life.augmented

AN4099
应用笔记

使用 STM32F0xx 和 STM32F3xx 微控制器为红外遥控协议实现发送器和接收器

前言

红外辐射是指电磁频谱中微波和可见光之间的区域。

红外辐射分两部分。近红外光与可见光的波长最接近，远红外与电磁频谱的微波区域更接近。

远程控制使用的是波长较短的波。信息通过电磁能而非线路进行发送和接收。

红外技术作为一种无线通信形式具有明显的优势。现今，几乎所有音频和视频设备都可以使用红外遥控手段来控制。在接收端，接收器检测光脉冲，并对其进行处理以检索/解码它们所包含的信息。

有多种流行的红外协议标准用于通过红外光传送数据，例如 RC5、SIRC...

本应用笔记旨在提供一种通用的解决方案，以使用 STM32F0xx 和 STM32F3xx 微控制器在软件中实现 IR 发送器（远程控制器件）和接收器。针对 RC5 和 SIRC 协议提供了软件实现示例。还可以根据需要支持和提供其它协议（有关更多信息，请联系意法半导体当地销售办事处）。

注： *本文档中描述的红外发送器和接收器解决方案使用 C 语言实现，并在 STM320518-EVAL (Config2)、STM32373C-EVAL 和 STM32303C-EVAL 演示包中提供，这些演示包可在 http://www.st.com 下载。*

表 1. 适用的产品

类型	适用的产品
微控制器	STM32F0xx STM32F3xx

目录

1 红外协议规范 6

1.1 RC5 协议基础 6

1.2 SIRC 协议基础 8

2 红外发送器 10

2.1 硬件注意事项 10

2.2 IR 发送器：通用解决方案 11

2.2.1 RC5 编码器解决方案 14

2.2.2 如何使用 RC5 编码器驱动程序 15

2.2.3 SIRC 编码器解决方案 16

2.2.4 如何使用 SIRC 编码器驱动程序 17

3 红外接收器 18

3.1 硬件注意事项 18

3.2 通用解决方案：使用配置为 PWM 输入模式的通用定时器的软件实现 . . . 18

3.3 RC5 协议解决方案 20

3.3.1 RC5 帧解码机制 20

3.3.2 RC5 解码库 22

3.3.3 如何使用 RC5 解码器驱动程序 23

3.4 SIRC 红外控制解决方案 25

3.4.1 软件实现 25

3.4.2 SIRC 库 27

3.4.3 如何使用 SIRC 解码器驱动程序 27

4 接口层 30

4.1 演示程序 30

4.1.1 使用 IRTIM 的发送器演示 30

4.1.2 使用配置为 PWM 模式的通用定时器的接收器演示 31

4.2 如何自定义 IR 驱动程序 32

4.2.1 IR 接收器驱动程序 32

4.2.2 IR 发送器驱动程序 34

5 结论 36

6 版本历史 37

表格索引

表 1. 适用的产品 1
表 2. RC5 时序 7
表 3. SIRC 时序 9
表 4. 实现示例 24
表 5. 实现示例 28
表 6. 红外协议参数相关的头文件定义列表 33
表 7. 文档版本历史 37

图片索引

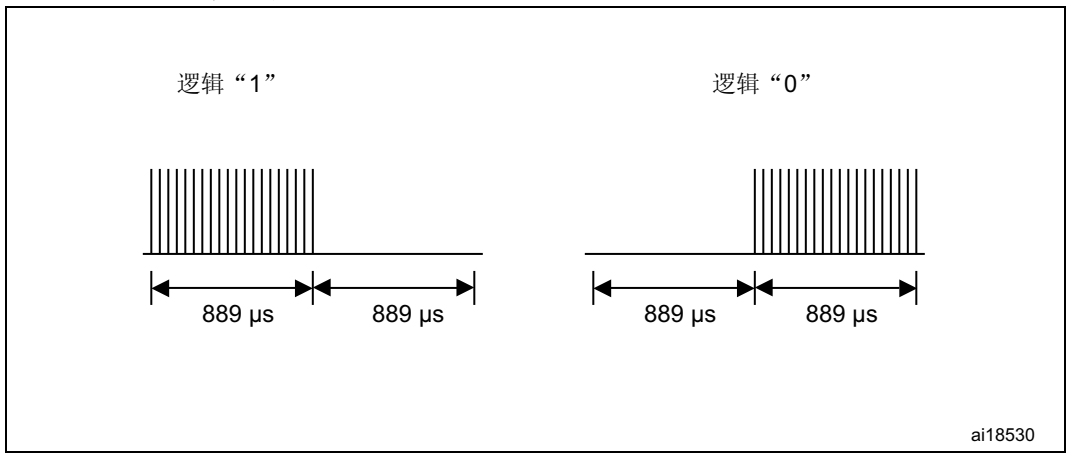
图 1. RC5 位表示 6
图 2. RC5 帧示例 7
图 3. RC5 空闲时间 7
图 4. 逻辑位的长度 8
图 5. 起始位的长度 8
图 6. SIRC 帧示例 9
图 7. 红外发送器的硬件配置 10
图 8. 硬件说明 11
图 9. 主循环流程图 12
图 10. 发送 IR 帧流程图 13
图 11. RC5 发送帧流程图 14
图 12. 曼彻斯特编码位 14
图 13. SIRC 发送帧流程图 16
图 14. SIRC 逻辑位转换 16
图 15. 硬件配置 18
图 16. 红外解码流程图 19
图 17. RC5 帧解码机制 20
图 18. 根据上升沿确定位：低脉冲 21
图 19. 根据下降沿确定位：高脉冲 21
图 20. RC5 解决方案流程图 22
图 21. SIRC 帧接收机制 25
图 22. SIRC 解决方案流程图 26
图 23. 应用层体系结构 30
图 24. IR 发送器演示 31
图 25. LCD 中显示的 RC5 接收的帧 31

1 红外协议规范

1.1 RC5 协议基础

RC5 编码是一个 14 位字，该字使用 36 kHz IR 载波频率的双相调制（也称为曼彻斯特编码）。所有位的长度均相等，为 1.778 ms，其中位时间的一半填充 36 kHz 载波脉冲，另一半空闲。逻辑 0 由位时间中的前一半脉冲串来表示。逻辑 1 由位时间中的后一半脉冲串来表示。36 kHz 载波频率的占空比为 33% 或 25%，以降低功耗。

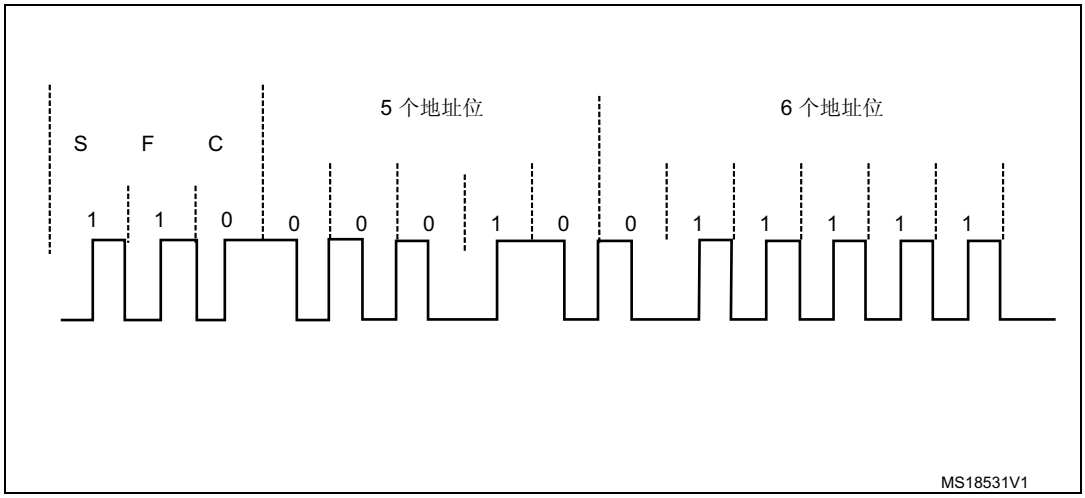
图 1. RC5 位表示



RC5 帧可以生成 2048 (32 x 64) 个不同的命令，这些命令分为 32 组。每组有 64 个不同的命令。RC5 帧包含下列字段。RC5 帧的示例如*图 2* 所示。

- **起始位 (S)：**长度为 1 位，始终为逻辑 1。
- **字段位 (F)：**长度为 1 位，表示发送的命令位于低位字段（逻辑 1 = 十进制数 0 到 63）还是高位字段（逻辑 0 = 十进制数 64 到 127）。该字段位是后来增加的，因为人们意识到每个设备 64 条命令是不够的。以前，该字段位与起始位结合在一起。许多设备仍在使用这种原始体系。
- **控制位或切换位 (C)：**长度为 1 位，每次按下按钮时切换。这使得接收设备可以区分两次连续的按钮按下操作（例如“1”、“1”代表“11”）。
- **地址：**长度为 5 位，可选择 32 种可能系统中的一种。
- **命令：**长度为 6 位（与字段位结合使用），表示 128 种可能的 RC5 命令中的一种。

