江 蘇 大 學

JIANGSU UNIVERSITY

微机接口课程设计报告

步进电机综合控制

设计时间：2019 — 2020 学年 第 1 学期

专业班级：XXXXXXXXXXX

学生姓名：zoxiii

学　　号：XXXXXXXXXX

指导教师：XXXXXXXXXXX

开课系部：计算机科学与技术系

2020 年 1 月 7 日

目录

步进电机综合控制

一、 课程设计目的

1）掌握综合控制键盘、数码管、步进电机的能力；

2）掌握有关中断服务程序的编制方法；

3）掌握时钟模块时钟信号分频系数计算的方法；

4）熟悉各常用可编程接口芯片的工作原理、使用方法及引脚连接特性，正确操作实验设备，能根据设计要求制定总体方案，完成硬件连线，理解应承担的责任；

5）能够根据要求编写出满足功能要求的软件代码，能够利用开发工具进行设计调试、协助分析运行结果、定位设计错误等；

6）具有创新意识。尝试对所要求的功能设计提出有效的改进设想并努力实现；或尝试增加新的系统功能并努力实现；

7）具有对微机输入输出系统进行需求分析的能力，能针对具体的问题获取新知识完成系统的设计；

8）能够以口头和书面方式准确地描述、总结所完成的设计和主要成果，撰写比较规范的课程设计报告。

二、 设计内容及具体要求

2.1 实验要求

1）8251、8253、8255、8259、0832、0809 等接口芯片模块必须用到 3 个或 3 个以上。

2）必须用到键盘和 LED 数码管模块进行相关的设置和显示。

3）必须要在原实验要求的基础上进行创新设计的思考和实现。

2.2 设计主要内容

1）通过按键实现外接步进电机模块的“启动”、“正反转”、“设定转速”、“设定步数”等功能。同时，将电机的当前转速和步数显示在数码管上。

2）对于步进模块要实现的功能通过键盘输入命令来控制，不同的按键对应不同的功能，具体的键盘的命令如下：

表 2-1 键盘命令表

命令	功能
A	启停步进电机
B##	步进电机以速度##转/分正转（顺时针）
C##	步进电机以速度##转/分反转（逆时针）
D##	步进电机前进##步（顺时针）
E##	步进电机后退##步（逆时针）

F	程序结束

3）数码管显示格式：

用左边 3 个数码管显示当前设定的命令以及对应命令设置的转速或步数，代表目前电机所处的功能状态，其具体步进显示格式如下：

表 2-2 高三位数码管显示格式表

格式	含义
B ##	表示以##速度正转；
C##	表示以##速度反转。
D##	步进电机前进##步（顺时针）
E##	步进电机后退##步（逆时针）

假设当前步数初值为 0，每正转一步，步数值加 1；反转一步，步数值减 1。所以用右边 5 个 LED 数码管显示当前的结果。具体步进显示格式如下：

表 2-3 低 5 位数码管显示格式表

格式	含义
0 ####	表示步数大于 0；
—####	表示步数小于 0。

2.3 步进电机原理

步进电机的驱动原理是通过它每相线圈的电流的顺序切换来使电机做步进式旋转，驱动电路由脉冲来控制，所以调节脉冲的频率便可改变步进电机的转速，微控制器最适合控制步进电机。另外，由于电机的转动惯量的存在，其转动速度还受驱动功率的影响，当脉冲的频率大于某一值（本实验为 f.>100hz）时，电机便不再转动。

实验电机共有四个相位（A,B,C,D），如果按照单 4 拍方式控制步进电机，假设用 8255 的 PC4~7 分别对应于 ABCD 四个相位。如果 PC7、PC6、PC5、PC4 的值按照 0001->0010->0100->1000->0001 的方式进行变化，则电机就实现了正向转动。如果 PC7、PC6、PC5、PC4 的值按照 1000->0100->0010->0001->1000 的方式进行变化，则电机就实现了反向转动。通过控制输出两次数值的时间间隔就可以改变转速。

由分析得到转动方式的相位如下：

表 2-4 转动方式对应相位转换表

转动方式	正转转动步骤	反转转动步骤
单	（A->B->C->D->A）	（C->B->A->D->C）
4	0001->0010->0100->1000->0001	0100->0010->0001->1000->0100
拍	10H->20H->40H->80H->10H	40H->20H->10H->80H->40H
双	（AB->BC->CD->DA->AB）	（DC->CB->BA->AD->DC）

4 拍	0011->0110->1100->1001	1100->0110->0011->1001->1100
	30H->60H->C0H->90H->30H	C0H->60H->30H->90H->C0H
单 双 8 拍	（A->AB->B->BC->C->CD->D->DA->A）	（B->BA->A->AD->D->DC->C->CB->B）
	0001->0011->0010->0110->0100->1100->1000->1001->0001	0010->0011->0001->1001->1000->1100->0100->0110->0010
	10H->30H->20H->60H->40H->C0H->80H->90H->10H	20H->30H->10H->90H->80H->C0H->40H->60H->20H

其中对于四相四拍，线圈换一次相转子转动 7.5^0，称为一步，转动一周由 48 步完成。由于 1 转/分=48 步/60 秒=0.8 步/秒，亦 1.25 秒/步，若转速为 n 转/分，则步进电机每 1.25/n 秒时间转一步。

对于四相八拍，线圈换一次相转子转动 3.75^0，称为一步，转动一周由 96 步完成。由于 1 转/分=96 步/60 秒=1.6 步/秒，亦 0.625 秒/步，若转速为 n 转/分，则步进电机每 0.625/n 秒时间转一步。

实验中可以根据 8253A 输入时钟信号的频率计算出 8253A 的分频系数。可根据需要选用计数分频电路 74LS393（四位二进制同步加减计数器）的某一输出信号作为 8253A 的 CLK 时钟信号。

