ARC

USB

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-3-6

目录

第 17 章 USB.....3
17.1 USB 简介.....3
17.2 USB 应用实例 ----- 通过触摸屏移动电脑鼠标.....3
17.2.1 实例描述.....3
17.2.2 硬件设计.....3
17.2.3 软件设计.....4

第17章 USB

17.1 USB 简介

USB ,是英文 Universal Serial BUS（通用串行总线）的缩写，而其中文简称为“通串线，是一个外部总线标准，用于规范电脑与外部设备的连接和通讯。是应用在 PC 领域的接口技术。USB 接口支持设备的即插即用和热插拔功能。USB 是在 1994 年底由英特尔、康柏、IBM、Microsoft 等多家公司联合提出的。

USB 各版本区别版本最大传输速率速率称号最大输出电流协议推出时间：

- USB1.0：1.5Mbps(192KB/s）低速(Low-Speed)500mA……1996 年 1 月
- USB1.1：12Mbps(1.5MB/s）全速（Full-Speed)500mA……1998 年 9 月
- USB2.0：480Mbps(60MB/s）高速（High-Speed)500mA……2000 年 4 月
- USB3.0：5Gbps(640MB/s）超速（Super-Speed)900mA……2008 年 11 月

USB 采用四线电缆，其中两根(D+, D-)是用来传送数据的串行通道，另两根(GND, VCC)为下游（Downstream）设备提供电源，

17.2 USB 应用实例 ----- 通过触摸屏移动电脑鼠标

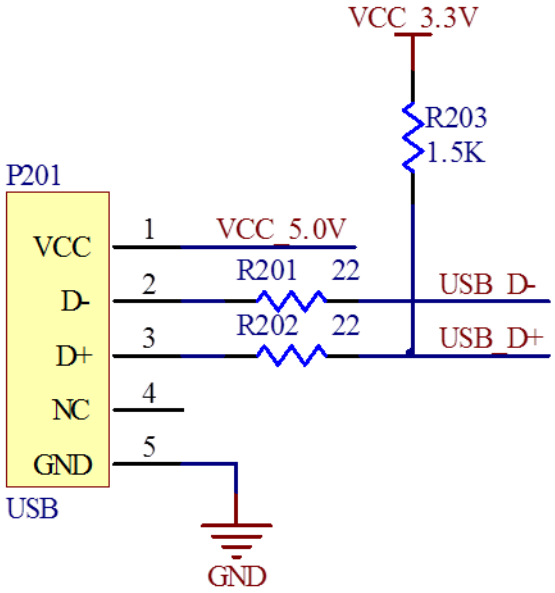
17.2.1 实例描述

本实例参照 ST 官方 USB 的 JoyStick 实例（3.3 版本），来控制电脑的鼠标，当你在触摸屏上划线是，该坐标值将会回传到电脑，可以看到鼠标随之移动。

17.2.2 硬件设计

该实例的 GPIO 分配和硬件原理图如下：

USB_D-	PA11
USB_D+	PA12



17.2.3 软件设计

下面重点介绍移植 JoyStick 到 ARC 平台，首先在 KEIL 的 USBMouse 项目内增加两个目录，分别用来存放 USB 驱动代码和 USB 中间层代码，名字分别为 USBFSPeriph_Driver 和 ARC_USB，将驱动文件 usb_core.c, usb_regs.c, usb_int.c, usb_sil.c, usb_mem.c, usb_init.c 包含到 USBFSPeriph_Driver 目录。将 hw_config.c, usb_prop.c, usb_istr.c, usb_desc.c, usb_pwr.c, usb_endp.c 包含到 ARC_USB 目录。

需要做修改的文件是 usb_desc.c，把那些 USB 描述改为你自己需要的。在 hw_config.c 内，把 Joystick_Send() 函数改为控制鼠标，

下面的代码是改动的代码，没有改动的代码请参考 ST 实例。

文件 USBMouse_main.c：

```
/**
  * @brief   Main program, usb control mouse example
  * @param   None
  * @retval None
  */
int main(void)
{
    uint8_t Mouse_Buffer[4];
    static uint8_t pre_x, pre_y;
    pen_state_struct *pen_st;
    pen_st = ARC_get_penstate();

    ARC_SysTick_Init();
    ARC_COM_Init();
```

```
    USART_Cmd(USART1, ENABLE);
    ARC_LCD_Init();
    ARC_LCD_On();

    ARC_TouchScreen_Init();
    SPI_Cmd(SPI1, ENABLE);
    ARC_TouchScreen_start();

    ARC_USBMouse_Init();

    ARC_LCD_Clear(LCD_COLOR_RED);
    ARC_LCD_ShowString(3, 3, "clear");

    ARC_Button_Init();

    while(1)
    {
        if (pen_st->force_adjust == 1)
        {
            ARC_TouchScreen_Adjust();
            pen_st->force_adjust = 0;
            ARC_LCD_Clear(LCD_COLOR_RED);
            ARC_LCD_ShowString(3, 3, "clear");
        }
        if (pen_st->pen_pressed == KEY_DOWN)
        {
            pen_st->pen_pressed = KEY_UP;
            do
            {
                if(ARC_TouchScreen_Rd_LCD_XY())
                {
                    pre_x = pen_st->x_converted;
                    pre_y = pen_st->y_converted;

                    if(pen_st->x_converted < 40 && pen_st->y_converted <
15)
                    {
                        ARC_LCD_Clear(LCD_COLOR_RED);
                        ARC_LCD_ShowString(3, 3, "clear");
                    }
                    else
                        ARC_LCD_DrawBigPoint(pen_st->x_converted,
pen_st->y_converted, LCD_COLOR_WHITE);
```

```
                        if(ARC_TouchScreen_Rd_LCD_XY())
                        {
                            if((abs(pen_st->x_converted - 20) < 18) &&
                               (abs(pen_st->y_converted - 8) < 6))
                            {
                                ARC_LCD_Clear(LCD_COLOR_RED);
                                ARC_LCD_ShowString(3, 3, "clear");
                            }
                            else

ARC_LCD_DrawBigPoint(pen_st->x_converted, pen_st->y_converted,
LCD_COLOR_WHITE);

                            if((bDeviceState == CONFIGURED) ||
PrevXferComplete)
                            {
                                Mouse_Buffer[0] = 0;
                                Mouse_Buffer[1] = (pen_st->x_converted -
pre_x) * 30;
                                Mouse_Buffer[2] = (pen_st->y_converted -
pre_y) * 30;
                                Mouse_Buffer[3] = 0;
                                Joystick_Send(Mouse_Buffer, 4);
                            }
                        }
                    }
                }while(!ARC_PEN_STATE());
            }
        }
}
```

