Z80 应　用

〔美〕 J.W.科夫朗　著

上海微电脑厂

1985年3月

序　言

你是否对微处理机控制系统的应用有过一种新的设想，但由于面对大量的技术细节而不得不放弃这种想法呢？要是这样的话，这种情况也并非你一人所有，微处理机控制系统的一些出色的应用都曾经历了这样的命运。设想往往是完美的，而且有这种设想的人，对于微处理机可能并不熟悉，因而不能如愿以偿。

本书就是为帮助 Z80 系统的用户摆脱这个困境而编写的。其目的是使用户对Z80微处理器有一个较深的了解。书中详细说明了 Z80 微处理器是如何与系统硬件相联系而构成一个系统的。还特别讨论了 ROM，静态 RAM，动态 RAM 和输入/输出设备。此外，在后面的章节中还分析了一些专用的外围 I/O 设备，其中有一些，如 SIO(串行输入输出)和 PIO(外围输入输出)芯片，是专为 Z80 的使用而制造的；而其它一些，比如 8253 定时器芯片及 8255 外围芯片则不然。

本书的对象可以是初学者，也可以是熟练的程序员，为此它提供了指导性的说明，清楚而准确的图示和详尽的实例。它还提供给用户充分的信息使之认识并实现他们自己的应用及设想。虽然本书并没有给出 Z80 微处理器的全部应用——某些应用必须来自于读者本身——但它却具备了实现任何新的应用所必需的知识。

内容简介

第一章，“Z80 中 ROM 的使用”从介绍各种类型的只读存贮器(ROM)开始叙述，然后说明普通的 ROM 和 PROM 器件怎样与 Z80 CPU 进行连接的方法。

第二章，“Z80 中静态 RAM 的使用，“说明了在Z80应用中，怎样使用静态随机存贮器(RAM)。讨论了公用 I/O 型和分离 I/O 型两种 RAM芯片并给出了这两种静态 RAM 系统的完整的图。

第三章，“Z80 的输入和输出”，叙述了 Z80 是怎样在电气上与输入输出设备进行通讯的。详细讨论和说明了标准的输入输出端口。

第四章，“Z80 中动态 RAM 的使用”，说明了动态RAM应用于 Z80 的方法。这章从讨论一个典型的动态 RAM 器件入手。然后进一步说明怎样根据 Z80 的内部结构将 RAM 与 Z80 接口。

第五章，“Z80 的中断”，主要讨论中断这一重要论题。分析了三种中断方式，每一类型都举了例子。

第六章，“8255 PIO 与 Z80 的连接及使用”，对 8255 外围 I/O 芯片(一种与 Z80 总线连接的器件)的使用作了说明，讨论了 8255 的每一种操作方式，并给出了许多编程的例子。

第七章。“8253 可编程定时器的应用”，叙述 8253 可编程定时器与 Z80 的连接方法。对这个器件与 Z80 的接口给出了详细说明。此外还举出了几个普通的编程例子。

第八章，“Z80-PIO的应用”，以 Z80 PIO 为中心，继续讨论外围芯片。而且给出了该器件的方块图，并讨论了各种操作方式和寄存器组。此外，给出了许多 Z80-PIO 应用的

编程实例。

第九章，“Z80-CTC 的应用”，讨论了计数器定时器芯片，并说明了 CTC 与 Z80 的连接方式。此外，给出了许多 CTC 应用的编程实例。

第十章，“串行通信导论”，讨论了串行通信这个论题。定义了波特率，启动位，停止位和标志电平等概念。用一个实例，说明了 8251 USART 与 Z80 的连接及应用。

第十一章，“Z80-SIO 的应用”，以 Z80-SIO 为核心，继续讨论串行通信的问题，并给出了几个使用 Z80-SIO 和对其进行编程的实际例子。

第十二章，“Z80 的静态激励测试”，由讨论 Z80 的静态激励测试(SST)作为本书的结束。这章里叙述了怎样能不用软件而使用这种方法调试系统。如果在一个现存的 Z80 系统中增加一个新的接口，那么用这种技术就能迅速地而且很容易地检查这个新的接口。