其中 8253A 的计数初值$N = F_{clk}/F_{out} = T_{out}/T_{clk}$

2.4 设计具体要完成的主要功能

1）8255，键盘初始化 → 可向步进电机输出相位值，对键盘扫描获取键值；

2）8253 → 8253 产生的频率信号产生中断信号传给 8086 系统；

3）8259 初始化，装载中断向量，设计中断子程序 →系统产生中断，执行中断子程序对步进电机进行控制；

4）LED 数码管显示，显示步进电机正在执行的功能及其状态 ；

5）扫描按键，根据按键判断命令，修改步进电机的相关参数，对电机进行启停，转动方式，转动方向，步数等等的控制。

6）8251 → 实现两台设备通信，同时控制两台设备的电机转动。

三、所需实验器件

3.1 D1 区：步进电机模块

[端口]：A,B,C,D 四个相位

[功能]：步进电机模块有 A,B,C,D 四个端口，通过其他设备向步进电机输入相位值，控制电机的相位切换，从而控制步进电机的转动。

步进电机具有旋转的角度正比于脉冲数，步进电机惯量低、定位精度高、无累积误差、控制简单等特点。

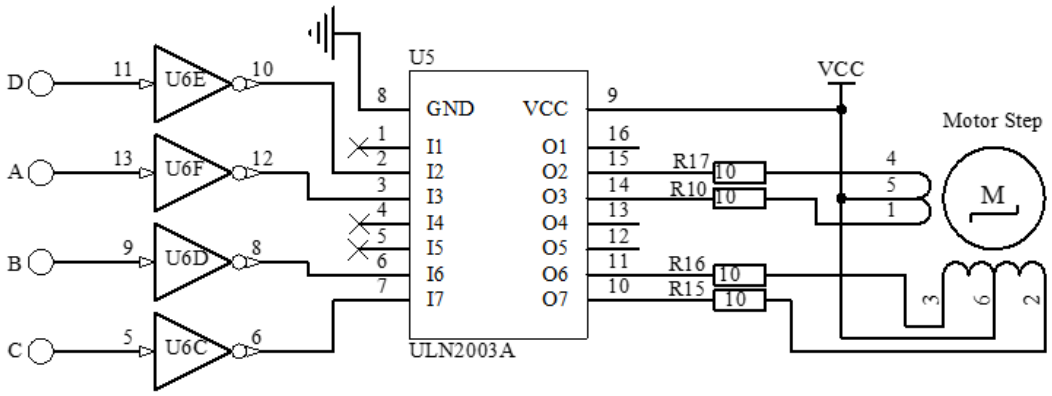


图 3-1 步进电机原理图

3.2 D3 区：8255A+数码管驱动电路

[端口]：CS：片选信号，低电平有效；

A0、A1：地址信号。

JP24：PC 口(键盘行)；

JP20：PB 口(键盘列)；

JP23：PA 口；

JP16：数码管段码

JP17：数码管段选

PC0~7：C 端口各个端口可单独操作

[功能]：PA,PB，PC0,PC1 控制键盘输入和数码管的显示；

PC4-输出相位值控制步进电机驱动；

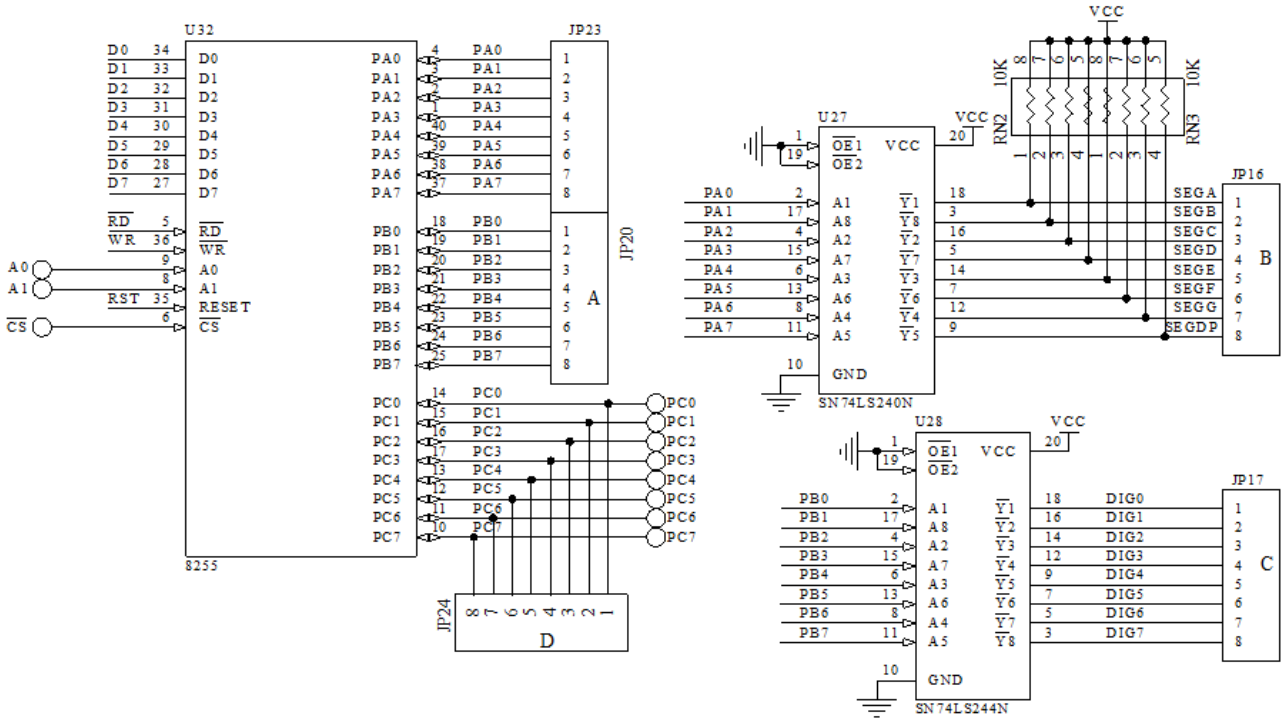


图 3-2 8255A+数码管驱动电路原理图

3.3 C4 区：8253A 定时模块

[端口]：CS：片选信号，低电平有效；

A0、A1：地址信号；

CLK0：计数器 0 的频率输入

OUT0：计数器 0 的频率输出

GATE：门控信号

[功能]：产生步进电机驱动的定时信号，通过不同的定时信号改变电机的速度。

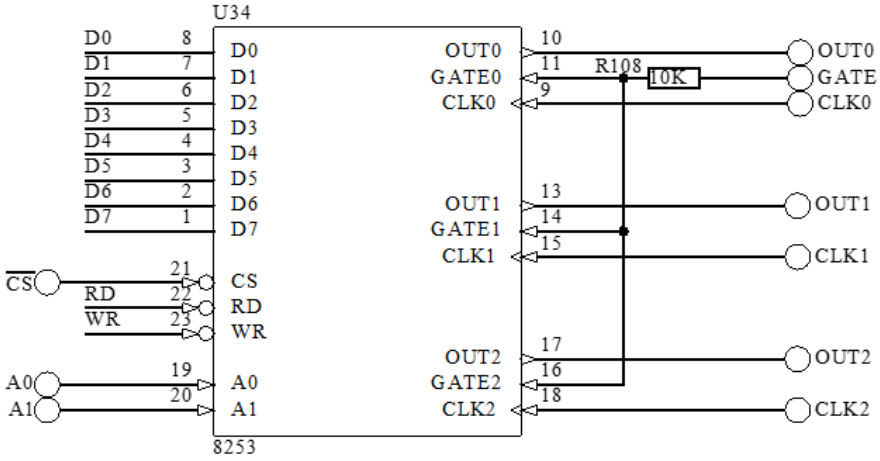


图 3-3 8253A 模块原理图

3.4 B3 区：8259A 中断控制器

[端口]：CS：片选信号，低电平有效；

A0：地址信号

INR0..INR7：中断输入

INTA：中断响应

[功能]：接收并处理来自 8253A 的定时中断请求，执行中断子程序对步进电机进行控制；

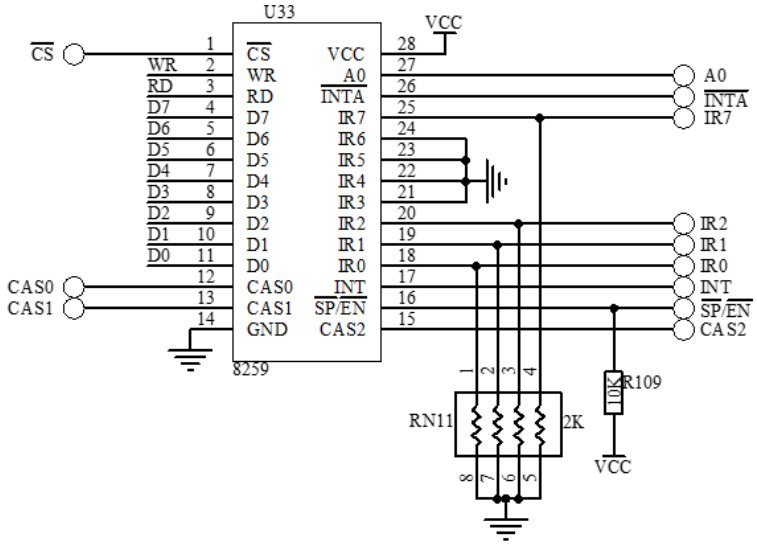


图 3-4 8259A 模块原理图

3.5 F5 区：小键盘和数码管等

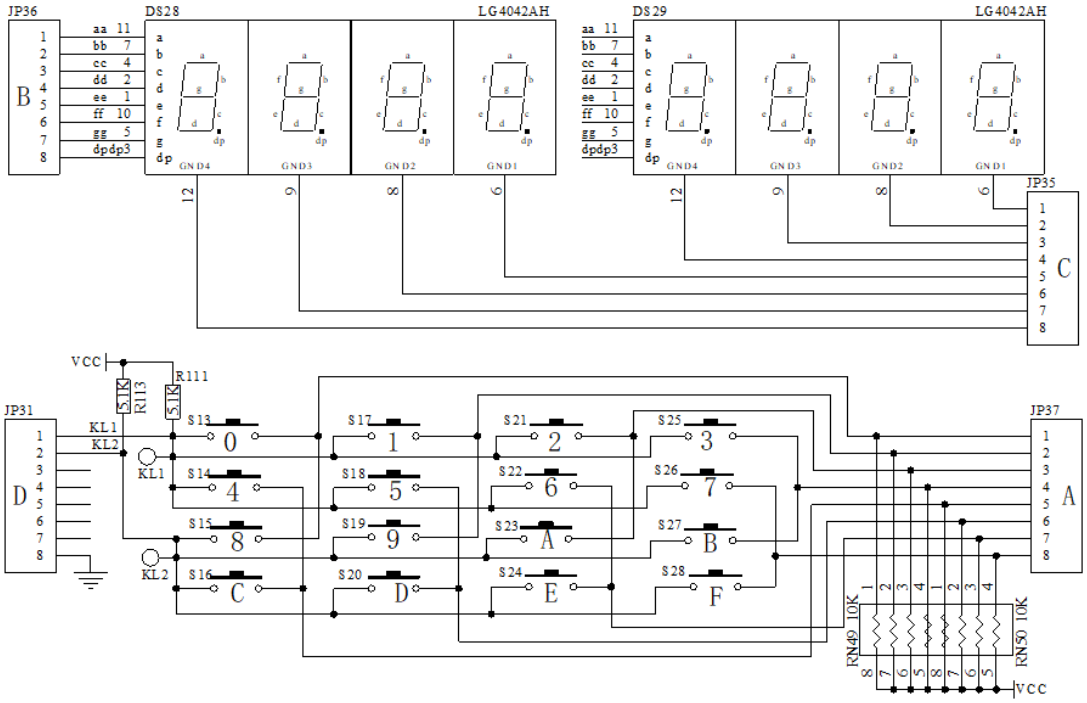


图 3-5 小键盘和数码管原理图

[端口]：A：按键的列线

B：数码管段码

C：数码管选择脚

D：按键的行线

[功能]：扫描获取按键，控制段码和位码在显示屏上显示需要显示的内容。

3.6 C3 区：8251 电路

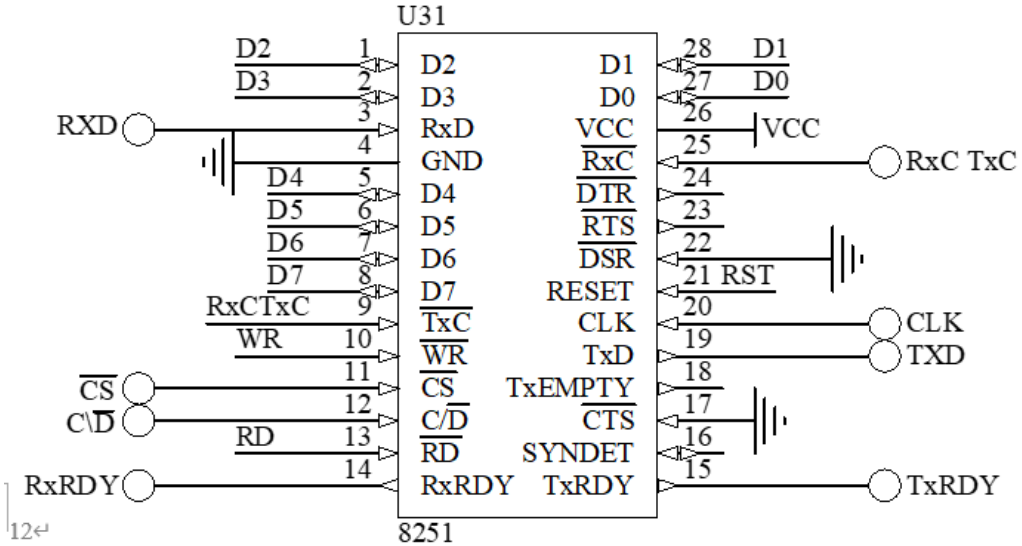


图 3-6 8251 电路原理图

[端口]：CS： 片选信号，低电平有效；

RxC、TxC： 收发时钟；

C/D：命令/数据；

RXD、TXD：串行收发；

CLK：时钟（8251 内部时序）。

TXRDY、RXRDY：查询发送数据或接收数据的信号

[功能]：实现两台设备数据的传输和通信。

3.7 F4 区：发光管、开关

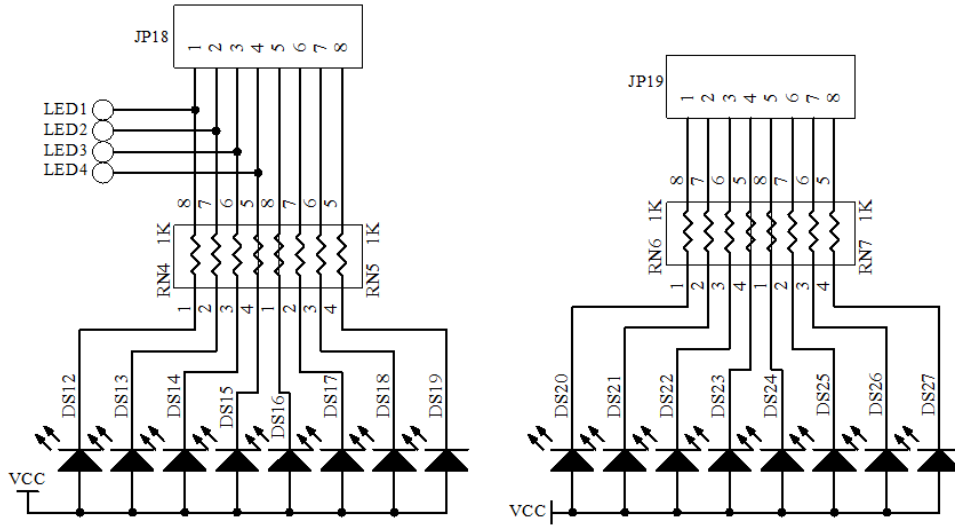


图 3-7 发光管原理图

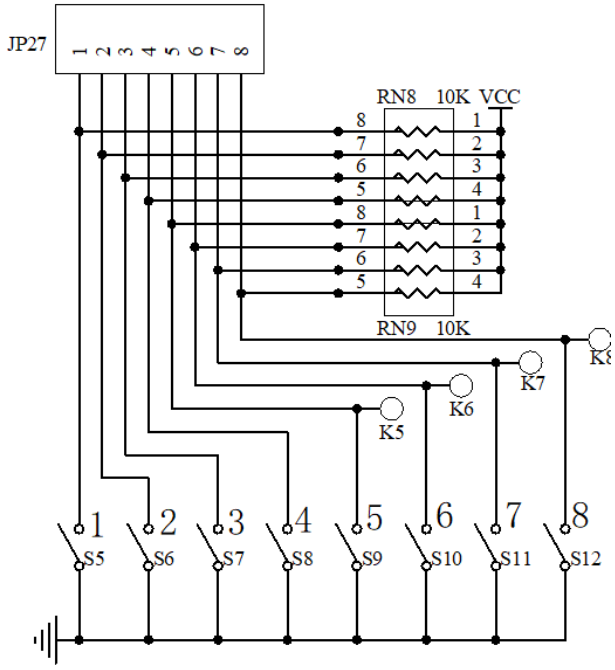


图 3-8 开关原理图

[端口]：发光管控制接口，0—灯亮，1—灯灭

开关控制接口；闭合—0 信号，断开—1 信号

[功能]：开关控制所需变量的输入，发光管观察 8251 通信时的标志信号

四、设计的方案及电路原理图

4.1 设计方案

首先通过查阅资料，了解步进电机的原理，然后分析讨论设计了如下内容：

4.1.1 硬件连线

表 4-1 硬件端口连线分配表

芯片	区域	端口	连接	区域	端口
步进电机	D1 区	A	连接	D3 区	PC4
	D1 区	B	连接	D3 区	PC5
	D1 区	C	连接	D3 区	PC6
	D1 区	D	连接	D3 区	PC7
8255	D3 区	8255 片选 CS*,A0,A1	连接	A3 区	系统 CS1*,A0 ,A1
	D3 区	PC0、PC1	连接	F5 区	KL1、KL2 (控制键盘 2 行)；
	D3 区	JP20(PB0-PB7)	连接	F5 区	A=JP37(1-8) （键盘列线）
	D3 区	(A 口) B=JP16(SEGA-SEGP)	连接	F5 区	B（JP36） （段码）
	D3 区	(B 口) C=JP17(DIG0-DIG7)	连接	F5 区	C（JP35） （位码）
	D3 区	PC0、PC1	连接	F4 区	K5,K6
8253	C4 区	8253 片选 CS*,A0,A1	连接	A3 区	系统 CS2*、A0 、A1；
	C4 区	CLK0	连接	B2 区	1M
	C4 区	GATE	连接	C1 区	VCC
	C4 区	OUT0	连接	B3 区	IR0
8259	B3 区	8259 片选 CS*、A0	连接	A3 区	系统 CS3*、A0
	B3 区	INT、INTA	连接	A3 区	INTR、INTA*
8251	C3 区	8521 片选 CS*,C/D*	连接	A3 区	系统 CS4*、A0
	C3 区	CLK	连接	B2 区	4M
	C3 区	TXC,RXC	连接	B2 区	62.5K
	C3 区	TXD	连接	C3 区	RXD
	C3 区	TXRDY RXRDY	连接	F4 区	LED1 LED2
预设计（未实现）					
0809	C2 区	0809 片选 CS,	连接	A3 区	系统 CS3,
	C2 区	ADDA,ADDB,ADDC	连接	A3 区	A0、A1、A2

	C2 区	CLK	连接	B2 区	500K
	C2 区	IN0	连接	F6 区	0~5V

4.1.2 芯片连接 8086 系统的各端口地址及分工

表 4-2 端口分配及分工表

芯片	硬件端口	端口地址	端口名称	端口作用
8255 [CS1]	A	270H	PORT8255_A	【方式 0 输出】段码；
	B	271H	PORT8255_B	【方式 0 输出】位码；列线；
	C	272H	PORT8255_C	【低 4 位输入】PC0,PC1 –>行线；PC2,PC3 –>开关量输入；【高 4 位输出】PC4,PC5,PC6,PC7 –>电机 A,B,C,D 四相位；
	控制 K	273H	PORT8255_K	写控制字，设置端口输入输出方式
8253 [CS2]	计数器 0	260H	PORT8253_0	分频产生定时中断
	计数器 1	261H	PORT8253_1	没有使用
	计数器 2	262H	PORT8253_2	没有使用
	控制 K	263H	PORT8253_K	写控制字
8259 [CS3]	偶址	250H	PORT8259_0	写控制字：ICW1
	奇址	251H	PORT8259_1	写控制字：ICW2,ICW4,OCW1
8251 [CS4]	偶址	240H	PORT8251_0	数据端口
	奇址	241H	PORT8251_1	控制端口
预设计（未实现）				
0809 [CS5]	IN0	230H	PORT0809	数据端口/控制端口

4.1.3 实验所需变量及参数设置

表 4-3 变量表

变量名	变量类型	变量含义
BITCODE	DB	存放位码值
STEGTAB	DB	存放段码值 0~F、“0.”、“-.”
KEYVALUE	DB	存放键值 0~F、“0.”、“-.”
KEYCODE	DB	存放扫描按键得到的行列码
BUFFER	DB	显示缓冲区
bFirst	DB	停机标志　0：启动　　中　1：停止
bClockwise	DB	转动方向标志　0：逆时针　1：顺时针
bNeedDisplay	DB	是否需要显示新的步数标志　1：需要显示
StepControl	DB	下一次要送给步进电机的相位值

SetCount	DB	##
StepCount	DB	当前步数绝对值
StepDec	DW	通过 DE 设置的步数
SpeedNumber	DB	通过 BC 设置的转速级数
JiShu	DB	通过转速级数计算得到的 8253 计数初值
StyleMove	DB	转动方式选择 0：单相 4 拍　1：双向 4 拍　2：单双 8 拍
ControlStyle	DB	相位值表，用于相位赋值
DONE	DB	D,E 模式下中断完成标志位 0：未完成 1：中断完成

4.1.4 程序模块设计

[1]初始化相关子程序

【INIT8253】8253 初始化——进行相应转速的设置，即对 1M 进行分频

【INIT8255】8255 初始化——初始化键盘及端口输入输出方式

【INIT8259】8259 初始化——中断初始化

【INIT8251】8251 初始化——设置接收机和发送机

【INITMOTOR】步进电机初始化——设置停机标志、显示缓冲区、转动方式

【INTERRUPT_VECTOR】中断向量表初始化——设置向量表偏移地址和段基地址，及保护断点

[2]主程序

【主程序】控制了系统整体的功能，是整个源程序必不可少的部分，它将各个实现了单独的功能的子程序模块连接起来，实现对电机的控制，其中主要包括如下的功能：

判断有无按键按下；

判断电机有无转动一步，进而设置数码管显示的步数；

判断 D,E 模式下设定的步数是否走完，进而决定中断是否结束；

判断电机的启动/停止状态；

判断按键功能和设置步数/转速等等。

[3]键盘和数码管显示相关子程序

【LED_DISPLAY】数码管显示——将显示缓冲区 buffer 通过数码管显示；

【SCANKEY】扫描按键——扫描键盘得到按键的列行码存入 AX；

再通过子程序【TRANSLATE】将列行码转换为键值——将扫描得到的列行码转换成相应键值存到 AL 中；

【DELAY】延时——进行一定时间的延时，消抖

[4]BUFFER 数据获取为变量赋值

【GetBuffer65】设置步数/转速——得到##值送入 SetCount，用于设置步数/转速；

【StepCountSet】得到当前步数值——将显示缓冲区后 4 位转换成 4 位数存

入 StepCount（当前步数值）；

[5]修改电机转动速度

【SetJiShu】设置 8253 计数值——##（SetCount）→SpeedNumber（转速级数）然后通过公式计算得到 JiShu（计数值）；

【SetStyleMove】设置步进电机的节拍方式——单相 4 拍/双向 4 拍/单双 8 拍

[6]中断相关子程序

【TIMERO】中断子程序——根据相位值 StepControl 送入 8255 控制步进电机转动：

将 bNeedDisplay 值置 1，即“已改变步数，需要显示”；

判断是否通过功能键 D 或 E 设置步数；

如果设置了步数，将设置的步数值-1；

发中断 EOI；

【StepControlSet】设置 StepControl（步进电机的下一步相位值）

判断电机转动方式——根据转动方式，设置下一步相位值

【AlterStep】步进电机步数调整

判断转动方向

判断当前步数是正是负

根据转动转动和步数正负对当前步数进行加减

判断当前步数绝对值是否到达 9999，到达则进行反转

[7]停机子程序

【HltFunction】功能 F 显示——按 F 显示全“F”表示停机

[7]8251 通信子程序

【SEND8251】一台设备通过 8251 发送数据给另一台设备

【RECEIVE8251】接收另一台设备传输过来的数据

4.2 电路的原理图

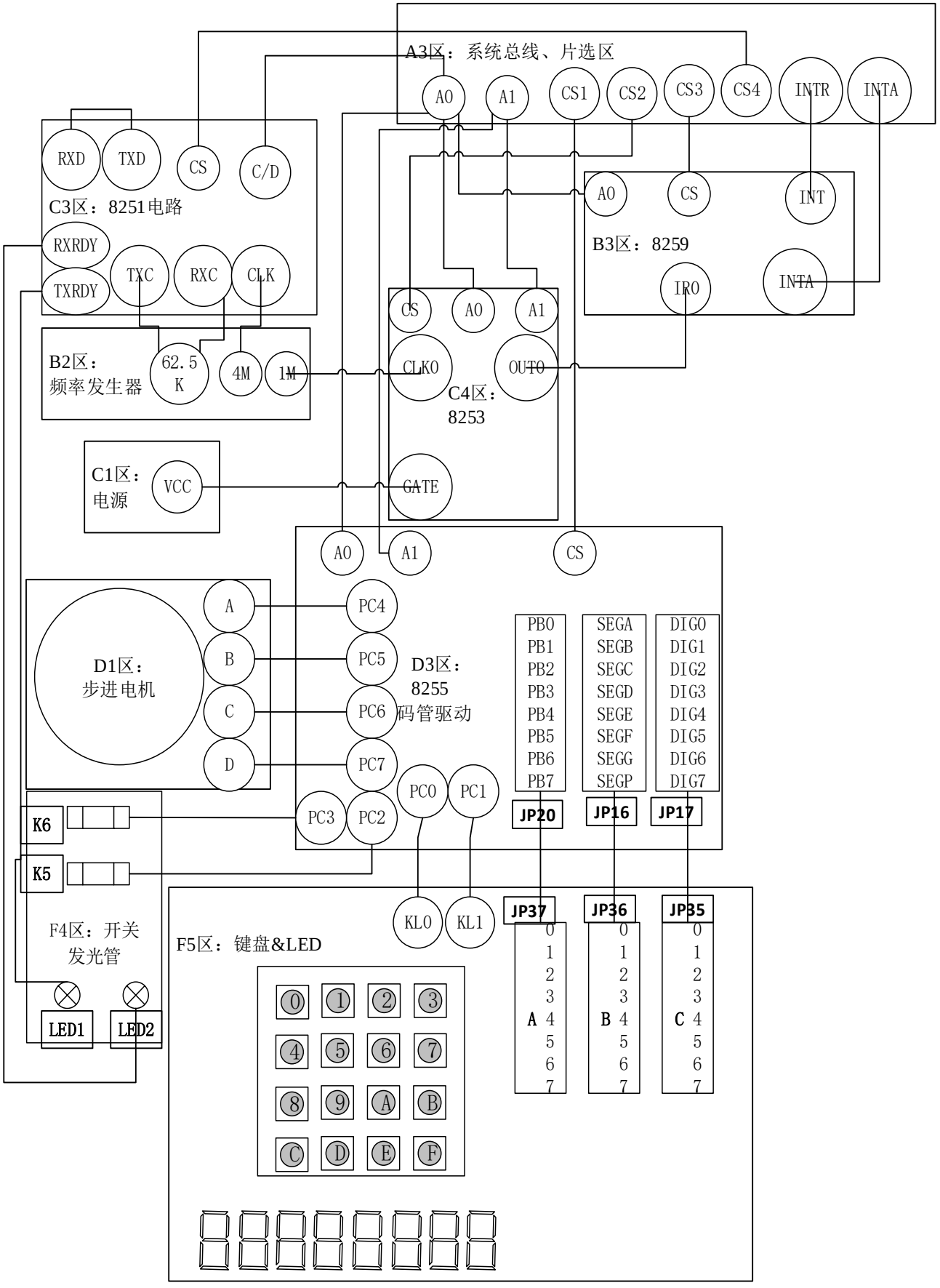

A3区：系统总线、片选区
A0
A1
CS1
CS2
CS3
CS4
INTR
INTA
RXD
TXD
CS
C/D
C3区：8251电路
RXRDY
TXRDY
TXC
RXC
CLK
A0
CS
INT
B3区：8259
IR0
INTA
CS
A0
A1
CLK0
OUT0
C4区：
8253
B2区：
频率发生器
62.5
K
4M
1M
C1区：
电源
VCC
GATE
A0
A1
CS
A
B
C
D
D1区：
步进电机
PC4
PC5
PC6
PC7
D3区：
8255
码管驱动
PC0
PC1
PC2
PC3
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
SEGA
SEGB
SEGC
SEGD
SEGE
SEGF
SEGG
SEGP
DIG0
DIG1
DIG2
DIG3
DIG4
DIG5
DIG6
DIG7
JP20
JP16
JP17
K6
K5
F4区：开关
发光管
LED1
LED2
KL0
KL1
JP37
JP36
JP35
F5区：键盘&LED
0 1 2 3
4 5 6 7
8 9 A B
C D E F

图 4-1 电路原理图

五、软件的流程图

步进电机的综合控制实验需要较多的模块来实现功能，所以在我们第一天对题目要求的仔细分析与思考之后，首先分工实现了一系列初始化模块以及电机步数转速的控制，之后再共同修改以及调试。以下着重描述自己所设计的模块（加*的部分）以及大家共同编写的部分（加【】的部分）

5.1 【主程序】

1）初始化部分-------首先是对使用的芯片以及键盘、步进电机等进行初始化，为后面的实验做准备。

2）等待设置步进电机节拍方式-------调用子程序对设置的节拍方式进行读取判断，如果设置了节拍方式则按照该节拍方式初始化电机，否则继续等待节拍方式的输入。

3）***START_1：*数码管显示+开始扫描按键**

[1]若有按键，则跳转到按键的功能进行设置；

[2]若没有按键，则判断标志位“bNeedDisplay”决定继续等待按键还是继续判断电机的状态位“bFirst”；

[3]根据具体的功能设置步进电机的参数，控制步进电机的转动。

4）我初始编写部分：

扫描按键和数码管显示的框架部分.

