NUC505 线控耳麦的实际应用

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	此应用文档是在说明如何让嵌入式软件开发工程师，在 NUC505 系列 USB UAC 的应用中，能够快速导入一般有线耳机麦克风 (Headset) 上控制键的入门应用。
Apply to	NuMicro® NUC505 series.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 大纲 4

1.1 耳麦接头的种类 4

1.2 四段式接头的定义 5

1.3 四段式接头耳麦原理图 6

1.4 单键式线控耳机麦克风 7

1.5 三键式线控耳机麦克风 8

2 NUC505 TINY BOARD硬件更改说明 9

2.1 NUC505与线控耳麦的应用 9

2.2 原开发板上耳机与麦克风电路图 10

2.3 更改后耳机与麦克风电路图 11

3 软件更改说明 12

3.1 主程序新增部分 12

3.2 ADC初始化与其中断子程序 13

3.3 Timer0初始化与其中断子程序 15

3.4 Keyboard Update子程序 16

3.5 Keyboard Report描述符 18

3.6 测试结果 19

4 修订历史 20

List of Figures

Figure 1-1常见耳机接头示意图 4
Figure 1-2四段耳机麦克风的定义图 5
Figure 1-3四段式接头耳麦原理图 6
Figure 1-4单键式线控耳机麦克风原理图 7
Figure 1-5三键式线控耳机麦克风原理图 8
Figure 2-1 NUC505与线控耳麦应用示意图 9
Figure 2-2原开发板上耳机与麦克风电路图 10
Figure 2-3更改后开发板耳机与麦克风电路图 11

1 大纲

此应用说明文档，是以 NUC505 系列为主控制器来开发线控耳机麦克风的相关应用当使用者按下线控耳机麦克风的键时，NUC505 ADC 输入端侦测到线控耳麦的麦克风输入端 DC 值的变化而得知是哪个键被按下，并透过 USB HID 类别(Keyboard Report) 实时回报给 USB Host，来控制接听/挂断(拨放/暂停)，音量增加(上一首)，音量降低(下一首)的应用参考。

