2012

三天入门 Cortex-M4
—Kinetis 系列

野火 Kinetics 开发板教程

综合测试例程，无线调试与无线图像传输实验。

freescale

作者：野火

野火嵌入式开发工作室

2012-11-19

综合例程讲解

此例程为无线调试和无线图像传输例程。

此例程拥有更多的创新亮点，为智能车比赛提供更加优越的调试条件和解决更多技术难题：

Kinetis 系列单片机中，野火首家独家提供快速稳定实现按键长按、短按、连按功能的例程；

目前其他的按键例程，清一色都是延时消抖

Kinetis 系列单片机中，野火首家独家提供硬件 SPI 驱动 NRF24L01+例程；

网上有软件 SPI 驱动的，我们提供硬件驱动

*Kinetis 系列单片机中，野火首家独家提供单片机之间实现图像无线传输功能，80*60 大小的二值化图像传输速度高达 37.5 帧每秒；*

网上有的是蓝牙模块与 PC 间图像传输

智能车比赛中，野火首家独家提供采集速度高达 150 帧每秒的摄像头模块，使用 K60 的 DMA 模块来采集；

智能车比赛专用摄像头中，没有如此高速度的摄像头模块。

无线调试

野火 K60 小霸王 板载 按键 PREV 、NEXT、ADD、SUB、OK、CANCEL 共 6 个按键，分别对应的功能为 切换上一个、切换下一个、变量值增加、变量值减少、确认、取消，支持长按、短按、连按功能。

假设有 8 位变量 var1、var2，16 位变量 var3、var4，32 位变量 var5、var6 需要通过按键调试来改变值，同时在 LCD 上显示进行人机交互。

我们在 main.c 里定义这几个变量（当然，你也可以在其他*.c 文件里定义）。

```
1.    u8  var1,var2;
2.    u16 var3,var4;
3.    u32 var5,var6;
```

目前支持 8 位，16 位，32 位无符号类型变量的无线传输。
如果需要，可以在这里赋初值。

在 key_event.c 里把地址写进数组里保存其地址。

```
1.    extern u8  var1,var2;
2.    extern u16 var3,var4;
3.    extern u32 var5,var6;
4.
5.    u32 * var_addr[VAR_MAX]={(u32 *)&var1,(u32 *)&var2,
6.                             (u32 *)&var3,(u32 *)&var4,
7.                             (u32 *)&var5,(u32 *)&var6};
```

不管是原来多少位的，都需要先转换为 32 位来进行处理。

变量顺序是有限制的，后面在 key_event.h 讲 对变量进行编号时就会讲到。

在 key_event.c 里也需要设置变量的最大值、最小值、及 LCD 显示的位置，以便人机交互。

var_addr 保存的是我们需要调试变量的地址。
num_info 保存的是变量调试的临时数据。
按键事件初始化的时候就好从 var_addr 数组里的变量地址中取值复制到 num_info 数组里；
当按下 OK 按键的时候，就会进行同步，无线发送数据出去，如果发送成功，就会把 num_info 数组的数据复制到 var_addr 数组里的变量地址指向区域。

```
1.    ui_var_info_t  num_info[VAR_MAX]=
2.    {
3.        //{val,oldval,minval,maxval,{x,y}}
4.        //val,oldval,会在调用 key_event_init 的时候从其对应变量里赋值过来，
5.        //所以这里的值可以随意写
6.        //需要设置 minval,maxval,{x,y}
7.        //务必注意最小值不要大于最大值
8.        {0,0,0,100,{90,0}},              //变量 var1,
9.        {0,0,0,100,{90,20}},             //变量 var2,
10.       {0,0,0,300,{90,40}},             //变量 var3,
11.       {0,0,0,300,{90,60}},             //变量 var4,
12.       {0,0,0,65540,{90,80}},           //变量 var5,
13.       {0,0,0,65540,{90,100}}           //变量 var6,
14.   };
```

