微电脑检测与控制

陈匡极　编著

上海交通大学出版社

内 容 简 介

本书通过典型的应用实例，详细分析和叙述了微型计算机在检测和控制方面的基本原理和方法。在内容的取材上尽量少而精，面向实际，面向过程。书中包括开关和显示接口、顺序控制实例、数据检测和处理、群控和巡回检测。本书在叙述方法上深入浅出、通俗易懂，既适用于大专院校作为微机原理及应用课程的参考教材，也可作为科技人员、高级技工的培训教材及自学读物。

微电脑检测与控制

出 版：上海交通大学出版社
（淮海中路1984弄19号）
印 刷：立信常熟印刷联营厂
开 本：787×1092(毫米) 1/16
印 张：10.25　插页 5
字 数：248,000
版 次：1990年 3 月　第 1 版
印 次：1990年 4 月　第 1 次
印 数：1—3,150
ISBN7-313-00613-6/TP·3
定 价：6.60元

前　言

当今社会正处在一个微电脑普及和开拓的新时代，各行各业急需求助于微电脑，以利于自身企事业的发展。不论是设备技术改造还是产品更新换代，都会与微电脑的检测和控制技术密切相关。掌握微电脑的应用需要一定的基础，要从硬件和软件两个方面对过程的实时控制和数据检测方法有一个基本的、清晰的、较为全面的了解，这就需要有一本较为实用的通俗性教材，本书就是为此目的而编写的。本书在编写内容上选用目前应用最广泛的Z80系列单板机作为讲述对象，它既适用于大学电专业或非电专业，又适用于中专作为微型计算机原理及应用课程的教学参考书，还特别适用于有关微电脑检测和控制方面高级技工技师培训班或工程师进修班等作教材。本书对广大科技工作者理解并掌握微电脑的应用都有重要的参考价值。

本书在内容叙述上由浅入深，循序渐进，易于接受，对程序和指令的注释较为详细，清晰易懂，并引导读者自己去分析程序，理解实时控制方面的基本知识，所以它也是一本较好的自学参考书。

本书在选材上尽量削枝强干，突出重点，详叙了微电脑在检测和控制应用方面最基本最必需的知识，体现了少而精的原则。首先从开关和显示接口出发，介绍了顺序控制，数据检测和群控巡测。这些内容都是生产过程中实现微电脑检测和控制所不可缺少的。

本书努力贯彻理论联系实际的原则，面向生产，为生产服务。本书运用微电脑用于工厂设备改造或产品更新方面的应用项目，作为过程检测和控制方面的典型实例，进行剖析，为读者指明了微电脑控制系统硬件和软件的设计方法，对当前正在进行的机电一体化技术改造和技术革新起到承上启下，抛砖引玉的作用。

本书由陈匡极编著。参加本书编写工作的还有钦光德、薛胜华、曹之农等同志。

在本书编写过程中，孙燕唐、李乃同、庞本等同志提出许多宝贵意见，特此表示衷心的感谢。

由于作者水平有限，对书中错误和欠妥之处，恳请读者批评指正。

编　者

1988.10

目　录

第一章　开关和显示器接口

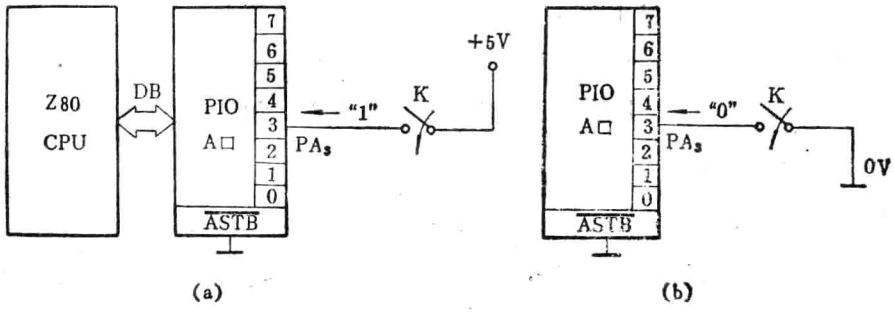
开关和显示器是自动控制中最基本最常用的器件，我们首先讨论其接口电路和控制程序的设计方法。

§1.1　开关接口电路

一、开关量信息

所谓开关量信息，就是代表开关所处的两种状态：闭合和断开。通常用二进制变量“1”或“0”来表示。

图1.1.1所示(a)电源端＋5V通过开关接在A□的PA_3数据线上。当开关接通时，输入到CPU的是高电平“1”（＋5V）。(b)接地端0V通过开关接在A□的PA_3数据线上。当开关接通时，输入到CPU的是低电平“0”（0V）。



(a) 输入高电平　　(b) 输入低电平

图1.1.1 开关量信息“1”和“0”

二、上拉电阻和下拉电阻

图1.1.1所示电路，当开关断开后，PA_3线上既不是高电平“1”态，也不是低电平“0”态，而是所谓的高阻抗状态，如图1.1.2所示，这时外界的干扰将使CPU产生误动作。