1.1 耳麦接头的种类

Figure 1-1显示常见两段式，三段式，四段式纯耳机或耳机麦克风接头的示意图。

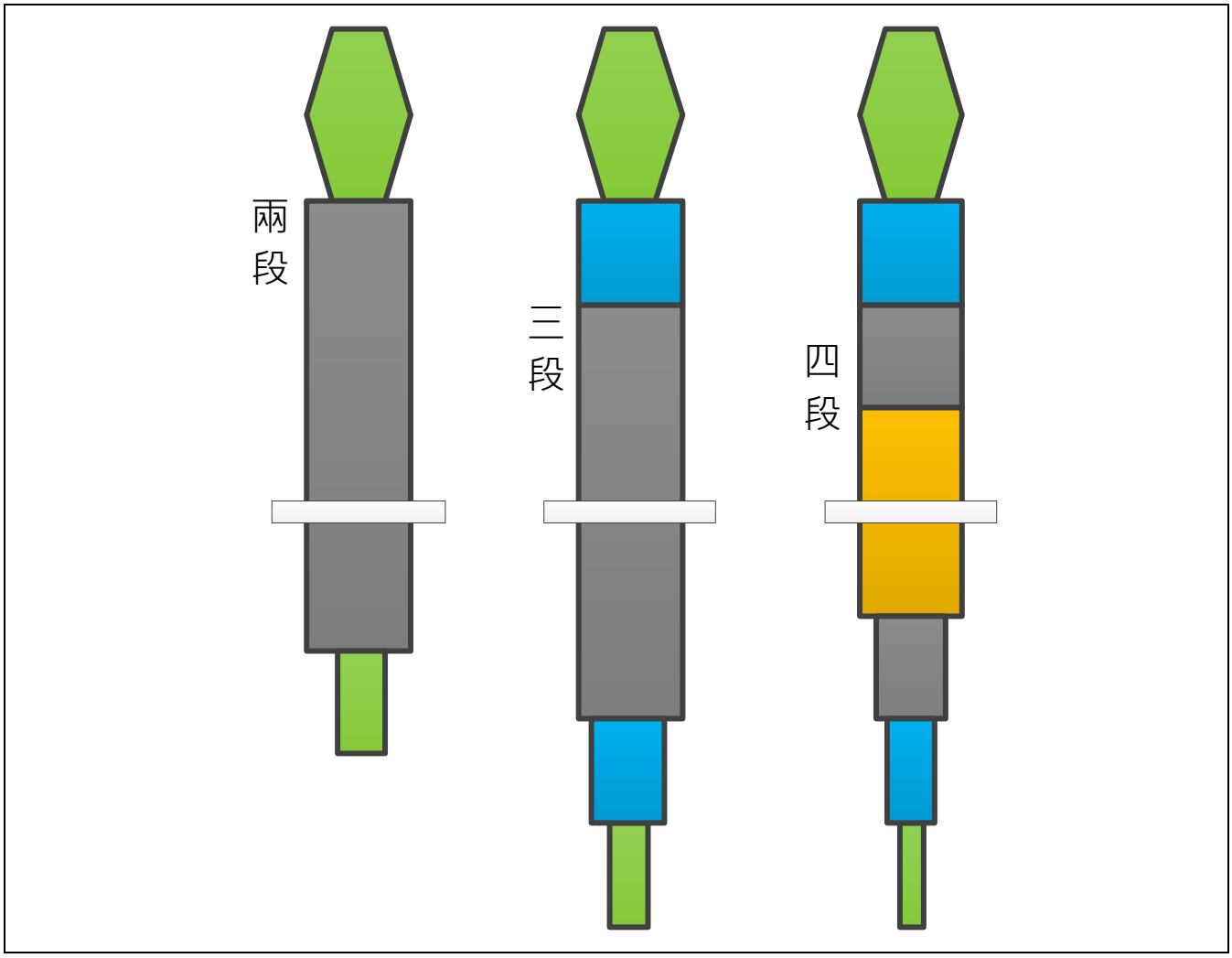
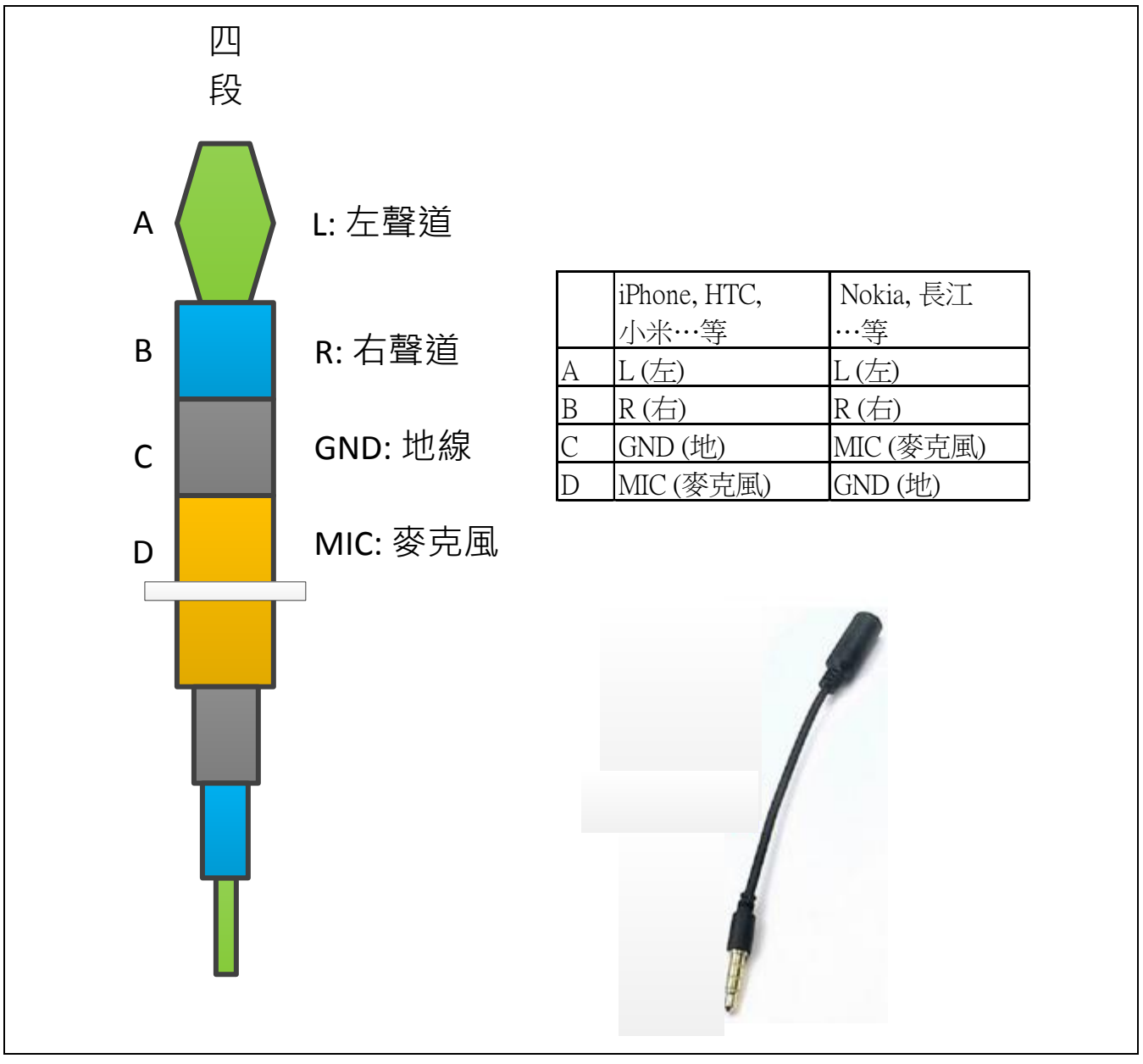


Figure 1-1常见耳机接头示意图

1.2 四段式接头的定义

Figure 1-2 所示是一般较常见的耳机麦克风四段式接头的定义，但不同厂商的所制造的耳机麦克风接头定义皆有所不同，如图中的表格所示，因此有些厂商会附加另一条转接线将 C 和 D 点对调，以适用厂牌手机。



	iPhone, HTC, 小米…等	Nokia, 長江 …等
A	L (左)	L (左)
B	R (右)	R (右)
C	GND (地)	MIC (麥克風)
D	MIC (麥克風)	GND (地)

