1.写在前面

AD7606 接口包含以下文件：

ad7606.h
ad7606.c

include.h
gpio.c
spi.c

testADC.c
Makefile

其中，include.h gpio.c spi.c 三个文件是对 gpio 子系统和 spi 子系统的封装，供 ad7606 接口层 API 调用

ad7606.h ad7606.c 是对 ad7606 芯片的直接控制接口
testADC.c Makefile 是测试程序以及组织脚本

本文档针对 ad7606 接口控制层作解释说明，底层子系统可以视为透明处理、

2.使用说明

将测试程序以及组织脚本以外的源代码拷贝到工程目录，并在核心处理层将 ad7606.h 的路径正确包含，并对工程的组织脚本做必要检查

3.接口函数使用说明

（1）ad7606 设备描述结构体

对一个 ad7606 设备的具体描述，其中包括 gpio 控制引脚，gpio 设备描述符，spi 接口设备描述符及设备文件路径

```
struct DeviceInfo{
    //gpio 引脚
    unsigned int num_cov;//开始转换信号
    unsigned int num_busy;//正在转换信号
    unsigned int num_cs;//片选信号
    unsigned int num_rest;//复位信号
    unsigned int num_os0;//速率选择第 0 位
    unsigned int num_os1;//速率选择第 1 位
    unsigned int num_os2;//速率选择第 2 位

    //以上 gpio 引脚对应以下的设备文件描述符
    int fd_cov;
    int fd_busy;
    int fd_cs;
    int fd_rest;
    int fd_os0;
    int fd_os1;
    int fd_os2;

    //spi 设备文件相关
    char *device;//spi 设备接口绝对路径
    int fd_spi;//spi 设备文件描述符

    };
```

（2）初始化 gpio

初始化 gpio 流程描述：

注册 gpio-->创建 gpio 目录-->打开 gpio 设备-->gpio 引脚电平初始化

参数：@*spi spi 设备描述结构体

执行成功返回 1，失败返回 0

int gpio_init(struct DeviceInfo *spi);

（3）初始化 spi 设备

初始化 spi 设备流程描述：

打开 spi 设备-->设置 spi 参数-->spi 设备复位

参数：@*spi spi 设备描述

执行成功返回 1，失败返回 0

int spi_init(struct DeviceInfo *spi);

（4）从 spi 接口读取数据，每次读取一组

数据读取流程描述：

转换信号：拉低 cov-->拉高 cov

获取 busy 电平：可以忽略

片选信号：拉低 cs

读取数据

恢复片选信号：拉高 cs

参数：@spi spi 设备描述

@buf 数据缓冲区

@len 读取长度

执行成功返回 1，失败返回 0

int spi_read_data(struct DeviceInfo *spi,uint8_t *buf,int len);

（5）接受指定组数的数据

反复执行 spi_read_data 函数，直到完成期待读取的个数

参数：@spi spi 设备描述

@buf 二维数据缓冲区，一维空间指示数据组数，二维空间描述具体数据

@num 读取组数

执行成功返回 1，失败返回 0

int spi_read_numdata(struct DeviceInfo *spi,uint8_t **buf,int num);

（6）关闭设备

关闭设备流程描述：

关闭 spi 设备-->关闭 gpio 设备-->注销 gpio 引脚

参数：@*spi spi 设备描述

执行成功返回 1，失败返回 0

int close_device(struct DeviceInfo *spi);

（7）将数据显示到终端，调试函数

参数：@*buf 数据缓冲

@len 数据长度

void show_data(uint8_t *buf,int len);

4.测试程序

进入目录，执行:$make

将生成的目标程序拷贝到开发板，进行验证