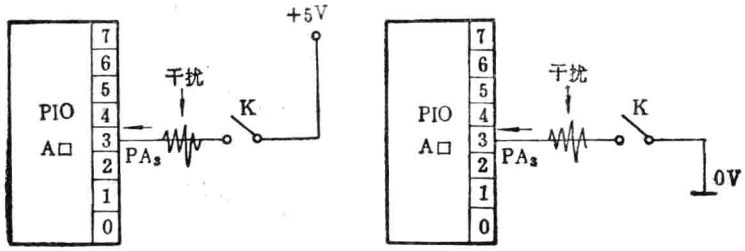


图1.1.2 干扰的引入

所以在硬件线路设计时，为了准确可靠，避免高阻抗状态，必须在端口引线上连接电

阻，若此电阻的另一端与电源＋5V相连接的，就称为上拉电阻。反之，若与地线0V相连的，就称为下拉电阻。

有了上拉电阻或下拉电阻，端口引线上不论开关接通或断开，都有了确定的电平。接上拉电阻的端口引线上，当开关扳断时，向CPU输入高电平“1”，当开关接通时，向CPU输入低电平“0”。接下拉电阻的端口引线上，当开关扳断时，向CPU输入低电平“0”，当开关接通时，向CPU输入高电平“1”。如图1.1.3所示。

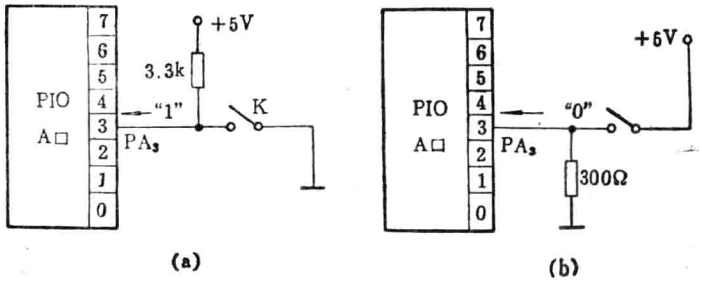


图1.1.3 开关接口电路
(a)上拉电阻；(b)下拉电阻

三、开关状态的读入和检测

要判断一个开关是否已经压合，仅凭上述硬件电路是不能完成的，还必须借助于软件程序。CPU用指令来检测PA_3线上是否已出现了低电平“0”，如果已出现“0”，就表明开关已经被压合。

程序1：检测开关的闭合

要求：将检测到的开关状态“1”或“0”，存放到2040H存储单元中，这样，2040H单元的内容就代表了开关状态的一种“标志”。若开关已闭合，则2040H单元中的标志置为00H。若开关未闭合，则2040H单元中的标志置为01H。

```
        ORG    2000H                  注        释
START:  LD     A,4FH          设置PIOA口为输入
        OUT(PIOCRA), A         (CRA)=(82H)⇐4FH输入
        LDHL, 2040H
        LD(HL), 00H            预置2040H单元为开关已闭合标志
        IN A, (PIODRA)         从PIO A数据端口读入开关的状态，送到累加器8位中只有
                               D3位是有效位，它代表了PA3线上所连接的开关的状态，
                               其余各位不接开关，读入的是无关的随机数，用X表示。
        AND  MASK              检测开关是否已闭合，可用AND指令的屏蔽码去屏蔽A。
                               因为开关接在PA3位上，故屏蔽码定为MASK=08H。执
                               行AND指令后，D3位保留下来，而其余无关位则被屏蔽
                               掉。
        JR Z, DONE             根据结果判转。
        INC (HL)               (HL)+1=(2040H)+1=00H+1=01H
DONE:   HALT
```

屏蔽码MASK的取值由开关接入PIO端口数据线$PA_{7\sim0}$中的哪一位而定，接入的位取1，其余位取0。

表1.1.1　MASK与开关接入位置的关系

开关位置	屏蔽码 MASK	
	二进制	十六进制
PA_0	0 0 0 0 0 0 0 1	01H
PA_1	0 0 0 0 0 0 1 0	02H
PA_2	0 0 0 0 0 1 0 0	04H
PA_3	0 0 0 0 1 0 0 0	08H
PA_4	0 0 0 1 0 0 0 0	10H
PA_5	0 0 1 0 0 0 0 0	20H
PA_6	0 1 0 0 0 0 0 0	40H
PA_7	1 0 0 0 0 0 0 0	80H

当开关接在第7位PA_7端口引线上时，则可用循环移位指令来检测开关的状态。程序改写如下：

```
        ORG  2000H                          注      释
START:  LD   A, 4FH                         同上一程序
        OUT  (PIOCRA), A
        LD   HL, 2040H
        LD   (HL), 00H
        IN   A, (PIODRA)
        RLA                                 把D7位移入CY位使CY=D7
        JR   NC, DONE                       若CY=0, 表示D7=0, 即: 开关已闭合, 就转DONE结束。
                                            若CY=1, 表示D7=1, 即: 开关未闭合, 则顺序执行。
        INC  (HL)
DONE:   HALT
```