5）我修改部分：

D,E 模式下设置步数只能设置为 0~99，若速度过快则电机很快就停止了 ，所以将变量步数 StepDec 设置为字变量，若输入 10，则控制步进电机转 10*10=100 步后再停止。

B,C 模式下将 StepDec 设置为“-1”代表没有设置步数，此时电机速度控制通过重设计数初值再对 8253 的计数器 0 重设计数初值改变中断信号的频率即改变步进电机的速度。

6）具体流程见流程图。

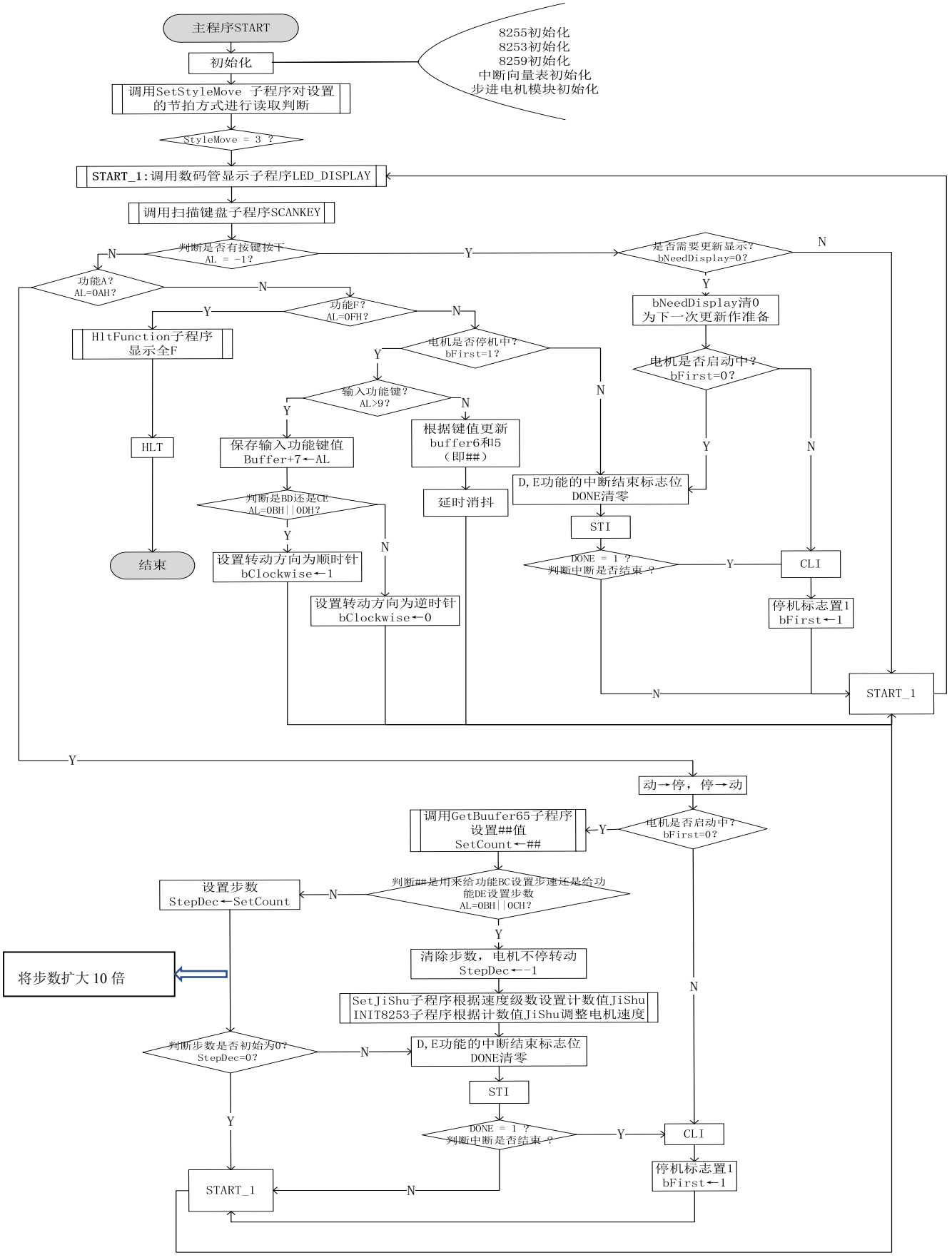

主程序START
初始化
8255初始化
8253初始化
8259初始化
中断向量表初始化
步进电机模块初始化
调用SetStyleMove 子程序对设置的节拍方式进行读取判断
StyleMove = 3 ?
START_1:调用数码管显示子程序LED_DISPLAY
调用扫描键盘子程序SCANKEY
判断是否有按键按下 AL = -1?
是否需要更新显示? bNeedDisplay=0?
功能A? AL=0AH?
功能F? AL=0FH?
HltFunction子程序 显示全F
HLT
结束
电机是否停机中? bFirst=1?
输入功能键? AL>9?
保存输入功能键值 Buffer+7←AL
判断是BD还是CE AL=0BH||0DH?
设置转动方向为顺时针 bClockwise←1
设置转动方向为逆时针 bClockwise←0
根据键值更新 buffer6和5 (即##)
延时消抖
bNeedDisplay清0 为下一次更新作准备
电机是否启动中? bFirst=0?
D,E功能的中断结束标志位 DONE清零
STI
DONE = 1 ? 判断中断是否结束 ?
CLI
停机标志置1 bFirst←1
START_1
动→停，停→动
调用GetBuufer65子程序 设置##值 SetCount←##
电机是否启动中? bFirst=0?
判断##是用来给功能BC设置步速还是给功能DE设置步数 AL=0BH||0CH?
设置步数 StepDec←SetCount
将步数扩大 10 倍
清除步数，电机不停转动 StepDec←-1
SetJiShu子程序根据速度级数设置计数值JiShu
INIT8253子程序根据计数值JiShu调整电机速度
判断步数是否初始为0? StepDec=0?
D,E功能的中断结束标志位 DONE清零
STI
DONE = 1 ? 判断中断是否结束 ?
CLI
停机标志置1 bFirst←1
START_1
Y
N

图 5-1 主程序流程图

5.2 *初始化模块

5.2.1 *INIT8255

该模块对 8255 芯片进行了初始化。由于对于步进电机的功能控制需要使用实验箱上的 2 行*8 列的小键盘，所以必须要使用 8255 的端口来对键盘和显示屏进行初始化。

数码管显示时需要 8255 输出段码和位码，所以需要两个端口工作在输出方式。

扫描按键时使用“列扫描”方式，所以需向列线输出，读入行线，所以需要有端口工作在输入方式，而键盘只有两行，所以可以控制 C 端口的低位输入。

控制步进电机时需要 4 个相位的输入，但又必须单独控制否则会对电机转动有影响，所以设置 C 端口的高 4 位工作在输出方式。

由于在之后实现功能的过程中设置了电机转动的三种节拍方式又增添了新的变量 StyleMove，所以为了使电机的设置更加灵活，并且 8255 的端口得到充分的运用，我们通过开关将设置的 StyleMove 值通过 C 端口读入，再进行其他设置。

即得到下表：

端口	工作方式	功能
A	方式 0 输出	（1）数码管显示时作为段码输出端口
B	方式 0 输出	（1）数码管显示时作为位码输出端口
		（2）扫描按键时作为列线输出全‘0’
C	低 4 位输入，高 4 位输出	（1）PC0PC1：扫描按键时作为行线输入
		（2）PC2PC3：作为变量 StyleMove 的输入
		（3）PC4~7：电机控制时作为相位值输出

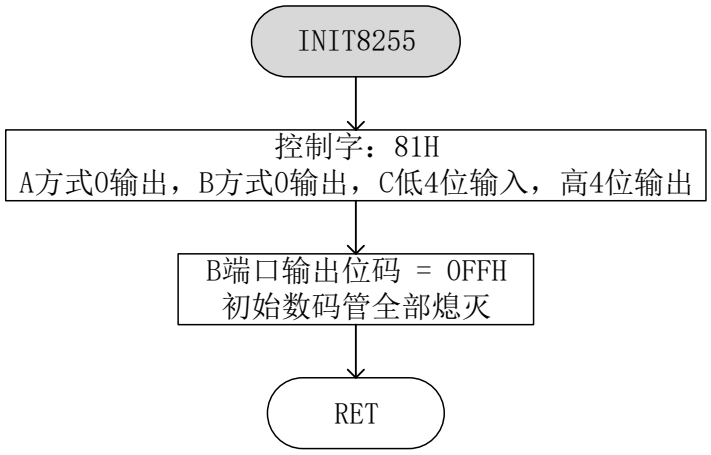


图 5-2 INIT8255 初始化流程图

5.2.2 *INIT8253

该模块对 8253 芯片进行了初始化。我们使用 8253 对接入的频率信号进行分频输出到 8259 作为定时中断信号，所以需要周期性的频率信号，而且在控制电机的过程中不需要 GATE 信号的改变，所以选择了使计数器工作在方式 2 下，这样就能够在 GATE 保持为高时，只需要写入一次计数值，就能够连续性的输出负脉冲。

由于我们初始连接的 CLK0 为 1M，根据我们的步进电机当转速设置为 n 时，对应的频率为 0.8*n Hz，周期为 1.25/n 秒/步，所以初始想要设置两个计数器来实现中断周期时长的控制即转速的控制，但在实际测试中发现步进电机在频率过于低时不会转动，最后在测试时得到计数值 JiShu 的大致的可以使电机进行转动的范围，在此基础上对 JiShu 进行控制，具体见 SetJiShu 子程序中介绍。

所以对于 8253 的初始化只需要使用一个计数器 0，使其工作在方式 2，使用了二进制计数输入得到的计数值即可。

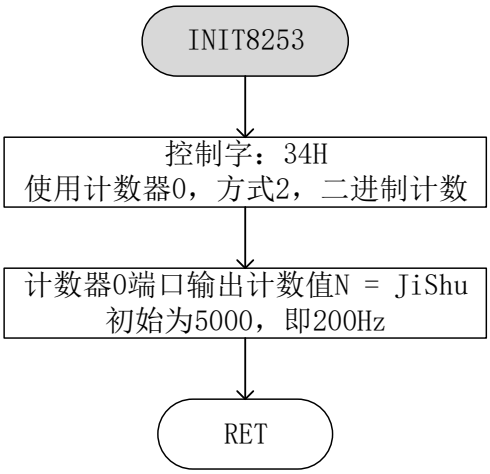


图 5-3 INIT8253 初始化流程图

5.2.3 *INIT8259

该模块是对芯片 8259 的初始化。我们需要使用 8259 产生中断然后控制电机的转动，所以只需要开放一个中断源接收 8253 传送过来的周期性的信号产生系统中断，系统跳转到中断子程序对电机进行一系列的控制。

没有需要使用多片 8259 级联的事件，所以 8259 单片方式下即可;

设置一个中断类型号，然后再初始化中断向量表；

设置为缓冲方式和非自动结束中断方式，非自动结束方式下，在中断服务完毕之后，中断服务寄存器 ISR 相应位不会自动清零，必须在中断服务程序结束之前，向 8259A 发中断结束命令 EOI，把对应位清零。

设置开放中断源 IR0，只开放一个中断源，不需要设置中断优先级。

所以需要写初始化命令字 ICW1,ICW2,ICW4 以及操作命令字 OCW1。

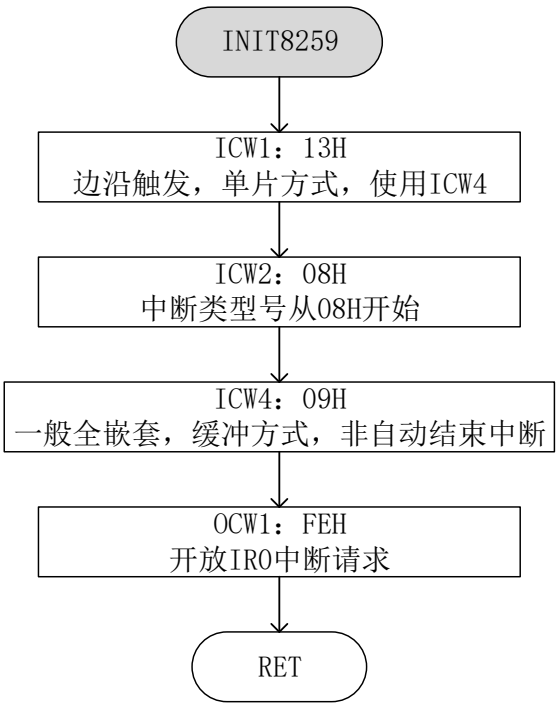


图 5-4 INIT8259 初始化流程图

5.2.4 *INITMOTOR

该模块是对步进电机的初始化以及显示步进电机执行功能的状态的数据缓冲区 BUFFER 初始化的。其中根据方案设计时设置的一系列步进电机的相关参数的含义需要对电机的状态进行初始化。

[1]首先初始时步进电机的状态应该为“停止”状态，所以变量 bFirst 置为‘1’代表电机初始状态；

[2]初始时应设置好下一次要传送给步进电机的相位值，这样在电机启动时能够立即传送给步进电机控制步进电机转动；

[3]初始时需要向数据缓冲区 BUFFER 里写入初始化显示的数据，同时也是步进电机初始功能方式的选择，这样在初始化后按下启动键电机就能够立即按照初始化设置开始转动.这里我们初始化是使步进电机步数为 0.0000，并选择了命令 D99，即顺时针转动 99*10 步后电机停止转动；

