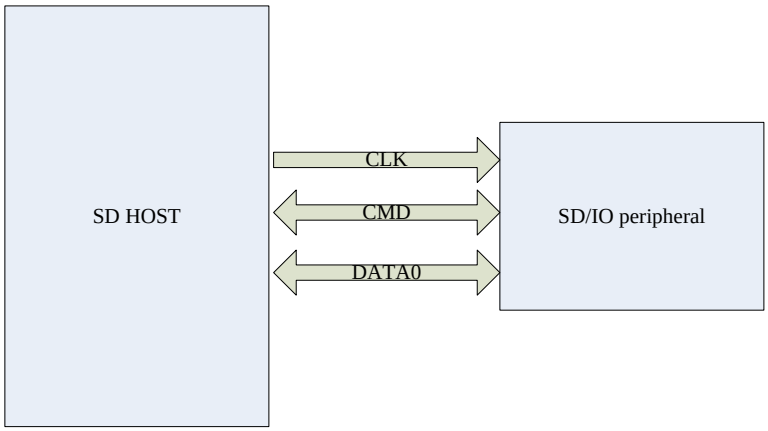
SDIO

1 SDIO 模块简介

支持 SDIO 接口标准。SD 是记忆卡的标准，SDIO 是在 SD 标准上定义了一种外设接口，就是 SD 的 I/O 接口的意思，SDIO 接口兼容以前的 SD 内存卡。SDIO controller 通过 SDIO 来和 SD 卡或者其它外设实现数据交互。

SDIO 协议保留了 SD 卡的读写协议，同时 SDIO 协议在 SD 卡协议置上添加了 CMD52/CMD53 命令。SDIO 和 SD 卡规范间的一个重要区别是增加了低速标准。SDIO 卡只需要 SPI 和 1 位 SD 传输模式。低速卡的目标应用是以最小的硬件开支支持低速 I/ O 能力。

SDIO controller 支持的时钟范围为 0 到 24MHz，在 SDIO 总线定义中，SDIO 的 1BIT 模式下 DAT0 用来传输数据，DAT1 用作中断线。但 O18 的 SDIOcontroller 不支持 DAT1 用作中断线。若需要只能将 DAT1 连接到 GPIO，利用 GPIO 的中断。其信号引脚定义如下：



2 SDIO 模块驱动接口设计

考虑到可能的对 SDIO 各种外设的支持，需要实现支持任意长度字节的传输，对 driver 进行层次划分，将 SDIO 驱动作为底层，所有 SDIO 外设包括 SD 卡的驱动都调用 SDIO controller 的底层驱动。

2.1 SDIO 相关说明

1、SDIO 所有 response 都经过 CMD 线，总共有 8 种 response type，R1~R7,其中 R1 分 R1 和 R1B 两种，有细微差别。

2、response 长度取决于响应类型。其中除了 R2 的 response 长度为 136bits 其余皆为 48bits。所以设置获取 response code length 的时候，若 response type 为 R2 请设置 16，其余设置为 5。

3、对 SD 协议而言，R4/R5 是 SDIO 额外支持的 response type。

2.2 函数列表

VOID SdioControllerInit(VOID)
VOID SdioEnableClk(VOID)
VOID SdioDisableClk(VOID)
VOID SdioSetClk(DWORD ClkIndex)
VOID SdioStartReciveData(BYTE* InBuf, WORD DataLen)
VOID SdioStartSendData(BYTE* OutBuf, WORD DataLen)
BOOL SdioSendCommand(BYTE Cmd, DWORD Param, WORD TimeOut)
VOID SdioGetCmdResp(BYTE *RespBuf, BYTE RespBufLen)
BOOL SdioIsDatTransDone(VOID)
BYTE SdioGetStatus(VOID)
BYTE SdioGetResponseStatus(VOID)
BOOL SdioIsBusy(VOID)
VOID SdioEndDatTrans(VOID)

基于不是所有 CMD 都会有响应，定义了超时机制与 ERR CODE。ERR CODE 用宏定义：
#define NO_ERR (0)
#define SEND_CMD_TIME_OUT_ERR (1)