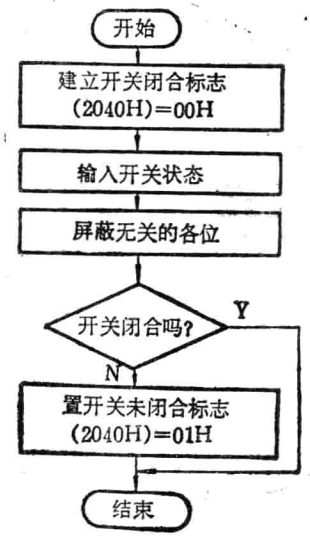


图1.1.4　检测开关闭合的流程图

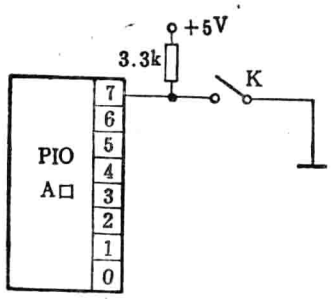
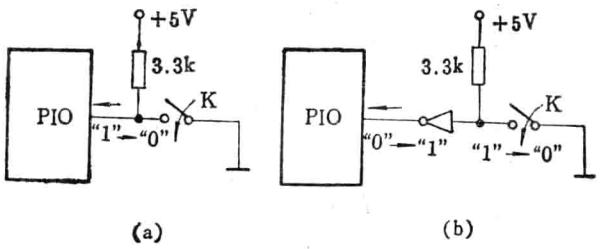


图1.1.5　开关接D_7位

四、带反相器的开关接口电路

图1.1.6(a)电路中当开关接通时，输向PIO的是低电平“0”，这与习惯相反。习惯的观念是开关接通输出“1”，开关断开输出“0”。为此，在开关输出端串联一只非门，使开关接通输出“1”，开关断开输出“0”，这就与习惯一致了，如图1.1.6(b)所示。



(a)与习惯不一致　　(b)与习惯一致

图1.1.6　带反相器的开关接口

反相器专用芯片可采用74LS00型，片内包含4只与非门如图1.1.7所示。

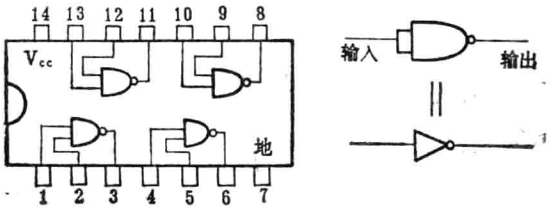


图1.1.7　74LS00反相器

§1.2　检测开关压合的次数

开关闭合的次数，一般都可用计数器来累计。在微机检测过程中，不需另外加接计数器，只要把通用寄存器或存储单元作为计数器，用增量指令INC使其加1，就成了加1计数器(或用

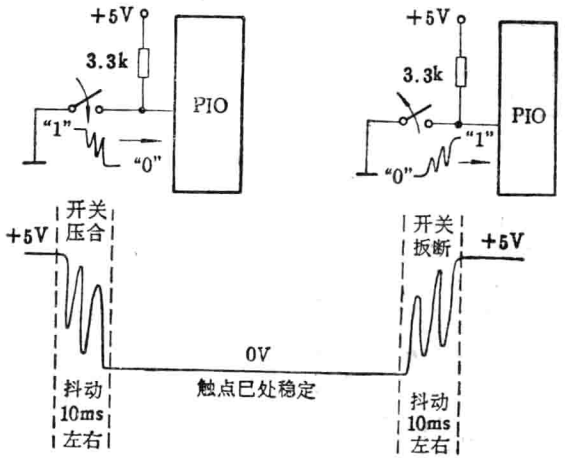


图1.2.1　抖动的产生

减量指令DEC使其减1，就成了减1计数器)。每当开关闭合一次，就用INC指令使存储单元的内容加1，闭合10次，就加了10次，从而达到了累计开关闭合次数的目的。事实上问题并非这样简单，因为开关在闭合一次的过程中，其机械触点要产生弹跳(即抖动)，这样，断开和闭合(即高低电平)之间的转换有很多次，如图1.2.1所示。而CPU采集信号的速度比开关的动作要快得多，CPU就会把开关抖动的次数误认为开关压合的次数进行累计，从而在计数控制中将会造成误动作。所以必须等待开关触点动作稳定后，再检测开关状态的信息。

为避免计数失误，在硬件方面可采用消抖电路，在软件方面可采用消抖程序。

一、消抖电路

1．R-S触发器组成的消抖电路

如图1.2.2所示，开关K通过R-S双稳态触发器接到PIO端口引线上。当开关K接a点时，使与非门A入低出高，与非门B全高出低，此低电位再反馈到A输入端，保证A始终出“1”，B始终出“0”。开关K闭合瞬间的抖动，不会影响PIO端口引线上的“1”电平，从而消除了抖动所带来的误操作。

