

# 77128303-1 Z80单板计算机使用手册



杭州大学

浙江永康无线电厂

香港金山公司

# 再 版 说 明

为了满足广大 TP 801 用户和读者的需要，本《使用手册》与《微处理器实用教材》两本书又重新排版付印。这次再版纠正了原版中一些印刷错误，并在文字上作了一些修改。由于目前大部分 TP801 用户均使用 TPBUG-A 监控程序，故在《使用手册》的第二章增写了“TPBUG-A 新增加的命令键”一节，并在第四章程序举例中，也相应的以使用 TPBUG-A 的命令为主来写。

编 者

1982 年 6 月 14 日

304

# 目 录

第一章 概述	( 1 )
1.1 引言	( 1 )
1.2 主要技术特性	( 1 )
1.3 功能简介	( 2 )
1.4 操作步骤及注意事项	( 3 )
1.5 《Z80 袖珍设计手册》使用说明	( 4 )
第二章 键盘操作说明	( 5 )
2.1 监控程序 TPBUG 简介	( 5 )
2.2 复位(RESET)按钮	( 6 )
2.3 16个十六进制数字键	( 6 )
2.4 MON (MONitor) 监控键	( 7 )
2.5 MEM EXAM(MEMory EXAMine 存储单元检查) 键	( 8 )
2.6 PORT EXAM (PORT EXAMine 口检查) 键	( 9 )
2.7 REG EXAM (REGister EXAMine 寄存器检查) 键	( 11 )
2.8 REG' EXAM (alternative REGister EXAMine 辅助寄存器检查) 键	( 12 )
2.9 BREAK POINT(设置断点)键	( 14 )
2.10 SINGLE STEP (单步执行程序) 键	( 15 )
2.11 EXEC (EXECute 连续执行程序) 键	( 16 )
2.12 CASS DUMP (CASSette DUMP 信息转储磁带) 键	( 16 )
2.13 CASS LOAD (CASSette LOAD 磁带输入) 键	( 16 )
2.14 PROM PROG (ePROM PROGrammer EPROM 写入) 键	( 17 )
2.15 NEXT 键	( 18 )
2.16 TPBUG-A 新增加的命令键	( 18 )
第三章 TP801—Z80单板计算机的结构和原理	( 24 )
3.1 概述	( 24 )
3.2 时钟电路	( 24 )
3.3 Z80—CPU	( 25 )
3.4 存储器	( 25 )
一、存储空间的分配	( 25 )
二、存储器译码	( 25 )
三、系统 RAM	( 27 )
四、PROM 1	( 27 )

五、PROM 2—EPROM .....	( 27 )
3.5 I/O 接口 .....	( 29 )
一、I/O 译码及空间分配 .....	( 29 )
二、Z80—PIO .....	( 30 )
三、Z80—CTC .....	( 30 )
四、键盘和显示 .....	( 31 )
五、录音机接口电路 (一) —转储 .....	( 33 )
六、录音机接口电路 (二) —输入 .....	( 34 )
3.6 其它部分 .....	( 34 )
一、单步逻辑 .....	( 34 )
二、中断电路 .....	( 34 )
三、复位电路 .....	( 34 )
四、电压保护电路 .....	( 35 )
五、S—100 总线 .....	( 35 )
六、布线区 .....	( 35 )
第四章 程序举例 .....	( 36 )
4.1 熟悉键盘操作和 TPBUG-A 命令的使用 .....	( 36 )
4.2 相对转移指令中偏移量的计算 .....	( 41 )
4.3 软件延时 .....	( 45 )
4.4 Z80—CTC 的应用 .....	( 46 )
4.5 Z80—PIO 的应用 .....	( 50 )
4.6 EPROM 的写入程序 .....	( 51 )
4.7 求十进制数的算术和 .....	( 55 )
附录一 TP801 原理图、安装位置图、零件表 .....	( 57 )
附录二 TP801 所用集成电路引脚图 .....	( 63 )
附录三 TPBUG 监控程序 .....	( 69 )
附录四 实验指导书 .....	( 133 )

# 第一章 概 述

## 1.1 引 言

近年来，微处理器/微型计算机的发展十分迅速。它的应用已深入到工业、农业、国防、科研、教育、管理以及日常生活（如家用电器、玩具）等各个领域。在自动控制和仪器仪表方面的应用尤为突出。随着大规模集成电路的发展，微处理器/微型计算机必将对计算机工业和计算机应用产生深远的影响。

为了推广微型计算机的应用，北京工业大学和香港京业公司合作，研制了一种新型的价格便宜而性能优良的 TP801—Z80 单板计算机。该机是使用 Z80 系列器件，做在一块印刷电路板上的完整的微型计算机。它结构简单，布局合理，功能齐全，用途广泛。

TP801—Z80 单板计算机作为“智能”部件，可用于生产过程控制、各种仪器和仪表或机械的单机控制、数据处理等等。它既可独立应用在小型自动控制系统中，又可用于分布系统中的前沿控制。该机尤其适合于初学者学习微型计算机的硬件、指令系统、编写程序的方法和技巧。因此，对大专院校学生和各行各业需要应用微型计算机的科技工作者来说，TP801—Z80 单板计算机也是一种经济实用的实验教学设备。

TP801—Z80 单板计算机还具有简易的开发功能，如在用户程序内可设置多至五个断点，可单步执行存于 RAM 或 PROM 中的程序，可对 2716/2758 EPROM 进行编程等。因而，可以将它用作调试样机，也可以直接将它用于微型机化 ( $\mu\text{c}$ -based) 产品中。TP801 可以在其布线区和两个 S—100 总线插座上扩展存储器板和 I/O 接口板等。为方便用户，已为 TP801 配置了接口扩展板，微型点阵式打印机，以及 ADC（模数转换器）和 DAC（数模转换器）板等，供用户选用。

## 1.2 主要技术特性

1. 中央处理单元为 Z80—CPU。
2. 时钟 ( $\phi$ ) 频率为 1.9968 MHz，便于使用 8080 A 的接口电路。晶体振荡器的频率为 3.9936 MHz。
3. RAM 为 4K 字节的 2114 静态读写存储器（共八片），也可只使用 2K 字节。
4. ROM 为 2K 字节，编入监控程序 TPBUG 或 TPBUG-A。
5. PROM 插座两个。可插入 4K 字节的 PROM 或 EPROM。
6. Z80—PIO 并行 I/O 接口芯片一个，它有两个 8 位可编程的 I/O 口，全供用户使用。
7. Z80—CTC 计数器/定时器芯片一个，它有四个通道：通道 0 供用户使用，其余由 TPBUG 或 TPBUG-A 使用。

8. 按键共 28 个, 16 个为十六进制数字键; 12 个为命令键。当使用 TPBUG 时, 这 12 个命令键的功能为:

- MEM EXAM (存储单元检查) 键
- PORT EXAM (口检查) 键
- REG EXAM (寄存器检查) 键
- REG' EXAM (辅助寄存器检查) 键
- EXEC (连续执行程序) 键
- SINGLE STEP (单步执行程序) 键
- MON (监控) 键
- NEXT 键
- BREAK POINT (设置断点) 键
- PROM PROG (EPROM 写入) 键
- CASS LOAD (磁带输入) 键
- CASS DUMP (信息转储磁带) 键

若使用 TPBUG-A, 则 12 个命令键具有二十种功能:

- MON' (换上挡的监控) 键
- MON (换下挡的监控) 键
- STEP (单步执行程序) 键
- EXEC (连续执行程序) 键
- DISP/MEM (偏移量计算/存储单元检查) 键
- REG/REG' (寄存器检查/辅助寄存器检查) 键
- DUMP/LAST (信息转储磁带/查上一个存储单元) 键
- LOAD/NEXT (磁带输入/下一) 键
- 2FB8/PROM (用户定义功能/EPROM 写入) 键
- 2FBA/MOVE (用户定义功能/存储块移动) 键
- 2FBC/PORT (用户定义功能/口检查) 键
- 2FBE/BP (用户定义功能/设置断点) 键

9. 显示器有六位 LED 数字显示, 通常左四位显示地址, 右两位显示数据。

10. 配有音频盒式磁带机接口。

11. 布线区为 2.5 × 7 英寸。

12. S-100 总线插孔两组。

13. 电源为 +5 V ± 5%, 1A。本机可以增配 7805 稳压器件, 这时输入须接 ≥ +7 V 直流电源。若要对 EPROM 进行写入, 尚须接入 +25 V ± 1V, 30mA 的电源。

## 1.3 功能简介

TP801—Z80 单板计算机系使用 Z80 系列器件。Z80—CPU 的指令系统有基本指令 158 条, 其中包含了 8080 A 的全部指令。此外, 还增加了数据块传送与查找、位操作

(置 1、置 0、测试)、相对转移、变址寻址、16 位的算术运算等。Z80—CPU 的内部寄存器也较 8080A 多,增设了 IX、IY,以及一套辅助寄存器。这些特点大大增强了 Z80—CPU 的处理功能,并非非常便于程序设计。Z80 的软件可以与 8080A 的软件相兼容。

本机使用八个廉价的静态读写存储器 2114,容量为 4 K 字节。板上配有 2K 字节的 ROM,存放监控程序 TPBUG 或 TPBUG-A,便于用户利用键盘或盒式磁带输入程序和数据,以及提供其他的软件功能。板上设有一个完整的 EPROM 编程器,允许用户将信息写入 2716/2758 EPROM 中,编程所需的电压为直流  $+25 \pm 1V$ ,最大电流为 30mA。

本机配有音频盒式磁带机接口,用转录线将盒式磁带录音机与机上插孔相连,可以方便地存取数据和程序。单板机上的红色发光二极管能指示磁带上有无信息。利用这只二极管的指示,可在一盘磁带上录制几个文件。RAM 和磁带之间的存取由命令键控制。板上配有六位数码显示,供显示地址和数据。通常左四位显示地址,右两位显示数据(对于 PC, SP, IX, IY 等 16 位寄存器而言,右四位显示数据)。板上还配有 16 个十六进制的数字键和 12 个命令键,它们的功能在第二章里介绍。

本机配有三个按钮和开关:按钮 S1 用来对整机提供 RESET (复位)信号。开关 S2 有两个位置可供选择,如果指向 MON RST 位置,则在复位后,显示器上出现“P”(或“-”),并扫描键盘的输入;如果指向 PROM1 RST 位置,则在初始化后进入 PROM 1 中的用户程序。开关 S3 用来选择 PROM2 中的 EPROM 处于“写入(PGM)”或“读出(READ)”的状态。

此外,板的左上方有两组 S—100 总线的插座孔。Z80—CPU 的地址、数据、控制总线都接到 S—100 总线插孔,以便在插座上扩充存储器及 I/O 接口板。板的左方有  $2.5 \times 7$  英寸的布线区,约可安装 25 片 IC (集成电路)。连接到布线区的信号有:

- Z80—CPU 的数据、地址和控制总线;
- Z80—PIO 两个口 (Port) 的数据线和联络线;
- Z80—CTC 通道 0 的输入和输出。

## 1.4 操作步骤及注意事项

一、请按以下规定连线:

1. 直流稳压电源、盒式录音机均接至交流 220V;
2. 用转录线将单板机上的 AUX 插孔与录音机上的 MIC 插孔相连;
3. 用转录线将单板机上的 EAR 插孔与录音机上的 MONITOR OUT 或 EAR PHONE 插孔相连;
4. 将单板机上的电源线接入直流稳压电源 +5V 输出。

二、设定单板机上的开关位置:

1. 通常将开关 S2 设定在 MON RST 位置;
2. 通常将开关 S3 设定在 READ 位置。

三、开机:

1. 接通直流稳压电源上的 +5V 开关;

2. 按下 RESET 按钮 S1, 计算机进入监控程序 TPBUG 或 TPBUG-A, 显示器上显示“P” (或“-”), 表明机器可接收命令。

四、输入用户程序, 并进行相应的调试和操作。请参阅第二和第四章。

五、如果进行第三步时, 不出现“P” (或“-”), 则须立即关闭电源, 仔细检查导线连接有否出错, 器件与插座的接触是否可靠, 各开关位置的设定是否正确, 直流稳压电源的输出电压是否满足  $+5V \pm 5\%$ 。

六、切忌用手直接触摸集成电路, 切忌对本机直接使用电源电压为交流 220V 的电烙铁, 以防损坏芯片。必须使用电烙铁时, 应将电源断开, 利用烙铁的余热进行焊接。

## 1.5 《Z80 袖珍设计手册》使用说明

TP801—Z80 单板计算机提供用户一本《Z80 袖珍设计手册》。这本小册子扼要叙述了 Z80—CPU 指令系统及其他芯片的编程要点。

该手册第 1 页是 Z80—CPU 内可供用户使用的寄存器。其中有一套与 8080 A 兼容的主寄存器, 另有一套相同的辅助寄存器, 此外还有一套专用寄存器。它们的使用请参考《微处理器实用教材—基础知识和 Z80 器件的使用》。第 2、3 页是摘要表示指令是如何影响各个标志位的。第 4 页是 8 位传送指令组, 黑体字的指令操作码与 8080A 相兼容, 其余的是 Z80 所独有的新指令, 新指令是变址的传送指令。第 5 页是对这些传送指令进行更为详细的说明。第 6、7 页是 16 位传送指令组, BC 和 DE 的内容可以直接由存储单元输入, 而不必通过 HL 寄存器。第 8、9 页是交换指令组和数据块传送和查找指令组。第 10、11 页是 8 位算术和逻辑指令组, 其中大多数与 8080A 相兼容, 新增加的指令也是一些变址操作指令。第 12、13 页是其他指令组, 其中有求补码指令和中断方式选择指令。中断方式 0 是仿照 8080A 的, 而方式 2 为 Z80 系列接口芯片所使用。

第 14、15 页是 16 位算术指令组。除增加了变址操作外, 还增加了两种只有小型计算机才具备的 ADC 和 SBC 指令。第 16、17 页提供了远较 8080A 更丰富的移位和循环移位操作的指令组。这些操作还能在以 HL, IX 和 IY 作为指针的存储单元上进行, 就象 6800 那样。此外还能将存储单元中的 BCD 数字与累加器 A 进行移位操作。第 18、19 页是位操作指令, 每一位都可被置位、复位和测试。第 20、21 页是转移指令组, 其中有二字节的相对转移, 而 8080A 只有三字节的转移指令。第 22、23 页是子程序调用和返回指令组, 其中只增加了两条新指令, RETI 和 RETN。这两条是从中断和不可屏蔽中断返回的指令。第 24、25 页是输入/输出指令。新增加的指令为以 C 寄存器作为 I/O 的指针, 数据传送的对象可以是 CPU 内 8 位寄存器。此外还可以进行数据块的输入/输出。第 26 页是 Z80—CPU 的中断结构。第 27、28 页是 Z80—PIO 和 CTC 的编程要点。第 29、30、31 页是 Z80—SIO 的编程要点。最后是 ASCII 字符表。



# 第二章 键盘操作说明

在这一章，逐个介绍本机中各个按键的使用，先扼要介绍监控程序 TPBUG 及相应的命令键的操作，最后一节介绍 TPBUG-A 及新增设的命令键的操作。更为详细的说明请参阅第四章及附录三。

## 2.1 监控程序 TPBUG 简介

TPBUG 固化在可擦的可编程序只读存储器 (EPROM) 中，在存储空间中它占用的地址为 0000—07FFH，此外，TPBUG 还使用了 112 个 RAM 单元。有了 TPBUG，就可以通过键盘向计算机输入机器语言水平的程序和数据。

TPBUG 可以划分为下列四个主要程序段：(1) 初始化、显示和键盘分析程序；(2) 键盘动作程序；(3) 实用子程序；(4) 表格。

### 一、初始化、显示和键盘分析程序

#### 1. 初始化 (0000—00F3H)

它设定栈指针  $SP = 2FC0H$ ，并将其保存在  $2FE2—2FE3H$  单元，然后设定 TPBUG 的栈指针  $SP = 2FA8H$ 。清除键盘状态标志。在最左端显示器所对应的显示单元 DISMEM 中填入“-”的代码 11H，其余填入空格的代码 10H。

#### 显示 DISUP (00F4—0122H)

它将六个用于显示的显示缓冲单元 (DISMEM—DSMEM5) 的内容依次轮流取出，送往 LED 显示器，每位数字保持显示一毫秒。

#### 3. 键盘分析 (0123—01CAH)

它等待并搜索键盘的输入。若没有按键压下，则又回到显示 DISUP 程序。若有按键压下，则对所按下的键进行分析。如果是数字键，则送入相应的显示单元；如果是命令键，则进入与该键功能相对应的键盘动作程序。

### 二、键盘动作程序 (0230—0633H)

本机有十二个命令键，相应的有十二个按键动作程序。这些动作程序的内容将在后面关于键盘操作的叙述中作简要介绍。

### 三、实用子程序 (0634—07A5H)

这些子程序都是供 TPBUG 中上述程序段使用的，也可以由用户调用。其中也包括一些中断服务程序。(TPBUG-A 中的实用子程序及入口地址与 TPBUG 相同)。

下面介绍几种常用的子程序。

#### 1. UIX3 (起始地址为 0634H)

它使用 IX 寄存器增 3, B 寄存器减 1。该子程序只影响 IX, B, F 三个寄存器。

#### 2. UFOR1 (起始地址为 063CH)

它将累加器 A 中的高 4 位作为一个十六进制数字写入 (IX) 单元, 将低 4 位写入 (IX + 1) 单元。该程序影响的寄存器有 A, B, F。

#### 3. D20MS (起始地址为 064FH)

它仅起延时 20ms 的作用, 即调用这个子程序后, 经过 20ms 后才返回, 该程序影响的寄存器有 H, L, F。

#### 4. UABIN (起始地址为 06B3H)

它将累加器 A 中的一个 ASCII 字符转换为二进制数, 再送回累加器 A。该程序使用的寄存器有 A 和 F。

#### 5. UBASC (起始地址为 06BBH)

它将累加器 A 中的低 4 位作为一个十六进制数字, 转换为 ASCII 字符, 再送回累加器 A。该程序使用的寄存器有 A 和 F。

### 四、表格 (07A6—07FFH)

## 2.2 复位 (RESET) 按钮

本机中有两种实现复位的方法: 一种是上电复位, 即一合上电源即自动地提供复位信号; 另一种是压下印刷线路板右方按钮开关 S1。

当开关 S3 处于 READ 位置, S2 处于 MON RST 位置时, 复位按钮的作用如下:

1. 使 Z80—CPU 处于初始状态。这些初始状态包括:

- 1) 中断允许触发器处于禁止状态;
- 2) 置寄存器 I, R 的内容为 00H;
- 3) 置程序计数器 PC 的内容为 0000H;
- 4) 置中断方式为 IM0。

在复位信号有效期间, 地址总线 and 数据总线处于浮动状态, 所有的输出信号均为无效。

2. 使本机从 0000H 开始执行监控程序的初始化部分, 详细内容请参阅上节及附录三。此时, 显示器的最左端应显示 TPBUG 的待命标志符号“-”。若是 TPBUG-A, 则待命标志符号为“P”(以下不再重复说明)。

3. 如果在用户程序中用 BREAK POINT (或 BP) 键设置了断点, 则压下复位按钮后, 所设置的断点被清除。

4. 如果开关 S2 处于 PROM1 RST 位置, 则 TPBUG 执行完初始化程序后, 自动转移到 PROM 1 中起始地址为 0800 H 的用户程序。这个功能使用户能方便地转入 PROM 1 中的用户程序, 而不必从键盘输入命令。

## 2.3 16 个十六进制数字键

这些键用来向计算机输入十六进制数字。这些数字可以是存储单元的地址、I/O 口

地址、寄存器标号、指令码，也可以是一些数据。每压下一个数字键，此数字即存入相应的显示缓冲单元。最先输入的数字送 (DISMEM = 2FF7H) 单元，依次为 (DSMEM1 = 2FF8H)，……(DSMEM5 = 2FFCH)，(DSMEM6 = 2FFDH)，(DSMEM7 = 2FFE H)。并将前六个单元的内容显示于六位LED显示器上，(DISMEM) 单元在最左边的LED上显示出来，具体操作如表 2.1 所示。

数字键与命令键的联合使用见以下各节。

表 2.1 数字键使用举例

按 键	显 示	说 明
<u>MON</u>	—	压下 MON 键或按复位按钮，显示“—”
<u>2</u>	2	压下第一个数字键 2
<u>0</u>	2 0	再压下数字键 0
<u>3</u> <u>4</u>	2 0 3 4	先后压下数字键 3, 4
<u>B</u> <u>6</u>	2 0 3 4 B 6	先后压下数字键 B, 6
<u>7</u>	2 0 3 4 B 6	压下第七个数字键 7 没有任何反应
<u>8</u>	—	压下第八个数字键 TPBUG 进行处理，由于没有命令键输入，连续输入 8 个数字毫无意义，所以控制进入 TPBUG 的初始化部分

## 2.4 MON (MONitor 监控) 键

MON 键有两种功能。一是中止现行程序的执行，当计算机执行 RAM 中的用户程序时（也即用户已压过 EXEC 键），压下 MON 键，通过 U15 单稳电路向 CTC 通道 2 的 C/T2 输入一个脉冲。由于在 EXEC 键的键盘动作程序（起始地址为 0230H）中，将 CTC 通道 2 设定为计数器工作方式，其初始常数为 01H，因而，C/T2 的脉冲导致 ZC2 上产生一个脉冲。这个脉冲通过 U33 和 U34 送往 CPU 的  $\overline{NMI}$  输入端，从而产生不可屏蔽中断，程序转入 0066H 及 01CBH 的中断服务程序。

这种功能在分析程序故障时尤其有用。例如，如果由于程序设计上的错误，或者由

于执行了一条暂停 (HALT) 指令, 或者由于陷入无休止循环, 那么用户可以使用 MON 键, 使控制返回到监控程序 TPBUG 而能保护 CPU 寄存器内容。这样, 用户就能知道压下 MON 键时程序执行到何处。

MON 键的另一种功能是, 中止或者退出当前的命令状态或输入数据, 使控制返回到 TPBUG 的初始化部分, 等待下一个按键输入, 以便输入新的命令或数字。

通常要进行一种新的操作方式之前, 均需压下 MON 键, 使显示器上出现 “-” 后, 才能压下新的命令键或数字键。

MON 键与复位按钮的作用相似, 都是使控制返回到 TPBUG。其不同点是: MON 键能保护 CPU 寄存器的状态以及 TPBUG 所使用的某些 RAM 专用单元 (主要是 键盘状态标志)。表 2.2 是说明 MON 键使用的例子。

表 2.2 MON 键使用举例

按 键	显 示	說 明
<u>MON</u>	—	准备接受命令
<u>2</u> <u>0</u> <u>0</u> <u>0</u>	2 0 0 0	輸入四个数字, 作为存儲单元地址
<u>MEM EXAM</u>	2 0 0 0    × ×	检查2000单元, ××为讀出值
<u>MON</u>	—	退出以上工作方式
<u>8</u> <u>4</u>	8 4	輸入口地址
<u>PORT EXAM</u>	8 4    × ×	检查口 84, ××为讀出值

## 2.5 MEM EXAM (MEMory EXAMine 存儲单元检查) 键

MEM EXAM 键用来检查或更改 RAM 单元的内容, 也就是通过键盘命令对 RAM 单元进行读、写操作。此外, 也可对 ROM 或 PROM 进行读操作。

当显示器最左端出现 “-” 标志后, 通过键盘输入表示地址的四个十六进制数字。先送地址的高位数字, 后送低位数字。每送入一个数字, 就立即显示在显示器上, 最高位显示在最左端。然后压下 MEM EXAM 键, 则在最右边两个显示器上出现该存储单元的内容。如果还没有送完四个数字的地址就压下 MEM EXAM 键, 那么压下此键不起任何作用, 右边

两个显示器上不会出现数字，也不影响以前输入的数字。在这种情况下，如果继续输入数字，在送完地址的四个数字之后再一次压下 MEM EXAM 键，那么在显示器上仍能出现该单元的内容。

表 2.3 MEM EXAM 键使用举例

按 键	显 示	說 明
<u>2</u> <u>1</u> -	2 1	压下两个数字键
MEM EXAM	2 1	此键不起任何作用
<u>3</u> <u>4</u>	2 1 3 4	繼續輸入数字3、4
MEM EXAM	2 1 3 4    × ×	显示該单元内容5、6

其次，介绍如何更改存储单元的内容。（参阅表 2.4）

在完成上述检查存储单元内容的操作之后，显示器上有六个数字，左边四个数字为地址，右边两个数字为存储单元的内容。如果再输入两个数字，这两个数字即被写入该存储单元，并被显示出来，原先的存储单元内容随即消失。值得注意的是，只有送完两个数字后，显示才会更新，如果只输入一个数字，则这个数字保存在 (DSMEM6 = 2FFDH) 单元中，不起任何作用。只有当第二个数字输入后，才会将这两个数字写入该存储单元中，然后从此存储单元中送至 DSMEM4 和 DSMEM5 单元中，并被显示出来。如果由于偶然的疏忽，用户企图更改 ROM 或空单元的内容，虽然在 DSMEM6 和 DSMEM7 单元会保存新输入的数字，但是它无法通过存储单元而进入 DSMEM4 和 DSMEM5 单元，从而不会更改右边显示的内容。此时用户会意识到自己的操作错误。

如果连续压下 MEM EXAM 键，则其结果只是重复读出并显示出该存储单元的地址和内容。如果压下 NEXT 键，则显示下一个单元的地址和内容。

## 2.6 PORT EXAM (PORT EXAMine 口检查) 键

PORT EXAM 键的功能与 MEM EXAM 键相似，它用来检查或更改口的内容。

当显示器最左端出现“-”标志后，通过键盘输入口地址的两个十六进制数字，先送地址的高位数字，后送地址的低位数字。由于最多只能访问 256 个 I/O 口，所以口地址只需要两个十六进制数字。然后压下 PORT EXAM 键，则在最右边两个显示器上将出现该口的内容。由于它的功能与 MEM EXAM 键相似，请读者参阅上节和本节的例题（见表 2.6），例中将扼要地介绍这些特性，从而不再作文字的叙述。

表 2.4

## 用 MEM EXAM 键更改存储单元的内容

按 键	显 示	说 明
<u>MON</u>	—	或按复位按钮
<u>2</u> <u>0</u> <u>0</u> <u>0</u>	2 0 0 0	依次输入存储单元地址
<u>MEM EXAM</u>	2 0 0 0 × ×	× × 为原读出值
<u>0</u> <u>6</u>	2 0 0 0 0 6	用 0 6 更改原来内容
<u>4</u>	2 0 0 0 0 6	只输入一个数字, 不会更改原来内容
<u>5</u>	2 0 0 0 4 5	送完两个数后, 更改原来内容
<u>NEXT</u>	2 0 0 0 × ×	显示下一单元的地址和内容
<u>7</u> <u>B</u>	2 0 0 1 7 B	用 7 B 更改原来内容
<u>MON</u>	—	准备检查新单元 0000H
<u>9</u> <u>0</u> <u>0</u> <u>6</u>	0 0 0 6	地址错了, 应是 0000
<u>MON</u>	—	消去上述数字
<u>0</u> <u>0</u> <u>0</u> <u>0</u>	0 0 0 0	输入正确的地址号
<u>MEM EXAM</u>	0 0 0 0 3 1	读出 0000H 单元内容
<u>4</u> <u>5</u>	0 0 0 0 3 1	再输入两个数字, 但无效, 因为不能更改 ROM 内容

值得注意的是，如果连续压下 PORT EXAM 键，则将连续显示该口的内容。这种特性对于CTC芯片是很有用的，通过这种操作可以了解CTC芯片中减1计数器的内容的变化情况。

压下 MON 键，可以使之退出 PORT EXAM 工作方式。

本机中共用11个I/O口，如表2.5所示。顺便指出，有些口只能读出，有些口只能写入，有些口（如PIO）包含若干个寄存器，读出和写入的操作与操作方式有关，使用时应予以注意。

**表 2.5 TP 801 使用的 I/O 口**

口 地 址	口	说 明
80H	PIO A 口数据寄存器	口内有数据输入寄存器和数据输出寄存器，读写过程与操作方式有关
81H	PIO B 口数据寄存器	同 上
82H	PIO A 口控制寄存器	只能写入，不能读出
83H	PIO B 口控制寄存器	同 上
84H	CTC 通道0	可读写，但读出的是减1计数器内容
85H	CTC 通道1	同 上
86H	CTC 通道2	同 上
87H	CTC 通道3	同 上
88—8BH	七段选择锁存器 SEGLH	只能写入，不能读出
8C—8FH	数字选择锁存器 DIGLH	同 上
90—93H	键盘选择三态输入缓冲器 KBSEL	只能读出，不能写入

## 2.7 REG EXAM (REGister EXAMine 寄存器检查) 键

REG EXAM 键用来检查或更改下列CPU寄存器的内容：A, B, C, D, E, F, H, L, I, IFF, PC, IX 和 IY。SP 寄存器的内容只能被检查，不能被更改。此键的使用与 MEM EXAM 相似，唯一的不同是，它不能和 NEXT 键配合使用。

先压下代表寄存器号的数字键，在显示器的左边出现该数字。然后压下 REG EXAM 键，显示器右边的两个或四个数字即为寄存器中的内容。IFF 是中断允许触发器的状态。00 表示禁止中断，04 表示允许中断。

如果要改变所显示寄存器的内容，只需再压下两个数字键（若是 IX、IY 或 PC，则需

表 2.6

PORT EXAM 键使用举例

按 键	显 示	說 明
<u>MON</u>	—	准备接受命令
<u>8</u>	8	输入口地址
<u>PORT EXAM</u>	8	未输完口地址即压下 PORT EXAM键, 不起任何 作用
<u>0</u>	8 0	输完口地址80
<u>PORT EXAM</u>	8 0      × ×	查口的内容, ××为读出值
<u>NEXT</u>	8 1      × ×	查下一个口的内容, ×× 为读出值
<u>MON</u>	—	准备接受命令, 检查口90
<u>8 8</u>	8 8	输入错误的口地址88, 应 为90
<u>MON</u>	—	准备接受新的命令
<u>9 0</u>	9 0	输入正确的口地址90
<u>PORT EXAM</u>	9 0      7 F	检查口90的内容

要输入四个数字)。

用户可以压下 MON 键使之退出 REG EXAM 工作方式, 或检查另一个寄存器。

在 TPBUG 的大多数按键工作方式下, CPU 寄存器的内容都保存在 RAM 的“用户寄存器存放区”。每当压下 EXEC 键 (连续执行程序) 或 SINGLE STEP 键 (单步执行程序) 而进入这两个按键的键盘动作程序时, 此动作程序首先将“用户寄存器存放区”的内容送入各 CPU 寄存器而后进入用户程序。每当执行程序时遇到了断点或单步执行程序结束时, 或者用 MON 键中止用户程序的执行时, 也就是说, 每当退出用户程序而进入 TPBUG 的控制时, CPU 中各个寄存器的内容又推入“用户寄存器存放区”, 以使用户进行更改和读出。

## 2.8 REG' RXAM (alternative REGiser EXAMine 辅助寄存器检查) 键

REG' EXAM 键用来检查或更改下列 CPU 辅助寄存器的内容: A', B', C', D',



表 2.7

## REG EXAM 键使用举例

按 键	显 示	說 明
<u>MON</u>	—	准备接受命令
<u>A</u> <u>REG EXAM</u>	A                    × ×	检查累加器 A 的内容, × × 为读出值
<u>8</u>	A                    × ×	用 8 4 写入 A, 输入第一个数字 8 没反应
<u>4</u>	A                    8 4	两个数字 8 4 都输入后, A 内容被更改
<u>MON</u>	—	准备接受命令
<u>2</u> <u>REG EXAM</u>	2            2 F    C 0	检查 SP 的内容, 为 2FC0H
<u>2</u> <u>F</u> <u>C</u> <u>2</u>	—	企图更改 SP 的内容为 2FC2, 没有成功
<u>1</u> <u>REG EXAM</u>	1            2 0    0 0	检查 PC 的内容, 为 2000H
<u>2</u>	1            2 0    2 0	2 进入显示
<u>3</u>	1            2 0    2 3	3 进入显示
<u>4</u>	1            2 0    2 3	没有响应
<u>5</u>	1            2 3    4 5	用 2,3,4,5 更改 PC 的内容
<u>NEXT</u>	—	NEXT 键不起作用

E', F', H' 和 L'。用 “'” 来表示辅助寄存器。该键的操作和使用与 REG EXAM 相同, 不再赘述。

## 2.9 BREAK POINT (設置断点) 鍵

该键用来在用户程序 (RAM 区) 中设置一至五个断点。在调试用户程序时, 利用断点可以在用户程序的某处暂停执行, 以便用户检查上一段程序设计中是否有发生差错, 进而修改程序。

表 2.8 設置断点的使用举例

按 鍵	显 示	說 明
<u>MON</u>	—	准备接受命令
<u>2</u> <u>0</u> <u>1</u> <u>5</u>	2 0 1 5	先輸入第一个断点地址
<u>BREAK POINT</u>	2 0 1 5	断点被接受, (BFLG)=1 2015被送往(2FE4)(2FE5) 单元
<u>MON</u>	—	
<u>2</u> <u>0</u> <u>2</u> <u>4</u>	2 0 2 4	輸入第二个断点地址
<u>BREAK POINT</u>	2 0 2 4	第二个点被接受, (BFLG) =2, 2024 被送往(2FE7)(2 FE8) 单元
<u>MON</u>	—	
<u>2</u> <u>0</u> <u>0</u> <u>0</u>	2 0 0 0	輸入要执行的程序的起始 地址
<u>EXEC</u>	2 0 1 5	从2000H 单元开始执行程序, 暂停于第一个断点处显示 断点地址 (即PC) 及累加 器 A 的内容
<u>EXEC</u>	2 0 2 4	繼續从 2015H 单元执行 程序暂停于第二个断点处, 显示该断点地址 (PC) 及累 加器 A 的内容, 上述三步可 反复进行
<u>MON</u>	—	控制轉回 TPBUG
<u>2</u>	2	輸入数字2
<u>BREAK POINT</u>	2	由于没有送完四个数就压 下BREAK POINT 鍵, 从而 使 (BFLG) = 0, 断点表 BPTAB中的信息不起作用, 显示保持原状