图 2. RC5 帧示例



为避免帧冲突，在两个连续帧之间插入一段特定宽度的空闲时间（参见*图 3*）。

空闲时间定义为 50 位宽。因此，一帧的周期为 64 x 1 位宽：

64 x 1.778 = 113.792 ms。

图 3. RC5 空闲时间

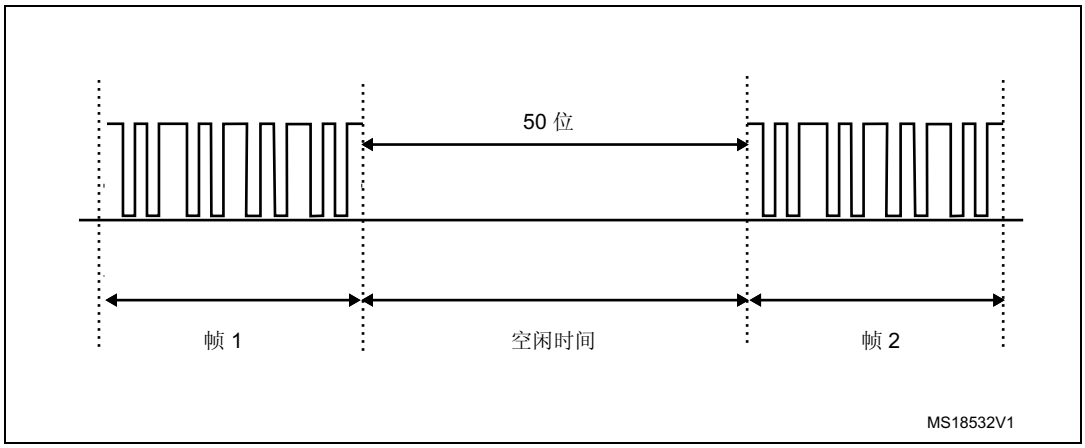


表 2. RC5 时序

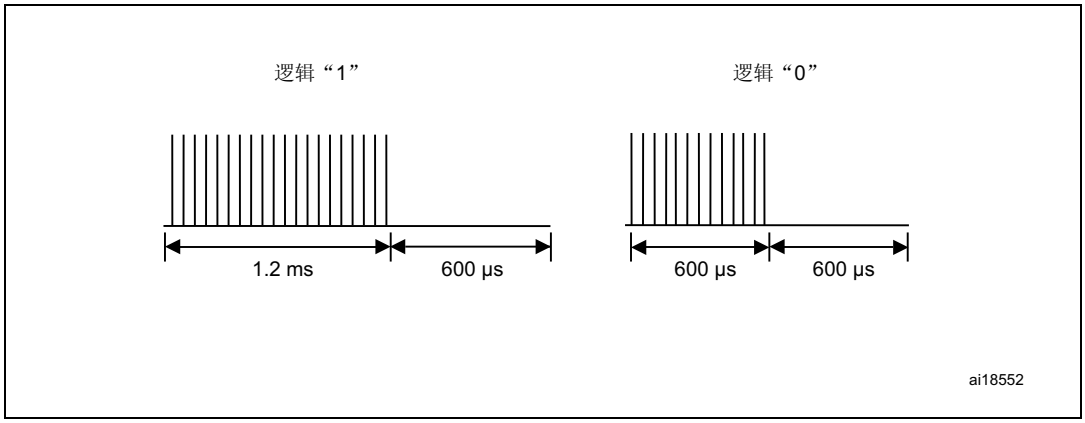
说明	最小值	典型值	最大值
RC5 半位周期	640 µs	889 µs	1140 µs
RC5 全位周期	1340 µs	1778 µs	2220 µs
RC5 消息时间	23.644 ms	24.889 µs	26.133 ms
RC5 消息重复时间	108.089 ms	113.778 ms	119.467 ms
载波脉冲位时间	27.233 µs	27.778 µs	28.349 µs

注： *红外协议的实现基于免费的 RC5 规范，该规范可从 http://www.sbprojects.com/knowledge/ir/rc5.php 下载。*

1.2 SIRC 协议基础

SIRC 代码是一个 12 位字。它使用 40 kHz IR 载波频率的调制。SIRC 协议使用脉冲串距离来对位进行编码。每个位的脉冲串都是由 600 us 长的 40 KHz 载波脉冲群组成。发送逻辑“1”需要 1.8 ms，而发送逻辑“0”需要 1.2 ms（*图 4*）。

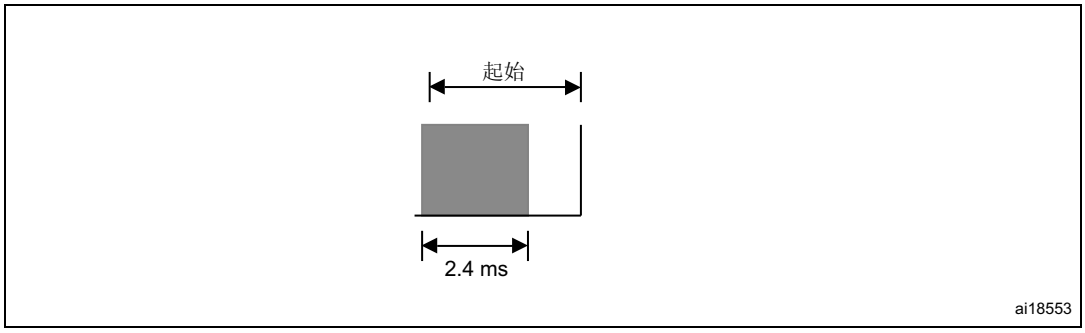
图 4. 逻辑位的长度



SIRC 帧包含下列字段。

- **起始位：**起始脉冲群始终为 2.4 ms 宽，然后是一个 0.6 ms 的标准间隔。
- **长度为 7 位的命令：**此字段有 7 位，用作命令字段。
- **长度为 5 位的地址：**此字段有 5 位，用作地址字段。

图 5. 起始位的长度

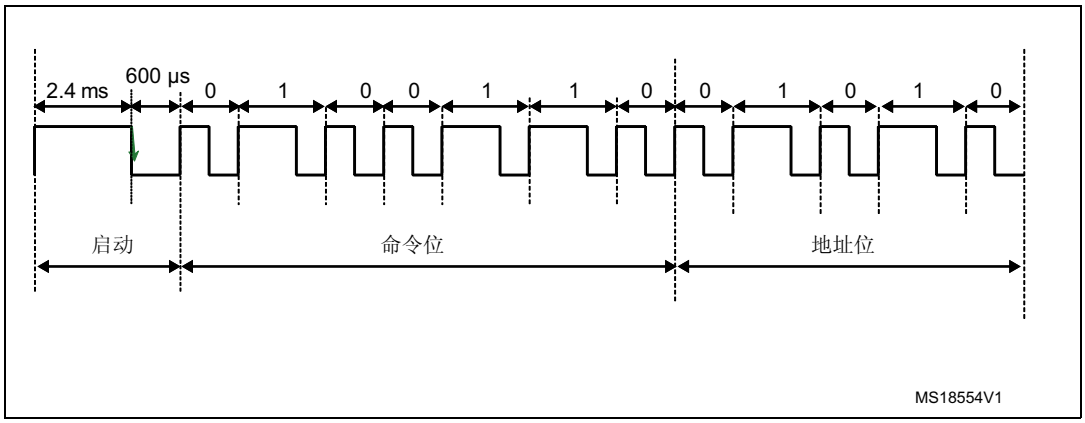


使用此协议时，先发送 LSB。因此，从 LSB 到 MSB 进行收集。由于先发送表示命令的 7 位，然后发送表示设备地址的 5 位，因此代码必须将收到的 12 位分割成 7 位和 5 位两组。

图 6 显示了 SIRC 帧的示例。

在此例中：命令为 26h (0100110b)，地址为 Ah (01010b)。

图 6. SIRC 帧示例



在两个连续帧之间插入一段空闲时间以避免冲突。每 45 ms 发送一次重复代码。

表 3. SIRC 时序

说明	典型值	最小值	最大值
同步脉冲高电平	2.4 ms	2.3 ms	2.6 ms
同步脉冲低电平	0.6 ms	0.55 ms	0.7 ms
位 0 周期	1.2 ms	1.1 ms	1.3 ms
位 1 周期	1.8 ms	1.7 ms	1.9 ms
SIRC 消息接收时间	45 ms	-	-
载波脉冲位时间	25 µs	-	-

注： *1 红外协议的实现基于免费的 SIRC 规范，该规范可从 http://www.sbprojects.com/knowledge/ir/sirc.php 下载。*

2 上表给出了本应用笔记中使用的数据脉冲宽度容差的概览。最小-最大 SIRC 时序可以由用户指定。

2 红外发送器

2.1 硬件注意事项

TX-IR LED 是用于红外串行数据链路和遥控应用的红外发送器。数据以所选的 36 kHz 或 40 kHz 载波频率进行调制，为红外数据通信和遥控应用提供简单的单芯片解决方案。

STM32F0xx 和 STM32F3xx 器件提供了用于遥控的红外接口 (IRTIM)。该接口可以与 IR LED 配合使用来执行遥控功能。

IR 数字接口可向红外二极管驱动电路输出数字信号。它可以输出任何现有调制类型的信号，调制类型取决于软件算法。

IR 接口非常容易配置，它使用两个 STM32 定时器（TIM16 和 TIM17）提供的两个信号。

TIM17 用于提供载波频率，TIM16 用于提供要发送的实际信号。

图 7. 红外发送器的硬件配置

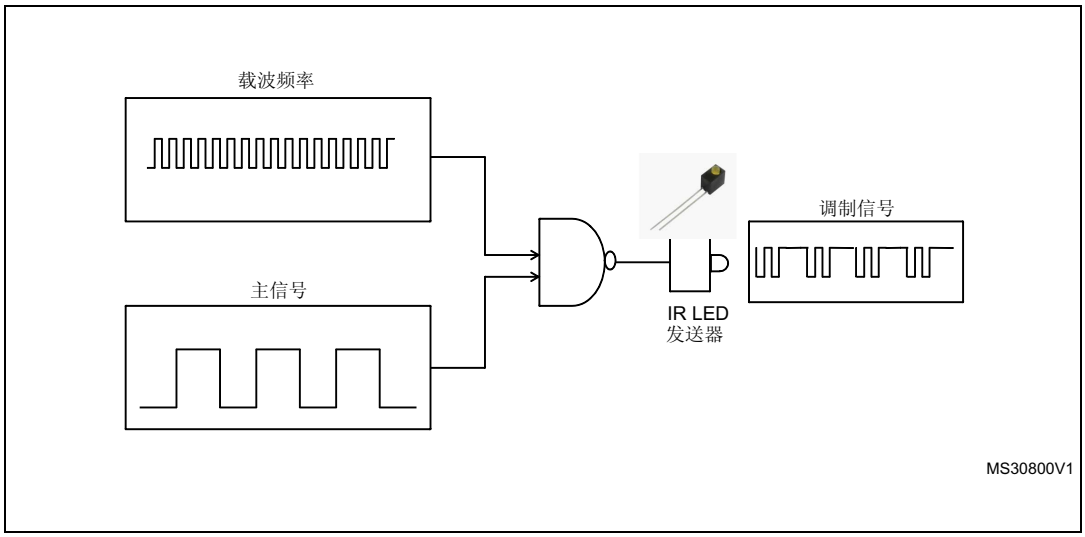
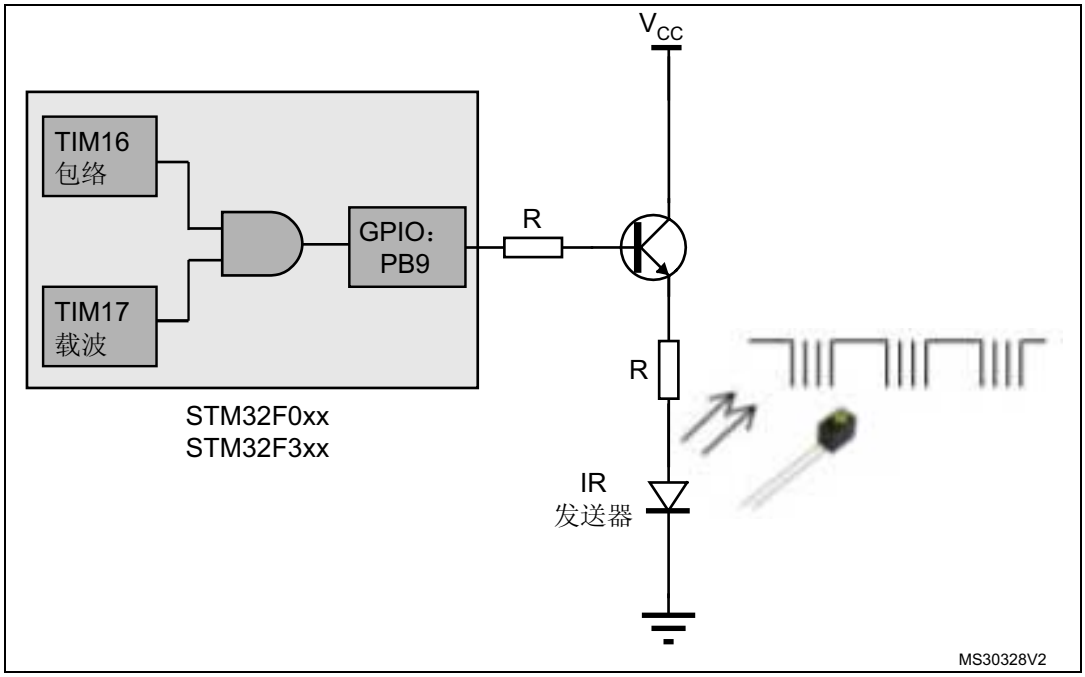