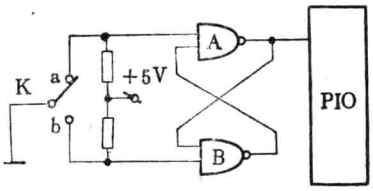


图1.2.2 R-S触发器消抖电路

2．施密特触发器组成的消抖电路

该电路由反相器，积分电路及施密特触发器组成，如图1.2.3所示。积分电路的功能是消除抖动，由R和C决定积分时间常数，它决定c点动态波形的前沿和后沿状态。R_2是电容C的泄流电阻。

施密特触发器的功能是把c点输入的缓慢变化的波形，整形成前后沿陡削的方波。

施密特触发器的专用芯片可选用74LS14，如图1.2.4所示。

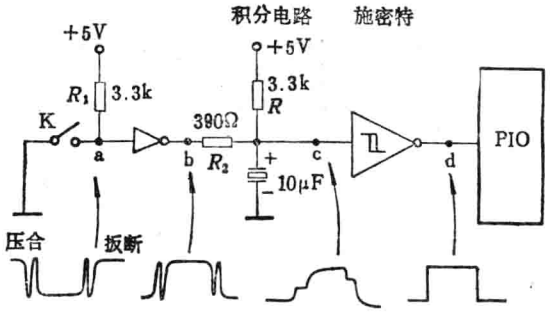


图1.2.3 施密特消抖电路

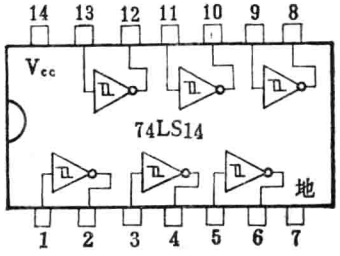


图1.2.4 施密特触发电路

二、消抖程序

在开关量输入通道中，消抖环节是必不可少的。除了用硬件实现消抖外，也可用软件来实现。当程序查到开关有一次闭合后，等待一段时间，避开触点的抖动，使触点在抖动期间，CPU不去读取开关的状态，而在触点抖动结束后，再去读取。否则会把抖动误认为闭合次数而累计出错。消除抖动的这段时间可由CPU执行延时程序来达到，延时程序结束，触点已处稳定状态。

程序2：检测开关闭合的次数（如图1.2.5）。

要求：(1) 用2040H存储单元作为计数器。开关每闭合一次，就使这计数单元加1，所以2040H单元的内容就代表了开关闭合的次数。

(2) 消抖延时程序作为子程序进行调用，延时长短由开关触点的弹跳时间决定。

两种消抖法的比较：

硬件消抖：不占用CPU的工作时间，CPU不必查询开关的抖动，不必延时等待。

软件消抖：节省集成电路芯片，但CPU需要查询并执行延时程序，浪费CPU的工作时间。

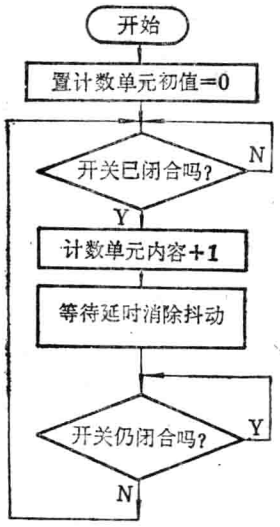


图1.2.5 检测开关的闭合次数

```
        ORG   2000H                    注        释
START:  LD    A, 4FH
        OUT   (PIOCRA), A              置A口为输入
        LD    HL, 2040H                置计数单元的地址指针HL=2040H
        LD    (HL), 00H                置计数器初值为00H (HL)=(2040H)←00H
RDSWH1: IN    A, (PIODRA)              读入开关状态
        AND   MASK                     查开关已闭合吗?
        JR    NZ, RDSWH1               否，等待开关闭合
        INC   (HL)                     是，计数器加1 (HL)+1=(2040H)+1
        CALL  DELAY                    调用延时子程序，等待10～20ms消除抖动。延时程序可另编，也可用TP801单板机延时20ms的子程序
RDSWH2: IN    A, (PIODRA)              再查开关仍处在闭合状态吗?只有当开关从闭合转为释放，才能为下次的再闭合计数作好准备。
        AND   MASK
        JR    Z, RDSWH2                若Z=1，表示开关仍处在闭合状态，就必须循环等待开关的断开。
        JR    RDSWH1                   若Z=0，表示开关已断开，就可循环回去重新查看下一次的闭合并计数加1。如果开关重复闭合100次，则从RDSWH1开始的这段程序就重复执行100次，计数器将增量100次，结果在2040H计数单元中将保存开关闭合的次数，即(2040H)=64H。而开关抖动的次数，则由于软件延时而被消除了，不会被计数单元所累计。
```