[4]其中电机的初始转动方向是通过判断 BUFFER+7 里的数据即初始选择的命令方式确定的。

电机初始化是一个很灵活的模块，一开始主要是为了在软件调试过程中测试各个功能设置的正确性。

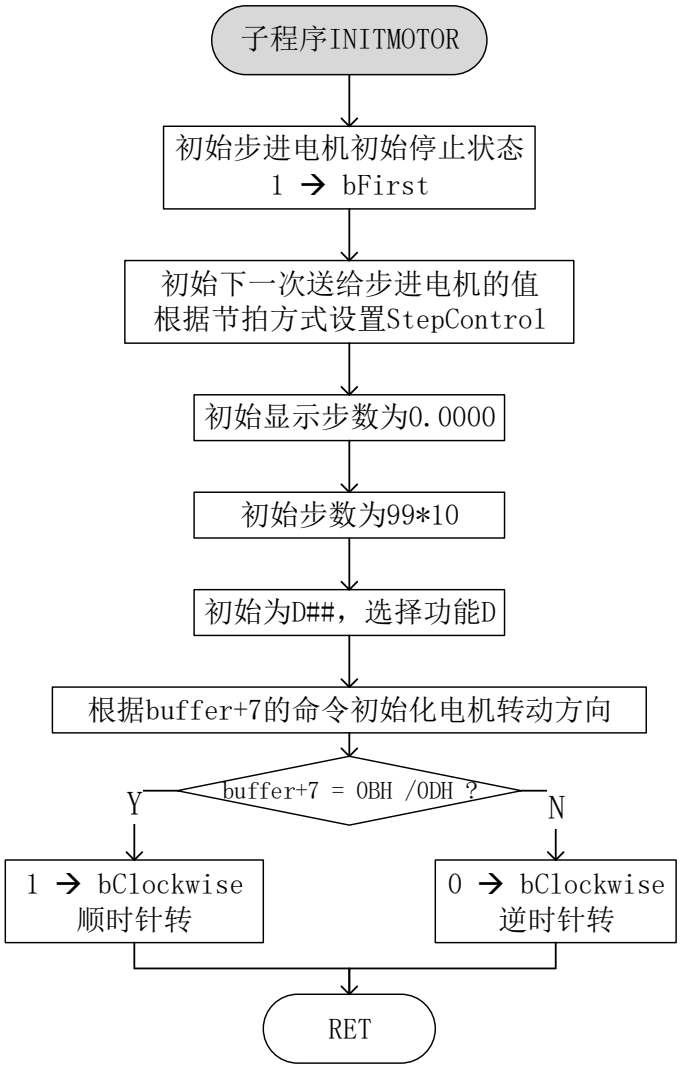


图 5-5 INITMOTOR 初始化流程图

5.2.5 *INTERRUPT_VECTOR

该模块是对中断向量表进行初始化，在我们编写了对应的中断子程序之后，需要将对应中断子程序的地址存入中断向量表中，这样在程序执行产生中断时，才能够跳转到相应的中断子程序执行，否则不可以，所以中断向量表的初始化是必不可少的。

然后对于中断向量表的初始化，主要就是如下步骤：

[1]设置中断向量表的段基地址；

[2]根据 8259A 初始化中的中断类型号设置中断向量表的地址；

[3]存入中断子程序的偏移地址和段基地址。

这里由于我们设置的中断类型号为 08H，所以中断向量表的地址为 08H*4=20H，设置的中断子程序名称为 TIMERO 即首先存入它的偏移地址，再存入它的段地址，这样中断向量表的初始化就完成了。

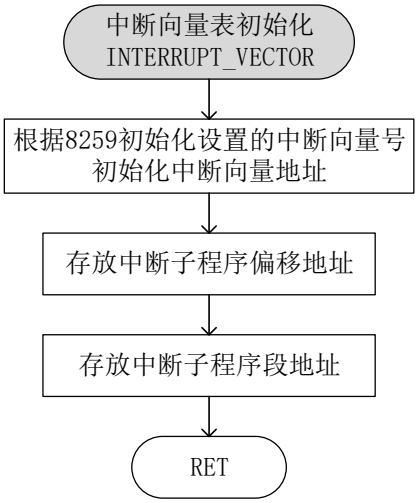


图 5-6 INTERRUPT_VECTOR 初始化流程图

5.2.6 *INIT8251

该模块是对芯片 8251 进行初始化。对于两台通信的设备，一个发送数据，一个接收数据使得两台设备能够同时控制步进电机进行转动。8251 通信可以选择中断方式下也可在查询方式下，但由于我们是通过中断控制电机的转动的，所以为保证电机的正常操作，我们选用查询方式对 8251 传送的数据进行输入输出操作。

8251A 用查询方式输入时，先设置模式字，然后设置控制字，就可以对 8251 进行初始化。对于 8251 数据发送器和 8251 数据接收器具体的初始化操作：

[1]首先都进行复位操作（包括向控制端口写 3 次 0 和软复位命令）

[2]写入模式字 4EH，即异步方式，不带奇偶校验，1 位停止位，字符长度位 8 位（由于需要传送字节量）；

[3]对于发送器，只需要允许发送，请求发送即可，则控制字为 21H；对于接收器，首先要清除错误标记，然后设置允许接收以及数据终端已准备好，则控制字为 16H。

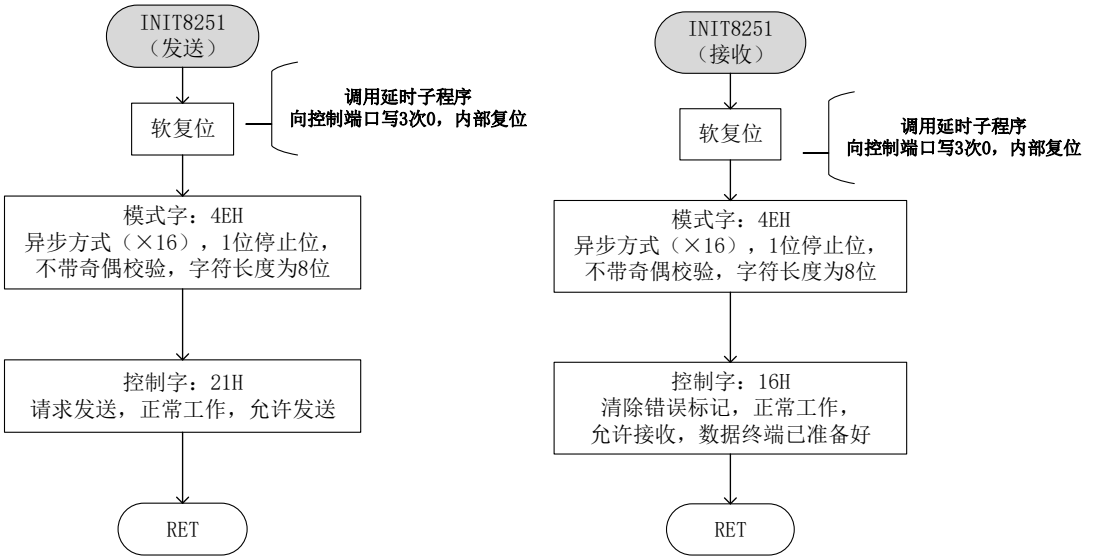


图 5-7 INIT8251 初始化流程图

5.3 SetStyleMove

为了使电机的节拍方式每次启动前能够较为自由的进行控制我们将其通过开关量输入。

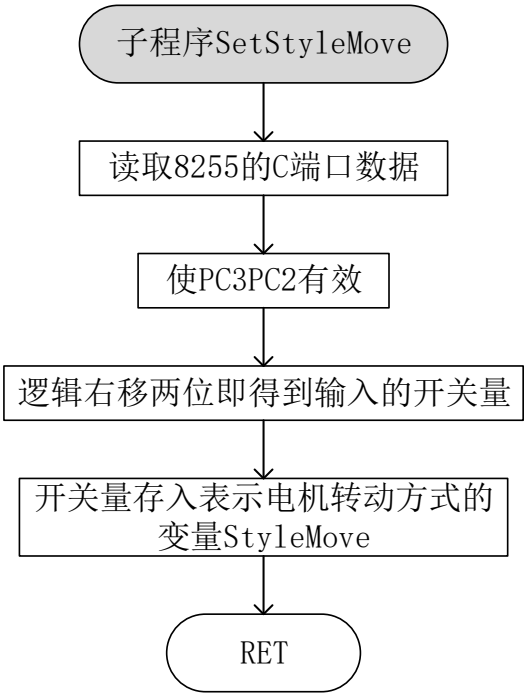


图 5-8 SetStyleMove 流程图

5.4 *LED_DISPLAY

该子程序是对缓冲区的内容进行查表转换然后将其输出到数码管上显示。

多位数码管显示程序程序的一般流程为：

[1]设置位码都无效，熄灭所有数码管；

[2]将一个数码管的自行吗送入段码端口；

[3]设置位码，点亮一个数码管，点亮一个数码管；

[4]适当延时后，重复以上过程。

按照该方法编写了该数码管多位显示不同的子程序。其中初始时需要设置段码表 SEGTAB，位码 BITCODE，以及需要显示的数据缓冲区 BUFFER。

根据题目要求，由于数码管由低往高第 4 位显示的数据 0/1 表示步数数据的正负，所以为了区分，在段码表中增加【0.】【—.】的段码显示分别为 40H 和 3FH ，为了便于在程序中进行查表查找，如要显示【0.】【—.】，即对应的 BUFFER+4 的内容分别写入 10H,11H 即可查表转换得到对应的段码显示在数码管上。

其中必须调用 DELAY 子程序进行延时才能正常显示否则会过快导致乱码显示数据。

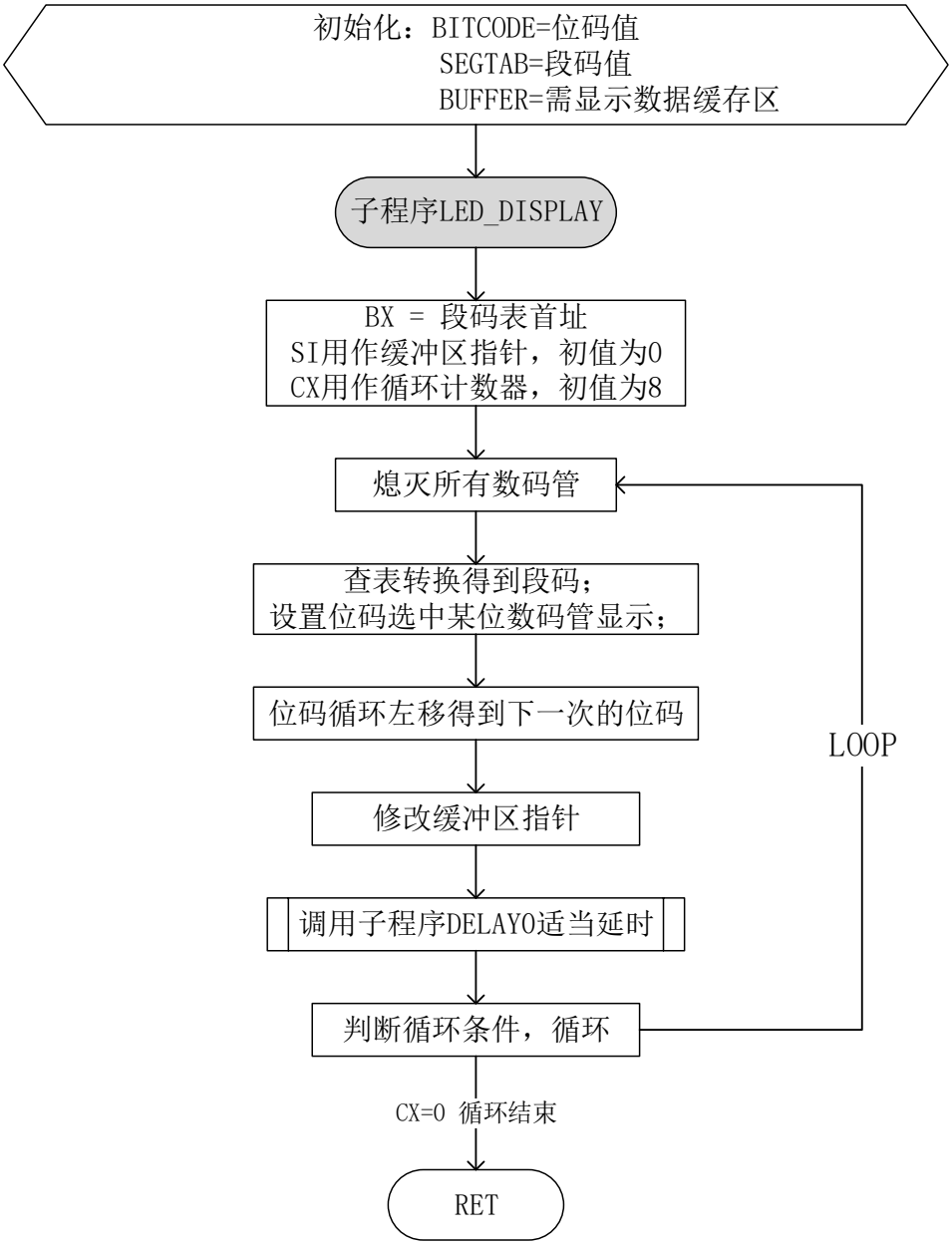


图 5-9 LED_DISPLAY 流程图

5.5 *SCANKEY

该模块是扫描按键的子程序。在前面对于 8255 的初始化中可了解到实验箱可以实现的是“列扫描”扫描按键。以下是列扫描按键的步骤：

[1]向列端口输出全“0”，并读取行端口，如果行端口信息为全 1，则退出；

[2]延时消抖；

[3]再次读取行端口，如果全“1”，说明无键按下；

[4]逐根列线输出“0”，并读取行端口，直到某次读到的行信息有 1 位为“0”，识别出按键。

[5]形成列行码

[6]调用 TRANSLATE 子程序对获得的按键的列行码进行转换得到键值存入 AL 中。

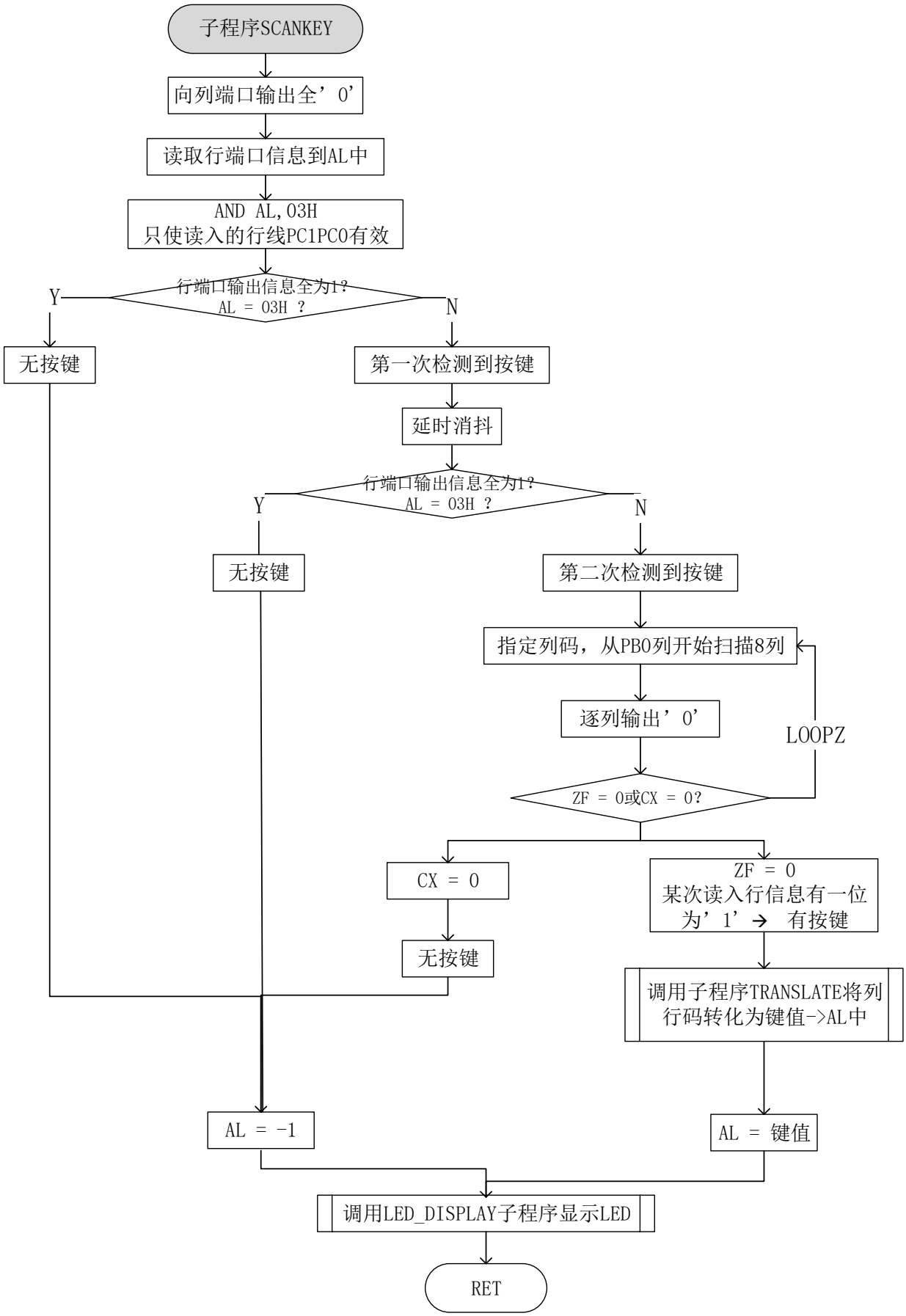

子程序SCANKEY
向列端口输出全’0’
读取行端口信息到AL中
AND AL,03H
只使读入的行线PC1PC0有效
行端口输出信息全为1?
AL = 03H ?
Y
N
无按键
第一次检测到按键
延时消抖
行端口输出信息全为1?
AL = 03H ?
Y
N
无按键
第二次检测到按键
指定列码，从PB0列开始扫描8列
逐列输出’0’
LOOPZ
ZF = 0或CX = 0?
CX = 0
无按键
ZF = 0
某次读入行信息有一位
为’1’→ 有按键
调用子程序TRANSLATE将列
行码转化为键值->AL中
AL = -1
AL = 键值
调用LED_DISPLAY子程序显示LED
RET