变量顺序是与 var_addr 一样的。

可在这里设定最大值和最小值，调试的时候就会限定变量的取值范围。

同时设置 LCD 显示的位置，便于人机交互。

编译的时候，调试板和小车模块的工程里变量尽管命名一样，但实际变量地址是不一样的，无线传输的时候，需要对变量进行编号，以确定不同的变量有对应的编号。

在 key_event.h 里实现对变量进行编号：

```
1.    //变量的发送与接收
2.    typedef enum
3.    {
4.        //变量的编号
5.
6.        /* 8 位变量 */
7.        VAR1,
8.        VAR2,
9.        VAR_8BIT = VAR2,          //8 位变量的结束编号
10.
```

8 位、16 位、32 位按先后顺序放入枚举里，放错了会导致程序运行异常。
var_addr 数组和 num_info 数组的元素的顺序与这里是一一对应的。

不同位数的变量，必须使用 VAR_xxBIT 来作为结尾，且等于上一个枚举元素的值。

```
    /* 16 位变量 */
    VAR3,
    VAR4,
    VAR_16BIT  = VAR4,          //16 位变量的结束编号

    /* 32 位变量 */
    VAR5,
    VAR6,
    VAR_32BIT  = VAR6,          //32 位变量的结束编号

    VAR_MAX,                    //变量数目
}var_tab_e;
```

利用这些编号，以及 var_addr 数组里保存变量的地址，我们就可通过这两个函数来修改其对应的值：

```
void save_var(var_tab_e var_tal,u32 var_data);    //根据编号保存变量的值
void get_var(var_tab_e var_tal,u32 *var_data);    //根据编号读取变量的值
```

例如我需要修改变量 var 的值为 20，除了直接赋值外，还可以根据变量的编号来赋值：

```
var1 = 20;                  //直接赋值
void save_var(VAR1,20);     //通过变量编号来赋值
```

由于我们的野火 K60 小霸王调试板 与 小车模块图像采集 的代码不是完全一样的，因此变量的地址是没法保证一样，无线传输的时候，直接传递变量的地址是行不通的，但变量的编号是肯定相同的，因此无线调试，传递的是变量的编号。

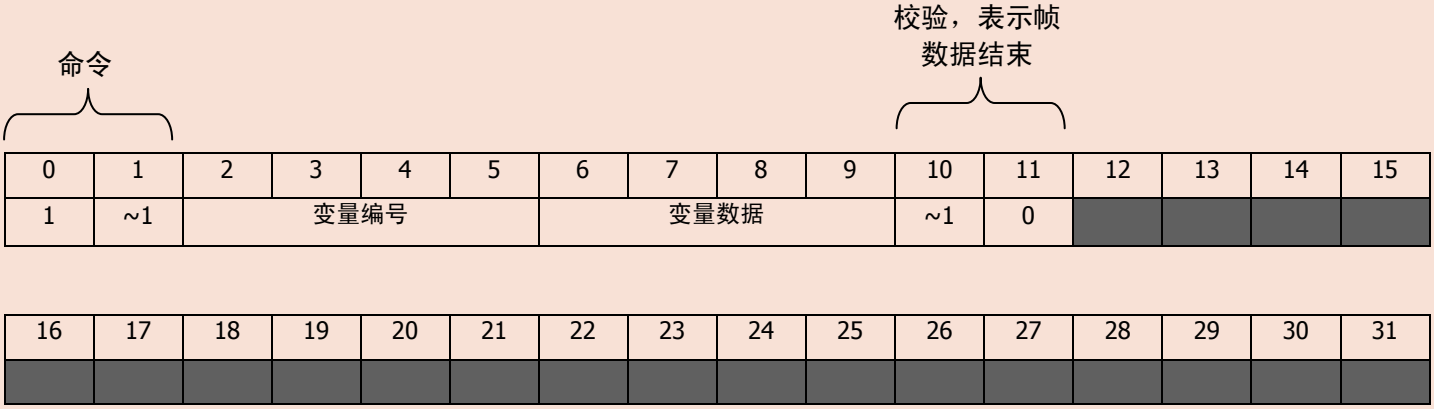
NRF24L01+模块，野火设置为 32 字节为一个数据包，而发送变量的值的数据是由 2 个字节命令 + 4 个字节变量编号 + 4 个字节的变量数据 + 2 个字节校验位组成，即共 12 个字节组成一个发送变量数据的帧，后面 32-12 = 20 个字节数据不进行处理。

