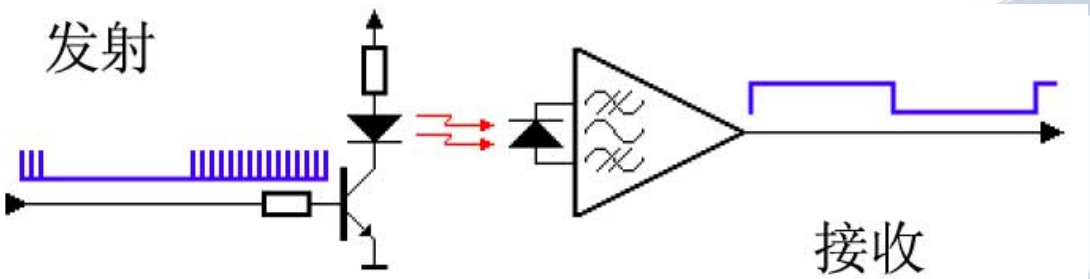
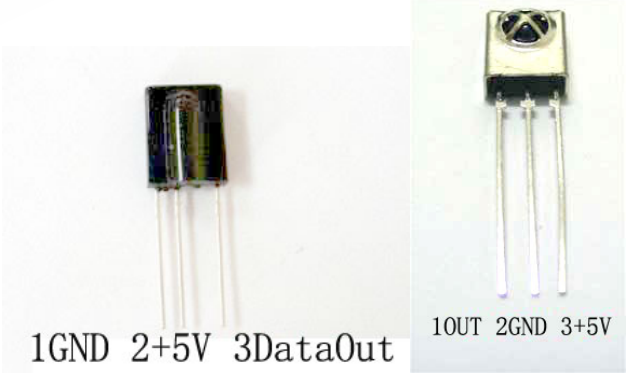
红外线遥控器已被广泛使用在各种类型的家电产品上，它的出现给使用电器提供了很多的便利。红外遥控系统一般由红外发射装置和红外接收设备两大部分组成。红外发射装置又可由键盘电路、红外编码芯片、电源和红外发射电路组成。红外接收设备可由红外接收电路、红外解码芯片、电源和应用电路组成。通常为了使信号能更好的被传输发送端将基带二进制信号调制为脉冲串信号，通过红外发射管发射。常用的有通过脉冲宽度来实现信号调制的脉宽调制（PWM）和通过脉冲串之间的时间间隔来实现信号调制的脉时调制（PPM）两种方法。



在同一个遥控电路中通常要使用实现不同的遥控功能或区分不同的机器类型，这样就要求信号按一定的编码传送，编码则会由编码芯片或电路完成。对应于编码芯片通常会有相配对的解码芯片或包含解码模块的应用芯片。在实际的产品设计或业余电子制作中，编码芯片并一定能完成我们要求的功能，这时我们就需要了解所使用的编码芯片到底是如何编码的。只有知道编码方式，我们才可以使用单片机或数字电路去定制解码方案。下面介绍的是笔者所收集整理的一些常用遥控编码芯片的编码方式和常用一体化接收芯片的引脚示意图。在最后还用实例介绍M50560－001P芯片的解码思路和应用实例程序的编写。

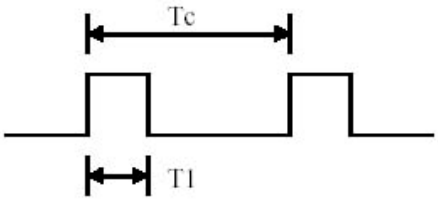
常用红外一体化接收头引脚示意



uPD6121，uPD6122，PT2222，SC6121，HS6222，HS6221

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

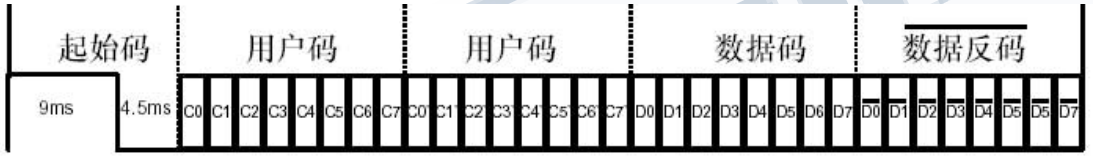
$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

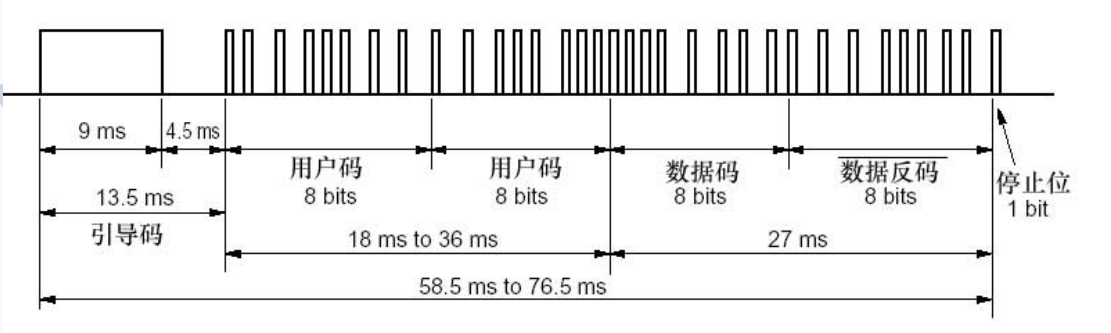
占空比 $= T1/Tc = 1/3$

数据格式.

数据格式包括了引导码、用户码、数据码和数据码反码，编码总占 32 位。数据反码是数据码反相后的编码，编码时可用于对数据的纠错。注意：第二段的用户码也可以在遥控应用电路中被设置成第一段用户码的反码。

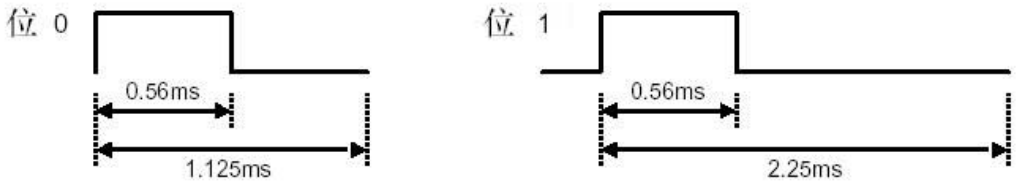


使用 455KHz 晶振时各代码所占的时间



位定义

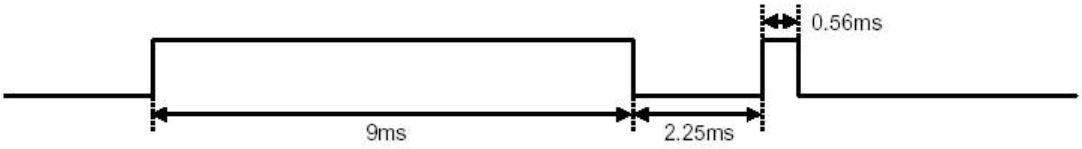
用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。区分‘0’和‘1’是利用脉冲的时间间隔来区分，这种编码方式称为脉冲位置调制方式，英文简写 PPM。



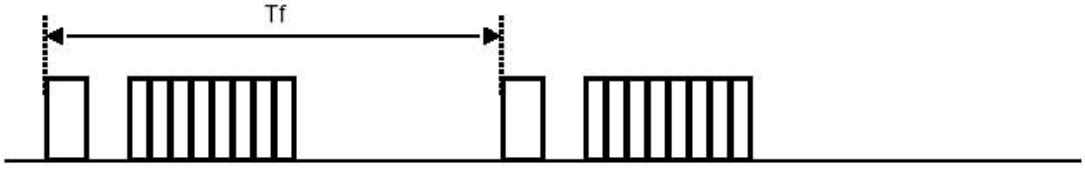
按键输出波形

uPD6121G 按键输出有二种方式：一种是每次按键都输出完整的一帧数据；另一种是按下相同的按键后每发送完整的一帧数据后，再发送重复码，再到按键被松开。

重复码

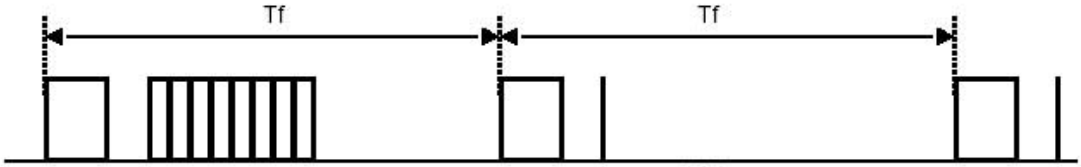


单一按键波形



Tf = 108ms 455KHz晶振

连续按键波形

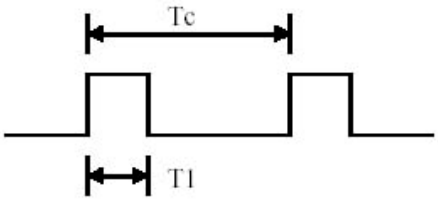


Tf = 108ms 455KHz晶振

TC9012、TC9028、TC9243、SC9012、SC9028、SC9243、HS9012

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

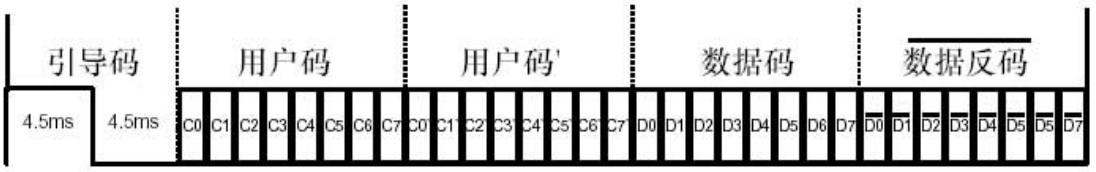
$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

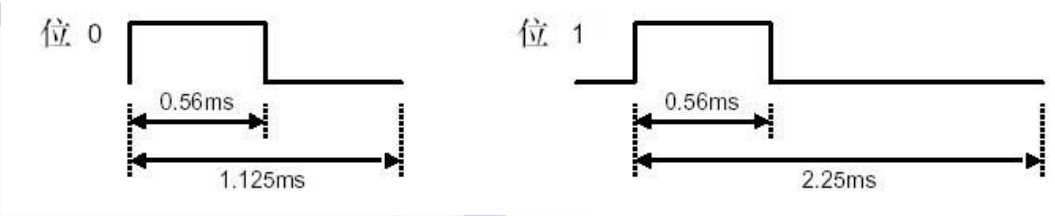
数据格式.

