PRODUCT NAME

TM58PR10

TITLE

TM58PR10 遥控器编码(UPG6121 SIMPLE 格式)

APPLICATION NOTE

1.功能简介
2.DEMO 程序
3.应用线路图

功能简介：

对于遥控器产品来说，如何利用端口的设置来进行正确的发码显得尤为重要。对于新推出的遥控器 MCU（TM58PR10）来说，以 REM 端口为载波发射端口，只需控制此端口，将发射的数据送入其中，便能完成你所需要的发码任务。

Note:

1. 在该 MCU 中，port B 及 port C 端口均有使用了下拉及唤醒功能。注意，port B 端口在任何情况下，都只能设置为输入，在 port C 端口，pc1 端口属于 I/O 口（作输入模式可外接 Led 显示）， pc0 口作输入及 OFF mode port，而 pc2 与 port B 端口具有同样的功能设置。
2. 与 port B 与 port C 端口的下拉及唤醒功能寄存器 PSTAT(地址为 24H)，其具体设置定义请参考以下：

Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PSTAT		PB Wake-up	PC Wake-up	PB Pull-down	PC1/ PC0 Pull-down	PC2 Pull-down	PC1 mode	X	PC0 mode
Setting	0	OFF	OFF	OFF	OFF	Disable	$\overline{LED}$	0	OFF
	1	ON	ON	ON	ON	Enable	PC1	0	IN
After cold reset		0	0	1	0	0	0	0	0

b0: 说明 PC_0 端口的输入模式。
0 = OFF (高阻抗); 1 = IN (输入模式).
b1: 保留;
b2: 说明 PC1/LED 端口的 I/O 模式。
0 = LED (输出模式); 1 = PC 1 (输入模式)
b3: 说明 PC2 下拉电阻的使用。
0 = 禁止 (无下拉); 1 = 允许 (有下拉).
b4: 说明在 PC1/ PC0 端口输入模式下拉电阻的使用。
0 = OFF (不使用); 1 = ON (使用).
b5: 说明 PB 端口下拉电阻的使用。
0 = OFF (不使用); 1 = ON (使用).
b6: 说明 PC 端口的唤醒功能。
0 = OFF; 1 = ON.
芯片可通过改变 PC 端口的状态被唤醒，PC 端口设成输入模式。
b7: 说明 PB 端口唤醒功能。
0 = OFF; 1 = ON.
芯片可以通过 PB 端口的状态被唤醒。