关于命令，可在 NRF24L0_MSG.h 里定义：

```
//控制命令消息类型
typedef enum
{
```

```
4.          ////////////////// COM 命令 ////////////////////
5.          //图像传输控制
6.          COM_IMG,                //发送图像命令:控制命令后，就直接接图像数据
7.          //事件控制
8.          COM_EVENT_INFORM,       //控制事件消息通知
9.
10.         COM_MAX      ,          //最大控制命令数目
11.     }crtl_e;
```

发送变量的值的命令就是 COM_EVENT_INFORM，即值为 1，所以发送变量的值的帧仅占用一个数据包，数据包的格式为：



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	~1	变量编号				变量数据				~1	0				

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

接收方接收到数据后，根据命令的不同而选择不同的函数来对数据进行处理，例如接收到 COM_EVENT_INFORM 命令后 就会执行 rx_com_event_inform 函数，函数里利用 变量编号来调用 save_var 函数， 把变量数据赋值到对应的变量里。

野火讲解到这里，基本的按键调试变量和无线发送变量的使用方法就讲解完毕了，不知道各位是否懂得使用呢？

无线调试的底层实现函数是比较复杂，所用的语法也是同学们很少看到的，因此野火在这里不再一一细讲，后续考虑再写一个教程来详细讲解各个底层代码的实现及各种项目开发常用的 C 语言语法。

如果需要增加调试变量，仅需要修改前面讲解的几个参数：var_addr、num_info、var_tab_e。

在野火原来的基础上再加多一个 8 位变量 var7 为例，需要修改的内容：

野火

野火就啰嗦一次，给各位同学们举个例子来讲解，不然部分基础相对薄弱的同学可能没法学会使用。

1. 定义变量

在 main.c 里定义变量：

```
1.    u8 var7;
```

2. 添加变量编号

在 key_event.h 里实现对变量进行编号：

```
1.    //变量的发送与接收
2.    typedef enum
3.    {
4.        //变量的编号
5.
6.        /* 8 位变量 */
7.        VAR1,
8.        VAR2,
9.        VAR7,
10.       VAR_8BIT = VAR7, //8 位变量的结束编号
11.
12.       /* 16 位变量 */
13.       VAR3,
14.       VAR4,
15.       VAR_16BIT  = VAR4,          //16 位变量的结束编号
16.
17.       /* 32 位变量 */
18.       VAR5,
19.       VAR6,
20.       VAR_32BIT  = VAR6,          //32 位变量的结束编号
21.
22.       VAR_MAX,        //变量数目
23.   }var_tab_e;
```

或者可以改成（后面的顺序也得改）

```
1.      /* 8 位变量 */
2.      VAR1,
3.      VAR7,
4.      VAR2,
5.      VAR_8BIT = VAR2, //8 位变量的结束编号
```

注意比较不同

3. 把变量地址写入 var_addr

必须与 var_tab_e 保持一样的顺序

```
1.      u32 * var_addr[VAR_MAX]={ (u32 *)&var1,(u32 *)&var2,(u32 *)&var7,
2.                                (u32 *)&var3,(u32 *)&var4,
3.                                (u32 *)&var5,(u32 *)&var6 };
```

4. 在 num_info 里设置变量的最大值、最小值、LCD 显示位置

```
    ui_var_info_t  num_info[VAR_MAX]=
    {
        //  {val,oldval,minval,maxval,{x,y}}
        //val,oldval,会在调用 key_event_init 的时候从其对应变量里赋值过来，
        //所以这里的值可以随意写
        //需要设置 minval,maxval,{x,y}
        //务必注意最小值不要大于最大值
        {0,0,0,100,{90,0}},              //变量 var1,
        {0,0,0,100,{90,20}},             //变量 var2,
        {0,0,40,50,{20,30}},             //变量 var7,
        {0,0,0,300,{90,40}},             //变量 var3,
        {0,0,0,300,{90,60}},             //变量 var4,
        {0,0,0,65540,{90,80}},           //变量 var5,
        {0,0,0,65540,{90,100}}           //变量 var6,
    };
```

