

## TS9638 触摸屏控制芯片技术规格

### 芯片简介：

TS9638 是新一代 4 线电阻式触摸屏控制核心芯片。TS9638 用创新的方法将触摸屏的控制接口转化为红外遥控接口。大大降低了系统开发人员的工作难度。

TS9638 开发接口简易，性能稳定，价格低廉，非常适合车载 DVD 和液晶电视的应用，是这类系统应用触摸屏的首选。TS9638 在模拟红外遥控数据的同时，还能发送连续坐标信息。这样就可以同时满足车载 DVD 与车载 GPS 双系统的需要。运用 TS9638 后，可以将触摸屏的应用扩展到许多方面，例如音响、工业控制领域。

### 该芯片具有以下几个鲜明的特色：

1. 提供 WINDOWS 平台的开发软件，用户只需要极短的时间就可以完成自己的开发。
2. 10bit（8 位物理精度+2 位软件精度）精度的 A/D 转换。
3. 内置超强的环境自适应控制电路。
4. 内置超强错误信息过滤与平滑算法。
5. 内置坐标变换算法。
6. 内置校正坐标信息存储接口。
7. 可模拟发送遥控器 HT622X 数据格式。
8. 对外围器件精度要求非常低，非常适合生产的需要。
9. 其外围电路非常简单，成本低廉。

### 工作原理：

TS9638 提供 10bit 的 A/D 采样精度获取触摸屏上的坐标位置。芯片在空闲时间会自动对触摸屏及外围环境因素进行测量，以保证外围环境的变化不影响 A/D 的测量结果。

TS9638 通过 USB 接口连接到 PC 上，开发工程师先设定好基本的数据。

工厂在生产阶段将数据下载到 TS9638 的系统中。

TS9638 根据 EEPROM 中设定的信息来发送 RS232 数据和 HT622X 数据。

### 技术规格：

工作电压：4.5 - 5.5V

工作电流：小于 10 毫安

工作温度：-5 - +70 摄氏度

储藏温度：-25 - +125 摄氏度

### 端口特性：

TS9638 通过 TXD 端口发送 RS232 数据，通过 HT622X\_OUT 端口发送模拟红外信号。

这两个端口在内部有 100K 的上拉电阻，默认状态为高电平。若要禁止某一个端口发送数据，只需要将该端口的状态拉低就可以了。

**数据格式：**

RS232 数据格式：

标准 RS232 数据格式，波特率：9600bps，8 个数据位，一个停止位，无校验位

数据格式：每组坐标有 5 个字节，每秒钟发送约 60 - 80 组坐标值。

BIT	7 6		5 4 3 2	1 0				
0 1 0			0 0 0 0	0 W				orking
1 0 0			0 X	11	X10	X9	X8	X7
2	0	X6	X5 X4	X3	X2	X1	X0	
3 0 0			0 Y	11	Y10	Y9	Y8	Y7
4	0	Y6	Y5 Y4	Y3	Y2	Y1	Y0	

Working 标志表示触摸屏是否发生触动

Working = 1，表示检测到触动；Working = 0，表示触动停止；

有数据时发送 81, XH, XL, YH, YL

抬笔发送一组 80, XH, XL, YH, YL

没有校正坐标以前，TS9638 发送的数据是原始坐标，X，Y 坐标范围均为（100 - 1400）  
一旦完成坐标校正，TS9638 发送的数据的范围将自动调整为 0 - 1024。

**注意：**

由于触摸屏安装时方位的差异，可能发生原始坐标中 X，Y 数据颠倒，最大、最小值颠倒的现象。一旦经过坐标校正后，TS9638 都能根据校正信息将数据调整成标准的方式：1024 × 1024，X 坐标为从左到右，Y 坐标为从上到下。左上角为系统坐标原点。

由于 TS9638 发送红外数据，会明显影响串口数据的传送速率。对于在 GPS 系统或者类似 PDA 的系统上，需要完成画线、写字的功能，会出现笔迹反应比较慢的现象。这时只要拉低红外端口就可以消除这个影响了。

**HT622X 数据格式：**

具体格式请参照附件《HT6221/2 解码成标准的键值》，内有解码的源代码。你也可以到 HOLTEK 公司的网站上下载相关芯片的文档。

TS9638 可以模拟 30 个设定的红外按键值，可以少于 30 个设定值，不能大于 30 个设定值。模拟按键的个数最少为 1，若设定按键个数为 0，系统默认设置的数据无效，此时系统只通过 RS232 发送原始数据，校正数据将不起任何作用。

HT622X 的数据由两部分组成，分别为两个字节的工厂编号和一个字节的按键值。所有的按键值只能一个有共同的工厂编号。发送数据时先发送工厂编号的低字节，再发送数据的高字节。每个字节发送时，先发送 MSB，最后发送 LSB。

在设置软件中，用户可以将触摸屏的工作区域切分成 n 个方块，n 的范围是 0-255。设置软件默认将工作区域分切成 16 × 16，共 256 个区域。用户可以在分切好的区域中选择不超过 30 个区域，并给每个区域定义要发送的红外码值。这 30 个区域中的红外码值可以重复。

