蔡莲红　黄顺珍　等　编著

APPLE
微型计算机系统（上）

——主机和监控系统

清华大学出版社

ISBN 7-302-00282-7
TP · 106 定价: 1.35 元

APPLE微型计算机系统

（上）

——主机和监控系统

蔡莲红　黄顺珍　方棣棠　吴文虎
胡起秀　　李树青　　钟玉琢　编著

清华大学出版社

内 容 简 介

Apple Ⅱ微机硬件设计巧妙、简单可靠，软件灵活紧凑，是国内外广泛使用的微型计算机系统。本册介绍了Apple Ⅱ plus主机硬件、监控系统及小汇编的工作原理。第一章为Apple微型计算机系统结构。第二章为主机硬件的组成和工作原理。第三章为Apple Ⅱ使用指南。第四章为监控程序分析。第五章为小汇编的工作原理。

本书力图将硬件和软件结合起来，深入浅出地讲清该机的工作原理，为系统维修人员、程序员以及广泛开发应用微机的人员提供详尽的资料。可供大学生和有关科技人员借鉴。

Apple 微型计算机系统

（上）

——主机和监控系统

蔡莲红　黄顺珍　方棣棠　吴文虎　编著
胡起秀　　李树青　　钟玉琢

责任编辑　贾仲良

☆

清华大学出版社出版

北京　清华园

北京昌平振南排版厂排版

中国科学院印刷厂印刷

新华书店北京发行所发行

☆

开本：787×1092 1/16　印张：15 1/8字数：377千字

1988年11月第1版　1988年11月第1次印刷

印数：00001—10000　　定价：5.35元

ISBN 7-302-00282-7/TP·106

目　录

前　言

目前，微型计算机在新技术革命的浪潮中起着越来越大的作用。由于它具有体积小，耗能少，价格低廉，性能可靠，操作维护简便，易于学习掌握，以及使用灵活等许多优点而受到普遍重视，正在被人们作为信息存储、转换、处理和传递的强有力工具，广泛应用于工业控制、国防设施、企业管理、科学研究、计算机教学等各个领域。

微型计算机有着自己的特点，要用好它，就需要学习掌握它。我们自1979年起，用"解剖一个麻雀"的方法解剖了Apple Ⅱ微型计算机，那时这种微机刚刚问世，现在这种微机在国内的用户已相当多。当时我们对Apple Ⅱ的硬件组成和监控系统进行了深入的分析研究，写出了分析报告。自1979年以来，经多人反复修改，先后五次易稿印刷，每次都因印量少而满足不了需求。这次我们又在举办多次学习班，倾听了学员及其它读者对该教材的反映，并作了较大修改之后才交付正式出版。我们写这本书的指导思想是详细分析该机的硬件组成，找出特点。对许多难点部分，比如显示彩色图形的原理与硬件，花较大的篇幅进行了阐述；监控程序部分也是逐条分析，理出头绪，找出规律，力图从软件与硬件的结合上把该机的各种基本功能分析清楚。这样做有利于掌握微机的原理，也为维护及进一步开发其功能奠定了基础。我们感到主机硬件组成和监控程序是微机本身的核心部分，也是掌握Apple Ⅱ的重点所在。因此，本书主要介绍这两部分内容。其它问题，只要有参考手册，是不难通过上机操作和编程实践来学会的。

由钟玉琢组织编写的本书的第一稿为本次出版打下了良好的基础。这次出版是在原有基础上，对全书的各个部分作了较大修改、补充，或添加进一些新的内容。硬件部分由蔡莲红、方棣棠、吴文虎、胡起秀、黄顺珍等人编写；软件部分的监控系统由黄顺珍编写，小汇编由李树青编写。

曾经参加分析解剖Apple Ⅱ工作，以及撰写分析报告的还有涂连华、徐风家、苗玉峰等人。

近年来，国内外研制了不少以6502为CPU的微型机，例如紫金Ⅱ、DJS-033、中华学习机，它们都与Apple Ⅱ兼容，为此作者特意写了附录D作了简介。本书对应用、开发这些兼容机也具有极好的参考价值。

限于编者水平，书中难免有错误与不当之处，望读者不吝指正。

编　者

第一章　Apple微型计算机系统结构

1.1　概　　述

Apple 微型计算机系统是微计算机系统中的低挡（即初级）机，或称个人计算机。市场上较为流行的是 Apple Ⅱ plus，这是微型计算机发展史上的一个典范。83 年以来，国内除进口了不少此类微机外，还开发了许多种Apple的兼容机，如紫金-Ⅱ、DJS-033。1987年，又研制了紫金-ⅡA（Apple Ⅱe 兼容机）以及 CEC-Ⅰ中华学习机、小蜜蜂等。以上机型的 MPU 均为 6502。它突出的特点是：价格便宜、结构灵活、使用方便，配套的硬件、软件十分丰富。

Apple 的主机内，有MPU、48K字节的随机存取存储器、12K字节的只读存储器，以及键盘和部分外设接口。若采用家用电视机作为字符和图形的显示设备，家用录音机作为外存储器，这样就构成了最便宜的Apple系统。

Apple 的硬件和软件都采用积木式结构。主机备有八位通用外部设备接口插座，能连接 800 多种接口卡，大大扩充了系统功能。若在外设插座上插上不同的微处理器卡，Apple 就变成了另外的机型，如插上 Z-80 soft card，再装入 CP/M 操作系统，能运行Z-80的所有高级语言软件包，执行Z-80、8080和8085的汇编语言。Apple机图文并茂、有声有色。图形颜色多达十六种，在高分辨率图形方式下，可以六种颜色控制到扫描光点。

Apple机使用方便。开机后自动引导磁盘操作系统或进入高级语言，同时，对工作环境无特殊要求。

在微机中，Apple 机的产销量名列前茅，它被广泛应用于科学计算、数据处理、企业管理、游戏和娱乐等各个领域。

Apple 的系统结构如图 1.1.1，下面将分别介绍该系统在软件和硬件上的一些特点。

1.2　主 机 硬 件

Apple是以Rockwell公司的6502为MPU的微处理机，时钟频率为 1MHz，八条数据线，十六条地址线。主机设计灵活巧妙、结构紧凑，很有独到之处。

1．存储器

主机大板上设有48K字节的RAM和12K字节ROM。ROM中存有监控程序(Monitor)和浮点BASIC（Apple soft）解释程序。RAM可扩展为 64K字节（或更多）。RAM扩展后，主机可使用其它高级语言，如PASCAL、FORTRAN 等。

2．显示

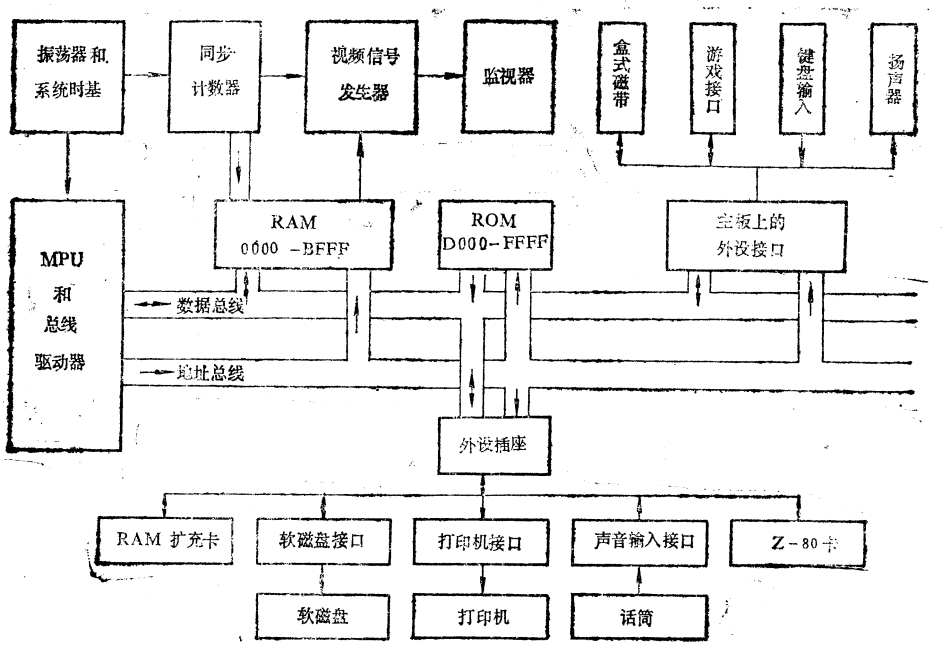


图 1.1.1 系统结构框图

Apple 的监视器可采用一般视频监视器或家用电视机。若采用家用电视机（制式）作显示，要将视频信号经一个视频信号调制器调制成高频，从电视机的天线输入端输入。调制器电路非常简单。若采用视频监视器，由主机后板备有的电缆插座输出视频信号，只要用电缆接通监视器和主机即可。

显示可有三种方式、多个页面。三种方式是：文本方式（40×24 字符）、低分辨率图形（以十六种颜色显示40×48个彩色方块）和高分辨率图形（以六种颜色显示 280×192彩点）。每种显示有两个页面。显示窗口可任意缩变，也可采用图形和文本混合显示的方式。

显示的内容存在主板的RAM中。视频信号发生电路简单、设计巧妙。

3．主机内的I/O接口

主机内的 I/O接口有：盒式录音机输入、输出；扬声器输出；键盘输入和游戏接口（四个模拟输入、三个一位输入、四个一位输出）。这些接口大大增加了主机的功能，也为扩充主机的使用范围提供了方便。接口的特点是硬件尽量简单，以降低主机的造价，其余的工作由软件完成。

1.3 系统装置和接口卡

Apple采用积木式的结构，因此系统组织灵活、结构紧凑，整个系统可大可小、变化多样。目前一般的系统组成为：主机、监视器、$5\frac{1}{4}$英寸软磁盘和打印机。

主板上装有八个外设接口插座，可以插上不同的接口板，带上不同的外部设备，除打印机、软磁盘外，还可连接图形输入板、多笔绘图仪及各种工业仪器仪表。到目前为止Apple的接口达几千种，比较常用的有：

- 串行接口卡（SERIAL INTERFACE CARD）
- 通信接口卡（COMMUNICATIONS INTERFACE CARD）
- 并行打印接口（PARALLEL PRINTER INTERFACE）
- 并行接口卡（PARALLEL INTERFACE CARD）
- 高级串行接口卡（SUPER SERIAL INTERFACE CARD）
- 80列显示卡（80×24 DISPLAY CARD）
- IEEE-488接口卡（IEEE-488 INTERFACE CARD）
- 彩色编码卡（PAL CARD）
- 时钟/日历卡（CLOCK/CALENDER CARD）
- 业余/实验卡（HOBBY/PROLOTYPING CARD）
- 软盘驱动卡（DISK DRIVE CARD）
- A/D＋D/A卡
- 16k RAM扩充卡（16k RAM EXPANSION CARD）
- SOFT卡（SOFT CARD）即Z-80卡
- EPROM写入卡
- 汉字卡
- 8088卡
- EPROM写入卡
- Omminet网络传送卡
- 声音合成卡

接口板上，除具有优良的接口电路外，一般都配有固化在ROM中的软件，功能齐全、使用方便。在这些接口卡及软件的支持下，使Apple机具有小型机的功能。随着计算机技术的发展，为满足各领域的需求，接口卡会越来越多，Apple机的功能将越来越强，应用也将越来越广泛。

1.4 系统软件和语言

在Apple中，除利用汇编语言编写程序外，还可使用BASIC语言。在操作系统支持下，可以使用多种语言。

1．整数BASIC和浮点BASIC

浮点BASIC解释程序固化在Apple Ⅱ plus主机ROM中，开机便可使用，方便可靠。主机配有16k RAM扩充卡时，也可使用整数BASIC语言，快速简洁。

2．DOS3.3和高级语言

DOS 3.3是Apple的磁盘系统DISK Ⅱ子系统上运行的操作系统，在它的支持下，可使用BASIC、PASCAL、FORTRAN等语言。

3．CP/M操作系统和高级语言

插上Z-80卡，主机转变为高速Z-80机，它允许8085、8080、Z-80系统的软件在本系统中运行。可使用的高级语言有BAS80、FORTRAN-80、BASIC、COBOL、PASCAL、AOGOL-80和C语言等。

1.5 应用软件

Apple的应用软件极其丰富，到目前为止，其软件多达上万种，涉及到的方面有科学计算、商业、管理、个人事务、教育等等。下面简要列出几种：

- 中小型企业管理和财务软件（THE CONTROLLER）
- 制图软件包（Apple GRAPHICS Ⅱ）
- 数据库类软件
- 商业作图软件（Apple Ⅱ BUSINESS GRAPHICS）
- 通信录软件（Apple POST）
- 调试测试类软件
- 图表绘制软件（Apple PLOT）
- 网络控制软件
- 乐理教学软件（MUSIC THEORY）
- 电路辅助设计软件
- DOS程序设计辅助软件（DOS TOOLKIT）
- 汉字处理软件
- 智力游戏（THE SHELL GAMES），等等。

第二章　Apple Ⅱ主机的组成和工作原理

为便于分析阐述Apple Ⅱ微型计算机系统主机的组成和工作原理，现将主机分为以下几个部分：

1. MPU和系统总线；
2. 振荡器和系统时基；
3. 存储器的组成和译码；
4. 随机存储器的选择电路；
5. RAM地址多路开关；
6. 外设插座；
7. 主板上的外设接口；
8. 同步计数器；
9. 视频信号发生器；
10. 键盘。

这些部分由时基总线、数据总线、地址总线、地址译码线以及一些控制信号线相互连接在一起。组成上述各部分的器件，除6502微处理器和字符发生器外，大部分是中规模TTL型集成电路。

下面将详细介绍上述各部分的组成、基本工作原理、逻辑关系及其作用。

2.1　MPU和系统总线

MPU和系统总线是由微处理器6502、总线驱动器（8T97和8T28）及一些与门、或门和非门组成的。下面分别介绍6502微处理器、总线驱动器的结构及基本工作原理。

2.1.1　6502微处理器（MPU）

Apple Ⅱ系统的中央处理器采用ROCWELL公司生产的6502微处理器，以后简称MPU。其基本指令56条，共有指令151条，时钟频率1MHz。一条指令的执行时间为3～8μs。数据线8条，是双向驱动的。地址线16条，是单向驱动的。

1. 6502微处理器的内部结构

如图2.1.1所示，6502微处理器和一般微处理器一样，其内部结构包括三个基本部分：

（1）算术逻辑部件（ALU）

它既能执行算术运算（如加法和减法等），又能执行逻辑操作，如逻辑“与”（AND）和逻辑或（OR）等。

（2）控制器部分

包括指令寄存器、指令译码器、中断逻辑部件、时钟发生器和时基控制器。时钟发

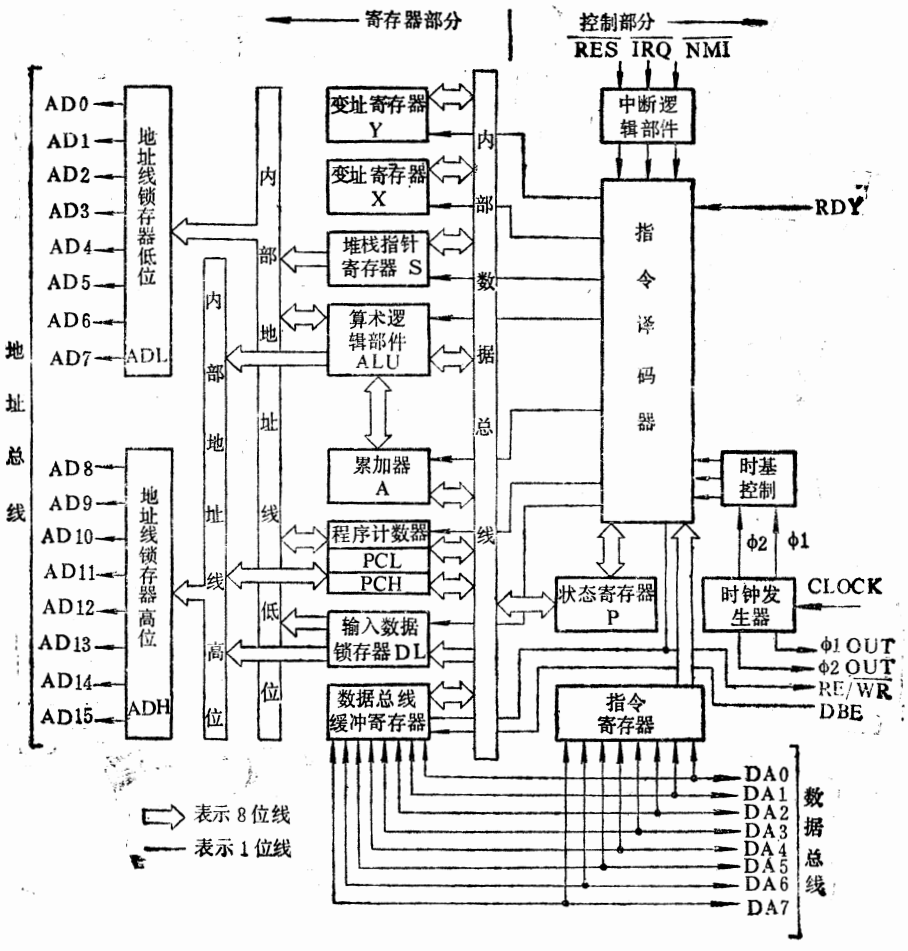


图 2.1.1 6502型微处理器的内部结构

生器提供定时的时钟脉冲。时基控制器提供一系列控制操作的控制信号。

（3）寄存器部分

包括变址寄存器（X）、变址寄存器（Y）、堆栈指针寄存器（S）、累加器（A）、程序计数器（高位PCH、低位PCL）、微处理器状态寄存器（P），这些寄存器被用来存放操作数、中间结果以及标志工作状态的信息等。

此外6502微处理器还有：数据总线缓冲寄存器、输入数据锁存器（DL）、地址总线锁存器（高位ADH、低位ADL）。

上述各部分在微处理器内通过内部总线互相联系。

2．6502微处理器的管脚引线及其功能

6502微处理器共有40条引线。其管脚分布如图2.1.2所示。

在图中：

V_{cc}　　电源线，电压＋5V。

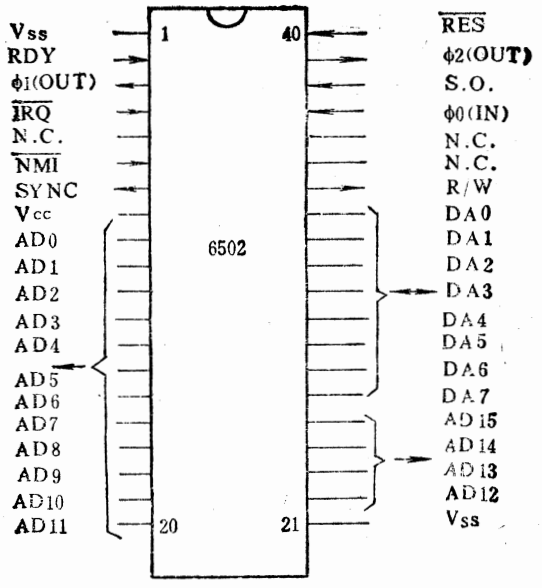


图 2.1.2 6502微处理器管脚图

V_{ss} 电源地线。

φ0 时钟脉冲输入线。时钟频率为 1.023MHz，它是由系统时基送给 MPU 的。6502微处理器内部还有一个时钟发生器，在φ0的控制下产生 φ1时钟脉冲。φ1 和φ0的关系为$\phi1=\overline{\phi0}$。

AD0～AD15 地址总线，共16条。每条引线单向输出。可带130pF的电容和一个标准的TTL负载。当处理器和随机存储器（RAM）、只读存储器（ROM）、外部设备交换信息时，这16条线给出地址码。地址总线上的信息，在 φ1 由低电位变为高电位约 300 ns 后才有效，并在 φ1＝0（低电位）期间始终保持有效。

DA0～DA7 数据总线，共8条。每条引线是双向驱动的，既可输入数据，也可输出数据。每条引线可带130pF的电容和一个标准的 TTL 负载。MPU和存储器之间数据和指令的传送都在这8条线上进行。当φ0＝1(φ0 为高电位)时，这条线上传送数据。当 φ0＝0（φ0 为低电位）时，这些数据线悬空（为高阻状态）。写周期时，数据线上的数据在φ0变高至少200 ns后才稳定可用。读周期时，数据在φ0 变低前应保持有效100 ns以上。

RE/$\overline{WR}$ 读写信号线，控制数据总线上数据的传送方向。读内存或外设时，这条线变成高电位，数据从系统总线进入MPU。写时这条线变成低电位，MPU把数据送给内存或外设。RE/$\overline{WR}$ 信号在 φ1 变高时发生变化，φ1变高后约300ns才稳定可用。

地址信号、数据信号、RE/$\overline{WR}$（读/写）信号与 φ0、φ1的时间关系如图 2.1.3 。

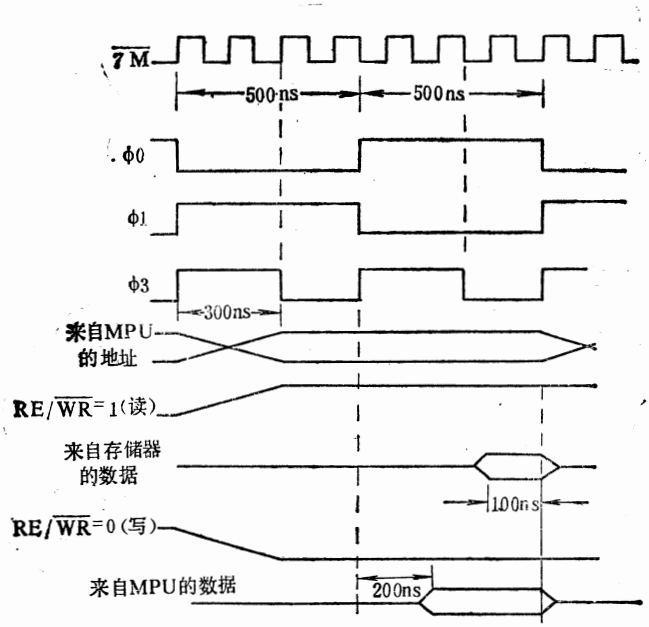


图 2.1.3 地址、数据、RE/$\overline{WR}$信号与φ0、φ1的时间关系

$\overline{NMI}$ 不可屏蔽的中断输入线。$\overline{NMI}$输入后，MPU一旦执行完当前的这条指令，MPU就被中断。此中断信号是不能被屏蔽的，即没办法阻止MPU承认这个信号。$\overline{NMI}$平时为高电位。当此线被外设拉低后，MPU就从内存的专用单元（FFFB和FFFA）取出新的程序计数向量（称作$\overline{NMI}$向量），也就是使PCH=(FFFB)；PCL=(FFFA)。而(FFFB)=03；(FFFA)=FB。然后MPU就从首地址03FB开始执行不可屏蔽的中断服务子程序。

$\overline{IRQ}$ 可屏蔽的中断请求输入线。平时为高电位，当外设申请中断时，把$\overline{IRQ}$拉低。$\overline{IRQ}$能被MPU中的状态寄存器的禁止位I屏蔽，禁止位I，一般由软件设置。如果中断禁止位标志I=1，中断被禁止，$\overline{IRQ}$信号不影响MPU。I=0时，中断请求才被承认。然后MPU从内存专用单元（FFFE和FFFF）取出新的程序计数向量（称为中断请求向量）也就是使PCH=(FFFF)，PCL=(FFFE)而(FFFF)=FA，(FFFE)=40。于是MPU就从FA 40首地址开始执行“中断服务子程序”。当MPU承认$\overline{IRQ}$之后，标志位I自动置1。MPU开始执行“中断服务程序”，这之后外设可使$\overline{IRQ}$变高。$\overline{IRQ}$变高后，一方面若I重新置0，可保证一个中断请求信号不致重复被承认如图2.1.4所示。另一方面若这时有更高级别的外设申请中断，那么刚才开始执行的“中断服务程序”也允许中断，以实现多级中断。

RDY “准备好”信号输入线。这条信号线被用来使MPU与存取速率较慢的外设同步。平时为高电位。当MPU与外设交换信息时，一旦MPU把地址信号送到系统地址总线上，接到外设插座上的外设就把RDY拉低，

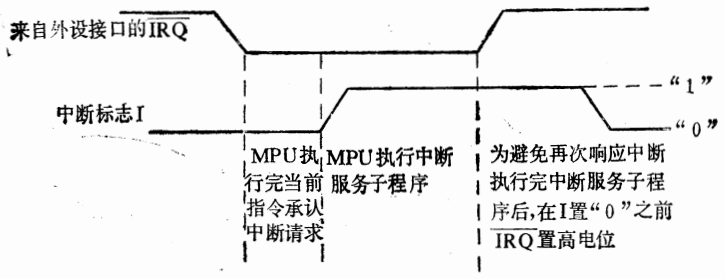


图 2.1.4 中断请求$\overline{IRQ}$与中断标志I的时间关系

MPU进入等待状态。只要RDY＝0，此等待状态就可持续，直到外设送到数据总线上的数据D0～D7有效后，外设才使RDY变高（即RDY＝1）。RDY＝1表示外设已准备就绪，已把数据送到系统数据总线。这时，MPU结束等待状态，开始执行当前的内部机器周期。

RDY线上的电位，在φ1＝1时改变，φ1＝0 时被承认，下一个 φ1＝1 时MPU才开始执行。

RDY线的主要目的是延迟一条程序读周期的执行，直到来自外设的数据有效为止。如果MPU正在执行写周期操作，则RDY不停止MPU的工作。当RDY在写周期由高变低，MPU将执行写周期的操作，紧接着在下一个读周期时（即RE/$\overline{WR}$＝1），MPU才停止工作。

$\overline{RES}$ 复位清零信号输入线。这个信号可以初始化微处理器。它可由合闸时的启动复位电路产生低电位；也可由使用者按RESET键产生低电位。$\overline{RES}$为低电位时，禁止从微处理器写入内存或外设接口。当$\overline{RES}$变成低电位后，6502微处理器将延迟六个周期，然后从内存的专用单元(FFFD、FFFC)中取出新的程序计数向量（称作复位向量）。也就是使程序计数器高位PCH等于FFFD单元中的内容，低位PCL等于FFFC中的内容。在自启动ROM中，FFFD中存的是FA，FFFC中存的是62。FA62是复位程序的入口地址。这样只要$\overline{RES}$变成低电位，MPU就从原来的工作状态中退出，自动进入复位程序并初始化MPU。

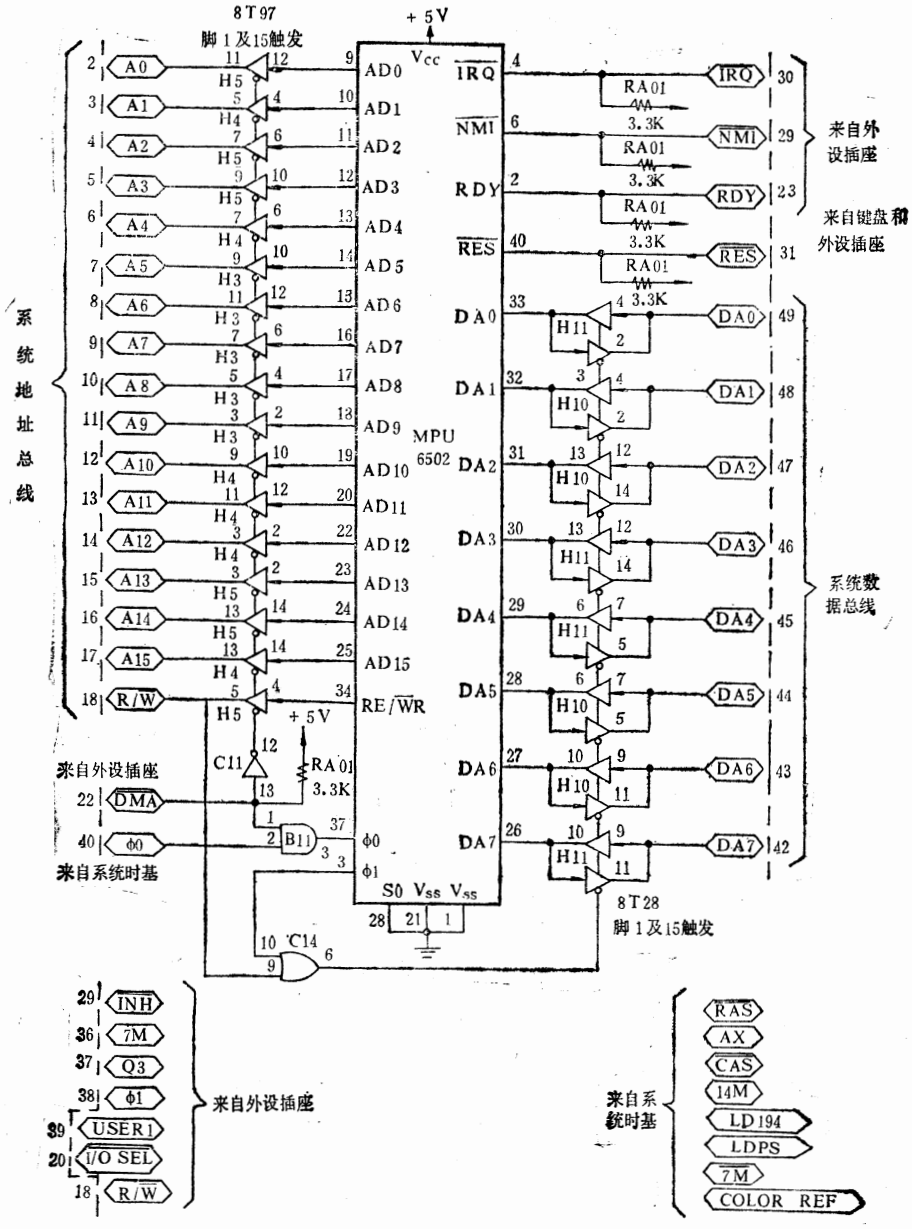
S.O 置状态寄存器第六位（溢出标志位V）。在Apple Ⅱ中此管脚接地，即不能由MPU的外面给MPU的溢出标志位置1。

图2.1.2中管脚7为同步信号SYNC线，脚39为φ2时钟信号输出线。Apple Ⅱ系统均未使用，故不再赘述。

2.1.2 MPU和系统总线阐述

1. A0～A15

Apple Ⅱ系统地址总线共16条。6502微处理器输出带负载能力较弱。为了增加输出带负载的能力，在6502微处理器的地址线AD0～AD15上接有单向驱动器8T97。当MPU向系统地址线送出信息时，因为8T97上的控制端1脚和15脚平时为低电位，所以



附注：虚线外的数字表示接到外设插座上的脚号

图2.1.5　MPU 和系统总线

MPU通过8T97把地址送到系统地址总线。当8T97上的控制端1脚和 15 脚为高电位 时，地址线AD0～AD15经驱动器的输出成高阻状态。MPU不给系统地址总线输出信息。系统地址总线A0～A15悬空，允许被其它设备控制。

在6502微处理器的读写信号线RE/$\overline{WR}$上也接有单向驱动器 8T97，其输出 为 系 统“读/写”信号R/$\overline{W}$。

单向三态驱动器上的控制信号端脚1和15是受$\overline{DMA}$信号控制的（DMA＝Direct Me-

mory Access). $\overline{DMA}$来自I/O外设插座脚24。当I/O设备要直接和存储器交换信息时，I/O设备使$\overline{DMA}$变为低电位。一方面$\overline{DMA}$的低电位经过C11反相器，脚12变成高电位，接到单向驱动器控制端1和15，使地址线AD0～AD15经驱动器的输出成高阻状态。MPU不再送出地址信息。同时，RE/$\overline{WR}$经驱动器的输出也变成高阻状态。另一方面$\overline{DMA}$的低电位封锁了B11门，处理器得不到从时基系统送来的时钟脉冲ϕ0，从而使6502微处理器停止工作。MPU与系统总线脱离。Apple Ⅱ系统处DMA状态。

2．D0～D7

Apple Ⅱ系统数据总线，共8条。6502微处理器的双向数据线DA0～DA7上接有双向三态驱动器8T28。双向三态驱动器受系统R/$\overline{W}$信号和MPU产生的时钟信号ϕ1的控制，逻辑电路如图2.1.6。

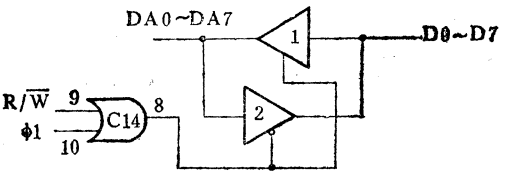


图 2.1.6　双向数据总线逻辑电路

按图2.1.6逻辑关系接的双向三态驱动器保证了当ϕ1＝1时或门C14的输出端8为高电位，这时MPU在数据总线上没有输出。

当ϕ1＝0时有两种情况：

（1）R/$\overline{W}$＝1　此时为读内存或读外设的情况。因为这时或门输出端8为高电位，三态门2为高阻状态。数据从总线D0～D7，经三态门1进入MPU的数据线DA0～DA7。数据总线D0～D7上的数据，在ϕ1变高前应保持有效100ns以上。

（2）R/$\overline{W}$＝0　此时为写内存或写外设的情况。这时或门输出端8为低电位；三态门1为高阻状态。数据从MPU的数据线DA0～DA7经三态门2输出到系统数据总线D0～D7。MPU送到数据总线上的数据，在ϕ1变低200ns之后稳定可用。

3．$\overline{RES}$

复位清零信号线。此线通过3.3kΩ　电阻接到＋5V上，平时为高电位。$\overline{RES}$来自键盘联接器，一方面接到MPU上，另一方面接到I/O外设插座上。

4．6502微处理器的信号输入端

中断请求信号输入端$\overline{IRQ}$、不可屏蔽的中断信号输入端$\overline{NMI}$、“准备好”信号输入端RDY都通过3.3kΩ电阻接到＋5V上，平时保持高电位。同时这些端点又接到I/O外设插座上，供I/O设备与MPU交换信息时使用。

5．系统总线还包括由时基系统获得的信号

$\overline{RAS}$　行选通信号，频率为2MHz；

AX　RAM地址选择开关的时间选通信号，频率为2MHz；

$\overline{CAS}$　列选通信号，频率为2MHz；

14M　14.318MHz信号；

LD194　视频信号发生器中移位寄存器74LS194的加载脉冲信号；

$\overline{LDPS}$　同步计数器加载信号，频率1MHz；

$\overline{7M}$　7M非信号，频率为7.159MHz；

COLOR REF　彩色参考信号，即彩色副载波信号，频率为3.58MHz。用于视

频发生器电路。

上述信号是怎样产生的，将在下一节中详细介绍。

2.2 振荡器和系统时基

振荡器和系统时基的原理图如图2.2.4（见下面）所示。从这张图可以得知，Apple Ⅱ微处理机系统的振荡器和系统时基信号产生电路是由74S175（四D触发器）、74S195（四位移位寄存器）、74LS153（双四选一选择器）、XT001（晶体）、2N4258（晶体管）、与门、或门、非门等组成。下面分别介绍主要时基信号产生的原理及各时基信号的时间关系。

2.2.1 14M信号

14M信号，由晶体振荡器产生。其电路如图2.2.1所示。

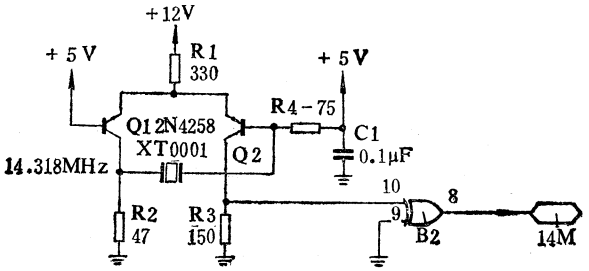


图 2.2.1 晶体振荡器电路图

图中XT001晶体、2N4258晶体管、电阻、电容组成晶体振荡器。其输出振荡信号再经异或门B2整形输出，就是14MHz信号。实际振荡频率为14.318MHz。14MHz信号是产生系统其它时间信号的基准。

2.2.2 7M和$\overline{7M}$信号

7MHz信号由14MHz信号经过二分频获得。二分频电路是由位于主电路板B1的D触发器实现的。为便于叙述，现在把位于主电路板B1的四D触发器中的四个D触发器分别命名为0号、1号、2号和3号D触发器。二分频逻辑图及波形图如图2.2.2。图中D触发器Q0的输出为7M信号。实际频率为7.159MHz。$\overline{Q0}$的输出为$\overline{7M}$信号。

2.2.3 3.5M信号

由14M信号经四分频获得。四分频电路是由位于主电路板B1的0号、1号D触发器和位于B2的异或门组成。其四分频逻辑图如图2.2.3。

下面分析一下四分频电路的工作过程。如果从零状态开始。Q0＝0，Q1＝0，经B2

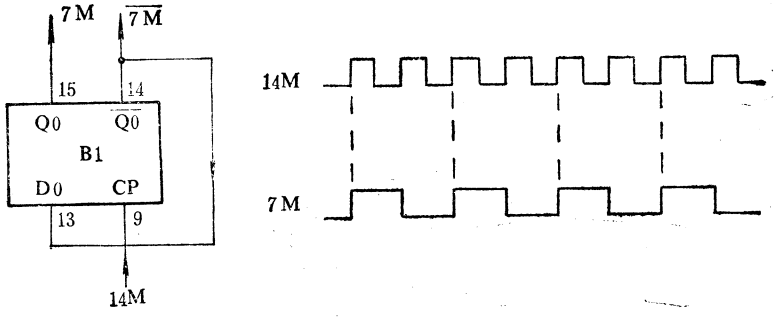


图 2.2.2 二分频逻辑图及波形图

异或后，D1＝0，又有D0＝$\overline{Q0}$＝1。第一个时钟脉冲过后，Q0＝1、Q1＝0，经B2异或后，D1＝1，又有D0＝$\overline{Q0}$＝0。第二个时钟脉冲过后，Q0＝0、Q1＝1，经B2异或后D1＝1，又有D0＝$\overline{Q0}$＝1。第三个时钟脉冲过后，Q0＝1、Q1＝1，过B2异或后D1＝0，又有D0＝$\overline{Q0}$＝0。第四个时钟脉冲之后，Q0＝0，Q1＝0，依次类推，可以得到表2.2.1中的时间关系及波形图。

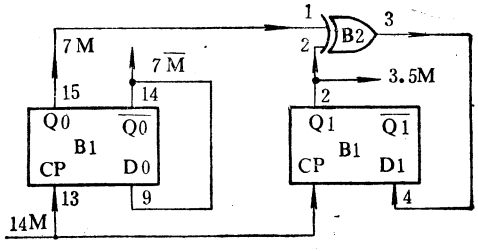


图 2.2.3 四分频逻辑电路图

表 2.2.1 二、四分频时间关系及波形图

CP	清0 1 2 3 4 5 6 7 8 0 ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑	频率Hz 14M	1 2 3 4 5 6
Q0	0 1 0 1 0 1 0 1 0	7M	0 1 0 1 0 1 0
D0＝$\overline{Q0}$	1 0 1 0 1 0 1 0 1	$\overline{7M}$	
Q1	0 0 1 1 0 0 1 1 0	3.5M	0 0 1 1 0 0 1
D1＝Q0⊕Q1	0 1 1 0 0 1 1 0 0		

逻辑图2.2.3中，四D触发器74S175中，2脚Q1的输出即为四分频信号，即3.5MHz信号，实际频率为3.580MHz。

2.2.4 $\overline{RAS}$、$\overline{CAS}$、AX、LD194、$\overline{LDPS}$及ϕ0信号

这些信号由14MHz信号经七分频获得。七分频电路主要由位于主电路板C2的四位移位寄存器74S195、位于C1的双四选一多路开关74LS153、位于B1的74S175中的二个D触发器及一些与门、与非门、或门组成。七分频电路的逻辑电路可参看图2.2.4。

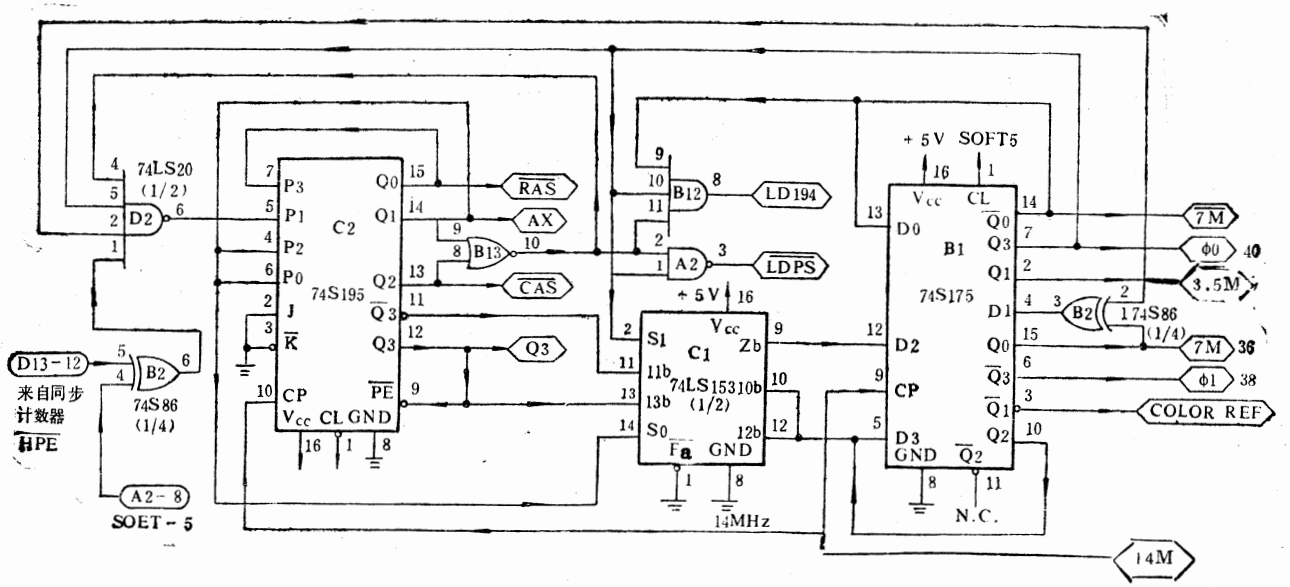


图 2.2.4 振荡器和系统时基原理图

为了便于说明七分频电路的工作原理及各种时基信号的产生过程，我们先分析四位移位寄存器74S195的工作过程。

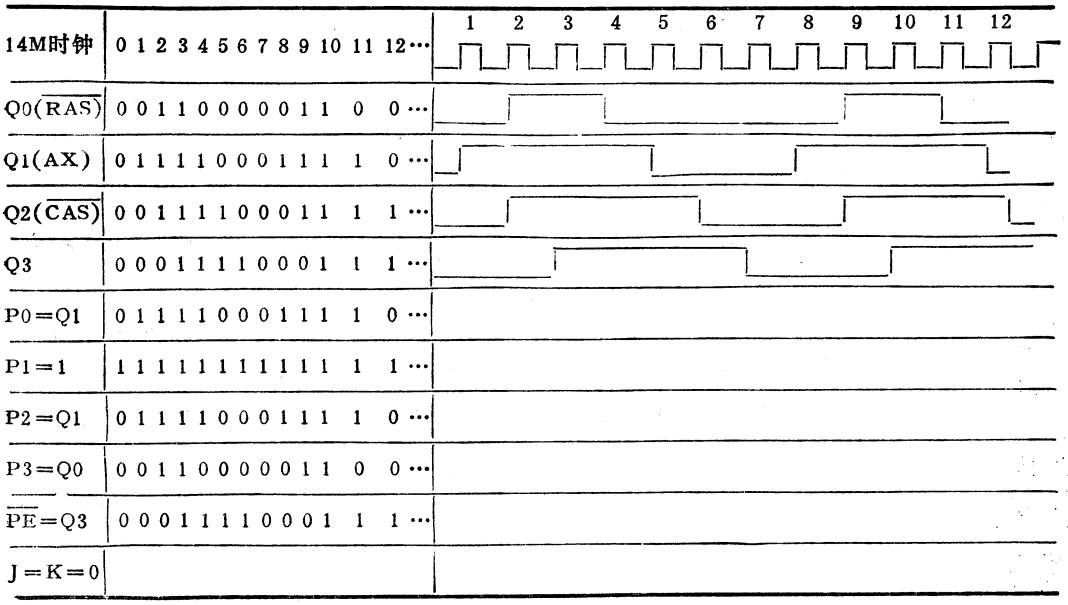
首先让我们看看四位移位寄存器74S195的功能：P0～P3是并行输入端，Q0～Q3是并行输出端，CP是时钟脉冲输入端。当$\overline{PE}$=0时，若来一时钟脉冲，则74S195置数。即P0～P3的信号送到Q0～Q3。当$\overline{PE}$=1时，若来一时钟脉冲，则74S195产生移位功能。此时若串行输入端J=K=0，则将0移到Q0，原Q0、Q1、Q2的信息移到Q1、Q2、Q3。74S195的详细功能请看书末附录。

其次我们将从第2.3节有关同步计算器的一段中得知，来自同步计数器的$\overline{HPE}$，64μs来一负脉冲，脉冲宽度为1μs，且SOFT5平时为高电位。因此74S195的P1输入端大部分时间是1。

另外，从图2.2.4连线可知74S195的J、K端是接地的。所以当$\overline{PE}$=Q3=1时，该片行使移位功能，将J、K的0移入低位。当$\overline{PE}$=Q3=0时，该片置数，P0～P3的信号送到输出端Q0～Q3。

根据上面三点和实际连线可得表2.2.2各信号的时间关系和波形图。

表 2.2.2　14M、$\overline{RAS}$、AX、$\overline{CAS}$、Q3的时间关系及波形图



下面简述四位移位寄存器74S195的工作过程。

如果从0状态开始，74S195的输出Q0～Q3均为0。由图2.2.4的连线知P0=P2=Q1=0，P3=Q0=0。P1=1，$\overline{PE}$=Q3=0。

第一个脉冲来时，因为$\overline{PE}$=0，74S195为置数状态，所以Q1、Q2、Q3分别变为0、1、0、0。P0=P2=Q1=1，P3=Q0=0，P1仍为1，$\overline{PE}$=Q3=0。

第二个脉冲来时，因为$\overline{PE}$=0为置数状态，所以74S195的输出Q0、Q1、Q2、Q3分别变为1、1、1、0。P0=P2=Q1=1，P3=Q0=1。P1仍为1，$\overline{PE}$=Q3=0。

第三个脉冲来时，因为$\overline{PE}$=0仍为置数状态，所以74S195的输出Q0～Q3均变为1，

P0～P3＝1，$\overline{PE}$＝Q3＝1。

第四个脉冲来时，因为$\overline{PE}$＝1，74S195变为移位状态，而且J＝K＝0。所以74S195的输出Q0、Q1、Q2、Q3分别变为 0、1、1、1。$\overline{PE}$＝Q3＝1。

第五、六、七个脉冲来时，由于$\overline{PE}$＝Q3＝1，74S195均为移位状态，所以均把J＝K＝0移入，使其输出Q0＝0，其它输出Q1、Q2、Q3依次移位。它们在不同脉冲时的状态如表2.2.2所示。

第八个脉冲来时，由于$\overline{PE}$＝Q3＝0，74S195变为置数状态。然后重复上述过程。最后可根据上述分析画出表 2.2.2 各信号的时间关系及波形图。图中信号$\overline{RAS}$、AX、$\overline{CAS}$、Q3分别对应于74S195的Q0、Q1、Q2、Q3的输出。

下面简述ϕ0（时钟脉冲）的产生过程，也就是在上述分析74S195四位移位寄存器的基础上，再进一步分析四选一多路开关及位于B1的D触发器的工作原理。

首先简单介绍一下四选一多路开关74LS153的功能。它的输出端Zb的电位是由选择端S1及S0的状态决定的。

即：

二进制		输出
S1	S0	Zb
0	0	0b
0	1	1b
1	0	2b
1	1	3b

其功能详见附录。

其次，我们可以看到，由移位寄存器的输出Q1、Q3、$\overline{Q3}$及连线图可得到表2.2.3七分频电路各信号的产生过程和图2.2.5即14M、ϕ0、ϕ1、AX的波形图。表中S0、1b 、3b分别来自移位寄存器的输出端Q1、$\overline{Q3}$、Q3。它们在不同脉冲时的状态可从表2.2.2中查出。S1来自D触发器的输出 Q3，Q3 由脉冲到来时D触发器输入端D3的状态决定。0b、2b来自D触发器的输出Q2，Q2由脉冲到来时D触发器的输入D2决定。根据S1、S0 的状态所表示的十进制数则可得到四选一多路开关的输出Zb。

表 2.2.3 七分频电路的产生过程

CP 14M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17 …
S0＝AX＝移位寄存器的Q1	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1	1 …
S1＝ϕ0＝D触发器的Q3	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1 …
0b＝D触发器的Q2	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1 …
1b＝移位寄存器的$\overline{Q3}$	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0 …
2b＝D触发器的Q2	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1 …
3b＝移位寄存器的Q3	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	1 …
D2＝多路开关的Zb	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1 …
D3＝D触发器的Q2	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1 …

在上述分析的基础上，让我们看看ϕ0的产生过程。

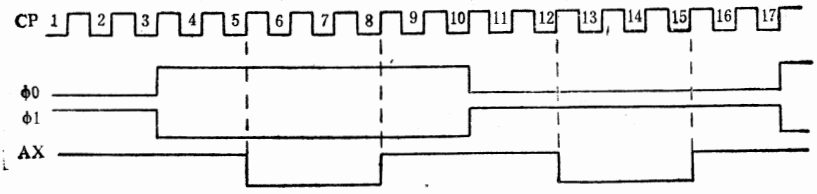


图 2.2.5 14M、ϕ0、ϕ1、AX的波形图

如果从0状态开始，74S195移位寄存器的Q1＝AX＝0，所以S0＝0。74S175 D 触 发器的Q3＝0，多路开关S1＝Q3＝0。由于S1＝0、S0＝0，故多路开关的输出Zb＝0b＝Q2＝0。而D触发器的D2＝Zb＝0，D3＝Q2＝0。

第一个脉冲来时，由移位寄存器的分析中知Q1＝AX＝1，所以S0＝1，D触发 器 的Q3，等于0状态时的D3＝0，Q2等于0状态时的D2＝0，多路开关S1＝Q3＝0，1b 等于移位寄存器的$\overline{Q3}$＝1。由于S1＝0、S0＝1，故Zb＝1b＝1，而D触发器的D2＝Zb ＝ 1 ，3D＝Q2＝0。

第二个脉冲来时，由移位寄存器分析中得知Q1＝AX＝1，S0＝Q1＝1。D 触发器的Q3，等于第一个脉冲时的D3＝0，Q2等于第一个脉冲时的D2＝1。所以多路开关S1＝Q3＝0，1b＝寄 存 器的$\overline{Q3}$＝1。由于S1＝0，S0＝1，故Zb＝1b＝1。而D触发器的D2＝Zb＝1，D3＝Q2＝1。

第三个脉冲来时，由移位寄存器分析中得知Q1＝AX＝1，S0＝Q1＝1。D触发器的Q3，等于第二个脉冲时的D3＝1，Q2等于第二个脉冲时的D2＝1，故多路开关S1 ＝ Q3＝1。由于S1＝1，S0＝1，故Zb＝3b，等于移位寄存器的Q3＝1，而D触发器的D2＝Zb＝1，D3＝Q2＝1……。

依次类推可得表2.2.3中七分频电路各信号产生过程和图2.2.5中的波形图。

又根据 LD194＝$\overline{7M}\cdot\phi0\cdot\overline{AX+\overline{CAS}}$，由 位于主电路板B12的与门输出脚8，可得到MHz的“LD194”信号。

又由于$\overline{LDPS}=\phi0\cdot\overline{AX+\overline{CAS}}$，由位于主电路板 A2 的与非 门 输 出 脚 3，可 得 到1MHz的“$\overline{LDPS}$”信号。

2.2.5 系统时基

根据以上分析，可将系统时基各时序信号的波形示于图2.2.6中。

各时序信号的作用简介如下；

7M　　是位于主电路板B1的四D触发器 74S175 的输出 Q0，用做字符显示器5×7点的光点信号；

COLOR REF　是位于主电路板B1的四D触发器74S175的输出$\overline{Q1}$，频率为3.5MHz，作为彩色电视副载波信号；

ϕ0、ϕ1　　是位于主电路板B1的四D触发器74S175的输出 Q3 和 $\overline{Q3}$，频 率 为1MHz，作为系统的时钟信号ϕ0＝$\overline{\phi1}$；

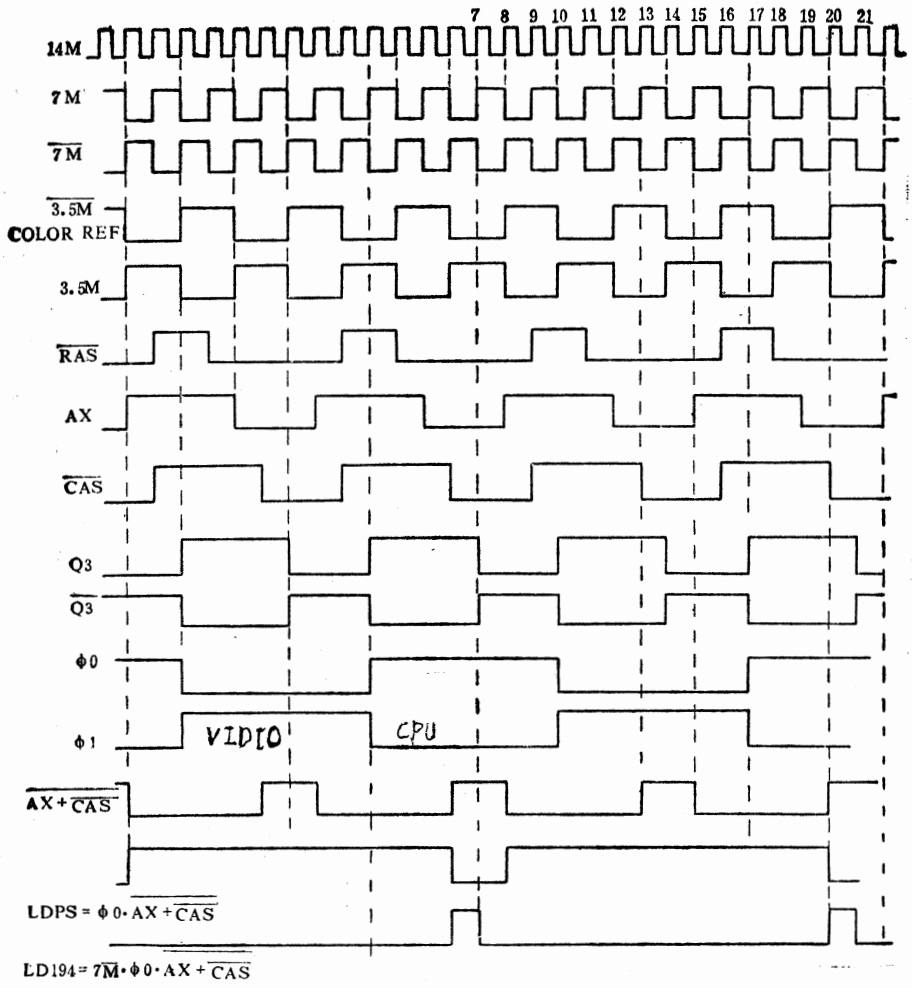


图 2.2.6 时基系统各时序信号的波形图

$\overline{RAS}$ 是位于主电路板C2的四位移位寄存器74S195的输出Q0，频率为2MHz，作动态RAM行地址选通信号；

AX 是位于主电路板C2的四位移位寄存器74S195的输出Q1，频率为 2MHz，作动态RAM地址选择开关的时间选通信号；

$\overline{CAS}$ 是位于主电路板 C2的四位移位寄存器74S195的输出Q2，频率为2MHz，作动态RAM列地址选通信号；

Q3 是位于主电路板C2的四位移位寄存器74S195的输出Q3，频率为 2MHz，是非对称信号，连到外设插座上，作备用信号；

LD194 是位于主电路板B12的与门第8脚的输出，频率为1MHz，作为视频信号发生器中，四位双向移位寄存器74LS194的时钟脉冲；

$\overline{LDPS}$ 是位于主电路板A2的与非门第3脚的输出，频率为1MHz，作同步计数器的加载时钟脉冲。

2.3 Apple地址空间的分配和译码

Apple的MPU具有十六位地址线，直接寻址可达65536个单元。这些单元被分成256页，每页256个单元，因此十六位地址的高八位可以看成是页地址（页号），低八位表示在某页中的位置。

Apple的寻址空间分成三部分：随机存取存储器（RAM）、只读存储器（ROM）和输入/输出设备码。Apple Ⅱ plus寻址空间基本分配如图2.3.1。

RAM存储区从0页到191页，占据的地址为0000～BFFF，共48K字节。这些存储器被系统程序和显示占去一部分，其余可由用户任意选用。

主板上有12K字节的只读存储器（ROM），分配给ROM的地址为D000～FFFF，其中存有磁盘自引导程序、监控程序和Apple soft（浮点BASIC）解释程序。这个地址空间也可被用于扩充的RAM，在0号外设插座插上扩充RAM卡（语言卡）时，系统可扩充12K字节的RAM，占用的地址为D000～FFFF。

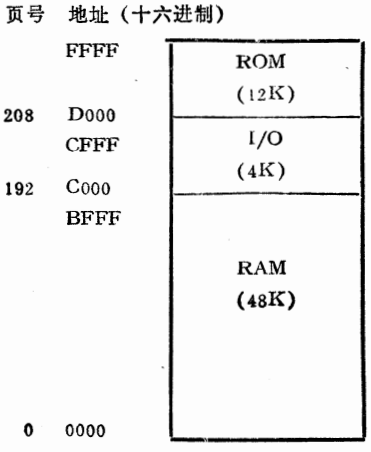


图 2.3.1 地址分配

Apple保留了4096个地址专用于输入/输出，这个区域的地址为C000～CFFF。

随着集成电路的发展，半导体存储器的集成度大大提高，体积急剧缩小，市场上已有64K×1位的存储器，但存储器的容量毕竟是有限的，因此Apple的RAM和ROM都是由多片集成电路组成。在"读/写"时，首先需要通过译码器选中不同的集成电路。在集成电路中，设有片允许控制端，它与电路内部的译码器或"读/写"控制器相联系，只有当片允许控制端出现规定的电位时（可称该片存储器被选中），才能根据所给的地址，对这一电路中的存储单元进行读或写。Apple的地址译码电路如图2.3.2。下面逐一说明译码电路产生的信号和作用。

2.3.1 ROM的片允许控制信号

早期生产的Apple机，其ROM均采用2K×8位（如2716）的集成电路。随着集成电路集成度的提高，价格的降低，目前Apple Ⅱ plus中均采用4K×8位的只读存储器（如2732），仅用三片就占满12K字节。在原印制板的基础上，组成ROM存储体的结构如图2.3.3。图中各片ROM相应位地址线分别连在一起，并接到Apple的系统地址总线上。各片相应位的数据线也连在一起，从ROM读出的八位数据送到系统数据总线上。

2732在单一的+5V电源下工作，它有二个控制端，只有这两者在逻辑上都得到满足，才能在输出端获得数据。片允许控制（$\overline{CE}$）是功率控制，用于器件选择。$\overline{CE}$为低电

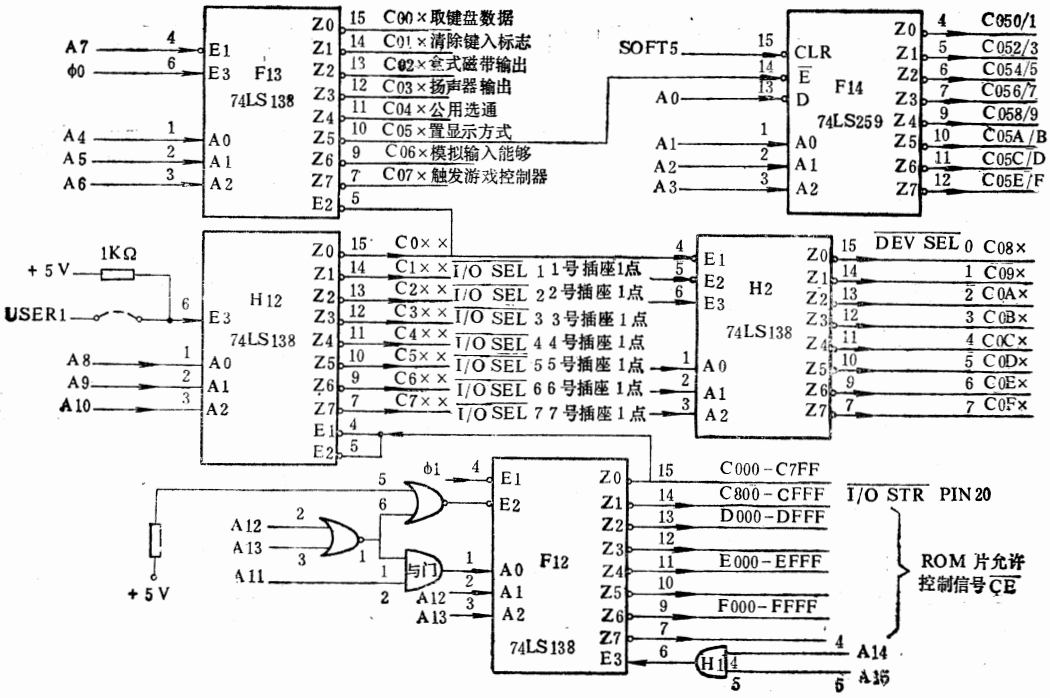


图 2.3.2 地址译码电路

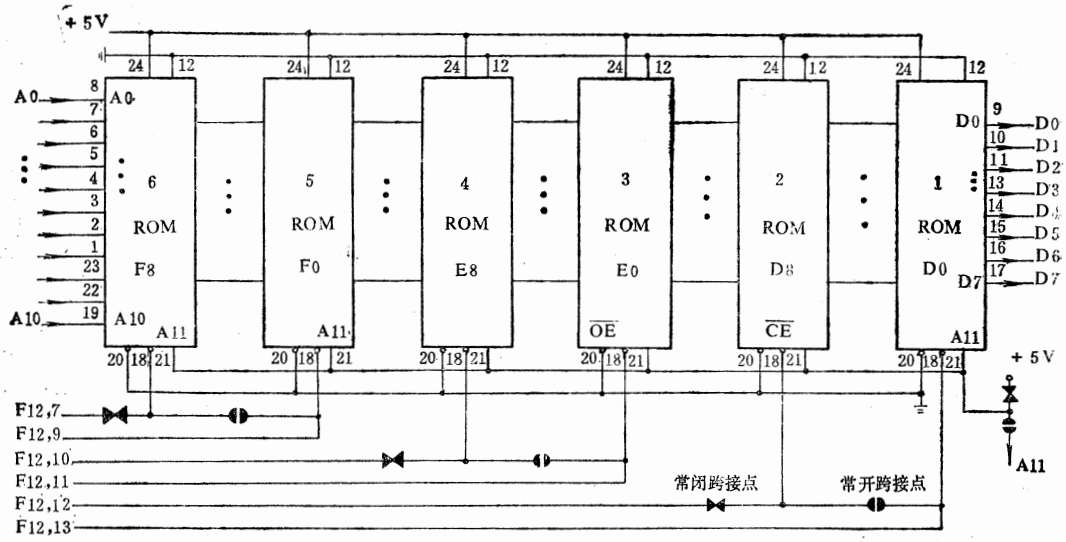


图 2.3.3 ROM存储体结构图

位时，器件被选中；$\overline{CE}$为高电位时，耗散功率可降低75%。输出允许（$\overline{OE}$）是输出控制，用于选通数据到输出端，它与器件选择无关，$\overline{OE}$为低电位时，允许输出数据。

在图2.3.3中，各片的输出允许$\overline{OE}$都连到一起，并接到地，$\overline{OE}$显然总是有效的。片允许控制端$\overline{CE}$和A11的连接考虑到ROM可能是2K×8位，或是4K×8位。当采用2K×8位的ROM时，六片ROM的21脚（A11）均接+5V。各片的$\overline{CE}$依次接到译码器74LS138（在主板上位于F12）的输出端上。而采用4K×8位ROM时，三个集成片分别插在图2.3.3中的1、3、5的位置上。这时仅需将与21脚连接的常闭跨接点断开，把常开跨接点接通。

74LS138(F12）是一片译码器，它的输出控制着ROM的选中与否。下面我们首先讨论F12能够译码的条件。从图2.3.2中可以看出，F12三个使能端的正逻辑表达式如下：

使能端E1＝$\phi 1$

使能端E2＝$\overline{\overline{A13}+\overline{A12}+\overline{INH}}$

使能端E3＝A15·A14

从器件索引中可知，当该片的使能端E1和E2同时为低电位且E3为高电位时，才可能有译码输出，即八个输出端之一为低电位；否则，译码输出端全部为高电位。也就是说，要使F12能够译码，则必须满足：

$\phi 1=0$，即时钟$\phi 1$为低的半周期。显然，也只有在$\phi 1=0$的半周内可以从ROM中读出数据，这与MPU的读写时序是相符的。

A15＝A14＝1，即地址总线的高两位为1。

先介绍一下$\overline{INH}$信号。$\overline{INH}$称作主板ROM选通译码的禁止信号，来自外设插座的32点。这一点通过1kΩ电阻接+5V，一般情况下为高电位。

下面讨论使能端E2在什么情况下为低电位。从逻辑表达式中可以看出，$\overline{INH}$和$\overline{A13+A12}$是或非关系，因此，它们中只要有一个是高电位，使能端E2就为低。一般情况下，由于$\overline{INH}$为高电位，所以无论A13和A12为何，F12都是可以译码的。

外设插座上的外设（或语言卡）也可将$\overline{INH}$置成低电位，而这时F12能否译码还要看总线地址A13和A12的值。A13和A12中有一个为1，则$\overline{A13+A12}=0$，F12的使能端E2就变高，F12不能有译码输出。这种情况相应的地址码为D000～FFFF。而当A13＝A12＝0时，$\overline{A13+A12}=1$，F12的使能端E2为低，F12能够译码。此时相应的地址码为C000～CFFF。当主机内插有16K RAM扩充卡（语言卡）时，由于扩充RAM占用的地址和主板上ROM的地址相同，也是D000～FFFF，在某个时间里，为了读写语言卡上的RAM，而不是主板上的ROM，就要使主板上的ROM不能被选中，即让$\overline{INH}$变低。若地址码在D000～FFFF范围内，F12的使能端变得无效，在适当的指令之后，就可以读写语言卡上的RAM了。与此同时，若需选通“输入/输出”设备，给出的地址码在C000～CFFF范围内，F12依旧能够译码。

综上所述，在$\phi 1=0$的半周期内，若在地址码为C000～CFFF或$\overline{INH}=1$，且地址码为D000～FFFF的条件下，F12有译码输出。

在译码器三个使能端有效的情况下，译码输出端中哪个是低电位，取决于译码输入端A2、A1和A0。F12的译码输入A2和A1分别接地址总线A13和A12，译码输入A0的逻辑表达式为：

$\overline{A13+A12}\cdot A11$

为直观起见，我们在表2.3.1中列出译码输入A0与三位总线地址的关系。从表中可以看出，只有在地址总线A12＝A13＝0时，译码输入A0才反映地址线A11的值。从前面的讨论可知F12在A14＝A15＝1时让使能端E3有效，所以当高四位地址码为C时，即在地址为C000～CFFF范围内，可区别是C000～C7FF还是C800～CFFF。而在D000～FFFF范围内，译码输入A0均为0，因此译码输出只能区别高四位地址是D、E还是F。

表 2.3.1 F12译码输入A0与地址总线的关系

地址总线			译码输入
A13	A12	A11	A0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

F12的使能输入和译码输入都没有涉及到地址的低十一位，即地址的低十一位可以为任意值。表2.3.2列出F12的译码情况。

从表中可以看出，F12的译码输出主要是供ROM作为片选信号的。按照地址总线上给出的地址码，选中某一ROM电路，以便读出其内容。

表 2.3.2 F12的译码情况

地址码	$\phi1$	$\overline{INH}$	选中	功能
C000～C7FF	0	0 1	Z0	外设I/O选择设备码
C800～CFFF	0	0 1	Z1	外设扩展ROM使能
D000～DFFF	0	1	Z2	ROM片选信号
E000～EFFF	0	1	Z4	ROM片选信号
F000～FFFF	0	1	Z6	ROM片选信号

C800～CFFF这2K地址是为外设插件板上的扩展ROM保留的。这个地址空间是所有外设插座都可以用的。注意：当不止一个外设插件板插入同一台Apple时，在某个时间里，只允许一个扩展ROM工作。为了使指定的扩展ROM能够工作，外设插件板上应设置一个触发器，其电路如图2.3.4所示。

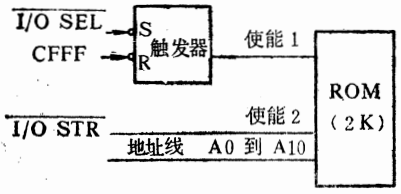


图 2.3.4 扩展ROM使能电路

CFFF是一个专用地址，所有具有扩充ROM的外设接口都应设置相应的译码电路，以识别这个地址。当主机或外设初始化时，用这个地址，将触发器置0，禁读各外设接口的扩充ROM。

当需要读某个外设接口的扩充ROM时，首先给出该外设所在插座的“输入/输出”设

备码，即可在该插座得到$\overline{\text{I/O SEL}}$有效信号，将触发器置1，然后再给出C800～CFFF范围内的地址码，$\overline{\text{I/O STR}}$有效信号使扩充ROM变得能够读出，就可以读出该扩充ROM了。

2.3.2 “输入/输出”选择信号（$\overline{\text{I/O SEL}}$）

图2.3.2中，H12也是一个三线/八线译码器，从上节知道，H12的使能端E1和E2的信号来自F12的输出Z0，地址码为C000～C7FF时，Z0有效。使能端E3通过1kΩ电阻接+5V。一般情况下，E3是有效的。而Apple的设计者在外设插座上设置了USER1，称作“用户1”（在插座39点），通过跨接线可将USER1和H12的使能端E3连接起来。这时，在任一外设插座的39点上出现低电位，H12将不能译码，即不能给出主板上的所有“输入/输出”设备码。

三条译码输入线分别接地址总线A10、A9和A8，H12的译码情况如表2.3.3。

表 2.3.3 H12的译码结果

使能端输入		译码输入			译码输出		功能
E1 (E2)	E3	A2	A1	A0	设备码	输出端	
C0000～C7FF	1	0	0	0	C0××	Z0	
	1	0	0	1	C1××	Z1	外设插座1上的$\overline{\text{I/O SEL}}$信号
	1	0	1	0	C2××	Z2	外设插座2上的$\overline{\text{I/O SEL}}$信号
	1	0	1	1	C3××	Z3	外设插座3上的$\overline{\text{I/O SEL}}$信号
	1	1	0	0	C4××	Z4	外设插座4上的$\overline{\text{I/O SEL}}$信号
	1	1	0	1	C5××	Z5	外设插座5上的$\overline{\text{I/O SEL}}$信号
	1	1	1	0	C6××	Z6	外设插座6上的$\overline{\text{I/O SEL}}$信号
	1	1	1	1	C7××	Z7	外设插座7上的$\overline{\text{I/O SEL}}$信号

如表2.3.3中所示，H12的译码输出主要是供给1到7号外设插座，译码有效时译码器输出的负脉冲作为输入/输出选择信号（$\overline{\text{I/O SEL}}$）。每个外设插座有256个地址可供使用。地址码为Cs00～CsFF，其中s为插座号。通常可在外设插件板上设置256×8位的ROM，其中存放外设的管理程序，而$\overline{\text{I/O SEL}}$信号送到ROM的使能端上。当微处理机涉及到Cs00～CsFF的地址码时，H12的相应译码输出端出现负脉冲，即相应插座的$\overline{\text{I/O SEL}}$点出现负脉冲，这时就可读出或执行ROM中的管理程序了。例如并行打印接口板插在1号外设插座上，当完成命令C100G后，就为并行输出做好了准备。实际上在执行C100G命令的过程中，1号插座$\overline{\text{I/O SEL}}$信号有效，读出并执行了接口板上从C100开始的程序。

一般的接口板可以插入除0号以外的任一个外设插座。设所插的插座号为s，启动命令的地址为Cs00（s=1…7）。

从图2.3.2中看出，H12的输出Z0并没有接到0号外设插座上，因此0号插座不象其它插座那样具有256个地址空间，也不能使用$\overline{\text{I/O SEL}}$信号。H12的输出端Z0的作用将在下面讨论。

2.3.3 外设接口板的设备选择信号($\overline{\text{DEV SEL}}$)

在图2.3.2中，H2也是一片译码器，型号为74LS138。使能端E1来自译码器H12的输出端Z0。当地址码为C000～C0FF时，H12的Z0输出负脉冲。使能端E2接时钟脉冲ϕ1。使能端E3接地址总线A7。即要求ϕ1=0，A7=1。

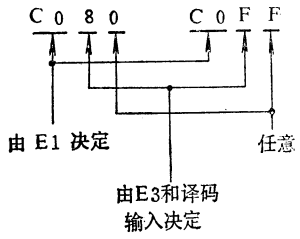


图 2.3.5 H2译码输出与输入的关系

H2的三个译码输入分别接地址总线A6、A5和A4，该片的译码不涉及地址总线A0～A3。综上所述，H2的译码输出的地址范围是C080～C0FF，如图2.3.5。译码结果如表2.3.4。

H2的译码输出被作为外设的设备选择信号$\overline{\text{DEV SEL}}$接到相应外设插座的41点上。当地址码为C080+s×时（s为插座号），经H2译码，在s号插座的41点产生负脉冲，用此

表 2.3.4 H2的译码的结果

使能端输入			译码输入			译码输出		去向		功能
E1	E2=ϕ1	E3=A7	A2	A1	A0	设备码	输出端	外设插座号	点号	
C0××	0	1	0	0	0	C08×	Z0	0	41	外设插座上的$\overline{\text{DEV SEL}}$信号
C0××	0	1	0	0	1	C09×	Z1	1	41	
C0××	0	1	0	1	0	C0A×	Z2	2	41	
C0××	0	1	0	1	1	C0B×	Z3	3	41	
C0××	0	1	1	0	0	C0C×	Z4	4	41	
C0××	0	1	1	0	1	C0D×	Z5	5	41	
C0××	0	1	1	1	0	C0E×	Z6	6	41	
C0××	0	1	1	1	1	C0F×	Z7	7	41	

脉冲可选择外设装置。上面地址码中的×表示任意值，即低四位地址为0～F中任一个都可以。当外设插件上的装置不止一个时，配合地址低四位的译码，最多可选择16个装置中的一个。例如A/D转换板上有16个通道，当A/D转换板插在5号插座时，用下面的一段程序，就可启动并读出16通道A/D转换结果。

```
        LDX  #00        变址寄存器置初值
        LDA  C0D0       首次启动0号通道A/D转换，设备码C080+50=C0D0
        NOP             延时
        NOP
LOOP:   LDA  C0D0, X    读出A/D转换结果
        STA  1000, X    存A/D转换结果
```

```
LDA C0D1,X    启动下一个通道A/D
INX           变址寄存器加1
CPX #10       16通道取完了吗?
BNE LOOP      未取完，继续取
RTS           返回
```

2.3.4 主板上外设设备码的译码

在6502的指令系统中，没有专设输入、输出指令，因此Apple Ⅱ的MPU把外部设备作为存储器的单元来对待。故每个外设占有存储器的一个地址。从外设输入一个数据，作为存储器的一次读操作；向外设输出一个数据，则作为存储器的一次写操作。这样，MPU对外设的操作可使用全部的存储器指令，使用方便。在讨论内存地址分配和译码时，也必须涉及到外设地址（设备码）的分配和译码。

主板上外设设备码的译码是由译码器F13（74LS138）完成的。F13的使能端E2来自译码器H12的输出端Z0，当地址码为C000～C0FF时，F13的E2得到负脉冲。使能端E3接时钟脉冲$\phi0$，使能端E1接地址总线A7，即要求$\phi0=1$，A7＝0。

F13的三个译码输入端分别接地址总线A6、A5和A4，该片的译码不涉及地址总线A3～A0。综上所述，H2有译码输出的地址范围是C000～C07F。译码结果如表2.3.5。

表 2.3.5 F13的译码结果

使能端输入			译码输入			译码输出		功能
E3＝$\phi0$	E2	E1＝A7	A2	A1	A0	设备码	输出端	
1	C0××	0	0	0	0	C00×	Z0	取键盘数据
1	C0××	0	0	0	1	C01×	Z1	清除键入标志
1	C0××	0	0	1	0	C02×	Z2	盒式磁带输出
1	C0××	0	0	1	1	C03×	Z3	扬声器输出
1	C0××	0	1	0	0	C04×	Z4	公用选通
1	C0××	0	1	0	1	C05×	Z5	置显示方式
1	C0××	0	1	1	0	C06×	Z6	模拟输入能够
1	C0××	0	1	1	1	C07×	Z7	触发游戏控制器