最后是附录 A，“Z80-SIO 内部寄存器介绍”，叙述了 Z80-SIO 中每一个内部寄存器的操作。

正如你所看到的，这本书在所有重要的论题方面都提供了详尽的信息，使您能更好地理解 Z80 微处理器。有了这些知识，你就能从事微处理机控制系统的任何设想或应用了。当你开始实现你的设想和应用时，就会发现，使用 Z80 微处理器不仅很容易而且也是很有乐趣的。因此，丢开你原来的印象，开始学习吧，让 Z80 为你工作。

前　言

在微型计算机领域中，由于Z-80的功能强、用得久、用得广，因而在八位微型机方面占有重要地位。本书着重叙述Z-80微处理机系列，由于8080微处理机部分器件与Z-80系列相兼容，因此本书也间接叙述一些8080系列中的典型器件。

微机的发展史就是其应用的历史。微机的应用是其发展的动力源泉，而微机应用的关键在于接口及其芯片以及汇编语言的使用。本书正是从这个目的出发而编写的。内容十分精炼，叙述由浅入深，概念清晰准确。该书既可作为Z80单板机和系统方面的培训班教材，又可作为自学参考书。

本书译自《Z80 Applications》(美国，1983年版)一书，由张建平译，叶安麒、沙建军和向子明等同志初校，全书译稿由陈伟昌工程师审校。由于时间仓促，水平有限，加之本书急于付诸使用，因此书中难免有疏漏和欠妥之处，请读者不吝指正。

译校者 1984.9.

目　录

第一章　Z80 中 ROM 的使用

引　言

现在我们开始学习Z80微处理器与只读存储器(ROM)的连接。首先，我们对ROM的操作作个一般的介绍，然后讨论ROM和Z80应用的几个重要方面。通过本章的学习，读者可以完整地了解ROM和Z80之间的通信方式，并能把这些知识用到各自的系统中。

1-1　什么是ROM?

大多数微处理机系统中都有这样一种存贮器，它可以存贮信息而且在电源断开后信息也不会丢失。这种存贮器就称为只读存贮器，简称ROM。ROM中的信息可以读出，但不能改变。ROM在微处理机系统中是很有用的，它在电源接通时，控制中央处理单元(CPU)使所有的外围硬软初始化，并把它们置成适当的逻辑状态。

微处理机系统中的非易失性存贮器有许多种类型：包括只读存贮器(ROM)，可编程只读存贮器(PROM)，可擦除的可编程只读存贮器(EPROM)，以及电可改写的只读存贮器(EAROM)(见图 1.1)。

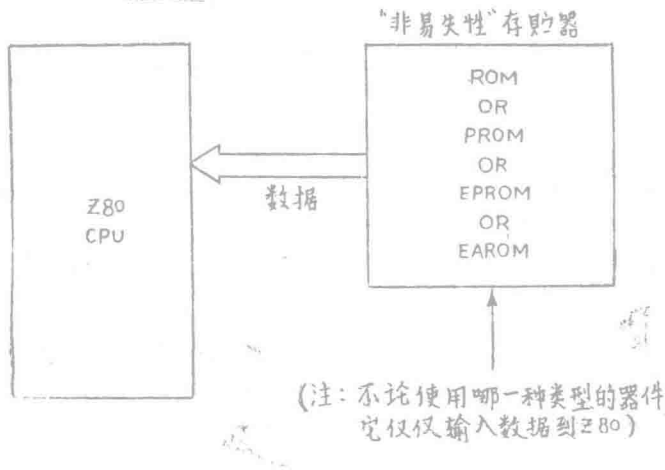

图 1.1　用于一个微处理机系统中的非易失性存贮器方块，非易失性ROM通常存放引导程序，在一接通电源时，首先执行这个程序，

在一个典型的Z80系统中，通常只使用一种类型的非易失性存贮器。但是，在你的应用中则可能要用到上述各类型的ROM，所以，我们把各种类型都介绍一下：

ROM：只读存贮器。制造厂把数据编程后送入存贮器。ROM用于大容量的不变数据的场合。(注：把数据编程送入ROM的过程是代价很大的)