**注意：以下内容与HT622X的数据格式定义不相同。**

若用户点击的位置没有设定红外编码，TS9638将直接发送HT622X的格式定义中的连发数据。这样便于主系统唤醒OSD菜单。若按键长期按动，大概每200ms，TS9638将重复发送红外码的全部内容，而不是HT622X定义的连发数据。

**芯片脚位图**

Y1 1	20	XCHECK
X1 2	19	YCHECK
Y2 3	18	CS
X2 4	17	SK
TXD 5	16	DI
IR_OUT 6	15	DO
GND 7	14	D+
NC 8	13	D-
C1 9	12	VCC
OSC1 10	11	OSC2

**脚位说明：**

1. Y 1：接触摸屏的引线
2. X 1：接触摸屏的引线
3. Y 2：接触摸屏的引线
4. X 2：接触摸屏的引线
5. TX D：RS232 串口信号输出
6. IR\_O UT：发送 HT622X 格式的信号
7. GND：地线
8. N C：不接任何东西
9. C1：接 10 $\mu$ 电容
10. O SC1：晶振输入脚
11. O SC2：晶振输出脚
12. V CC：电源
13. D -：接 USB 接口
14. D +：接 USB 接口
15. D O：接 93c46 的 DO
16. D I：接 93c46 的 DI
17. SK：接 93c46 的 SK
18. CS：接 93c46 的 CS
19. YCHECK：Y 环境测量辅助接口
20. XCHECK：X 环境测量辅助接口

**使用方法：****第一步：**

开发工程师先通过遥控设定软件，生成生产用数据初始化文件。工程师可以设定产品的名称、版本号、屏幕切分参数、红外编码。

第二步：

在工厂生产时，运行生产测试软件，开发工程师编写的初始化文件被自动打开。工人依照生产测试软件的指示，校正触摸屏，软件自动将各种设定好的信息写入 TS9638 系统中。