必须与 var_tab_e 保持一样的顺序。这里设置最大值为 50，最小值为 40，LCD 实现的位置为 x=20，y=30。

无线图像传输

野火 K60 小霸王采用野火鹰眼摄像头模块来采集图像，采集速度高达 150 帧每秒；采用 NRF24L01+模块传输 80*60 的二值化图像，传输速度为 37.5 帧每秒。

图像采集与无线发送的步骤是很简单的：

1. 定义缓存区和图像缓冲区指针

```
u8  nrf_buff[CAMERA_SIZE + MAX_ONCE_TX_NUM];      //预多
u8 *img_bin_buff = (u8 *)(((u8 *)&nrf_buff) + COM_LEN);
```

```
3.          //二值化图像的 buf 指针，由于开头有 COM_LEN 个字节是留给命令，所以需要加 COM_LEN
```

nrf_buff 缓存区是用于无线模块发送数据的缓存区，采集图像时直接把图像采集到这个缓存区里，这样可以节省内存和复制时间。

img_bin_buff 是图像缓存区指针，直接指向 nrf_buff 缓存区。

2.初始化图像，把图像缓存区地址传递进去，采集图像时直接采集到图像缓存区里

```
1.    Ov7725_Init(img_bin_buff);                    //摄像头初始化
```

3.调用图像采集函数采集图像

```
1.    ov7725_get_img();                             //采集图像
```

ov7725_get_img 函数的实现是在 OV7725.c 里完成：

```
1.    void ov7725_get_img()
2.    {
3.        img_flag = IMG_START;                     //开始采集图像
4.        PORTA_ISFR=~0;          //写 1 清中断标志位(必须的，不然回导致一开中断就马上触发中断)
5.        enable_irq(87);                           //允许 PTA 的中断
6.        while(img_flag != IMG_FINISH)             //等待图像采集完毕
7.        {
8.            if(img_flag == IMG_FAIL)              //假如图像采集错误，则重新开始采集
9.            {
10.               img_flag = IMG_START;             //开始采集图像
11.               PORTA_ISFR=~0; //写 1 清中断标志位(必须的，不然回导致一开中断就马上触发中断)
12.               enable_irq(87);                   //允许 PTA 的中断
13.           }
14.       }
15.   }
```

4. 等待场中断来临

场中断里关闭场中断，启动 DMA 传输

```
1.    /*************************************************************************
2.    *                           野火嵌入式开发工作室
3.    *
```

```
4.      *  函数名称：PORTA_IRQHandler
5.      *  功能说明：PORTA 端口中断服务函数
6.      *  参数说明：无
7.      *  函数返回：无
8.      *  修改时间：2012-1-25     已测试
9.      *  备     注：引脚号需要自己初始化来清除
10.     ********************************************************************/
11.     void PORTA_IRQHandler()
12.     {
13.         u8  n = 0;     //引脚号
14.         n = 29;                                         //场中断
15.         if(PORTA_ISFR & (1 << n))                       //PTA29 触发中断
16.         {
17.             //场中断需要判断是场结束还是场开始
18.             if(img_flag == IMG_START)                   //需要开始采集图像
19.             {
20.                 img_flag = IMG_GATHER;                  //标记图像采集中
21.                 disable_irq(87);
22.                 DMA_EN(CAMERA_DMA_CH);                  //使能通道 CHn 硬件请求
23.                 DMA_DADDR(CAMERA_DMA_CH) = (u32)IMG_BUFF;     //恢复地址
24.             }
25.             else if(img_flag == IMG_GATHER)             //图像采集中进入场中断，即图像采集完毕
26.             {
27.                 while(1);                               //DMA 采集异常
28.             }
29.             else                                        //图像采集错误
30.             {
31.                 disable_irq(87);                        //关闭 PTA 的中断
32.                 img_flag = IMG_FAIL;                    //标记图像采集失败
33.             }
34.             PORTA_ISFR  = ~0;                           //场中断里，全部都要清中断标志位
35.             return;                                     //场中断触发了，就不需要处理行中断
36.         }
37.         PORTA_ISFR  = ~0;                   //写 1 清中断标志位
38.     }
```

