Timer Capture IR

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Use Timer 2 capture function to get IR data
BSP Version	NUC230/240 Series BSP CMSIS v3.01.002
Hardware	NuTiny–EVB–NUC240–LQFP100 V1.0

1 Function Description

1.1 Introduction

This example code demonstrates how to use Timer 2 capture function to get an IR data that is compatible with NEC IR protocol.

1.2 Principle

Figure 1-1 shows the block diagram of this example code, including the NUC240 MCU, IR Receiver Module and IR Encoder/Transmitter.

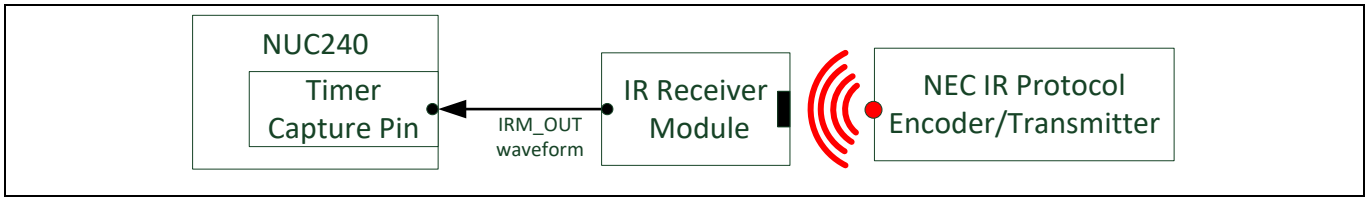


Figure 1-1 Block Diagram

The NEC Infrared Protocol (NEC IR) features are:

1. Started by a Leader signal and its (9 ms low duration + 4.5 ms high duration)
2. 8-bit Address code, and then 8-bit Address Inverse code
3. 8-bit Command code, and then 8-bit Command Inverse code
4. Last 562.5 us low duration is a Stop signal of this IR transfer

Figure 1-2 shows the NEC IR protocol (NEC_IR) and the output waveform (IRM_OUT) after the IR Receiver Module receives the signal.

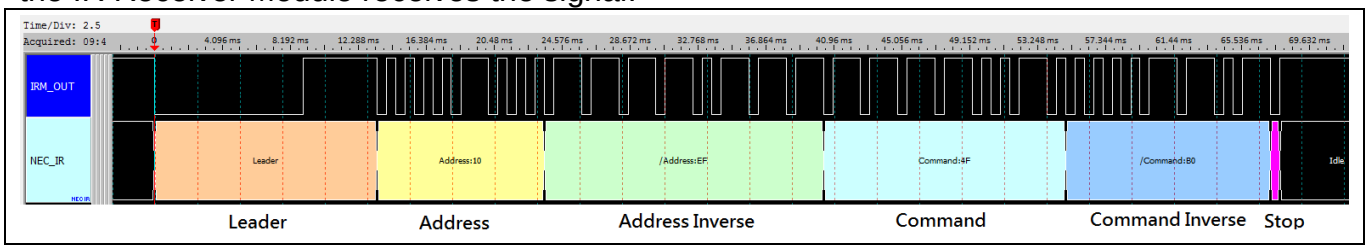


Figure 1-2 NEC IR Protocol and IR Receiver Module Waveform

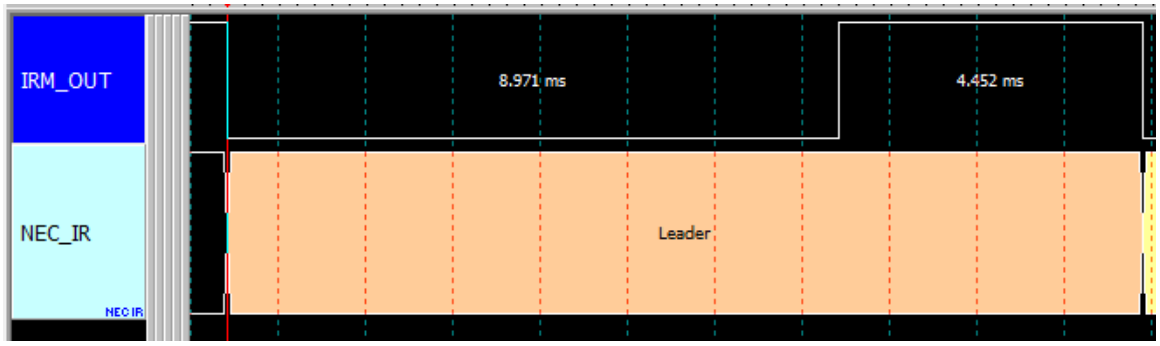
The value of each bit in the Address/Command is determined by the state and length of the received waveform (IRM_OUT).