或者修改寄存器和 I/O 口中的内容。

设置断点的操作如表 2.8 所示。每当输入一个断点后，断点标志 (BFLG = 2 FF 4 H 单元) 的内容增 1，断点地址被送往起始地址为 2FF4H 的“断点表” BPTAB。该表共有 15 个单元，每个断点占用三个存储单元，其中两个存放断点地址，另一个存放用户程序断点处指令操作码的第一个字节。在输入断点地址后按下 BREAK POINT 键，显示的地址应暗后复明，表示断点已被接受。如果输入的断点数多于五个，则多设的断点输入将是无效，此时显示器上出现标志“-”。最初输入的五五个断点仍保持不变。

当执行设有断点的用户程序时，先执行第一条指令，然后进入不可屏蔽的中断服务程序，装配断点，即将 RST8 (CFH) 单字节指令代替各个断点的指令操作码的第一个字节，而此指令操作码的第一字节均被送往 BPTAB。然后继续执行程序，直到断点处，执行 RST8 指令。接着进入起始地址为 0008H 的服务程序，它暂停程序的执行，CPU 寄存器的内容均保存在“用户寄存器存放区”，并将断点表 BPTAB 中所有的断点指令操作码的第一个字节 (不管是否已经执行) 送回用户程序，取代 RST8 指令，以便用户进行检查或更改，此时显示 PC 和 A 的内容。

如果按下 EXEC 键，可继续执行用户程序，如果程序中还有其余断点，则其过程与上述相同。

在每个断点处暂停用户程序时，用户可用 MON 等键对已执行的程序和执行结果 (寄存器和 I/O 口) 进行检查或修改。

如果用户是清除已设置的断点，可以使用下列三种方法：

1. 按下复位按钮。
2. 按下 SINGLE STEP 键。
3. 在输入断点地址的四个数字之前，也就是说在输入零到三个数字后，紧接着压下 BREAK POINT 键。

上述三种方法都是将断点标志 (BFLG) 清零。从而使断点表中的数据无效。

通常 MON 键对设置断点没有任何影响，但是下列情况是例外。如果程序遇到 HALT 指令而停止，此时压下 MON 键，使控制转回 TPBUG，但是用户程序中的 RST8 指令并没有被取代，而仍然留在用户程序中。在调试过程中，用户应该注意这种情况的发生。

## 2.10 SINGLE STEP (单步执行程序) 键

此键可以用来每次执行程序中的一条指令。执行完后，显示 PC (即下一条指令的地址) 和 A 的内容。此时，用户可以使用 MON 键以及其他的按键来检查或修改程序、I/O 口或 CPU 寄存器。

用户可以反复地压下 SINGLE STEP 键，使程序一步一步地执行，显示器上可以看到下一次要执行的指令的地址。对于条件转移 (conditional jump) 和条件转子 (conditional call) 指令，这种特性是很有用的。用户可以了解程序是否发生转移以及跳转到什么地址，也就是说，测试的“条件”是否满足。在一步一步地执行程序时，用户可以更改 PC 的内容，强使程序转移到一个新的地址。

不仅RAM中的程序可以单步地执行,而且ROM,PROM<sub>1</sub>, EPROM中的程序也可以单步地执行。值得注意的是,单步执行与CTC通道2有关的指令,很易产生差错,用户应尽量避免这种情况。

## 2.11 EXEC (EXECute 连续执行程序) 键

这个键用来连续执行RAM,ROM,或EPROM的程序。它有两种使用方式:

1. 先输入要执行的程序起始地址的四个数字,然后压下EXEC键,即从此地址开始执行程序。如果要程序的起始部分开始执行,通常使用这种操作方式。见表2.8。
2. 仅仅压下EXEC键。此时从保存在“用户寄存器存放区”中的PC的现行地址开始执行程序。如果在连续执行程序时遇到断点,或是单步执行程序而暂停程序的执行,而此后要求连续执行程序,则只需压下EXEC键即可。这种操作方式请参阅表2.8,

## 2.12 CASS DUMP (CASSette DUMP 信息转储) 键

此键用来将RAM中的信息转储到盒式录音机的磁带中。在转储过程中使用美国“堪萨斯(Kansas)城标准”。传送信息的速率为300波特(Baud),即以八个2400赫音频脉冲信号表示“1”,以四个1200赫音频脉冲信号表示“0”。在一个文件的转储过程中,开头有40秒全“1”的导引信号,末尾有5秒全“1”的结尾信号。

转储的操作过程如下:

1. 用转录线将本机的J2(AUX)端与录音机的“AUXIARY”或“MIC”输入端相联。
2. 将磁带装入录音机。
3. 利用MEM EXAM键,将内存中需要转储的信息的起始地址装入2FC0H和2FC1H单元(高字节置入2FC0H,低字节置入2FC1H)。
4. 利用MEM EXAM键,将内存中需要转储的信息的最终地址置入2FC2H和2EC3H单元(高字节置入2FC2H,低字节置入2FC3H)。
5. 压下MON键,使显示器出现“-”。将录音机置成录音方式,然后压下CASS DUMP键。“-”将消失。
6. 不需要进行音量调节,因为录音机内有AGC自动增益控制。当转储完成后,显示器上再次出现标志“-”。此时按下录音机的STOP键,停止走带。整个转储过程至少要45秒。

## 2.13 CASS LOAD (CASSette LOAD 磁带输入) 键

此键用来将录音机磁带中的信息输入RAM。输入的操作过程如下:

1. 用转录线将录音机的“MONITOR OUT”或“EAR PHONE”孔与本机的J1(EAR)孔相接。
2. 将磁带走到相应位置。

3. 将录音机的高音控制调到最大，低音控制调到最小，音量控制调到最小。
4. 压下 MON 键，显示器上出现标志“-”。然后压下 CASS LOAD 键，“-”标志消失。
5. 将录音机置成放音方式，逐渐增大音量，直至 LED（发光二极管）出现亮光。然后再增大音量20%。在整个输入过程中，LED 将保持发光。
6. 如果输入过程取得成功（即核实了各个记录的“检查和”Checksum\*），则TPBUG将显示标志“-”。此时可以关闭录音机。

7. 如果输入过程中发现某个记录的“检查和”有差错，则显示下一个要输入的记录的第一个字节的存放地址。这种情况表明，刚才输入的记录有差错，而在这个有差错记录以前的输入是成功的。此时应重新进行输入，并核实音量和音调设置是否恰当。

8. 在同一条磁带上可以记录好几个文件（file）。当走带进入一个文件时，LED 显示器发亮；而进入文件之间的间隙时，则 LED 不发亮。用户可以利用这个特点来识别不同的文件。

用户也可以用语言标志或录音机上的走带计数器对文件进行识别。

## 2.14 PROM PROG (ePROM PROGrammer, EPROM 写入) 键

在本机中，可以将 RAM 中起始地址为 2000H 的数组写入插在 PROM2 插座（起始地址为 1000H）中的 2716/2758 型 EPROM。对 EPROM 进行写入，还需要有一个  $\pm 1\text{V}$ ，30mA 的辅助电源。此电源应接到线路板上右边标有标记  $\pm 25\text{V}$  的焊点上。

对 EPROM 进行写入的操作过程如下：

1. 将需要写入的 EPROM 用紫外灯穿过器件窗口进行照射，以擦除其中原有的内容，即各单元内容均应为 FF。如果是未使用过的新器件，则可以免去此步。

所用紫外线的波长为  $2537\text{\AA}$ ，照射强度为  $12000\mu\text{W}/\text{cm}^2$ ，照射能量为  $15\text{W}\cdot\text{sec}/\text{cm}^2$ 。通常照射距离为 2.5~5.0cm，照射时间为 20—40 分钟，即可擦除 EPROM 的内容。

2. 在关闭电源的情况下，将 EPROM 插入 PROM2 插座。

3. 合上 +5V 和 +25V 电源。

4. 用 MEM EXAM 键或 CASS LOAD 键将所需要写入 EPROM 的数据输入 RAM（从 2000H 开始）。

5. 压下 MON 键，LED 显示器上出现标志“-”。

6. 向微型机输入四个十六进制数字，表示要向 EPROM 写入的字节数。先送高位数字。

7. 将开关 S3 置于 PGM 位置，压下 PROM PROG 键，显示器熄灭，并进行写入。每个字节的写入约需要 52 ms。

\* 通常要输入的文件（可以是“批数据”也可以是一个程序）由好几个记录构成。记录的格式详见第四章，每个记录有一个“检查和”。

写入完毕后，有两种可能的结果：

一（多半）是显示器上出现标志“-”。这表示写入 EPROM 的内容与 RAM 核对后无误。

另一是在显示器上出现六个数字，分别表示 EPROM 中第一个与 RAM 内容不相符的单元的地址（左边四个数字）和内容（右边两个数字）。如果压下 NEXT 键，键盘动作程序将继续核对 EPROM 中的内容。如果没有错误，则显示器上出现“-”。如果有错误，则继续给予显示，直至用 NEXT 键检查完毕为止。在写入 EPROM 过程中出现错误，往往是由于使用了没有“擦净”的 EPROM，或者所用的 EPROM 是废品。

在写入完成以后，将开关 S3 置于 READ 位置。

还有一点需要提醒用户注意，在 EPROM 写入过程中要插入 52.5ms 的等待（Wait）状态。在等待状态中 Z80—CPU 暂停对动态存储器的刷新，从而使动态存储器中的内容丢失。如果用户需要扩充使用动态存储器，则应对此问题予以注意。本机中使用静态存储器，故不会产生上述问题。

在第四章中我们将介绍一种程序，它能够从 RAM 或 PROM1，ROM 的任意单元向 PROM2 的任意单元进行写入。在 RAM 顶部（2F90—2FFFH）有 112 个供 TPBUG 使用的专用单元，不能存放 EPROM 写入的内容。

## 2.15 NEXT 键

NEXT 键用于下述三种操作方式：检查存储器，检查口，检查 EPROM 的下一个写入错误。它们的操作和特性已如前述，此处不再重复。

如果在其他操作方式中使用 NEXT 键，则不会产生任何反应。此时在显示器上出现“-”，控制转入 TPBUG 的初始化部分。

## 2.16 TPBUG-A 新增加的命令键

TP801 单板计算机配用经改进的监控程序 TPBUG-A，丰富了键盘功能。也就是说，虽然与 TPBUG 一样，仍为 2K 字节，但是 TPBUG-A 使 TP801 单板机中的八个命令键成为双功能键。除了一个作为双功能键的上挡和下挡选择外，TP801 单板机配用 TPBUG-A 后相当于增加了七个新的命令键。

TP801 单板计算机配用 TPBUG-A 的键盘如右图所示。

图中带黑框的按键，功能与原配用 TPBUG 时完全相同。除此以外的按键中，NEXT 键、MEM（存储单元检查）键、REG（寄存器检查）键、REG'（辅助寄存器检查）键、PORT（口检查）键、PROM（EPROM 写入）键、DUMP（信息转储磁带）键、LOAD（磁带输入）键、BP（BREAK POINT，设置断点）键等，其控制功能与使用 TPBUG 相同，仅键盘的位置有所不同（有的是上挡，有的是下挡键）。另外八个命令键，则完全是新设置的，他们是：

1. MON'（交换）键

2FB8 PROM	2FBA MOVE	2FBC PORT	2FBE B P
DISP MEM	REG' REG	DUMP LAST	LOAD NEXT

7 H	8 L	9	A	MON'
4 IX	5 IY	6 I	B	MON
1 PC	2 SP	3 IFF	C	STEP
0	F	E	D	EXEC

此键除了具备 MON 键的基本功能——使系统进入监控程序待命状态外，尚完成双功能键的上下挡更换。

以 REG'/REG 键为例，上挡键为 REG'，下挡键为 REG。当按下 MON' 后再按此键，TPBUG-A 接受上挡键命令进行 REG' 访问。如果先按下的不是 MON'，而是 MON，TPBUG-A 则进行 REG 访问。为提示用户监控程序处于上下挡中不同状态，按下 MON' 和 MON 的标志是不相同的。前者为撇号“'”，后者为“P”（TPBUG 为“-”号）。

例：假定 A 寄存器内容为 AA，A' 寄存器内容为 BB，用 REG'/REG 命令键读出它们的内容，并更换 A' 寄存器内容为 06，其操作如表 2.9。

其余双功能键的换挡操作与上例所示原理相同。

## 2. LAST (查上一个存储单元) 键

此系新增加的一个常用命令键，用以访问显示器现行存储单元的上一个存储单元。这一点与 NEXT 键正好相反。但 NEXT 键不仅可以访问（读或写）下一个存储器单元，而且还可以访问下一输入/输出口，以及 EPROM 编程时的下一个错误检查。而 LAST 键只能检查存储器。从下例可知，用户在操作时，不仅可以连续按下 LAST 键来顺序显示上一个存储单元内容，而且可以交叉使用 LAST 和 NEXT 键，灵活地更换所要访问的存储单元。

例，假定 2000H 至 2003H 诸单元中存有 00，11，22，33 等数据，用 LAST 键检查它们，其操作如表 2.10。

表 2.9

MON' 键使用举例

按 键	显 示	說 明
<u>MON</u>	P	TPBUG-A 待命标志“P”
<u>A</u>	A	要求访问 A 寄存器
<u>REG'/REG</u>	A A	执行下档命令, 显示 A 内容
<u>MON'</u>	'	结束上档命令, 标志为“'”
<u>A</u>	A	要求访问 A 辅助寄存器
<u>REG'/REG</u>	A' B B	这次执行上档命令
<u>0</u>	A' B B	向 A' 寄存器写入 06H
<u>6</u>	A' 0 6	完成写入

表 2.10

LAST 键使用举例

按 键	显 示	說 明
<u>MON</u>	P	TPBUG-A 待命, 下档键有效
<u>2 0 0 3</u>	2 0 0 3	给出存储单元地址
<u>DISP/MEM</u>	2 0 0 3 3 3	下档有效, 存储器检查
<u>DUMP/LAST</u>	2 0 0 2 2 2	下档有效, 查上一个存储单元
<u>DUMP/LAST</u>	2 0 0 1 1 1	访问上一个存储单元
<u>DUMP/LAST</u>	2 0 0 0 0 0	访问上一个存储单元
		用户欲返回 2001 并写入 A6
<u>LOAD/NEXT</u>	2 0 0 1 1 1	下档有效, 检查下一个存储单元
<u>A</u>	2 0 0 1 1 1	写入 A6
<u>6</u>	2 0 0 1 A 6	完成写入



从此例中可知，由于一开始按下的是 MON 键，故各命令键均为下挡有效。

### 3. MOVE (存储块移动) 键

只要六位显示器左边四位显示一个地址，不论右边两位是否显示该单元内容，按一下 MOVE 键，则从所示地址单元开始，直到 2EFFH 单元为止，各存储单元内容均向下移动一个字节。显示器原显示地址单元的内容被清零（相当于插入了一条单字节的空操作指令 NOP）。

TP801 单板计算机在其 4K 字节 RAM 之中，分配给用户可用的区域为 2000H 至 2F87H。使用 MOVE 键可以移动的存储单元范围为 2000H 至 2EFFH。也就是说，从 2F00H 开始至 2F87H 为止的 136 个字节不会向下移动。这 136 个字节可放置数据表格和变量。

例：表 2.11 显示 MOVE 键的功能，所示存储单元内容是假设的，此操作的目的是想在 2300 处插入一条新的指令（两字节），使用 MOVE 键使 2300 开始至 2EFF 为止的内容移动。

此键在用户程序调试中，为修改程序提供了方便。

表 2.11 MOVE 键使用举例

	按 MOVE 前		按 MOVE		再按一次 MOVE	
显示	2300		2301	AF	2302	AF
操作结果	2300	AF	2300	00	2300	00
	2301	22	2301	AF	2301	00
	2302	00	2302	22	2302	AF
	2303	2F	2303	00	2303	22
	⋮		⋮		⋮	
	⋮		⋮		⋮	
	2EFE	FF	2EFE	xx	2EFE	xx
	2EFF	BB	2EFF	FF	2EFF	xx
2F00	AA	2F00	AA	2F00	AA	

### 4. DISP (相对转移偏移量计算) 键

此键所完成的命令与使用 TPBUG 时起始地址 00C0H 的程序是一样的，即进行相对转移指令操作数——偏移量的计算，显示计算结果，提示计算的有效性（是否超过单字节范围）；计算结果写入指令的操作数字节。与 TPBUG 相比，TPBUG-A 选用寄存器对存储源指令地址（IY 寄存器对）和转移目的地址（IX 寄存器对），并用 DISP 按键直接启动程序，从而方便了用户。

例：某 JR 指令的地址为 2300H，转移目的地址为 22F0H，计算相对偏移量的操作如表 2.12 所示：

如果用户忘记跳转指令的地址，可用接续表 2.12 的操作方法找回，并进行检查一下计算结果是否已写入指令的第二字节。

表 2.12

DISP 键使用举例

按 键	显 示	说 明
<u>MON</u>	P	“P”提示：TPBUG-A待命且双功能键下档有效
<u>IY</u>	5	IY寄存器对访问，“5”代表“IY”。
<u>REG'/REG</u>	5 × × × ×	下档有效，“×”为无用数
<u>2 3 0 0</u>	5 2 3 0 0	输入源指令地址
<u>MON</u>	P	TPBUG-A 待命
<u>IX REG'/REG</u>	4 × × × ×	IX 寄存器对访问，“4”代表 IX
<u>2 2 F 0</u>	4 2 2 F 0	输入转移目的地址
<u>MON'</u>	,	TPBUG-A 待命，进入上档键有效
<u>DISP/MEM</u>	F F E E	上档键 DISP 有效 计算结果为 EE。FF 指示负跳转（如为 00 则指示是正跳转，如为 FF 和 00 以外的数，则计算结果超出跳转指令允许范围）

按 键	显 示	说 明
<u>MON</u>	P	TPBUG-A 待命
<u>IY REG'/REG</u>	5 2 3 0 0	IY 访问，下档键有效，显示跳转指令地址 2300
<u>MON</u>	P	结束上面表的命令
<u>2 3 0 0 DISP/MEM</u>	2 3 0 0 × ×	2300存储单元访问，MEM下档命令有效，“××”为转移指令操作码
<u>LOAD/NEXT</u>	2 3 0 1 E E	NEXT 下档命令有效，显示操作数 EE 已写入

5~8. 2FB8, 2FBA, 2FBC, 2FBE (用户程序启动) 键四个

此四个按钮留给用户定义，用于启动四个用户程序。四个程序的地址应依次存于 RAM 中专门设置的用户程序启动地址表中 (2FB8H~2FBFH)。每个地址的低字节在前，高字节在后。四个地址值是由用户指定的。四个键不必按次序使用。

地址表的内容，可用键盘写入。如果用户程序已固存于 EPROM 中者，或转录于盒

式磁帶中，为方便操作，建议用户使用如下简单的引导程序，由其完成地址表的建立操作（UPADD1~UPADD4 分别为四个起始地址）。

```
LD    HL, UPADD1
LD    (2FB8H), HL
LD    HL, UPADD2
LD    (2FBAH), HL
LD    HL, UPADD3
LD    (2FBCH), HL
LD    HL, UPADD4
LD    (2FBEH), HL
HALT
```

此引导程序的起始地址可以是一个易于记忆的地址（例如 2000，或为 PROM1 插座的首地址 0800H）。

例：           UPADD1 = 2100           UPADD2 = 2300  
              UPADD3 = 2500           UPADD4 = 2700

在执行了前述引导程序建表之后，用户根据需要启动各用户程序的操作如表 2.13 所示。

表 2.13                            用户程序启动键的使用举例

按 键	显 示	說 明
<u>MON'</u>	'	TPBUG-A 待命
<u>2FBA MOVE</u>	根据程序安排显示	上档命令有效，开始执行第二号用户程序
<u>RESET</u>	P	用复位键停止程序
<u>MON'</u>	'	使上档命令有效
<u>2FBE BP</u>	根据程序安排显示	执行上档命令：执行第四号用户程序
<u>RESET</u>	P	停止程序
<u>MON'</u>		使上档命令有效
<u>2FB8 PROM</u>	根据程序安排显示	执行上档命令：执行第一号用户程序 ⋮

为方便操作，减少误操作的可能，对双功能键的布局作了精心的安排。用户在调试程序中，通常使用的按键都设置在下档，即用户按下 MON' 键的次数通常是很少的。

# 第三章 dT801—Z80单板计算机 的结构和原理

## 3.1 概 述

TP801—Z80 单板计算机的原理框图如图 3.1 所示。它有三条总线：数据总线 DB、地址总线 AB、控制（总）线CB。挂到这三条总线上的器件有下列三类：

1. Z80—CPU；
2. 存储器：有 ROM，PROM1，PROM2，RAM；
3. 接口电路：有 Z80—PIO，Z80—CTC，键盘和显示，录音机的接口；此外，尚有译码电路、时钟和复位电路等。详细的原理图请见附录一。

下面对电路的各个部分以及本机的一些功能作进一步的介绍。

## 3.2 时钟电路

时钟信号用来协调微型机内各部分的动作，使其有条不紊地进行操作。时钟电路示

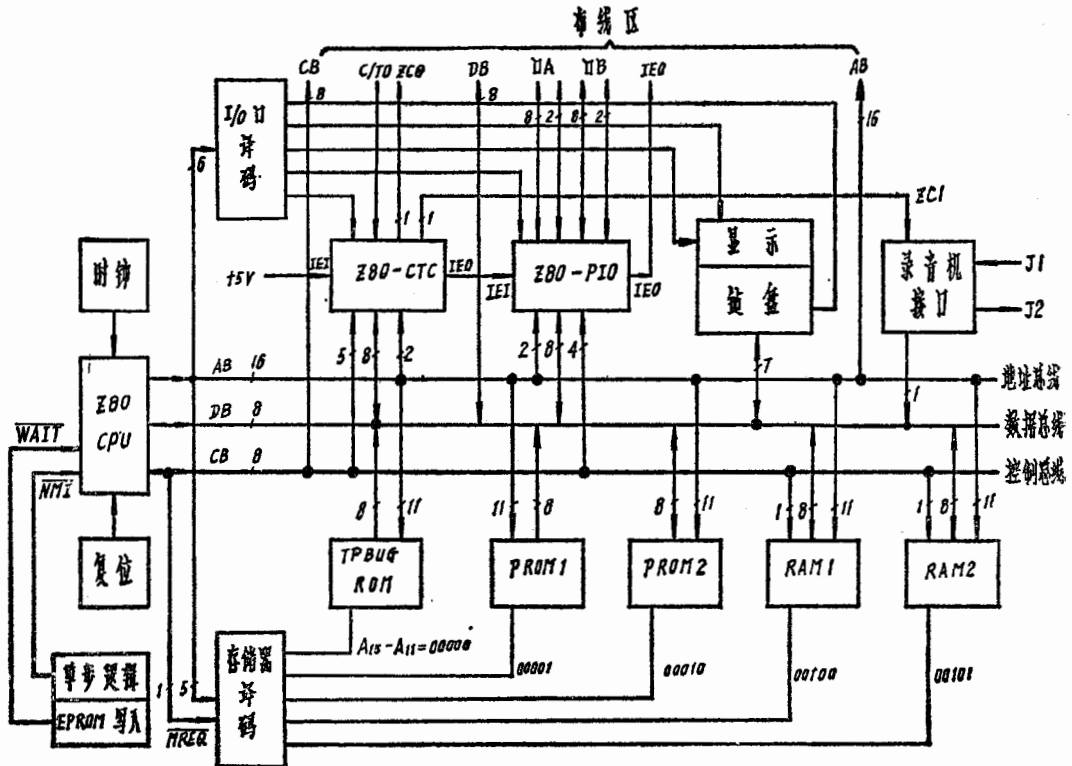


图 3.1 TP801 单板计算机原理图

于图 3.2。晶体振荡电路振荡的频率为 3.9936 MHz (兆赫)，这个数值恰好是堪萨斯标准音频磁带机接口所要求的 1200/2400Hz (赫) 和 300 波特 (Baud) 的整倍数。这个频率由 D 触发器 U30 分频而成为 1.9968MHz 的 CPU 时钟频率。CPU 时钟频率略低于 2 MHz，时钟周期接近于 500 ns，从而可以使用 8080A 的接口芯片以及廉价的存储器。这个 D 触发器的输出通过 74LS04 反相器接到 Z80—CPU，PIO 和 CTC。

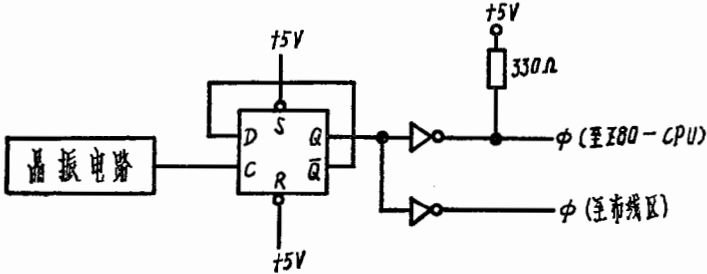


图 3.2 时钟电路

### 3.3 Z80—CPU

Z80—CPU 有一条 16 位的地址总线、一条 8 位的双向数据总线、一条 13 位的控制(总)线。13 个控制信号中本机使用 9 个，未使用  $\overline{\text{HALT}}$ 、 $\overline{\text{RFSH}}$ 、 $\overline{\text{BUSRQ}}$ 、 $\overline{\text{BUSAk}}$  信号 ( $\overline{\text{BUSRQ}}$  输入线接高电位，使其无效)。所有的总线均接向布线区和 S—100 总线插孔，以供用户附加电路时使用。关于 CPU 的详细介绍请参考《微处理器实用教材：基础知识和 Z80 器件的使用》及 Z80—CPU 技术手册。

### 3.4 存储器

#### 一、存储空间分配

存储空间的分配及片选信号见表 3.1。

TPBUG-A 监控程序安排在最下面的 2 K 字节 (其地址号为 0000—07 FFH)。

0800—0FFFH 为 2K 字节的 PROM<sub>1</sub>，其中可存放用户的应用程序。

1000—17FFH 为 2K 字节的 PROM<sub>2</sub>，如果用户的应用程序超过 2K 字节，则可继续存放在 PROM 2。此外，TPBUG-A 还可对该插座中的 EPROM 进行写入。

1800—1FFFH 2K 字节，没被使用。

2000—27FFH 为 2K 字节的 RAM<sub>1</sub>。其中 2400—27 FFH 的 1 K 字节为任选。

2800—2FFFH 为 2K 字节的 RAM<sub>2</sub>。其中 2800—2 BFFH 的 1 K 字节为任选。

RAM (即 RAM<sub>1</sub> 和 RAM<sub>2</sub>) 的存储分配如表 3.2 所示。

3000—37FFH 为 2K 字节，没被使用。

3800—3FFFH 为 2K 字节，没被使用。

#### 二、存储器译码

存储器的译码是由 U24 的 74LS138 八中取一译码器来完成，具体电路如图 3.3 所

表 3.1

TP 801 的 存 储 分 配

地 址	器 件	A15-A11	A10-A0	译码器的有效输出
3800—3FFFH	没 用	00111	可 变	$\overline{y7} = \overline{CS7}$
3000—37FFH	没 用	00110	可 变	$\overline{y6} = \overline{CS6}$
2800—2FFFH	2 K RAM 2 (U20-U 23)	00101	可 变	$\overline{y5} = \overline{CS5} = \overline{RAM2 SEL}$
2000—27FFH	2 K RAM 1 (U16-U 19)	00100	可 变	$\overline{y4} = \overline{CS4} = \overline{RAM1 SEL}$
1800—1FFFH	没 用	00011	可 变	$\overline{y3} = \overline{CS3}$
1000—17FFH	2 K PROM2 (U9)	00010	可 变	$\overline{y2} = \overline{CS2} = \overline{PROM2 SEL}$
0800—0FFFH	2 K PROM1 (U8)	00001	可 变	$\overline{y1} = \overline{CS1} = \overline{PROM1 SEL}$
0000—07FFH	2 K ROM (U7)	00000	可 变	$\overline{y0} = \overline{CS0} = \overline{MON SEL}$

表 3.2

RAM 的 存 储 分 配

地 址 空 间		用 途	字 节 数		插座编号
TPBUG	TPBUU-A		TPBUG	TPBUG-A	
2FFF 2FC0	2FFF 2FC0	TPBUG 和 TPBUG-A 使用的 RAM 暂存区和断点表	64	64	U 20—U 23 RAM 2
	2FBF 2FB8	TPBUG-A 四个用户程序入口 地址	0	8	
2FBF 2FA8	2FB 7 2FA 0	用户程序寄存器存放区, 用户 程序栈工作区	24	24	
2FA7 2F90	2F9F 2F88	监控程序工作区	24	24	
2F8F 2C00	2F87 2C00	RAM2 的用户程序工作区(1)	912	904	
2BFF 2800	2BFF 2800	RAM2 的用户程序工作区(2) —任选	1K	1K	
27FF 2400	27FF 2400	RAM1 的用户程序工作区(2) —任选	1K	1K	U 16—U 19 RAM 1
23FF 2000	23FF 2000	RAM1 的用户程序工作区(1)	1K	1K	

示。译码器的每根输出接至 2K 存储器的  $\overline{CE}$  端或  $\overline{CS}$  端, 故本机可配 16K 存储器。译码器规定 A15 和 A14 应为 0, 故 16K 存储器的地址空间为 0000—3FFFH, 占有下方的 16K 字节。

译码器的输出都接向 U27 的 16 个引线孔, 如图 I .1 所示。如果要改变各个 2K 存储器的地址空间, 则可以切断印刷线路板引线孔之间的铜箔连接, 按装 16 只引脚的插座, 其中插入具有相应连线的插头。

### 三、系统 RAM

U16—U19 为总计 2K 字节的 RAM (2114 型), U20—U23 为另外的 2K 字节 RAM。RAM 中部分单元用作 TPBUG-A 暂存区和栈区, 如表 3.2 所示。RAM 与 CPU 之间的连接如图 3.3 所示。

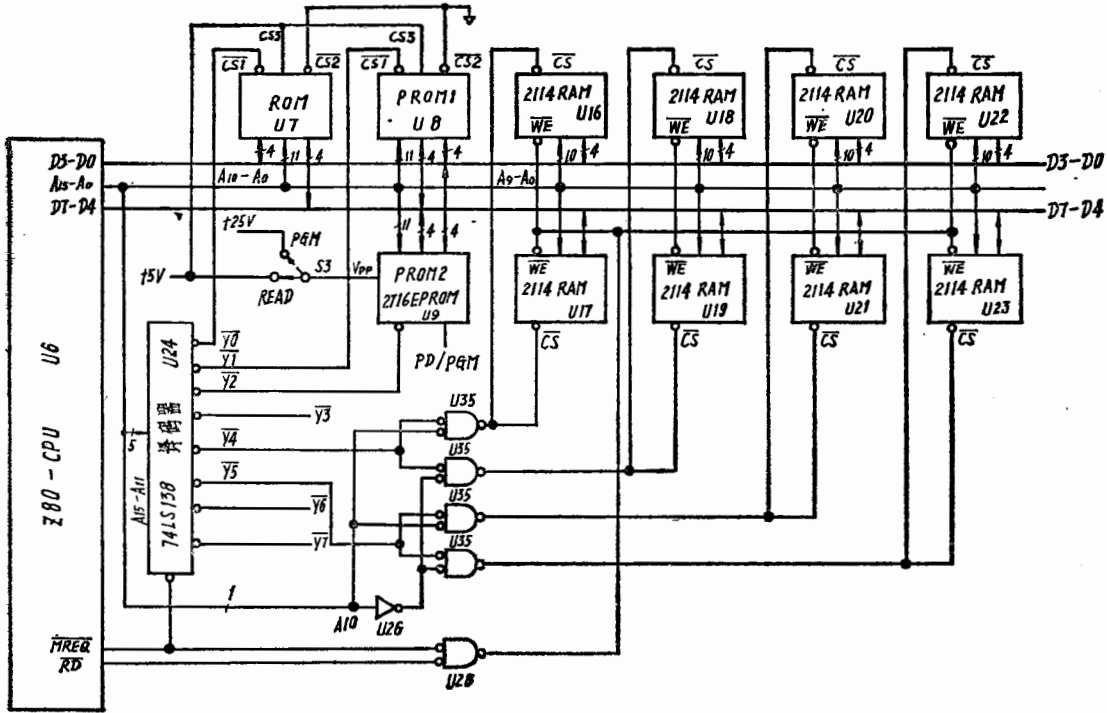


图 3.3 RAM 与 CPU 之间的连接

### 四、PROM1

当 RESET 信号为有效时, 从 TPBUG 的 0000H 单元开始执行程序, 在 00CDH 单元的指令检查开关 S2 的位置。如果 S2 置于 MON RST 位置, 则继续执行 TPBUG 监控程序, 在显示器上显示 “-” 或 “P”, 并扫描键盘的输入。如果 S2 置于 PROM 1 RST 位置, 则转移到起始地址为 0800H 的 PROM1 中的程序, 从而可以不必通过键盘输入命令而进入用户程序。

### 五、PROM2—EPROM

本节结合硬件和软件来简要介绍 EPROM 的写入过程。

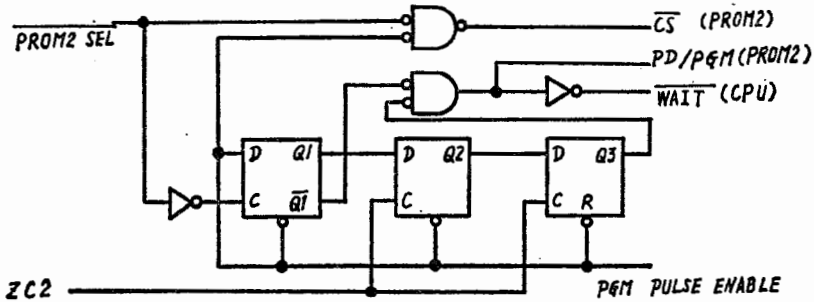
首先复习 2716/2758 型 EPROM 的操作方式, 如表 3.3 所示。为了对 EPROM 进行写入, 要求 S3 开关置于 PGM 位置, 即 EPROM 的 Vpp 引脚接至 +25V 电源。此外要求 CS = 1、PD/PGM 引脚输入一个 TTL 电平的脉宽为 50—55 毫秒的正脉冲。在写入时, 地址总线上应出现 EPROM 被写入单元的地址, 数据总线应出现被写入的内容。

