AN3174
Application note

Implementing an RC5 infrared remote control receiver with the STM32F10xx microcontrollers

Introduction

Nowadays, almost all audio and video equipment can be controlled using an infrared remote control. RC5 is the most popular protocol for transmitting data via infrared light. It was developed by Philips in the late 1980's.

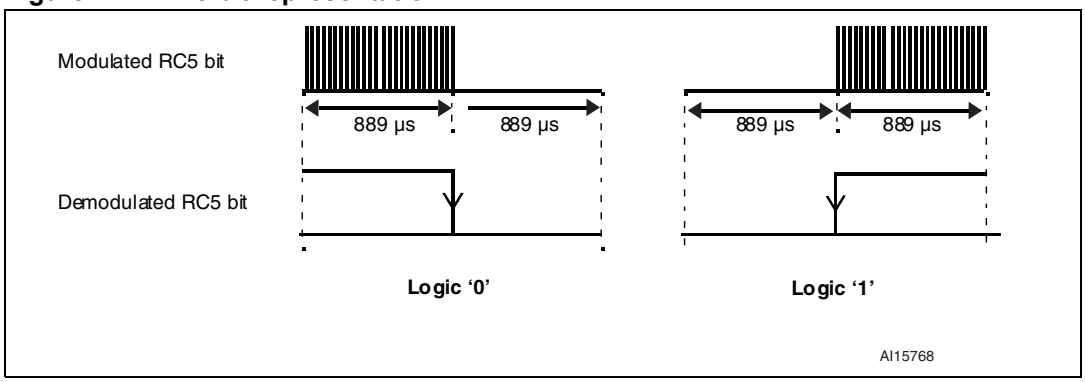
This application note describes a software solution for implementing an RC5 receiver using the STM32F10xx microcontrollers.

Contents

1 RC5 protocol basics

The RC5 code is a 14-bit word, it uses bi-phase modulation (also called Manchester coding) of a 36 kHz IR carrier frequency. All bits have an equal length of 1.778 ms, with half of the bit time filled with a burst of the 36 kHz carrier and the other half being idle. A logical zero is represented by a burst in the first half of the bit time. A logical one is represented by a burst in the second half of the bit time (refer to *Figure 1.*). The duty cycle of the 36 kHz carrier frequency is 33% or 25% which reduces power consumption.

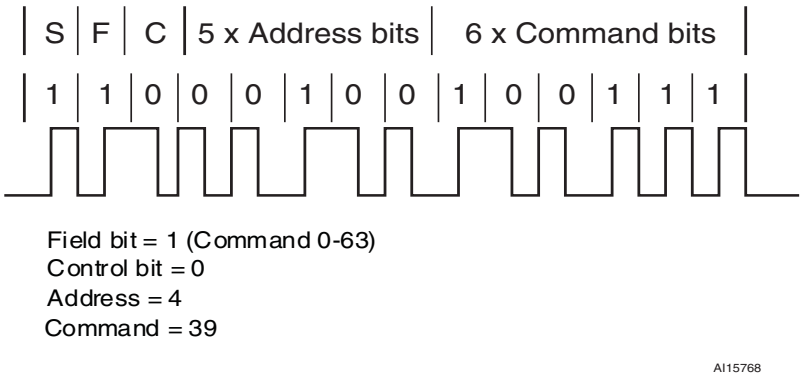
Figure 1. RC5 bit representation



The RC5 frame can generate 2048 (32x64) different commands organized on 32 groups, each group has 64 different commands. A RC5 frame contains the following fields (An example of RC5 frame is shown in *Figure 2.*):

- **Start bit (S):** 1 bit length, always logic 1.
- **Field bit (F):** 1 bit length, which denotes whether the command sent is in the lower field (logic 1 = 0 to 63 decimal) or the upper field (logic 0 = 64 to 127 decimal). The field bit was added later by Philips when it was realized that 64 commands per device were insufficient. Previously, the field bit was combined with the start bit. Many devices still use this original system.
- **Control bit or Toggle bit (C):** 1 bit length, which toggles each time a button is pressed. This allows the receiving device to distinguish between two successive button presses (such as "1", "1" for "11").
- **Address:** 5 bits length, that selects one of 32 possible systems.
- **Command:** 6 bits length, that (in conjunction with the field bit) represents one of the 128 possible RC-5 commands.

