AN1705C ATK-PAJ7620 手势识别模块

使用说明

本应用文档（AN1705C）将教大家如何在 ALIENTEKT 探索者 STM32F407 开发板上使用 ATK-PAJ7620 手势识别传感器模块。

本文档分为如下几部分：

1， ATK-PAJ7620 模块简介

2， 硬件连接

3， 软件实现

4， 验证

1、ATK-PAJ7620 模块简介

ATK-PAJ7620_V1.2(V1.2 是版本号，下面均以 ATK-PAJ7620 表示该产品)是 ALIENTEK 推出的一款高性能手势识别传感器模块。该模块采用原相科技（Pixart）公司的 PAJ7620U2 芯片，芯片内部集成了光学数组式传感器，以使复杂的手势和光标模式输出，自带九种手势识别，支持上、下、左、右、前、后、顺时针旋转、逆时针旋转和挥动的手势动作识别，以及支持物体接近检测等功能。

ATK-PAJ7620 模块具有：体积小、灵敏度高、支持中断输出、兼容 3.3V/5V 系统、使用方便等特点，模块通过 6 个 2.54mm 间距的排针与外部连接，模块外观如图 1.1 所示：

图 1.1 ATK-PAJ7620 模块外观图

1.1 PAJ7620U2 简介

PAJ7620U2 是原相科技（PixArt）公司推出的一款光学数组式传感器，内置光源和环境光抑制滤波器集成的 LED，镜头和手势感测器在一个小的立方体模组，能在黑暗或低光环境下工作。同时传感器内置手势识别，支持 9 个手势类型和输出的手势中断结果。并且内置接近检测功能，可用于感测物体接近或离开。

PAJ7620U2 的特点包括：

①IIC 接口，支持高达 400Khz 通信速率。

②内置 9 个手势类型（上、下、左、右、前、后、顺时针旋转、逆时针旋转、挥动），支持输出中断。

③支持接近检测功能，检测物体体积大小和亮度。

④待机功耗电流 15uA。

⑤抗灯光干扰

PAJ7620U2 的模块功能框图如图 1.1.1 所示：

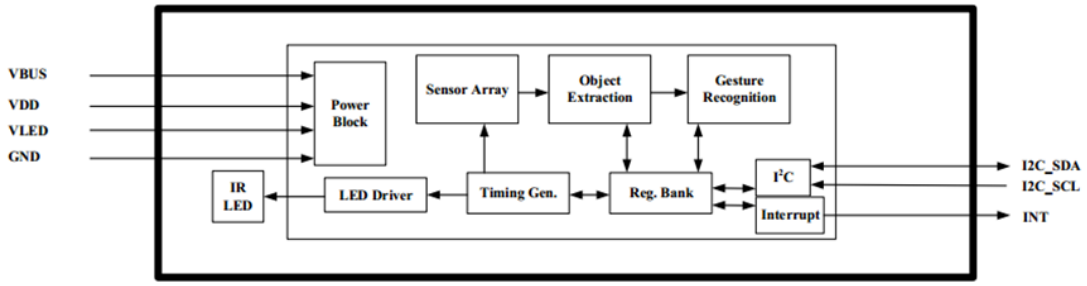

图 1.1.1 PAJ7620U2 功能框图

在图 1.1.1 框图可以看到，PAJ7620U2 内部自带 LED 驱动器，传感器感应阵列、目标信息提取阵列和手势识别阵列。PAJ7620U2 工作时通过内部 LED 驱动器，驱动红外 LED 向外发射红外线信号，当传感器阵列在有效的距离中探测到物体时，目标信息提取阵列会对探测目标进行特征原始数据的获取，获取的数据会存在寄存器中，同时手势识别阵列会对原始数据进行识别处理，最后将手势结果存到寄存器中，用户可根据 I2C 接口对原始数据和手势识别的结果进行读取。

1.2 手势操作说明

PAJ7620U2 内部自带了 9 个手势识别，分别是“上”、“下”、“左”、“右”、“前”、“后”、“顺时针旋转”、“逆时针旋转”、“挥动”。使用时传感器的开窗口位置需朝上，如图 1.2.1 所示：

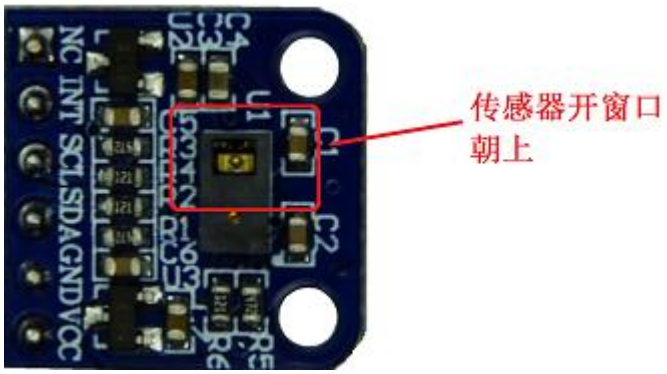

图 1.2.1 传感器朝向位置（开窗口向上）

手在传感器的上方，保持与传感器的垂直距离，做出图 1.2.2(1)和(2)所示的手势，可以分别得出“上”、“下”、“左”、“右”、“顺时针旋转”、“逆时针旋转”、“挥动”的识别结果。