EPROM 写入的电路如图 3.4(a) 所示, 操作的时间图如图 3.4(b) 所示。在写入前应做完一切准备工作, 即用紫外线擦除 EPROM 中的内容(即各单元内容都为全 1)。写入时, 应在未接入电源的情况下插入插座 U9, 再接上电源 +5V 和 25V, 将要写入的内

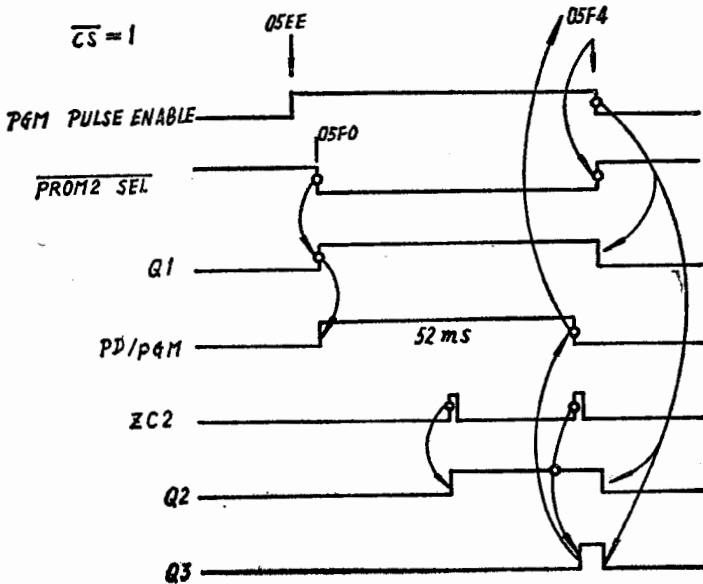
表 3.3

EPROM 的工作方式

工作方式	管脚	PD PGM	$\overline{CS}$	Vpp	数据綫状态
读		0	0	+5V	D <sub>out</sub>
禁止	止	0	1	+5V	高阻抗
待机	机	1	x	+5V	高阻抗
写入	入	┌ 52ms ─┘		+25V	D <sub>in</sub>
校验	核	0	0	+25V	D <sub>out</sub>
禁止	写入	0	1	+25V	高阻抗



(a) 电 路



(b) 间 时 图

图 3.4 EPROM 写入的电路和时间图

容输入起始地址为 2000H 的 RAM 单元，将需要写入的字节数用数字键送入，字节数用四个数字表示，显示在左端四个 LED 显示器上，也就是说，它们存放在 DISMEM—DS



MEM3 存储单元内。然后将 S3 开关置于 PGM 位置，压下 MON 键使显示“-”。最后压下 PROM PROG 键，进入该键的动作程序（起始地址为 05D3H）：

首先将 CTC 通道 2 设定为定时器工作方式，每隔 26 毫秒送出一个脉冲，但并不需要产生中断。其次执行 05EEH 单元的命令，使 PGM PULSE ENABLE 信号变高电平，从而撤除对 Q1—Q3 触发器的封锁。下一条为 05F0H 单元的 LDI 指令，开始对 EPROM 进行写入，地址总线上出现 1000H，数据总线上出现 2000H 单元的内容， $\overline{\text{PROM2 SEL}}$  变为低电平，Q1 变为高电平，PD/PGM 变为高电平， $\overline{\text{WAIT}}$  变为低电平（有效），从而使 CPU 进入等待状态。当 CTC 通道 2 的 ZC2 发出第二个脉冲（即 52 毫秒以后）时，Q3 变高，从而使 PD/PGM 变低， $\overline{\text{WAIT}}$  变高，CPU 结束等待状态。执行完 LDI 指令后，继续执行下一条指令（05F4H），使 PGM PULSE ENABLE 变低，从而使  $\overline{\text{PROM2 SEL}}$  变高、Q1—Q3 变低。这一单元的写入过程到此结束。写入下一单元的过程与此相同。

2758 与 2716 的写入过程基本相同，两者只有一只引脚不同，即 2758 的引脚 19 为 AR 而不是 2716 的 A10。通过跳接线（jumper）可将 AR 端接至 +5V，或接至地，或接至地址总线的 A10。

### 3.5 I/O 接口

#### 一、I/O 译码及空间分配

I/O 译码电路如图 3.5 所示。U36 单元的八中取一译码器将 I/O 口每四个为一组，每次选中一组，如表 3.4 所示。表中  $\overline{\text{PS5}}—\overline{\text{PS7}}$  接向布线区以供用户使用。对于 PIO 和 CTC，可以用 A1 和 A0 线将表中所示地址分配给有关单元。

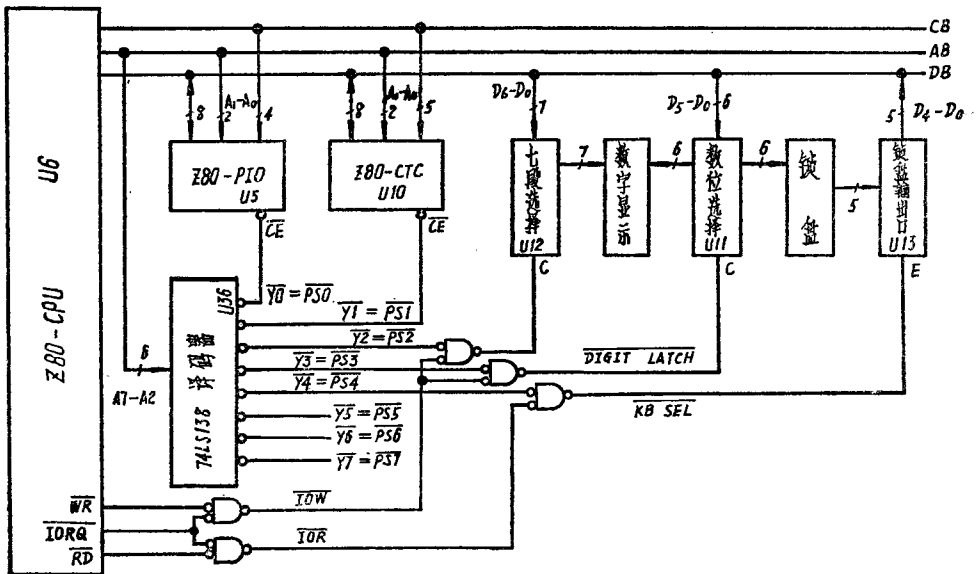


图 3.5 I/O 译码电路

表 3.4

A7—A2	译码器输出	器 件	A1 A0	口	口 地 址
100000	$\overline{Y_0} = \overline{PS_0} = \overline{PIO SEL}$	Z80—PIO U5	0 0	口A 数据寄存器	80 H
			0 1	口B 数据寄存器	81 H
			1 0	口A 控制寄存器	82 H
			1 1	口B 控制寄存器	83 H
100001	$\overline{Y_1} = \overline{PS_1} = \overline{CTC SEL}$	Z80—CTC U10	0 0	通道 0	84 H
			0 1	通道 1	85 H
			1 0	通道 2	86 H
			1 1	通道 3	87 H
100010	$\overline{Y_2} = \overline{PS_2} = \overline{SEG LH}$	74 LS273 U 12 八锁存器	× ×	七段选择 (只写)	88—8B H
100011	$\overline{Y_3} = \overline{PS_3} = \overline{DIG LH}$	74 LS273 U 11 八锁存器	× ×	数位选择 (只写)	8C—8 FH
100100	$\overline{Y_4} = \overline{PS_4} = \overline{KB SEL}$	74 LS244 U 13 八缓冲器	× ×	读键值 (只读)	90—93 H
100101	$\overline{Y_5} = \overline{PS_5}$	没使用			94—97 H
100110	$\overline{Y_6} = \overline{PS_6}$	没使用			98—9B H
100111	$\overline{Y_7} = \overline{PS_7}$	没使用			9C—9FH

## 二、Z80—PIO

Z80—PIO 是通用的并行 I/O 口芯片。它有两个 8 位的口。每个口有两根联络线。Z80—PIO 的两个口全部供用户使用，本机并不占用，因而这些线都接向布线区，可供用户附加电路。

每个口都可以使用联络线进行中断控制，并采用 IM2 中断方式。

## 三、Z80—CTC

Z80—CTC 有四个通道：

通道 0 一供用户使用，中断服务程序的起始地址为 2FD6H。用户可在 2FD6 处安排一个转移指令，转移到中断服务程序。

通道 1 一在本机中用于“CASS DUMP”键的动作程序，该程序将 RAM 中信息转储到盒式录音机的磁带中。该通道的中断服务程序的起始地址为 0732H。

通道 2 一在本机中用于 PROM PROG 键、SINGLE STEP 键和 MON 键的动作程序，

在 PROM PROG 键动作程序中，通道 2 仅用来产生 52ms 的脉冲。在 SINGLE STEP 键和 MON 键的动作程序中，通道 2 用来产生不可屏蔽中断。它们都没有使用 IM2 中断方式，因而也不需要中断矢量。

通道 3 — 在本机中用于“CASS LOAD”键的动作程序，该程序将盒式录音机磁带中的信息传送到 RAM 中。该通道的中断服务程序的起始地址为 079DH。

各个通道的中断服务程序的起始地址表放在 ROM<sub>1</sub> 中，如下表所示：

地址	内容	
→ 07F8	D 6	} 通道 0 中断服务程序起始地址 2FD6
07F9	2 F	
→ 07FA	3 2	} 通道 1 中断服务程序起始地址 0732
07FB	0 7	
→ 07FC	D D	} 通道 2 中断服务程序起始地址 2FDD*
07FD	2 F	
→ 07FE	9 D	} 通道 3 中断服务程序起始地址 079D
07FF	0 7	

CTC 四个通道的中断矢量按偶数依序排列，中断矢量低字节写入 CTC 通道 0，其格式如下：

D7							D0
1	1	1	1	1	×	×	0

此值为通道的号，由 CTC 自动填入

通道 0 的输入和输出都接向布线区以供用户使用。

#### 四、键盘和显示

本机有六个七段 LED 显示器，可以显示十六进制数，也可以显示其它的一些特定字符，在 TPBUG 监控程序中可显示“-”和“'”和“空”。

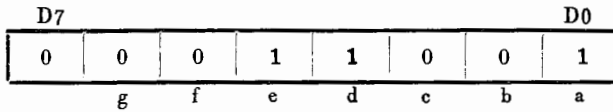
所显示的数据（或字符）写入 U12 的八锁存器 74LS273（口地址为 88H），至于要显示哪一位则由 U11 单元的八锁存器 74LS273（口地址为 8CH）来选择。

七段 LED 显示器的图形以及段号与数据位的对应关系如图 3.6 所示。LED 显示器最左边的一位命名为 DG0 位，其余的依次为 DG1, DG2, DG3, DG4, DG5。例如，要在最左边的 LED 显示器 DG0 位置上显示“4”，可以通过下列程序段来完成：

```
LD    A, 19H
OUT   (88), A
LD    A, 20H
OUT   (8C), A
```

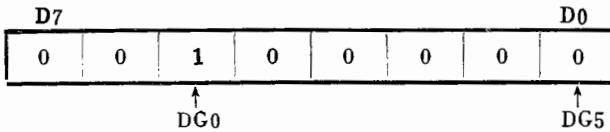
\*如果用户需要将通道 2 设定为 IM2 中断方式，则 (2FDD) — (2FDF) 将存放中断服务程序的第一条指令，通常是一条转移指令，从而这三个单元不能被 TPBUG 用作 (IF)，(PUFLR)，(RFLG)，用户应予注意。监控程序不要求通道 2 请求 IM2 方式的中断。

其中 19H 的二进制表示为:



D7 不起作用, 对于 D6—D0 来说 0 为有效, 故 b, c, f, g 段亮, 得“4”。

数 20H 的二进制表示:



D7, D6 不起作用, 对于 D5—D0 来说, 1 为有效, 故显示最左边的 DG0 位。

实际的显示过程并非如此简单。请参考 TPBUG 监控程序中 00F4—0120H 的 DISUP 程序段以及起始地址为 07A6H 的七段显示表 SEGPT。

晶体管 Q1—Q7 用来提高驱动能力, U1—U3 驱动器用来增加吸收电流的能力。U1 不仅扫描显示, 还能扫描键盘。例如, 如图 I.1 (见附录一) 所示, 当口 (8C) U11 中数为 08H (即 L3 线为低电平, 其余线为高电平) 时可检查 4, 5, 6, B, MON 键有否被压下, 如果键 6 被压下, 则 R2 线呈低电平, 其余线呈高电平, 即口 (90) 中低五位为 11011B。由 08H 和 11011B 两个数字可以求出键的偏移量。(功能键的键偏移量都大于等于 10H。例如 EXEC 键为 10H, SS 键为

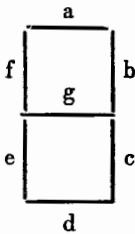


图 3.6 (a) 七段显示器

数字	D7	D6	D5	D4	D3	D2	D1	D0	字形
	g	f	e	d	c	b	a		
0	1	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	1	1
2	0	1	0	0	1	0	0	0	2
3	0	1	1	0	0	0	0	0	3
4	0	0	1	1	0	0	0	1	4
5	0	0	1	0	0	1	0	0	5
6	0	0	0	0	0	1	0	0	6
7	1	1	1	1	0	0	0	0	7
8	0	0	0	0	0	0	0	0	8
9	0	0	1	1	0	0	0	0	9
A	0	0	0	1	0	0	0	0	A
B	0	0	0	0	0	0	1	1	b
C	1	0	0	0	1	1	0	0	C
D	0	1	0	0	0	0	0	1	d
E	0	0	0	0	1	1	0	0	E
F	0	0	0	1	1	1	0	0	F
'	1	1	1	1	1	0	0	1	'
空	1	1	1	1	1	1	1	1	
-	0	1	1	1	1	1	1	1	-

(b) 七段显示器的代码与字形的关系

11H 等, 详见起始地址为 07B9H 的键值表 KYTBL。)

详细的键盘分析程序 DECKY 请参考附录三的 TPBUG。

## 五、录音机接口电路 (一) — 转储 (Dump)

将存放在 RAM 中易失的程序信息转储到廉价的盒式磁带中, 通常可以使用市场上出售的廉价的录音机和录音磁带。在转储过程中我们采用了“堪萨斯 (Kansas) 城标准”的记录技术。

用于盒式磁带上记录数据的格式必须遵守两个标准: 记录“1”和“0”的堪萨斯城标准和记录数据块的 Intel Hex Format。

堪萨斯城标准是 1975 年 9 月 7 日和 8 日在美国堪萨斯城由 BYTE 杂志主持的专题讨论会上制定的。这次专题讨论会的目的是, 为业余爱好者的录音磁带记录技术制订标准。堪萨斯城标准的要点是 (本机是符合这些规定的):

- 1) 八个频率为 2400Hz 的周波表示逻辑 1;
- 2) 四个频率为 1200Hz 的周波表示逻辑 0;
- 3) 一个记录字符的构成为: 一个逻辑 0 的起始位、7 或 8 个数据位、两个或两个以上的停止位 (本机使用一个七位的 ASCII 数据字符和一个停止位);
- 4) 七个 ASCII 数据位的次序为: 最低位在先, 最高位在后;
- 5) 在数据块之前有 30 秒以上的全 1 导引段, 之后有 5 秒的全 1 结尾段;
- 6) 数据的传送速率为 300 波特 (每位用 3.33ms);
- 7) 数据块内容不作规定。

因为堪萨斯城标准没有对所记录数据块的内容作出规定, 所以选用另一个标准——Intel Hex Format 一来规定数据块的结构。该标准的要点是:

- 1) 在数据块内的每个记录以冒号 (:) 开始, 以回车和换行符号结束;
- 2) 一切信息用 ASCII (7 位, 无奇偶校验) 表示;
- 3) 数据记录格式:

字节 1	冒号 (:) 分解符
字节 2 — 3	该记录中二进制的字节数, 最多为 16 个二进制字节 (即 32 个 ASCII 字节)。
字节 4 — 5	该记录起始地址的高位字节。
字节 6 — 7	该记录起始地址的低位字节。
字节 8 — 9	记录类型: ASCII “00”。
字节 10 — 11	除了分解符与回车和换行符以外的所有字节的“检查和”, 此“检查和”为各二进制字节和的负数。

回车和换行

- 4) “文件结束”记录格式:

字节 1	冒号 (:) 分解符
字节 2 — 3	ASCII “0”
字节 4 — 5	ASCII “0”
字节 6 — 7	ASCII “0”

字节 8—9

“01”记录类型；ASCII “01”

字节 10—11

“检查和”

用于转储的接口电路如图 I .1 下部所示。利用程序（详细内容请参考 TPBUG 中起始地址为 04CAH 的 CASS DUMP 键盘动作程序）设定 CTC 通道 1 为定时器工作方式，使 ZC1 产生 4800Hz 或 2400Hz 的脉冲，通过 U14 的分频，即可得到 2400Hz 或 1200Hz 的脉冲，再通过阻容（R33 和 C7）滤波，滤去其中的高频分量，即可接至录音机的 Auxiliary 或 MIC 输入端。

## 六、录音机接口电路（二）—输入（Load）

从 J1 端输入录音机的脉冲信息，U4 为限幅与整形电路，使 U5 能有一个不畸变的方波输入。J1 端输入的脉冲的峰—峰值应为 2V 左右，输入电平的高低可以用 U4 驱动 LED 显示器来指示。U14 和 U15 是频率检测器，用来鉴别 1200Hz 和 2400Hz。当 U7 输出 1200Hz 脉冲（“0”）时，U14 的  $\bar{Q}$  输出为 0，当 U7 输出 2400Hz 脉冲（“1”）时，U14 的  $\bar{Q}$  输出为 1。 $\bar{Q}$  的数据由 CPU 通过 U13 来读取，并拼装成一个 ASCII 字符，读入的 ASCII 字符再由程序转换成二进制数。

可见，在 TPBUG 的控制下，Z80—CPU 和 Z80—CTC 能实现 UART（通用异步接收发送器）的功能，以接收异步的串行数据，并形成并行的字，然后存入存储器。

## 8.6 其他部分

### 一、单步逻辑

当压下 SINGLE STEP 键后，进入该键的动作程序（起始地址为 0272H）。它将 CTC 通道 2 设定为定时器工作方式，并恢复现场，即从栈（用户寄存器存放区）中弹出各个寄存器的内容，然后执行用户的指令。在开始执行用户指令时，CTC 通道 2 的 ZC2 输出一个脉冲，这个脉冲通过 U33 和 U34 送向 CPU 的  $\overline{NM}$ I 输入端，因而在这一条用户指令执行完毕后，即返回 TPBUG，执行 0066H 单元的指令。此后程序为（起始地址为 01CBH）将关闭 CTC 通道 2，并保护现场。最后进入起始地址为 007EH 的程序段，显示 PC 和累加器 A 的内容。

### 二、中断电路

Z80 系列中 I/O 接口器件内均设有中断控制电路，从而与 Intel 8080/8085 系列不同，不必使用专用的中断控制器件。中断的优先级根据该芯片在电路中的位置来决定。本机中 CTC 具有较高的中断优先级，PIO 具有较低的中断优先级。PIO 芯片的 IE0 线接向布线区，供用户再增加 I/O 芯片时使用。

### 三、复位电路（Reset）

图 3.7 示出了本机所用的复位电路。有两种情况可以产生复位信号：一是上电复位，在刚接通电源时，由于电容 C17 的作用，使 RESET 短暂的保持低电平信号零；一是使用 RESET 按钮，在压下此按钮期间，RESET 保持低电平。

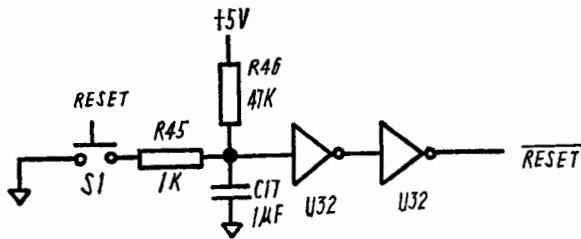


图 3.7 复位电路

#### 四、电压保护电路（任选）

本机使用 7805 集成电路。它的输入应大于 +7V。它的输出将稳定在 +5V。

#### 五、S—100 总线

本机可安装两个 S—100 总线插座。这种接口通常能与静态存储器和 I/O 扩充板兼容。如果接口需要 8080 所特有的控制信号，例如 SYNC，INTA，DBIN，POC，PWR，PRD 等，则尚需在布线区附加一些逻辑电路，以进行信号变换；并/或将附加的信号接到 S—100 总线。在印刷线路板上方有 +8V，±18V 电源的接线端，以便接入相应的电源。

#### 六、布线区

布线区可供用户附加 25—30 个集成电路，以便扩充存储器、增加 CRT 接口及其他电子实验电路。Z80—CPU，PIO，CTC（部分）的信号线以及译码出来的系统信号都引到布线区的附近，以使用户增加集成电路时使用这些信号。电源线和地线在印刷线路板的背面平行走线，使集成电路紧挨着这一阻抗的电源，从而减少用户电路的噪声干扰。

# 第四章 程序举例

本章目的是想通过几个程序的导入、执行和调试，使用户进一步熟悉键盘的操作功能和本机的使用方法，以及程序设计的一些基本知识。

## 4.1 熟悉键盘操作和 TPBUG-A 命令的使用

设有以下程序使用TPBUG-A命令，通过键盘操作，输入单板机\*。

```

                ORG      2000H
2000    3E  AA    LD  A, 0AAH; 将立即数 AA 送入 A 累加器
2002    06  BB    LD  B, 0BBH; 将立即数 BB 送入 B 寄存器
2004    78                LD  A, B    ; 寄存器 B 内容送入 A累加器
2005    0E  CC    LD  C, 0CCH; 将立即数 CC 送入 C 寄存器
2007    79                LD  A, C    ; 寄存器 C 内容送入 A累加器
2008    3E  AA    LD  A, 0AAH; 立即数 AA 送入 A累加器
200A    76                HALT        ; 暂停
    
```

使用 TPBUG-A 命令进行的操作如表 4.1:

表 4.1

按 键	显 示		说 明
	ADDRESS	DATA	
<u>RESET</u> (S1)	<b>P</b>		按下开关, S1—RESET, 立刻显示标志“P”
<u>2</u> <u>0</u> <u>0</u> <u>0</u>	2 0 0 0		输入程序的首地址2000
<u>MEM</u>	2 0 0 0	× ×	使用MEM键读出存储单元2000的内容××
<u>3</u> <u>E</u>	2 0 0 0	3 E	对2000存储单元写入操作码3E
<u>NEXT</u>	2 0 0 1	× ×	用NEXT键顺序对存储单元进行读出操作
<u>A</u> <u>A</u>	2 0 0 1	A A	对存储单元2001写入操作数AA
<u>NETX</u>	2 0 0 2	× ×	读出2002存储单元
<u>0</u> <u>6</u>	2 0 0 2	0 6	对存储单元2002写入操作码06

\* 本例以及下面各例中，所有用户按下的按键，都在它下面划一横线，而 TPBUG-A 的响应，在显示器上显示的就表示在方框中。数字如无特殊标记，均为十六进制数字。



按 鍵	显 示		說 明
	ADDRESS	DATA	
<u>NEXT</u>	2 0 0 3	× ×	讀出 2003 存儲單元
<u>B B</u>	2 0 0 3	b b	对存儲單元2003写入操作数 BB
<u>NEXT</u>	2 0 0 4	× ×	讀出存儲單元 2004
<u>7 8</u>	2 0 0 4	7 8	对存儲單元2004写入单字节指令 78
<u>NEXT</u>	2 0 0 5	× ×	讀出存儲單元 2005
<u>0 E</u>	2 0 0 5	0 E	对存儲單元2005写入操作碼 0E
<u>NEXT</u>	2 0 0 6	× ×	讀出存儲單元2006
<u>C C</u>	2 0 0 6	C C	对存儲單元2006写入操作数 CC
<u>NEXT</u>	2 0 0 7	× ×	讀出存儲單元2007
<u>7 9</u>	2 0 0 7	7 9	对存儲單元2007写入单字节指令 79
<u>NEXT</u>	2 0 0 8	× ×	讀出存儲單元2008
<u>3 E</u>	2 0 0 8	3 E	对存儲單元2008写入操作碼 3E
<u>NEXT</u>	2 0 0 9	× ×	讀出存儲單元2009
<u>A A</u>	2 0 0 9	A A	对存儲單元2009写入操作数 AA
<u>NEXT</u>	2 0 0 A	× ×	讀出存儲單元 200A
<u>7 6</u>	2 0 0 A	7 9	对存儲單元 200A 写入单字节暂停指令 76
<u>MON</u>	P		用 MON 鍵中止对存儲單元的讀、写操作，以便进行其他操作
<u>A</u>	A		讀出 A 累加器

续表 4.1

按 鍵	显 示		說 明
	ADDRESS	DATA	
<u>REG</u>	A	× ×	在程序未执行前, A 的内容是随机数
<u>MON</u>	P		
<u>B</u>	b		讀出 B 寄存器
<u>REG</u>	b	× ×	程序未执行, B 寄存器的内容是随机数
<u>C</u>	C		讀出 C 寄存器
<u>REG</u>	C	× ×	程序未执行, C 寄存器的内容是随机数
<u>MON</u>	P		中止上述操作。下面用 STEP 鍵, 采用单步方式执行 ORG 2000 程序
<u>PC(1)</u>	1		
<u>REG</u>	1      ×   ×	× ×	讀出程序計数器 PC
<u>2</u> <u>0</u> <u>0</u> <u>0</u>	1      2   0	× ×	单步执行程序方式, 首先要修改程序計数器 PC 内容, 使它指向程序首地址
<u>STEP</u>	2   0   0   2	A A	执行完 2000 指令, 立即数 AA 送入累加器 A, PC 指向下一条指令的地址。“DATA”显示器显示 A 的内容“AA”
<u>STEP</u>	2   0   0   4	A A	2002 指令执行完毕, 立即数 BB 送入 B 寄存器, PC 指向下一条指令地址。这时累加器 A 内容不变, “DATA”仍然显示“AA”
<u>MON</u>	P		
<u>B</u>	b		
<u>REG</u>	b	b b	讀出 B 寄存器的内容, 验证指令的执行
<u>STEP</u>	2   0   0   5	b b	B 寄存器内容送入 A 累加器, PC 指向下一条指令的地址

按 鍵	显 示		說 明
	ADDRESS	DATA	
<u>STEP</u>	2 0 0 7	b b	立即数 CC 送入 C 寄存器, PC 指向下一条指令的地址。A 累加器的内容不变
<u>STEP</u>	2 0 0 8	C C	C 寄存器的内容送入 A 累加器, PC 指向下一条指令的地址
<u>STEP</u>	2 0 0 A	A A	
<u>MON</u>	P		中止 STEP 操作方式
<u>A REG</u>	A	A A	先按下 A, 后按 REG 键, 读出 A 累加器内容
<u>MON</u>	P		
<u>B REG</u>	b	b b	读出 B 寄存器内容
<u>MON</u>	P		
<u>C REG</u>	C	C C	读出 C 寄存器内容
<u>MON</u>	P		
<u>PC REG</u>	1 2 0	0 A	读出 PC 的内容, 它现在指向 200A 存储单元, 是存储暂停指令的地址
<u>MON</u>	P		中止读出方式, 下面开始设置多个断点操作, 如在 2002、2005、2008、200A 设置断点
<u>2 0 0 2</u>	2 0 0 2		先送入四位数码表示断点单元地址
<u>B P</u>	2 0 0 2		按下 BP 键, 断点被接受
<u>MON</u>	P		
<u>2 0 0 5</u> <u>B P</u>	2 0 0 5		先按下四位地址数码键, 后按 BP 键, 在 2005 单元设置断点

续表 4.1

按 键	显 示 ADDRESS	DATA	说 明
<u>MON</u>	P		
<u>2 0 0 8</u> BP	2 0 0 8		在 2008 单元设置断点
<u>MON</u>	P		
<u>2 0 0 A</u> BP	2 0 0 A		在 200A 单元设置断点
<u>MON</u>	P		中止设置断点的操作, 下面用 EXEC 键执行程序
<u>2 0 0 0</u>	2 0 0 0		送入程序首地址 2000, 按 EXEC 键, 执行程序
<u>EXEC</u>	2 0 0 2	A A	程序从 2000 单元开始执行, 显示第一个断点地址和执行第一条指令后, A 的内容 AA
<u>EXEC</u>	2 0 0 5	b b	显示第二个断点地址和执行此断点前面的指令后, A 的内容 "bb"
<u>EXEC</u>	2 0 0 8	C C	显示第三个断点地址以及完成此断点前面的指令后, A 的内容 "CC"
<u>EXEC</u>	2 0 0 A	A A	显示第四个断点, 以及完成此断点前面的指令后, A 的内容 "AA"
<u>RESET</u>	P		按动 RESET 或 STEP 或送入三个以下的数字键后按下 BP 键, 均可取消断点
<u>2 0 0 0</u>	2 0 0 0		输入程序首地址
<u>EXEC</u>			从程序首地址开始执行程序, 一直执行到暂停指令。显示器不显示
<u>MON</u>	P		按下 MON 键, 回到 TPBUG-A 控制, 显示标志 "P"
<u>PC</u> <u>REC</u>	1 2 0	0 b	检查 PC, 它指向存储 HALT 指令的下一个存储单元的地址
<u>MON</u>	P		重新回到 TPBUG-A 控制, 可相应检查各寄存器内容, 与上述执行程序方式相比较

## 4.2 相对转移指令中偏移量的计算

本例的内容是计算相对转移指令的偏移量。偏移量是以十六进制表示的带符号的 2 的补码。TPBUG-A 具有自动计算并填写相对偏移量的功能。相对转移源地址装入 IY 寄存器对，目的地地址装入 HL 寄存器对，偏移量则装入以 (IY + 1) 内容为地址的存储单元，在 TPBUG-A 中，计算并自动填入偏移量的子程序入口地址是 00C0H。但要求转移源地址装入 DE，目的地地址装入 HL，详细框图及程序注释如下示：

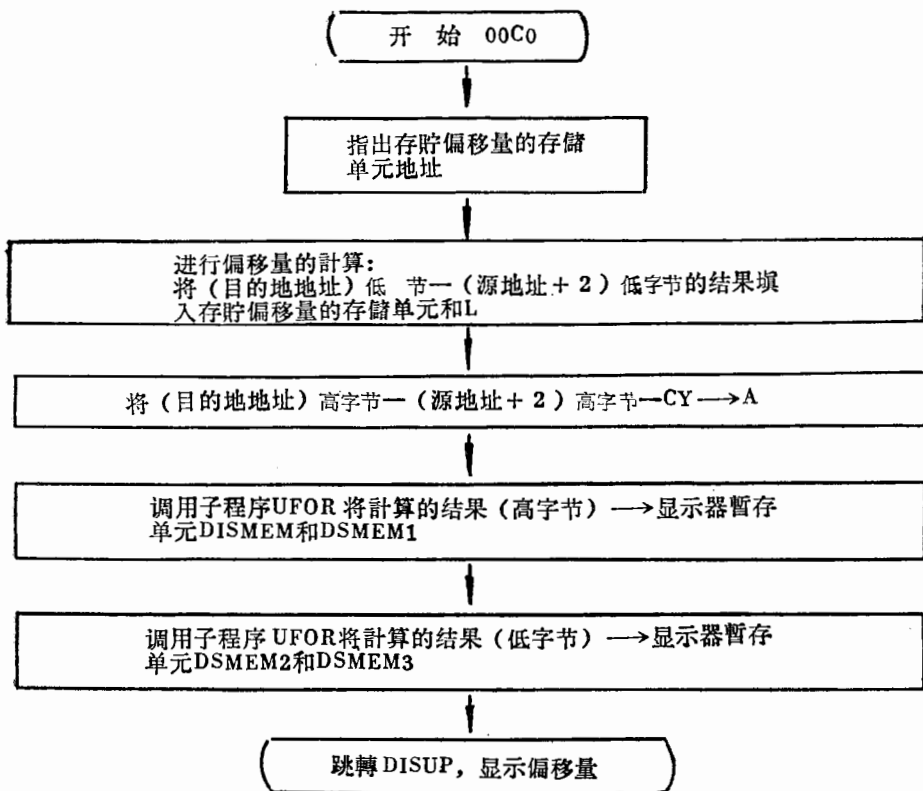
；子程序

```

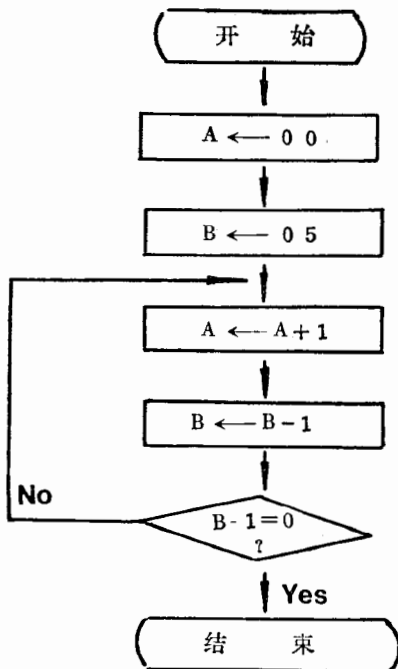
                                ORG  00C0H
00C0    1813    JP    RESTR3-S
00D5    13    RESTR3: INC  DE    ; 指出填写偏移量所在存储单元地址
00D6    D5     PUSH DE    ; 将贮存偏移量的存储单元地址转入 IX
00D7    DDE1   POP  IX     ; IX←-DE
00D9    13     INC  DE
00DA    7D     LD   A, L
00DB    93     SUB  E      ; A←-(目的地地址)低字节
                                - (源地址 + 2)低字节
00DC    6F     LD   L, A
00DD    DD7700 LD (IX + 0), A ; 将计算结果(低字节)填入贮存偏移量的
                                存储单元
00E0    7C     LD   A, H
00E1    9A     SBC  A, D    ; A←-(目的地地址)高字节
                                - (源地址 + 2)高字节 - CY
00E2    DD21F72F LD IX, DISMEM
00E6    CD3C06  CALL UFOR1 ; 利用 TPBUG 的子程序 UFOR1, 将计算
                                结果(高字节)写入显示暂存单元 2FF7
                                (DISMEM)和2FF8(DSMEM1)
00E9    DD21F92F LD IX, DSMEM2
00ED    7D     LD   A, L
00EE    CD3C06  CALL UFOR1 ; 调用子程序 UFOR1, 将计算结果(低字
                                节)写入显示暂存单元2FF9(DSMEM2)
                                和2FFA(DSMEM3)
00E1    C3F400  JP   DISUP ; 调用 TPBUG 子程序 DISUP 显示计算
                                结果(四位数字)