下面是以 TM58PR10 扫描 8*11 的 DEMO 程序及线路图，其发码的格式为 UPG6121 SIMPLE,程序的功能及设置请参考程序中的注释。

```
;--------------------------------------------------------------------------
;ic :tm58pr10_24pin
```

```
;titile : uPG612simple. asm
;振荡频率:4M
;key count: 88key
;date :2004-11-03
;--------------------------------------------------------------------------
;Note:
;code_head:                  hight:9.0ms    low:4.5ms
;client_code:   8bit
;reverse_code: 8bit
;              code_0        hight:560us    low:560us
;              code_1        hight:560us    low:1670us

;data_code:     8bit
;reverse_code: 8bit
;              code_0        hight:560us    low:560us
;              code_1        hight:560us    low:1670us

;stop_bit_code               hight:560us    low:37.0ms

;Last send  repeat code from the lead_code to the stop_bit_code.
;--------------------------------------------------------------------------

;-----------------------spicial resister--------------------------------1.
iar             equ.    0X00
timr0           equ     0x01
pc              equ     0x02
status          equ     0x03
bsr             equ     0x04
ra              equ     0x08
rb              equ     0x06
rc              equ     0x07

modl            equ     0x20
modh            equ     0x21
irq             equ     0x22
pgs             equ     0x23
pstat           equ     0x24

cg_ctrl         equ     0x25
timer           equ     0x26
osc_adj         equ     0x27

;-----------status resister bit------------2.
c               equ     0
dc              equ     1
z               equ     2
pd              equ     3
;-----------instruction flag---------------3
a               equ     0
```

```
m               equ     1

;-------------general resister-------------
ra_buf          equ     0x08
scan_lp         equ     0x09
send_lp         equ     0x0a
flag            equ     0x0b
key_value       equ     0x10
key_hang        equ     0x11
key_lie         equ     0x12
del_m0          equ     0x13
del_m1          equ     0x14
tm_r            equ     0x15
cont_t          equ     0x16
acc_buf         equ     0x17
status_buf      equ     0x18
check_r1        equ     0x19
check_r2        equ     0x1a

temp            equ     0x1b
public_l        equ     0x1c
public_h        equ     0x1d
temp_buf        equ     0x1e
val_cnt         equ     0x1f
;-----------------------------------------------------------------------
;Program start here.
;-----------------------------------------------------------------------
        org     3feh                    ;中断程序入口地址
        lgoto   tmr0_int                ;进入中断子程序
        org     3ffh
        movla   7fh                     ;程序在 ROM 中的起始地址
        org     000h
        movam   27h
        lgoto   main                    ;进入主程序

;----------------------------------键码表-----------------------------------------
seek_code
        addam           pc,     m

        retla           0x38            ;/sw01
        retla           0x28            ;/sw02
        retla           0xfa            ;/sw03
        retla           0xc2            ;/sw04
        retla           0xf2            ;/sw05
        retla           0xea            ;/sw06
        retla           0xd2            ;/sw07
        retla           0xe2            ;/sw08
```

```
retla       0xca        ;/sw09
retla       0xda        ;/sw10
retla       0x7a        ;/sw11
retla       0x42        ;/sw12
retla       0x72        ;/sw13
retla       0x6a        ;/sw14
retla       0x52        ;/sw15
retla       0x62        ;/sw16

retla       0x4a        ;/sw17
retla       0x5a        ;/sw18
retla       0xba        ;/sw19
retla       0x82        ;/sw20
retla       0xb2        ;/sw21
retla       0xaa        ;/sw22
retla       0x10        ;/sw23
retla       0x18        ;/sw24

retla       0x20        ;/sw25
retla       0x00        ;/sw26
retla       0x30        ;/sw27
retla       0x08        ;/sw28
retla       0xa0        ;/sw29
retla       0x80        ;/sw30
retla       0x88        ;/sw31
retla       0x92        ;/sw32

retla       0xa2        ;/sw33
retla       0x8a        ;/sw34
retla       0x9a        ;/sw35
retla       0x90        ;/sw36
retla       0xb0        ;/sw37
retla       0x98        ;/sw38
retla       0x3a        ;/sw39
retla       0x02        ;/sw40

retla       0x32        ;/sw41
retla       0x2a        ;/sw42
retla       0x12        ;/sw43
retla       0x22        ;/sw44
retla       0x0a        ;/sw45
retla       0x1a        ;/sw46
retla       0xf8        ;/sw47
retla       0xc0        ;/sw48

retla       0xf0        ;/sw49