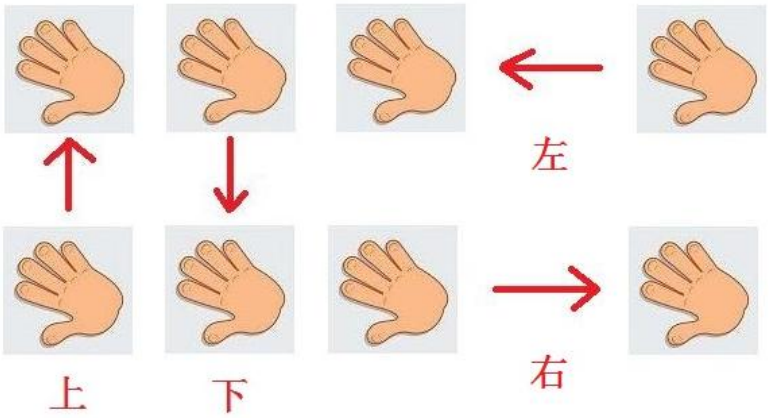

图 1.2.2（1）“上”、“下”、“左”、“右”手势

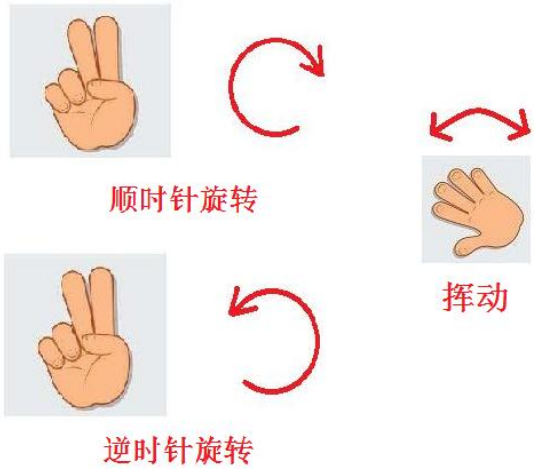

图 1.2.2（2）“顺时针旋转”、“逆时针旋转”、“挥动”手势

手在传感器的上方，与传感器的垂直方向上距离有相对的变化，做出图 1.2.2(3)所示的手势，可以得出“前”、“后”的识别结果。

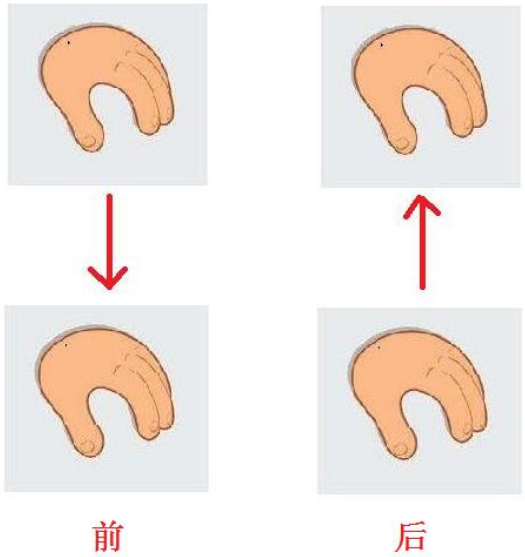

图 1.2.2（3）“前”、“后”手势

1.3 模块常用寄存器简介

PAJ7620U2 内部总共有上百个寄存器，但官方的也只是对部分的寄存器进行讲解，这里我们仅介绍 PAJ7620U2 常用的几个寄存器。其他的请大家参考：PAJ7620U2 芯片手册。

这里我们先提及下，在 PAJ7620U2 的内部有两个 BANK 寄存器区域，分别是 BANK0 和 BANK1。不同的区域用于访问不同的功能寄存器，但想访问其中的 BANK 区域下的寄存器，需在访问前发送控制指令进入该寄存器区域，具体控制指令如表 1.3.1 所示：

BANK 区域	地址	数值
BANK0	0xEF	0x00
BANK1	0xEF	0x01

表 1.3.1 控制指令

从表可以看到，进入 BANK0 区域往传感器 0xEF 地址写 0x00 数值，而 BANK1 区域往传感器 0xEF 地址写 0x01 数值。

首先，我们介绍下 BANK1 下的 PAJ7620U2 使能工作寄存器，该寄存器地址为 0X72，寄存器描述如图 1.3.2 所示：

Register Bank 1, ADDR 0x72, Enable/Disable PAJ7620U2		
NAME	Reserved	Enable
BIT #	[7:1]	[0]
ACCESS	Write as 0	R/W
DEFAULT	0	0
	0x00	

NAME	FUNCTION/OPERATION
Enable	1: Enable PAJ7620U2 0: Disable PAJ7620U2

图 1.3.2 使能工作寄存器

该寄存器用于使能 PAJ7620U2 工作，这里我们只关心 bit0 位，设置为 1，则使能 PAJ7620 工作，设置为 0，则失能 PAJ7620U2 工作。

接下来我们介绍下 BANK0 下的挂起管理寄存器，该寄存器地址为 0X03，各位描述如图 1.3.3 所示：

Register Bank 0, ADDR 0x03, I²C Suspend Command		
NAME	Reserved	Suspend
BIT #	[7:1]	[0]
ACCESS	Write as 0	W
DEFAULT	0	1
	0x01	

NAME	FUNCTION/OPERATION
Suspend	Write 1: Enter suspend state (wake up by writing I2C slave ID (default: 0x73), Refer to topic "I²C Bus Timing Characteristics and Protocol"

图 1.3.3 挂起管理寄存器各位描述

其中，Suspend bit0 位用来控制挂起，要使 PAJ7620U2 进入挂起状态，并非设置该位为 1 就可以，手册上有具体说明如何进入和退出挂起，如图 1.3.4 所示：

To enter the suspend state, first disable the PAJ7620U2 by writing Register Bank 1, ADDR 0x72 with 0x00 then process the I²C suspend command by writing Register Bank 0, ADDR 0x03 with 0x01.
To exit the suspend state, first process the I²C wake-up command by writing the slave ID (Refer to topic "I²C Bus Timing Characteristics and Protocol") then enable the PAJ7620U2 by writing Register Bank 1, ADDR 0x72 with 0x01.

图 1.3.4 进入和退出挂起过程

进入挂起前，先配置 BANK1 的 PAJ7620U2 使能工作寄存器 0X72 为 0X00，失能 PAJ7620U2 工作，然后再往 BANK0 的挂起管理寄存器 0X03 写 0X01，才真正的挂起 PAJ7620U2。而退出挂起到唤醒工作，则需执行 3 个步骤：

（1）往 PAJ7620U2 发送写命令，以触发唤醒，命令格式如图 1.3.5 所示：

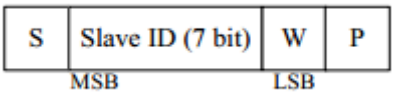

图 1.3.5 唤醒命令格式

唤醒命令格式就是 I2C 通讯的命令格式，从图可以看到，在发送写命令后不用等待 PAJ7629U2 的应答，直接发送停止就可以了。

（2）发送完唤醒命令后，需等待大于 700us 的时间，然后读取 PAJ7620U2 的 0X00 寄存器，判断是否为 0X20 数值，若不是则继续执行步骤 1 继续唤醒。直到读取 0X00 寄存器