数据格式包括了引导码、用户码、数据码和数据反码，编码共占 32 位。数据反码是数据码反相后的编码，编码时可用于对数据的纠错。用户码是可以用二极管在遥控应用电路板上定义，这样可以把同一型号的芯片用在不同的设备中，也称系统码（system code）。



位定义

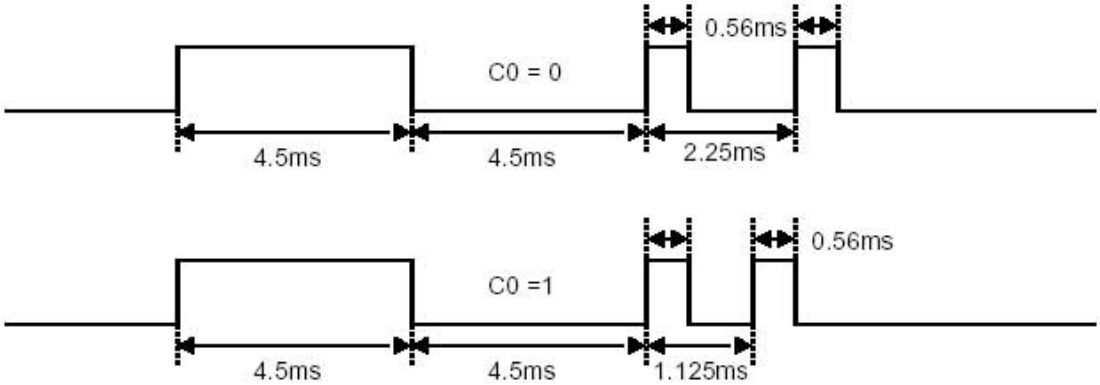
用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。编码方式为 PPM。



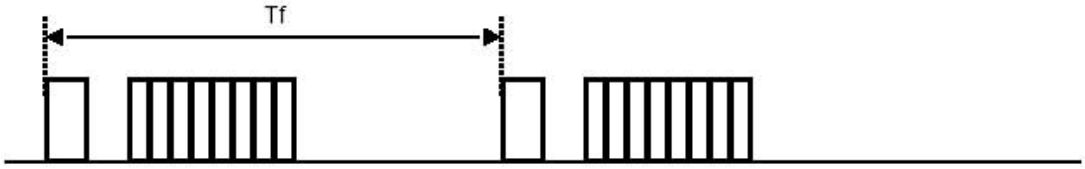
按键输出波形

按键保持按下状态时每发送完整的一帧数据后，再发送重复码，再到按键被松开。此芯片用两种不同的重复码，当用户码的 C0 位为 1 时用一种，C0 位为 0 时使用另一种

重复码

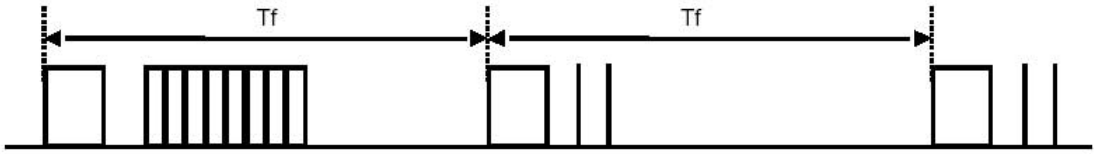


单一按键波形



Tf = 108ms 455KHz晶振

连续按键波形

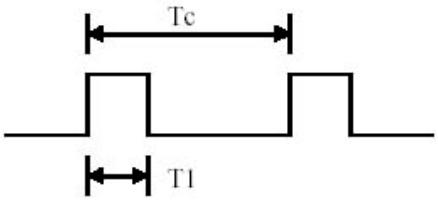


Tf = 108ms 455KHz晶振

M50560-001、M50560-003、M34280、PT2560、SC50560、HS50560

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

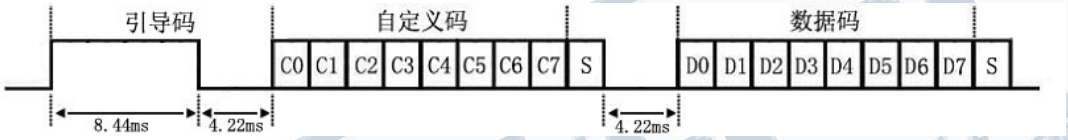
fCAR = 1/Tc = fOSC/12 ≈ 38KHz

fOSC是晶振频率

占空比 = T1/Tc = 1/3

数据格式.

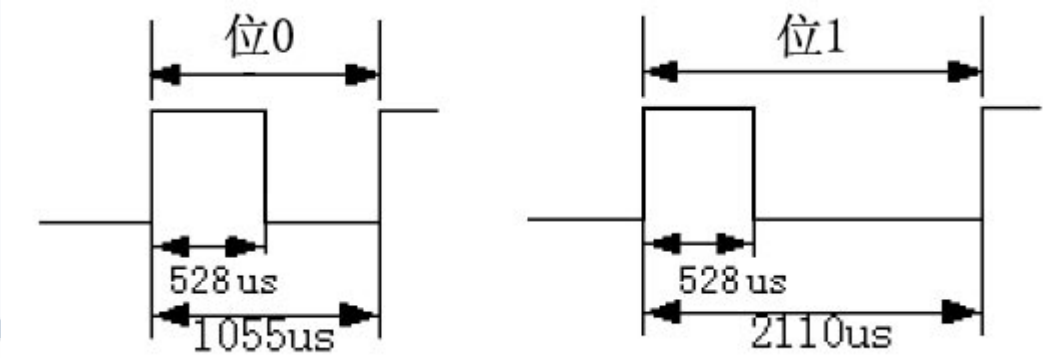
数据格式为每一帧数据包括 8 位自定义码和 8 位数据码，共 16 位，自定义码和数据码后还有同步位。



使用 455 晶振时一帧数据的示例。

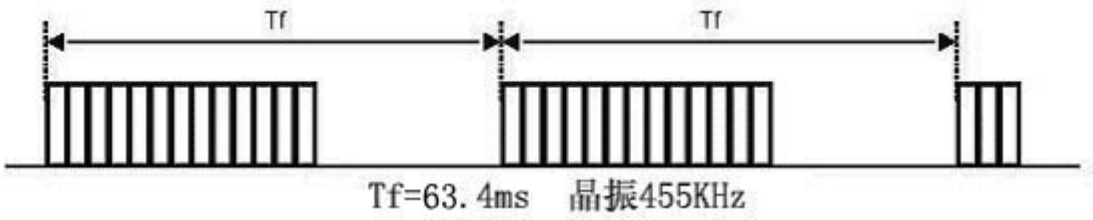
位定义

用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。位 1 的时间是位 0 的两倍。位编码方式是 PPM 方式。



按键输出波形

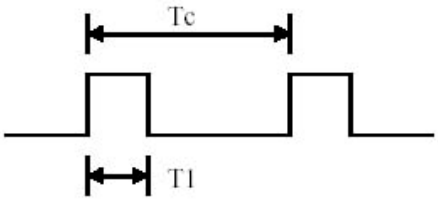
按键按下后输出一帧数据，Tf 周期后再输出另一帧数据，重复输出直到按键松开。



LC7461、LC7462

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

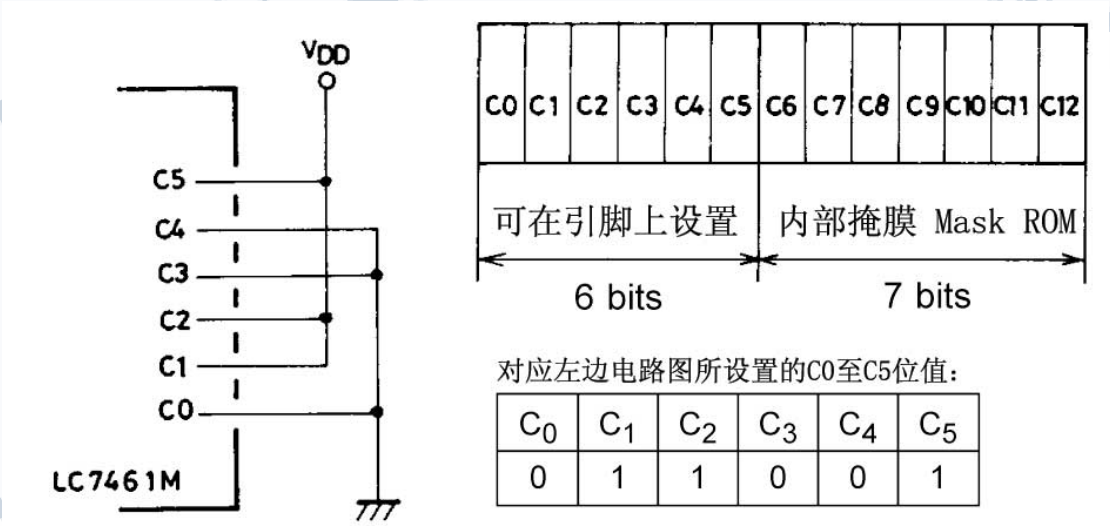
f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

数据格式.

数据格式包括了引导码、用户码、用户反码、数据码和数据反码。用户码有 13 位，其中 C0 至 C5 位可以通过设置芯片的 C0-C5 引脚的高低电位为调整，C6 至 C12 则是在制造时掩膜生成，不同的遥控器可以是不一样的。芯片有反码输出，这样可以很大程序减少了接收的误码率。

C0 至 C12 定义

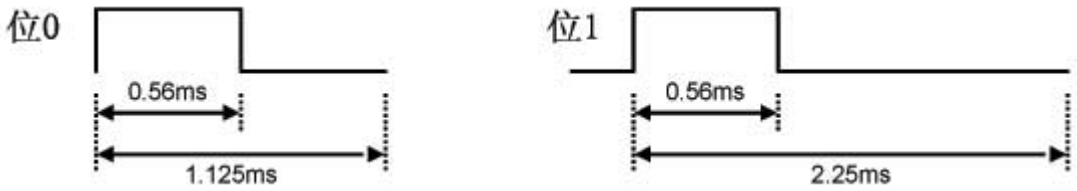


对应左边电路图所设置的C0至C5位值：

C_0	C_1	C_2	C_3	C_4	C_5
0	1	1	0	0	1

位定义

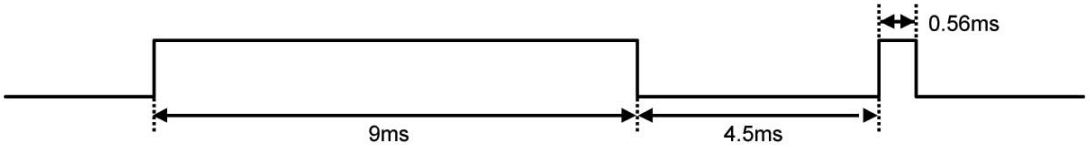
用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。编码方式为 PPM。



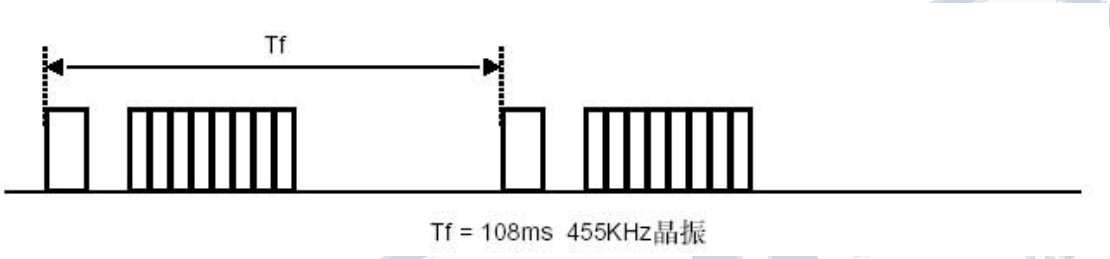
按键输出波形

LC7461M 按键输出有二种方式：一种是每次按键都输出完整的一帧数据；另一种是按下相同的按键后每发送完整的一帧数据后，再发送重复码，再到按键被松开。

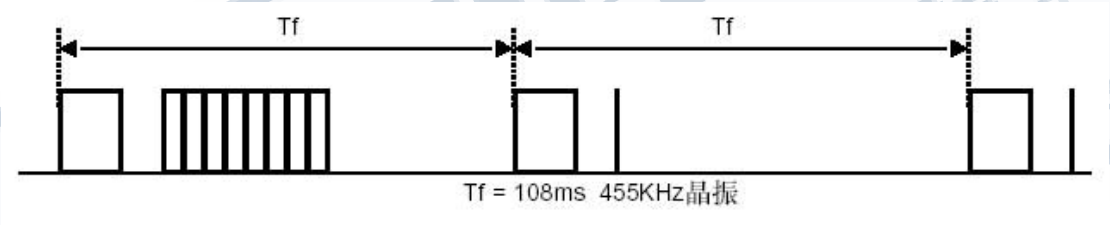
重复码



单一按键波形



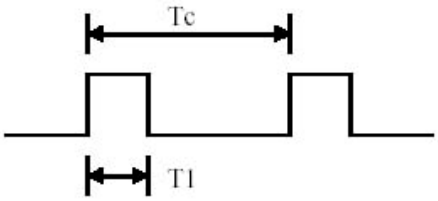
连续按键波形



M3004LAB1、M3004LD

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

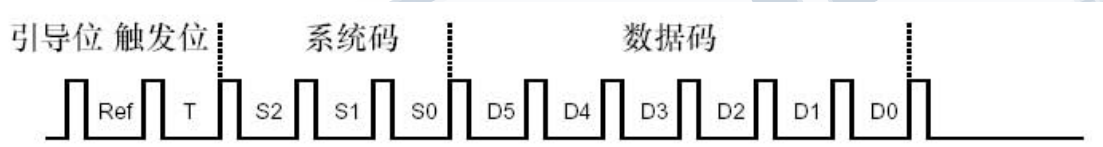
f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