- Logic 0: 562.5 us low duration + 562.5 us high duration, total 1125 us
- Logic 1: 562.5 us low duration + 1687.5 us high duration, total 2250 us

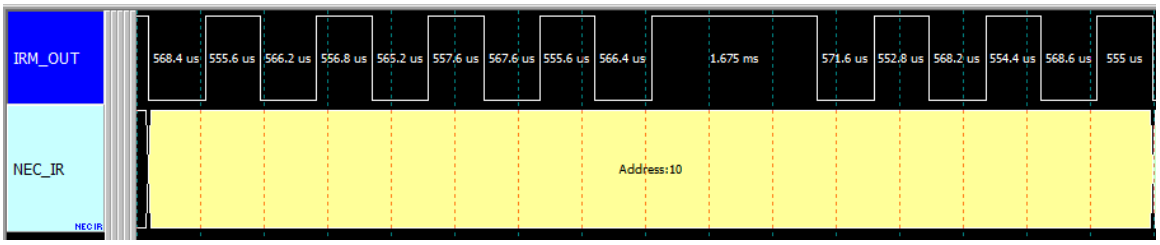
According to the above description and figure, when the IR Receiver Module receives the IR signal, a falling edge-triggered is generated. As long as the user can capture two consecutive falling edge-triggered intervals, it is possible to know the incoming signal from this interval. Therefore, this example code uses the capture function of the NUC240 Timer to get the consecutive falling edge-triggered interval received by the IR Receiver Module.

According to the Figure 1-2 , each part of the NEC IR protocol is described as follows:

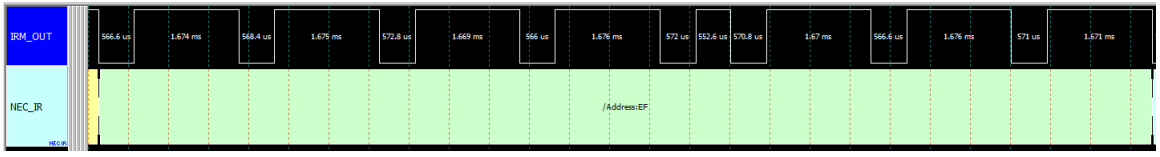
- Leader signal interval



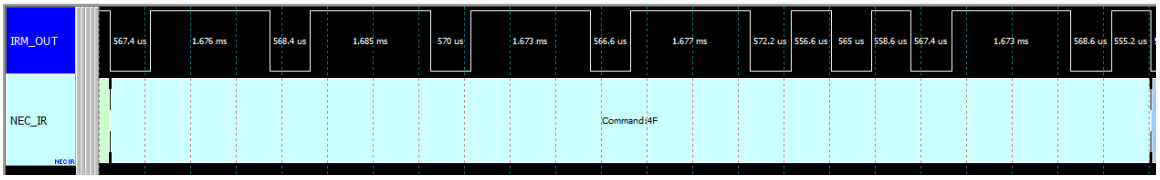
- Address code (0x10)



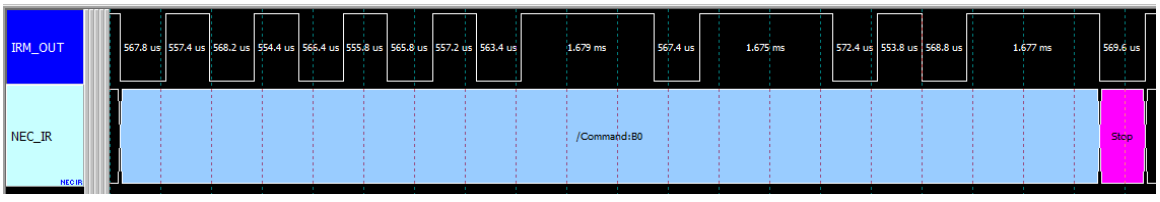
- Address Inverse code (0xEF)



- Command code (0x4F)



- Command Inverse code (0xB0) and Stop signal interval



1.3 Demo Result

The execution result can be output on UART, baud rate 115200, as shown in Figure 1-3 on PC.