值为 0X20，则唤醒成功。

（3）唤醒成功后（由于在挂起时，把 PAJ7620U2 给关闭了）往 BANK1 的使能工作寄存器 0X72 写 0X01，使能 PAJ7620U2 工作。

按照上面的三个步骤，就可以让 PAJ7620U2 从挂起到唤醒工作了。这里需注意一下，PAJ7620U2 首次上电，传感器也是工作在挂起状态，同样也需唤醒其工作。

接着我们看下 BANK0 下的手势检测输出中断使能寄存器 1，该寄存器地址为 0X41，各位描述如图 1.3.6 所示：

	Register Bank 0, ADDR 0x41, Gesture Detection Interrupt Flag Mask							
NAME	Counter-Clockwise Mask	Clockwise Mask	Backward Mask	Forward Mask	Right Mask	Left Mask	Down Mask	Up Mask
BIT #	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
ACCESS	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
DEFAULT	1	1	1	1	1	1	1	1
	0xFF							

NAME	FUNCTION/OPERATION
Counter Clockwise Mask	1: Counter clockwise gesture will generate an interrupt 0: Counter clockwise gesture will not generate an interrupt
Clockwise Mask	1: Clockwise gesture will generate an interrupt 0: Clockwise gesture will not generate an interrupt
Backward Mask	1: Backward gesture will generate an interrupt 0: Backward gesture will not generate an interrupt
Forward Mask	1: Forward gesture will generate an interrupt 0: Forward gesture will not generate an interrupt
Left Mask	1: Left gesture will generate an interrupt 0: Left gesture will not generate an interrupt
Right Mask	1: Right gesture will generate an interrupt 0: Right gesture will not generate an interrupt
Down Mask	1: Down gesture will generate an interrupt 0: Down gesture will not generate an interrupt
Up Mask	1: Up gesture will generate an interrupt 0: Up gesture will not generate an interrupt

图 1.3.6 手势检测输出中断使能寄存器 1

该寄存器作用于手势识别，bit0~bit7 位用于使能不同手势识别结果的中断输出，默认值为 0XFF，其中 bit0 位为“上”、bit1 位为“下”、bit2 位为“左”、bit3 位为“右”、bit4 位为“前”、bit5 位为“后”、bit6 位为“顺时针旋转”、而 bit7 位为“逆时针旋转”。对应位设置为 1，则使能，当检测到对应的手势识别时，会输出对应手势识别结果中断。若对应位设置为 0，则关闭手势识别结果中断。

接着我们看下 BANK0 下的手势检测输出中断使能寄存器 2，该寄存器地址为 0X42，各位描述如图 1.3.7 所示：

	Register Bank 0, ADDR 0x42, Gesture Detection Interrupt Flag Mask	
NAME	Reserved	Wave Mask
BIT #	[7:1]	[0]
ACCESS	Write as 0000000	R/W
DEFAULT	1111111	1
	0xFF	

NAME	FUNCTION/OPERATION
Wave Mask	1: Wave gesture will generate an interrupt 0: Wave gesture will not generate an interrupt

图 1.3.7 手势检测输出中断使能寄存器 2

该寄存器也是作用于手势识别，其中只有 bit0 位有作用，bit1-bit7 为保留位，寄存器默认值为 0XFF。bit0 为用于使能手势识别“挥动”的输出中断，当 bit0 位设置为 1 时，则使

能“挥动”手势识别输出中断，设置 0，则关闭输出中断。

接着我们看下 BANK0 的手势识别中断标志寄存器 1，该地址为 0X43，各位描述如图 1.3.8 所示：

	Register Bank 0, ADDR 0x43, Gesture Detection Interrupt Flag							
NAME	Counter Clockwise	Clockwise	Backward	Forward	Right	Left	Down	Up
BIT #	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
ACCESS	R	R	R	R	R	R	R	R
DEFAULT	-	-	-	-	-	-	-	-
	-							

NAME	FUNCTION/OPERATION
Counter Clockwise	1: Counter clockwise gesture be detected 0: No Counter clockwise gesture be detected
Clockwise	1: Clockwise gesture be detected 0: No Clockwise gesture be detected
Backward	1: Backward gesture be detected 0: No Backward gesture be detected
Forward	1: Forward gesture be detected 0: No Forward gesture be detected
Right	1: Right gesture be detected 0: No Right gesture be detected
Left	1: Left gesture be detected 0: No Left gesture be detected
Down	1: Down gesture be detected 0: No Down gesture be detected
Up	1: Up gesture be detected 0: No Up gesture be detected

图 1.3.8 手势识别中断标志寄存器 1

该寄存器作用于手势识别，实现手势识别输出结果的中断标志。当 BANK0 寄存器 0X41 使能了对应手势中断位后，当 PAJ7620U2 检测到内置的手势，则对应的寄存器手势标志会置 1，读取标志可清除对应的中断标志位。用户根据读取对应的状态，可知道当前手势识别的结果。

接着我们看下 BANK0 的手势识别中断标志寄存器 2，该地址为 0X44，各位描述如图 1.3.9 所示：

	Register Bank 0, ADDR 0x44, Gesture Detection Interrupt Flag	
NAME	Reserved	Wave
BIT #	[7:1]	[0]
ACCESS	R	R
DEFAULT	-	-
	-	

NAME	FUNCTION/OPERATION
Wave	1: Wave gesture be detected 0: Wave gesture be detected

图 1.3.9 手势识别中断标志寄存器 2

该寄存器是承接着 0X43 寄存器，同样也是作用于手势识别，实现手势识别输出结果的中断标志，其中只有 bit0 位有作用，bit1-bit7 为保留位。当 BANK0 寄存器 0X42 使能了“挥动”手势中断位后，PAJ7620U2 检测到“挥动”的手势，则对应的寄存器手势标志会置 1，读取标志可清除对应的中断标志位。用户根据读取对应的状态，可知道当前手势识别的结果。

接下来我们看下 BANK0 的检测物体亮度寄存器，该地址为 0XB0，寄存器描述如图 1.3.10 所示：

Register Bank 0, ADDR 0xB0, Object Brightness, Report object brightness	
ADDRESS	ADDR 0xB0
NAME	ObjectAvgY[8:1]
BIT #	[7:0]
ACCESS	R
DEFAULT	-

NAME	FUNCTION/OPERATION
ObjectAvgY	Report Object Brightness (Max. value 255).