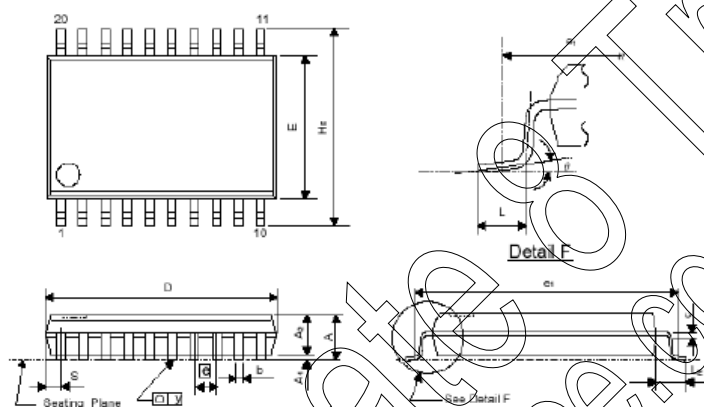
第三步：

数据设定好后就可以对主系统整体进行测试了。

封装信息

SOP 20L (W.B.) Outline Dimensions

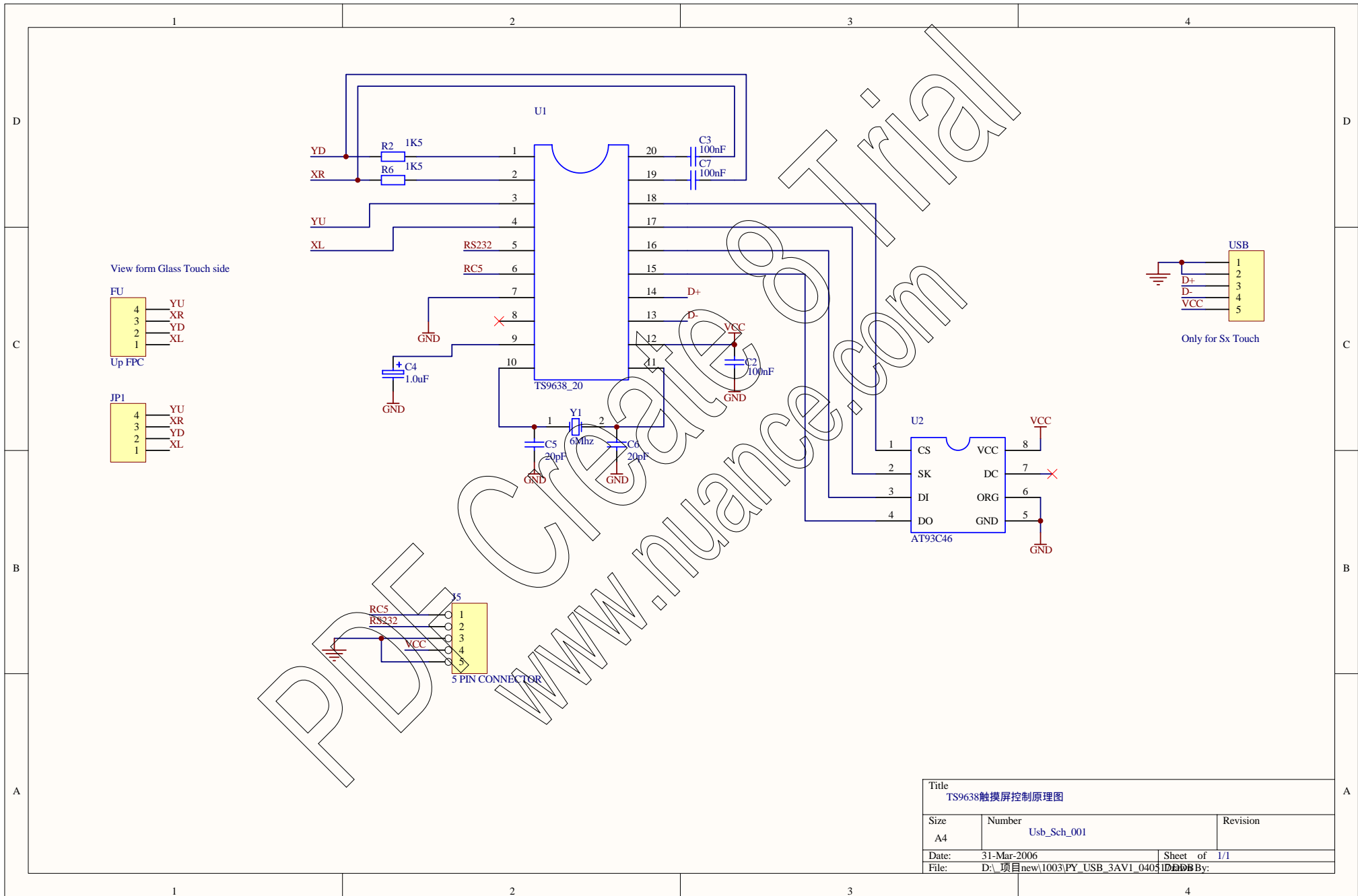
unit: inches/mm



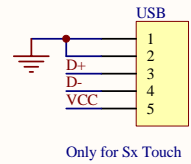
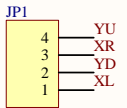
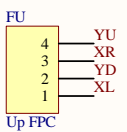
Symbol	Dimensions in inches	Dimensions in mm
$\lambda$	0.06 Max	2.65 Max
A1	0.004 Min.	0.10 Min.
A2	0.062 ± 0.005	2.38 ± 0.13
b	0.016 +0.004 -0.003	0.41 +0.10 -0.05
C	0.010 +0.004 -0.002	0.25 +0.10 -0.05
D	0.500 ± 0.02	12.60 ± 0.51
E	0.295 ± 0.010	7.49 ± 0.25
$\bar{E}$	0.050 ± 0.006	1.27 ± 0.15
A <sub>1</sub>	0.378 NOM.	9.50 NOM.
H <sub>1</sub>	0.406 ± 0.012	10.31 ± 0.31
L	0.032 ± 0.008	0.81 ± 0.20
LE	0.055 ± 0.008	1.40 ± 0.20
S	0.042 Max	1.07 Max
y	0.004 Max	0.10 Max
$\theta$	0° ~ 10°	0° ~ 10°

Notes:

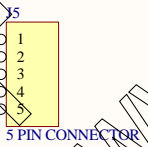
1. The maximum value of dimension D includes end flash.
2. Dimension E does not include resin fins.
3. Dimension e<sub>1</sub> is for PC Board surface mount pad pitch design reference only.
4. Dimension S includes end flash.



View form Glass Touch side



Only for Sx Touch



Title TS9638触摸屏控制原理图		
Size A4	Number Usb_Sch_001	Revision
Date: 31-Mar-2006	Sheet of 1/1	
File: D:\项目\new\1003\PY_USB_3AV1_0403	ID: 111111 By:	

## HT6221/2 解码成标准的键值

(本程序选自广州周立功单片机发展有限公司音响单片机控制软件平台, 程序员: 李奇刚)

### 一. HT6221 遥控器芯片简介

#### 1. 特征

- \* 工作电压: 1.8V~3.5V
- \* Dout 输出 38KHz
- \* 最小发射字: 一个字
- \* 一个 455KHz 的陶瓷或晶体
- \* 16 位地址码
- \* 8 位数据码
- \* ppm 代码方式
- \* 最大活动键     HT6221: 32 键  
                    HT6222: 64 键

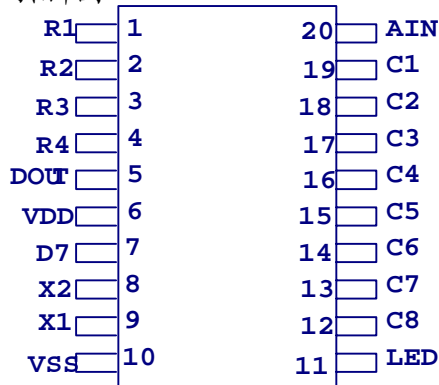
#### 2. 应用

- \* 电视和录像录音机控制器
- \* 夜盗警报系统
- \* 烟火警报系统
- \* 车门控制器
- \* 汽车警报系统
- \* 安全系统
- \* 其它遥控系统

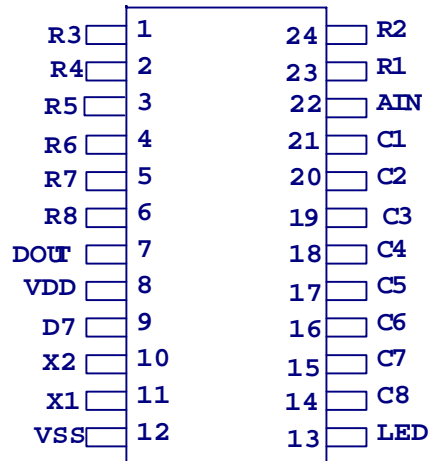
#### 3. 概述

HT6221/HT6222 能编码 16 位地址码和 8 位数据码, HT6221/HT6222 包含 32 键(K1 ~ K32) 和 64 键(K1 ~ K64)