PROM：可编程只读存贮器。由用户把数据编程后送入存贮器。在写入时，高电压脉冲加在(或称"烧")集成电路的金属片或多晶硅上，把逻辑"1"和"0"写入指定的地址单元。

数据一送入，就不能改变了。这种存贮器的操作速度一般比其它可编程存贮器的速度快得多。

EPROM：可擦除的可编程存贮器。用户利用高压信号把数据编程送入存贮器。这和PROM编程相似。向复盖在集成电路上的透明窗照射紫外线，就可以把所存的数据擦除。经过规定的曝光时间之后，全部数据都可以擦除，然后可以用新的数据重新对它编程。这种存贮器常用于那些程序需要经常改动的开发工作。

EAROM：电可改写的只读存贮器。这种存贮器由用户把程序数据送给它，其方法类似于EPROM。它们之间的主要差异是，EAROM的内容可用电来擦除而不用紫外线。

无论使用的是哪一种非易失性存贮器，它们的操作特性都是很相似的。下面我们将来讨论这一类存贮器的几个重要参数。（注：本书中所用的ROM一词泛指上面提到的各种非易失性存贮器）

1-2 ROM的主要操作特性

现在我们来考察一下ROM的一些重要操作特性。首先，ROM只能被CPU读出，因而命名为“只读存储器”。其次，地址加到地址输入线上时就可取出ROM中的信息，ROM中地址输入线的数目取决于ROM中数据的内部结构。

注意一下ROM的规格，就可以确定存贮器中数据的内部结构。例如，ROM的结构可以是1024×8，2048×8，4096×8等几种规格。其中第一个数字表示一块集成电路片上所包含的地址单元数。第二个数字则表示从每个独立的地址单元中读出的并行数据的位数。

下面的步骤能帮助我们确定ROM的地址线数。如果我们说存贮器的规格是2048×8，则2048就表示独立的地址单元数，而且也是地址线上二进制的不同组合数。换句话说，$2^X=2048$，即X=11其中X是地址线的数目。因此，如果存贮器为4096×2，则$2^X=4096$，即地址线应为X=12条。正象你所看到的那样，这个等式适用于任何地址单元数的ROM。

注意，地址单元的确切数目往往用比较接近的千位数表示，并简写为“K”（即千）。例如1024×8的ROM可以表示为1K，4096×8的ROM可以表示为4K等等。

ROM的最后一个操作特性就是，所有数据都是以并行方式从ROM中读出。换句话说，当允许数据向采样数据总线输出时，微处理器把数据选通至一个内部寄存器。

图1.2是ROM操作功能的方块图。在这个图中，我们看到一个输入，称作“片选”（chipselect）。这条输入线的功能就是接通或关断ROM的数据输出线。在片选输入线有效时，ROM的数据输出线也有效，并根据编程的数据，或输出逻辑“1”或输出逻辑“0”。在片选无效时，数据输出就处于第“三态”（tri-state），即高阻抗状态。换句话说，片选输入线可允许或禁止ROM的输出。在本章的后面部份，我们要确切地说明这条线的使用方法。

1-3 从ROM读取数据时的电信号序列

现在我们讨论图1.2。每次从ROM中读取数据时，都有一个电信号序列发生（下面列出）。请记住，无论使用哪一种CPU，都一定要按照这个序列读数。现在我们就对这个序列进行分析（后面还要专门讨论Z80怎样完成这个序列）。