```

## 计算相对转移指令偏移量的子程序流程图



## 主程序流程图



通过键盘操作，输入下面程序

```

                                ORG 2000H
2000  3E 00                      LD A,00
2002  06 05                      LD B,05H
2004  3C                          LOOP: INC A
2005  10 ?                        DJ NZ,LOOP-$
2007  76                          HALT
    
```

表 4.2

按 键	显 示		说 明
	ADDRESS	DATA	
<u>RESET</u>	P		等待键盘命令
<u>2 0 0 0</u> <u>MEM</u>	2 0 0 0	× ×	输入程序
<u>3 E</u>	2 0 0 0	3 E	
<u>NEXT 0 0</u>	2 0 0 1	0 0	
<u>NEXT 0 6</u>	2 0 0 2	0 6	
<u>NEXT 0 5</u>	2 0 0 3	0 5	
<u>NEXT 3 C</u>	2 0 0 4	3 C	
<u>NEXT 1 0</u>	2 0 0 5	1 0	
<u>NEXT</u>	2 0 0 6	× ×	2006 单元不装入信息，这是准备填入计算的结果——相对转移偏移量
<u>NEXT 7 6</u>	2 0 0 7	7 6	
<u>MON</u>	P		中止输入程序的操作，准备进行计算偏移量的操作
<u>IX(4) REG</u>	4 × ×	× ×	目的地地址→ IX
<u>2 0 0 4</u>	4 2 0	0 4	
<u>MON</u>	P		

按 鍵	显 示 ADDRESS DATA	說 明
<u>IY(5) REG</u>	5    ×   × ×   ×	源地址→IY 1
<u>2 0 0 5</u>	5    2   0 0   5	
<u>MON/</u>	/	上档键有效, 显示上档标志“/”
<u>DISP</u>	F   F F   d	显示偏移量FDH, 即-3H的2的补码, 因Z80指令系统规定相对偏移量为两字节(即从+127D~-128D), 所以实际偏移量为FDH, FF表示跳转方向
<u>MON</u>	P	
<u>2 0 0 6</u> <u>MEM</u>	2   0   0   6 F   d	检查2006单元已自动填入偏移量FD
<u>MON</u>	P	下面用STEP方式执行程序
<u>PC REG</u>	1    ×   × ×   ×	修改程序计数器
<u>2 0 0 0</u>	1    2   0 0   0	将PC指向程序的首地址
<u>STEP</u>	2   0   0   2 0   0	执行完第一条指令, 显示A的内容和下一条指令的地址
<u>STEP</u>	2   0   0   4 0   0	
<u>STEP</u>	2   0   0   5 0   1	第一次执行“INCA”指令, A第一次增量
<u>STEP</u>	2   0   0   4 0   1	循环回去, 这时若检查B寄存器, B的内容是多少?
<u>STEP</u>	2   0   0   5 0   2	累加器A第二次增量
<u>STEP</u>	2   0   0   4 0   2	循环回去
<u>STEP</u>	2   0   0   3 0   3	A第三次增量
<u>STEP</u>	2   0   0   4 0   3	循环回去
<u>STEP</u>	2   0   0   5 0   4	A第四次增量



按 键	显 示 ADDRESS DATA	說 明
<u>STEP</u>	2 0 0 4      0 4	循环回去
<u>STEP</u>	2 0 0 5      0 5	A 第五次增量。这时若检查 B 的内容，它是 01
<u>STEP</u>	2 0 0 7      0 5	执行条件转移指令“DJNZ”，当条件 B-1=0 成立时，即依次执行下一条指令，而不作转移
<u>MON</u>	P	等待键盘指令

### 4.3 软件延时

下面这个程序是使字符“8”，从显示器的右端向左端不停地循环移动。在每个位置上停留的时间，可以由改变装入 B 寄存器的时间常数来决定，约由 20ms 到 5.2sec。

；程序

```

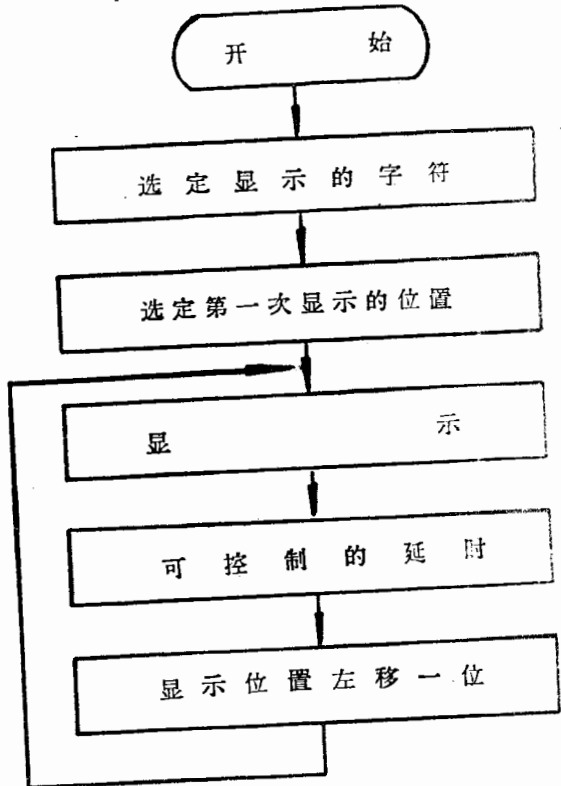
                                ORG 2000H
2000  3E00      LD A,00
2002  D388      OUT(88H),A      ; 选定字符“8”
2004  3E01      LD A,01H        ; 选定右边第一个显示器亮
2006  D38C      LOOP: OUT(8CH),A
2008  06FF      LDB,0FFH        ; B←延时时间常数
200A  CD4F06    LOOP1: CALL D20MS ; 调用 TPBUG中延时约 20ms 子程序
200D  10FB      DJ NZ,LOOP1-$ ; 最长延时约5.2sec
200F  07        RLCA           ; 左移一位
2010  18F4      JR LOOP-$      ; 循环回去

```

首先输入程序，按下 MON，输入程序首地址 2000H，按下 EXEC，字符“8”则在显示器上从右到左循环移动。按下 MON，程序则停止执行。控制延时子程序 D20MS 的调用次数（也就是控制装入 B 寄存器的时间常数）就可以控制字符在显示器上的停留时间。

1

## 程 序 流 程 图



### 4.4 Z80-CTC 的应用

		ORG	2000H	
2000	3E21	LD	A, 21H	
2002	ED47	LD	I, A	； 设定中断矢量高字节 I←21H-
2004	310023	LD	SP, 2300H	； 建立栈指示器
2007	3E00	LD	A, 00	
2009	D384	OUT	(84H), A	； 外部设备提供中断矢量低字节
200B	3EA5	LD	A, 0A5H	
200D	D384	OUT	(84H), A	； 设定 CTC 的工作方式, 输入通道控制字
200F	3EFF	LD	A, 0FFH	
2011	D384	OUT	(84H),	； 输入时间常数, 延时约 33ms
2013	3E01	LD	A, 01H	； 设定第一次显示 01
2015	ED5E	IM2		； 设定中断方式 2
2017	FB	LOOP;	EI	
2018	76	HALT		
2019	C31720	JP	LOOP	

； 中断服务程序入口地址表

2100 00 22

； 中断服务程序

2200 FB EI ； 开中断

2201 07 RLC A ； A 累加器左移一位

2202 ED 4D RETI ； 中断返回

本例程序是使 Z80-CTC 的 0 通道工作在定时器方式, 经过一个整定延迟时间后 (本例整定延迟时间约 33ms), 发出一个中断请求。改变输入到 CTC 的时间常数, 可以改变请求中断的时间。

输入程序, 按以下步骤进行调试:

1. 采用 STEP 工作方式, 首先修改程序计数器 PC 为 2000H。以稍慢的节奏, 顺序步进执行程序到显示地址为 2018H 时, 稍停一会。

2. 继续按动 STEP 键, 注意观察显示器显示 A 累加器的内容。每执行完一次 2201H 指令, A 的内容便左移一位, 显示器按 01, 02, 04, 08, 10, 20, 40, 80 的次序循环变化。

3. 当 2201H 地址显示时, 停止按 STEP。按下 MON 键, 检查栈指示器 SP 的内容, 这时应为 22FEH。这是因为从栈顶地址 2300H 压入了两字节。检查 22FEH 和 22FFH 单元的内容, 应该是 18H 和 20H, 这是中断返回的地址 2018H。

4. 按下 MON 键, 然后再按 STEP 键, 直至显示 2018H。这时, 再检查栈指示器 SP 内容, 应该是 2300H。这表明已经执行了中断返回 RETI 指令, 中断返回地址 2018H 已从栈中弹出。

5. 用 BP 键, 在 2200H 设置一个断点, 用 EXEC 键连续执行程序。先输入首地址 2000H, 按下 EXEC 键, 立刻显示断点地址 2200H 和这时 A 累加器的内容。连续不断按动 EXEC, A 累加器内容不断变化, 而且再按一次, 改变显示一次, 按 01, 02, 04, 08, 10, 20, 40, 80 的顺序改变。

可利用 RESET 键使 CTC 复位。

下面这个程序, 是将本例稍加修改后, 再与 4.3 例结合起来的一个简单表演程序。

输入程序, 按下 RESET 键, 输入程序首地址 2000H, 按下 EXEC 键, 字符“8”快速从显示器右端移动到左端, 经过十次循环之后, 立刻变为较慢的一次移动, 接着又是十次较快的循环移动, 又接着一次慢速移动……, 这样不停地在显示器上循环移动。根据需要, 可以很方便地改变显示的字符和循环的速度。按下 RESET 键, 移动立刻停止回到 TPBUG-A 控制。

； 程序

```
                ORG    2000H
2000    3 E 21    LD     A, 21 H
2002    ED 47    LD     I, A      ; 设定中断矢量高字节
2004    3 E 00    LD     A, 00
2006    D 384    OUT    (84H), A  ; 外部设备提供中断矢量低字节
```

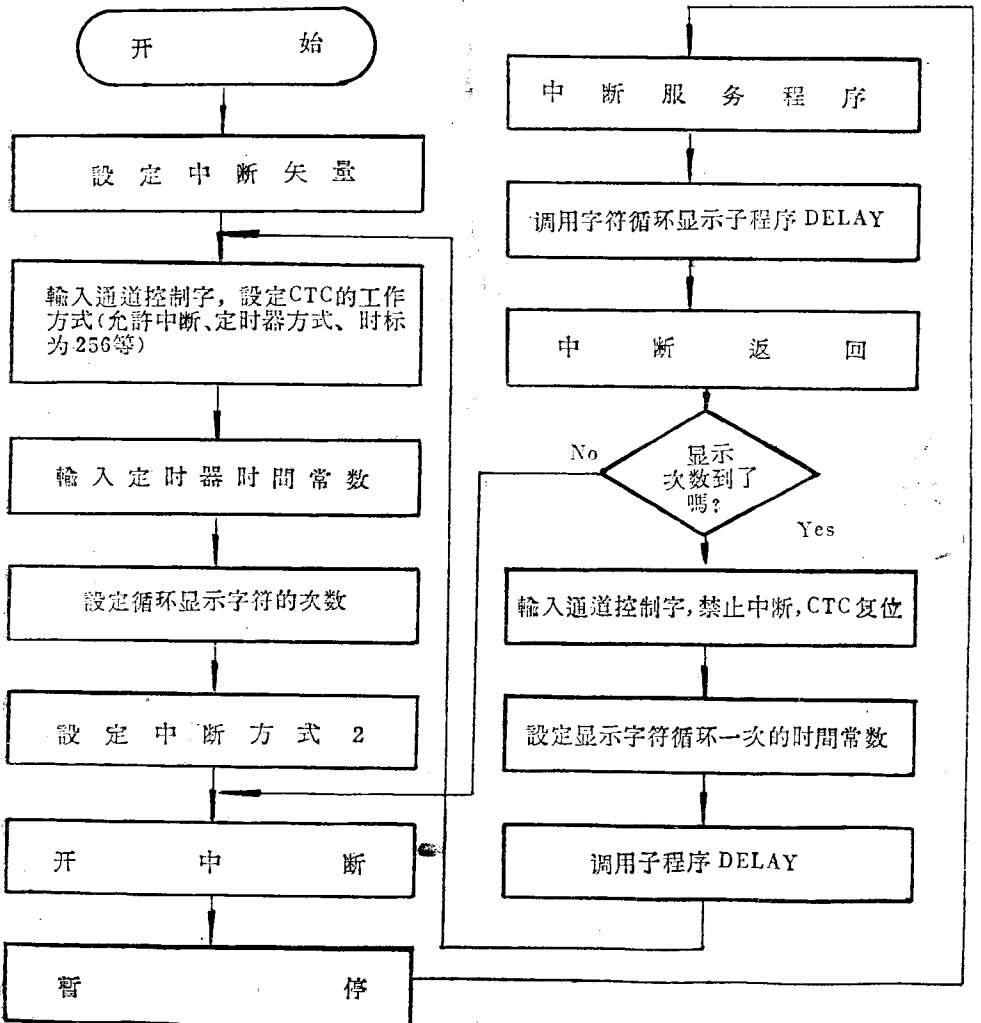
2008	3EA5	ST, LD A,0A5H	
200A	D384	OUT (84H),A	;输入通道控制字, 设定 CTC 工作方式
200C	3EFF	LD A,0FFH	
200E	D384	OUT (84H),A	;输入时间常数
2010	0EF6	LD C,0F6H	;设定快速循环次数(10D 次)
2012	ED5E	IM2	;设定中断方式 2
2014	FB	LOOP, EI	;开中断
2015	76	HALT	
2016	0C	INC C	
2017	20FB	JR NZ,LOOP-\$	;C≠0 循环返回
2019	3E03	LD A,03H	
201B	D384	OUT (84H),A	;禁止通道中断
201D	1E5F	LD E,5FH	;设定字符循环一次的时间常数
201F	CD5020	CALL DELAY	;调用子程序
2022	C30820	JP ST	;循环返回, 重新工作
			; 中断服务程序入口地址表
2100	0022		
			; 中断服务程序
2200	1E0A	LD E,0AH	;设定字符循环速度的时间常数
2202	CD5020	CALL DELAY	
2205	ED4D	RETI	
			; 子程序
2050	3E00	DELAY, LD A,00	;选定字符“8”
2052	D388	OUT (88H),A	
2054	3E01	LD A,01H	
2056	D38C	LOOP1, OUT (8CH),A	
2058	43	LD B,E	
2059	CD4F06	LOOP2, CALL D20MS	
205C	10FB	DJ NZ,LOOP2-\$	
205E	CB6F	BIT 5,A	;测试 A 的第 5 位
2060	2003	JR NZ,LOOP3-\$	
2062	07	RLCA	
2063	18F1	JR LOOP1-\$	
2065	C9	LOOP3, RET	

注: (1) 这个 [DELAY] 子程序, 是将 4.3 例稍加修改而成。这种设计方法, 不仅易于调试和检查, 而且也节省存储单元。

(2) 若想显示别的字符, 可参考下表, 修改 DELAY 子程序的第一条指令 LD A, 00H 中的立即数 (即修改 2051 存储单元的内容)。

显示的 字符	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
立即数	40	79	24	30	19	12	02	78	00	18	08	03	46	21	06	0E

### 程 序 流 程 图



## 4.5 Z80-PIO 的应用

首先用一个 10 K $\Omega$  电阻, 将 Z80-PIO 的  $\overline{\text{ASTB}}$  脚连接到电源 +5 V 端, 然后输入下列程序, 并用 BP 键在 2200 H 单元设置断点。

输入程序首地址 2000 H, 按下 EXEC 键, CPU 执行程序到暂停指令 HALT, 显示器无显示。

取一导线, 使  $\overline{\text{ASTB}}$  脚碰一下地 (GND), 相当于输入一负脉冲, 此脉冲的上升沿使 PIO 发出一中断请求 ( $\overline{\text{INT}}$ ), CPU 将执行中断服务程序, 把由 BP 键设置的断点 2200H 显示出来。

；程序

```
                                ORG 2000 H
2000    3E 21                    LD  A,21 H
2002    ED 47                    LD  I,A      ; 设定中断矢量高字节
2004    3E 00                    LD  A,00
2006    D3 82                    OUT (82H),A ; 外部设备提供中断矢
                                量低字节
2008    3E 4F                    LD  A,4 FH
200A    D3 82                    OUT (82H),A ; 对 PIO 的 A 口设置
                                方式 1
200C    3E 87                    LD  A,87 H
200E    D3 82                    OUT (82H),A ; 设置允许中断控制字
2010    ED 5E                    IM2      ; 中断方式 2
2012    FB                        EI
2013    18 FE                    LOOP; JR  LOOP
； 中断服务程序入口地址表
2100    00 22
； 中断服务程序
2200    FB                        EI
2201    ED 4D                    RETI      ; 中断返回
```

可以将 4.4 例的中断服务程序的 DELAY 子程序作为本例的中断服务程序。这时, 输入程序之后, 按下 RESET 键, 输入程序首地址 2000H, 按下 EXEC 键, 显示器无显示, 将  $\overline{\text{ASTB}}$  脚碰一下地, 字符“8”立刻从显示器右端移往左端, 最后停留在左端不动。再碰一次, 再重复一次上述现象。若将  $\overline{\text{ASTB}}$  脚碰一下地的操作得不好时, 字符“8”会循环移动两次才停留在左端不动。这时, 可以用一个 10K $\Omega$  电阻把  $\overline{\text{ASTB}}$  脚与地连接, 然后输入程序, 用 EXEC 键执行程序, 然后通过一个 3 K $\Omega$  电阻碰一下 +5 V 端, 字符立即从显示器右端移到左端并停留在左端不动。再碰一次, 再重复上述现象一

次。若条件许可，用 TTL 电平的单脉冲触发就更好些。

## 4.6 EPROM 的写入程序

本机可以将用户所需要的程序写入到 EPROM (1K 字节的 2758 或 2K 字节的 2716) 中。在前面按键操作说明中已讲到，可以使用 PROM 键，将从 2000H 单元开始的程序复制到从 1000H 单元开始的 EPROM 中去。复制的字节数，由按键盘输入的 4 位十六进制数字表示。这个操作虽然简便，但源程序和目的地程序的首地址均被限定在 2000H 和 1000H，有时使用就不方便。下列两个应用程序的例子，用户可以将任意地址存储器数据块复制到 EPROM 中去。

这两个应用程序，如果是经常使用的话，可以事先录制在盒式录音机的磁带上。需要时，从本系统盒式磁带机接口 J1 输入 RAM 中；或者将程序写在 EPROM 里，而将这个写有程序的 EPROM 插在 EPROM1 插座上，供在线使用。

在第一个应用程序中，使用了 Z80 数据块传送指令。它可以用来将贮存在存储器中的数据块（源数据）传送到任意地址的存储单元（目的地数据）；它也可以将插在 PROM1 或 PROM2 插座上的 EPROM 数据复制到 RAM 中去。

使用这个程序之前，先用 MON 和 REG 键对 Z80 寄存器进行予置数据：

HL——源数据地址，高字节在 H；

DE——目的地数据地址，高字节在 D；

BC——数据传送的总字节数，高字节在 B。

然后执行下列数据块传送程序：

```
2050  EDB0          LDIR          ; 数据块自动传送
2052  C39C00       JP RESTAR    ; 传送完毕，回到 TPBUG
```

第二个例子，是用来对 EPROM 进行编程用的应用程序。可以将贮存在 RAM(本系统原有的或用户扩充的 RAM)、ROM、或者插在 EPROM1 插座上的 EPROM 中的数据复制到插在 PROM2 插座上的 EPROM 中去。

每次使用本程序之前，先按下 RESET 键，然后修改下述存储单元内容；

2FC0H—源数据地址的高字节；

2FC1H—源数据地址的低字节；

2FC2H—目的地数据地址的高字节；

2FC3H—目的地数据地址的低字节；

2FC4H—要复制的字节数的高位；

2FC5H—要复制的字节数的低位。

通过键盘输入下述程序，或通过 J1 口由盒式磁带机将已录制好的本例程序信息输入单板机。需要复制到 EPROM 中去的源数据（存于 RAM、ROM 或 EPROM）也应校核准确。按下 MON，显示“P”标志，由键盘输入首地址 2000H，用 EXEC 键执行下述程序，将源数据写入 EPROM，这时显示器中标志消失。过一段时间（这时间长短由

要复制的字节数决定，如复制 2K 字节，大约需 3—4 分钟，千万要有耐心等待！），若一切正常即重新显示标志“P”，表示复制完毕。若显示器显示数字，这表示复制过程有错，显示出出错地址和内容，按下 NEXT 键，逐个检查出错单元，直至出现“P”标志。详细内容请参阅第二章有关 EPROM 写入部分内容。

程序；

```

                                ORG    2000 H
2000    3E01    PROG; LD    A, 01
2002    32DA2F    LD    (PRFEL), A    ; 置标志
2005    3AC02F    LD    A, (2FC0H)
2008    67      LD    H, A
2009    3AC12F    LD    A, (2FC1H)
200C    6F      LD    L, A    ; 将源数据首地址→HL
200D    3AC22F    LD    A, (2FC2H)
2010    57      LD    D, A
2011    3AC32F    LD    A, (2FC3H)
2014    5F      LD    E, A    ; 将目的地址首地址→DE
2015    3AC42F    LD    A, (2FC4H)
2018    47      LD    B, A
2019    3AC25F    LD    A, (2FC5H)
201C    4F      LD    C, A    ; 将字节数→BC
201D    D5      PUSH DE    ; 保存目的地数据首地址
201E    E5      PUSH HL    ; 保存源数据首地址
201F    C5      PUSH BC    ; 保存字数以备校核用
2020    3E25    PROG1; LD    A, 25 H
2022    D386    OUT    (86H), A    ; 输入控制字，设置 CTC 为定时器工作
2024    3ECB    LD    A, 0CBH
2026    D386    OUT    (86H), A    ; 时间常数，约 26ms
2028    3E80    LD    A, 80H
202A    D38C    OUT    (DIGLH), A    ; 发出 PGM·PULSE·EN = 1
202C    EDA0    LDI    ; 插入等待信号 WAIT
202E    3E00    LD    A, 0
2030    D38C    OUT    (DIGLH), A    ; 使 PGME·PULSE·EN = 0
2032    3E03    LD    A, 03
2034    D386    OUT    (86H), A    ; CTC 复位
2036    EA2020    JP    PE, PROG1    ; 若 BC - 1 ≠ 0 则返回
2039    C1      POP    BC    ; 弹出字节数，开始进行校对
203A    E1      POP    HL    ; 弹出源数据首地址

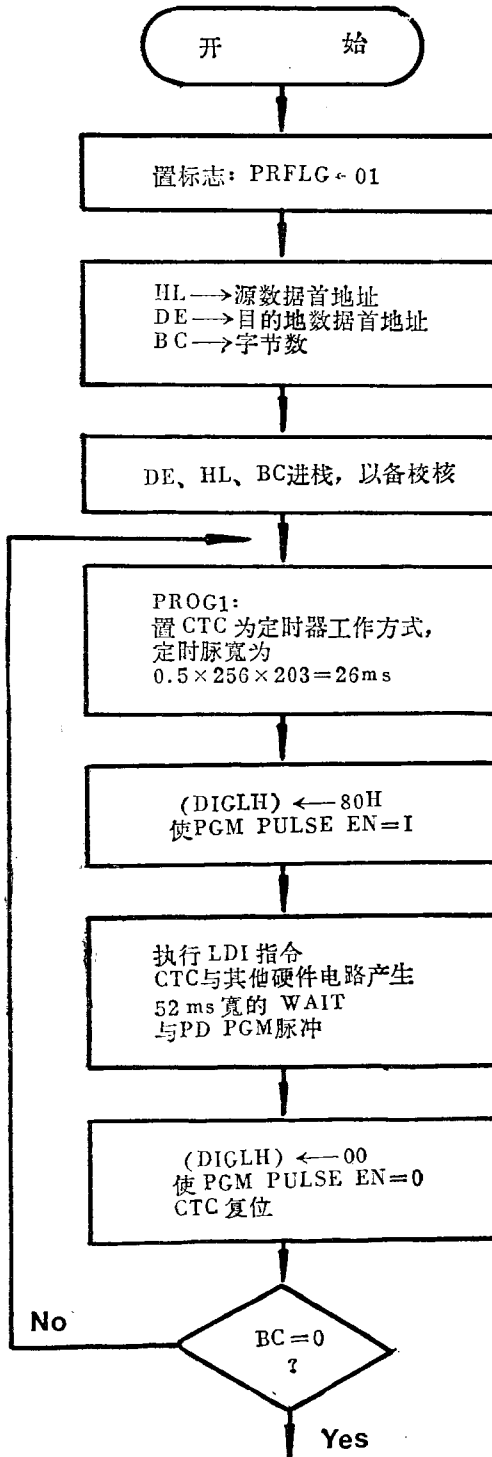
```

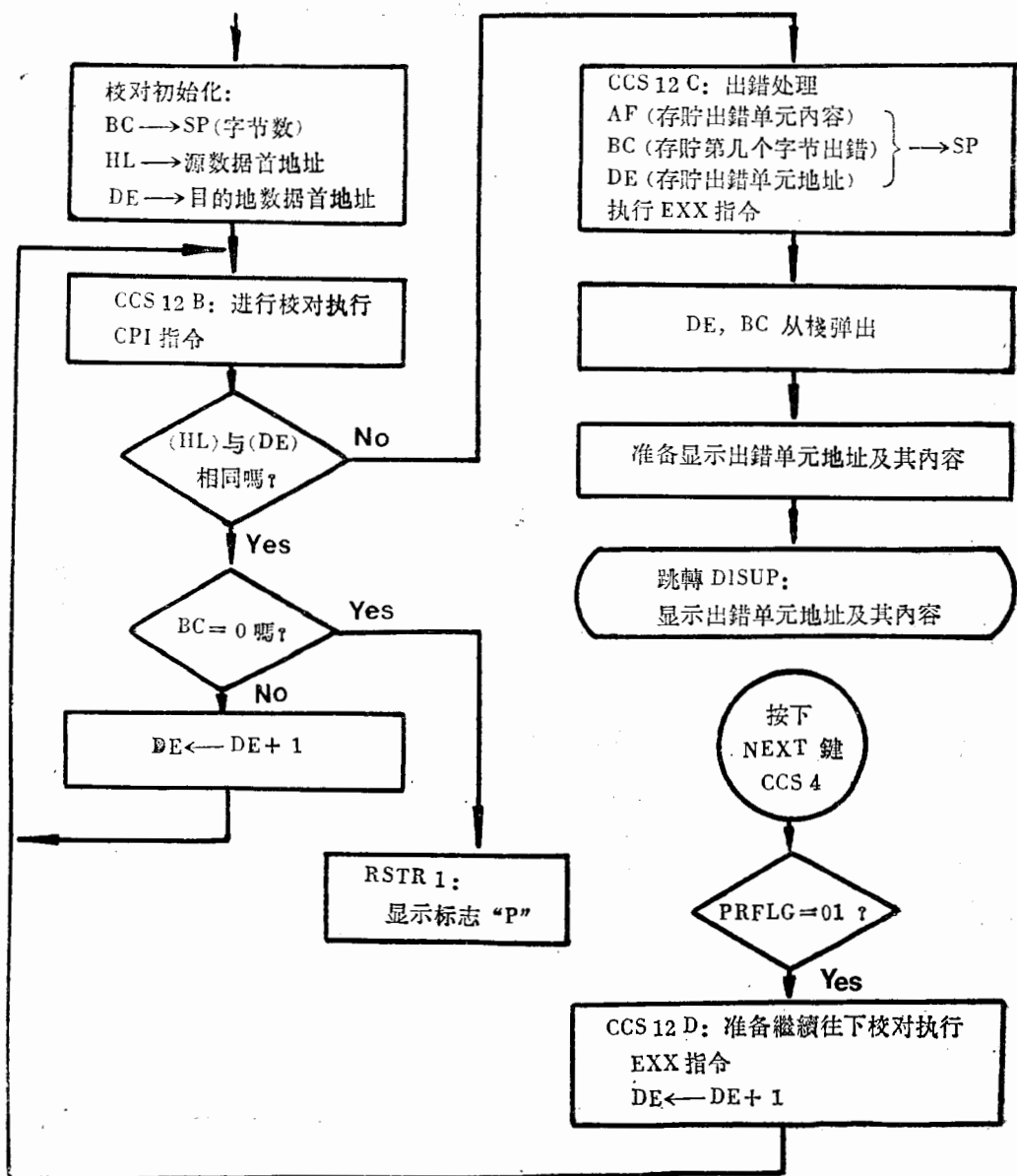


203B D1  
203C C30406

POP DE ; 弹出目的地数据首地址  
JP CCS12B ; 转到 TPBUG-A

### 程 序 流 程 图





## 4.7 求十进制数的算术和

下列程序是将两个 5 位十进制数(或和数不大于 6 位的 6 位数)进行算术加法运算。两数长度是相同的, 和数贮存在指定存储单元, 并将和数在显示器上显示出来。

； 数据

2030——贮存字节数(两位十进制数为 1 字节, 5 位十进制数为 3 字节)；

2031、2032、2033——贮存被加数, 按顺序先贮存低字节数；

2041、2042、2043——贮存加数, 按顺序先贮存低字节数；

2051、2052、2053——贮存和数、按顺序先贮存低字节数。

； 例子  $895867 + 91787 = 987654$

字节数: (2030) = 03

被加数: (2031) = 67      加数: (2041) = 87      和数: (2051) =

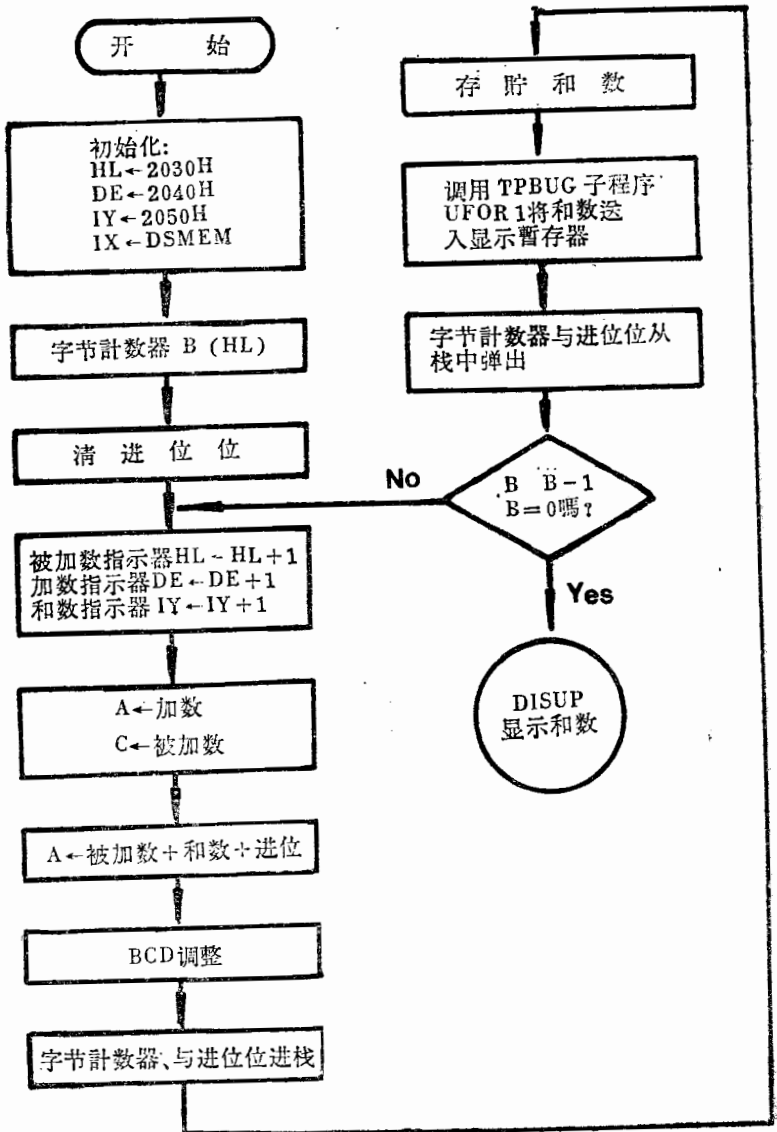
(2032) = 58                      (2042) = 17                      (2052) =

(2033) = 89                      (2043) = 09                      (2053) =

； 程序

			ORG	2000 H		
2000	21 30 20	LD	HL,	2030H	； 初始化	
2003	11 40 20	LD	DE,	2040H		
2006	FD 21 50 20	LD	IY,	2050H		
200A	DD 21 FB 2F	LD	IX,	(DSMEM4)		
200E	46	LD	B,	(HL)	； 计数器←字节数	
200F	AF	XOR	A		； 清进位位	
2010	23	LOOP:	INC	HL	； 指向被加数存储单元地址	
2011	13		INC	DE	； 指向加数存储单元地址	
2012	FC 23		INC	IY	； 指向和数存储单元地址	
2014	1A		LD	A,	(DE)	； 取出加数
2015	4E		LD	C,	(HL)	； 取出被加数
2016	89		ADC	A,	C	； 带进位加
2017	27		DAA			； BCD 调整
2018	FD 7700		LD	(IY+0),	A	； 贮存和数
201B	C5		PUSH	BC		； 记忆计数器内字节数
201C	F5		PUSH	AF		； 记忆进位位
201D	CD 3C06		CALL	UFOR1		； 显示初始化
2020	DD2B		DEC	IX		
2022	DD2B		DEC	IX		
2024	F1		POP	AF		
2025	C1		POP	BC		
2026	10 E8		DJ	NZ,	LOOP—\$	
2028	C2 F400		JP	DISUP		； 显示和数