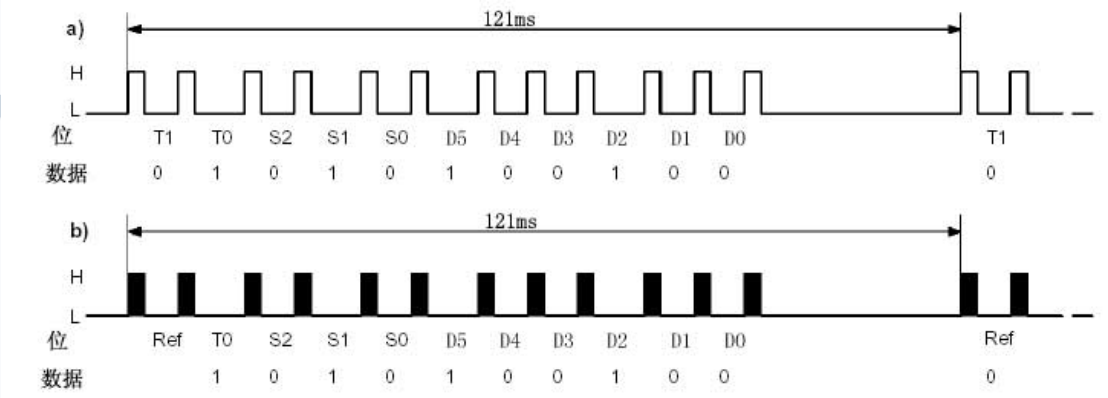
数据格式.

M3004LAB1有二种发射模式，一种是普通的模式闪烁模式(Flashed)，另一种是调制模式(Modulated)。前者的数据格式包括二个触发位(toggle bits)、三个系统位和六个数据位；后者则包括一个引导位(reference time)、一个触发位、三个系统位和六个数据位。其中引导位是只能是逻辑位1。

数据内容



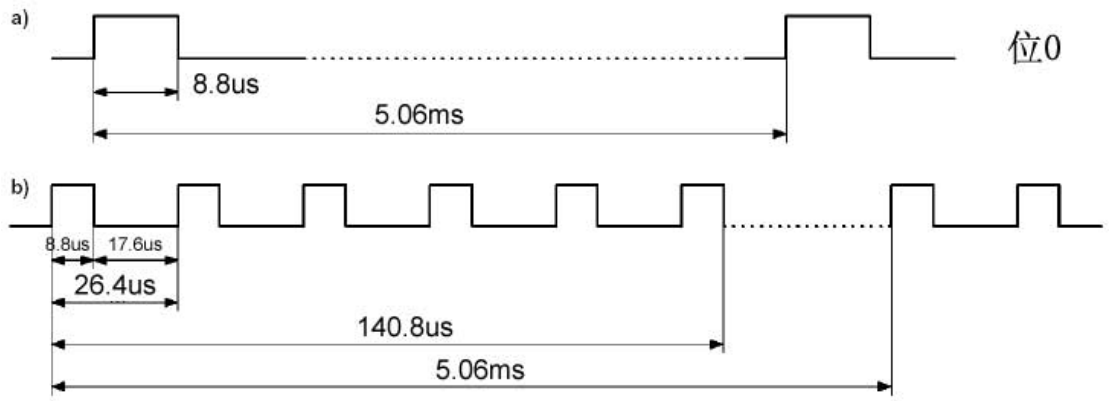
二种发射模式的数据格式(a 普通模式、**b** 调制模式**)**



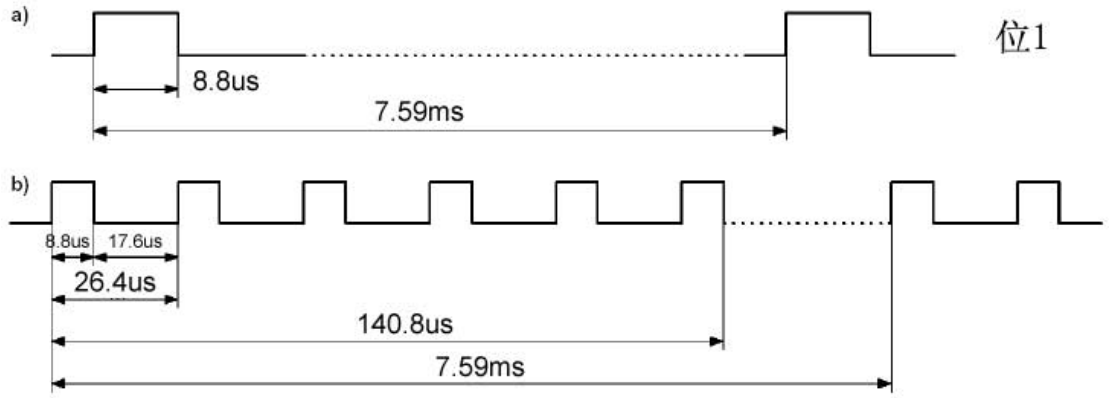
位定义

触发位、系统码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。参看下面的波形图时会发现高电平或被调制部分在位 1 或位 0 都是一样的，在使用单片机进行解码时可以考虑用对波形中低电平的脉宽来进行判断是位 1 还是位 0。

逻辑位 0 的波形(a 普通模式、**b** 调制模式**)**

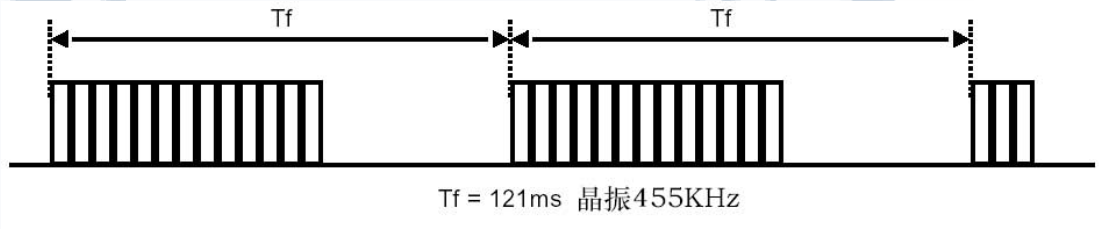


逻辑位 1 的波形



按键输出波形

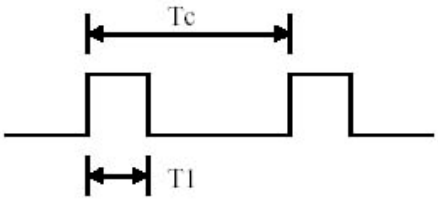
每次按键都输出完整的一帧数据，按键一直被按下时则不断输出同样的一帧数据。



SAA3010，HS3010，SC3010

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

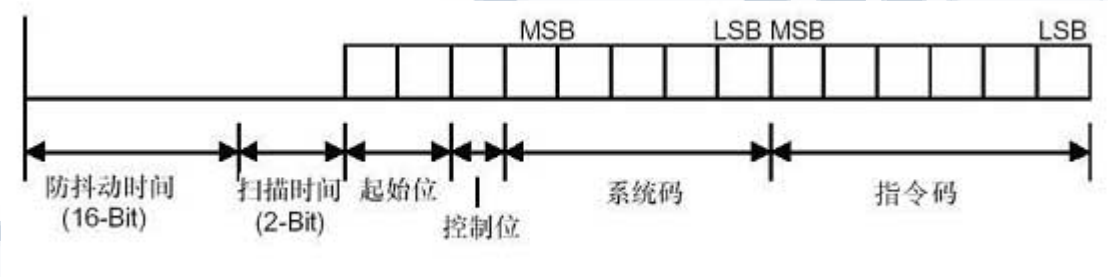
$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

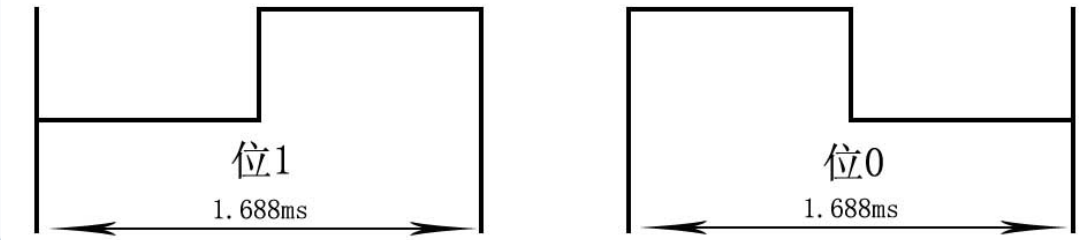
数据格式.

SAA3010是使用RC－5编码，有2位起始位、1位控制位、系统码占5位、指令码占6位，一帧数据共占14位。在第一次按下遥控按键后芯片要经过16位的防抖动时间和2位的扫描时间才会发送第一帧数据。



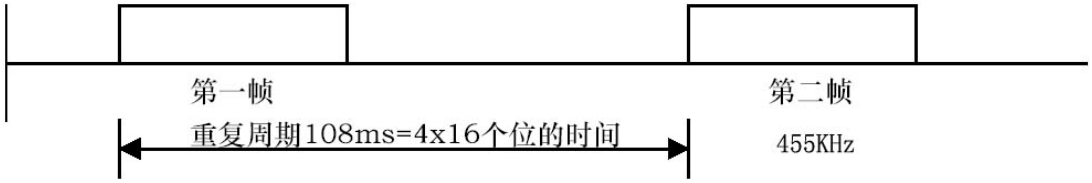
位定义

SAA3010 的位传送方式是采用双相位，位 1 和位 0 的相位正好是相反的。在解码时可以用定时采样的方式进行解码，一个位采样二次，分别在位波形的三分之一和三分之二处进行采样，如位 1 用这种方法采样的值就是 0 和 1。



按键输出波形

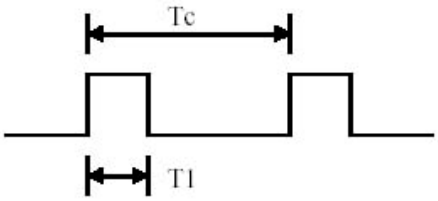
每次按键都输出完整的一帧数据，按键一直被按下时则不断输出同样的一帧数据。



uPD1986

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

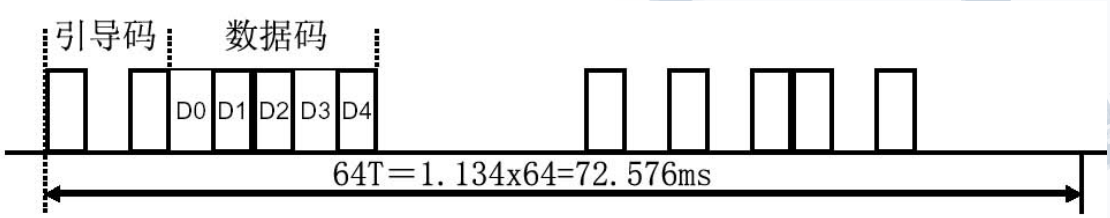
f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