1. CPU把一个地址输入给ROM，这个地址指出了数据所在的内部位置。ROM中有

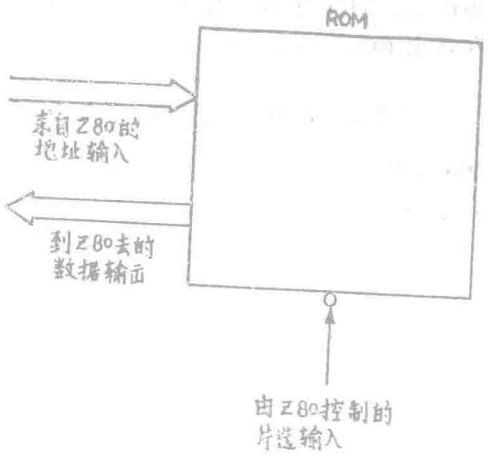

图 1.2 一个典型 ROM 器件的功能方块图。地址输入用来选择要输出的内部数据。当器件被选中时，就输出数据。

成千上万字节的数据，而地址则从中找出一个字节作为输出。

2。然后 CPU 等待一段时间(称作存取时间，时间的长短随 ROM 不同而异，它大约为 100～300 毫微秒)，因而使存贮器能对地址进行译码，并把选出的数据送到数据输出线上。

3．片选有效，使数据输出到系统数据总线上。这时，ROM 的数据就出现在系统数据总线和 CPU 的数据输入引脚上。接着，CPU 就把数据选通到一个内部寄存器。

4．片选无效，从系统数据总线上消除 ROM 的数据。

CPU 每次按照这个序列从ROM读取数据。图 1.3 是这个序列的一般时序图。

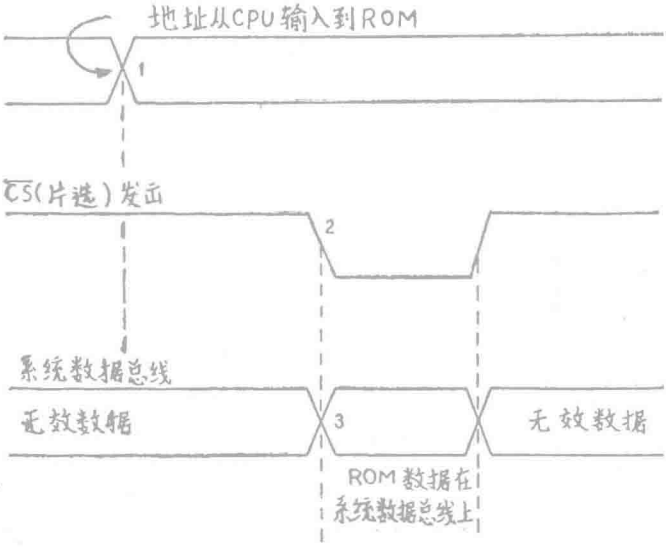

图 1.3 从 ROM 中读数据的一般时序：

1．地址从 CPU 输入到 ROM 器件。 2．发出片选。

3．允许 ROM 数据输出到数据总线上。

CPU 都是设计成在这个范围内进行操作的。微处理器借助于内部的时序电路控制着这个序列，因此在大多数情况下，用户都不必去考虑这个序列。不过，对它有一定的了解也是很重要的，因为这样在使用 ROM 及理解电路的操作原理时就简单多了。

1-4 Z80 总线的连接

现在我们把 Z80 和 ROM 连接起来。图 1.4 说明了 Z80 的数据线和地址线与 ROM 的实际连接方法。这些连接都是很简单的。注意，在图 1.4 中 2716 EPROM 的片选线没有连接，这是因为我们准备先讨论地址映象。