Figure 2. Example of an RC5 frame



To avoid frame collisions, an idle time is inserted between two frames with a specific width (see *Figure 3.*).

The idle time is defined as 50 bits wide. So the periodicity of a frame is 64 x 1 bit width: 64 x 1.778 = 113.792 ms (exactly 113.788 ms).

Figure 3. RC5 idle time

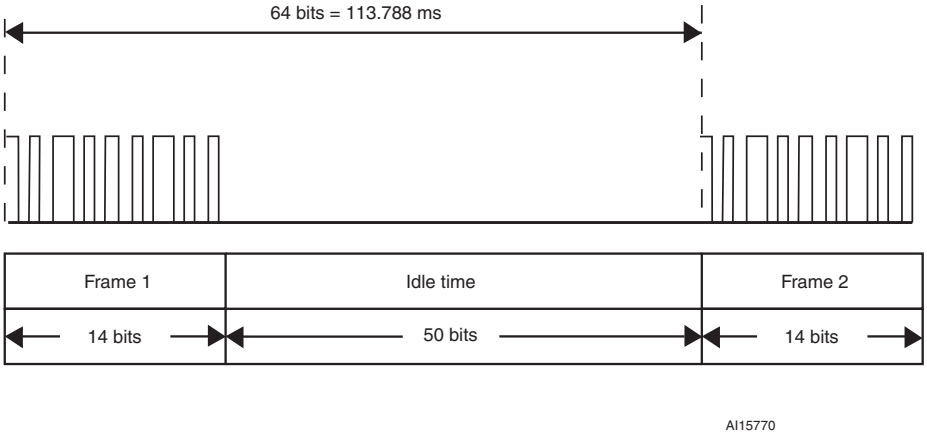


Table 1. RC5 timings

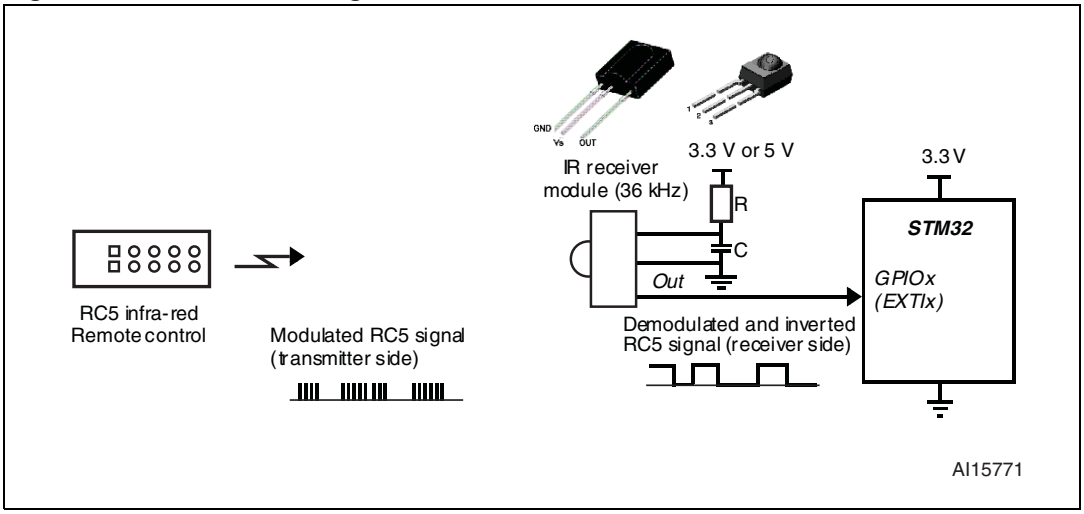
Description	Min.	Typical	Max.	Units
RC5 Half bit period	640	899	1140	µs
RC5 Full bit period	1340	1778	2220	µs
RC5 message time	23.644	24.889	26.133	ms
RC5 message repetition time	108.089	113.778	119.467	ms
Carrier pulse bit time	27.233	27.778	28.345	µs

2 Hardware considerations

To improve noise rejection, the pulses are modulated at 36 kHz. The easiest way to receive these pulses is to use an integrated IR-receiver/demodulator module like the TSOP1736 (5 V supply version) or TSOP34836 (3.3 V supply version) or other equivalent part number (refer to *Figure 4.*). These are 3-pin devices that receive the infrared burst and output the demodulated bit stream on the output pin which is connected directly to one of the STM32 microcontroller's GPIO pins. The GPIO pin used is selected by the user (refer to the section *Section 3.2.3 on page 8*). If TSOP1736 is used, the selected GPIO should be Five volt Tolerant (FT). The output of the IR module is inverted compared to the transmitted data (the data is idle high and logic '0' becomes logic '1' and vice versa)

Note: The IR module needs two external components: a capacitor and a resistor (refer to the related IR module datasheet for their values).