Figure 1-2四段耳机麦克风的定义图

1.3 四段式接头耳麦原理图

Figure 1-3 显示一般四段式接头的耳机麦克风原理图。

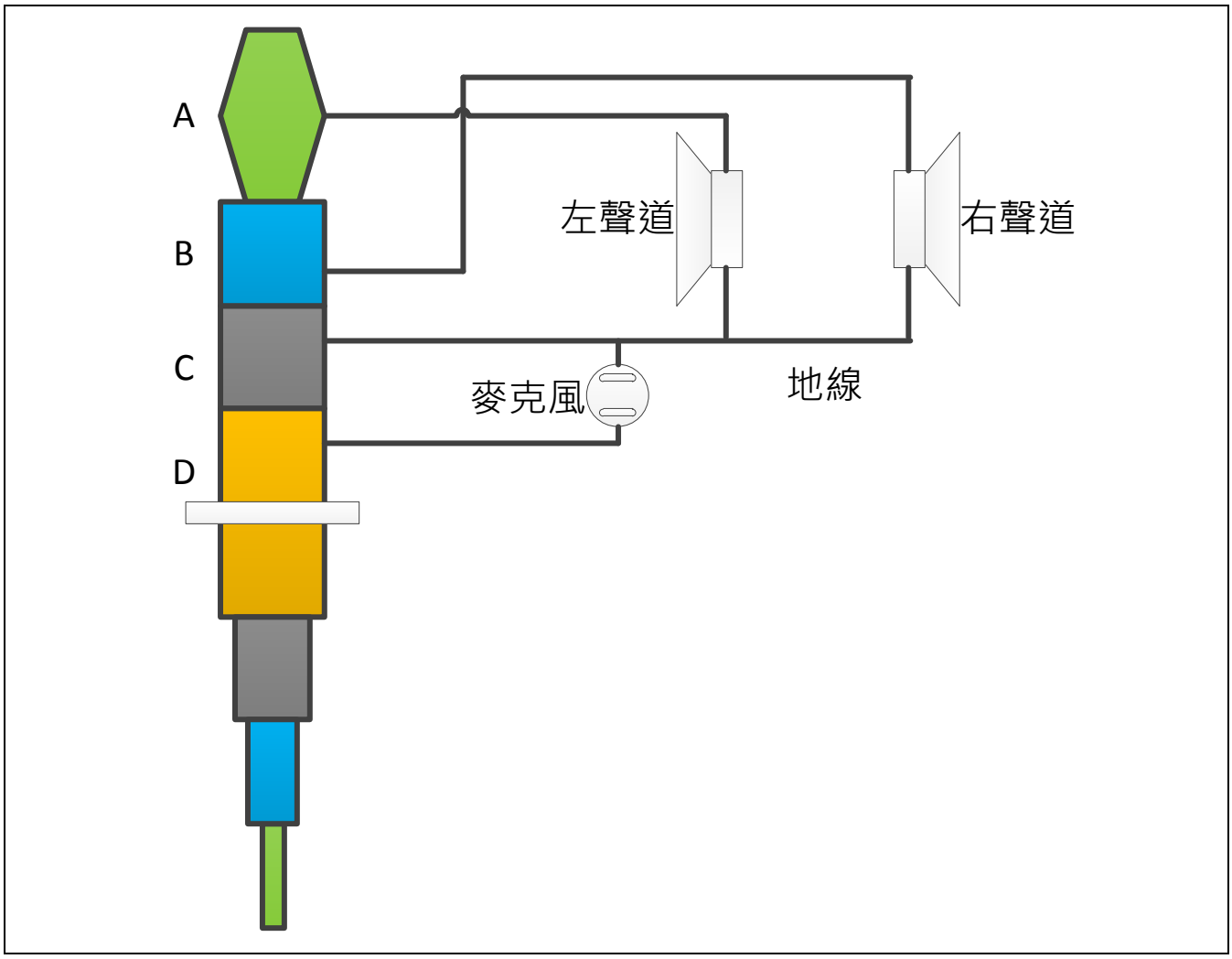


Figure 1-3 四段式接头耳麦原理图

1.4 单键式线控耳机麦克风

一般市面上销售单键式线控耳机麦克风的原理图如 Figure 1-4，此单键多为来电接听或挂断的功能。

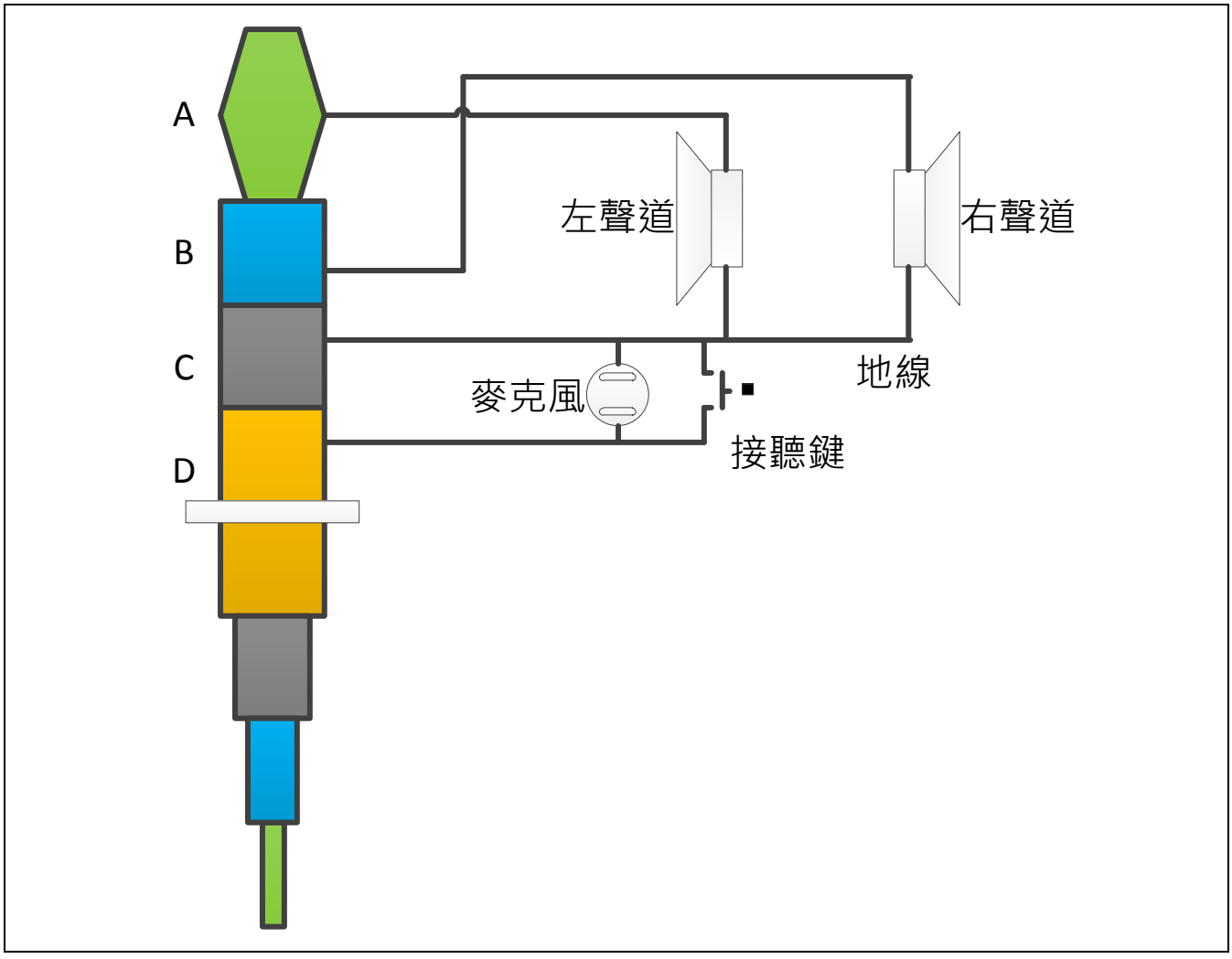


Figure 1-4 单键式线控耳机麦克风原理图

1.5 三键式线控耳机麦克风

一般市面上销售三键式线控耳机麦克风的原理图如 Figure 1-5，此三键的功能分别为来电接听或挂断(播放或暂停)，音量增加(或上一首)，音量减少(或下一首)，这些功能之所以不同，通常取决于各家手机的设定，使用者可以在手机上利用 APP 来做这些键的功能调整。