retla       0xe8        ;/sw50
retla       0xd0        ;/sw51
```

```
        retla       0xe0        ;/sw52
        retla       0xc8        ;/sw53
        retla       0xd8        ;/sw54
        retla       0x78        ;/sw55
        retla       0x40        ;/sw56

        retla       0x70        ;/sw57
        retla       0x68        ;/sw58
        retla       0x50        ;/sw59
        retla       0x60        ;/sw60
        retla       0x48        ;/sw61
        retla       0x58        ;/sw62
        retla       0xb8        ;/sw63
        retla       0xa8        ;/sw64

        retla       0x38        ;/sw65
        retla       0x28        ;/sw66
        retla       0xfa        ;/sw67
        retla       0xc2        ;/sw68
        retla       0xf2        ;/sw69
        retla       0xea        ;/sw70
        retla       0xd2        ;/sw71
        retla       0xe2        ;/sw72

        retla       0xca        ;/sw73
        retla       0xda        ;/sw74
        retla       0x7a        ;/sw75
        retla       0x42        ;/sw76
        retla       0x72        ;/sw77
        retla       0x6a        ;/sw78
        retla       0x52        ;/sw79
        retla       0x62        ;/sw80

        retla       0x4a        ;/sw81
        retla       0x5a        ;/sw82
        retla       0xba        ;/sw83
        retla       0xb2        ;/sw84
        retla       0xaa        ;/sw85
        retla       0x10        ;/sw86
        retla       0x18        ;/sw87
        retla       0x20        ;/sw88

;----------------------------延时子程序-----------------------------------
delkms                          ;延时 10ms
        movla       .10
        movam       del_m0
loop1
        movla       .250
        movam       del_m1
```

```
loop2
        nop
        decmsz      del_m1,m
        lgoto       loop2
        nop
        decmsz      del_m0,m
        lgoto       loop1
        ret

;---------------------------tmr0    interrupt--------------------
tmr0_int:
        movam       acc_buf         ;将累加器内容送入 a_temp 中进行保护
        swapm       status,a        ;将状态寄存器 status 高低半字节内容
                                    ;相互交换结果送累加器中
        movam       status_buf      ;最后将结果送入 status_temp 中进行保护

        incm        tm_r,m          ;tm_r 加 1 送入 m
        movla       .138            ;;给 tmr0 重装值
        movam       timr0
timer0_int_end:
        bcm         irq,0           ;清 tmr0 中断请求标志
        swapm       status_buf,a    ;将 status_temp 寄存器高低半字节内容相互交换
                                    ;结果送累加器中
        movam       status          ;累加器内容送状态寄存器 status,以恢复进中
                                    ;断前的 status 状态
        movm        acc_buf,a
        reti                        ;中断返回，执行该指令后，程序将自动开中断

;----------------------------------主程序-----------------------------------
main
        clra
        iodir       ra              ;porta 口设为输出态
        clrm        ra              ;porta 口地址初始化

        clrm        ra_buf          ;general resisterd 的地址初始化
        clrm        scan_lp
        clrm        send_lp
        clrm        key_value
        clrm        key_hang
        clrm        key_lie
        clrm        del_m0
        clrm        del_m1
        clrm        tm_r
        clrm        cont_t
        clrm        acc_buf
        clrm        status_buf
        clrm        temp
```

```
        clrm        public_l
        clrm        public_h
        clrm        temp_buf

        clrm        irq         ;清中断标志
        movla       0x46        ;modl,modh 寄存器的设置
        movam       modl
        movla       0x8c
        movam       modh
        clrm        cg_ctrl     ;清 cg_ctrl

;********************按键扫描*******************
Scan_key:
        movla       0xfd        ;唤醒及下拉功能设定
        movam       pstat
        nop
        nop
        nop
        nop
;---------------------------------- ----------------------------------
scan_rb0:
        movla       b'10000000' ;
        movam       ra_buf      ;将 a 中的值送入 porta 的缓冲器
        movla       .8          ;扫描 8 次
        movam       scan_lp
        movla       .0          ;扫描第 0 列
        movam       key_lie
scan_rb0_loop:
        movm        ra_buf,a    ;将 ra_buf 缓冲器的值送入寄存器 a 中
        movam       ra
        nop
        nop
        nop
        btmss       rb,0        ;
        lgoto       scan_rb0_nt ;
        lcall       delkms      ;调用延时程序，按键消抖
        btmss       rb,0        ;再次进行按键扫描
        lgoto       scan_rb0_nt ;
        decm        scan_lp,a
        movam       key_hang    ;确有按键盘按下，将按键所在的行送入寄存器 a 中
        lgoto       key_dsp     ;跳转到按键处理程序
scan_rb0_nt:
        bcm         status,c    ;将状态寄存器 status 的 bit0 置 0
        rrm         ra_buf,m    ;
        decmsz      scan_lp,m   ;判断 scan_lp 减 1 是否为 0
        lgoto       scan_rb0_loop;不为 0，则继续扫描
;---------------------------------- ----------------------------------
```

```
scan_rb1:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .1
        movam       key_lie
scan_rb1_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,1
        goto        scan_rb1_nt

        lcall       delkms          ;延时消抖
        btmss       rb,1            ;判断 rb 是否全为 0,
        lgoto       scan_rb1_nt     ;为 0，继续扫描，
        decm        scan_lp,a       ;不为 0,继续扫描
        movam       key_hang
        lgoto       key_dsp
scan_rb1_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb1_loop
;--------------------------------- ---------------------------------
scan_rb2:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .2
        movam       key_lie
scan_rb2_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,2
        lgoto       scan_rb2_nt

        lcall       delkms
        btmss       rb,2
        lgoto       scan_rb2_nt
        decm        scan_lp,a
```

```
        movam       key_hang
        lgoto       key_dsp
scan_rb2_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb2_loop
;---------------------------------- ----------------------------------
scan_rb3;
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .3
        movam       key_lie
scan_rb3_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,3
        lgoto       scan_rb3_nt

        lcall       delkms
        btmss       rb,3
        lgoto       scan_rb3_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rb3_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb3_loop
;---------------------------------- ----------------------------------
scan_rb4:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .4
        movam       key_lie
scan_rb4_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
```

```
        nop
        btmss       rb,4
        lgoto       scan_rb4_nt

        lcall       delkms
        btmss       rb,4
        lgoto       scan_rb4_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rb4_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb4_loop
;---------------------------------- ----------------------------------
scan_rb5:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .5
        movam       key_lie
scan_rb5_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,5
        lgoto       scan_rb5_nt

        lcall       delkms
        btmss       rb,5
        lgoto       scan_rb5_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rb5_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb5_loop
;---------------------------------- ----------------------------------
scan_rb6
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
```

```
        movla       .6
        movam       key_lie
scan_rb6_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,6
        lgoto       scan_rb6_nt

        lcall       delkms
        btmss       rb,6
        lgoto       scan_rb6_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rb6_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rb6_loop
;---------------------------------- ----------------------------------
scan_rb7:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .7
        movam       key_lie
scan_rb7_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rb,7
        lgoto       scan_rb7_nt

        lcall       delkms
        btmss       rb,7
        lgoto       scan_rb7_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rb7_nt:
        bcm         status,c
        rrm         ra_buf,m
```

```
        decmsz      scan_lp,m
        lgoto       scan_rb7_loop
;---------------------------------- ----------------------------------
 scan_rc0:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .8
        movam       key_lie
scan_rc0_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rc,0
        lgoto       scan_rc0_nt

        lcall       delkms
        btmss       rc,0
        lgoto       scan_rc0_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rc0_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rc0_loop
;---------------------------------- ----------------------------------

scan_rc1:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .9
        movam       key_lie
scan_rc1_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rc,1
        lgoto       scan_rc1_nt
        lcall       delkms
        btmss       rc,1
```

```