文件 ARC_USBMouse.c：

```
/**
  * @brief   Initialize USBMouse.
  * @param   None
  * @retval None
  */
void ARC_USBMouse_Init(void)
{
    ARC_USB_RCC_Init();
    ARC_USB_NVIC_Init();
    ARC_USB_EXTI_Init();
    USB_Init();
}
```

文件 ARC_RCC.c

```
/**
   * @brief   Configures USB clocks.
   * @param   None
   * @retval None
   */
void ARC_USB_RCC_Init(void)
{
   /* Select USBCLK source */
   RCC_USBCLKConfig(RCC_USBCLKSource_PLLCLK_1Div5);

   /* Enable the USB clock */
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_USB, ENABLE);
}
```

文件 ARC_NVIC_API.c：

```
/**
   * @brief   Initialize NVIC of USB mouse.
   * @param   None
   * @retval None
   */
void ARC_USB_NVIC_Init(void)
{
   NVIC_InitTypeDef NVIC_InitStructure;

   /* Enable the USB interrupt */
   NVIC_InitStructure.NVIC_IRQChannel = USB_LP_CAN1_RX0_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);

   /* Enable the USB Wake-up interrupt */
   NVIC_InitStructure.NVIC_IRQChannel = USBWakeUp_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
   NVIC_Init(&NVIC_InitStructure);
}
```

文件 ARC_EXTI.c：

```
/**
   * @brief   Initialize EXTI of USB.
   * @param   None
   * @retval None
```

```
   */
void ARC_USB_EXTI_Init(void)
{
      EXTI_InitTypeDef EXTI_InitStructure;

      /* Configure the EXTI line 18 connected internally to the USB IP */
      EXTI_ClearITPendingBit(EXTI_Line18);
      EXTI_InitStructure.EXTI_Line = EXTI_Line18;
      EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
      EXTI_InitStructure.EXTI_LineCmd = ENABLE;
      EXTI_Init(&EXTI_InitStructure);
}
```

文件 usb_desc.c：

```
/* USB String Descriptors (optional) */
const uint8_t Joystick_StringLangID[JOYSTICK_SIZ_STRING_LANGID] =
{
      JOYSTICK_SIZ_STRING_LANGID,
      USB_STRING_DESCRIPTOR_TYPE,
      0x09,
      0x04
}; /* LangID = 0x0409: U.S. English */

const uint8_t Joystick_StringVendor[JOYSTICK_SIZ_STRING_VENDOR] =
{
      JOYSTICK_SIZ_STRING_VENDOR, /* Size of Vendor string */
      USB_STRING_DESCRIPTOR_TYPE,    /* bDescriptorType*/
      /* Manufacturer: "ARC" */
      'A', 0, 'R', 0, 'C', 0
};

const uint8_t Joystick_StringProduct[JOYSTICK_SIZ_STRING_PRODUCT] =
{
      JOYSTICK_SIZ_STRING_PRODUCT,          /* bLength */
      USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */
      'A', 0, 'R', 0, 'C', 0, ' ', 0, 'M', 0, 'o', 0, 'u', 0,
      's', 0, 'e'
};
uint8_t Joystick_StringSerial[JOYSTICK_SIZ_STRING_SERIAL] =
{
      JOYSTICK_SIZ_STRING_SERIAL,           /* bLength */
      USB_STRING_DESCRIPTOR_TYPE,        /* bDescriptorType */
      'A', 0, 'R', 0, 'C', 0, ' ', 0, ' ', 0, ' ', 0, ' ', 0
};
```

文件 hw_config.c：

```
/*******************************************************************************
* Function Name : Joystick_Send.
* Description    : prepares buffer to be sent containing Joystick event infos.
* Input          : Mouse_Buffer: Mouse position to be sent.
* Output         : None.
* Return value   : None.
*******************************************************************************/
void Joystick_Send(uint8_t *Mouse_Buffer, uint32_t buffer_size)
{
  /* Reset the control token to inform upper layer that a transfer is ongoing */
  PrevXferComplete = 0;

  /* Copy mouse position info in ENDP1 Tx Packet Memory Area*/
  USB_SIL_Write(EP1_IN, Mouse_Buffer, buffer_size);

  /* Enable endpoint for transmission */
  SetEPTxValid(ENDP1);
}
```