图 1.3.10 检测物体亮度寄存器

该寄存器实现获取检测物体亮度值。在接近检测下，当物体在 PAJ7620U2 的有效检测距离内，读取该寄存器能获得物体的亮度，亮度值为 0~255。

接下来我们看下 BANK0 的检测物体体积大小寄存器，地址为 0XB2 和 0XB1，寄存器描述如图 1.3.11 所示：

Register Bank 0, ADDR 0xB1, ADDR 0xB2, Object Size		
ADDRESS	ADDR 0xB2	ADDR 0xB1
NAME	ObjectSize[11:8]	ObjectSize[7:0]
BIT #	[3:0]	[7:0]
ACCESS	R	R
DEFAULT		

NAME	FUNCTION/OPERATION
ObjectSize	Report Object Size (Max. value 900).

图 1.3.11 检测物体体积大小寄存器

物体的体积大小值由两个寄存器值组合而成，分别是 0XB2 寄存器的低四位值和 0XB1 寄存器八位值。在接近检测下，当物体在 PAJ7620U2 的有效检测距离内，读取这两个寄存器能获得物体体积大小，体积值为 0~900。

以上就是常用的寄存器介绍，更多的寄存器介绍说明请看 PAJ7620U2 芯片手册。

1.4 模块实验流程

本例程实验主要测试 PAJ7620U2 的手势识别和接近检测（物体的体积大小和亮度）的功能（采用轮询方式，没有使用中断引脚），具体的工作流程如图 1.4.1 所示：

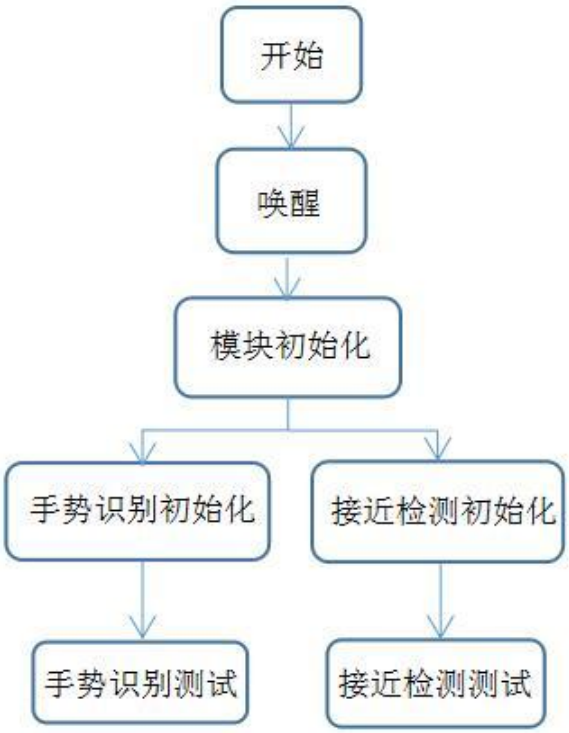

图 1.4.1 工作流程图

在前面的寄存器介绍有提及过，PAJ7620U2 在初次上电时会处于挂起状态，所以需唤醒才能使用，唤醒后需要初始化，但在 PAJ7620U2 的初始化中，需配置多个寄存器，而有些寄存器手册没提及到，不过幸好的是手册上有提供配置数组给用户，我们直接调用就可以了。同样手势识别和接近检测，手册也有提供初始化配置数组，我们也是调用就可以了。

在实验例程中，手势识别和接近检测通过按键进行选择，下面我们说下：

1：手势识别测试：通过 KEY1 按键进入此项测试。实现 PAJ7620U2 自带的 9 个手势识别检测，向上（Up）、向下（Dowm）、向左（Left）、向右（Right）、向前（Forward）、向后（Backward）、顺时针旋转（Clockwise）、逆时针旋转（Counterclockwise）和挥动（Wave）。当识别到正确的手势，DS1 灯会闪烁，同时手势结果显示在 LCD 屏幕上，并且串口会输出。DS0 灯闪烁提示程序正在运行，按下 KEY_UP 按键，可返回主菜单页面。

2，接近检测测试：通过 KEY0 按键进入此项测试。实现读取 PAJ7620U2 接近物体的体积大小和亮度的传感器数据，显示在 LCD 屏幕上，并串口输出，同时 DS0 灯闪烁，提示程序正在运行，当按下 KEY_UP 按键，可返回主菜单页面。

2、硬件连接

2.1 硬件准备资源

本实验所需要的硬件资源如下

1，ALIENTEK 探索者 STM32F407 开发板 1 个

2，TFTLCD 模块

3，ATK-PAJ7620 模块 1 个

4，USB 线一条（用于供电和模块与电脑串口调试助手通信）

2.2 模块与开发板连接

ATK-PAJ7620 模块可直接与 ALIENTEKT 探索者 STM32F407 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，ATK MODULE 与 MCU 连接原理图如图 2.2.1 所示：