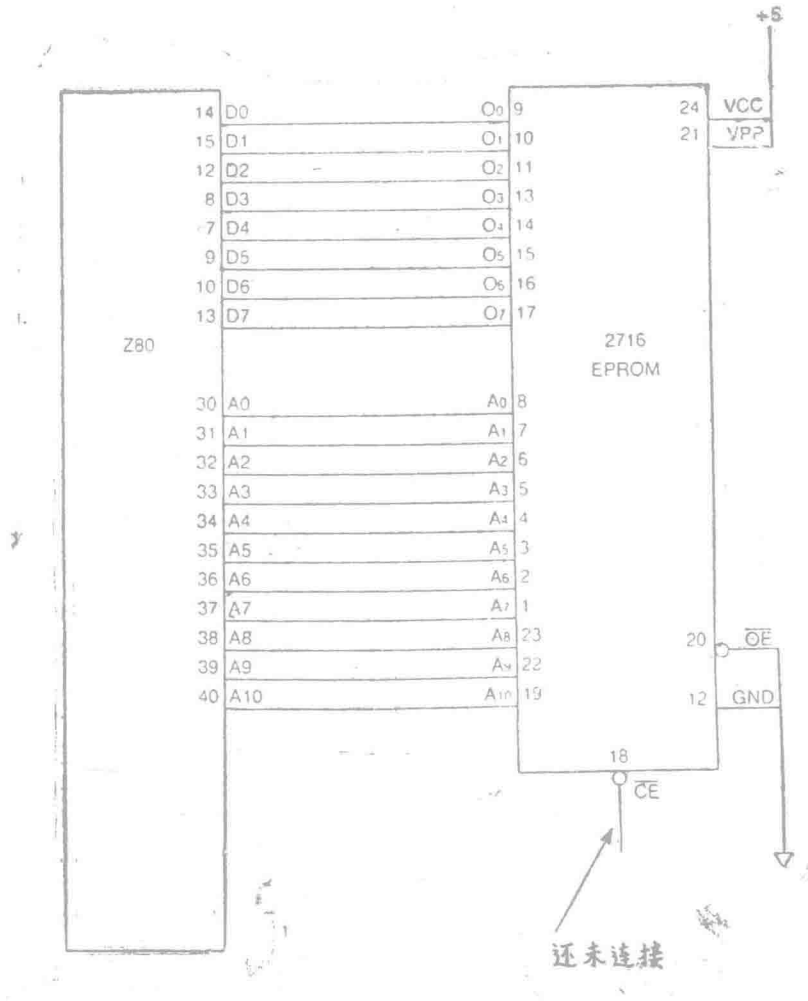

图 1.4 Z80微处理器和 2716 EPROM 之间采用无缓冲的地址和数据线连接图示。

1-5 地址映象

Z80 共有 16 条物理地址输出线，标为 A_0—A_{15}。这就是说Z80能直接访问 2^{16} 即 65,536 个独立的存贮单元。例如，2716 EPROM 有 2048 个物理地址单元，因此就需要有一种方法从总共65，536 即 64K 个可能单元中仅选出 2048 个单元。换句话说，我们选择哪

2048 个单元呢？65，536 除以 2048 等于 32，就是说我们有 32 种可能的选择方法。

在建立硬件系统之前，设计者必须先构造一个存贮器映象。存贮器映象用来指定给任何特定的 ROM，RAM，输入或输出设备的地址单元。图 1.5 就是一个典型的存贮器映象。在这个图中我们可以看到，整个 65，536 个地址单元被分为若干个功能块。这些块指明了留给 ROM 和 RAM 的地址单元。

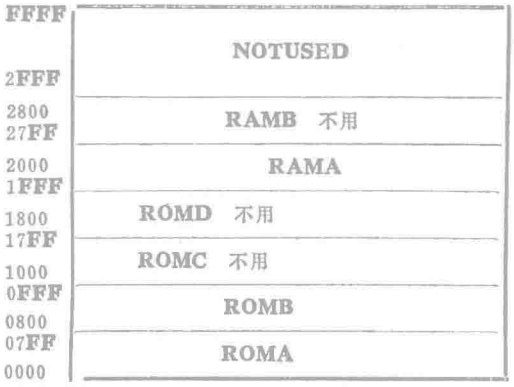

图 1.5 一个典型微处理机系统的存贮器映象。在设计任何硬件之前，通常要构造一个存贮器映象，指明怎样分配存贮空间。

注" 如图 1.5 所示，Z80 最低地址值(即 0000H)通常留给 ROM。这因为外部总清开关或开机总清电路总清Z80时，地址总线总是要求第一条指令操作码存放在 0000H 地址单元。