位时间 $= T = 43Tc = 1.134ms$

数据格式.

uPD1986 的每一帧数据占 64 位的时间，包含二段相同的编码，每段各占 32 位时间。其中每段有引导码 3 位，数据码 5 位。

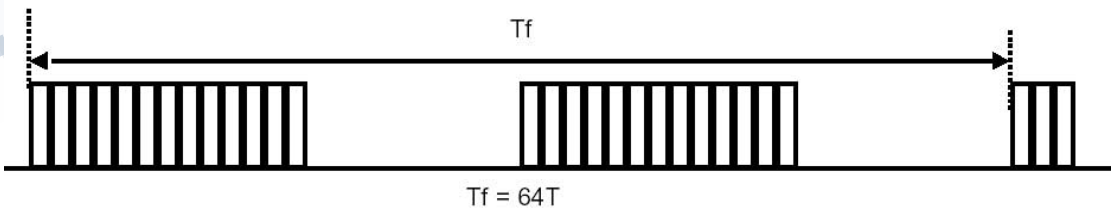


位定义

uPD1986 的位定义十分简单，中用到高低电平来表示。这样的编码方式在使用单片机解码时应注意在采集每一位波形时，可能会出现的干扰的情况。

按键输出波形

每次按键都输出完整的一帧数据，按键一直被按下时则不断输出同样的一帧数据。



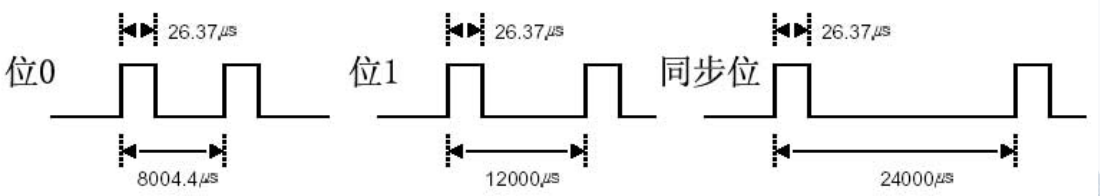
MV400

数据格式

MV400信号没有调制，数据格式包括同步脉冲和5位数据位，在发送数据位前先发送一个同步位脉冲，接着发送数据位，数据位发完后再发一个同步位脉冲。

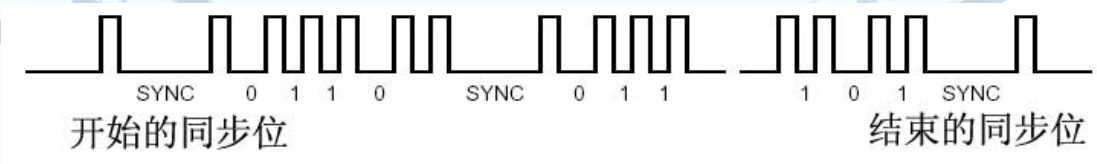
位定义

MV400 的位是根据一个高电平脉冲后的低电平的宽度来判断。



按键输出波形

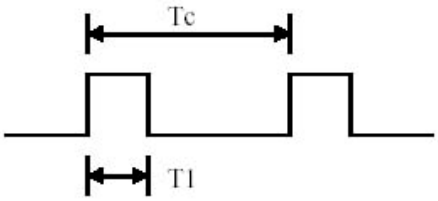
每次按键都输出完整的一帧数据，按键一直被按下时则不断输出同样的一帧数据。按键按下后先发送开始同步位，再发送数据，紧接再发送一个同步位，按键还在按下时则再发送一帧数据和一个同步位并不断循环，直到按键松开以一个结束同步位结束。



LR3715M

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

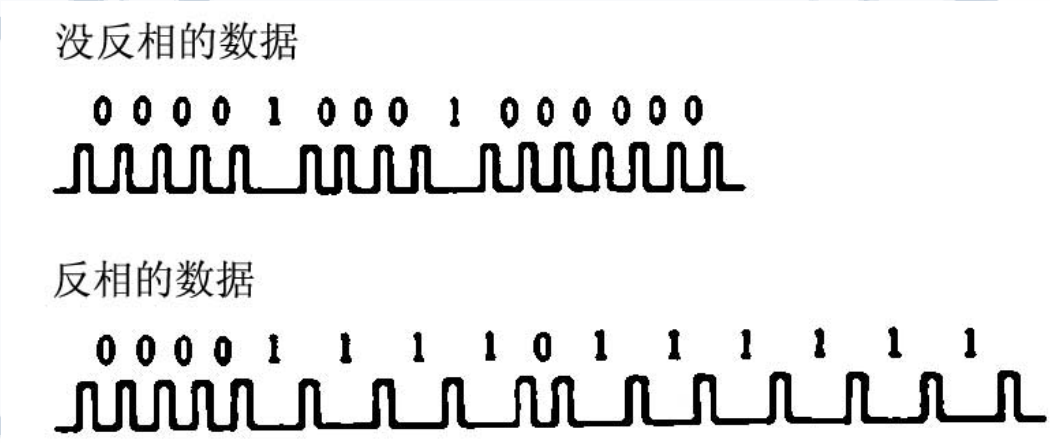
占空比 $= T1/Tc = 1/3$

位时间 $= T = 10Tc = 0.264ms$

数据格式

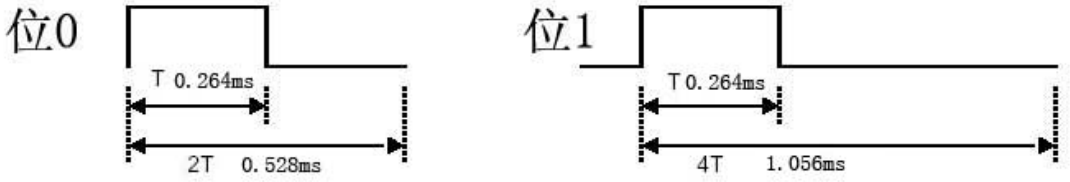
LR3715M的数据格式包括有5位的系统码、6位的数据码和4位扩展数据码，共占15位。其中最后一位C14是反相判断位。当C14等于0时说明这一帧数据是没有反相的数据，当C14等于1时则说明这一帧数据是反相的数据。解码时就可以根据C14的值判断当前数据是否是反相数据。要注意的是在反相数据帧中只有数据码和扩展码是反相的。

数据反相



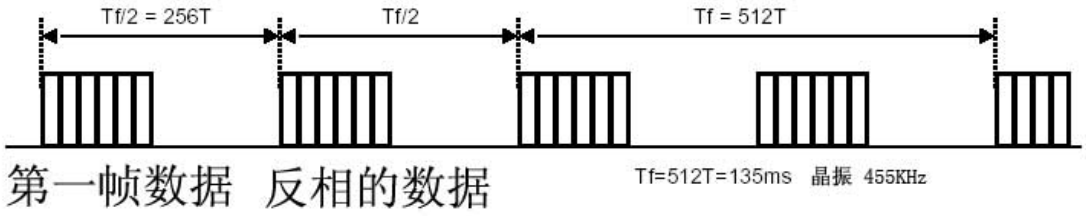
位定义

LR3715M 的位定义在解码时可以根据高脉冲后的低脉冲宽度来进行判断。



按键输出波形

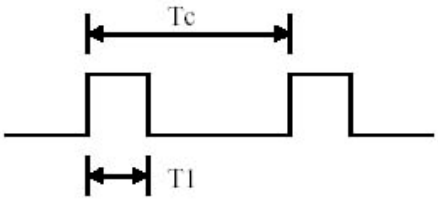
遥控的按键按下后芯片先发送一帧数据，256T时间后，再发送一帧反相的数据，按键一直被按下时则一直反复发送数据和反相数据直到按键松开。



Zenith CG1

载波波形

使用 480KHz 晶体，经内部分频电路，信号被调制在 40KHz，占空比为 3 分之 1。



调制频率（晶振使用480KHz时）

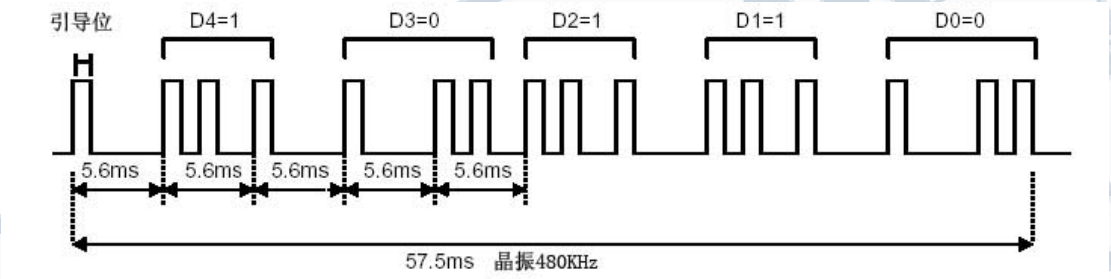
$f_{CAR} = 1/Tc = f_{OSC}/12 = 40KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

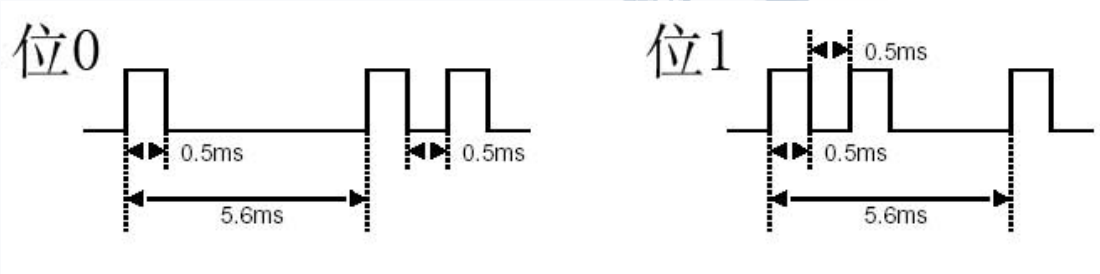
数据格式.