§1.3 检测多位开关的位置

一、多位开关的接口电路

如图1.3.1是一个8位开关，其动触点接地，8个静触点7～0接消抖电路，并通过上拉电阻接+5V电源。

当动触点不在位上，即没有接通任一静触点时，PIO的8根数据线上都出现高电平“1”。

编码为FFH。

当动触点接通某一静触点时，则该位数据线上将出现低电平“0”，而其余数据线上仍为“1”。8位开关位置不同，8位数据线送向CPU的电平信息编码也不同。如表1.3.1所示。

表1.3.1 开关位置信息编码表

开关闭合位置	开关量信息编码（二进制）	（十六进制）
0	1 1 1 1 1 1 1 0	FEH
1	1 1 1 1 1 1 0 1	FDH
2	1 1 1 1 1 0 1 1	FBH
3	1 1 1 1 0 1 1 1	F7H
4	1 1 1 0 1 1 1 1	EFH
5	1 1 0 1 1 1 1 1	DFH
6	1 0 1 1 1 1 1 1	BFH
7	0 1 1 1 1 1 1 1	7FH
不在位	1 1 1 1 1 1 1 1	FFH

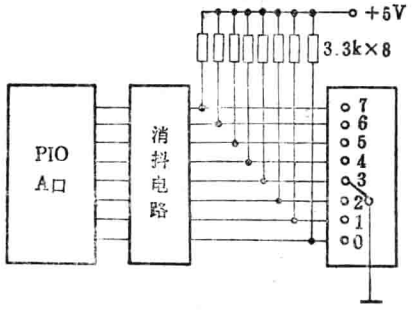


图1.3.1 8位开关接口电路

程序3：检测多位开关的位置(如图1.3.2)

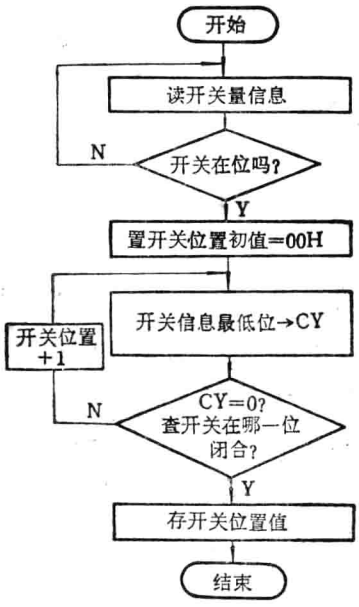


图1.3.2 检测多位开关位置的流程图

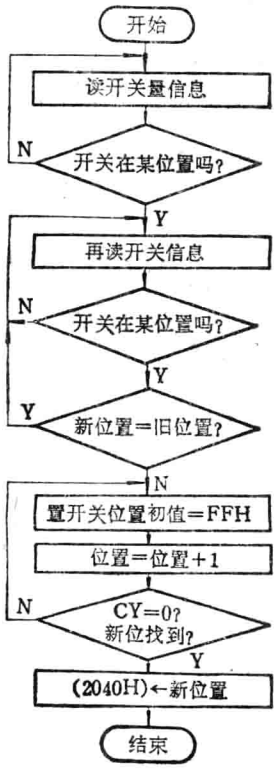


图1.3.3 检测多位开关位置是否改变流程图

要求：(1)设多位开关位置值有8位，即：0～7。

(2)CPU读入开关量信息，查找开关在哪一位置上闭合。

(3)将这一位置值送内存2040H单元保存。

```
        ORG  2000H                    注        释
START:  LD   A, 4FH          置A口为输入
        OUT(PIOCRA), A
RDSWH:  IN   A, (PIODRA)     读入开关量信息A⇐(80H)=××××  ××××
        CP   0FFH            开关在某一位置上吗?
                             如果不在位则读入后A=11111111=FFH
                             比较结果A-FFH=FFH-FFH=00H→置标志Z=1。
                             如果在第3位则读入后A=11110111=F7H
                             比较结果A-F7H≠00H→置标志Z=0。
        JR   Z, RDSWH        若Z=1，不在位，循环等待开关扳到某位上。若Z=0，
                             在位上，顺序执行下一条
        LD   B, 00H          置开关位置初值=00H。
TEST:   RRA                  查找开关在哪一位闭合?可将A的每一位从D0开始逐个
                             移入进位位CY
        JR   NC, DONE        再判CY,若CY=0,则开关闭合位置已找到,转DONE。
                             这时B的内容即开关的位置号。如图1.3.1开关在D3闭
                             合则需循环三次，B将增量3次，故B=03H
        INC  B               若CY=1，表示该位是非闭合位，必须将开关位置加1，
                             即B+1
        JR   TEST            再循环回去继续查找。
DONE:   LD   HL, 2040H       将开关位置值存入2040H单元
        LD(HL), B            (HL)=(2040H)⇐B=03H
        HALT
```

程序4：检测多位开关位置的改变(如图1·3·3)

