武汉芯源半导体有限公司
WUHAN XINYUAN SEMICONDUCTOR CO., LTD

CW32W031 Interrupt In Advance Reference

Application note

Rev 1.0

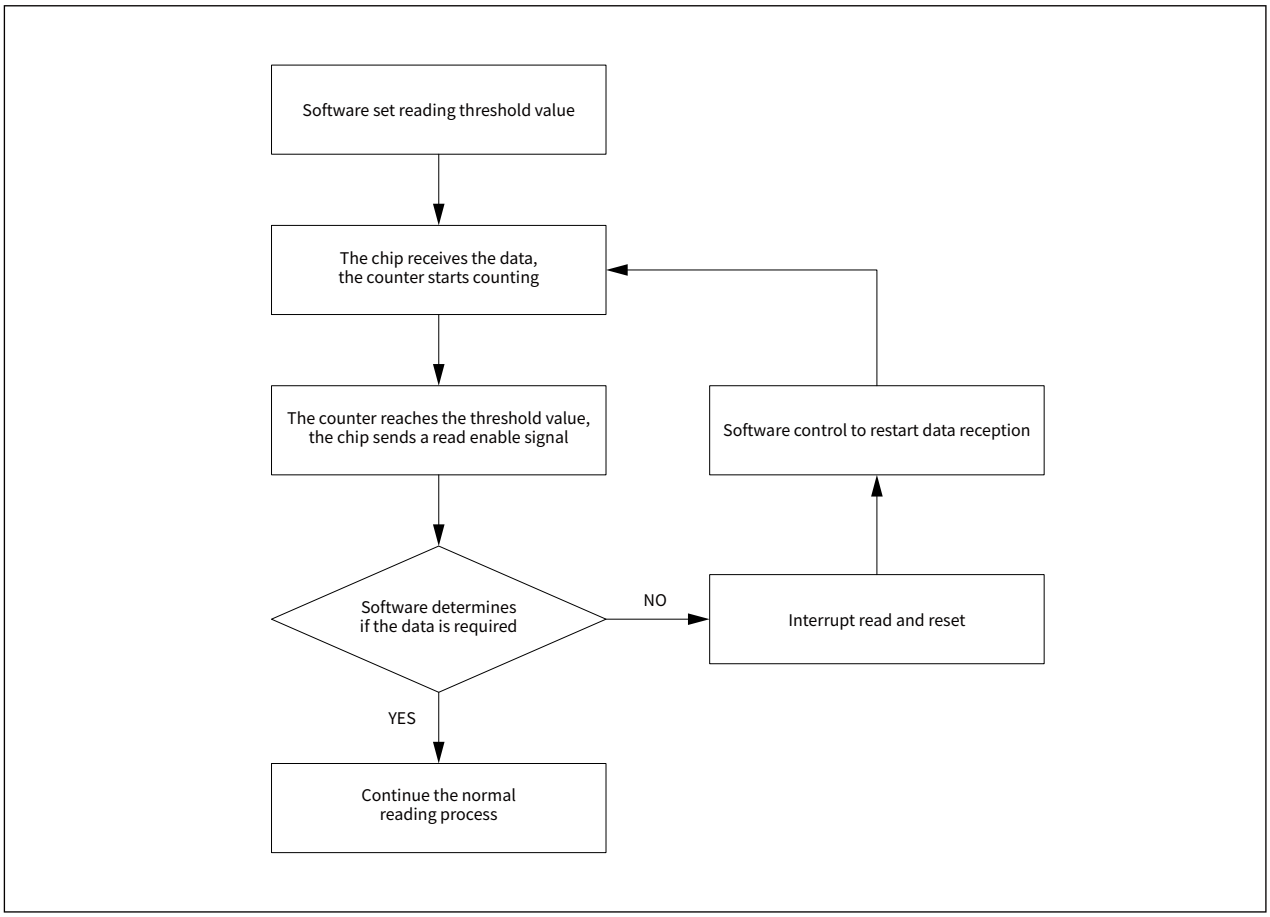
Contents

1 Introduction

The CW32W031 advance interrupt function is used when the chip reads a frame of data, looks at the data that has been parsed, determines if it is what you want, and then decides whether to continue reading or to discard the frame of data.

The flow chart is shown below:

Figure 1-1 Interrupt in advance



2 Software design reference

2.1 Software design process

1. Chip initialization;
2. Configuring the chip to advance interrupt mode;
3. The packet header length, or counter threshold, is set via a register. The advance interrupt function requires setting the number of bytes to be checked from the first and how many bytes of data to check (only 8 bytes or 16 bytes are supported, indicated by PLHD_LEN8/PLHD_LEN16 respectively);
4. The chip enters reception mode;
5. The chip receives the data, the internal counter starts counting and adds 1 to a byte received until the counter reaches the packet header length after which the chip generates an advance interrupt signal for software to read;
6. The software determines if it is the data it wants, if so, it continues to read the data, if not, it stops reading.

2.2 Software design verification

2.2.1 Validation steps

1. The sending module periodically sends 100 bytes of data packets with the first 30 bytes of data being:

```
uint8_t tx_buf[256]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,0,1,2,3,4,5,6,7,8,9};
```

2. The receive module is configured in advance interrupt mode and set to check a total of 16 bytes of data, starting at byte 5;

```
rf_set_plhd_rx_on(5,PLHD_LEN16);
```

3. When the advance interrupt signal is generated, printing the data obtained by the advance interrupt;
4. Continue to receive data and print out all the data in this frame;
5. View the print results via the serial debugging assistant.

