ORG 0000H

LJMP START

ORG 0003H

LJMP X0\_INT

ORG 0013H

LJMP X1\_INT

ORG 0030H

START: ;初始化

SETB IT0 ;外部中断0 跳

SETB IT1 ;外部中断1 跳

SETB EX0 ;打开外部中断0

SETB EX1 ;打开外部中断1

SETB EA

MOV SP,#50H ;堆栈指针 初始位置50H，

MOV A,#0F0H ;F0H送给A

LOOP: MOV P1,A ;A传送给P1。P1输出值给led灯

CALL DELAY ;延时

CPL A ;取反

SJMP LOOP ;循环，等待中断

X0\_INT:

PUSH ACC ;保护数据

PUSH PSW

MOV A,#0FFH ;全灭

MOV R2, #4 ;亮灭共四次

X0\_1:

MOV P1,A ;P1输出

CALL DELAY ;延时

CPL A ;取反，全亮或全灭

DJNZ R2,X0\_1 ;判断R2是否为零并自减，跳转

POP PSW ;取出数据

POP ACC

RETI ;中断返回

X1\_INT:

PUSH ACC

PUSH PSW

SETB RS1

MOV A,#0FEH

MOV R2,#8 ;流水灯亮8次

X1\_1:

MOV P1,A

CALL DELAY

RL A

DJNZ R2,X1\_1

POP PSW

POP ACC

RETI

DELAY: ;@11.0592MHz

NOP

NOP

NOP

PUSH 30H ;保护数据

PUSH 31H

PUSH 32H

MOV 30H,#4

MOV 31H,#93 ;定时

MOV 32H,#152

NEXT:

DJNZ 32H,NEXT

DJNZ 31H,NEXT

DJNZ 30H,NEXT

POP 32H

POP 31H ;恢复数据

POP 30H

RET

END

1.

ORG 0000H

AJMP START

ORG 000BH ;定时器0的中断向量地址

AJMP TIME0 ;跳转到真正的定时器程序处

ORG 30H

START:

MOV 30H,#00H ;软件计数器预清0

MOV TMOD,#00000001B ;定时/计数器0工作于方式1

MOV TH0, #04bH

MOV TL0, #0fdH ;初始值

SETB EA ;开总中断允许

SETB ET0 ;开定时/计数器0允许

SETB TR0 ;定时/计数器0开始运行

MOV A,#00H;

LOOP: AJMP LOOP

TIME0: ;定时器0的中断处理程序

INC 30H

MOV R0,30H

CJNE R0,#20,T\_RET ;30H单元中的值是否为20

T\_L1:

MOV P1,A

CPL A;

MOV 30H,#0 ;清软件计数器

T\_RET:

MOV TH0,#04bH

MOV TL0,#0D7H ;重置定时常数

RETI

DELAY: ;@11.0592MHz

NOP

NOP

NOP

PUSH 30H ;保护数据

PUSH 31H

PUSH 32H

MOV 30H,#4

MOV 31H,#93 ;定时

MOV 32H,#152

NEXT:

DJNZ 32H,NEXT

DJNZ 31H,NEXT

DJNZ 30H,NEXT

POP 32H

POP 31H ;恢复数据

POP 30H

RET

END

2.

ORG 0000H

AJMP START

ORG 30H

START:

MOV SP,#50H

MOV TMOD,#01000000B ;定时/计数器1作计数用,0不用全置0

MOV TL1,#00H

SETB TR1 ;启动计数器1开始运行.