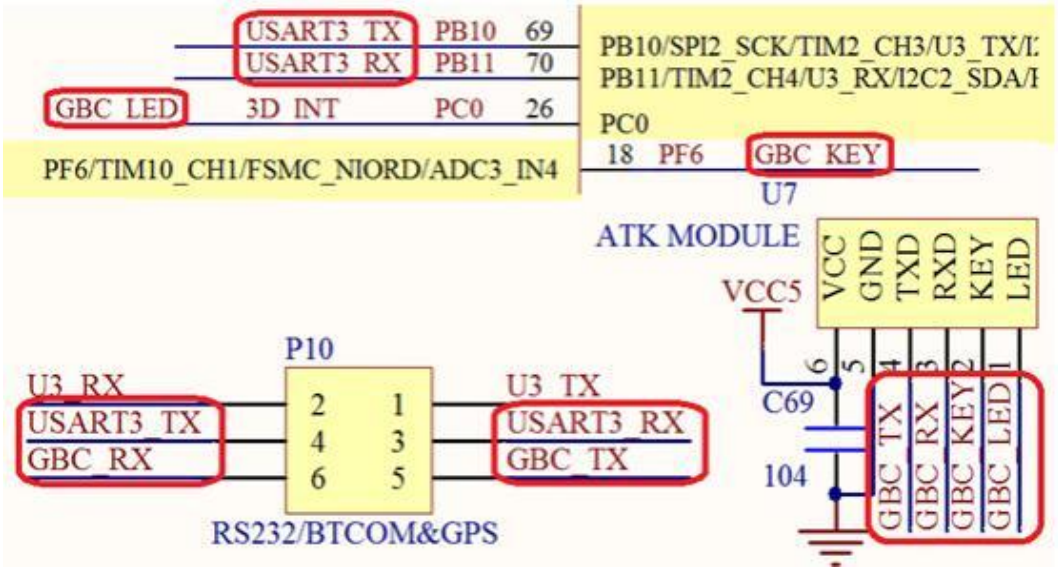

图 2.2.1 ATK-MODULE 接口与 MCU 连接关系

从上图看出，ATK MODULE 接口，使用时必须将 P10 的 USART3_TX（PB10）和 GBC_RX 以及 USART3_RX（PB11）和 GBC_TX 连接，才能完成和 STM32 的连接。探索者 F407 只需要用跳线帽去短接就可以了，连接好后，探索者 F407 开发板与 ATK-PAJ7620 模块的连接关系如表 2.3.2 所示：

ATK-PAJ7620 手势识别传感器模块与开发板连接关系				
ATK-PAJ7620 模块	VCC	GND	SCL	SDA
探索者 STM32F407 开发板	5V	GND	PB10	PB11

表 2.3.2 ATK-PAJ7620 模块与探索者 STM32F407 开发板连接关系图

ATK-PAJ7620 模块插入到开发板的 ATK MODULE 接口，如图 2.3.3 所示：

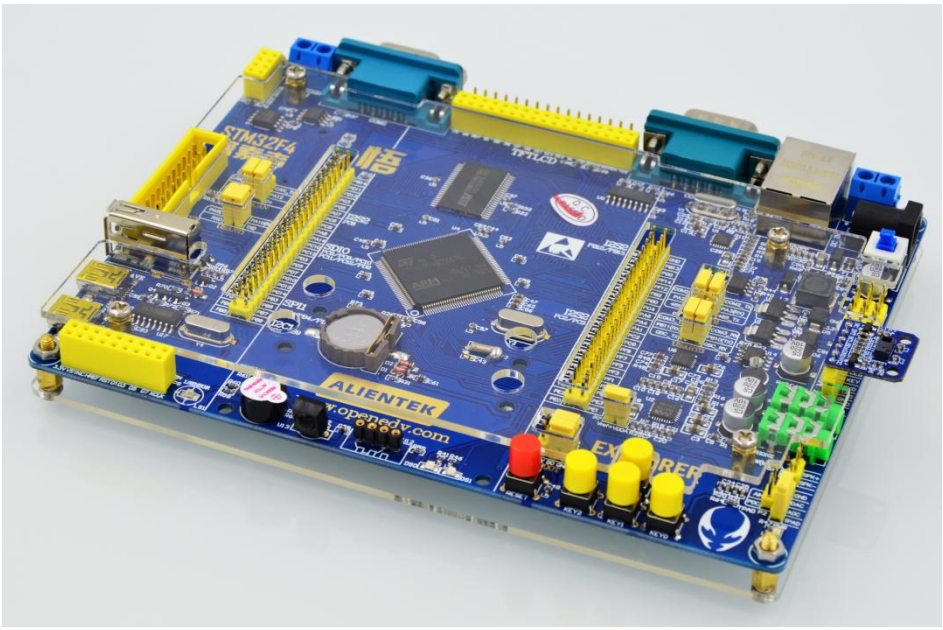

图 2.2.4 ATK-PAJ7620 模块与开发板对接实物图

3、软件实现

本实验在探索者 STM32F407 开发板的 IIC 实验基础上进行修改。

由于例程中没有用到 24C02，所以，先去掉 24cxx.c 和 myiic.c。然后在 HARDWARE 文件夹下新建一个 PAJ7620U2 的文件夹。然后新建 paj7620u2.c、paj7620u2.h、paj7620u2_iic.c、paj7620u2_iic.h、paj7620u2_cfg.h 共 5 个文件，paj7620u2.c 文件为手势识别和接近检测的驱动，paj7620u2_iic 文件为底层的 IIC 驱动，而 paj7620u2_cfg.h 文件为存放上电初始化、手势识别初始化、以及接近检测初始化的配置数组。接着在工程目录 HARDWARE 文件夹中添加 paj7620u2_iic.c 和 paj7620u2.c 文件，最后添加头文件包含路径。最终的工程如图 3.1 所示：