Zenith 的 CG1 编码格式每一帧数据包含了一个引导位和 4 个数据位。每一位数据位则由数据位和数据位的反相位组成，也就是一帧数据要占用 9 个位的时间，引导位是单位，数据位是双位。如下图所示的 D4 位就是由逻辑位 1 和逻辑位 0 所表示，在解码时就可以根据这个规则判断接收到的数据是否正确。



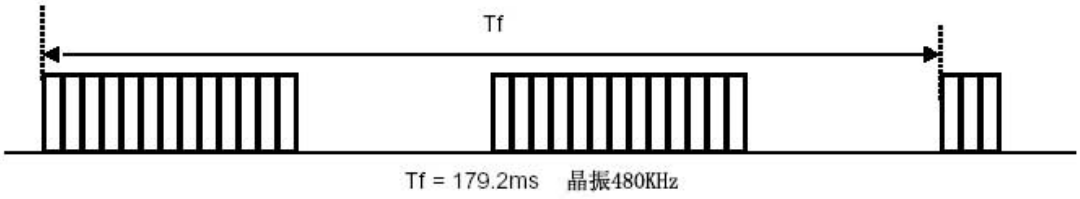
位定义

每一个逻辑位占 5.6ms，而数据位则还有相反位，数据位每位就占 11.2ms。



按键输出波形

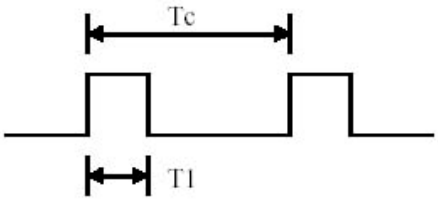
按键按下后输出二帧数据，周期Tf为179.2ms，重复输出直到按键松开。



Zenith CG2

载波波形

使用 480KHz 晶体，经内部分频电路，信号被调制在 40KHz，占空比为 3 分之 1。



调制频率（晶振使用480KHz时）

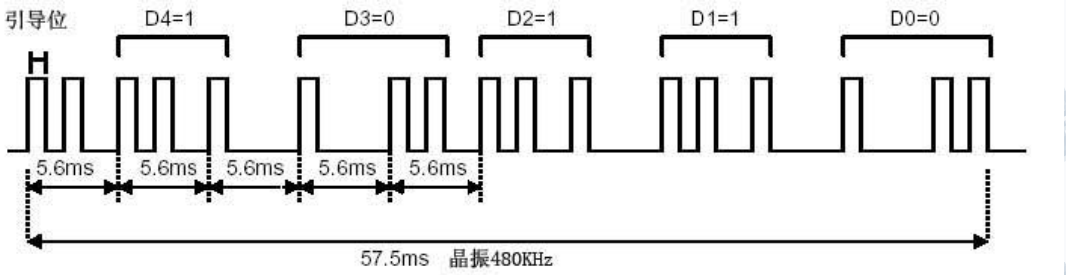
$f_{CAR} = 1/Tc = f_{OSC}/12 = 40KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

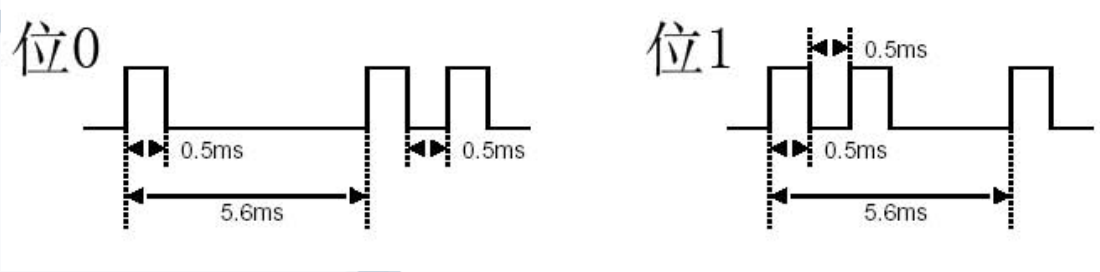
数据格式.

Zenith 的 CG2 编码格式和 CG1 格式是基本相同的，每一帧数据包含了一个引导位和 4 个数据位，所不同的是 CG1 的引导位是逻辑 0，CG2 则是逻辑 1。



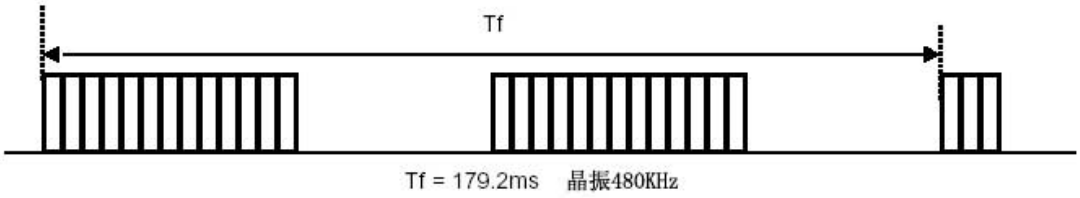
位定义

每一个逻辑位占 5.6ms，而数据位则还有相反位，数据位每位就占 11.2ms。



按键输出波形

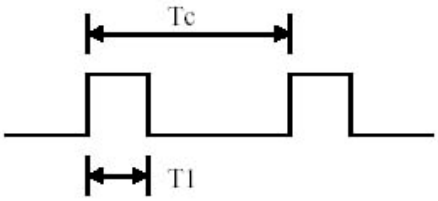
按键按下后输出二帧数据，周期Tf为179.2ms，重复输出直到按键松开。



SONY D7C6

载波波形

使用 480KHz 晶体，经内部分频电路，信号被调制在 40KHz，占空比为 3 分之 1。



调制频率（晶振使用480KHz时）

fCAR = 1/Tc = fOSC/12 = 40KHz

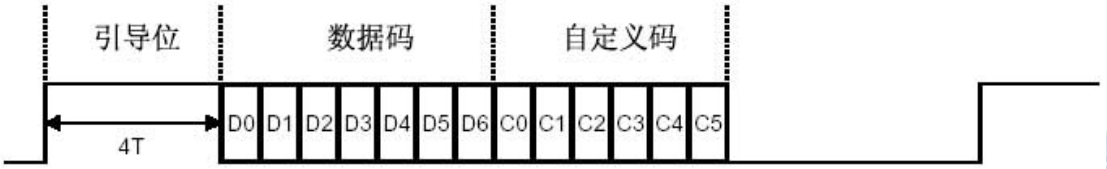
fOSC是晶振频率

占空比 = T1/Tc = 1/3

位时间 = T = 24Tc = 0.6ms

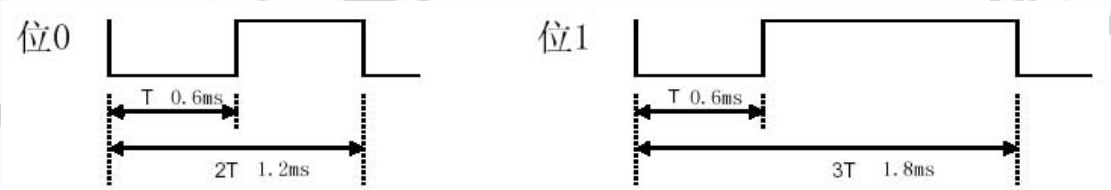
数据格式.

SONY 的 D7C6 编码格式，在经过 4T 的引导时间后，输出 7 位数据码和 6 位自定义码。



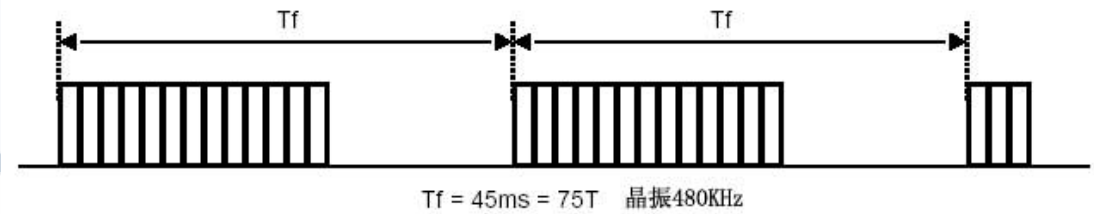
位定义

下图可以看出D7C6编码格式的位定义是用高电平的宽库还区分，逻辑位1的宽度要比逻辑位0多出一个T周期。



按键输出波形

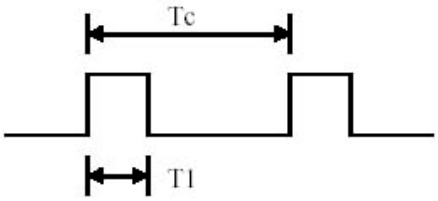
按键按下后输出一帧数据，周期Tf为45ms，重复输出直到按键松开。



SONY D7C8

载波波形

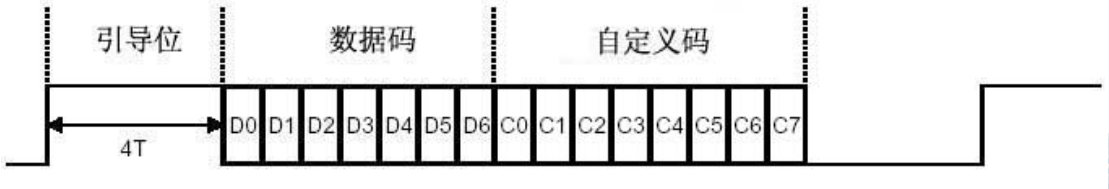
使用 480KHz 晶体，经内部分频电路，信号被调制在 40KHz，占空比为 3 分之 1。



调制频率（晶振使用480KHz时）
fCAR = 1/Tc = fOSC/12 = 40KHz
fOSC是晶振频率
占空比 = T1/Tc = 1/3
位时间 = T = 24Tc = 0.6ms

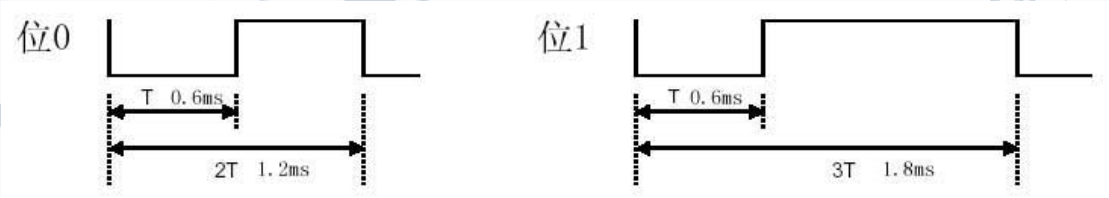
数据格式.

SONY 的 D7C8 编码格式，在经过 4T 的引导时间后，输出 7 位数据码和 8 位自定义码。



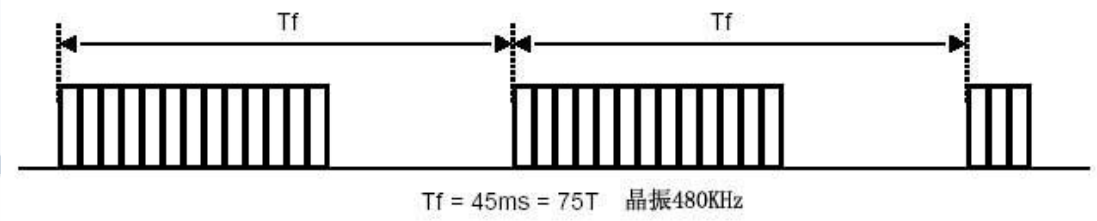
位定义

下图可以看出D7C8编码格式的位定义是用高电平的宽度来区分，逻辑位1的宽度要比逻辑位0多出一个T周期。



按键输出波形

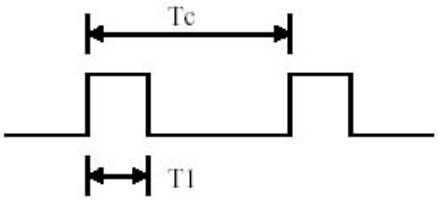
按键按下后输出一帧数据，周期Tf为45ms，重复输出直到按键松开。



MN6014 C5D6

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

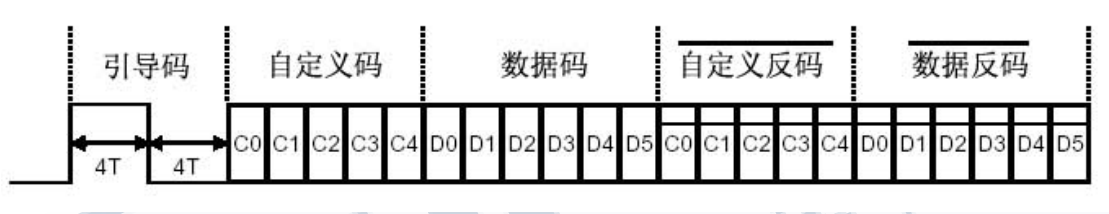
f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

位时间 $= T = 32Tc = 0.844ms$

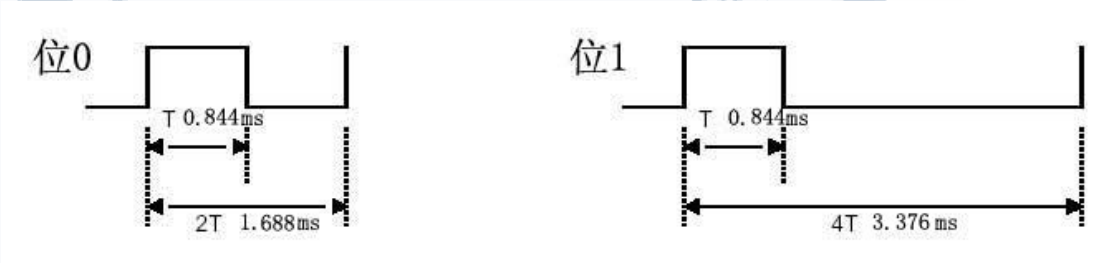
数据格式.