图 8. 硬件说明



2.2 IR 发送器：通用解决方案

基于 STM32 的红外发送器解决方案可将所有 RC5 和 SIRC 指令发送到所有 RC5 和 SIRC 接收器设备。

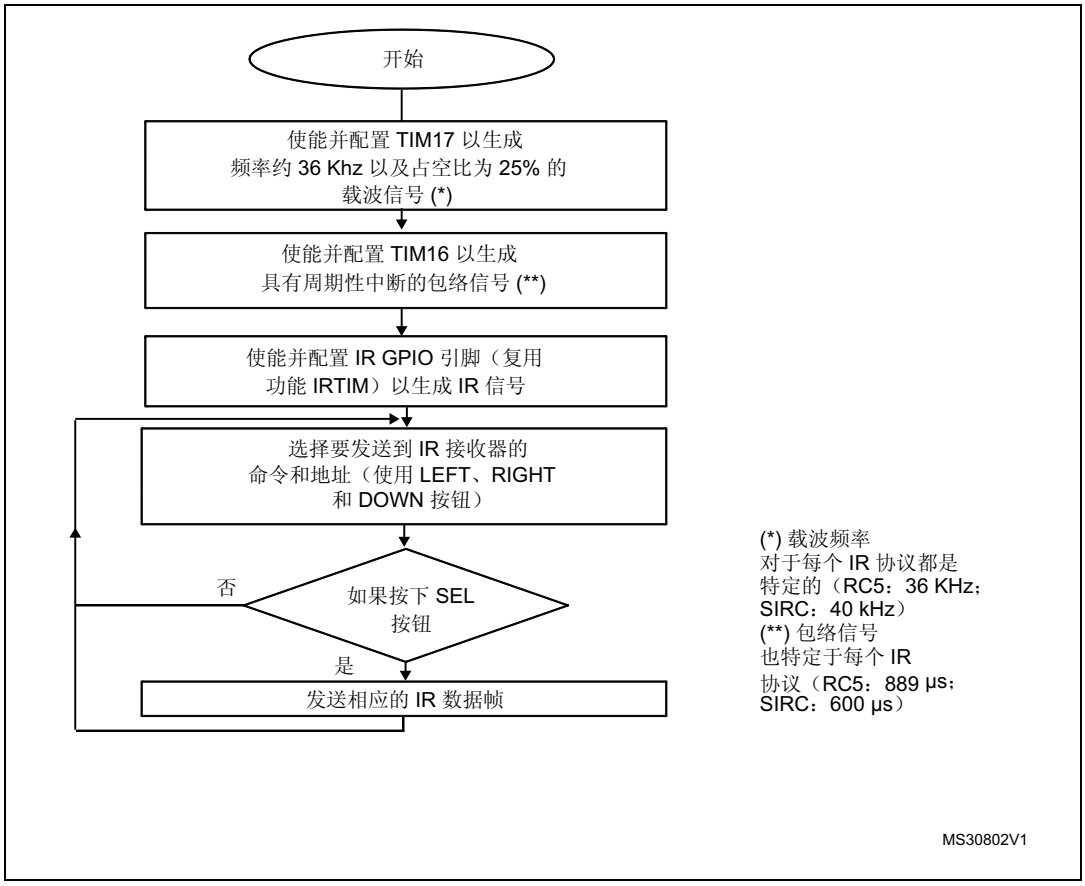
该应用解决方案使用四个外设。

- IRTIM：（带定时器的 IR 接口）使用 TIM16 和 TIM17 生成 IR 信号。
  - TIM17：(Timer17) 为 RC5 协议和 SIRC 协议分别提供频率为 36 kHz 和 40 kHz 的载波信号。
  - TIM16：(Timer16) 提供要发送的主要信号（RC5 帧或 SIRC 帧）。
- GPIO：（通用 I/O）提供与遥控按钮和 IR-LED 连接的 I/O。
- CLK：（时钟控制器）使能时钟并为定时器提供正确的时钟频率。

要生成红外遥控信号，必须正确配置 TIM16 通道 1 (TIM16_OC1) 和 TIM17 通道 1 (TIM17_OC1) 以生成正确的波形。通过对两个定时器输出比较通道编程可以获得所有标准 IR 脉冲调制模式。红外功能在 TIM_IR 引脚上输出。通过使能 GPIOx_AFRx 寄存器中的相关复用功能位（PB9 直接控制红外 LED）来激活此功能。

主程序流程如*图 9* 所示。

图 9. 主循环流程图



TIM17 的目标是生成载波信号。

TIM17_Period = (SystemCoreClock / FrequencyCarrier) - 1

TIM16 用于生成包络信号。

TIM16_Period = (SystemCoreClock / FrequencyEnvelop) - 1

模块初始化（IRTIM，帧字段）后，应用程序等待 SEL 按钮按下以发送 IR 数据。*图 10* 显示了*发送帧*流程图。

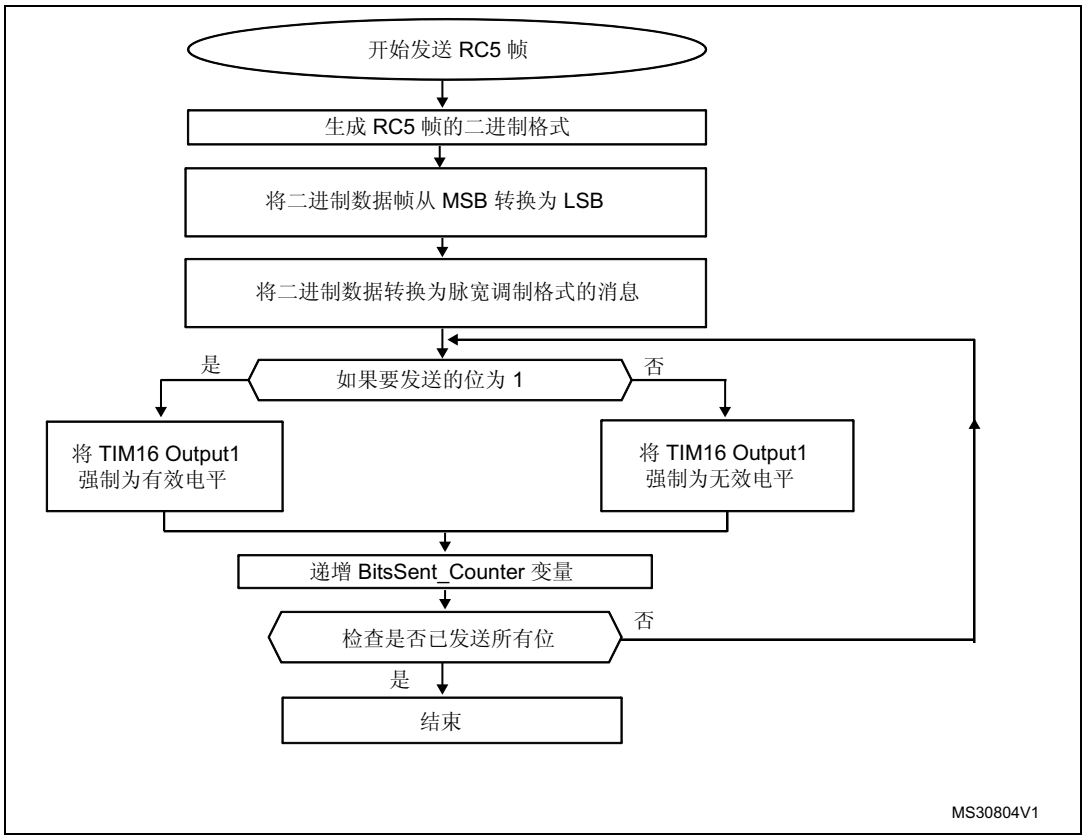
图 10. 发送 IR 帧流程图

2.2.1 RC5 编码器解决方案

RC5 编码机制

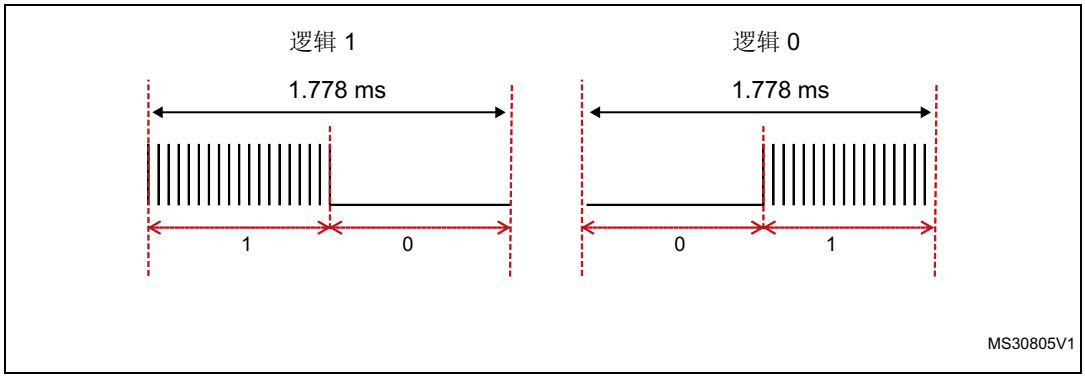
第 14 页的图 11 显示了生成 RC5 帧的方式。在 TIM16 更新中断例程期间将调用所述的流程图。

图 11. RC5 发送帧流程图



在曼彻斯特编码中，逻辑“0”由位中心的 0 到 1 转换表示，逻辑“1”由位中心的 1 到 0 转换表示。*图 12* 概括了曼彻斯特编码规则。

图 12. 曼彻斯特编码位



RC5 编码库

RC5 编码器驱动程序基于以下函数。

RC5_Encode_Init()

此函数初始化不同的外设（GPIO、定时器...）。

RC5_Encode_SendFrame()

此函数发送曼彻斯特格式的 RC5 帧。

RC5_Encode_SignalGenerate()