图 5-10 SCANKEY 流程图

5.6 *TRANSLATE

该模块是对扫描按键获取的列行码进行转换的子程序。由于列行码较难在程序中进行判断跳转执行对应的操作，所以在获取按键时需要得到按键的键值，这样汇编程序就会便于书写和理解，所以在内存中定义对应的列行码表和键值表通过查表转换得到按键的键值。

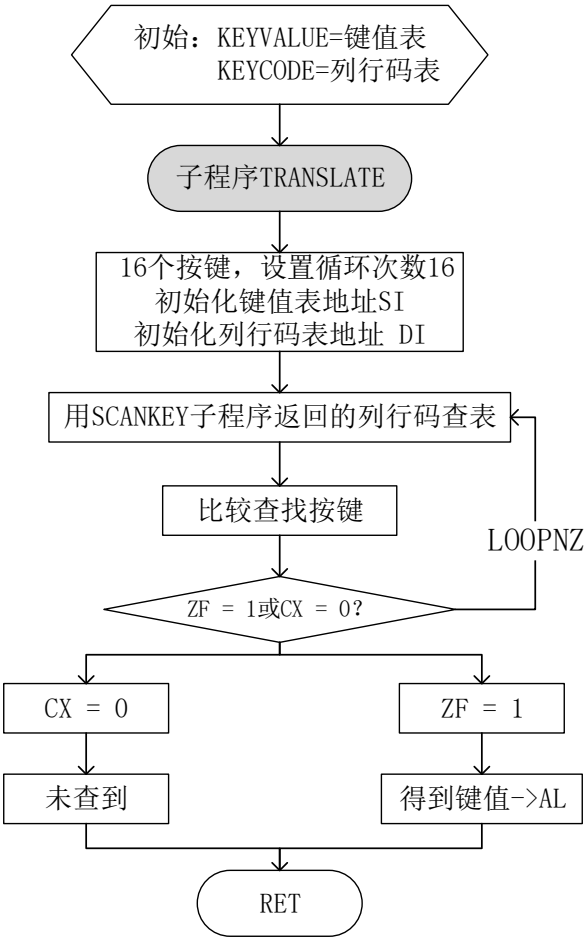


图 5-11 TRANSLATE 流程图

5.7 GetBuffer65

在通过按键输入转速/步数时，通过子程序 GetBuffer65 将输入的十进制数据存入对应的变量 SetCount 中。

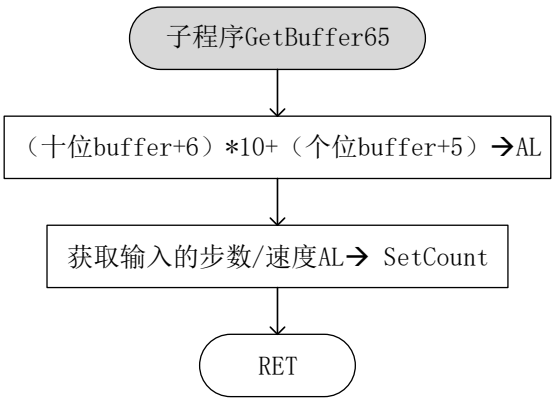


图 5-12 GetBuffer65 流程图

5.8 SetJiShu

我负责的部分【公式推导】：

在测试时由于设置计数初值过大过小都会造成电机不转动，所以根据对速度计数和计数值之间关系分析推出如下公式来得到计数初值：

$$JiShu = 5000 + 100 * (101 - SpeedNumber)$$

[1]首先对于输入的速度存到了共享变量 SetCount 中，其范围为 0~99；

[2]将 SetCount 对应地转换为速度级数即对其进行加 1 操作，存入 SpeedNumber 中；

[3]由于测试中发现电机在频率信号差距较小的范围内，肉眼可见的电机的转速变化较小，所以在测试时发现计数初值为 5000，即 200Hz 下步进电机转动较快，而在计数初值为 15000，即 200/3Hz 时，步进电机的转速明显变慢，所以引入了一个表示“通过实际测试的计数初值的范围”的一元一次函数对输入的速度级数进行转换得到对应的步进电机转动的速度。

具体如下：

初始为200Hz，即0.005 秒/步
然后根据速度级数对应JiShu的范围15000~5100
则【单4拍和双四拍】对应的中断周期范围为0.015秒/步~0.0051秒/步
则【单双8拍】对应的中断周期范围为0.0075秒/步~0.00255秒/步

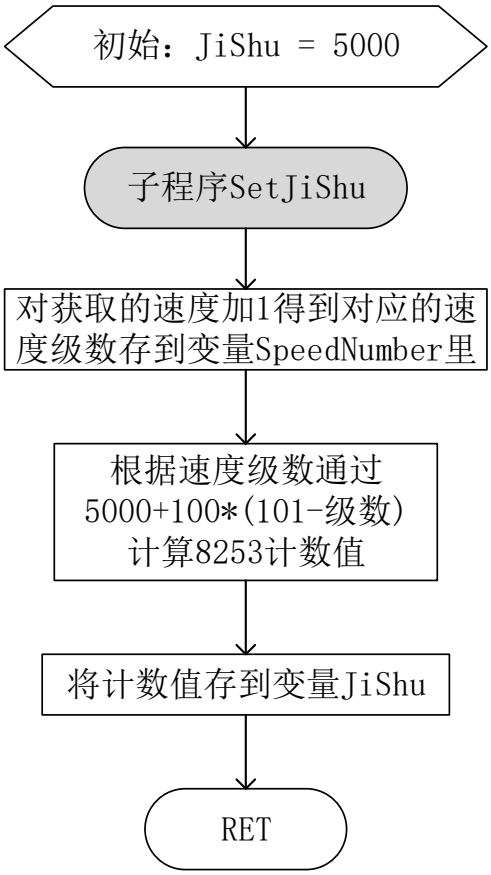


图 5-13 SetJiShu 流程图

5.9 StepCountSet

获取数码管上显示的目前电机的转动步数，存入变量 StepCount 中。

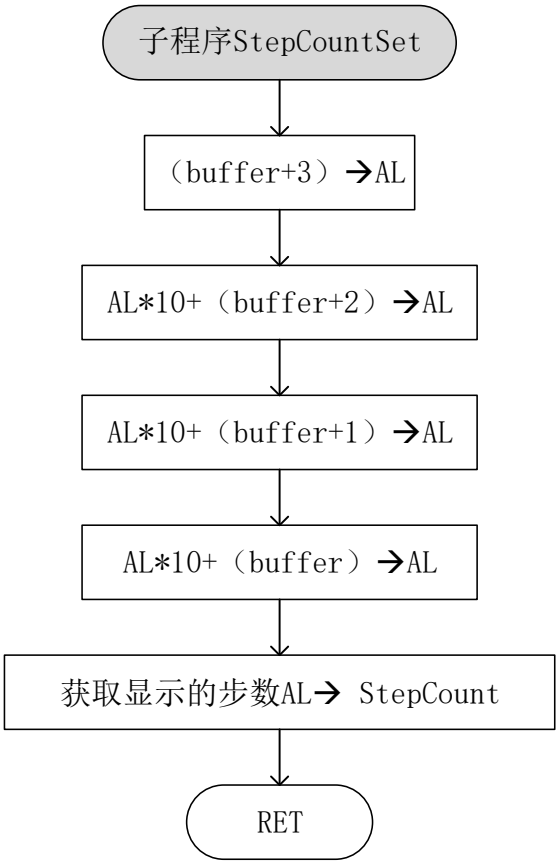


图 5-14 StepCountSet 流程图

5.10 【DELAY 系列】

延时子程序：

DELAY0：延时 5ms，使多位数码管同时显示；

DELAY1：延时 250ms，用于防止输入数字时速度过快；

DELAY2：延时 100μs，用于扫描按键时按键消抖；

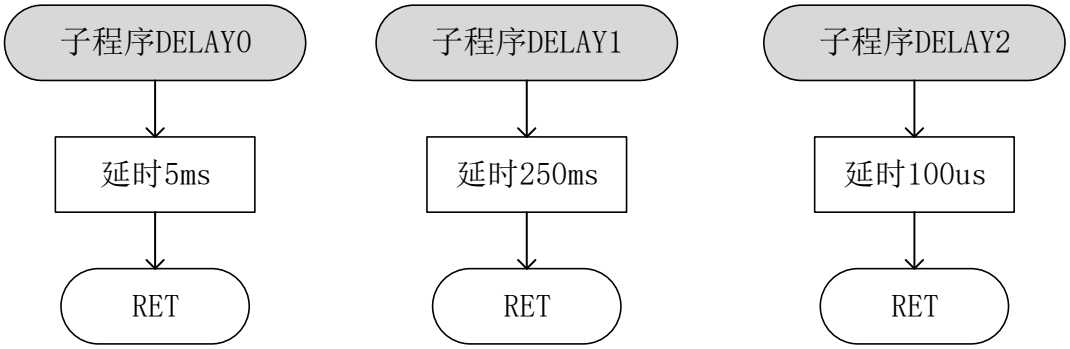


图 5-15 DELAY 系列流程图

5.11 *HltFunction

该子程序用于在数码管上显示全 F，表示此时停机，程序结束。

具体步骤如下：

[1]熄灭所有数码管（每次重新显示前必须执行的操作）；

[2]输出“F”的段码；

[3]输出位码全 0，所有数码管都显示“F”

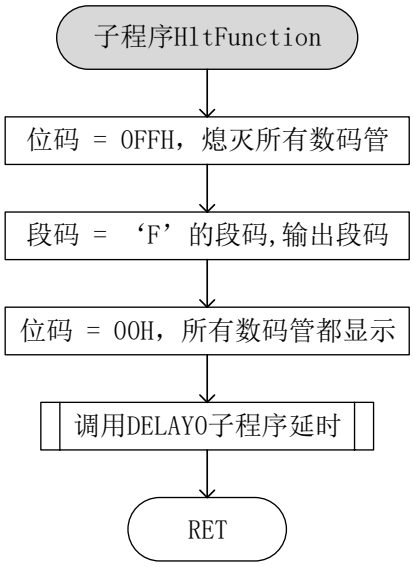


图 5-16 HltFunction 流程图

5.12 StepControlSet

该模块实现了对步进电机下一步节拍值的设置。

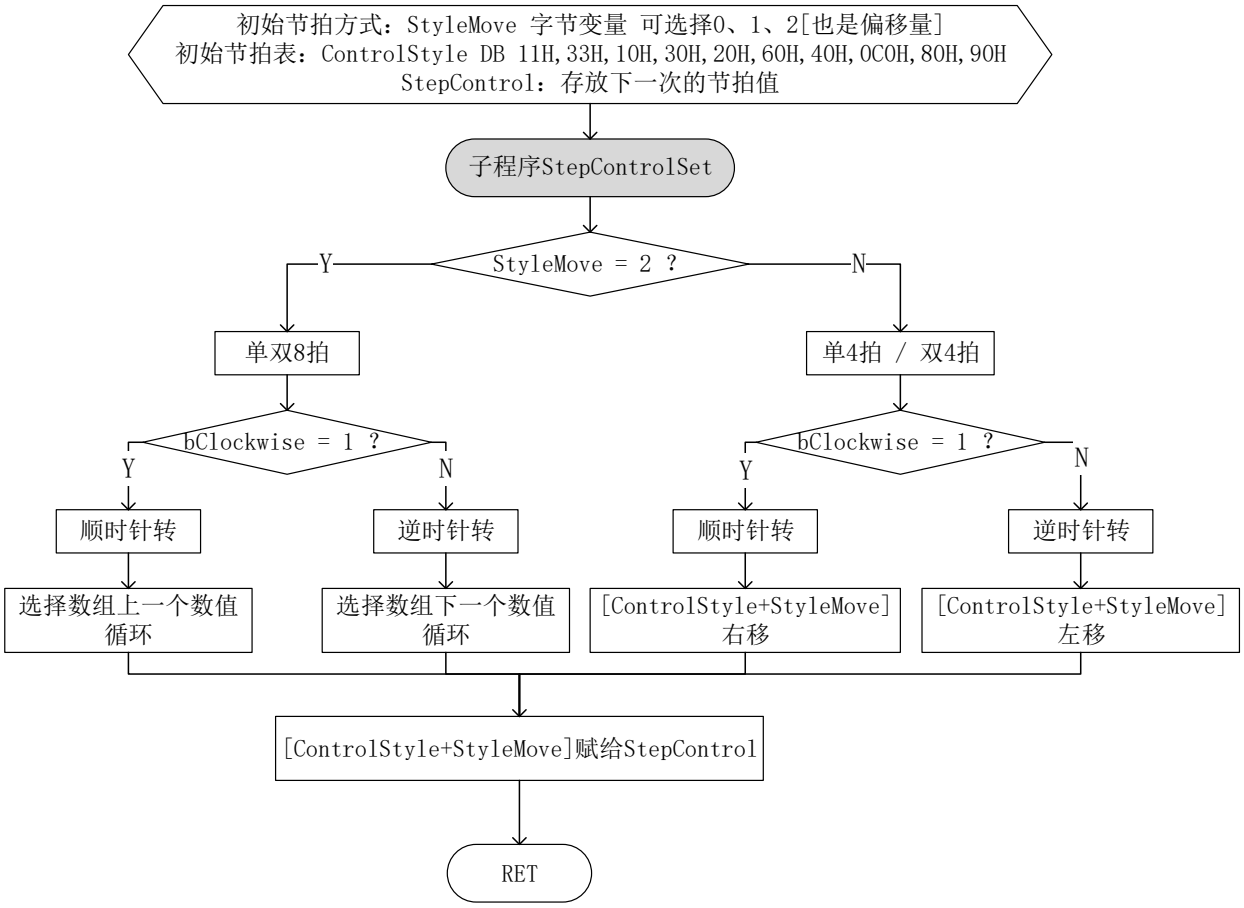


图 5-17 StepControlSet 流程图

5.13 AlterStep

该子程序用于对数码管显示的步数进行更新操作。

我修改的部分：提出在正负切换时对特殊跳转的设置，即-0001 到 00000 的跳转，以及 00000 到-0001 的跳转。这样显示的数据范围就为[-9999，09999].

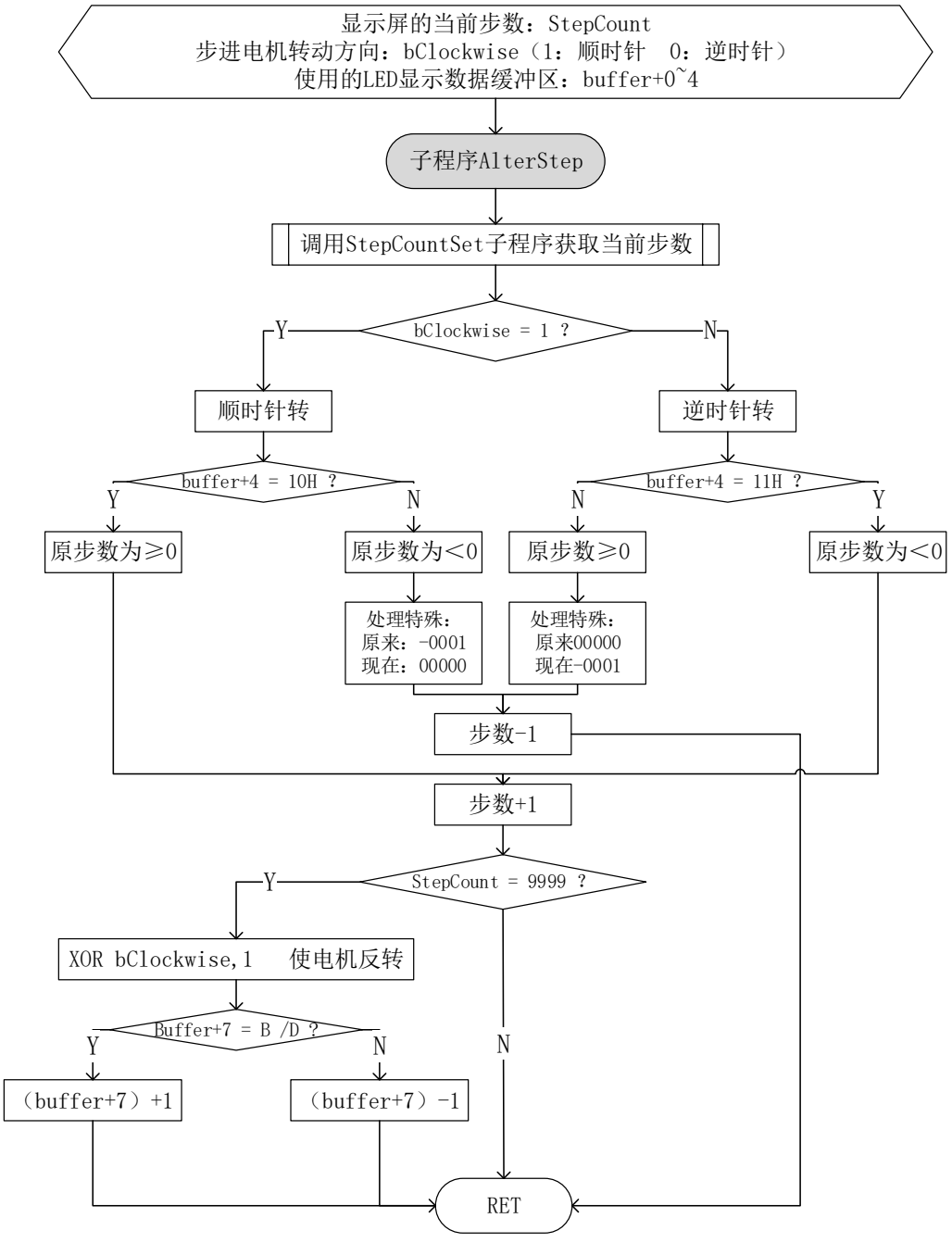
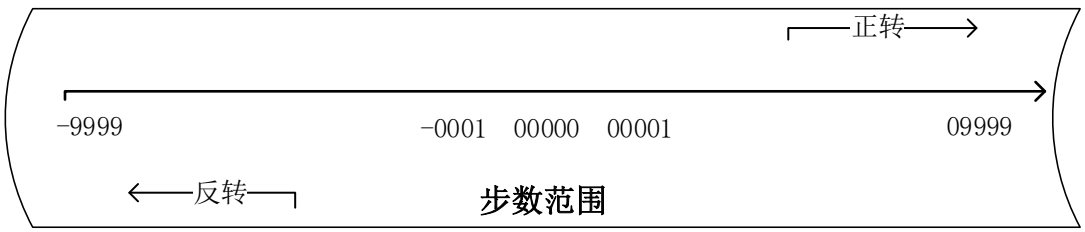