#### 4. 引脚图

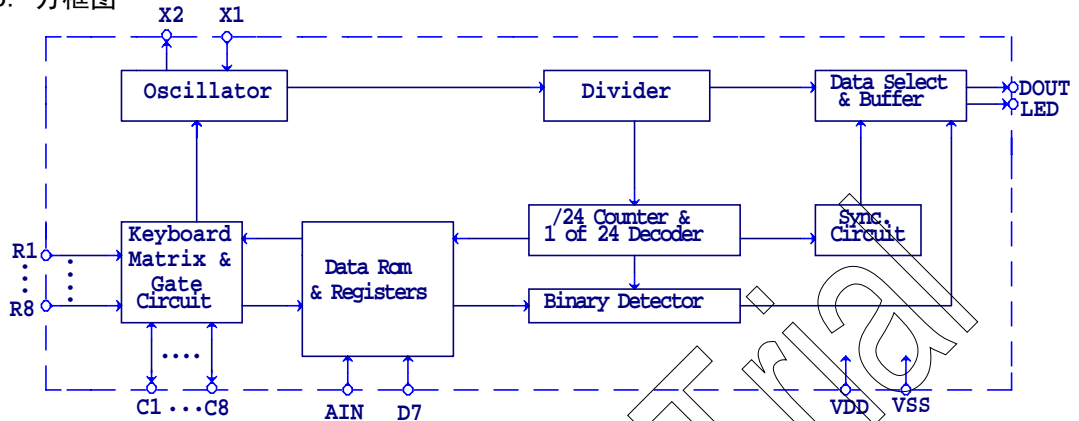


HT6221  
-20 DIP/SOP



HT6222  
-24 DIP/SOP

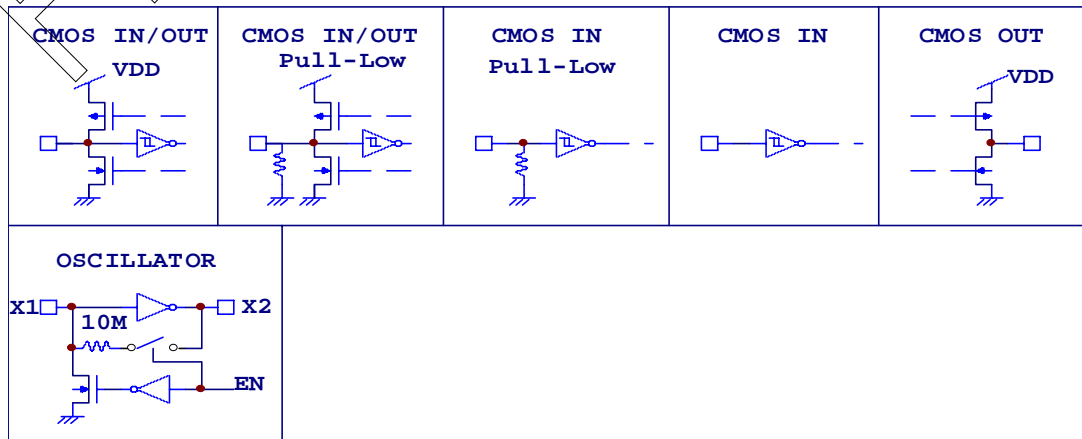
5. 方框图



6. 引脚说明

引脚号	引脚名称	I/O	描述
1~6	R3~R8	输入	键盘行控制, 高电平有效
7	DOUT	输出	串行数据输出引脚, 38KHz 发射频率
8	V <sub>DD</sub>	输入	1.8V~3.5V
9	DT	输入	最重要数据位(DT)代码设置
10	X2	输出	455KHz 振荡器输出
11	X1	输入	455KHz 振荡器输入
12	V <sub>SS</sub>	输入	地
13	LED	输出	发射输出
14~21	C8~C1	输入/输出	键盘列控制
22	AIN	输入	低 8 位地址码输入
23~24	R1~R2	输入	键盘行控制, 高电平有效

7. 内部连接应用电路



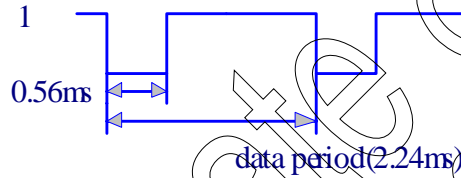
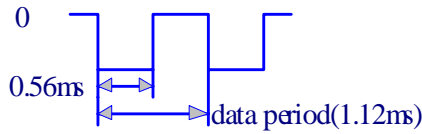
## 二. 代码的特征