此函数通过监视 TIM16 的输出电平来生成帧信号。在 TIM16 更新中断期间将调用此函数，以处理输出信号。

2.2.2 如何使用 RC5 编码器驱动程序

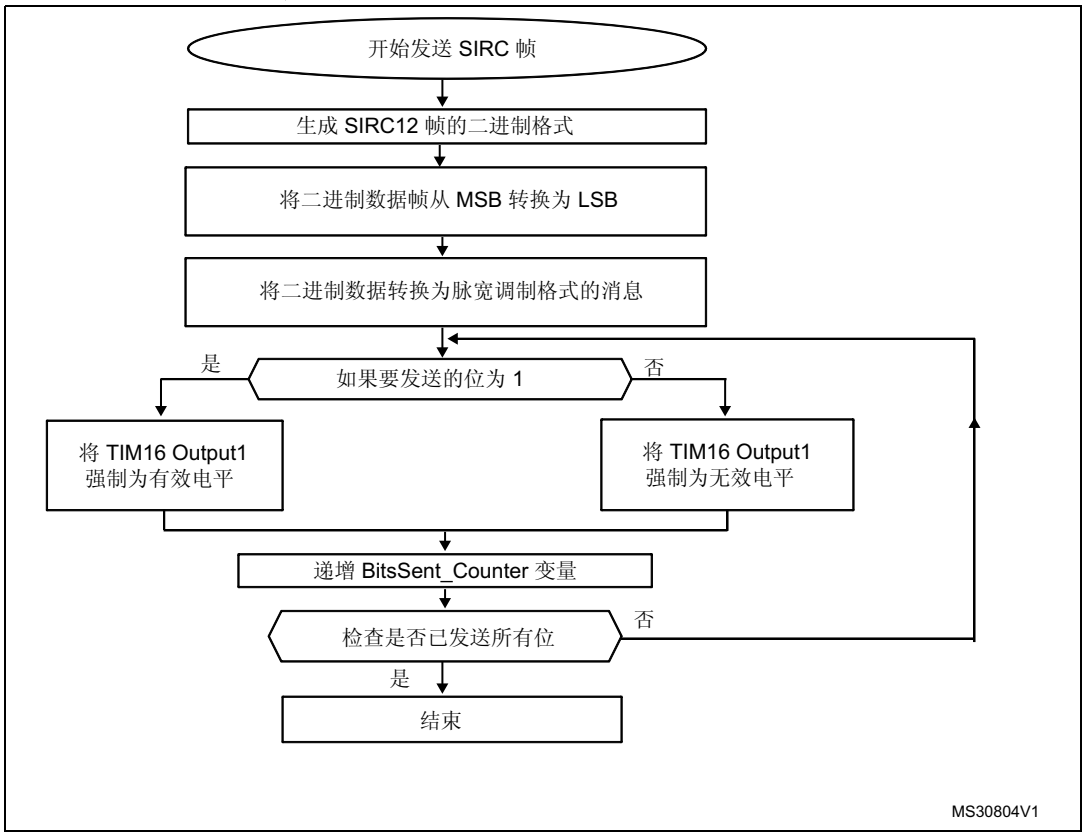
要使用 RC5 编码器驱动程序，请按以下步骤操作。

- 调用函数 RC5_Encode_Init() 以配置 RC5 编码所需的定时器和 GPIO 硬件资源。
- 调用函数 RC5_Encode_SendFrame() 以发送 RC5 帧。
- 在 TIM16 更新中断对 RC5 帧使用脉宽调制进行编码。

2.2.3 SIRC 编码器解决方案

SIRC 编码机制

图 13. SIRC 发送帧流程图

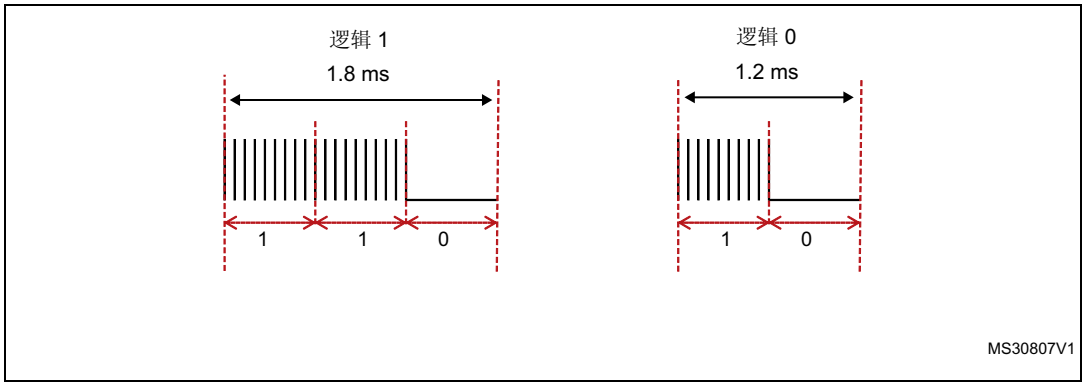


生成二进制格式的帧后，每个逻辑位都转换为表示脉宽调制格式的“0”和“1”组合。

例如，发送逻辑“1”需要 1.8 ms，其中 1.2 ms 为高电平，600 µs 为低电平。所选时基为 600 µs，因此，逻辑“1”转换为 110。

对于逻辑“0”，则需要 1.2 ms，其中 600 µs 为高电平，600 µs 为低电平。它将转换为 10（参见*图 14*）。

图 14. SIRC 逻辑位转换



SIRC 编码库

SIRC 编码器驱动程序基于以下函数。

SIRC12_Encode_Init()

此函数初始化不同的外设（GPIO、定时器、NVIC...）。

SIRC12_Encode_SendFrame()

此函数发送 SIRC12 帧格式的脉宽调制。

SIRC12_Encode_SignalGenerate()

此函数通过监视 TIM16 的输出电平来生成帧信号。在 TIM16 更新中断期间将调用此函数，以处理输出信号。

2.2.4 如何使用 SIRC 编码器驱动程序

要使用 SIRC 编码器驱动程序，请按以下步骤操作。

- 调用函数 SIRC12_Encode_Init() 以配置 SIRC 编码所需的定时器和 GPIO 硬件资源。
- 调用函数 SIRC12_Encode_SendFrame() 以发送 SIRC 帧。
- 在 TIM16 更新中断对 SIRC 帧使用脉宽调制进行编码。

3 红外接收器

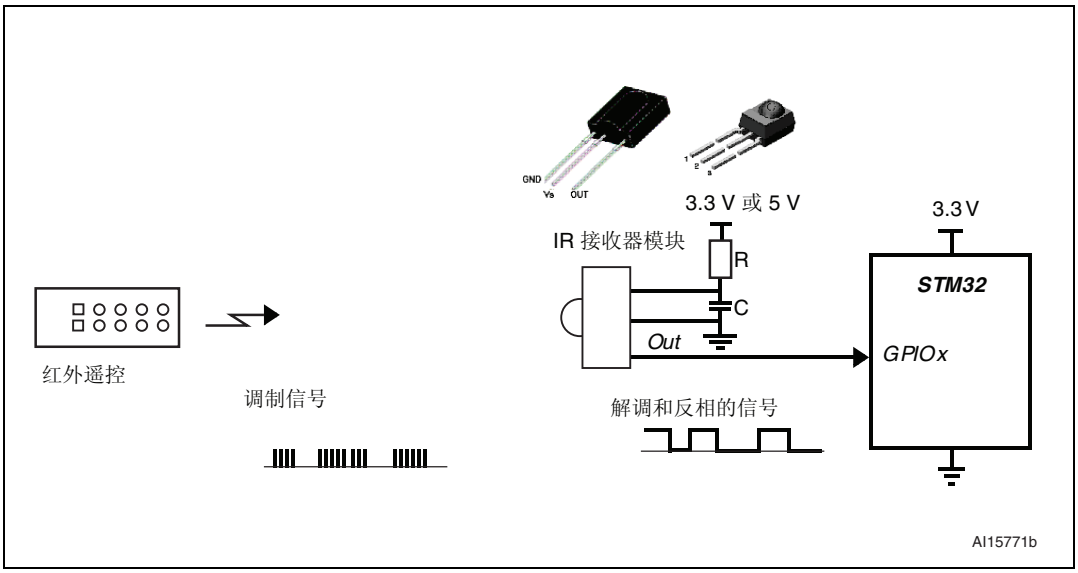
3.1 硬件注意事项

要改善噪声抑制能力，可在 36 kHz、38 kHz 或 40 kHz 附近调制 IR 脉冲。接收这些脉冲的最简单方法是使用集成的 IR 接收器/解调器模块，例如 TSOP1736（5 V 电源版）或 TSOP34836（3.3 V 电源版）或其它等效料号（参见*图 15*）。

这些器件是 3 引脚器件，可接收红外脉冲并在输出引脚上输出解调的位流，该输出引脚与 STM32 微控制器的一个 GPIO 引脚或通用定时器输入捕捉通道直接连接。如果使用 TSOP1736，所选的 GPIO 必须为 5 V 容限 (FT)。IR 模块的输出与发送的数据呈反向（数据空闲为高电平，逻辑“0”变为逻辑“1”，反之亦然）。

注： *IR 模块需要两个外部元件：一个电容和一个电阻（关于它们的值，请参见相关 IR 模块数据手册）。*

图 15. 硬件配置



3.2 通用解决方案：使用配置为 PWM 输入模式的通用定时器的软件实现

可以使用 STM32 微控制器内置的定时器外设之一来解码每个红外协议。此定时器可以配置为 PWM 输入模式并用于对红外帧位进行采样。当出现极性相反的边沿时，将激活定时器输入捕捉功能。