在图中，一般音量增加键的 R1 建议值约在 200 欧姆，而音量减少键的 R2 建议值约在 600 欧姆，例如小米耳麦 R1 值是 218 欧姆，R2 值是 579 欧姆。

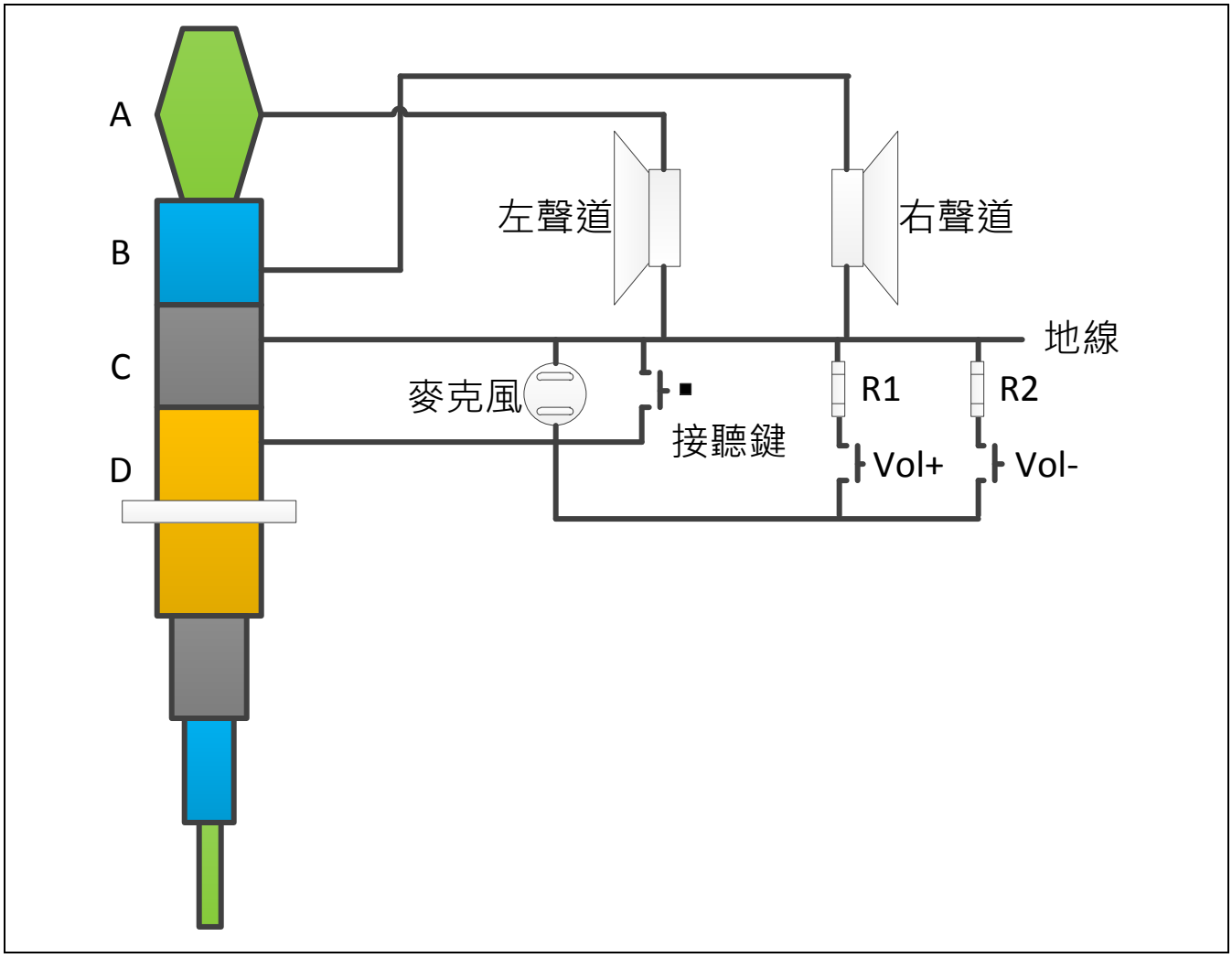


Figure 1-5三键式线控耳机麦克风原理图

2 NUC505 Tiny Board硬件更改说明

2.1 NUC505 与线控耳麦的应用

Figure 2-1 是 MCU NUC505 与四段式接头耳麦应用连接示意图，耳麦上的麦克风的输入除了接到 NUC505 的麦克风的输入管脚 MIC1P 外，还须将其拉至 ADC 的输入管脚，在此应用说明中是使用了ADC 通道 2 的输入脚来做 ADC 的转换，以识别耳麦上何键被按下。