对于表中每个设备码的作用，我们将结合接口的工作原理进行讲解。

2.3.5 屏幕开关

主板上的F14（74LS259）是一片锁存器，每片中包括可寻址的八个锁存器。259有一个使能端$\overline{E}$（14脚），当$\overline{E}$为低电位时，该片能够锁存数据，而输入数据锁存在哪个锁存器中，由锁存地址选择决定。该片有三条锁存地址选择线，经过内部译码电路可选中八个锁存器中的一个。F14的使能端接F13的Z5（10脚），当地址码为C050～C05F时，F13的Z5输出负脉冲，即此时F14可锁存数据。F14的锁存地址选择分别接地址总线的A3、A2和A1。总线地址A0作为锁存数据，这样，高十二位地址为C05，再配上低四位

地址表示的十六个地址码，就形成了十六个专用的设备码。

F14的输出作为屏幕和四个一位输出的软开关。我们先列出译码结果如表2.3.6，然后逐一说明之。

表 2.3.6 F14的输出情况

CLR	E	A3	A2	A1	A0	设备码	功　　能
1	C05X	0	0	0	0	C050	置图形方式
					1	C051	置文本方式　TEXT　MODE
				1	0	C052	置全部文本或全部图形
					1	C053	置文本和图形混合方式　MIX　MODE
			1	0	0	C054	置显示第一页
					1	C055	置显示第二页　PAGE2
				1	0	C056	置低分辨率图形方式
					1	C057	置高分辨率图形方式　HIRES
		1	0	0	0	C058	一位输出0，低电位
					1	C059	一位输出0，高电位
				1	0	C05A	一位输出1，低电位
					1	C05B	一位输出1，高电位
			1	0	0	C05C	一位输出2，低电位
					1	C05D	一位输出2，高电位
				1	0	C05E	一位输出3，低电位
					1	C05F	一位输出3，高电位

Apple的监视器显示有三种方式：

1．文本方式：每帧显示24行字符，每行40个字符。

2．低分辨率图形：显示彩色方块，每帧48行，每行40块。计有十六种颜色。各彩色块之间没有空隙。

3．高分辨率图形：显示彩色点阵。每行280个点，每列192个点，有六种颜色。

监视器显示的信息均存在主板的RAM中，对于每种显示方式都有两个存储区与之相对应，称为第一页（初始页）和第二页。

在显示图形时，还可以在屏幕下部留下四行，仍显示文本内容，称为混合方式。归结起来，共有十种屏幕方式，它是由F14给出的软开关组合而成，表2.3.7列出各屏幕软开关的组合和功能。

表2.3.7中，每一种屏幕方式下，表中所列软开关必须全部设置。例如混合方式下的文

表 2.3.7 屏幕开关的组合和功能

软件开关 / 页别 / 屏幕方式	第一页	第二页
全部是文本	C054 C051	C055 C051
全部是低分辨率图形	C054 C056 C052 C050	C055 C056 C052 C050
全部是高分辨率图形	C054 C057 C052 C050	C055 C057 C052 C050
文本和低分辨率图形混合	C054 C056 C053 C050	C055 C056 C053 C050
文本和高分辨率图形混合	C054 C057 C053 C050	C055 C057 C053 C050

本和低分辨率图形第一页，需要设置四个软开关，采用如下程序即可：

```
LDA C050    置图形方式
LDA C053    置文本和图形混合方式
LDA C056    置低分辨率图形方式
LDA C054    置显示第一页
```

四个一位输出，有的书上也称之为信号灯。它们都能够作为机外一些电子装置的输入。实际上，每个一位输出就是经过锁存的一对设备码的最低位。四个一位输出使用的地址码为C058～C05F，当涉及到偶数地址时，输出的逻辑电平为“0”，电位接近0V；当涉及到奇数地址时，输出的逻辑电平为“1”，电位接近5V。

四个一位输出连到游戏联接器J14的12、13、14和15脚上，以供外电路使用。

2.4 RAM的选择电路

2.4.1 主板上的RAM

Apple的主板上共有24片RAM，分设在C行、D行和E行上。结构如图2.4.1。

图2.4.1中，每片RAM的容量是16K×1位，即同一片上的16K个存储单元都在同一数据位上。如位于C3、D3和E3的RAM的输入都接到数据总线D0上，读出的数据通过锁存器又送到数据总线D0上。类似，其它同一列上的三片RAM，其数据输入、输出也都在同一数据位上，那么同一行上的八片RAM中地址相同的存储单元就构成了一个计算机字节。八片RAM构成了16K个八位字节，三行共48K字节，这就是我们常说的Apple主板上有48K RAM的结构。

16K×1位RAM的型号和规格很多，我们以4116为例，在图2.4.2中列出其外引线。每片RAM有16条引线，其中电源输入端三个，一个接地端。行地址选通$\overline{RAS}$、列地址选通$\overline{CAS}$各一个。地址码输入端七个。我们知道一片4116中有16384个存储单元。寻址16K，需要14条地址线，但4116只有16条引线，所以地址被分成两部分：行地址和列地址。4116内部的存储单元被组织成矩阵的形式，如128×128，这样由七位行地址和七位列地址的重叠就可选中所需的存储单元。在集成电路内部设地址锁存器，用选通的办

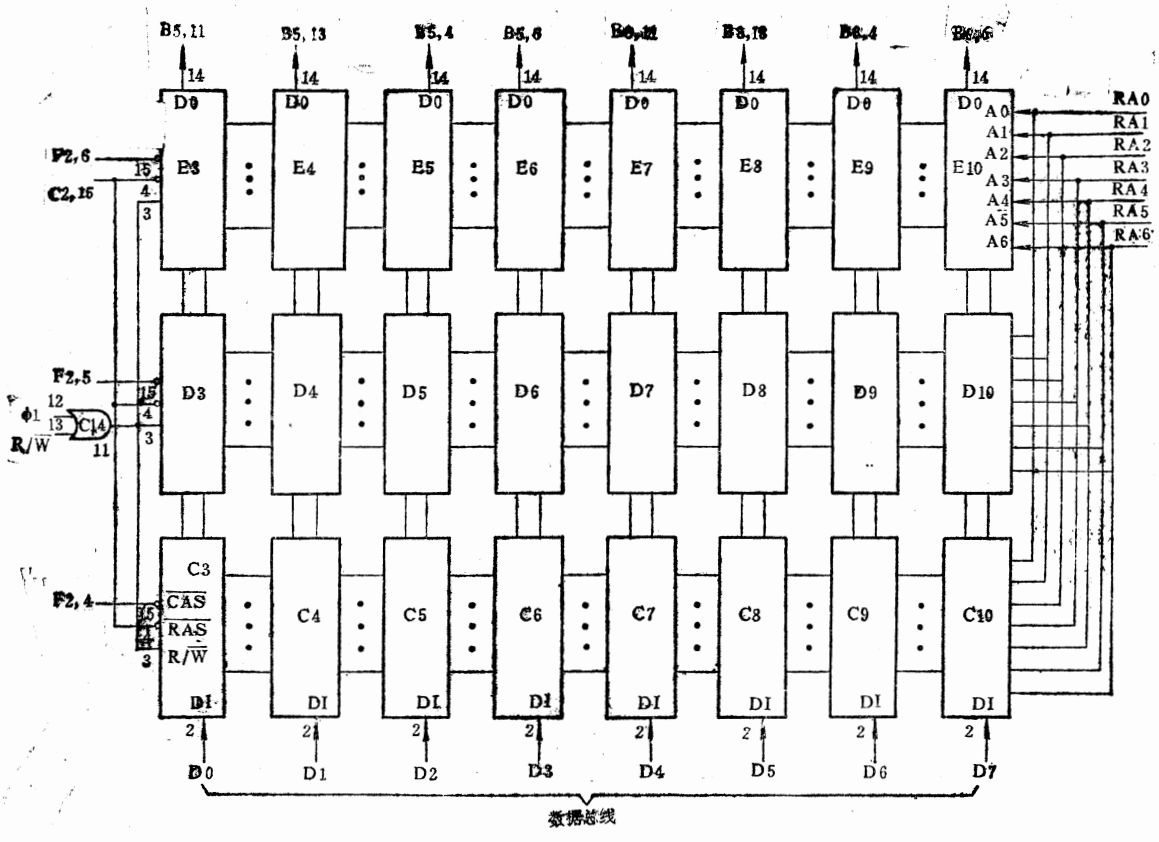


图 2.4.1 主板上的RAM

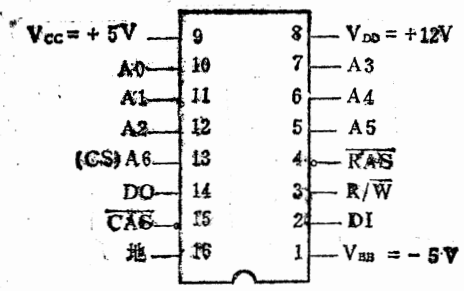


图 2.4.2 4116引脚图

法，利用行地址选通信号$\overline{RAS}$把首先出现的7位地址送行地址锁存器，由随后出现的列地址选通信号$\overline{CAS}$把后出现的7位地址送列地址锁存器，这样就可读出所需的存储单元。4116的内部结构示意图如图2.4.3。

当某一行被选中时，这一行的128个存储单元都被选通到读出放大器去，在那里每一个存储单元的逻辑电平都被鉴别、锁存和重写。但在列地址译码后，只选中这128个放大器中的一个，把它送至输出锁存和缓冲器中去。

一条数据输入线和一条数据输出线。一般情况下，数据输出线为高阻状态，只是在读周期内，在存取数据直到$\overline{CAS}$回到高电位之前，才是逻辑“1”或“0”。

4116有一个控制信号端R/$\overline{W}$。写入RAM时，R/$\overline{W}$必须是低电位。读RAM时，R/$\overline{W}$=1。

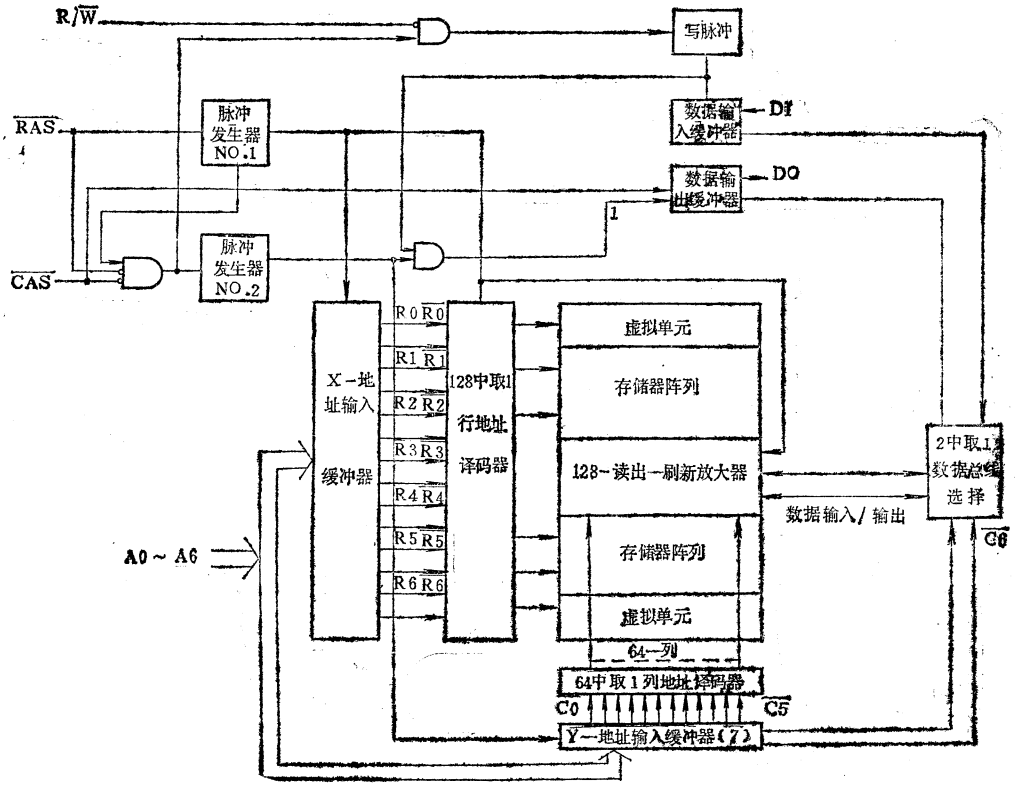


图 2.4.3 4116内部结构图

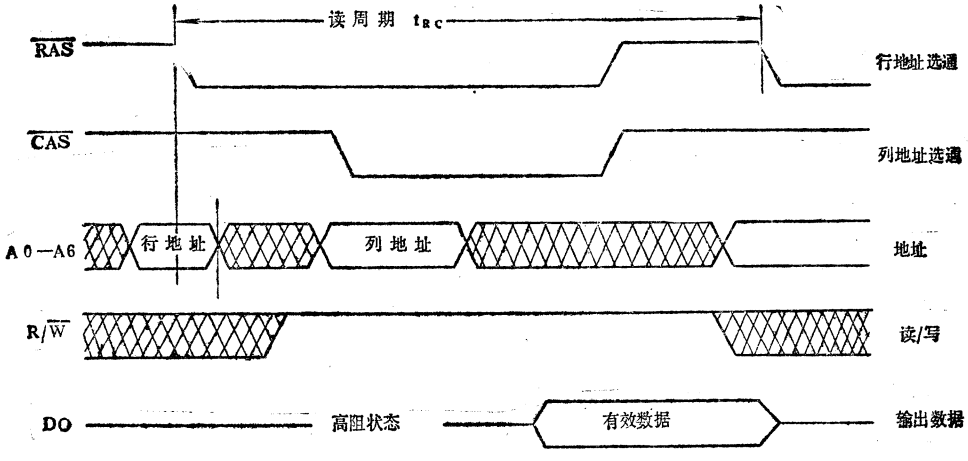


图 2.4.4 4116的读周期时序

读周期时序如图2.4.4。读周期从$\overline{RAS}$脉冲变低开始。但地址必须在$\overline{RAS}$变低前有效。同样列地址也必须在$\overline{CAS}$有效以前有效。由于4116内部有地址锁存器，故在所要求

的列地址保持时间以后，在读写周期完成以前，外界的输入地址可以改变。而$\overline{RAS}$和$\overline{CAS}$有效要保持到数据有效之后。

Apple中常采用的RAM为4116P-3，它的读周期最小为375ns。

写周期时序中，$\overline{RAS}$信号、$\overline{CAS}$信号以及它们与地址信号之间的关系，与读周期相同。注意：这时R/$\overline{W}$必须为低。为保证外部数据能可靠地写入，要求数据在选通信号（$\overline{CAS}$信号与R/$\overline{W}$信号的较晚出现者）有效前已经稳定，并要保持一定的时间。

4116是动态随机存取存储器，其特点是集成度高、功耗低、价格便宜，但要求刷新。典型的刷新周期为2ms。在正常的读写周期中，由七位行地址所选中的一行可被刷新。

计算机中的RAM，通常都是由MPU通过总线给出地址进行读写的。在Apple中，监视器显示的内容也是存在系统RAM中的。为叙述方便，我们用显示存储器这个术语表示存储显示内容的RAM，而用MPU存储器这个术语表示MPU读写的RAM。实际上，MPU可以读写整个存储区，包括显示存储器。宏观上看，MPU和显示是同时工作的，那么它们是如何共享系统RAM的呢？图2.4.5表示了RAM地址的形成和时序。从图中看出，MPU存储器和显示存储器是分时被读写的。

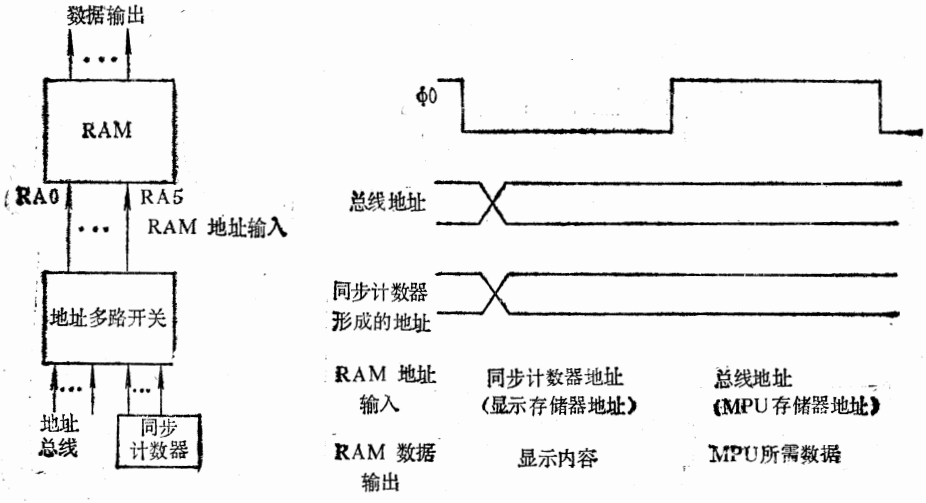


图 2.4.5 RAM地址的形成和时序

从2.1节中可知，MPU是在φ0＝0时在地址总线上建立地址；而在φ0＝1时，在数据总线上得到有效地址。那么，对于RAM来说，只要RAM的读写周期小于500ns，我们就可以在φ0＝1期间，将MPU送来的总线地址送给RAM，读出有效数据送MPU或将MPU的数据写入RAM；而读显示存储器可在φ0＝0期间完成。Apple就是这样做的。

RAM的地址输入端接有多路开关，开关交替地给出MPU存储器地址和显示存储器地址。在φ0＝0期间，地址多路开关没有把总线上已经建立的地址送RAM，而是将同步计数器形成的地址送RAM，读出（禁写）显示存储器内容，供显示用。在φ0＝1期间，再将总线上的地址送RAM，对RAM进行读写。可见，在一个时钟周期内，读写了两次RAM。

我们知道，读写一次RAM，需要送去两组地址：行地址和列地址。因此，在一个时钟周期内，需要给每片RAM的地址输入端送去四组地址。为了使这些地址分时有效，系统时基设置了行选通信号$\overline{RAS}$、列选通信号$\overline{CAS}$，还有AX信号等。图2.4.6列举了一些选通信号，并标明了RAM地址和选通信号有效的时序。

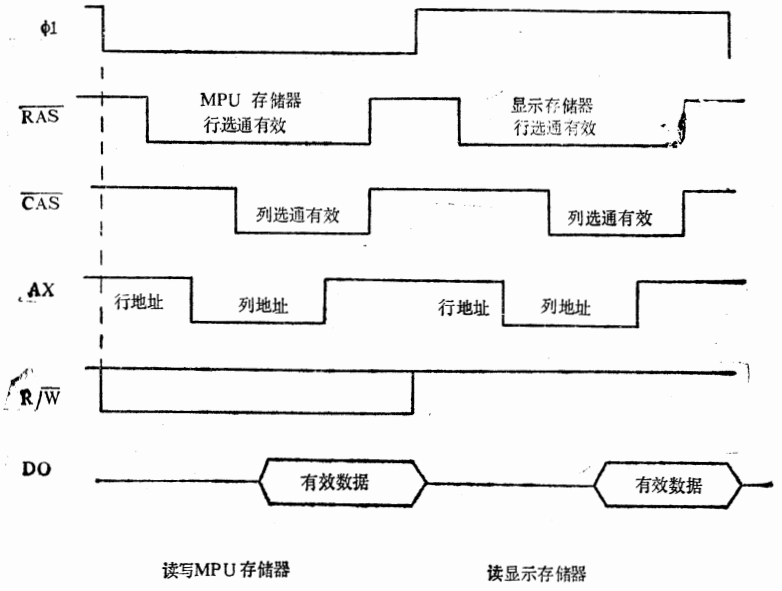


图 2.4.6 RAM地址和选通信号有效时序

图2.4.7是RAM选择电路。图中J1是二选一开关，选择控制信号为φ0。F2是二线/四线译码器。它们共同产生RAM的列选通信号。

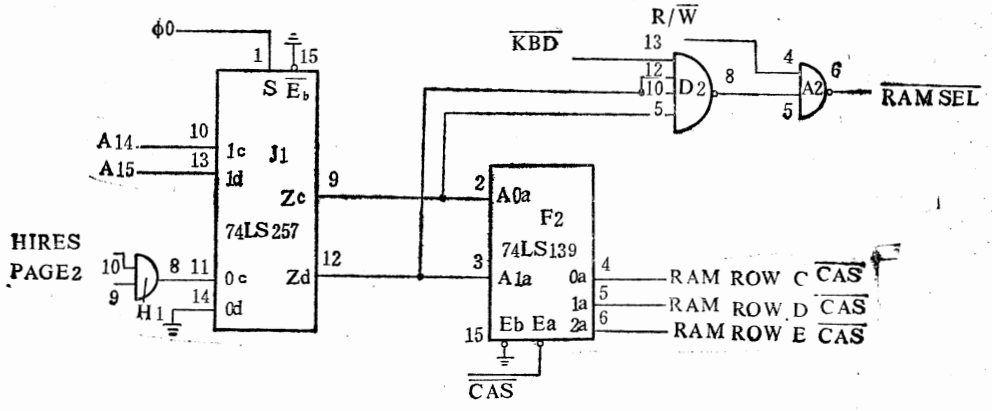


图 2.4.7 RAM的列选通信号电路

2.4.2 RAM的行地址选通

RAM的行地址选通没有专门设置电路，只是将时基电路产生的$\overline{RAS}$信号直接连到

RAM的行地址选通端。它的作用是将首先出现在RAM地址输入端的地址置入RAM的行地址输入缓冲器。

2.4.3 RAM的列地址选通

每一行上的八片RAM构成16K字节，为了寻找一个存储单元，需要14位地址。

Apple的主板上有48K字节RAM，分设在三行上，而为了寻址48K，需要16位地址。不同行上的RAM，低14位地址范围相同，只有高二位地址不同。因此可将高两位地址A15和A14译码，利用译码输出的有效信号来区别某行RAM是否正在读写。从前面4116的管脚介绍中知道，它没有设片选端，而设有列地址选通$\overline{CAS}$端，因此可将译码输出信号送到地址选通端。

同理，在读显示存储器时，送到RAM列地址选通端的信号被用以确定该种显示方式下的显示存储器范围。

图2.4.7中，J1（74LS257）是一片二选一开关。选择输入信号为$\phi 0$。当$\phi 0=0$时，J1的输出反映0组输入信号，这时J1送出的是显示存储器列选通信号。$\phi 0=1$时，J1的输出反映1组输入信号，这时送出MPU存储器列选通信号。图2.4.7中，F2（74LS139）是二线/四线译码器，使能端Eb接地，Ea接系统时基信号$\overline{CAS}$，因此F2只有在列选通信号$\overline{CAS}$为低时，才有译码输出。J1选择输出和F2译码输出分别如表2.4.1和表2.4.2。

表 2.4.1 J1选择输出

选择输入	选择输出		功能
S=$\phi 0$	Zc	Zd	
0	HIRES·PAGE2	0	显示存储器列选通信号
1	A14	A15	MPU存储器列选通信号

表 2.4.2 F2的译码结果

$\phi 0$	使能输入	译码输入		译码输出
	Ea	A1a=0	A0a=HIRES·PAGE2	
0	$\overline{CAS}$	0	0	0a=0 选中C行 RAM
		0	1	1a=0 选中D行 RAM
1	$\overline{CAS}$	A1a=A15	A0a=A14	
		0	0	0a=0 选中C行 RAM
		0	1	1a=0 选中D行 RAM
		1	0	2a=0 选中E行 RAM

1．MPU存储器列地址选通信号

从表2.4.2中可以看出，在$\phi 0=1$半周期内，若总线地址A15和A14都为零，当系统时基$\overline{CAS}$变为有效时，F2的0a端的输出变为负。现F2的0a端是连到主板上C行RAM的列地址选通端的，那么，RAM列地址选通的有效，可将出现在RAM地址输入端的地址置入RAM内的Y地址输入缓冲器。译码器F2对总线地址低十四位无要求，因此，当地址码在0000～3FFF之间时，C行RAM的列地址选通端均可出现负脉冲，其负脉冲出现的时刻与系统时基$\overline{CAS}$一致。从MPU的角度来看，可以讲C行RAM编址在0000～3FFF。

同理，F2的1a在A15＝0、A14＝1时，有负脉冲输出；F2的2a端在A15＝1，A14＝0时有负脉冲输出。它们分别连到D行和E行RAM的列地址选通端，因此D行、E行RAM的编址分别为4000～7FFF、8000～BFFF。

2．显示存储器列选通信号

在$\phi 0=0$期间，F2送出显示存储器的列地址选通信号。由于这时A1a总为零，因此显示存储器的列选通信号完全由A0a决定。A0a＝0选中C行RAM，A0a＝1选中D行RAM。

下面我们列表于表2.4.3中，以说明A0a与显示方式的关系。从表中看出，只有高分辨率图形的第二页时，才选中D行RAM作显示存储器，其它显示方式下，其显示存储器均选在C行RAM中。

表 2.4.3 显示存储器列选通信号

HIRES	PAGE2	A0a	显示方式	显示存储器
0	0	0	低分辨率图形第一页	C行RAM
0	1	0	低分辨率图形第二页	C行RAM
1	0	0	高分辨率图形第一页	C行RAM
1	1	1	高分辨率图形第二页	D行RAM

从上节讨论中知道，C行RAM的高两位地址A15和A14为00，D行RAM的高两位地址为01。即然高分辨率图形第二页的存储器在D行RAM中，那么高分辨图形第二页显示存储器的高两位地址也是01。同理，其它显示方式的显示存储器地址的高两位应为00。至于显示存储器的低十四位地址码，我们将在下节给出。

2.4.4 RAM的数据输出

从RAM读出的数据，锁存在寄存器中，寄存器的输出送到显示移位寄存器，数据可用于显示。寄存器的数据又可通过二选一多路开关送系统数据总线。电路如图2.4.8。

图2.4.8中，B5和B7（74LS174）都是四D触发器，从RAM读出的数据送到触发器的D端。在$\overline{RAS}$有效期间，RAM的行地址、列地址已送入RAM，数据已建立，因此可用$\overline{RAS}$的后沿将数据打入D触发器中。被寄存的数据除直接供给显示移位寄存器外，还被送到二选一多路开关B5和B7的输入端。多路开关用以把键盘输入数据和RAM数据

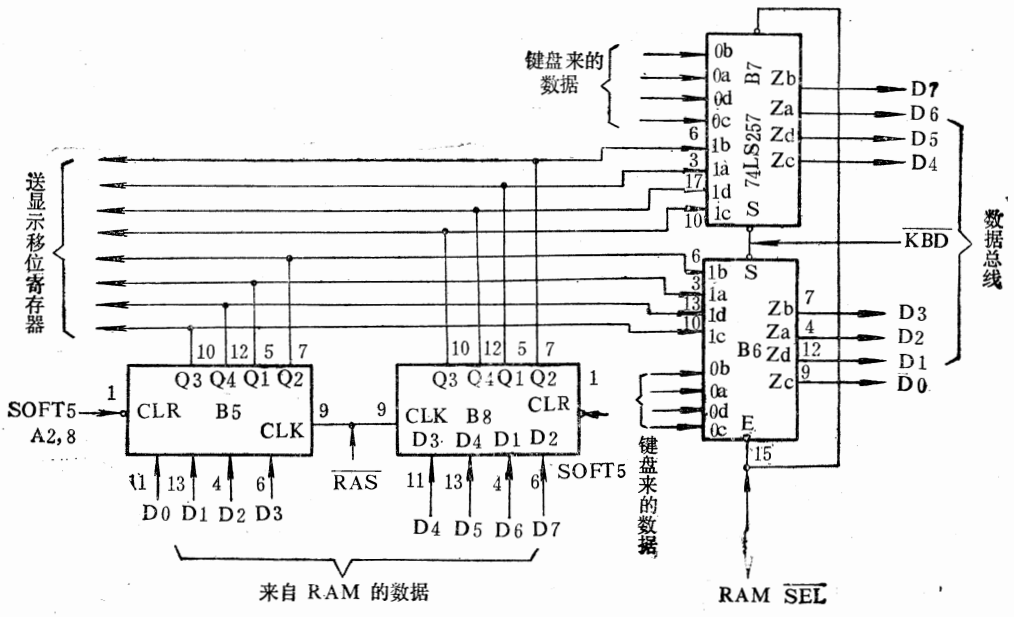


图 2.4.8 RAM的数据输出

送数据总线,其工作原理可参看2.10节。

2.5 RAM地址多路开关

由上节的分析可知，在ϕ0的一个周期内，需要给出四组地址码：MPU存储器行地址和列地址、显示存储器行地址和列地址。图2.5.1中RAM地址多路开关的作用就是分时将这四组地址送给RAM。

2.5.1 各器件输出与输入的关系

图2.5.1中，E14是一片四位全加器，每一位的输出都等于本位两个输入信号之和，再加上低一位的进位。其输出和输入的关系如表2.5.1。

E11、E12、E13和C1是四选一数据选择器。选择输入信号为AX和 ϕ0，AX 和 ϕ0 状态的组合，选择输出四组地址码中的一组，送RAM的七位地址输入端。输出信号与选择输入的关系如表2.5.2，其时序如图2.4.6。C1选择输入S1＝ϕ0、S0＝AX，选择器C1的输出功能与表2.5.2相同，只是输出端序号不同。

C12是二选一数据选择器，选择控制端为S。S为0时，其输出等于0组输入；S＝1时，其输出等于1组输入。C12的S＝HIRES，即表示C12在高分辨率图形方式（HIRES＝1）和低分辨率图形方式（HIRES＝0）下送出不同的地址。C12的输出与输入的关系如表2.5.3。

全加器E14和二选一数据选择器作为四选一数据选择器的输入，选择器就分时送出

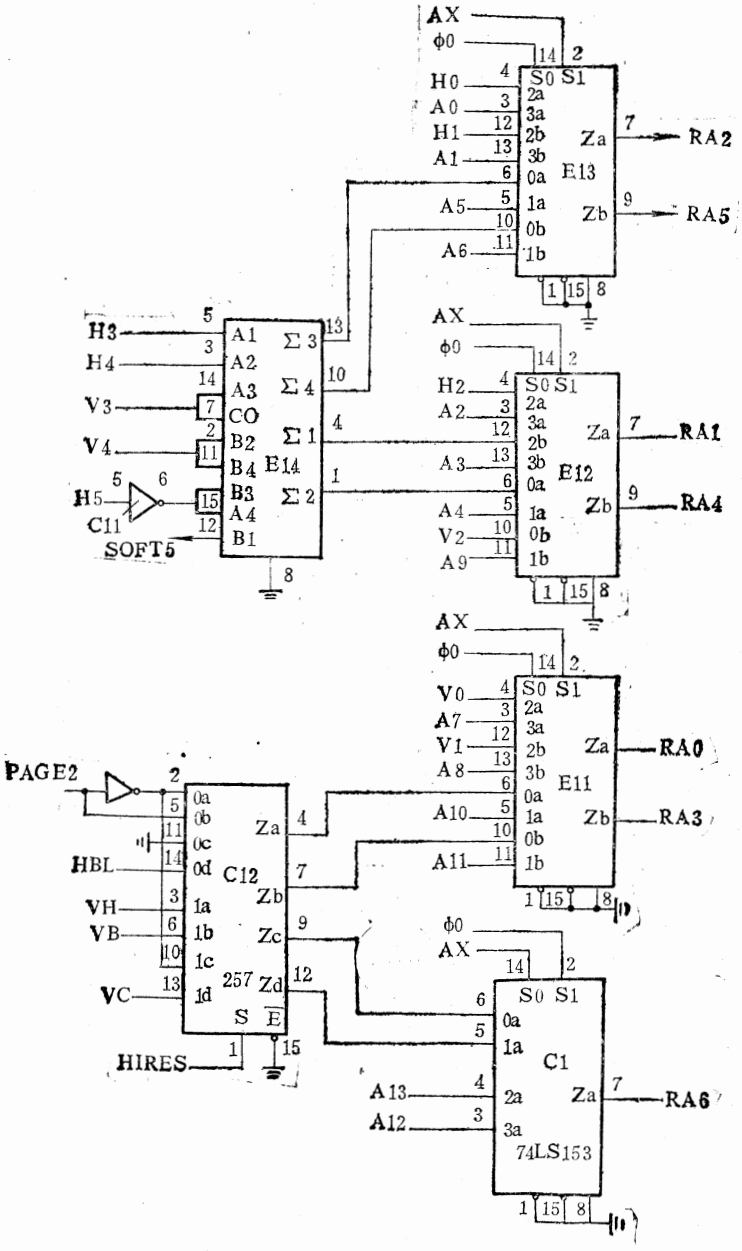


图 2.5.1 RAM地址多路开关

了四组地址。这就是MPU存储器和不同显示方式下显示存储器的低十四位地址。

2.5.2 MPU存储器地址

在φ0＝1时，四选一数据选择器选中两组地址，如表2.5.4。这两组地址就是 MPU 存储器低十四位地址。这十四位地址与上节讨论的高两位地址组成十六位 地 址。 这 样

表 2.5.1 全加器输出和输入的关系

序号	输入		低一位进位	输出
	A	B	C	Σ
1	H3	1	C0=V3	Σ1=H3+V3+1
2	H4	V4	C1=H3+V3+1	Σ2=H4+V4+C1
3	V3	$\overline{H5}$	C2=H4+V4	Σ3=V3+ $\overline{H5}$+C2
4	$\overline{H5}$	V4	C3=V3+ $\overline{H5}$	Σ4=$\overline{H5}$+V4+C3

表 2.5.2 数据选择器输出与选择输入的关系

选择输入		输出	
S1=AX	S0=ϕ0	Z	功能
0	0	Za=0a Zb=0b	显示存储器列地址
1	0	Za=2a Zb=2b	显示存储器行地址
0	1	Za=1a Zb=1b	MPU 存储器列地址
1	1	Za=3a Zb=3b	MPU 存储器行地址

表 2.5.3 C12的输出与输入的关系

选择输入	输出	
S		功能
0	Zi=0i	低分辨率图形存储器地址
1	Zi=1i	高分辨率图形存储器地址

MPU就可以寻址48K字节，读写主板上RAM的任何单元。

表 2.5.4 MPU存储器地址

RAM地址							功能
RA6	RA5	RA4	RA3	RA2	RA1	RA0	
A12	A1	A3	A8	A0	A2	A7	行地址
A13	A6	A9	A11	A5	A4	A10	列地址

2.5.3 显示存储器地址

图2.5.1中，四选一数据选择器在$\phi 0=0$时，再给出两组地址，供读显示存储器用。由于显示有多种方式，所需存储器的地址和范围均不同，因此，在RAM地址选择器（多路开关）的前面又设置了二选一地址选择器（C12）和全加器（E14）。这样就可以送出不同显示方式下的显示存储器地址。根据各器件的工作原理，列出地址多路开关送出的显示存储器地址，如表2.5.5。

表 2.5.5 显示存储器低十四位地址

RAM地址							功能
RA6	RA5	RA4	RA3	RA2	RA1	RA0	
HIRES·Vc +$\overline{\text{HIRES}}$·HBL	H1	H3+V3 +1	V1	H0	H2	V0	显示存储器行地址
HIRES· $\overline{\text{PAGE2}}$	$\overline{\text{H5}}$ + V4+C3	V2	$\overline{\text{HIRES}}$·PAGE2 +HIRES·VB	$\overline{\text{H5}}$+V3 +C2	V4+H4 +C1	HIRES.VA +$\overline{\text{HIRES·PAGE2}}$	显示存储器列地址

表中Hi（i=0～5）表示同步计数器中水平计数器相应位的输出。Vj（j=0～3）表示同步计数器中垂直计数器相应位的输出。我们将在下面讨论这些地址的具体数值。

2.5.4 显示存储器地址计算

在进行显示存储器地址计算之前，先简单介绍同步计数器输出与显示的关系。详细内容可参看2.8节。VC、VB和VA是字符点阵行计数器。每个字符点阵为5×8，点阵中自上而下的八行对应着VC、VB和VA的000～111。水平计数器H5、H4、H3、H2、H1、H0是一行中字符个数计数器，当它们计数到011000时，一行的字符显示开始，直到111111，共40个字符，水平计数器的其它时间为行逆程和显示两边的消隐部分。垂直计数数V4、V3、V2、V1、V0是显示行数计数器。垂直计数器在00000～10111范围内，屏幕显示，共24行。计数器的其它时间为场消隐时间。以上结果详见表2.5.6。

在前面表2.5.2中，我们列出了与显示存储器地址对应的MPU存储器地址。MPU存储器地址来自系统地址总线。若显示存储器地址也采用相同的编址，那么由同步计数器形成的地址就便于计算了。如RAM地址输入端RA5，MPU存储器行地址是总线地址A1，与之相对应，显示存储器行地址是水平计数器H1，我们就认为H1表示显示存储器地址的A1位。其它位地址也作类似考虑。

表2.5.7中，根据表2.5.5，按总线地址位顺序列出由同步计数器形成的显示存储器地址和总线地址的对应关系。并补进高四位地址。

从表中可以看出，显示存储器地址A15总是0。A14的值我们在2.4节中讨论过，在显示高分辨率图形第二页时A14=1，其它显示方式下，A14=0。A13=HIRES·$\overline{\text{PAGE2}}$，$\overline{\text{PAGE2}}$ 表示锁存器F14输出PAGE2的非，置显示第一页后为1。因此，A13在显示高分

表 2.5.6 同步计数器和显示的关系

垂直计数器					点阵行计数器			水平计数器						显示	
V4	V3	V2	V1	V0	VC	VB	VA	H5	H4	H3	H2	H1	H0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	行消隐	第一行40个字符
					⋮					⋮					
					1	1	1	0	1	0	1	1	1		
					0	0	0	0	1	1	0	0	0	第一行第一个字符	
		⋮			⋮										
1	0	1	1	1	1	1	1								
					0	0	0	0	1	1	0	0	1		
					⋮					⋮					
					1	1	1	1	1	1	1	1	1		
					0	0	0	0	0	0	0	0	0	显示一场	
					⋮					⋮					
					1	1	1	1	1	1	1	1	1		
1	1	0	0	0	0	0	0	0	0	0	0	0	0	场消隐	
		⋮			⋮					⋮					
1	1	1	1	1	1	1	1	1	1	1	1	1	1		

表 2.5.7 显示存储器地址

总线地址	A7	A6	A5	A4	A3	A2	A1	A0
同步计数器	V0	$\overline{H5}$ +V4 +C3	$\overline{H5}$ +V3 +C2	H4+V4+C1	H3+V3+1	H2	H1	H0
总线地址	A15	A14	A13	A12	A11	A10	A9	A8
同步计数器	0	HIRES· PAGE2	HIRES· $\overline{PAGE2}$	HIRES·VC +$\overline{HIRES}$·HBL	HIRES·VB +$\overline{HIRES}$·PAGE2	HIRES·VA +$\overline{HIRES}$·PAGE2	V2	V1

辨率图形第一页时，A13＝1，在其它显示方式下，A13＝0。表中写明：

$$A12 = HIRES \cdot VC + \overline{HIRES} \cdot HBL$$

$$A11 = HIRES \cdot VB + \overline{HIRES} \cdot PAGE2$$

$$A10 = HIRES \cdot VA + \overline{HIRES} \cdot \overline{PAGE2}$$

在这三个逻辑表达式中，$\overline{HIRES}$ 表示译码器F14输出HIRES的非，置低分辨率图形显示方式后为1。下面分两种情况讨论这三位地址：①在高分辨率图形方式下，HIRES＝1、$\overline{HIRES}$ ＝0，这时A12、A11和A10三位地址就分别由VC、VB和VA决定，即这三位地址码的值与字符点阵行数计数器一致。②在低分辨率图形方式下，$\overline{HIRES}$＝1、

HIRES=0，A12、A11和A10分别由HBL、PAGE2和 $\overline{\text{PAGE2}}$ 决定，HBL是视频信号发生器产生的消隐控制信号，在字符显示期间，HBL=0；消隐（保持黑）期间，HBL=1。因此A12也是在字符显示期间为0；消隐期间为1。PAGE2在置显示第二页时为1，$\overline{\text{PAGE2}}$ 在置显示第一页时为1，因此在低分辨率图形方式下，显示第一页时，A10=1，A11=0；显示第二页时，A11=1，A10=0。综上所述，现将这几位地址码列于表2.5.8。

表 2.5.8 与显示方式有关的几位地址码

	A15	A14	A13	A12	A11	A10
低分辨率图形第一页	0	0	0	0	0	1
低分辨率图形第二页	0	0	0	0	1	0
高分辨率图形第一页	0	0	1	VC	VB	VA
高分辨率图形第二页	0	1	0	VC	VB	VA

1．低分辨率图形/文本方式第一页显示存储器地址

在这种显示方式下，地址的高六位是000001。我们根据同步计数器各位的输出，按照表2.5.7可以计算出其它地址位的值。以显示第一行字符为例，计算出的地址码列于表2.5.9。可以看出显示第一行字符的地址码是0400～0427。换句话说，显示第一行字符的ASCII码依次放在地址码为0400到0427的随机存储器中。

表 2.5.9 文本显示第一行地址

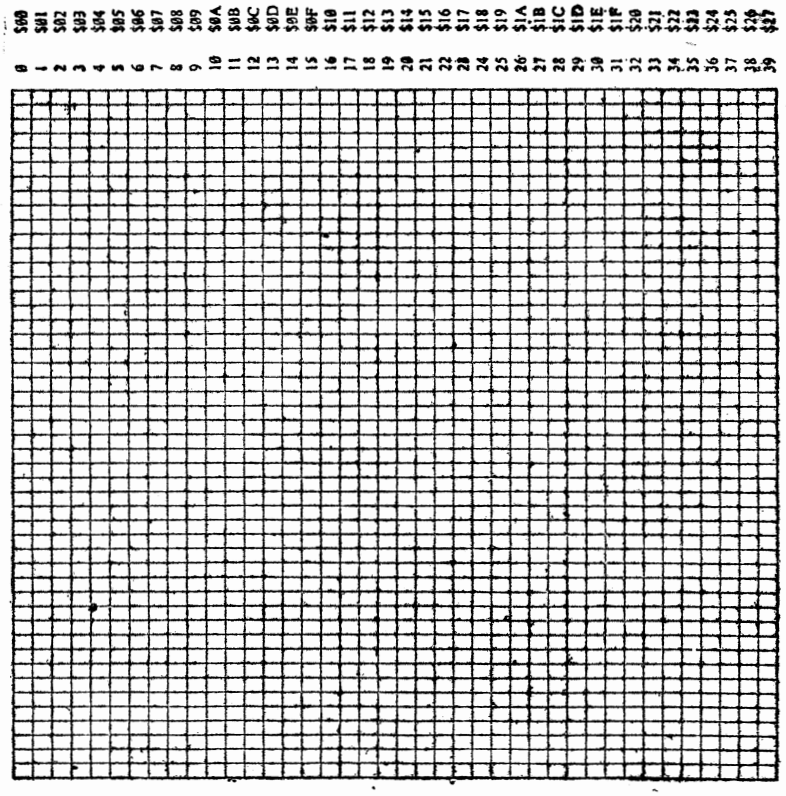
A15～A11	A10	A9～A6	A5 A4 A3 A2 A1 A0	十六进制地址码 高位	低位
00000	1	0000	0 0 0 0 0 0	04	00
			0 0 0 0 0 1	04	01
			0 0 0 0 1 0	04	02
			0 0 0 0 1 1	04	03
			0 0 0 1 0 0	04	04
			⋮	⋮	
			1 0 0 1 1 1	04	27

显示第二行字符时，地址A6～A0同第一行，而垂直计数器V0变成1，V0相当于总线地址A7。因此，V0加1，地址相应增加80（十六进制数），第二行字符显示存储器地址为0480～04A7。依次类堆，可算出第三行到第七行显示存储器的地址，参看表2.5.10。

显示第八行到第十五行时，垂直计数器V3变了，它影响到地址A5和A3，V3由0变1，显示存储器相应地址加28（十六进制）。以第八行为例：显示第八行时，垂直计数器V2、V1和V0又恢复到0，仅V3=1。因此，第八行显示存储器地址为0428～044F。读者

表 2.5.10 文本方式第一页的显示存储器地址

$400	1024
$480	1152
$500	1280
$580	1408
$600	1536
$680	1664
$700	1792
$780	1920
$428	1064
$4A8	1192
$528	1320
$5A8	1448
$628	1576
$6A8	1704
$728	1832
$7A8	1960
$450	1104
$4D0	1232
$550	1360
$5D0	1488
$650	1616
$6D0	1744
$750	1872
$7D0	2000



可试计算其它行显示存储器地址，再与表2.5.10比较之。

若显示低分辨率图形/文本第二页，则地址的高六位为000010，其它地址位数值不变，可计算得地址范围为0800～0BFF。

以上是从硬件角度所分析的显示存储器地址。软件中是根据显示的横坐标和纵坐标来计算显示存储器地址的。横坐标CH表示将要显示的字符在一行中的位置，显示完一个字符，CH加1。纵坐标CV表示显示的当前行数。计算显示存储器时，首先根据纵坐标CV计算显示行的首地址（基本地址），然后再加上CH的数值，就可确定显示存储器的地址，监控程序中的基本地址计算程序如下：

```
FBC1  PHA          A为显示纵坐标CV的值，A进栈。
      LSR          A右移一位
      AND#03       保留纵坐标的1和0位
      ORA#04       A“或”04送A
      STA  BASH    存基本地址高位
FBC9  PLA          A退栈，A为显示纵坐标
      BCC  BSCLC2  判纵坐标的奇偶性
```

```
FBCE    ADC#7F      奇数：地址低位加80
BSCLC2  STA BASL
        ASL         A左移一位
        ASL         A左移一位
        ORA BASL
        STA BASL    存基本地址低位
        RTS
```

我们先分析一下表2.5.10中基本地址与纵坐标的关系，再来看这段程序，它的算法就很容易看明白了。

表 2.5.11 基本地址高位与纵坐标的关系

基本地址高位	04		05		06		07	
纵坐标	十进制	二进制	十进制	二进制	十进制	二进制	十进制	二进制
	0	00000	2	00010	4	00100	6	00110
	1	00001	3	00011	5	00101	7	00111
	8	01000	10	01010	12	01100	14	01110
	9	01001	11	01011	13	01101	15	01110
	16	10000	18	10010	20	10100	22	10110
	17	10001	19	10011	21	10101	23	10111

表2.5.11中列出基本地址高位与显示纵坐标之间的关系。纵坐标以二进制写出。请注意，纵坐标的二、一位（最低位为0位），若把这两位看成是一、0位，那么它们形成的数，加上4，刚好就是基本地址高位。按着这个规律，从FBC1开始的五条程序，就由纵坐标算出基本地址的高位BASH。

从表2.5.10中可以看出，纵坐标为偶数时，基本地址低位小于80；纵坐标为奇数时，基本地址低位大于80。若从偶数纵坐标加1得到奇数坐标，那么地址低位在原来基础上加80即可。在表2.5.12中仅列出纵坐标为偶数时，基本地址低位与纵坐标的关系。请特别注意纵坐标二进制表示中四、三位和基本地址低位的关系。上述程序中，从FBC9开始就是用来形成基本地址低位的。它首先根据FBC2中的移位，判断纵坐标是奇还是偶，若纵坐标是奇数，进位位C＝1，程序顺序执行，基本地址加80；若纵坐标是偶数，C＝0，跳过FBCE那条程序，基本地址不加80。然后将保留下来的纵坐标四、三位左移两次，形成新的六、五位，再和原来的四、三位求逻辑或就得到了基本地址低位。

当我们已经知道屏幕对应的内存地址后，可将显示内容直接送进内存，以得到快速显示。在文本方式下，把待显示字符的ASCⅡ码送显示存储器。在低分辨率图形方式下，由于十六种颜色只需半个字节（四位二进制），所以在一个存储单元内可放两种颜色编码，它对应着屏幕上两个彩色块。若把显示一个字符的屏幕范围分成上下两小块，那么存

表 2.5.12 基本地址低位与偶数纵坐标的关系

基本地址低位	00		28		50	
纵坐标	十进制	二进制	十进制	二进制	十进制	二进制
	0	00000	8	01000	16	10000
	2	00010	10	01010	18	10010
	4	00100	12	01100	20	10100
	6	00110	14	01110	20	10110

储器中一个字节的高四位对应下面那个小块，而低四位对应上面那个小块。可以看出低分辨率图形显示所需存储器大小同文本显示。实际上，它们占用的地址也是相同的。

2．高分辨率图形方式的显示存储器地址

从表2.5.7中容易看出，高分辨率图形的高三位地址码，第一页时A15、A14、A13为0、0、1，第二页时为0、1、0。与低分辨率图形方式对比，高分辨率图形显示存储器地址的低十位A9～A0没有改变，只是地址A12、A11和A10分别与字符点阵行计数器VC、VB和VA有关。当VC＝VB＝VA＝0时，高分辨率图形显示存储器地址低十二位与低分辨率图形显示存储器地址相同，那么很容易写出这时的显示存储器地址，如表2.5.10。点阵中的这些行，相当于文本方式下，各行字符点阵的第一行。

字符点阵行计数器低位VA影响显示地址A10，显然VA加1，显示地址增加400（相当于十进制数1024）。按此规律，我们可以算出整个高分辨率图形显示存储器地址。

为了快速画出高分辨率彩色图形，也可将彩色编码直接送入显示存储器内。

2.5.5 RAM的刷新

Apple中的RAM采用的是动态随机存取存储器。动态存储器是靠电容来存储信息。由于泄漏电流的存在，电容上的电压会逐渐放掉。为了保持电容上的信息，必须不断地进行重写，或称作刷新。一般要求2ms刷新一遍。这意味着，在2ms内，RAM中所有的存储单元至少要刷新一次。

4116在每次读或写周期，由七位行地址所选中的一整行均被刷新。微处理机中，MPU读写存储器的地址是随机的，它访问那些与正在执行程序有关的存储器，而暂时没有读写的存储器就不能被刷新，这样就会丢失这些存储单元内的信息。因此有的计算机中要附加专门的电路来控制RAM的刷新，也有的计算机是由CPU来控制，如Z80 CPU在取指周期的最后两个时钟周期送出刷新地址，刷新一行。其后，地址顺序增加。

Apple中没有专门设置刷新控制电路，而是采用隐蔽刷新的方式。在MPU不读写存储器的时间里，刷新存储器。这种刷新过程不被觉察，也不引起MPU延迟执行。

从前面的讨论中知道，在一个φ0周期，读写了两次RAM；MPU存储器和显示存储器。MPU存储器的读写地址是随机的，而显示存储器的读地址是很有规律的。我们也知道，正常的读写周期刷新了由行地址选中的一行。如果显示存储器行地址安排的得

当，是否能满足刷新的需求呢？事实证明是可以的。

重新列出显示存储器的行地址如下：

RAM地址	RA6	RA5	RA4	RA3	RA2	RA1	RA0
	HIRES·VC	V1	V0	V3+H3+1	H2	H1	H0
同步计数器	+ $\overline{\text{HIRES}}$ ·HBL						

从上面同步计数器形成的显示存储器行地址看出：Apple 不能实现RAM的顺序刷新，有些存贮器刷新周期较短，有些存储器刷新周期较长。为此，我们来分析最长的刷新周期是多少。

同步计数器时钟信号的频率为1.023MHz，最低位H0的周期约为2μs。以此，我们列出与显示存储器行地址有关的同步计数器位的周期：

RAM行址		RA5	RA4			RA2	RA1	RA0
同步计数器	V3	V1	V0	Vc	H3	H2	H1	H0
周期	8ms	2ms	1ms	0.5ms	16μs	8μs	4μs	2μs

可以看出，RAM的行地址中RA4、RA2、RA1、RA0的变化周期小于2ms，RA5的变化周期等于2ms。

RAM的地址输入端RA3的信号为V3+H3+1。RA3和V3、H3的关系如下：

V3	H3	RA3
0	0	1
0	1	0
1	0	0
1	1	1

可以看出，当V3=0时，RA3等于H3的反；V3=1时，RA3等于H3。RA3的变化周期与H3变化周期相同，也是16μs。

RAM地址输入端RA6的输入信号为：

HIRES·VC+ $\overline{\text{HIRES}}$ ·HBL。监视器扫描一行的时间约为64μs：显示时间为40μs；保持黑（HBL）的时间约为24μs，此时HBL=1。因此在低分辨率图形方式时，RA6的变化周期约为64μs。

在高分辨图形方式，RA6的状态取决于VC，这时RA6的变化周期同VC的变化周期，时间为0.5ms。

综上所述，RAM显示存储器行地址中RA5的变化周期最长，为2ms，完全满足RAM刷新周期的要求。地址中其它位的变化周期小于2ms，表示有些存储器刷新周期也小于2ms，或者说有些存储器在2ms内刷新了多次。

最后还值得一提的是，Apple中三排RAM显示存储器的行地址是相同的，而且采用了系统时基中的行选通信号$\overline{\text{RAS}}$作RAM的行选通，因此，在行地址有效期间，三排RAM的行地址均有效。那么刷新也是在三排RAM中同时进行的。

总之，Apple是用显示存储器的行地址在正常读周期中完成RAM刷新的。

2.6 外设插座

在主机印刷电路板的后部，装有八个外部设备插座，它是 Winchester＃2HW250-C0-111型50点印刷板插座。这八个插座自左至右的(从正前方看)编号为0，1，…，7。可以在这些插座上插入各种各样的外设接口板。那么，怎样合理而有效地利用这些外设插座，是用户普遍关心的问题。图 2.6.1 中画出了其中一个插座的引线图。表 2.6.1 按引脚顺序列出各引线的名称和作用。

下面我们对一些引脚做补充说明：

1. 三条设备码选通线

从2.3节中知道，Apple保留了 4K地址空间供 I/O 使用。而外设插座上的外设可使用的设备码范围是 C080～CFFF。这些设备码分成三部分，通过主板上的译码器将产生三类设备选通信号，现分述如下：

（1）装置选择 $\overline{\text{DEV SEL}}$

当插座号为 s 时，该插座的装置选择设备码为 C080＋s×。× 表示任意值。地址总线上出现这个范围内的地址码时，主板上的译码器 H2输出负脉冲，送至相应插座的 41 点，作为 $\overline{\text{DEV SEL}}$信号。同时，外设板上也可用这个设备码的低四位作为设备号，以区别不同的设备或装置。

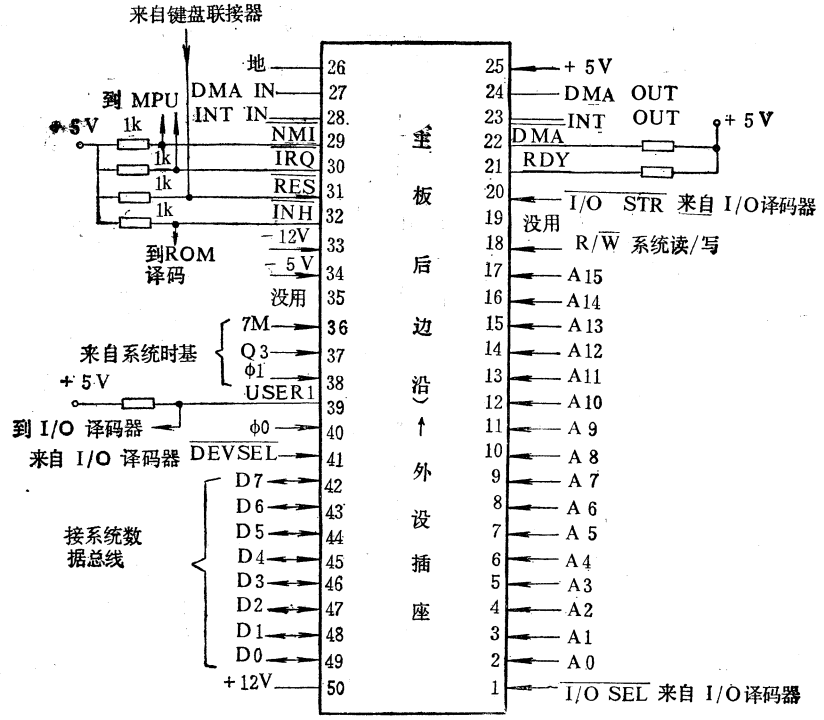


图 2.6.1 外设插座

（2）I/O 选择 $\overline{\text{I/O SEL}}$

表 2.6.1 外设插座引脚说明

引脚号	信号名称	说　　明
1	$\overline{\text{I/O SEL}}$	输入/输出选择。这条线通常为高。当MPU访问CS页存储器时才变低。其中S为各外设插座的插座号(S=1…7)。这个信号在ϕ0 周期有效。可驱动10个LSTTL负载。0号插座不出现该信号。
2～17	A0～A15	经过缓冲的地址总线。线上地址在ϕ1周期变得有效,并保持到ϕ0结束为止。可驱动5个LSTTL负载。
18	R/$\overline{\text{W}}$	缓冲后的"读/写"控制信号。该信号与地址总线同时有效。"读"周期为高;"写"周期为低。可驱动2个LSTTL负载。
19	SYNC	视频信号中的同步信号。只有在7号插座上出现。该引脚接C13(74LS51)的8脚。
20	$\overline{\text{I/O STR}}$	外设板上扩展ROM选通信号。当地址总线上出现C800到CFFF之间的任何地址时,在ϕ0期间这条线变低,可驱动4个LSTTL负载。
21	RDY	准备好。6502的RDY输入。若在ϕ1周期内，这条线拉低，就会使MPU暂停,而地址总线保持当前访问的单元的地址。
22	$\overline{\text{DMA}}$	直接存储器存取。这条线通过1kΩ电阻接 +5V。平时保持高电平。一旦将这条线拉低,将切断6502的时钟输入和地址总线，MPU暂停工作。
23	INT OUT	链式中断输出端。送低一级优先权的设备中去。此脚一般接28脚(INT IN)。
24	DMA OUT	链式DMA输出端。送低一级优先权的设备中去。此脚一般接27脚(DMA IN)
25	+5V	+5V电源。为所有外设能够提供500mA电流。
26	GND	系统接地端。
27	DMA IN	链式DMA输入端。来自高一级优先权的设备。此脚一般接24脚(DMA OUT)
28	INT IN	链式中断输入端。来自高一级优先权设备。此脚一般接23脚(INT OUT)
29	$\overline{\text{NMI}}$	不可屏蔽的中断请求信号。
30	$\overline{\text{IRQ}}$	中断请求信号。这条线被拉低时,如果6502的中断标志I没有被置位(I=0),Apple开始一个中断周期,6502将跳到中断服务子程序。子程序的首地址存在03FE和03FF中。
31	$\overline{\text{RES}}$	复位信号,这条线被拉低时,微处理器开始RESFS周期
32	$\overline{\text{INH}}$	主板上ROM(D000～FFFF)的禁止信号。这点通过电阻接+5V,当这点拉低后,主板上的ROM被禁读。
33	−12V	−12V电源。为所有外设能够提供200mA电流。
34	−5V	−5V电源。为所有外设能够接供200mA电流。
35	COLOR REF	彩色参考信号。频率为3.58MHz。只有第7号插座上才有此信号,其它插座上无此信号。
36	7M	7MHz信号。这条线能驱动2个LSTTL负载
37	Q3	2MHz非对称信号。这条线能驱动2个LSTTL负载。

续表

引脚号	信号名称	说　　明
38	ϕ1	MPU的1相时钟。频率为1MHz。这条线能驱动2个LSTTL负载
39	USER1	这条线被拉低时，所有的I/O地址不能被译码。
40	ϕ0	MPU的0相时钟信号。频率为1MHz。这条线能驱动2个LSTTL负载。
41	$\overline{\text{DEV SEL}}$	设备选择信号。当地址总线上的地址为C0M0到C0MF之间的任何地址时，在ϕ0周期，被选中的插座上的这条线变成有效。这条线能驱动10个LSTTL负载。其中M=8+S，S为选中的插座号。
42—49	D0—D7	经过缓冲的双向数据总线。在写周期内，这些线上的数据在ϕ0周期，至少200ns后才稳定可用。在读周期中，ϕ0周期结束前至少要稳定100ns。每条数据线能驱动1个LSTTL负载。
50	+12V	+12V电源。为所有外设能够提供250mA电流。

当插座号为S时，该插座的I/O选择设备码为CS××，计256个地址。通常外设接口的管理程序固化在板上的ROM中，而ROM占用的地址就是I/O选择设备码的地址。地址总线上出现CS××的地址码时，主板上译码器H12的相应端输出负脉冲，并送到相应插座的1点，作为$\overline{\text{I/O SEL}}$信号。同时，外设也可用这个设备码的低八位作为板上ROM的地址，读出该外设的管理程序。

（3）外设扩展ROM选通 $\overline{\text{I/O STR}}$

外设板上扩展ROM的地址空间为C800～CFFF。地址总线上出现这个范围内的地址码时，主板上的译码器F12输出负脉冲，并送到全部外设插座的20点，作为$\overline{\text{I/O STR}}$信号。同时外设也可用这个设备码的低十一位作为扩展ROM的地址读出其程序。注意CFFF是个专用地址，请参看2.3节。

表2.6.2中列出各插座可使用的设备码范围。

表 2.6.2 “输入/输出”选通信号的设备码

插座号	设备码		
	$\overline{\text{DEV SEL}}$	$\overline{\text{I/O SEL}}$	$\overline{\text{I/O STR}}$
0	C080～C08F	无地址空间	C800～CFFF 8个外设插座公用
1	C090～C09F	C100～C1FF	
2	C0A0～C0AF	C200～C2FF	
3	C0B0～C0BF	C300～C3FF	
4	C0C0～C0CF	C400～C4FF	
5	C0D0～C0DF	C500～C5FF	
6	C0E0～C0EF	C600～C6FF	
7	C0F0～C0FF	C700～C7FF	

2．与MPU直接有关的信号

与MPU直接有关的信号的详细说明在2.1节。这些信号有：

16根地址线　A15～A0经过缓冲，来自MPU。

8根数据线　D7～D0经过缓冲、双向。

“读/写”控制线　R/$\overline{W}$

不可屏蔽中断信号　$\overline{NMI}$

准备好信号　RDY

复位信号　$\overline{RES}$

中断请求信号　$\overline{IRQ}$

直接存贮器存取　$\overline{DMA}$

3．时钟信号

连到插座上的时钟信号有φ0、φ1、Q3、7M和彩色同步信号COLOR REF。与之有关的信息，请参看2.2节。

4．与译码器有关的信号

与译码器有关的信号是$\overline{INH}$和USER1。请参看2.3节。

5．电源

外设接口板的电源均由主机供给。连到插座上的电源共有四种：±5V、±12V。注意：外设从主机电源索取的电流是有限制的。

6．链式中断和链式DMA

外设插座上还备有链式中断输入、输出端（INT IN、INT OUT）和链式DMA输入、输出端（DMA IN、DMA OUT），可供用户建立链式中断、链式DMA结构。我们仅以链式中断为例，作一简单的说明。

当有多个外设需要采用中断方式与主机交换数据时，可以把它们的中断申请输出“线或”在一起，连到MPU的$\overline{IRQ}$端。在这些外设向主机申请中断时，主机应能判断申请来自哪个外设，也应能根据中断的优先级别，决定先为哪个外设服务。这就是优先级排队和中断识别的问题。6502无优先级排队功能，这个工作只有由软件或附加硬件来完成。

软件排队就是利用程序查询的办法来判断中断来自哪个外设。一旦查出中断源，便立即转入该外设的服务程序。那么先被查询的外设优先权级别就高。例如在Apple监控程序中，查询了BRK中断和用户中断，具体程序如下：

```
FA40: STA  $45  保存累加器A的内容在45中
      PLA       状态字P退栈，A为原状态字内容
      PHA       状态字P进栈
      ASL       A左移
      ASL       A左移
```

```
        ASL          A左移，将状态字中BRK中断标志，移至累加器的最高位
FA47    BMI FA4C     状态字中B=1，转入处理BRK中断
FA49    JMP (03FE)   跳至用户中断服务子程序
FA4C    ┌──────┐
        │BRK中断 │
        │处理程序│
        └──────┘
```

当MPU接收到中断请求信号（$\overline{IRQ}$变低）后，若中断请求被承认，MPU首先将程序计数器（PCH、PCL）和状态字寄存器（P）顺序压入堆栈，然后取出中断向量。自启动ROM中，$\overline{IRQ}$服务子程序的地址为FA40。执行这段程序时，首先把累加器内容存在专用地址中。然后将状态字退栈。根据状态字判断有没有BRK中断，若有，则转入处理BRK中断子程序；若无BRK中断，转用户中断服务程序。从程序的执行顺序可以看出，BRK中断的优先级高于用户中断。

若申请中断的外设数量增加，识别程序也相应加长，这必然要影响实现中断的速度。

Apple Ⅱ中借助外设插座的INT IN和INT OUT，在外设接口板的少量硬件配合下，可以形成链式中断结构，这样能够简化识别程序，提高实现中断的速度。每一个插座的INT OUT接到下一号插座的INT IN，构成一个“菊花链”，如图2.6.2所示。显然，0号插座内的外设优先级最高，7号插座内的外设优先级最低。

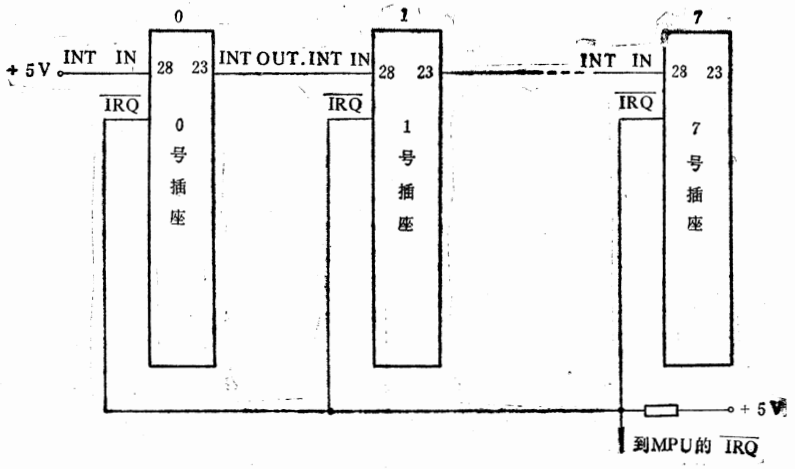


图 2.6.2 链式中断结构

当某个外设需要中断时，必须检查它的INT IN信号，若INT IN为低（即高一级的INT OUT为低），表示有高优先级的外设正在申请中断，本外设不能向主机申请中断。若INT IN为高，本外设可以向主机申请中断，使$\overline{IRQ}$变低。同时将它的INT OUT变

低，以便通知比它优先级低的外设，为了实现上述功能，各外设接口必须具有相应的硬件。

2.7 主板上的外设接口

主板上的外设接口包括扬声器、键盘输入、盒式磁带“输入/输出”、游戏器等等。键盘输入接口将在2.10节中介绍，本节简要说明其它几个接口。

有两个片子是这几个接口共用的，一片是译码器74LS138，位于F13，它的作用是给出接口设备码的选通信号（请参看2.3节）。另一片是“8选1”多路开关74LS251，位于H14。它的作用是将接口信号送数据总线。

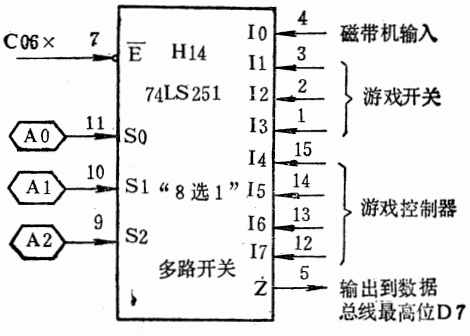


图 2.7.1 多路开关H14的输入与输出

2.7.1 “8选1”多路开关

顾名思义，“8选1”就是从八个输入信号中选择一个信号送输出端输出。先将该片的输入、输出列出，如图2.7.1。

下面分析用以选择输入信号的设备码。“8选1”多路开关的设备码由使能端$\overline{E}$和选择端S0、S1和S2决定。从该片特性可知，$\overline{E}$为低电位时，该片有数据输出，否则呈高阻状态。现$\overline{E}$接F13的Z6，当选通设备码C06×时，F13的Z6输出为低电位，这时该片有数据输出。那么到底选择八路输入的哪一路，则由S0、S1和S2决定。现S0、S1和S2分别接地址总线的A0、A1和A2。A3可以是任意数，因此“8选1”多路开关的设备码可分为如下两种情况：

$$A3=\begin{cases}0 & \text{设备码 C060～C067}\\ 1 & \text{设备码 C068～C06F}\end{cases}$$

这说明，每路开关将有两个设备码，如表2.7.1所示。多路开关的输出送数据总线最高位D7。

表 2.7.1 多路开关的设备码与功能

A15～A4	A3	A2A1A0	设备码	功能
C06	0或1	0 0 0	C060或C068	选通磁带机入
C06	0或1	0 0 1	C061或C069	选通SW0
C06	0或1	0 1 0	C062或C06A	选通SW1
C06	0或1	0 1 1	C063或C06B	选通SW2
C06	0或1	1 0 0	C064或C06C	选通PDL0
C06	0或1	1 0 1	C065或C06D	选通PDL1
C06	0或1	1 1 0	C066或C06E	选通PDL2
C06	0或1	1 1 1	C067或C06F	选通PDL3

2.7.2 扬声器

扬声器接口的线路很简单，从设备码C03×选通到扬声器放声的电路如图2.7.2所示。

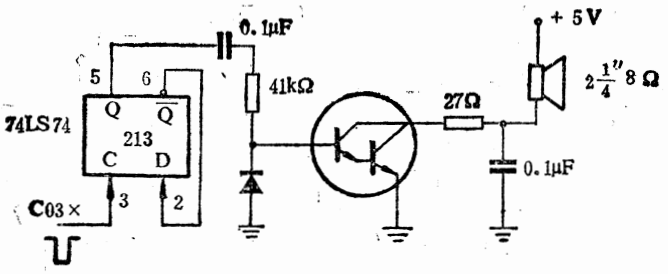


图 2.7.2 扬声器接口

D触发器接成计数器形式，设备码C03×每选通一次，D触发器就翻转一次，信号经电容耦合到放大器放大后，送扬声器发声。

发声频率取决于选通设备码C03×的频率。由于选通C03× 两次，放大器输出变化一个周期，故发声基频是选通设备码频率的1/2。设每隔500μs选通一次 C03×，则发声频率为：

$$\frac{1}{2}\times\frac{1}{500\times10^{-6}}=1000\text{Hz}$$

如下的这小段程序，从地址T起动，就能使扬声器发声。

```
T: LDA C030     选通扬声器设备码
   LDX #6B      决定发声频率，X越大，频率越低
Q: DEX          }
   BNE Q        }延时
   JMP T
```

监控系统中有一段程序是用于发声的，每执行一次该段程序，扬声器就发一声“嘟”，其程序如下：

```
FBE2  LDY #C0     决定发声长短，y为C0（十进制 192）时，发声时间为
                  105ms
FBE4  LDA #0C     决定发声频率，该值越大，频率越低。该值为 0C（十
                  进制12）时，基频为916Hz
      JSR FCA8    延时。A为0C（十进制12）时，延时529μs
      LDA C030    选通扬声器设备码
      DEY
      BNE FBE4
      RTS
```

延时子程序如下：

```
FCA8  SEC
FCA9  PHA
FCAA  SBC#01
      BNE FCAA
      PLA
      SBC#01
      BNE FCA9
      RTS
```

从附录C中，查出各条指令执行的拍数，就可算出执行FCA8这段延时程序的时间。设A为转入这段程序时累加器的初值，则延时时间由下述公式计算：

$$(2.5A^2+13.5A+7)\mu s。$$

2.7.3 盒式磁带

1．输出到盒式磁带

输出到盒式磁带机的接口非常简单，和扬声器输出接口差不多，设备码是C02×。设备码选通线作为一个计数器的计数脉冲，计数器的输出经120:1的衰减，送给盒式磁带机。电路见图2.7.3。

设备码C02×每选通一次，计数器就翻转一次，因此送往磁带机的信号是方波，幅值为几十毫伏。

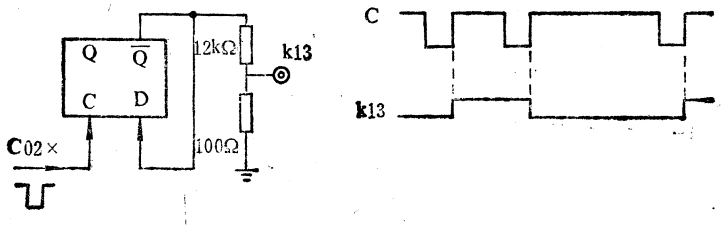


图 2.7.3 输出到盒式磁带机的接口及波形

这部分电路是简单的，但“设备码C02×选通一次，计数器就翻转一次，送出方波”，这个概念是重要的。监控系统中有一段较复杂的“写磁带子程序”，分析那段程序时要用到这个概念。关于信号在磁带机上的起始标志、记录格式、校验，都由那段程序完成。第四章将对那段程序做详细的分析。

2．盒式磁带输入（计算机接收磁带机输入的信息）

从磁带机输入的信息经电容C耦合到运算放大器的输入端，再经限幅放大，送到“8选1”多路开关，最后送到数据总线最高位D7，选通磁带输入的设备码是C060或C068，这部分电路如图2.7.4。

来自盒式磁带的信号接近方波，由K12输入。放大器接成正反馈形式，是为了对信号进行整形放大。放大器输出为方波，并送往多路开关。

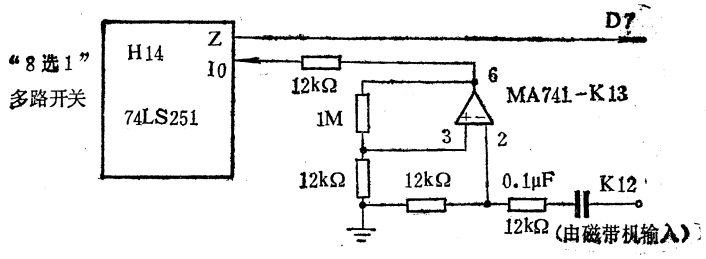


图 2.7.4 盒式磁带机输入接口

监控系统中专门有一段"读磁带"子程序，关于如何识别记录的开始，如何把数据整理送到内存，如何检验数据对不对等问题，均由那段程序完成。第四章将对此做详细分析。

2.7.4 游戏

游戏部分包括一位输出、游戏开关、游戏控制器、公共选通等，其接口电路如图2.7.5。

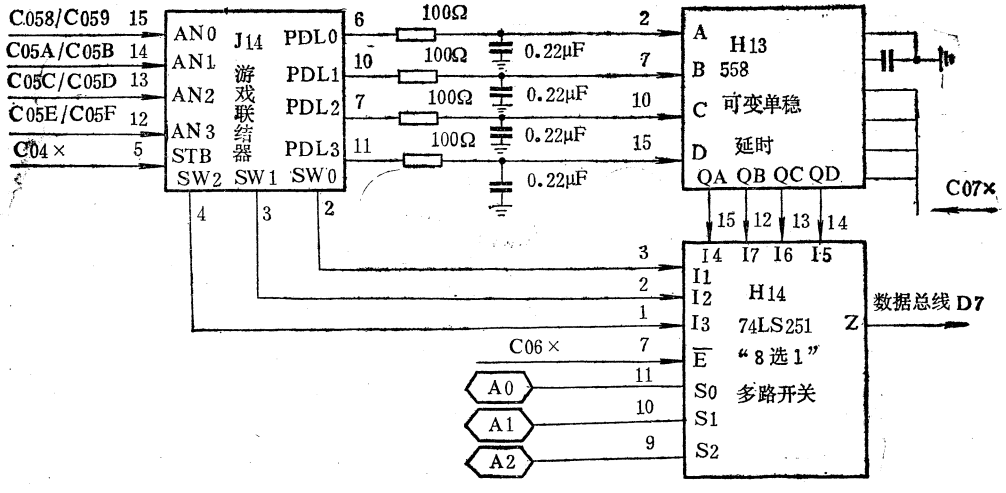


图 2.7.5 游戏接口

下面对各项游戏进行具体介绍：

1．一位输出

本机设置了几个设备码，当这些设备码被选通时，就在相应的端点出现高电平或低电平，用户可通过游戏联结器把这些信号引出，按需要做各种用途。例如，经放大后接通指示灯、电铃等。本机有四个这样的设备码，故有四个一位输出。

这四个一位输出来自带锁存器的译码器74SL259，2.3.5节中已提及这些设备码，

表 2.7.2 一位输出的设备码和功能

设备码	功能	设备码	功能
C058	清AN0	C059	置AN0
C05A	清AN1	C05B	置AN1
C05C	清AN2	C05D	置AN2
C05E	清AN3	C05F	置AN3

其功能如表 2.7.2 所示。

例如下面这段程序，从T起动，即可在AN0 输出方波。

```
T: LDA C058          清 AN0
   LDX #30
Q: DEX               }延时
   BNE Q
   LDA C059          置AN1
   LDX#10
R: DEX               }延时
   BNE R
   JMP T
```

2. 游戏开关

游戏开关就是在机外接开关，通过游戏联接器与板上接口相连，电路非常简单。以SW0为例，其电路如图2.7.6所示。

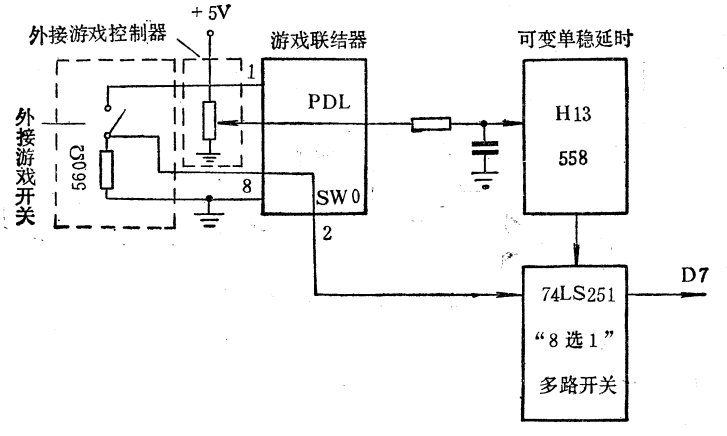


图 2.7.6 外接游戏开关和游戏控制器

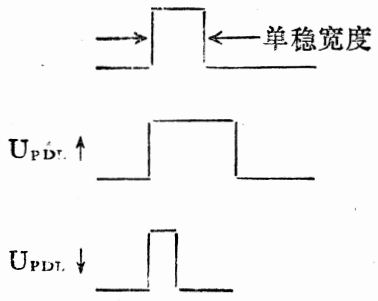
开关合上时，SW0为高电位，断开时为低电位，通过“8选1”多路开关，把其状态送往数据总线D7，选通游戏开关的设备码如下：