要求：检测多位开关是否从一个位置转到另一位置(设从3扳向4)，并将开关的新位置值

```
        ORG  2000H                    注        释
START:  LD   A, 4FH          置A口为输入
        OUT (PIOCRA), A
RDSWH1: IN   A, (PIODRA)     读开关信息
        CP   0FFH            开关在某一位置上吗?
        JR   Z, RDSWH1       否，等待开关扳到某位上。
        LD   B, A            是，开关在某一位置上则将开关信息代码暂存于B中
                             B⇐A=11110111=F7H
RDSWH2: IN   A, (PIODRA)     再读开关信息
        CP   0FFH            开关在某一位置上吗?
        JR   Z, RDSWH2       否，等待开关扳到某位上。
        CP   B               是，将A新位置开关信息代码与原位置开关信息代码相比
                             较，新旧位置是否相同?
                             若开关仍在原位3，则F7H-F7H=00H，置标志→Z=1。
                             若开关已在新位4，则F7H-EFH≠00H，置标志→Z=0。
        JR   Z, RDSWH2       若Z=1，则循环等待。
        LD   B, 0FFH         若Z=0，置开关位置初值=FFH。
TEST:   INC  B               从初值FFH开始每次增1，以确定开关新的位置值。
        RRA                  从D0开始将A各位逐位右移至进位位CY。
        JR   C, TEST         再判CY：若CY=1，表示为非闭合位，再继续查找。
                             若CY=0，表示新闭合位已找到，即B=04H。
        LD   HL, 2040H       新的位置值存入2040H单元中
        LD   (HL), B         (2040H)⇐B=04H
        HALT
```

04H存入2040H单元内保存。

§1.4 键盘接口电路

键盘是微型机控制系统中最常用的输入设备。通过它可以向微机输入程序、数据及各种控制命令。键盘分编码和非编码两种。下面讨论的是一般常用的非编码键盘。

一、简易键盘接口电路

简易键盘相当于一组开关。如图1·4·1所示，图中共有8个按键，各个键的一端接PIO(或74LS244)的端口，另端接地。CPU检测键盘是否有键按下，哪一个键按下，与前面检测开关的方法是相同的，这里不再赘述。

简易键盘的线路简单，按键识别也容易，但引线太多，一个键，需要一根引线。64个键就有64根引线，需要8个端口。所以按键较多时，常用矩阵键盘。

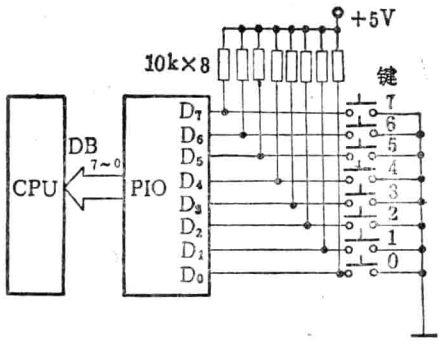


图1.4.1 简易键盘接口

二、4×4矩阵键盘接口电路

矩阵键盘就是把按键开关按行、列排列成矩阵形式，使引线大为减少。64个键只需要8根行线和8根列线，一个8位输出锁存器和一个8位输入三态缓冲器。如图1·4·2是16个键的键盘排成4行×4列矩阵。输出锁存器可选74LS273，输入缓冲器可选74LS244。也可共用一个PIO端口，工作在位控方式3，高4位作输出，低4位作输入，如图1·4·3所示为A口作输入，B口作输出，各用一个PIO端口。