### 1. HT6221 键码的形成

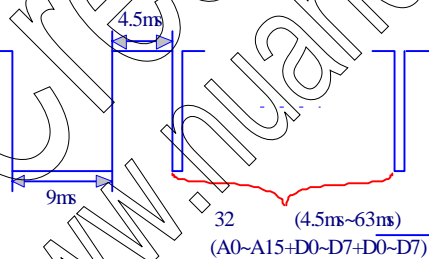
当一个键按下超过 36ms, 振荡器使芯片激活, 如果这个键按下且延迟大约 108ms, 这 108ms 发射代码由一个起始码 (9ms), 一个结果码 (4.5ms), 低 8 位地址码 (9ms~18ms), 高 8 位地址码 (9ms~18ms), 8 位数据码 (9ms~18ms) 和这 8 位数据的反码 (9ms~18ms) 组成。如果键按下超过 108ms 仍未松开, 接下来发射的代码 (连发代码) 将仅由起始码 (9ms) 和结束码 (2.5ms) 组成。

### 2. 代码格式 (以接收代码为准, 接收代码与发射代码反向)

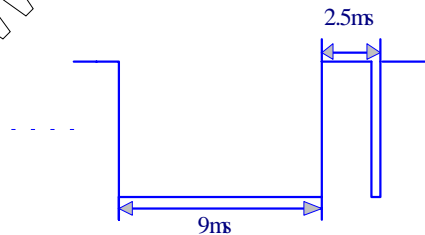
#### ①位定义



#### ②单发代码格式



#### ③连发代码格式



注: 代码宽度算法: 16 位地址码的最短宽度:  $1.12 \times 16 = 18\text{ms}$

16 位地址码的最长宽度:  $2.24\text{ms} \times 16 = 36\text{ms}$

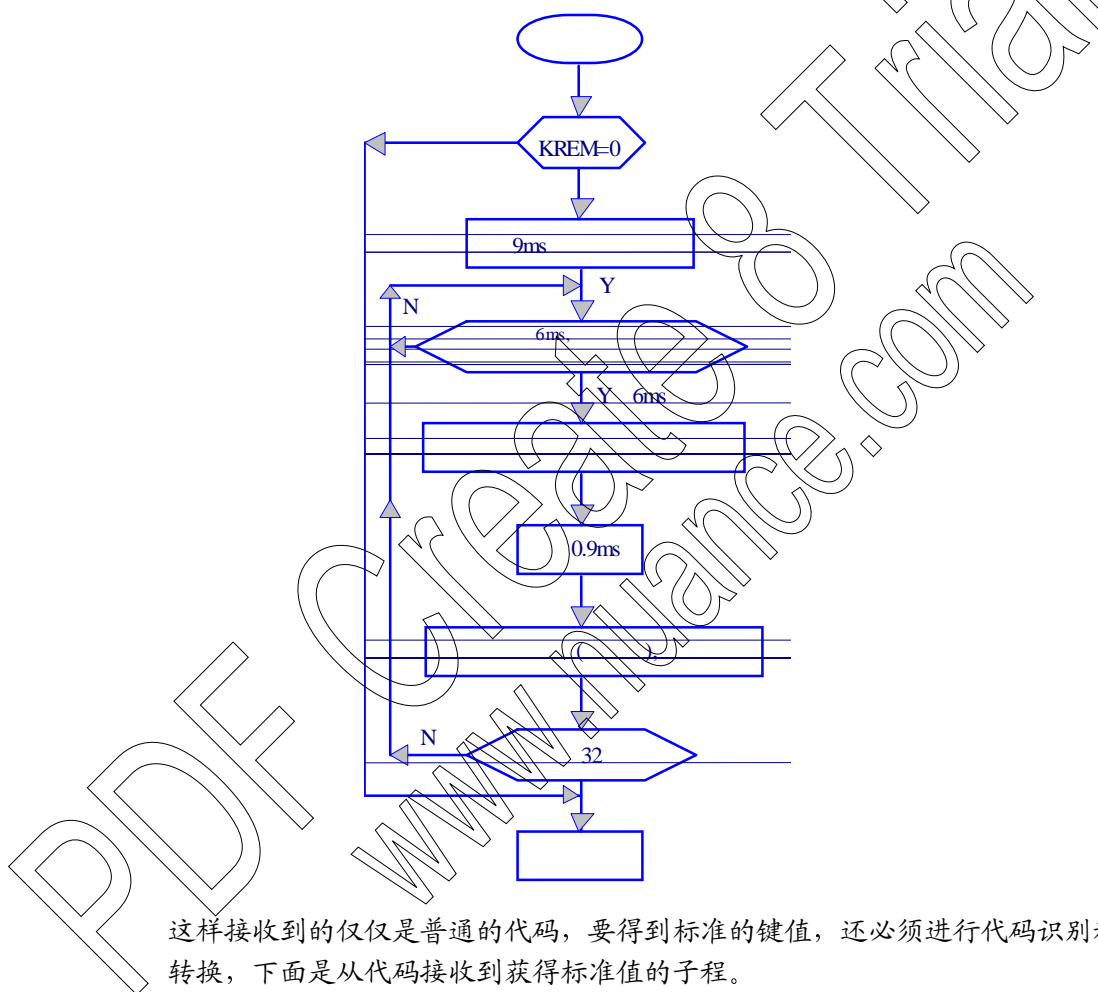
易知 8 位数据代码及其 8 位反代码的宽度和不变:  $(1.12\text{ms} + 2.24\text{ms}) \times 8 = 27\text{ms}$