# 程序流程图



## 附 录 一

**TP801 原理图、安装位置图、零件表**

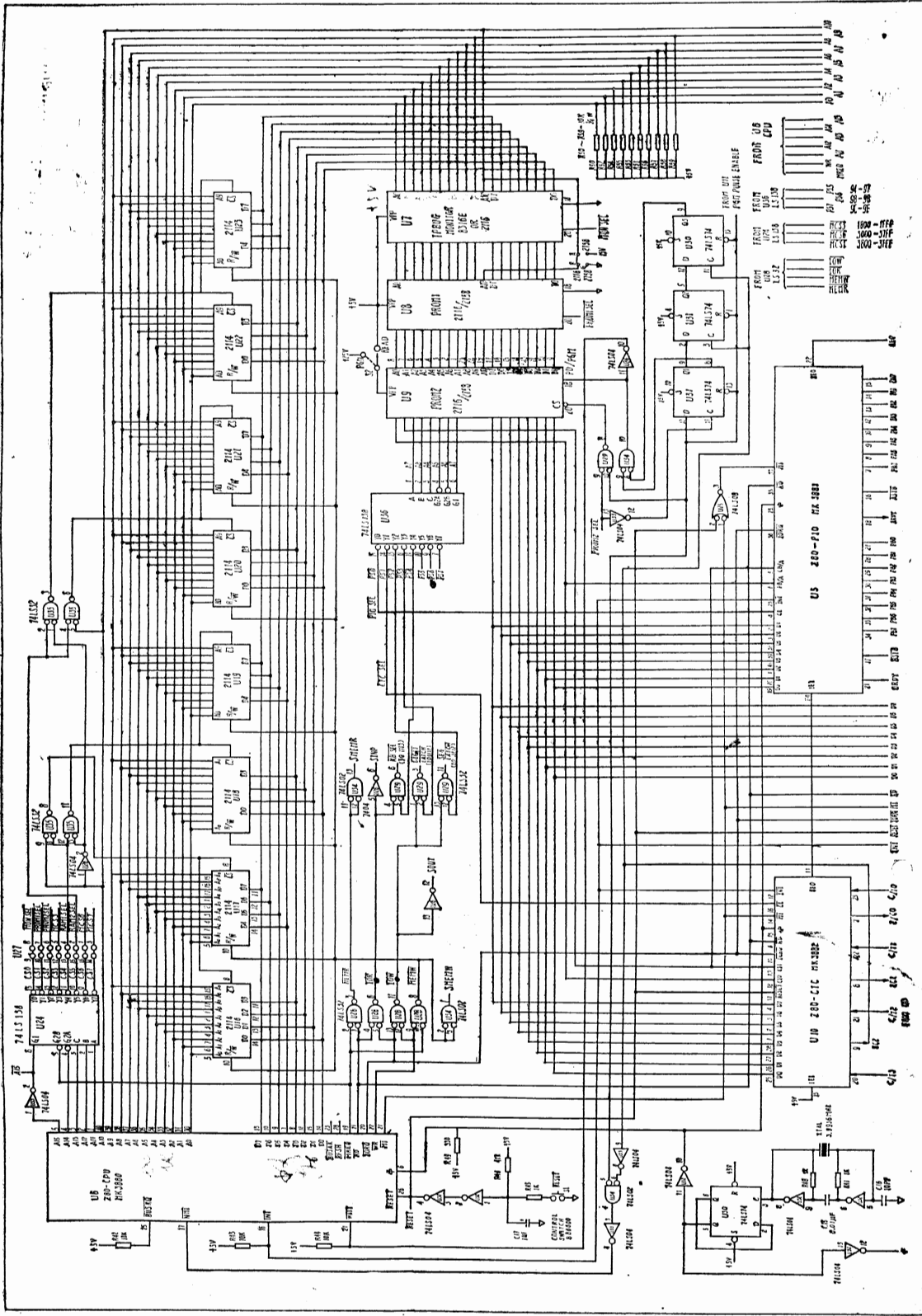


图 1.1 TP801-Z80 单板计算机线路原理图 1



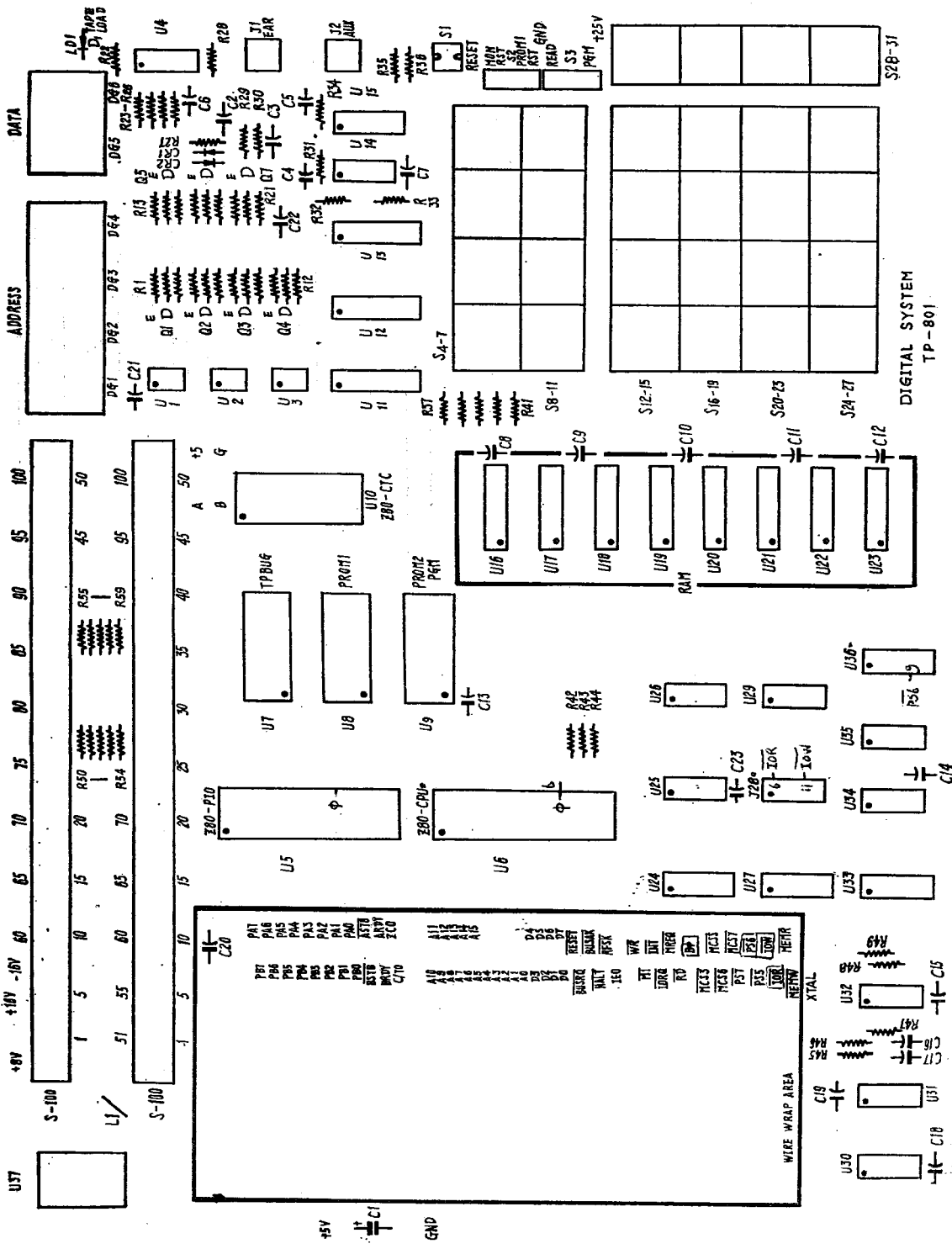


图 1.3 TP801-Z80 单板计算机安装位置图



TP 801-Z80 单板计算机零件表

序号	型号	名称 (英文)	中文名称	数量	元件代号
1	Z-80 CPU		中央处理器	1	U6
2	Z-80 PIO		并行输入输出电路	1	U5
3	Z-80 CTC		计数器定时器电路	1	U10
4	2114 RAM		450 ns 读写存储器	4/8	U16~U19 U20~U23
5	ROM--TPBUG/TPBUG-A		只读存储器TPBUG或TPBUG-A	1	U7
6	74LS138	Decoder	译码器	2	U24, U36
7	74LS32	Quad OR Gate	四或门	3	U28, U29, U35
8	74LS04	Hex Inverting Buffer	六反相器	2	U26, U33
9	7404	Hex Inverting Buffer	六反相器	1	U32
10	74LS08	Quad And Gate	四与门	1	U25
11	74LS244	Octal Buffer	8位缓冲器	1	U23
12	74LS273	Octal Latch	8位D锁存器	2	U11, U12
13	74LS74	Dual Latch	双D锁存器	2	U30, U31
14	74LS02	Quad NOR Gate	四或非门	1	U34
15	75452P	Peripheral Driver	驱动器	3	U1, U2, U3
16	MC14538BCP	CMOS Monostable	CMOS双单稳态触发器	1	U15
17	MC14013BCP	CMOS Latch	CMOS双D锁存器	1	U14
18	LM339	Quad Comparator	四比较器	1	U4
19	Single Digit Display		数码显示器	6	DG1~DG6
20	9012	PNP Transistor	PNP三极管	7	Q1~Q7
21	IN4148	Diode	二极管	2	CR1~CR2
22	3.9936 MHz	Parallel Resonant Cristal	并联谐振晶体	1	XTAL
23	Red LED	Indicator	红发光二极管	1	LDI
24	4.7K Ohm Resistor 1/4W 5%		4.7 K电阻色标: 黄、紫、红	1	R2, R5, R8, R11, R14, R17, R20, R32
25	10K Ohm Resistor 1/4W 5%		10K电阻色标: 棕、黑、橙	7	R36
26	68 Ohm Resistor 1/4W 5%		68Ω电阻 色标: 兰、灰、黑	29	R1, R4, R7, R10, R13, R16, R19, R25, R28, R37~R44, R50~R59
27	1K Ohm Resistor 1/4W 5%		1K电阻 色标: 棕、黑、红	7	R3, R6, R9, R12, R15, R18, R21 R23, R29, R30, R35, R45, R47, R48

28	470K Ohm Resistor $\frac{1}{4}$ W5%	470K电阻色标: 黄、紫、红、黄	2	R31, R34
29	220K Ohm Resistor $\frac{1}{4}$ W5%	220K电阻色标: 红、红、黄	1	R24
30	100K Ohm Resistor $\frac{1}{4}$ W5%	100K电阻色标: 棕、黑、橙、黑	3	R26, R37, R33
31	300 Ohm Resistor $\frac{1}{4}$ W5%	330 $\Omega$ 电阻色标: 橙、紫、黑	1	R49
32	47K Ohm Resistor $\frac{1}{4}$ W5%	47K电阻色标: 黄、紫、黑、橙	1	R46
33	100 Ohm Resistor $\frac{1}{4}$ W5%	100 $\Omega$ 电阻色标: 棕、黑、棕	1	R22
34	0.1 $\mu$ F Capacitor 20%	0.1 $\mu$ F电容	15	C3, C6, C8~C14, C18~C23
35	1 $\mu$ F Capacitor 20%	1 $\mu$ F钽电容	1	C17
36	10 $\mu$ F Capacitor 20%	10 $\mu$ F钽电容	1	C1
37	0.0047 $\mu$ F Capacitor 20%	0.0047 $\mu$ F电容	1	C7
38	10PF Capacitor 20%	10PF电容	1	C16
39	0.01 $\mu$ F Capacitor 20%	0.01 $\mu$ F电容	1	C15
40	620PF Capacitor 20%	620PF电容	2	C4, C5
41	0.047 $\mu$ F Capacitor 20%	0.047 $\mu$ F电容	1	C2
42	Push Button Control Switch	按钮	1	S1
43	Slide Switch	控制开关	2	S2, S3
44	Key Switch	键开关	28	S4~S31
45	Key Top with Transparent Cap	带透明罩的键帽	28	
46	Audio Jack	录音机插孔	2	J1, J2
47	8pin IC Socket	8脚插座	3	U1, U2, U3
48	14Pin IC Socket	14脚插座	12	U4, U14, U25, U26, U28, U29, U35
49	16Pin IC Socket	16脚插座	3	U15, U24, U36
50	18Pin IC Socket	18脚插座	4/8	U16~U19/U20~U23
51	20Pin IC Socket	20脚插座	3	U11, U12, U13
52	24Pin IC Socket	24脚插座	3	U7, U8, U9
53	28Pin IC Socket	28脚插座	1	U10
54	40Pin IC Socket	40脚插座	2	U5, U6
55	Printed Circuit Board 354 $\times$ 264mm	印刷电路板	1	
56	Plastic Support 15mm	塑料支柱	14	
57	Steel Screw M3 $\times$ 12 S, T R/H	M3 $\times$ 12螺钉	14	

## 附 录 二

### TP801 所用集成电路引脚图

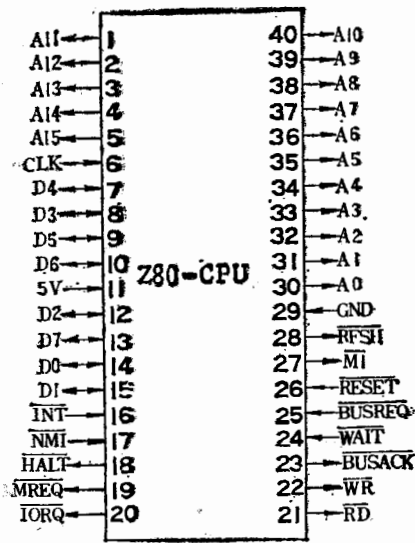


图 I.1 Z80-CPU 引脚图

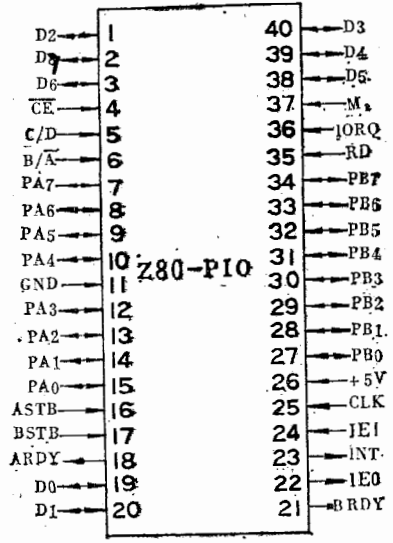


图 I.2 Z80-PIO 引脚图

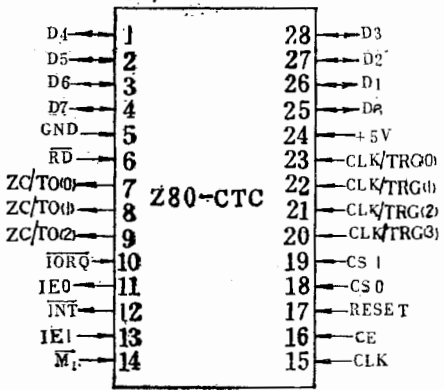


图 I.3 Z80-CTC 引脚图

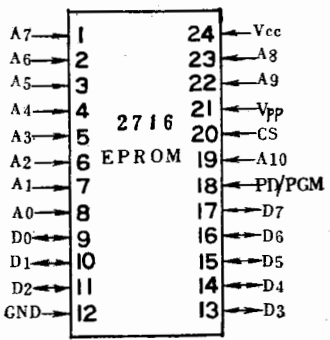


图 I.4 2716 EPROM 引脚图

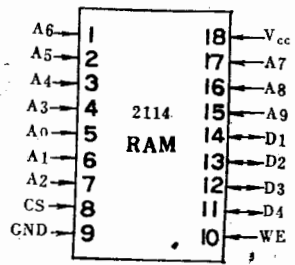


图 I.5 2114 RAM 引脚图

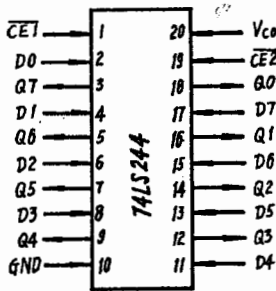


图 I.6 74LS244 八缓冲器引脚图

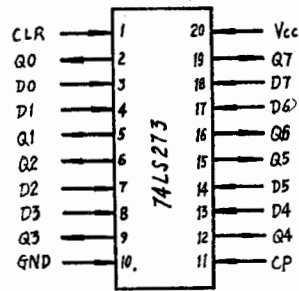


图 I.7 74LS273 八 D 锁存器引脚图

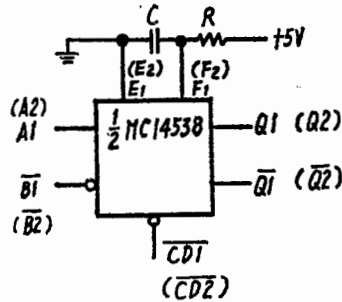
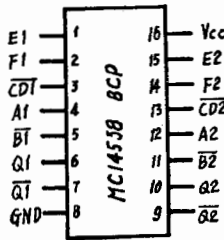


图 I.8 CMOS—MC14538BCP 双单稳态触发器引脚图及框图

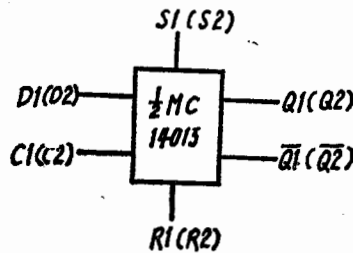
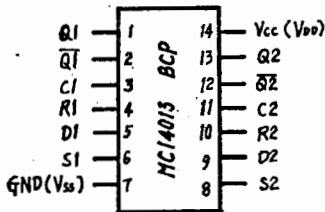


图 I.9 CMOS—MC14013BCP 双 D 锁存器引脚图及框图



G1	G2A	G2B	C	B	A	输出
1	0	0	0	0	0	$\bar{Y}0=0$ 其余为 1
1	0	0	0	0	1	$\bar{Y}1=0$ 其余为 1
1	0	0	0	1	0	$\bar{Y}2=0$ 其余为 1
1	0	0	0	1	1	$\bar{Y}3=0$ 其余为 1
1	0	0	1	0	0	$\bar{Y}4=0$ 其余为 1
1	0	0	1	0	1	$\bar{Y}5=0$ 其余为 1
1	0	0	1	1	0	$\bar{Y}6=0$ 其余为 1
1	0	0	1	1	1	$\bar{Y}7=0$ 其余为 1
不是上述情况			x	x	x	全部输出为 1

图 I.10 74LS138 八中取一译码器引脚图及真值表

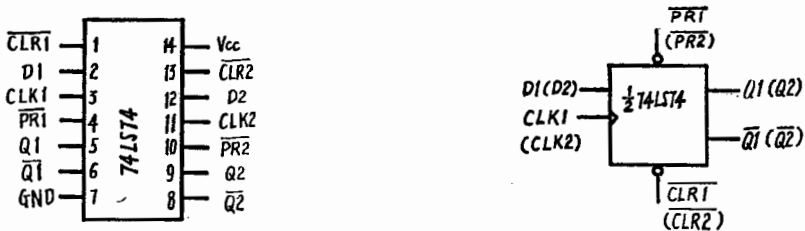


图 I .11 74LS74 双 D 锁存器的引脚图及框图



图 I .12 LM339 四比较器的引脚图



图 I .13 74LS04 六反相器的引脚图



图 I .14 74LS08 四与门的引脚图

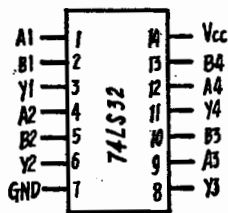


图 I .15 74LS32 四或门的引脚图

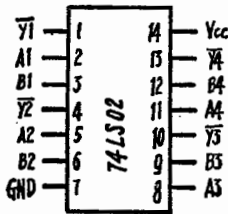
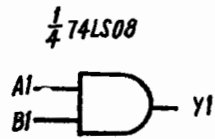


图 I .16 74LS02 四或非门的引脚图

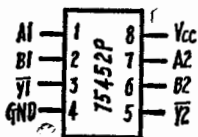
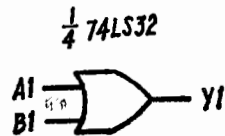
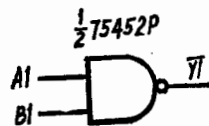


图 I .17 75452P 双驱动器的引脚图







## 附录三

### TPBUG 监控程序表

## LOAD MAP (存储分配)

		起始地址	结束地址
DK1: RESTR. OBJ[1]	REL	BEG ADDR 0000	END ADDR 00F3
DK1: DISUP. OBJ[1]	REL	BEG ADDR 00F4	END ADDR 0122
DK1: DECKY. OBJ[1]	REL	BEG ADDR 0123	END ADDR 0633
DK1: UTIL. OBJ[1]	REL	BEG ADDR 0634	END ADDR 07F4
DK1: UTILR. OBJ[1]	ABS	BEG ADDR 07F8	END ADDR 07FF

## GLOBAL CROSS REFERENCE TABLE (通用标号索引表)

SYMBOL (标号)	ADDR (地址)	REFERENCES (使用索引)
ARFLG	2FE0	06A4 03C5 02F0
BFLG	2FF4	0681 04BF 04BB 049E 0275 00A9
BPTAB	2FE4	067E 0250
CTC1P	07FA	04D5
CTC3L	07FE	0759
D20MS	064F	0172 0135
DECKY	0123	0121
DIG2	2FF5	068F 036C 01A6
DIG4	2FF6	0692 0498 0396 0235 01AC
DISMEM	2FF7	06AA 0687 065E 0617 05BB 0470 0443 03B6 02F4 00F5 00E4 00B4 0080
DISUP	00F4	062E 05CB 04C8 04B3 03B2 0393 0369 02EB 02D6 01BC 01A3 014D 0132 00F2 00D0 009D
DSMEM1	2FF8	017F 00B9
DSMEM2	2FF9	061F 05C3 0420 0378 0340 0334 00EB 00BC



BFLG	2FDF	06A1	03BF	030B	
RST16	2FC4	0011			
RST24	2FC7	0019			
RST32	2FCA	0021			
RST40	2FCD	0029			
RST48	2FD0	0031			
RST56	2FD3	0039			
SEGPT	07A6	0102			
SSFLG	2FF3	0695	027A	0202	01F9
STKPT	2FE2	0485	0468	0346	023F 01F6 01E6 01CD 00A1 0083 004B 0043 000A
STKPT1	2FE3	033C			
UABIN	06B3				
UBASC	06BB				
UFGCR	0686	00AF			
UFOR1	063C	062B	0623	061B	05C7 05BF 0423 03E5 03A7 0388 035D 034D 0343 0337 032E
UFOR2	0659	0318	0304	02E8	02E1 02D3 02CB 02C3 00EF 00E7 009A 0091 0088
UFOR3	067C	05D9	04A3	03E9	03D3 039F 037E 02D9 02BE 023B
UFOR4	06A7	04A6	0248	0205	0062
UIF	2FDD	0437	035A	0295	01F2 00AC 005F
UIX3	0634	04B6	0260	007A	
ULACC	06DB	05B3	05AC	05A5	05A1 059D 0599
UPACC	06C4				
UPACCS	06D5	056B	0566	055F	054E 0549 0544 053C 0538 0534 052D 0526

GLOBAL SYMBOL TABLE (通用标号表)

ARFLG	2FE0	BFLG	2FF4	BPTAB	2FE4	CTC1P	07FA
CTC3L	07FE	D20MS	064F	DECKY	0123	DIG2	2FF5
DIG4	2FF6	DISEM	2FF7	DISUP	00F4	DSMEM1	2FF8
DSMEM2	2FF9	DSMEM3	2FFA	DSMEM4	2FFB	DSMEM5	2FFC
DSMEM6	2FFD	DSMEM7	2FFE	FLG24	2FD9	INCHR	0758
INCHR4	079D	KEYPTR	2FDB	KYTBL	07B9	MFLG	2FE1
NMIS	01CB	OTCHR	06E4	OTCHR1	06F7	OTCHR6	0732
PELG	2FDE	PRFLG	2FDA	PUNHEH	2FC2	PUNHEL	2FC3
PUNHSH	2FC0	PUNHSL	2FC1	REGTB	07D5	REGTBP	07E5
REMBP2	007E	RESTAR	009F	RESTR1	00AE	RFLG	2FDF
RST16	2FC4	RST24	2FC7	RST32	2FCA	RST40	2FCD
RST48	2FD0	RST56	2FD3	SEGPT	07A6	SSFLG	2FF3
STKPT	2FE2	STKPT1	2FE3	UABIN	06B3	UBASC	06BB
UFGCR	0686	UFOR1	063C	UFOR2	0659	UFOR3	067C
UFOR4	06A7	UIF	2FDD	UIX3	0634	ULACC	06DB
UPACC	06C4	UPACCS	06D5				

地址	目的码	语句序号	源语句
		0002	程序名称 RESTR
		0003;	重新启动(RST), 不可屏蔽中断(NMI)和移去断点(BP)程序
		0004;	版本 1.7 5/18/78
		0005	PSECT REL
		0006	GLOBAL DSMEM4
		0007	GLOBAL DSMEM1
		0008	GLOBAL DISUP
		0009	GLOBAL UIF
		0010	GLOBAL UIX3
		0011	GLOBAL STKPT
		0012	GLOBAL RST16
		0013	GLOBAL RST24
		0014	GLOBAL RST32
		0015	GLOBAL RST40
		0016	GLOBAL RST48
		0017	GLOBAL RST56
		0018	GLOBAL DSMEM5
		0019	GLOBAL DSMEM2
		0020	GLOBAL DISMEM
		0021	GLOBAL DSMEM3
		0022	GLOBAL UFGCR
		0023	GLOBAL UFOR1
		0024	GLOBAL NMIS
		0025	GLOBAL UFOR3
		0026	GLOBAL RESTAR
		0027	GLOBAL REMBP2

地址	目的码	语句序号	源语句
>0000		0028	GLOBAL RESTR 1
		0029	GLOBAL BFLG
		0030	ORG 0000H
		0031;	※※※※重新启动指令处理※※※※
		0032;	※※※※复位入口点—RST0※※※※
		0033;	※※※※RST0用于复位, RST8用于断点, 其它所有
		0034;	的RST指令如RST16、RST24、RST32、RST
		0035;	48和RST56都在RAM中分配存储单元, 用
		0036;	户可以在那里设置一条转移指令, 转移到存在
		0037;	RAM中的服务程序。当进入断点后, 用户程
		0038;	序所有的寄存器都被推入堆栈, 而后将所有的
		0039;	断点从用户程序中移走
		0040;	※※※※复位入口点—RST0
0000	31C02F	0041	LD SP, 2FC0H
0003	C39F00	0042	JP RESTAR ; 转去完成初始化
		0043;	※※※※断点入口—RST8 ; 开始重新启动表
>0008		0044	ORG 0008H
0008	ED73E22F	0045	BPENT: LD (STKPT), SP; 保存用户的栈指针SP
000C	F5	0046	PUSH AF ; 用户寄存器进栈
000D	C5	0047	PUSH BC
000E	1803	0048	JR BPENT2-\$
		0049;	※※※※用户使用RST16的入口
0010	C3C42F	0050	JP RST16 ; 转到RAM
0013	ED57	0051	BPENT2: LD A, I ; 中断向量寄存器I送累加器A, 中断允许触发器IFF 送标志寄存器F

地址	目的码	语句序号	源语句
0015	F3	0052	DI
0016	1803	0053	JR BPFNT3-\$
0018	C3C72F	0054, ※※※※	用户使用 RST24 的入口
001B	D5	0055	JP RST24
001C	E5	0056	BPENT3; PUSH DE
001D	F5	0057	PUSH HL
001E	1003	0058	PUSH AF
0020	C3CA2F	0059	JR BPENT4-\$
0023	08	0060, ※※※※	用户使用 RST32的入口
0024	D9	0061	JP RST 32
0025	F5	0062	BPENT4; EX AF, AF'
0026	1803	0063	EXX
0028	C3CD2F	0064	PUSH AF
002B	C5	0065	JR BPENT5-\$
002C	D5	0066, ※※※※	用户使用 RST40 的入口
002D	E5	0067	JP RST40
002E	1803	0068	BPENT5; PUSH BC
0030	C3D02F	0069	PUSH DE
0033	DDE5	0070	PUSH HL
0035	00	0071	JR BPENT6-\$
0036	1803	0072, ※※※※	用户使用 RST48 的入口
		0073	JP RST48
		0074	BPENT6; PUSH IX
		0075	NOP
		0076	JP BPENT7-\$

，保护 I 和 IFF

，取辅助寄存器的内容

，空一字节



地址	目的码	源语句	源语句
0038	C3D32F	JP RST56	0077; ※※※※用户使用RST56的入口
003B	FDE5	0079 BPENT7; PUSH IY	
003D	3E03	LD A,03H	
003F	D386	OUT (CTC2),A	; 使键盘不能通过CTC2请求中断
0041	DD2AE22F	LD IX,(STKPT)	; 取用户栈指针SP
0045	DD23	INC IX	
0047	DD23	INC IX	
0049	DD22E22F	(STKPT),IX	
004D	DD7EFE	LD A,(IX-2)	; 测试PC的低字节
0050	B7	OR A	; 设置状态标志
0051	2003	JR NZ,BPENT8-\$	
0053	DD35FF	DEC (IX-1)	; 高字节减1
0056	DD35FE	DEC (IX-2)	; 低字节减1
0059	DD7EF4	LD A,(IX-12)	; 在堆栈中的IFF送A
005C	E604	AND 04H	; 保留F中的IFF,屏蔽掉其它位
005E	32DD2F	LD (UIF),A	; 保存供后面使用
0061	CD7C06	CALL UFOR3	; 取断点表地址,断点数存入B寄存器
0064	1803	JR BPNT8A-\$	
0066	C3CB01	JP NMIS	; 不可屏蔽中断入口
0069	2813	Z,REMBP2-\$	; 若无断点,转去显示PC和A寄存器内容
0098;	※※※※	※※※※	※※※※
0099;	※※※※	※※※※	※※※※
0100;	※※※※	※※※※	※※※※

0099; 当进入监控程序TPBUG时移去断点,在执行或继续执行用户程序命令时,恢复断点

地址	目的码	源语句	源语句
006B	DD7E02	LD A,(IX + 2)	; 取要恢复的操作码
006E	FECF	CP 0CFH	
0070	2807	JR Z,REMBP1-\$	; 如多断点则转移
0072	DD6E01	LD L,(IX + 1)	
0075	DD6600	LD H,(IX + 0)	; 取断点地址
0078	77	LD (HL),A	; 恢复操作码
0079	CD3406	CALL UIX3	; 取下一个断点地址, 断点数目计数器 B 减 1
007C	20ED	JR NZ,REMBP-\$	; 再次返回
007E	DD21F72F	LD IX,DISMEM	; 指向显示缓冲区首地址 DISMEM
0082	2AE22F	LD HL,(STKPT)	; 栈顶置入 HL
0085	2B	DEC HL	
0086	7E	LD A,(HL)	; 取 PC 的高字节
0087	CD3C06	CALL UF0R1	; 写入显示缓冲区
008A	DD23	INC IX	
008C	DD23	INC IX	
008E	2B	DEC HL	
008F	7E	LD A,(HL)	; 取 PC 的低字节
0090	CD3C06	CALL UF0R1	; 写入显示缓冲区
0093	DD23	INC IX	
0095	DD23	INC IX	
0097	2B	DEC HL	
0098	7E	LD A,(HL)	; 取栈存的 A
0099	CE3C06	CALL UF0R1	; 写入显示缓冲区

地址	目的码	语句序号	源语句
009C	C3F400	0126	DISUP JP 转显程序
		0127	完成重新启动的程序
009F	ED73E22F	0128	RESTAR, LD (STKPT), SP ; 保护用户的 SP
00A3	31A82F	0129	SP, 2FA8H ; 对监控程序的 SP 置初值
00A6	3E00	0130	LD A, 00H
00A8	32F42F	0131	LD (BFLG) A ; 清断点标志 BFLG
00AB	32DD2F	0132	LD (UIF), A ; 清用户 IFF2 标志
00AE	CD8606	0133	RESTR1; CALL UFGCR ; 清各个标志字节
00B1	3E11	0134	LD A, 11H ; 送提示符“-”
00B3	32F72F	0135	LD (DISMEM), A ;
00B6	3E10	0136	LD A, 10 H ; 送空白符
00B8	32F82F	0137	LD (DSMEM1), A ;
00BB	32F92F	0138	LD (DSMEM2), A ;
00BE	1802	0139	JR RESTR2-\$ ;
00C0	1813	0140	JR RESTR3-\$ ; 相对偏移量 (OFFSET) 计算程序的入口
00C2	32FA2F	0141	RESTR2; LD (DSMEM3), A ;
00C5	32FB2F	0142	LD (DSMEM4), A ;
00C8	32FC2F	0143	LD (DSMEM5), A ;
00CB	DB90	0144	IN A, (KBSEL) ; 检测开关 S2 的位置
00CD	CB6F	0145	BIT 5, A ; 测试
00CF	C2F400	0146	JP NZ, DISUP ; 转显程序
00D2	C30008	0147	JP 0800H ; 转向 PROM1 的起始地址