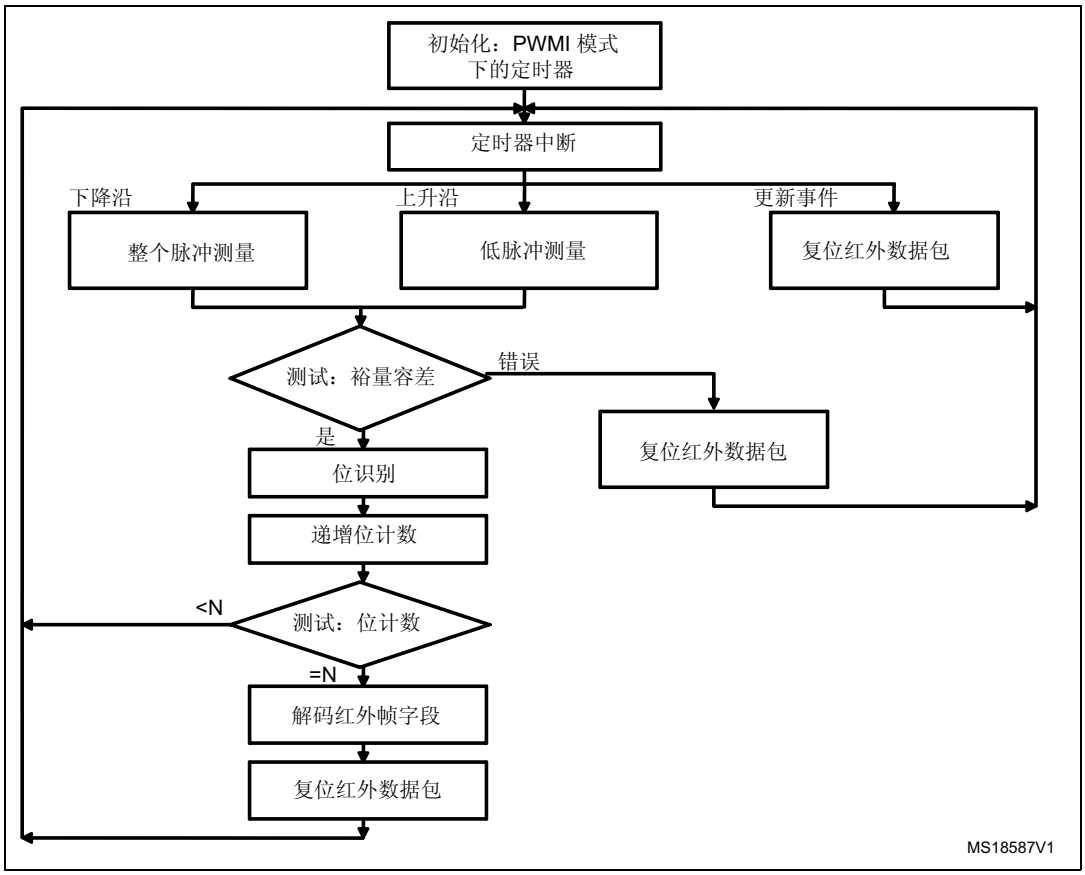
定时器生成三种类型的中断。

- 在每个下降沿生成的中断：可用于测量整个脉冲（两个连续下降沿之间的持续时间）。
- 在每个上升沿生成的中断：可用于测量低脉冲（下降沿和上升沿之间的持续时间）。
- 更新事件：用于在定时器计数器溢出时将红外数据包置于默认状态（位计数、数据和状态）。