∴ 32 位代码的宽度为  $(18\text{ms} + 27\text{ms}) \sim (36\text{ms} + 27\text{ms})$

## 三. 解码方法及软件说明



1. 解码的关键是如何识别“0”和“1”，从位的定义我们可以发现“0”、“1”均以0.56ms的低电平开始，不同的是高电平的宽度不同，“0”为0.56ms，“1”为1.68ms，所以必须根据高电平的宽度区别“0”和“1”。如果从0.56ms低电平过后，开始延时，0.56ms以后，若读到的电平为低，说明该位为“0”，反之则为“1”，为了可靠起见，延时必须比0.56ms长些，但又不能超过1.12ms，否则如果该位为“0”，读到的已是下一位的高电平，因此取 $(1.12ms + 0.56ms) / 2 = 0.84ms$ 最为可靠，一般取0.84ms左右均可。
2. 根据码的格式，应该等待9ms的起始码和4.5ms的结果码完成后才能读码。
3. 从上述两点，我们可得到解码程序的流程图。



这样接收到的仅仅是普通的代码，要得到标准的键值，还必须进行代码识别和代码转换，下面是从代码接收到获得标准值的子程。

KREM; 与接收头相连的 I/O 口

1AH,1BH,1CH,1DH; 存放代码的 4 个连续单元

```
YAO_KONG:   CLR     EA
            JNB     KREM,REMOT1
            SJMP    REM_BAK ; 平时 KREM 为高电平，所以当 KREM=1 时，
                               ; 表示无键按下，应立即返回
```

```
REMOT1:     JNB     KREM,$ ; 等待 9ms 的起始码发送完
            MOV     R2,#32 ; 32 表示代码共 32 位，也可以送 24，这样
```

; 接收到的 24 位码将不包括数据代码的  
; 反代码

-----

; 代码接收

```

BYTE1:      MOV      R3,#250
BYTE2:      MUL      AB          ; 延时约 6ms, 可以稍长或稍短, 但不能
                                ; 小于 4.5ms, 也不能太长. 太长连击时
                                ; 将影响程序运行速度
                                JNB      KREM,BYTE3;
                                DJNZ    R3,BYTE2 ;由于结果码为 4.5ms,如果小于 4.5ms,
                                ; 结果码未发送完, 读得的码值将出错
BYTE3:      JNB      KREM,$      ;等待高电平, 保证读每一位的起点一致
    
```

-----

```

                                MOV      R3,#150
                                DJNZ    R3,$      ;延时 0.9ms, 延时范围为 0.56ms~1.12ms
    
```

-----

```

                                MOV      C,KREM
                                MOV      R3,#4
                                MOV      R0,#1DH
BYTE4:      MOV      A,@R0
                                RLC      A
                                MOV      @R0,A
                                DEC      R0
                                DJNZ    R3,BYTE4
                                DJNZ    R2,BYTE1
    
```

;至此 32 位代码已全部接收完成, 并存放在 1AH~1DH 中,  
; 依次为低 8 位地址码, 高 8 位地址码, 8 位数据码,  
; 8 位数据的反代码

-----

; 代码识别

```

M OV      A,1AH
XRL      A,#03          ;3 为地址低 8 位的值, 对于不同的遥控器
                                ; 有不同的地址值

JNZ      REM_BAK
MOV      A,1BH
XRL      A,#0FCH      ;FCH 为地址高 8 位的值
JNZ      REM_BAK
MOV      A,1CH
CPL      A
XRL      A,1DH          ; 如果地址码不对或接收到的数据码两单元
                                ; 不反向均当错码, 本程序当无键按下处理
    
```

```

                JNZ      REM_BAK

;-----
; 代码转换
                MOV      R2,#21      ; 21 为遥控器面板按键数
                MOV      DPTR,#TAB_REMOT
LOOKUP_1:      MOV      A,R2
                MOVC     A,@A+DPTR
                XRL      A,1CH
                JZ       REM_BAK0
                DJNZ     R2,LOOPUP_1
REM_BAK0:      MOV      A,R2      ; R2 中的值即为标准的键值
                SJMP     END_YK
REM_BAK:      CLR      A
END_YK:      SETB     EA
                RET
    
```

;-----  
; 代码转换表，表中的值为面板上相应键的代码  
; 对于不同的遥控器，表中的值应做相应的改变

```

TAB_REMOT:
                DB      0 0H
;
                VCD     DVD     AUX     TUNER     STM     TSV-4     6
                DB      0C0H, 0D0H, 0E8H, 0F0H, 0E0H, 0C8H
;
                UP      DOWN     FM/AM     MEMORY     A/B
                DB      00H, 20H, 48H, 68H, 58H
;
                1       234
                DB      0D8H, 0F8H, 40H, 60H, 50H, 70H
;
                AUTO     VOL+     VOL-     MUTE
                DB      28H, 10H, 30H, 78H
    
```

说明:此程序可在需要的地方任意调用 (LCALL YA0\_KONG), 返回后, 累加器中的值即为标准的键值, 如果 A=0 则不予处理 (可能原因有: 无键按下, 错码或非本机所用的遥控器的操作), 程序中的延时均以 4MHz 的晶振为准, 若用不同的晶振, 只需改变相应值, 符合注释中的延时时间即可。