Figure 4. Hardware configuration

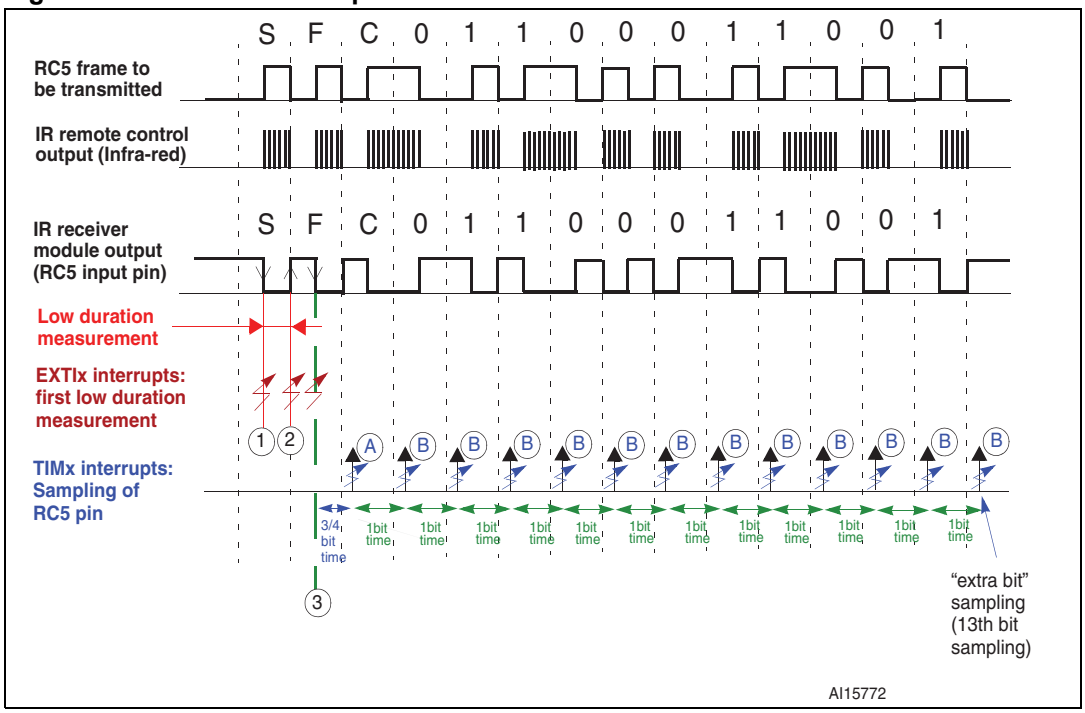


3 Software implementation

3.1 RC5 frame reading mechanism

Figure 5. shows how the RC5 frame is received. Principally, two of the STM32 microcontroller's embedded peripherals are used for this purpose: EXTI and a timer (TIMx). The STM32 pin connected to the IR module's output pin can be any GPIO selected by the user (see *Section 3.2: How to use the RC5 library on page 7*).

Figure 5. RC5 frame reception mechanism



3.1.1 EXTI interrupt events

The EXTI interrupt is used to start and stop TIMx in order to measure the first low duration to validate the header timing of the RC5 frame. Refer to events 1, 2 and 3 in *Section Figure 5.: RC5 frame reception mechanism on page 6*

- First EXTI interrupt event (1): the TIMx counter is initialized and enabled.
- Second EXTI interrupt event (2): the TIMx counter is disabled, read and then initialized. The value read from the counter gives the measured duration. The 3rd execution of the EXTI interrupt depends on the measured duration:
  - If the duration is within the tolerance range of one half bit time, the EXTI is not disabled and the EXTI interrupt occurs for the 3rd time, which enables TIMx. TIMx then starts sampling the RC5 data. In this case the *Field* bit will be recognized as a logical 1.
  - If the duration is within the tolerance range of one bit time, the EXTI is then disabled at this moment and the TIMx update event interrupt as well as the TIMx counter are enabled to start sampling the RC5 data. In this case the *Field* bit will

recognized as a logical 0. If the duration seems to be a glitch, the system will be initialized for the next RC5 frame.