低脉冲持续时间和整个脉冲持续时间用于确定位值。如果持续时间在位时间的容差范围内，便可以识别位值（逻辑 0、逻辑 1 或报头）。

下面的流程图概述了红外解码过程。

图 16. 红外解码流程图



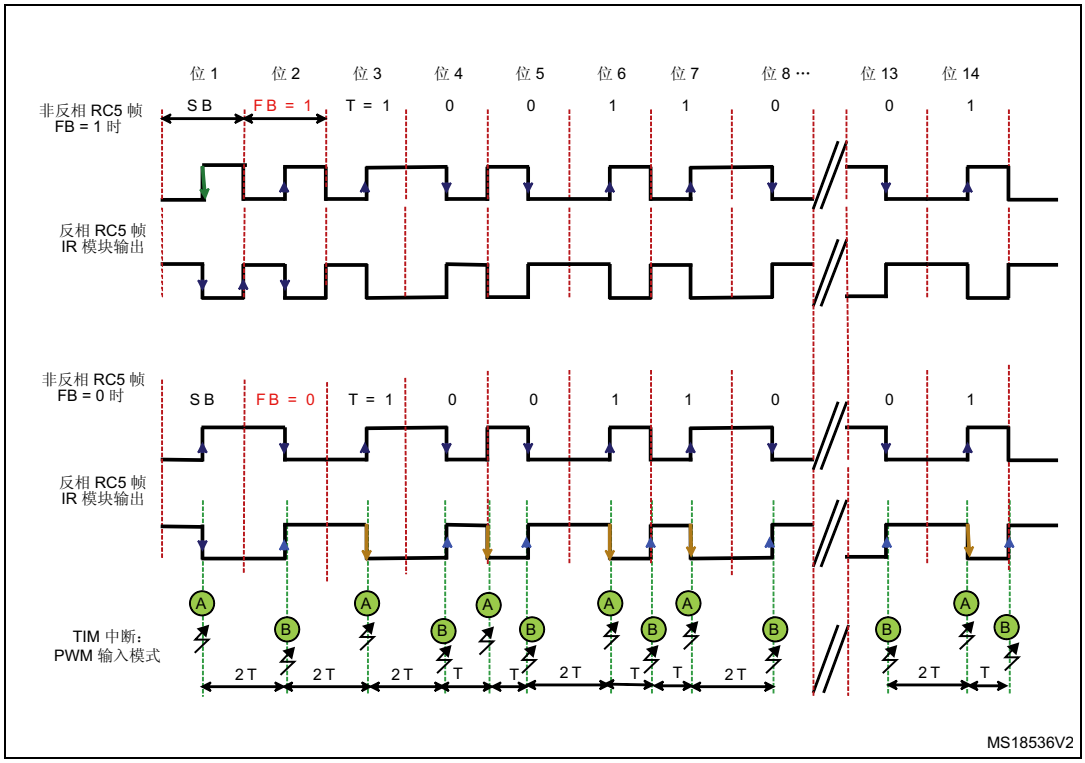
3.3 RC5 协议解决方案

3.3.1 RC5 帧解码机制

图 17 显示了接收 RC5 帧的方式。将 STM32 微控制器内置的外设之一用于以下目的：配置为 PWM 输入模式的定时器。

此输入可以在下降沿和上升沿捕捉当前定时器值，还可以在这两个边沿生成中断。此特性使得测量 RC5 脉冲高电平和低电平的时间变得容易。

图 17. RC5 帧解码机制



- **定时器中断事件：**下降沿

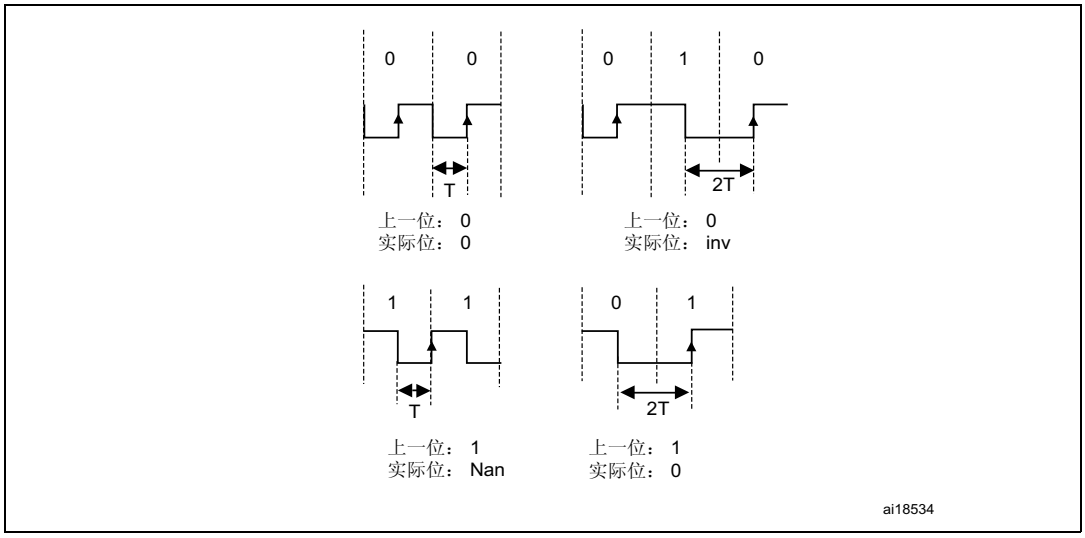
  A： 定时器中断用于测量两个连续下降沿之间的时间段（整个脉冲持续时间）。

- **定时器中断事件：**上升沿

  B： 定时器用于测量下降沿和上升沿之间的持续时间（低脉冲持续时间）。

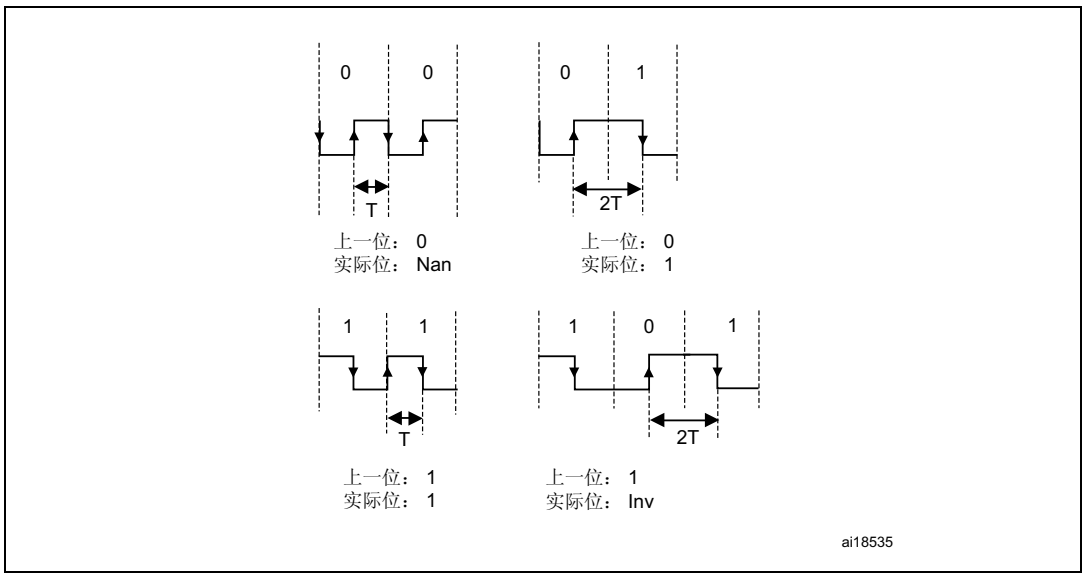
这两个持续时间用于确定位值。每个位值的确定都与上一位有关。

图 18. 根据上升沿确定位：低脉冲



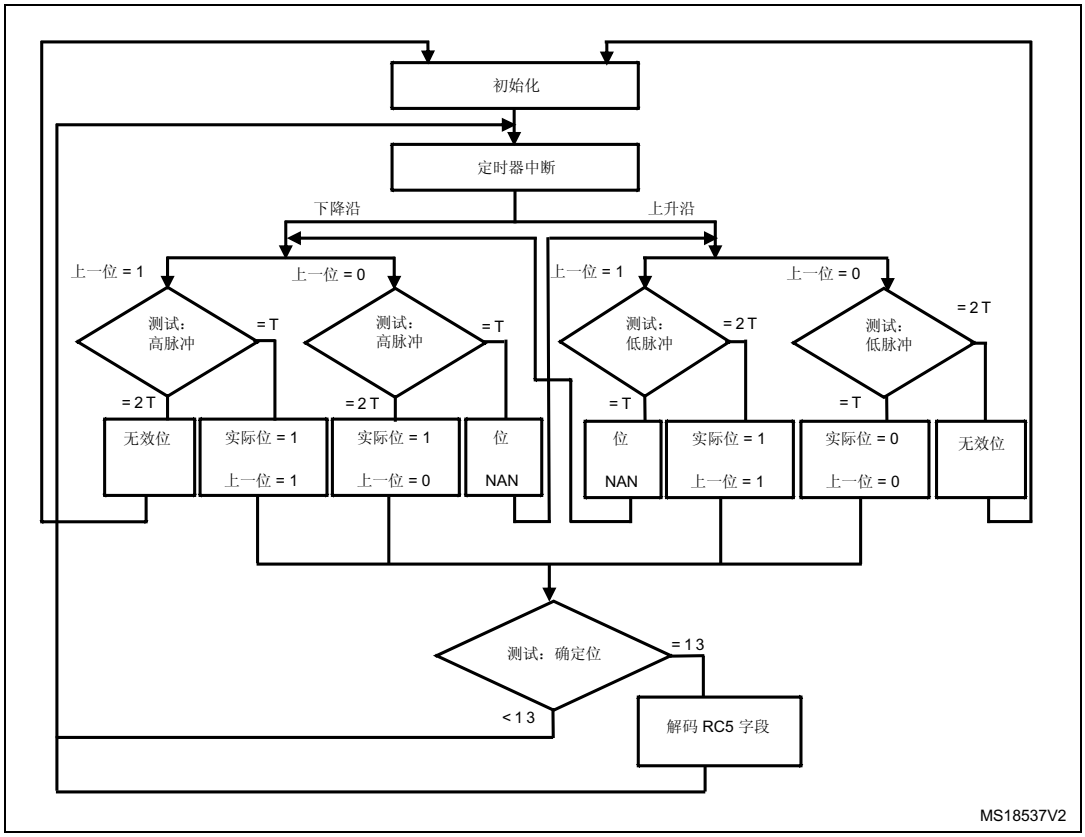
- 如果低脉冲持续时间等于 T 并且确定的上一位是“0”=> 实际位是***逻辑 0***。
- 如果低脉冲持续时间等于 2T 并且确定的上一位是“0”=> 实际位是 ***Inv***（**无效情况**：在这种情况下不能释放）。
- 如果低脉冲持续时间等于 T 并且确定的上一位是“1”=> 实际位是 ***Nan***（**无位**：此位在下一个下降沿确定）。
- 如果低脉冲持续时间等于 2T 并且确定的上一位是“1”=> 实际位是***逻辑 0***。

图 19. 根据下降沿确定位：高脉冲



- 如果高脉冲持续时间等于 T 并且确定的上一位是“0”=> 实际位是 ***Nan***（**无位**：此位在下一个上升沿确定）。
- 如果高脉冲持续时间等于 2T 并且确定的上一位是“0”=> 实际位是***逻辑 1***。
- 如果高脉冲持续时间等于 T 并且确定的上一位是“1”=> 实际位是***逻辑 1***。
- 如果高脉冲持续时间等于 2T 并且确定的上一位是“1”=> 实际位是 ***Inv***（**无效情况**：在这种情况下不能释放）。

图 20. RC5 解决方案流程图



3.3.2 RC5 解码库

RC5 驱动程序非常易于使用。

RC5_Init()

此函数初始化不同的外设（GPIO、定时器...）。

RC5_ResetPacket()

此函数将数据包结构设置为默认状态。主要在 TIM2_IRQHandler 例程中调用此函数。每次定时器溢出时都会调用此函数，以复位 RC5 数据包。

RC5_Decode(RC5_Frame_TypeDef *rc5_frame)

该函数应在用户应用程序中调用。它可解码 RC5 接收到的消息。以下结构包含 RC5 帧的不同值。

```
typedef struct
{
   __IO uint8_t FieldBit;  /* 字段位字段  */
   __IO uint8_t ToggleBit; /* 切换位字段 */
   __IO uint8_t Address;   /* 地址字段    */
```

```
    __IO uint8_t Command;    /* 命令字段     */
  } RC5_Frame_TypeDef ;
```

当 RC5FrameReceived 标志等于 YES 时，执行 IR_RC5_decode ()。

RC5_DeInit()

此函数对不同的外设（GPIO、定时器...）取消初始化。

TIM2_IRQHandler ()

该函数处理 TIM 捕捉比较中断。

- **定时器下降沿事件：**用于测量两个连续下降沿之间的时间段（整个脉冲持续时间）。
- **定时器上升沿事件：**用于测量下降沿和上升沿之间的持续时间（低脉冲持续时间）。
- **更新事件（超时事件）：**复位 RC5 数据包。定时器溢出设置为 3.7 ms。

低脉冲持续时间和整个脉冲持续时间用于确定位值。每个位值的确定都与上一位有关。

3.3.3 如何使用 RC5 解码器驱动程序

要使用 RC5 解码器驱动程序，请按以下步骤操作。

- 调用函数 RC5_Init() 以配置 RC5 解码所需的定时器和 GPIO 硬件资源。
- TIM2 捕捉比较和更新中断用于解码 RC5 帧，如果正确接收到一帧，将设置全局变量“RC5FrameReceived”来通知应用程序。
- 随后应用程序应调用函数 RC5_Decode() 来检索所接收到的 RC5 帧。

代码示例

```
#include "rc5_decode.h"

/* IR_FRAME 将保存 RC5 帧（地址、命令...） */
RC5_Frame_TypeDef IR_FRAME;

/* 初始化 RC5 驱动程序 */
RC5_Init();

while(1)
{
   /* 对接收到的 RC5 帧进行解码并将其存储在 IR_FRAME 变量中
   */
   RC5_Decode(&IR_FRAME);

   /* 此处添加用于处理刚接收到的帧（即 IR_FRAME 变量）的代码，否则它将被下一帧覆盖 */
   ...
}
```

注： 1 *TIMx_IRQHandler ISR 的代码在 stm32f0xx_it.c 或 stm32f3xx_it.c 中。*

- 如果在应用程序中使用一个或两个中断，请务必小心操作：
  - 在这些 ISR 中添加应用程序代码，或
  - 在应用程序代码中复制这些 ISR 的内容。

2 *可以在“ir_decode.h”文件中使用不同的定义声明来轻松定制应用程序，使其符合硬件需求。请参见下表。*

表 4. 实现示例

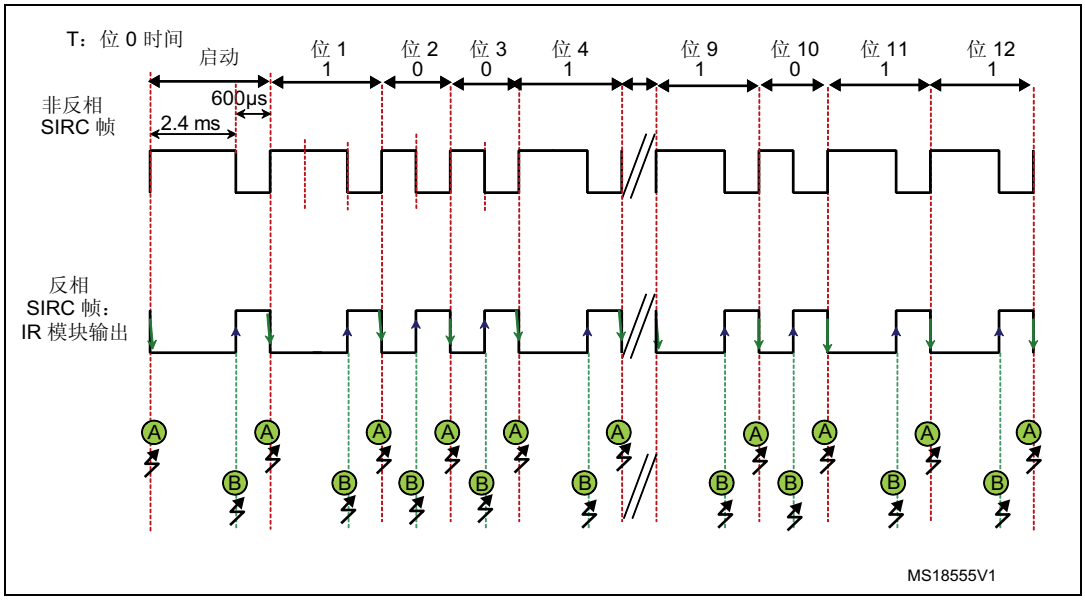
定义名称	说明	STM32F0xx	STM32F30x	STM32F37x
#define IR_TIM	用于 IR 解码的定时器(1)	TIM2	TIM1	TIM3
#define TIM_PRESCALER	TIM 预分频器 计算此参数以将 1 µs 作为时基。TIM 频率 (MHz) /（预分频器 + 1）	47	71	71
#define IR_TIM_CLK	所使用定时器的 APB 时钟	RCC_APB1Periph_TIM2	RCC_APB2Periph_TIM1	RCC_APB1Periph_TIM3
#define IR_TIM_IRQn	IR TIM IRQ	TIM2_IRQn	TIM1_CC_IRQn	TIM3_IRQn
#define IR_TIM_Channel	IR TIM 通道	TIM_Channel_2	TIM_Channel_2	TIM_Channel_2
#define IR_GPIO_PORT	IR 输出连接的端口(1)	GPIOB	GPIOA	GPIOB
#define IR_GPIO_PORT_CLK	IR 引脚 GPIO 端口时钟	RCC_APB2Periph_GPIOB	RCC_AHBPeriph_GPIOA	RCC_AHBPeriph_GPIOB
#define IR_GPIO_PIN	IR 连接的引脚(1)	GPIO_Pin_3	GPIO_Pin_9	GPIO_Pin_5