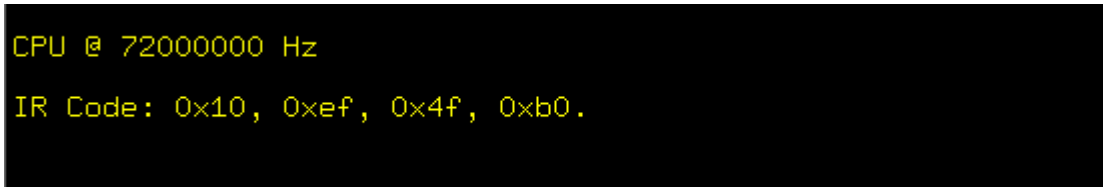


Figure 1-3 Execution Result on UART

2 Code Description

Declare the variables for storing the falling edge interval and its index in the capturing operation.

```
#define IR_FRAME_TOUT       (100000) // Received time-out interval: 100 ms

volatile uint32_t gu32IRINRIdx = 0;     // To indicate the interval index of IR frame
volatile uint32_t gu32IsIRINRDone = 0;  // To indicate an IR frame is completed or not
volatile uint32_t gau32IRINR[33] = {0}; // To save all intervals of IR frame
```

Implement the TMR2_IRQHandler() for capturing the falling edge interval of PB.2 by Timer 2 capture function.

```
void TMR2_IRQHandler(void)
{
    uint32_t interval;

    /* To check if receiving IR signal interval has time-out */
    if (TIMER_GetIntFlag(TIMER2) == 1)
    {
        /* IR received has time-out and reset the receive state */
        gu32IRINRIdx = 0;
        gu32IsIRINRDone = 0;
        TIMER_ClearIntFlag(TIMER2);
        return ;
    }

    /* To capture the two falling edge-triggered interval */
    if (TIMER_GetCaptureIntFlag(TIMER2) == 1)
    {
        /* Clear Timer2 capture interrupt flag */
        TIMER_ClearCaptureIntFlag(TIMER2);

        if (PB2 == 0) // capture pin status
        {
            /* Reload TIMER2 counter to 0 */
            TIMER_SET_CMP_VALUE(TIMER2, IR_FRAME_TOUT);

            if (gu32IRINRIdx == 0)
            {
```

```
                /* It's starting signal to capture the 1st interval - "Leader" */
                gu32IRINRIdx++;
            }
            else
            {
                /* Save to interval to gau32IRINR[] */
                interval = TIMER_GetCaptureData(TIMER2);
                //printf("INR: %d(%d)\n", interval, gu32IRINRIdx);
                gau32IRINR[(gu32IRINRIdx - 1)] = interval;
                gu32IRINRIdx++;

                if (gu32IRINRIdx > 33)
                {
                    /* Get a complete IR frame interval data */
                    gu32IRINRIdx = 0;
                    gu32IsIRINRDone = 1;
                }
            }
        }
        else
        {
            /* Reset the receive state */
            gu32IRINRIdx = 0;
            gu32IsIRINRDone = 0;
            printf("\nERROR: unexpected capture event !!!\n\n");
        }
    }
}
```

In main.c, the user needs to initialize the Timer 2 capture function and configure a Timer 2 counter with a counter unit is 1 us. After executing TIMER_Start(), the signal interval received by the IR Receiver Module can be captured in TMR2_IRQHandler( ).

```
    ……
    /* Initial Timer2 default setting */
    TIMER_Open(TIMER2, TIMER_PERIODIC_MODE, 1);

    /* Configure Timer2 setting for external capture function, and one Timer2 counter is 1
us. */
    TIMER_SET_PRESCALE_VALUE(TIMER2, ((SystemCoreClock / 1000000) - 1));
    TIMER_SET_CMP_VALUE(TIMER2, IR_FRAME_TOUT);
```

```
    TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_FREE_COUNTING_MODE,
TIMER_CAPTURE_FALLING_EDGE);
    TIMER_EnableCaptureInt(TIMER2);

    /* Enable Timer2 NVIC */
    NVIC_EnableIRQ(TMR2_IRQn);

    /* Start Timer2 capture function */
    TIMER_Start(TIMER2);
    ……
```