5. DMA 中断处理

DMA 中断来了，表示图像采集完成，需要清中断标志位，标志图像完成。

```
1.      void DMA0_IRQHandler()
2.      {
3.          DMA_DIS(CAMERA_DMA_CH);              //关闭通道 CHn 硬件请求
4.          DMA_IRQ_CLEAN(CAMERA_DMA_CH);            //清除通道传输中断标志位
5.          img_flag = IMG_FINISH ;
6.      }
```

ov7725_get_img 里不断等待 img_flag 为 IMG_FINISH ，实际上就是等待 DMA 中断到来。

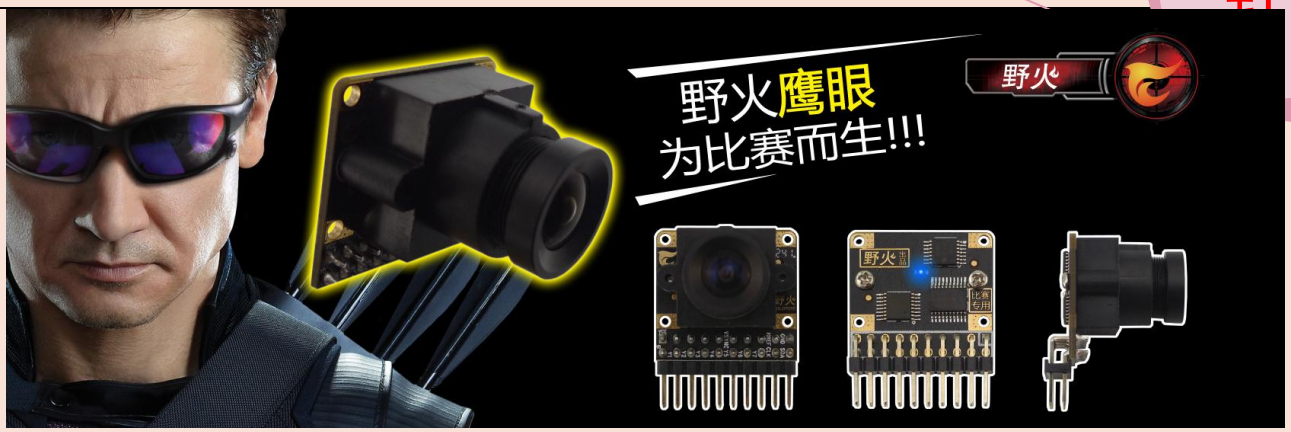
6.图像采集完成后，无线发送 图像

只需传递命令和缓存区给 NRF_MSG_send 函数即可完成图像发送功能，NRF_MSG_send 函数会根据命令来决定发送数据的长度，用户不需要关系数据的长度。

```
1.      NRF_MSG_send(COM_IMG,nrf_buff);          //发送数据，发送长度由 com 对应的函数决定
```

由于使用中断发送，NRF_MSG_send 函数执行完后，图像还没发送完毕的，我们需要不断查询是否还在发送状态，确认图像发送完毕了才采集下一幅图像。在图像发送过程中，我们是可以利用这些空余时间来完成其他任务，例如 LCD 显示，图像处理等可被中断的，且不会破坏图像发送缓存区的任务。

```
1.      LCD_Img_Binary(site,size,(u16 *)img_bin_buff);       //显示图像
2.      do  {
3.          status = NRF_MSG_send_state();       //获取发送状态
4.      }while( status == TX_ISR_SEND);          //如果图像发送中，则需要继续等待发送完成
5.
6.      /*判断发送状态*/
7.      if(status == TX_ISR_SUCCEED )
8.      {
9.          printf("\r\n 图像发送成功！ \r\n");
10.     }
11.     else if( status == TX_ISR_FAIL )
12.     {
13.         printf("\r\n 图像发送失败。 \r\n");
14.     }
```


野火鹰眼
为比赛而生!!!
野火