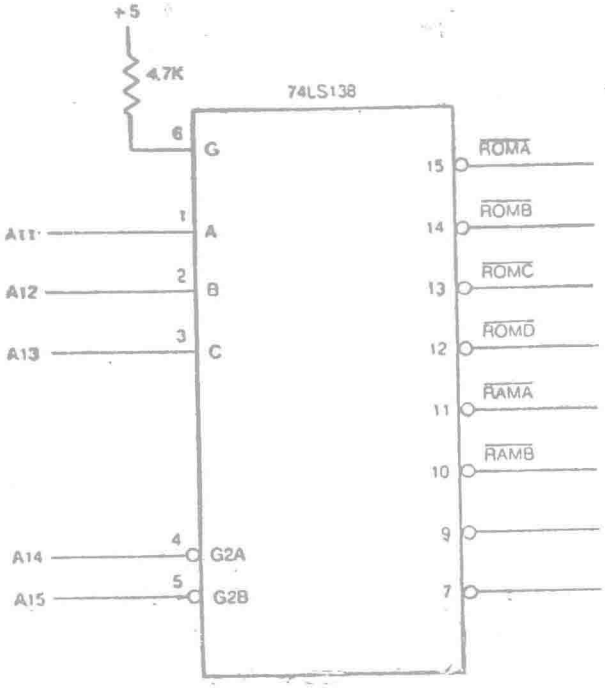

图 1.6 实现图 1.5 中存贮映象的硬件电路

1-6 生成ROM片选信号

由图1.5我们可以看到，每当Z80输出0000H—07FFH之间的地址时，ROMA就被允许。为了指出地址总线上的地址是在这个范围之内，就需要有一个电信号，图1.6示出了这样一种电路。

在图1.6的电路中，每当A_{11}，A_{12}，A_{13}，A_{14}和A_{15}都是逻辑"0"时，74LS138的输出引脚15就为逻辑"0"。在限定的地址范围内，引脚15保持为"0"，图1.7的表中列出了74LS138的输出引脚与地址总线上地址输出的对应关系。

A15	A14	A13	A12	A11	A10----------A0	HEX	PIN*=0
0	0	0	0	0	0 ---------- 0	0000	15
0	0	0	0	0	1 ---------- 1	07FF	15
0	0	0	0	1	0 ---------- 0	0800	14
0	0	0	0	1	1 ---------- 1	0FFF	14
0	0	0	1	0	0 ---------- 0	1000	13
0	0	0	1	0	1 ---------- 1	17FF	13
0	0	0	1	1	0 ---------- 0	1800	12
0	0	0	1	1	1 ---------- 1	1FFF	12
0	0	1	0	0	0 ---------- 0	2000	11
0	0	1	0	0	1 ---------- 1	27FF	11
0	0	1	0	1	0 ---------- 0	2800	10
0	0	1	0	1	1 ---------- 1	2FFF	10

图 1.7 图1.6的存贮器映象

从图1.7中我们可以看到，当A_{11}变为"1"时，74LS138的引脚14转向逻辑"0"，并且引脚1为逻辑"1"。这个结果表明，仅当Z80的地址输出是在限定的0000H—07FFH之间时，引脚1才为"0"。同样的推也可以应用于表中其它的输出引脚。

图1.6电路只是在地址空间中选择一个唯一地址块的方法。(还有许多种方法)

1-7 存贮器读信号的产生

现在我们已几乎可以把片选输入与Z80连接起来，但是在此之前，我们首先必须学习一下Z80发出的，能够启动和停止数据传送的定时控制信号。这个信号称为存贮器读或$\overline{MEMR}$信号。$\overline{MEMR}$信号和存贮器选择信号一起使用，就能使ROM在正确的时间输出数据至系统数据总线。图1.8是Z80生成$\overline{MEMR}$信号的一种方法。