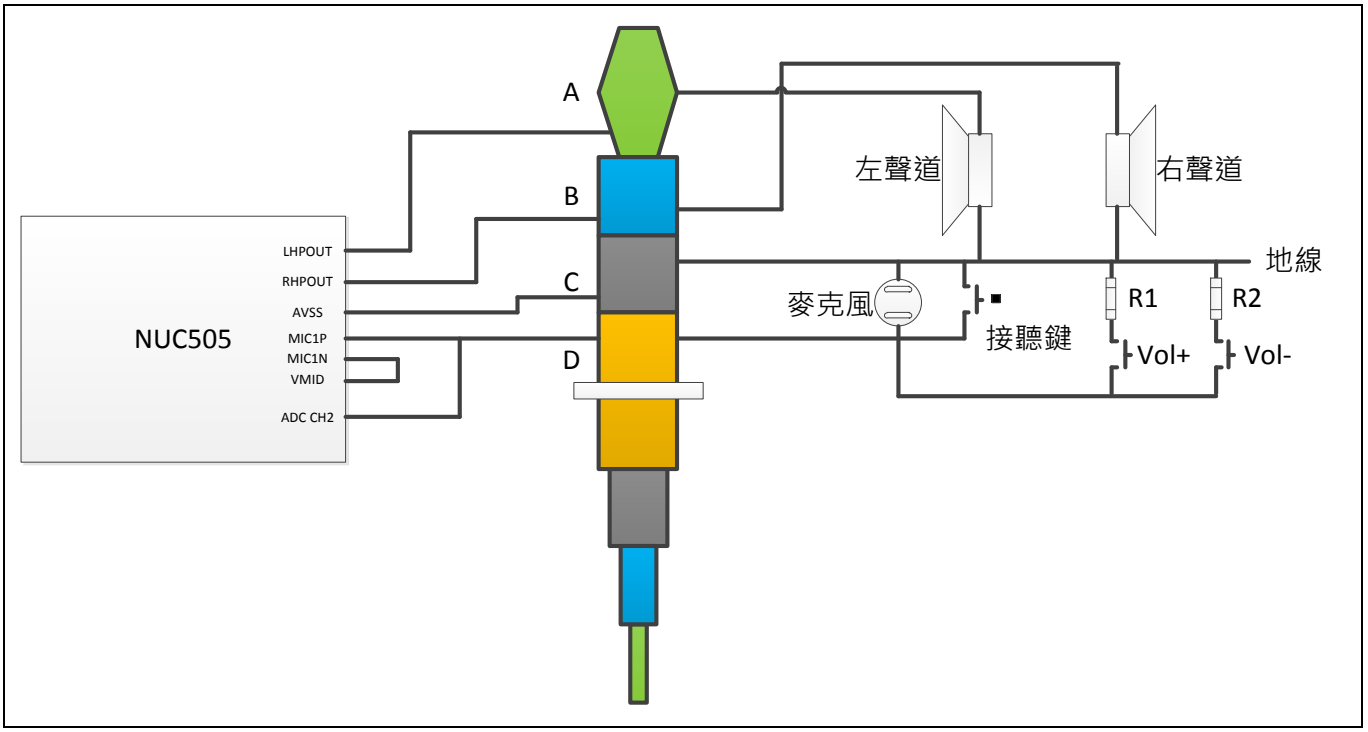


Figure 2-1 NUC505与线控耳麦应用示意图

2.2 原开发板上耳机与麦克风电路图

原本 NUC505 开发板子(NuTiny-EVB-NUC505_QFN88)上耳机与麦克风的电路图如 Figure 2-2 所示，麦克风的电路是双端输入(MIC1P 与 MIC1N)，而耳机部分是三段式接头的设计，板子的电路无法连接到线控耳麦的麦克风头，而且开发板上原先 JP1 因为板子出厂预设是 LINE IN 的偏压，所以 JP1 的端点 1 与 2 是短路的。