图 5-18 AlterStep 流程图



5.14 【TIMERO】

中断子程序主要执行向步进电机输出相位的操作，在对此调试与修改之后，

将获取下一步的相位值以及步数更新的子程序移入中断程序进行判断，解决了步数显示不同步的问题。

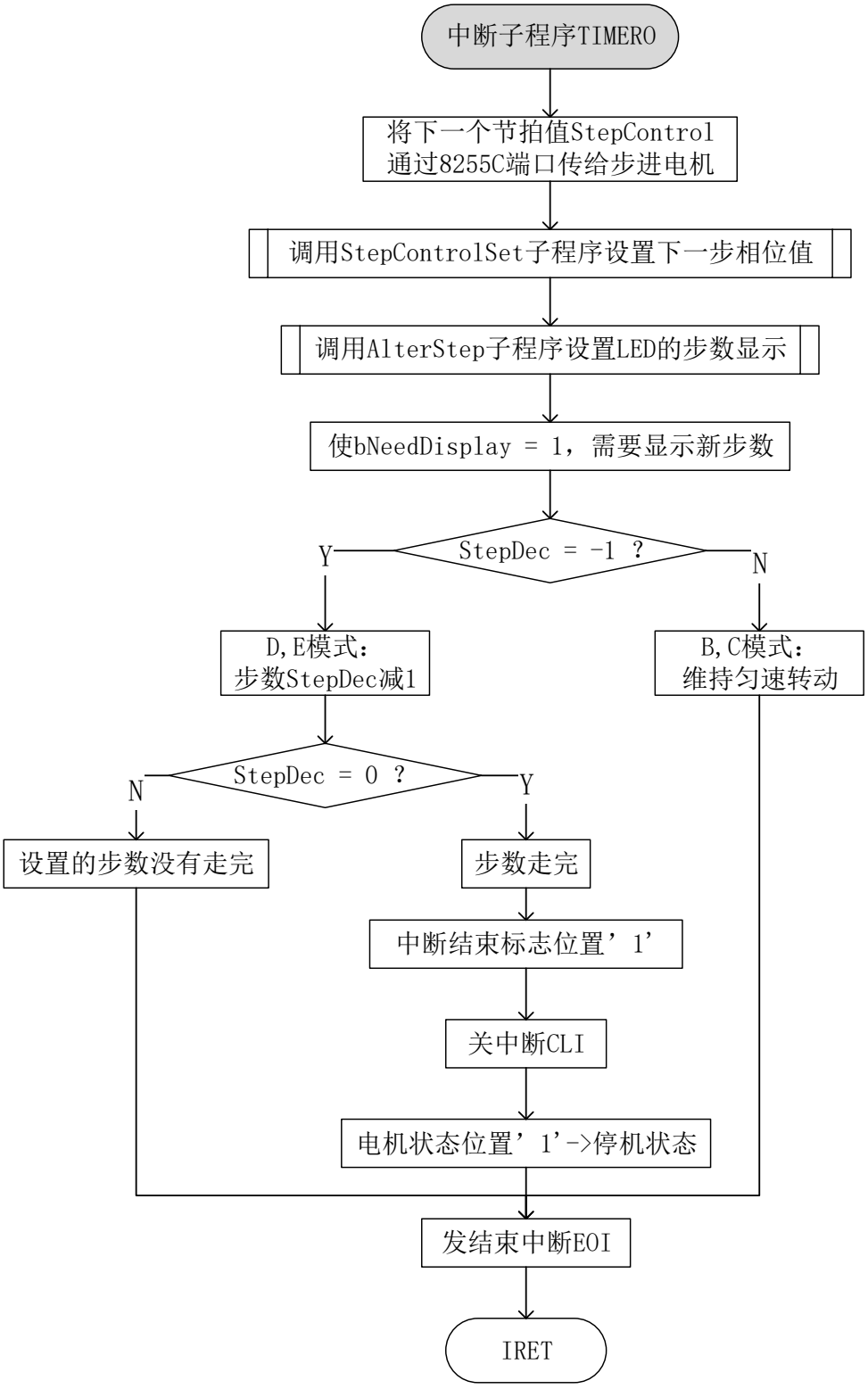


图 5-19 TIMER0 流程图

5.15 *SEND8251

初始化之后读状态字，查询到 TXRDY（发送器准备好引脚）为 1，可以从数

据端口发送数据；

我们想要实现一个 8086 系统开启后，即将其 BUFFERsend 数据缓冲区的数据通过其 8251 将并行数据转换为串行数据然后发送给另一个 8086 系统的 8251；

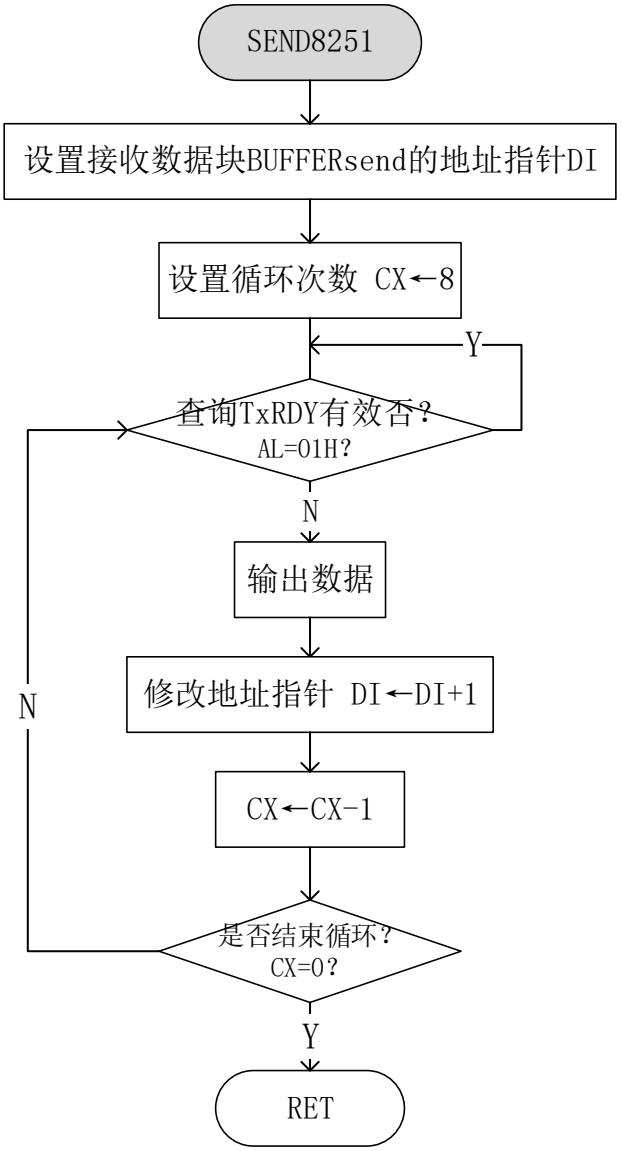


图 5-20 SEND8251 流程图

5.16 *RECEIVE8251

初始化之后读状态字，查询到 RXRDY（接收器准备好引脚）为 1，可以从数据端口接收数据；

一个 8086 系统将从另一个 8086 系统接收到的串行数据转换为并行数据存到相应的变量中，对其步进电机进行初始化，然后该电机便可以启动，并可执行一系列的命令。

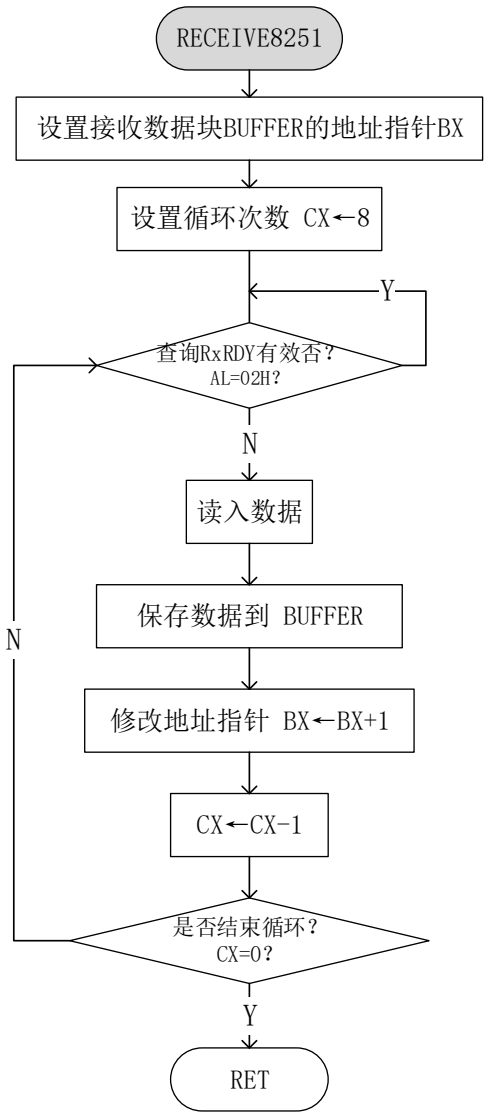


图 5-21 RECEIVE8251 流程图

六、软硬件调试过程

6.1 设计的软硬件调试过程及方法

1）首先测试所用连线是否通路。

2）关闭电源后，硬件连线。

3）成功编译.asm 文件后，装载.dob 文件，进行单步测试。

4）根据单步测试的每一步结果，观察是否符合预期需求：

若符合预期所想，进行第 5 步；

否则，针对错误的结果，对源程序进行相应的修改，重新编译加载程序，进行第 4 步，直到符合预期为止。

5）全速运行程序，观察是否符合预期需求；

若符合预期所想，进行第 6 步；

否则，针对错误的结果，对源程序进行相应的修改，重新编译加载程序，

进行第 4 步，直到符合预期为止。

6）思考是否可以对其进行改进，若对其改进，跳转到第 4 步，否则程序测试成功并已完成。

6.2 调试过程中出现的问题及解决方法

6.2.1 PC 指针超出范围

由于一些子程序忘记写 RET，虽然可以编译程序，但会出现指针跑飞的现象，出现 PC 指针超出范围的错误。

【解决方法】仔细检查源程序，加上 RET。

6.2.2 设置功能参数问题

功能参数为两位数，当只键入一次数字值时，会出现两个数码管上同时修改为键入值，比如设置“10”，会出现设置成“11”的情况。

【解决方法】在键入 0~9 时进行适当的延时。

6.2.3 步数误差问题

当电机处于 DE 功能下时，理应在指定步数走完后停止，实际发现电机总是比设置的步数多走了几步之后才会停止。

【原因分析】经过多次调试，我们猜测，应该与中断频率有关。当转速较慢时，误差较小；转速较快时，误差较大。中断速度太快，导致主程序判断不及时，必然存在步数误差。

6.2.4 转速问题

当设置的转动频率过快或过慢时，步进电机将处于停止状态。

虽然速度可设置 1~100 级，但变化幅度并不大。

【解决方法】通过合理的数学公式（已在前文给出）将设置的速度级数转换成合理的计数值，使电机可在 1~100 级的速度下转动。

6.2.5 跳转范围越界

在调试过程中，出现过主程序太长，导致无法通过条件转移指令跳转到相应位置的问题，即超出了条件转移指令的下一条指令到目的地址之间的相对位移量（必须在-128~+127 内）。

【解决方法】在不影响主程序逻辑结构的前提下，将缩短条件转移指令和跳转模块的距离。

6.2.6 按键控制子程序问题

一开始我们的设计思想是：扫描按键后，通过调用 ChooseControl 子程序进行功能的判断和设置。但在实际调试过程中，总存在问题导致电机无法按照指定的功能正常启停。

【解决方法】将 ChooseControl 子程序的功能直接放入主程序中。

6.2.7 停机问题

[1]步进电机无法通过按键 A 正常停止转动，之后也无法通过按键 A 正常启动。

[2]DE 功能下，电机在走完指定步数后仍不予停止。

【解决方法】设置停机标志 bFirst 判断电机的运行状态，将对 StepDec 的判断移入中断子程序 TIMERO 中。

6.2.8 转动方向问题

以单 4 拍为例，按照原理理解，正转应为 A→B→C→D→A，但在实际运行过程中，正转却为 C→B→A→D→C。

【解决方法】以实际硬件操作为主。

6.2.9 显示问题

起初，B 端口既是列线又是位码输出，在进行列扫描时会向列端口输出全“0”测试有无按键按下，此时导致 8 个数码管都被选中显示全 0。

【解决方法】在 SCANKEY 扫描按键子程序中调用 LED_DISPLAY 数码管显示子程序。

此外，为了使多个数码管同时显示，必须进行一定时间的延时，否则将会出现数码管从左到右依次显示的问题。

6.2.10 键入 A 问题

在调试时我们发现，有时按 A 电机没有反应，需要再按一两次电机才会启停。

【原因分析】可能存在按键抖动，可适当加入一定的延时，但仍存在按键抖动现象。而且如果加入的延时不恰当，会出现数码管屏闪且显示全“8”的问题。

6.2.11 当前步数不同步问题

无论转速如何变化，当前步数的改变频率总是不变，与实际步数的改变不同步。

【解决方法】在中断子程序 TIMERO 内进行当前步数的更新。

七、设计总结

7.1 系统已经实现的功能

1）能够实现题目中扫描按键 A~F 并执行对应指令的功能；

2）能够使步进电机在跨度较大的速度范围下运行；

3）能够在步进电机启动状态下，若按下除启停键“A”和停机键“F”以外的其他按键无效，对步进电机的状态和功能不影响；

4）能够通过开关量输入设置步进电机的节拍方式，且步进电机的 3 种节拍方式下的转动都能实现；

5）能够实现两个 8086 系统之间的通信，从而控制两个步进电机的转动；

6）能够实现在步进电机运行状态下步数达到最大值后会反向转动；

7.2 有待改进的地方

1）可以使步进电机实现变速转动

目前系统只能够在 B,C 模式下按照设定的速度匀速运行，而不能够实现变速功能。

2）步数误差问题

目前系统在 D,E 模式下设置一定的步数让电机停下来，但实际显示的步数要

大一些，有一定的误差。

3）按键抖动问题

目前系统在按键“A”的操作上有仍有按键抖动的问题，有时不能正确地对电机启停。

4）

7.3 改进的措施

1）实现变速转动

一种是若设置在电机启动状态下按除“A”“F”外的按键有效，则可在 B,C 模式下修改速度，使得步进电机变速运行，但这种方式对步进电机的影响较大，若设置的速度相差较大则电机有可能会卡住不转。

另一种可以通过 0809 对 0~5V 电压进行模数转换，然后将其持续循环的输入给 8253 模块，修改其频率达到变速的效果。这种方式下速度不会突然跳变，对电机的影响较小。

7.4 创新方法

7.4.1 电机反转

我们设置若当前步数的绝对值已到达 9999 步计数上限时，改变电机的转动方向，同时修改显示的功能键值。

7.4.2 开关量设置转动方式

通过拨动 K5、K6 可设置电机的转动方式，是单四拍、双四拍还是单双八拍。

7.4.3 通过 8251 通信

利用 8251 可以实现步进电机之间的控制：发送机将自身 buffer 缓冲区中的数据送给接收机，接收机成功接收后保存在自身的 buffer 缓冲区中，此后根据 buffer 中的数据设定功能和功能参数，按 A 进行转动。