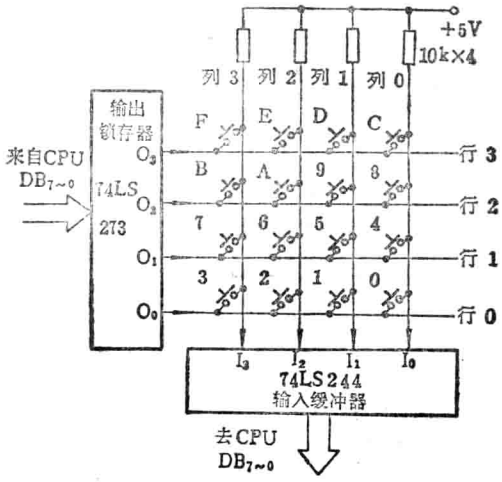


图1.4.2 矩阵键盘接口

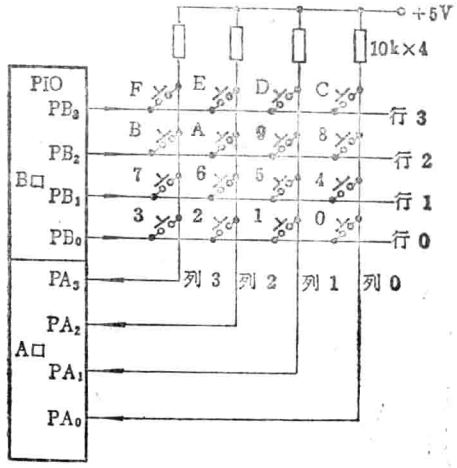


图1.4.3 矩阵键盘接口

(一)检测是否有键闭合——全扫描

要判断是否有键按下，可采用扫描键盘的方法，先进行全扫描。全扫描就是对整个键盘同时进行一次扫描。如图1.4.4所示。

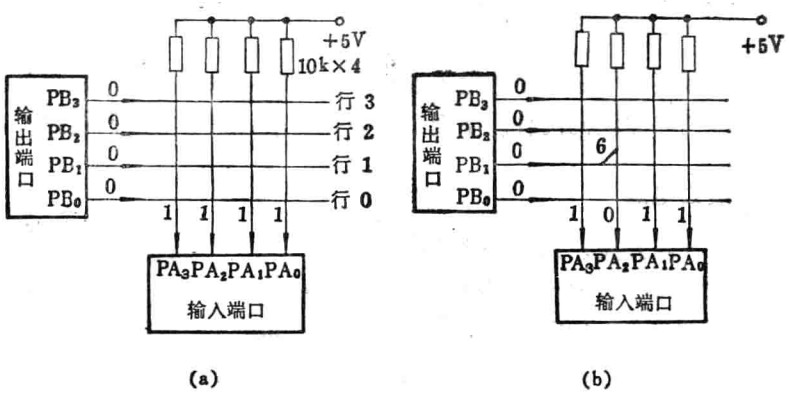


图1.4.4 全扫描

(a)无键闭合时　　(b)有键闭合时

输出$PB_{3\sim0}$=0000　　输出$PB_{3\sim0}$=0000

输入$PA_{3\sim0}$=1111　　输入$PA_{3\sim0}$=1011

1．首先由CPU通过锁存器273或PIOB口低4位输出全扫描电平，即$PB_{3\sim0}$=0000送到键盘各行线上。

2．然后CPU通过缓冲器244或PIOA口低4位读入键盘列线数据。

3．CPU再判断读入的数据各位是“0”还是“1”，若全为“1”表示无键按下，如图1.4.4(a)。若非全“1”表示有键按下，如图1.4.4(b)。

(二)检测是哪个键被按下——行扫描

通过全扫描发现有键按下，就用逐行扫描方法再确定是哪个键被按下。如图1.4.5所示。

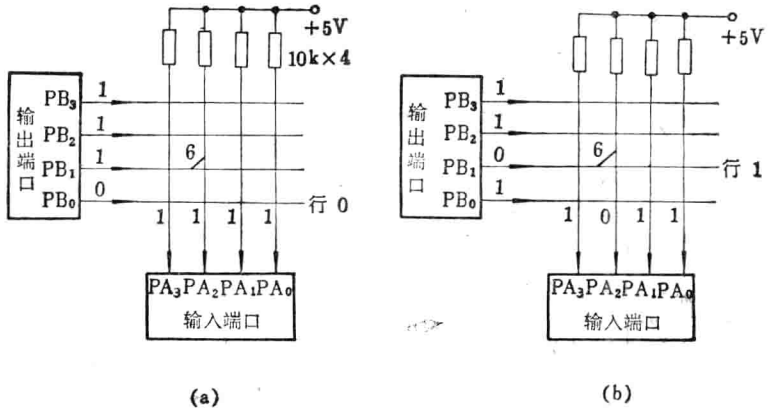


图1.4.5 行扫描

(a)扫描第0行：　　(b)扫描第1行：

输出$PB_{3\sim0}$=1110　　输出$PB_{3\sim0}$=1101

输入$PA_{3\sim0}$=1111　　输入$PA_{3\sim0}$=1011

无键闭合　　有键闭合，在第1行第2列键号值=6

1．先扫描第0行，CPU通过输出端口向行线0输出“0”电平，而其余行输出“1”电平。

如图1.4.5(a)。

2．再由CPU通过输入端口读入列的数据。

3．CPU再判断读入的数据中哪一位出现“0”，就可确定按键在哪一列上。如果读入的列值全是“1”，说明这一行所有按键都未被按下，接着再扫描第1行。以此类推。如图1.4.5(b)。