- Third EXTI interrupt event (3): this interrupt occurrence depends on the duration of the first low duration, see (2). When the interrupt occurs, TIMx is enabled and starts sampling the RC5 data.

3.1.2 TIMx interrupt events

TIMx is used to sample each bit of the RC5 frame after checking the timing of the first low duration of the frame.

TIMx interrupts are executed 13 times during an RC5 frame in order to sample all the bits. The Start bit (S) and Field bit (F) are not sampled by TIMx and an « extra bit » is sampled at the end of the RC5 frame to be sure that all bits have been received and an idle state is present.

- TIMx interrupt event (A): at this time, the RC5 pin is sampled by a single reading of the GPIO input data register. In this interrupt service routine, the TIMx is configured to generate a periodic interrupt each bit time.
- TIMx interrupts event (B): at this time, the RC5 pin is sampled by a single reading of the GPIO input data register and the interrupt service routine checks if the number of data bits has reached 13 (n = 13: 14-2+1) If yes, the TIMx counter and the TIMx update interrupt are disabled.

As we can see, reading from the GPIO input data register directly reflects the value of the bit. If the read value is at low level this implies that the bit value is logic ‘0’. If the value read is at high level this implies that the bit value is logic‘1’.

3.2 How to use the RC5 library

The RC5 driver is very simple to use. There are four functions accessible for the user.

3.2.1 RC5_Receiver_Init() function

This function is intended to initialize the different peripherals used by the RC5 driver: GPIOs, EXTI and TIMx. It should be called after the user clock configuration.

3.2.2 RC5_Sample_Data() function

This function is used to sample the RC5 data. It should be called in the RC5_TIM_IRQ_Handler routine (TIMx_IRQHandler) in the stm32f10x_it.c file. The RC5_IR_Receiver.h file should be included in the stm32f10x_it.c file. By default, TIM2 is used. You can use any timer (TIMx in the list below) by modifying the defines in the RC5_IR_Receiver.h file(path: \STM32F10x_AN3174_FW_VX.Y.Z\Libraries\ STM32F10x_RC5_Emul_Receiver_Lib\inc\RC5_IR_Receiver.h) as follows:

Example:

If you want to use TIM3, make these modifications:

```
#define RC5_TIM                 TIM3
#define RC5_TIM_CLK             RCC_APB1Periph_TIM3
#define RC5_TIM_IRQn            TIM3_IRQn
```

```
#define RC5_TIM_IRQ_Handler     TIM3_IRQHandler
```

- You can choose any of the STM32F10x family timers.

3.2.3 RC5_MeasureFirstLowDuration() function

This function measures and validates the first low duration of the RC5 frame. When this timing is in the range of the allowed timings, the function enables the RC5 frame sampling. This function should be called in the appropriate EXTI interrupt handler (in stm32f10x_it.c file) depending on the GPIO used for the RC5 input pin.

By default, GPIOB.01 is used as the RC5 input pin. You can use any GPIO by modifying the defines in the RC5_IR_Receiver.h file path: \STM32F10x_AN3174_FW_VX.Y.Z\Libraries\STM32F10x_RC5_Emul_Receiver_Lib\inc\RC5_IR_Receiver.h).

Example:

If you want to use GPIOD.09, make these modifications:

```
#define RC5_GPIO_PORT             GPIOD
#define RC5_GPIO_CLK              RCC_APB2Periph_GPIOD
#define RC5_GPIO_PIN              GPIO_Pin_9
#define RC5_EXTI_PORT_SOURCE      GPIO_PortSourceGPIOD
#define RC5_EXTI_PIN_SOURCE       GPIO_PinSource9
#define RC5_EXTI_IRQn             EXTI9_5_IRQn
#define RC5_EXTI_LINE             EXTI_Line9
#define RC5_EXTI_IRQ_Handler      EXTI9_5_IRQHandler
```

RC5_MeasureFirstLowDuration should be called in the RC5_EXTI_IRQHandler.