开关： SW0 SW1 SW2

设备码：C061或C069 C062或C06A C063或C06B

3．游戏控制器

游戏控制器 PDL就是在机外接电位器，如图2.7.6所示。调节电位器得到连续可变的电压，经阻容滤波送到位于H13的片 子 558。这是一个可变单稳延时脉冲，调节电位器就是调节单稳脉冲的宽度，如下所示：



这个信号也送到"8选1"多路开关，最后送到数据总线最高位D7，四个游戏控制器的设备码为：

游戏控制器	PDL0	PDL1	PDL2	PDL3
设备码	C064或C06C	C065或C06D	C066或C06E	C067或C06F

每选通一次游戏控制器的设备码，就把该路单稳输出送到数据总线D7，程序不停地查询D7的状态，就可知单稳的宽度，用这个宽度来反映PDL的位置。

监控系统中有一段程序专门用于检测单稳的宽度，程序如下：

```
FB1E: LDA C070     启动单稳可变延时
      LDY #00
      NOP
      NOP
FB25: LDA C064,X   X取0～3时，可分别选通PDL0～PDL3
      BPL FB2E     D7＝0时，表示单稳延时结束
      INY          计数器加1
      BNE FB25     继续测单稳宽度
      DEY
FB2E: RTS
```

从该段程序返回时，Y的值即可表示PDL的位置。

4．公用选通

公用选通是指一些备用的设备码。当设备码 C04X被选通时，译码器F13输出一负脉冲，用户可从游戏联接器取出此信号，按需使用。

2.8 同步计数器

2.8.1 同步计数器的功能

同步计数器在Apple机中完成两个功能：其一是控制一系列同步脉冲信号出现的时

刻和持续的时间，这些信号是电视显示所要求的；其二是形成显示字符或显示彩色图形的行计数和列计数信号，这两个信号的作用请参看2.5节。

为了讲明对同步计数器的要求，有必要先简单介绍一下有关全电视信号的基本知识。

我们知道，Apple机可以经过一个非常简单的射频调制器与一般NTSC制式的彩色电视机连接，也可以直接与彩色监视器连接。为了本节和下一节叙述的需要，首先介绍一下NTSC彩色电视标准。这种制式是美国、加拿大、日本等国采用的一种彩色电视制式。它的整个电视屏幕为525行扫描线，采用隔行扫描制。每帧图象有两场，场频60Hz，帧频30Hz。每秒可显示30帧图，因为每一帧有525行，故行频为525×30＝15750Hz，每行的扫描周期约为63.56μs。在一行的视频信号中包含有四种信号(见图2.8.1，Apple中的信号与一般彩色电视信号略有不同)：

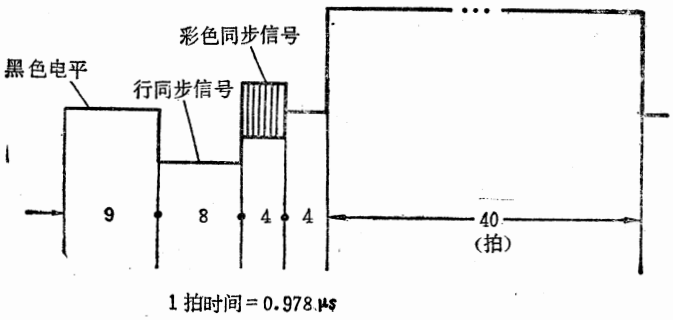


图 2.8.1 Apple的行视频信号

1．图象信号：它在光点扫描的正程起作用，详见下节。

2．行消隐信号：电平是“黑色”电平，在这个信号作用期间屏幕是黑的，它的作用一是保证看不到回扫线，二是保证显示在屏幕中央的一行40个字符（或相当于40个字符宽度的彩色图形）的左右两侧为黑色。

3．行同步信号：电平比行消隐信号更“黑”，它的作用是同步每一行的信号。

4．彩色同步信号：这是3.58MHz的正弦信号，作为彩色副载波信号的相位基准，用以确定图象信号所应该有的色调，这部分较详细的介绍放在下节。

在一场的视频信号中，除了包含上述四种行频信号之外，还要有：

1．场消隐信号：它是保证使显示在屏幕中央的24行字符以外的空间（屏幕的顶部和底部）保持黑色；使光点在每场扫描结束时从屏幕底回扫到顶时消隐掉。

2．场同步信号：起同步每一场的信号的作用。

综上所述，电视屏幕上要显示一幅图象，需要六种信号。消隐信号和同步信号是依一定规律循环出现的信号，而图象信号（储于指定的内存单元）在调往屏幕显示的过程中，也是周而复始地从内存单元中依次调出，也要有一定的规律。要保证这六种信号步调协调的依一定规律出现，就需要有一个同步计数器。比如在一行扫描中，经过约40μs显示图象的时间之后，接着要有约24μs的行消隐时间，保证40个字符宽的显示区以外的

屏幕（左右两侧）为黑色，同时还保证当光点从右端快速返回到左端时，消隐掉回扫线。在这期间，先要有一个约 8μs 的行同步脉冲，再接着留下约 4μs的时间，以便放彩色同步信号。这一切周而复始出现。可见这四种信号都与电视行频一致。同理，我们可以分析出，场消隐信号与场同步信号与电视的场频一致。总之，这六种信号有一定的时间顺序，又各自有确定的持续时间，同步计数器对这六种信号在什么时候产生，持续多长时间，起控制与协调作用。

2.8.2 同步计数器的组成及工作原理

图2.8.2画出了同步计数器的接线图。它是由四块74LS161同步计数器组成，该片的逻辑功能参见附录A。

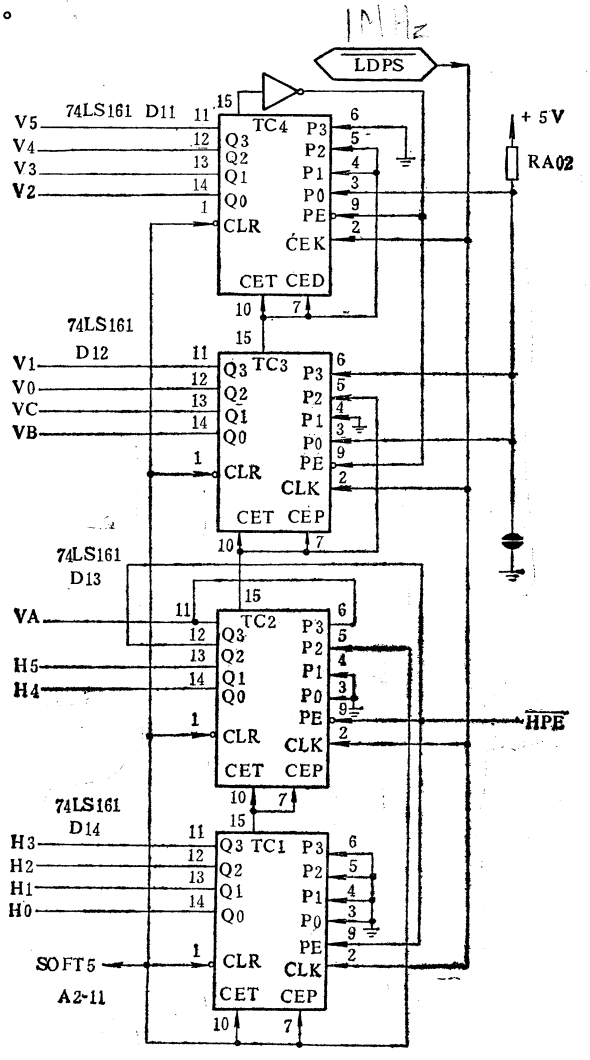


图 2.8.2 同步计数器

要看懂这张图，必须：一、记住PE、CET、CEP电位高低控制看电路的工作状态；二、记住所要产生的电视信号对该片提出的要求。

本节一开始我们就已经提到，同步计数器要提供三种同步信号，在这里我们将之具体化：

第一组：H5 H4 H3 H2 H1 H0，被称之为水平计数信号，约每 1μs计一个数（实际上是每 0.9777902μs计一个数），每一次循环出现65种状态，其中出现两次 全 0 状态，每一次循环的时间为63.56μs，恰为行扫描时间。

第二组：V4 V3 V2 V1 V0，被称之为垂直计数信号，每 8个行扫描时间计一个数，即每显示一行字符它计一个数。

第三组：VC VB VA，在显示字符时用确定字符点阵（每个 字 符 水平七个点，垂直八个点）的垂直位置。每隔一个行扫描时间计一个数，八个行扫描时间是这个三位计数器的循环周期。这部分的详细说明见2.9.3中的内容。

下一节我们将会看到，正是用了这三组共14个同步信号，经过一定的组合，去控制显示字符（或图形）所需的地址信号与同步信号。

在有了上述三组信号的一般概念之后，我们来着手分析如图2.8.2所示 的 硬件，从逻辑上是如何满足要求的。分析之前要记住：

1．当PE＝0，不管CEP和CET是0还是1，片子都工作在数据预置状态。这时在同步脉冲CLK上升沿触发下，将P3、P2、P1、P0置入Q3、Q2、Q1、Q0。

2．当PE＝1，CEP＝CET＝0时，保持原状态不变。

3．当PE＝1，CEP＝CET＝1时，片子处于同步计数工作状态，每来一个同步 脉冲CLK（上升沿触发），计一个数。

4．必须注意，从片子内部的逻辑电路看出，它的动态进位输出TC为1的条件，除了需要Q3·Q2·Q1·Q0＝1111外，还需使能端CET＝1。对TC来说，这五个信号 是“与”的关系：

$$TC=Q3\cdot Q2\cdot Q1\cdot Q0\cdot CET$$

5．以 $\overline{LDPS}$ 作为时钟脉冲（见2.2节）加到CLK端，它是约1MHz的脉冲。

将前三点归纳成下表，对于分析图2.8.2是有益的。

PE	0	1	1
CEP、CET	0/1	0	1
工作状态	预置	保持	计数

下面我们用列表的办法来分析图2.8.2，先分析D13和D14，参见表2.8.1。

我们假定初始状态两个片子的Q3Q2Q1Q0均为 0000，因 此 $\overline{HPE}=Q2_{D13}=0$注，从而使两 片 的PE＝ $\overline{HPE}$ 均为低电平。故当第一个时钟脉冲到来时，两块电路都工作在数据预置状态：

注，$Q2_{D13}$表示D13片的Q2端

表 2.8.1 D14与D13片的状态转换表

	D14 CEP＝CET＝CLR＝SOFT5＝1 P0＝P1＝P2＝P3＝0							D13 P2＝CLR＝SOFT5＝1 P0＝P1＝0									
计数脉冲	PE＝$\overline{HPE}$	计数脉冲来前工作状态	H3 ↑Q3	H2 ↑Q2	H1 ↑Q1	H0 ↑Q0	TC1 ↑TC	CET、CEP＝TCL1	PE＝$\overline{HPE}$	P3＝VA	工作状态	VA ↑Q3	$\overline{HPE}$ ↑Q2	H5 ↑Q1	H4 ↑Q0	TC2 ↑TC
			0	0	0	0	0					0	0	0	0	0
1	0	预置计数	0	0	0	0	0	0	0	0	预置	0	1	0	0	0
2	1		0	0	0	1	0	0	1	0	保持	0	0	0	0	0
3	1		0	0	1	0	0	0	1	0				⋮		
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
16	1		1	1	1	1	1	0	1	0		0	1	0	0	0
17	1		0	0	0	0	0	1	1	0		0	1	0	1	0
18	1		0	0	0	1	0	0	1	0	计数	0	1	0	1	0
19	1		0	0	1	0	0	0	1	0	保持	0	1	0	1	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
32	1		1	1	1	1	1	0	1	0		0	1	0	1	0
33	1		0	0	0	0	0	1	1	0	计数	0	1	1	0	0
34	1		0	0	0	1	0	0	1	0	保持	0	1	1	0	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
48	1		1	1	1	1	1	0	1	0				⋮		
49	1		0	0	0	0	0	1	1	0	计数	0	1	1	1	0
50	1		0	0	0	1	0	0	1	0	保持	0	1	1	1	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
64	1		1	1	1	1	1	0	1	0				⋮		
65	1		0	0	0	0	0	1	1	0	计数	1	0	0	0	0
66	0	预置	0	0	0	0	0	0	0	1	预置	1	1	0	0	0
67	1	计数	0	0	0	1	0	0	1	0	保持	1	1	0	0	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
82	1		0	0	0	0	0	1	1	0	计数	1	1	0	1	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
98	1		0	0	0	0	0	1	1	0	计数	1	1	1	0	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
114	1		0	0	0	0	0	1	1	0	计数	1	1	1	1	0
⋮	⋮				⋮			⋮	⋮	⋮				⋮		
129	1		1	1	1	1	1	0	1	0	保持	1	1	1	1	1
130	1		0	0	0	0	0	1	1	0	计数	0	0	0	0	0

D14的Q3 Q2 Q1 Q0预置为0000，D13的Q3 Q2 Q1 Q0预置为0100。这样，在第一个同步脉冲作用后，两个片子的PE就变成了高电平，从而使D14进入同步计数状态，使D13变作状态保持。

对于D14，从第2个同步脉冲到第65个同步脉冲，这段时间间隔为D14的基本周期，

表 2.8.2 D12与D11片的状态转换表

	D12 CLR＝SOFT5＝1，P1＝0 P0＝P3＝1									D11 CLR＝SOFT5＝1 P3＝0，P0＝1								
TC2 序号	CET、CEP＝TC2	PE＝$\overline{TC4}$	P2＝TC2	工作状态	V1 ↑Q3	V0 ↑Q2	VC ↑Q1	VB ↑Q0	TC3 ↑TC	CET、CEP＝TC3	PE＝$\overline{TC4}$	P1、P2＝TC3	工作状态	V5 ↑Q3	V4 ↑Q2	V3 ↑Q1	V2 ↑Q0	TC4 ↑TC
					0	0	0	0	0					0	0	0	0	0
1	1	1	1	计数	0	0	0	1	0	0	1	0	保持	0	0	0	0	0
2	1	1	1	计数	0	0	1	0	0	0	1	0		0	0	0	0	0
3	1	1	1	计数	0	0	1	1	0	0	1	0		0	0	0	0	0
4	1	1	1	计数	0	1	0	0	0	0	1	0		0	0	0	0	0
⋮	⋮	⋮	⋮	⋮			⋮			⋮	⋮	⋮				⋮		
⋮	⋮	⋮	⋮	⋮			⋮			⋮	⋮	⋮				⋮		
15	1	1	1	计数	1	1	1	1	0	⋮	⋮	⋮						
16*	0	1	0	保持	1	1	1	1	1					0	0	0	0	0
16	1	1	1	计数	0	0	0	0	0	1	1	1	计数	0	0	0	1	0
17	1	1	1	计数	0	0	0	1	0	0	1	0	保持	0	0	0	1	0
18	1	1	1	计数	0	0	1	0	0	0	1	0	保持	0	0	0	1	0
⋮		⋮											保持			⋮		
32	1	1	1	计数	0	0	0	0	0	1	1	1	计数	0	0	1	0	0
⋮													保持			⋮		
48	1	1	1	计数	0	0	0	0	0	1	1	1	计数	0	0	1	1	0
⋮													保持			⋮		
64	1	1	1	计数	0	0	0	0	0	1	1	1	计数	0	1	0	0	0
⋮													保持			⋮		
240	1	1	1	计数	0	0	0	0	0	1	1	1	计数	1	1	1	1	0
⋮													保持			⋮		
256*	0	1	0	保持	1	1	1	1	1	0	1	0	保持	1	1	1	1	1
256	1	0	1	预置	1	1	0	1	0	1	0	1	预置	0	1	1	1	0
257	1	1	1	计数	1	1	1	0	0	0	1	0	保持	0	1	1	1	0
258	1	1	1	计数	1	1	1	1	0	0	1	0	保持	0	1	1	1	0
259	1	1	1	计数	0	0	0	0	0	1	1	1	计数	1	0	0	0	0
260	1	1	1	计数	0	0	0	1	0	0	1	0	保持	1	0	0	0	0
⋮				计数			⋮									⋮		
387	1	1	1	计数	1	1	1	1	0	1	1	1	计数	1	1	1	1	0
388*	0	1	0	保持	1	1	1	1	1	0	1	0	保持	1	1	1	1	1
388	1	1	1	预置	1	1	0	1	0	1	0	1	预置	0	1	1	1	0

约63.56μs。从第66个同步脉冲到第130同步脉冲，又构成第二个基本周期，也是63.56μs，依此类推。

对于D13，从第2个时钟脉冲到第16个时钟脉冲为数据保持状态。在第16个时钟脉冲作用过后，D14片计满1111，同时进位位TC1＝1，这就给D13做好了准备计数的条件，

即CET＝CEP＝TC1＝1，PE＝1。当第17个时钟脉冲到来时，计一个数，Q3 Q2 Q1 Q0从0100变到0101。第17个脉冲过后，由于D14的进位位TC1回零了，使得D13片又进入数据保持状态。依此类推，在第33、49、65个时钟脉冲作用时，为计数状态。到第65个时钟脉冲作用过后，由于这时Q3 Q2 Q1 Q0＝1000，$\overline{HPE}$＝Q2＝0，使得D13与D14片做好了数据预置的准备。当第66个时钟脉冲到来时，D13片预置成1100。这里必须指出的是，到第114个时钟脉冲到来时，D13片计数成1111，但是并没有进位，即TC2＝0，而不是等于1。原因就是前面所说的TC2为5个量相"与"的结果，其中有一个量CET＝TC1＝0，故TC2＝0。这件事可以这样解释：在第114个时钟脉冲作用期间，D14与D13同时在计数，只要D14一计数，就会使TC1从1变为0，当D13计完数，Q3 Q2 Q1 Q0变为1111时，TC1却为0了。从这里可以得出结论：D13在计数状态时，是不可能使本身的进位位TC2为1的。从表2.8.1可以看出，只有在第129个时钟脉冲到来时，片子处于数据保持工作状态下，才使进位位TC2＝1。其原因就是：这时的D14片计满了1111，同时它的进位位TC2＝1。从这里还可以看出，TC2为1出现在VA（即Q3）由1变0的前一拍时间。这一点很重要，这是分析D12与D11片的基础。从表8.2.1可见，V_A每63.56μs变一次号，周期为两个行扫描时间，故可知每127.12μs就会出现一次TC2＝1。

在分析清楚D13、D14片工作原理之后，我们结合表2.8.2来分析D11、D12片的工作。由于这两个片子的状态变化频率比较低，我们就不再用时钟脉冲的序号作为造表的顺序了，而是用前一片的进位TC2的序号作为表上第一列的顺序。换言之，我们不是去看每隔1μs时间Q3 Q2 Q1 Q0如何变化，而是看每隔127.12μs时间，它是如何变化的。

同样，我们假定开机之后两个片子的原始状态Q3 Q2 Q1 Q0为0000，$\overline{TC4}$＝1，在TC2为1时，D12工作在计数状态，在时钟脉冲作用下，Q3 Q2 Q1 Q0由0000变至0001。在这以后的两个行扫描间隔时间内（＜127.12μs），由于前一片D13的进位TC2＝0，因而D12工作在数据保持状态。到第二次TC2为1时，D12再计一次数，只有一拍时间，接着又是长时间的数据保持状态。到了第15次TC2为1时，计满1111，但是这时的进位位TC3不为1，这是因为在计数的同时TC2回零。只有在该片作第16次计数之前（在表中我们用符号16*来表示，它距第16次计数只差1μs时间）、D13、D12两片同时工作在数据保持状态，而TC2＝1，使TC3＝1，从而为下一片D11计数做好了准备。下一个时钟脉冲一到D11就从0000变为0001。以后每隔16×127.12μs，D11计1个数。我们把它称作每隔16个双倍行扫时间D11计一个数。当到了第256个双倍行扫时间的前一拍时（差约1μs），D11片的Q3 Q2 Q1 Q0TC4为11111，而D11、D12两片的PE＝$\overline{TC4}$＝0，这时是数据预置工作状态。当下一个时钟脉冲到来时，D12被预置成1101，D11被预置成0111。从这以后电路才进入到有效循环。之后，又在这基础上计数，计到两片均达1111后，又预置成1101和0111。如此循环下去。循环周期是131×127.12μs，或者说是131个双倍行扫时间。

最后，我们把四片连起来一起看，得到三组计数信号：

第一组，水平计数信号H5 H4 H3 H2 H1 H0每1拍（约1μs）计一个数，循环周期为65拍，约63.56μs，即行扫描时间。

第二组：垂直计数信号V5 V4 V3 V2 V1 V0每8个行扫描时间计一个数，循环周期为131个双倍行扫时间，恰为电视每一场的时间。

第三组：字符点阵垂直位置计数信号VC、VB、VA每一行扫时间计一个数，循环周期是8个行扫时间。

2.8.3 计数信号与显示屏幕的对应关系

为使读者得到直观的概念，我们画出了图2.8.3。在这张图中屏幕的位置与计数信号有一一对应关系。图中的大框表示整个屏幕，里面的小框表示字符（或图形）显示区，共可显示24行×40列字符。某行某列的字符在什么时候出现在屏幕上，是由水平计数信号H5 H4 H3 H2 H1 H0与垂直计数信号V4 V3 V2 V1 V0来决定的。比如0行0列字符是在H5 H4 H3 H2 H1 H0＝011000与V4 V3 V2 V1 V0＝00000时刻出现的。

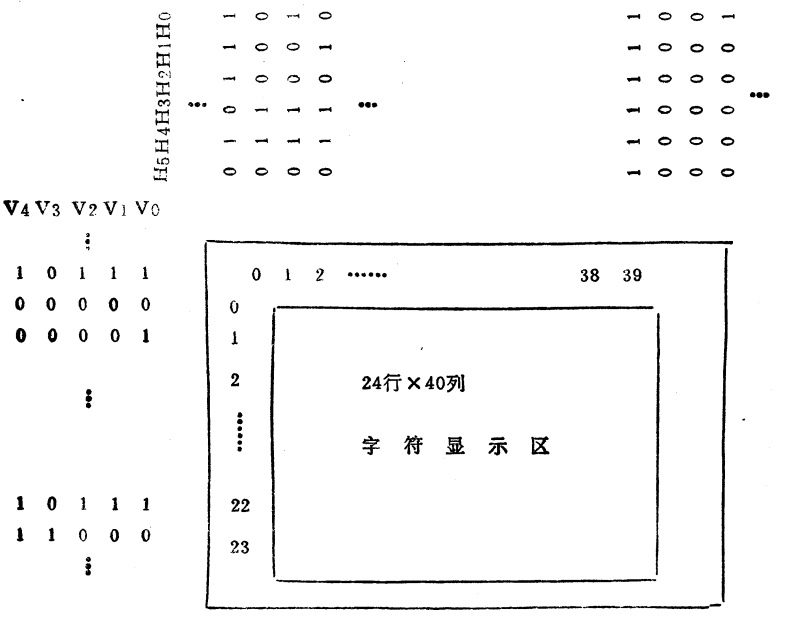


图 2.8.3 屏幕位置与计数信号的对应关系

2.9 视频信号发生器

有了前一节同步计数器的基本知识之后，我们着手研究为了进行显示所必需的全电视信号，接着还要分析八种不同的显示方式在电路上是怎样实现的。我们假定读者不太熟悉电视机的原理，特别是不熟悉彩色电视机的原理。因此我们需要补充一些这方面的基本知识，否则，要想弄清这部分的工作原理是相当困难的。

视频信号发生器的硬件线路画在图2.9.20上。下面我们就根据视频信号本身的分类一一加以分析。

2.9.1 同步信号与消隐信号

同步信号包括行同步信号、场同步信号和色同步信号，消隐信号包括行消隐信号与场消隐信号。在分析这部分电路时，为便于读者掌握，我们想用“倒”过来的方式，先将

设计者的思想告诉读者，反过来去验证电路是否能满足设计思想。

先看水平计数信号，我们依次列出了六个变量的取值表（见表2.9.1），在表的右侧注上信号的用途。这张表就是设计思想。

表 2.9.1 水平计数信号六个变量的取值表

H5	H4	H3	H2	H1	H0	用途
0	0	0	0	0	0	
0	0	0	0	0	0	
0	0	0	0	0	1	（9拍）
		⋮				
0	0	0	1	1	1	用作水平消隐信号出现时间（共25拍）
0	0	1	0	0	0	用作行同步信号出现时间（8拍）
0	0	1	0	0	1	
		⋮				
0	0	1	1	1	1	
0	1	0	0	0	0	用作彩色同步信号出现时间（4拍）
0	1	0	0	0	1	
0	1	0	0	1	0	
0	1	0	0	1	1	
0	1	0	1	0	0	（4拍）
		⋮				
0	1	0	1	1	1	
0	1	1	0	0	0	用作屏幕上一行40个字符出现的时间（40拍）
0	1	1	0	0	1	
		⋮				
1	1	1	1	1	1	

下面我们就从这张表出发，导出同步信号与消隐信号的逻辑关系。

1．水平消隐信号

我们将水平消隐信号所涉及到的那部分变量取值表分作两部分：高三位和低三位。低三位H2 H1 H0的所有8种可能的取值组合都出现，故从逻辑上讲，水平消隐信号与之无关。再看高三位，H5必须为0，而H3和H4不能同时为1。因此，可以推断出行消隐信号（HBL）的逻辑表达式为

$$HBL=\overline{H5}\cdot\overline{H4\cdot H3}$$
$$=\overline{H5+H4\cdot H3}$$

用简单的“与”、“或”、“非”门就能实现这个表达式。所说明的逻辑过程有了这个表达式，我们再到电路图上去找它的线路就易如反掌了。这个电路由74LS51-C13实现，画在图2.9.1中。

从表2.9.1可以看出，这个HBL信号要维持25拍时间（约24μs），它的作用是控制电视屏幕，使之在这25拍时间里保持黑色。

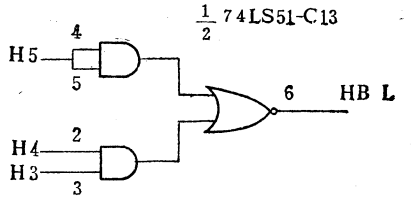


图 2.9.1 行消隐信号的逻辑电路

2．行同步信号

仍依上法，对于行同步信号取值表的低三位的8种组合都出现，故与之无关。高三位需H5 H4为0，H3为1。因此有

$$\text{行同步信号}=H3\cdot\overline{H4}\cdot\overline{H5}$$

为了节省器件，可以利用已经得到的HBL信号同H3进行逻辑“乘”，就可以得到行同步信号。证明如下：

$$\begin{aligned}HBL\cdot H3&=(\overline{H5}\ \overline{H4H3})H3\\&=(\overline{H5}\ \overline{H4}+\overline{H5}\ \overline{H3})H3\\&=H3\ \overline{H4}\ \ \overline{H5}\end{aligned}$$

其实，HBL同H3相“与”的这一关系，可以从六位取值表中直接看出。因为行同步信号的取值表包含在水平消隐信号的取值表中，它占消隐时间的一部分，它的突出标志是H3为1，或者说在行消隐期间又符合H3为1的那段时间，就是行同步脉冲持续的时间。产生这个同步信号的具体电路，将在分析场同步信号后一并给出。

3．彩色同步信号（COLOR BURST）

在上述的六变量取值表中，有4拍被用作彩色同步信号出现的时间，我们称之为T_{COLOR}，由于这段时间包含在水平消隐信号出现的时间之内，其特殊处为H4＝1，H3＝0，H2＝1，H5＝0。故可以用下述的逻辑乘式得出

$$T_{COLOR}=H4\ \overline{H3}\ \ \overline{H2}\ \ \overline{H5}$$

这个式子尚可化简为

$$T_{COLOR}=HBL\cdot H4\overline{H2}$$

这只要将$\overline{H5}\cdot\overline{H4H3}$代入到HBL中去，即可看出。读者对此可自己验证。用这个信号去控制彩色副载波（COLOR REF）通过门的时间，这是一个“与”的关系。门的输出就是彩色同步信号（COLOR BURST）。

$$\text{COLOR BURST}=T_{COLOR}\cdot\overline{COLORREF}$$

$$=HBL\cdot H4\cdot\overline{H2}\cdot\overline{\text{COLOR REF}}$$

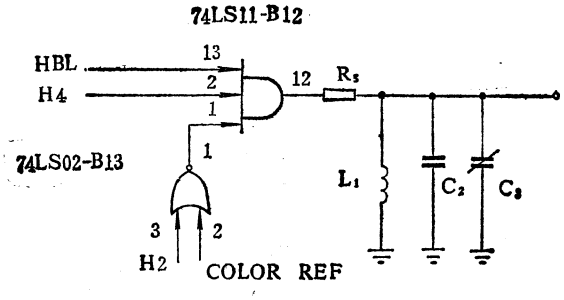


图 2.9.2 形成彩色同步信号电路

根据这个表达式，我们就可以找出实现它的线路，如图2.9.2所示。

在图2.9.2上，COLOR REF信号来自时基电路（参看图2.2.2），它是3.58MHz方波，经过上述电路之后，这个方波从74LS11-B12的12脚输出时倒相180°，后面我们将会讲到，这是彩色同步信号所要求的。前面我们讲过，彩色同步信号出现的时间是4拍，约4μs，可导出方波将有14个（3.5×4）。彩色电

视要求彩色同步信号应该是正弦波，因此在线路上加了由R_5、L_1、L_2和C_3组成的滤波电路，目的是滤去方波中的高频成分，使之（尽可能）成为正弦波。C_3是一个5～50pF的半可调电容，它可以改变彩色同步信号的相位，从而改变输出的彩色图形的色调。详细原理将在后面介绍。

4．场消隐信号

我们采用上面的分析思路，首先列出垂直计数信号五个变量的取值表（表2.9.2），表上的一拍时间为8个行扫描时间。

表 2.9.2 垂直计数信号五个变量的取值

V4	V3	V2	V1	V0	用途
0	0	0	0	0	用作屏幕上出现24行字符的时间（24拍）
0	0	0	0	1	
		⋮			
1	0	1	1	1	
1	1	0	0	0	用作场消隐信号出现时间（共8拍）
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	用作场同步信号出现时间(1拍)
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

分析出现场消隐信号的取值，发现V2、V1、V0的所有8种组合均出现，故逻辑表达式与之无关，而V4 V3同时为1是这8拍的共同特征。

故　　场消隐信号＝V4 V3

5．保持黑信号

这个信号是行消隐信号与场消隐信号的叠加，作用是：无论在行回扫期间还是在场回扫期间，都使屏幕保持黑色，同时还使屏幕中央的24行×40列字符宽度之外的区域保持黑色。具体线路如图2.9.3。

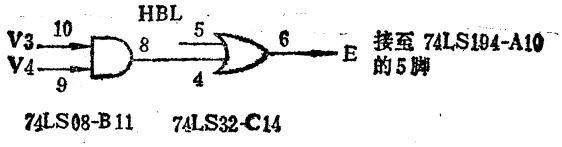


图 2.9.3 形成保持黑信号

保持黑信号＝HBL＋V4 V3

这个保持黑信号接至74LS194-A10的5脚（见图2.9.20），用于区别有无屏幕显示。

6．场同步信号

从表2.9.2可以看出，V4 V3 V2 V1 V0＝11100是出现场同步信号的一拍时间，为8个行扫时间，在一拍时间里要求有如图2.9.4那样的场同步信号波形。分析这一波形，其左半边为占空比3∶1的脉冲。据表2.8.1可知，H4为每16μs（约）计一个数，H5为每32μs(约)计一个数。如果使两者"或"起来，就可以得到上述信号的左半部波形，而要得到整个波形，还需"与"上$\overline{VC}$。因为$\overline{VC}$出现的时间是4×64μs，在图2.9.4左半边是$\overline{VC}$＝1，右半边$\overline{VC}$＝0，在左半边时间里让H4＋H5信号通过，在右半边时间里不让H4＋H5通过，这就是"与"的逻辑关系。整个这一段波形又要在V4 V3 V2 V1 V0＝11100这一拍时间内出现（8个行扫描时间），故得场同步信号的表达式为

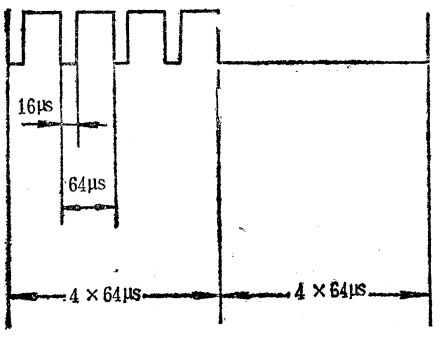


图 2.9.4 场同步信号波形

场同步信号＝$\overline{V0}$ $\overline{V1}$ V2 V3 V4 $\overline{VC}$ (H4＋H5)

图2.9.5画出了与这个表达式相应的线路，图中A点就是这个表达式的内容。注意：在看

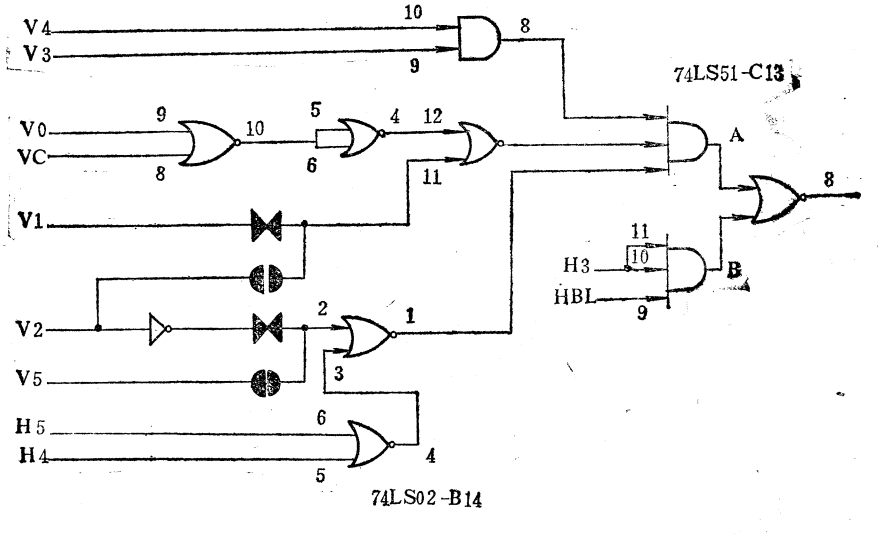


图 2.9.5 行、场同步信号线路

这张图时，⋈是跨接线连上的符号；◑是跨接线断开的符号。因为我们现在分析的是在场频为60Hz的情况，如果场频为50Hz，则反过来，让◑号处的跨接线连通，让⋈号处的跨接线断开，当然，相应的表达式也要略有改变，为

场同步信号（50Hz场频）＝$\overline{V0}$ $\overline{V2}$ V3 V4 $\overline{V5}$ $\overline{VC}$(H4＋H5)

从电视角度要求行同步信号与场同步信号合起来，倒个相输出。图2.9.5上的A点为场同步信号，B点为行同步信号。两者经过“或非”门，从74LS 51-C13的 8 脚 输 出，称之为SYINC（同步）信号。

$$SYINC=\overline{\overline{V0}\ \overline{V1}\ V2\ V3\ V4\ \overline{VC}(H4+H5)+H3HBL}$$

在图2.9.6中，我们画出了在场消隐期间同步信号SYINC的波形。从 图 中可见，在

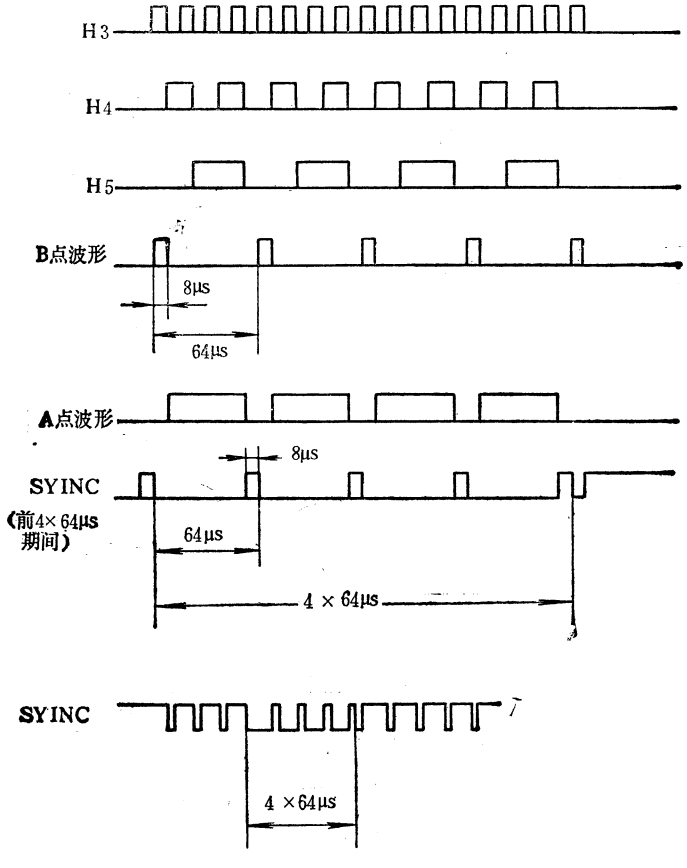


图 2.9.6 在场消隐期间同步信号SYINC波形

4×64μs时间内，有4个脉宽为56μs的负脉冲作为场同步用。而在这4个 脉冲之前和之后有一连串的脉宽为8μs的负脉冲作行同步用，以保证在场逆程期间行同步不 受 影 响。由图可见行、场同步脉冲的宽度是不一样的，相差 7 倍之多。在电视接收机中，正是利用了这个差别，采用不同时间常数的简单的RC微分或积分电路，将它们分离开来，分 别 去控制扫描电路的行频或场频，从而实现同步的目的。

2.9.2 产生彩色图象信号

如果我们想让电视屏幕显示如图2.9.7的彩条图案，Apple是怎样给出这一图案的？它的全电视信号应该是什么样的？它的软件怎样与之配合？它的硬件又是怎样产生这一信号的？要回答这一系列问题，必须首先普及一下彩色电视中有关彩色信号的知识。先

看看彩色电视对显示彩条图案有何要求，再来分析Apple是怎样通过硬件和软件来满足这一要求的。

1．美国NTSC制彩色编码方式

现代电视均以三个基色作为组成各种彩色的基础。这三个基色是：

R——红色

G——绿色

B——蓝色

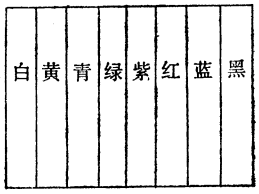


图 2.9.7 电视屏幕上的彩条信号

NTSC制规定三个基色的亮度系数为：

$L_R=0.299$

$L_G=0.587$

$L_B=0.114$。

将三个亮度系数小数点后第三位四舍五入之后，得出总的亮度信号Y与三个基色信号大小间的关系：

$$Y=0.30R+0.59G+0.11B$$

如$U_R=U_G=U_B=1$（可视作三个基色均取单位电压），则亮度$U_Y=0.30+0.59+0.11=1$，这时显示白色。如$U_R=U_G=1$，$U_B=0$，则$U_Y=0.30+0.59+0=0.89$，这时显示黄色，也就是说红色和绿色可以混合出黄色。比较白色与黄色，他们的亮度不同，如果是显示在黑白电视机上，表现出不同的灰度。我们知道，要显示出彩色来，光有亮度信号Y是不够的，还要加上两个能反映颜色的色差信号：

$$U_{R-Y}=U_R-U_Y$$

$$U_{B-Y}=U_B-U_Y$$

有人会问，为什么不直接用U_R、U_G和U_B呢？这是为了满足彩色电视与黑白电视兼容的需要，同时也有关于信号频带方面问题的考虑。

我们设想一条线要同时传送这三个未加处理过的信号，如果将三者叠加起来，送到接收端，此时再想把它们分开则是怎么也办不到的。在NTSC制中采用正交调制的方法，用一个3.58MHz的副载波去调制两个色差信号。为了能够使调制后的色差信号不致相互混淆，一个用正弦信号调制，一个用余弦信号调制。框图画在2.9.8上。调制后的色度信号C为这两个正交色度分量的和：

$$C=(B-Y)\sin\omega_{sc}t+(R-Y)\cos\omega_{sc}t$$

其中$\omega_{SC}=2\pi f_{SC}$，f_{SC}——副载波频率。在NTSC制式下选$f_{SC}=3.58MHz$。

我们用矢量来表示上述色度信号及其两个正交分量，如图2.9.9。在图中色度信号矢量为它的两个正交分量的矢量和。这个矢量可以用振幅和相位来描述，如果知道两个分量的大小，就可以求出振幅和相位：

$$F_c=\sqrt{(R-Y)^2+(B-Y)^2}$$

$$\phi=\operatorname{arctg}\frac{R-Y}{B-Y}$$

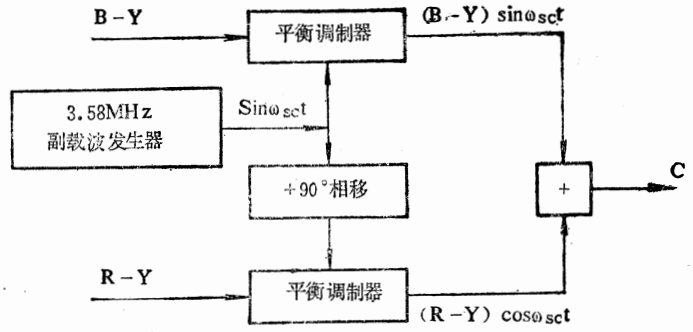


图 2.9.8 正交调制形成彩色信号框图

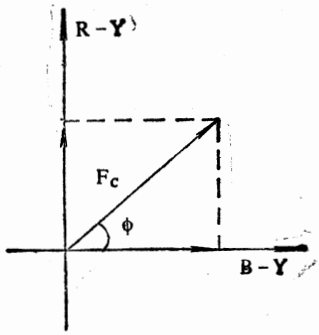


图 2.9.9 色度信号矢量图

不同色调不同深浅的彩色，它们的相位和振幅不同。相位ϕ反映色差信号的比例，代表彩色的色调；而振幅F_C由色差信号的大小所决定，反映彩色的饱和度。我们举一个日常生活中的例子来说明什么叫彩色饱和度。比如一件蓝色外衣，刚买来时湛蓝湛蓝的，我们称它饱和度深。下水之后，经烈日爆晒，退了色，变白了些，我们说它饱和度浅了。如果全白了，饱和度为零。因此饱和度的深浅反映了彩色光中所含有白光的多少。所含的白光越多，则颜色越浅。综上所述，我们可以这样说：F_C的大小只影响颜色的深浅，而ϕ的大小却决定着色调，对分析是何种彩色，后者更为重要。

因为色度信号$C=F_C\sin(\omega_{sc}t+\phi)$要与亮度信号Y加起来一起传送，这就要求适当压缩$F_C$，否则会使总的信号的动态范围大大超过亮度信号的黑色电平和白色电平，影响黑白电视接收机的收看效果。一般规定黑色电平为0，白色电平为1时，彩色信号（亮度信号＋色度信号）Y＋C的动态范围限制在＋1.33至－0.33范围。这样做兼顾了黑白电视和彩色电视的收看效果，使两者均不会产生明显失真。这就需要将色差信号B－Y和R－Y分别乘上两个小于1的因子a和b：

$$a=0.493$$

$$b=0.877$$

于是，色度信号的振幅和相位就相应改为：

$$Fc=\sqrt{[a(B-Y)]^2+[b(R-Y)]^2}$$

$$\phi=\mathrm{arctg}\frac{b(R-Y)}{a(B-Y)}$$

根据上式，我们把如图2.9.7的彩条的亮度信号和色度信号计算列于表2.9.3。

由表2.9.3可以看出：

（1）八个彩条可以由不同的基色信号R、G、B配出。比如R=G=1，B=0，可配出黄色，R=B=1，G=0，可配出紫色，等等。

（2）八个彩条各有不同的亮度Y。如果用黑白电视机来显示这八个彩条的话，则

表 2.9.3 彩条的亮度信号与色度信号

	白	黄	青	绿	紫	红	蓝	黑
R	1	1	0	0	1	1	0	0
G	1	1	1	1	0	0	0	0
B	1	0	1	0	1	0	1	0
Y	1	0.89	0.70	0.59	0.41	0.30	0.11	0
B—Y	0	−0.89	0.30	−0.59	0.59	−0.30	0.89	0
R—Y	0	0.11	−0.70	−0.59	0.59	0.70	−0.11	0
a(B—Y)	0	−0.4388	0.1479	−0.2909	0.2909	−0.1476	0.4388	0
b(R—Y)	0	0.0965	−0.6139	−0.5174	0.5174	0.6139	−0.0965	0
Fc	0	0.44	0.63	0.59	0.59	0.63	0.44	0
ϕ	—	167°	283°	241°	61°	103°	347°	—
Fc 与色同步信号幅度比	0	2.20	3.15	2.95	2.95	3.15	2.20	0
相对色同步信号相位	—	−13°	103°	61°	−119°	−77°	167°	—

有八个不同的灰度，依次从最亮（白色）到最暗（黑色）。

（3）这些彩条是依表上所列R、G、B比例配出的，都是百分之百的饱和度。在这种情况下，每个彩条有如表中所列的色度信号的幅值F_c和相角ϕ。彩条不同相角ϕ不同。

在图 2.9.10中，我们画出了这八种彩色的色度信号的矢量图。图中矢量长短等于

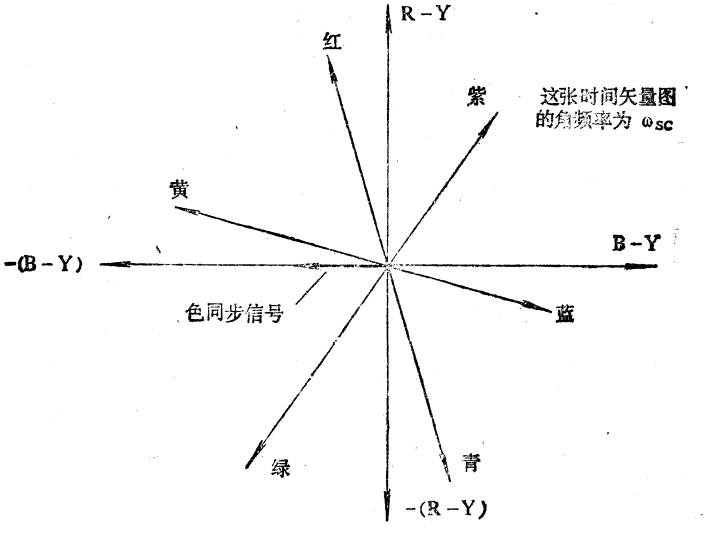


图 2.9.10 彩条色度信号矢量图

F_c值，而相角ϕ是从B－Y轴算起的。比如黄色$\phi=167°$，在第二象限。白色无所谓相角，因为它的色度信号为零，故坐标原点代表白色。白色只有亮度信号Y。黑色的亮度信号与色度信号均为零。在图中，我们还标出了色同步信号的矢量。按照NTSC制规定，色

同步信号矢量长度为0.2，相位为180°。实际上，这一色同步信号矢量在接收端（即彩色电视机中）是作为一把“尺子”，用相对于它的相位来决定接收到的彩色信号的色调，用相对于它的幅度比来决定彩色的饱和度（深浅）。前面已经说过，这个色同步信号出现在行消隐期间，紧跟在行同步信号的后面，频率也是3.58MHz。

下面我们来看一下如果整幅图象是黄色，它的彩色电视信号在一行里是什么样的。按照表 2.9.3中所列的值，黄色（饱和度为100%的）的亮度信度Y＝0.89，色度信号的幅值与色同步信号的幅度比为2.20，色度信号的相位（相对色同步信号）为－13°。彩色信号应为亮度信号与色度信号的叠加。在图 2.9.11（a）中我们画出了一行中的彩色信号，在图 2.9.11（b）中，画出了将时间轴拉长后这一信号的波形，它与色同步信号画在一起以资比较。从图可见，彩色信号是一个直流分量与一个3.58MHz的正弦波的叠加，这一正弦波落后于色同步信号的相位13°（黄色）。

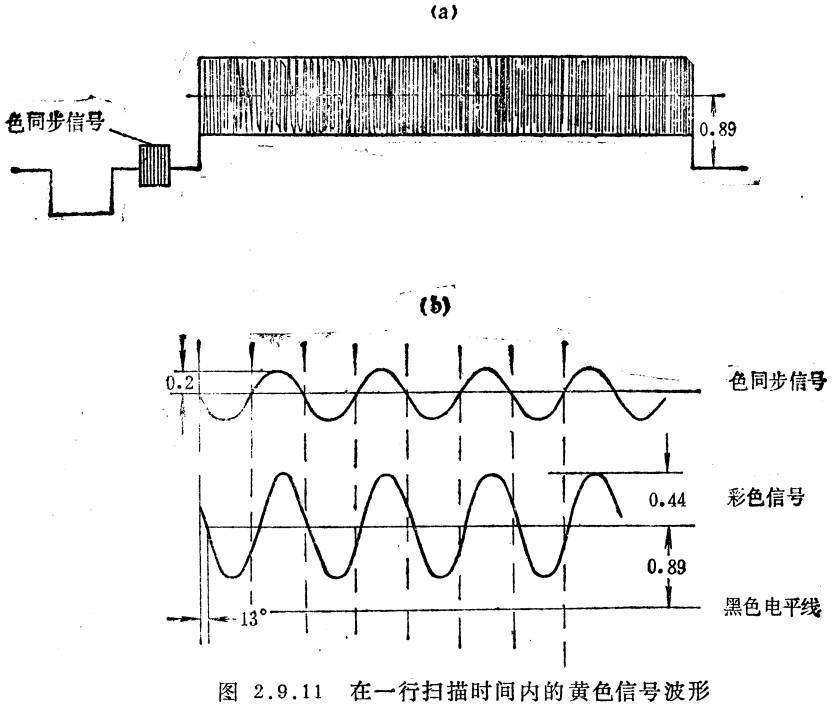


图 2.9.11　在一行扫描时间内的黄色信号波形

我们知道，彩色电视的发送设备是将色同步信号和彩色信号一起传送的。电视机接收到这两个信号后，将按照图 2.9.12所示的框图解出R、G、B信号。这里必须声明，实际的NTSC制式从人的视觉以及传送信号的频带方面考虑，还要将色度信号C分解为I、Q分量，真正传送的是Y、I、Q三个量。我们这里简化了这一步骤，而这并不影响我们对这一问题的理解。或者说，我们用图 2.9.12这一简化的框图，对于搞清APPLE是如何产生彩色信号、以及接收机又是如何解调出R、G、B信号这样的问题，已经足够了。下面我们就从解调过程示意框图出发，来讲解出R、G、B信号的过程。简述如下：彩色信号经带阻滤波器后，滤去3.58MHz的色度信号，其输出就是亮度信号Y。考虑到

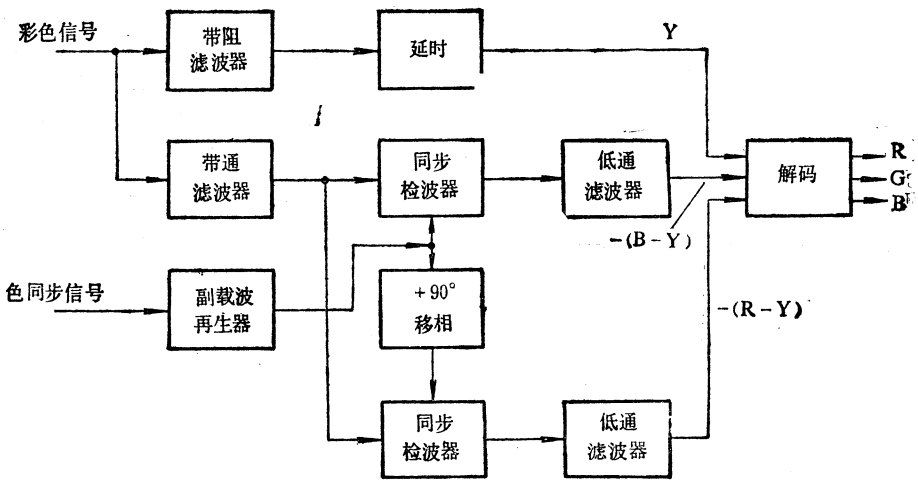


图 2.9.12 电视接收机解调过程示意框图

Y要与下面解出的色差信号B－Y和R－Y同时到达解码网络，故加入一定的延时电路，此为其一。其二，彩色信号经过带通滤波器，去掉亮度信号，保留下了色度信号，再分送到两个同步检波器上，而两个同步检波器的副载波又是被色同步信号所激励。一个与色同步信号同相，另一个比色同步信号超前90°相位。这样，就可以解出两个色差信号，经低通滤波之后，再与亮度信号一起，通过解码网络，还原出R、G、B信号。

为了使读者对彩色信号有直观印象，我们在图2.9.13中给出了八种彩条在一行扫描中的亮度信号与色度信号，并给出了拉长时间轴放大后的波形。

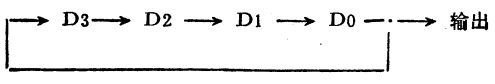
2．Apple微处理机所产生的彩色信号

我们已经讲了一般NTSC制式发送和接收彩色图象的原理。下面我们要研究Apple微处理机所产生的彩色信号。大家知道，在Apple中的这部分电路用的不是模拟电子电路，而是数字电路，它所产生的彩色信号不是正弦波而是方波。在图2.9.14上画出了Apple所能产生的16种方波波形。下面我们就来分析这些方波是怎样产生的，以及怎样用方波来表示不同的色调。

在Apple中，表示彩色的方波信号是用非常巧妙的办法形成的，其核心器件是一个四位移位寄存器。我们知道，四位数码D3 D2 D1 D0可以有16种组态，每一种组态被可以用来代表一种彩色。比如0001代表洋红，0010代表深蓝，等等。这种用四位代码形成表示彩色的方波的过程可以分为两步：

第一步，先将D3 D2 D1 D0打入移位寄存器；

第二步，在14MHz时钟脉冲作用下，让移位寄存器作循环移位输出，方向如下图：



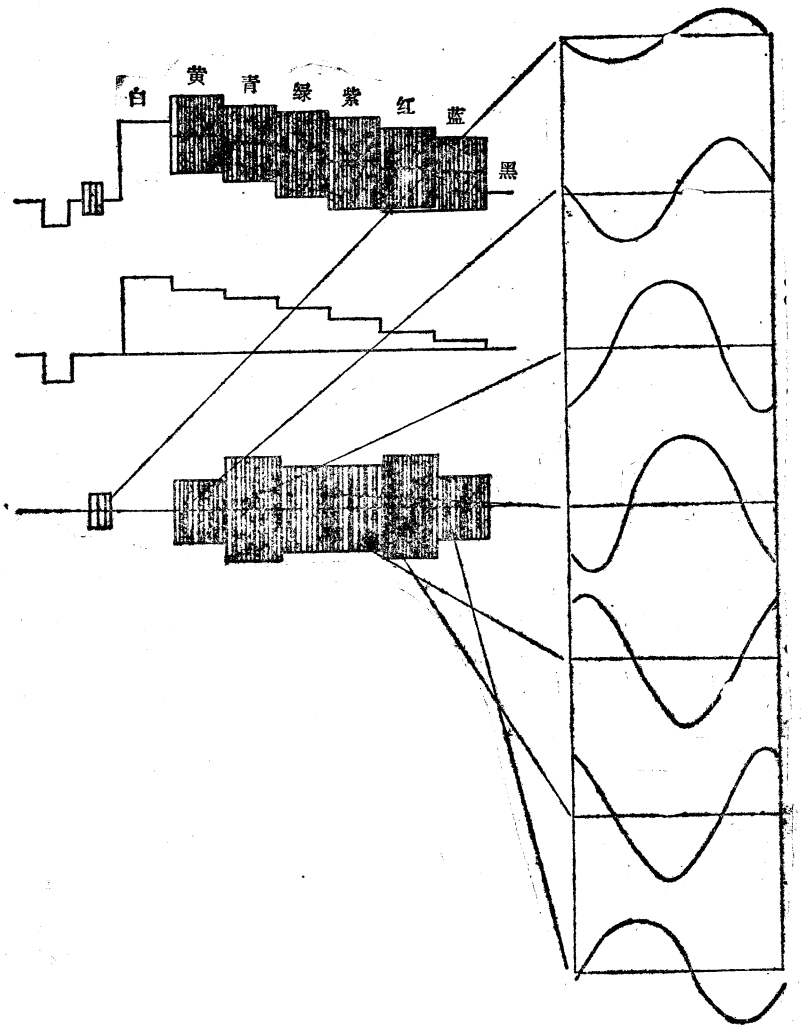


图 2.9.13 在一行扫描中八种彩条的波形

这时循环移位输出的波形就是我们所要的方波波形。

比如，洋红色的编码D3D2D1D0＝0001，经循环移位得到如图2.9.15的波形。在图的左半部分，列出了依时间变化的D3D2D1D0循环移位值，相邻两组取值的时间间隔为1/14μs；因为要将D0这一位拿去输出，故只要依时间顺序将D0的取值画出来，就可得到图的右半部分所表示的方波，其频率恰为3.5MHz。

读者可能会提出：前面讲过，要出现彩色需要有亮度信号Y和3.5MHz的正弦载波的色度信号。可是这里的洋红色只有方波信号，能出彩色吗？要解答这个问题，就要回顾一下图2.9.12中的彩色电视接收机示意解调过程框图。在这个框图中，输入的彩色信号先要通过带阻滤波器和带通滤波器。Apple输出的3.5MHz的方波经过带阻滤波器后只剩下平均值（直流分量），就是亮度信号Y。同样的方波经带通滤波器，将高次谐波分

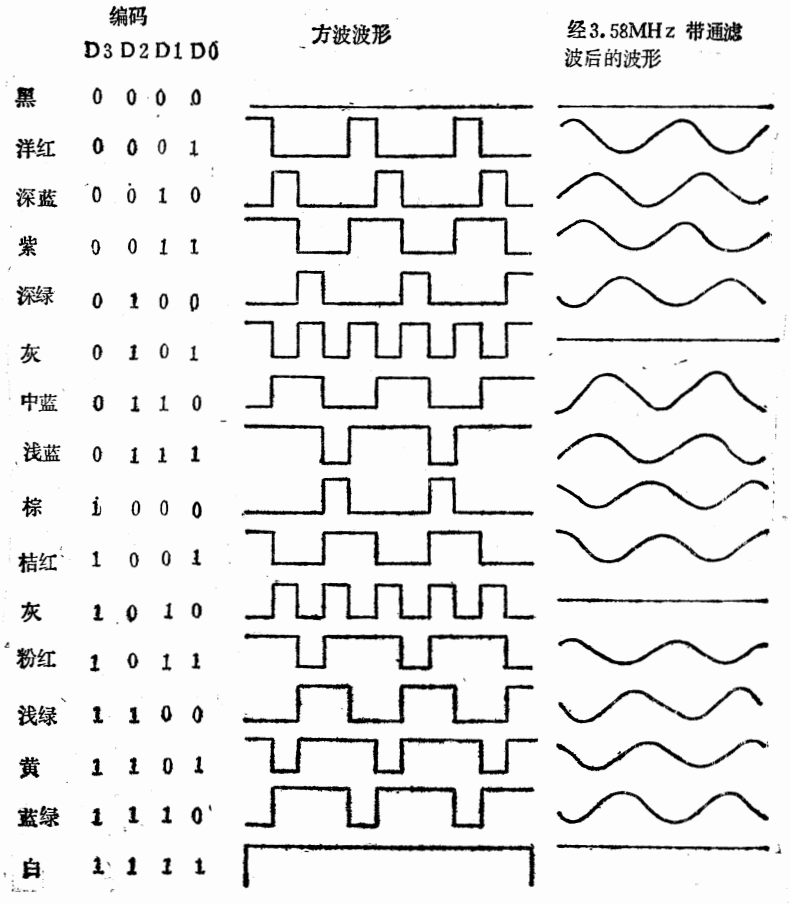


图 2.9.14 Apple产生的表示彩色的方波波形

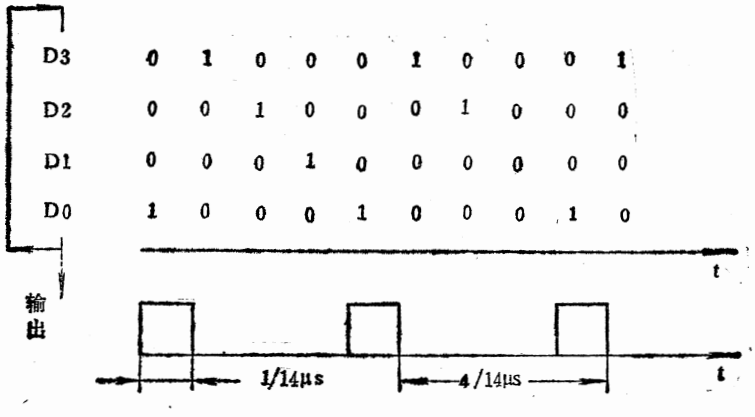


图 2.9.15 四位数码经循环移位得到方波波形

量滤除，输出的是3.5MHz的正弦波，就是色度信号，它有着确定的幅值和相位。从原理上讲，方波被分解为亮度信号和色度信号的方法，就是用付里叶分析，取平均值和基波。在图 2.9.16上，画出了一个方波被分解之后取平均值和基波的例子。在图 2.9.14中画出了十六种彩色依其编码所得方波，还画出了与这些方波所对应的正弦波。需要说明的是，对于编码为0101或1010的方波，频率不是3.5MHz，而是7MHz，故经带通滤波器之后，全部被滤除。而两种方波经带阻滤波之后，有直流分量，等于0.5，故呈灰色。

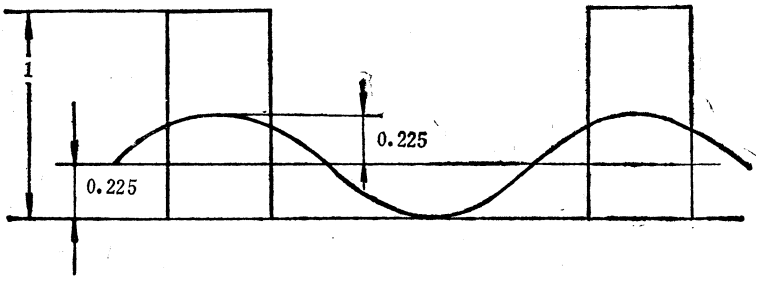


图 2.9.16 分解方波取平均值和基波

在表 2.9.4中，列出了这十六种方波信号经分解后得到的亮度信号Y，色度信号的幅值F_c和色度信号的相角ϕ。这个相角是相对色同步信号画出的。因为在Apple中是可以用半可变电容C_3来调色同步信号的相角的（见图 2.9.2），我们在这里就假定调到深绿色的ϕ角领先这个色同步信号60°，其它色度信号据此定位。从表中看出：

表 2.9.4 APPle中16种彩色相应的亮度信号与色度信号

编码 D3D2D1D0	显示彩色	亮度信号 Y	色度信号幅值Fc	色度信号相角 ϕ
0000	黑	0	0	—
0001	洋红	0.25	0.225	−120°
0010	深蓝	0.25	0.225	150°
0011	紫	0.50	0.318	−165°
0100	深绿	0.25	0.225	60°
0101	灰	0.50	0	—
0110	中蓝	0.50	0.318	105°
0111	浅兰	0.75	0.225	150°
1000	棕	0.25	0.225	−30°
1001	桔红	0.50	0.318	−75°
1010	灰	0.50	0	—
1011	粉红	0.75	0.225	−120°
1100	浅绿	0.50	0.318	15°
1101	黄	0.75	0.225	−30°
1110	蓝绿	0.75	0.225	60°
1111	白	1	0	—

（1）彩色都不是标准的100%饱和度，因为这些信号是数字电路产生的，在一定意义上来说幅度不可控，相角也不可能变化很"细"（最小变化45°角）。因此，从屏幕上看到的彩色都不太标准。

（2）比较洋红和粉红，两者的相角φ均为－120°，色度信号的幅值也一样，为0.225，唯一的差别是，前者亮度信号为0.25，而后者高达0.75，可见亮度信号不同则颜色不同。

（3）编码为0101和1010的两种颜色均为灰色，都只有亮度信号（Y＝0.5），而无色度信号。

据表2.9.4画出了色度信号的矢量图（图2.9.17）。在图上的同一个矢量有可能代表两种彩色，其差别是亮度信号不同。我们将亮度信号Y注在矢量边上，以资区分。

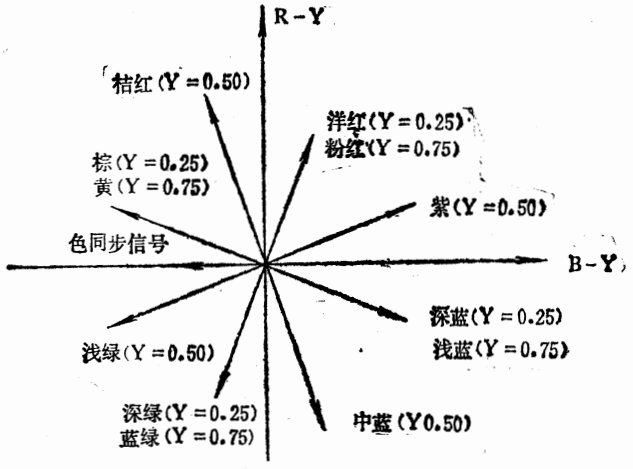


图 2.9.17　Apple在低分辨率情况下产生的彩色色度信号矢量图

3．全电视信号输出

全电视信号是图象信号（字符或彩色图形）、同步信号SYNC和副载波信号的叠加。叠加是在一只晶体三极管上进行的。接成如图2.9.18所示的求和电路，信号由基极输入，全电视信号由射极输出。依图可写出如下方程式：

$$\left[\frac{U_S-U_E}{R_8}+\frac{U_C-U_E}{R_6}+\frac{U_G-U_E}{R_7}\right]\cdot\beta=\frac{U_E}{R_{11}}$$

进一步化简为

$$\frac{U_S}{R_8}+\frac{U_C}{R_6}+\frac{U_G}{R_7}=\frac{U_E}{R_E}$$

$$\alpha_S U_S+\alpha_C U_C+\alpha_G U_G=U_E$$

其中

$$\alpha_S=\frac{R_E}{R_8}$$

$$\alpha_C=\frac{R_E}{R_6}$$

$$a_G=\frac{R_E}{R_7}$$

$$R_E=\frac{1}{\frac{1}{R_8}+\frac{1}{R_6}+\frac{1}{R_7}+\frac{1}{\beta R_{11}}}$$

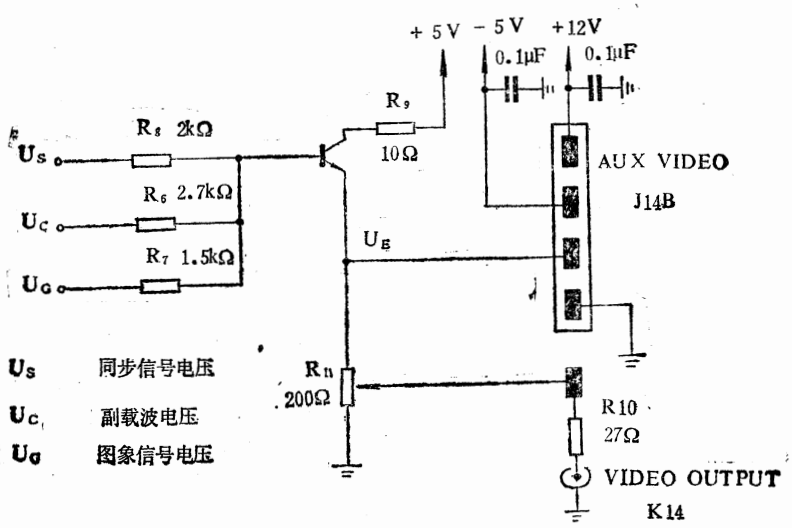


图 2.9.18 全电视信号由三种信号叠加输出

我们关心的是三种输入信号所占的比例。这三者的比值为

$$a_S:a_C:a_G=\frac{1}{R_8}:\frac{1}{R_6}:\frac{1}{R_7}$$

将$R_8=2k\Omega$、$R_6=2.7k\Omega$、$R_7=1.5k\Omega$代入上式得：

$$a_S:a_C:a_G=1:0.74:1.33$$

在图2.9.19中画出了一行信号的波形，图中画出的只是三种信号波形的相对大小。

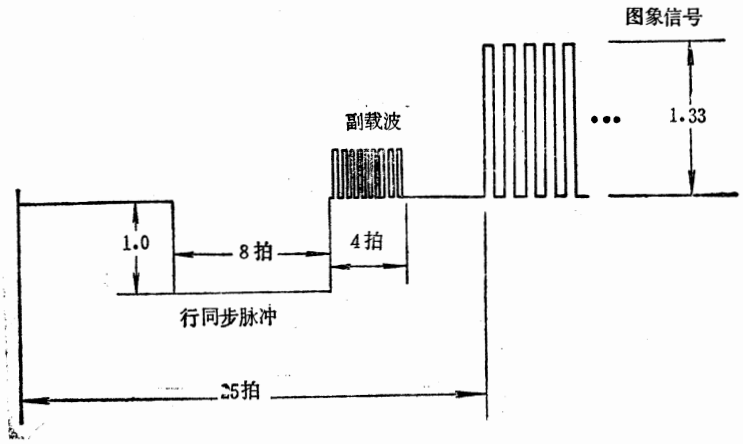


图 2.9.19 一行全电视信号波形

因为副载波电压（见图 2.9.2）通过R_5、L_1、C_2、C_3衰减了，故加到图2.9.18U_C端的电压要较U_S低。

图2.9.18中四引脚插座（AUX VIDEO JI48）是接射频调制器用的，而插孔（VIDEO OUTPUT K14）是作为直接接彩色监视器的插孔。

2.9.3 屏幕显示

从表2.9.1和表2.9.2得到，当V4 V3 V2 V1 V0为00000～10111、H5 H4 H3 H2 H1 H0为011000～111111时，屏幕将显示字符或彩色图形。屏幕显示方式分24行文本方式、混合方式（4行文本，其余为图形）和全屏幕图形方式三种。图形又分低分辨率图形和高分辨率图形两种。低分辨率图形全屏幕产生40×48小方块的16种彩色图形，高分辨率图形全屏幕产生280×192点的6种彩色图形。

为了改变显示方式，通过对C05X设备码进行读或写来实现，它将TEXT MODE、MIX MODE、HIRES和PAGE2四个信号置1或置0。它们分别是

TEXT MODE	C050 置0	设置图形工作方式
	C051 置1	设置文本工作方式
MIX MODE	C052 置0	设置全屏幕图形
	C053 置1	设置下面4行为文本
PAGE2	C054 置0	显示初始页
	C055 置1	显示第二页
HIRES	C056 置0	设置低分辨率图形
	C057 置1	设置高分辨率图形

在屏幕显示期间，视频信号发生器根据以上四个信号的状态、同步计数器的计数、以及在相应内存地址的取数，产生字符显示或彩色图形显示。这部分逻辑图见图2.9.20。

在2.5.4节我们已经讨论过在屏幕显示期间，根据同步计数器的计数和显示初始页还是第二页在相应内存地址取数的情况。也就是显示文本和低分辨率图形时，初始页占0400～07FF地址，第二页占0800～0BFF地址；显示高分辨率图形时，初始页占2000～3FFF地址，第二页占4000～5FFF地址。在φ0的一周期内（约1μs）取数一次，送到RAM锁存数据的输出D7、D6、…、D0。图 2.9.20的逻辑电路就是根据每微秒改变一次的D7、D6、…、D0的数得出显示需要的信号，然后送至图中的©端（©端与图 2.9.18 中的U_C连接）。

在屏幕显示时，共有三种显示方式：①显示字符，②显示低分辨率图形，③显示高分辨率图形。在分别讨论这三种显示方式之前，我们先讨论一下图2.9.20中的一些共同问题。

我们先看该图左上部的门电路和触发器。它们的功能是根据前面提到的四个信号转换成G、H、I三点的信号，用以控制是显示字符还是图形。但这里没有用到PAGE2这一信号，因为它只影响取数的来源（不同的内存区），这在2.5.4节中已讨论过。

$$F=\overline{V2\cdot V4\cdot MIX\ MODE+TEXT\ MODE}$$

TEXT MODE＝1时是文本方式，也就是显示全屏幕字符；MIX MODE＝1时屏幕下面

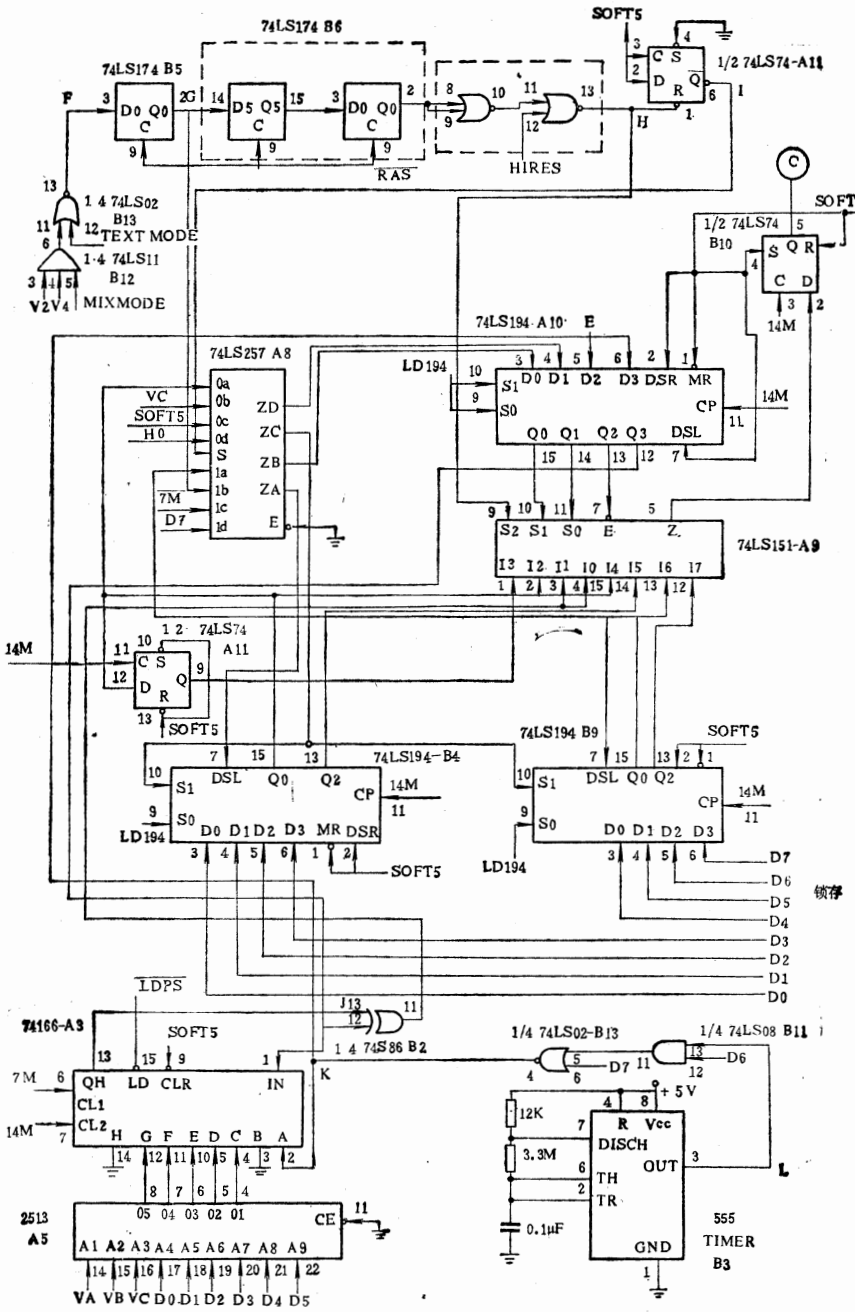


图 2.9.20 屏幕显示的逻辑电路

4行显示字符，而V2＝V4＝1正好对应下面4行（参看表 2.9.2），因此从上面的逻辑关系可以得到：显示字符时F＝0，显示图形时F＝1。

F输入到三个串接的触发器，由2MHz的时钟脉冲$\overline{RAS}$触发。每个触发器仅起将F点

来的信号延时1/2μs的作用，而不改变信号的状态。因此G点与F点一样，也是显示字符时G＝0，显示图形时G＝1。

三个触发器后面的第一个"或门"仅起反相作用，第二个"或门"当HIRES＝0时（显示低分辨率图形时）也只起反相作用，而在HIRFS＝1时（显示高分辨图形时）使H＝0。

因此显示字符时H＝0，显示低分辨率图形时H＝1，显示高分辨率图形时H＝0。

紧接着是一个触发器，它的时钟端C和输入端D已固定接在高电平，预置端S已固定接地。清除端R接来自H点的信号，当它低时，输出端$\overline{Q}$变高；它高时，输出端$\overline{Q}$变低，起反相作用。也就是显示字符和高分辨率图形时I＝1，显示低分辨率图形时I＝0。

G、H、I三信号的作用，在以下几节中分别叙述。

细心的读者一定会提出这样一个问题：对于显示来说，我们从内存地址取数是1μs取一次，而在显示字符或图形时在40μs期间要显示280点（每个字符宽度横向分7点，这在下面要详细叙述），也就是取一次数要显示7点。这是怎样做到的呢？它是靠移位寄存器将输入一字节的八位数逐位输出的。因此在图2.9.20中有三个移位寄存器，它们的作用是将输入的信号进行移位输出。一个八位移位寄存器（74166-A3）供字符显示时移位用，两个四位双向移位寄存器（74LS194-B4和B9）供图形显示时移位用。它们的输入都是从内存中取来的数（或取来后经过变换的数）。74LS194的时钟输入端接14MHz时钟信号，也就是1/14μs进行一次数据打入、位移或保持（由S1，S0的状态来控制）。74166的时钟CL1、CL2分别接到7MHz和14MHz时钟信号，我们在讨论显示字符时再讨论它们之间的关系。

移位寄存器的输出经74LS151-A9八输入端的选择和多路开关选择输出（由S2、S1、S0的状态控制），而S2、S1、S0的状态又由74LS257-A8和74LS194-A10进行控制。四位二选一数据选择器A8还控制B4和B9两个移位寄存器的工作。A9的输出经B10触发器输出到©点。

74LS 194-A10本来是移位寄存器，但是它们的S1、S0接在一起，连到LD194端。而LD194信号是1/14μs宽的1MHz脉冲，当它为正时，S1＝S0＝1，A10工作在并行打入状态；其余13/14μs为负，S1＝S0＝0，A10工作在保持状态。因此A10的输出每隔1μs改变一次。它的D0、D1输入端来自A8的ZB和ZD，它的输出Q0、Q1接到A9的S1、S0，用以控制A9的输入选择。D2端来自2.9.1中分析过的E点（见图2.9.3），在消隐时E＝1，使它的输出Q2＝1，连接到A9的使能端，使它的输出Z总为0。只有在显示时E＝0，它的输出Q2＝0，连接到A9的使能端，使它的输出跟随被选的输入端变化。

B4和B9的清除端MR恒接正，这两个片子总处在工作状态；右移输入端也恒接正。从下面的分析可以看到，这和两片子的工作没有关系。

从下面开始，我们分三部分讨论显示的三种方式，在每部分我们只绘出该部分要用到的器件和输入、输出端，不用的部分在图中省略，上面讨论过的部分也省略，这样每张图要相对简单清楚一些，容易把问题分析清楚。

1．显示字符

显示字符时的简化逻辑图如图2.9.21所示。

根据上节的分析，在显示字符时，I＝1接到数据选择器A8的通道选择端S，选择1b，

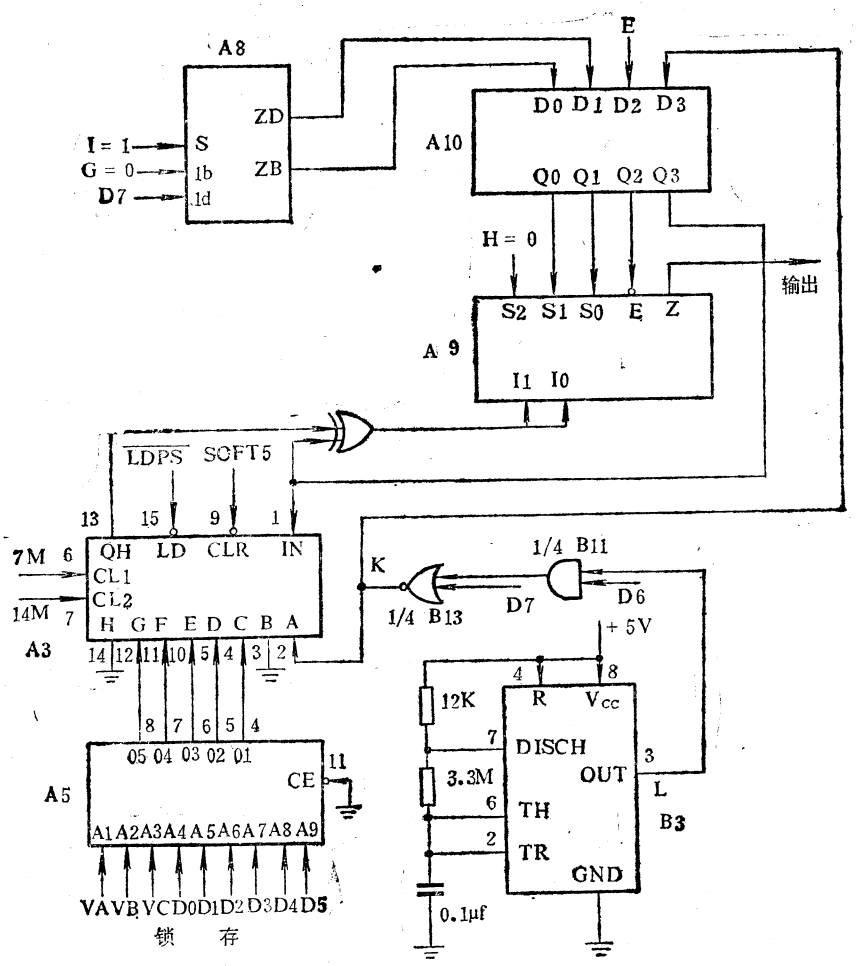


图 2.9.21 显示字符时的简化逻辑图

1d为ZB、ZD的输入端，1d接D7。1b接到G点，根据上面的分析，当显示字符时 G=0。ZB、ZD经过A10打入到Q0和Q1，也就是使A9的S1S0=01或00，A9的S2端接到H点，根据上面的分析，当显示字符时H=0，因此A9的S2S1S0为001或000。因此它的输出Z接通I1或I0，而I1或I0是B2的输出。B2有两个输入，上面的输入J来自八位移位寄存器A3的输出端QH；下面的输入来自A10的Q3，它是使字符黑白相反或闪烁的。

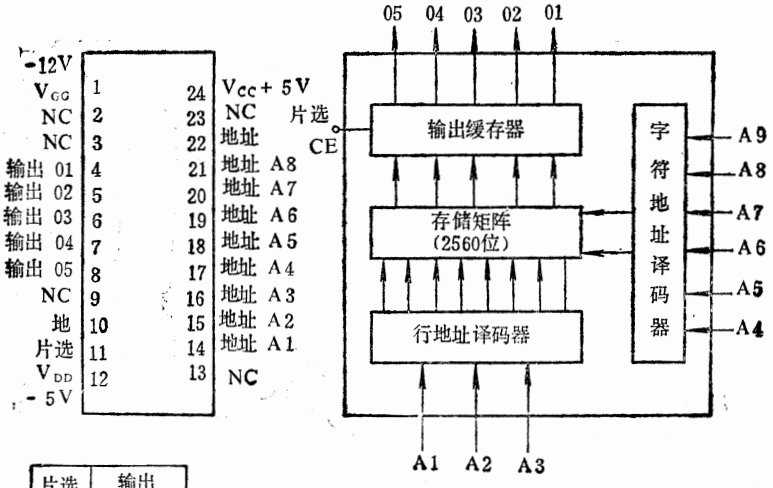
我们先说上面的输入，因为它是被用来产生字符本身的，是主要部分。A3的输入来自2513-A5，它是64×8×5字符发生器，即它能产生64个字符的信号，每个字符由8×5的点阵组成。

八位移位寄存器A3和字符发生器A5合在一起能完成一个什么任务呢？前面我们已经说过，从内存地址取数是1μs取一次。在显示字符时取的是该字符的ASCII码，显然ASCII码不能直接用于显示，而要经过一定的变换。将ASCII码变为用8×5点阵表示的该字符，该点阵产生不同的黑白组合就形成字符显示。图 2.9.22右上角举了"S"字符的点阵值的例子。为了得到不同字符的点阵，字符发生器需要输入字符的ASCII码和显示

的行数（即点阵八行中的哪一行）。

A3	A2	A1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

05	04	03	02	01
0	0	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	0	0
0	1	1	1	0
0	0	0	0	1
1	0	0	0	1
0	1	1	1	0



片选	输出
0	数据
1	开路

图 2.9.22　字符发生器2513

2513字符发生器的地址线共有9根，其中A4～A9是64个字符的ASCII码的低6位。可以显示的64个字符如表 2.9.5。

A3、A2、A1是表示每一字符点阵的八行中的具体行数，因此VC、VB、VA（在同步计数器的计数中，它们用以表示一个字符点阵中八行的行数）与它们连接。它的数据输出线一共有五根，对应每一个字符每一行中五个位置是0还是1。图 2.9.22以“S”为例，给出8×5点阵中每一点的0，1值。存储矩阵中存了64个字符和每一个字符八行的所有5位数。根据字符地址译码器和行地址译码器的结果，输出相应的显示信息。因此存

表 2.9.5 64个可显示的字符

000000	e	010000	P	100000	空格	110000	0
000001	A	010001	Q	100001	!	110001	1
000010	B	010010	R	100010	"	110010	2
000011	C	010011	S	100011	#	110011	3
000100	D	010100	T	100100	$	110100	4
000101	E	010101	U	100101	%	110101	5
000110	F	010110	V	100110	&	110110	6
000111	G	010111	W	100111	'	110111	7
001000	H	011000	X	101000	(	111000	8
001001	I	011001	Y	101001	)	111001	9
001010	J	011010	Z	101010	*	111010	:
001011	K	011011	[	101011	+	111011	;
001100	L	011100	\	101100	,	111100	<
001101	M	011101	]	101101	—	111101	=
001110	N	011110	∧	101110	.	111110	>
001111	O	011111	_	101111	/	111111	?