程序5：检测键盘按键

检测键盘按键的流程如图1.4.6所示。

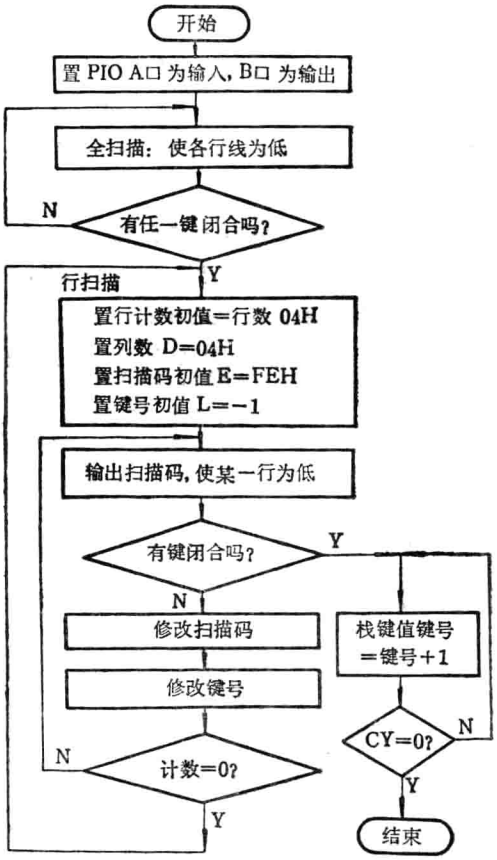


图1.4.6 检测键盘按键流程图

```
        ORG  2000H                    注        释
START:  LD   A, 4FH          置A口为输入
        OUT  (PIOCRA), A
        LD   A, 0FH          置B口为输出
        OUT  (PIOCRB), A
        SUB  A               全扫描：B口输出使所有行线为低电平“0”。见图1.4.4(a)
        OUT(PIODRB), A
WATKEY: IN   A, (PIODRA)     输入列数据
        AND  0FH             屏蔽掉无用的列位，保存有关的列位。
        CP   0FH             有任一键闭合吗？
                             若 D3D2D1D0=1111，即无键按下时，则比较结果 A = 00000000为全。置标志Z=1
                             若 D3D2D1D0不为全1，即有键按下时，则比较结果 A≠00000000非全0，置标志Z=0
```

```
        JR  Z, WATKEY          若无键按下，则返回等待按键闭合。
                               若有键按下，则做行扫描。
LOOP1: LD  B, 04H              行扫描：置行计数器B初值＝行数4
       LD  C, PIODRB           建立端口地址指针
       LD  D, 04H              置列数D＝4列
       LD  E, 0FEH             置扫描码初值使行0为低电平
       LD  L, 0FFH             置键号初值L＝FFH，即为－1
       OUT (C), E              (DRB)←E＝FEH输出使最低位D0＝“0”扫描一根行线
LOOP2: IN  A, (PIODRA)         读入列数据
       AND 0FH                 屏蔽无用列位
       CP  0FH                 有列线接“0”吗？有键闭合吗？
       JR  NZ, KEYDN           若有闭合键，则转至键译码去寻找键值。
       RLC E                   若无闭合键，则修改扫描码，准备扫描下一行。
       LD  A, L                修改键号作为扫描下一行的初值
       ADD A, D
       LD  L, A
       DJNZ LOOP2              如未扫完，转扫下一行
       JR  LOOP1               如已扫完，无键按下，重新开始扫描。
KEYDN: INC L                   找键值：进行键译码  L＋1＝键号加1
       RRA                     该列线为低吗？即把列线各位右移至CY，当CY＝1，表示
                               该列上无按键。若CY＝0，表示闭合键就在该列上。
       JR  C, KEYDN            当CY＝1，循环继续查找键值
                               当CY＝0，按键的键值就在L中，即L＝键号，结束。
DONE:  HALT
```

三、8×8矩阵键盘

8×8矩阵键盘共有64个键，其接口电路如图1.4.7所示。

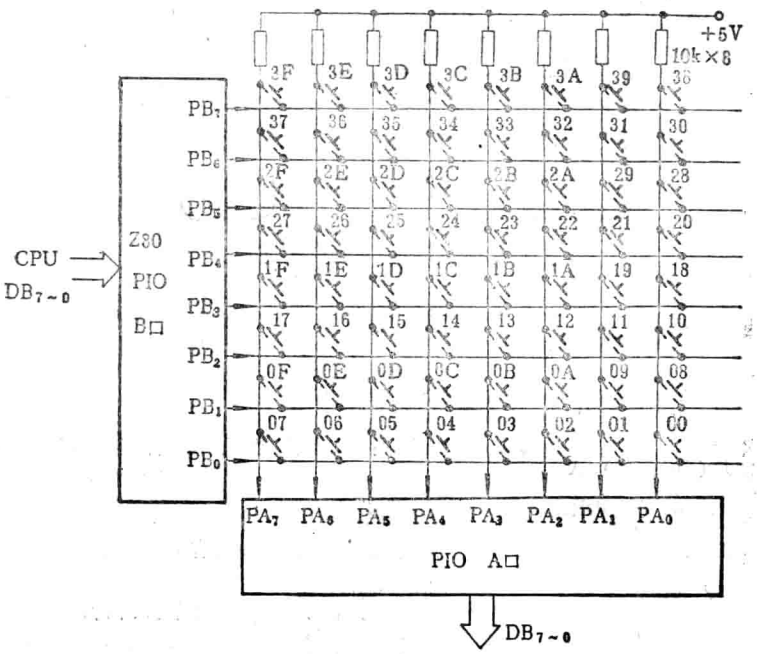


图1.4.7 8×8矩阵键盘接口电路