MN6013 芯片的 C5D6 编码格式每帧数据由引导码、5 位自定义码、6 位数据码、5 位自定义反码和 6 位数据码反码组成。解码时可以根据引导码来判断数据帧是否开始被接收，反码则可以和自定码数据码作比较来判断当前数据帧是否被正确接收。



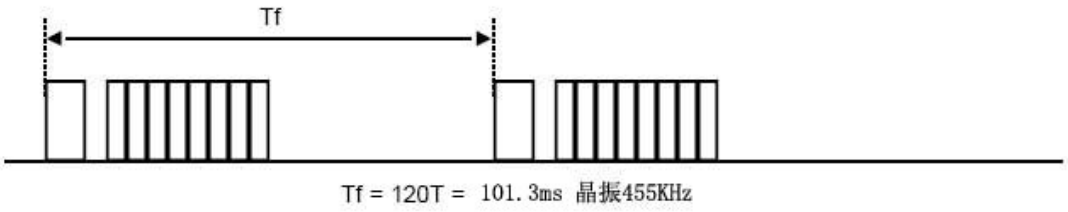
位定义

下图可以看出 C5D6 编码格式的位定义是用低电平的宽度来区分，逻辑位 1 的低电平宽度要比逻辑位 0 多出二个 T 周期。编码方式为 PPM。



按键输出波形

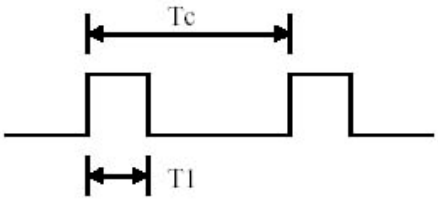
按键按下后输出一帧数据，周期Tf为101.3ms，重复输出直到按键松开。



MN6014 C6D6

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

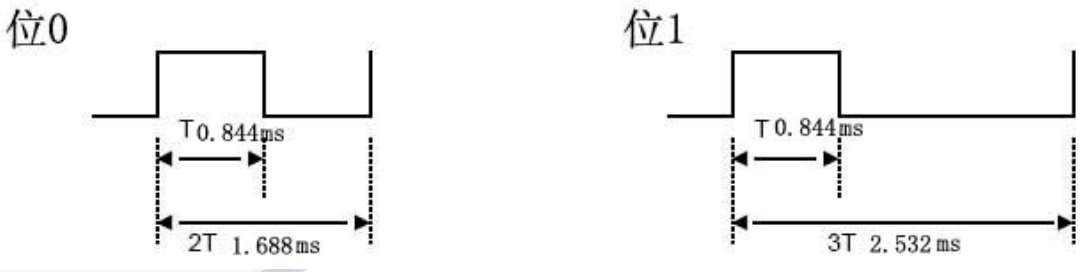
位时间 $= T = 32Tc = 0.844ms$

数据格式.

MN6013 芯片的 C6D6 编码格式每帧数据由引导码、6 位自定义码、6 位数据码、6 位自定义反码和 6 位数据码反码组成。解码时可以根据引导码来判断数据帧是否开始被接收，反码则可以和自定码数据码作比较来判断当前数据帧是否被正确接收。

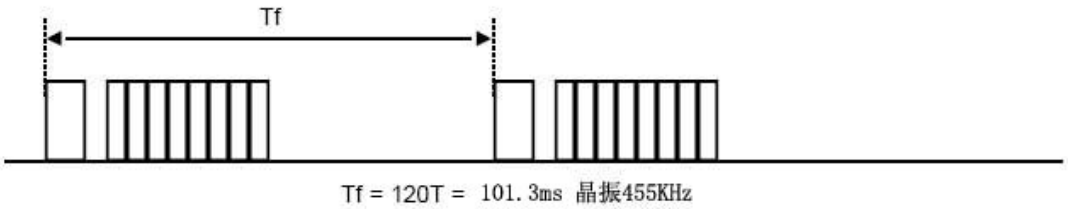
位定义

下图可以看出 C6D6 编码格式的位定义是用低电平的宽度来区分，逻辑位 1 的低电平宽度要比逻辑位 0 多出一个 T 周期。编码方式为 PPM。



按键输出波形

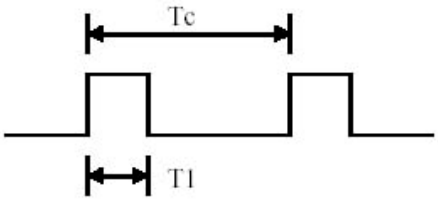
按键按下后输出一帧数据，周期Tf为120T，重复输出直到按键松开。



LC7464、AEHA

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

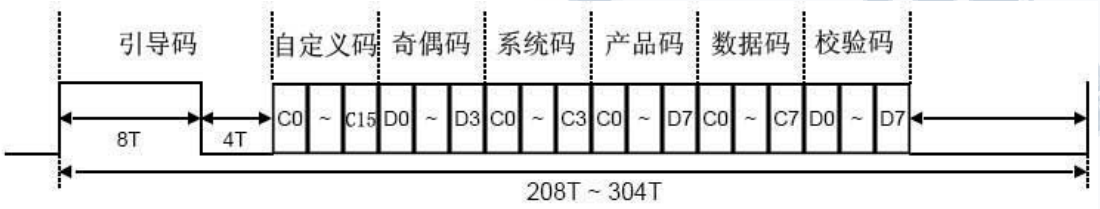
f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

位时间 $= T = 16Tc = 0.422ms$

数据格式.

LC7464 编码数据格式在 12T 的引导码后输出 16 位自定义码，4 位奇偶码，4 位系统码，8 位产品码，8 位数据码，8 位校验码。在解码时可以用校验码来校验数据帧是否正确被接收。



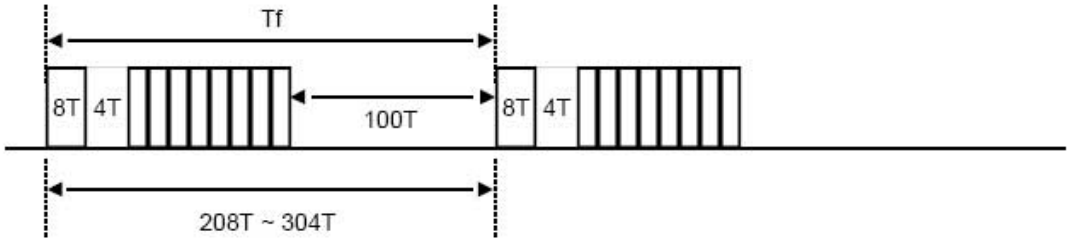
位定义

下图可以看出 AEHA 的编码格式的位定义是用低电平的宽度来区分，逻辑位 1 的低电平宽度要比逻辑位 0 多出二个 T 周期。编码方式为 PPM。

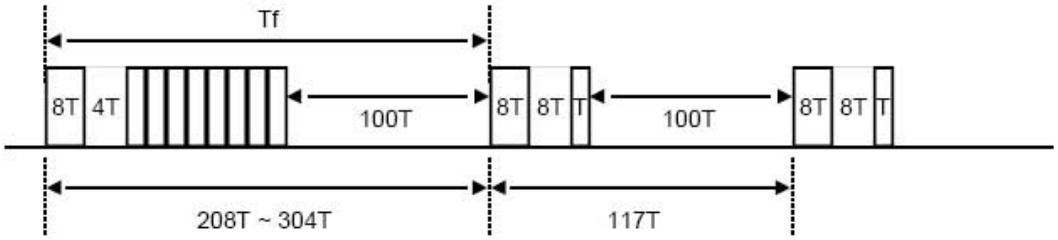
按键输出波形

LC7464 按键输出有二种方式：一种是每次按键都输出完整的二帧数据；另一种是按下相同的按键后每发送完整的一帧数据后，再发送重复码，再到按键被松开。

单一按键波形



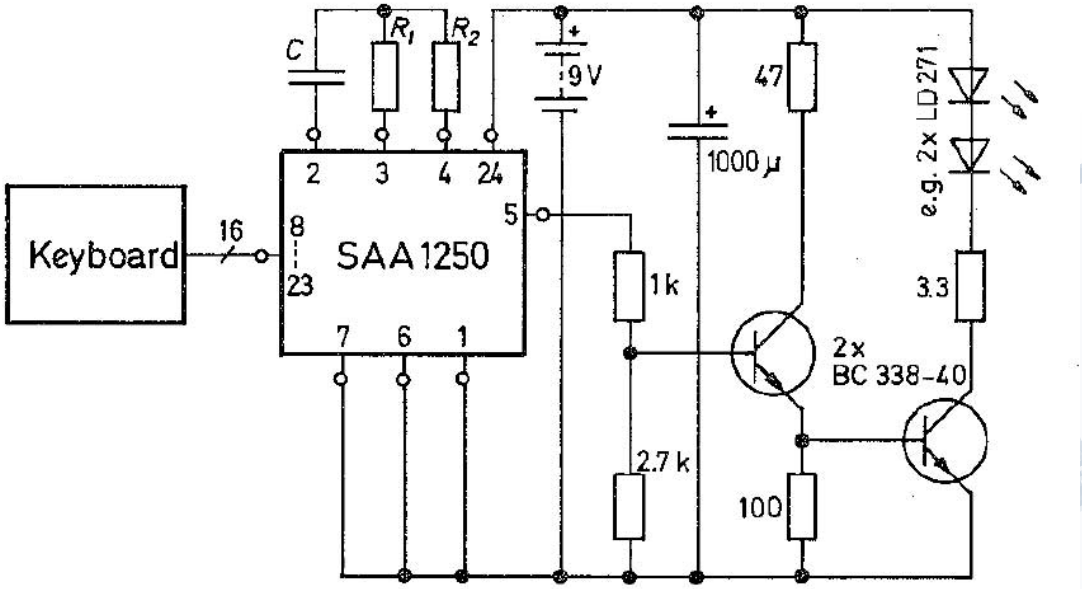
连续按键波形



IRT1250、IRT1260、SAA1250、SAA1260

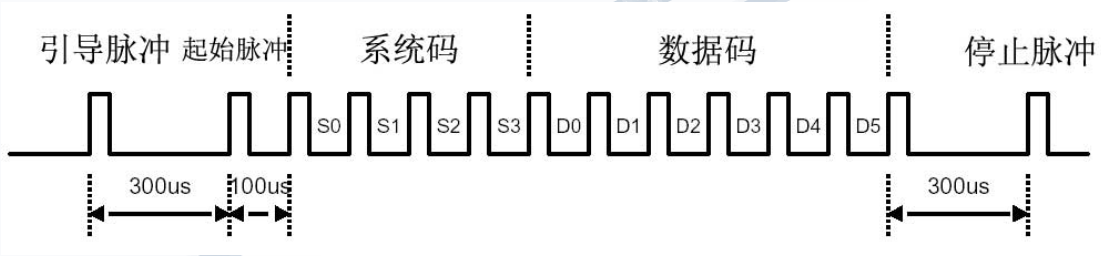
这几款芯片所用的编码格式是一样的，但把支持的电源和频率有所不同，这里只用典型的SAA1250说明。

SAA1250的所使用频率是160－220KHz，输出信号无载波，电压为6－9V，典型用应用电路如下。



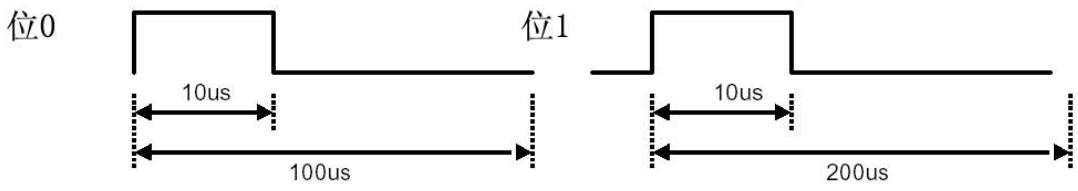
数据格式.