储矩阵中一共存了64×8×5=2560位数。

一个字符在屏幕上显示的次序应该是05、04、03、02、01。由于字符与字符之间还要有空隙，在每个字符的左右各有一列空白，所以一个字符所占的实际宽度为七点，也就是在字符发生器产生的五点左右各有一点、数值为0。

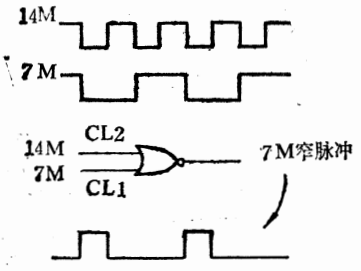


图 2.9.23 74166内部时钟逻辑图

字符发生器的输出05、04、03、02、01送到移位寄存器74166-A3的G、F、E、D、C端。A3的时钟端CL2接14M，时钟禁止端CL1接7M。所以实际时钟是7MHz（见图 2.9.23）。加载和移位端LD接至信号 $\overline{LDPS}$，$\overline{LDPS}$为负时，A5的05～01打入到 A3 的 QG～QC，而QH和QB是0，经过6次移位依次将 QG、QF、QE、QD、QC、QB移至QH（也就是J点）。这样在一个字符的一行期间显示了两个0和中间属于该字符的5点。

现在再来看B2输入端的下面部分。

如果它为1时，则B2输出端为$\overline{J}$，原来是黑底白字变成白底黑字，那么什么时候B2输入端的下面部分为1呢？这点来自A10的Q3，而它的对应输入端为D3。D3点信号的逻辑表达式为

$$K=\overline{L\cdot D6+D7}$$

下面我们将要分析L点，它是555计时器B3的输出，它接成振荡器形式，振荡频率约3Hz。

当D6＝D7＝0时，K＝1，B2输出端为$\overline{J}$。

当D7＝0 D6＝1时，L＝0，K＝1；L＝1，K＝0

K产生约3Hz的振荡，因而使字符显示产生3Hz的闪烁（即白底黑字，与黑底白字

相间出现）。由于字符发生器不用D7、D6两位，因而正好用来改变显示方式。当D7为1时，不管D6为0还是1，K均为0，显示正常黑白方式。

3Hz的振荡信号，由555计时器产生。它的管脚和逻辑图如图 2.9.24所示。

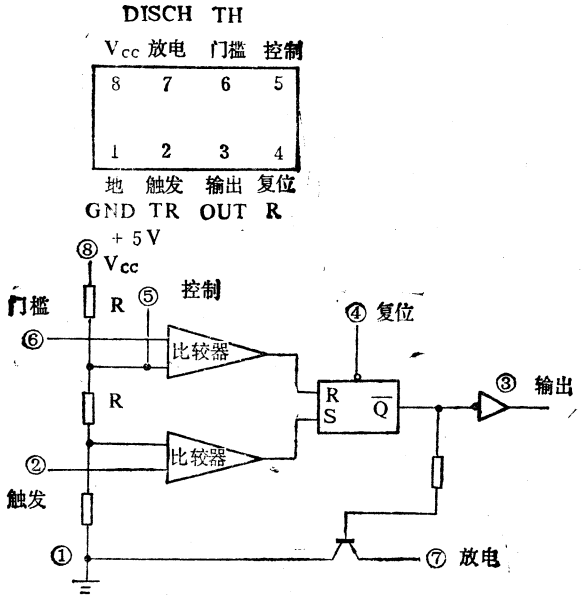


图 2.9.24 555计时器

在上面的比较器的下端输入一个较高的电压，同时在下面的比较器的上端输入一个较低的电压。从图 2.9.21得知门槛与触发端接在一起，是一个电容上的充电电压。开始时电容上的电压较低，上面的比较器输出低电平，下面的比较器输出高电平，使触发器的$\overline{Q}$为高电平。电容继续充电，直至下面的比较器输出低电平，上面的比较器输出高电平，使$\overline{Q}$变低；通过放电端使晶体管导通，电容放电。当电容上的电压低到一定程度时，触发器的输出$\overline{Q}$又变高，放电停止，又继续充电。

当$\overline{Q}$由低变高时，输出端3由高变低，因此输出振荡的电压。振荡频率决定于外接电阻、电容的数值。按照图 2.9.21所接的电阻、电容值，充电和放电的时间常数约为

$$3.3M\cdot 0.1\mu = 0.33s$$

2．显示低分辨率图形

显示低分辨率图形时的简化逻辑图如图 2.9.25所示。

根据2.9.3中的分析，在显示低分辨率图形时I=0，H=1。A8的通道选择端S为0时，选择0a、0b、0c、0d为ZA、ZB、ZC、ZD的输入端。A9的S2=1，S1=VC，S0=H0。因此S2、S1、S0与VC、H0的关系和A9选择的输入端如表 2.9.6所示。

4、I5和I6、I7分别来自B4和B9的Q0、Q2，而B4和B9的输入分别接到RAM锁存数据的输出D0～D3和D4～D7。每隔1μs根据同步计数器的计数从对应的内存地址读出数据，把该数据的低四位D0～D3打入到B4，高四位D4～D7打入到B9。从上表可知，数据

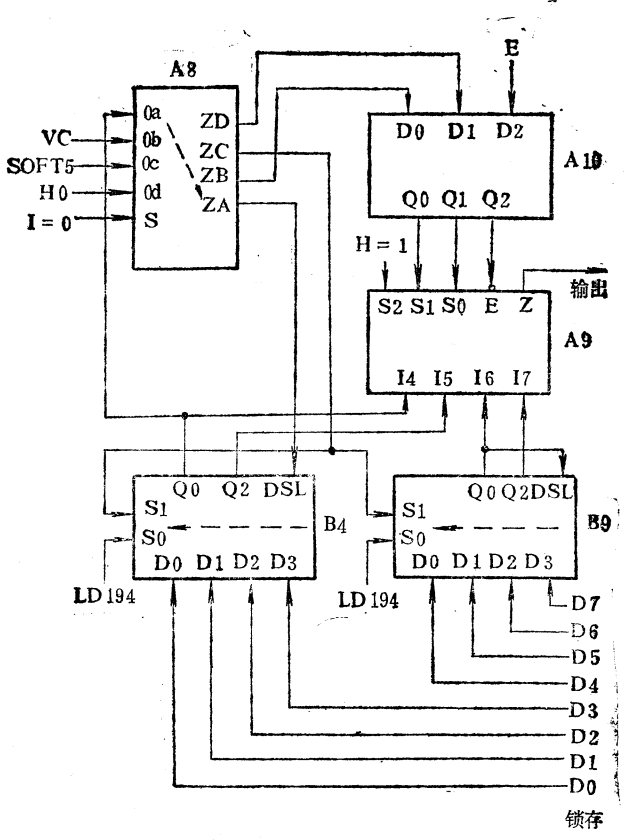


图 2.9.25 显示低分辨率图形时的简化逻辑图

表 2:9.6

VC	H0	S2	S1	S0	选择的输入端
0	0	1	0	0	I4
0	1	1	0	1	I5
1	0	1	1	0	I6
1	1	1	1	1	I7

的低四位对应VC＝0，也就相当于一个字符范围内的上面四行（相当于图形的偶数行）；数据的高四位对应，VC＝1，相当于一个字符范围内的下面四行（相当于图形的奇数行）。H0＝0相当于一行中第偶数个字符的位置，H0＝1相当于第奇数个字符的位置。

在Apple Ⅱ型微处理机中，规定了0～15的数，分别对应不同的彩色（表 2.9.4）。把这个数送到显示图形所对应的内存地址，如果是偶数行，就送到该地址的低四位，奇数行则送到高四位。

下面，我们通过一个例子说明图2.9.25的逻辑图是如何根据对应内存地址的取数来

输出所需的彩色信号的。

如果我们想在第一行画上一根浅绿线(高度为一个小方块，相当于半个字符)，在第二行画上一根黄线，在对应的内存地址应该存什么数？第一行在计算机内部被称为第0行，属于偶数行；第二行被称为第1行，属于奇数行。它们对应的内存地址为0400～0427。因此我们在0400～0427的低四位预先送上1100（因为浅绿色＝12），高四位预先送上1101（因为黄色＝13）。也就是给0400～0427送上DC。然后我们从显示第一行的最左边开始讨论。在LD194信号为正时，B4将0400所存的D0～D3打入到Q0～Q3，以后每隔1/14μs移位一次，并将Q0的值通过A9的I4送到Z端输出。Q0同时经过A8的0a→ZA送回B4的左移输入端DSL，这样B4的Q0、Q1、Q2、Q3四位的0、0、1、1就形成循环左移，在14个1/14μs内串行输出00110011001100。之后，B4又将0401的D0～D3打入到Q0～Q3。但这时由于H0＝1为奇数，所以A9的输入端为5，即输入为B4的Q2。而Q0～Q3的初始值还是0、0、1、1，14个1/14μs输出11001100110011，它与第一个14点相接成为图2.9.26的数串从Z端输出。它与副载波之间的相位决定了它是浅绿色（参见图2、9、14）这样的

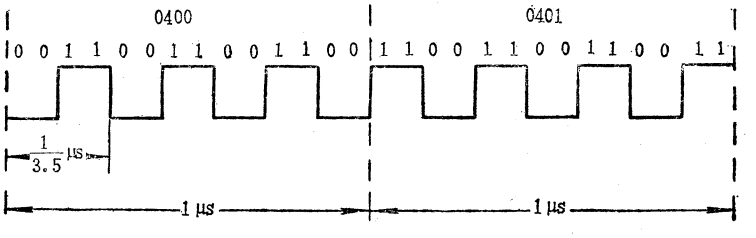


图 2.9.26 显示浅绿色时的输出波形

动作同样地重复四行（字符点阵宽度的四行）。在显示第二行(彩色方块的第二行)时，VC＝1，根据表2.9.6，A9的选择输入端为I6或I7，在LD194信号为正时，B9将0400所存的D4～D7打入到Q0～Q3，随后的工作原理和VC＝0时一样，从A9的Z端输出数串10111011101110……，如图2.9.27所示。它与副载波之间的相位决定了它是黄色（参见图2.9.14），这样的动作也同样地重复四行。

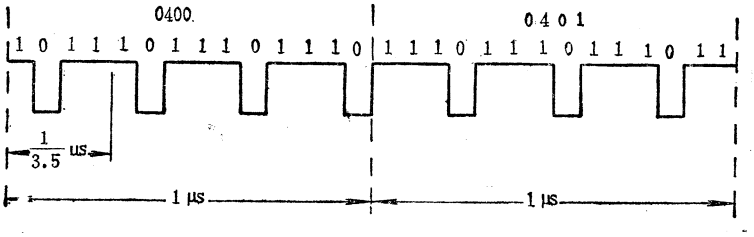


图 2.9.27 显示黄色时的输出波形

3. 显示高分辨率图形

显示高分辨率图形时的简化逻辑图如图2.9.28所示。

根据2.9.3中的分析，在显示高分辨率图形时，I＝1，H＝0，G＝1。A8的通道选择端S＝1时，选择1a～1d为ZA～ZD的输入端。A9的S2＝0，S1＝1，S0＝D7；也就

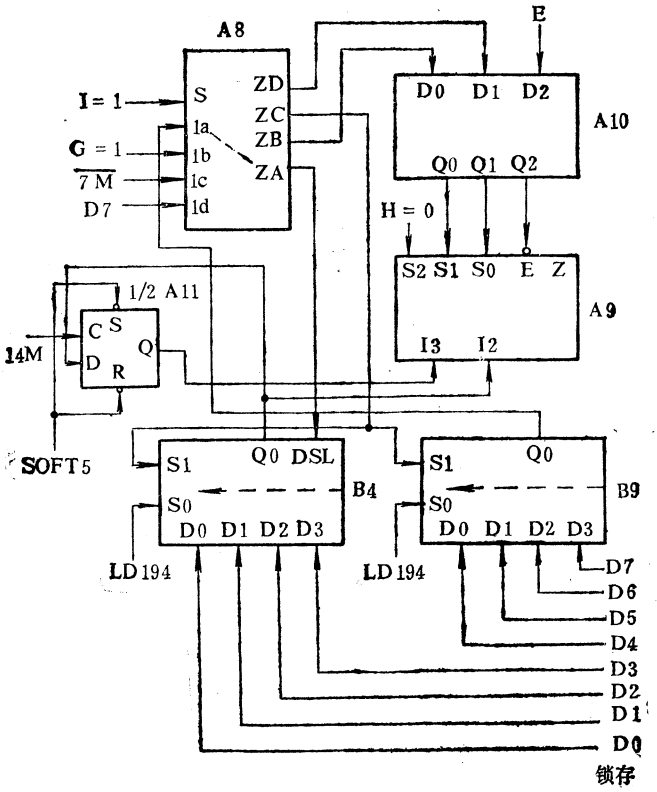


图 2.9.28 显示高分辨率图形时的简化逻辑图

是当D7＝0时，S2S1S0＝010，A9选择I2为输入端；当D7＝1时，S2S1S0＝011，A9选择I3为输入端。

由于A8的1C接到$\overline{7M}$，通过ZC接到B4和B9的S1，而S0接到LD194，当它为正时，$\overline{7M}$也为正，S1S0为11，RAM锁存数据D0～D7分别打入到B4和B9的Q0～Q3。当LD194为负时，S0＝0，S1＝0或1，因此S1S0分别为00和10相间，也就是保持和左移相间，1μs内移位6次。一个字节的数只有低七位起作用。当D7＝0时，B4的Q0（由D0打入）首先经I2到Z输出。移位时B9的Q0通过A8的1a、ZA输入到B4的左移输入端，使D4～D6接在D0～D3的后面输出。

我们还是以在第一行输出浅绿色为例说明图2.9.28的工作原理。我们预先在第一行显示对应的内存地址2000开始存以下数据（注意：这里的一行指的是高分辨率图形的一行，它相当于低分辨率图形一行的1/4）：

2000　2001　2002　……

2A　55　2A　……

也就是如图2.9.29所示。

所得波形和低分辨率显示浅绿色时一样，但每微秒只输出七点数据，每一点对应存储器所存储数据的一位，每位都可以独立置数，每一行也可独立置数（显示低分辨率图形时，每四行置同样的数）。

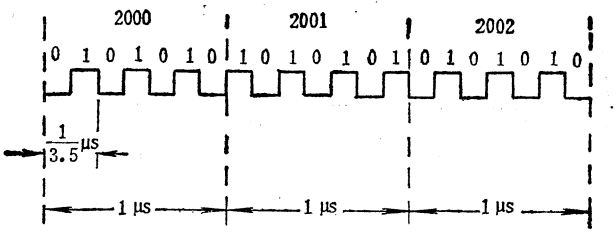


图 2.9.29 显示浅绿色时的输出波形

当D7＝1时，由于A9的输入选择为I3，而I3来自左下角的1/2A11触发器的输出 Q。而触发器的输入来自B4的Q0，因此Q0的值延时1/14μs进入I3。所以当D7＝1时与上例相比，波形右移1/14μs，相位发生变化，使彩色由浅绿色变为桔红。但由于一个字节的七位均由D7控制，因此七点中不可能有几点是浅绿，有几点是桔红。但可以有几点是浅绿，有几点是紫色（均是D7＝0）；或有几点是桔红，有几点是中蓝（均是D7＝1）。第一行彩色为桔红、紫色和中蓝时，对应的内存地址从2000开始存以下数据：

	2000	2001	2002 ……
桔红	AA	D5	AA ……
紫色	55	2A	55 ……
中蓝	D5	AA	D5 ……

从上面的分析可以看到，在显示高分辨率图形时，内存中的每一位数对应图形中的一个点。而要形成对应某一种彩色的一个周期的波形则需要两个点。因此要形成一个点宽度的不同彩条相间是不可能的。下面举一个例子来说明这个问题，如果我们想显示一列绿色与一列紫色相间，而且第一列是绿色，则显示绿色时内存中所存的数与输出波形已在前面给出。下面在图2.9.30绘出显示紫色时内存中所存的数和输出波形。

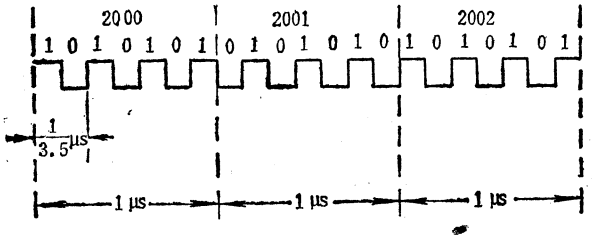


图 2.9.30 显示紫色时的输出波形

考虑到显示时从低位到高位取数，第一行第一列为绿色，2000中的D0应为0；第二列为紫色，D1也是0。依此类推，从2000开始存的都是00。这样，显示的既不是绿，也不是紫，而是黑色（参见表2.9.4）。如果我们改为第一列是紫色，第二列是绿色。结果从2000开始存的都是7F，显示的都是白色（参见表2.9.4）。

2.10 键　盘

Apple Ⅱ微计算机有一个打印机式的键盘，共有52个键。其键盘编码采用美国标准 信

息交换码（ASCII）。96个大写的ASCII代码字符中有91个可由键盘直接产生；还有US、FS、[\、_5个字符不能由键盘直接产生；另外还有两个换挡键（SHIFT），六个特殊键（CTRL、ESC、RESET、REPT、←、→）。表2.10.1中列出了各个键和与之对应的ASCII代码。由表2.10.2可得到ASCII代码十进制与十六进制的对应关系。

表 2.10.1 键和与之对应的ASCII代码

键名	按单个键	与CTRL键同时按	与SHIFT键同时按	与CTRL、SHIFT键同时按	键名	按单个键	与CTRL键同时按	与SHIFT键同时按	与CTRL、SHIFT键同时按
space	$A0	$A0	$A0	$A0	RETURN	$8D	$8D	$8D	$8D
0	$B0	$B0	$B0	$B0	G	$C7	$87	$C7	$87
1!	$B1	$B1	$A1	$A1	H	$C8	$88	$C8	$88
2"	$B2	$B2	$A2	$A2	I	$C9	$89	$C9	$89
3#	$B3	$B3	$A3	$A3	J	$CA	$8A	$CA	$8A
4$	$B4	$B4	$A4	$A4	K	$CB	$8B	$CB	$8B
5%	$B5	$B5	$A5	$A5	L	$CC	$8C	$CC	$8C
6&	$B6	$B6	$A6	$A6	M	$CD	$8D	$DD	$9D
7'	$B7	$B7	$A7	$A7	N^	$CE	$8E	$DE	$9E
8(	$B8	$B8	$A8	$A8	O	$CF	$8F	$CF	$8F
9)	$B9	$B9	$A9	$A9	P@	$D0	$90	$C0	$80
:*	$BA	$BA	$AA	$AA	Q	$D1	$91	$D1	$91
;+	$BB	$BB	$AB	$AB	R	$D2	$92	$D2	$92
,<	$AC	$AC	$BC	$BC	S	$D3	$93	$D3	$93
-=	$AD	$AD	$BD	$BD	T	$D4	$94	$D4	$94
.>	$AE	$AE	$BE	$BE	U	$D5	$95	$D5	$95
/?	$AF	$AF	$BF	$BF	V	$D6	$96	$D6	$96
A	$C1	$81	$C1	$81	W	$D7	$97	$D7	$97
B	$C2	$82	$C2	$82	X	$D8	$98	$D8	$98
C	$C3	$83	$C3	$83	Y	$D9	$99	$D9	$99
D	$C4	$84	$C4	$84	Z	$DA	$9A	$DA	$9A
E	$C5	$85	$C5	$85	→	$88	$88	$88	$88
F	$C6	$86	$C6	$86	←	$95	$95	$95	$95
					ESC	$9B	$9B	$9B	$9B

2.10.1 键盘电路的组成

Apple Ⅱ的内装键盘电路可分为下述各部分：键盘阵列、MM5740译码器、自激振荡器、REPEAT（重复）信号发生器及一些非门、与非门、或非门组成的电路。键盘电路图如图2.10.1所示。下面分别简单介绍各个部分的构成及功能。

1. 键盘阵列

Apple Ⅱ键盘的52个键中，除RESET、REPT、CTRL、及二个SHIFT键外，都属于键盘阵列。这47个键分布在X方向的9个位置及Y方向的10个位置组成的阵列中，详见键

表 2.10.2 ASCII码十进制与十六进制的对应关系

十进制	十六进制	128 $80	144 $90	160 $A0	176 $B0	192 $C0	208 $D0	224 $E0	240 $F0
0	$0	NUL	DLE		0	@	P		p
1	$1	SOH	DC1	!	1	A	Q	a	q
2	$2	STX	DC2	"	2	B	R	b	r
3	$3	ETX	DC3	#	3	C	S	c	s
4	$4	EOT	DC4	$	4	D	T	d	t
5	$5	ENQ	NAK	%	5	E	U	e	u
6	$6	ACK	SYN	&	6	F	V	f	v
7	$7	BEL	ETB	,	7	G	W	g	w
8	$8	BS	CAN	(	8	H	X	h	x
9	$9	HT	EM	)	9	I	Y	i	y
10	$A	LF	SUB	*	:	J	Z	j	z
11	$B	VT	ESC	+	;	K	[	k	{
12	$C	FF	FS	,	<	L	\	L	|
13	$D	CR	GS	-	=	M	[	m	{
14	$E	SO	RS	.	>	N	^	n	~
15	$F	SI	US	/	?	O	_	o	rub

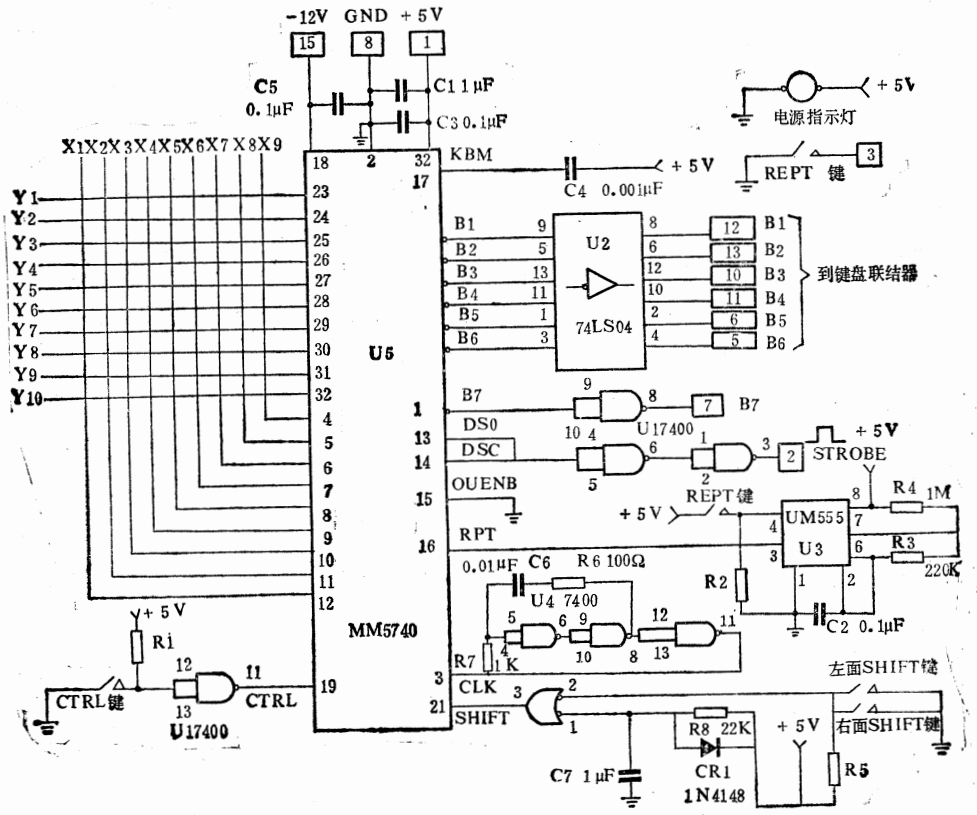


图 2.10.1 键盘电路图

盘阵列示意图2.10.2。X方向的每一条线与MM5740键盘译码器的X1～X9输出端相连接。Y方向的每一条线与MM5740的Y1～Y10输入端相连接。

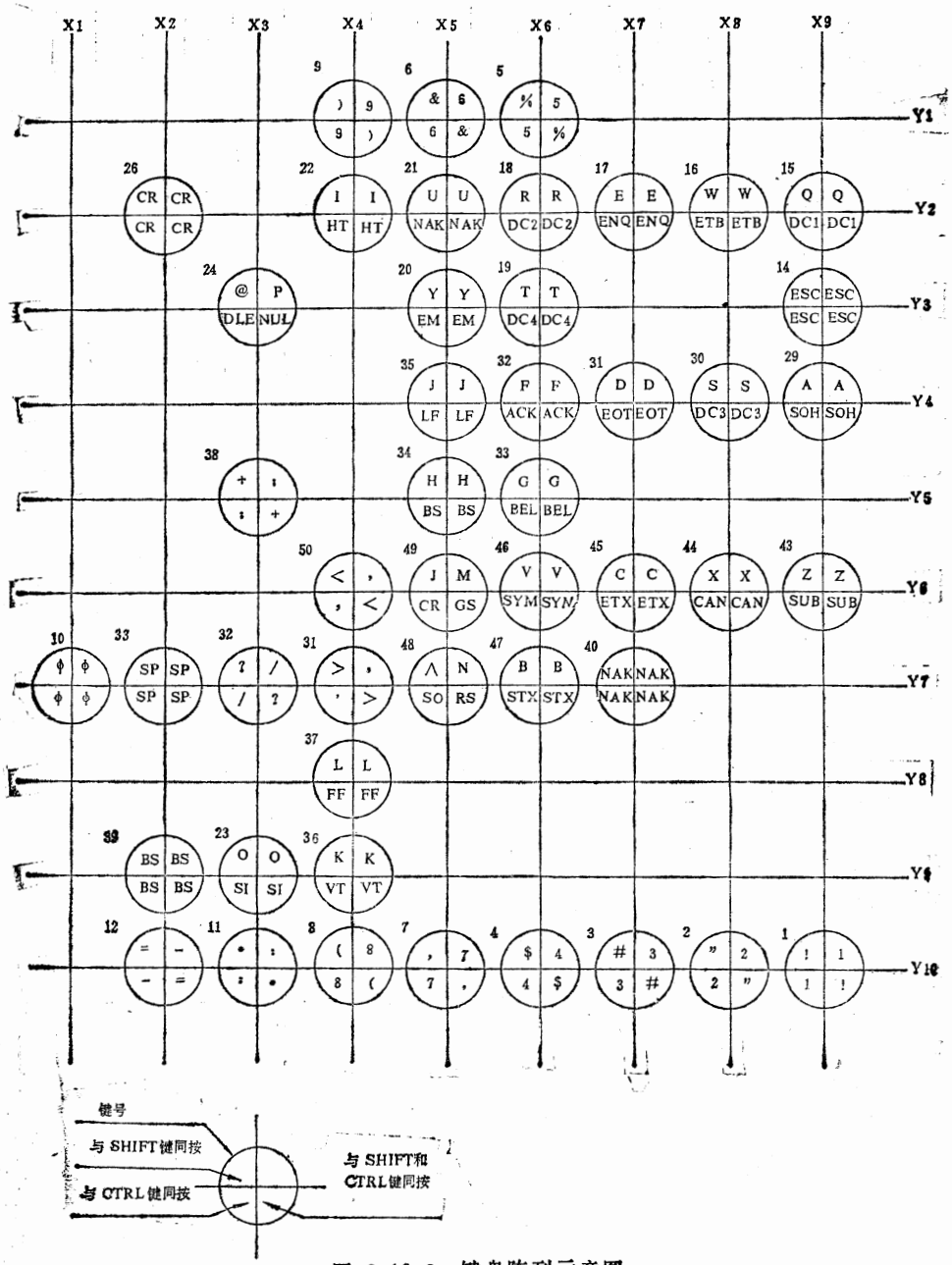


图 2.10.2 键盘阵列示意图

2. 键盘译码器MM5740的结构及管脚功能

（1）MM5740的结构框图如图2.10.3所示

（2）MM5740的管脚功能

MM5740的管脚图如图2.10.4所示

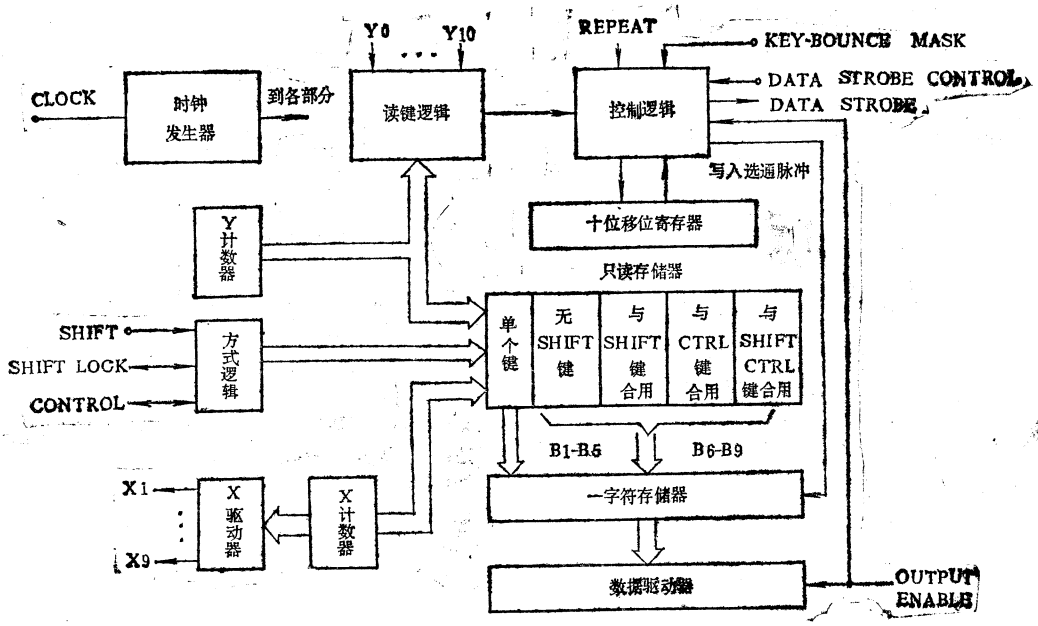


图 2.10.3 键盘译码器MM5740的结构框图

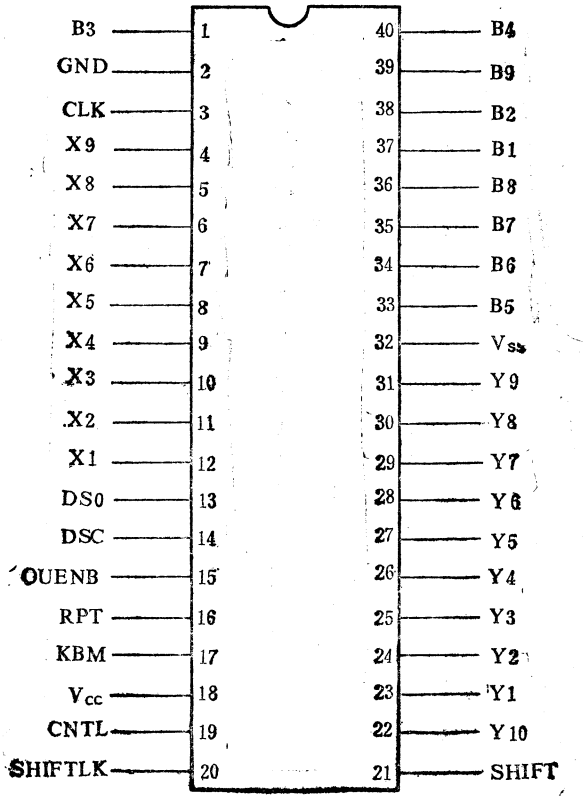


图 2.10.4 键盘译码器MM5740管脚图

键盘译码器MM5740的管脚功能如下：

名　称	脚号	说　　明
X⑬～X⑨	④～⑫	是MM5740的输出端，用来驱动键盘开关阵列。以900μs的扫描周期，依次给阵列中的X1～X9的9条线加正脉冲。
Y1～Y10	22～31	是MM5740的输入端。把键盘阵列的Y1～Y10输入到MM5740。这些端经键盘开关阵列连到X1～X9的驱动线上。Y1～Y10平时为低电位，当按下某键时，对应的某一个Y被拉到高电位。
B1～B9	1,33～40	是MM5740的数据输出端。键的代码由这9位输出组成。这些输出是具有和TTL兼容的三态输出。当"输出使能"(ENB)端(15脚)为高电位时，这些输出端为高阻状态。
DSO	13	是MM5740的数据选通输出端(DATA STROBE OUTPUT)。这条脚的功能是表示键盘已经送出有效数据，而且随时准备好承认。高电位表示实际数据的选通。数据选通可以是电位方式、也可以是脉冲方式。
DSC	14	是芯片的数据选通控制(即DATA STROBE CONTROL)输入端。这一端是为了输入一控制信号，以控制"数据选通输出"信号的脉冲宽度。当与数据选通输出（13脚）连接时，数据选通输出的脉宽为一位时钟脉冲的宽度。数据选通输出的脉冲宽度可以通过在数据选通输出和选通控制间加一RC网络来改变。在数据选通采用电位方式时，可以接到V_{SS}(+5V)或数据选通输出上。
OUENB	15	是芯片的"能够输出"(OUTPUT ENABLE)的控制端。这一端是三态数据输出B1～B9的控制端。同时还控制数据选通返回到空载条件（低电位）。在电位选通方式时需要这样。
RPT	16	是芯片的重复信号(REPEAT)输入端。这是为经重复键(REPT)接受一重复信号而设置的。重复信号的每个正向跳变都会使芯片发出一数据选通信号。因此如果10Hz信号经重复键开关送入RPT输入端，当任一数据键和REPT键同时按下时，每秒钟将发出10个字符数据选通信号。
KBM	17	是防止键颤引起误码的输入端。(KEY-BOUNCE MASK)。这一端通过一个0.001μF的小电容接电源(+5V)。
SHIFT	21	是芯片的控制端。当这一端被拉到高电位时，译码器将呈现SHIFT字符方式。
CNTL	19	是芯片的控制端。这一端被拉到高电位时，译码器将呈现CONTROL（控制）字符方式。
HIFT	LK 20	可作输入端，也可作输出端。当按下"SHIFT LOCK"键（SHIFT锁定键）时，仍保持SHIFT方式，同时这一端还可作为输出去驱动一只指示灯。当再按SHIFT键时，又可恢复上述功能。
CLK	3	由这一端加入时钟脉冲。
V_{CC}	32	+5V电源输入端。
GND	2	接地端。
V_{SS}	18	−12V电源输入端。

（3）Apple Ⅱ系统中键盘译码器的连线及工作原理

由键盘电路图2.10.1中可看到，键盘阵列的X1～X9共9条线连到MM5740的4～12脚。Y1～Y10共十条线连到MM5740的23～32脚。

MM5740在自激振荡器的控制下，以900μs的扫描周期，快速扫描键盘上的键盘阵列。按下一个键后，通过X1～X9、Y1～Y10共19条信号线的状态及SHIFT、CTRL键所决定的逻辑方式，由MM5740中的只读存储器译码输出后，将一个字符的9位代码送到"一个

字符存储器"中存储起来。由于Apple Ⅱ中MM5740的输出允许端15脚是接地的，因此在MM5740中，一旦"控制逻辑"输出一写入选通脉冲，其"一个字符存储器"就将对应的ASCII码经数据驱动器输出。输出端共9位，Apple Ⅱ仅用7位B1～B7。如果不按新键，这个键的ASCII码就一直存储在"一个字符存储器"中，直到按新键，其内容才改变。

按下REPEAT键，REPEAT信号发生器经MM5740中的控制逻辑电路产生重复脉冲，触发"一个字符存储器"。从而使MM5740的输出B1～B7，重复输出同一ASCII码。

在Apple Ⅱ中，MM5740的"数据选通控制"与"数据选通输出"是接在一起的。故按一键后，经MM5740中的"控制逻辑电路"产生一"数据选通脉冲"输出。输出脉冲经位于U1的7400二个与非门驱动后，作为"键盘选通脉冲 STROBE"输出。STROBE的脉冲宽度为10μs。

3. 自激振荡器

自激振荡器的电路图如图2.10.5所示。自激振荡器是用来给键盘译码器内部的时钟发生器提供触发脉冲的。自激振荡器由键盘印刷板上U4的7400的三个与非门组成。这个振荡器的振荡频率受键盘电路板上的C_6、R_6和R_7的控制。

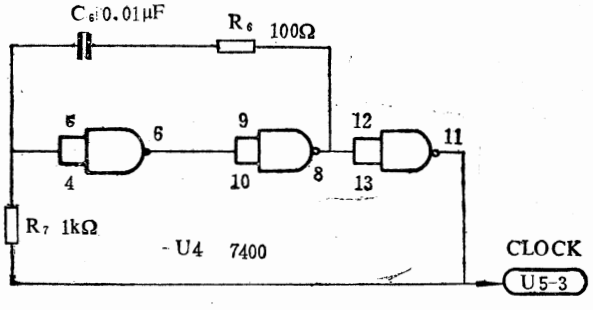


图 2.10.5 自激振荡器电路图

4. REPEAT信号发生电路

当键盘上的REPT键单独被按下时，产生一个和上一次按的键相同的键码。按下某一字符键的同时再按REPT键，其作用相当于以每秒按十次的速率重复按这个键。若这时释放REPT键或释放字符键，都可停止重复。为了实现上述功能，键盘上REPT键的触点连接至位于键盘电路板U3的LM555定时电路芯片上。此芯片和电容C_2、电阻R_3、R_4产生约10Hz的"REPEAT"信号。电阻R_2是为限制REPT键的触点电流而设置的。若220kΩ(R_3)电阻用较小的电阻值替换，则REPT键就会以更快的速率重复所按的字符键。为便于说明REPEAT信号的产生原理，将LM555的内部电路及其与R_2、R_3、R_4及C_2的连线示于图2.10.6。

从LM555的内部电路看，它有二个比较器。若电源电压为V_{CC}（第8脚），则比较器Ⅰ的基准电压$V_{B1}=\frac{2}{3}V_{CC}$；比较器Ⅱ的基准电压$V_{A2}=\frac{1}{3}V_{CC}$。合闸后电源经电阻R_3、R_4向电容C_2充电。当$V_{C_2}<\frac{1}{3}V_{CC}$时，$V_{A1}<V_{B1}$，比较器Ⅰ的输出为低即$I_{出}=0$。而

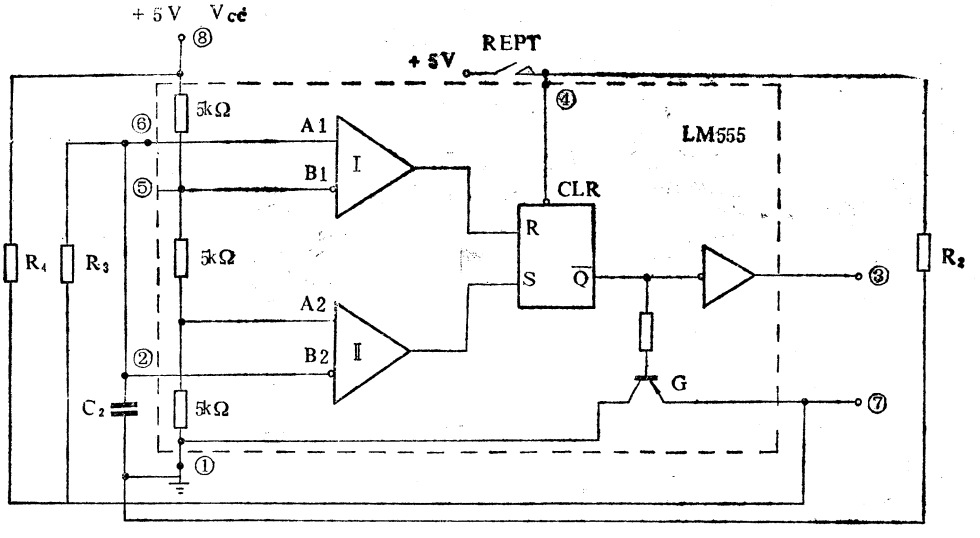


图 2.10.6 REPEAT信号发生电路

$V_{A2}>V_{B2}$，比较器Ⅱ输出为高，即Ⅱ出＝1。故双稳态触发器输出$\overline{Q}$为高，即$\overline{Q}$＝1。LM555输出③为低，即③＝0。随着电源经R_4、R_3向C_2继续充电。当$V_{C_2}>\frac{1}{3}V_{CC}$时，Ⅰ出＝0，Ⅱ出＝0。故触发器保持原状态，$\overline{Q}$＝1，，③＝0。

当$V_{C_2}>2/3V_{CC}$时，Ⅰ出＝1，Ⅱ出＝0故$\overline{Q}$＝0，③＝1。因为$\overline{Q}$＝0放电管G导通，故C_2经R_3和放电管放电。

当放电到$V_{C_2}<\frac{2}{3}V_{CC}$时，Ⅰ出＝0、Ⅱ出＝0，则$\overline{Q}$保持，即$\overline{Q}$＝0。③＝1，$C_2$继续放电。

当$V_{C_2}<\frac{1}{3}V_{CC}$时，Ⅰ出＝0，Ⅱ出＝1，则$\overline{Q}$＝1，③＝0，放电管G关闭。电源端⑧经$R_4$、$R_3$向$C_2$充电。

由于电容C_2的不断充放电，就使LM555的输出③产生周期性的脉冲信号。该信号作为REPEAT键的时钟信号，连接到键盘译码器MM5740的16脚。

电容C_2的充电时间$t_{充}\approx0.693(R_4+R_3)C_2$；电容$C_2$的放电时间$t_{放}\approx0.693R_3\cdot C_2$；振荡周期$T=t_{充}+t_{放}\approx0.693(R_4+2R_3)\cdot C_2$；根据图2.10.6所给的参数可求出$T\approx0.097s$所以REPEAT信号的振荡频率$=\frac{1}{T}\approx10Hz$。

5．SHIFT信号的产生

如图2.10.7所示，MM5740上的SHIFT信号一方面可由按键盘左面的或右面的SHIFT键来产生低电位，把这个信号送到位于U4的低电位或非门输入端2，则U4的输出端3，产生高电位，作为SHIFT信号，送到MM5740的21脚，以使键盘以SHIFT方式运

行。另一方面，在合闸瞬间由于电容C_7两端电压不能突变，U4的1端为低电位，这时输出端③为高电位，等C_7的电位充到U4的关门电位时，1端为高电位。因为2端为高电位，故低电位或非门U4的输出3为低电位，即合闸时产生一个SHIFT脉冲。

6．CTRL信号的产生

CTRL键按下后，使键盘电路上位于U1的7400与非门的输入端12和13变成低电位。反相后该与非门的输出11（此时为高电位）作为CTRT信号，连到MM5740的19端。经MM5740的方式逻辑电路送到只读存储器。参见图2.10.2。

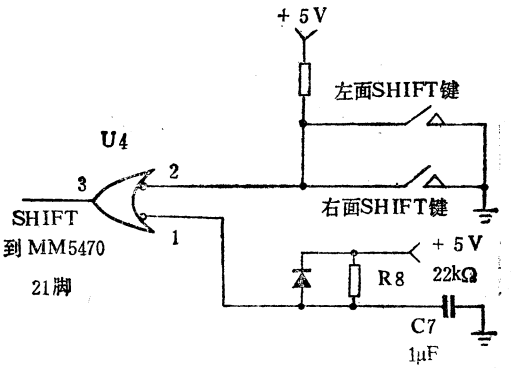


图 2.10.7　SHIFT信号产生电路

CTRL和SHIFT键本身不产生键代码，它们和其它某些键同时按下时，可改变那些键的键代码和功能。产生何种键代码，可由表2.10.1查出。这些键代码主要由MM5740中的只读存储器决定。

2.10.2　键盘接口的组成及其工作原理

键盘接口可分为开机复位电路、键盘联接器、D触发器、四位二选一三态输出的数据选择器（2片）四个部分。键盘接口电路图如图2.10.8。

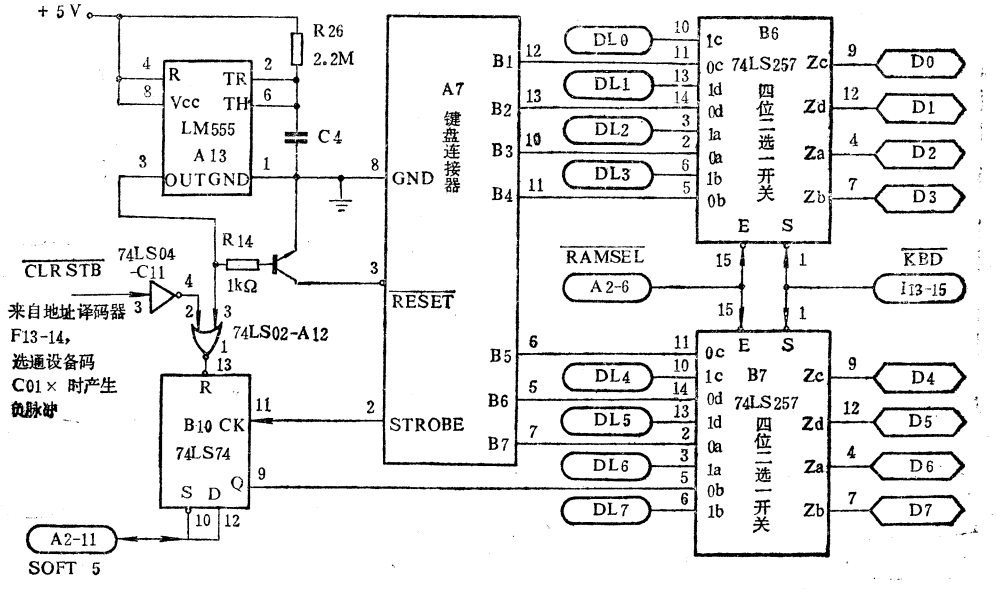


图 2.10.8　键盘接口电路图

1．开机复位电路

开机复位电路是由位于主电路板A13的LM555定时器和位于A12的或非门组成。其电路图如图2.10.9。

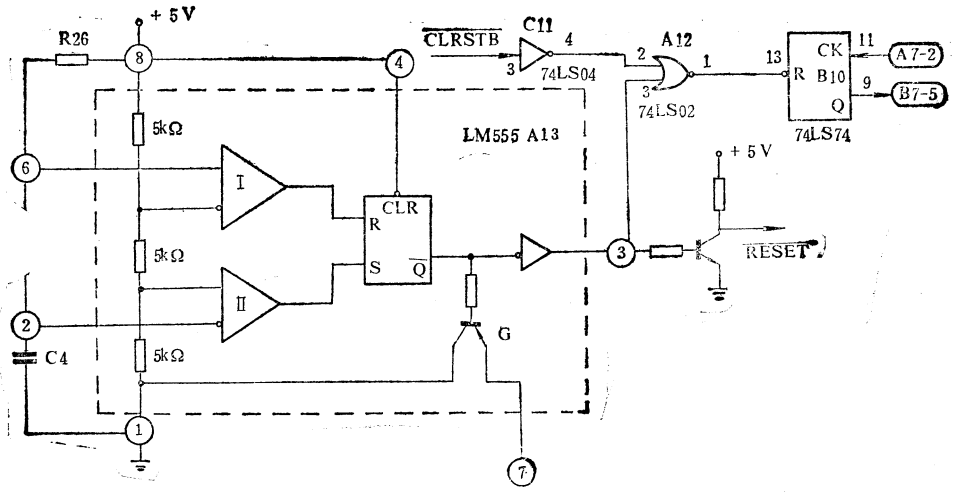


图 2.10.9 开机复位电路

从图2.10.9可知，合闸后电源经电阻R_{26}向电容C_4充电。当$V_{C4}<\frac{1}{3}$电源电压V_{cc}时，LM555中的比较器Ⅰ输出低电位；比较器Ⅱ输出高电位，即$\overline{Q}$为高电位。这就使LM555的输出③为低。随着电源经R_{26}向C_4充电，当$V_{C4}>\frac{2}{3}$电源电压时，比较器Ⅰ输出高电位，比较器Ⅱ输出低电位，使$\overline{Q}$为低，LM555输出③为高电位。这就是说合闸接通电源时，LM555输出端③产生一个正脉冲。这个正脉冲一方面经位于主电路板A12的74LS02或非门反相后，触发位于B10的D触发器，使之清零。另一方面经三极管反相放大后产生$\overline{\text{RESET}}$信号，这个信号经位于A7的键盘联接器送到MPU的$\overline{\text{RESET}}$端，使MPU开始一个总清周期。详见2.1.1节中MPU和总线驱动器一节。

$\overline{\text{RESET}}$信号除由开机复位电路产生外，按RESET键也能产生。键盘上的RESET键不产生ASCII码。RESET键直接经键盘联接器与MPU联接。因此当此键按下后，产生的$\overline{\text{RESET}}$信号使MPU开始一个清零周期。

2. 键盘联接器

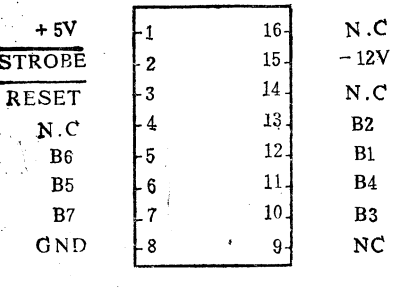


图 2.10.10 键盘联接器引线图

位于主电路板A7的键盘联接器是将键盘电路产生的7位键代码信号B1～B7、一个选通脉冲STROBE和一个复位清零信号RESET通过键盘联接器送到主电路板上。B1～B7被送到位于B6、B7的二选一数据选择器的一组输入端。STROBE被送到位于B10的D触发器的触发脉冲端。$\overline{\text{RESET}}$被送到MPU和外设插座的$\overline{\text{RESET}}$输入端。

键盘联接器的引线如图2.10.10。各引线信号的说明见表2.10.3。

表 2.10.3 键盘联接器的信号说明

引 腿	名 称	说 明
1	+5V	+5V电源，该引腿流出的总电流必须小于120mA
2	STROBE	键盘的选通脉冲输出端，每按一次键，此线就送出一个脉冲宽度为10μs以上的脉冲，选通可以是两种极性的。
3	$\overline{\text{RESET}}$	送到微处理器的$\overline{\text{RESET}}$端，平常为高电位，按RESET键时此线被拉低。
4、9、14、16	N.C	没有连线。
5～7、10～13	B5～B7、B10～B13	ASCII键盘的七位数据输入。
8	地	系统地。
15	−12V	−12V电源。最大电流应小于50mA。

3．D触发器

用以判断有无键入的标志。D触发器输出Q端接到二选一多路开关的一个输入端0b，最后Q端的信号将被送到数据总线最高位D7，用以判断有无键入。当D触发器被置1，即Q=1时，将使D7=1，表示有键入。反之，当D触发器被复位，即Q=0时，将使D7=0，表示无键入或键码已被取走。在正常工作情况下，有键入时通过键盘联接器送来一个选通脉冲STROBE将D触发器置1。数据取走后，通过软件使D触发器置0（复位）。这只需要在软件程序中排一条涉及C01X（X表示0～F的任意数）的指令就可以了，如BIT C010。监控系统中专门有一段键入程序行使这些功能，下面将予以介绍。开机时，开机复位电路中LM555产生一个正脉冲，通过或非门反向后将D触发器复位。

4．二选一多路开关

由位于B6、B7的两片四位二选一多路开关（二选一数据选择器）74LS257构成8位，其输入来自两路：一路是带锁存器的RAM输出（图上标着DL0～DL7），另一路是键盘联结器的七位数据B1～B7。加上D触发器的Q端。二选一多路开关的输出接到数据总线上。

具体什么时候选通RAM的输出数据（DL0～DL7）到总线，什么时候选通键盘数据（B1～B7）到总线，这是分析键盘输入接口最关键的问题。而要搞清这一问题，就需要弄清这二片多路开关的输出控制端E（15脚）和通道选择端S(1脚)所接的信号。

E为输出控制端。从74LS257的功能表（查阅书末附录A）可知，当E为低时，该片有数据输出，E为高时，其数据输出端呈高阻状态。E端信号来自RAM选择电路的$\overline{\text{RAMSEL}}$信号。产生此信号的逻辑电路如图2.10.11。

在与非门D2的四个输入端中，脚13接$\overline{\text{KBD}}$信号。当读键盘的设备码C00X选通时，该脚为低电位。当选中RAM时。RAM选择电路中位于J1的74LS257的脚9和12一定有一

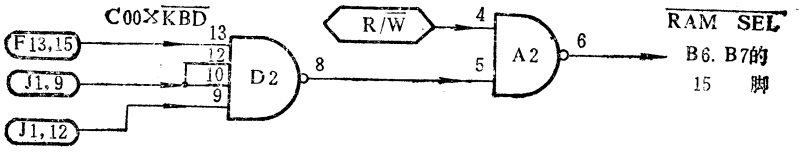


图 2.10.11 产生 RAMSEL 的逻辑电路图

个为低电位。

在读周期时（即R/$\overline{W}$=1），与非门A2的第4脚为高电位，这时与非门D2的三个输入端中，只要有一个出现低电位，$\overline{RAMSEL}$就为低电位。位于B6、B7的二选一多路开关就有输出。这就是说，在读周期时，当RAM选中或要读键盘输入时，$\overline{RAMSEL}$为低电位，这时B6、B7二片选通，其数据送到数据总线。

S为通道选择，当$\overline{RAMSEL}$为低时，由S决定选择哪一路数据到总线，S为低时，选择来自键盘联接器数据B0～B6和位于B10的D触发器的Q端的信号。S为高时，选择带锁存的RAM输出信号DL0～DL7。S端的信号来自位于F13的译码器的输出脚15($\overline{KBD}$)。$\overline{KBD}$在选通读键盘设备码C00X时，为低电位。

综上所述可以得出结论：在读周期时，当读键盘设备码C00X选通时，$\overline{RAMSEL}$为低电位，$\overline{KBD}$也为低电位，就把ASCII键盘代码B1～B6和键盘选通信号（位于B10的D触发器的Q端）送往数据总线。当有RAM地址(0000～BFFF)选通时，$\overline{RAMSEL}$为低，$\overline{KBD}$为高，就把带锁存的RAM输出数据DL0～DL7送往数据总线。

5．键盘信息的读取

在Apple监控系统中专门有一段读取键盘信息的子程序(KEYIN)。这段程序的功能是：判有无键盘输入；若有，就取走键盘代码并把D触发器清零；若没有，就踏步等待。这段程序的入口地址为FD1B。程序如下：

```
FD1B INC 4E      随机数低位+1
FD1D BNE FD21
FD1F INC 4F      随机数高位+1
FD21 BIT C000    B1～B7⇒数据总线D0～D6，Q(B10的Q)⇒数据总线D7。因为设备码C000被选通。同时D7被送到MPU的状态寄存器第8位(N标志位)。
FD24 BPL FD1B    有键入时，D7=1即N=1；无键入时，D7=0，即N=0。若无键入，则返回FD1B等待键入。
FD26 STA (28),Y  有键入时，改变屏幕上闪烁光标的位置。
FD28 LDA C000    B0～B6⇒D0～D6，Q (B10的Q)⇒D7。D0～D7送累加器。即键代码送累加器A。
FD2B BIT C010    给位于B10的D触发器清零，表明键码已取走。
FD2E RTS         子程序返回
```

上述基本指令的功能请参看附录B。

一般来说，键盘信息从数据总线送到MPU的累加器经过软件进行一系列分析判别后，除编辑字符、控制字符外都一方面被送到键盘的数据缓存区（0200—02FF），另一方面送到RAM的字符显示区。

2.10.3 几个特殊键的使用及功能

下面仅对几个特殊键的一般使用方法及功能作一简单介绍。这些特殊键在不同语言下使用时，可能还有不同功能。详见Apple Ⅱ各种语言的使用手册。

1. SHIFT键：

在Apple Ⅱ的键盘上按下一个字符键，屏幕上显示该键的字符或双字符键的下部字符。若同时按下字符键和 SHIFT 键，即显示该键的上部字符。这是由于键盘译码器在上述两种情况下译码输出不同。详见2.10.1节。

2. ESC键

此键具有编辑功能。按下ESC键，Apple Ⅱ进入Escape方式。在该方式下，有11个键具有特殊功能，这11个键被称为"Escape码"。若按一下ESC 键，然后按"Escape码"可执行相应的功能。若在Escape方式下，按了非"Escape码"，那么Apple Ⅱ不理采此次按键，并结束Escape方式。

Apple Ⅱ的11个Escape码中，有8个是作纯光标移动。它只移动光标，不改变屏幕显示和已输入行的内容。另外三个是用于清除屏幕的一部分或全部。6个Escape码(@、A、B、C、D、E、F）操作后都会结束Escape方式，另四个 Escape 码（I、J、K、M）则不结束Escape方式。下面分别介绍它们的功能。

ESC @　称为"复原并清除"，它清除整个屏幕，同时将光标置于左上角。

ESC A　按ESC键后再按A键，使光标右移一格。

ESC B　按ESC键后再按B键，使光标左移一格。

ESC C　按ESC键后再按C键，光标下移一行。若光标已在屏幕窗口的最底行，按ESC C后，屏幕窗口内的文本向上卷一行，光标不动。

ESC D　按ESC键后再按D键，光标上移一行。若光标已在屏幕顶行，按ESC D后光标不动。

ESC E　称为"清除至行末"。按ESC键后再按E键，将屏幕窗口内光标所在的那一行从光标开始清除原文至行末。

ESC F　称为"清除至屏幕末"。按ESC键后再按F键，它将清除屏幕窗口内光标右方和下方的全部内容。光标位置不变。通常用它来清除屏幕上杂乱不用的内容。

ESC K　光标右移一格。

ESC J　光标左移一格。

ESC M　光标下移一行。

ESC I　光标上移一行。

最后四个Escape码不结束Escape方式，故可以连续使用 Escape 码。例如，按ESC

键后按I键，光标上移一行，接着再按M键，光标就下移一行。

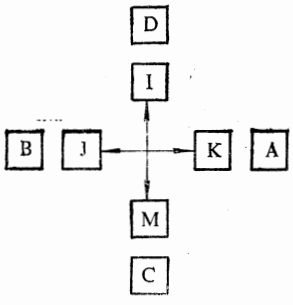


图 2.10.12 光标移动方向示意图

上述这些移动光标的键按图2.10.12排列。由中心到每个键的方向，恰好对应于Escape码移动光标的方向。

3．“←”键和“→”键

此两键也具有编辑和修改功能。按一次“←”键，左移光标一格，删除一个字符。按一次“→”键右移光标一格，并复制改写的字符。“←”键改正打字错误。“→”键有利于重新输入一行的其余部分。“→”键与纯光标移动结合起来，可反复拷贝和编辑屏幕上的信息。

4．CTRL键

为叙述方便，在某些规定的字母上，加上标C来表示控制字符。例如B^C表示同时按CTRL键和B键。控制字符命令不能在屏幕上显示。不同程序语言控制字符表示的功能，详见该程序语言的使用说明。下面仅介绍通用的一些控制字符的作用和功能。

B^C 在监控状态下（以＊为催向符）同时按CTRL键和B键，然后按RETURN键，将使Apple Ⅱ进入BASIC状态，并置用户存储器使用区域，使最高存储区到现有的最高存储单元，最低到2048。

C^C 分两种情况：

1．在监控状态下，按C^C，然后按RETURN键使 Apple Ⅱ 在不破坏现行程序下进入BASIC状态。

2．在执行其它程序语言时，按C^C均可立即停止程序的执行和列表功能，并显示暂停处的行数，如用CONT命令，则可继续执行。

E^C 在监控状态下，按E^C然后按RETURN键，则可显示6502微处理器内，状态寄存器的内容。

G^C 按G^C则喇叭发出响声。

H^C 按一次H^C，光标左移一格，并删除 Apple Ⅱ 相应内存的字符代码，而不删除屏幕上的字符。

J^C 按一次J^C，发出一换行信号。

S^C 按S^C暂时停止程序的执行和列表功能，再按S^C可使被S^C暂停的程序继续下去。

U^C 按一次U^C，光标右移一格，并复制改写的字符。

X^C 按X^C可使Apple Ⅱ 删除刚才打入的一行信息。

Y^C 在监控状态下，按Y^C然后按RETURN键，则从规定的存储单元03F8，执行用户指定的程序。

K^C 按一外设所在槽号的数字键，再按K^C则可将 Apple Ⅱ 的键盘输入转到接在该槽上的外设上。

P^C 按某外设槽号的数字键，再按P^C，则可将Apple Ⅱ 的输出转到接在该槽上的外设上。

5．RETURN键

打入每条命令或语句后，都必须按RETURN键。因为Apple Ⅱ以RETURN键的代码—8D作为命令或语句输入结束的标志。否则Apple Ⅱ无法知道一条命令或语句是否结束。

6．RESET键

当Apple Ⅱ的BASIC语言卡不工作时，按RESET键，使Apple Ⅱ自动转入磁盘启动程序。进入浮点BASIC。

当Apple Ⅱ的BASIC语言卡工作在整数BASIC时，按RESET键立即中断任何程序的执行，使Apple Ⅱ复位，并在整个屏幕窗口上置全部文本方式，并进入监控状态。按RESET键，不破坏现存的BASIC语言或机器语言程序。

7．REPT键

与其它键同时按，会以每秒10个字符的速率重复该键，若这时释放其它键或REPT键都可停止重复。

第三章　Apple Ⅱ 使用指南

在本章，将对Apple Ⅱ的使用和命令作一个基本的介绍，使读者对该机能有一个初步的认识，以便正确合理地使用该系统。对于读者需要深入了解的某些部分的内容，本章将随时指出其应参考的资料。

3.1　主机的启动

Apple Ⅱ的使用很方便。在接好外设后，打开位于主机后部左边的电源开关，键盘左下角的指示灯即亮，同时喇叭发出"嘟"的一声，在屏幕上方显示"APPLE]["，屏幕左下角出现提示符"]"，说明主机工作在"Apple soft"(浮点BASIC)方式。此时，用户就可以使用BASIC命令了。

若接好了磁盘驱动器，则先把系统主盘小心地插入驱动器，关上门，就可以打开主机电源开关。这时驱动器上的红灯变亮，盘片转动起来，在几秒钟内就可将磁盘的管理程序——操作系统（DOS）从磁盘调入内存，最后屏幕上出现提示符"]"。驱动器上的红灯熄灭，盘片停止转动。现在，用户不但可以使用BASIC命令，也可以使用操作系统命令了。

有关BASIC语言可参考《INTEGER BASIC REFERENCE MANUAL》、《APPLESOFT REFERENCE MANUAL》等。有关操作系统命令请参考《The DOS Manual》或《Apple微型计算机系统》下册磁盘操作系统。

3.2　键　　盘

Apple Ⅱ的大部分程序和语言都要通过键盘输入。这也是该机的基本输入方式。Apple Ⅱ有一个打字机式的52个键的键盘，采用美国标准信息交换码（ASCII码），绝大部分的ASCII代码可由键盘直接产生。另外还有两个换挡键（SHIFT），六个特殊键（CTRL、ESC、RESET、REPT、←、→）。这里简单介绍一下特殊键的功能与使用，详细内容请参考本书第二章第十节。SHIFT和CTRL键本身不产生键码，它们仅仅是改变其它键所产生的键码。若同时按下字符键和SHIFT键，就显示字符键的上部字符；若只按字符键，则显示该键的下部字符。同时按下CTRL键和字符键，产生的键码可作为具有控制功能的编码。这些键码（或称控制字符）不能在屏幕上显示。ESC键具有编辑功能，按ESC键后，Apple Ⅱ就进入编辑工作方式。此时有11个键具有特殊功能，如移动光标、改变屏幕显示内容等。REPT键称重复键，若单独按下该键，就会产生和上一次按的键相同的键码。若同时按下REPT和其它一个字符键，就会重复显示该字符。"←"和"→"键也具有编辑和修改功能。按一次"←"键，删除一个输入到计算机中的字

符。按一次"→"键，复制一个屏幕上光标所在位置的字符。RESET键称作复位键，按下此键（必须与CTRL键同时按下），一切处理都停止。释放后，计算机开始一个复原周期。只有当机器工作在非正常情况下，才使用RESET键，复原机器，重新开始工作。RETURN键被用来结束命令或语句的输入。记住，欲结束命令或语句的输入，就按RETURN键。

3.3 屏幕显示

在主机的后部右边有一个标有"VIDEO"的同轴电缆插座，用一根电缆将该插座与监视器连接起来，监视器就可显示所要求的信息了。

"VIDEO"插座输出的是全电视信号，它由同步信号SYNC、图象信号和副载波信号叠加而成。输出幅度可从0V调到1V（峰值）。

Apple Ⅱ的屏幕显示方式有三种：文本方式，能显示24行数字、符号或字母，每行40个，字符由7×5的点阵组成；低分辨率图形方式，能显示40×48个彩色方块，每个方块可有十六种颜色；高分辨率图形方式，能显示280×192的彩色点阵，共有六种颜色。通过程序或命令可以改变显示方式。

文本方式下显示的字符可以是正常的（黑底亮字）、反相的（亮底黑字）或闪烁的（正反相间），且屏幕上显示的区域可通过软件实现缩变。

视频显示的信息存储在系统RAM的专用区域中。一个存储单元的数据能控制屏幕相应位置的一个字符或两个彩色方块或一行上的七个点。在文本或低分辨率图形方式下，用1024个存储单元存储显示信息。在高分辨率图形方式下，用8192个单元。实际上，对于每种显示方式，都有两个存储区可以作为显示信息的来源，分别被称为第一页和第二页。

Apple Ⅱ显示的图象功能较强，该机的BASIC语言中扩充了独特的画图语句，这大大扩充了该机的应用范围。

有关显示存储区的地址、显示方式选择、视频信号发生器等的详细内容请参考本书第二章。

3.4 外存储设备

为了长期保存计算机存储器中的程序或数据，可将其存在磁带或磁盘中。显然，磁带或磁盘上的信息也可以取回到计算机中，而不必一次一次地从键盘输入。

一般的录音机都可用来存储程序。在计算机后部右方有两个插座，分别标有"IN"和"OUT"，"IN"表示磁带上的信息要从这个插座输入到计算机；"OUT"表示计算机内的信息要从这个插座送到录音机。用两根转录线，分别将插座"IN"、"OUT"与磁带录音机的插座"EAR"和"MIC"连接起来。

用SAVE命令，可将主机内存中的BASIC程序转储到磁带中，其步骤如下：

（1）在键盘上打入 SAVE；

（2）按下录音机上的PLAY和RECORD键；

（3）按RETURN键。

大约十秒钟左右，主机喇叭发出“嘟”的一声，表示此程序开始存储。当存储结束时主机会发出第二声“嘟”，屏幕上出现催问符。

用LOAD命令可将磁带上的BASIC程序装入内存。步骤如下：

（1）倒带到程序的开头；

（2）在键盘上打入LOAD。

（3）按下录音机的PLAY键。

（4）按RETURN键。

等待响过两声“嘟”之后，若屏幕上出现提示符，表明装入成功。若屏幕上出现错误信息或什么也没有，请检查录音机的音量、程序开头位置等，然后再重新操作一次。

在监控工作方式下，内存中的汇编程序或数据可用监控命令W写入磁带；磁带中的信息用R命令装入内存。有关监控命令的使用以及磁带的记录格式等内容，请参考本书第四章。

用磁带机存储信息所进行的操作不太方便，而且速度慢，目前最常用的外存储设备是软磁盘。把磁盘驱动器的电缆连到磁盘接口卡上，第一个驱动器的电缆插头应插在标有DRIVE1的插座上。然后将接口卡插入主机内的外设插座上，一般是插在6号插座。

开机引导磁盘操作系统的操作在前面已介绍过。如果开机自引导没有成功，请小心地检查一下系统是否装配好，试着重新启动。你也可以使用命令将磁盘操作系统装入内存。在BASIC方式下，引导DOS的命令是：