        lgoto       scan_rc1_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rc1_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rc1_loop

;---------------------------------- ---------------------------------
scan_rc2:
        movla       b'10000000'
        movam       ra_buf
        movla       .8
        movam       scan_lp
        movla       .10
        movam       key_lie

scan_rc2_loop:
        movm        ra_buf,a
        movam       ra
        nop
        nop
        nop
        btmss       rc,2
        lgoto       scan_rc2_nt

        lcall       delkms
        btmss       rc,2
        lgoto       scan_rc2_nt
        decm        scan_lp,a
        movam       key_hang
        lgoto       key_dsp
scan_rc2_nt:
        bcm         status,c
        rrm         ra_buf,m
        decmsz      scan_lp,m
        lgoto       scan_rc2_loop
        lgoto       sleep_wkup    ;无键按下，扫描结束，进入 sleep_wkup

;---------------------------------- ---------------------------------
sleep_wkup:
        movla       0xfd          ;唤醒功能的设置
        movam       pstat
        movla       0xff          ;portb and portc 唤醒
        movam       ra            ;porta 输出高电平，使得有键按下后使 portb
                                  ;portc 端口电平变化
```

```
        nop
        nop                     ;无按键按下，进入睡眠状态
        sleep
        nop                     ;sleep 唤醒后回到 3FFH
        nop


;-------------------------按键处理程序;--------------------------------
key_dsp:
        bcm         status,c    ;得键值
        rlm         key_hang,m
        rlm         key_hang,m
        rlm         key_hang,m
        rlm         key_hang,m
        rlm         key_hang,m

        rlm         key_hang,m
        rlm         key_lie,m
        rlm         key_hang,m
        rlm         key_lie,m
        rlm         key_hang,m
        rlm         key_lie,m

        movm        key_lie,a
        movam       key_value   ;键值移入 KEY_VALUE

;---------------------------------- ----------------------------------;获得键码

        movm        key_value,a ;将 key_value 移入 a 中
        lcall       seek_code   ;调用键码表 seek_code
        movam       temp        ;将 key_value 送入 temp 中
        movam       temp_buf    ;将 temp 的值送入 temp_buf 缓冲器保护

;---------------------------------- ----------------------------------;发送头码
send_code_loop_st:
        lcall       code_head
        lcall       code_560us
;---------------------------------- ----------------------------------发送键码
        movla       00h         ;客户码设置初始值
        movam       public_h
        movla       0ffh
        movam       public_l

        movla       .24         ;当 send_lp=16 时，发客户码的正码及其反码
        movam       send_lp     ;余下 send_lp=8 时，发键码的正码
send_code_loop11:
        rlm         temp,m      ;temp 中的 key_value 进位循环左移
```

```
        rlm         public_l
        rlm         public_h
        btmsc       status,c        ;判断状态寄存器的 bit0 是否为 0，
                                    ;不为 0 则调用发码子程序
                                    ;code_0 子程序
        lgoto       send_code_10;为 0，则调用发码 code_1 子程序

send_code_00
        lcall       code_0          ;调用发码 code_0 子程序
        lgoto       send_code_end11     ;

send_code_10
        lcall       code_1          ;调用发码 code_1 子程序

send_code_end11
        decmsz      send_lp,m       ;判断发码的位减 1 是否为 0,为 0,则发
        lgoto       send_code_loop11;不为 0 则继续发码
;---------------------------------- ----------------------------------发送键码-
send_key_code:
        movla       .8              ;发键码的反码 send_lp=8
        movam       send_lp
        comm        temp_buf,a      ;将 temp_buf 的键值取反送入寄存器 a 中
        movam       temp            ;将键值的反码送入 temp 中
send_code_loop22:
        rlm         temp,m          ;temp 中的 key_value 进位循环左移
        btmsc       status,c        ;判断状态寄存器的 bit0 是否为 0，
                                    ;不为 0 则调用发码子程序 code_0
        lgoto       send_code_11;为 0，则调用发码 code_1 子程序
send_code_01:
        lcall       code_0          ;调用发码 code_0 子程序
        lgoto       send_code_end22;
send_code_11:
        lcall       code_1          ;调用发码 code_1 子程序
send_code_end22:
        decmsz      send_lp,m       ;判断发键码的反码是否结束
        lgoto       send_code_loop22;没有，继续发码
read_wait0:
        movla       .149            ;给 tmr0 重装值
        movam       timr0
        clrm        tm_r            ;清 tm_r
        movla       b'00011010'     ;tmr0 采用内部时钟计数,预分频器分配给 tmr0,
        movam       timer           ;分频比为 1:4
        movla       b'10000000'
        movam       irq             ;开放 tmr0 中断功能
;---------------------------------- ----------------------------------
```