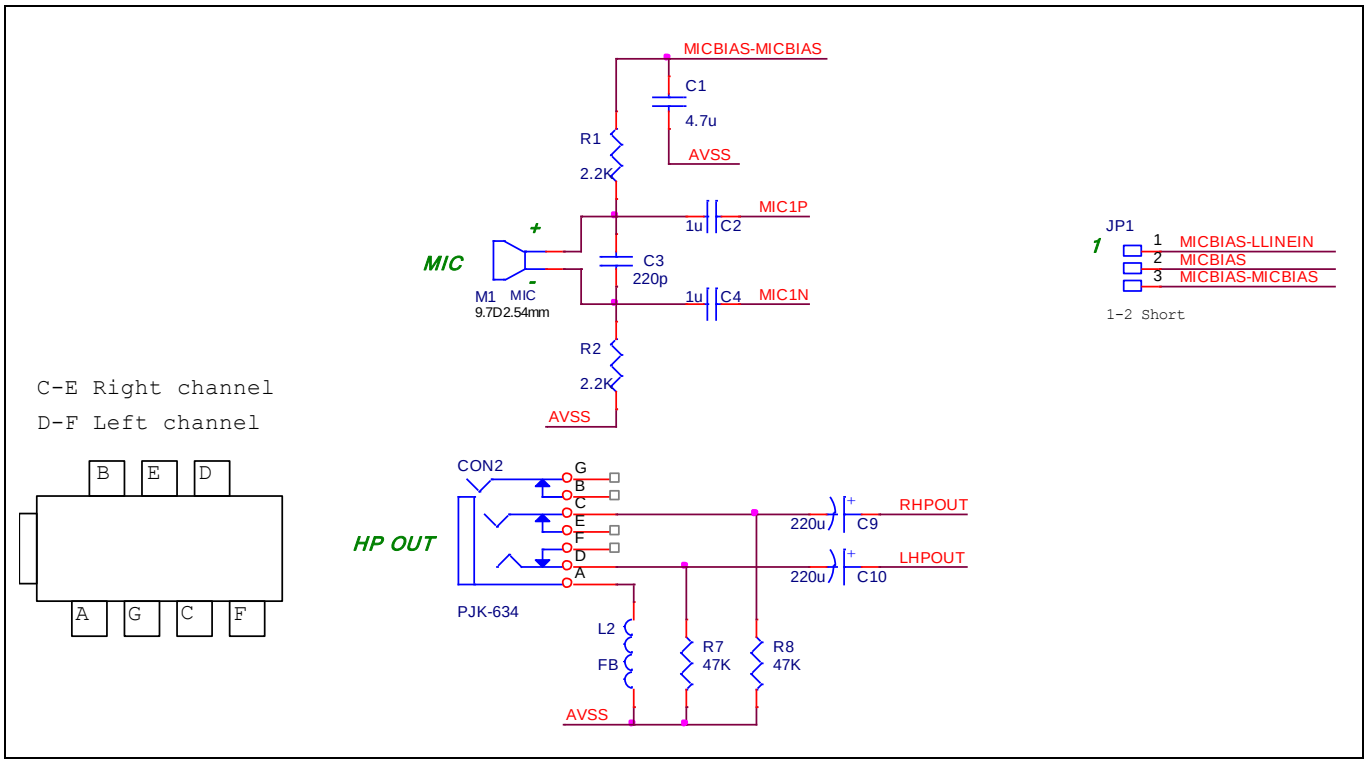


Figure 2-2原开发板上耳机与麦克风电路图

2.3 更改后耳机与麦克风电路图

Figure 2-3 是开发板更改后耳机与麦克风的电路图，麦克风的电路改成单端输入(图中第 1~5 点)，而耳机部分需改成四段式接头的设计(图中第 6~7 点)，把线控耳麦的麦克风输入信号，透过耳机孔(CON2)上的A点接到开发板上来当作麦克风的输入源，最后需要再把 MIC_IN 连接到 P71 (图中第 8 点)，P71 是 MCU NUC505-QFN88 的管脚 71，它是 GPIO PA.2，也是 ADC 通道 2 的输入脚，当使用者按下线控耳麦上的任一个键时，MIC_IN 点的 DC 值也会跟着变化，就可以利用 ADC 转换后值的不同大小来判断是哪个键被按下，进而透过 USB HID 类别 (Keyboard Report) 回报给 Host 来做计算机或手机端的操控。