PR#s或IN#s（s是磁盘接口卡所在的插座号，s=1,…,7）

使用操作系统的显著特点之一，就是信息的存储和索取是根据文件名进行的，而且操作系统会自动高效地完成上述工作。把一个BASIC程序存到磁盘上的命令是带有文件名的SAVE命令，格式如下：

SAVE〔文件名〕

从盘上读一个BASIC程序的命令格式如下：

LOAD〔文件名〕

如果在上述两个命令中忘记输入文件名，系统将认为你准备和磁带交换信息。

操作系统还有很多有用的命令，请参考《Apple微型计算机系统》下册。那里将详细介绍磁盘的使用、操作系统命令的语法、功能和执行过程，以及磁盘接口卡的硬件工作原理等。

3.5 打印输出

Apple Ⅱ通过接口可以连接多种打印机。目前比较常用的打印机型号是MX-80（FX-80）。将打印机的扁平电缆接到打印机接口卡上，然后把打印机接口卡插在主机内的外设插座上，一般插在1号插座。

需要打印输出时，首先将打印机侧面的电开关拨向ON的位置，然后键入命令：

PR#s（s是打印机接口卡所在的插座号）

上述命令执行后，计算机送到监视器上的信息将同时在打印机上印出。例如执行LIST命令，内存中的程序清单不但列于屏幕，而且也被打印在纸上。

要退出打印状态，键入命令：

PR＃0

MX-80打印机一行可打印80个字符。只用PR＃s命令转去打印时，打印的行宽仅为40。若想扩大行宽，需再键入命令：

POKE1656＋s，L（s是打印接口卡所在的插座号，L是希望的行宽）

MX-80打印机的功能很强，为了充分发挥其功能，打印机接口卡上装有2K字节的管理程序。在该程序的控制下，可以实现放大、缩小、加密等打印方式，也可以具有位图象打印功能，这时打印机作为屏幕的硬拷贝设备，只要输入控制命令CTRL Q即可。

3.6 监控命令简表

Apple Ⅱ有一个高效率的控制程序，大约有2K字节，主要功能有：管理存储器，管理外部设备和对微处理机系统进行简单操作和运算。详细内容请参考本书第四章。这里仅列出命令简表，供读者快速查找。

监控命令简表

命令类别	命令符	实例	功能
检验存储器内容		1000.2000	检验(显示)指定存储区域的内容
修改存储器内容		2000:55	修改从指定存储器开始的存储器内容
转移存储器内容	M	1000<2000.3000M	把2000～3000存储区的内容转移到1000以后
核实存储器内容	V	1000<2000.3000V	核实两个存储区域的内容，有差别时，分别将两个存储器的地址和内容显示在屏幕上
磁带输入	R	300.380R	读磁带上的信息送指定的存储区域
输出到磁带	W	300.380W	把指定存储区的信息转储到磁带上
置显示方式	I		建立反的显示方式(亮底黑字)
	N		建立正常的显示方式(黑底亮字)
反汇编	L	1000L	列出从指定地址开始的20条指令的记忆码
执行指令命令	G	1000G	从指定地址开始执行机器指令，直到返回指令
	S	1000S	执行指定地址的一条机器指令
扫描追踪命令	T	1000T	从指定地址开始执行指令，直到断点，并在屏幕上显示A、X、Y、S、P的内容

续表

命令类别	命令符	实　例	功　能
启动小汇编		F666G	启动小汇编程序，屏幕将出现小汇编提示符"!"
控制字符命令	CTRL E①		显示A、X、Y、S、P的内容
	CTRL Y①		从特定地址$03F8开始执行用户确定的机器指令
	CTRL B		进入BASIC工作方式，清除现存的BASIC程序
	CTRL C		进入BASIC工作方式，不清除现存的BASIC程序
	CTRL G①		喇叭发出"嘟"声
	CTRL H①		光标退一格，并删除从计算机来的字符与"←"键具有相同的功能
	CTRL J①		仅发出换行信号
	CTRL U①		光标进一格，并复制所改写的字符，与"→"键具有相同的功能
	CTRL X①		删除刚刚键入的字符
置输入通道	S CTRL K	6CTRL K	设置S号外设插座上的外设为输入设备，S=1，…，7
置输出通道	S CTRL P	6CTRL P	设置S号外设插座上的外设为输出设备，S=1，…，7
十六进制运算命令	+	25+A5	完成十六进制加法，并显示计算结果
	−	A5−3E	完成十六进制减法，并显示计算结果
编辑字符	ESC I②		光标上移一行
	ESC M②		光标下移一行
	ESC J②		光标左移一格
	ESC K②		光标右移一格
	ESC D		光标上移一行
	ESC C		光标下移一行
	ESC B		光标左移一格
	ESC A		光标右移一格
	ESC E		从光标处开始清除原文到行末
	ESC F		从光标处开始清除原文到页末
	ESC @		清除整个屏幕，并将光标置于屏幕左上角

表注：①该命令之后不用按RETURN键。其它命令后都必须按RETURN键。
②该编辑命令适合于装有自启动监控ROM的机器。操作时，先按一下ESC键，再按相应字符键，实现其编辑功能，且此时不退出编辑状态。即，可连继进行编辑工作。若按了非编辑用字符键，才退出编辑状态。
另外三个编辑字符，每实现一次编辑功能都要先按一次ESC键。

3.7 整数BASIC命令简表

Apple Ⅱ整数BASIC程序是一个解释系统。它是一个简洁可靠的语言，特别适用于管理和控制。BASIC具有很灵活的对话功能，用户可通过命令使计算机接收输入程序、印出程序清单，并可以随时查找或修改程序，也可以在程序的执行过程中扫描变量的变化、检查程序的执行。用户还可以用命令来中断当前正在执行的程序，并显示停止的行数。解释系统还能检查程序中词法、语法及语义上的错误，并及时告诉用户，以便及时修改。

整数BASIC提供了八个函数。本语言还具有低分辨率作图的能力。

有关整数BASIC的详细内容可参考《INTEGER BASIC REFERENCE MANUAL》。

下面列出整数BASIC命令简表。

整数BASIC命令简表

类别	符　号	说　明
命令	AUTO	置自动行数方式
	CLR	清除现存的BASIC变量
	CON	继续执行由CTRL C所停止的程序
	DEL	删除程序
	DSP	置检查变量方式
	HIMEM	设置BASIC使用的最高内存地址
	GOTO	程序转移
	GR	置最大图形显示方式
	LIST	在屏幕上列出程序
	LOAD	从盒式磁带读BASIC程序
	LOMEM	置BASIC使用的最低内存地址
	NEW	清除现存的BASIC程序
	NO DSP	清除对变量的检查方式
	NO TRACE	清除程序执行的检查方式
	RUN	运行程序
	SAVE	存一个BASIC程序到盒式磁带上
	TEXT	置最大文本方式
	TRACE	置程序执行的检查方式
运算符	+	加
	−	减
	*	乘
	/	除
	↑	乘方
	=	赋值

续表

类型	符　号	说　明
运算符	#或<>	不等
	>	大于
	<	小于
	>=	大于或等于
	<=	小于或等于
	AND	逻辑与
	OR	逻辑或
	NOT	逻辑非
	MOD	余数
	( )	优先运算符
函数	ABS	求绝对值
	ASC	给出变量串的ASCII码的十进制数
	LEN	给出变量串的当前长度
	PDL	给出游戏电位器的位置
	PEEK	给出存储器内的十进制数
	RND	给出随机数
	SCRN	给出屏幕某位置的颜色
	SGN	给出表达式符号
语句	CALL	调用机器语言子程序
	COLOR=	确定彩色图形的颜色
	DIM	定义变量或变量串的长度
	END	停止执行程序
	FOR…NEXT	形成循环操作
	GOSUB	调用BASIC语言子程序
	HLIN	画一条彩色水平线
	IF…NEXT	条件转移语句
	INPUT	从I/O设备输入信息
	IN#	改变输入通道
	LET	赋值说明,LET是可选的
	PLOT	画一个彩色方块
	POKE	修改存储器内容
	POP	退出一个堆栈地址
	PRINT	输出信息
	PR#	改变输出通道
	REM	注释语句
	RETURN	BASIC子程序返回语句
	TAB	水平移动光标
	VLIN	画一条彩色垂直线
	VTAB	垂直移动光标

3.8 Applesoft BASIC (浮点BASIC) 快速参考指南

在Apple Ⅱ plus中，浮点BASIC解释程序被固化在ROM中。因此，开机后就可以使用浮点BASIC语言。这是一个功能非常强的可用来进行科学计算的语言。有效数字为9位，数字大小可达$\pm 10^{38}$。与整数BASIC语言相比：增加了实数类型的数、数组或变量串的维数，还增加了高分辨率作图的能力以及许多有用的命令。详细内容可参考《Applesoft BASIC Programming Reference Manual》。这里仅给出浮点BASIC快速参考指南：

简单变量

类型	名字	范围
实数	AB	±9.99999999E+37
整数	AB%	±32767
字符串	AB$	0到255个字符

A是字母，B是字母或数字。

数组变量

类型	名字及实例
实数	AB (3,12,7)
整数	AB% (3,12,7)
字符串	AB$ (3,12,7)

数组的规模应受有效存储器的限制

算术操作

= 变量赋值
— 负
∧ 指数
* 乘
/ 除
+ 加
- 减

关系或逻辑操作

= 等于
< > 不等于
< 小于
> 大于
<= 小于或等于
>= 大于或等于

NOT　逻辑“非”

AND　逻辑“与”

OR　逻辑“或”

关系式或逻辑表达式的值是1时，为真；值是0时，为假。关系算符还可用于字符串比较。

系统命令

LOAD　从磁带装入一个程序。

SAVE　把一个程序存到磁带上。

NEW　删除现有程序。

RUN　从标号最小的语句运行程序。

RUN 477　从标号为477的语句开始运行程序。

STOP　暂停程序的运行，并显示停止的行数。

END　停止程序的运行，不回答任何信息。

CTRL C　暂停程序的运行和列表。

CONT　继续执行由STOP、END或CTRL C停止的程序。

TRACE　置检查方式，例出程序执行的行数。

NOTRACE　退出TRACE的检查方式。

PEEK(X)　给出存储器X单元的内容。

POKEX,13　修改存储器内容，把13放在地址为X的存储单元中。

WAIT X, Y, Z　条件等待语句，直到地址X里的内容“或”Z之后，再“与”Y不等于0，才执行程序的下一个命令。

CALL X　调用开始地址为X的机器语言子程序。

USR (X)　传送X到机器语言子程序。

HIMEM:　置BASIC程序使用的最高内存地址。

LOMEM:　置BASIC变量使用的最低内存地址。

编辑或与格式有关的命令

LIST　列出全部程序

LIST X,Y　列出标号为X到标号为Y的程序段。

DEL,X,Y　删除标号为X到标号为Y的程序段。

REM XYZ　XYZ是程序注释。

VTAB Y　垂直移动光标到Y行 (Y=1,…,24)。

HTAB X　水平移动光标到位置X (X=1,…,40)。

TAB (X)　仅用在PRINT (打印) 语句中，水平移动光标到位置X(X=1,…,40)。

POS (0)　给出光标的当前水平位置 (0,…,39)

SPC (X)　仅用在打印语句中，放X个空格在两项打印之间。

HOME　清除屏幕，而且放游标在屏幕的左上角。

CLEAR　置所有变量到零

FRE (0)　给出用户可用的内存总量

FLASH	置计算机显示输出为闪烁
INVERSE	置计算机显示输出为亮底黑字
NORMAL	置正常显示输出
SPEED=X	置字符输出速率 (0～255)
ESC A	光标向右移动一格
ESC B	光标向左移动一格
ESC C	光标向下移动一格
ESC D	光标向上移动一格
→	输入光标所在位置的字符，光标右移一格。
←	删除光标所在位置的字符，光标左移一格。
CTRL X	清除正在输入的一行。

数组和变量串

DIM A (X, Y, Z)	确定数组A的最大下标，为 (X+1)·(Y+1)·(Z+1) 个实元素保留存储单元。
DIM A$ (X, Y)	确定A$的最大下标，它可以存 (X+1)·(Y+1) 个字符串元素，每个字符串的长度可多达255个。
LEN (A$)	给出A$中字符的个数。
STR $ (X)	把X变换成一个以数值出现的字符串。
VAL (A$)	把A$当作数，给出它的值，直到第一个非数字字符。
CHR$ (X)	给出相应的ASCII字符。
ASC (A$)	给出A$第一个字符的ASCII码。
LEFT $ (A$, X)	给出A$中X字符左面的字符。
RIGHT $(A$, X)	给出A$中X字符右面的字符。
MID $(A$, X, Y)	给出A$中从X算起的第Y个字符。
STORE A	存一个数值数组A到磁带上
RECALL B	从磁带上读回一个数组，数组B必须是已经正确定义过的。

输入/输出命令

INPUT S$	在屏幕上显示"？"，等待用户键入字符串S$。
INPUT"XYZ"; A	在屏幕上显示XZY，等待用户键入实数A的值。
GET A$	等待用户键入一个字符的值，不需要按RETURN键。
DATA X,"Y",Z	建立一个数据表，这个表可用READ语句来读。
READ A	将DATA表中的下一个数据赋值给变量A。
RESTORE	再一次准备从DATA表中第一个数据开始读数据。
PRINT"X="; X	在屏幕上显示X=和X的值。
IN#S	置S号外设插座上的外设为输入设备 (S=1,…,7)。
PR#S	置S号外设插座上的外设为输出设备 (S=1,…,7)。
LET X=Y	将Y的值赋给X
DEF FNA (X)=X+28/X	定义函数FNA

在以后的使用中，FNA的变量将被定义过的表达式X所替代。例如FNA(4)=11

控制语句命令

GOTO 347　转移到347语句

IF X=3 THEN STOP　如果X=3是真执行STOP命令；X=3是假，程序跳到下一条语句。

FOR X=1 TO 20 STEP 4…NEXT

执行从FOR到相应的NEXT之间的语句，直到X>20。

第一次X=1，每执行一次NEXT语句，X增加4。若没有指定STEP的大小则STEP为1。

NEXT X　定义FOR…NEXT循环的底，X是可选的

GOSUB 33 调用从33行开始的子程序

RETURN　由GOSUB调用的子程序的结束标志

POP　退出一个堆栈返回地址

ON X GOTO 397，12，458　跳到由X指定的GOTO行数，若X=3程序转到458行。

ON X GOSUB 397，12，458 调用由X指定的子程序。若X=2程序调用起始行数为12的子程序。

ONERR GOTO 4500　若程序有错误，跳到错误处理子程序，起始行数为4500。

RESUME　错误处理程序的返回语句。

画图和游戏控制

低分辨图形

GR　置低分辨率图形方式，屏幕下方可显示四行文本，清屏幕到黑。

COLOR=X　置彩色（X=0，…，15）。

PLOT X，Y　画一个彩色块，水平位置在X（X=0，…，39），垂直位置在Y（Y=0，…，39）。

HLIN X1，X2 AT Y　从X1，Y点到X2，Y点画一条水平线。

VLIN Y1，Y2 AT X　从X，Y1点到X，Y2点画一条垂直线。

SCRN（X，Y）　给出屏幕上X，Y点的彩色。

高分辨率图形

HGR　置高分辨率图形第一页，屏幕下方可显示四行文本，清屏幕到黑。

HGR2　置高分辨率图形第二页，清屏幕到黑。

HCOLOR=X　置颜色（X=0，…，5）。

HPLOT X，Y　画一个彩色点，水平座标为X，垂直座标为Y；

X=0，…，279，Y=0，…，159（HGR）或Y=0，…，191（HGR2）

HPLOT X1，Y1 TO X2，Y2　画一条线，从X1，Y1点到X2，Y2点

SHLOAD　从磁带上装一个图形表。

DRAW S AT X，Y　按预先装入的S号图形表画图，起始点在X，Y，颜色由HCOLOR确定。

XDRAW S AT X,Y　按S号图形表画图，每一点的颜色都是该点颜色的补码。

ROT＝X　旋转XDRAW或DRAW图形，ROT＝0是垂线；ROT＝16顺时针旋转90°；ROT＝32顺时针旋转180°。

SCALE＝X　置XDRAW或DRAW的比例（X＝1,…,255）。

PDL（X）　给出表示X号游戏电位器位置的数从0到255（X＝0,…,3）

PEEK（X－16287）　若X号游戏开关被按住，给出的数值大于127。（X＝0,…,2）

PEEK（－16336）　喇叭发声。

函数

SIN（X）　给出弧度X的正弦值。

COS（X）　给出弧度X的余弦值。

TAN（X）　给出弧度X的正切值。

ATN（X）　给出弧度X的余切值。

INT（X）　给出小于或等于X的最大整数。

RND（1）　给出从0到0.999999999之间的随机数。

RND（0）　给出上一次得到的随机数。

RND（－3）　给出4.48217199E－8，对于每个不同的负自变量得到一个不同的固定的数，这之后，具有正自变量的RND将遵循一固定的顺序。

SGN（X）　若X＜0，则给出－1；
X＝0，则给出0；
X＞0，则给出1。

ABS（X）　给出X的绝对值

SQR（X）　给出X的正的平方根

EXP（X）　给出e（2.71828）的X次方

LOG（X）　给出X的自然对数值

3.9 Apple Ⅱ plus 地址空间分配与专用地址表

表A　地址空间的分配

地址区域	存储器	用处
0000～BFFF	RAM	系统专用、堆栈、键盘缓存区、显示存储器、用户使用区
C000～CFFF	I/O	外设设备码、输入/输出、设备选择设备码、外设扩展ROM
D000～FFFF	ROM	Applesoft BASIC或整数BASIC解释程序、监控程序

表B　RAM的结构和使用

地址范围	用　　处	地址范围	用　　处
0000～00FF	系统程序	0800～0BFF	文本/低分辨率图形显示第二
0100～01FF	系统堆栈	0C00～01FF	页用户使用空间
0200～02FF	键盘输入缓存区	2000～3FFF	高分辨率图形显示第一页
0300～03FF	监控向量地址	4000～5FFF	高分辨率图形显示第二页
0400～07FF	文本/低分辨率图形显示第一页	6000～BFFF	用户使用空间

注：在同一时间，Apple只占用一组显示地址，其余显示地址可供用户使用。

表C　监控使用0页存储器表

	低　位															
地址	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00																
10																
20	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
30	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
40	•	•	•	•	•	•	•	•	•	•					•	•
50	•	•	•	•	•	•										
60																
70																
80																
90																
A0																
B0																
C0																
D0																
E0																
F0																

表D　整数BASIC使用0页存储器表

地址	低位 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00																
10																
20																
30																
40											•	•	•	•		
50						•	•	•	•	•	•	•	•	•	•	•
60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
D0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
E0																
F0																

表E　Applesoft BASIC使用0页存储器表

地址	低位 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	•	•	•	•	•	•					•	•	•	•	•	•
10	•	•	•	•	•	•	•	•	•							
20																
30																
40																
50	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
D0	•	•	•	•	•	•			•	•	•	•	•	•	•	•
E0	•	•	•		•	•	•	•	•	•	•					
F0	•	•	•	•	•	•	•	•	•							

表F 向量表

地 址	用 处	地 址	用 处
03F0 03F1	保存BRK中断的服务程序入口	03F8 03F9 03FA	保存一条"JMP"指令，指令中的地址为处理CTRL Y命令的子程序入口
03F2 03F3	软件入口向量	03FB 03FC 03FD	保存一条"JMP"指令，指令中的地址是处理NMI中断的服务程序入口
03F4	电源建立字节	03FE 03FF	保存IRQ中断的服务程序入口
03F5 03F6 03F7	保存一条"JMP"指令		

表G 输入/输出设备码

地 址	用 处
C00X	取键盘数据
C01X	清除键入标志
C02X	盒式磁带输出
C03X	扬声器输出
C04X	公用选通
C050	置图形显示方式
C051	置文本显示方式 TEXT MODE
C052	置全部文本或全部图形方式
C053	置文本和图形混合方式 MIX MODE
C054	置显示第一页
C055	置显示第二页 PAEG2
C056	置低分辨率图形方式
C057	置高分辨率图形方式 HIRES
C058	置0号一位输出为0， AN0＝0
C059	置0号一位输出为1， AN0＝1
C05 A	置1号一位输出为0， AN1＝0
C05B	置1号一位输出为1， AN1＝1
C05C	置2号一位输出为0， AN2＝0
C05D	置2号一位输出为1， AN2＝1
C05E	置3号一位输出为0， AN3＝0
C05F	置4号一位输出为1， AN3＝1
C060 C068	选通磁带输入

续表

地　址	用　处
C061 C069	选通0号游戏开关 SW0
C062 C06A	选通1号游戏开关 SW1
C063 C06B	选通2号游戏开关 SW2
C064 C06C	选通0号游戏电位器 PDL0
C065 C06D	选通1号游戏电位器 PDL1
C066 C06E	选通2号游戏电位器 PDL2
C067 C06F	选通3号游戏电位器 PDL3
C07X	触发游戏控制器
C08X	0号外设插座的设备选择地址
C09X	1号外设插座的设备选择地址
C0AX	2号外设插座的设备选择地址
C0BX	3号外设插座的设备选择地址
C0CX	4号外设插座的设备选择地址
C0DX	5号外设插座的设备选择地址
C0EX	6号外设插座的设备选择地址
C0FX	7号外设插座的设备选择地址
C1XX	1号外设插座的I/O地址
C2XX	2号外设插座的I/O地址
C3XX	3号外设插座的I/O地址
C4XX	4号外设插座的I/O地址
C5XX	5号外设插座的I/O地址
C6XX	6号外设插座的I/O地址
C7XX	7号外设插座的I/O地址
C800～CFFF	外设接口板扩展ROM地址

第四章　监控系统分析

4.1　概　　述

Apple监控系统程序放在ROM中，从F800～FFFF，约占2K内存单元，它的主要作用是：

（1）管理存储器：

检验、改变、移动、核实存储器内容。

（2）管理外部设备：

管理键盘输入、字符显示器输出、盒式磁带录音机输入和输出、打印机输出等多种外部设备。

（3）对微处理机系统进行简单的操作运算：

按给定地址起动运行和单步执行机器指令，停止运行机器指令，多重命令执行，进行简单的十六进制运算等。

通过分析，我们感到Apple监控系统结构比较紧凑、灵活，使用起来效率比较高，这在下面的分析中就将可以看出。

4.2　Apple监控命令介绍

在下面介绍的监控命令中，请注意如下问题：

“地址（adrs）”是一个四位十六进制的数字，而“数据(data)”是一个二位十六进制的数字；

在每一条命令的末尾要按一下〈RETURN〉键（回车），标志着命令结束。机器接收到〈RETURN〉键的键码3D，就去执行这条命令。

Apple微机监控命令如表4.1所示。

另外还有一些专门的控制字符和编辑字符。

当机器处于监控程序或BASIC程序时，这类命令都能执行。

控制字符是用某些字母加一个上标“C”来表示的，如G^C、B^C等。在按下〈CTRL〉键的同时，敲打所指定的字母，就可以得到控制字符的命令。控制字符是不能在屏幕上显示的。请注意：在B^C和C^C后面必须跟着按一次〈RETURN〉键。

屏幕的编辑字符是用某些字母加一个下标“E”来表示，如A_E、D_E等。按下〈ESC〉键，松开，然后敲打指定的字母就可以得到相应的编辑字符命令。编辑字符仅把信息送给显示荧光屏，而不送数据到存储器，例如编辑字符A_E是将游标右移一格，而不复制原文，它与控制U（U^C）不同，U^C除将游标右移外，并复制原文，注意二者的区别。

表 4.1 APPLE微机监控命令

命令类别及格式	实　例	说　明
(一)检验存储器		
地址(adrs)	*40F2	检验（显示）单一存储单元(如40F2)内容
地址1·地址2	*1024·1048	检验(显示)从地址1(如1024)到地址2(如1048)存储区域的内容
单按〈RETURN〉键	*〈RETURN〉	检验(显示)下面8个存储器单元的内容
·地址2	*·4096	检验(显示)从现行的存贮单元到地址2(如4096)之间存储单元的内容
(二)改变存储器内容		
地址:数据␣数据	*A256:EF␣20	从给定地址单元(如A256)开始将数据（如EF、20)依次存放到存储器中 "␣"表示空格,以下同。
数据␣数据	*F0␣A2	从上次用来存放数据的地址之后开始,将数据(如F0、A2)存入存储器中。
(三)转移存储器内容		
地址1<地址2·地址 3M	*1000<B010·B410M	把地址2到地址3区域内的当前数据复制到从地址1开始的存储单元中。
(四)核实存储器内容		
地址1<地址2·地址 3V	*1000<B010·B410V	核实从地址2到地址3区域内的数据块与从地址1开始的数据块是否完全相等,若有不等,就显示出来。
(五)磁带机输入/输出		
地址1·地址2R	*3100·4FF0R	读磁带机内的数据进入所规定的存储器地址区域内,记录长度必须和存储器区域相等,否则就出错。
地址1·地址2W	*8400·9FFFW	将指定的存储器区域内的数据写到磁带上
(六)显示方式:		
I	*I	建立倒的电视方式(白底黑字)
N	*N	建立通常的电视方式(黑底白字)
(七)反汇编程序		
地址 L	*C100L	将指定地址(如C100)开始的20条指令翻译成6502的汇编记忆符,并显示出来
L	*L	将当前存贮器地址开始的下面的20条指令翻译成汇编记忆符,并显示出来
(八)启动小汇编		
F666G	*F666G	起动小汇编,APPLe用"!"回答
按CTRL—RESET键	! CTRL—RESET	退出小汇编程序并返回到监控系统
(九)监控执行和调整程序		
地址G	*3000G	从指定地址(如3000)开始启动机器程序
地址T	*8000T	从指定地址(如8000)开始连续扫描追踪程序,直到击中一个断点为止,断点发生在指令 00处(BRK),然后返回到监控程序,并打开6502状态寄存器
(新监控无此命令)		〔请看注意事项(一)〕

续表

命令类别及格式	实　例	说　明
地址S (新监控无此命令)	*50C0S	从指定地址(如50C0)开始单步(单条)执行程序，每按一次S，就执行一条指令，并打开6502状态寄存器，显示其内容，以供参考〔参看注意事项(一)〕
(控制E)	*E^{C}	(同时按下CTRL键和E键)，打开6502状态寄存器，并显示其内容，以供参考。
(控制Y)	*Y^{C}	从规定的存贮器单元(03F8)开始执行用户所确定的机器语言子程序。
(十)十六进制的算术运算：		
数据1＋数据2	*78＋34	实现数据1和数据2的十六进制加法，并显示结果(例子中计算结果为AC)
数据1－数据2	*AE－65	实现数据1和数据2的十六进制减法，并显示结果(例子中的计算结果为49)
(十一)置输入/输出通道：		
(X)(控制P)	*5P^{C}	设置打印机输出的I/O槽号。(如X＝5，则置打印机输出槽号为5)
(X)(控制K)	*2K^{C}	设置键盘输入的I/O槽号。(如X＝2，则置键盘输入槽号为2)
(十二)多重命令	*100L␣200G	如果用一个空格分隔不同的监控命令，多重的命令就可放在同一行上。
	*LLL	单一字母的监控命令可以重复，无需空格

现将〈RESET〉键、控制字符和编辑字符的作用归纳如下：

〈RESET〉键：

Apple微机分Apple Ⅱ和Apple Ⅱ plus，Apple Ⅱ是Apple Ⅱ plus的前身，目前国内使用的大都是Apple Ⅱ plus，两种机器对〈RESET〉键的使用方法及功能略有区别。

对于Apple Ⅱ，按〈RESET〉键将立即中断任何程序的执行，将计算机复位，并能在最大屏幕窗口置全部文本方式，机器进入监控系统程序，Apple Ⅱ用一个“*”和“嘟”声回答。按〈RESET〉键并不破坏现存的BASIC或机器语言程序。

对于Apple Ⅱ plus，需将CTRL键与RESET键连用，而且在按CTRL-RESET时，通常是进入浮点BASIC，屏幕上将出现提示符“〕”。

控制B (B^{C})：

当机器处于监控系统中(用“*”号应答)时，敲控制B和〈RETURN〉将使机器进入BASIC程序中，并置存储器使用区域为：最高到现存的存储器最高单元，最低到2048。

控制C (C^{C})：

当机器处在BASIC程序中时，按控制C和〈RETURN〉后，程序就暂停，并显示暂停处的行数字，程序可用一个CON命令继续执行下去。如果机器处在监控程序中，按

控制C和<RETURN>将使机器在不破坏现行程序下进入BASIC程序。

控制G（G^C）：

响铃（喇叭发出"嘟"声）。

控制H（H^C）：

游标返回，并从计算机内（不是从荧光屏上）删去多写的字符。Apple的键盘右边有一个标以"←—"的键，专门用来左移，在不利用控制字符H^C命令时，"←—"键也提供同样功能。

控制J（J^C）：

仅发生换行信号。

控制U（U^C）：

与H^C相反，游标右移并复制所改写的字符，Apple键盘右边有一个标以"—→"的键，供右移用，其功能与控制U相同。

控制X（X^C）：

立即抹去当前一行。

编辑字符：

A_E，K_E　将游标移到右边。

B_E，J_E　将游标移到左边。

C_E，M_E　将游标移到下面。

D_E，I_E　将游标移到上面。

E_E　从游标处清除原文到行末。

F_E　从游标处清除原文到页末。

$@_E$　游标移到页的顶端，清除整个页面的原文。

注意事项：

（1）6502状态寄存器被打开，无论什么时候在屏幕上显示的都是最近的一次结果。为了修改它们，敲打"：数据␣数据<CR>"，这些数据就提供给每个寄存器了。

例如，状态寄存器A＝3C，X＝EF，Y＝00，P＝32，S＝F2，

*：FF　仅把FF赋给A。

*：FF␣00␣33　把FF，00，33分别赋给A、X和Y寄存器。

若只改变S寄存器，你必须首先重复敲打A、X、Y和P的数字，然后敲打赋给S的数值。

（2）用这种方法仅仅可选1～7号槽口，不能选0号槽口（$0P^C$或$0K^C$就使机器重新恢复到固有的电视显示和键盘通道），而且只有当Apple接口卡所插槽号与命令中所用槽号一致时，这些命令才被执行。

4.3　监控系统主程序框图及说明

Apple Ⅱ和Apple Ⅱ plus的监控程序略有区别，进入监控系统的方法也略有不同。Apple Ⅱ的起动向量是FF59，开机和按RESET键均直接进入监控系统，从BASIC

语言程序执行一条CALL-151命令也能进入监控系统，主程序粗框图和细框图分别见图4.1和图4.2。

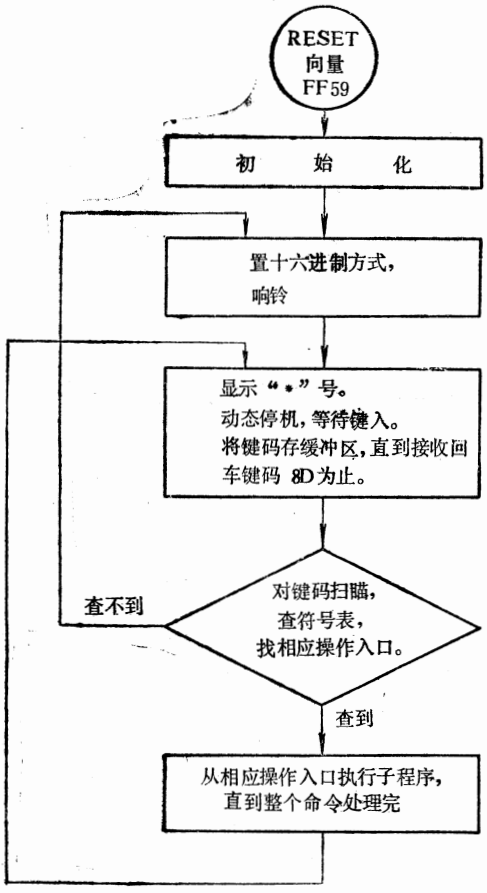


图 4.1　监控系统粗框图

Apple Ⅱ plus 的起动向量是 FA62，开机时不能直接进入监控系统，而是搜索和往主机调磁盘操作系统，这段程序与监控系统关系不大，本文称它为磁盘起动程序，我们把它放在本章最后进行分析。Apple Ⅱ plus通常是在BASIC语言程序下执行一条CALL-151命令进入监控系统。

图4.2的⑦号框右边有一个圆圈标着“CALL-151 入口”，指出了执行这一命令进入监控的入口地址，这个地址是FF69.

从FF59起动的监控程序常称为“老监控”，与此对应，从FA62起动的监控程序称为“新监控”。新监控和老监控在进入监控系统以后的主程序框图结构是一样的，也就是图4.2中的第⑦框到⑮框，至于这几个框中的具体内容，对于两种监控系统来说则不完全一样。下面从上将简单分析从FF59起动的老监控主程序，这样的好处是对监控系统有一个比较完整的概念，在具体分析每一框内容时将以新监控内容为主。整个分析方法对两种监控都是适用的。

下面将有次序地对图 4.2 细框进行解释。

①～④号框初始化。

敲〈RESET〉键进入监控系统。要进行初始化工作，初始化内容包括如下：

（1）置显示器方式

显示器的工作方式有两种，一种是屏幕背景是黑的，字符是亮的，这叫正常工作方式；另一种是背景是亮的，字符是暗的，这叫反工作方式，在程序中具体做法如下：

正常工作方式：FF→32

非正常工作方式：3F→32

（2）置文本方式，初始化显示屏幕窗口

用设备码C056设置高分辨率（HIRES）图形，用设备码C054设置显示器初始页（PRIMARY），用设备码C051设置文本方式（Text mode）。显示窗口最大面度是24行，每行40个字，在监控系统中，对窗口有编辑功能，上下左右以及游标的位置都可以设置，在初始化程序中设置为最大窗口：

0⟶22（窗口顶），

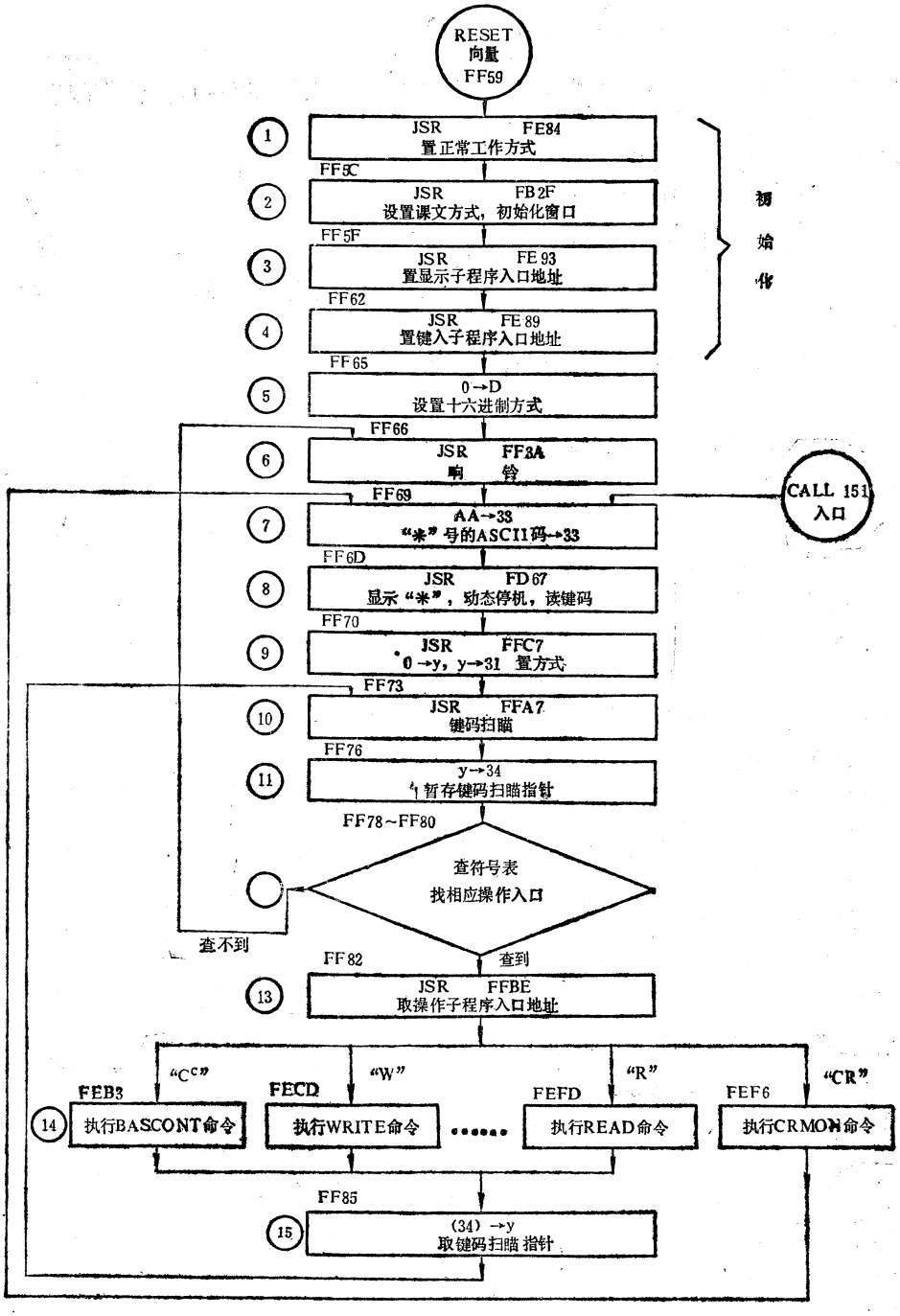


图 4.2 监控系统主程序细框图

0⟶20（窗口左），

28⟶21（窗口右），

18⟶23（窗口底），

17⟶25（游标垂直位置CV）。

此外，还有状态字清0：0⟶48；基本地址计算：根据CV＝17，计算结果得：

地址低位＝D0　　地址高位＝07

07D0为屏幕最末行最左边字符所对应的内存地址。

（3）置显示子程序入口地址，作法如下：

F0→36，　FD→37

FDF0为显示子程序入口，从程序结构的角度，这相当于把0号通道设置为字符显示器输出。

（4）置键入子程序入口地址，作法如下：

1B→38，FD→39

FD1B为键盘输入子程序入口，从程序结构的角度，这相当于把0号通道设置为键盘输入。

（5）号框　置十六进制工作方式，因为监控命令都是以十六进制方式出现。

（6）号框　喇叭发出"嘟"声（响铃）。

（7）号框　"＊"号的ASCII码⟶33

Apple用以传送字符的编码（从键盘输入的键码和系统送往显示器、打印机的字符编码等）是把ASCII码最高位加1。"＊"的ASCII码是2A，最高位加1就成AA，因此AA就是Apple用以传送"＊"号的编码，本文亦称其为ASCII码。

（8）号框　显示催问符（如"＊"、"！"等）；动态停机，等待键入，把键码如实送到0200开始的缓冲区，直到接收〈RETURN〉键码8D为止。

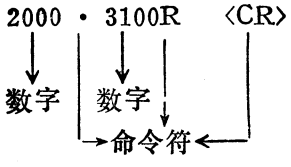
（9）号框　置方式，0⟶31，监控程序执行过程中，31单元的内容是可变化的，程序将根据31单元的内容做不同处理和分支。

（10）～（12）号框　对缓冲区键码扫描，换码、查符号表。

从0200开始，对缓冲区键码逐一扫描，若是数据，经换码后送有关单元中，若是命令符，经换码后就去查符号表，以便转到相应子程序去执行（详见第4.4节）。

（13）号框　取操作子程序入口地址（详见第4.4节）。

（14）～（15）号框　一条监控命令往往需要执行好几次子程序操作才能完成，例如命令



包含二处数字，三处命令符，系统每扫描到一个命令符就要执行一次子程序操作，扫描到"."就转到FE18子程序去处理，扫描到"R"就转到FEFD子程序去处理，等等，直到处理完"CR"码才表明整个命令处理完。

4.4 介绍几段子程序

1. 读键码子程序FD67

程序功能：

（1）显示催问符，如“*”、“!”等。

（2）动态停机，等待键入。

（3）除编辑字符外，每敲进任何一个键（包括空格和控制字符），程序都如实把键码取来并存到缓冲区，缓冲区在0200～02FF，共256个单元。

（4）当一条命令的字符数≥249个时，喇叭发出"嘟"声，以示警告。

（5）当一条命令的字符数到256个时，命令超长，认为无效，并显示“\”。

读键码子程序如下所示：

```
        FD3D    (32)→A          }取显示方式压入堆栈
        FD3F    A↓              }
        FD40    FF→A            }置通常的显示方式(黑底白字)
        FD42    A→32            }
        FD44    (0200+(x))→A 从缓冲区取键码
        FD47    JSR  FDED  显示
        FD4A    A↑              }恢复原来显示方式
        FD4B    A→32            }
        FD4D    (0200+(x))→A 从缓冲区取键码
        FD50    A-88       88为“退格”键码
        FD52    BEQ  FD71  是“退格”转
        FD54    A-98       98为“X^c”键码，X^c是用于抹去当前一行
        FD56    BEQ  FD62  是“X^c”转
        FD58    X-F8
        FD5A    BCC  FD5F  字数不到$F8 (249) 转
        FD5C    JSR  FF3A  响铃，字数≥249时响铃以示警告
        FD5F    X+1→X
        FD60    BNE  FD75  字数未到256转
        FD62    DC→A       DC为“\”键码
        FD64    JSR  FDED  显示“\”，抹去当前一行
入口→FD67       JSR  FD8E  回车、换行
        FD6A    (33) →A         }显示催问符，如“*”、“!”等
        FD6C    JSR  FDED       }
        FD6F    1→X        X为缓冲区指针
        FD71    X→A
        FD72    BEQ  FD67  结果为0转
        FD74    X-1→X
```

```
FD75    JSR  FD35    取键码
FD78    A—95         95为“进格”键码
FD7A    BNE  FD7E    不是“进格”转
FD7C    ((28,29)+y)→A
FD7E    A—EO
FD80    BCC  FD84
FD82    A.DF→A
FD84    A→0200, X
FD87    A—8D         8D为回车键码，命令结束符
FD89    BNE  FD3D    不是回车转
FD8B    JSR  FC9C    从游标处清0到行末
FD8E    8D→A
FD90    BNE  FDED    “条转指令”在此当“无转指令”用
FDED    JMP  (36)    转屏幕显示，36、37存的是显示子程序入口地址FDF0
FD2F    JSR  FD0C    取键码
FD32    JSR  FBA5    判是否是编辑字符
FD35    JSR  FD0C    取键码
FD38    A—9B         9B为〈ESC〉键码
FD3A    BEQ  FD2F    是〈ESC〉键码转
FD3C    RTS
FD0C    (24) →y      }取屏幕字符
FD0E    ((28,29)+y)→A}
FD10    A↓
FD11    A.3F→A       }
FD13    ORA  40      }最高两位变为01，出闪烁
FD15    A→(28), y    }
FD17    A↑
FD18    JMP  (38)    转键入子程序，38、39放的是键入子程序入口地址FD1B
FD1B    INC  4E      }
FD1D    BNE  FD21    }4E,4F为随机数地址，此处无用，求随机数时用到
FD1F    INC  4F      }
FD21    BIT  C000    选通键入数据
FD24    BPL  FD1B    最高位为0,无键入
FD26    A→(28), y    替换屏幕上闪烁信号
FD28    LDAC000      取键码
FD2B    BIT  C010    发清0脉冲，清键入标志
FD2E    RTS
FB97    SEC
FB98    JMP  FC2C    转屏幕编辑（见下面屏幕显示与编辑程序）
FB9B    A→y
FB9C    (FA48+y) →A
FB9F    JSR  FB97
FBA2    JSR  FD0C    取键码
FBA5    A—CE         }判是否是编辑字符IE、jE、KE、ME。
FBA7    BCS  FB97    }
FBA9    A—C9         }IE：游标上移
FBAB    BCC  FB97    }jE：游标左移
FBAD    A—CC         }KE：游标右移
FBAF    BEQ  FB97    }ME：游标下移
FBB1    BNE  FB9B
```

读这段程序时请注意FD32这条指令，AppleⅠ和AppleⅠplus是不一样的.如下所示：

Apple Ⅰ　FD32 JSR FC2C

```
AppleIIplus   FD32 JSR   FBA5
                编辑入口：FC2C
                FC2B   RTS
                FC2C   A⊕C0                    @E   清屏幕
                FC2E   BEQ FC58
AF 游标右移      FC30   A+FD+C→A                C=1
←————————      FC32   BCC  FBF4               判是AE、BE否
BF 游标左移      FC34   BEQ  FC10
←————————      FC36   A+FD+C→A                C=1
CE 游标下移      FC38   BCC  FC66               判是CE、DE否
←————————      FC3A   BEQ  FC1A
DF 游标上移      FC3C   A+FD+C→A                C=1
←————————      FC3E   BCC  FC9C               判是EE、FE否
EE 从游标处      FC40   BNE  FC2B
←————————
   清零到行末    FC42   (24)→y                  从游标处清零到页末
                FC44   (25)→A
                FC46   A→堆栈
                FC47   JSR  FC24
                FC4A   JSR  FC9E
                FC4D   0→y
                FC4F   堆栈→A
                FC50   A+0+C→A
                FC52   A-(23)
                FC54   BCC  FC46
                FC56   BCS  FC22
                FC58   (22)→A
                FC5A   A→25
                FC5C   0→y
                FC5E   y→24
                FC60   BEQFC46
```

2、屏幕显示与编辑程序

本程序具有如下功能：

（1）对可显示字符，立即在屏幕上予以显示。

（2）对不可显示字符，如回车、换行、退格、响铃等，程序立即执行其相应功能。

（3）对如下单次编辑字符

A_E——游标右移　　B_E——游标左移

C_E——游标下移　　D_E——游标上移

E_E——从游标处清到行末

F_E——清屏幕

程序予以一次编辑。所谓一次编辑（以游标左移为例）即为：每敲一次〈ESC〉键和B键，游标左移一次。

（4）对如下多次编辑字符

I_E——游标上移　　J_E——游标左移

K_E——游标右移　　M_E——游标下移

程序予以多次编辑。所谓多次编辑即为：敲了一次〈ESC〉键后，可连续敲I、J、K、M，

其功能均能实现，无需每次敲〈ESC〉键

编辑字符仅对屏幕起编辑作用，不改变缓冲区内容。

下面介绍几个专用地址：

(20)——窗左（初值00）

(21)——窗右（初值28）

(22)——窗顶（初值00）

(23)——窗底（初值18）

(24)——水平计数

(25)——垂直计数

显示子程序入口是FDF0，下面予以介绍：

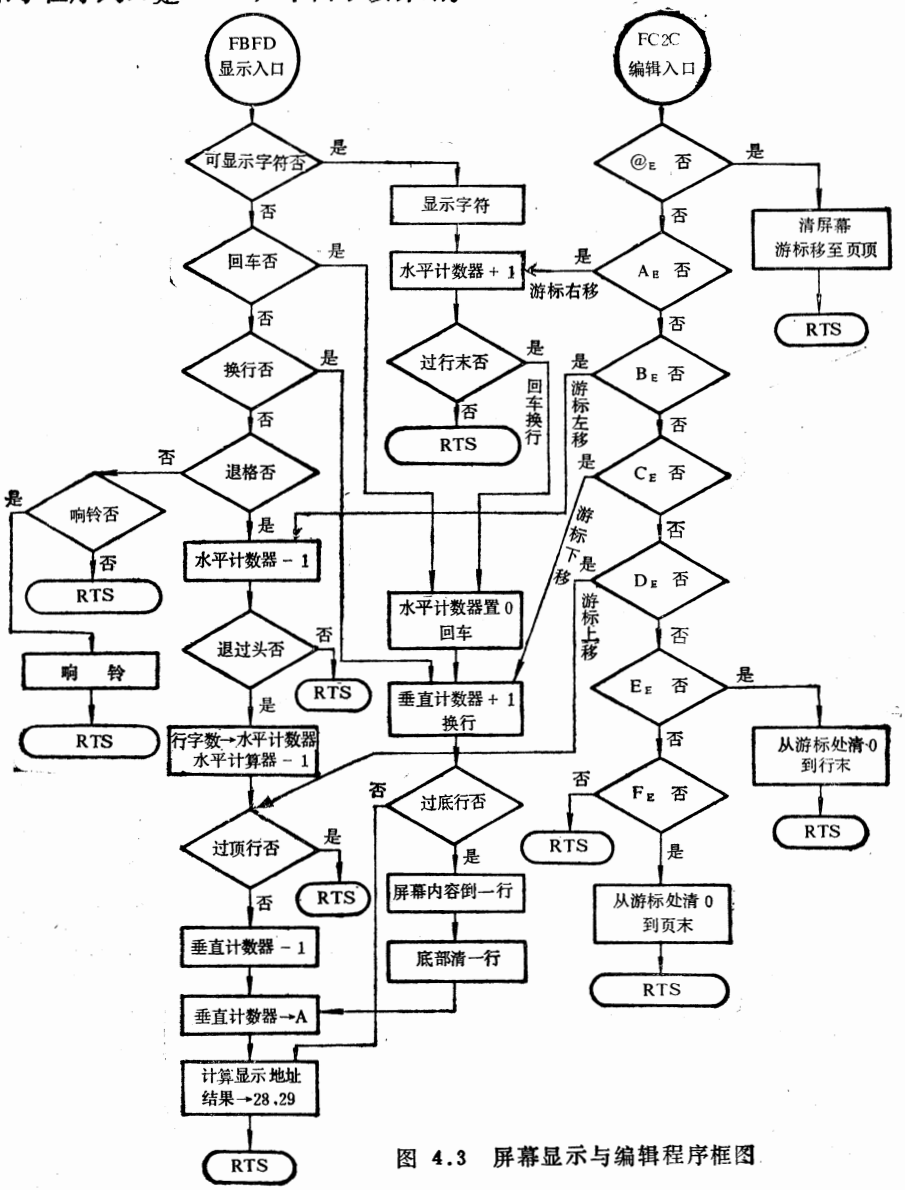
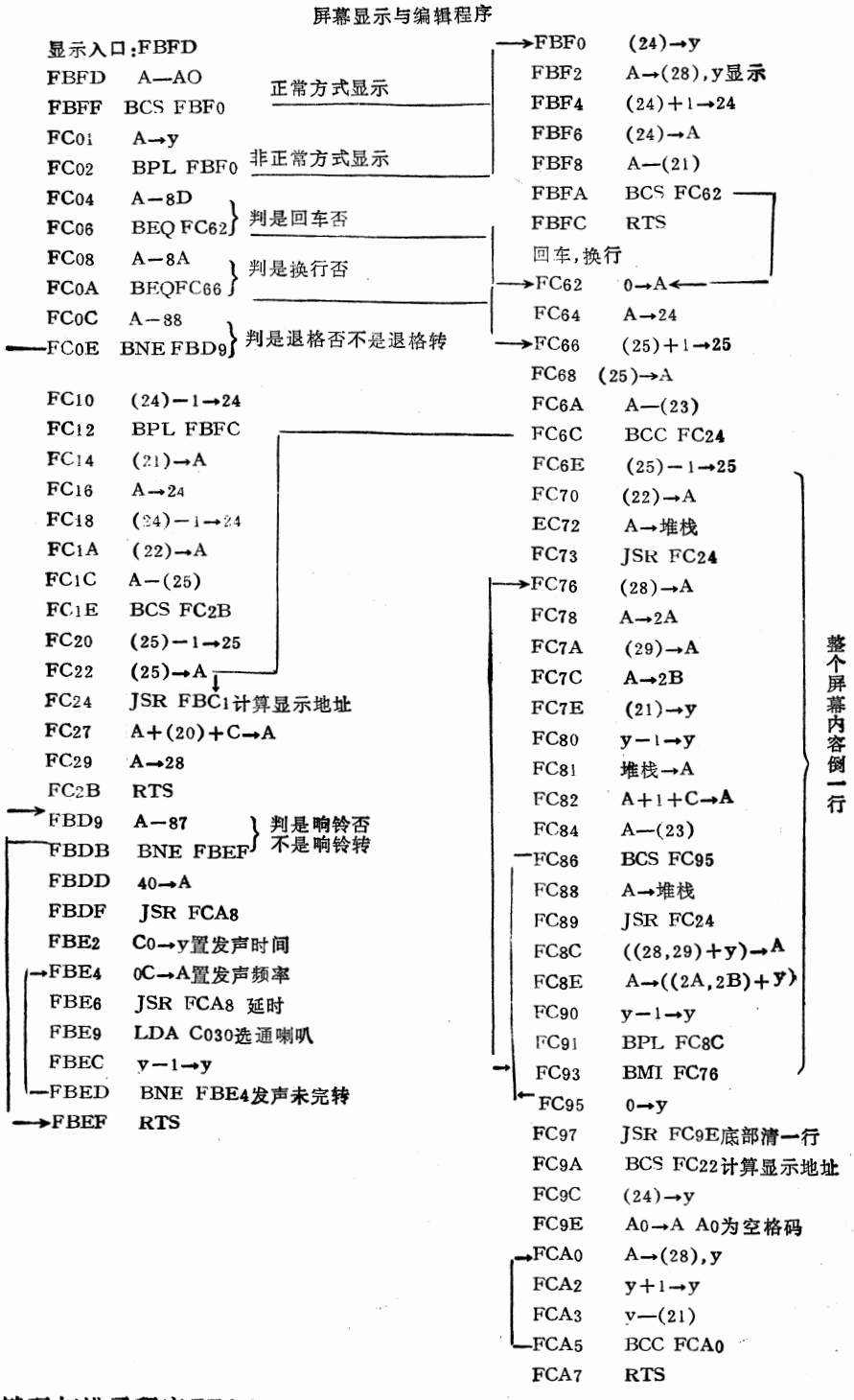


图 4.3 屏幕显示与编辑程序框图

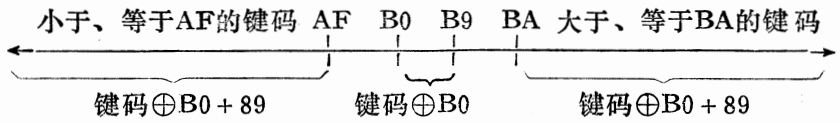


3. 键码扫描子程序FFA7

程序功能：

（1）对缓冲区键码扫描并换码

如前所述，从键盘敲进去的键码通过FD67子程序如实存到缓冲区，这段程序就是以缓冲区首地址0200开始，逐一顺序扫描，并进行换码，换码规律如下所示：



做这样换码的好处是：对于0～15的数字，换码后取低4位，得0～F，正好便于机器做十六进制运算，如表4.2所示。

表 4.2　数字换码运算

十进制数	键码	换码运算	取低4位
0	B0	B0⊕B0＝0	0
1	B1	B1⊕B0＝1	1
2	B2	B2⊕B0＝2	2
3	B3	B3⊕B0＝3	3
4	B4	B4⊕B0＝4	4
5	B5	B5⊕B0＝5	5
6	B6	B6⊕B0＝6	6
7	B7	B7⊕B0＝7	7
8	B8	B8⊕B0＝8	8
9	B9	B9⊕B0＝9	9
10(A)	C1	C1⊕B0＋89＝FA	A
11(B)	C2	C2⊕B0＋89＝FB	B
12(C)	C3	C3⊕B0＋89＝FC	C
13(D)	C4	C4⊕B0＋89＝FD	D
14(E)	C5	C5⊕B0＋89＝FE	E
15(F)	C6	C6⊕B0＋89＝FF	F

（2）若命令中包含数字，则换码后存到3E、3F，或转存到3C、3D和40、41。

（3）若遇到非数字码，换码后从该段程序退出，转入下段程序去查符号表，以便转入相应子程序处理。

由于从该段程序退出时，累加器带着换码后的新码（称为符号码）去查符号表（CHRTBL），因此可想而知，符号表存的不是Apple键码，而只能是符号码，符号表在FFCC～FFE2，如表4.3所示。

表4.3中CHRTBL部分，每单元内容（符号码）与之所对应命令符的键码符合如下等式：

符号码＝键码⊕B0＋89

如FFCC存的是CTRL-C的符号码BC，与CTRL-C的键码83符合如下等式：

BC＝83⊕B0＋89

键码扫描子程序如下所示。

```
键码扫描子程序
入口地址:FFA7
FFA7   0→X
FFA9   X→3E
FFAB   X→3F
FFAD   (0200,y)→A从缓冲区取键码,y为指针,y的初始值是0
FFB0   y+1→y
FFB1   A⊕B0→A
FFB3   A-0A
FFB5   BCC FF8A        数字0～9
FFB7   A+88+C→A C=1,A+89→A
FFB9   A-FA
FFBB   BCS FF8A        数字10～15
FFBD   RTS 遇到非数字退出

FF8A   03→X
FF8C   ASL
FF8D   ASL
FF8E   ASL
FF8F   ASL             高四位移出不要
FF90   ASL
FF91   ROR 3E
FF93   ROR 3F
FF95   X-1→X           低四位→3E,3F
FF96   BPL FF90
FF98   (31)→A
FF9A   BNE FFA2
FF9C   (3F,X)→A
FF9E   A→3D,X
FFA0   A→41,X
FFA2   X+1→X           (31)=0时,A2⇒A1,A3
                       (31)≠0时,A2不送A1,A3
FFA3   BEQ FF98
FFA5   BNE FFAD
```

A_1:3C,3D

A_2:3E,3F

A_3:40,41

4．子程序FFBE～取操作子程序入口地址

顾名思义，这段程序是取子程序入口地址的，以便转到预定子程序去执行相应功能。

所有操作子程序入口地址都是FE××，即高位都是FE，低位存放在FFE3～FFF9，构成子程序入口地址表（SUBTBL）。

符号表（CHRTBL）与子程序入口地址表（SUBTBL）有一一对应关系，如表4.3所示。

从FBE这段子程序首先取FE（子程序入口地址高位）压入堆栈，由指令FFBE、FFC0完成，然后根据查符号表时变址器y的数值得到SUBTBL取数，压入堆栈，由指令FFC1，FFC4完成。但必须注意，SUBTBL存的不是真正低位，而是“低位－1”，这是因为到子程序返回时，程序完成如下操作：PC↑，PC＋1→PC。“PC↑”表示从堆栈弹出二个数到指令计数器，也就是把子程序入口地址“低位－1”和“高位”从堆栈弹到指令计数器。“PC＋1→PC”的含义是指令计数器自动加1，然后程序转到指令计数器指出的入口地址。由于指令计数器作了自动加1操作，为了使程序转到预定入口地址，因而，SUBTBL存的数应当是入口地址低位－1。

现举例说明符号表与子程序入口地址表的对应关系：

CHRTBL第一个单元存的是BC，代表C^c，与之对应的SUBTBL第一个单元存的是B2，B2＋1＝B3。这表示：C^c的符号码是BC，其子程序入口地址是FEB3。如此等等。

取入口地址子程序如下所示：

```
FFBE  FE→A      取入口地址高位
FFC0  A→堆栈    高位压入堆栈
FFC1  (FFE3,y) →A  从SUBTBL取入口地址 "低位-1"
```

FFC4　A→堆栈　　"低位－1"压入堆栈
FFC5　(31)→A　　取方式
FFC7　0→y
FFC9　y→31　　置方式

FFCB　RTS　　PC↑　入口地址从堆栈弹出到指令计数器
PC+1→PC　指令计数器自动+1，程序转至相应的操作子程序。

表 4.3　符号表、子程序入口地址表及其对应关系

CHRTBL			SUBTBL		实　际
单 元 号	内容(符号码)	代表命令符	单　元　号	内容(地址低位-1)	入口地址
FFCC	BC	CTRL-C	FFE3	B2	FEB3
FFCD	B2	无用(注)	FFE4	C9	无用(注)
FFCE	BE	CTRL-E	FFE5	BE	FEBF
FFCF	B2	无用(注)	FFE6	C1	无用(注)
FFD0	EF	V	FFE7	35	FE36
FFD1	C4	CTRL-K	FFE8	8C	FE8D
FFD2	B2	CTRL-y	FFE9	C4	FEC5
FFD3	A9	CTRL-P	FFEA	96	FE97
FFD4	BB	CTRL-B	FFEB	AF	FEB0
FFD5	A6	—	FFEC	17	FE18
FFD6	A4	+	FFED	17	FE18
FFD7	06	M	FFEE	2B	FE2C
FFD8	95	<	FFEF	1F	FE20
FFD9	07	N	FFF0	83	FE84
FFDA	02	I	FFF1	7F	FE80
FFDB	05	L	FFF2	5D	FE5E
FFDC	F0	W	FFF3	CC	FECD
FFDD	00	G	FFF4	B5	FEB6
FFDE	EB	R	FFF5	FC	FEFD
FFDF	93	:	FFF6	17	FE18
FFE0	A7	·	FFF7	17	FE18
FFE1	C6	CR	FFF8	F5	FEF6
FFE2	99	BLANK(空格)	FFF9	03	FE04

注：对于新监控，FFCD、FFCF、FFE4、FFE6这四个单元是没有用的，对于老监控，这四个单元是有用的，其内容及意义如下.

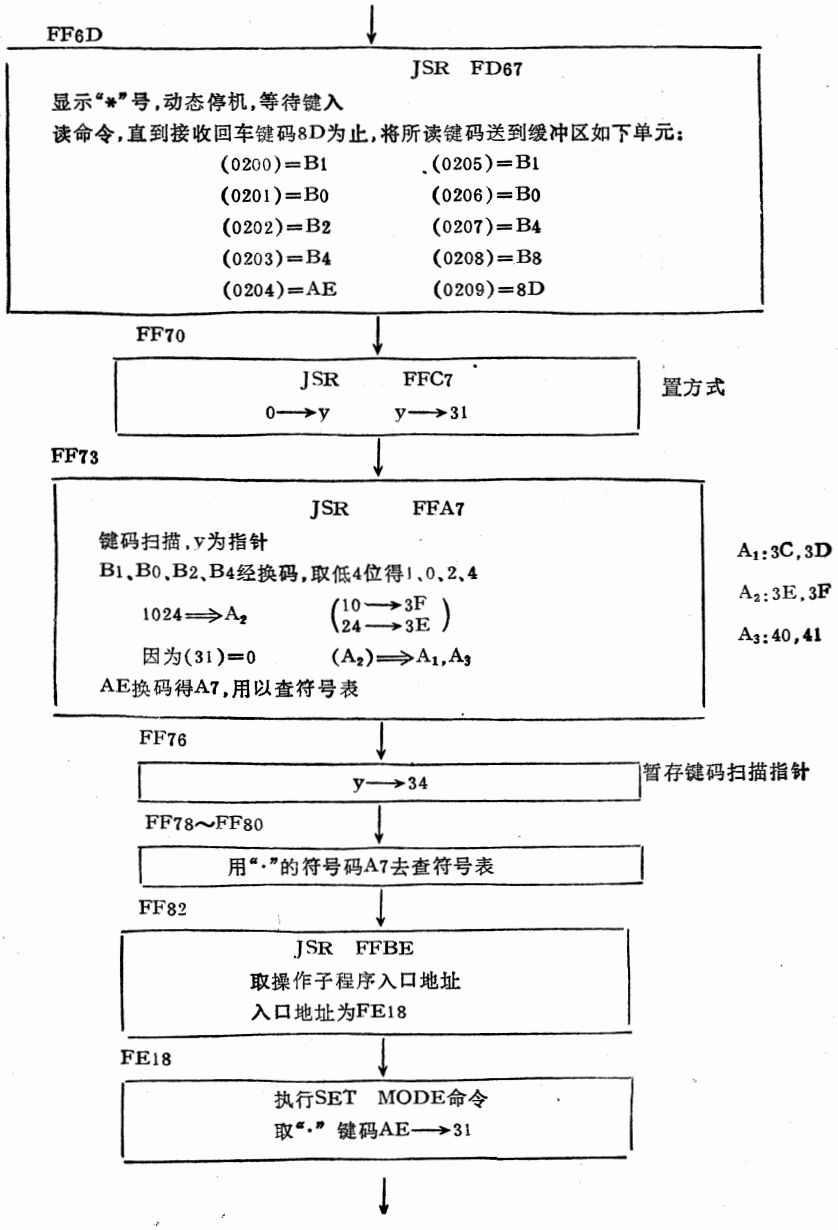
单 元 号	内容(符号码)	代表命令符	单 元 号	内容(地址低位-1)	实　际入口地址
FFCD	B2	CTRL-Y	FFE4	C9	FECA
FFCF	ED	T	FFE6	C1	FEC2
FFD2	EC	S	FFE9	C3	FEC4

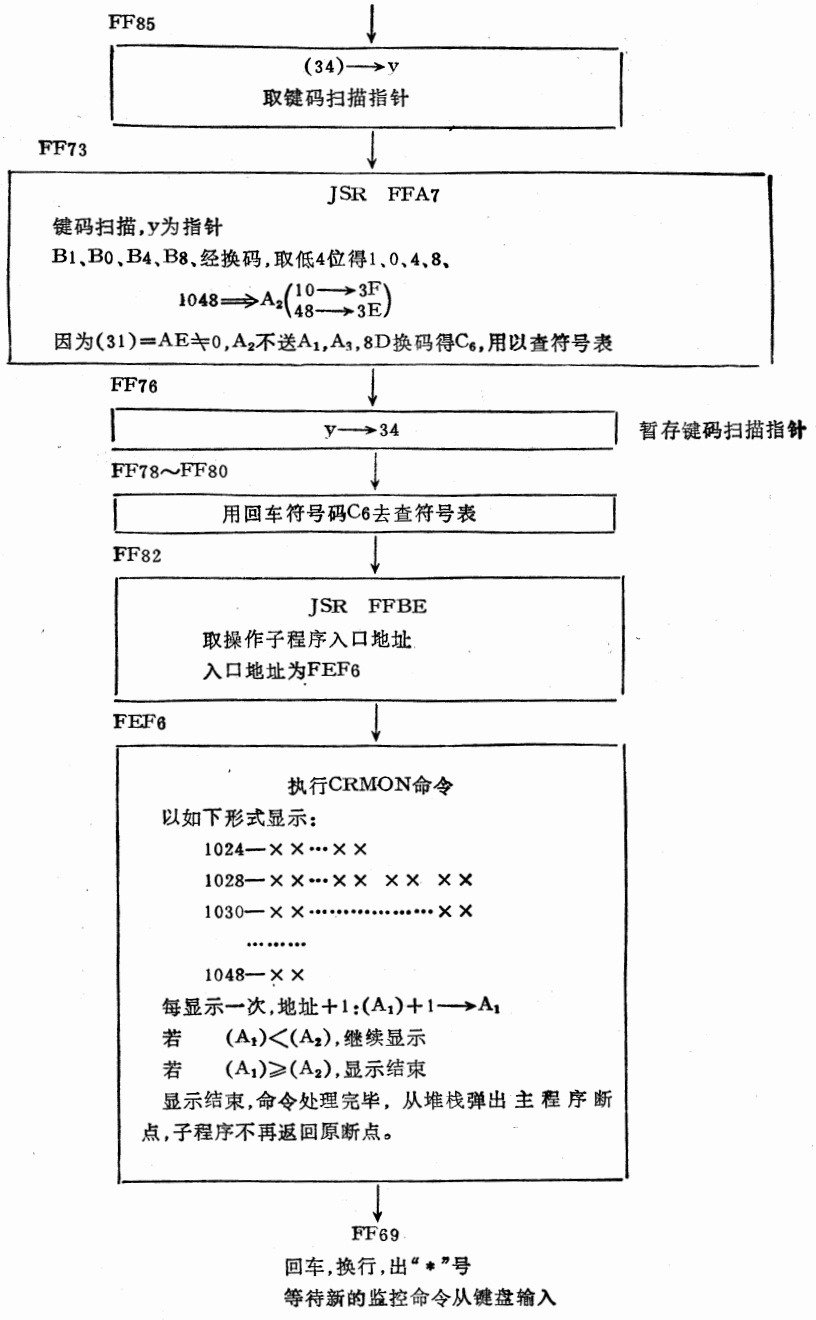
4.5 Apple监控命令执行过程流程图

下面是Apple监控系统所有命令的流程图。读者在读这些流程图时，请与图4.2主程序细框图互相对照，这样就比较容易读懂。所有的流程图都是从FF6D开始画，因为从主程序细框图来看，Apple始终是在FF6D等待键盘输入。

1. * 1024·1048〈CR〉命令的执行过程

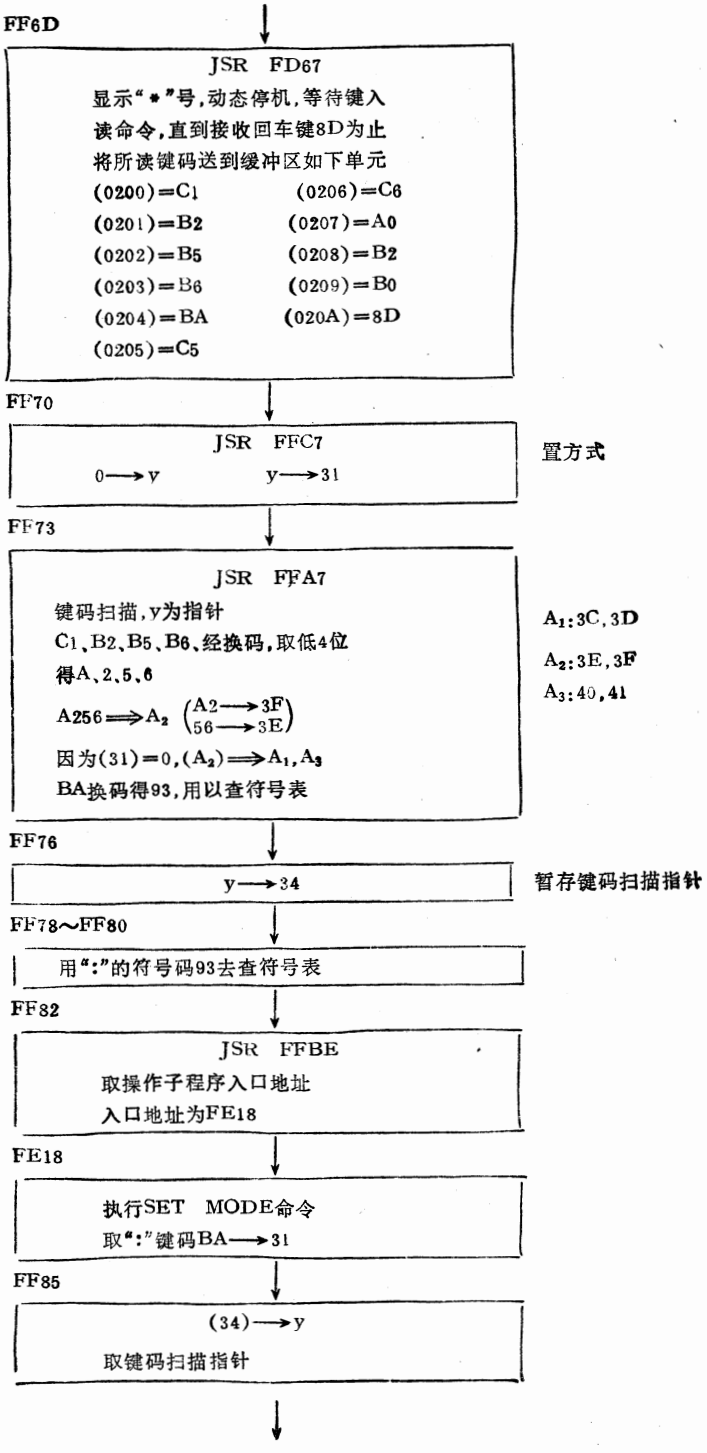
这条命令的键码是：B1 B0 B2 B4 AE B1 B0 B4 B8 8D

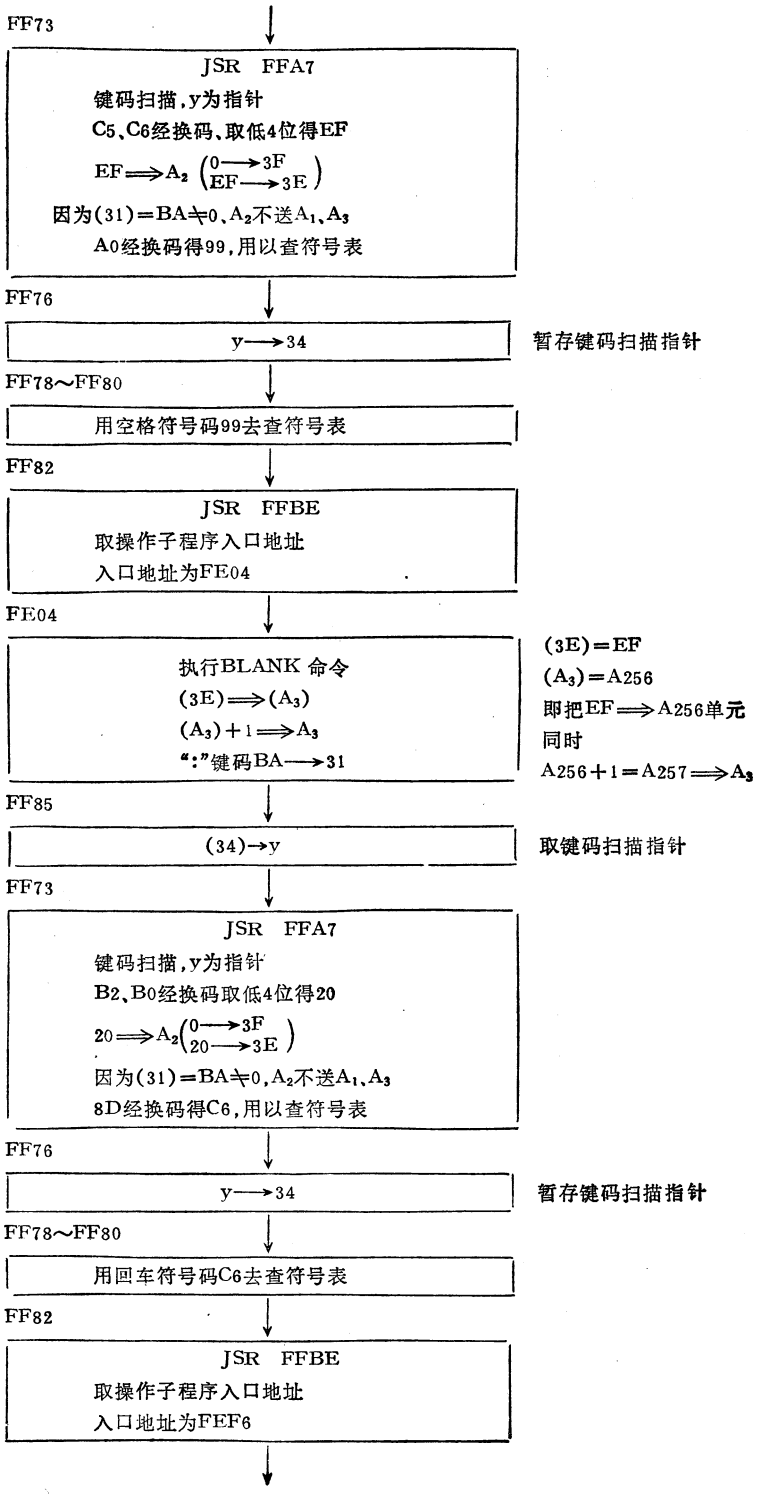


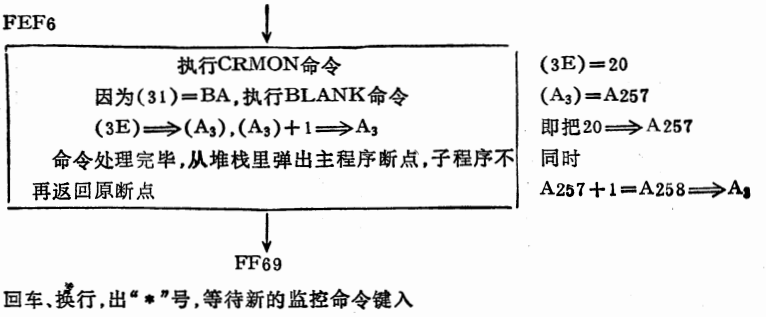


2. *A256: EF␣20<CR>命令的执行过程

这条命令的键码是: C1 B2 B5 B6 BA C5 C6 A0 B2 B0 8D

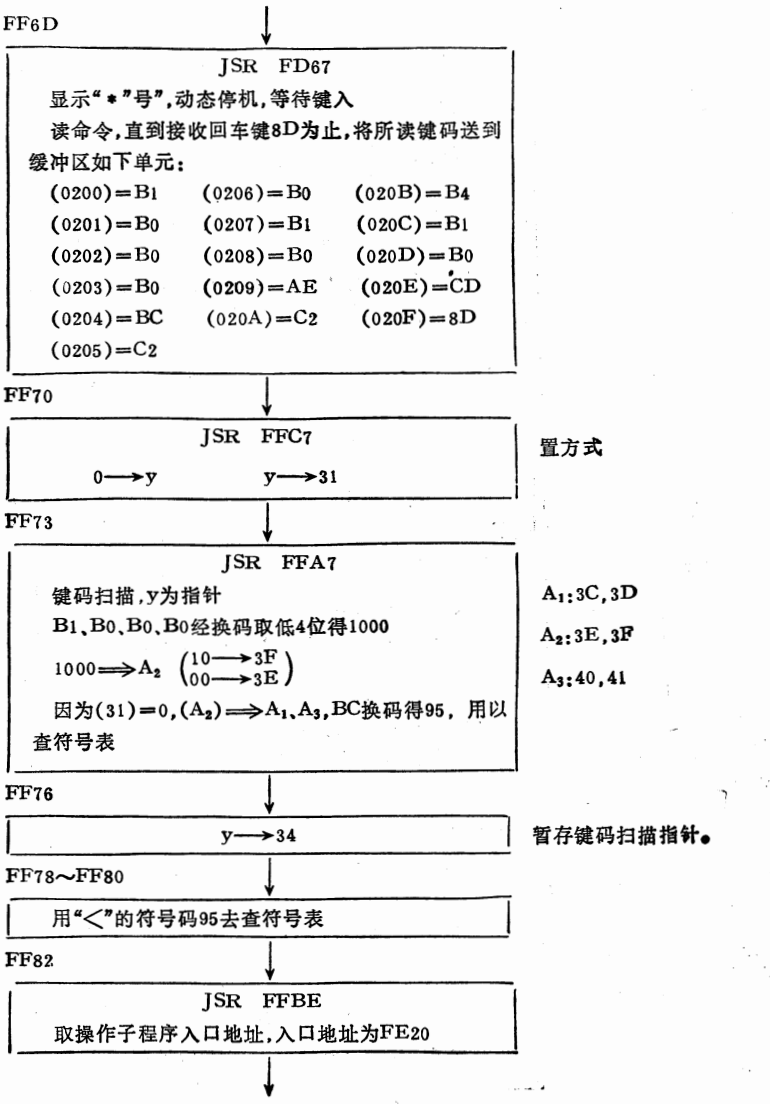

FF6D
JSR FD67
显示"*"号,动态停机,等待键入
读命令,直到接收回车键8D为止
将所读键码送到缓冲区如下单元
(0200)=C1 (0206)=C6
(0201)=B2 (0207)=A0
(0202)=B5 (0208)=B2
(0203)=B6 (0209)=B0
(0204)=BA (020A)=8D
(0205)=C5
FF70
JSR FFC7
0→y y→31
置方式
FF73
JSR FFA7
键码扫描,y为指针
C1、B2、B5、B6、经换码,取低4位
得A、2、5、6
A256⟹A2 (A2→3F, 56→3E)
因为(31)=0,(A2)⟹A1,A3
BA换码得93,用以查符号表
A1:3C,3D
A2:3E,3F
A3:40,41
FF76
y→34
暂存键码扫描指针
FF78～FF80
用":"的符号码93去查符号表
FF82
JSR FFBE
取操作子程序入口地址
入口地址为FE18
FE18
执行SET MODE命令
取":"键码BA→31
FF85
(34)→y
取键码扫描指针

