求终端具备告知操作者有错的某些方式，一般通过闪烁一个显示灯或通过提供一些其它引起操作者注意的特殊指示器。

另一个问题是操作者怎么知道输入是遗漏的或处理不正确。有些终端简单地在延迟一个很长时间之后开锁。操作者注意到，忙碌灯已关但没有收到回答。于是操作者试图再次输入。在一、二次尝试之后，操作者应向监控人员报告有故障。

不少的设备故障也是可能的。除了显示器，键盘和处理器之后，还有通讯错或故障以及中央计算机故障等问题。

校正传输错

数据传输一般将包括错误检查和校正过程。某些可能是：

1. 奇偶校提供错误检出能力，但无纠正手段。接受者将需要请求重发的某种手段，以及发送者将保存数据的拷贝直至正确接收被确认。但奇偶校的实现是很简单的。

2. 短的信息可以使用精心设计的方案。例如响应终端的是/否可以加以编码，以提供栓错和纠错的能力。

3. 接受的确认以及有限次的重试可触发一个指示器，以告知操作者通讯故障（不能无差错传输信息）或中央计算机故障(在一定的时间周期里没有响应)。这样的方案，如同灯测试一样简化了故障的诊断。

通讯或中央计算机故障指示器也使终端"开锁"，也即允许它接受另一次输入。如果终端在校验进行之中不接受输入，这样做是必须的。终端也可在某一定的较长时间延迟之后开锁。某些输入可以是予作诊断用的，即某些信用卡号码用来检查终端的内部操作和测试显示器。

评　　述

问题的定义是软件开发的一部分，其重要性如同它作为任何其它工程任务的一部份一样。注意，它不需要任何程序设计或通晓计算机，更确切地说，它是基于对系统的理介和正确的工程判断。微处理器提供了灵活性和有限的智能，使设计者能够提供范围较宽的功能。

问题的定义与任何特定计算机、计算机语言或开发系统无关。然而，它对特定的应用需要什么类型的计算机及其速度，以及设计者能对硬件和软件做出什么折衷提供指导。虽然对计算机能力的了解有助于设计者提出可能的实现过程，但问题定义阶段并不取决于使用哪一种计算机。