1. 有关可用 STM32 资源的更多详细信息，请参见产品数据手册。

3.4 SIRC 红外控制解决方案

3.4.1 软件实现

图 21. SIRC 帧接收机制

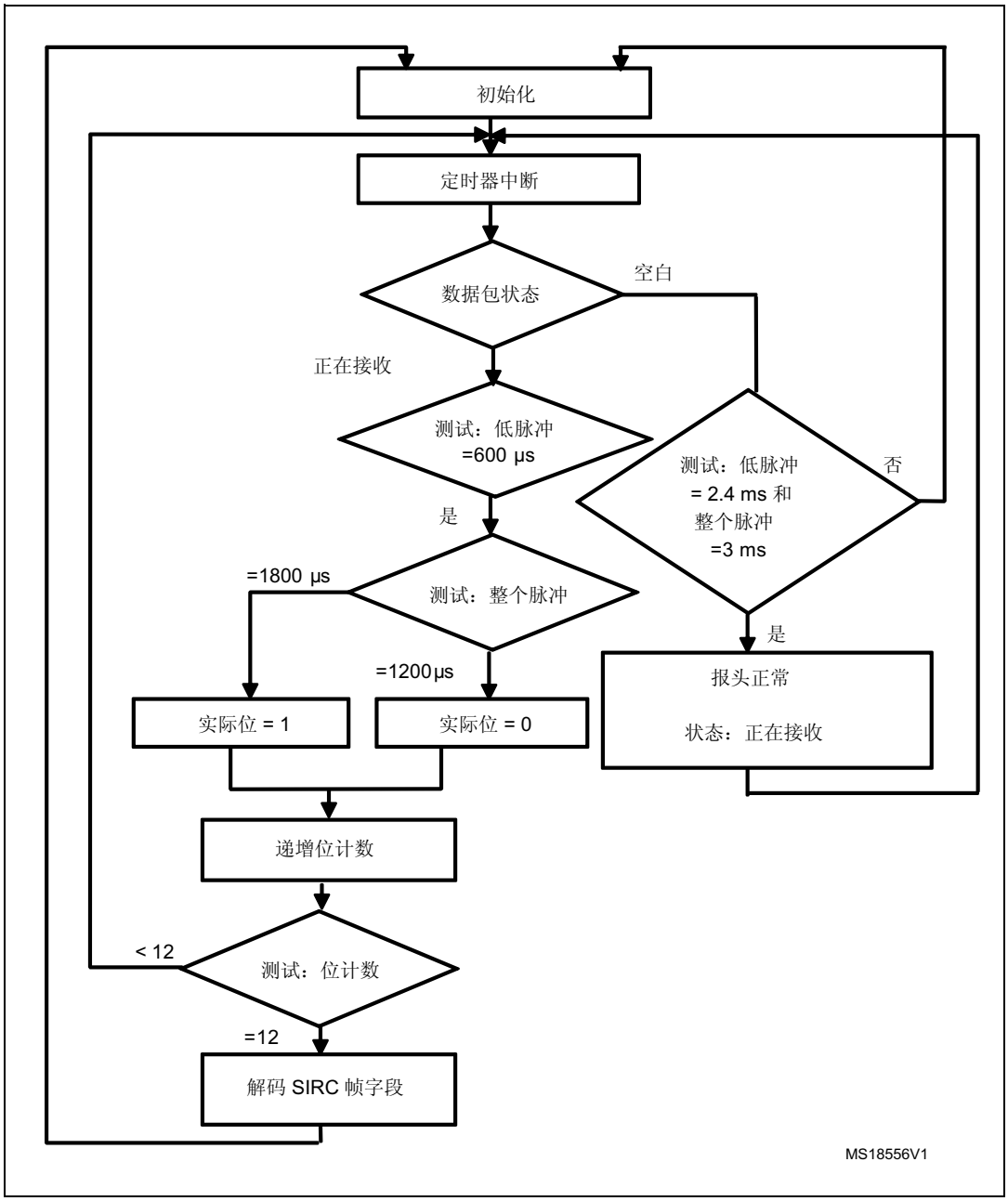


定时器中断：在 PWM 输入模式下

定时器用于对 SIRC 帧的不同位进行采样。可以在下降沿和上升沿捕捉当前定时器值，还可以在这两个边沿生成中断。此特性使得测量 SIRC 脉冲的完整时间和低电平时间变得容易。

- 如果测量的周期等于 T = 1200 µs 并且低脉冲持续时间等于 T/2 = 600 µs => 该位为***逻辑“0”***。
- 如果测量的周期等于 3T/2 = 1800 µs 并且低脉冲持续时间等于 T = 1200 µs => 该位为***逻辑“1”***。
- 如果测量的整个周期等于 3000 µs 并且低脉冲持续时间等于 2400 µs => 该位为***“起始位”***。

图 22. SIRC 解决方案流程图



3.4.2 SIRC 库

SIRC_Init()

此函数初始化用于 SIRC 协议的不同外设。

SIRC_Decode (SIRC_Frame_TypeDef *sirc_frame)

该函数应在用户应用程序中调用。它可解码 SIRC 接收到的消息。它将包含 IR 帧不同值的结构作为参数。

```
typedef struct
{
   __IO uint8_t Command;          /* 命令字段     */
   __IO uint8_t Address;          /* 地址字段     */
} SIRC_Frame_TypeDef;
```

当 IRFrameReceived 标志等于 YES 时必须执行 SIRC_decode ()。

SIRC_ResetPacket()

此函数将 IR 数据包置于默认状态。此函数在 TIM2_IRQHandler 例程中调用。每次定时器溢出时都会调用此函数，以复位 IR 数据包。

SIRC_DeInit()

此函数对用于 SIRC 协议的不同外设取消初始化。

TIM2_IRQHandler ()

该函数处理 TIM 捕捉比较中断。

- **定时器下降沿事件**：用于测量两个连续下降沿之间的不同时间段，以识别帧位。
- **定时器上升沿事件**：用于测量下降沿和上升沿之间的持续时间（低脉冲持续时间）。
- **更新事件（超时事件）**：复位 RC5 数据包。定时器溢出设置为 4 ms。

位值由这两个持续时间确定。

3.4.3 如何使用 SIRC 解码器驱动程序

要使用 SIRC 解码器驱动程序，请按以下步骤操作。

- TIM2 捕捉比较和更新中断用于解码 IR 帧。如果正确接收到一帧，将设置全局变量“IRFrameReceived”来通知应用程序。
- 随后应用程序应调用函数 SIRC_Decode() 来检索接收到的 IR 帧。
- 可以轻松定制此驱动程序以适合任何其它红外协议，只需按照相应的红外协议规范（位持续时间、报头持续时间、边缘容差、位数 ...）以及命令和设备描述表来修改 sirc_decode.h 中的定义。

代码示例

```
#include "sirc_decode.h"

/* SIRC_FRAME 将保存 SIRC 帧（地址、命令...） */
SIRC_Frame_TypeDef SIRC_FRAME;

/* 初始化 SIRC 驱动程序 */
SIRC_Init();

while(1)
   {
   /* 对接收到的 SIRC 帧进行解码并将其存储在 SIRC_FRAME 变量中 */
   SIRC_Decode(&SIRC_FRAME);

   /* 此处添加用于处理刚接收到的帧（即 SIRC_FRAME 变量）的代码，否则它将被下一帧覆盖 */
   ...
   }
```

注： 1 *TIMx_IRQHandler ISR 的代码在 stm32f0xx_it.c 或 stm32f3xx_it.c 驱动程序中。*

- 如果在应用程序中使用一个或两个中断，请务必小心操作：
  - 在这些 ISR 中添加应用程序代码，或
  - 在应用程序代码中复制这些 ISR 的内容。

2 *可以在“ir_decode.h”文件中使用不同的定义声明来轻松定制应用程序，使其符合硬件需求。*

表 5. 实现示例

定义名称	说明	STM32F0xx	STM32F30x	STM32F37x
#define IR_TIM	用于 IR 解码的定时器(1)	TIM2	TIM1	TIM3
#define TIM_PRESCALER	TIM 预分频器 计算此参数以将 1 µs 作为时基。TIM 频率 (MHz) /（预分频器 + 1）	47	71	71
#define IR_TIM_CLK	所使用定时器的 APB 时钟	RCC_APB1Periph_TIM2	RCC_APB2Periph_TIM1	RCC_APB1Periph_TIM3
#define IR_TIM_IRQn	IR TIM IRQ	TIM2_IRQn	TIM1_CC_IRQn	TIM3_IRQn
#define IR_TIM_Channel	IR TIM 通道	TIM_Channel_2	TIM_Channel_2	TIM_Channel_2
#define IR_GPIO_PORT	IR 输出连接的端口(1)	GPIOB	GPIOA	GPIOB

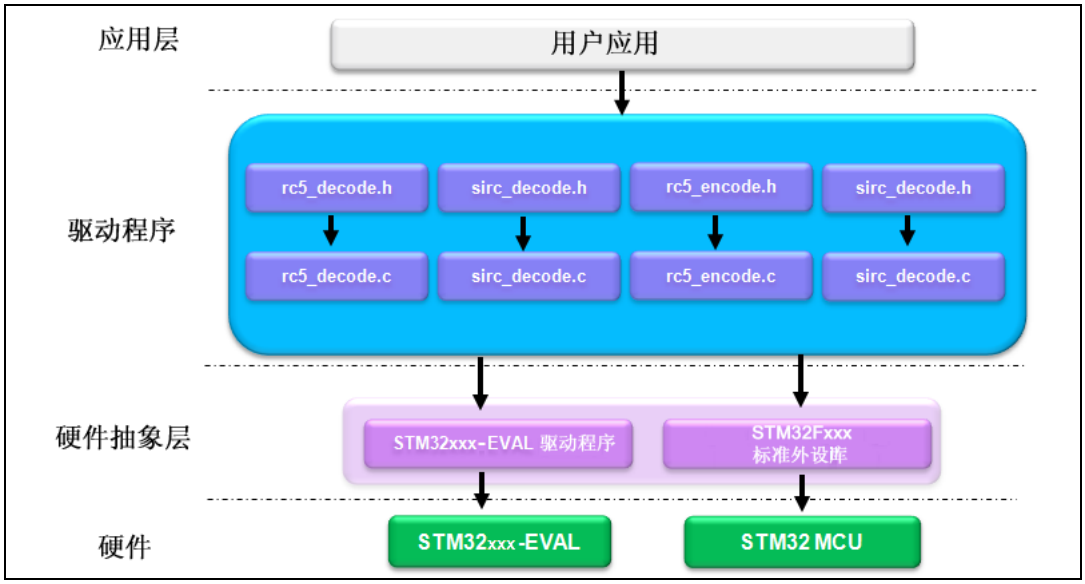
表 5. 实现示例（续）

定义名称	说明	STM32F0xx	STM32F30x	STM32F37x
#define IR_GPIO_PORT_CLK	IR 引脚 GPIO 端口时钟	RCC_APB2Periph_GPIOB	RCC_AHBPeriph_GPIOA	RCC_AHBPeriph_GPIOB
#define IR_GPIO_PIN	IR 连接的引脚(1)	GPIO_Pin_3	GPIO_Pin_9	GPIO_Pin_5

1. 有关可用 STM32 资源的更多详细信息，请参见产品数据手册。

4 接口层

图 23. 应用层体系结构



许多相似的红外协议，比如说 SIRC 协议，只是在时序参数存在差别。这些协议由 sirc_decode.c/sirc_encode.c 函数处理。用户只需更新时序值。

还有一些协议完全不同，它们由特定函数（例如 RC5 及其相关驱动程序 rc5_decode.c/rc5_encode.c）管理。

每个协议都有特定的结构帧。IR_FRAME 是指向所选红外协议结构的指针，它包含通信所需的主要信息（设备地址和命令）。

4.1 演示程序

为确保快速入门，本文档中介绍的红外发送器和接收器使用 C 语言实现，并在 STM320518-EVAL(Config2)、STM32373C-EVAL 和 STM32303C-EVAL 演示包中提供。