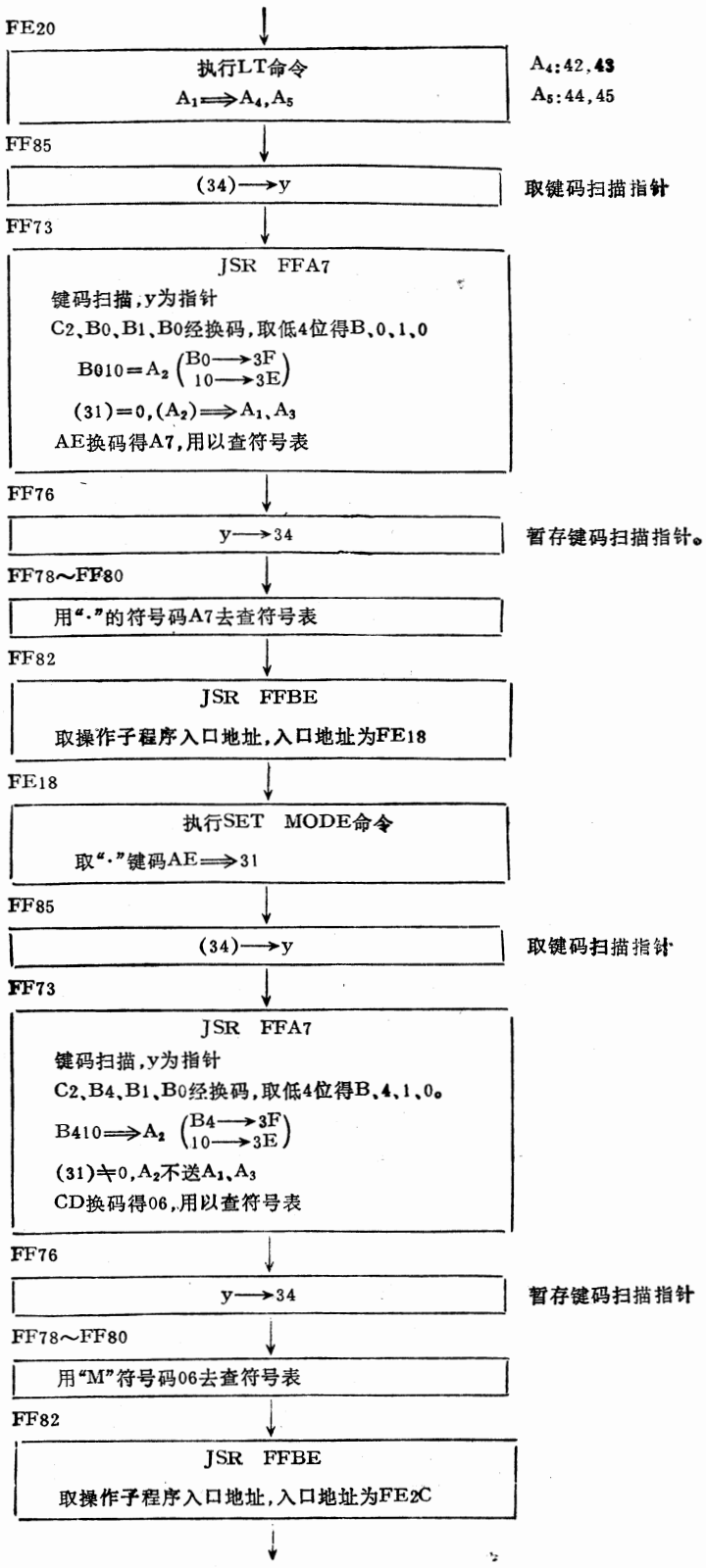

FF73
JSR FFA7
键码扫描,y为指针
C5、C6经换码、取低4位得EF
EF⟹A2 (0⟶3F, EF⟶3E)
因为(31)=BA≠0、A2不送A1、A3
A0经换码得99,用以查符号表
FF76
y⟶34
暂存键码扫描指针
FF78～FF80
用空格符号码99去查符号表
FF82
JSR FFBE
取操作子程序入口地址
入口地址为FE04
FE04
执行BLANK 命令
(3E)⟹(A3)
(A3)+1⟹A3
":"键码BA⟶31
(3E)=EF
(A3)=A256
即把EF⟹A256单元
同时
A256+1=A257⟹A3
FF85
(34)→y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
B2、B0经换码取低4位得20
20⟹A2(0⟶3F, 20⟶3E)
因为(31)=BA≠0,A2不送A1、A3
8D经换码得C6,用以查符号表
FF76
y⟶34
暂存键码扫描指针
FF78～FF80
用回车符号码C6去查符号表
FF82
JSR FFBE
取操作子程序入口地址
入口地址为FEF6

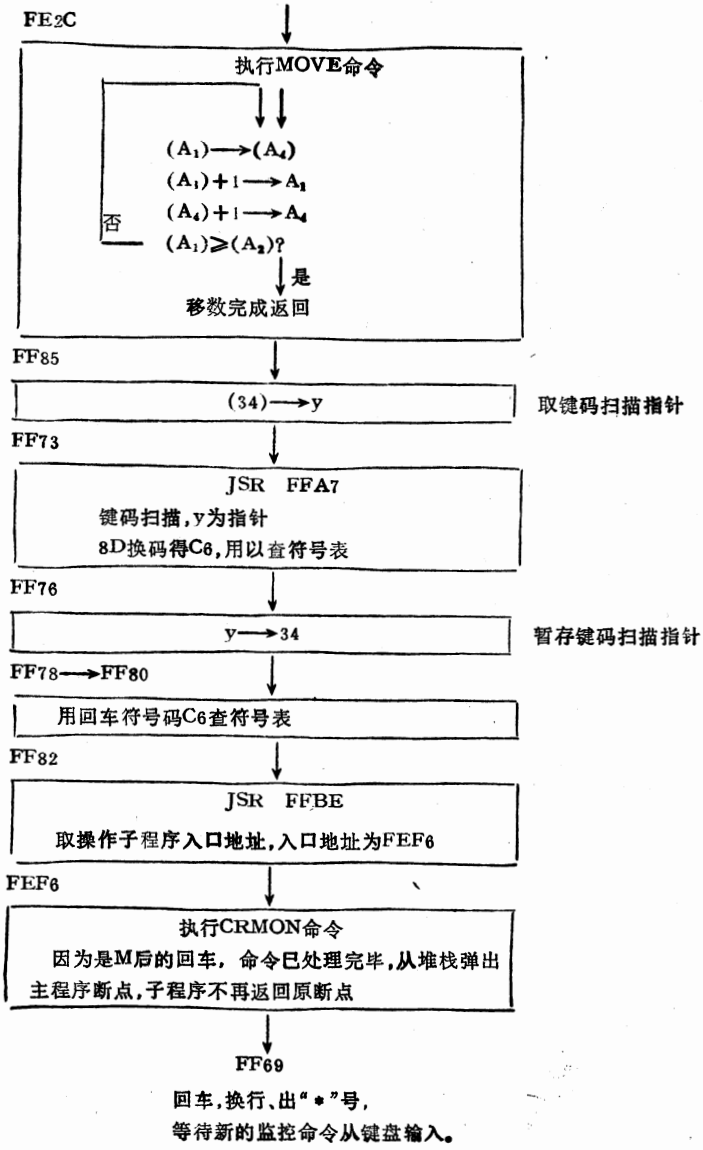


3. *1000<B010·B410M<CR>

这条命令的键码是：B1 B0 B0 B0 BC C2 B0 B1 B0 AE C2 B4 B1 B0 CD 8D

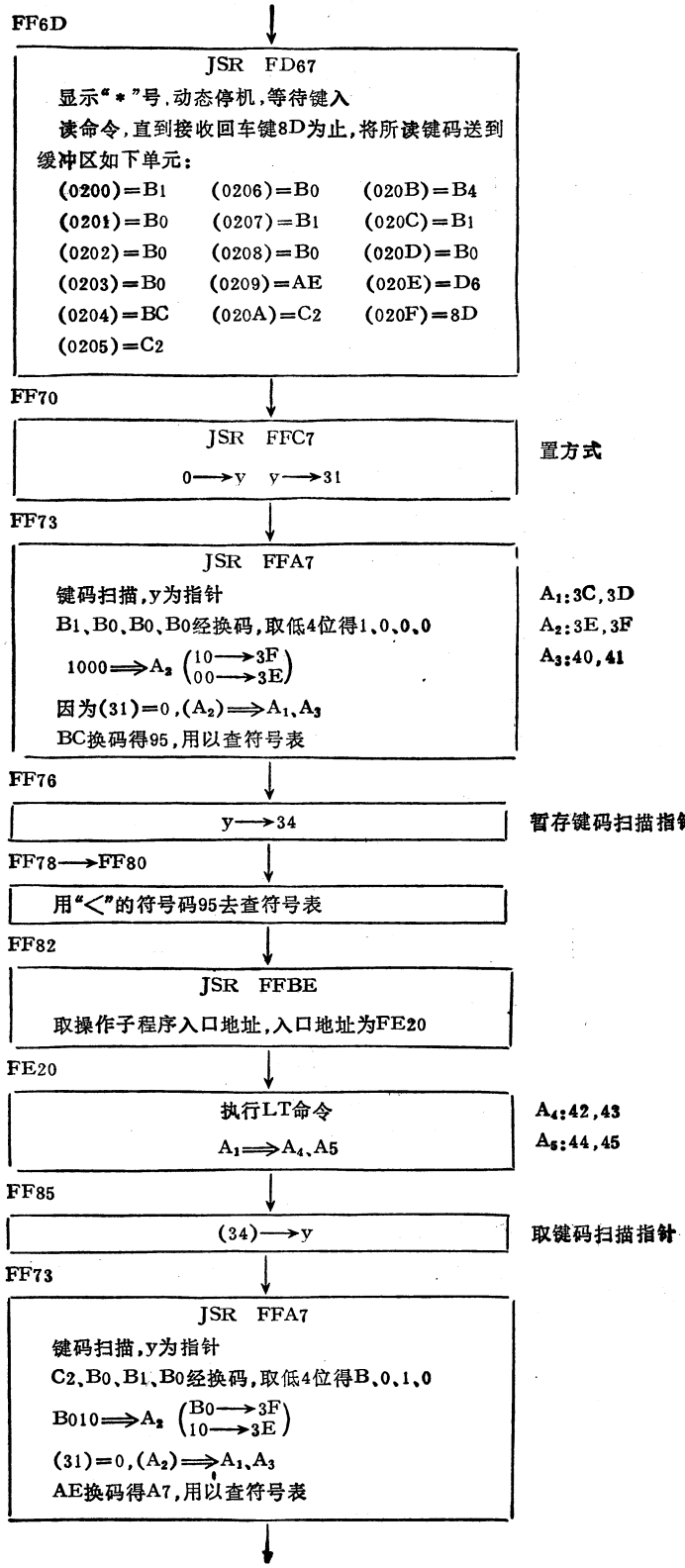


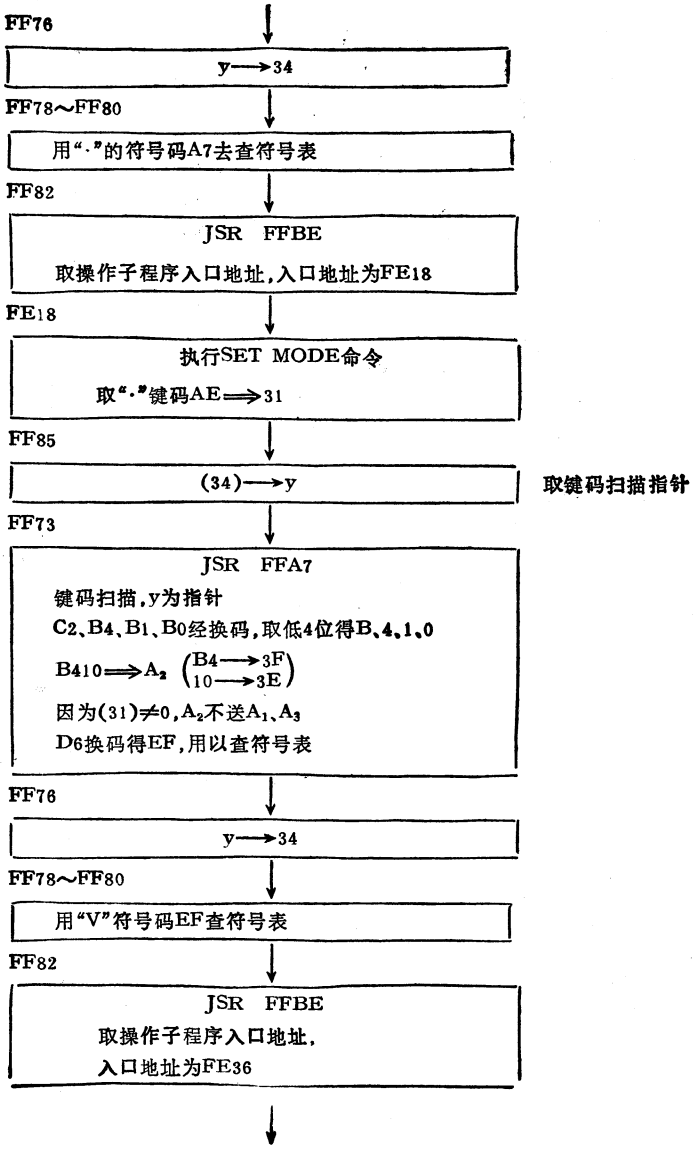

FE20
执行LT命令
$A_1 \Longrightarrow A_4, A_5$
A_4:42,43
A_5:44,45
FF85
(34)⟶y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
C2、B0、B1、B0经换码,取低4位得B、0、1、0
$B010 = A_2 \begin{pmatrix} B0 \longrightarrow 3F \\ 10 \longrightarrow 3E \end{pmatrix}$
(31)=0,$(A_2) \Longrightarrow A_1, A_3$
AE换码得A7,用以查符号表
FF76
y⟶34
暂存键码扫描指针。
FF78～FF80
用"."的符号码A7去查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE18
FE18
执行SET MODE命令
取"."键码AE⟹31
FF85
(34)⟶y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
C2、B4、B1、B0经换码,取低4位得B、4、1、0。
$B410 \Longrightarrow A_2 \begin{pmatrix} B4 \longrightarrow 3F \\ 10 \longrightarrow 3E \end{pmatrix}$
(31)≠0,A_2不送A_1、A_3
CD换码得06,用以查符号表
FF76
y⟶34
暂存键码扫描指针
FF78～FF80
用"M"符号码06去查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE2C

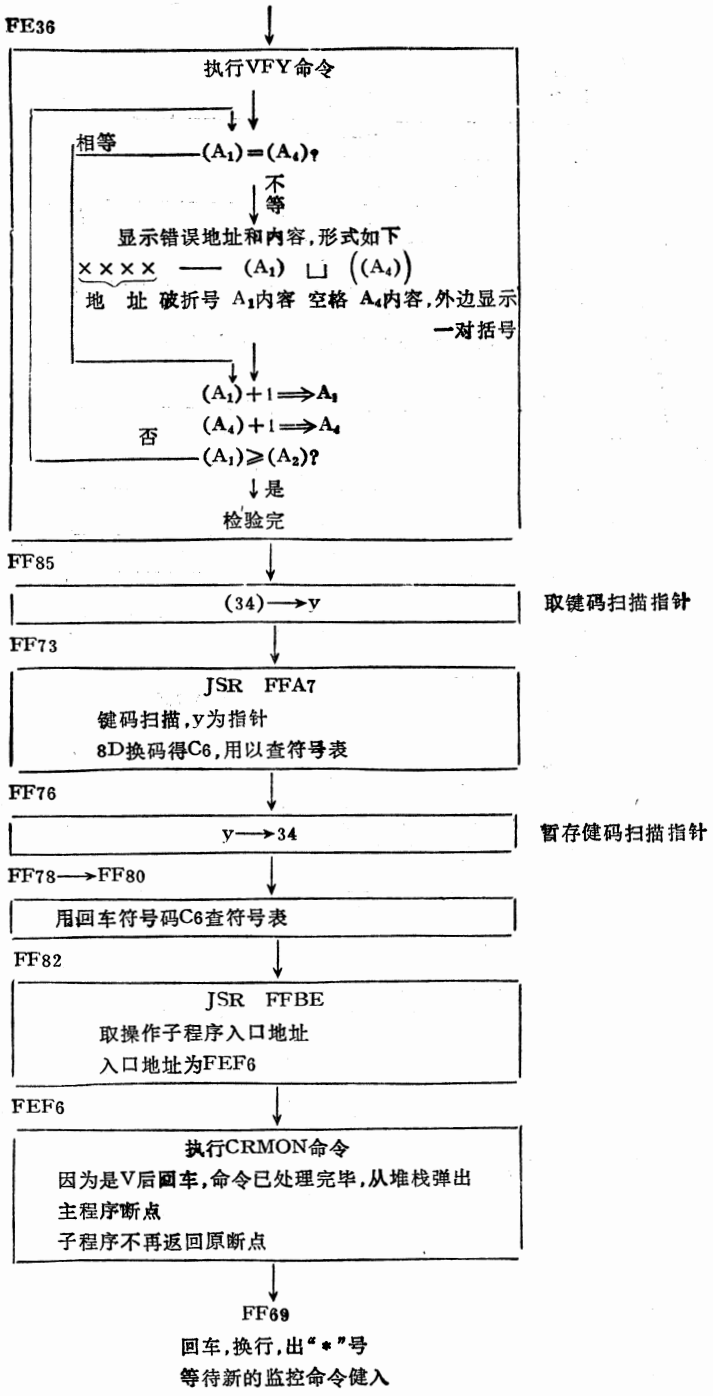


4. *1000<B010·B410V<CR>

这条命令的码键是：B1 B0 B0 B0 BC C2 B0 B1 B0 AE C2 B4 B1 B0 D6 8D


FF6D
JSR FD67
显示"*"号,动态停机,等待键入
读命令,直到接收回车键8D为止,将所读键码送到
缓冲区如下单元:
(0200)=B1 (0206)=B0 (020B)=B4
(0201)=B0 (0207)=B1 (020C)=B1
(0202)=B0 (0208)=B0 (020D)=B0
(0203)=B0 (0209)=AE (020E)=D6
(0204)=BC (020A)=C2 (020F)=8D
(0205)=C2
FF70
JSR FFC7
0⟶y y⟶31
置方式
FF73
JSR FFA7
键码扫描,y为指针
B1、B0、B0、B0经换码,取低4位得1、0、0、0
1000⟹A2 (10⟶3F 00⟶3E)
因为(31)=0,(A2)⟹A1、A3
BC换码得95,用以查符号表
A1:3C,3D
A2:3E,3F
A3:40,41
FF76
y⟶34
暂存键码扫描指针
FF78⟶FF80
用"<"的符号码95去查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE20
FE20
执行LT命令
A1⟹A4、A5
A4:42,43
A5:44,45
FF85
(34)⟶y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
C2、B0、B1、B0经换码,取低4位得B、0、1、0
B010⟹A2 (B0⟶3F 10⟶3E)
(31)=0,(A2)⟹A1、A3
AE换码得A7,用以查符号表


FF76
y⟶34
FF78～FF80
用"·"的符号码A7去查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE18
FE18
执行SET MODE命令
取"·"键码AE⟹31
FF85
(34)⟶y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
C2、B4、B1、B0经换码,取低4位得B、4、1、0
B410⟹A2 (B4⟶3F, 10⟶3E)
因为(31)≠0,A2不送A1、A3
D6换码得EF,用以查符号表
FF76
y⟶34
FF78～FF80
用"V"符号码EF查符号表
FF82
JSR FFBE
取操作子程序入口地址,
入口地址为FE36



5. *8400•9FFFW〈CR〉

这是一条把内存8400～9FFF的内容往磁带机上写的命令,整个过程的波形如 图4.3 所示。

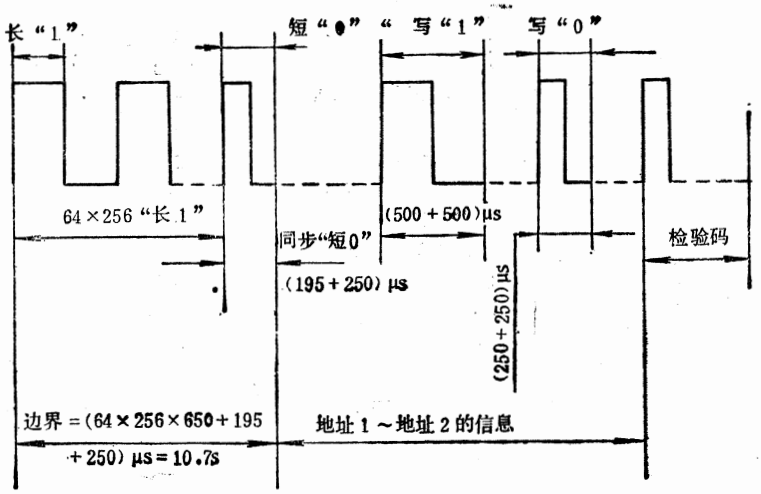


图 4.4 信息写往磁带的波形

这个波形是完全通过分析程序得到的，现将与这波形有关的两段程序摘录如下：

入口地址:FECD

①	FECD	40→A	
②	FECF	JSR FCC9	
③	FED2	27→y	2
④	FED4	0→x	2
⑤	FED6	A⊕(3C,x)→A	6累计检验码
⑥	FED8	A↓	3
⑦	FED9	(3C,x)→A	6
⑧	FEDB	JSR FEED	6写一个字节
⑨	FEDE	JSR FCBA	6计算地址
⑩	FEE1	1D→y	2
⑪	FEE3	P↑	4
⑫	FEE4	C=0转	2判断记录长度,够长时C=1
⑬	FEE6	22→y	2
⑭	FEE8	JSR FEED	6
⑮	FEEB	结果为0转	2或3
⑯	FEED	10→x	2
⑰	FEEF	←A	2
⑱	FEF0	JSR FCD6	6
⑲	FEF3	结果不为0转	2或3
⑳	FEF5	RTS	6
㉑	FF3A	87→A	
㉒	FF3C	JMP FDED	

㉓	FCC9	┌→4B→y	2
㉔	FCCB	JSR FCDB	6
㉕	FCCE	└结果≠0转	2或3
㉖	FCDC	A+FE+C→A	2
㉗	FCD2	└C=1转	2
㉘	FCD4	21→y	2
㉙	FCD6	JSR FCDB	6
㉚	FCD9	y+1→y	2
㉛	FCDA	y+1→y	2
㉜	FCDB	┌→y−1→y	2
㉝	FCDC	└结果≠0转	2或3
㉞	FCDE	┌C=0转	2
㉟	FCE0	32→y	2
㊱	FCE2	┌→y−1→y	2
㊲	FCE3	└结果≠0转	2或3
㊳	FCE5	└→LDY C020	4
㊴	FCE8	2C→y	2
㊵	FCEA	x−1→x	2
㊶	FCEB	RTS	6

每条指令左边按顺序标上号是为了下面的叙述方便。指令右边标出每条指令执行的时间，单位是节拍，也可理解为微秒，因1节拍$=10^{-6}/1.023$秒≒1微秒。

下面分析5个问题。在分析过程中，请注意㊳号指令LDYC020已在第3.7节指出过。C020是写磁带设备码。C020每选通一次，波形翻转一次，两次选通C020之间的时间间隔就是脉冲（半波）宽度。另外需要交待的是，程序从FECD入口时，C=1，X=1。

（1）“长1”宽度

“长1（Long1）”和“短0（short 0）是针对方波个数和宽度而言，前者个数多而宽度大，后者只有两个半波而且很窄。

列出在“长1”情况下，两次选通设备码C020中间所执行的指令，然后把执行这些指令的时间加起来，即为“长1”宽度。这些指令（按执行顺序列出）为：

指令号：㊴ ㊵ ㊶ ㉕ ㉓ ㉔ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳

时间（拍）：2 2 6 3 2 6 75×5 2 2 50×5 4

因而“长1”宽度$=2+2+6+3+2+6+75\times5+2+2+50\times5+4=654\fallingdotseq650$（“拍”或“微秒”）。“长1”个数是由1号指令给累加器赋以40所决定的。㉗号指令共循环了41次，而每执行一次㉗号指令（最后一次除外），㊵指令就循环256次，因此，“长1”个数$=64\times256$个 “长1”总时间$=64\times256\times650(\mu s)$

（2）“短0”宽度

写完“长1”，紧接着就写“短0”，这时C=0，“短0”波形如下页上图。

从写完“长1”最后一条边开始到再次选通C020设备码，这期间所执行的指令就决定了“短0”第一个半波宽度。这些指令为：

指　令　号：㊴ ㊵ ㊶ ㉕ ㉖ ㉗ ㉘ ㉙ ㉜ ㉝ ㉞ ㊳

时间（拍）：2　2　6　2　2　2　2　6　33×5　2　4

因而，“短0”第一个半波宽度＝2＋2＋6＋2＋2＋2＋2＋6＋33×5＋2＋4＝195(拍或μs)。

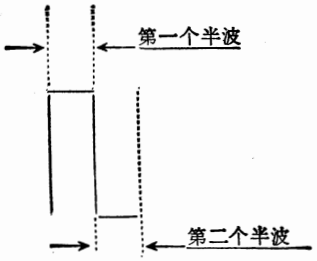
“短0”第二个半波宽度由下列指令决定：

指　令　号：㊴ ㊵ ㊶ ㉚ ㉛ ㉜ ㉝ ㉞ ㊳

时间（拍）：2　2　6　2　2　46×5　2　4

“短0”第二个半波宽度＝2＋2＋6＋2＋2＋46×5＋2＋4＝250(μs)

所以，同步“短0”的总时间是（195＋250)μs

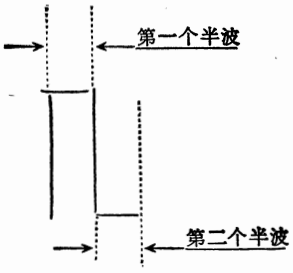


边界的总时间＝(64×256×650＋195＋250)μs＝10.7s

（3）写信息时，写“1”的宽度

写信息的大致过程是这样：

⑯号指令，$(10)_{16}\rightarrow X$，含义是一个字节有8位，每写一位波形翻转二次，对X要减2次，写一个字节要减$(16)_{10}$次，因而X的初始值是$(16)_{10}$。⑰号指令是把一个字节逐位移出到C，根据C是0还是1，以决定写“0”还是写“1”，由⑱号指令转出去写。



要分析写“1”或写“0”的方波宽度，就要假设⑰号指令移出一个“1”或“0”，然后根据C＝1或C＝0找出两次执行㊳号指令期间所经指令，即可决定写“1”或写“0”的方波宽度。写“1”或写“0”的波形如上：

写“1”时第一个半波由下列指令决定：

指　令　号：㊴ ㊵ ㊶ ⑲ ⑰ ⑱ ㉙ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳

时间（拍）：2　2　6　3　2　6　6　44×5　2　2　50×5　4

第一个半波宽度＝2＋2＋6＋3＋2＋6＋6＋44×5＋2＋2＋50×5＋4

＝505(μs)≒500(μs)

第二个半波宽度由下列指令决定：

指　令　号：㊴ ㊵ ㊶ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳

时间（拍）：2　2　6　2　2　46×5　2　2　50×5　4

第二个半波宽度＝2＋2＋6＋2＋2＋46×5＋2＋2＋50×5＋4

＝502≒500（μs）

所以，写“1”的时间是（500＋500）μs

（4）写信息时，写“0”的宽度

分析写“0”的方波宽度与分析写“1”时的方法相同，只要注意到写“0”时C＝0就可以了。

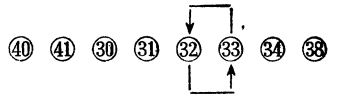
第一个半波由下列指令决定：

时间（拍）：2 2 6 3 2 6 6 44×5 2 4

第一个半波宽度＝2＋2＋6＋3＋2＋6＋6＋44×5＋2＋4

＝253≒250(μs)

第二个半波由下列指令决定：



时间（拍）：2 2 6 2 2 46×5 2 4

第二个半波宽度＝2＋2＋6＋2＋2＋230＋2＋4＝250(μs)

所以写“0”的时间是（250＋250）μs。

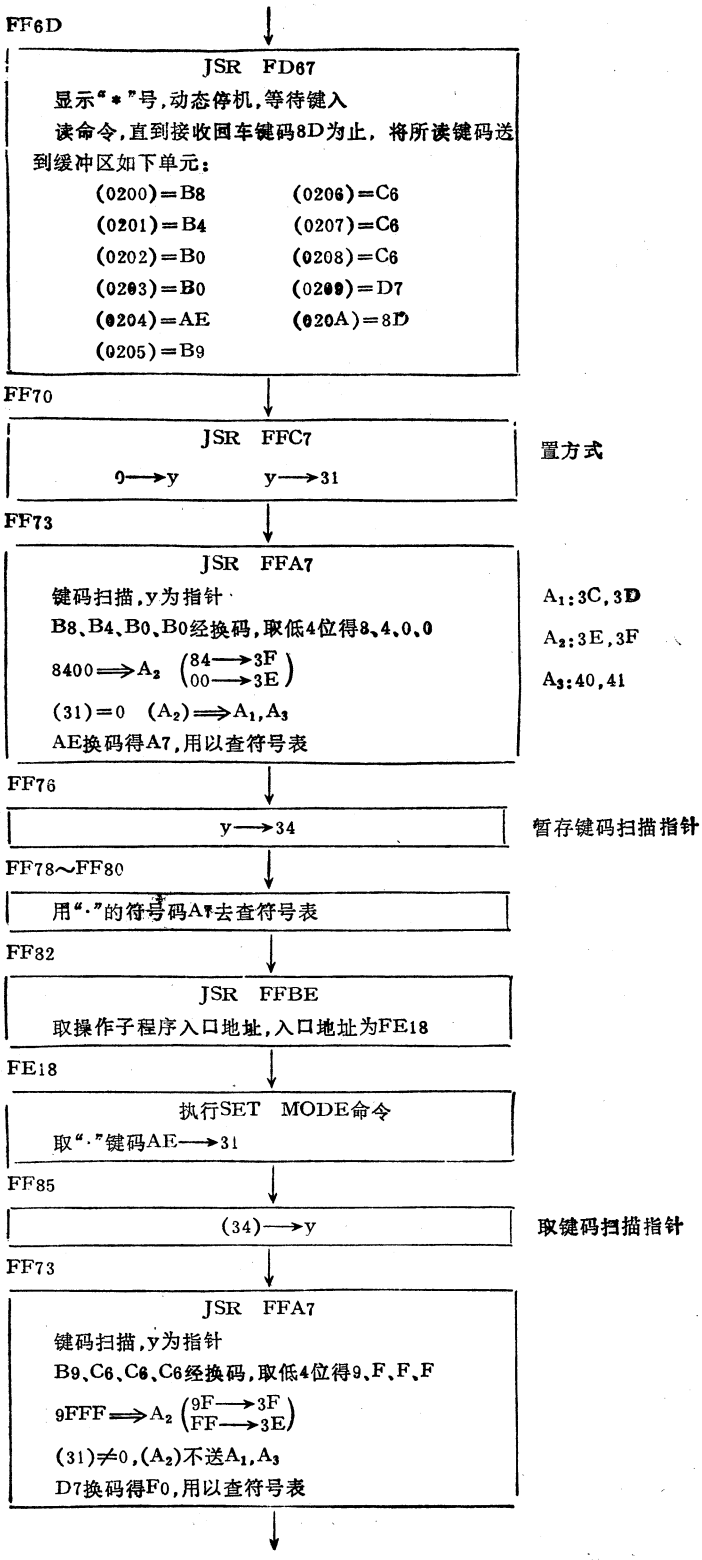
（5）检验码的形成及记录

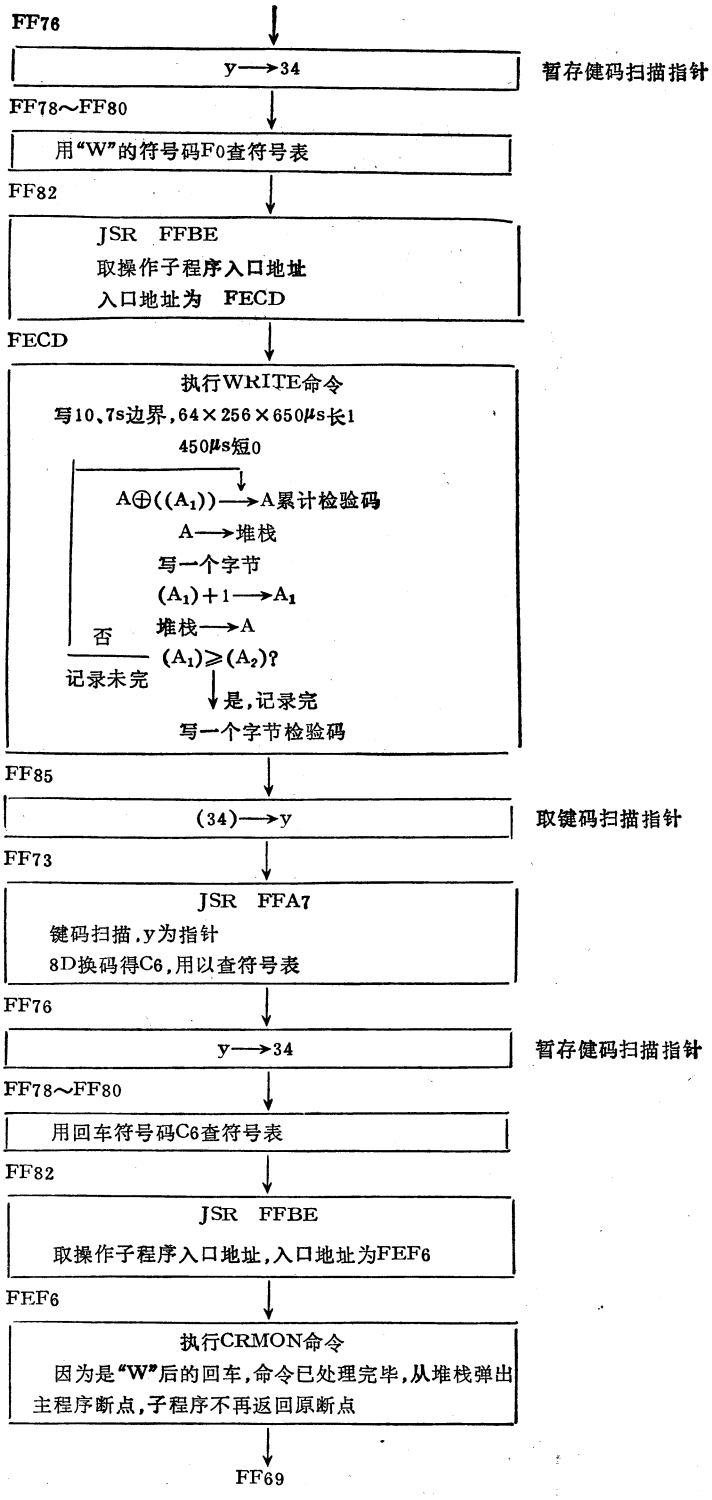
检验码是多次执行⑤号指令EOR（3C，X）形成的，其起始值是利用子程序FCC9的运算结果。从②号指令转到FCC9写边界，写完边界又返回到③号指令时，累加器的值是FF，这就是检验码的起始值。然后，每次在写一个字节之前，执行一次⑤号指令，把累加器和要记录的字节作“异或”运算，直到最后一个字节。因此，检验码与记录长度内每个字节有关。

检验码的记录由⑭号指令所执行的子程序来完成。

最后交待一下，⑩和⑬号指令是在写一个字节过程中管写第一位方波宽度的。写其他各位时，方波宽度均由㊴号指令和㉟号指令管着。写完一个字节后。因为要判断是否够记录长度了，执行了从③号到⑫号指令，耗费了一些时间。为了保证在整个记录过程中，除边界外任何时候写“1”和写“0”的长度均为预定值，因而写每一字节的第一位之前，决定方波宽度的数据不能采用㊴号指令所赋的值，而应适当减小。⑩、⑬以及③号指令就是为此而设计的。

下面分析命令＊8400·9FFFW〈CR〉的框图，这条命令的键码是：B8 B4 B0 B0 AE B9 C6 C6 C6 D7 8D。


FF6D
JSR FD67
显示"*"号,动态停机,等待键入
读命令,直到接收回车键码8D为止,将所读键码送到缓冲区如下单元:
(0200)=B8 (0206)=C6
(0201)=B4 (0207)=C6
(0202)=B0 (0208)=C6
(0203)=B0 (0209)=D7
(0204)=AE (020A)=8D
(0205)=B9
FF70
JSR FFC7
0→y y→31
置方式
FF73
JSR FFA7
键码扫描,y为指针
B8、B4、B0、B0经换码,取低4位得8、4、0、0
8400⇒A2 (84→3F 00→3E)
(31)=0 (A2)⇒A1,A3
AE换码得A7,用以查符号表
A1:3C,3D
A2:3E,3F
A3:40,41
FF76
y→34
暂存键码扫描指针
FF78～FF80
用"."的符号码A7去查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE18
FE18
执行SET MODE命令
取"."键码AE→31
FF85
(34)→y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
B9、C6、C6、C6经换码,取低4位得9、F、F、F
9FFF⇒A2 (9F→3F FF→3E)
(31)≠0,(A2)不送A1,A3
D7换码得F0,用以查符号表



回车，换行，出"*"号，等待新的键盘输入

6. *3100·4FF0R〈CR〉

这是一条从磁带机上读信息并送往内存的指令。从前面刚介绍的往磁带机上存信息的指令我们知道了信息写往磁带的过程，这是了解本指令的基础。根据信息在磁带上是怎么写的，我们不难想到读的过程必须包括下列步骤：

（1）寻找边界，即寻找“长1”和同步“短0”。

（2）找到同步“短0”后，紧接着读到的就是所需信息。根据方波宽度以确定每一位是“0”还是“1”。

（3）在读的过程中不断累计检验码，读完规定字节数后校核检验码，以确定所读信息长度是否正确。

我们把读磁带机有关的三段指令摘录如下，并在指令旁边做了一些注解。反映这三段指令基本思想的框图，已在下面命令流程图中FEFD“执行READ命令”那一框中列出，读者根据框图及指令注释仔细琢磨，是不难理解其含义的。应当指出的是，FCFA这段子程序很关键，它是理解本指令的核心。一般地说，从FCFA入口，FD0B返回，是读到了半波（⎍或⎓）；从FCFD入口，FD0B返回，是读到了一条跳边（⌊或⌋）。

读磁带的三段程序如下：

入口地址：FEFD

```
FEFD  JSR FCFA   检验到⎍或⎓,⌊,⌋
FF00  16→A       延时3.6秒,返回时A=FF
FF02  JSR FCC9
FF05  A→2E       FF(检验码初始值)→2E
FF07  JSR FCFA   检测到⎍或⎓、⌊、⌋,目的是找一条跳边
FF0A  →24→y      找同步短0的第一个半波
FF0C  JSR FCFD
FF0F  —C=1转     未找到就继续找
FF11  JSR FCFD找同步短0的第二半波
FF14  3B→y
FF16  →JSR FCEC  读一个字节
FF19    A→(3C,X) 存一个字节
FF1B    A⊕2E     累计检验码
FF1D    A→2E     寄存检验码
FF1F    JSR FCBA 计算地址
FF22    35→y
FF24  —C=0转     计录长度未读完转
FF26  JSR FCEC   读检验码
FF29  A—(2E)     校核检验码
FF2B  →结果为0转 检验码正确转
FF2D    C5→A     显示“E”
FF2F    JSR FDED
FF32    D2→A     显示“RR”
FF34    JSR FDED
FF37    JSR FDED
FF3A  —87→A      响铃
FF3C    JMP FDED
```

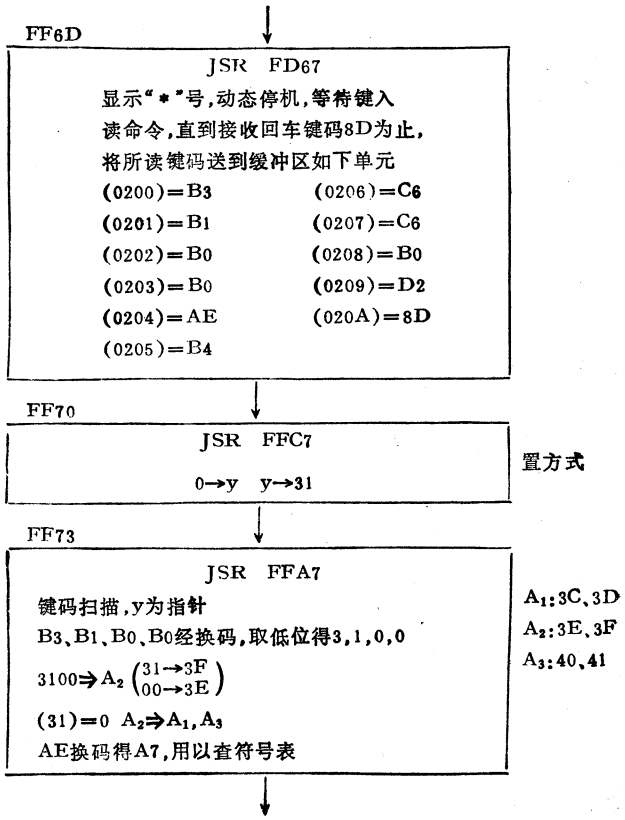
读“⎍”或“⎓”子程序

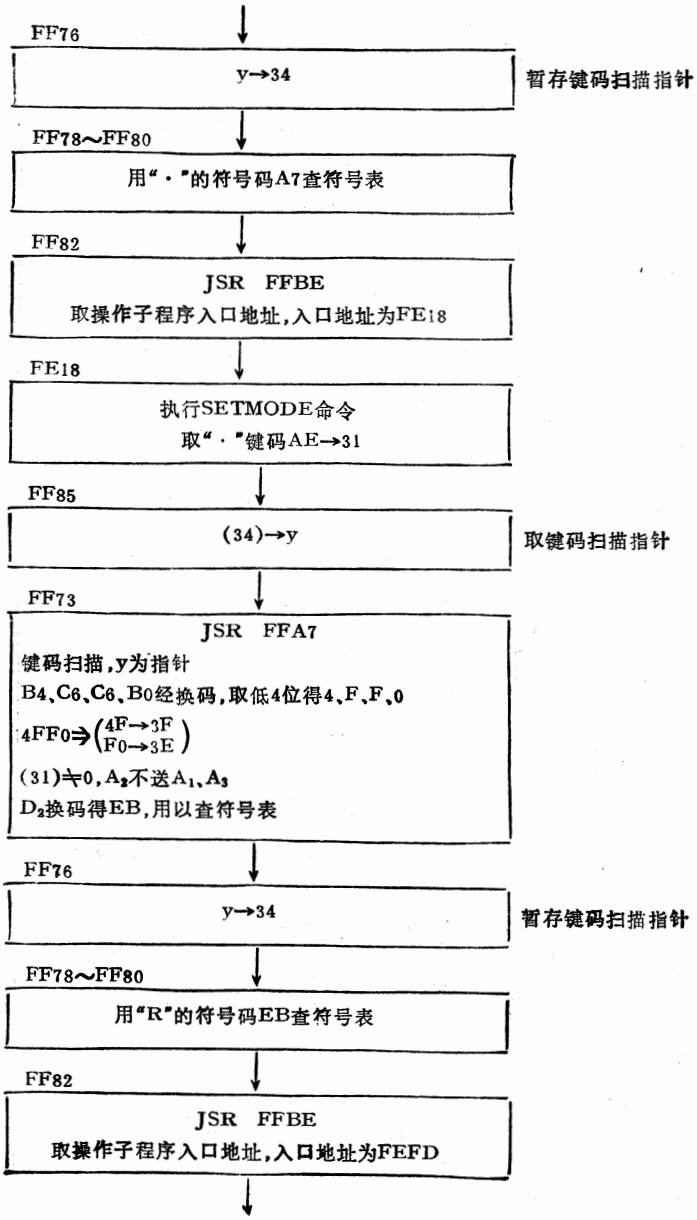
```
FCFA   JSR FCFD
FCFD  ┌→y－1→y
FCFE  │  LDA C060   读磁带
FD01  │  A⊕2F→A
FD03  └─正值转        (2F)跟A反号时程序往下走,否则循环,程序下走时,A中最高位为1

FD05     A⊕2F→A
FD07     A→2F       使(2F)最高位变号,2F最高位寄存波形状态
FD09     y－80
FD0B     RTS
```

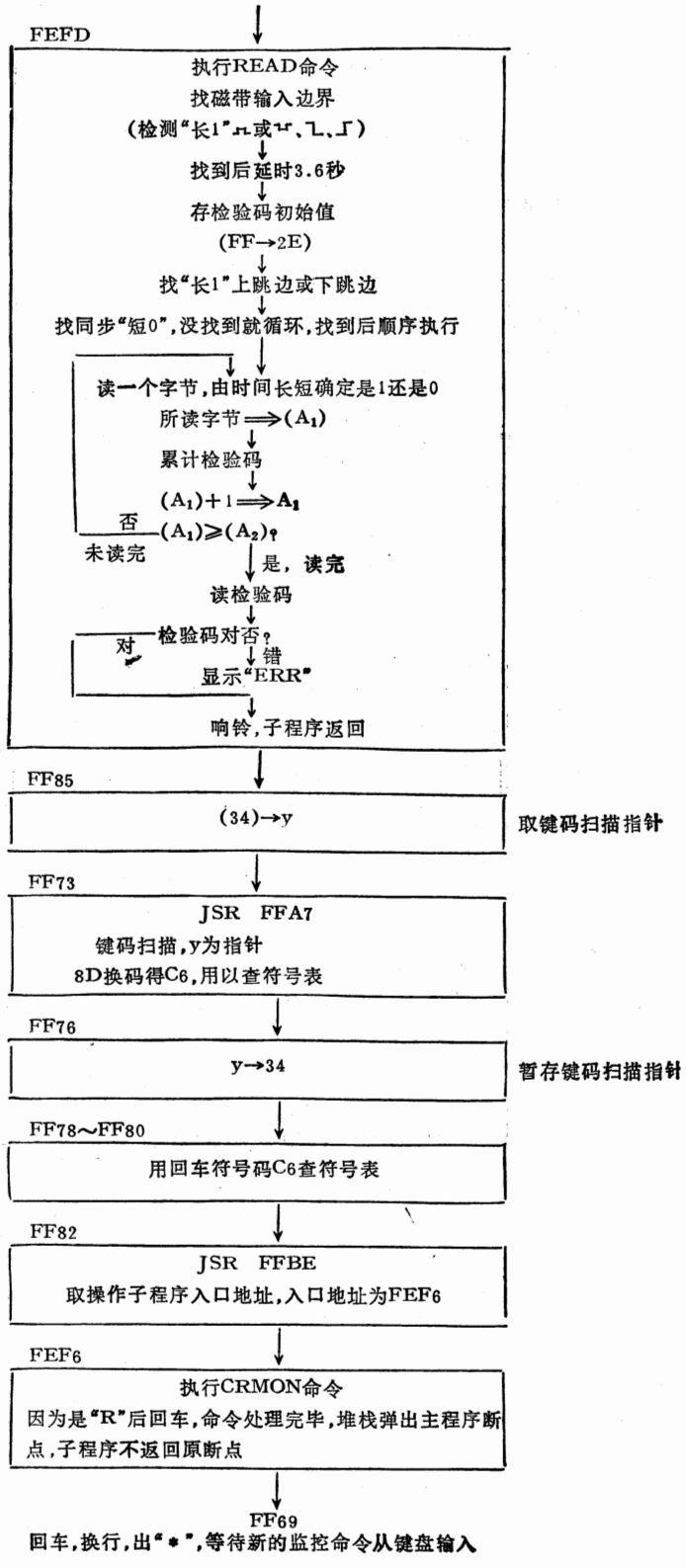
读一个字节子程序

```
FCEC   08→X   一个字节读8位
FCEE  ┌→A→堆栈
FCEF  │  JSR FCFA  读一位
FCF2  │  堆栈→A
FCF3  │  ROL        所读的1位移入A
FCF4  │  3A→y
FCF6  │  x－1→x
FCF7  └─结果不为0转
FCF9     RTS
```

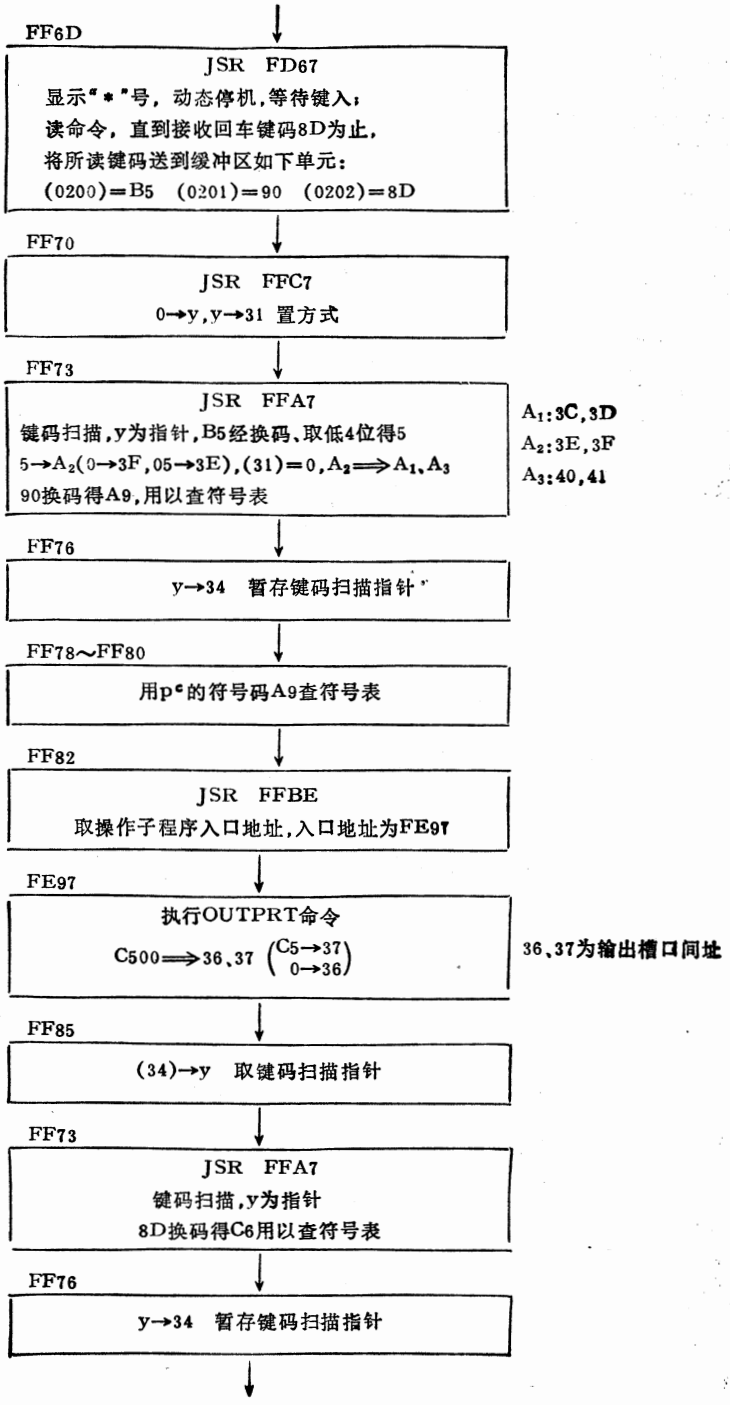
现分析命令＊3100·4FF0R〈CR〉的框图，这条命令的键码是B3 B1 B0 B0 AE B4 C6 C6 B0 D2 8D。

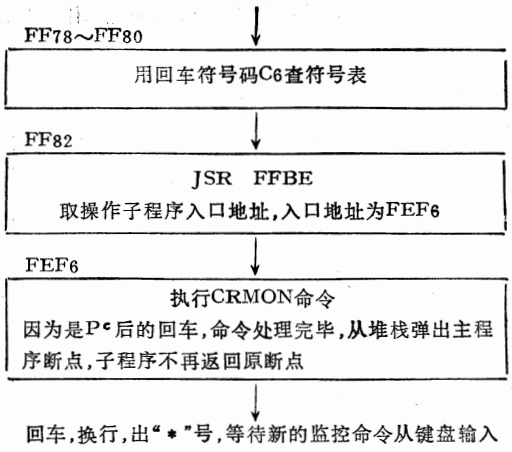



FF76
y→34
暂存键码扫描指针
FF78～FF80
用"·"的符号码A7查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FE18
FE18
执行SETMODE命令
取"·"键码AE→31
FF85
(34)→y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
B4、C6、C6、B0经换码,取低4位得4、F、F、0
4FF0⇒(4F→3F, F0→3E)
(31)≠0,A2不送A1、A3
D2换码得EB,用以查符号表
FF76
y→34
暂存键码扫描指针
FF78～FF80
用"R"的符号码EB查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FEFD

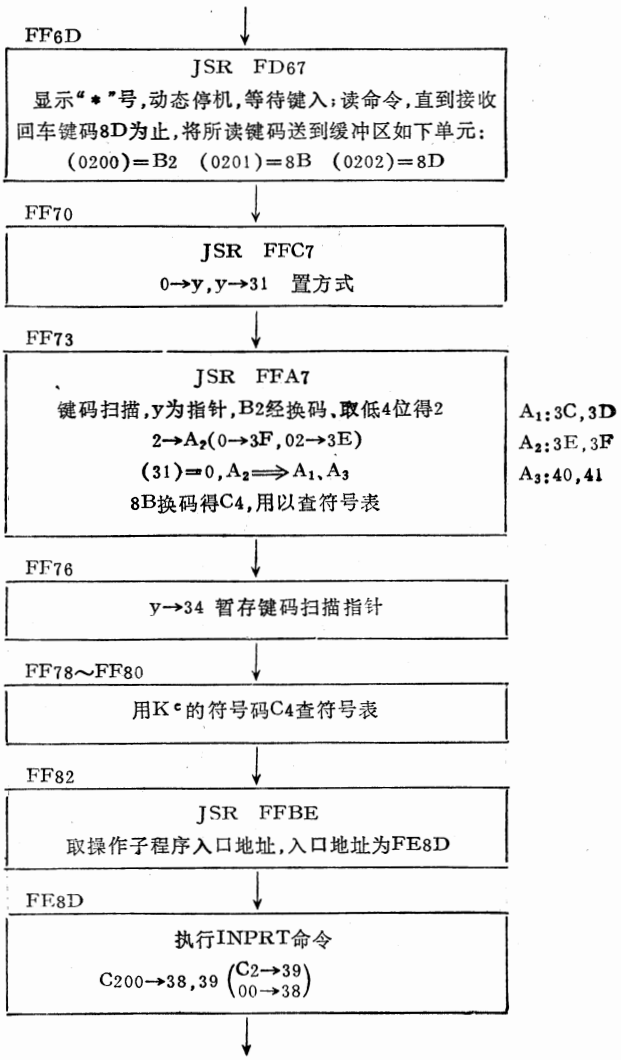

FEFD
执行READ命令
找磁带输入边界
(检测"长1"┌┐或┐┌、┐└、┘)
找到后延时3.6秒
存检验码初始值
(FF→2E)
找"长1"上跳边或下跳边
找同步"短0",没找到就循环,找到后顺序执行
读一个字节,由时间长短确定是1还是0
所读字节⟹(A1)
累计检验码
(A1)+1⟹A1
否
(A1)≥(A2)?
未读完
是,读完
读检验码
检验码对否?
对
错
显示"ERR"
响铃,子程序返回
FF85
(34)→y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
8D换码得C6,用以查符号表
FF76
y→34
暂存键码扫描指针
FF78～FF80
用回车符号码C6查符号表
FF82
JSR FFBE
取操作子程序入口地址,入口地址为FEF6
FEF6
执行CRMON命令
因为是"R"后回车,命令处理完毕,堆栈弹出主程序断点,子程序不返回原断点
FF69
回车,换行,出"*",等待新的监控命令从键盘输入

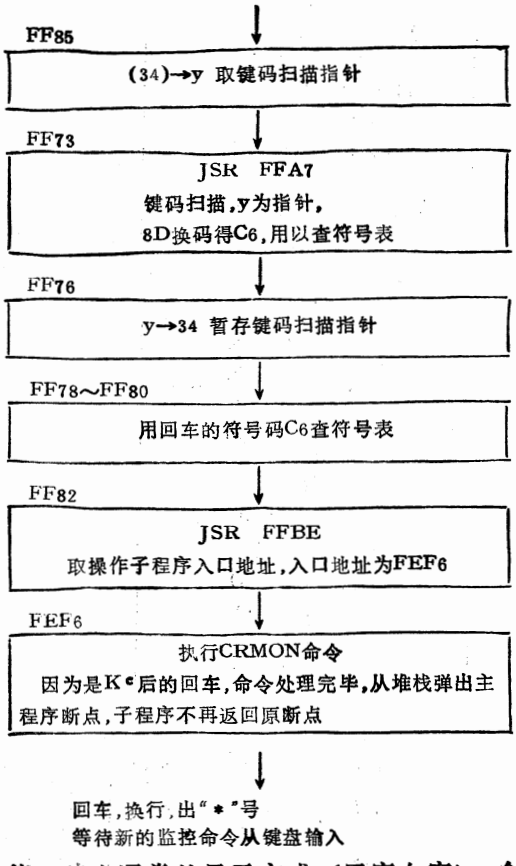
7. *5P^{c}<CR> 这条命令是设置5号槽口为打印机输出，这条命令的键码是：B5 90 8D





8. *2K^c〈CR〉 这条命令是设置2号槽口为键盘输入，这条命令的键码是：B2 8B 8D



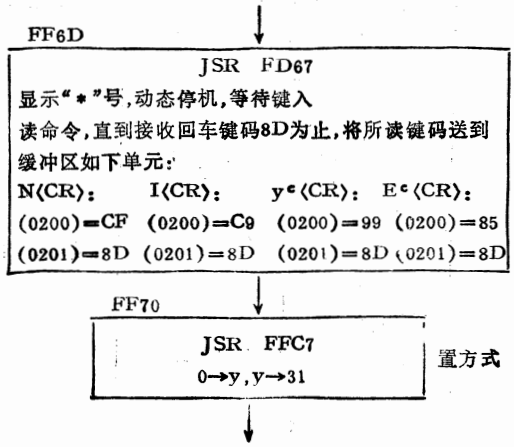


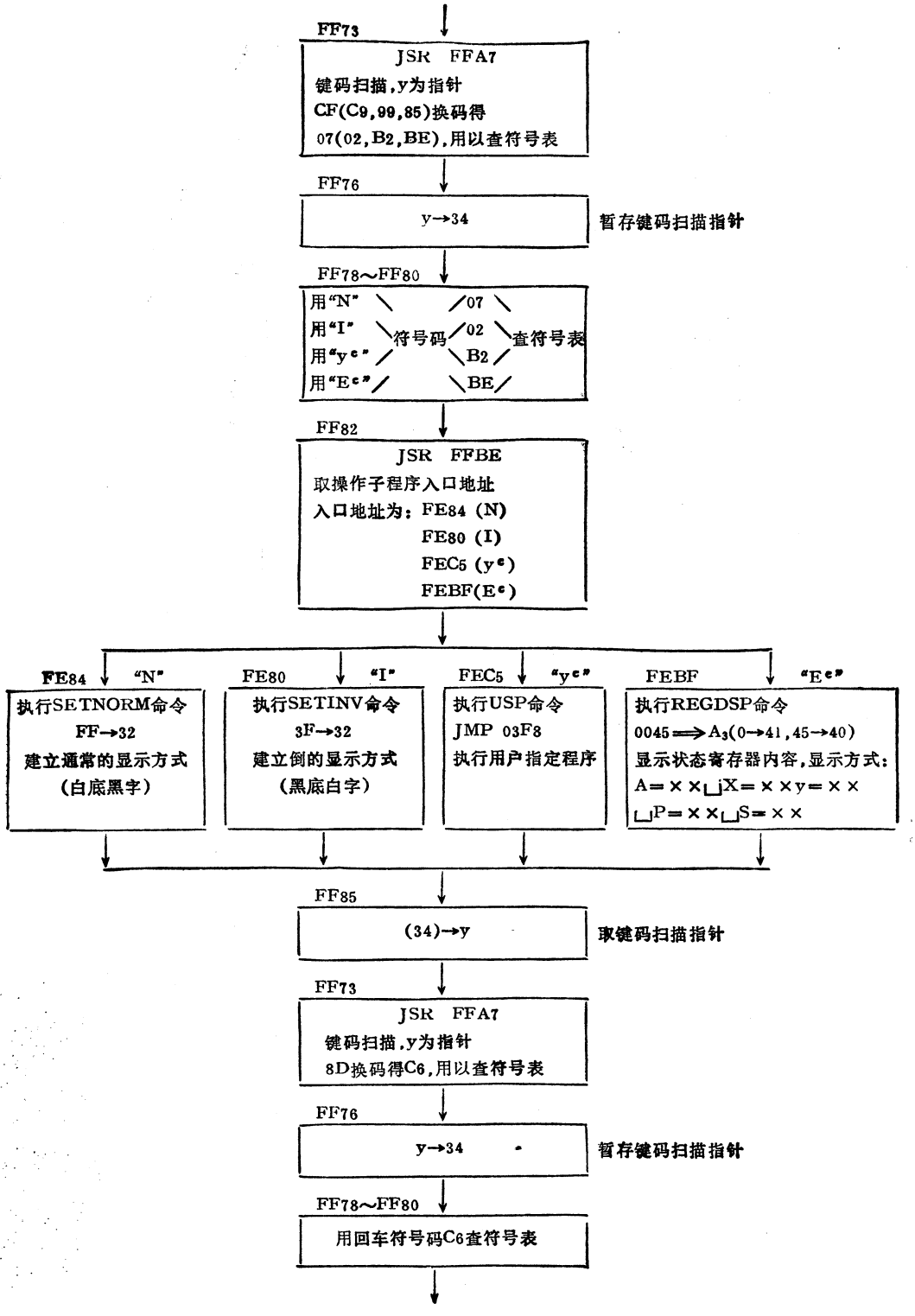
9. *N〈CR〉 功能：建立通常的显示方式（黑底白字），命令键码：CF，8D

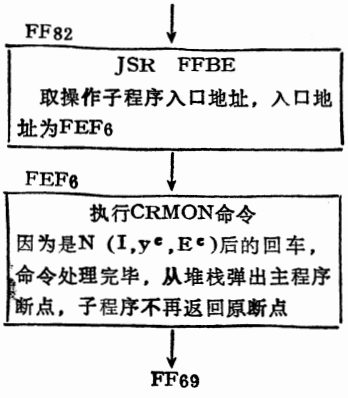
10. *I〈CR〉 功能：建立倒的显示方式（白底黑字），命令键码：C9，8D

11. *Y^c〈CR〉 功能：转至预定地址03F8，执行用户指定程序，命令键码：99，8D

12. *E^c〈CR〉 功能：打开6502状态寄存器，显示其内容，以便修改，命令键码：85，8D



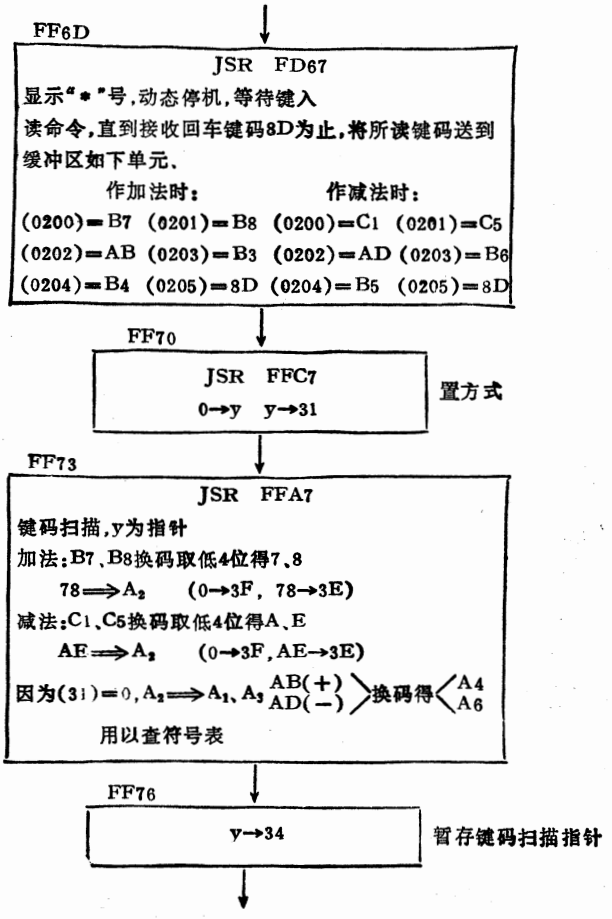

FF73
JSR FFA7
键码扫描,y为指针
CF(C9,99,85)换码得
07(02,B2,BE),用以查符号表
FF76
y→34
暂存键码扫描指针
FF78～FF80
用"N"
用"I"
用"y^e"
用"E^e"
符号码
07
02
B2
BE
查符号表
FF82
JSR FFBE
取操作子程序入口地址
入口地址为: FE84 (N)
FE80 (I)
FEC5 (y^e)
FEBF(E^e)
FE84
"N"
执行SETNORM命令
FF→32
建立通常的显示方式
(白底黑字)
FE80
"I"
执行SETINV命令
3F→32
建立倒的显示方式
(黑底白字)
FEC5
"y^e"
执行USP命令
JMP 03F8
执行用户指定程序
FEBF
"E^e"
执行REGDSP命令
0045⟹A3(0→41,45→40)
显示状态寄存器内容,显示方式:
A=××␣X=××y=××
␣P=××␣S=××
FF85
(34)→y
取键码扫描指针
FF73
JSR FFA7
键码扫描,y为指针
8D换码得C6,用以查符号表
FF76
y→34
暂存键码扫描指针
FF78～FF80
用回车符号码C6查符号表

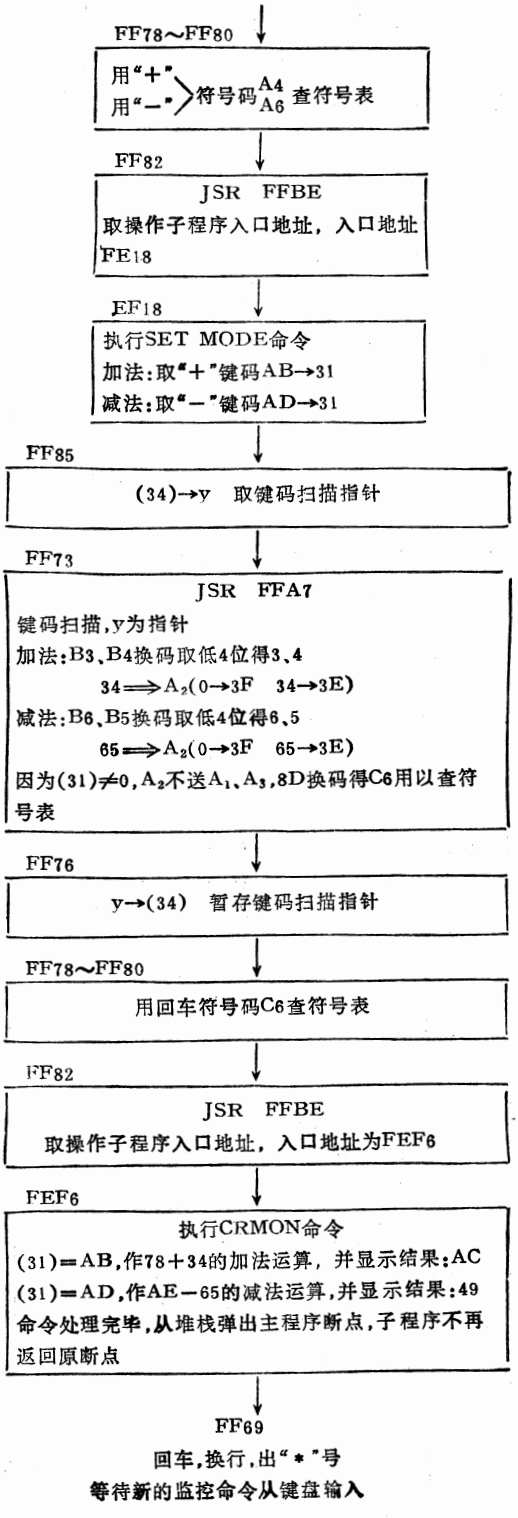


回车，换行，出"*"号，等待新的监控命令从键盘输入

13. *78+34〈CR〉 功能：作十六进制加法运算，命令键码：B7 B8 AB B3 B4 8D

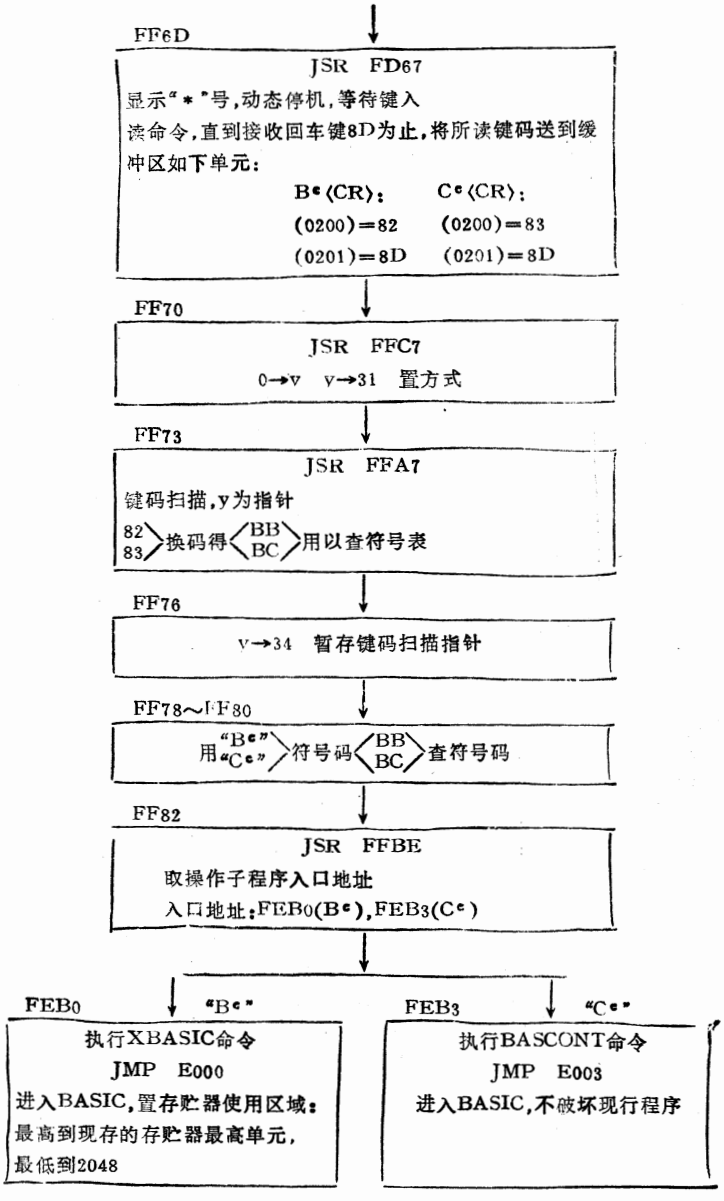
14. *AE-65〈CR〉 功能：作十六进制减法运算，命令键码：C1 C5 AD B6 B5 8D




FF78～FF80
用"+"
用"-"
符号码A4 A6 查符号表
FF82
JSR FFBE
取操作子程序入口地址，入口地址FE18
EF18
执行SET MODE命令
加法：取"+"键码AB→31
减法：取"-"键码AD→31
FF85
(34)→y 取键码扫描指针
FF73
JSR FFA7
键码扫描，y为指针
加法：B3、B4换码取低4位得3、4
34⟹A2(0→3F 34→3E)
减法：B6、B5换码取低4位得6、5
65⟹A2(0→3F 65→3E)
因为(31)≠0，A2不送A1、A3，8D换码得C6用以查符号表
FF76
y→(34) 暂存键码扫描指针
FF78～FF80
用回车符号码C6查符号表
FF82
JSR FFBE
取操作子程序入口地址，入口地址为FEF6
FEF6
执行CRMON命令
(31)=AB，作78+34的加法运算，并显示结果：AC
(31)=AD，作AE-65的减法运算，并显示结果：49
命令处理完毕，从堆栈弹出主程序断点，子程序不再返回原断点
FF69
回车，换行，出"*"号
等待新的监控命令从键盘输入