7.4.4 0809 控制、蜂鸣器提示

在实现了 8251 通信的前提下，将发送机设置为主机，将接收机设置为从机。主机设定不变，从机修改为：

1）无法通过按键设定功能和参数，只能通过主机发送指定的功能和参数控制从机步进电机进行转动；

2）通过 ADC0809 对旋转黑色旋钮产生的电压进行模数转换，当电压为 5V 时，控制从机电机停止，从而实现用旋钮控制从机电机启停的功能：

[1]只有在从机电机停止的情况下，才能通过 8251 通信，接收主机发来的数据；

[2]只有在主机启动状态下，按下任意键，将数据传送给从机。

3）此时因为不需要按键控制，8255 的 C 口低 4 位全部未用，此时可将 C 口低 4 位设置成方式 0 输出。将 PC0 连接到蜂鸣器的 Ctrl 端口，实现功能如下：

[1]当从机电机工作在 DE 模式下，走完设定步数时，通过 PC0 发信号给 Ctrl，提示步数已走完；

[2]旋钮控制从机停止后，通过蜂鸣器进行提示。

八、课程设计的收获及心得体会

8.1 收获

1）第一次了解到步进电机。也在课程设计的过程中对它有了一定的理解和掌握。

2）在和同组的小伙伴一起设计子程序，编写主程序以及软硬件调试的过程中也再一次感受到了小组合作的好处和不易。

3）在基本设计内容中对 8255，8253，8259 和 8251 等芯片有了更加深入的理解，也逐渐体会到了它们的一些应用范围。

4) 软硬件调试过程中遇到了很多问题，通过对它们的分析和解决，增强了分析问题、解决问题的能力。

5）在创新过程中了解了蜂鸣器，0809 等器件的功能，在设计创新时，得到了更多独立思考的经验。

8.2 心得体会

通过本次接口课程设计，实现了步进电机的控制。总的来说结果是好的，我们成功地实现了实验的基本要求，但是过程真的是非常艰辛的。

由于硬件设备只能在实验室用，所以在宿舍只能够一直修改程序，无法测试，无法立即发现问题，而每次在实验室要是有一个问题也是需要修改这个问题修改很久的，所以很难合理地使用到实验箱。

我们几乎除了我们班的实验时间也会去机房找空的设备修改我们的程序，但相对来说可能在逻辑思考上的能力还是不够，常常在修改完程序之后还是会有新的错误出现，这可能是程序员的宿命？一直都在改 BUG，但是功夫不负有心人，我们最终修改的程序算是实现的功能很完善了。

除此之外，在思考创新点时，我们有想过很多想法，但是有些由于硬件设备端口比较少的问题，没能实现我们的想法，还有一些是我们认为可以实现但我们在之前也已经耗费了很长的时间，所以没有充足的时间来进行思考设计，这算是比较遗憾的地方。