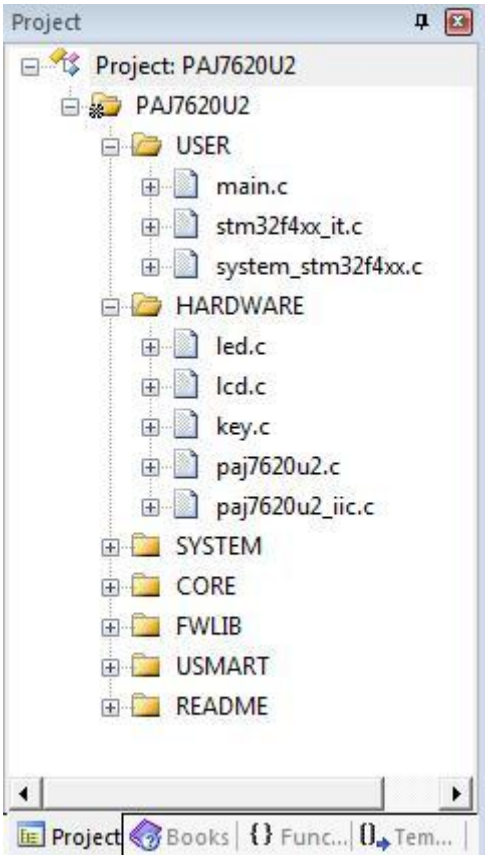

图 3.1ATK-PAJ7620 模块测试实验工程截图

本例程代码比较简单，这里我们先说下 paj7620u2_cfg.h 文件，该文件内容是关于初始化配置数组的，代码如下：

```
#define INIT_SIZE sizeof(init_Array)/2
//上电初始化数组
const unsigned char init_Array[][2] = {
    {0xEF,0x00},
    {0x37,0x07},
    {0x38,0x17},
    .....省略部分设置
    {0x73,0x35},
    {0x74,0x00},
    {0x77,0x01},
};
```

```
#define PROXIM_SIZE sizeof(proximity_arry)/2
//接近检测初始化数组
const unsigned char proximity_arry[][2]={
    {0xEF,0x00},
    {0x41,0x00},
    {0x42,0x00},
    {0x6B,0x50},
    ........省略部分设置
    {0x6C,0xC3},
    {0x6D,0x50},
    {0x6E,0xC3},
    {0x74,0x05},
};

#define GESTURE_SIZE sizeof(gesture_arry)/2
//手势识别初始化数组
const unsigned char gesture_arry[][2]={
    {0xEF,0x00},
    {0x74,0x00},
    ........省略部分设置
    {0xEF,0x00},
    {0x41,0xFF},
    {0x42,0x01},
};
```

以上代码，我们省略了部分（全部贴出来太长了），该文件包含了三个配置数组，分别作用于上电时的初始化、手势识别以及接近检测。该数组为二维数组，每个数组的第一个字节为寄存器号（也就是寄存器地址），第二个字节为要设置的值，比如{0xEF,0X00}，就表示在 0xEF 地址，写入 0x00 这个值。

接下来我们说下 paj7620u2_iic.c 文件，该文件包括 PAJ7620U2 的 IIC 底层驱动读写函数，引脚 IO 初始化以及唤醒命令的函数，关于底层和 IO 初始化就不做讲解了。这里我们讲解下唤醒命令函数，唤醒命令函数代码如下：

```
//PAJ7620U2 唤醒命令
void GS_WakeUp(void)
{
    GS_IIC_Start();//开始
    GS_IIC_Send_Byte(PAJ7620_ID);//发写命令
    GS_IIC_Stop();//释放总线
}
```

该函数实现触发唤醒，在前面有说到，唤醒需发送唤醒命令，而唤醒命令其实就是写指令，在发送写指令给 PAJ7620U2 后，不必等待 PAJ7620U2 返回的应答，直接释放总线就可以触发一次唤醒。

接下来我们说下 paj7620u2.c 文件，该文件包括手势识别和接近检测等代码，这里我们讲下：

（1）paj7620u2_selectBank()函数，具体代码如下：

```
//选择 PAJ7620U2 BANK 区域
void paj7620u2_selectBank(bank_e bank)
{
    switch(bank)
    {
        case BANK0: GS_Write_Byte(PAJ_REGITER_BANK_SEL,PAJ_BANK0);break;
                    //BANK0 寄存器区域
        case BANK1: GS_Write_Byte(PAJ_REGITER_BANK_SEL,PAJ_BANK1);break;
                    //BANK1 寄存器区域
    }
}
```

该函数实现选择 BANK 区域，通过 bank 的入口参数选择要进入 BANK 的区域。

（2）paj7620u2_wakeup()函数，具体代码如下：

```
//PAJ7620U2 唤醒
u8 paj7620u2_wakeup(void)
{
    u8 data=0x0a;
    GS_WakeUp();//唤醒 PAJ7620U2
    delay_ms(5);//唤醒时间>700us
    GS_WakeUp();//唤醒 PAJ7620U2
    delay_ms(5);//唤醒时间>700us
    paj7620u2_selectBank(BANK0);//进入 BANK0 寄存器区域
    data = GS_Read_Byte(0x00);//读取状态
    if(data!=0x20) return 0; //唤醒失败

    return 1;
}
```

该函数实现对 PAJ7620U2 唤醒，在代码中我们发送两次 GS_Wake()唤醒命令，因为发送一次，可能会遇到唤醒不起 PAJ7620U2。在发送唤醒命令后需等待大于 700us 的时间，然后切换到 BANK0 区域，读取 0x00 寄存器，判断读取的值是否为 0x20，若是则唤醒成功，否则唤醒失败。

（3）paj7620u2_init()函数，具体代码如下：

```
//PAJ7620U2 初始化
//返回值：0:失败  1:成功
u8 paj7620u2_init(void)
{
    u8 i;
    u8 status;

    GS_i2c_init();//IIC 初始化
    status = paj7620u2_wakeup();//唤醒 PAJ7620U2
    if(!status) return 0;
    paj7620u2_selectBank(BANK0);//进入 BANK0 寄存器区域
    for(i=0;i<INIT_SIZE;i++)
```

```
    {
        GS_Write_Byte(init_Array[i][0],init_Array[i][1]);//初始化 PAJ7620U2
    }
    paj7620u2_selectBank(BANK0);//切换回 BANK0 寄存器区域

    return 1;
}
```

该函数实现对 PAJ7620U2 初始化，先对 IIC 通信的 IO 引脚初始化，然后调用 paj7620u2_wakeup()函数对 PAJ7620U2 进行唤醒，唤醒成功后，由于后面要调用 init_Array 初始化数组，初始化时是从 BANK0 区域开始的，所以这里先选择进入 BANK0 区域，初始化数组配置完毕后，又切换回 BANK0 区域，这时 PAJ7620U2 初始化完成。