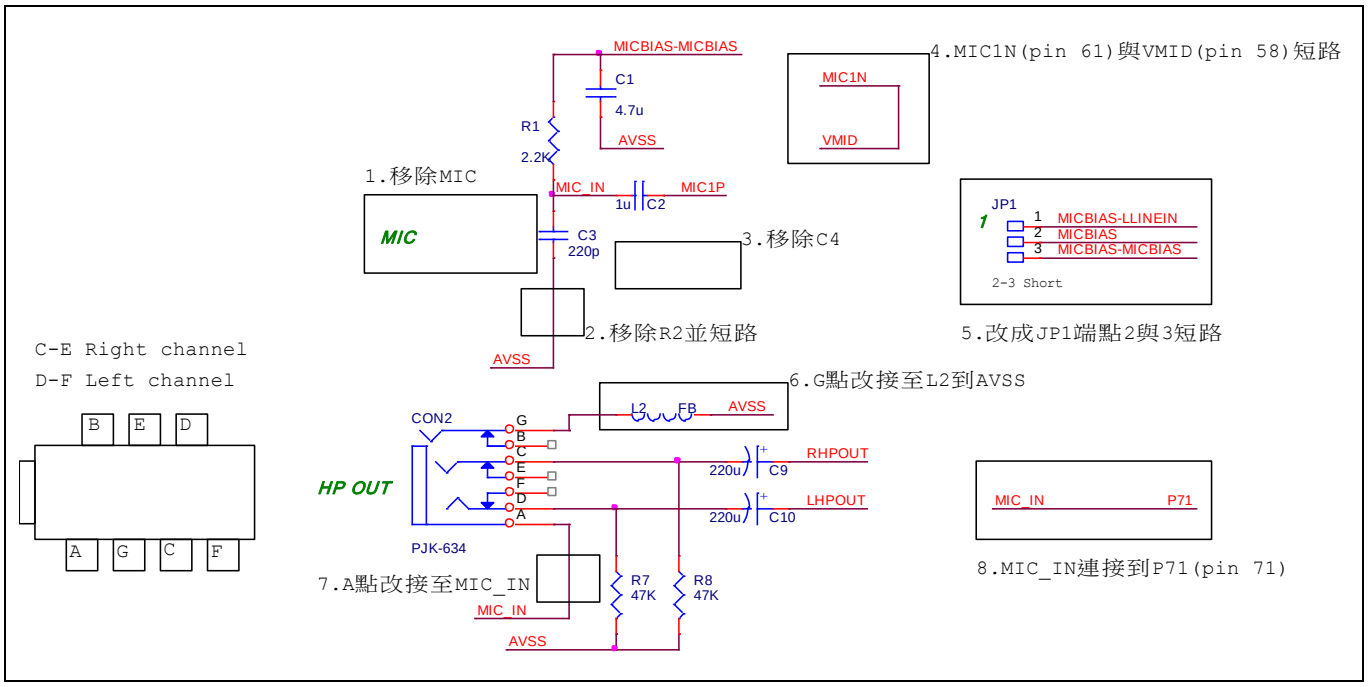


Figure 2-3更改后开发板耳机与麦克风电路图

3 软件更改说明

3.1 主程序新增部分

主程序更改部分主要增加了 ADC 和 Timer0 的初始化，以及在 While(1) loop 无穷循环里增加了以 Timer0 的中断事件来触发 ADC 开始做转换，并判断 ADC 转换后的值来界定是否有按下线控耳麦上的任一个按键，最后透过 USB HID 类别的 Keyboard Report 来回报给 USB Host。

```
volatile uint8_t u8ADF = 0;
volatile uint8_t PP_Key = 0;
volatile uint8_t VI_Key = 0;
volatile uint8_t VD_Key = 0;

/*---------------------------------------------------------------------------------------------*/
/*  Main Function                                                                              */
/*---------------------------------------------------------------------------------------------*/
int32_t main(void)
{
    …
    …

    /* Init ADC Channel 2 (PA.2) */
    ADC_Init();

    /* Enable Timer0 NVIC */
    Timer0_Init();

    …
    …

    while(1)
    {
        /* Trigger ADC to converter */
        if(TM0_INT){
           TM0_INT = 0;
           ADC_EnableHWTrigger(ADC, ADC_CH_2_MASK, 0x20);
         }
         else{
           if(u8ADF){
```

```
            u8ADF = 0;
            u16Data = ADC_GET_CONVERSION_DATA(ADC, NULL);
            printf("%d\n", u16Data);
            if(u16Data == 0)
               PP_Key = 1;
            else if((u16Data > 0) && (u16Data <= 400))
               VI_Key = 1;
            else if((u16Data > 400) && (u16Data <= 800))
               VD_Key = 1;
            else{
               PP_Key = 0;
               VI_Key = 0;
               VD_Key = 0;
            }
         }
       }
      …
      …

       /* Update Key Code */
        HID_UpdateKbData();

      …
      …

   }
}
```

3.2 ADC 初始化与其中断子程序

下面是 ADC 的初始化子程序，以及 ADC 的中断服务程序，本范例只开启 ADC 通道 2 (GPIO PA.2) 作 ADC 的输入管脚，当 ADC 转换成功后即发生 ADC 的中断，中断服务程序里设置了 u8ADF 的标志。

```
void ADC_Init(void)
{
    /* Enable ADC Module clock */
    CLK_EnableModuleClock(ADC_MODULE);
```

```
    /* ADC module clock from EXT */
    CLK_SetModuleClock(ADC_MODULE, CLK_ADC_SRC_EXT, 4); /* 12M/16/(4+1) = 150000 */

    /* Reset ADC */
    SYS_ResetModule(ADC_RST);

    /* Open ADC */
    ADC_Open(ADC, NULL, NULL, ADC_CH_2_MASK);

    /* Enable ADC ADC_IF interrupt */
    ADC_EnableInt(ADC, ADC_ADF_INT);
    NVIC_EnableIRQ(ADC_IRQn);

    /*---------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                         */
    /*---------------------------------------------------------------------------------*/
    /* Configure multi-function pins for AIN2 */
    SYS->GPA_MFPL = (SYS->GPA_MFPL & (~SYS_GPA_MFPL_PA2MFP_Msk)) |
SYS_GPA_MFPL_PA2MFP_ADC_CH2;

    /* Set analog shared pin input type */
    SYS_SetSharedPinType(SYS_PORT_A, SYS_PIN_2, 0, 0);

}

void ADC_IRQHandler(void)
{
    uint32_t u32Flag;

    /* Get ADC conversion finish interrupt flag */
    u32Flag = ADC_GET_INT_FLAG(ADC, ADC_ADF_INT);

    if(u32Flag & ADC_ADF_INT)
    {
        u8ADF = 1;
        ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT);
    }

    /* To avoid the the synchronization issue between system and APB clock domain */
    u32Flag = ADC_GET_INT_FLAG(ADC, ADC_ADF_INT);
}
```

3.3 Timer0 初始化与其中断子程序

下面是 Timer0 的初始化子程序，以及 Timer0 的中断服务程序，Timer0 的设置是每秒发生 10 次中断，亦即每 100ms 发生一次中断，在 Timer0 的中断服务程序里设置了 TM0_INT 的标志。

```
volatile uint32_t TM0_INT = 0;

void Timer0_Init(void)
{
    /* Enable Timer0 clock */
    CLK_EnableModuleClock(TMR0_MODULE);

    /* Timer0 module clock from EXT */
    CLK_SetModuleClock(TMR0_MODULE, CLK_TMR0_SRC_EXT, 0);

   /* Open Timer0 in periodic mode, enable interrupt and 10 interrupt tick per second
*/
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 10);
    TIMER_EnableInt(TIMER0);

    /* Enable Timer0 NVIC */
    NVIC_EnableIRQ(TMR0_IRQn);

    /* Start Timer0 counting */
    TIMER_Start(TIMER0);

}

void TMR0_IRQHandler(void)
{
    if(TIMER_GetIntFlag(TIMER0) == 1)
    {
        /* Clear Timer0 time-out interrupt flag */
        TIMER_ClearIntFlag(TIMER0);

        TM0_INT = 1;
    }
}
```