0148, 相对偏移量计算子程序—自动地计算相对偏移

0149, 量并将计算结果放入存储器并显示

地址	目的码	语句序号	源语句
		0150;	HL放相对转移的目的地的操作码地址
		0151;	DE放带有相对偏移量的相对转移指令所在的地址
00D5	13	0152	RESTR3; INC DE DE ; 指向偏移量
00D6	D5	0153	PUSH DE DE
00D7	DDE1	0154	POP IX IX ; 保护偏移量的地址
00D9	13	0155	INC DE DE ; 指向下一条指令的操作码
00DA	7D	0156	LD A,L A,L ; 取低字节
00DB	93	0157	SUB E E ; 减
00DC	6F	0158	LD L,A L,A ; 保存偏移量
00DD	DD7700	0159	LD (IX + 0), A ; 将偏移量填入需要的存储单元
00E0	7C	0160	LD A,H A,H ; 取高字节
00E1	9A	0161	SBC A,D A,D ; 减
00E2	DD21F72F	0162	LD IX,DISMEM IX,DISMEM
00E6	CD3C06	0163	CALL UFOR1 UFOR1 ; 将高字节减法结果写入显示缓冲区, 如是FF或00, 则为超越正常范围
00E9	DD21F92F	0164	LD IX,DSMEM2 IX,DSMEM2
00ED	7D	0165	LD A,L A,L
00EE	CD3C06	0166	CALL UFOR1 UFOR1 ; 写偏移量到显示缓冲区
00F1	C3F400	0167	JP DISUP DISUP ; 转显示程序
>0086		0168	CTC2: EQU 86H ; 单步/EPROM编程
>0090		0169	KBSEL: EQU 90H ; MONITOR/PROM1开关指向检测
		0170	END

地址	目的码	语句序号	源语句
		0002	程序名称 DISUP
		0003;	更新显示程序
		0004;	版本0.3 3/13/78
		0005	PSECT REL
		0006	GLOBAL DISMEM
		0007	GLOBAL SEGPT
		0008	GLOBAL DECKY
		0009	GLOBAL DISUP
		0010	GLOBAL DISUP
>00F4		0011	ORG 00F4
		0012;	更新显示程序
		0013;	功能: 从显示缓冲区移数据到数码显示
		0014;	入口: 被显示的数据在六个存储单元内, 起始单元为 DISMEM
		0015;	
		0016;	出口: 刷新所有的六位数据—使用的
		0017;	寄存器是: IX, AB, D, E, H, L,
		0018;	寄存器的使用: HL是指向 DISMEM 中现行单元的指示器
		0019;	IX是指向字形表的指示器
		0020;	B是现行数位指示器
		0021;	E保持被显数据
		0022;	A和D是暂存寄存器
		0023	DISUP; LD HL, DISMEM; 指向显示缓冲区的首地址 DISMEM
		0024	LD B, 020H ; 指向左边数位
		0025	DISUP; LD E, (HL) ; 取第一个字节
		0026	LD D, 00H ; D寄存器清零

| 00F4 21F72F  
 81 | 00F7 0620  
 | 00F9 5E  
 | 00FA 1600

地址	目的码	语句序号	源语句
00FC	3E00	0027	LD A,00H
00FE	D38C	0028	OUT (DIGLH),A
0100	DD21A607	0029	LD IX,SECP T
0104	DD19	0030	ADD IX,DE
0106	DD7E00	0031	LD A,(IX+0)
0109	D388	0032	OUT (SEGLH),A
010B	78	0033	LD A,B,
010C	D38C	0034	OUT (DIGLH),A
010E	1E2D	0035	LD E,2DH
0110	1D	0036	DISUP2; DEC E
0111	3E00	0037	LD A,00H
0113	BB	0038	CP E
0114	20FA	0039	JR NZ,DISUP2-\$
0116	3E01	0040	LD A,01H
0118	B8	0041	CP B
0119	2805	0042	JR Z,DISUP3-\$
011B	23	0043	INC HL
011C	CB38	0044	SRL B
011E	18D9	0045	JR DISUP1-\$
0120	C32301	0046	DISUP3; JR DECKY
>008C		0047	DIGLH; EQU 8CH
>0088		0048	SEGLH; EQU 88H
		0049	END

; 关显示  
 ; 变址寄存器指向字形表  
 ; 为获得字形变址作加法  
 ; 得到字形  
 ; 输出字形到字形选择锁存器  
 ; 输出到数位选择锁存器  
 ; 设置 1 ms 延时  
 ; 延时循环  
 ; B = 1 ?  
 ; 是, 转出口  
 ; 否, 指向下一位  
 ; 移位到下一位  
 ; 转键盘分析程序  
 ; 只写, 数位选择  
 ; 只写, 字形选择

地址	目的码	语句序号	源语句
		0002	ORG 0123H
		0003	程序名称 DECKY
		0004	； 键分析和执行
		0005	； 版本 2.1 5/17/78
		0006	PSECT REL
		0007	GLOBAL UFOR4
		0008	GLOBAL UFOR1
		0009	GLOBAL UFOR2
		0010	GLOBAL UFOR3
		0011	GLOBAL REMBP2
		0012	GLOBAL D20MS
		0013	GLOBAL KEYPTR
		0014	GLOBAL STKPT
		0015	GLOBAL STKPT1
		0016	GLOBAL SSFLC
		0017	GLOBAL UPACCS
		0018	GLOBAL OTHCR1
		0019	GLOBAL OTHCR6
		0020	GLOBAL UFOR1
		0021	GLOBAL BPTAB
		0022	GLOBAL DIG4
		0023	GLOBAL DIG2
		0024	GLOBAL DIG8
		0025	GLOBAL MFLG
		0026	GLOBAL BFLG

地址	目的码	语句序号	源语句
		0027	GLOBAL RFLG
		0028	GLOBAL ARFLG
		0029	GLOBAL PFLG
		0030	GLOBAL PRFLG
		0031	GLOBAL INCHR
		0032	GLOBAL PUNHSH
		0033	GLOBAL PUNHSL
		0034	GLOBAL PUNHEH
		0035	GLOBAL PUNHEL
		0036	GLOBAL REGTB
		0037	GLOBAL RESTAR
		0038	GLOBAL RESTR1
		0039	GLOBAL DISUP
		0040	GLOBAL REGTBP
		0041	GLOBAL DISMEM
		0042	GLOBAL RFLG
		0043	GLOBAL UIF
		0044	GLOBAL DSMEM1
		0045	GLOBAL DSMEM2
		0046	GLOBAL DSMEM3
		0047	GLOBAL DSMEM4
		0048	GLOBAL DSMEM5
		0049	GLOBAL DSMEM6
		0050	GLOBAL DSMEM7
		0051	GLOBAL ULACC



地址	目的码	语句序号	源语句
		0052	GLOBAL FLG24
		0053	GLOBAL UIX3
		0054	GLOBAL KYTBL
		0055	GLOBAL CTC1P
		0056	GLOBAL NMIS
		0057	GLOBAL DECKY
		0058;	功能: 空白数码显示, 扫描键矩阵有否键闭合, 如
		0059;	发现键闭合, 就产生一个 20 ms 的延时, 用以
		0060;	消除键抖动。每次扫描键矩阵的一行, 并检
		0061;	测所有的列。如果闭合的是一个十六进制的
		0062;	数字键, 就将其放到存储器显示缓冲区的下
		0063;	一个空白单元中(指针为 KEYPTR)。输入
		0064;	两位数就设置标志(DIG2)。当已输入了四
		0065;	位数, 就设置标志(DIG4), 当已输入了八位
		0066;	数, 就调用存储器交换程序。如经分析识别
		0067;	为一个命令键, 相应的处理程序将从跳转表
		0068;	(JPTAB) 中找到。命令键的处理程序也包
		0069;	括在本模块中。
		0070;	
		0071;	入口: 无什么要求, 跟在更新显示程序 DISUP 后面执行。
		0072;	出口: 如果是命令键, 在执行命令之后转出。如果是十六进制数字键, 数据已写入显示缓冲
		0073;	区(DISMEM)之后转出。
		0074;	使用的寄存器:
		0075;	A—暂存键值

地址	目的码	语句序号	源语句
		0076;	B—数值值
		0077;	C—暂存由行和列数据确定的特征字
		0078;	HL—键值查找表 (KYTBL) 指针
		0079;	HL—显示缓冲区指针 (KEYPTR)
		0080;	BC—暂存—多用于计算出要存放在 DISMEM 中的偏移量
		0081;	EXX—在 EPROM 编程的校对过程中保存指针用。在显示了错误之后, 使用 NEXT 键
		0082;	继续校对时, 恢复指针用。
0123	3E7F	0083 DECKY; LD	A, 7FH
0125	D388	0084 OUT	(SEGLH), A ; 关显示
0127	3E3F	0085 LD	A, 3FH
0129	D38C	0086 OUT	(DIGLH), A ; 输出, 使所有的行为低
012B	DB90	0087 IN	A, (KBSEL) ; 输入列数据
012D	E61F	0088 AND	1FH ; 屏蔽无用的位 (D5, D6, D7)
012F	FE1F	0089 CP	1FH ; 是否有键按下?
0131	CAF400	0090 JP	Z, DISUP ; 否, 返回显示
0134	CD4F06	0091 CALL	D20MS ; 为了除去键抖动而延时 20ms
0137	0E8C	0092 LD	C, DIGLH
0139	0601	0093 LD	B, 01H
013B	ED41	0094 KEYDN1; OUT	(C), B ; 选中其中一行键
013D	DB90	0095 IN	A, (KBSEL) ; 输入列数据
013F	E61F	0096 AND	1FH ; 屏蔽无用的位 (D5, D6, D7)
0141	FE1F	0097 CP	1FH ; 是否有键按下?
0143	200A	0098 JR	NZ, KEYDN2—\$ ; 是, 转键译码
0145	CB20	0099 SLA	B ; 否, 选择下一行
0147	3E40	0100 LD	A, 40H

地址	目的码	语句序号	源语句	
0149	B8	0101	CP B	; 是否全扫描完?
014A	20EF	0102	JB NZ,KEYDN1-\$	; 否, 返回循环
014C	C3F400	0103	JP DISUP	; 是, 转显示
014F	0E00	0106	KEYDN2: LD C,00H	
0151	0D	0107	KEYDN3: DEC C	
0152	CB38	0108	SRL B	
0154	20FB	0109	JR NZ,KEYDN3-\$	; 当B = 0 时, 才往下执行
0156	CB21	0110	SLA C	
0158	CB21	0111	SLA C	
015A	CB21	0112	SLA C	
015C	CB21	0113	SLA C	; 得到字节高四位
015E	81	0114	ADD A,C	; 与 A 相加
015F	21B907	0115	LD HL,KYTL	; 设置表的指针
0162	BE	0116	KEYDN4: CP (HL)	; 在表中找到等于 A 的项
0163	2804	0117	JR Z,KEYDN5-\$	; 是, 找到此项
0165	23	0118	INC HL	
0166	04	0119	INC B	; 否, 调整指针
0167	18F9	0120	JR KEYDN4-\$	; 循环返回
0169	DB90	0121	KEYDN5: IN A, (KBSEL)	; 检测键释放
016B	E61F	0122	AND 01FH	; 屏蔽其它输入位
016D	FE1F	0123	CP 01FH	
016F	20F8	0124	JR NZ,KEYDN5-\$	; 循环直到键释放为止

地址	目的码	语句序号	源语句	
0171	CD4F06	0125	CALL	D20MS ; 为消去键抖动而延时 20ms
0174	78	0126	LD	A, B ; 取键值
0175	FE10	0127	CP	10H
0177	3045	0128	JR	NC, KEYDN6—\$ ; 若是命令键, 则转到分析
0179	2ADB2F	0129	LD	HL, (KEYPTR) ; 若是十六进制数字键, 则存入显示缓冲区
017C	70	0130	LD	(HL), B
017D	B7	0131	OR	A
017E	01F82F	0132	LD	BC, DSMEM1
0181	ED42	0133	SBC	HL, BC ; 输入的是第 2 位数吗?
0183	2820	0134	JR	Z, KEYDNA—\$ ; 是, DIG2 标志置 1
0185	B7	0135	OR	A ; 清进位标志
0186	01F82F	0136	LD	BC, DSMEM3
0189	2ADB2F	0137	LD	HL, (KEYPTR)
018C	ED42	0138	SBC	HL, BC ; 输入的是第 4 位数吗?
018E	281B	0139	JR	Z, KEYDN8—\$ ; 是, DIG 4 标志置 1
0190	B7	0140	OR	A
0191	01FE2F	0141	LD	BC, DSMEM7
0194	2ADB2F	0142	LD	HL, (KEYPTR) ; 十六进制数字键, 存入显示缓冲区中
0197	ED42	0143	SBC	HL, BC ; 显示缓冲器已写入 8 位数吗?
0199	2816	0144	JR	Z, KEYDN9—\$ ; 是, 跳转
019B	2ADB2F	0145	KEYDN7; LD	HL, (KEYPTR) ; 取键指针
019E	23	0146	INC	HL ; 指针加 1
019F	22DB2F	0147	LD	(KEYPTR), HL ; 保护 HL 内容
01A2	C3F400	0148	JP	DISUP ; 转出

地址	目的码	语句序号	源语句
01A5	21F52F	0149	KEYDNA,LD HL,DIG2
01A8	34	0150	INC (HL)
01A9	18F0	0151	JR KEYDN7-\$
01AB	21F62F	0152	KEYDN8,LD HL,DIG4
01AE	34	0153	INC (HL)
01AF	18EA	0154	JR KEYDN7-\$
01B1	CDB403	0155	KEYDN9,CALL ALTER
01B4	2ADB2F	0156	LD HL,(KEYPTR)
01B7	2B	0157	DEC HL
01B8	22DB2F	0158	LD (KEYPTR),HL ; 把指针向回调整到进入六位数时的位置
01BB	C3F400	0159	JP DISUP ; 存 HL
01BE	D610	0160;	找到命令键处理程序的地址, 此时键值在 A 中
01C0	4F	0161	KEYDN6,SUB 10H ; 第 16 个键的偏移量为 0, 第 17 个键的偏移量为 3, 先求差值
01C1	81	0162	LD C,A
01C2	81	0163	ADD A,C ; 二倍差值
01C3	4F	0164	ADD A,C ; 三倍差值
01C4	0600	0165	LD C,A
01C6	21C0C2	0166	LD B,00H
01C9	09	0167	LD HL,JPTAB ; 取表的首地址
01CA	E9	0168	ADD HL,BC ; 得到表的项号地址 (首地址加偏移量)
01CB	ED73E22F	0169	JP (HL)
01CF	F5	0170;	不可屏蔽中断服务程序 (单步或中止程序时使用)
		0171	NMIS,LD (STKPT),SP ; 保存用户的 SP
		0172	PUSH AF

地址	目的码	语句序号	源语句
01D0	3E03	0173	LD A,03H
01D2	D386	0174	OUT (CTC2),A ; 关闭 CTC
01D4	ED57	0175	LD A,I ; I 送 A, IFF 送 F
01D6	C5	0176	PUSH BC ; 保存各寄存器
01D7	D5	0177	PUSH DE
01D8	E5	0178	PUSH HL
01D9	F5	0179	PUSH AF
01DA	08	0180	EX AF,AF'
01DB	D9	0181	EXX
01DC	F5	0182	PUSH AF
01DD	C5	0183	PUSH BC
01DE	D5	0184	PUSH DE
01DF	E5	0185	PUSH HL
01E0	DDE5	0186	PUSH IX
01E2	FDE5	0187	PUSH IY
01E4	DD2AE22F	0188	LD IX,(STKPT) ; 取用户 SP
01E8	DD23	0189	INC IX ; 调整指针到用户 SP 的实际值
01EA	DD23	0190	INC IX ; IFF 在堆栈内的位置
01EC	DD7EF4	0191	LD A,(IX-12) ; 屏蔽其它位, 保留 IFF
01EF	E604	0192	AND 04H ; 存入 UIF
01F1	32DD2F	0193	LD (UIF),A ; 保存用户 SP
01F4	DD2E22F	0194	LD (STKPT),IX ; 单步方式吗?
01F8	3AF32F	0195	LD NMIS1; LD A,(SSFLG) ; 设置状态标志
01FB	B7	0196	OR A ; 否, 是一个由键盘给出的中断
01FC	CA2301	0197	JP Z, DECKY

地址	目的码	语句序号	源语句
01FF	3E00	0198	NNIS2; LD A, 00H
0201	32F32F	0199	LD (SSFLG), A ; 清标志 SSFLG
0204	CD7C06	0200	CALL UF0R3 ; 取断点数目
0207	CA7E00	0201	JP Z, REMBP2 ; 无断点, 转去显示 PC 和 A
020A	1842	0202	CCS1G—\$ ; 转装配断点, 使键盘中断有效, 并返回用户程序
020C	C33002	0203	; 命令键跳转送表
020F	C37202	0204	JPTAB; JP CCS1 ; EXEC (连续执行程序)
0212	C3A102	0205	JP CCS2 ; SS (单步执行程序)
0215	C3A402	0206	JP CCS3 ; MON (监控)
0218	C3ED02	0207	JP CCS4 ; NEXT (下一)
021B	C30803	0208	JP CCS5 ; REG' DISP (辅助寄存器单元检查)
021E	C36B03	0209	JP CCS6 ; REG DISP (寄存器检查)
0221	C39503	0210	JP CCS7 ; PORT EXAM (口检查)
0224	C39704	0211	JP CCS8 ; MEM EXAM (存储单元检查)
0227	C3CA04	0212	JP CCS9 ; BP (设置断点)
022A	C38105	0213	JP CCS10 ; PUNCH TAPE (信息转储磁带)
022D	C3D305	0214	JP CCS11 ; LOAD TAPE (磁带输入)
0230	3E00	0215	JP CCS12 ; PROC PROM (EPROM写入)
0232	D38C	0216	; EXEC 键处理程序
0234	3AF62F	0217	CCS1; LD A, 00H
0237	B7	0218	OUT (DIGLH), A ; 清显示
0238	280D	0219	LD A, (DIG4)
023A	CD5906	0220	OR A ; 设置状态标志
		0221	JR Z, CCS1A—\$ ; 如果没有输入 4 位数, 则转到继续执行方式
		0222	CALL UF0R2 ; 将输入的四个数作为起始地址, 置入 HL

地址	目的码	语句序号	源语句
023D	DD2AE22F	0223	LD IX,(STKPT) ; 设置用户 SP
0241	DD75FE	0224	(IX-2),L
0244	DD74FF	0225	(IX-1),H ; 更换 PC 为新的值
0247	CD7C06	0226 ; 继续执行方式	UFOR3 ; 测试断点是否有效
024A	202B	0227 CCS1A: CALL	NZ,CCS2A--\$ ; 是, 单步执行一步
024C	1816	0228 JR	CCS1D--\$ ; 若断点无效, 允许键盘用 NMI 中断中止用户程序并用 RET 进入用户程序
024C	1816	0229 JR	
024E	DD21E42F	0230 ; 装配断点并保存用户操作码	IX,BPTAB ; 设定断点表指针
0252	DD6600	0231 CCS1C: LD	H,(IX+0)
0255	DD6E01	0232 CCS1CA LD	L,(IX+1)
0258	7E	0233 LD	A,(HL) ; 取操作码地址
0259	0ECF	0234 LD	C,0CFH ; 取操作码
025B	71	0235 LD	(HL),C ; 置 RST8 操作码 CF 到 C
025C	DD7702	0236 LD	(IX+2),A ; 设置一个断点
025F	CD3406	0237 LD	UIX3 ; 保存用户操作码
0262	20EE	0238 CALL	NZ,CCS1CA--\$ ; 指向下一个单元
0264	3E04	0239 JR	; 为处理更多的断点而循环返回
0266	D38C	0240 ; 允许键盘中止用户程序	A,04H ; 允许 NMI 中断
0268	2E45	0241 CCS1D: LD	(DIGLH),A
026A	D886	0242 OUT	A,45H ; CTC 的 ZC/T0 待命
026C	3E01	0243 LD	(CTC2),A
026E	D386	0244 OUT	A,01H ; 若输入一个负脉冲, CTC 即发生中断请求
026E	D386	0245 LD	(CTC2),A
026E	D386	0246 OUT	



地址	目的码	语句序号	源语句	源语句
0270	1812	0247	JR	CCS2B—\$ ; 转恢复用户寄存器
0272	3E00	0248 ; SINGLE STEP	键处理程序	
0274	32F42F	0249 CCS2;	LD A, 00H	
0277	3E01	0250	LD (BELG), A	; 清除断点
0279	32F32F	0251 CCS2A;	LD A, 01H	
027C	3E07	0252	LD (SSFLG), A	; 设置单步方式标志
027E	D386	0253	LD A, 07H	
0280	3E0B	0254	OUT (CTC2), A	; 禁止 CTC2 中断
0282	D386	0255	LD A, 0BH	; 启动 CTC, 时间常数为 176
0284	FDE1	0256	OUT (CTC2), A	; 等待 NMI 中断
0286	DDE1	0257 ; 恢复寄存器;		
0288	E1	0258 CCS2B;	POP IY	
0289	D1	0259	POP IX	
028A	C1	0260	POP HL	
028B	F1	0261	POP DE	
028C	08	0262	POP BC	
028D	D9	0263	POP AF	
028E	F1	0264	EX AF, AF'	; 辅助寄存器
028F	ED47	0265	EXX	
0291	E1	0266	POP AF	; I 和 IFF
0292	D1	0267	LD I, A	; 恢复 I
0293	C1	0268	POP HL	
0294	3ADD2F	0269	POP DE	
		0270	POP BC	
		0271	LD A, (UIF)	; 取 IFF

地址	目的码	语句序号	源语句	
0297	B7	0272	OR A	; 设置状态
0298	C29E02	0273	JP NZ, CCS2C	; 转允许中断
029B	F1	0274	POP AF	
029C	F3	0275	DI	; 关中断
029D	C9	0276	RET	; 返回用户程序
029E	F1	0277	CCS2C; POP AF	
029F	FB	0278	EI	; 允许中断
02A0	C9	0279	RET	; 返回用户程序
		0280	; MON 键处理程序——强迫返回监控程序 RESTR1 (00AE)	
		0281	; 并保存用户寄存器	
02A1	C3AE00	0282	CCS3; JP RESTR1	; 清标志及进行显示处理
		0283	; NEXT 键处理程序	
		0284	; 用于选择下一个存储单元, 下一个口或	
		0285	; 在 EPROM 写入程序中的下一个错误	
02A4	3AE12F	0286	CCS4; LD A, (MFLG)	
02A7	FE01	0287	CP 01H	; 是存储器检查方式吗?
0A29	2812	0288	JR Z, CCS4B—\$	
02AB	3ADE2F	0289	LD A, (PFLG)	
02AE	FE01	0290	CP 01H	; 是口检查方式吗?
02B0	2826	0291	JR Z, CCS4C—\$	
02B2	3ADA2F	0292	LD A, (PRFLG)	
02B5	FE01	0293	CP 01H	; 是 EPROM 写入方式吗?
02B7	C2AE00	0294	CCS4A; JP NZ, RESTR1	; 否, 按无意义键处理
02BA	C33006	0295	JP CCS12D	; EPROM 写入方式有效
		0296		; 转显示下一个错误

地址	目的码	语句序号	源语句	
02BD	CD5906	0297	CALL CCS4B;	UFOR2
02C0	23	0298	INC	HL
02C1	7C	0299	LD	A, H
02C2	CD3C06	0300	CALL	UFOR1
02C5	DD23	0301	INC	IX
02C7	DD23	0302	INC	IX
02C9	7D	0303	LD	A, L
02CA	CD3C06	0304	CALL	UFOR1
02CD	DD23	0305	INC	IX
02CF	DD23	0306	INC	IX
02D1	7E	0307	LD	A, (HL)
02D2	CD3C06	0308	CALL	UFOR1
02D5	C3F400	0309	JP	DISUP
02D8	CD5906	0310	CALL	UFOR2
02DB	4C	0311	LD	C, H
02DC	0C	0312	INC	C
02DD	79	0313	LD	A, C
02DE	CD3C06	0314	CALL	UFOR1
02E1	ED78	0315	IN	A, (C)
02E3	DD21FB2F	0316	LD	IX, DSMEM4
02E7	CD3C06	0317	CALL	UFOR1
02EA	C3F400	0318	JP	DISUP
02ED	3E01	0319	辅助寄存器检查 (REG' DISP键)	
02EF	3E02F	0320	CCS5;	A, 01H
		0321	LD	(ARFLG), A

; 将 DISMEM 的前四个字节形成一个地址送 HL  
; 选择下一个存储单元地址

; IX 已经指向 DISMEM, 更新前两位数

; 更新次两位数

; 读新的存储单元内容

; 更新最后两位数

; 转显示

; 将 DISMEM 的前四个字节形成一个地址送 HL

; 仅 H 有意义

; 选择下一个口地址

; IX 已经指向 DISMEM

; 读入

; 指向最后两位数

; 更新最后两位数

; 转显示

; 置位辅助寄存器检查标志 ARFLG

地址	目的码	语句序号	源语句	
02F2	DD21F72F	0322	LD IX, DISMEM	; 指向显示缓冲区 DISMEM单元
02F6	3E12	0323	LD A, 12H	; 撇号送 A
02F8	DD7701	0324	LD (IX+1), A	; 写入显示缓冲区
02FB	CD5504	0325	CALL ALTER5	
02FE	7E	0326	LD A, (HL)	; 送入累加器
02FF	DD21FB2E	0327	LD IX, DSMEM4	; 指向显示缓冲区后两位
0303	CD3C06	0328	CALL UFOR1	; 写 A 到显示缓冲区
0306	185A	0329	JR CCS6C--\$	
0308	3E01	0330	; 主寄存器检查 (REG DISP键)	
030A	32DF2F	0331	LD A, 01H	; 置寄存器检查标志 RFLC
030D	CD6E04	0332	LD (RFLG), A	
0310	380A	0333	CALL ALTER6	
0312	7E	0334	JR C, CCS6A--\$	; 是对应键小于 6 的寄存器则转移
0313	DD21FB2F	0335	LD A, (HL)	; 取 (HL) 单元内容送 A
0317	CD3C06	0336	LD IX, DSMEM4	; 指向显示缓冲区的后两个单元
031A	1846	0337	CALL UFOR1	; 写 A 到 DSMEM 4, 5
031C	FE03	0338	JR CCS6C--\$	
031E	2835	0339	; PC, SP, IX, IY 寄存器处理	
0320	FE02	0340	CP CCS6A; CP 3	; 是键 3 吗?
0322	2817	0341	JR Z, CCS6B--\$	; 是, 为 IFF
0324	7E	0342	CP 2	; 是键 2 吗?
0325	DD21FB2F	0343	JR Z, CCS6A1--\$	; 是, 为栈指示器 SP
0329	DD22DB2F	0344	LD A, (HL)	; 低字节送 A
		0345	LD IX, DSMEM4	
		0346	LD (KEYPTR), IX	; 为写入四位新数做准备

地址	目的码	语句序号	源语句	
032D	CD3C06	0347	CALL	UFOR1 ; 第一个字节写到 DSMEM4,5
0330	23	0348	INC	HL ; 高字节送 A
0331	7E	0349	LD	A, (HL)
0332	DD21F92F	0350	LD	IX, DSMEM2 ; 指向前两位
0336	CD3C06	0351	CALL	UFOR1 ; 第二个字节写到 DSMEM2,3
0339	182D	0352	JR	CCS6D—\$ ; 转显示
033B	3AE32F	0353	CCS6A1; LD	A, (STKPT1)
033E	DD21F92F	0354	LD	IX, DSMEM2
0342	CD3C06	0355	CALL	UFOR1 ; 第一字节送 DSMEM2, 3
0345	3AE22F	0356	LD	A, (STKPT)
0348	DD21FB2F	0357	LD	IX, DSMEM4
034C	CD3C06	0358	CALL	UFOR1 ; 第二字节送 DSMEM4, 5
034F	DD22DB2F	0359	LD	(KEYPTR), IX ; 为写入四位新数作准备
0353	1813	0360	JR	CCS6D—\$
		0361	IFF 处理	
0355	DD21FB2F	0362	CCS6B; LD	IX, DSMEM4
0359	3ADD2F	0363	LD	A, (UIF) ; 取得 UIF 的值 4 或 0
035C	CD3C06	0364	CALL	UFOR1 ; 写入 DISMSM
035F	21FE2F	0365	LD	HL, DSMEM7
0362	21FD2F	0366	CCS6C; LD	HL, DSMEM6 ; 设置指针以备更换 IFF 时用
0365	22DB2F	0367	LD	(KEYPTR), HL
0368	C3F400	0368	CCS6D; JP	DISUP
		0369	口检查 (PORT EXAM键)	
036B	3AF52F	0370	CCS7; LD	A, (DIG2)
036E	FE01	0371	CP	1

地址	目的码	语句序号	源语句	
0370	2020	0372	JR	NZ, CCS7A—\$
0372	32DE2F	0373	LD	(PFLG), A
0375	3E10	0374	LD	A, 10H
0377	32F92F	0375	LD	(DSMEM2), A
037A	32FA2F	0376	LD	(DSMEM3), A
037D	CD5906	0377	CALL	UFOR2
0380	4C	0378	LD	C, H
0381	ED78	0379	IN	A, (C)
0383	DD21FB2F	0380	LD	IX, DSMEM4
0387	CD3C06	0381	CALL	UFOR1
038A	DD23	0382	INC	IX
038C	DD23	0383	INC	IX
038E	DD22DB2F	0384	LD	(KEYPTR), IX
0392	C3F400	0385	JP	DISUP
0395	3AF62F	0386	; 存储单元检查 (MEM EXAM键)	
0398	B7	0387	LD	A, (DIG4)
0399	2816	0388	OR	A
039B	32E12F	0389	JB	Z, CCS8A—\$
039E	CD5906	0390	LD	(MFLG), A
03A1	7E	0391	CALL	UFOR2
03A2	DD21FB2F	0392	LD	A, (HL)
03A6	CD3C06	0393	LD	IX, DSMEM4
03A9	DD23	0394	CALL	UFOR1
03AB	DD23	0395	INC	IX
		0396	INC	IX

; 验证是否输入了两位数

; 是, 置位口检查标志 PFLG

; 对(DSMEM2)(DSMEM3)填“空”符

; 取得口地址送 HL

; 传送前两位数到 C 寄存器

; 从该口输入

; 指向数据位 (最后两位)

; 写入显示缓冲区第 5,6 单元

; 为口内容改变作准备

; 设置口改变时的指针

; 转显示

; 验证是否输入了四位数

; 是, 置位存储检查标志 MFLG 为 1

; 四位数形成地址送 HL

; 取得存储器数据

; 指向正确的数位

; 写入显示缓冲区 (DSMEM4) (DSMEM5)

地址	目的码	语句序号	源语句	语意
03AD	DD2DB2F	0397	LD	(KEYPTR), IX ; 为更换数据准备指针
03B1	C3F400	0398	JP	DISUP
03B4	DD21F72F	0399	ALTER; LD	IX, DISMEM
03B8	3ADE2F	0400	LD	A, (PFLG)
03BB	B7	0401	OR	A,
03BC	202A	0402	JR	NZ, ALTER2—\$ ; 口内容的改变
03BE	3ADF2F	0403	LD	A, (RFLG)
03C1	B7	0404	OR	A
03C2	2035	0405	JR	NZ, ALTR3—\$ ; 寄存器内容的改变
03C4	3AE02F	0406	LD	A, (ARFLG)
03C7	B7	0407	OR	A
03C8	C24104	0408	JP	NZ, ALTR4 ; 辅助寄存器内容的改变
03CB	3AE12F	0409	LD	A, (MFLG)
03CE	B7	0410	OR	
03CF	CAAE00	0411	JP	Z, RESTR1 ; 不正确操作, 转回 RESTR1
03D2	CD5906	0412	CALL	UFOR2 ; 取得地址送 HL
03D5	DD7E06	0413	LD	A, (IX + 6) ; 取得字节的高四位 (半字节)
30D8	CD8F04	0414	CALL	ALTER7
03DB	DDB607	0415	OR	(IX + 7) ; 与字节的低四位相或
03DE	77	0416	LD	(HL), A ; 此字节写入存储器中
03DF	7E	0417	LD	A, (HL) ; 再读回 A
03E0	DD21FB2F	0418	ALTER1; LD	IX, DSMEM4
03E4	CD3C06	0419	CALL	UFOR1 ; 把新的数据写入显示缓冲区 (DSMEM4)
03E7	C9	0420	RET	(DSMEM5)
03E8	CD5906	0421	ALTER2; CALL	UFOR2 ; 取得的地址送 HL

地址	目的码	语句序号	源语句	
03EB	DD7E06	0422	LD A, (IX + 6)	; 取得字节的高四位
03EE	CD8E04	0423	CALL ALTER7	
03F1	DD8607	0424	OR (IX + 7)	; 与字节的低四位相或
03F4	4C	0425	LD C, H	
03F5	ED79	0426	OUT (C), A	; 写入口中
03F7	18E7	0427	JR ALTER1—\$	
0428主寄存器改变内容				
03F9	CD6E04	0429	ALTR3: CALL ALTER6	
03FC	380D	0430	JR C, ALTR3A—\$	; 是键号小于6的16位寄存器和1FF吗?
03FE	DD7E06	0431	LD A, (IX + 6)	; 取得字节的高四位
0401	CD8E04	0432	CALL ALTER7	
0404	DDB607	0433	OR (IX + 7)	; 与字节的低四位相或
0407	77	0434	LD (HL), A	; 修改保护在寄存器存放区内的内容
0408	C3E003	0435	JP ALTER1	
040B	FE03	0436	ALTR3A: CP 3	; 是键3吗?
040D	2824	0437	JR Z, ALTR3B—\$	; 是, 1FF
040F	FE02	0438	CP 2	; 是键2吗?
0411	282A	0439	JR Z, ALTR3C—\$	; 是, SP
0413	DD7E04	0440	LD A, (IX + 4)	; 取字节的高四位
0416	CD8E04	0441	CALL ALTER7	
0419	DDB605	0442	OR (IX + 5)	; 与字节的低四位相或
041C	23	0443	INC HL	
041D	77	0444	LD (HL), A	; 修改高字节
041E	DD21F92F	0445	LD IX, DSMEM2	
0422	CD3C06	0446	CALL UPOR1	; 写入数据到显示缓冲区