3.2.4 RC5_Decode() function

This function is intended to be called in the user application. It decodes the RC5 received messages. It returns a structure that contains the different values of the RC5 frame.

```
typedef struct
{
  __IO uint8_t ToggleBit;  /* Toggle bit field */
  __IO uint8_t Address;    /* Address field */
  __IO uint8_t Command;    /* Command field */
} RC5Frame_TypeDef;
```

RC5_decode() should be called when the RC5_FrameReceived flag is equal to YES.

Example of usage:

```
  /* System Clocks Configuration */
  RCC_Configuration();

  /* Initialize RC5 reception */
```

```
RC5_Receiver_Init();

while (1)
{
   /* If RC5 frame has been received, then decode it */
   if (RC5_FrameReceived == YES)
   {
     /* Get the RC5 frame */
      RC5_Frame = RC5_Decode();
   }
}
```

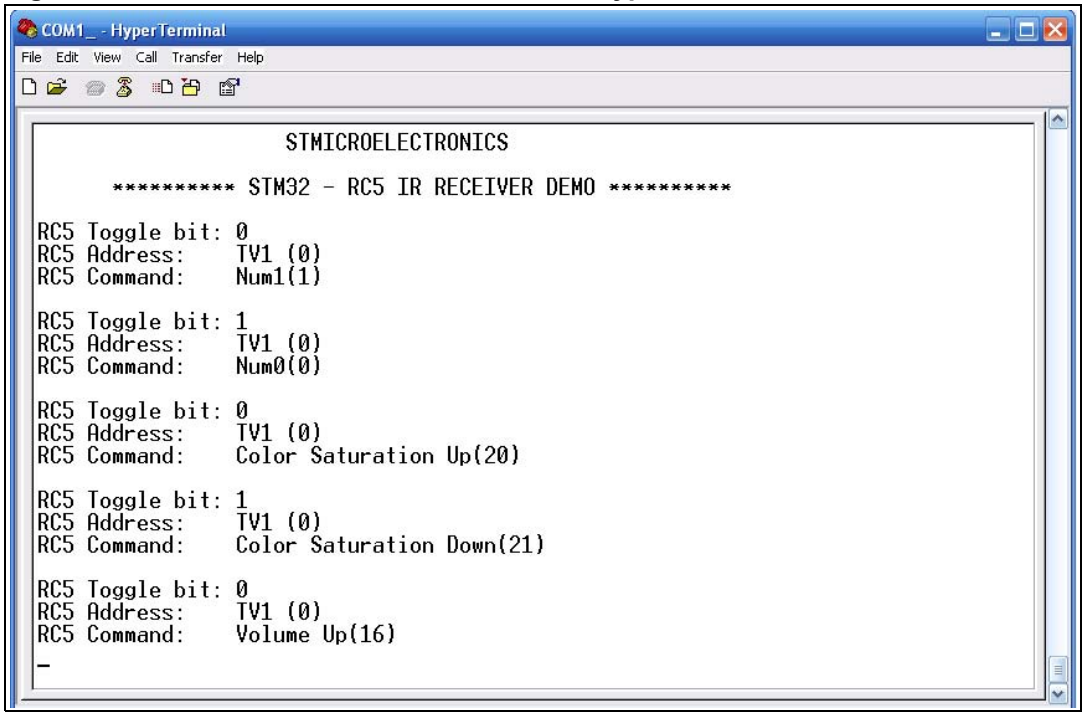
3.3 RC5 Demo description

The RC5 demo consists of receiving RC5 messages and sending them to the hyperterminal (see *Figure 6.*) using USART1 (115200 baud, 8-bit data, No parity, No Flow Control). Each RC5 message is displayed in 3 parts:

- The value of the toggle bit.
- The device which transmitted the RC5 with its decimal value in brackets.
- The command to be executed and its decimal value in brackets.

When an RC5 message is received, LED1 will toggle on the board.

Figure 6. RC5 received frames shown in the hyperterminal



4 Conclusion

This application note provides a solution for implementing a RC5 receiver in software using an EXTI and a general purpose Timer (TIMx). The driver is very simple to use and it supports standard and extended RC5 formats, which is not the case for several of the RC5 drivers offered by other manufacturers.

5 Revision history

Table 2. Document revision history

Date	Revision	Changes
01-Apr-2010	1	Initial release.