4. 从上述解码过程我们不难发现, 对于连发代码, 解码得的值 1AH~1DH 全为 0FFH, 所以软件如果需要处理连击, 我们只须在代码识别前判断 1AH~1DH 是否全为 0FFH, 是则有连击现象, 这样建一个连击标志, 再返回, 软件根据这个标志, 结合上一次读得的键值便可进行相应的连击操作, 直到按键松开, 连击标志才被清除。具体操作如下:

在代码识别前插入

```

                MOV      A,1AH
                ANL      A,1BH
    
```

```

ANL      A,1CH
ANL      A,1DH
XRL      A,#0FFH
JNZ      DAN_JI
SETB     FLAG_LIANJI      ;建连击标志
SJMP     END_YK
DAN_JI:  NOP
    
```

另外子程的最后几条指令改成

```

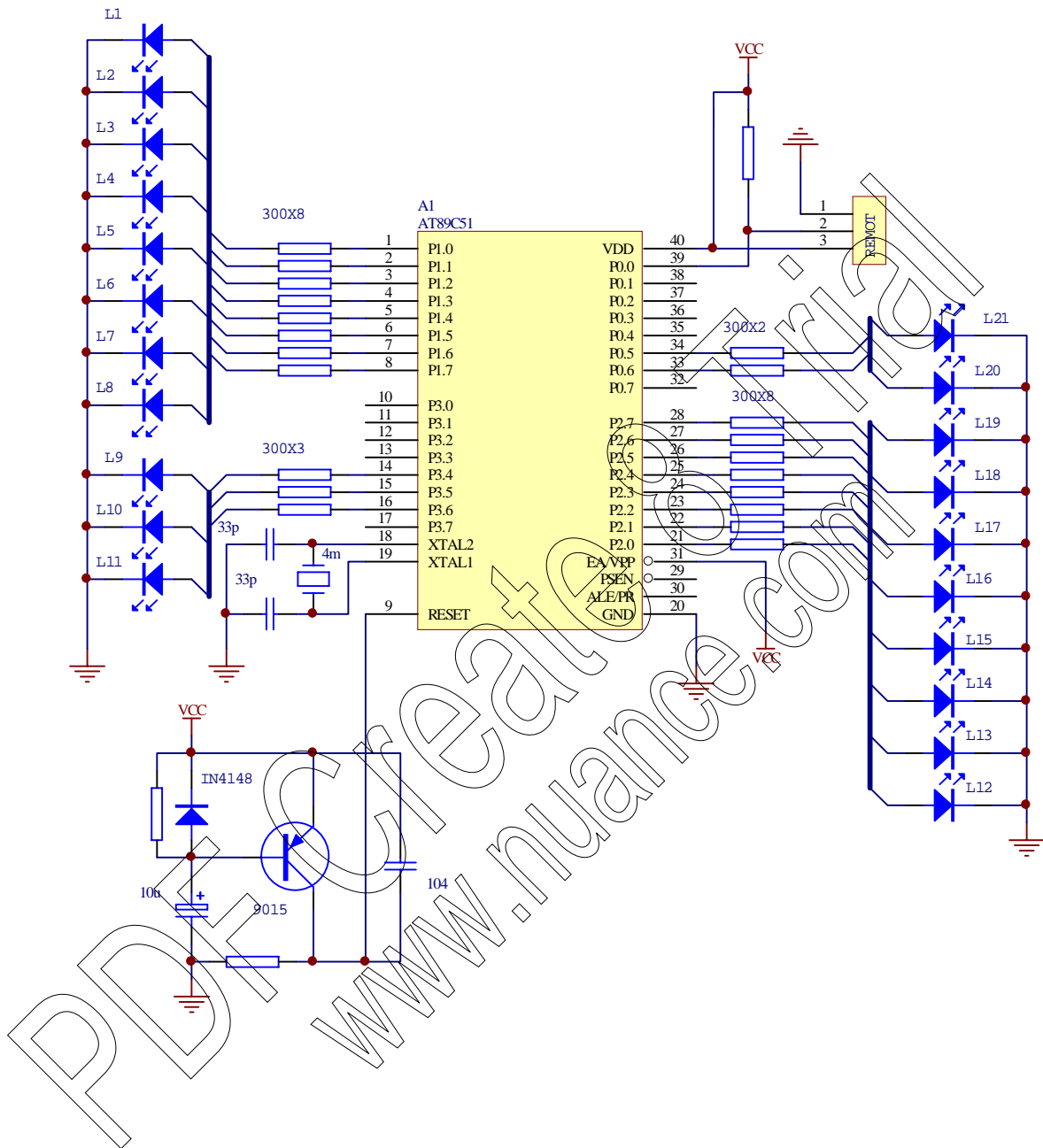
REM_BAK0: MOV     A,R2
          SJMP     END_YK0
REM_BAK:  CLRA
END_YK0:  CLR     FLAG_LIANJI      ;清连击标志
END_YK:  SETB     EA
          RET
    
```

#### 四. 应用实例

读者也许会问，只要解得遥控器的代码就可以了，转换成 1~n 标准值有什么优点呢？看了后面的简单实例，不难发现它有如下好处：

- ①. 在应用系统中，带遥控器的仪器，一般都带按键，而且二者功能相同，转换成标准值后，遥控按键散转表格可以与键盘散转表格复用，这样能节省一定的空间。
- ②. HT622 1/2 最大可支持 32 或 64 个按键，一般系统只使用其中的一部分，这样可能会使遥控器按键的代码毫无规律，为了处理这样的代码，软件人员要么想方设法通过复杂的算法找出那些代码不是规律的规律，要么干脆不管那么多，排列一大堆“CJNE A, #DATA, NEXT”指令判断，使键值判断变得拖沓冗长。使用前述方法则清楚明了，简单易行。
- ③. 对于不同的遥控器，本程序只要改变代码转换表即可，对再开发大有益处。