典型的T周期为100us，经过3个T周期的引导脉冲后，是一个起始脉冲说明编码开始传送。编码包括4位系统码和6位数据码，最后经过3个T周期的停止脉冲结束一帧数据的输出。



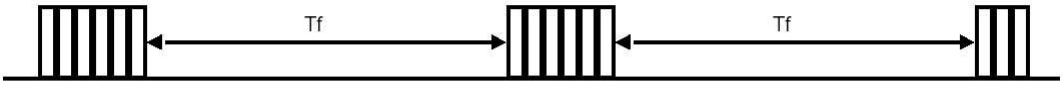
位定义

位 1 和位 0 的高电平部分都是一个 10us 的脉冲，位 1 比位 0 的宽度多一个 T 周期时间。



按键输出波形

按键按下后输出一帧数据，延时 Tf 周期后再输出另一帧数据，重复输出直到按键松开。

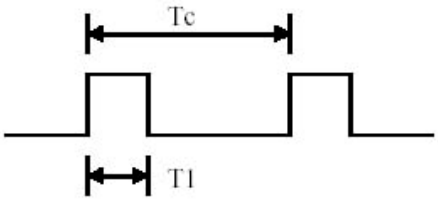


Tf=140ms

TC9148、HS9148、SC9148、BL9148

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

fCAR = 1/Tc = fOSC/12 ≈ 38KHz

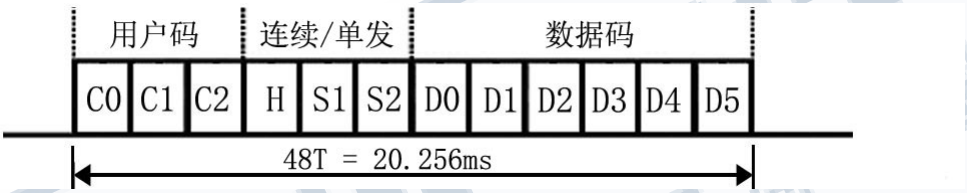
fOSC是晶振频率

占空比 = T1/Tc = 1/3

位时间 = T =16Tc = 0.422ms

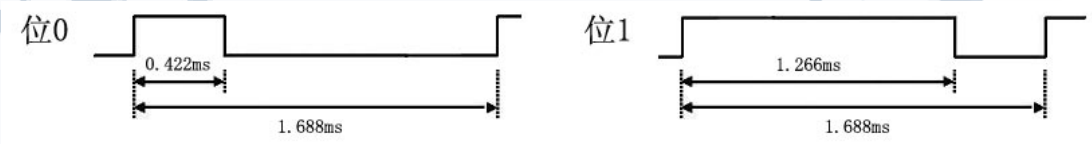
数据格式.

数据格式包括了用户码、单发/连续标识，数据码，编码共占 12 位。用户码可以标识不同的系统，H、S1、S2 分别标识连续和单发码。数据码则标识键盘值。



位定义

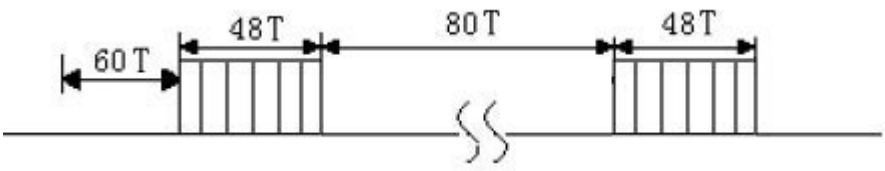
用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。位 0 的脉冲宽度是(1/focs)*192,秒=0.422ms，位 1 则是它的三倍。



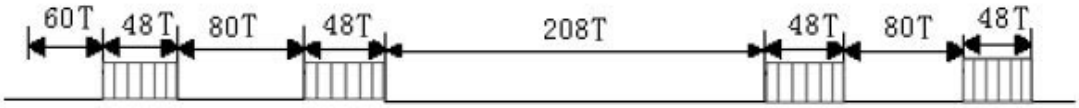
按键输出波形

TC9148 的应用电路可以设置使用单发码和连发码。单发码的按键在按下去后经过 60T 的防抖动时间，传送 2 帧数据后停止传送，要再发送则需先松开按键，再次按下。连续码则在按键按下后不断的按周期重复传送数据，直到松开按键。

单发码



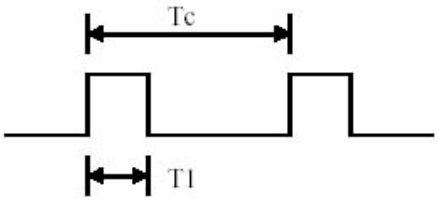
连续码



M50462、HS50462、SC50462、AS50462

载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。



调制频率（晶振使用455KHz时）

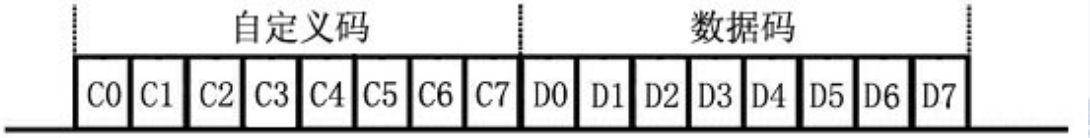
$f_{CAR} = 1/Tc = f_{OSC}/12 \approx 38KHz$

f_{OSC}是晶振频率

占空比 $= T1/Tc = 1/3$

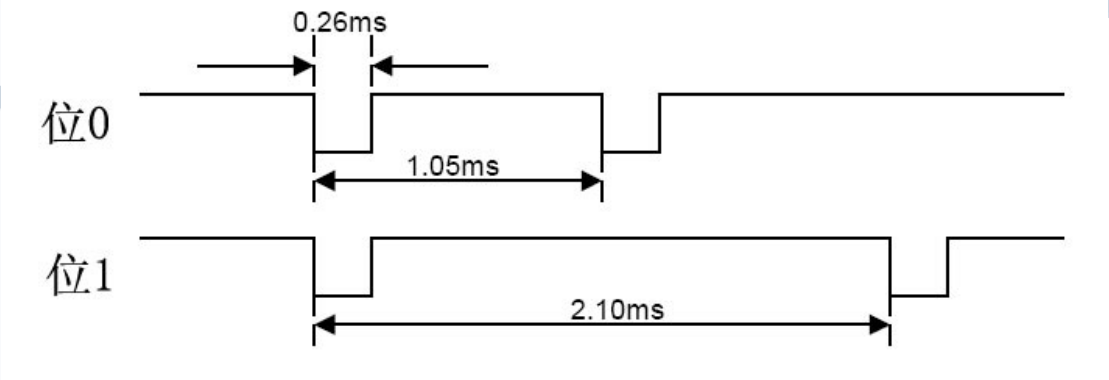
数据格式.

数据格式为每一帧数据包括 8 位自定义码和 8 位数据码，共 16 位。有些资料在数据码后还会有一个 S 停止位，不过在实际解码时可以忽略它的存在。



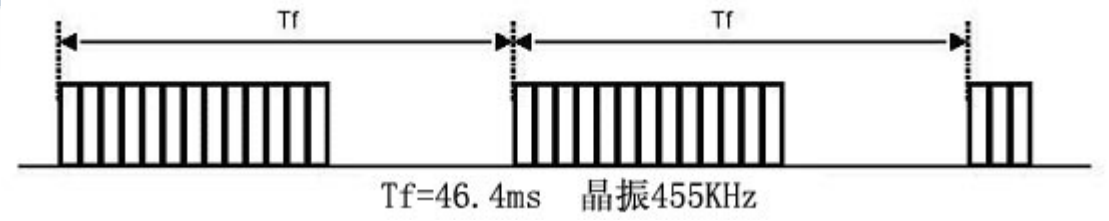
位定义

用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。位 1 的时间是位 0 的两倍。位编码方式为 PPM。



按键输出波形

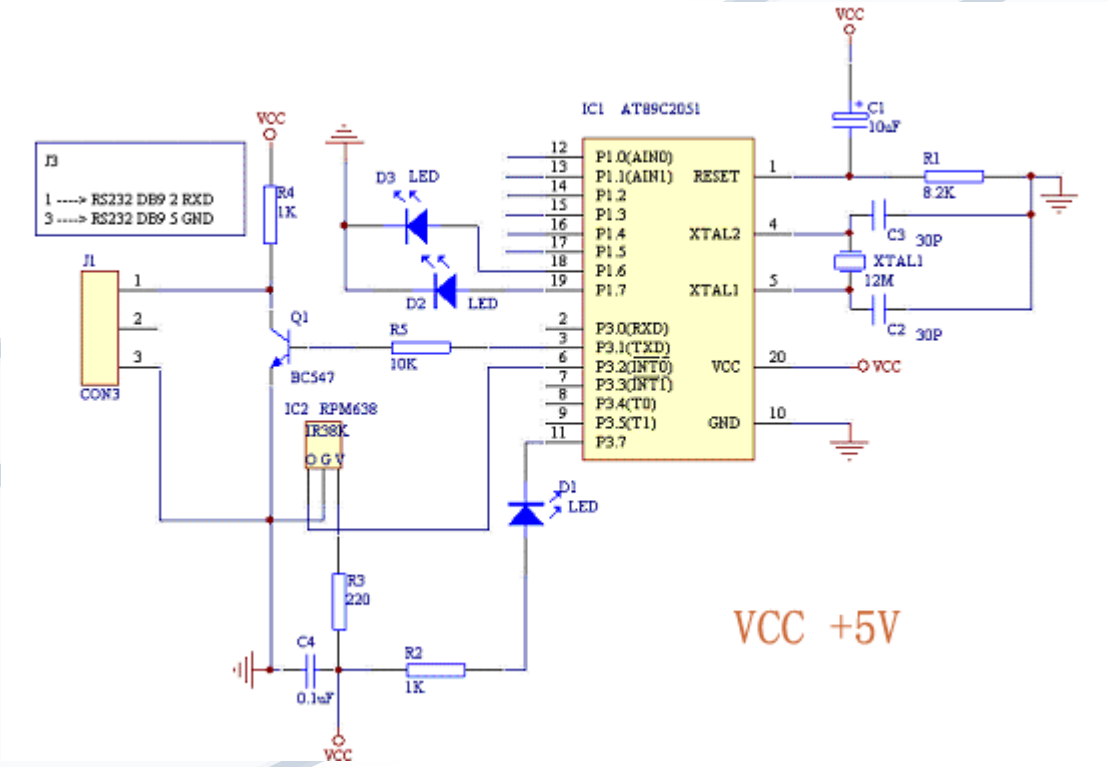
按键按下后输出一帧数据，Tf 周期后再输出另一帧数据，重复输出直到按键松开。



实例

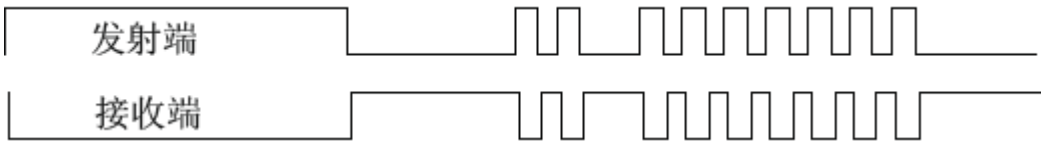
上面的资料对于刚开始了解红外遥控编码的读者来说，可能看完后还是不太清楚如何去用单片机对其进行解码。下面笔者就使用一个简单的实例去简要的说明一下笔者自己对遥控解码经验。笔者手头上正好有一个众合牌 H-105A 型兼容 M50560-001 的电视遥控器，使用芯片是 MAGIC II，这款芯片可以兼容好几个不同的芯片，这里只讨论 M50560。