After receiving a complete set of IR signal interval data, the meaning of all intervals can be resolved in the main.c to obtain the actual IR code, including the Address code, Address Inverse Code, Command Code and Command Inverse Code.

```
    ……
    while (1)
    {
        /* In NEC IR protocol, an IR frame format consists of
            [Leader interval] +
                [8-bit Address] + [8-bits Address Inverse] +
                [8-bit Command] + [8-bit Command Inverse] +
            [Stop interval]
        */
        /* Get one IR frame data */
        if (gu32IsIRINRDone)
        {
            /* Check if is valid "Leader" signal, 13 ~ 14 ms*/
            if ((gau32IRINR[0] > 13000) && (gau32IRINR[0] < 14000))
            {
#if 0 // enable for debugging

                for (i = 0; i < 33; i++)
                {
                    printf("INR: %d (%d)\n", gau32IRINR[i], i);
                }

                printf("---------------\n\n");
#endif

                /* Parse IR code[0] */
                offset = 1;
```

```
    for (i = 0; i < 8; i++)
    {
        interval = gau32IRINR[(i + offset)];

        if (interval > 2000)
            au8IRCode[0] |= (1 << i);  // get data "1"
        else
            au8IRCode[0] |= (0 << i);  // get data "0"
    }

    /* Parse IR code[1] */
    offset += 8;

    for (i = 0; i < 8; i++)
    {
        interval = gau32IRINR[(i + offset)];

        if (interval > 2000)
            au8IRCode[1] |= (1 << i);  // get data "1"
        else
            au8IRCode[1] |= (0 << i);  // get data "0"
    }

    /* Parse IR code[2] */
    offset += 8;

    for (i = 0; i < 8; i++)
    {
        interval = gau32IRINR[(i + offset)];

        if (interval > 2000)
            au8IRCode[2] |= (1 << i);  // get data "1"
        else
            au8IRCode[2] |= (0 << i);  // get data "0"
    }

    /* Parse IR code[3] */
    offset += 8;

    for (i = 0; i < 8; i++)
```

```
                {
                    interval = gau32IRINR[(i + offset)];

                    if (interval > 2000)
                        au8IRCode[3] |= (1 << i);  // get data "1"
                    else
                        au8IRCode[3] |= (0 << i);  // get data "0"
                }
            }

            printf("IR Code: 0x%x, 0x%x, 0x%x, 0x%x.\n",
                   au8IRCode[0], au8IRCode[1], au8IRCode[2], au8IRCode[3]);

            /* Clear au8IRCode[] data */
            for (i = 0; i < 4; i++)
                au8IRCode[i] = 0x0;

            /* Clear gau32IRINR[] data */
            for (i = 0; i < 33; i++)
                gau32IRINR[i] = 0x0;

            /* Reset the receive state */
            gu32IRINRIdx = 0;
            gu32IsIRINRDone = 0;
        }
    }
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVersion 5.28

Hardware Environment

- Circuit component
  - NuTiny–EVB–NUC240–LQFP100 V1.0
  - IR module, IRM-2638
- Diagram
  - Connect the IR Receiver Module output signal to the NUC240 Timer 2 capture pin (PB.2) to capture the IR signal interval.
  - Connect UART TX (PB.1) pin to PC UART RX to display the execution result of example code on PC.

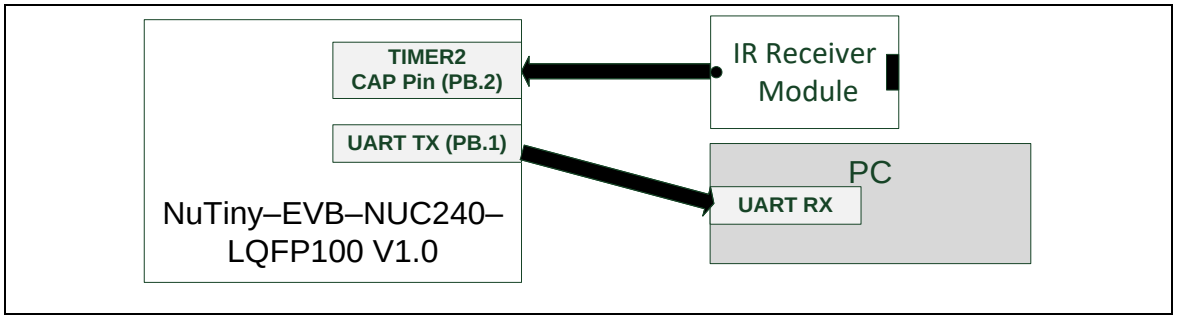


Figure 3-1 Connect NUC240 Pins to IR Module Output and PC UART RX

4 Directory Information

- EC_NUC240 _TIMER_Capture_IR_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click TIMER_Capture_IR.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Sep. 18, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.