##### 1. 实例电路



程序清单及说明:

```

KREM    EQU    P0.0
L1      EQU    P1.0
L2      EQU    P1.1
L3      EQU    P1.2
L4      EQU    P1.3
L5      EQU    P1.4
L6      EQU    P1.5
L7      EQU    P1.6
L8      EQU    P1.7
L9      EQU    P3.4
L10     EQU    P3.5
L11     EQU    P3.6
L12     EQU    P2.0
L13     EQU    P2.1
L14     EQU    P2.2
L15     EQU    P2.3
L16     EQU    P2.4
L17     EQU    P2.5
L18     EQU    P2.6
L19     EQU    P2.7
L20     EQU    P0.6
L21     EQU    P0.5
;-----
                ORG    0000H
                AJMP   START0
                ORG    0030H
START0:        MOV     SP, #60H
START:         ACALL  YAO_KONG    ;调用解码子程
                JZ     START
;-----
                RL     A
                MOV   DPTR, #TAB_KEY
                JMP   @A+DPTR
;-----
TAB_KEY:      AJMP   START
                AJMP   KEY1
                AJMP   KEY2
                AJMP   KEY3
    
```

```
AJMP KEY4
AJMP KEY5
AJMP KEY6
AJMP KEY7
AJMP KEY8
AJMP KEY9
AJMP KEY10
AJMP KEY11
AJMP KEY12
AJMP KEY13
AJMP KEY14
AJMP KEY15
AJMP KEY16
AJMP KEY17
AJMP KEY18
AJMP KEY19
AJMP KEY20
AJMP KEY21
```

```
-----
;
KEY1:      ACALL  CLEAR_IO
           SETB   L1
           AJMP  START
```

```
-----
;
KEY2:      ACALL  CLEAR_IO
           SETB   L2
           AJMP  START
```

```
-----
;
KEY3:      ACALL  CLEAR_IO
           SETB   L3
           AJMP  START
```

```
-----
;
KEY4:      ACALL  CLEAR_IO
           SETB   L4
           AJMP  START
```

```
-----
;
KEY5:      ACALL  CLEAR_IO
           SETB   L5
           AJMP  START
```

```
-----
;
KEY6:      ACALL  CLEAR_IO
           SETB   L6
           AJMP  START
```

```
-----
;
KEY7:      ACALL  CLEAR_IO
```

```

                SETB    L7
                AJMP    START
;-----
KEY8:          ACALL   CLEAR_IO
                SETB    L8
                AJMP    START
;-----
KEY9:          ACALL   CLEAR_IO
                SETB    L9
                AJMP    START
;-----
KEY10:         ACALL   CLEAR_IO
                SETB    L10
                AJMP    START
;-----
KEY11:         ACALL   CLEAR_IO
                SETB    L11
                AJMP    START
;-----
KEY12:         ACALL   CLEAR_IO
                SETB    L12
                AJMP    START
;-----
KEY13:         ACALL   CLEAR_IO
                SETB    L13
                AJMP    START
;-----
KEY14:         ACALL   CLEAR_IO
                SETB    L14
                AJMP    START
;-----
KEY15:         ACALL   CLEAR_IO
                SETB    L15
                AJMP    START
;-----
KEY16:         ACALL   CLEAR_IO
                SETB    L16
                AJMP    START
;-----
KEY17:         ACALL   CLEAR_IO
                SETB    L17
                AJMP    START
;-----
KEY18:         ACALL   CLEAR_IO

```



```

                SETB    L18
                AJMP   START
;-----
KEY19:         ACALL  CLEAR_IO
                SETB    L19
                AJMP   START
;-----
KEY20:         ACALL  CLEAR_IO
                SETB    L20
                AJMP   START
;-----
KEY21:         ACALL  CLEAR_IO
                SETB    L21
                AJMP   START
;-----
CLEAR_IO:     MOV     P0, #0
                MOV     P1, #0
                MOV     P2, #0
                MOV     P3, #0
                RET
;-----
                END

```

- ① 该程序是应用“HT6221 解码成标准键值”的典型例子。KEY1、KEY2……KEY21 分别表示不同的功能模块，也就是说根据解得的键值按照需要做具体的事。以示明显，这里分别用 L1、L2……L21 共 21 个发光管代替程序要做的事，当不同的键按下时，对应的发光管亮，其余的管均灭。
- ② 如果读者有兴趣，不防一试。需要注意的是，实验前必须把你手上的遥控器的代码填在代码转换表的相应位置，代码的得来不难。在“代码识别”前设一断点，运行程序，按不同的键，程序运行到断点停下时，“1CH”单元的内容即为当前按键的代码。同时应该用 1AH 中的内容替换程序中的低 8 位地址“3”，1BH 中的内容替换程序中高 8 位地址“0FCH”。如此做好后，便大功告成，这时有且只有你手上的遥控器可以控制发光管了。