2.2.2 SDK Sample

main.c reference code:

```
ret = rf_init();
if(ret != OK)
{
    printf(" RF Init Fail");
    while(1);
}
rf_set_default_para();
rf_set_plhd_rx_on(5,PLHD_LEN16);
rf_enter_continous_rx();
while (1)
{
    if(rf_get_recv_flag() == RADIO_FLAG_PLHDRXDONE)
    {
        rf_set_recv_flag(RADIO_FLAG_IDLE);
        printf("###Plhd Rev :##\r\n");
        for(i = 0; i < RxDoneParams.PlhdSize; i++)
        {
            printf("0x%02x ", RxDoneParams.PlhdPayload[i]);
        }
        printf("\r\n");
    }
    if(rf_get_recv_flag() == RADIO_FLAG_RXDONE)
    {
        rf_set_recv_flag(RADIO_FLAG_IDLE);
        printf("Rx : SNR: %f ,RSSI: %f \r\n", RxDoneParams.Snr, RxDoneParams.Rssi);
        for(i = 0; i < RxDoneParams.Size; i++)
        {
            printf("0x%02x ", RxDoneParams.Payload[i]);
        }
        printf("\r\n");
        cnt ++;
        printf("###Rx cnt %d##\r\n", cnt);
    }
    if((rf_get_recv_flag() == RADIO_FLAG_RXTIMEOUT) || (rf_get_recv_flag() == RADIO_FLAG_
    RXERR))
    {
        rf_set_recv_flag(RADIO_FLAG_IDLE);
        printf("Rxerr\r\n");
    }
}
```

The sample code configures the advance interrupt mode and sets up to check 16 bytes of data starting at byte 5. After receiving the advance interrupt signal, printing out the data received by the advance interrupt in the main function, and continuing to receive; The module then generates a receive interrupt signal, and the main function prints the complete received data content.

If the reception needs to be stopped in advance, simply execute "PAN3028_rst()" after the advance interrupt signal is received, i.e:

```
if(rf_get_recv_flag() == RADIO_FLAG_PLHDRXDONE)
{
    rf_set_recv_flag(RADIO_FLAG_IDLE);
    printf("###Plhd Rev :##\r\n");
    for(i = 0; i < RxDoneParams.PlhdSize; i++)
    {
        printf("0x%02x ", RxDoneParams.PlhdPayload[i]);
    }
    printf("\r\n");
    PAN3028_rst();
}
```

2.2.3 Validation results

The serial debugging assistant shows the following results:

```
###Plhd Rev :##
0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e
0x0f 0x10 0x11 0x12 0x13 0x00
Rx : SNR: 11.464460 ,RSSI: -85.000000
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
###Rx cnt 1##
###Plhd Rev :##
0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e
0x0f 0x10 0x11 0x12 0x13 0x00
Rx : SNR: 8.774908 ,RSSI: -82.000000
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
###Rx cnt 2##
```

According to the results, an advance interrupt occurred in the receive module, the specified data was acquired and the receive was continued and the complete packet was received.

2.3 Logic analyser capture

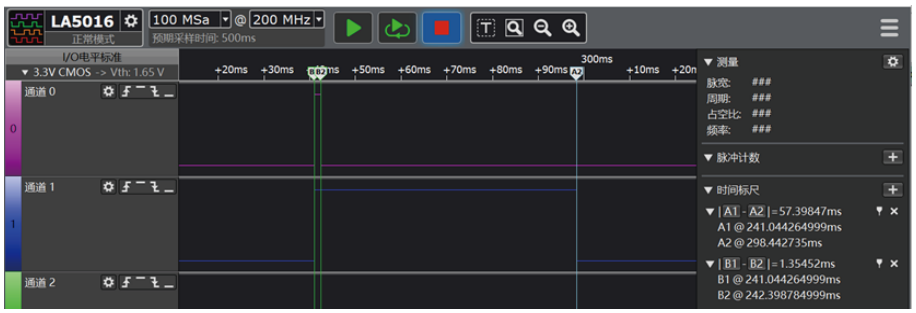
2.3.1 Validation steps

1. The sending module periodically sends data packets;
2. The receive module uses the advance interrupt receive mode and the normal receive mode for reception respectively;
3. Use channel1 of the logic analyser to capture the advance interrupt signal and channel1 to capture the normal receiver signal.

2.3.2 Validation results

The result of the capture is shown in the figure below:

Figure 2-1 Logic analyser capture result



As can be seen from the result, the advance interrupt receive mode generates an advance interrupt at 1.35ms for the user to determine. The normal receive mode takes 57.39ms to generate a full receive interrupt.

2.4 Notice

The advance interrupt function only supports reading two data lengths, i.e. 8 bytes/16 bytes, indicated by PLHD_LEN8/PLHD_LEN16 respectively. No custom parameters can be used.

The advance interrupt function to obtain data uses the PAN3028_plhd_receive() interface function, which differs from the normal packet PAN3028_recv_packet() interface function with a different internal FIFO address.

3 Revision history

Table 3-1 Document revision history

Date	Revision	Changes
May 18, 2023	Rev 1.0	Initial release.