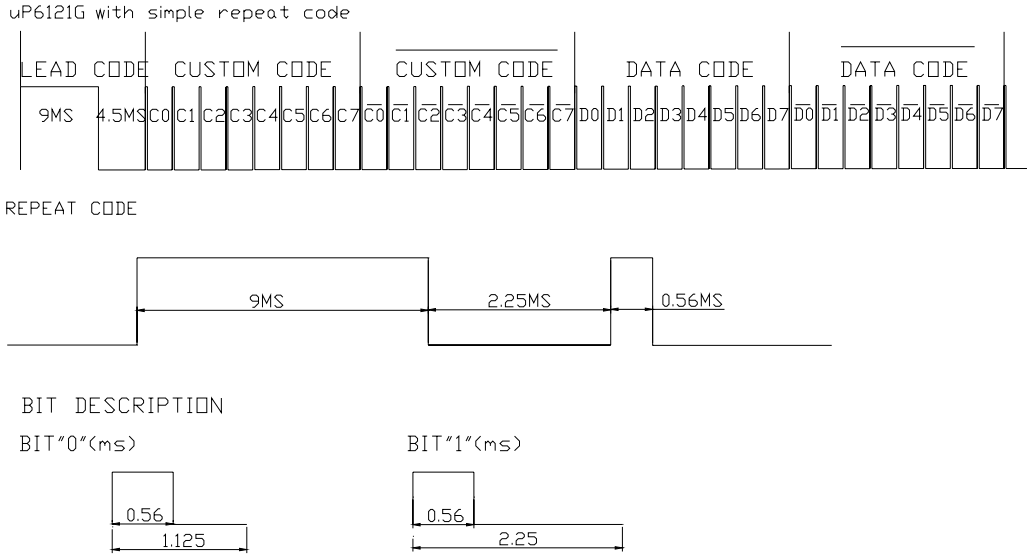
```
read_wait:
        btmsc       rb,0          ;判断按键是否抬起
        lgoto       read_wait_nt  ;没有，则等待，读取时间
        btmsc       rb,1
        lgoto       read_wait_nt
        btmsc       rb,2
        lgoto       read_wait_nt
        btmsc       rb,3
        lgoto       read_wait_nt
        btmsc       rb,4
        lgoto       read_wait_nt
        btmsc       rb,5
        lgoto       read_wait_nt
        btmsc       rc,0
        lgoto       read_wait_nt
        btmsc       rc,1
        lgoto       read_wait_nt
        btmsc       rc,2
        lgoto       read_wait_nt
        btmsc       rb,6
        lgoto       read_wait_nt
        btmsc       rb,7
        lgoto       read_wait_nt
        lgoto       dsp_key_end   ;按键抬起，发码结束
read_wait_nt:
        movm        tm_r,a
        xorla       .66           ;定时 37.0ms
        btmss       status,z      ;判断时间是否到，没到则继续等待
        lgoto       read_wait
        lcall       code_head     ;重复发码
        lcall       code_560us
        lgoto       read_wait0
dsp_key_end;
        clrm        tm_r
        movla       b'00011010'
        movam       timer         ;
        clrm        irq
        lgoto       sleep_wkup     ;停止发码，进入 sleep

;--------- ----------------------发头码子程序;--- --------------------------

code_head:
        movla       b'00011010'   ;tmr0 采用内部时钟计数,预分频器分配给 tmr0
        movam       timer         ;分频比为 1:4
        movla       .122
        movam       timr0         ;给 tmr0 重装值
        movla       b'10000000'
```

```
        movam       irq             ;开放 tmr0 中断功能
        clrm        tm_r
        bsm         cg_ctrl,3       ;启动载波
        movla       .16
        xoram       tm_r,a          ;定时发码为 9.0ms
        btmss       status,z
        lgoto       $-3
        bcm         cg_ctrl,3       ;停止载波发送
        bcm         cg_ctrl,2       ;启动 level
        movla       .155
        movam       timr0
        clrm        tm_r            ;
        movla       .8
        xoram       tm_r,a          ;定时发 level 为 4.50ms
        btmss       status,z
        lgoto       $-3
        clrm        tm_r
        clrm        irq             ;清中断标志
        ret
;-----------------------------------------------------
code_560us:
        movla       b'00011010'     ;tmr0 采用内部时钟计数,预分频器分配给 tmr0
        movam       timer           ;分频比为 1:4
        movla       .122            ;给 tmr0 重装值
        movam       timr0
        movla       b'10000000'     ;开放 tmr0 中断功能
        movam       irq
        clrm        tm_r
        bsm         cg_ctrl,3       ;开启载波
        movla       .1              ;发送载波的时间：560us
        xoram       tm_r,a
        btmss       status,z
        lgoto       $-3
        bcm         cg_ctrl,3
        clrm        tm_r
        clrm        irq             ;清中断标志
        ret
    ;------------------ -----------发码子程序;----------------- ----------------------
code_0;
        movla       b'00011010'     ;tmr0 采用内部时钟计数,预分频器分配给 tmr0
        movam       timer           ;分频比为 1:4
        movla       .130            ;给 tmr0 重装值
        movam       timr0
        bcm         cg_ctrl,2
        movla       b'10000000'     ;开放 tmr0 中断功能
        movam       irq
```

```
        clrm        tm_r
        movla       .1          ;发送 levle 560us
        xoram       tm_r,a
        btmss       status,z
        lgoto       $-3
        movla       .122        ;;给 tmr0 重装值
        movam       timr0
        clrm        tm_r
        bsm         cg_ctrl,3   ;开启载波
        movla       .1          ;发送载波的时间：560us
        xoram       tm_r,a
        btmss       status,z
        lgoto       $-3
        bcm         cg_ctrl,3   ;停止载波
        clrm        tm_r
        clrm        irq         ;清中断标志
        ret
;---------------------------------- ----------------------------------
code_1:
        movla       b'00011010' ;tmr0 采用内部时钟计数,预分频器分配给 tmr0
        movam       timer       ;分频比为 1:4
        movla       .150        ;给 tmr0 重装值
        movam       timr0
        bcm         cg_ctrl,2
        movla       b'10000000' ;开放 tmr0 中断功能
        movam       irq
        clrm        tm_r
        movla       .3          ;发送 levle 1670us
        xoram       tm_r,a
        btmss       status,z
        lgoto       $-3
        movla       .122
        movam       timr0       ;;给 tmr0 重装值
        clrm        tm_r
        bsm         cg_ctrl,3   ;开启载波
        movla       .1          ;发送载波的时间：560us
        xoram       tm_r,a
        btmss       status,z
        lgoto       $-3
        bcm         cg_ctrl,3   ;停止载波
        clrm        tm_r
        clrm        irq         ;清中断标志
        ret
;---------------------------------- ----------------------------------
        end
```

Note: **Upd6121** 发码的波形如下所示：

A single pulse, modulated with 4MHz。



应用线路图如下：