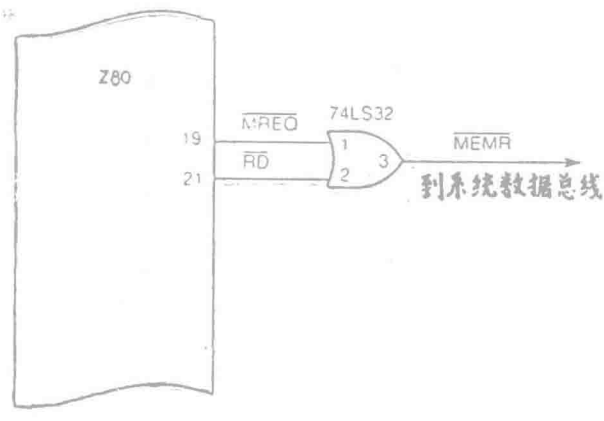

图 1.8 把Z80微处理器的控制信号"或"起来生成系统存贮器读控制信号的电路。

在图 1.8 中，当 Z80 与存贮器通信时，$\overline{MREQ}$ 置为逻辑“0”，这个动作是由由软件确定的。$\overline{RD}$ 是 Z80 发出的定时控制信号，每当 Z80 准备接收从存贮器或输入/输出设备来的数据时，它就变为逻辑“0”。

1-8 片选线的连接

Z80 与 ROM 的完整连接如图 1.9 所示。当存贮器选择和存贮器读($\overline{MEMR}$)信号都为逻辑“0”时，传送到 2716EPROM 的片选输入逻辑“0”有效。图 1.10 是 ROM 读期出间的时序以及重要的信号