3.4 Keyboard Update 子程序

当 ADC 转换后，判断其值并确认线控耳麦上有键被按下时，USB Device 需要实时更新键值，并回报给 USB Host 来做相对应的操作。

```
#define HID_CTRL_MUTE         0x01
#define HID_CTRL_VOLUME_INC   0x02
#define HID_CTRL_VOLUME_DEC   0x04

#define HID_CTRL_EJECT        0x08
#define HID_CTRL_PLAY         0x01
#define HID_CTRL_STOP         0x02
#define HID_CTRL_PAUSE        0x04
#define HID_CTRL_NEXT         0x08
#define HID_CTRL_PREVIOUS     0x10
#define HID_CTRL_RECORD       0x20
#define HID_CTRL_REWIND       0x40
#define HID_CTRL_FF           0x80

void HID_UpdateKbData(void)
{
    int32_t n;
    int32_t volatile i;
    uint32_t key = 0xF;

    n = 8;
    if(g_u8EPCReady)
    {
        if(USBD->EP[EPC].EPDATCNT & 0xFFFF)
            return;

        key = PP_Key | VI_Key | VD_Key;

        if(key == 0)
        {
            for(i = 0; i < n; i++)
                g_buf[i] = 0;
        }
        else
        {
```

```
            if(PP_Key){
                PP_Key = 0;
                g_buf[1] = HID_CTRL_PAUSE;          /* Play/Pause */
            }else if(VI_Key){
                VI_Key = 0;
                g_buf[0] = HID_CTRL_VOLUME_INC;     /* Vol + */
            }else if(VD_Key){
                VD_Key = 0;
                g_buf[0] = HID_CTRL_VOLUME_DEC;     /* Vol - */
            }
        }

        /* Set transfer length and trigger IN transfer */
        while (1)
        {
            if (!(USBD->DMACTL & USBD_DMACTL_DMAEN_Msk))
                  break;

            if (!USBD_IS_ATTACHED())
                  break;
        }

        USBD_SET_DMA_READ(HID_IN_EP_NUM);
        USBD_SET_DMA_ADDR((uint32_t)&g_buf[0]);
        USBD_SET_DMA_LEN(8);
        USBD_ENABLE_DMA();
        while (1)
        {
            if (!(USBD->DMACTL & USBD_DMACTL_DMAEN_Msk))
                  break;

            if (!USBD_IS_ATTACHED())
                  break;
        }
        USBD->EP[EPC].EPRSPCTL = USB_EP_RSPCTL_SHORTTXEN;
        g_u8EPCReady = 0;
        return;
    }
}
```

3.5 Keyboard Report 描述符

下面是多媒体键盘的报告描述符，但本范例只有用到 Play/Pause，Volume Increment 和 Volume Decrement 的 usages。

```
__align(4)  uint8_t gu8KeyboardReportDesc[] =
{
    0x05, 0x0C, // Usage Page (Consumer)
    0x09, 0x01, // Usage(Consumer Control)
    0xA1, 0x01, // Collection(Application )
    0x15, 0x00, // Logical Minimum(0x0 )
    0x25, 0x01, // Logical Maximum(0x1 )
    0x09, 0xE2, // Usage(Mute)
    0x09, 0xE9, // Usage(Volume Increment)
    0x09, 0xEA, // Usage(Volume Decrement)
    0x75, 0x01, // Report Size(0x1 )
    0x95, 0x03, // Report Count(0x3 )
    0x81, 0x02, // Input(Data, Variable, Absolute, No Wrap, Linear, Preferred State,
No Null Position, Bit Field)
    0x75, 0x01, // Report Size(0x1 )
    0x95, 0x05, // Report Count(0x5 )
    0x81, 0x03, // Input(Constant, Variable, Absolute)
    0x09, 0xB0, // Usage(Play)
    0x09, 0xB7, // Usage(Stop)
    0x09, 0xCD, // Usage(Play/Pause)
    0x09, 0xB5, // Usage(Scan Next Track)
    0x09, 0xB6, // Usage(Scan Previous Track)
    0x09, 0xB2, // Usage(Record)
    0x09, 0xB4, // Usage(Rewind)
    0x09, 0xB3, // Usage(Fast Forward)
    0x75, 0x01, // Report Size(0x1 )
    0x95, 0x08, // Report Count(0x8 )
    0x81, 0x02, // Input(Data, Variable, Absolute, No Wrap, Linear, Preferred State,
No Null Position, Bit Field)
    0x09, 0x00, // Usage(Undefined)
    0x75, 0x08, // Report Size(0x8 )
    0x95, 0x06, // Report Count(0x6 )
    0x81, 0x02, // Input(Data, Variable, Absolute, No Wrap, Linear, Preferred State,
No Null Position, Bit Field)
    0x09, 0x00, // Usage(Undefined)
    0x75, 0x08, // Report Size(0x8 )
```

```
    0x95, 0x08, // Report Count(0x8 )
    0x91, 0x00,
    0xC0
};
```

3.6 测试结果

下表是两款线控耳机麦克风中的 R1 与 R2 的值，并在麦克风输入端上实测到的 DC 值与 ADC 转换后的值，提供作参考。

		MIC+ V (ADC Value)			電阻	
	未按 Key	上 Key	中 Key	下 Key	R1 (ohm)	R2 (ohm)
小米耳麥	1.97 V (2450)	223 mV (274)	0	515 mV (638)	218	579
FOXXRAY耳麥	1.71 V (2127)	238 mV (293)	0	461 mV (570)	235	503

4 修订历史

Date	Revision	Description
2016.07.29	1.00	1. 初次发行版本

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.