（4）Gesrure_test()函数，具体代码如下：

```
//手势识别测试
void Gesrure_test(void)
{
    u8 i;
    u8 status;
    u8 key;
    u8 data[2]={0x00};
    u16 gesture_data;
    u8 ledflash=0;

    paj7620u2_selectBank(BANK0);//进入 BANK0
    for(i=0;i<GESTURE_SIZE;i++)
    {
        GS_Write_Byte(gesture_arry[i][0],gesture_arry[i][1]);//手势识别模式初始化
    }
    paj7620u2_selectBank(BANK0);//切换回 BANK0
    i=0;
    POINT_COLOR=BLUE;//设置字体为蓝色
    LCD_Fill(30,170,300,300,WHITE);
    LCD_ShowString(30,180,200,16,16,"KEY_UP: Exit the test");
    LCD_ShowString(30,210,200,16,16,"Gesrure test");
    POINT_COLOR=RED;//设置字体为蓝色
    while(1)
    {
        key = KEY_Scan(0);
        if(key==WKUP_PRES)
        {
            GS_Write_Byte(PAJ_SET_INT_FLAG1,0X00);//关闭手势识别中断输出
            GS_Write_Byte(PAJ_SET_INT_FLAG2,0X00);
            break;
        }
        status = GS_Read_nByte(PAJ_GET_INT_FLAG1,2,&data[0]);//读取手势状态
```

```
if(!status)
{
    gesture_data =(u16)data[1]<<8 | data[0];
    if(gesture_data)
    {
        switch(gesture_data)
        {
            case   GES_UP:
                   LCD_ShowString(110,250,200,16,24,"UP                ");
                   printf("Up\r\n");    ledflash=1; break; //向上
            case   GES_DOWM:
                   LCD_ShowString(100,250,200,16,24,"Dowm              ");
                   printf("Dowm\r\n"); ledflash=1; break; //向下
            case   GES_LEFT:
                   LCD_ShowString(100,250,200,16,24,"Left              ");
                   printf("Left\r\n"); ledflash=1; break; //向左
            case   GES_RIGHT:
                   LCD_ShowString(100,250,200,16,24,"Right             ");
                   printf("Right\r\n"); ledflash=1; break; //向右
            case   GES_FORWARD:
                   LCD_ShowString(80,250,200,16,24,"Forward          ");
                   printf("Forward\r\n"); ledflash=1; break; //向前
            case   GES_BACKWARD:
                   LCD_ShowString(80,250,200,16,24,"Backward         ");
                   printf("Backward\r\n"); ledflash=1; break; //向后
            case   GES_CLOCKWISE:
                   LCD_ShowString(70,250,200,16,24,"Clockwise      ");
                   printf("Clockwise\r\n"); ledflash=1; break; //顺时针
            case   GES_COUNT_CLOCKWISE:
                   LCD_ShowString(50,250,200,16,24,"AntiClockwise");
                   printf("AntiClockwise\r\n"); ledflash=1; break; //逆时针
            case   GES_WAVE:
                   LCD_ShowString(100,250,200,16,24,"Wave              ");
                   printf("Wave\r\n"); ledflash=1; break; //挥动
            default:    ledflash=0; break;
        }
        if(ledflash)//DS1 闪烁
        {
            LED1=0;delay_ms(80);LED1=1;delay_ms(80);
            LED1=0;delay_ms(80);LED1=1;delay_ms(80);
            delay_ms(300);
            LCD_ShowString(40,250,200,16,24,"                                  ");
            ledflash=0;
```

```
                    }
                }
            }
            delay_ms(50);
            i++;
            if(i==5)
            {
                LED0=!LED0;//提示系统正在运行
                i=0;
            }
        }
    }
```

该函数实现手势识别功能，一开始先对 PAJ7620U2 手势识别检测进行初始化，配置其 gesture_arry 数组的参数值。这里说一下，该数组有使能 9 个手势识别的中断标志输出的配置，初始化完毕后，在 while 循环中一直读取手势中断标志寄存器，当识别到相应的手势，手势标志会置 1，读取寄存器标志会自动清 0。根据手势的标志，在 LCD 屏幕上显示识别到的手势结果，DS1 灯会闪烁，同时手势结果打印到串口上，DS0 灯闪烁表示检测正在运行。当按下 KEY_UP 按键，关闭 9 个手势识别的检测输出，退出测试。

（5）Ps_test()函数，具体代码如下：

```
//接近检测测试
void Ps_test(void)
{
    u8 i;
    u8 key;
    u8 data[2]={0x00};
    u8 obj_brightness=0;
    u16 obj_size=0;
    paj7620u2_selectBank(BANK0);//进入 BANK0
    for(i=0;i<PROXIM_SIZE;i++)
    {
        GS_Write_Byte(proximity_arry[i][0],proximity_arry[i][1]);//接近检测模式初始化
    }
    paj7620u2_selectBank(BANK0);//切换回 BANK0
    i=0;
    POINT_COLOR=BLUE;//设置字体为蓝色
    LCD_Fill(30,170,300,300,WHITE);
    LCD_ShowString(30,180,200,16,16,"KEY_UP: Exit the test");
    LCD_ShowString(30,210,200,16,16,"Ps test");
    LCD_ShowString(30,240,200,16,16,"Brightness");
    LCD_ShowString(160,240,200,16,16,"Size");
    POINT_COLOR=RED;//设置字体为蓝色

    while(1)
    {
```

```
            key = KEY_Scan(0);
            if(key==WKUP_PRES) break;

            obj_brightness = GS_Read_Byte(PAJ_GET_OBJECT_BRIGHTNESS);//读取物体亮度
            data[0] = GS_Read_Byte(PAJ_GET_OBJECT_SIZE_1);//读取物体大小
            data[1] = GS_Read_Byte(PAJ_GET_OBJECT_SIZE_2);
            obj_size = ((u16)data[1] & 0x0f)<<8 | data[0];
            LCD_ShowxNum(50,270,obj_brightness,3,24,0);
            LCD_ShowxNum(152,270,obj_size,3,24,0);
            printf("obj_brightness = %d\r\n",obj_brightness);
            printf("obj_size:%d\r\n",obj_size);
            delay_ms(100);
            i++;
            if(i==5)
            {
                LED0=!LED0;//提示系统正在运行
                i=0;
            }
        }
    }
```

该函数实现接近检测，获取接近物体的体积大小和亮度的数据。同样开始对接近检测进行初始化，配置其 proximity_arry 的数组，初始化完成后，在 while(1)循环中，读取获取体积大小和亮度的寄存器，读到的数据显示在 LCD 屏幕上，同时打印到串口上，DS0 闪烁表示检测正在运行，当按下 KEY_UP 按键，退出测试。