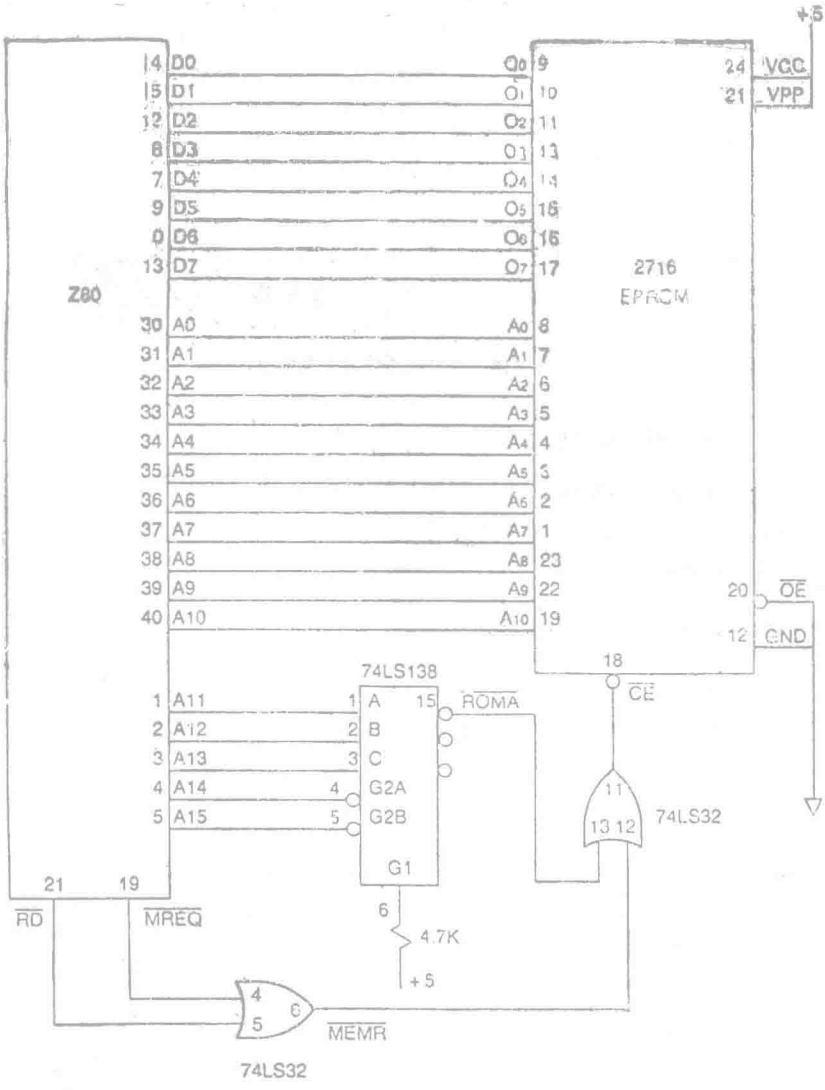

图 1.9 2716 EPROM 和 Z80 微处理器之间的完整连接图。本图包括了用来按照存贮器映象来使存贮器工作的逻辑。

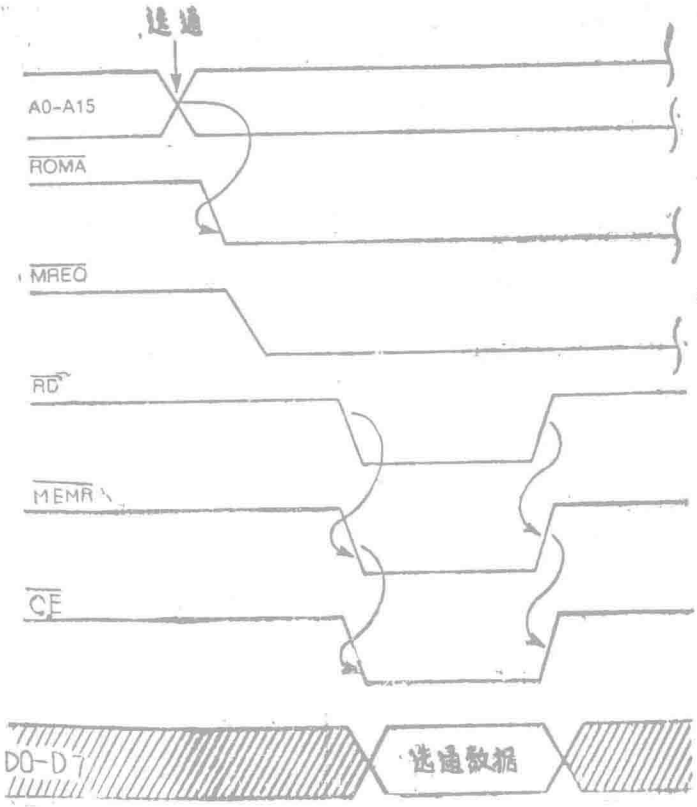

图 1.10　图 1.9 中重要信号关系的时序

1-9 另一种方法

图 1.11 是另一种能使 ROM 数据准时输送到系统数据总线上的技术。它利用2716内部的译码功能。在 ROM 的数据向系统总线输出之前，引脚 20 的 $\overline{OE}$ 和引脚 18 的 $\overline{CE}$ 信号都必须为逻辑"0"。按照先前的方法，引脚 20 的 OE 输入接地，这个输入总是处于有效状态。

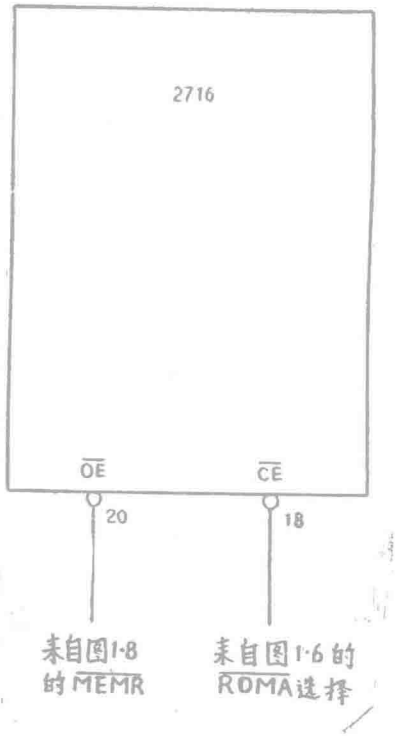

图 1.11　使用输出允许引脚 20 及片选引脚 18 控制 2716 器件的图示。