ESP8266 IR Remote Control User Guide

Version 0.3

Espressif Systems IOT Team

http://bbs.espressif.com/

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The WiFi Alliance Member Logo is a trademark of the WiFi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Table of Contents

1. Introduction to Infrared Transmission

Users can request the sample codes of infrared transmission by sending an e-mail to feedback@espressif.com.

This document introduces how to implement transmitting or receiving remote control codes using the 32-bit NEC IR transmission protocol as an example.

1.1. Transmitting

Users can use the following methods to transmit carrier wave:

- BCK of I2S
- 38KHz carrier frequency generated by WS pin
- Carrier wave generated by any GPIO via sigma-delta function. However, the duty ratio of carrier wave generated by sigma-delta is around 20%, thus MTMS pin (GPIO14) is suggested, for this pin can generate standard square wave at a carrier frequency of 38KHz and a duty ratio of 50% exactly.

In the sample codes, data transmission queue is generated via the DSR TIMER interface of system FRC2, while a state machine driving the transmission of infrared data is also generated.

Considering that the timing precision of transmitting NEC infrared code should reach a level of μs, when initiating IR TX, `system_timer_reinit` should be invoked to improve the timing precision of FRC2. In `user_config.h`, enable the definition of `USE_US_TIMER`, then interface function `os_timer_arm_us` can be invoked to implement precise timing at the level of μs.

1.2. Receiving

The receiving of remote control codes is implemented via edge-triggered interrupt. When one system is substracted from one another, the result is the duration time of the wave. This can be processed by software state machine `ir_intr_handler`.

Note:

- Receiving of infrared remote control codes is implemented via GPIO interrupt. However, the system can only register only one IO interrupt handler program at the same time. If other IOs also need interrupts, please handle these interrupts in the same processing program by determine the source of interrupt and deal with them accordingly.
- In non-OS version of SDK, functions with ICACHE_FLASH_ATTR properties, including print function os_printf defined in IROM section of the Flash, should NOT be invoked in the whole process of interrupt handling process such as GPIO, UART, FRC, etc.

2. Parameters Configuration

All kinds of parameters related to transmitting and receiving of infrared remote control codes can be configured in `ir_tx_rx.h`.

2.1. Config Parameters for Transmitting

```
#define GEN_IR_CLK_FROM_IIS 0
                        // Config the mode of carrier
                        // 1: IIS clock signal generates carrier wave for transmission
                        // 0: generate carrier wave for transmission under GPIO sigma-delta model
                        // Suggest using MTMS pin to implement infrared transmitting function.
```

```
                        // Config the register function and multiplexing function of infrared pins
#define IR_GPIO_OUT_MUX     PERIPHS_IO_MUX_GPIO5_U
#define IR_GPIO_OUT_NUM     5
#define IR_GPIO_OUT_FUNC    FUNC_GPIO5
```

2.2. Config Parameters for Receiving

```
                        // Config the buffer size via infrared receiving
#define RX_RCV_LEN    128
```

```
                        // Config the GPIO register function and multiplexing function of infrared pins
#define IR_GPIO_IN_NUM      14
#define IR_GPIO_IN_MUX      PERIPHS_IO_MUX_MTMS_U
#define IR_GPIO_IN_FUNC     FUNC_GPIO14
```

Other parameters:

`#define USE_US_TIMER` can be defined in `user_config.h`

2.3. Modes of Transmitting Carrier Waveform

2.3.1. Mode 1: IIS Clock Mode

MTMS pin, or GPIO14 is used to transmit carrier waveform under IIS clock mode. Please refer to Figure 1 below.

```
#define GEN_IR_CLK_FROM_IIS      1
#define IR_GPIO_OUT_MUX          PERIPHS_IO_MUX_MTMS_U
#define IR_GPIO_OUT_NUM          14
#define IR_GPIO_OUT_FUNC         FUNC_GPIO14
```

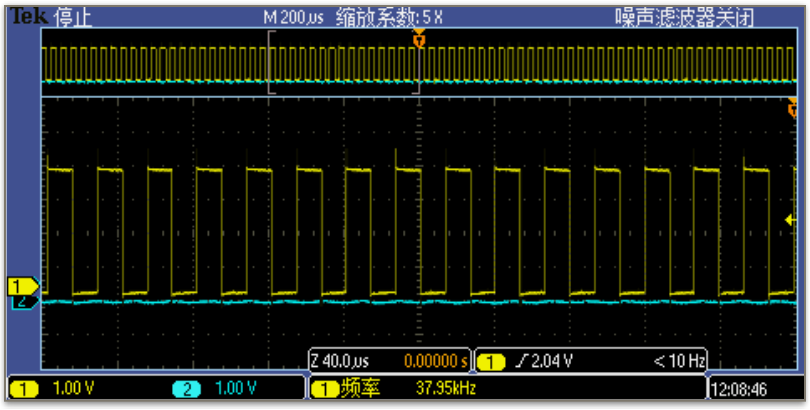

Figure 1 IIS Clock Mode to Transmit Carrier Waveform

2.3.2. Mode 2: Sigma-delta Mode

Any GPIO can be used to transmit carrier waveform under sigma-delta mode. Please refer to Figure 2 below.

```
#define GEN_IR_CLK_FROM_IIS      0
#define IR_GPIO_OUT_MUX          PERIPHS_IO_MUX_GPIO5_U
#define IR_GPIO_OUT_NUM          5
#define IR_GPIO_OUT_FUNC         FUNC_GPIO5
```

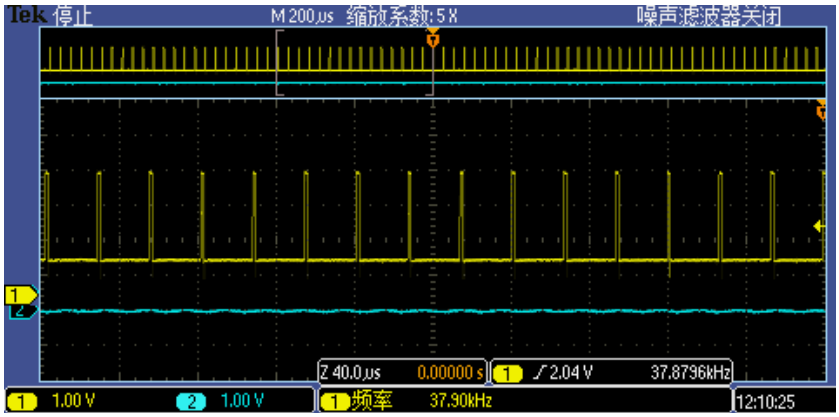

Figure 2 Sigma-delta Mode to Transmit Carrier Waveform

3. Functions of Infrared Sample Codes

The below functions can be implemented using infrared sample codes provided by Espressif Systems:

- Functions of infrared transmitting and receiving can be invoked in the initialization process, and a 4s loop timer can be configured to transmit infrared remote control codes.
- Check the ring buffer of infrared remote control codes simultaneously. If there is any data in the queue, it will be printed out.
- If any carrier waveform in comply with NEC infrared remote control protocol is received by the state machine of infrared receiver, the instruction fields will be stored in the ring buffer of infrared receiving codes.