地址	目的码	语句序号	源语句
0425	DD7E04	0447	LD A, (IX + 4) ; 取字节的高四位
0428	CD8E04	0448	CALL ALTER 7 ; 与字节的低四位相或
042B	DDB605	0449	OR (IX + 5) ; 与字节的低四位相或
042E	2B	0450	DEC HL ; 修改低字节
042F	77	0451	LD (HL), A ; 修改低字节
0430	C3E003	0452	JP ALTER 1 ; 获得新值
0433	3AFE2F	0453	ALTR3B; LD A, (DSMEM7), IX ; 获得新值
0436	32DD2F	0454	LD (UIF), A ; 修改 UIF
0439	32FC2F	0455	LD (DSMEM5), A ; 写入显示缓冲区
043C	C9	0456	RET ; 哑指令
043D	E1	0457	SP 的内容是不容许改变的
043E	C3AE00	0458	ALTR3C; POP HL ; 哑指令
		0459	JP RESTR1 ; 哑指令
0441	DD21F72F	0460	辅助寄存器改变内容
0445	CD5504	0461	ALTR4; LD IX, DISMEM ; 指向存放在 DISMEM, 其中存有代表寄存器的键号
0448	DD7E06	0462	CALL ALTER 5 ; 取字节的高四位
044B	CD8E04	0463	LD A, (IX + 6) ; 取字节的高四位
044E	DDB607	0464	CALL ALTER 7 ; 与字节的低四位相或
0451	77	0465	OR (IX + 7) ; 与字节的低四位相或
0452	C3E003	0466	LD (HL), A ; 修改在寄存器存放区内的内容
0455	DD5E00	0467	JP ALTER 1 ; 转写新数据到显示缓冲区
0458	0600	0468	ALTER5; LD E, (IX + 0) ; 取存于 DISMEM, 其中存有代表寄存器的键号
045A	1600	0469	LD B, 00H ; 取存于 DISMEM, 其中存有代表寄存器的键号
045C	21E507	0470	LD D, 00H ; 取存于 DISMEM, 其中存有代表寄存器的键号
		0471	LD HL, REGTBP ; 指向辅助寄存器表

地址	目的码	语句序号	源语句
045F	19	0472	HL, DE ; 加键值偏移量
0460	4E	LD	C, (HL) ; 取值
0461	79	LD	A, C
0462	FE19	CP	19H ; 是非法值吗?
0464	CAAE00	JP	Z, RESTR1 ; 是, 重新启动监控程序
0467	2AE22F	LD	HL, (STKPT) ; 否, 指向用户 SP
046A	B7	OR	A ; 清进位位
046B	ED42	SBC	HL, BC ; 从SP 减去偏移量
046D	C9	RET	
046E	DD21F72F	LD	IX, DISMEM ; 指向 DISMEM, 其中存有代表寄存器的键号
0472	DD5E00	LD	E, (IX) ; 得到代表寄存器的键号
0475	0600	LD	B, 00H
0477	1600	LD	D, 00H
0479	21D507	LD	HL, REGTB ; 指向寄存器表 REGTB
047C	19	ADD	HL, DE ; 加键值偏移量
047D	4E	LD	C, (HL) ; 得到表中相应项的内容
047E	79	LD	A, C
047F	FE19	CP	19H ; 是非法值 (即 25D) 吗?
0481	CAAE00	JP	Z, RESTR1 ; 是, 重新启动
0484	2AE22F	LD	HL, (STKPT) ; 否, 指向用户 SP
0487	7B	LD	A, E ; 重新得到键值
0488	B7	OR	A ; 清进位位
0489	ED42	SBC	HL, BC ; 从用户 SP 减去表中相应项内容
048B	FE06	CP	6 ; 判断是何寄存器
048D	C9	RET	

地址	目的码	源语句	源语句
048E	CB27	0497 ALTER7: SLA	A
0490	CB27	0498	SLA A
0492	CB27	0499	SLA A
0494	CB27	0500	SLA A
0496	C9	0501	RET
		0502;	断点的设置
		0503;	如断点表还有空间就在断点表中设置一个入口
		0504;	实际上断点是在执行命令中才放入 RAM 之中的
		0505	
0497	3AF62F	0506 CCS9:	LD A, (DIG4) ; 取标志
049A	B7	0507	OR A ; 设置状态
049B	2005	0508	JR NZ, CCS90-\$ ; 输入了四位数吗?
049D	32F42F	0509	LD (BFLG), A ; 否, 则清除所有的断点
04A0	1810	0510	JR CCS91-\$
04A2	CD5906	0511 CCS90:	CALL UFOR2 ; 得到断点地址在 HL
04A5	CD7C06	0512	CALL UFOR3 ; 得到断点状态
04A8	2810	0513	JR Z, CCS9B-\$ ; 无断点, 转去在表内设置一个断点
04AA	78	0514	LD A, B
04AB	FE05	0515	CP 05H ; 表满了吗?
04AD	2006	0516	JR NZ, CCS9A-\$ ; 否, 继续
04AF	C3AE00	0517	JP RESTR1 ; 是, 清显示
04B2	C3F400	0518 CCS91:	JP DISUP
04B5	CD3406	0519 ;	在表中找到第一个未使用的空间
04B8	20FB	0520 CCS9A:	CALL UIX3 ; IX + 3和B - 1
		0521	JR NZ, CCS9A-\$ ; 返回, 直到找到自由空间

地址	目的码	语句序号	源语句
04BA	3AF42F	0522	在表内插入新断点
04BD	3C	0523	CCS9B; LD A, (BFLG)
04BE	32F42F	0524	INC A ; 标志 BFLG 增1
04C1	DD7400	0525	LD (BELG), A
04C4	DD7501	0526	LD (IX), H ; 把断点地址写入断点表内
04C7	C3F400	0527	LD (IX + 1), L
		0528	JP DISUP
		0529	按 KC 格式将信息转储到盒式磁带
		0530	起始地址在 DE, 结束地址在 HL
		0531	在此程序中, 中断始终有效
04CA	CDA706	0532	CCS10; CALL UFOR4 ; 清显示
04CD	3E01	0533	LD A, 1
04CF	32D92F	0534	LD (FLG24), A ; 为逻辑 1 而设置 FLG24 标志为 1
04D2	ED5E	0535	IM 2
04D4	01FA07	0536	LD BC, CTC1P
04D7	78	0537	LD A, B
04D8	ED47	0538	LD I, A
04DA	79	0539	LD A, C
04DB	D384	0540	OUT (CTC0), A ; 中断矢量
04DD	3E85	0541	LD A, 85H
04DF	D385	0542	OUT (CTC1), A ; CTC 中断允许
04D1	3E1A	0543	LD A, 1AH
04D3	D385	0544	OUT (CTC1), A ; 送时间常数
04E5	3AC02F	0545	LD A, (PUNHSH)
04E8	57	0546	LD D, A

地址	目的码	语句序号	源语句	
04E9	3AC12F	0547	LD	A, (PUNHSL) ; 得到起始地址
04EC	5F	0548	LD	E, A
04FD	3AC22F	0549	LD	A, (PUNHEH)
04F0	67	0550	LD	H, A
04F1	3AC32F	0551	LD	A, (PUNHEL) ; 得到结束地址
04F4	6F	0552	LD	L, A
04F5	AF	0553	XOR	A ; 清进位位
04F6	ED52	0554	SBC	HL, DE ; 计算数据块大小
04F8	23	0555	INC	HL
04F9	E5	0556	PUSH	HL ; 计算结果存在 HL 中
04FA	210000	0557	LD	HL, 0000H
04FD	0603	0558	LD	B, 03H ; 设置计数值以获得40秒延时
04FF	FB	0559	EI	
0500	76	0560	CCS10A, HALT	; 等待 CTC1 中断
0501	2D	0562	DEC	L
0502	20FC	0563	JR	NZ, CCS10A-\$
0504	25	0564	DEC	H
0505	20F9	0565	JR	NZ, CCS10A-\$ ; 延时循环
0507	10F7	0566	DJNZ	CCS10A-\$ ; 40 秒全1 开头段
0509	3E3A	0567	CCS10B; LD	A, 03AH ; 以冒号开始
050B	CDF706	0568	CALL	OTCHR1 ; 输出冒号
050E	AF	0569	XOR	A ; 清进位位
050F	011000	0570	LD	BC, 10H
0512	E1	0571	POP	HL

地址	目的码	语句序号	源语句	HL, BC	HL 和最大字节数 16 相比
0513	ED42	0572	SBC	HL, BC	
0515	3009	0573	JR	NC, CCS10C—\$	; NC = >HL > 10H
0517	09	0574	ADD	HL, BC	; 当少于16个字节时, 恢复数据块字节数
0518	85	0575	ADD	A, L	; 如L=0, 就会设置状态标志
0519	47	0576	LD	B, A	
051A	2E00	0577	LD	L, 0	
051C	283C	0578	JR	Z, CCS10F—\$	
051E	1801	0579	JR	CCS10D—\$	
0520	41	0580	CCS10C; LD	B, C	; B = 数据块大小
0521	E5	0581	CCS10D; PUSH	HL	; 保护进栈
0522	0E00	0582	LD	C, 0	; 清检查和
0524	78	0583	LD	A, B	; 转储记录长度
0525	CDD506	0584	CALL	UPACCS	
0528	3AC02F	0585	LD	A, (PUNHSH)	
052B	67	0586	LD	H, A	
052C	CDD506	0587	CALL	UPACCS	
052F	3AC02F	0588	LD	A, (PUNHSL)	; 转储起始地址低字节
0532	6F	0589	LD	L, A	
0533	CDD506	0590	CALL	UPACCS	
0536	AF	0591	XOR	A	; 为转储“记录类型”置累加器为0
0537	CDD506	0592	CALL	UPACCS	
053A	7E	0593	CCS10E; LD	A, (HL)	; 将被转储的数据送累加器 A
053B	CDD506	0594	CALL	UPACCS	
053E	23	0595	INC	HL	
053F	10F9	0596	DJNZ	CCS10E—\$	

地址	目的码	语句序号	源语句
0541	97	0597	SUB A
0542	91	0598	SUB C
0543	CDD506	0599	UPACCS ; 转储“检查和”
0546	3E0D	0600	A, 0DH ; 回车字符
0548	CDD506	0601	UPACCS
054B	3E0A	0602	A, 0AH ; 换行字符
054D	CDD506	0603	UPACCS
0550	7C	0604	A, H
0551	32C02F	0605	(PUNHSH), A ; 保存地址高字节
0554	7D	0606	A, L
0555	32C12F	0607	(PUNHSL), A ; 保存地址低字节
0558	18AF	0608	JR CCS10B--\$ ; 继续转储更多的数据
055A	0603	0609	CCS10F; LD B, 3
055C	AF	0610	CCS10G; XOR A
055D	4F	0611	LD C, A
055E	CDD506	0612	CALL UPACCS ; 转储“文件结束”
0591	10F9	0613	DJNZ CCS10G--\$
0563	3E01	0614	LD A, 1
0565	CDD506	0615	CALL UPACCS
0568	97	0616	SUB A
0569	91	0617	SUB C
056A	CDD506	0618	CALL UPACCS ; 转储“文件结束”的检查和
056D	21FFFF	0619	CCS10H; LD HL, 0FFFFH ; 设置5秒全1结尾的计数值
0570	FB	0620	EI ; 开中断
0571	76	0621	HALT

地码	目的码	语句序号	源语句
0572	2D	0622 ; ※※※※※※※※※※	
0573	20FD	0623 CCS10J; DEC	L
0575	25	0624 JR	NZ, CCS10J—\$
0576	20FA	0625 DEC	H
0578	F3	0626 JR	NZ, CCS10J—\$
0579	3F03	0627 DI	
057B	D385	0628 LD	A, 03H
057D	F3	0629 OUT	(CTC1), A ; 关 CTC, 停止发出脉冲冲波
057F	C3AE00	0630 DI	; 关中断
		0631 JP	RESTR1 ; 转重新启动监控程序
0581	ED5E	0632 ; 从盒式磁带向存储器输入数据 ( Kansas city 格式)	
0583	21FF0F	0633 CCS11, IM	2 ; 选择中断方式 2
0586	0620	0634 LD	HL, 0FFFH ; 开始15秒延时
0588	2D	0635 LD	B, 20H
0589	20FD	0636 CCS11A; DEC	L
058B	25	0637 JR	NZ, CCS11A—\$
058C	20FA	0638 DEC	H
058E	10F8	0639 JR	NZ, CCS11A—\$
		0640 DJNZ	CCS11A—\$
0590	CD5807	0641 ; 磁带现在进入加载程序部分—启动主循环	
0593	D63A	0642 CCS11G; GALL	INCHR ; 取得第一个字
0595	20F9	0643 SUB	03AH ; 检测是否是冒号
0597	4F	0644 JR	NZ, CCS11G—\$ ; 循环返回
0598	CDDB06	0645 LD	C, A ; “检查和”预置 0
		0646 CALL	ULACC ; 收到记录长度及其检查和



地址	目的码	语句序号	源语句
059B	47	0647	LD B, A
059C	CDDB06	0648	CALL ULACC ; 收到地址的高字节
059F	57	0649	LD D, A
05A0	CDDB06	0650	CALL ULACC ; 收到地址的低字节
05A3	5F	0651	LD E, A
05A4	CDDB06	0652	CALL ULACC ; 收到记录类型
05A7	3D	0653	DEC A ; 记录类型为“文件结束”吗?
05A8	F5	0654	PUSH AF
05A9	2807	0655	JR Z, CCS11J-\$ ; 如是则转移
05AB	CDDB06	0656	CALL CCS11H; ULACC ; 不是, 接收数据
05AE	12	0657	LD (DE), A ; 存储数据
05AF	13	0658	INC DE ; 为接收更多数据而循环
05B0	10F9	0659	DJNZ CCS11H-\$
05B2	CDDB06	0660	CALL CCS11J; ULACC ; 接收“检查和”
05B5	AF	0661	XOR A ; 清累加器
05B6	81	0662	ADD A, C
05B7	2814	0663	JR Z, CCS11K-\$ ; 用检查和校验无误
05B9	DD21F72F	0664 ;	指示接收差错
05BD	7A	0665	LD IX, DISMEM ; 指向显示缓冲区
05BE	CD3C06	0666	LD A, D
05C1	DD21F92F	0667	CALL UFOR1 ; 显示高字节
05C5	7B	0668	LD IX, DSMEM2
05C6	CD3C06	0669	LD A, E
05C9	F1	0670	CALL UFOR1 ; 显示低字节
		0671	POP AF ; 恢复堆栈指针 SP

地址	目的码	语句序号	源语句
05CA	C3F400	0672	JP DISUP ; 转显示
05CD	F1	0673	CCS11K, POP AF ; 判断是否为“文件结束”
05DE	20C0	0674	JP NZ, CCS11G—\$ ; 转入下一个记录的循环
05D0	C3AE00	0675	JP RESTR1 ; 重新启动监控程序
		0676	; 用于 2758 和 2716 EPROM 的 EPROM 编程器
		0677	; 从以 2000H 为起始地址的 RAM 传送数据
		0678	; 到起始地址为 1000H 的 EPROM
		0679	; 被传送的字节数 (由四个十六进制数表示)
		0680	; 存放在显示缓冲区内
05D3	3E01	0681	CCS12, LD A, 01H
05D5	32DA2F	0682	LD (PRFLG), A ; EPROM 编程标志置位
05D8	CD5906	0683	CALL UFOR2 ; 得到字节数
05DB	E5	0684	PUSH HL ; 保护
05DC	C1	0685	POP BC
05DD	E5	0686	PUSH HL
05DE	210020	0687	LD HL, 2000H ; RAM 源地址送 HL
05E1	110010	0688	LD DE, 1000H ; EPROM 目的地址送 DE
05E4	3E25	0689	CCS12A; LD A, 25H ; CTC 用于 26ms 定时输出 ZC/T02 不产生中断
05E6	D386	0690	OUT (CTC2), A
05E8	3ECB	0691	LD A, 0CBH ; 0CBH = 203D
05EA	D386	0692	OUT (CTC2), A ; 时间常数
05EC	3E80	0693	LC A, 80H ; 清显示
05EE	D38C	0694	OUT (DIGLH), A ; 发出 EPROM 编程允许脉冲(PGM PULSE ENABLE)
05F0	EDA0	0695	LDI ; 插入等待状态直到 CTC2 定时输出两次
05F2	3E00	0696	LD A, 00H

地址	目的码	语句序号	源语句
05F4	D38C	0697	OUT (DIGLH), A ; 清 EPROM 编程允许脉冲冲
05F6	3E03	0698	LD A, 03H ; CTC2 复位
05F8	D386	0699	OUT (CTC2), A
05FA	EAE405	0700	JP PE, CCS12A ; 如 BC-1 ≠ 0, 循环返回
05FD	C1	0702	POP BC ; 恢复字节数
05FE	210020	0703	LD HL, 2000H ; HL 指向 RAM
0601	110010	0704	LD DE, 1000H ; DE 指向 EPROM
0604	1A	0705	LD A, (DE) ; 取 EPROM 数据
0605	EDA1	0706	CPI ; 与 RAM 比较
0607	2006	0707	JR NZ, CCS12C--\$ ; 有错, 指出错误
0609	E2AE00	0708	JP PO, RESTR1 ; 无错—返回
060C	13	0709	INC DE ; 更新 DE
060D	18F5	0710	JR CCS12B--\$ ; 检查下一个字节
060F	F5	0711	错误处理程序
0610	C5	0712	CCS12C; PUSH AF ; 保护错误数据
0611	D5	0713	PUSH BC ; 保护字节计数
0612	D9	0714	PUSH DE ; 保护错误地址
0613	D1	0715	PXX ; 保护主寄存器
0614	C1	0716	POP DE ; DE 为 EPROM 出错处地址
0615	DD21F72F	0717	POP BC ; BC 为字节计数
0619	7A	0718	LD IX, DISMEM
061A	CD3C06	0719	LD A, D
061D	DD21F92F	0720	CALL UFOR1 ; 高字节送显示缓冲区
		0721	LD IX, DSMEM2

地址	目的码	语句序号	源语句	
0621	7B	0722	LD A, F	
0622	CD3C06	0723	CALL UFOR1	; 低字节送显示缓冲区
0625	DD21FB2F	0724	LD IX, DSMEM2	
0629	F1	0725	POP A, E	; 取回错误数据
062A	CD3C06	0726	CALL UFOR1	; 错误数据送显示缓冲区
062D	C3F400	0727	JP DISUP	; 显示
0630	D9	0728	在错误显示期间 NEXT 键输入	
0631	13	0729	CCS12D; EXX	; 取得返回的各方针
0632	18D0	0730	INC DE	; 为测试下一个错误而增1
		0731	JR CCS12B—\$	; 为测试其它错误而循环
		0732	*** ** * 赋值	
>008C		0733	DIGLH; EQU 8CH	; 只写, 数位选择输出出口
>0090		0734	KBSEL; EQU 90H	; 只读, 键盘选择输入入口
>0084		0735	CTC0; EQU 84H	; 中断矢量
>0085		0736	CTC1; EQU 85H	; 磁带接口—转储
>0086		0737	CTC2; EQU 86H	; 单步/EPROM 编程器
>0087		0738	CTC3; EQU 87H	; 磁带接口—输入
>0088		0739	SEGLH; EQU 88H	; 只写, 字形锁存器输出出口
		0740	END	

地址	目的码	语句序号	源语句
		0002	程序名称 UTIL
		0003	; 实用程序块
		0004	; 版本1.6 2/24/80
>0634		0005	ORG 0634H
		0006	PSECT REL
		0007	GLOBAL DIG2
		0008	GLOBAL DIG4
		0009	GLOBAL DIG8
		0010	GLOBAL UIX3
		0011	GLOBAL UFOR1
		0012	GLOBAL D20MS
		0013	GLOBAL UFOR2
		0014	GLOBAL UFOR3
		0015	GLOBAL UFOR4
		0016	GLOBAL BPTAB
		0017	GLOBAL SSFLG
		0018	GLOBAL UABIN
		0019	GLOBAL UBASC
		0020	GLOBAL UPACC
		0021	GLOBAL UPACCS
		0022	GLOBAL ULACC
		0023	GLOBAL OTCHR
		0024	GLOBAL KYTBL
		0025	GLOBAL REGTB
		0026	GLOBAL REGTBP

地址	目的码	语句序号	源语句
		0027	GLOBAL DISMEM
		0028	GLOBAL STKPT
		0029	GLOBAL SECTP
		0030	GLOBAL KEYPTR
		0031	GLOBAL OTCHR
		0032	GLOBAL OTCHR1
		0033	GLOBAL INCHR
		0034	GLOBAL FLG24
		0035	GLOBAL RFLG
		0036	GLOBAL ARFLG
		0037	GLOBAL BFLG
		0038	GLOBAL PFLG
		0039	GLOBAL CTC1P
		0040	GLOBAL MFLG
		0041	GLOBAL PRFLG
		0042	GLOBAL CTC3L
		0043	GLOBAL CTC1P
		0044	GLOBAL UEGCR
		0045	GLOBAL OTCHR6
		0046	GLOBAL INCHR4
		0047	IX加3而B减1
		0048	用于查断点表, 该表存有
		0049	断点地址和用户操作码
0634	DD 23	0050	UIX 3; INC IX
0636	DD 23	0051	INC IX

地址	目的码	源语句
0638	DD23	INC IX
063A	05	DEC B
063B	C9	RET ; 转用户检测断点溢出
0655		IX 指向显示缓冲区的下一个和下一个要写入的单元
0656		写入的数据来自累加器
0657		UFOR1 执行这一写操作
0658	UFOR1	LD B, A ; 保护A的内容
0659	AND FH	; 屏蔽掉字节的高四位
0660	LD (IX+1), A	; 写入字节的低四位
0661	LD A, B	; 取回完整字节
0662	SRL A	
0663	SRL A	
0664	SRL A	
0665	SRL A	; 字节的高四位移至低四位
0666	LD (IX+0), A	; 将字节的高四位写到显示缓冲区
0667	RET	
0668		延时 20 毫秒后返回
0669	D20MS;	LD HL, 08FFH
0670	D20MS1;	DEC L
0671	JR NZ, D20MS1--\$	
0672	DEC H	
0673	JR NZ, D20MS1--\$	
0674	RET	
0675		将在显示缓冲区的前四位数组成地址置于 HL 中,
0676		用于存储器口的更新命令, 设置断点时也用到

地址	目的码	语句序号	源语句
0659	DD21F77F	0077	UFOR2, LD IX, DISMEM ; 指向 DISMEM
065D	DD7E00	0078	LD A, (IX) ; 取得第一位数
0660	CB27	0079	SLA A
0662	CB27	0080	SLA A
0664	CB27	0081	SLA A
0666	CB27	0082	SLA A
0668	DDB601	0083	OR (IX+1) ; 在 A 内移到高四位
066B	67	0084	LD H, A ; 引入低四位
066C	DD7E02	0085	LD A, (IX+2) ; 存入 H
066E	CB27	0086	SLA A ; 取来第二字节的高四位
0671	CB27	0087	SLA A
0673	CB27	0088	SLA A
0675	CB27	0089	SLA A
0677	DDB603	0090	OR (IX+3) ; 引入低四位
067A	6F	0091	LD L, A ; 完成指针 HL 的建立
067B	C9	0092	RET
067C	DD21E42F	0093;	置 IX 断点表地址而(BFLG)标志放入 B 中
0680	3AF42F	0094	UFOR3; LD IX, BPTAB ; IX 指向表的首地址
0683	B7	0095	LD A, (BFLG) ; 有效的断点数目
0684	47	0096	OR A ; 设置状态
0685	C9	0097	LD B, A ; 转用户测试状态标志
		0098	RET
		0099;	标志初始化程序
0686	21F72F	0100	UFGCR; LD HL, DISMFM
0689	22DB2F	0101	LD (KEYPTR, HL); 指向显示缓冲区



地址	目的码	语句序号	源语句
068C	3E00	0102	LD A, 00H
068E	32E52F	0103	LD (DIG2), A ; 各个标志置 0
0691	32F62F	0104	LD (DIG4), A
0694	32F32F	0105	LD (SSFLG), A
0697	32DA2F	0106	LD (PRFLG), A
069A	32DE2F	0107	LD (PFLG), A
069D	32E12F	0108	LD (MFLG), A
06A0	32DF2F	0109	LD (RFLG), A
06A3	32E02F	0110	LD (ARFLG)A
06A6	C9	0111	RET
		0112	将空码送入显示缓冲区——DISMEM
06A7	0608	0113	UFOR4: LD B, 08H
06A9	21F72F	0114	LD HL, DISMEM
06AC	3E10	0115	LD A, 10H ; 不显示字的空码送 A
06AE	77	0116	UFOR4A:LD (HL), A
06AF	23	0117	INC HL
06B0	10FC	0118	DJNZ UFOR4A—\$ ; 循环送空码直到送完为止
06B2	C9	0119	RET
		0120	将累加器中的一个 ASCII 码数转换为二进制数
06B3	D630	0121	UABIN: SUB 30H
06B5	FE0A	0122	CP 10
06B7	F8	0123	RET M
06B8	D607	0124	SUB 7
06BA	C9	0125	RET
		0126	将累加器中的一个 4 位二进制数转换为 ASCII 码

地址	目的码	语句序号	源语句
06BB	E60F	0127	UBASC: AND 0FH ; 屏蔽掉字节的高四位
06BD	C690	0128	ADD A, 90H
06BF	27	0129	DAA
06C0	CE40	0130	ADC A, 40H
06C2	27	0131	DAA
06C3	C9	0132	RET
06C4	D9	0133	将累加器内容按两个字进行转储——首先转储高四位
06C5	F5	0134	UPACC: EXX ; OTCHR 使用辅助寄存器
06C6	0F	0135	PUSH AF ; 保护累加器
06C7	0F	0136	RRCA
06C8	0F	0137	RRCA
06C9	0F	0138	RRCA
06CA	CDF406	0139	RRCA
06CD	E1	0140	CALL OTCHR
06CE	E60F	0141	POP AF
06D0	CDF406	0142	AND 0FH ; 屏蔽掉高四位
06D3	D9	0143	CALL OTCHR
06D4	C9	0144	EXX ; 恢复寄存器
		0145	RET
06D5	F5	0146	求检查和, 而后把累加器内容按两字节转储
06D6	81	0147	UPACCS: PUSH AF ; 保护累加器
06D7	4F	0148	ADD A, C ; 求检查和而后送入 C 寄存器
06D8	F1	0149	LD C, A
06D9	18E9	0150	POP AF
		0151	JR UPACC-\$ ; 累加器内容转储

地址	目的码	源语句	话句序号
06DB	C5	PUSH BC	0152 ; 从磁带中读两个字符并转换为二进制数, 这个二进制数存放在 A 累加器中, 并将求得的检查和保存在 C 寄存器
06DC	CD5807	INCHR	0153 ; 保护 C 寄存器 (检查和)
06DF	CDB306	CALL UABIN	0154 ; 读字符的第一位
06E2	07	RLCA	0155 ; 转换为二进制数
06E3	07	RLCA	
06E4	07	RLCA	
06E5	07	RLCA	
06E6	4F	LD C, A	0156 ; 移位到高四位
06E7	CD5807	CALL INCHR	0157 ; 保留第一位数
06EA	CDB306	CALL UABIN	0158 ; 取第二位数
06ED	B1	OR C	0159 ; 转换成二进制数
06EE	C1	POP BC	0160 ; 合并两个四位
06EF	F5	PUSH AF	0161 ; 取回检查和
06F0	81	ADD A, C	0162 ; 保护累加器内容
06F1	4F	LD C, A	0163 ; 将八位二进制数加到检查和
06F2	F1	POP AF	0164 ; 存入 C 寄存器
06F3	C9	RET	0165 ; 取回累加器内容
			0171 ; 程序使用 CTC 的通道 1 作为基本时基 (4800Hz 或 2400Hz)
			0172 ; 人口是: 放一个二进制数在累加器的低四位, 本程序将这个二进制数转换为 ASCII, 并将此码和一个起始位以
			0173 ; 及两个停止位一起发出, 从 CTC1 的输出端 (ZC/T0)
			0174 ; 输出的是两倍于 Kansas City 标准频率的频率 (1 为 4800Hz, 0 为 2400Hz)。

地址	目的码	语句序号	源语句	
06F4	CDBB06	0177	OTCHR: CALL UBASC	; 转换为A CSII 码
06F7	FB	0178	OTCHR1: EI	
06F8	57	0179	LD D, A	; 累加器的内容送 D 寄存器
06F9	3F10	0180	LD A, 10H	; 为 CTC 中断将 A 予置初值
06FB	2E0A	0181	LD L, 0AH	; 字位的计数
06FD	CB12	0182	RL D	; 进位位送 D0
06FF	FE01	0183	OTCHR2: CP 01H	; A = 1 吗? 等待到 A = 1 为止
0701	20FC	0184	JR NZ, OTCHR2-\$	; 等待 CTC 下一次计数到 0, 直到最 后一次 (A = 1) 为止
0703	47	0185	LD B, A	; 保存累加器内容
0704	3E00	0186	LD A, 00H	
0706	32D92F	0187	LD (FLG24), A	; 为发出起始位而清标志 (FLG24)
0709	78	0188	LD A, B	
070A	76	0189	OTCHR3: HALT	
		0190	*** ** ** ** **	
070B	37	0191	SCF	; 设置进位位
070C	CB1A	0192	RR D	; 右移 D 寄存器一次
070E	2D	0193	DEC L	
070F	2007	0194	JR NZ, OTCHR4-\$	
0711	3E01	0195	LD A, 01H	
0713	32D92F	0196	LD (FLG24), A	; 字发完——跟随发 1
0716	F3	0197	DI	; 出口, 禁止中断
0717	C9	0198	RET	
0718	FE01	0199	OTCHR4: CP 1	; 反复等待下一次计数到 0, 直到最后一次 (A = 1) 为止。

地址	目的码	语句序号	源语句
071A	20FC	0200	JR NZ, OTCHR4-\$ ; 还未到, 则等待
071C	CB42	0201	BIT 0, D ; 测试下一位
071E	2009	0202	JR NZ, OTCHR5-\$ ; 下一位是1
0720	47	0203	LD B, A
0721	3E00	0204	LD A, 00H ; 下一位是0, 将0送入累加器
0723	32D92F	0205	LD (FLG24), A ; 清 (FLG24) 标志
0726	78	0206	LD A, B
0727	18E1	0207	JR OTCHR3-\$
0729	47	0208 OTCHR5:	LD B, A
072A	3E01	0209	LD A, I
072C	32D92F	0210	LD (FLG24), A ; 置 (FLG24) 标志为1
072F	78	0211	LD A, B ; 恢复累加器
0730	18D8	0212	JR OTCHR3-\$ ; 等待字符存储结束
0732	3D	0213 ;	在转储期间的 CTC1 中断服务程序
0733	2020	0214 OTCHR6:	DEC A ; A 为周期计数器
0735	DD21D92F	0215	JR NZ, OTCHR8-\$ ; 计数未完, 转中断返回
0739	DDCB0046	0216	LD IX, FLG24
073D	200C	0217	BIT 0, (IX+0) ; 测试 (FLG24) 标志
073F	3E85	0218	JR NZ, OTCHR7-\$ ; (FLG24) 为0 则不转移
0741	D385	0219	LD A, 85H
0743	3E34	0220	OUT (CTC1), A ; 新控制字中断有效
0745	D385	0221	LD A, 34H
0747	3E08	0222	OUT (CTC1), A ; 为 1200Hz 设时间常数
0749	180A	0223	LD A, 8 ; 计数 8 个周期
		0224	JR OTCHR8-\$ ; 返回

地址	目的码	语句序号	源语句
074B	3E85	0225	OTCHR7: LD A, 85H
074D	D385	0226	OUT (CTC1), A ; 新控制字
074F	3E1A	0227	LD A, 1AH
0751	D385	0228	OUT (CTC1), A ; 为2400Hz设时间常数
0753	3E10	0229	LD A, 10H ; 计数16个周期
0755	FB	0230	OTCHR8: EI
0756	ED4D	0231	RETI
		0232	输入的位率已在 (BITRT) 范围内均分
		0233	在出口: 字存在累加器内
		0234	使用的寄存器是: A, B, H
		0235	允许中断且 CTC1 按位率中断
0758	21FE07	0236	INCHR: LD HL, CTC3L
075B	7C	0237	LD A, H ; 设置 I 寄存器
075C	ED47	0238	LD I, A ; CTC 中断矢量
075E	7D	0239	LD A, L ; 一个字的位计数值
075F	D384	0240	OUT (CTC0)
0761	0608	0241	INCHR1: LD B, 08H
0763	FB	0242	EI
0764	3E00	0243	LD A, 0 ; 清字
0766	2600	0244	LD H, 0 ; 输入数据
0768	DB90	0245	INCHIA: IN A, (KBSEL)
076A	CB7F	0246	BIT 7, A
076C	20FA	0247	JR NZ, INCHIA-\$ ; 为找到起始位“0”而循环返回
076E	3EA5	0248	LD A, 0A5H ; 设中断, 定标系数为 256
0770	D387	0249	OUT (CTC3), A ; CTC3 控制字



地址	目的码	语句序号	源语句
0797	3E03	0274	INCHR3: LD A, 03H
0799	D387	0275	OUT (CTC3), A ; CTC3 复位, 甄别错误
079B	18C4	0276	JR INCHR1-\$ ; 为接收另外的字符循环返回
079D	DB90	0277	; 在输入期间 CTC 中断服务程序
079F	FB	0278	INCHR4: IN A, (KBSEL) ; 输入数据
07A0	ED4D	0279	EI
07A2	CB0C	0280	RETI
07A4	18DD	0281	INCHR5: RRC H
		0282	JR INCHR2-\$ ; 等待下一位
		0283	***赋值表*** ; ***
>0084		0284	; CTC 读为取减一计数器的值, CTC 的写为设置计数器
>0085		0285	CTC0: EQU 84H ; 为用户使用
>0086		0286	CTC1: EQU 85H ; 音频磁带—信息转储
>0087		0287	CTC2: EQU 86H ; 单步和 EPROM 编程器
>0090		0288	CTC3: EQU 87H ; 音频磁带—磁带输入
		0289	KBSEL EQU 90H ; 磁带数据
		0290	; ***七段显示字形
07A6	40	0291	SEGPT DEFB 40H ; 0
07A7	79	0292	DEFB 79H ; 1
07A8	24	0293	DEFB 24H ; 2
07A9	30	0294	DEFB 30H ; 3
07AA	19	0295	DEFB 19H ; 4
07AB	12	0296	DEFB 12H ; 5
07AC	02	0297	DEFB 02H ; 6
07AD	78	0298	DEFB 78H ; 7