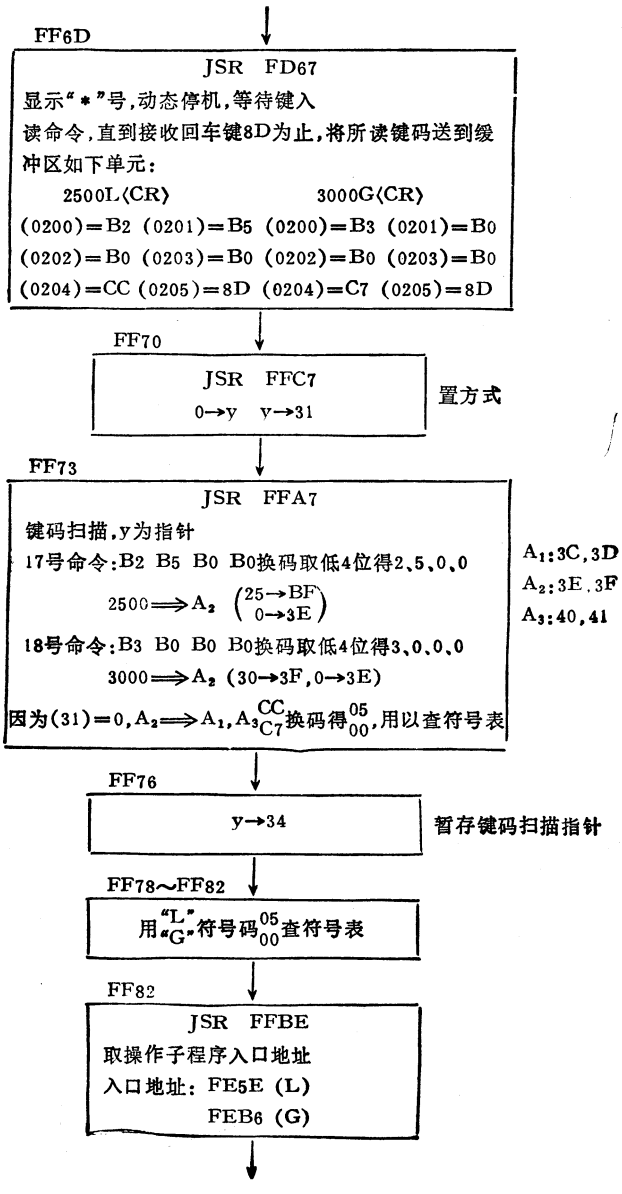
15. *Bᵉ〈CR〉 命令键码：82 8D

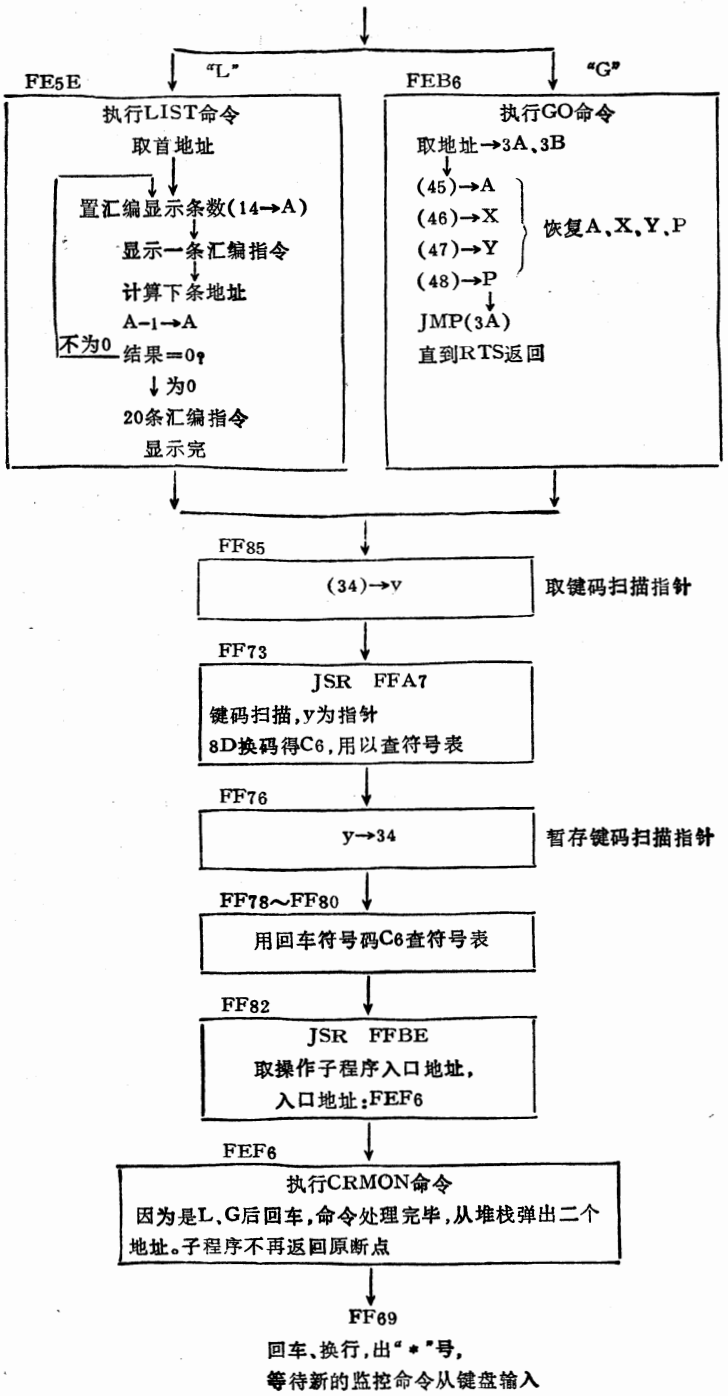
16. *Cᵉ〈CR〉 命令键码：83 8D



17. *2500L〈CR〉 命令键码：B2 B5 B0 B0 CC 8D

18. *3000G〈CR〉 命令键码：B3 B0 B0 B0 C7 8D


FF6D
JSR FD67
显示"*"号,动态停机,等待键入
读命令,直到接收回车键8D为止,将所读键码送到缓
冲区如下单元:
2500L〈CR〉 3000G〈CR〉
(0200)=B2 (0201)=B5 (0200)=B3 (0201)=B0
(0202)=B0 (0203)=B0 (0202)=B0 (0203)=B0
(0204)=CC (0205)=8D (0204)=C7 (0205)=8D
FF70
JSR FFC7
0→y y→31
置方式
FF73
JSR FFA7
键码扫描,y为指针
17号命令:B2 B5 B0 B0换码取低4位得2、5、0、0
2500⟹A2 (25→BF, 0→3E)
18号命令:B3 B0 B0 B0换码取低4位得3、0、0、0
3000⟹A2 (30→3F, 0→3E)
因为(31)=0,A2⟹A1,A3 CC C7换码得 05 00,用以查符号表
A1:3C,3D
A2:3E,3F
A3:40,41
FF76
y→34
暂存键码扫描指针
FF78～FF82
用"L""G"符号码 05 00查符号表
FF82
JSR FFBE
取操作子程序入口地址
入口地址: FE5E (L)
FEB6 (G)



4.6 部分程序注释

1. 回车符<CR>子程序FEF6

监控命令的最后一个键码必定是〈CR〉码8D。如前所述，在执行命令的过程中，只有处理完〈CR〉码才表明命令处理完毕(＊B[c]〈CR〉和＊C[c]〈CR〉除外)。回车符〈CR〉的子程序入口是FEF6。

大部分监控命令的回车符前紧挨的是某一命令符，如＊3100·4FFOR〈CR〉，＊1000＜B010·B410M〈CR〉等。紧挨〈CR〉前是R和M，这类命令的功能主要由回车前的命令符(R、M等)子程序处理。另一类命令，回车符前紧挨的不是命令符，这部分命令有8条（类）：

＊40F2〈CR〉	＊A256：EF␣20〈CR〉
＊1024.1048〈CR〉	＊：F0␣A2〈CR〉
＊〈CR〉	＊78＋34〈CR〉
＊.4096〈CR〉	＊AE－65〈CR〉

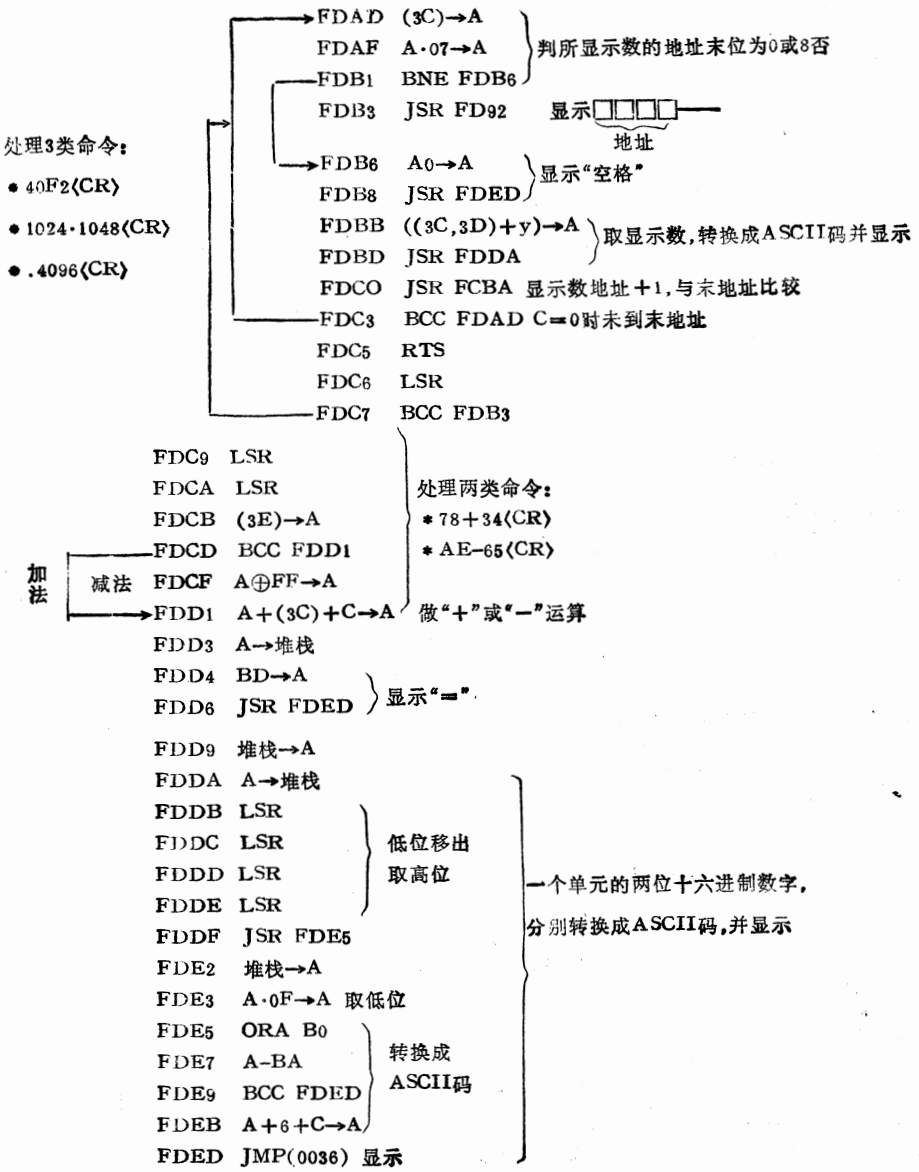
这8条命令的功能主要靠子程序FEF6处理，因此，深入了解子程序FEF6就能正确理解上述8条命令的处理过程，同时，对理解所有监控命令也是有帮助的。

子程序FEF6如下：

```
入口地址：FEF6
FEF6    JSR FE00
FEF9    堆栈→A   ┐
FEFA    堆栈→A   ┘ 整个命令处理完毕,从堆栈弹出主程序断点，程序不再返回原断点
FEFB    BNE FF69
          ↓
        FF69
        回车，换行，显示"＊"号，等待新的监控命令从键盘输入
FE00    (34)-1→34    34寄存键码扫描指针
        FE02  BEQ FDA3
        FE04  x--1→x
        FE05  BNE FE1D
        FE07  A－BA     BA为":"键码
        FE09  BNE FDC6
        FE0B  A→31      ┐ 处理两类命令：
        FE0D  (3E)→A    │ ＊A256:EF␣20〈CR〉
        FE0F  A→(40),y  │ ＊:F0␣A2〈CR〉
        FE11  (40)＋1→40│ 31存方式
        FE13  BNE FE17  │ 3E放数字
        FE15  (41)＋1→41│ 40、41放地址
        FE17  RTS       ┘ y＝0
        FE1D  A→31
        FE1F  RTS
        FDA3  (3C)→A
        FDA5  ORA 07   所显示的末数地址取成"7"或"F"
        FDA7  A→3E     ┐
        FDA9  (3D)→A   ┘ 末地址→3E,3F
        FDAB  A→3F
```

处理1类命令 ＊〈CR〉

回车及空格前为命令符



2．磁盘自起动程序FA62

主要功能：

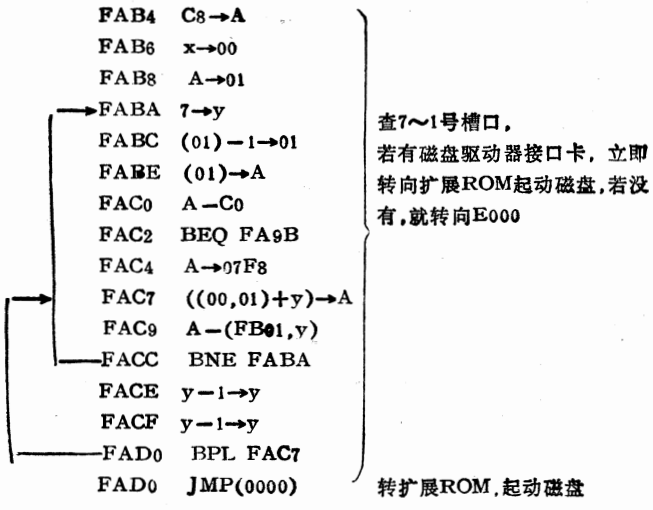
（1）初始化，详见第4.3节；

（2）清屏幕，在屏幕顶部显示商标（如"APPLE Ⅱ"等）；

（3）查I/O槽口，若有磁盘驱动器接口卡，即转向接口卡ROM程序，准备往主机内存调操作系统，否则立即转至E000（BASIC入口）。

程序如下：

```
        FA62
         ↓
FA62    CLD
FA63    JSR FE84  置正常显示方式
FA66    JSR FB2F  设置课文方式,最大窗口        } 初始化
FA69    JSR FE93  置显示器输出入口地址
FA6C    JSR FE89  置键盘输入入口地址
FA6F    LDA C058  清AN0
FA72    LDA C05A  清AN1
FA75    LDA C05D  置AN2                        } AN0～AN3为4个一位输出
FA78    LDA C05F  置AN3
FA7B    LDA CFFF  锁扩展ROM
FA7E    BIT C010  清键入标志
FA81    0→D       置十六进制方式
FA82    JSR FF3A  响铃
FA85    (03F3)→A                               } 开机时,03F3和03F4中为随机数，不可
FA88    A⊕A5→A                                 } 能有如下关系：(03F3)⊕A5=(03F4)。故，
FA8A    A-(03F4)                               } 程序走FAA6
FA8D    BNE FAA6
FA8F    (03F2)→A
←FA92   BNE FAA3
 FA94   E0→A
 FA96   A—(03F3)
←FA99   BNE FAA3
 FA9B   03→y
 FA9D   y→03F2
 FAA0   JMP E000
→FAA3   JMP (03F2)      调操作系统以后：
                        (03F3,03F2)=9DBF
                        不调操作系统：
                        (03F3,03F2)=E003
FB60    JSR FC58        清屏幕
FB63    08→y            C1——→040F
→FB65   (FB08,y)→A      D0——→0410
 FB68   A→040E,y        D0——→0411
 FB6B   y—1→y           CC——→0412
←FB6C   BNE FB65        C5——→0413
FB6E    RTS             A0——→0414
                        DD——→0415
                        DB——→0416
                        即在屏幕顶部显示：APPLE Ⅱ
FAA6    JSR FB60
FAA9    05→x            59——→03F0
→FAAB   (FAFC,x)→A      FA——→03F1
 FAAE   A→03EF,x        00——→03F2
 FAB1   x—1→x           E0——→03F3
←FAB2   BNE FAAB        45——→03F4
```



结 束 语

Apple监控系统结构较紧凑，使用效率较高。把这部分内容分析得透彻，对于灵活使用机器是很有帮助的，同时也为进一步分析、开发微机系统的其他软件打下了基础。欲将它分析清楚，深刻理解监控系统主程序框图是个关键。在此基础上分析几条或大部分监控命令的执行过程，整个监控系统也就面目全清了。

第五章　Apple Ⅱ的小汇编程序

5.1　小汇编程序简介

Apple Ⅱ所配有的小汇编程序的功能是把汇编语言指令转化为机器语言（机器代码）指令，即把便于记忆的以英文字母和其它字符构成的记忆符号指令变换成机器执行用的十六进制操作码。这一转化过程叫作汇编，完成这一变换的程序即汇编程序，Apple Ⅱ的小汇编只能汇编单条指令，不能完成汇编语言的源程序到二进制目的程序的转换。

Apple Ⅱ的MPU是6502微处理器，6502微处理器有56个记忆符号指令名字，13种寻址方式。由不同的记忆符与不同的寻址方式组合后构成151条汇编语言基本指令。小汇编将对这151条汇编语言指令汇编成151条十六进制操作码指令，一条汇编语言指令唯一地对应一条十六进制操作码指令。相反的转换过程叫反汇编，对于Apple Ⅱ，这两种功能都有，本章只讨论汇编程序。

小汇编程序驻留在ROM或盘中，其主程序段约占320个内存地址单元，范围是F500～F63C，详见小汇编程序清单（F500～F666）。对于在执行小汇编程序时要用到的包括在监控程序中的子程序，本章不作分析，需要时可查阅监控部分。

Apple Ⅱ小汇编程序完成一条指令的汇编，可以分以下两个步骤：第一步，计算机对所要汇编的汇编语言指令加以分析处理，把汇编指令中的英文字母记忆符号的ASCII码进行码变换处理，每个字母只取5位二进制码。一般记忆符是由三个英文字母组成，所以共有15位二进制码。最低位补零构成两个二位的十六进制代码，将其存入约定的两个地址单元中。对于汇编指令中的寻址方式加以处理，变换，最后得到一个二位的十六进制的数，以表示这条指令的寻址方式，将其存入另一个约定好的地址单元中。第二步是计算机对可能在00～FF二位十六进制数范围内出现的机器操作代码逐个加以剖析，每取一个二位十六进制代码，经分析查表可得出三个二位十六进制代码，其中之一是表示此操作码相对应的寻址方式。另外二个二位十六进制码是表示该操作码所对应的记忆符号部分。用第二步中分析操作码所得的三个两位十六进制码与第一步中存入三个约定地址中的十六进制码逐个比较，当三对码都符合时，汇编成功，即刚被分析的那个十六进制操作码就是所要汇编的那条指令的操作码，否则换一个操作码，重复上述分析，比较，直到完全符合为止。可见每汇编一条指令，操作码就得一个一个地被检查分析。

从Apple Ⅱ十六进制操作码表中看到，当操作码从00到FF变化时，共有256个码，而Apple Ⅱ实际上只有151条指令，即只有151个机器操作码。所以其中有105个码属于无用操作码，机器遇到这种码时拒绝汇编，以出错处理。Apple Ⅱ所用的这种汇编，方法简单，费时间，但对于指令较少的Apple Ⅱ倒也方便，查找平均次数为128次。

归纳起来，小汇编程序具有以下两个功能：

（1）对于所汇编的汇编语言指令进行语法检查，如有错，则机器拒绝执行，响一声铃，并在错误位置下面打印出一个箭头“∧”。

（2）把汇编语言指令汇编成机器操作码，将汇编结果存入指定地址中，并将结果显示在屏幕上；汇编完成后屏幕上再次出现小汇编催问符“！”，等待下一条输入。

5.2 小汇编程序使用的内存地址约定

A1L 003C
A2L 003E
A3L 0040
} 汇编指令操作数低位暂存

A1H 003D
A2H 003F
A3H 0041
} 汇编指令操作数高位暂存

PCL 003A 参考地址低位暂存
PCH 003B 参考地址高位暂存
A4L 0042 汇编指令记忆符右8位暂存
A4H 0043 汇编指令记忆符左8位暂存
A5L 0044 汇编指令寻址方式暂存
002E 由FMT2表查得的寻址方式暂存
002F 指令字节长度暂存
F962～F9A4 经变换后可求得查FMT2表的变址X
F9A6～F9B3 寻址方式码
F9B4～F9B9 放标点及字符“,”、“)”、“,”、“#”、“(”、“$”
F9BA～F9BF 放字符“Y”、“0”、“X”、“$”、“$”、“0”
F9C0～F9FF 放记忆符号名字的左8位
FA00～FA3F 放记忆符号名字的右8位

5.3 小汇编中的表结构说明

5.3.1 概述

在汇编程序中，几个表结构起着极重要的作用，这节需说明的共有以下几种表：

表5.1：英文字母变换表
表5.2：6502记忆符号指令名字十六进制码表
表5.3：机器操作码分类表
表5.4：记忆符号指令名字分类表
表5.5：A类记忆符表
表5.6：B类记忆符表
表5.7：C类记忆符表
表5.8：D类记忆符表
表5.9：E类记忆符表
表5.10：寻址方式代码表
表5.11：寻址方式书写格式表

通过对以上几个表的含义和构造的理解，以及几个表之间的联系，可以了解小汇编程序的设计思想和巧妙的数据安排，这对了解整个汇编程序是很有帮助的。所以在本章后面的部分将扼要介绍几个表的构成以及它们之间的关系。

我们知道对于每一条汇编语言指令（或者叫记忆符号指令）包括两方面的内容：第

一是记忆符号名字，表示这条指令的基本操作 如“A DC”、“SBC”分别表 示“加”和“减”运算，一般记忆符号名字是用三个英文字母表示，6502微处理器的汇编语言指令共有56个记忆符号名字；第二是寻址方式，即形成执行地址的方式，比如一条加法指令，在“加”的记忆符号名字之后要跟随一个有关“被加数”的执行地址的描述。寻址方式的表达 方式一般由标点符号和其它字符及十六进制的数组合而成，6502微处理器的汇编语言指 令共有13种寻址方式（详见表5.10）。

5.3.2 汇编记忆符号名字表

因为一个记忆符号名字通常以三个英文字母表示，而键盘输入的每个英文字母是 以ASCII码表示（8位二进制），这样三个字母就需要24位二进制或6位十六进制来表示，为了节省内存，可对每个字母的ASCII码进行码变换，每个字母只用5位二 进 制 表 示。从ASCII到5位二进制码的变换过程及结果详见表5.1。

表 5.1 英文字母变换表

英文字母	ASCII码	左移一位后	(A)-BE-$\overline{C}$→A ($\overline{C}$=0)	左移两位后取头5位	对应十进制数
A	41	82	C4	00010	2
B	42	84	C6	00011	3
C	43	86	C8	00100	4
D	44	88	CA	00101	5
E	45	8A	CC	00110	6
F	46	8C	CE	00111	7
G	47	8E	D0	01000	8
H	48	90	D2	01001	9
I	49	92	D4	01010	10
J	4A	94	D6	01011	11
K	4B	96	D8	01100	12
L	4C	98	DA	01101	13
M	4D	9A	DC	01110	14
N	4E	9C	DE	01111	15
O	4F	9E	E0	10000	16
P	50	A0	E2	10001	17
Q	51	A2	E4	10010	18
R	52	A4	E6	10011	19
S	53	A6	E8	10100	20
T	54	A8	EA	10101	21
U	55	AA	EC	10110	22
V	56	AC	EE	10111	23
W	57	AE	F0	11000	24
X	58	B0	P2	11001	25
Y	59	B2	F4	11010	26
Z	5A	B4	F6	11011	27

注意：

（1）英文字母中的ASCII码是7位二进制，如表5.1中第二列所示。键入后高8位自动配1构成8位，左移一位，使进位位为1，所以$\overline{C}=0$。

（2）英文字母的变换在F5C0～F5C8一段程序中完成，详见后面举例。

（3）表5.1是英文字母（原来用ASCII码）经变换后得5位二进制码。

一个记忆符号名字有三个英文字母，可用15位二进制码表示，最低位补一位0构成16位二进制，将16位再分成左8位和右8位两部分，左边8位构成一个二位的十六进制数，占一个内存单元；右边8位构成另一个二位16进制数，占用另一个内存单元。可见，经过变换后，一个记忆符号名字只用两个内存单元就够了。6502微处理器汇编语言记忆符号名字表见表5.2。

注意：

（1）记忆符右8位R只有7位，人为地将最低位补一个“0”构成2位十六进制。

表 5.2 6502汇编指令记忆符号名字十六进制码表

序 号	记忆符名字	记忆符左8位 L	记忆符右8位 R	左H①	右H
1	ADC	00010001	01001000	11	48
2	AND	00010011	11001010	13	CA
3	ASL	00010101	00011010	15	1A
4	BCC	00011001	00001000	19	08
5	BCS	00011001	00101000	19	28
6	BEQ	00011001	10100100	19	A4
7	BIT	00011010	10101010	1A	AA
8	BMI	00011011	10010100	1B	94
9	BNE	00011011	11001100	1B	CC
10	BPL	00011100	01011010	1C	5A
11	BRK	00011100	11011000	1C	D8
12	BVC	00011101	11001000	1D	C8
13	BVS	00011101	11101000	1D	E8
14	CLC	00100011	01001000	23	48
15	CLD	00100011	01001010	23	4A
16	CLI	00100011	01010100	23	54
17	CLV	00100011	01101110	23	6E
18	CMP	00100011	10100010	23	A2
19	CPX	00100100	01110010	24	72
20	CPY	00100100	01110100	24	74
21	DEC	00101001	10001000	29	88
22	DEX	00101001	10110010	29	B2
23	DEY	00101001	10110100	29	B4
24	EOR	00110100	00100110	34	26
25	INC	01010011	11001000	53	C8
26	INX	01010011	11110010	53	F2

续表

序　号	记忆符号名字	记忆符左8位 L	记忆符右8位 R	左H	右H
27	INY	01010011	11110100	53	F4
28	JMP	01011011	10100010	5B	A2
29	JSR	01011101	00100110	5D	26
30	LDA	01101001	01000110	69	46
31	LDX	01101001	01110010	69	72
32	LDY	01101001	01110100	69	74
33	LSR	01101101	00100110	6D	26
34	NOP	01111100	00100010	7C	22
35	ORA	10000100	11000100	84	C4
36	PHA	10001010	01000100	8A	44
37	PHP	10001010	01100010	8A	62
38	PLA	10001011	01000100	8B	44
39	PLP	10001011	01100010	8B	62
40	ROL	10011100	00011010	9C	1A
41	ROR	10011100	00100110	9C	26
42	RTI	10011101	01010100	9D	54
43	RTS	10011101	01101000	9D	68
44	SBC	10100000	11001000	A0	C8
45	SEC	10100001	10001000	A1	88
46	SED	10100001	10001010	A1	8A
47	SEI	10100001	10010100	A1	94
48	STA	10100101	01000100	A5	44
49	STX	10100101	01110010	A5	72
50	STY	10100101	01110100	A5	74
51	TAX	10101000	10110010	A8	B2
52	TAY	10101000	10110100	A8	B4
53	TSX	10101101	00110010	AD	32
54	TXA	10101110	01000100	AE	44
55	TXS	10101110	01101000	AE	68
56	TYA	10101110	10000100	AE	84

①下角“H”表示十六进制

（2）Apple Ⅱ将表5.2中的右边8位（记忆符）放入内存地址FA00～FA3F范围内，命名为“MNEMR表”；将表5.2中的记忆符左8位放入内存地址F9C0～F9FF范围内，命名为“MNEML表”。

（3）一个记忆符号名字，必须在“MNEML表”中查得8位，然后再在“MNEMR表”中取另外8位，两部分拼在一起构成一个完整的记忆符名字（详见程序举例）。

5.3.3 十六进制操作码分类表

6502微处理器共有151个基本指令，从指令操作码来看，有一些指令的最末三位总是

"000"；有一些指令的最末三位总是"100"；有一些指令的最末四位总是"1010"；还有一些指令码的最末两位是"10"和"01"。根据这些特点，可以把151条基本指令分成A、B、C、D、E5大类，构成表5.3。

表 5.3 机器操作码分类表

类 别	操 作 码	操作码特征	指令条数
A类	XXXXX000	低4位是0_H、8_H①	31
B类	XXXYY100	低4位是4_H、C_H	15
C类	1XXX1 010	低4位总是A_H	6
D类	XXXYYY10	末两位是10	36
E类	XXXYYY01	末两位是01	63
总数			151

①下角"H"表示十六进制

按末4位，末3位和末两位的特征同样可把56条记忆符号指令的名字分类得表5.4：

表 5.4 记忆符号名字分类表

类别	操 作 码	记 忆 符 号 名 字	记忆符号指令条数
A类	XXXXX000	其余	28条
B类	XXXYY100	BIT，JMP，STY，LDY，CPY，CPX。	6条
C类	1XXX1 010	TXA，TXS，TAX，TSX，DEX，NOP。	6条
D类	XXXYYY10	ASL，ROL，LSR，ROR，STX，LDX，DEC，INC。	8条
E类	XXXYYY01	ORA，AND，EOR，ADC，STA，LDA，CMP，SBC。	8条
总数			56条

注意：

（1）比较表5.3与表5.4，发现记忆符号名字表中的分类条数与表3中同类的操作码条数不等，这当然是正确的。因为同一记忆符号名字由于寻址方式的不同，对应的操作码数要增加。如E类记忆符号名字，每条有8种寻址方式，所以每条对应8个操作码，8条对应64个操作码，但实际上只有63个操作码，因为有一个记忆符号名字只有七种寻址方式。

（2）C类记忆符名字的指令只有一种寻址方式，所以六个名字只对应6条操作码。

（3）A类记忆符号名字指令只有一种寻址方式，本来28条记忆符号名字指令应对应28个操作码，但由于B类中的操作码"C0"，"A0"，"E0"最后三位不符合"100"，所以将这三条操作码合并到A类中。因此，A类增加三条成为"31"。

在表5.3和表5.4中，除了分类特征码外，还有"X""Y"，其中"X"表示记忆符号名字特征，而"Y"表示指令的寻址方式特征。为了说明操作码中哪几位二进制码表示记忆符名字的特征，哪几位二进制码表示寻址方式特征，请看下面各类操作码与记忆符号名字的关系表（表5.5～表5.9）。

表 5.5 A类XXXXX000（由二进制码看到，低三位全为0，其余六位为X）

记忆符	操作码	二进制码	记忆符	操作码	二进制码
BRK	00	00000000	SEI	78	01111000
PHP	08	00001000	DEY	88	10001000
BPL	10	00010000	BCC	90	10010000
CLC	18	00011000	TYA	98	10011000
JSR	20	00100000	TAY	A8	10101000
PLP	28	00101000	BCS	B0	10110000
BMI	30	00110000	CLV	B8	10111000
SEC	38	00111000	INY	C8	11001000
RTI	40	01000000	BNE	D0	11010000
PHA	48	01001000	CLD	D8	11011000
BVC	50	01010000	INX	E8	11101000
CLI	58	01011000	BEQ	F0	11110000
RTS	60	01100000	SED	F8	11111000
PLA	68	01101000	LDY	A0	10100000
BVS	70	01110000	CPY	C0	11000000
			CPX	E0	11100000

表 5.6 B类操作码XXXYY100

记忆符	操作码			
	十六进制	二进制		
		XXX	YY	100
BIT	24	001	00	100
BIT	2C	001	01	100
JMP	4C	010	01	100
JMP	6C	011	01	100
STY	84	100	00	100
STY	9C	100	01	100
STY	8C	100	01	100
LDY	A4	101	00	100
LDY	B4	101	10	100
LDY	AC	101	01	100
LDY	BC	101	11	100
CPY	C4	110	00	100
CPY	CC	110	01	100
CPX	E4	111	00	100
CPX	EC	111	01	100

表 5.7 C类操作码1XXX1010

记忆符	操作码			
	十六进制	二进制		
		1	XXX	1010
TXA	8A	1	000	1010
TXS	9A	1	001	1010
TAX	AA	1	010	1010
TSX	BA	1	011	1010
DEX	CA	1	100	1010
NOP	EA	1	110	1010

表 5.8 D类操作码XXXYYY10

记忆符	操作码		记忆符	操作码	
	十六进制	二进制		十六进制	二进制
		XXX YYY 10			XXX YYY 10
ASL	0A	000 010 10	ROL	2A	001 010 10
ASL	06	000 001 10	ROL	26	001 001 10
ASL	16	000 101 10	ROL	36	001 101 10
ASL	0E	000 011 10	ROL	2E	001 011 10
ASL	1E	000 111 10	ROL	3E	001 111 10
LSR	4A	010 010 10	ROR	6A	011 010 10
LSR	46	010 001 10	ROR	66	011 001 10
LSR	56	010 101 10	ROR	76	011 101 10
LSR	4E	010 011 10	ROR	6E	011 011 10
LSR	5E	010 111 10	ROR	7E	011 111 10
STX	86	100 001 10	LDX	A2	101 010 10
STX	96	100 101 10	LDX	A6	101 001 10
STX	8E	100 011 10	LDX	B6	101 101 10
DEC	C6	110 001 10	LDX	AE	101 011 10
DEC	D6	110 101 10	LDX	BE	101 111 10
DEC	CE	110 011 10	INC	E6	111 001 10
DEC	DE	110 111 10	INC	F6	111 101 10
			INC	EE	111 011 10
			INC	FE	111 111 10

表 5.9 E类操作码XXXYYY01

记忆符	操作码		记忆符	操作码	
	十六进制	二进制		十六进制	二进制
		XXX YYY 01			XXX YYY 01
ORA	01	000 000 01	AND	21	001 000 01
ORA	11	000 100 01	AND	31	001 100 01
ORA	05	000 001 01	AND	25	001 001 01
ORA	15	000 101 01	AND	35	001 101 01
ORA	09	000 010 01	AND	29	001 010 01
ORA	19	000 110 01	AND	39	001 110 01
ORA	0D	000 011 01	AND	2D	001 011 01
ORA	1D	000 111 01	AND	3D	001 111 01
EOR	41	010 000 01	ADC	61	011 000 01
EOR	51	010 100 01	ADC	71	011 100 01
EOR	45	010 001 01	ADC	65	011 001 01
EOR	55	010 101 01	ADC	75	011 101 01
EOR	49	010 010 01	ADC	69	011 010 01
EOR	59	010 110 01	ADC	79	011 110 01
EOR	4D	010 011 01	ADC	6D	011 011 01
EOR	5D	010 111 01	ADC	7D	011 111 01
STA	81	100 000 01	LDA	A1	101 000 01
STA	91	100 100 01	LDA	B1	101 100 01
STA	85	100 001 01	LDA	A5	101 001 01
STA	95	100 101 01	LDA	B5	101 101 01
STA	99	100 110 01	LDA	A9	101 010 01
STA	8D	100 011 01	LDA	B9	101 110 01
STA	9D	100 111 01	LDA	AD	101 011 01
CMP	C1	110 000 01	LDA	BD	101 111 01
CMP	D1	110 100 01	SBC	E1	111 000 01
CMP	C5	110 001 01	SBC	F1	111 100 01
CMP	D5	110 101 01	SBC	E5	111 001 01
CMP	C9	110 010 01	SBC	F5	111 101 01
CMP	D9	110 110 01	SBC	E9	111 010 01
CMP	CD	110 011 01	SBC	F9	111 110 01
CMP	DD	110 111 01	SBC	ED	111 011 01
			SBC	FD	111 111 01

从表5.5到表5.9可以看出：

（1）A类操作码最后三位全是0，前面5位从00000～11111共32个码，除了80之外，其余各代表一个记忆符号名字。

（2）B类操作码倒数第0、1、2三位为“100”，表示分类特征。头三位第7、6、5三位为“001”，“100”，“101”，“110”，“111”共5种码，分别代表5种记忆符号名字。中间两位

为“00”,“01”，“10”三个码对应三种不同寻址方式。

（3）C类码最高位总是1，低4位总是“1010”，第6、5、4三位代码对应6种记忆符号名字。

（4）D类最末两位为“10”，第7、6、5三位共8种码对应8种记忆符号名字，第4、3、2三位对应5种不同的寻址方式。

（5）E类最末两位为“01”，第7、6、5位沿水平方向变化，分别对应8种不同的记忆符号名字。第7、6、5三位在每一列都是相同的，因为它们对应的是同一个记忆符名字，可见这三位是记忆符名字的特征。第4、3、2三位沿列方向变化，表示不同寻址方式。同样道理，对于D、C、B、A类也一样，“X”表示记忆符特征，“Y”表示寻址方式特征。

5.3.4 查找记忆符名字的方法

了解了上述几种表的结构之后，现在需要分析一下如何使用存放在内存中的MNEML表和MNEMR表来确定一个十六进制操作码所对应的记忆符号名字。这里还要说明一点，MNEML表和MNEMR表是以A、B、C、D、E类分类排放的，所以对于一个已知的操作码，首先要判断该码属于哪一类，然后根据不同类的码进行不同的变换，目的是保留“X”特征，去掉“Y”特征。

变换结果

A类：XXXXX000⇒000XXXXX

B类：XXXYY100⇒00100XXX

C类：1XXX1010⇒00101XXX

D类：XXXYYY10⇒00110XXX

E类：XXXYYY01⇒00111XXX

箭头左是机器原操作码，箭头右是经变换后得查名字表的变址。

例如已知一机器操作码为“96”，

1 0 0 1 0 1 1 0

XXXYYY ←末两位为“10”属于D类

变换得：

XXXYYY10⇒00110XXX＝00110100

所以变址值为34

由 F9C0＋34查MNEML表得A5

FA00＋34查MNEMR表得72

将A572与表5.2对照得STX。

5.3.5 查找寻址方式的方法

FMT1表存储在地址F962～F9A4范围内，FMT2表存储在地址F9A6～F9B3范围内。FMT2中存储的内容是操作码对应的寻址方式，共13种（见表5.10）。FMT1表中存储的内容再经过变换是查FMT2时的变址，而FMT1的变址是由十六进制操作码经变换而得

（详见程序举例）。

表 5.10 寻址方式代码表

序号	代码	格　式	说　明
1	00	ERR	错　误
2	21	IMM	立即数
3	81	ZPAG	0页地址
4	82	ABS	绝对地址
5	00	IMPLIED	隐寻址
6	00	ACCUMULATOR	累加器寻址
7	59	(ZPAG,X)	0页变址X间址
8	4D	(ZPAG),Y	0页间址变址Y
9	91	ZPAG,X	0页地址变址X
10	92	ABS,X	绝对地址变址X
11	86	ABS,Y	绝对地址变址Y
12	4A	(ABS)	绝对地址间址
13	85	ZPAG,Y	0页变址Y
14	9D	RELATIVE	相对寻址

5.4 小汇编程序的汇编过程

5.4.1 启动小汇编

在监控命令状态下按F666G，机器将进入小汇编，进入小汇编的标志是响一声铃，屏幕上显示催向符“！”。游标闪烁，说明机器已进入小汇编状态等待输入汇编命令。

5.4.2 输入汇编语言指令

在催向符“！”下，当你按入一条汇编命令时，机器将该条命令存入0200～02FF缓冲区中。汇编命令的输入格式有以下三种：

（1）！1234：STA $ 2346

（2）！␣STA $ 2789

（3）！$ C8ϕϕL

第一种格式表示把一条汇编指令如“STA $ 2346”汇编成机器操作码，汇编结果顺序存入参考单元地址1234中，例如“8D”存1234单元，低位地址“46”存入1235地址单元，高位地址“23”存入1236单元中。

第二种格式表示继续汇编下一条指令，这时参考地址可以省略，但要在“！”与STA之间按一个空格。机器将汇编结果存入下面可利用的地址单元，如上面两条排在一起时，则在1237单元中存入“8D”，地址低位“89”存入1238单元中，地址高位“27”存入1239单元中。

第三种情况是在小汇编状态下，如果想执行监控命令，只要在“!”后按一个“$”，则后面可按入任何监控命令，当一条监控命令执行完后，机器将自动返回小汇编。但要注意，在执行完监控命令之后，汇编下一条汇编指令时，参考地址（存储汇编结果的地址单元）要重新按一次，以免发生错误。

机器对于输入汇编指令格式，首先要进行错误检查，如发现有错，则拒绝汇编，响铃，并在错误之处打箭头“Λ”以提醒注意。另外对于上述三种格式的输入，机器识别后转入不同指针地址执行。

机器将输入的汇编指令一个字符一个字符地从缓冲区取出，如果没有错误，即把“!”后的参考地址存入003B（存高位）和003A（存低位）保存。

5.4.3 机器对记忆符号名字的处理

机器对参考地址和冒号后的记忆符号名字的处理，首先将用ASCII码表示的英文字母进行码变换，变换的结果是将ASCII码变成5位二进制，例如英文字母“S”，它的ASCII码是“53”，变换过程如下：

（1）原码　　1010011“53”

（2）最高位补1　11010011

（3）左移一位　　10100110“A6”（C=1）

（4）减　　A6－BE－$\overline{C}$=E8

（5）左移两次取头5位得“10100”

同理，对其余两个英文字母也这样处理。三个输入的英文字母码对应15位二进制码，最低位补一个0，凑足16位以表示一个记忆符号名字。假如你按入的记忆符名字是“CMP”，则它对应的16位二进制码是0010、0011、1010、0010，再把它分成左8位用2位十六进制数表示，即“23”；右8位用另外2位十六进制数表示，即A2。机器把左8位“23”存入约定地址单元0043中，把右8位“A2”存入约定地址单元0042中。

对于任何输入的汇编语言记忆符号名字，机器都要做如上处理，得出2个二位十六进制码，以表示这个记忆符名字（详见表5.2中6502记忆符号十六进制码），并把这两对数存入0042和0043单元中以备最后比较时使用。

5.4.4 机器对寻址方式的处理

6502微处理器的汇编指令的寻址方式共有13种（见表5.10），这13种方式对应13个十六进制代码。现在需要讨论机器如何根据输入的汇编指令求出它的寻址方式十六进制代码。为了说明问题，先看表5.11，寻址方式书写格式与代码对照关系。

从表5.11寻址方式的表达式中看出，寻址方式的表达方法是十六进制数和一些字符的组合，用到的字符有

“$”,“(”,“#”,“,”,“)”及“X”,“Y”

而且这些字符之间与十六进制数的组合有一定规律。

其中：

（1）“$”之后一般都是表示地址的数

表 5.11 寻址方式书写格式表

序号	寻址方式	汇编语言形式举例	操　作	寻址方式代码
1	立即数	CMP # $57	(A)—57	21
2	0页地址	CMP $55	(A)—(0055)	81
3	绝对地址	CMP $3355	(A)—(3355)	82
4	隐寻址	CLC	0→C	00
5	累加器寻址	ASL A	←	00
6	0页变址X间址	CMP($55,X)	(A)—(0056+(X))(0055+(X))	59
7	0页间址变址Y	CMP($55),Y	(A)—((0056)(0055)+(X))	4D
8	0页地址变址X	CMP $55,X	(A)—(0055+(X))	91
9	绝对地址变址X	CMP $3355,X	(A)—(3355+(X))	92
10	绝对地址变址Y	CMP $3355,Y	(A)—(3355+(Y))	86
11	绝对地址间址	JMP($3355)	跳转3355	4A
12	0页变址Y	LDX $55,Y	(0055)+(Y))→X	85
13	相对寻址	BEQ $55		9D

（2）“(”之后有可能是数，也有可能是“$”

（3）“#”之后是“$”

（4）“,”后一般是X或Y字母

（5）“)”之后有“，”或“Y”字母

（6）“$”,“(”，“#”字符出现在十六进制数之前，而“,”，“)”，“Y”，“X”出现在十六进制数之后。

根据上述这些特点，机器在读寻址方式时，一般分三个步骤：

（1）读十六进制地址数前可能出现的字符

（2）读十六进制地址数

（3）读十六进制地址数后可能出现的字符

检查时把上述字符配对检查，配对列表如下：

,	)	,	地址后的第一个字符	#	(	$	记忆符名字后的第一个字符
Y	0	X	地址后的第二个字符	$	$	0	记忆符名字后的第二个字符

检查步骤如下：

第一：检查记忆符后是否“$”？接着将“$”后的字符与“0”比较，这实际上是一个空动作，因为“$”后一般没有别的字符。

第二：检查记忆符后的第一个字符是否“(”？如果是，接着检查“(”后是否是“$”？

第三：检查记忆符后的第一个字符是否“#”？如果是，接着检查“#”后是否是“$”？

以上三步检查完后，读操作数（十六进制地址或立即数），接着检查三次操作数后

面可能会出现的其余符号。

第四：检查操作数后的第一个字符是否“，”？如果是，接着检查紧跟着的是否“X”

第五：检查操作数后的第一个字符是否“)”，如果是，接着将跟随圆括弧之后的字符与“0”字符比较。这一步实际上是一个空动作（详见程序）。

第六：检查是否“，”，如果是，后面是否是“Y”？经过以上六次检查，便可确定寻址方式，最后形成一个十六进制码，将此码存入0044单元中，以备比较时使用。

需要说明一点，对于相对寻址方式，除上述循环查找之外，还要单独处理（详见程序举例）。

读完寻址方式后，如果继续读时遇到回车，机器停止读数，这时说明一条输入的汇编指令已被读完，经过处理之后已把输入的信息码保存到指定单元，等待下一步比较时使用。

003A：存储参考地址低位

003B：存储参考地址高位

0042：存储汇编指令记忆符名字右8位

0043：存储汇编指令记忆符号名字左8位

0044：存储汇编指令寻址方式代码

5.4.5 十六进制操作码的扫描

机器对上述键入的汇编指令分析，存储之后，汇编过程仍未结束，留下的问题是如何找出输入汇编指令的十六进制操作码。方法是机器对003D单元中的十六进制码逐个分析（简称扫描），每取一个十六进制码，对它加以剖析，得出三个十六进制码。其中两个代表与该操作码对应的记忆符号名字，另一个代表寻址方式。将这三个码与0042、0043、0044单元中的内容逐个比较，如果完全符合，说明当前003D中的十六进制操作码恰好就是输入汇编指令的操作码，否则003D换一个码，重复上述分析比较，直到完全符合为止。

下面需要说明一下机器如何将003D中的一个十六进制码分解成记忆符号名字的十六进制码和寻址方式十六进制码。这段工作由子程序F88E完成，详见程序举例，下面简要说明一下思路。

5.4.6 寻址方式代码的确定

将003D中的码送累加器A，首先排除无用操作码，因为从6502的十六进制操作码表看出，有151个码是有用指令，105个码是无用操作。而003D中的码变换与十六进制操作码表完全一致，也有256种可能，其中有151个码是有用的，其余105个码是无用操作码，机器遇到无用操作码就跳过去不汇编，使003D换一个码再继续执行。

最末两位都是1的码，还有“89”是无用操作码，另外还有40个无用操作码。由003D取出代码，首先排除这些无用操作码后，如果遇到有用码，经过变换，得出一个查FMT1表的变址值，由FMT1表查得的结果再处理得另一个变址值去查FMT2表，由FMT2表得出的十六进制码即为寻址方式代码，将这个码存入002E单元中。

5.4.7 记忆符号名字十六进制码的确定

寻址方式码确定后，机器把相应的003D中的内容再送回累加器A进行分析，目的是求出记忆符号名字。求记忆符号名字的方法是，据003D中的内容经过变换，求出一个“变址值”，由这个变址值去查MNEML表和MNEMR表。

由前面分析过的内容所知，MNEML表和MNEMR表的排列次序是按A、B、C、D、E分类排列的，即同类的记忆符名字放在一起。所以在查表前首先要确定一下该操作码属于哪一类，知道其所属类别之后，再求“变址值”。“变址值”的求取方法各类略有区别：

A类机器操作码：

原码：XXXXX000
第一次右移：0XXXXX00
第二次右移：00XXXXX0
第三次右移：000XXXXX

A类
XXXXX000⇒000XXXXX

B类机器操作码：

原码：XXXYY100
第一次右移：0XXXYY10
第二次右移：00XXXYY1
第三次右移：000XXXYY
第四次右移：0000XXXY
第五次右移：00000XXX
与20V（或）：00100XXX

B类
XXXYY100⇒00100XXX

C类机器操作码：

原码：1XXX1010
第一次右移：01XXX101
第二次右移：001XXX10
第三次右移：0001XXX1
第四次右移：00001XXX
与20V（或）：00101XXX

C类
1XXX1010⇒00101XXX

D类机器操作码：

原码：XXXYYY10
第一次右移：0XXXYYY1
第二次右移：00XXXYYY
第三次右移：000XXXYY
第四次右移：0000XXXY
与 20V ：0010XXXY
第五次右移：00010XXX
与 20V ：00110XXX

D类：
XXXYYY10⇒00110XXX

E类机器操作码：

原码：XXXYYY01
第一次右移：0XXXYYY0
第二次右移：00XXXYYY
第三次右移：000XXXYY
与 20V　：001XXXYY
第四次右移：0001XXXY
与 20V　：0011XXXY
第五次右移：00011XXX
与 20V　：00111XXX

E类

XXXYYY01⇒00111XXX

经过变换，最后得出：

A类：XXXXX000⇒000XXXXX

B类：XXXYY100⇒00100XXX

C类：1XXX1010⇒00101XXX

D类：XXXYYY10⇒00110XXX

E类：XXXYYY01⇒00111XXX

箭头左边是原码，箭头右边是查记忆符名字的变址值。例如003D中的码是8D，首先确定它属于E类，按E类变换，得变址00111XXX，XXX＝100，代入得变址值为00111100＝3C，以3C值查MNEML表和MNEMR表，可得出记忆符号名字为“STA”。

5.4.8 比较判决输出结果

对003D中操作码的扫描分析，得出其对应的寻址方式码与记忆符号名字代码，与输入汇编语言指令经变换后已存入约定地址单元中的内容逐个比较，即将扫描变换得出的寻址方式码与0044中的内容比较；扫描变换查MNEML表得来的结果与0043中的内容比较；扫描变换查MNEMR表得来的结果与0042中的内容比较。三次比较结果都符合则汇编成功，三次比较中有一次不符合都算失败，003D换一个码再变换比较，直到从003D中找出一个码与输入汇编指令符合为止。如果全符合了，说明当前003D中的十六进制码就是输入汇编指令所对应的机器操作码。如果全符合，则还要做下面几件事：

（1）汇编结果送参考地址保存，例如输入汇编指令为“↓1238:STA $3456”则将

“8D”送1238单元

“56”送1239单元

“34”送123A单元

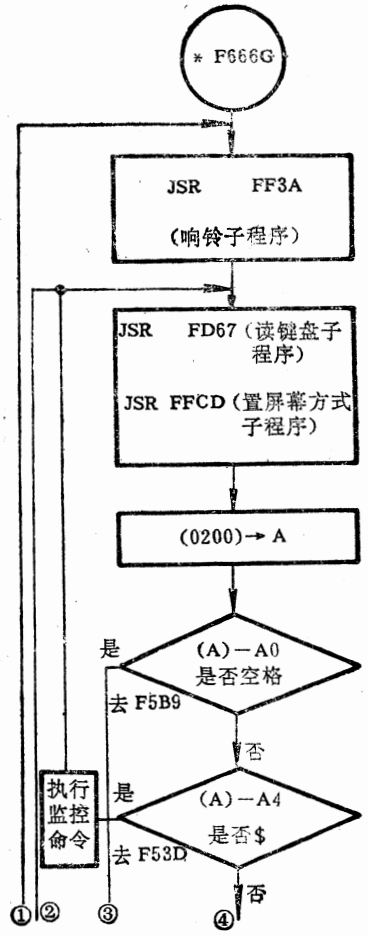
（2）将汇编结果显示于屏幕，显示结果前把原先的输入清除，游标向上滚两行，然后显示参考地址汇编结果，在汇编结果后，显示该条汇编指令。显示格式：

1238- 8D 56 34 STA $3456

（3）计算新的指令地址，并保存于003B和003A单元中。如上例中执行完该条指令，003B中的内容是“12” 003A中的内容是“3B”。

（4）转读键盘子程序，屏幕显示“↓”，等下一条输入。

5.5 小汇编程序的总体框图



在监控状态下，按 * F666G 启动小汇编

响铃

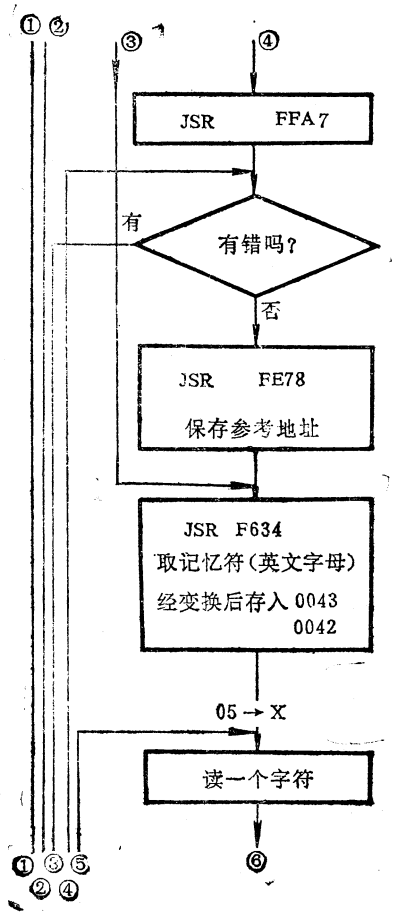
显示"！"，出现闪烁游标。

按一行命令，自动存入 0200～02FF 缓冲区，遇到回车停止存。

判存入 0200 单元中的第一个字符。

是空格，去 F5B9，汇编指令存入下一个可利用的地址单元。

是 $，去 F53D，执行监控命令后重新返回小汇编。



读参考地址。

是否":"及其它错误。

地址高位存入 PCH (003B)。

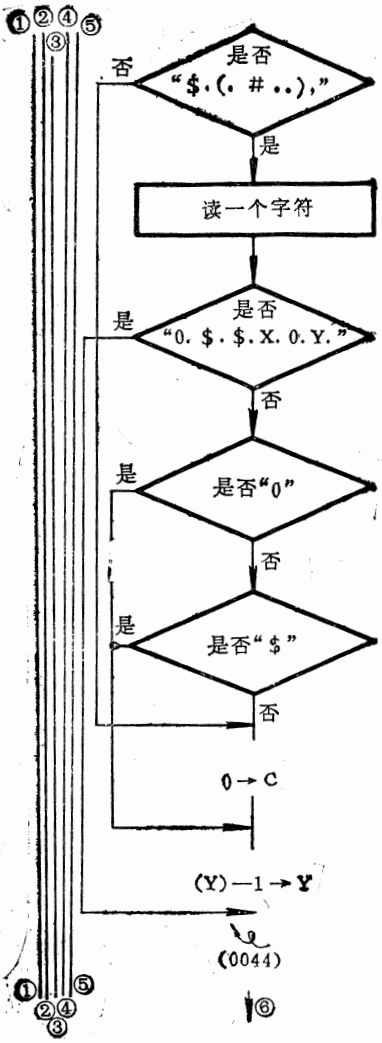
地址低位存入 PCL (003A)。

记忆符号左 8 位存入 0043 单元。

记忆符号右 8 位存入 0042 单元。

X 作为计数(检查六种情况)。

读记忆符号名字后的第一个字符。



当 X = 5, 4, 3 时，判是否"$"，"("，"#"？当 X = 2, 1, 0 时，查","，")"，","？如都不是，则跳过

读记忆符名字后的第二个字符。

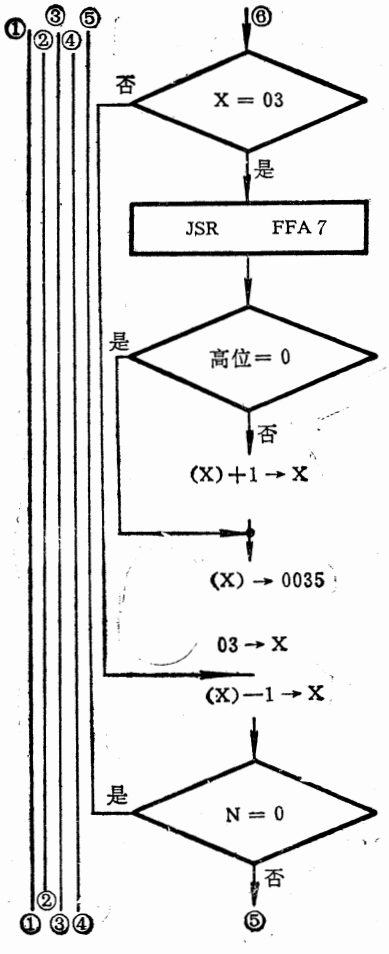
当 X = 5, 4, 3 且第一个字符是"$"，"("，"#"时，查后面是否为"0"，"$"；
当 X = 2, 1, 0 且第一个字符是","，")"，","时，查后面是否为"X"，"0"，"Y"。

在"$"和")"之后 0→A 使程序跳过 0→C。

当 X = 4 第一个字符是"#"，第二个又不是"$"时，使程序跳过 0→C。

0044 中形成格式化字节（寻址方式代码的中间结果）。

遇到符合的字符0044 中循环进入一个"1"，否则是"0"，共六次。



X = 03 转 FFA7 子程序读操作数。

X ≠ 03 继续循环检查。

由 FFA7返回时， (X) = 00 没有读操作数，
(X) = 01 读完操作数后。

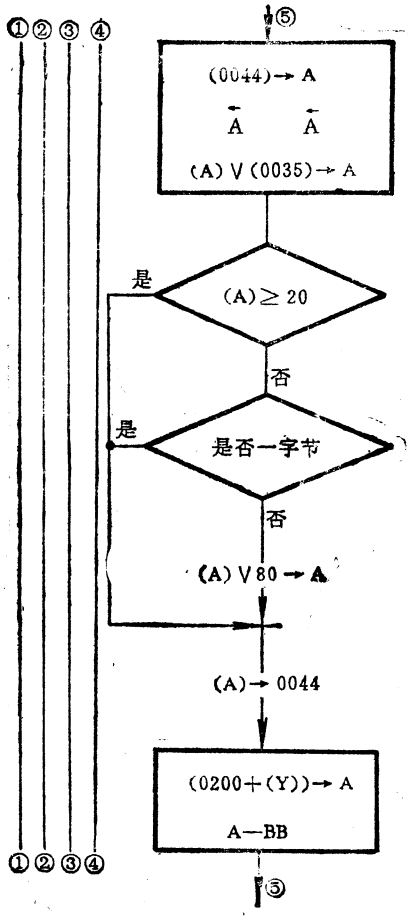
高位等于 0 时字节长度为 1。

高位不为 0 时字节长度为 2。

0035 保存字节长度："0"是 1 字节；"1"是 2 字节；"2"是 3 字节。

恢复 X 计数。

X = FF 说明六种情况已检查完(0044 中的格式化字节已形成)，否则继续循环。



(0044)去掉高2位，与字节长度进行"或"运算，结果送A。

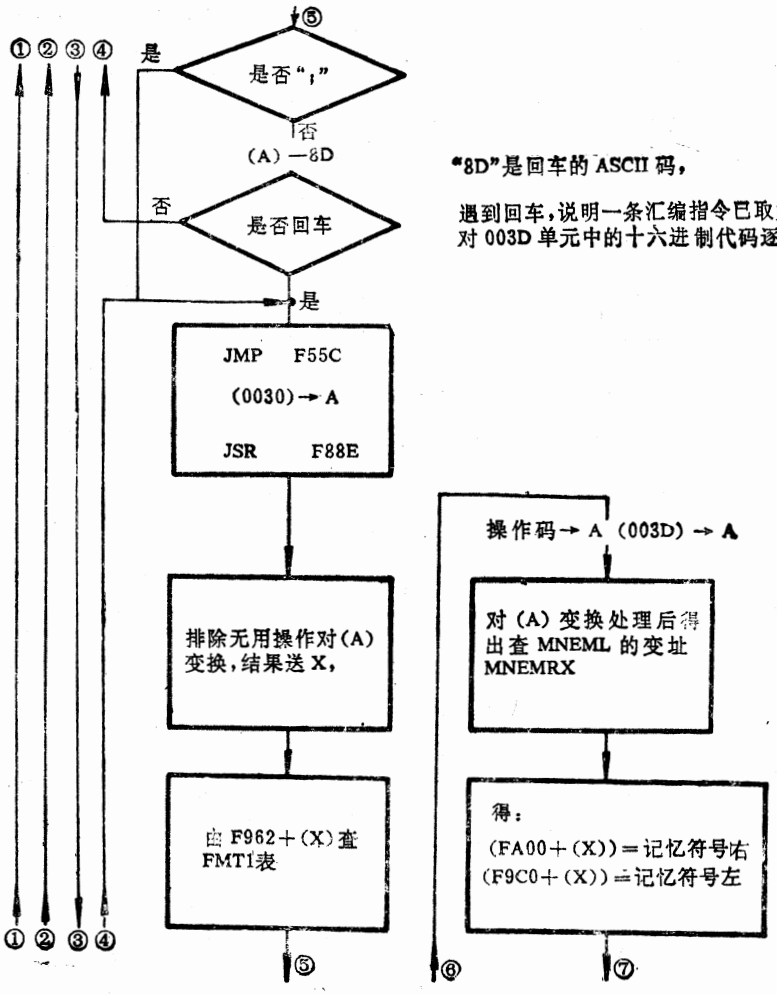
(A)≥20直接做(A)→0044。

(A)＜20，且是一字节(A)→0044，

否则(A)∨80→0044

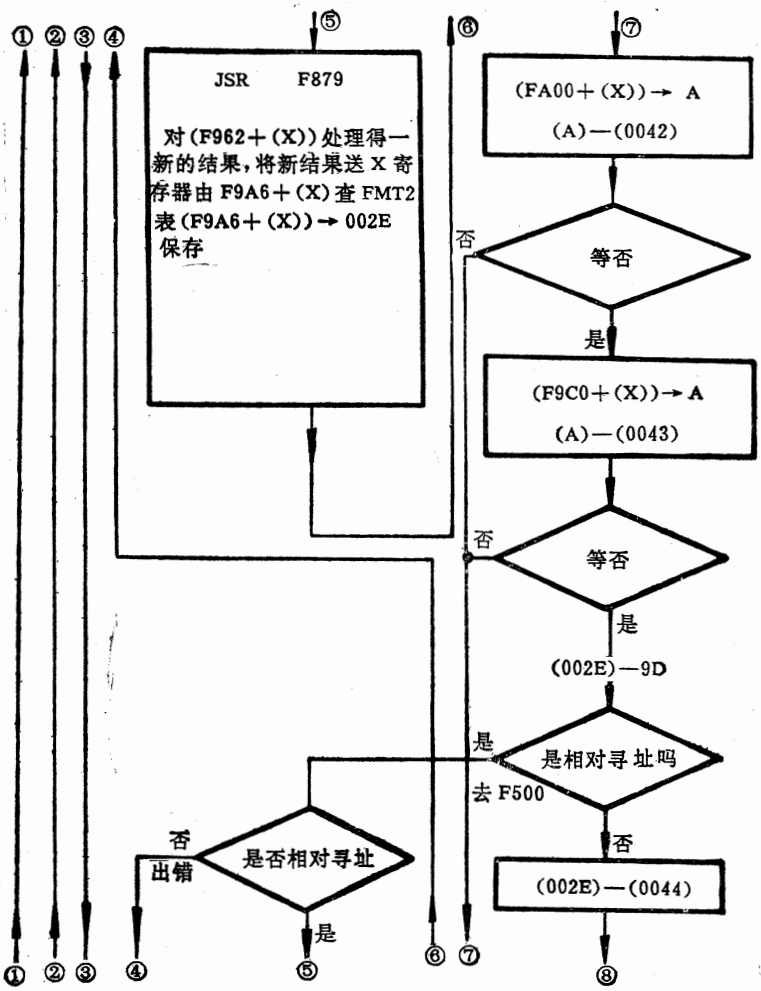
这段程序的作用是：将输入汇编指令的寻址方式形成一个十六进制代码存入0044单元中。

"BB"是";"的ASCII码，";"后可加注释。



“8D”是回车的ASCII码，

遇到回车，说明一条汇编指令已取完，应转向F88E，对003D单元中的十六进制代码逐个扫描分析。



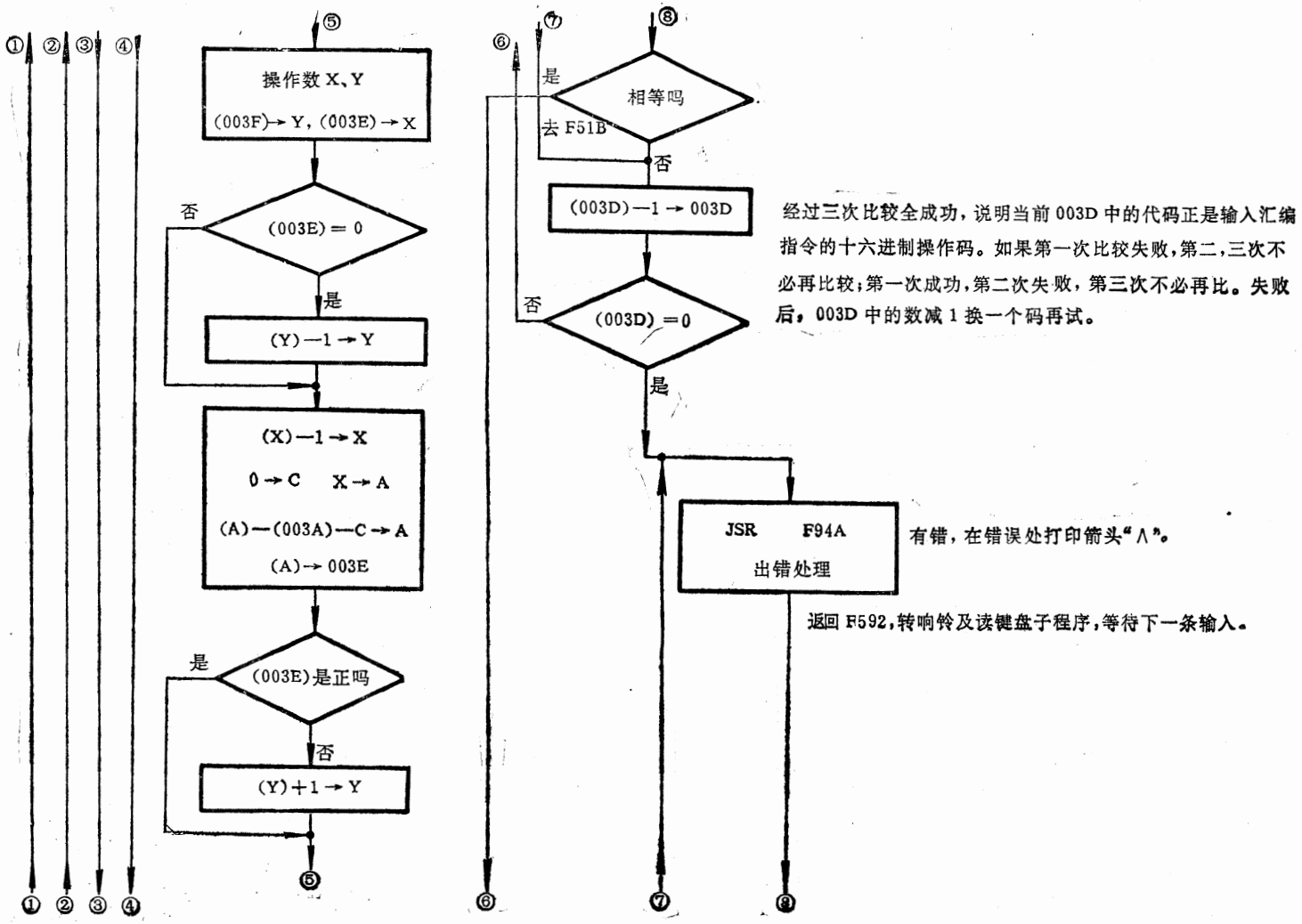
第一次比较：(FA00+(X))与(0042)比较
(FA00+(X))中的内容是键入记忆符号右8位
(0042)中的内容是查“MENMR表”所得的结果。

第二次比较：(F9C0+(X))与(0043)比较
键入记忆符号左8位与查“MENML表”的结果进行比较。

是否相对寻址格式？如果是相对寻址，转F500计算相对地址。

第三次比较(002E)与(0044)

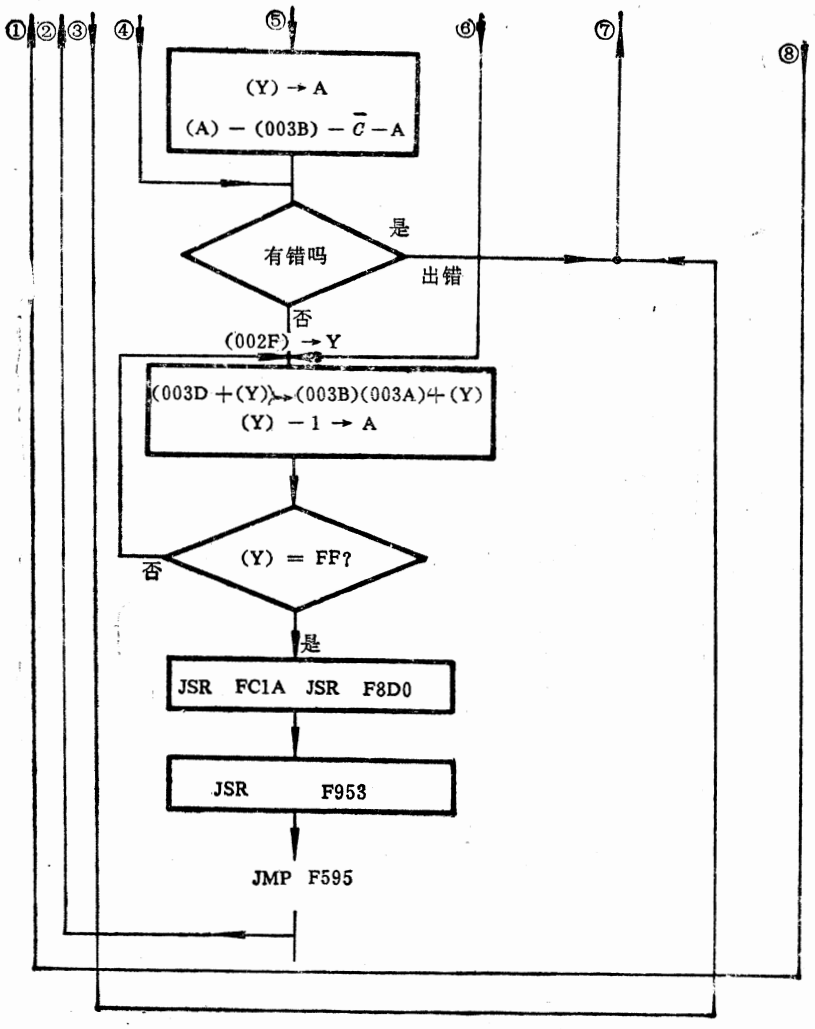
键入汇编指令的寻址方式与查“FMT2表”的结果比较。



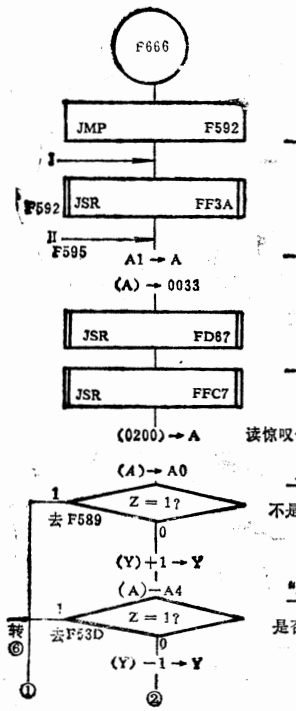
经过三次比较全成功，说明当前 003D 中的代码正是输入汇编指令的十六进制操作码。如果第一次比较失败，第二，三次不必再比较；第一次成功，第二次失败，第三次不必再比。失败后，003D 中的数减 1 换一个码再试。

有错，在错误处打印箭头“∧”。

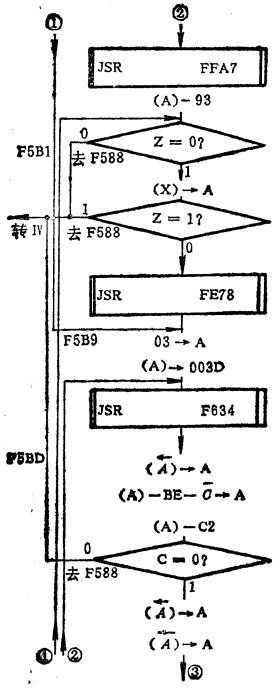
返回 F592，转响铃及读键盘子程序，等待下一条输入。


①
②
③
④
⑤
⑥
⑦
⑧
(Y) → A
(A) − (003B) − $\bar{C}$ − A
有错吗
是
出错
否
(002F) → Y
(003D + (Y)) → (003B)(003A) + (Y)
(Y) − 1 → A
(Y) = FF?
否
是
JSR FC1A JSR F8D0
JSR F953
JMP F595

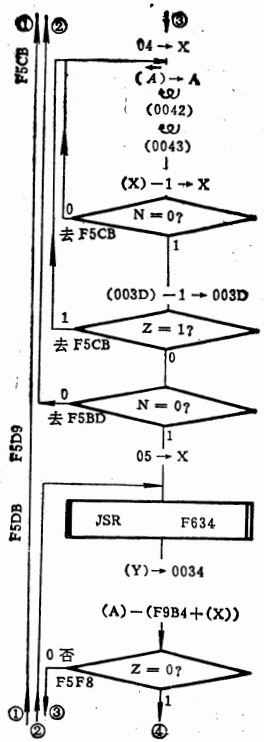
5.6 小汇编程序的举例



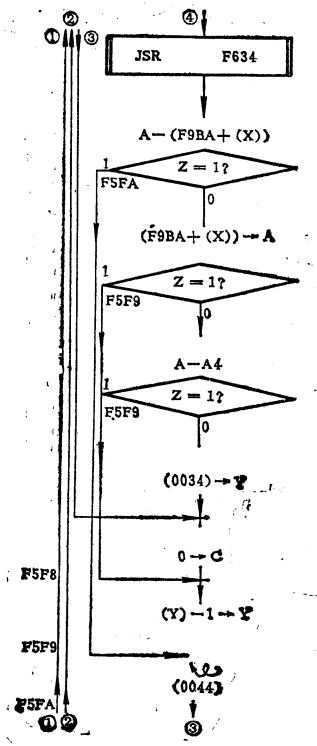
说明	举例					
	1	2	3	4	5	6
启动小汇编。	!1234: STA $3355CR	!␣CMP #$57CR	!␣ CMP ($55),YCR	!␣ASL	! ␣ BNE $1264CR	! ␣ JMP ($3355) CR
响铃。						
"!"的ASCII码送0033。 读输入汇编指令。 置方式。 读惊叹号"!"后第一个字符的ASCII码， A0是空格的ASCII码。	(A) = B1	(A)= A0	(A)= A0	(A)= A0	(A)= A0	(A)= A0
不是空格为Z = 0，是空格则Z = 1。	Z = 0 1 → Y	Z = 1	Z = 1	Z = 1	Z = 1	Z = 1
"A4"是"$"的ASCII码， 是否"$"？是则Z = 1，执行监控， 否则Z = 0，顺序执行。	Z = 0 (Y) = 0					



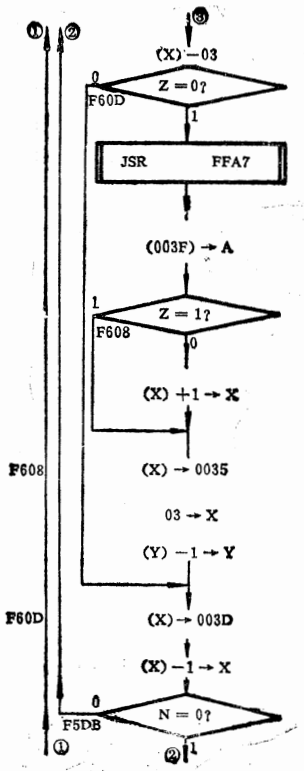
读参考地址("1"与";"之间的数)。	"12"存 003F 003D "34"存 003E 003C 0041 0040					
";"的 ASCII 码"BA"经变换得"93"，是";"为 Z＝1，不是";"则 Z＝0。从 FFA7 返回时(X)＝01，	Z＝1 01→A					
未读到地址返回时(X)＝0。出错。(003D)→003B (003C)→003A 保存	(003B)＝12 (003A)＝34					
记忆符号名字，英文字母计数。	"S""T""A"	"C""M""P"	"C""M""P"	"A""S""L"	"B""N""E"	"J""M""P"
读英文字母 ASCII 码，	(A)＝53 54 41	43 4D 50	43 4D 50	41 53 4C	42 4E 45	4A 4D 50
英文字母的 ASCII 码高位补 1，	(A)＝A6 A8 82	86 9A A0	86 9A A0	82 A6 98	84 9C 8A	94 9A A0
左移 C＝1，	(A)＝E8 EA C4	C8 DC E2	C8 DC E2	C4 E8 DA	C6 DE CC	D6 DC E2
(A)＜C2 非法，						
C＝1 合法，	C＝1	C＝1	C＝1	C＝1	C＝1	C＝1
去掉高两位不要。	A0 A8 10	20 70 88	20 70 88	10 A0 68	18 78 30	58 70 88



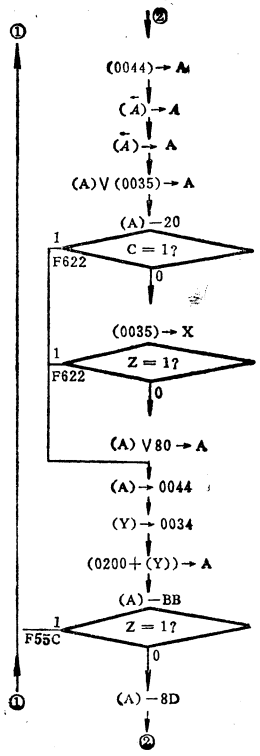
位计数每个字母循环左移5次。	"S" "T" "A" 10100 10101 00010	"C" "M" "P" 00100 01110 10001	"C" "M" "P" 00100 01110 10001	"A" "S" "L" 00010 10100 01101	"B" "N" "E" 00011 01111 00110	"J" "M" "P" 01011 01110 10001
	0043 0042	0043 0042	0043 0042	0043 0042	0043 0042	0043 0042
15位全移入0043，0042后再补一位0	A5 44	23 A2	23 A2	15 1A	1B CC	5B A2
构成十六位，左8位保存于0043中，右八位保存在0042中。						
字母计数是否为0？第三个字母多移一次，低位补0。						
寻址方式标志符计数。	绝对地址 $3355 < CR >	立即数 #$57<CR>	另页间址变址:($55),Y	累加器 ASL < CR >	相对寻址 $1264<CR>	间址JMP($3355)<CR>
	(Y) = 8	(Y) = 4	(Y) = 4	(Y) = 4	(Y) = 4	(Y) = 4
读一个字符，返回前(Y)+1→Y，Y是读缓冲区0200～02FF的变址。	(X) = 5 (Y) = 8 读"$" (X) = 4, 3 (Y) = 9 读"3" (X) = 2, 1, 0 (Y) = 13 读"CR"	(X) = 5,4,3 (Y) = 4, 读"#" (X) = 2,1,0(Y) = 8, 读"CR"	(X) = 5,4(Y) = 4,读"(" (X) = 2,1(Y) = 8 读")" (X) = 0(Y) = 9, 读","	(X) = 5, 4, 3 (Y) = 4 读"CR" (X) = 2, 1, 0 (Y) = 4 读"CR"	(X) = 5,(Y) = 4读"$" (X) = 4,3(Y) = 9读"1" (X) = 2, 1, 0 (Y) = 9 读"CR"	(X) = 5,4(Y) = 4 读"(" (X) = 2,1(Y) = 9 读")" (X) = 0,(Y) = 10读"CR" (X) = 3,(Y) = 6 读"3"
	(X) = 5, Z = 1 其余 Z = 0	(X) = 3, Z = 1 其余 Z = 0	(X) = 4, 1, 0 Z = 1 其余 Z = 0	全部 Z = 0	(X) = 5 Z = 1 其余 Z = 0	(X) = 5, 1 Z = 1 其余 Z = 0