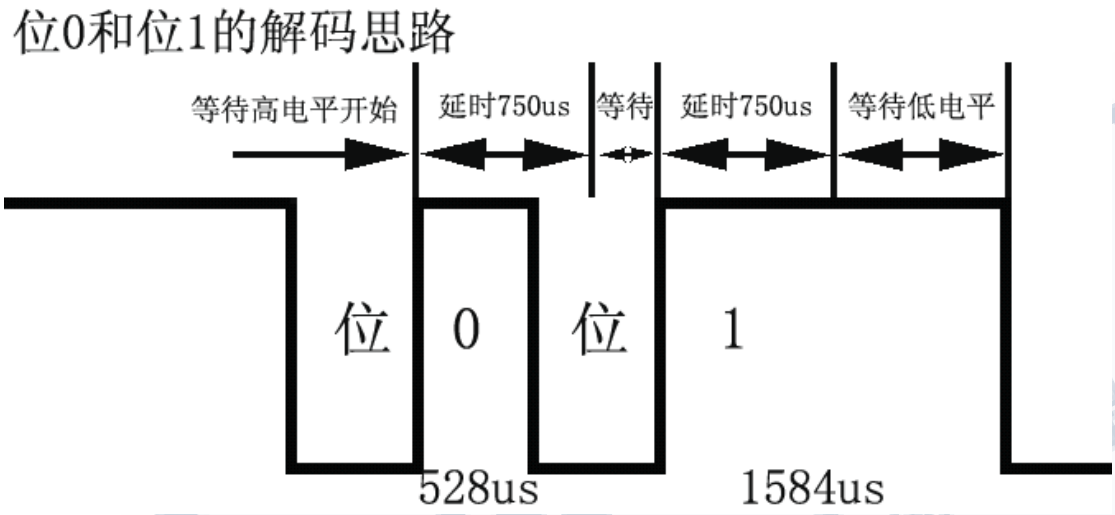
为了实验，先来制作一个实验用的电路。这里选用 AT89C2051 做 CPU，红外信号接收则使用普通的 38K 一体化接收头，它已集成了解调和放大功能。在 2051 的 IO 脚上接两个 LED 来实验遥控控制功能，控制 LED 的亮和灭。为了能更直观的看到解码后的编码值，使用上 2051 的串口输出，连接电脑后可以直观的看到解码后的编码值，以判断解码是否成功。该电路有二点要注意：1、D3 和 D2 是直接 IO 驱动的，选用亮度低的 LED 时可以加上拉电阻或使用 D1 的驱动方式提高亮度，2、R3 和 C4 是不可以少的，否则可能会因干扰过大而无法完成实验。



实验电路准备好后，就来看看如何用程序去分析位 0 和位 1。从资料上看位 0 和位 1 所不同之处就是在高电平脉冲后的低电平脉宽不一样，位 0 约为 528us，位 1 约为 1584us。这里有一点要注意的是，资料上的波形是指遥控芯片输出的波形，而一般的接收头在接收到信号时是输出低电平的，也就是说接收头输出的波形正好和遥控芯片输出的相反。下图就是其中一段引导码和自定义码发射端和接收端的波形，其中自定义码的值为 02H。在接收端位 1 的高电平宽度约为 1584us，位 0 约为 528us，程序上可以这样判断一个位的值：在位开始时接收头的引脚是低电平，等待低电结束，高电平开始后，延时 750us，读引脚的电平，高电平为位 1，低电平为位 0，当前位如果是 0 时先前延时 750us 这时已到了下一位的低电平上，可以读下一个位了，当前位如果是 1 时先前延时 750us 这时还没有结束位 1 的高电平，这时要等待下一个位的低电平才可以开始读下一位，循环 8 次就可以读完一个码。这种判断位的方法同样可以用在 uPD6121、TC9012 等位定义为 PPM 方式的芯片解码中。还有很多的其它的方法可以达到同样的效果，如触发中断后再使用定时中断进行数据的采集，无论使用何

种方法都要按位定义的规则来进行程序的编写，如果采集的数据无法准确判断每一个位，那么解码将不会有正确的结果。

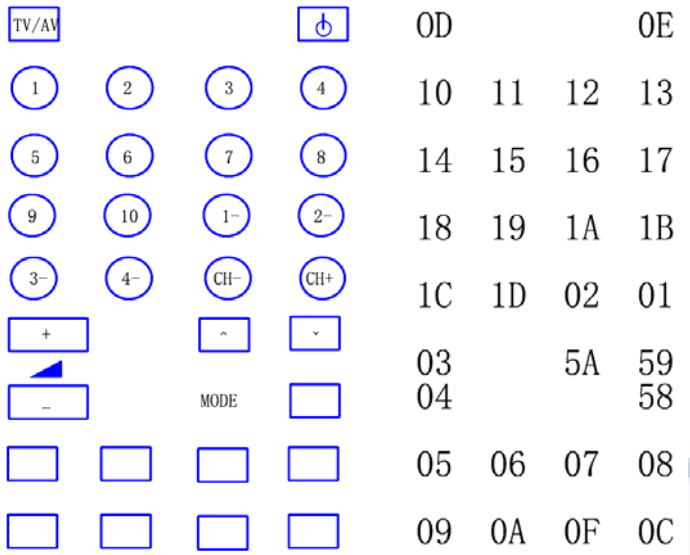




M50560 的编码中并没有输出反码，无法通过正反码相对比的方法来纠错。在抗干扰方面除了做硬件上的抗干扰，解码 M50560 时软件可以对引导码和间隔码做一定的检查来更大程度的减少误码产生。笔者的方法是当有红外信号触发 CPU 中断时，延时 7ms(这个时间不能超过 8.44ms)后判断是否输入引脚仍然是低电平,是则继续，否则退出中断。接着用同样的方法判断引导码的后半部分。在实际电路上这种方法也得到很好表现。

这个电路连接 PC 串口后，可以用串口调试软件，使用 4800 比特率来查看遥控的自定义码（系统码）和键码。程序中只定义了 1 号和 2 号键来控制 D2,D3 的亮灭，读者朋友可以自行修改加入到自己的应用电路中去。笔者查找到 MagicII 芯片相关资料，并参对了众合 H－105A 的电路，程序解出来的编码和资料说明的完全一样，说明程序是正确的。该遥控的自定码是 02H，下图是各键所对应的键值和串口调试时的截图。

当手上有一个遥控器，而自己又没有这个遥控芯片的资料时，可以用逻辑分析仪得到输出波形来对它进行分析，没有逻辑分析仪也可以按 http://www.cdle.net/wzadmin/download.asp?autoid=312 网友提供的资料自制一个简单的分析器，它的电路原理和本文的实验电路基本一样，只要更换晶振和程序就可以使用。其它一些相关的资料和制作也可以访问笔者主持的网站 http://www.cdle.net



实例程序如下

/*--

红外遥控解码

(M50560 电视遥控器)

明浩 E-mail: pnzwzw@163.com pnzwzw@cdle.net

一体化接收头输出端拉 P3.2(int0)，P1 为控制输出端。
可以扩展到 32 路或更多
输出为低电平有效
----------*/

```
#include <AT89x051.h>

void InitCom(void);
void ComOutChar(unsigned char OutData);
void DelayA(void);
void DelayB(void);

void main(void)
{
 unsigned int TempCyc;

 InitCom(); //初始化串口
 EA = 1; //允许 CPU 中断
 IT0 = 1; //INT0 下降沿有效
 EX0 = 1; //开 INT0 中断;
 ComOutChar(1);
 ComOutChar(5);
 ComOutChar(3);
 do
      {
       for (TempCyc=0; TempCyc<35000; TempCyc++)
            P3_7 = 0;
       for (TempCyc=0; TempCyc<30000; TempCyc++)
            P3_7 = 1;  //工作指示 LED
      }
 while(1);
}

//INT0 中断
void INT0Fun(void) interrupt 0 using 2
{
 unsigned char IRCode[2], IROK;
 unsigned int TempCyc, TempCycB, TempCycA;
 EX0 = 0; //外部中断 0 关闭
 IROK = 0;
 DelayA(); //延时等待引导码的前半部结束
 DelayA();
 if (!P3_2)   //检验前半部是否过早结束，防干扰
      {
```

```
    for (TempCycA=0; TempCycA<2; TempCycA++)
        {
         DelayA();
         if (P3_2)   //检验前半部是否过早结束，防干扰
             {
              for (TempCyc=0; TempCyc<300; TempCyc++)
                  if (!P3_2) break; //等待第一个位,
              if (TempCyc<300) //超时检验
                  {
                   for (TempCyc=0; TempCyc<8; TempCyc++)
                       {
                        while(!P3_2); //等待 P3_2 拉高，开始位的下部分
                        DelayB(); //这里没设超时，实际应用在多功能的设计时应设超时
                        IRCode[TempCycA] = IRCode[TempCycA]>>1;
                        if (P3_2) //当延时 750us 后 P3_2 仍为高则当前位为 1
                            {
                             IRCode[TempCycA] = IRCode[TempCycA] | 0x80;
                             for (TempCycB=0; TempCycB<100; TempCycB++)
                                 if (!P3_2) break; //等待下个位  当位 1 时高电平为 1.5ms,
                             if (TempCycB>99) //之前已延时了 750us，所以超时应大于 1.5ms-750us
                                 goto endchk; //这里最大为 1ms
                            }
                       }
                  }
              else
                  goto endchk;    //超时
             }
         else
             goto endchk;
         IROK++; //当自定码和数据码都完成时为 2
        }
    }
endchk:
 if (IROK==2)
    {
     ComOutChar(IRCode[0]);
     ComOutChar(IRCode[1]); //连接 PC 串口查看自定义码和数据码
     if (IRCode[1]==0x10) //1 号键   //只演示点亮 2 只 LED，读者可以自行扩展控制别的器件
         P1_7 = ~P1_7;
     if (IRCode[1]==0x11) //2 号键
         P1_6 = ~P1_6;
     for (TempCyc=0; TempCyc<300; TempCyc++)
         DelayA(); //延时
    }
```

```
  EX0 = 1;
}

//向串口输出一个字符（非中断方式）
void ComOutChar(unsigned char OutData)
{
  SBUF = OutData; //输出字符
  while(!TI); //空语句判断字符是否发完
  TI = 0; //清 TI
}

//串口初始化  晶振为 12M    方式 1  波特率 4800
void InitCom(void)
{
  SCON = 0x50; //串口方式 1,允许接收
  TMOD = 0x21; //定时器 1 定时方式 2,定时 0 为模式 1，16 位模式
  TCON = 0x40; //设定时器 1 开始计数
  TH1 = 0xF3; //设波特率为 4800
  TL1 = 0xF3;
  PCON = 0x80; //波特率加倍控制,SMOD 位
  RI = 0; //清收发标志
  TI = 0;
  TR1 = 1; //启动定时器
}

void    DelayA(void)
{
  unsigned int TempCyc;
  for (TempCyc=0; TempCyc<650; TempCyc++); //3.5
}

void DelayB(void)
{
  unsigned int TempCyc;
  for (TempCyc=0; TempCyc<93; TempCyc++); // 0.75ms
}
```