地址	目的码	语句序号	源语句	
07AE	00	0229	DEFB	00H ; 8
07AF	18	0300	DEFB	18H ; 9
07B0	08	0301	DEFB	08H ; A
07B1	03	0302	DEFB	03H ; b
07B2	46	0303	DEFB	46H ; C
07B3	21	0304	DEFB	21H ; d
07B4	06	0305	DEFB	06H ; E
07B5	0E	0306	DEFB	0EH ; F
07B6	7F	0307	DEFB	7FH ; 空 (不显示)
07B7	3E	0308	DEFB	3FH ; 提示符“-”
07B8	7D	0309	DEFB	7DH ; 撇号标记“/”
07B9	FF	0310 ; ※※※查找键值表	0FFH	; 0 B=01, A=0F
07BA	EF	0311 KYTBL: DEFB	0EFH	; 1 B=02, A=0F
07BB	F7	0312 DEFB	0F7H	; 2 B=02, A=17
07BC	FB	0313 DEFB	0FBH	; 3 B=02, A=1B
07BD	DF	0314 DEFB	0DFH	; 4 B=04, A=0F
07BE	E7	0315 DEFB	0E7H	; 5 B=04, A=17
07BF	EB	0316 DEFB	0EBH	; 6 B=04, A=1B
07C0	CF	0317 DEFB	0CFH	; 7 B=08, A=0F
07C1	D7	0318 DEFB	0D7H	; 8 B=08, A=17
07C2	DB	0319 DEFB	0DBH	; 9 B=08, A=1B
07C3	DD	0320 DEFB	0DDH	; A B=08, A=1D
07C4	ED	0321 DEFB	0EDH	; B B=04, A=1D
07C5	FD	0322 DEFB	0FDH	; C B=02, A=1D
07C6	0D	0323 DEFB	00DH	; D B=01, A=1D
		0324 DEFB		

地址	目的码	语句序号	源语句	
07C7	0B	0325	DEFB	00BH ; E B = 01, A = 1B
07C8	07	0326	DEFB	07H ; F B = 01, A = 17
07C9	0E	0327	DEFB	0EH ; EXEC B = 01, A = 1E
07CA	FE	0328	DEFB	0FEH ; SS B = 02, A = 1E
07CB	EE	0329	DEFB	0EEH ; MON B = 04, A = 1E
07CC	DE	0330	DEFB	0DEH ; NEXT B = 08, A = 1E
07CD	CD	0331	DEFB	0CDH ; REG'DISP B = 10, A = 1D
07CE	CB	0332	DEFB	0CBH ; REG DISP B = 10, A = 1B
07CF	C7	0333	DEFB	0C7H ; PORT EXAM B = 10, A = 17
07D0	BF	0334	DEFB	0BFH ; MEM_EXAM B = 10, A = 0F
07D1	BD	0335	DEFB	0BDH ; BP B = 20, A = 1D
07D2	BB	0336	DEFB	0BBH ; PUNCH B = 20, A = 1B
07D3	B7	0337	DEFB	0B7H ; LOAD B = 20, A = 17
07D4	AF	0338	DEFB	0AFH ; PROG B = 20, A = 0F
07D5	19	0339 ;	RECTB, DEFB	19H ; 键 0 不显示
07D6	02	0341	DEFB	02H ; 键 1 = PC
07D7	02	0342	DEFB	02H ; 键 2 = SP
07D8	0C	0343	DEFB	0CH ; 键 3 = IFF 显示用户 IFF2
07D9	16	0344	DEFB	16H ; 键 4 = IX
07DA	18	0345	DEFB	18H ; 键 5 = IY
07DB	0B	0346	DEFB	0BH ; 键 6 = I
07DC	09	0347	DEFB	09H ; 键 7 = H
07DD	0A	0348	DEFB	0AH ; 键 8 = L
07DE	19	0349	DEFB	19H ; 键 9 不显示

0339 ; ※※寄存器表——相应键与在用户寄存器存放区内位置的相对值

地址	目的码	语句序号	源语句
07DF	03	0350	DEFB 03H ; 键 A = A
07E0	05	0351	DEFB 05H ; 键 B = B
07E1	06	0352	DEFB 06H ; 键 C = C
07E2	07	0353	DEFB 07H ; 键 D = D
07E3	08	0354	DEFB 08H ; 键 E = E
07E4	04	0355	DEFB 04H ; 键 F = F
		0356 ;	***辅助寄存器组
07E5	19	0357	REGTBP; DEFB 19H ; 键 0 不显示
07E6	19	0358	DEFB 19H ; 键 1 不显示
07E7	19	0359	DEFB 19H ; 键 2 不显示
07E8	19	0360	DEFB 19H ; 键 3 不显示
07E9	19	0361	DEFB 19H ; 键 4 不显示
07EA	19	0362	DEFB 19H ; 键 5 不显示
07EB	19	0363	DEFB 19H ; 键 6 不显示
07EC	13	0364	DEFB 13H ; 键 7 = H
07ED	14	0365	DEFB 14H ; 键 8 = L'
07EE	19	0366	DEFB 19H ; 键 9 不显示
07EF	0D	0367	DEFB 0DH ; 键 A = A'
07F0	0F	0368	DEFB 0FH ; 键 B = B'
07F1	10	0369	DEFB 10H ; 键 C = C'
07F2	11	0370	DEFB 11H ; 键 D = D'
07F3	12	0371	DEFB 12H ; 键 E = E'
07F4	0E	0372	DEFB 0EH ; 键 F = F' ,
07F5	19	0373	DEFB 19H
07F6	19	0374	DEFB 19H
07F7	19	0375	DEFB 19H
		0376	END

地址	目的码	语句序号	源语句
>07F8		0002	UTILR 程序名称
		0003;	实用RAM和常数
		0004;	版本 1.2 5/13/78
		0005	07F8H ORG
		0006	ABS PSECT
		0007	CTC3L GLOBAL
		0008	CTC1P GLOBAL
		0009	OTCHR6 GLOBAL
		0010	INCHR4 GLOBAL
		0011;	CTC中断矢量表
07F8	D62F	0012	CTC0 DEFW CTC0V
07FA	2307	0013	CTCT1P; DEFW OTCHR6
>07FC		0014	CTC2; DEFS 2 ; 分配地址到RAM ; 转储中断矢量 FAH ; 未用中断 ; 转储中断矢量 FEH
07FE	9D07	0015	CTC3L DEFW INCHR4
>2FC0		0016	ORG 2FC0H
		0017	DIG2 GLOBAL
		0018	DIG4 GLOBAL
		0019	DIG8 GLOBAL
		0020	BPTAB GLOBAL
		0021	SSFLC GLOBAL
		0022	UIF GLOBAL
		0023	DISMEM GLOBAL
		0024	DSMEM1 GLOBAL
		0025	DSMEM2 GLOBAL
		0026	DSMEM3 GLOBAL

地址	目的码	源语句号	源语句
		0027	GLOBAL DSMEM4
		0028	GLOBAL DSMEM5
		0029	GLOBAL DSMEM6
		0030	GLOBAL DSMEM7
		0031	GLOBAL STKPT
		0032	GLOBAL STKPT1
		0033	GLOBAL RST16
		0034	GLOBAL RST24
		0035	GLOBAL RST32
		0036	GLOBAL RST40
		0037	GLOBAL RST48
		0038	GLOBAL RST56
		0039	GLOBAL KEYPTR
		0040	GLOBAL FLG 24
		0041	GLOBAL RFLG
		0042	GLOBAL ARFLG
		0043	GLOBAL BFLG
		0044	GLOBAL PFLG
		0045	GLOBAL MFLG
		0046	GLOBAL PRFLG
		0047	GLOBAL PUNHSH
		0048	GLOBAL PUNHSL
		0049	GLOBAL PUNHEH
		0050	GLOBAL PUNHEL
		0051, ※※※	RAM 变量

地址	目的码	语句序号	源语句
>2FC0		0052	PUNSH; DEFS 1 ; 转录起始地址高字节
>2FC1		0053	PUNHSL; DEFS 1 ; 转录起始地址低字节
>2FC2		0054	PUNHEH; DEFS 1 ; 转录结束地址高字节
>2FC3		0055	PUNHEL; DEFS 1 ; 转录结束地址低字节
>2FC4		0056	RST16; DEFS 3 ; 为处理 RST16—56 指令, 用户在此设置转移指令
>2FC7		0057	RST24; DEFS 3 ;
>2FCA		0058	RST32; DEFS 3 ;
>2FCD		0059	RST40; DEFS 3 ;
>2FD0		0060	RST48; DEFS 3 ;
>2FD3		0061	RST56; DEFS 3 ;
>2FDC		0062 ;	***为 CTC0 中断用户插入转移指令
		0063	CTC0V; DEFS 3 ; CTC0 中断将指向此处
		0064 ;	***CTC3 中断用
>2FD9		0065	FLG24 DEFS 1 ; 转录逻辑 1 时置 1
>2FDA		0066	PRFLG; DEFS 1 ; EPROM 编程标志
>2FDB		0067	KEYPTR DEFS 2 ; 为了写入下一个显示缓冲单元而设的指针
>2FDD		0068	UIF; DEFS 1 ; 用户 IFF2 在此
>2FDE		0069	PFLG; DEFS 1 ; 口检查标志
>2FDF		0070	RFLG; DEFS 1 ; 寄存器检查标志
>2FE0		0071	ARFLG; DEFS 1 ; 辅助寄存器检查标志
>2FE1		0072	MFLG; DEFS 1 ; 存储器检查标志
>2FE2		0073	STKPT; DEFS 1 ; 用户栈指针的高字节
>2FE3		0074	STKPT1; DEFS 1 ; 用户栈指针的低字节
		0075 ;	***断点表的格式为:
		0076 ;	设置断点处的地址为两个字节(高、低), 跟着一个字节即是被 RST8 指令替换掉的用户

地址	目的码	语句序号	源语句
>2FE4		0077	; 操作码的第一字节
>2FF3		0078	BPTAB; DEFS 15 ; 断点地址和移来的操作码
>2FF4		0079	SSFLG; DEFS 1 ; 单步方式标志;
>2FF5		0080	BFLG DEFS 1 ; 已设置的断点数目
>2FF6		0081	DIG2 DEFS 1 ; 已输入 2 位数字的标志
>2FF7		0082	DIG4 DEFS 1 ; 已输入 4 位数字的标志
>2FF8		0083	DISMEM DEFS 1 ; 存储器显示缓冲区
>2FF9		0084	DSMEM1 DEFS 1
>2FFA		0085	DSMEM2 DEFS 1
>2FFB		0086	DSMEM3 DEFS 1
>2FFC		0087	DSMEM4 DEFS 1
>2FFD		0088	DSMEM5 DEFS 1
>2FFE		0089	DSMEM6 DEFS 1
		0090	DSMEM7 DEFS 1
		0091	END





## 附录四

### 实验指示书

## 实验一 熟悉键盘操作

一、实验目的：熟悉按键操作及 TPBUG-A

二、实验设备：TP801 单板机一台

5V 稳压电源一台

三、实验准备工作：

①启动稳压电源，将其输出电压调到  $5V \pm 5\%$ ；

②将单板机的电源线“+”端、“地”端分别接到稳压电源的“+”、“-”端。

四、实验内容：

本实验通过执行下面的程序，一方面熟悉通过键盘命令输入程序的操作，另一方面通过显示器的显示，部分地观察程序的执行，并熟悉 STEP, MON, BP, EXEC.....等命令键的使用，简单了解调试程序的过程。

		ORG	2000H	
2000	3EAA	LD	A, 0AAH	
2002	06BB	LD	B, 0BBH	
2004	78	LD	A, B	
2005	0ECC	LD	C, 0CCH	
2007	79	LD	A, C	
2008	3EAA	LD	A, 0AAH	
200A	76	HALT		

五、实验步骤：

1. 通过键盘操作，将 ORG 2000H 程序送入单板机。（被操作的按键均标以横线，TPBUG-A 响应均以方框表示显示器显示。）

按 键	显 示		
<u>RESET</u> (S1)	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">P</td></tr> </table>	P	
P			
<u>2 0 0 0</u> <u>MEM</u>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">2 0 0 0</td></tr> </table> <table border="1" style="display: inline-table; border-collapse: collapse; margin-left: 20px;"> <tr><td style="padding: 5px;">x x</td></tr> </table>	2 0 0 0	x x
2 0 0 0			
x x			
<u>3 E</u>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">2 0 0 0</td></tr> </table> <table border="1" style="display: inline-table; border-collapse: collapse; margin-left: 20px;"> <tr><td style="padding: 5px;">3 E</td></tr> </table>	2 0 0 0	3 E
2 0 0 0			
3 E			
<u>NEXT</u> <u>A A</u>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">2 0 0 1</td></tr> </table> <table border="1" style="display: inline-table; border-collapse: collapse; margin-left: 20px;"> <tr><td style="padding: 5px;">A A</td></tr> </table>	2 0 0 1	A A
2 0 0 1			
A A			
<u>NEXT</u> <u>0 6</u>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">2 0 0 2</td></tr> </table> <table border="1" style="display: inline-table; border-collapse: collapse; margin-left: 20px;"> <tr><td style="padding: 5px;">0 6</td></tr> </table>	2 0 0 2	0 6
2 0 0 2			
0 6			
<u>NEXT</u> <u>B B</u>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 5px;">2 0 0 3</td></tr> </table> <table border="1" style="display: inline-table; border-collapse: collapse; margin-left: 20px;"> <tr><td style="padding: 5px;">b b</td></tr> </table>	2 0 0 3	b b
2 0 0 3			
b b			

按 键	显 示
<u>NEXT</u> <u>7</u> <u>8</u>	2 0 0 4      7 8
<u>NEXT</u> <u>0</u> <u>E</u>	2 0 0 5      0 E
<u>NEXT</u> <u>C</u> <u>C</u>	2 0 0 6      C C
<u>NEXT</u> <u>7</u> <u>9</u>	2 0 0 7      7 9
<u>NEXT</u> <u>3</u> <u>E</u>	2 0 0 8      3 E
<u>NEXT</u> <u>A</u> <u>A</u>	2 0 0 9      A A
<u>NEXT</u> <u>7</u> <u>6</u>	2 0 0 A      7 6
<u>MON</u>	P

2. 在未执行程序前，检查有关寄存器内容，填写入显示框内：

<u>A</u> <u>REG</u>	A	00
<u>MON</u>	P	
<u>B</u> <u>REG</u>	b	00
<u>MON</u>	P	
<u>C</u> <u>REG</u>	C	00
<u>MON</u>	P	

3. 用STEP键，单步执行程序。首先调整程序计数器PC，使PC指向程序首地址2000H，然后才按动STEP键，注意观察显示器的变化。

当显示器显示地址为200AH时，按下MON键，检查并记录有关寄存器内容，与步骤2进行比较。

<u>MON</u>	P	200A
<u>A</u> <u>REG</u>	A	00
<u>MON</u>	P	
<u>B</u> <u>REG</u>	b	00

<u>MON</u>	P	
<u>C REG</u>	C	
<u>MON</u>	P	
<u>PC REG</u>	1	
<u>MON</u>	P	

4. 用 BP 键，设置断点。

在 2002、2005、2008、200A 单元设置断点，并用键 EXEC 执行程序，注意观察显示器的变化，比较与 STEP 方式执行程序有什么不同。

<u>MON</u>	P	
<u>2 0 0 2 BP</u>	2 0 0 2	
<u>MON</u>	P	
<u>2 0 0 5 BP</u>	2 0 0 5	
<u>MON</u>	P	
<u>2 0 0 8 BP</u>	2 0 0 8	
<u>MON</u>	P	
<u>2 0 0 A BP</u>	2 0 0 A	
<u>MON</u>	P	

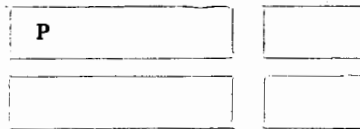
然后输入首地址 2000H，连续按动 EXEC 键执行程序。

当执行程序到显示器显示 200AH 地址时，停止按动 EXEC 键，与步骤 3 一样，检查有关寄存器内容：

<u>A</u>	A	15
<u>B</u>	b	10
<u>C</u>	C	CC
<u>PC</u>	1	001

5. 清除断点。用 EXEC 键连续执行程序。

MON



2 0 0 0 EXEC

- 问题：(1) 用 EXEC 键连续执行程序，为什么显示器不显示？若要显示器自动显示标志“P”，应如何修改程序？
- (2) 检查 PC 内容，对比三种方式执行程序，PC 内容为什么有异、同？
- (3) 清除断点有几种方式？
- (4) 思考：当采用 STEP 方式执行程序，若要显示 DATA 的显示器循环不止的显示 AA, BB, CC, 如何修改本程序？

## 实验二 相对转移中偏移量的计算

一、实验目的：熟悉单板机的性能，了解 Z80 的指令。

在 Z80 系统中，有许多指令需要计算相对转移的偏移量，而这些偏移量是以二进制的补码表示的。在本系统中，具有自动计算并填写相对偏移量的功能。

二、实验设备：同实验一

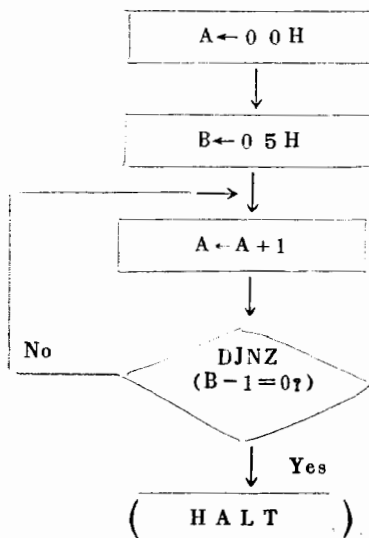
三、实验准备工作：同实验一

四、实验内容：

### 相对地址的计算

地 址	机 器 码	操 作 码
2000	3E00	LD A, 00H
2002	0605	LD B, 05H
2004	3C	→ LOOP: INC A
2005	<u>10 ?</u>	← DJNZ, LOOP-\$
2007	76	HALT

框 图



## 五、实验步骤：

### 1. 操作键盘，输入程序：

<u>RESET</u>		P	
<u>2 0 0 0</u> <u>MEM</u> <u>3 E</u>		2 0 0 0	3 E
<u>NEXT</u> <u>0 0</u>		2 0 0 1	0 0
<u>NEXT</u> <u>0 6</u>		2 0 0 2	0 6
<u>NEXT</u> <u>0 5</u>		2 0 0 3	0 5
<u>NEXT</u> <u>3 C</u>		2 0 0 4	3 C
<u>NEXT</u> <u>1 0</u>		2 0 0 5	1 0
<u>NEXT</u>		2 0 0 6	× ×
<u>NEXT</u> <u>7 6</u>		2 0 0 7	7 6
<u>MON</u>		P	

注意：在 2006 单元不输入内容，因为要将计算的相对偏移量填入此单元。

### 2. 自动计算并填写偏移量：

<u>IX REG</u> <u>2 0 0 4</u>	4 2 0	0 4
<u>MON</u>	P	

(说明：标号 LOOP 单元目的地地址 2004H 装入 IX 寄存器对。)

<u>IY REG</u> <u>2 0 0 5</u>	5 2 0	0 5
<u>MON'</u>	,	

(说明：现行 \$ 单元源地址 2005H 装入 IY 寄存器对。)

<u>DISP</u>	F F	F d
-------------	-----	-----

(FF 表示向上偏移，如 0 0 表示向下偏移，Fd 为偏移值。)

<u>MON</u> <u>2 0 0 6</u> <u>MEM</u>	2 0 0 6	F d
--------------------------------------	---------	-----

(检查 2006，看偏移量是否已装入存储器。)

### 3. 用 STEP 单步键来检查程序的执行过程：

<u>MON</u>	<u>1</u>	<u>REG</u>	1	×	×	×	×	
使 PC→2000H								
<u>2</u>	<u>0</u>	<u>0</u>	<u>0</u>	1	2	0	0	
<u>STEP</u>				2	0	0	2	
<u>STEP</u>				2	0	0	4	
<u>STEP</u>				2	0	0	5	
第一次增 A								
<u>STEP</u>				2	0	0	4	
循环返回								
<u>STEP</u>				2	0	0	5	
第二次增 A								
<u>STEP</u>				2	0	0	4	
<u>STEP</u>				2	0	0	5	
第三次增 A								
<u>STEP</u>				2	0	0	4	
<u>STEP</u>				2	0	0	5	
第四次增 A								
<u>STEP</u>				2	0	0	4	
<u>STEP</u>				2	0	0	5	
第五次增 A								
B 被减到 1								
<u>MON</u>	<u>B</u>	<u>REG</u>	b				0	1
<u>MON</u>	<u>STEP</u>				2	0	0	7

(B 减量, 若非零则转移, 现在 B 为零, 所以 PC→2007H)

选做题:

下列程序使字符“8”在显示器内从右方移到左方

2000	3E00	LD	A, 00H
2002	D388	OUT	(88H), A
2004	3E01	LD	A, 01H
2006	D38C	LOOP,	OUT (8CH), A
2008	CD4F06	CALL	D20MS
200B	CD4F06	CALL	D20MS
200E	CD4F06	CALL	D20MS
2011	CD4F06	CALL	D20MS
2014	CD4F06	CALL	D20MS
2017	07	RLCA	
2018	18EC	JR	LOOP-\$

注：“CALL D20MS”：调用管理程序中 D20MS 子程序，“D20MS”表示延时 20ms。

思考：①能否将上面选作题稍作修改，可以很方便地控制字符在显示器上停留时间？

②若要显示别的字符，应如何修改程序？若要从左方第一位右移到右方，应如何修改程序？要求每次停止时间都很均匀，又如何修改程序？

③能否使“A”、“P”两个字符轮流在显示器上循环移动？

### 实验三 程序设计（一）

一、实验目的：利用已掌握的 Z80 指令系统，进行一些简单程序设计，并通过实验，熟悉调试程序的过程。

二、实验设备：同实验一

三、实验准备工作：同实验一

四、实验内容：

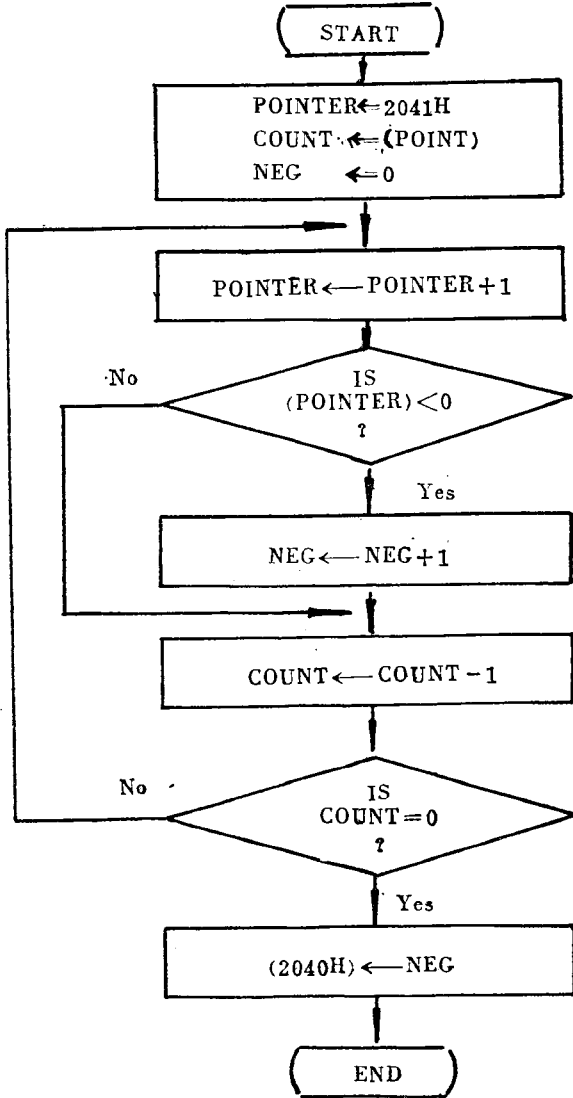
1. 在一个有正、负数的数据块中，找出负数的个数。假定每个数都是 8 位二进制数，并将每个数的第 7 位 (b7) 作为符号位。数据块的长度存放在 2041H 单元，数据从 2042H 单元开始存放，负数的个数存放在 2040H 单元。

例如：

(2040H) = ?  
 (2041H) = 06H  
 (2042H) = 53H  
 (2043H) = F4H  
 (2044H) = 85H  
 (2045H) = 30H  
 (2046H) = 58H  
 (2047H) = 3AH



# 流 程 图



请大家查找指令表，将机器码填入。

2000

```

ORG    2000H
LD     HL, 2041H
LD     B, (HL)
LD     C, 00
LOOP1: INC  HL
LD     A, (HL)
AND   A
JP    P, LOOP2
  
```

```

        INC    C
LOOP2:  DI    NZ, LOOP1
        LD    A, C
        LD    (2040H), A
        HALT

```

注：AND A 这条指令是用来判别 A 的内容的第 7 位 D7 是 0 还是 1（即 A 的内容是正数还是负数），当 D7 = 1 时，标志寄存器 F 的 S = 1，反之，S = 0。请问：能完成这种功能的指令还有那些？请按你的意见，修改上面程序，并通过实验，验证计算机执行程序的结果。

2. 在一个数据块中找出最大数。假定这些数都是 8 位二进制数，且是不带符号位的正整数。数据块的长度存放在 2040H 单元，数据块从 2042H 单元开始存放，最后找出的最大数存放在 2040H 单元。

例如：

```

(2040H) = ?
(2041H) = 06H
(2042H) = 57H
(2043H) = 89H
(2044H) = A7H
(2045H) = 17H
(2046H) = E8H
(2047H) = 95H

```

请大家查找指令表，将对应机器码填入下面程序

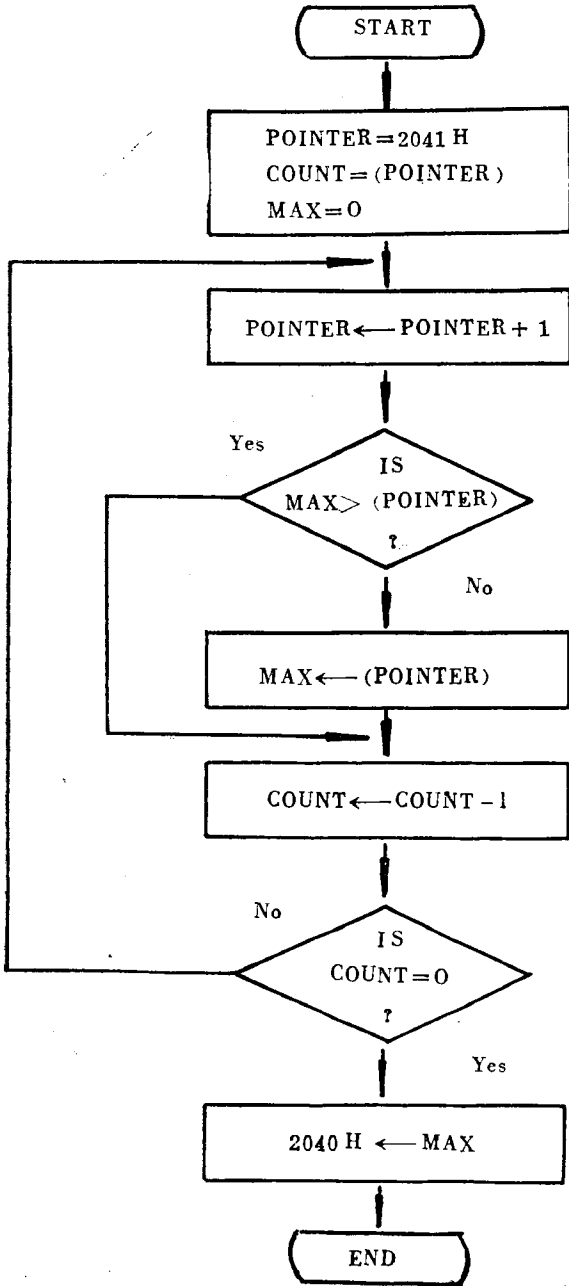
```

2000          ORG    2000H
              LD    HL, 2041H
              LD    B, (HL)
              SUB   A
LOOP1:        INC   HL
              CP   (HL)
              JR   NC, LOOP2
              LD   A, (HL)
LOOP2:        DJ  NZ, LOOP1
              LD   (2040H), A
              HALT

```

按下 MON，检检 2040H 单元内容。

流 程 图



## 实验四 程序设计 (二)

一、实验目的: 同实验三

二、实验设备: 同实验一

三、实验准备工作: 同实验一

四、实验内容:

1. 8 位二进制数求和  $38H + 55H + 26H + 12H + 23H = ?$

5 个八位数相加, 这组数的个数放在(2041H)单元, 和存在(2040H)单元。这组数从(2044H)单元开始存放(假定和数也是一个八位数, 暂不进行和数大于八位数的计算)

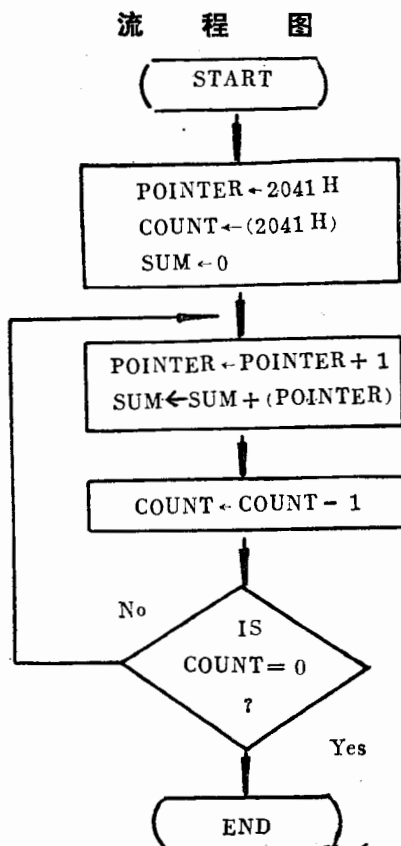
数据: (2041H) = 05H (2044H) = 26H

(2042H) = 38H (2045H) = 12H

(2043H) = 55H (2046H) = 23H

和数: (2040H) = E8H

根据下面流程图, 请大家进行程序设计, 然后在计算机上进行调试。



输入程序, 从首地址 2000H 开始执行, 按  $\overline{MON}$  键, 检查 2040H 地址单元内容, 即是和数。

2. 8 位二进制数求和, 和数是 16 位二进制数

$$0C8H + 0FAH + 96H + 0F9H + 0E8H = ?$$

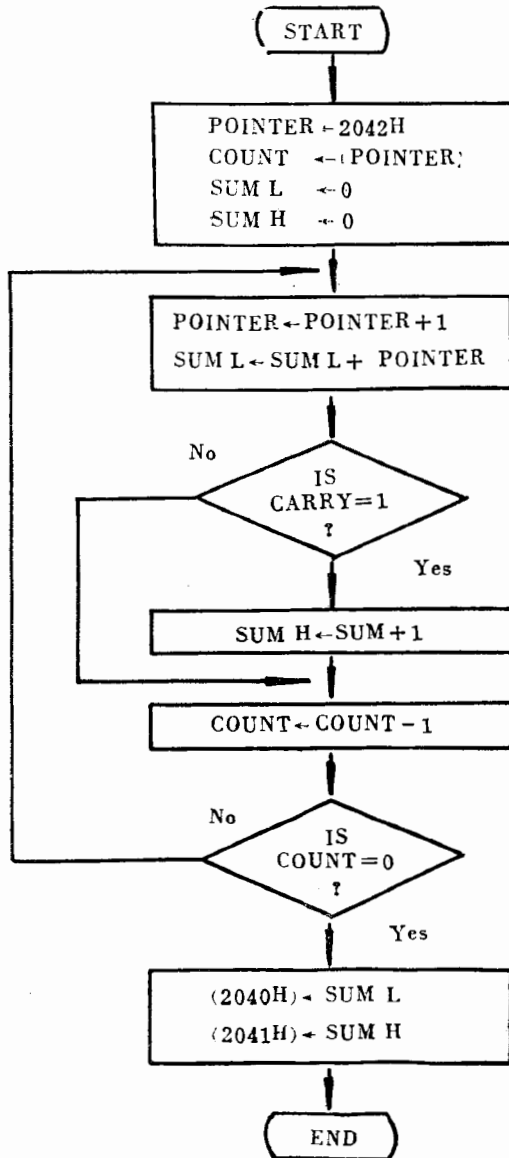
和的低位存放 (2040H) 单元, 高位存放 (2041H) 单元, 数组的个数存放 (2042H) 单元, 数据从 (2043H) 开始存放;

和数 (2040H) = ?  
 (2041H) = ?

数据 (2042H) = 05H (2045H) = 96H  
 (2043H) = 0C8H (2046H) = 0F9H  
 (2044H) = 0FAH (2047H) = 0E8H

输入程序: 从首地址 2000H 开始执行, 按 MON 键, 检查 2040H 和 2041H 地址单元

### 流 程 图



内容，即是和数。

选作题：①将BCD码转换为二进制码。

本程序是将存储在2040H和2041H单元的BCD码29D转换成二进制代码，存放在2042H单元。

；数据

(2040H) = 02

(2041H) = 09

(2042H) = 1DH = 29Decimal

ORG 2000H

LD HL, 2040H

LD A, (HL)

ADD A, A

LD B, A

ADD A, A

ADD A, A

ADD A, B

INC HL

ADD A, (HL)

INC HL

LD (HL), A

HALT

②参考实验二的选作题，设计一个这样的程序：将由键盘输入的一个字符，反复的由慢到快地移动若干次。