最后我们说下 main.c，代码如下：

```
int main(void)
{
    delay_init(168);                    //初始化延时函数
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分组 2
    uart_init(115200);                  //初始化串口波特率为 115200
    LED_Init();                         //初始化 LED
    LCD_Init();                         //LCD 初始化
    KEY_Init();                         //按键初始化
    usmart_dev.init(84);                //初始化 USMART
    POINT_COLOR=RED;                    //设置字体为红色
    LCD_ShowString(30,50,200,16,16,"Explorer STM32");
    LCD_ShowString(30,70,200,16,16,"Sensor PAJ7620U2 TEST");
    LCD_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");
    LCD_ShowString(30,110,200,16,16,"2017/8/23");
    POINT_COLOR=BLUE;                   //设置字体为蓝色
    while(!paj7620u2_init())//PAJ7620U2 传感器初始化
    {
        printf("PAJ7620U2 Error!!!\r\n");
        LCD_ShowString(30,140,200,16,16,"PAJ7620U2 Error!!!");
```

```
            delay_ms(500);
            LCD_ShowString(30,140,200,16,16,"                              ");
            delay_ms(500);
            LED0=!LED0;//DS0 闪烁

        }
        printf("PAJ7620U2 OK\r\n");
        LCD_ShowString(30,140,200,16,16,"PAJ7620U2 OK");
        while(1)
        {
            paj7620u2_sensor_test();//PAJ7620U2 传感器测试
        }
    }
```

此部分代码比较简单，对用到的外设进行初始化，然后通过调用 paj7620u2_sensor_test()函数，进入 ATK-PAJ7620 模块的主测试程序，对 ATK-PAJ7620 的手势识别和接近检测功能进行测试。

另外，为了方便大家调试，我们在本例程的 USMART 添加了 GS_Write_Byte 和 GS_Read_Byte 两个函数，这样，我们就可以通过串口调试助手，改写和读取 PAJ7620U2 的寄存器数据了，方便大家调试。

至此，软件实现部分介绍完了，我们接下来看代码验证。

4、验证

首先，请先确保硬件都已经连接好了：

1，ATK-PAJ7620 模块与 ALIENTEK 探索者 STM32F407 开发板连接（连接方式见 2.3 小节）

2，ALIENTEK 探索者 STM32F407 开发板插上 TFTCLD 液晶。

3，給 ALIENTEK 探索者 STM32F407 开发板供电。

代码编译成功之后，我们将代码下载到 STM32 开发板上。LCD 界面显示如图 4.1 所示：

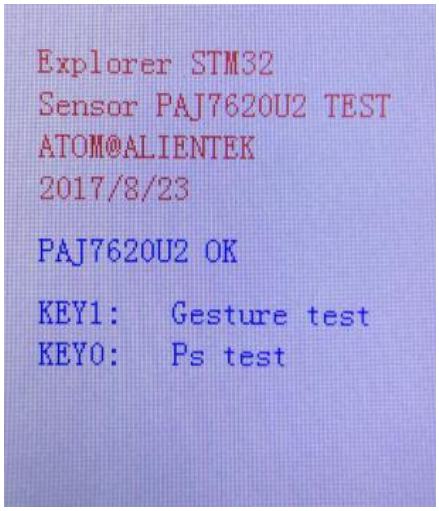

图 4.1 本测试实验界面

可以看到，LCD 屏幕显示 PAJ7620U2 OK，表示 PAJ7620U2 传感器初始化通过，同时显示了 KEY1/KEY0 测试功能选项，KEY1 手势识别测试、KEY0 接近检测测试。

4.1 手势识别测试

在主菜单界面，按 KEY1，则可进入此项测试，此项测试为手势识别测试，主界面如图 4.1.1 所示：

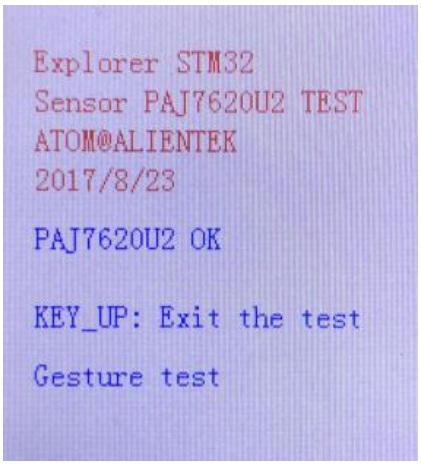

图 4.1.1 手势识别测试界面

当手在 ATK-PAJ7620 模块上方手势滑动，会识别出 9 个的手势识别，识别结果显示在 LCD 屏幕上，同时 DS1 闪烁。如图 4.1.2 为“后”的识别效果。

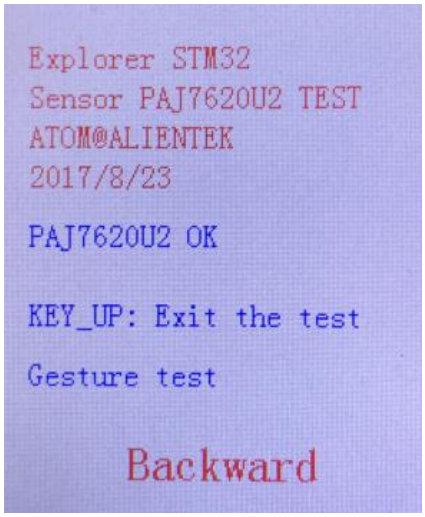

图 4.1.2 手势识别

按下 KEY_UP 按键可退出手势识别测试，返回主菜单页面。

4.2 接近检测测试

在主菜单页面，按 KEY0，可进入此项测试，此项测试接近检测功能。接近检测测试主界面如图 4.2.1 所示：

图 4.2.1 接近检测测试界面

当手在模块上方动作时，模块检测到物体的体积大小和亮度数据也在变化如图 4.2.2 所示。

图 4.2.2 接近检测

最后，按下 KEY_UP 按键可退出接近检测测试，返回主菜单页面。

至此，关于 ATK-PAJ7620 模块的使用介绍，我们就讲完了，本文档介绍了 ATK-PAJ7620 模块的使用，有助于大家快速学会 ATK-PAJ7620 模块的使用。

正点原子@ALIENTEK
2017-12-4
公司网址：www.alientek.com
技术论坛：www.openedv.com
资料下载地址：http://www.openedv.com/thread-233690-1-1.html
电话：020-38271790
传真：020-36773971