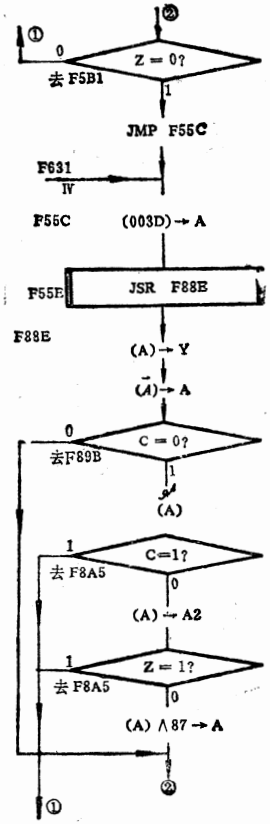
读字符，返回前(Y)+1→Y。	(X)＝5,(Y)＝9 读"3" (X)＝4，3 跳过 (X)＝2，1，0 跳过	(X)＝3(Y)＝5读"$" (X)＝5，4 跳过 (X)＝2，1，0 跳过	(X)＝4,(Y)＝5 读"$" (X)＝1,(Y)＝9 读"," (X)＝0,(Y)＝10 读"Y" (X)＝5，3，2 跳过	(X)＝5，4，3，2，1，0 跳过	(X)＝5,(Y)＝5,读"1" (X)＝4,3,2,1,0 跳过	(X)＝4(Y)＝5读"$" (X)＝1(Y)＝11读"CR" (X)＝5，3，2，0跳过
当首字符是"$""）"符合后，0→A。	(X)＝5,Z＝0 其余跳过 (X)＝5 时 0→A 其余跳过	(X)＝3,Z＝1 其余跳过 (X)＝5，4，3，2，1，0 全跳过。	(X)＝4，0　Z＝1 (X)＝1　　Z＝0 其余跳过 (X)＝1　0→A (X)＝5，4，3，2，0 跳过	全跳过 全跳过	(X)＝5,Z＝0 其余跳过 (X)＝5，0→A 其余全跳过	(X)＝4,Z＝1 (X)＝1,Z＝0 其余跳过 (X)＝1　0→A 其余全跳过
当(X)＝5，4，3，2，1，0.不是标志符号时，0→C，《0044》中循环进一个 0。	(X)＝4,3,2,1,0 0→C	(X)＝5，4,2,1,0 0→C	X＝5，3，2　0→C	全部 0→C	(X)＝4，3，2，1，0 0→C	(X)＝5，3，2，0 0→C
是标志符 C＝1， 《0044》中循环进一个 1。	(X)＝5， C＝1	(X)＝3,C＝1	(X)＝4，1，0 C＝1		X＝5,C＝1	(X)＝4，1 C＝1



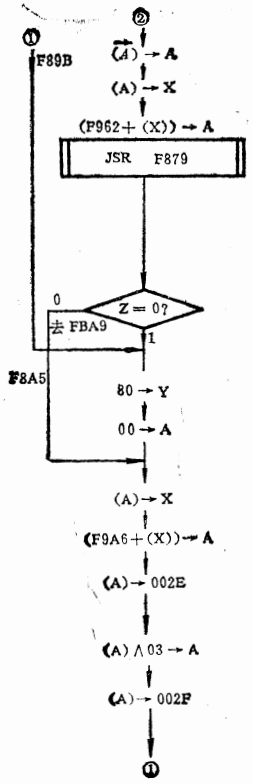
操作数前面的标志符是否读完?						
	(0044)＝××100000	(0044)＝××001000	(0044)＝××010011	(0044)＝××000000	(0044)＝××100000	(0044)＝××010010
读操作数,由FFA7返回时	33→003F	00→003F 57→003E	00→003F 55→003E		12→003F	33→003F
(X)＝01末到数(X)＝00 高位送A,高位＝0,Z＝1, 否则Z＝0。	55→003E (A)＝33	(A)＝0	(A)＝0		64→003E (A)＝12	55→003E (A)＝33
Z＝1, (X)＝01。 Z＝0, (X)＝02。						
Z＝1,(0035)＝01。 Z＝0,(0035)＝02。 重新置计数,	02→0035	01→0035	01→0035	0→0035	02→0035	02→0035
标志符不符合,退回原处。 (003D)＝0时退出,						
(X)＝FF时退出。						



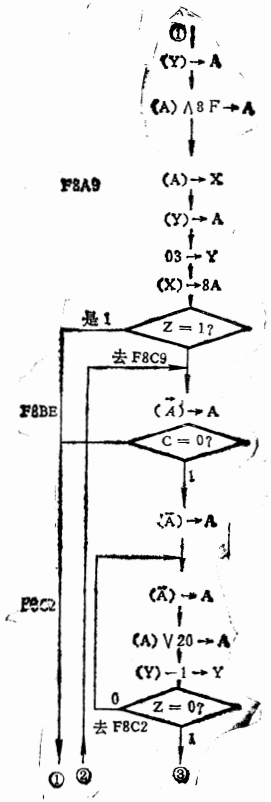
	(A)=××100000	(A)=××001000	(A)=××010011	(A)=××000000	(A)=××100000	(A)=××010010
去掉高两位。	去掉高两位 (A)=80 (A)=82	去掉高两位 (A)=20 (A)=21	去掉两位 (A)=4C (A)=4D	去掉高两位 (A)=00 (A)=00	去掉高两位 (A)=80 (A)=82	去掉高两位 (A)=44 (A)=46
	(A)>20 C=1	(A)>20 C=1	(A)>20 C=1	(A)<20 C=0	(A)>20 C=1	(A)>20 C=1
				(X)=0 Z=1 跳至 F622		
	(0044)=82	(0044)=21	(0044)=4D	(0044)=00	(0044)=82	(0044)=46
是否“,”号？标志输入汇编命令已结束。						
是否回车？标志输入汇编命令已结束。						



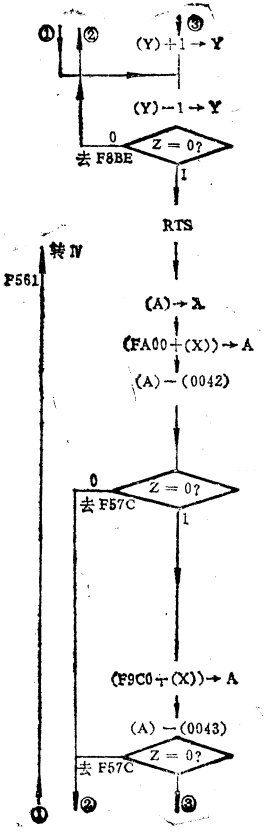
003D 单元初始值为 0，逐个扫描，寻找与输入汇编指令相符合的操作码。						
从 F88E 是扫描 003D 单元中内容的子程序。 检查最末位是否为 1，	(Y) = 8D (A) = 46	(Y) = C9 (A) = 64	(Y) = D1 (A) = 68	(Y) = 0A (A) = 05	(Y) = D0 (A) = 68	(Y) = 6C (A) = 36
最末位为 1，C = 1。 最末位为 0，C = 0。	C = 1 (A) = A8	C = 1 (A) = B2	C = 1 (A) = B4	C = 0	C = 0	C = 0
倒数 1，2 都是 1，属于无用操作。 排除 A2 无用操作。	C = 0	C = 0	C = 0			
	(A) = 83	(A) = 82	(A) = 84			



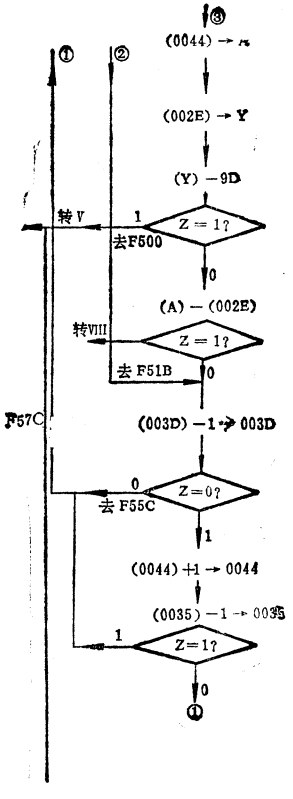
(X)是查FMT1表的变址。	(A)=41 C=1 F962+41=F9A3	(A)=41 C=0 F962+41=F9A3	(A)=42 C=0 F962+42=F9A4	(A)=02 C=0 F962+02=F964	(A)=34 C=0 F962+34=F996	(A)=1B C=01 F962+01=F963
查FMT1表得结果。	(A)=31	(A)=31	(A)=87	(A)=54	(A)=0D	(A)=3B
当C=1 $\overrightarrow{A4}$次， 将低4位去掉。						
(A)∧#0F→A	(A)=03	(A)=01	(A)=07	(A)=04	(A)=0D	(A)=0B
(X)是查FMT2的变址。	(X)=03	(X)=01	(X)=07	(X)=04	(X)=0D	(X)=0B
查FMT2表，	F9A6+03=F9A9	F9A6+01=F9A7	F9A6+07=F9AD	F9A6+04=F9AA	F9A6+0D=F9B3	F9A6+0B=F9B1
查得寻址方式码，	(002E)=82	(002E)=21	(002E)=4D	(002E)=00	(002E)=9D	(002E)=4A
送002E保存。						
	(A)=02	(A)=01	(A)=01	(A)=00	(A)=01	(A)=02
保存字节长度。	02→002F	01→002F	01→002F	00→002F	01→002F	02→002F



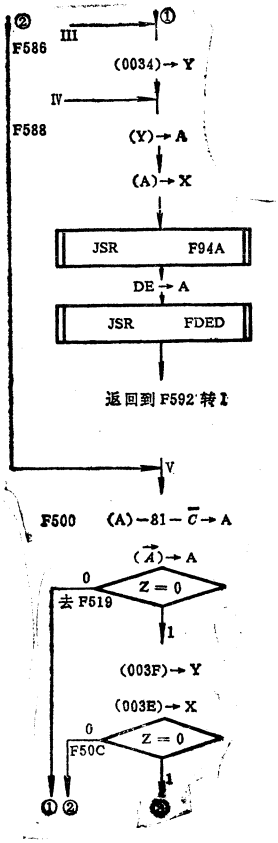
求"MNEML表""MNEMR表"的变址	8D→A	C9→A	D1→A	0A→A	D0→A	6C→A
	(A) = 8D (X) = 8D	(A) = 89 (X) = 89	(A) = 81 (X) = 81	(A) = 0A (X) = 0A	(A) = 80 (X) = 80	(A) = 0C (X) = 0C
	(A) = 8D (Y) = 03	(A) = 89 (Y) = 03	(A) = 81 (Y) = 03	(A) = 0A (Y) = 03	(A) = 80 (Y) = 03	(A) = 0C (Y) = 03



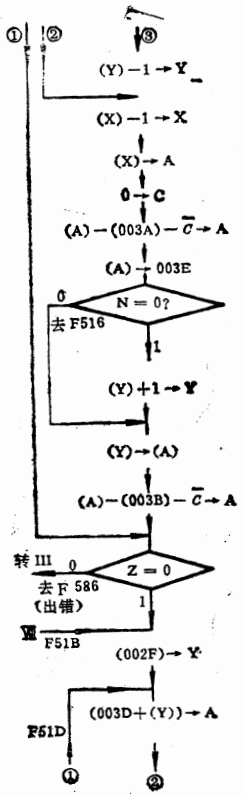
由 (X) 值查“MNEMR 表”。	(A) =3C	A = 3E	(A) = 3E	A = 30	(A) = 1A	(A) = 23
将查 MNEMR 表的结果与(0042)中内容(0042中存放有输入汇编指令记忆符号各字右 8位码)进行第一次比较。	由 FA3C 查得 44 → A	由 FA3E 查得 A2 → A	由 FA3E 查得 A2 → A	由 FA30 查得 1A → A	由 FA1A 查得 CC → A	由FA23 查得 A2 → A
Z=1，第一次比较成功。顺序执行 Z=0，第一次比较失败，换操作码。						
由 (X) 值查“MNEML 表”，将查 MNEML 表结果与(0043) 中内容 (0043 放有输入汇编记忆符号左 3 位) 进行第二次比较。	由 F9FC 查得 A5 → A	由 F9FE 查得 23 → A	由 F9FE 查得 23 → A	由 F9F0 查得 15 → A	由 F9DA 查得 1B → A	由 F9E3 查得 5 B → A



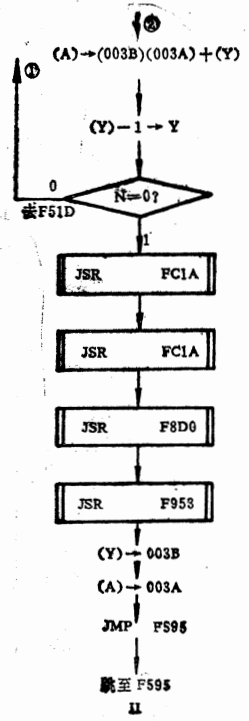
Z = 0 失败，换码。 Z=1，成功。汇编指令寻址方式码送 A 查 FMT2 得寻址方式码送 Y						
是否相对寻址？ Z = 1，转 F500 计算位置数，Z = 0，顺序执行	Z = 0	Z = 0	Z = 0	Z = 0	是相对寻址 Z=1	Z=0
第三次寻址方式码比较 Z = 1 成功转 F51B Z = 0 失败换操作码	Z=1	Z = 1	Z = 1	Z=1		Z = 1
换下一个操作码继续比较。 (003D) ≠ 0 返回						



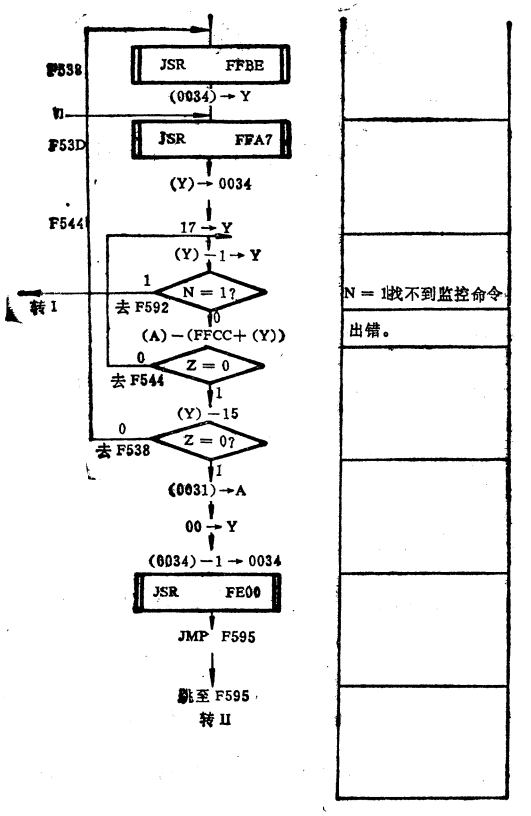
错误处理子程序，在出错处打"∧"。					
					转入F500前 C=1,(0044)→A, 82→A 82−81−0→A (A)=0
操作数高位					操作数高位 12→Y, 操作数低位 64→X
操作数低位					Z=0低位不为0



低位为0时考虑低位向高位借位，(Y)－1→Y。						
					63→A	
					63－3C－1→A　(003A)＝3C 26→003E　C＝1	
					12→A 12－12－0→A Z＝1正常	
Z＝0出错。 字节长度。	02→Y	01→Y	01→Y	00→Y	01→Y	02→Y
	(Y)＝02时(003F)→1236 (Y)＝01时(003E)→1235	(Y)＝01时(003E)→1238 (Y)＝00时(003D)→1237	(Y)－01时(003E)→123A (Y)＝00时(003D)→1239	(003D)→123B	(Y)＝01时(003E)→123D (Y)＝00时(003D)→123C	(Y)＝02时(003F)→1240 (Y)＝01时(003E)→123F



	(Y)＝00时(003D)→1234 (1236)＝33 (1235)＝55 (1234)＝8D	(1238)＝57 (1237)＝C9	(123A)＝55 (239)＝D1	(123B)＝0A	(123D)＝26 (123C)＝D0	(Y)=00时(003D)→123E (1240)＝33 (123F)=55 (123E)＝6C
(Y)＝FF退出						
游标向上滚两行						
显示一条汇编指令 重新计算指令地址						
保存新的地址高位 保存新的地址低位	12→003B 37→003A	12→003B 39→003A	12→003B 3B→003A	12→003B 3C→003A	12→003B 3E→003A	12→003B 41→003A
执行完上述六条汇编命令后内存中保存的结果→	1234……8D 1237……C9 1239……D1 123B……0A 123C……D0 123E…6C	55 33 STA $3355 57 CMP #$57 55 CMP ($55),Y ASL 26 BNE $1264 55 33 JMP ($3355)				


F538
JSR FFBE
(0034)→Y
F53D
JSR FFA7
(Y)→0034
F544
17→Y
(Y)−1→Y
1
N = 1?
转1
去F592
0
(A)−(FFCC+(Y))
0
Z = 0
去F544
1
(Y)−15
0
Z = 0?
去F538
1
(0031)→A
00→Y
(0034)−1→0034
JSR FE00
JMP F595
跳至F595
转II
N = 1找不到监控命令
出错。

附　录

附录A　Apple主机器件索引

•功能表说明

H＝高电平（稳态）

L＝低电平（稳态）

↑＝从低电平转到高电平

↓＝从高电平转到低电平

×＝不定态

Z ＝三态输出的高阻状态

Z_i＝输出端

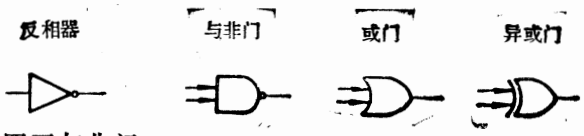
Q_0＝建立稳态输入条件前Q的电平

$\overline{Q_0}$＝建立稳态输入条件前Q的补码电平或 $\overline{Q}$ 的电平

⎍高电平脉冲

⎎低电平脉冲

•门电路符号（正逻辑）



74LS00　双输入端四正与非门

74LS02　双输入端四正或非门

74LS04　六倒相器

74LS08　双输入端四正与门

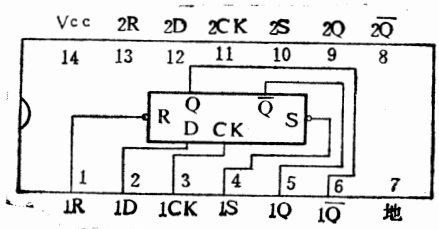
74LS11　3 输入端三正与门

74LS20　4 输入端二正与非门

74LS32　双输入端四正或门

74S86　双输入四异或门

•**74LS74**　正沿触发双D触发器

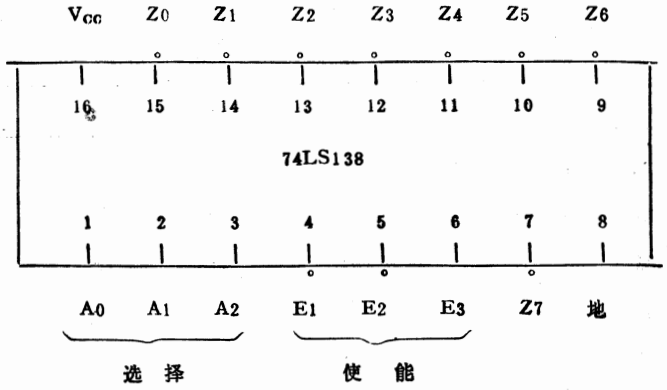


功能表

输入				输出		功能
S	R	CK	D	Q	$\overline{Q}$	
L	H	×	×	H	L	预置
H	L	×	×	L	H	清除
L	L	×	×	H*	H*	
H	H	↑	H	H	L	寄存
H	H	↑	L	L	H	寄存
H	H	L	×	Q_0	$\overline{Q_0}$	不变

*这种情况是不稳定的。即当预置和清除输入回到高电平时，状态将不能保持。

•74LS138 三线-八线译码器

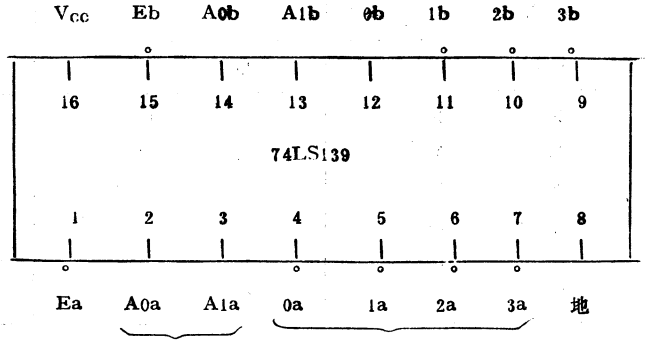


功能表

译码输入					译码输出								译码有效
使能		选择											
E3	E1(E2)	A1	A2	A0	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	
×	H	×	×	×	H	H	H	H	H	H	H	H	
L	×	×	×	×	H	H	H	H	H	H	H	H	
H	L	L	L	L	L	H	H	H	H	H	H	H	Z0
H	L	L	L	H	H	L	H	H	H	H	H	H	Z1
H	L	L	H	L	H	H	L	H	H	H	H	H	Z2
H	L	L	H	H	H	H	H	L	H	H	H	H	Z3
H	L	H	L	L	H	H	H	H	L	H	H	H	Z4
H	L	H	L	H	H	H	H	H	H	L	H	H	Z5
H	L	H	H	L	H	H	H	H	H	H	L	H	Z6
H	L	H	H	H	H	H	H	H	H	H	H	L	Z7

•74LS139

•74LS139双二线-四线译码器

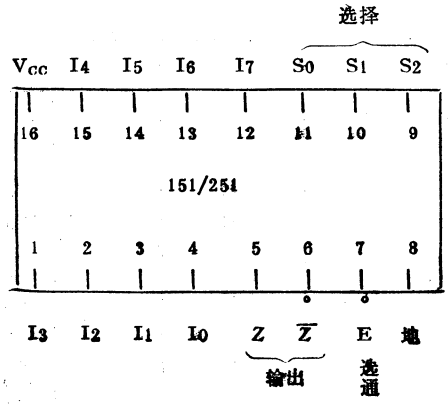


功能表

输入			输出				译码有效
使能	选择						
Ea	A1a	A0b	0a	1a	2a	3a	
H	×	×	H	H	H	H	
L	L	L	L	H	H	H	0a
L	L	H	H	L	H	H	1a
L	H	L	H	H	L	H	2a
L	H	H	H	H	H	L	3a

b组译码与a组相同

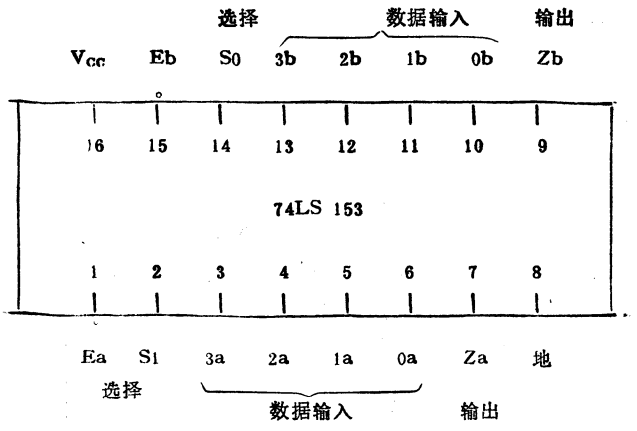
•74LS151/251 八选一数据选择器（多路开关）251是三态输出



功能表

输入				输出	
选择			选通		
S2	S1	S0	E	Z	$\overline{Z}$
×	×	×	H	L	H
L	L	L	L	I0	$\overline{I0}$
L	L	H	L	I1	$\overline{I1}$
L	H	L	L	I2	$\overline{I2}$
L	H	H	L	I3	$\overline{I3}$
H	L	L	L	I4	$\overline{I4}$
H	L	H	L	I5	$\overline{I5}$
H	H	L	L	I6	$\overline{I6}$
H	H	H	L	I7	$\overline{I7}$

•74LS153　双4选一数据选择器（多路开关）

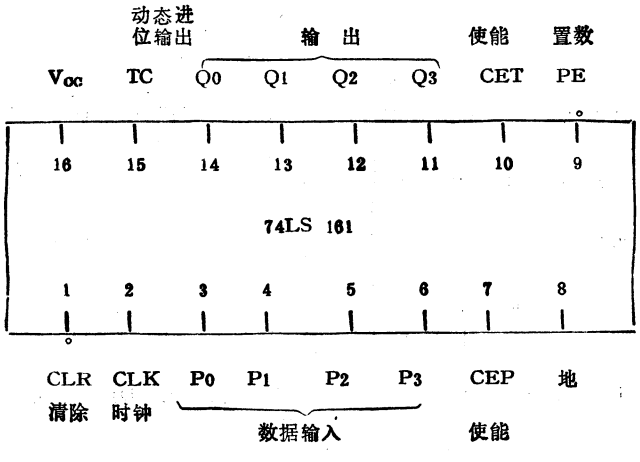


功能表

选择输入		数　据　输　入				选　通	输　出
S1	S0	0Y	1Y	2Y	3Y	E	ZY
×	×	×	×	×	×	H	L
L	L	L	×	×	×	L	L
L	L	H	×	×	×	L	H
L	H	×	L	×	×	L	L
L	H	×	H	×	×	L	H
H	L	×	×	L	×	L	L
H	L	×	×	H	×	L	H
H	H	×	×	×	L	L	L
H	H	×	×	×	H	L	H

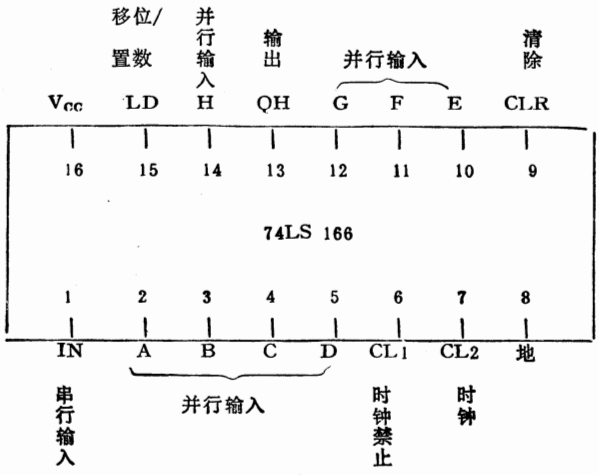
选择输入S0和S1是两部分公用.
数据输入表示a组或b组

•74LS161　同步四位计数器



这种计数器是全编程的，即输出可预置到任何值。当预置是同步时，在置数输入上将建立一低电平，封住计数器，不管使能端为何电平，在下一个时钟来之前，使输出与建立数据一致。

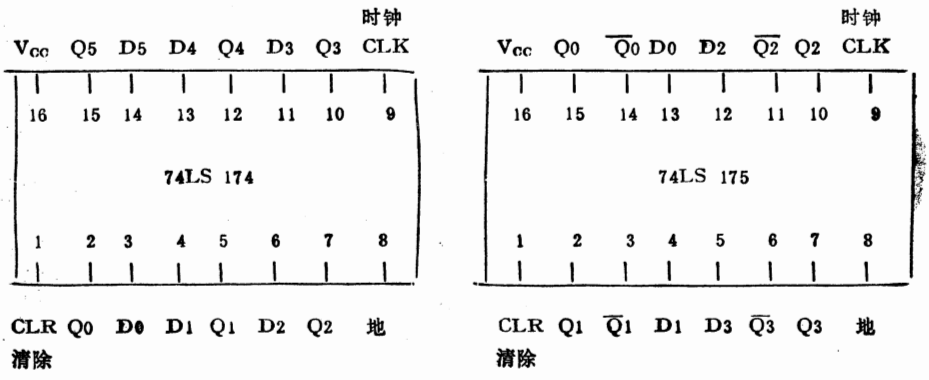
•74LS166　八位移位寄存器



功能表

输入						输出			功能
清除	移位/置数	时钟禁止	时钟	串行	并行 A…H	QA	QB	QH	
L	×	×	×	×	×	L	L	L	清除
H	×	L	L	×	×	QA_0	QB_0	QH_0	维持
H	L	L	↑	×	a…h	a	b	h	置数
H	H	L	↑	H	×	H	QA_n	QG_n	移位 补高
H	H	L	↑	L	×	L	QA_n	QG_n	移位 补低
H	×	H	↑	×	×	QA_0	QB_0	QH_0	禁止

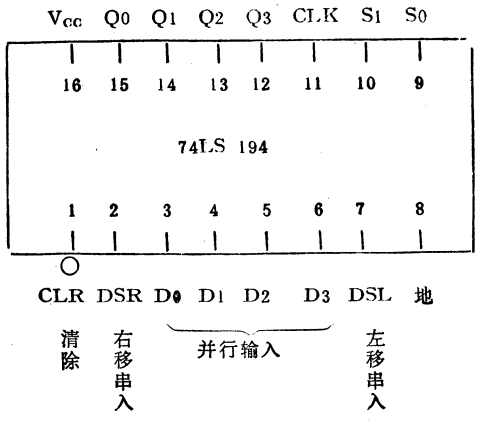
•74LS174/175　带清除端六/四D触发器



功能表

输入			输出	
CLR	CLK	D	Q	$\overline{Q}$
L	×	×	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	×	Q_0	$\overline{Q_0}$

•74LS194 四位双向通用移位寄存器

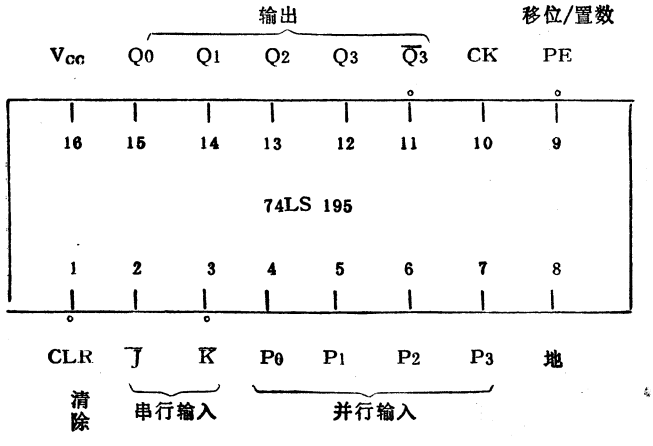


功能表

输入										输出			
CLR	模式		CLK	串行		并行				Q0	Q1	Q2	Q3
	S1	S0		左	右	D0	D1	D2	D3				
L	×	×	×	×	×	×	×	×	×	L	L	L	L
H	×	×	L	×	×	×	×	×	×	QD0	QD1	QD2	QD3
H	H	H	↑	×	×	d0	d1	d2	d3	d0	d1	d2	d3
H	L	H	↑	×	H	×	×	×	×	H	$Q0_n$	$Q1_n$	$Q2_n$
H	L	H	↑	×	L	×	×	×	×	L	$Q0_n$	$Q1_n$	$Q2_n$
H	H	L	↑	H	×	×	×	×	×	$Q1_n$	$Q2_n$	$Q3_n$	H
H	H	L	↑	L	×	×	×	×	×	$Q1_n$	$Q2_n$	$Q3_n$	L
H	L	L	×	×	×	×	×	×	×	$Q0_n$	$Q1_n$	$Q2_n$	$Q3_n$

d0、d1、d2、d3＝输入D0、D1、D2、D3端相应的稳定态输入电平

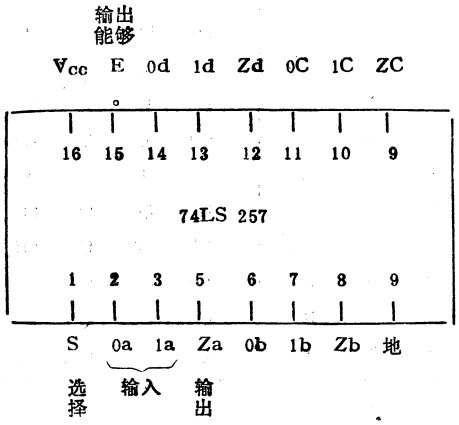
•74LS 195 四位并行存取移位寄存器



功能表

输入									输出				
CLR	PE	CK	串行		并行				Q0	Q1	Q2	Q3	$\overline{Q3}$
			J	$\overline{K}$	P0	P1	P2	P3					
L	×	×	×	×	×	×	×	×	L	L	L	L	H
H	L	↑	×	×	p0	p1	p2	p3	p0	p1	p2	p3	$\overline{p3}$
H	H	L	×	×	×	×	×	×	$Q0_0$	$Q1_0$	$Q2_0$	$Q3_0$	$\overline{Q3_0}$
H	H	↑	L	H	×	×	×	×	$Q0_0$	$Q0_0$	$Q1_n$	$Q2_n$	$\overline{Q2_n}$
H	H	↑	L	L	×	×	×	×	L	$Q0_n$	$Q1_n$	$Q2_n$	$\overline{Q2_n}$
H	H	↑	H	H	×	×	×	×	H	$Q0_n$	$Q1_n$	$Q2_n$	$\overline{Q2_n}$
H	H	↑	H	L	×	×	×	×	$\overline{Q0_n}$	$Q0_n$	$Q1_n$	$Q2_n$	$\overline{Q2_n}$

•74LS257　四2选1数据选择器

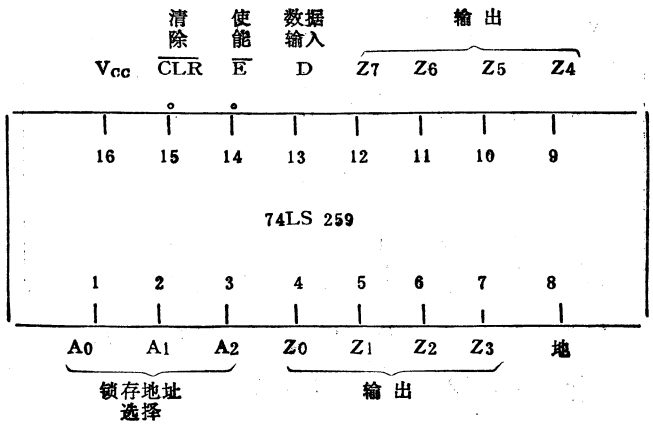


功能表

输入				输出
E	S	a	b	Zi
H	×	×	×	Z
L	L	L	×	L
L	L	H	×	H
L	H	×	L	L
L	H	×	H	H

Zi中i=a, b, c, d

•74LS259　8位可寻址锁存器



功能表

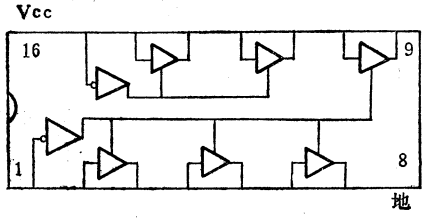
输入		寻址锁存器输出	其他输出	功能
$\overline{CLR}$	$\overline{E}$			
H	L	D	Zi0	寻址锁存
H	H	Zi0	Zi0	存储器
L	L	D	L	8线解复器
L	H	L	L	清除

锁存选择表

选择输入			编址
A2	A1	A0	锁存器
L	L	L	Z0
L	L	H	Z1
L	H	L	Z2
L	H	H	Z3
H	L	L	Z4
H	L	H	Z5
H	H	L	Z6
H	H	H	Z7

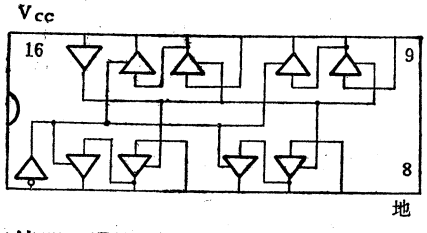
•8T97（SN74367）原码三态总线驱动器

8T97

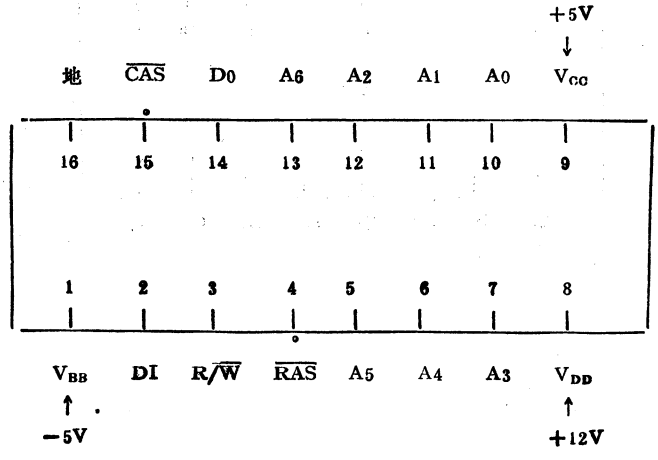


•8T28 原码三态输出驱动器

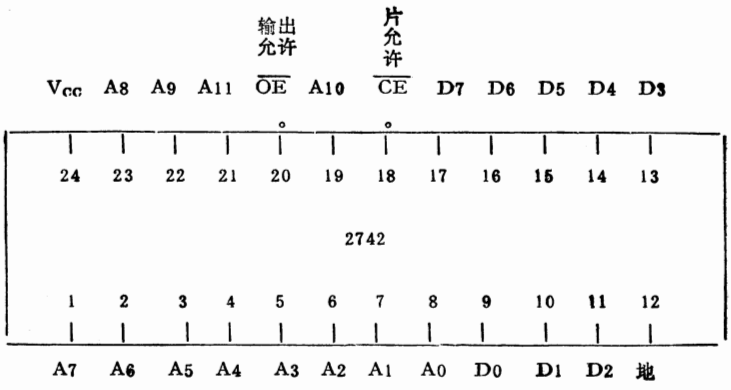
8T28



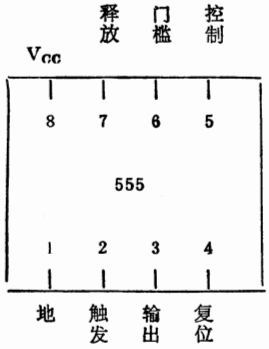
•4116 16K×1位随机存储器（RAM）



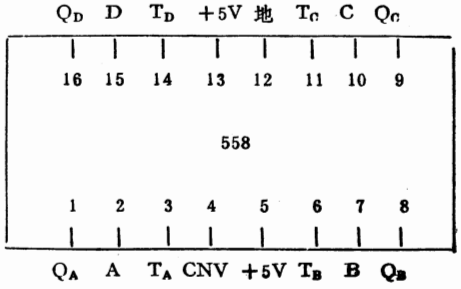
•2732　4K×8位只读存储器（ROM）



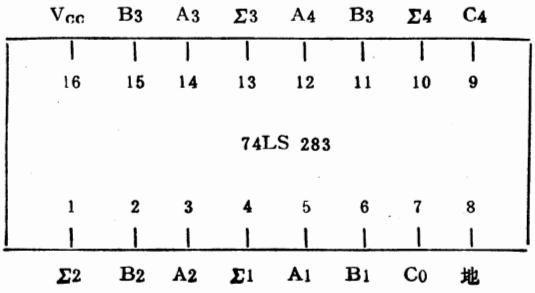
•555 精确计时器



•558四计时器



•74LS283　快速进位四位二进制全加器



功能表

输入				输出					
				当C0=L/当C2=L			当C0=H/C2=H		
A1/A3	B1/B3	A2/A4	B2/B4	Σ1/Σ3	Σ2/Σ4	C2/C4	Σ1/Σ3	Σ2/Σ4	C2/C4
L							H		
H				H				H	
L	H			H				H	
H	H				H		H	H	
L		H			H		H	H	
H		H		H	H				H
L	H	H		H	H				H
H	H	H				H	H		H
L			H		H		H	H	
H			H	H	H				H
L	H		H	H	H				H
H	H		H			H	H		H
L		H	H			H	H		H
H		H	H	H		H		H	H
L	H	H	H	H		H		H	H
H	H	H	H		H	H	H	H	H

注：表中未注电平处均为低电平L。
先用A1，B1，A2，B2和C0的输入条件确定Σ1、Σ2的输出和内部进位C2的值。然后用C2、A3、B3、A4和B4的值来确定Σ3和Σ4和C4的输出。

附录 B 6502指令

THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

A	Accumulator
X,Y	Index Registers
M	Memory
$\overline{C}$	Borrow
P	Processor Status Register
S	Stack Pointer
✓	Change
—	No Change
+	Add
∧	Logical AND
−	Subtract
⊕	Logical Exclusive Or

FIGURE 1. ASL-SHIFT LEFT ONE BIT OPERATION

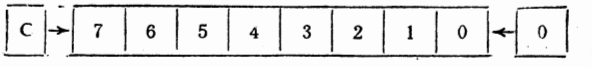
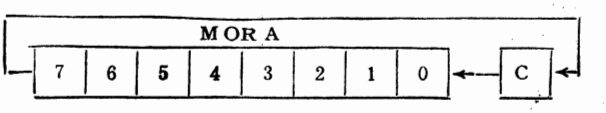
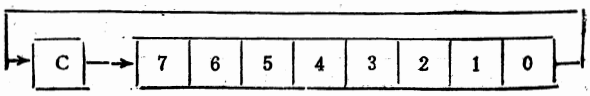


FIGURE 2. ROTATE ONE BIT LEFT(MEMORY OR ACCUMULATOR)



↑	Transfer From Stack
↓	Transfer To Stack
→	Transfer To
←	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	Operand
#	Immediate Addressing Mode

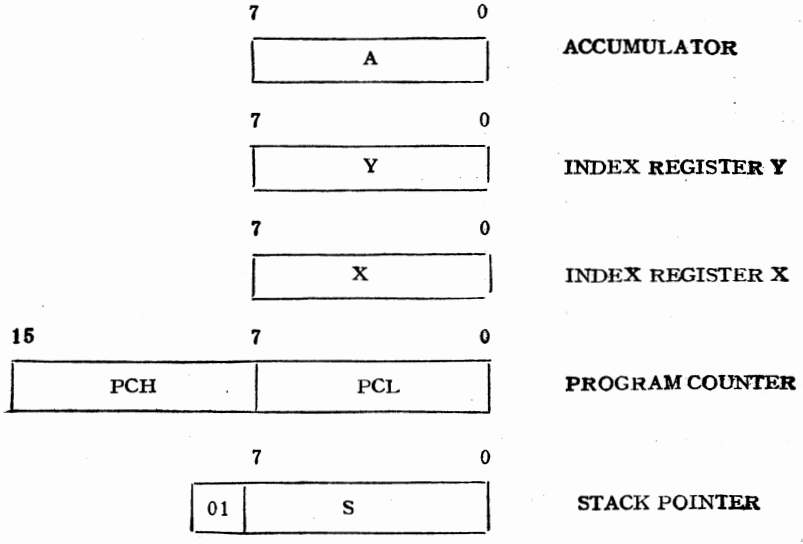
FIGURE 3.

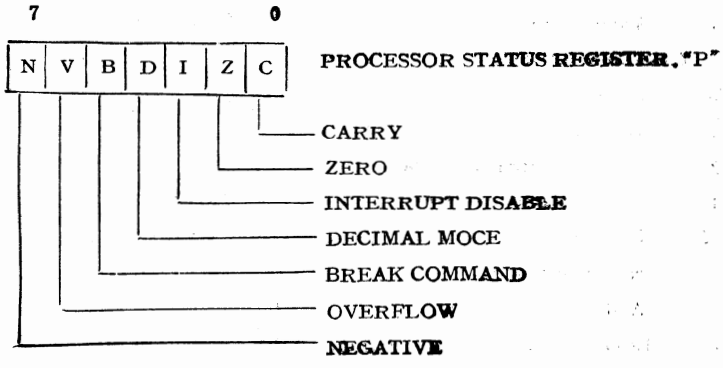


NOTE 1, BIT — TEST BITS

Bit 6 and 7 are transferred to the status register. If the result of A ∧ M is zero then Z = 1, otherwise Z = 0.

PROGRAMMING MODEL





INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
ADC						
Add memory to accumulator with carry	A+M+C→A,C	Immediate	ADC #Oper	69	2	✓✓✓——✓
		Zero Page	ADC Oper	65	2	
		Zero Page,x	ADC Oper ,x	75	2	
		Absolute	ADC Oper	6D	3	
		Absolute,X	ADC Oper ,X	7D	3	
		Absolute,Y	ADC Oper ,Y	79	3	
		(indirect,X)	ADC (Oper,X)	61	2	
		(indirect),Y	ADC (Oper),Y	71	2	
AND						
"AND" memory with accumulator	A∧M→A	Immediate	AND #Oper	29	2	✓✓————
		Zero Page	AND Oper	25	2	
		Zero Page,X	AND Oper ,X	35	2	
		Absolute	AND Oper	2D	3	
		Absolute,X	AND Oper ,X	3D	3	
		Absolute,Y	AND Oper ,Y	39	3	
		(Indirect,X)	AND (Oper,X)	21	2	
		(Indirect),Y	AND (Oper),Y	31	2	
ASL						
Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator	ASL A	0A	1	✓✓✓———
		Zero Page	ASL Oper	06	2	
		Zero Page,X	ASL Oper,X	16	2	
		Absolute	ASL Oper	0E	3	
		Absolute,X	ASL Oper,X	1E	3	
BCC						
Branch on carry clear	Branch on C=0	Relative	BCC Oper	90	2	——————

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
BCS Branch on carry set	Branch on C=1	Relative	BCS Oper	B0	2	———————
BEQ Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	———————
BIT Test bits in memory With accumulator	A∧M, M_7→N, M_6→V	Zero Page Absolute	BIT* Oper BIT* Oper	24 2C	2 3	M_7✓———M_6
BMI Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	———————
BNE Branch on result not zero	Branch on Z=0	Relative	BNE Oper	D0	2	———————
BPL Branch on result plus	Branch on N=0	Relative	BPL Oper	10	2	———————
BRK Force Break	Forced Interrupt PC+2↓P↓	Implied	BRK*	00	1	———1——
BVC Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	———————
BVS Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	———————
CLC Clear carry flag	0→C	Implied	CLC	18	1	———0——
CLD Clear decimal mode	0→D	Implied	CLD	D8	1	—0————

Nate 1 Bit 6 and 7 are transterred to the status register. If the result of A∧M is o then Z=1 otherwise Z=0

Note 2 A BRK command cannot be masked by setting 1.

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
CLI	0→I	Implied	CLI	58	1	— — — 0 — —
CLV Clear overflow flag	0→V	Implied	CLV	B8	1	0 — — — — —
CMP Compare memory and accumulator	A−M	Immediate	CMP #Oper	C9	2	✓✓✓— — —
		Zero page	CMP Oper	C5	2	
		Zero page.X	CMP Oper.X	D5	2	
		Absolute	CMP Oper	CD	3	
		Absolute.X	CMP Oper.X	DD	3	
		Absolute.Y	CMP OPer.Y	D9	3	
		(Indirect.X)	CMP (Oper.X)	C1	2	
		(Indirect).Y	CMP (Oper).Y	D1	2	
CPX Compare memory and ndex X	X−M	Immediate	CPX #Oper	E0	2	✓✓✓— — —
		Zero page	CPX Oper	E4	2	
		Absolute	CPX Oper	EC	3	
CPY Compare memory and ndex Y	Y−M	Immediate	CPY #Oper	C0	2	✓✓✓— — —
		Zero Page	CPY Oper	C4	2	
		Absolute	CPY Oper	CC	3	
DEC Decrement memory by one	M−1→M	Zero Page	DEC Oper	C6	2	✓✓— — — —
		Zero Page.X	DEC Oper.X	D6	2	
		Absolute	DEC Oper	CE	3	
		Absolute.X	DEC Oper.X	DE	3	
DEX Decrement index X by one	X−1→X	Implied	DEX	CA	1	✓✓— — — —
DEY Decrement index Y by one	Y−1→Y	Implied	DEY	88	1	✓✓— — — —

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P"Status Reg. N Z C I D V
EOR "Exclusive-Or"memory with accumulator	A V M→A	Immediate	EOR # Oper	49	2	✓✓————
		Zero Page	EOR Oper	45	2	
		Zero Page.X	EOR Oper .X	55	2	
		Absolute	EOR Oper	4D	3	
		Absolute.X	EOR Oper .X	5D	3	
		Absolute.Y	EOR Oner .Y	59	3	
		(Indirect.X)	EOR (Oper.X)	41	2	
		(Indirect).Y	EOR (Oper).Y	51	2	
INC Increment memory by one	M+1→M	Zero Page	INC Oper	E6	2	✓✓————
		Zero Page.X	INC Oper.X	F6	2	
		Absolute	INC Oper	EE	3	
		Absolute X	INC Oper.X	FE	3	
INX Increment index X by one	X+1→X	Implied	INX	E8	1	✓✓————
INY Increment index Y by one	Y+1→Y	Implied	INY	C8	1	✓✓————
JMP Jump to new location	(PC+1) →PCL	Absolute	JMP Oper	4C	3	——————
	(PC+2) →PCH	Indirect	JMP (Oper)	6C	3	
JSR Jump to new location saving return address	PC+2↓ (PC+1) →PCL (PC+2) →PCH	Absolute	JSR Oper	20	3	——————

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
LDA						
Load accumulator with memory	M→A	Immediate	LDA #Oper	A9	2	✓✓−−−−
		Zero Page	LDA Oper	A5	2	
		Zero Page,X	LDA Oper,X	B5	2	
		Absolute	LDA Oper	AD	3	
		Absolute,X	LDA Oper,X	BD	3	
		Absolute,Y	LDA Oper,Y	B9	3	
		(Indirect,X)	LDA (Oper,X)	A1	2	
		(Indirect),Y	LDA (Oper),Y	B1	2	
LDX						
Load index X with memory	M→X	Immediate	LDX #Oper	A2	2	✓✓−−−−
		Zero Page	LDX Oper	A6	2	
		Zero Page,Y	LDX Oper,Y	B6	2	
		Absolute	LDX Oper	AE	3	
		Absolute,Y	LDX Oper,Y	BE	3	
LDY						
Load index Y with memory	M→Y	Immediate	LDY #Oper	A0	2	✓✓−−−−
		Zero Page	LDY Oper	A4	2	
		Zero Page,X	LDY Oper,X	B4	2	
		Absolute	LDY Oper	AC	3	
		Absolute,X	LDY Oper,X	BC	3	
LSR						
Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator	LSR A	4A	1	0✓✓−−−
		Zero Page	LSR Oper	46	2	
		Zero Page,X	LSR Oper,X	56	2	
		Absolute	LSR Oper	4E	3	
		Absolute,X	LSR Oper,X	5E	3	

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
NOP No operation	No Operation	Implied	NOP	EA	1	– – – – – –
ORA "OR" memory with accumulator	A V M→A	Immediate	ORA #Oper	09	2	✓✓– – – –
		Zero Page	ORA Oper	05	2	
		Zero Page.X	ORA Oper.X	15	2	
		Absolute	ORA Oper	0D	3	
		Absolute.X	ORA Oper.X	1D	3	
		Absolute.Y	ORA Oper.Y	19	3	
		(Indirect.X)	ORA (Oper.X)	01	2	
		(Indirect).Y	ORA (Oper).Y	11	2	
PHA Push accumulator on stack	A ↓	Implied	PHA	48	1	– – – – – –
PHP Push processor status on stack	P ↓	Implied	PHP	08	1	– – – – – –
PLA Pull accumulator from stack	A ↑	Implied	PLA	68	1	✓✓– – – –
PLP Pull processor status from stack	P ↑	Implied	PLP	28	1	From Stack
ROL Rotate one bit left (memory or accumulator	(See Figure 2)	Accumulator	ROL A	2A	1	✓✓✓– – –
		Zero Page	ROL Oper	26	2	
		Zero Page.X	ROL Oper.X	36	2	
		Absolute	ROL Oper	2E	3	
		Absolute.X	ROL Oper.X	3E	3	

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
ROR Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator	ROR A	6A	1	✓✓✓— — —
		Zero Page	ROR Oper	66	2	
		Zero Page.X	ROR Oper .X	76	2	
		Absolute	ROR Oper	6E	3	
		Absolute.X	ROR Oper .X	7E	3	
RTI Return from interrupt	P↑ PC↑	Implied	RTI	40	1	From Stack
RTS Return from subroutine	PC↑ .PC+1→ PC	Implied	RTS	60	1	— — — — — —
SBC Subtract memory from accumulator with borrow	A−M−$\overline{C}$→A	Immediate	SBC #Oper	E9	2	✓✓✓— — —
		Zero Page	SBC Oper	E5	2	
		Zero Page.X	SBC Oper .X	F5	2	
		Absolute	SBC Oper	ED	3	
		Absolute.X	SBC Oper .X	FD	3	
		Absolute.Y	SBC Oper .Y	F9	3	
		(Indirect .X)	SBC (Oper.X)	E1	2	
		(Indirect) .Y	SBC (Oper).Y	F1	2	
SEC Set carry flag	1→C	Implied	SEC	38	1	— — 1 — — —
SED Set decimal mode	1→D	Implied	SED	F8	1	— — — — 1 —
SEI Set interrupt disable status	1→I	Implied	SEI	78	1	— — — 1 — —

续表

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg. N Z C I D V
STA Store accumulator in memory	A→M	Zero Page	STA Oper	85	2	------
		Zero page, X	STA Oper, X	95	2	
		Absolute	STA Oper	8D	3	
		Absolute, X	STA Oper, X	9D	3	
		Absolute, Y	STA Oper, Y	99	3	
		(Indirect, X)	STA (Oper, X)	81	2	
		(indirect), Y	STA (Oper), Y	91	2	
STX Store index X in memory	X→M	Zero Page	STX Oper	86	2	------
		Zero Page, Y	STX Oper, Y	96	2	
		Absolute	STX Oper	8E	3	
STY Store index Y in memory	Y→M	Zero Page	STY Oper	84	2	------
		Zero Page, X	STY Oper, X	94	2	
		Absolute	STY Oper	8C	3	
TAX Transfer accumulator to index X	A→X	Implied	TAX	AA	1	✓✓----
TAY Transfer accumulator to index Y	A→Y	Implied	TAY	A8	1	✓✓----
TSX Transfer sfack pointer to index X	S→X	Implied	TSX	BA	1	✓✓----
TXA Transfer index X to accumulator	X→A	Implied	TXA	8A	1	✓✓----
TXS Transfer index X to stack pointer	X→S	Implied	TXS	9A	1	------
TYA Transfer index Y to accumulator	Y→A	Implied	TYA	98	1	✓✓----

HEX OPERATION CODES

00-BRK
01-ORA-(indirect, X)
02-NOP
03-NOP
04-NOP
05-ORA-Zero Page
06-ASL-Zero Page
07-NOP
08-PHP
09-ORA-Immediate
0A-ASL-Accumulator
0B-NOP
0C-NOP
0D-ORA-Absolute
0E-ASL-Absolute
0F-NOP
10-BPL
11-ORA-(Indirect),Y
12-NOP
13-NOP
14-NOP
15-ORA-Zero Page,X
16-ASL-Zero Page,X
17-NOP
18-CLC
19-ORA-Absolute,Y
1A-NOP
1B-NOP
1C-NOP
1D-ORA-Absolute,X
1E-ASL-Absolute,X
1F-NOP
20-JSR
21-AND-(Indirect, X)
22-NOP
23-NOP
24-BIT-Zero Page
25-AND-Zero Page
26-ROL-Zero Page
2F-NOP
30-BMI
31-AND-(indirect),Y
32-NOP
33-NOP
34-NOP
35-AND-Zero Page,X
36-ROL-Zero Page,X
37-NOP
38-SEC
39-AND-Absolute,Y
3A-NOP
3B-NOP
3C-NOP
3D-AND-Absolute,X
3E-ROL-Absolute,X
3F-NOP
40-RTI
41-EOR-(Indirect, X)
42-NOP
43-NOP
44-NOP
45-EOR-Zero Page
46-LSR-Zero Page
47-NOP
48-PHA
49-EOR-Immediate
4A-LSR-Accumulator
4B-NOP
4C-JMP-Absolute
4D-EOR-Absolute
4E-LSR-Absolute
4F-NOP
50-BVC
51-EOR(Indirect),Y
52-NOP
53-NOP
54-NOP
55-EOR-Zero Page,X
5E-LSR-Absolute,X
5F-NOP
60-RTS
61-ADC-(indirect, X)
62-NOP
63-NOP
64-NOP
65-ADC-Zero Page
66-ROR-Zero Page
67-NOP
68-PLA
69-ADC-Immediate
6A-ROR-Accumulator
6B-NOP
6C-JMP-Indirect
6D-ADC-Absolute
6E-ROR-Absolute
6F-NOP
70-BVS
71-ADC-(Indirect),Y
72-NOP
73-NOP
74-NOP
75-ADC-Zero Page,X
76-ROR-Zero Page,X
77-NOP
78-SEI
79-ADC-Absolute,Y
7A-NOP
7B-NOP
7C-NOP
7D-ADC-Absolute, X NOP
7E-ROR-Absolute, X NOP
7F-NOP
80-NOP
81-STA-(Indirect, X)
82-NOP
83-NOP
84-STY-Zeropage

27-NOP
28-PLP
29-AND-Immediate
2A-ROL-Accumulator
2B-NOP
2C-BIT-Absolute
2D-AND-Abtolute
2E-ROL-Absolute
8D-STA-Absolute
8E-STX-Absolute
8F-NOP
90-BCC
91-STA-(Indirect),Y
92-NOP
93-NOP
94-STY-Zero page,X
95-STA-Zero page,X
96-STX-Zero page,Y
97-NOP
98-TYA
99-STA-Absolute,Y
9A-TXS
9B-NOP
9C-NOP
9D-STA-Absolute,X
9E-NOP
9F-NOP
A0-LDY-Immebiate
A1-LDA-(Indirect,X)
A2-LDX-Immediate
A3-NOP
A4-LDY-Zero Page
A5-LDA-Zero Page
A6-LDX-Zero Page
A7-NOP
A8-TAY
A9-LDA-Immediate
AA-TAX
AB-NOP
AC-LDY-Absolute
AD-Absolute LDA
56-LSR-Zero Page,X
57-NOP
58-CLI
59-EOR-Absolute,Y
5A-NOP
5B-NOP
5C-NOP
5D-EOR-Absolute,X
B4-LDY-Zero Page,X
B5-LDA-Zero Page,X
B6-LDX-Zero Page,Y
B7-NOP
B8-CLV
B9-LDA-Absolute,Y
BA-TSX
BB-NOP
BC-LDY-Absolute, X
BD-LDA-Absolute,X
BE-LDX-Absolute,Y
BF-NOP
C0-CPY-Immediate
C1-CMP-(Indirect,X)
C2-NOP
C3-NOP
C4-CPY-Zero Page
C5-CMP-Zero Page
C6-DEC-Zero Page
C7-NOP
C8-INY
C9-CMP-Immediate
CA-DEX
CB-NOP
CC-CPY-Absolute
CD-CMP-Absolute
CE-DEC-Absolute
CF-NOP
D0-BNE
D1-CMP-(Indirect), Y
D2-NOP
D3-NOP
D4-NOP
85-STA-Zero Page
86-STX-Zero Page
87-NOP
88-DEY
89-NOP
8A-TXA
8B-NOP
8C-STY-Absolute
DB-NOP
DC-NOP
DD-CMP-Absolute,X
DE-DEC-Absolute,X
DF-NOP
E0-CPX-Immediate
E1-SBC-(Indirect,X)
E2-NOP
E3-NOP
E4-CPX-Zero Page
E5-SBC-Zero Page
E6-INC-Zero Page
E7-NOP
E8-INX
E9-SBC-Immediate
EA-NOP
EB-NOP
EC-CPX-Absolute
ED-SBC-Absolute
EE-INC-Absolute
EF-NOP
F0-BEQ
F1-SBC-(Indirect),Y
F2-NOP
F3-NOP
F4--NOP
F5-SBC-Zero Page,X
F6-INC-Zero Page,X
F7-NOP
F8-SED
F9-SBC-Absolute, Y
FA-NOP
FB-NOP

AE-LDX-Absolute
AF--NOP
B0-BCS
B1-LDA-(Indirect),Y
B2-NOP
B3-NOP
D5-CMP-Zero page,X
D6-DEC-Zero page X
D7-NOP
D8-CLD
D9-CMP-Absolute,Y
DA-NOP
FC-NOP
FD-SBC-Absolute,X
FE-INC-Absolute,X
FF-NOP

附录C 6502指令执行拍数表

记忆符	直接数	绝对	零页	累加器	已包含	(间址,X)	(间址,Y)	零页,X	绝对,X	绝对,Y	相对	间址	零页,Y	
ADC	2	4	3			6	5	4	4	4				(1)
AND	2	4	3			6	5	4	4	4				(1)
ASL		6	5	2				6	7					
BCC											2			(2)
BCS											2			(2)
BEQ											2			(2)
BIT		4	3											
BMI											2			(2)
BNE											2			(2)
BPL											2			(2)
BRK					7									
BVC											2			(2)
BVS											2			(2)
CLC					2									
CLD					2									
CLI					2									
CLV					2									
CMP	2	4	3			6	5	4	4	4				(1)
CPX	2	4	3											
CPY	2	4	3											
DEC		6	5					6	7					
DEX					2									
DEY					2									
EOR	2	4	3			6	5	4	4	4				(1)
INC		6	5					6	7					
INX					2									
INY					2									
JMP		3										5		
JSR		6												

续表

记忆符	直接数	绝对	零页	累加器	已包含	(间址，X)	(间址，Y)	零页，X	绝对，X	绝对，Y	相对	间址	零页，Y	
LDA	2	4	3			6	5	4	4	4				(1)
LDX	2	4	3							4			4	(1)
LDY	2	4	3					4	4					(1)
LSR		6	5	2				6	7					
NOP					2									
ORA	2	4	3			6	5	4	4	4				
PHA					3									
PHP					3									
PLA					4									
PLP					4									
ROL		6	5	2				6	7					
ROR		6	5	2				6	7					
RTI					6									
RTS					6									
SBC	2	4	3			6	5	4	4	4				(1)
SEC					2									
SED					2									
SEI					2									
STA		4	3			6	6	4	5	5				
STX		4	3										4	
STY		4	3					4						
TAX					2									
TAY					2									
TSX					2									
TXA					2									
TXS					2									
TYA					2									

(1) 假如跨页加1。
(2) 同页转加1，跨页转加2。
(3) 每拍执行时间0.9778μs。

附录D　Apple Ⅱ兼容机的发展现状

近年来，国内外开发生产了多种Apple Ⅱ兼容机，如Apple Ⅱe、DJS-033、紫金Ⅱ、紫金ⅡA、中华学习机、小蜜蜂等。DJS-033、紫金Ⅱ与Apple Ⅱ完全兼容，具体问题请参考本书有关章节。紫金ⅡA、中华学习机等与Apple Ⅱe兼容，中华学习机增加了汉字处理功能。这里以Apple Ⅱe的特性为主导，简介Apple Ⅱ兼容机的特点以及它们之间的差别。

Apple Ⅱe与Apple Ⅱ兼容，功能有所增强，也更易操作。它能够运行Apple Ⅱ上的

所有软件。与Apple Ⅱ相比，Apple Ⅱe的主要特点是采用了大规模专用门阵电路，如MMU（存储器管理部件）、IOU（输入输出部件）和PAL（可编程阵列逻辑），大大减小了器件的数量，提高了系统工作的可靠性。从功能上讲，系统增强了存储器管理功能，尽管6502的直接寻址范围为64K字节，但Apple Ⅱe可以管理128K或更多的存储器。

下面分几个方面介绍：

1．键盘

Apple Ⅱe有一个类似于打印机式的键盘，共63个键。键盘编码采用美国标准信息交换码（ASCII）。键盘除提供字符编码外，还有四个移动光标键：左、右、上和下，它们的编码分别是$08、$15、$0A和$0B。键盘上还有七个特殊键，它们不产生ASCII码。如，你不能直接读CONTROL、SHIFT、CAPSLOCK键，但你按下它们中的一个，便可改变你按下的另一个键的编码。Apple Ⅱ的键盘没有CAPSLOCK键（锁定键）。按下锁定键，所有再按的26个英文字母都是大写；抬起锁定键。则所有再按的英文字母都是小写。Apple Ⅱe比Apple Ⅱ多了两个画着苹果图样的键（一个是实心的，称为SOLID-APPLE；另一个是空心的，称为OPEN-APPLE），它们与游戏输入口相连，与RESET键一起，用于选择不同的复位方式。Apple Ⅱe有三种复位方式：开机复位（冷启动）、热启动和强迫冷启动。按CONTROL键，再按一下RESET键，可引起机器以热启动方式复位。若按CONTROL和OPEN-APPLE键，再按一下RESET键，便可不关电源开关而实现冷启动，即强迫冷启动。按CONTROL和SOLID-APPLE键，再按一下RESET键，Apple Ⅱe即刻执行固化在ROM中的自检程序，然后进入正常的运行之中。

中华学习机的键盘有69个键，它没有SOLID-APPLE键。以TEST键替代了OPEN-APPLE键。它还增加了中文、西文键，用于中文输入和英文输入的转换。它的Quit键相当于CTRL-C，用于从监控返回BASIC。中华学习机的键盘还有5个功能键。F1：进入ASCII方式。F2：进入拼音方式。F3：进入区位方式。F4和F5供用户使用，它们的键码是$14和$06。

键盘通过电缆和26点的连接器与主机相连。读$C000，即可得到最近一次按键的编码；Apple Ⅱ键盘的设备码是$C00×（×＝0…F）。写$C010即可清除键入标志；Apple Ⅱ清除键入标志的设备码是$C01×（×＝0…F）。

2．存储器管理

Apple Ⅱe主板上的动态存储器（主存储器）为64K字节，由8片64K×1位的RAM组成。主RAM基本部分的地址范围是$0000-$BFFF，共48K字节。这些RAM供用户存放程序和数据，或被磁盘操作系统占用。$D000-$FFFF地址区间是存储体切换空间，ROM和RAM均可占用。主板上有16K只读存储器（内部ROM），它是由两片8K×8位ROM组成。占用$D000-$FFFF地址区间的内部ROM中存有APPLESOFT BASIC解释程序和监控程序。内部ROM的另一部分占用的地址是$C100-$CFFF，其中存有80列显示的管理程序和自检程序。

Apple Ⅱe具有对第二个64K存储器（辅存储器）的管理功能。配有辅存储器的电路

板插在60点的外设插座上。如80列文本显示卡，可为Apple Ⅱe扩充1K存储器，但它只能作为显示缓冲器。这是因为在80列显示方式下，需要2K显示存储器，而主存储器仅提供1K，而另1K只得由80列显示卡提供。如果80列卡上有64KRAM，若采用80列显示方式，其中1K仍作显示缓冲器，其它63K可用来存储程序和数据。在有辅存储器的情况下存储器分配情况如图D-1。

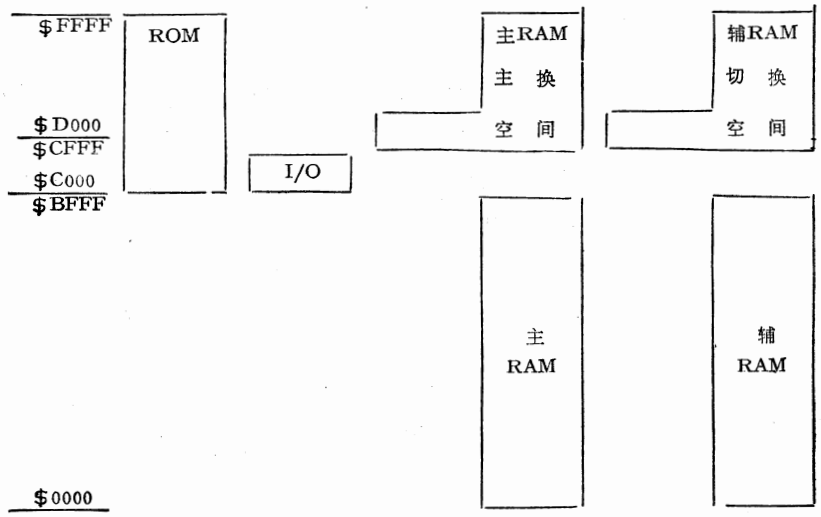


图 D-1 存储器分配

地址空间 $D000-$FFFF是存储体切换空间。它可被主板上的ROM占用，也可被主RAM或辅RAM占用。在后一种情况下，$D000-$DFFF又可对应两个4K存储器。以上存储体的切换均通过改变软开关的状态来实现。软开关的地址是 $C080-$C08F。Apple Ⅱe复位时，系统置ROM于可读，置主RAM于可写。

主、辅存储器转换也由一系列软开关控制。用户需要仔细安排这些开关状态，以实现正确地转换和读写。

所有对存储器的管理均由存储器管理部件（MMU）实现。MMU的引脚图如图D-2。引脚信号的功能如下：

$\phi 0$： 系统时基信号，频率为1MHz。

Q3： 非对称的2MHz信号。

$\overline{PRAS}$： 存储器行地址选通。

RA0-RA7：复用地址输出。

R/$\overline{W}$： 6502读写控制信号。

$\overline{INH}$： 禁止访问主存储器信号。

$\overline{DMA}$： 外部设备直接访问存储器控制信号。

$\overline{EN80}$ ： 访问辅存储器允许信号。

$\overline{KBD}$： 读键盘输入数据选通信号。

$\overline{ROMEN2}$：此信号有效时，允许访问内部ROM空间 $E000-$FFFF。

$\overline{ROMEN1}$：此信号有效时，允许访问内部ROM空间 $C100-$DFFF。

MD7：软开关状态输出位。

R/$\overline{W}$245：控制I/O数据总线与系统数据总线之间的流通方向。

$\overline{RAMEN}$：访问主存储器允许信号。

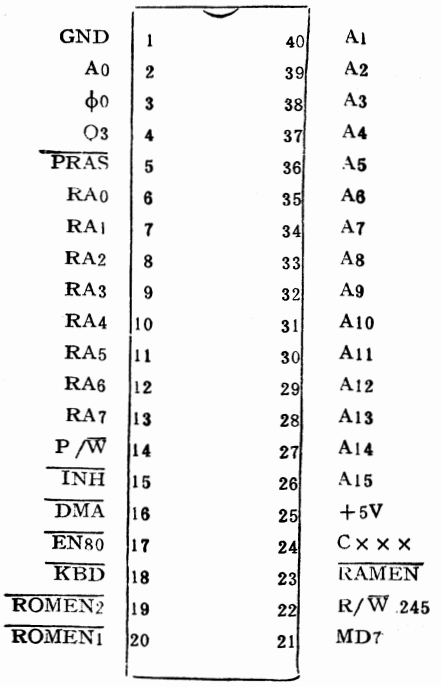


图 D-2 MMU引脚图

C×××：外设卡上存储器选通信号。

MMU中设有多种软开关。用户利用这些软开关对存储器进行分配、管理，使存储空间得以充分灵活地利用。为保持与Apple Ⅱ的兼容性，Apple Ⅱe的I/O设备码大多与Apple Ⅱ相同。只是Apple Ⅱe另需要一些设备码，为此它占用了$C000-$C010地址区间，表D-1列出这些软开关的名称和功能。

中华学习机具有汉字处理功能，提供拼音和区位输入方法，主机内配有全点阵汉字字库，其中包括国标一、二级汉字点阵。汉字系统占用辅存储器空间。辅存的$4000-$7FFF地址区间为ROM区，用作汉字点阵库。由于汉字点阵库为256K字节，因此这16K地址重复使用16次。$8000-$BFFF也是ROM区，存放转换码表。在输入汉字时，利用该表，将输入码转换为汉字内码。$D000-$FFFF这12K ROM存放汉字处理程序。

表 D-1 Apple Ⅱe新增的软开关

软开关名称	置位/复位地址	读/写	读状态设备码
80 STORE	$C001/$C000	W	$C018
RAMRD	$C003/$C002	W	$C013
RAMWRT	$C005/$C004	W	$C014
SLOTCXROM	$C007/$C006	W	$C015
ALTZP	$C009/$C008	W	$C016
SLOTC3ROM	$C00B/$C00A	W	$C017
80 COL	$C00D/$C00C	W	$C01F
ALTCHARSET	$C00F/$C00E	W	$C01E
TEXT	$C051/$C050	R/W	$C01A
MIX	$C053/$C052	R/W	$C01B
PAGE2	$C055/$C054	R/W	$C01C
HIRES	$C057/$C056	R/W	$C01D

3．输入输出部件（IOU）和外设插座

Apple Ⅱe和中华学习机的I/O部件与Apple Ⅱ相同，只是输入/输出控制由专用门

阵电路IOU来实现。IOU控制屏幕显示，产生读显示缓冲器地址，并将读出的显示数据和视频显示控制信号一起送视频信号合成器，IOU还产生录音机、扬声器的输出信号以及四个一位输出的信号。

IOU的引脚如图D-3。引脚信号的功能如下：

φ0：	系统时基信号，频率为1MHz。
Q3：	非对称的2MHz信号。
SEGA、SEGB、VC：	字符点阵垂直计数器。
$\overline{80VID}$：	80列显示使能信号。
CASSO：	录音机输出信号。
SPKR：	扬声器输出信号。
MD7：	软开关状态输出位。
AN0-AN3：	四个一位输出信号。
R/$\overline{W}$：	6502读写控制信号。
$\overline{RESET}$：	系统复位信号。
RA0-RA7：	复用地址线。
$\overline{PRAS}$：	存储器行地址选通。
A6：	6502地址线。
$\overline{C0\times\times}$：	IOU内部地址译码使能信号。
AKD：	按键的标志信号。
KSTRB：	键盘选通信号。
VID7、VID6：	显示数据的D7和D6。
$\overline{RA9}$、$\overline{RA10}$：	视频显示的控制位。
$\overline{CLRGAT}$：	彩色同步信号的选通信号。
$\overline{WNDW}$ ：	显示消隐信号。
$\overline{SYNC}$ ：	显示同步信号。

IOU内部有两组与显示有关的计数器：水平计数器和垂直计数器，IOU也生成屏幕扫描所需的同步信号。值得指出的是，Apple Ⅱe中的彩色信号采用NTSC制。由于我国电视采用PAL制，中华学习机中增加了把NTSC制转换为PAL制的电路。因此中华学习机可通过调制器直接与PAL制电视机相连。

Apple Ⅱe的主板上保留7个50点的外设插座，序号为1-7。插座上的信号与Apple Ⅱ相同。Apple Ⅱe增加了一个60点的辅助外设插座。80列显示卡，内存扩充卡、高分辨率显示卡均可插在此座。辅助外设插座提供了许多在50点插座上得不到的信号，这些信号大多与显示有关。这对诊断维修机器也是非常有益的。

4. 时基发生器

Apple Ⅱ中的时基信号由一系列移位寄存器产生，Apple Ⅱe和中华学习机中则由可编程阵列逻辑电路PAL完成。PAL的型号是PAL16R8，它有8个信号输入端和8个信号

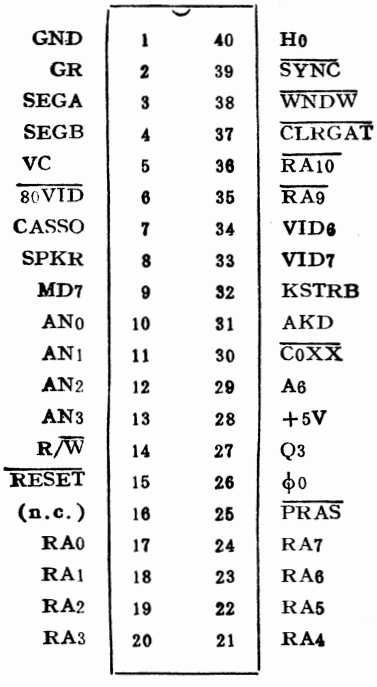


图 D-3 IOU引脚图

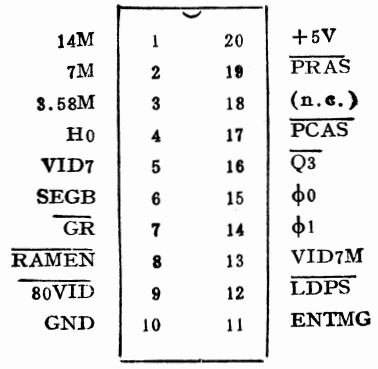


图 D-4 PAL引脚图

输出端。PAL的引脚如图D-4，引脚功能如下：

14M： 频率为14MHz的输入信号。

7M： 频率为7MHz的输入信号。

3.58M： 频率为3.58MHz的输入信号。

V1D7： 视频显示数据D7。

H0： 水平计数器最低位。

SEGB： 字符点阵垂直计数器。

$\overline{GR}$： 视频显示图形方式使能信号GR的非信号。

$\overline{RAMEN}$：RAM列地址选通的使能信号。

$\overline{80VID}$： 80列字符显示使能信号。

ENTMG：主时基使能信号。

$\overline{LDPS}$： 视频显示移位寄存器的装载使能信号。

VID7M：视频显示时钟。7M和14M。

ϕ1，ϕ0：视率为1MHz的系统时钟。

Q3： 非对称的2MHz信号。

$\overline{PCAS}$： RAM列地址选通信号。

$\overline{PRAS}$： RAM行地址选通信号。

随着技术的发展，新出现的Apple Ⅱ兼容机，功能将不断改善、增强。但它们的原理仍与Apple Ⅱ相同。