要用一个字概括的话，那肯定是“累”，但也算是“累并快乐着”，也是不错的哈哈，为自己加油！

九、附录程序清单

9.1 send.asm(发送方完整程序)

```
.MODEL TINY
PORT8255_A    EQU 270H              ;CS1
PORT8255_B    EQU 271H
PORT8255_C    EQU 272H
PORT8255_K    EQU 273H
PORT8253_0    EQU 260H              ;CS2
PORT8253_1    EQU 261H
PORT8253_2    EQU 262H
PORT8253_K    EQU 263H
PORT8259_0    EQU 250H              ;CS3
PORT8259_1    EQU 251H
PORT8251_0    EQU 240H
PORT8251_1    EQU 241H              ;CS4
.STACK 100
.DATA
    BITCODE          DB        ?                              ;存放位码值
    SEGTAB           DB        0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,
88H,83H,0C6H,0A1H,86H,8EH,40H,3FH          ;段码->LED 灯编码（0~F+"0."+"-."）
    KEYVALUE         DB
00H,01H,02H,03H,04H,05H,06H,07H,08H,09H,0AH,0BH,0CH,0DH,0EH,0FH
  ;键值表 +【10H -> "0."】+【11H -> "-."】
    KEYCODE          DW
0FE02H,0FD02H,0FB02H,0F702H,0EF02H,0DF02H,0BF02H,7F02H,0FE01H,0FD01H,0FB01
H,0F701H,0EF01H,0DF01H,0BF01H,7F01H    ;键盘列行码
    BUFFER           DB        8 DUP(?)                       ;数码管上显示内容缓存区
    bFirst           DB        0                              ;启停步进电机（0：启动；1：
停止）
    bClockwise       DB        0                              ;电机转动方向（1：顺时针；
0：逆时针）
    bNeedDisplay     DB        0                              ;已转动一步，需要显示新的步
数
    StepControl      DB        0                              ;给步进电机下一步的相位值
    SetCount         DB        0                              ;步进电机的步数/转速
```

```
    StepCount       DW      0                       ;数码管显示的步数值
    StepDec         DW      0                       ;D,E 设置的步数
    SpeedNumber     DB      0                       ;B,C 设置的转速
    JiShu           DW      5000                    ;8253 计数初值 5000(0.005 秒/
步)
    StyleMove       DB      0                       ;方式
    ControlStyle    DB      11H,33H,10H,30H,20H,60H,40H,0C0H,80H,90H        ;拍
值
    DONE            DB      0                       ;设置的步数已走完及中断完成
标志位
    BUFFERsend      DB      01H,02H,03H,04H,10H,09H,09H,0BH         ;发送数据缓
冲区，初始为【B990.4321】
.CODE
START:
    MOV AX,@DATA
    MOV DS,AX               ;装载 DS
    ;----------初始化部分
    CALL INIT8253           ;8253 初始化
    CALL INIT8255           ;8255 初始化
    CALL INIT8259           ;8259 初始化
    CALL INIT8251           ;8251 初始化
    CALL INTERRUPT_VECTOR       ;中断向量表初始化
    CALL INITMOTOR          ;步进电机模块初始化
    CALL SEND8251           ;向另一个实验箱的 8251 发送数据
    ;----------等待设置步进电机节拍方式
WaitingSet:
    CALL SetStyleMove       ;调用子程序对设置的节拍方式进行读取判断
    CMP StyleMove,3         ;判断是否设置了节拍方式
    JZ WaitingSet           ;还未设置节拍方式继续等待
    ;----------开始显示并扫描按键
START_1:
    CALL LED_DISPLAY        ;调用数码管显示子程序
WAITEKEY:
    CALL SCANKEY            ;调用扫描键盘子程序
    CMP AL,-1               ;判断是否有按键按下
    JNZ START_2             ;有按键按下，则跳转开始判断按键
    CMP bNeedDisplay,0      ;无按键，则判断电机有没有转动
```

```
    JZ    WAITEKEY                ;无按键，电机没有转动，则继续等待按键
    MOV bNeedDisplay,0            ;清零为下一次转动做准备
    CMP bFirst,0                  ;电机转动了一步，且此时没有按键按下，判断电机此
时的状态
    JZ    Exec1                   ;电机处于启动状态-->则开中断启动
    JMP StopNEXT                  ;电机处于停机状态-->则关中断停止
    ;-------------判断按键+设置参数
START_2:                          ;开始判断按键
    CMP AL,0AH                    ;如果输入 A
    JNZ START_OTHER               ;若果不是 A 跳转到执行
    XOR bFirst,1                  ;启停步进电机
    CMP bFirst,0                  ;判断是启动还是停止步进电机
    JNZ StopNEXT                  ;bFirst=1，跳转到 StopNEXT
StartNEXT:                        ;bFirst=0，启动步进电机
    CALL Getbuffer65              ;设置 SetCount
    CMP buffer+7,0BH              ;如果是功能 B
    JZ BC                         ;跳转到 BC，设置步速
    CMP buffer+7,0CH              ;如果是功能 C
    JZ BC                         ;跳转到 BC，设置步速
DE:                               ;功能 DE，设置步数
    MOV AL,SetCount               ;AL <-- SetCount
    MOV BL,10                     ;BL <-- 10
    MUL BL                        ;设置步数为 10*SetCount
    MOV StepDec,AX                ;再将步数送给 StepDec(即输入 99，设置步数为 990
步)
    CMP StepDec,0                 ;若初始设置为 0
    JZ    START_1                 ;跳转到开始继续等待按键
    JMP Exec1                     ;否则开中断
BC:                               ;功能 BC，设置步速
    MOV StepDec,-1                ;清除步数
    CALL SetJiShu                 ;调用子程序设置 8253 计数值
    CALL INIT8253                 ;设置新的速度
Exec1:                            ;开始启动步进电机
    MOV DONE,0                    ;D,E 功能下，中断结束标志位清零
    STI                           ;开中断
    CMP DONE,1                    ;判断中断是否结束
    JNZ START_1                   ;没有结束跳转到开始继续扫描按键等待
```

```
StopNEXT:                    ;bFirst=1，停止步进电机
    CLI                      ;关中断
    MOV bFirst,1             ;再赋给 bFirst 一次值
    JMP START_1              ;跳转到开始继续扫描按键
START_OTHER:                 ;如果输入 0~9 或 B~F
    CMP AL,0FH               ;如果输入 F
    JZ F                     ;跳转到 F 使步进电机停机
    CMP bFirst,1             ;判断电机当前状态
    JZ SHURU                 ;电机处于停机状态，则跳转设置 0~9，B~E
    JMP Exec1                ;电机处于启动状态，则除 A,F 按键外其他按键不能进
行操作
SHURU:
    CMP AL,9                 ;如果输入 0~9
    JA START_BCDE            ;将键值送入 buffer+6 和 buffer+5，即步数/步速
    MOV AH,buffer+5          ;根据输入的键值从右到左进行步速/步数更新
    MOV buffer+6,AH
    MOV buffer+5,AL
    CALL DELAY1              ;调用延时子程序：防止按键输入过快
    JMP START_1              ;否则跳转显示刚才设置的数据或命令,然后继续等待
按键
START_BCDE:                  ;如果输入 B,C,D,E
    MOV buffer+7,AL          ;将 B,C,D,E 的功能键值送入 buffer+7
    CMP AL,0BH               ;如果输入 B
    JZ BD                    ;跳转到 BD，设置转动方向（顺时针）
    CMP AL,0DH               ;如果输入 D
    JZ BD                    ;跳转到 BD，设置转动方向（顺时针）
    MOV bClockwise,0         ;否则为功能 CE，设置转动方向为逆时针
    JMP START_1
BD:                          ;功能 BD，设置为顺时针
    MOV bClockwise,1
    JMP START_1              ;跳转到开始继续扫描按键等待
F:                           ;F--->程序结束
    CLI                      ;关中断
    CALL HltFunction         ;调用停机显示‘F’子程序
    HLT                      ;停机
;---------------------------------------------------------------
;子程序名：TIMERO
```

```
;功能：中断子程序
;影响变量：StepDec,StepControl,bNeedDisplay
;影响寄存器：AX,DX
TIMERO PROC NEAR
    PUSH AX
    PUSH DX
    MOV AL,StepControl          ;将下一次要送给步进电机的值送给 AL
    MOV DX,PORT8255_C           ;将 StepControl 通过 8255C 口输出
    OUT DX,AL
    CALL StepControlSet         ;设置下一步相位值
    MOV bNeedDisplay,1          ;设置需要显示新步数
    CALL AlterStep              ;调用步数调整子程序对显示屏显示的步数进行调整
    CMP StepDec,-1              ;判断是否设置步数
    JZ TIMERO_1                 ;没有设置步数即 B,C 模式下，直接发结束中断 EOI
    DEC StepDec                 ;设置步数，中断一次步数-1
    CMP StepDec,0               ;判断 D,E 功能下设置的步数 StepDec 有没有走完
    JNZ TIMERO_1                ;如果走完了需要关中断停止，并使 bFirst 为 1
    MOV DONE,1                  ;中断完成，标志位置'1'
    CLI                         ;关中断
    MOV bFIRST,1                ;电机-->停机状态
TIMERO_1:
    MOV DX,PORT8259_0           ; 发结束中断 EOI
    MOV AL,20H
    OUT DX,AL
    POP DX
    POP AX
    IRET
TIMERO ENDP
;---------------------------------------------------------------
;子程序名：SetJiShu
;功能：设置 8253 计数值得到不同的频率
;入口参数：SetCount
;出口参数：JuShu，SpeedNumber
;影响寄存器：AX,BX
SetJiShu PROC NEAR
    PUSH AX                     ;保护现场
    PUSH BX
```

```
    INC SetCount                ;输入 0 对应 1 级速度，以此类推
    MOV AL,SetCount             ;保存速度级数到 SpeedNumber
    MOV SpeedNumber,AL
    MOV AL,101                  ;根据速度级数通过 5000+100*(101-级数)计算 8253
计数值
    SUB AL,SetCount
    MOV BL,100
    MUL BL
    ADD AX,5000
    MOV JiShu,AX                ;将计数值传给 JiShu
    POP BX                      ;恢复现场
    POP AX
    RET
SetJiShu ENDP
;---------------------------------------------------------------
;子程序名：INITMOTOR
;功能：步进电机模块初始化+BUFFER 初始化
INITMOTOR PROC NEAR
    MOV bFirst,1                ;初始步进电机初始停止状态
    MOV BL,StyleMove            ;将偏移量送入 StyleMove
    MOV AL,[ControlStyle+BX]
    MOV StepControl,AL          ;初始下一次送给步进电机的值
    MOV buffer,0                ;初始化显示 D990.0000
    MOV buffer+1,0
    MOV buffer+2,0
    MOV buffer+3,0              ;初始显示步数为 0.0000
    MOV buffer+4,10H
    MOV buffer+5,9              ;初始步数为 990
    MOV buffer+6,9
    MOV buffer+7,0DH            ;初始为 D##
    CMP buffer+7,0BH            ;根据 buffer+7 的命令初始化电机转动方向
    JZ BDinit
    CMP buffer+7,0DH
    JZ BDinit
    MOV bClockwise,0
BDinit:
    MOV bClockwise,1
```

```
    RET
INITMOTOR ENDP
;-------------------------------------------------------------
;子程序名：AlterStep
;功能：步进电机的步数调整
;入口参数：步数（buffer 低 4 位）数组首地址 BX，步数位数 CX
;出口参数：buffer 低 4 位
;影响寄存器：BX,CX
;问题修改：处理特殊数字跳转及设置 9999 反转
AlterStep PROC
    PUSH CX                         ;保护现场
    PUSH BX
    CALL StepCountSet               ;获取显示屏当前步数
    MOV CX,4                        ;设置步数位数
    LEA BX,buffer                   ;数组首地址
    CMP bClockwise,1                ;判断正/反转
    JZ ClockwiseSet
AntiClockwiseSet:                   ;反转设置
    CMP BYTE PTR[BX+4],11H          ;判断步数是正是负
    JZ AddStep                      ;步数为负，+1
    CMP StepCount,0                 ;步数为 0.0000
    JNZ AntiClockwiseSet1
    MOV BYTE PTR[BX+4],11H
    INC BYTE PTR[BX]                ;则步数变为-.0001
    JMP AlterStep1
AntiClockwiseSet1:
    JMP SubStep                     ;步数为正，-1
ClockwiseSet:                       ;正转设置
    CMP BYTE PTR[BX+4],10H          ;判断步数是正是负
    JZ AddStep                      ;步数为正，+1
    CMP StepCount,1                 ;步数是否为-.0001
    JNZ ClockwiseSet1
    MOV BYTE PTR[BX+4],10H
    DEC BYTE PTR[BX]                ;步数变为 0.0000
    JMP AlterStep1
ClockwiseSet1:
    JMP SubStep                     ;步数为负，-1
```

```
AddStep:
    INC BYTE PTR [BX]                ;步数+1
    CMP BYTE PTR [BX],0AH            ;低位是否产生进位
    JNZ AlterStep1                   ;无进位，则退出
    MOV BYTE PTR [BX],0              ;有进位，处理进位
    INC BX
    LOOP AddStep
    SUB BX,4
    MOV CX,4                         ;四位都有进位则达到最大值
AddStep1:                            ;设置最大值 0.9999
    MOV BYTE PTR [BX],9
    INC BX
    LOOP AddStep1
    XOR bClockwise,1                 ;步数>9999,使电机反转
    CMP BUFFER+7,0BH                 ;同时修改转动方式
    JZ BD_fanzhuan
    CMP BUFFER+7,0DH
    JZ BD_fanzhuan
    DEC BUFFER+7
    JMP AlterStep1
BD_fanzhuan:
    INC BUFFER+7
    JMP AlterStep1
SubStep:
    DEC BYTE PTR [BX]                ;步数-1
    CMP BYTE PTR [BX],0FFH           ;低位是否产生借位
    JNZ AlterStep1                   ;无借位，则退出
    MOV BYTE PTR [BX],9              ;有借位，处理借位
    INC BX
    LOOP SubStep
AlterStep1:
    POP BX                           ;恢复现场
    POP CX
    RET
AlterStep ENDP
;---------------------------------------------------------------
;子程序名：StepControlSet
```

```
;功能：判断电机转动方式，并送下一次送入 StepControl 的值
;入口参数：ControlStyle 转动方式数组，以及偏移量 DI（StyleMove）
;出口参数：StepControl
;影响寄存器：BX,AX
;问题修改：顺时针-->右移,逆时针-->左移
StepControlSet PROC NEAR
    PUSH BX
    PUSH AX
    XOR BX,BX
    MOV BL,StyleMove            ;将偏移量送入 StyleMove
    MOV AL,[ControlStyle+BX]    ;将转动方式的相位值送给 StepControl
    MOV StepControl,AL
    CMP BX,2                    ;是否是单双 8 拍的转动方式
    JNB DanShuang8              ;如果是单双 8 拍，则跳转到 DanShuang8
    CMP bClockwise,1            ;否则，判断电机转动方向
    JZ  ControlClockwise        ;如果是正转，跳转到 ControlClockwise
ControlAntiClockwise:           ;如果是反转，顺势左移 1 位
    ROL [ControlStyle+BX],1
    JMP SCNEXT
ControlClockwise:               ;如果是正转，顺势右移 1 位
    ROR [ControlStyle+BX],1
    JMP SCNEXT
DanShuang8:                     ;单双 8 拍的转动方式
    CMP bClockwise,1            ;判断电机转动方向
    JZ DSClockwise              ;如果是正转，跳转到 DSClockwise
DSAntiClockwise:                ;如果是反转，选择数组下一个值送入 StepControl
    INC BX
    CMP BX,10                   ;相位值循环(偏移量 DI 增加)
    JNZ SCNEXT
    MOV BX,9
    JNZ SCNEXT
DSClockwise:                    ;如果是正转，选择数组上一个值送入 StepControl
    DEC BX
    CMP BX,1                    ;相位值循环(偏移量 DI 减少)
    JNZ SCNEXT
    MOV BX,2
    JNZ SCNEXT
```

```
SCNEXT:
    MOV StyleMove,BL            ;保存此时的偏移量值进 StyleMove
    MOV AL,[ControlStyle+BX]
    MOV StepControl,AL          ;将下一步相位值给到 StepControl
    POP AX
    POP BX
    RET
StepControlSet ENDP
;-------------------------------------------------------------------
;子程序名：StepCountSet
;功能：把 buffer0-3（当前步数值）送入 StepCount
;入口参数：buffer0-3
;出口参数：StepCount
;影响寄存器：AX,BX
StepCountSet PROC NEAR
    PUSH AX
    PUSH BX                         ;保护现场
    MOV AL,buffer+3                 ;（buffer+3）-->AL
    MOV BX,10
    MUL BL
    ADD AL,buffer+2                 ;AL*10+（buffer+2）-->AL
    MUL BL
    ADD AL,buffer+1                 ;AL*10+（buffer+1）-->AL
    ADC AH,0
    MUL BX
    ADD AL,buffer                   ;AL*10+（buffer）-->AL
    ADC AH,0
    MOV StepCount,AX                ;转动步数送入 StepCount
    POP BX                          ;恢复现场
    POP AX
    RET
StepCountSet ENDP
;---------------------------------------------------------------
;子程序名：[INIT8255]
;功能：8255 初始化
INIT8255 PROC NEAR
    MOV DX,PORT8255_K
```

```
    MOV AL,81H                  ;A 方式 0 输出，B 方式 0 输出，C 低 4 位输入，高 4 位
输出
    OUT DX,AL
    MOV   DX, PORT8255_B
    MOV AL,0FFH                 ;位码全 1，初始数码管全部熄灭
    OUT DX,AL
    RET
INIT8255 ENDP
;---------------------------------------------------------------
;子程序名：INIT8253
;功能：8253 初始化
INIT8253 PROC NEAR
    MOV DX,PORT8253_K
    MOV AL,34H                  ;00110100 计数器 0，方式 2，二进制计数
    OUT DX,AL
    MOV DX,PORT8253_0
    MOV AX,JiShu                ;10000+50*(101-n)
    OUT DX,AL
    MOV AL,AH
    OUT DX,AL
    RET
INIT8253 ENDP
;---------------------------------------------------------------
;子程序名：[INIT8259]
;功能：8259 初始化
INIT8259 PROC NEAR
    MOV DX,PORT8259_0
    MOV AL,13H                  ;[ICW1]00010011,边沿触发，单片方式，使用 ICW4
    OUT DX,AL
    MOV DX,PORT8259_1
    MOV AL,08H                  ;[ICW2]00001000,中断类型号从 08H 开始
    OUT DX,AL
    MOV AL,09H                  ;[ICW4]00001001,一般全嵌套，缓冲方式，非自动结束中
断
    OUT DX,AL
    MOV AL,0FEH                 ;[OCW1]11111110,开放 IR0 中断请求
    OUT DX,AL
```

```
    RET
INIT8259 ENDP
;---------------------------------------------------------------
;子程序名：[INIT8251]
;功能：8251 初始化
INIT8251 PROC NEAR
    ;--------软复位----------
    MOV CX,3                    ;
    XOR AL,AL                   ;
    MOV DX,PORT8251_1           ;
AGA:OUT DX,AL                   ;连续写入 3 个 00H
    CALL DELAY0                 ;每次写入后延时一定的时间
    LOOP AGA                    ;
    MOV AL,40H                  ;
    OUT DX,AL                   ;写入 40H
    CALL DELAY0
    ;------8251 初始化--------
    MOV AL,4EH                  ;01001110B 设置方式字【异步×16，数据位 8 位，不带
奇偶校验位，1 位停止位】
    OUT DX,AL                   ;
    CALL DELAY0
    MOV AL,21H                  ;00100001B 设置控制字【允许发送，请求发送】
    OUT DX,AL                   ;
    RET
INIT8251 ENDP
;---------------------------------------------------------------
;子程序名：[INTERRUPT_VECTOR]
;功能：中断向量表初始化
;入口参数：子程序 TIMERO
;影响寄存器：ES,AX,BX
INTERRUPT_VECTOR PROC NEAR
    PUSH ES
    PUSH AX
    PUSH BX
    MOV AX,0
    MOV ES,AX
    MOV BX,08H*4                ;BX<-中断向量地址
```

```
    MOV AX,OFFSET TIMERO            ;中断子程序地址
    MOV ES:[BX],AX              ;存放偏移地址
    MOV AX,SEG TIMERO
    MOV ES:[BX+2],AX            ;存放段地址
    POP BX
    POP AX
    POP ES
    RET
INTERRUPT_VECTOR ENDP
;---------------------------------------------------------------
;子程序名：DELAY0
;功能：延时子程序：延时使多位数码管同时显示
;影响寄存器：CX
DELAY0 PROC NEAR
    PUSH CX
    MOV CX,500
        LOOP $
    POP CX
    RET
DELAY0 ENDP
;---------------------------------------------------------------
;子程序名：DELAY1
;功能：延时子程序：延时用于防止输入数字时过快
;影响寄存器：CX
DELAY1 PROC NEAR
    PUSH CX
    MOV CX,25000
    LOOP $
    POP CX
    RET
DELAY1 ENDP
;---------------------------------------------------------------
;子程序名：DELAY2
;功能：延时子程序：用于扫描按键时按键消抖
;影响寄存器：CX
DELAY2 PROC NEAR
    PUSH CX
```

```
    MOV CX,10
    LOOP $
    POP CX
    RET
DELAY2 ENDP
;---------------------------------------------------------------
;子程序名：Getbuffer65
;功能：设置步进电机的步数/速度
;入口参数：十位 buffer+6，个位 buffer+5
;出口参数：步数/速度值 SetCount
;影响寄存器：AX,BX
Getbuffer65 PROC
    PUSH AX
    PUSH BX                          ;保护现场
    MOV AL,buffer+6                  ;转动步数送入 SetCount
    MOV BX,10
    MUL BL
    ADD AL,buffer+5
    MOV SetCount,AL
    POP BX                           ;恢复现场
    POP AX
    RET
Getbuffer65 ENDP
;---------------------------------------------------------------
;子程序名：[HltFunction]
;功能：数码管显示全 F 子程序
;影响寄存器：AX,DX
;调用子程序：DELAY0
HltFunction PROC NEAR
    PUSH AX
    PUSH DX                      ;保护现场
    MOV   AL, 0FFH               ;位码=0FFH
    MOV   DX, PORT8255_B
    OUT   DX, AL                 ;熄灭所有数码管
    MOV   AL,8EH                 ;F 的段码
    MOV   DX, PORT8255_A
    OUT   DX, AL                 ;送出段码
```

```
    MOV   AL,0                    ;所有位都显示
    MOV   DX, PORT8255_B
    OUT   DX, AL                  ;送出位码
    CALL DELAY0                   ;进行适当的延时
    POP DX                        ;恢复现场
    POP AX
    RET
HltFunction ENDP
;---------------------------------------------------------------
;子程序名：[LED_DISPLAY]
;功能：数码管显示子程序
;入口参数：  BITCODE=位码值
;            SEGTAB=段码值
;            BUFFER=需显示数据缓存区
;影响寄存器：AX,CX,DX,SI
;调用子程序：DELAY0
LED_DISPLAY PROC NEAR
    PUSH AX
    PUSH CX
    PUSH DX
    PUSH SI                       ;保护现场
    LEA   BX,SEGTAB               ;BX 为段码表首址
    MOV BITCODE,0FEH              ;
    MOV   SI, 0                   ;SI 用作缓冲区指针，初值为 0
    MOV   CX, 8                   ;CX 用作循环计数器，初值为 8
ONE:
    MOV   AL, 0FFH                ;位码=0FFH
    MOV   DX, PORT8255_B
    OUT   DX, AL                  ;熄灭所有数码管
    MOV   AL, BUFFER[SI]          ;从缓存区得到待查元素在表中的序号
    XLAT                          ;查表转换
    MOV   DX, PORT8255_A
    OUT   DX, AL                  ;送出一位信息的字形码（段码）
    MOV   AL,BITCODE              ;位码
    MOV   DX, PORT8255_B
    OUT   DX, AL                  ;送出位码，选中某位数码管显示段码
    ROL   BITCODE, 1              ;循环左移有效“0”--->形成下一个位码
```

```
    INC     SI                          ;修改输出缓冲区指针
    CALL DELAY0                         ;进行适当的延时使多位数码管同时显示
    LOOP    ONE                         ;循环，显示下一位信息
    POP SI                              ;恢复现场
    POP DX
    POP CX
    POP AX
    RET
LED_DISPLAY ENDP
;-------------------------------------------------------------------
;子程序名：【SCANKEY】
;功能：列扫描按键得到键值
;出口参数：AL=键值
;影响寄存器：CX,DX
;调用子程序：TRANSLATE    \      DELAY2
SCANKEY PROC NEAR
    PUSH CX
    PUSH DX                 ;保护寄存器
    MOV DX,PORT8255_B;B【列线】
    MOV AL,0
    OUT DX,AL           ;列输出全 0
    MOV DX,PORT8255_C;C【行线】
    IN AL,DX            ;读取 C 端口（行端口）
    AND AL,3
    CMP AL,03H          ;检测行信息(2 行->03H)是否全为 1，判断有无按键
    JZ NO_KEY           ;无按键时，转移后返回-1
    CALL DELAY2             ;调用延时子程序延时消抖
    IN AL,DX            ;读取行端口
    AND AL,03H
    CMP AL,03H          ;检测行信息是否全为 1，判断有无按键
    JZ NO_KEY           ;无按键时，转移后返回-1
    MOV AH,0FEH             ;指定列码，从 PB0 列开始扫描
    MOV CX,8            ;最多扫描 8 列
NEXT:MOV AL,AH              ;列码->AL
    ROL AH,1            ;形成下一列的列码，为扫描下一列做准备
    MOV DX,PORT8255_B
    OUT DX,AL           ;输出列码
```

```
    MOV DX,PORT8255_C
    IN AL,DX              ;读取行码
    AND AL,03H
    CMP AL,03H            ;ZF=0 或 CX=0 则退出循环
    LOOPZ NEXT
    JZ NO_KEY             ;ZF=1,无按键，转移后返回-1
    ROR AH,1              ;存放形成的列行码->AX（恢复刚才的列码）
    CALL TRANSLATE        ;列行码转换为键值->AL
    JMP EXIT
NO_KEY:
    MOV AL,-1             ;没有按键 AL=-1
EXIT:
    CALL LED_DISPLAY      ;显示 LED（防止显示乱码）
    POP DX
    POP CX
    RET
SCANKEY ENDP
;---------------------------------------------------------------
;子程序名：【TRANSLATE】
;功能：列行码转换为键值
;入口参数：AX=扫描到的按键的列行码
;          KEYVALUE=键值表
;          KEYCODE=列行码表
;影响寄存器：CX,SI,DI
TRANSLATE PROC NEAR
    PUSH CX
    PUSH SI
    PUSH DI               ;保护寄存器
    MOV CX,16             ;16 个按键，设置循环次数
    LEA SI,KEYVALUE-1     ;初始化键值表地址
    LEA DI,KEYCODE-2      ;初始化列行码表地址
SCANTAB:
    INC SI                ;用 SCANKEY 子程序返回的列行码查表
    ADD DI,2              ;修改指针
    CMP AX,[DI]           ;比较查找按键
    LOOPNZ SCANTAB        ;ZF=1 或 CX=0 则退出循环
    JNZ QUIT              ;ZF=0 则结束
```

```
    MOV AL,[SI]             ;得到键值
QUIT:
    POP DI                  ;恢复寄存器
    POP SI
    POP CX
    RET
TRANSLATE ENDP
;---------------------------------------------------------------
;子程序名：【SetStyleMove】
;功能：设置电机转动方式
;入口参数：开关量
;出口参数：StyleMove=电机转动方式
;影响寄存器：CX,SI,DI
SetStyleMove PROC
    PUSH AX
    PUSH DX
    MOV DX,PORT8255_C
    IN AL,DX                ;读取 8255C 端口
    SHR AL,1
    SHR AL,1                ;逻辑右移两位
    AND AL,3                ;使 PC3PC2 有效即得到输入的开关量
    MOV StyleMove,AL        ;开关量存入表示电机转动方式的变量 StyleMove
    POP DX
    POP AX
    RET
SetStyleMove ENDP
;---------------------------------------------------------------
;子程序名：SEND8251
;功能：发送 BUFFERsend 初始数据
;影响寄存器：DI,AX,DX,
SEND8251 PROC
    PUSH DI
    PUSH AX
    PUSH DX
    LEA DI,BUFFERsend           ;设置发送数据快地址指针
    MOV CX,8                    ;设置计数器初值
    ;------发送数据----------
```

```
SEND:
    MOV DX,PORT8251_1          ;
    IN AL,DX                   ;
    AND AL,01H                 ;查询 TxRDY 有效否?
    JZ SEND                    ;TxRDY=0，无效则等待
    MOV DX,PORT8251_0          ;
    MOV AL,[DI]                ;向 8251 输出 1 个字节数据
    OUT DX,AL                  ;
    INC DI                     ;修改地址指针
    LOOP SEND                  ;循环
    POP DX
    POP AX
    POP DI
    RET
SEND8251 ENDP
    END START
```

9.2 receive.asm（接收方程序，这里只贴出与发送方不同的地方）

【1】在主程序里修改为 CALL RECEIVE8251

【2】INITMOTOR 需要注释掉对 buffer 数据缓冲区的赋值，通过通信获取数据。

【3】INIT8251

```
;-------------------------------------------------------------
;子程序名：[INIT8251]
;功能：8251 初始化
INIT8251 PROC NEAR
    ;--------软复位----------
    MOV CX,3                   ;
    XOR AL,AL                  ;
    MOV DX,PORT8251_1          ;
AGA:OUT DX,AL                  ;连续写入 3 个 00H
    CALL DELAY0                ;每次写入后延时一定的时间
    LOOP AGA                   ;
    MOV AL,40H                 ;
    OUT DX,AL                  ;写入 40H
```

```
    CALL DELAY0
    ;------8251 初始化--------
    MOV AL,4EH                ;01001110B 设置方式字【异步×16，数据位 8 位，不带奇偶校验位，1 位停止位】
    OUT DX,AL                 ;
    CALL DELAY0
    MOV AL,16H                ;00010110H 设置控制字【清除错误标志，允许接收，数据终端准备好】
    OUT DX,AL                 ;
    RET
INIT8251 ENDP
```

【4】RECEIVE8251

```
;-------------------------------------------------------------
;子程序名：RECEIVE8251
;功能：接收初始数据子程序
;影响变量：BUFFER
;影响寄存器：BX,AX,DX
RECEIVE8251 PROC
    PUSH BX
    PUSH AX
    PUSH DX
    LEA BX,BUFFER             ;设置接收数据块地址指针
    MOV CX,8                  ;设置计数器初值
    ;------接收数据----------
RECEIVE:
    MOV DX,PORT8251_1         ;
    IN AL,DX                  ;
    TEST AL,02H               ;查询 RxRDY 有效否？
    JZ RECEIVE                ;RxRDY=0，无效则等待
    MOV DX,PORT8251_0         ;
    IN AL,DX                  ;RxRDY=1，读入数据
    MOV [BX],AL               ;保存数据
    INC BX                    ;修改地址指针
    LOOP RECEIVE              ;循环
    POP DX
    POP AX
    POP BX
```

```
    RET
RECEIVE8251 ENDP
    END START
```