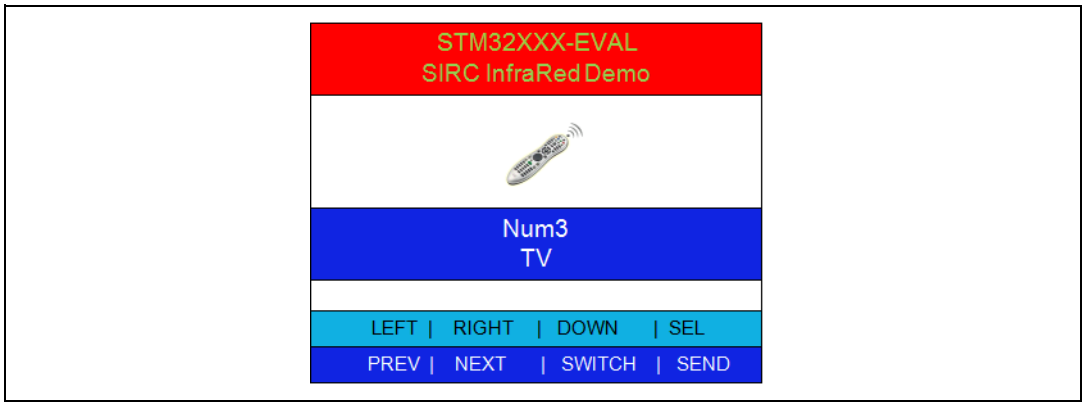
4.1.1 使用 IRTIM 的发送器演示

此演示旨在发送在 LCD 上显示的 IR 消息。

每个 IR 消息以两部分显示。

- IR 设备接收器。
- 要执行的命令。

图 24. IR 发送器演示



4.1.2 使用配置为 PWM 模式的通用定时器的接收器演示

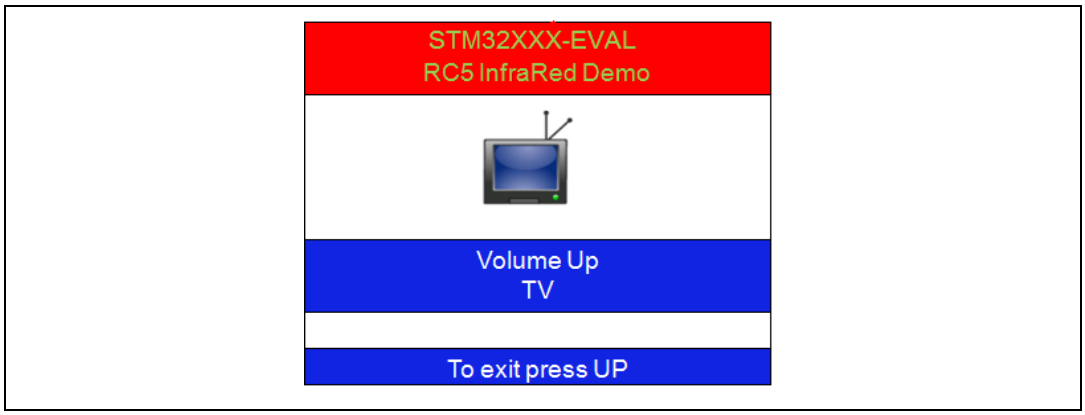
此演示旨在接收 IR 消息并将这些消息发送到 LCD。

每个 IR 消息以两部分显示。

- 发送 IR 帧的设备。
- 要执行的命令。

下图显示了使用 PWMI 方法的 RC5 解码器（请参见*第 18 页的第 3.2 节*）。

图 25. LCD 中显示的 RC5 接收的帧



4.2 如何自定义 IR 驱动程序

4.2.1 IR 接收器驱动程序

要在用户应用程序中包含基于 PWM 输入解决方案的红外解码器驱动程序，必须：

1. 将相应 IR 协议的头文件添加到项目中。
   示例：rc5_decode.h。
2. 将与 IR 协议对应的 .c 文件添加到项目中。
   示例：rc5_decode.c。
3. 在 main() 中调用协议初始化函数
   示例：`RC5_Init();`
4. 将 TIMx 中断函数添加到 *stm32f0xx_it.c 或 stm32f3xx_it.c*。
   示例：

```
void TIM2_IRQHandler (void)
{
   static uint32_t ICValue1;
   static uint32_t ICValue2;
   /* IC1 中断*/
   if((TIM_GetFlagStatus(IR_TIM, TIM_FLAG_CC1) != RESET))
   {
      TIM_ClearFlag(IR_TIM, TIM_FLAG_CC1);
      /* 获得输入捕捉值 */
      ICValue2 = TIM_GetCapture1(IR_TIM);
      /* RC5 */
      RC5_DataSampling(ICValue2 - ICValue1, 0);
   }
   /* IC2 中断 */
   else if((TIM_GetFlagStatus(IR_TIM, TIM_FLAG_CC2) != RESET))
   {
      TIM_ClearFlag(IR_TIM, TIM_FLAG_CC2);
      /* 获得输入捕捉值 */
      ICValue1 = TIM_GetCapture2(IR_TIM);
      RC5_DataSampling(ICValue1 , 1);
   }
   /* 检查 IR_TIM 标志是否置 1。*/
   else if ((TIM_GetFlagStatus(IR_TIM, TIM_FLAG_Update) != RESET))
   {
      /* 清除 IR_TIM 的挂起标志*/
      TIM_ClearFlag(IR_TIM, TIM_FLAG_Update);
      RC5_ResetPacket();
   }
}
```

5. 在文件 main.c 中为 IR 协议定义一个结构。
   示例：
   ```
   RC5_Frame_TypeDef  IR_FRAME。
   ```
6. 在 main() 中调用解码函数。
   示例：
   ```
   void main(void)
   {
      RC5_Init();
      while(1)
      {
      RC5_Decode(&IR_Frame).
      }
   }
   ```

支持任意 IR 协议所需的更改

可以使用此解决方案支持任意红外协议，只需在头文件中做几处更改并更新命令和设备表。

– 创建一个类似于 sirc_decode.h 文件的头文件（例如：ir_protocol_name.h）。更改定义以使其适应所选 IR 协议的规范（位持续时间最小值/最大值、标头持续时间最小值/最大值、总位数、超时...）

表 6. 红外协议参数相关的头文件定义列表

定义	含义	SIRC 协议设置示例
IR_Time_OUT_US	超时 (µs)	4050
IR_BITS_COUNT	位数	11
IR_TOTAL_BITS _COUNT	总位数	11
IR_ONTIME_MIN_US	最小低脉冲 (µs)	(600 - 60)
IR_ONTIME_MAX_US	最大低脉冲 (µs)	(1200 + 60)
IR_HEADER_LOW_MIN_US	最小报头低脉冲 (µs)	(2400 - 150)
IR_HEADER_LOW_MAX_US	最大报头低脉冲 (µs)	(2400 + 150)
IR_HEADER_WHOLE_MIN_US	最小报头完整持续时间 (µs)	(2400 + 600 - 60)
IR_HEADER_WHOLE_MAX_US	最大报头完整持续时间 (µs)	(2400 + 600 + 60)
IR_VALUE_STEP_US	位 0 与位 1 之间的步进值 (µs)	600

表 6. 红外协议参数相关的头文件定义列表（续）

定义	含义	SIRC 协议设置示例
IR_VALUE_MARGIN_US	裕量 (µs)	100
IR_VALUE_00_US	位 0 持续时间 (µs)	1200

注： *IR 指 IR 协议的名称。例如，SIRC_HEADER_LOW_MIN_US。*

– 更改 IR_Frame_TypeDef 结构中的 IR 协议帧字段。

```
typedef struct
{

/* IR 帧的结构（地址、命令...）*/

} IR_Frame_TypeDef;
```

– 在 sirc_decode.c 文件中，为 IR 协议添加适当的 IR_Commands 和 IR_devices 表。

4.2.2 IR 发送器驱动程序

要在用户应用程序中包含基于 IRTIM 解决方案的红外编码器驱动程序，必须：

1. 将相应 IR 协议的头文件添加到项目中。
   示例：rc5_encode.h。
2. 将与 IR 协议对应的 .c 文件添加到项目中。
   示例：rc5_encode.c。
3. 在 main() 中调用协议初始化函数
   示例：`RC5_Encode_Init();`
4. 将 TIMx 中断函数添加到 *stm32f0xx_it.c 或 stm32f3xx_it.c*。
   示例：

```
void TIM16_IRQHandler(void)
{
  RC5_Encode_SignalGenerate(RC5_FrameManchestarFormat);

  /* 清除 TIM16 更新中断 */
  TIM_ClearITPendingBit(TIM16, TIM_IT_Update);
}
```

5. 在 main() 中调用编码函数。
   示例：

```
void main(void)
{
   RC5_Encode_Init();
   while(1)
   {
      RC5_Encode_SendFrame(Address, Instruction, Control)
   }
}
```

5 结论

本应用笔记为使用通用定时器在软件中实现 IR 发送器/接收器提供了解决方案。

IR 编码应用程序使用 STM32F0xx 和 STM32F3xx 微控制器，并包含一个功能强大的硬件调制器 (IRTIM)，该调制器结合了来自两个内部定时器的信号来驱动 IR 接口。此特性使微控制器特别适合于需要 IR 信号生成能力的应用程序。

IR 解码应用程序可以使 IR 解决方案集成到 HDMI-CEC 模块中，以在给定环境下支持所有各种视听产品的高级控制功能。

6 版本历史

表 7. 文档版本历史

日期	版本	变更
2012 年 05 月 02 日	1	初始版本。
2012 年 10 月 23 日	2	增加了对 STM32F3xx 系列微处理器的支持。

请仔细阅读：

中文翻译仅为方便阅读之目的。该翻译也许不是对本文档最新版本的翻译，如有任何不同，以最新版本的英文原版文档为准。

本文档中信息的提供仅与ST产品有关。意法半导体公司及其子公司（“ST”）保留随时对本文档及本文所述产品与服务进行变更、更正、修改或改进的权利，恕不另行通知。

所有ST产品均根据ST的销售条款出售。

买方自行负责对本文所述ST产品和服务的选择和使用，ST概不承担与选择或使用本文所述ST产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为 ST 授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在ST的销售条款中另有说明，否则，ST对ST产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

意法半导体的产品不得应用于武器。此外，意法半导体产品也不是为下列用途而设计并不得应用于下列用途：（A）对安全性有特别要求的应用，例如，生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）汽车应用或汽车环境，且/或（D）航天应用或航天环境。如果意法半导体产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向意法半导体发出了书面通知，采购商仍将独自承担因此而导致的任何风险，意法半导体的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML或JAN正式认证产品适用于航天应用。

经销的ST产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致ST针对本文所述ST产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大ST的任何责任。

ST和ST徽标是ST在各个国家或地区的商标或注册商标。

本文档中的信息取代之前提供的所有信息。

ST徽标是意法半导体公司的注册商标。其他所有名称是其各自所有者的财产。

意法半导体集团公司

澳大利亚 - 比利时 - 巴西 - 加拿大 - 中国 - 捷克共和国 - 芬兰 - 法国 - 德国 - 中国香港 - 印度 - 以色列 - 意大利 - 日本 - 马来西亚 - 马耳他 - 摩洛哥 – 菲律宾 – 新加坡 - 西班牙 - 瑞典 - 瑞士 - 英国 - 美国

www.st.com