LOOP:

MOV A,TL1

CPL A

MOV P1,A

AJMP LOOP

END

miaoL equ 77h

miaoH equ 76h

timercount equ 40h

keyvalue equ 50h

org 0000h

ljmp main

org 000bh

ljmp timer0\_ser

org 0030h

main:

mov r0,#70h

mov a,#0

mov r7,#08h

lp:

mov @r0,a

inc r0

djnz r7,lp

lcall inittimer

lcall openint

lcall starttimer0

mov timercount,#0ah

main1:

lcall disp

lcall keycan

sjmp main1

disp:

push 07h

push 06h

push 00h

push acc

mov r7,#08h

mov r6,#0feh

mov r0,#70h

lp2:

mov a,r6

setb p2.7

mov p0,a

clr p2.7

mov dptr,#tab

mov a,@r0

movc a,@a+dptr

setb p2.6

mov p0,a

clr p2.6

lcall delay

inc r0

mov a,r6

rl a

mov r6,a

djnz r7,lp2

pop acc

pop 00h

pop 06h

pop 07h

ret

delay:

push 03h

push 04h

mov r3,#04h

d2: mov r4,#04fh

d1: djnz r4,d1

djnz r3,d2

pop 04h

pop 03h

ret

inittimer:

mov tmod,#01h

mov th0,#3ch

mov tl0,#0b0h

ret

openint:

setb et0

setb ea

ret

starttimer0:

setb tr0

ret

stoptimer:

clr tr0

ret

timer0\_ser:

push acc

mov th0,#3ch

mov tl0,#0b0h

djnz timercount,rtn

mov timercount,#0ah

inc miaoL

mov a,miaoL

cjne a,#0ah,rtn

mov miaoL,#0

inc miaoH

mov a,miaoH

cjne a,#06h,rtn

mov miaoH,#0

rtn:

pop acc

reti

keycan:

mov r3,#00h ;col

mov r4,#00h ;row

mov p3,#0fh

nop

nop

mov a,p3

cjne a,#0fh,k1

sjmp keyrtn

;puanduan hang

k1:

cpl a

anl a,#0fh

jb acc.0,row1

jb acc.1,row2

jb acc.2,row3

jb acc.3,row4

row1:

mov r4,#00h

sjmp next

row2:

mov r4,#01h

sjmp next

row3:

mov r4,#02h

sjmp next

row4:

mov r4,#03h

next:

mov p3,#0f0h

nop

nop

mov a,p3

cjne a,#0fh,k2

sjmp keyrtn

k2:

cpl a

anl a,#0f0h

swap a

jb acc.0,col1

jb acc.1,col2

jb acc.2,col3

jb acc.3,col4

col1:

mov r3,#00h

sjmp keyhandle

col2:

mov r3,#01h

sjmp keyhandle

col3:

mov r3,#02h

sjmp keyhandle

col4:

mov r3,#03h

keyhandle:

mov a,r4

mov b,#04h

mul ab

add a,r3

mov keyvalue,a

keywait: mov p3,#0fh

mov a,p3

cjne a,#0fh,keywait

lcall handlekey

keyrtn:

ret

handlekey:

mov a,keyvalue

cjne a,#00h,kk1

inc miaoL

mov a,miaoL

cjne a,#0ah,kkrtn

mov miaoL,#00h

sjmp kkrtn

kk1:

inc miaoH

mov a,miaoH

cjne a,#06h,kkrtn

mov miaoH,#00h

kkrtn:

ret

tab: db 3Fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh,77h,7ch

end

endchar equ '$'

org 0000h

ljmp main

org 0023h

ljmp serial\_ser

org 0030h

main:

mov p1,#0fh

mov tmod,#20h

mov th1,#0fdh

mov tl1,#0fdh

setb tr1

mov scon,#50h

mov dptr,#str

mov r6,#00h

mov a,r6

movc a,@a+dptr

mov sbuf,a

setb es

setb ea

sjmp $

serial\_ser:

jnb ri,send

clr ri

mov a,sbuf

cjne a,#31h,rets

mov p1,#0f0h

sjmp rets

send:

clr ti

inc r6

mov a,r6

movc a,@a+dptr

cjne a,#endchar,sendnext

sjmp rets

sendnext:

mov sbuf,a

rets:

reti

str: db 'hello world!','$'

end