3 SDIO 模块接口函数描述

3.1 SdioControllerInit

原型：
VOID SdioControllerInit(VOID)
参数：
VOID
描述：
初始化sdio controller
返回值：
VOID

3.2 SdioSetClk

原型：
VOID SdioSetClk(DWORD ClkIndex)

参数：

ClkIndex：分频参数，0代表2分频，1代表4分频，依次类推

描述：

将提供给sd卡的clk进行分频

返回值：

VOID

3.3 SdioEnableClk

原型：

VOID SdioEnableClk(VOID)

参数：

VOID

描述：

打开sd模块clk

返回值：

VOID

3.4 SdioDisableClk

原型：

VOID SdioDisableClk(VOID)

参数：

VOID

描述：

关闭sd模块clk

返回值：

VOID

3.5 SdioSendCommand

原型：

BOOL SdioSendCommand(BYTE Cmd, DWORD Param, WORD TimeOut)

参数：

Cmd：要发送的命令

Param：要发送的命令伴随的参数

TimeOut：等待发送命令完成超时

描述：

发送命令给sdio外围设备

返回值：

NO_ERR：发送命令成功

SEND_CMD_TIME_OUT_ERR 获得CMD done信号超时

3.6 SdioGetCmdResp

原型：

VOID SdioGetCmdResp(BYTE *RespBuf, BYTE RespBufLen)

参数：

RespBuf：存储response

RespBufLen：需要获得的response长度，请参看章节2.1或者SDIO SPECIFICATION

描述：

获得所发送命令的响应

返回值：

VOID

3.7 SdioStartReciveData

原型：

VOID SdioStartReciveData(BYTE* InBuf, WORD DataLen)

参数：

InBuf：将接收到的数据存入此buf

DataLen：需要接收的数据长度

描述：

接收sdio数据总线上的数据

返回值：

VOID

3.8 SdioStartSendData

原型：

VOID SdioStartSendData(BYTE* OutBuf, WORD DataLen)

参数：

OutBuf：将需要发送的的数据存入此buf

DataLen：需要发送的数据长度

描述：

开始数据传输

返回值：

VOID

3.9 SdioGetResponseStatus

原型：

BYTE SdioGetResponseStatus(VOID)

参数：

VOID

描述：

从command的response中获得card status。

返回值：

VOID

3.10 SdioGetStatus

原型：

BYTE SdioGetStatus(VOID)

参数：

VOID

描述：

每个block data写入card，将会有一个BYTE的response，该函数将获得这些response。

返回值：

Sd status

3.11 SdioIsDatTransDone

原型：

BOOL SdioIsDatTransDone(VOID)

参数：

VOID

描述：

等到data传输结束信号

返回值：

1 means done。

3.12 SdioEndDatTrans

原型：

VOID SdioEndDatTrans(VOID)

参数：
VOID
描述：
结束本次传输
返回值：
VOID

3.13 SdioIsBusy

原型：
BOOL SdioIsBusy (VOID)
参数：
VOID
描述：
指示sd卡是否busy
返回值：
0 means busy

4 典型应用

下面以 SDK 中 SD 卡驱动中对 SDIO 的应用为例进行介绍

a. 检测 SD 卡是否连接

卡座有卡插人检测引脚，检测该引脚的电平就可以判断是否有卡插入。

b. 配置 sdio 接口在 GPIO 上的位置

调用 GpioSdIoConfig (uint8_t ModeSel)函数，参数 ModeSel 为 0,1,2,3 表示选取相应位置的 GPIO 口作为 SDIO 接口，参数 ModeSel 为 0xFF 表示将上述指定位置引脚恢复为 GPIO。引脚对应关系如下：

0: sd0
sd_cmd(io) a[2]
sd_clk(o) a[3]
sd_dat(io) a[4]
1: sd1
sd_cmd(io) a[21]
sd_clk(o) a[20]
sd_dat(io) a[19]
2: sd2
sd_cmd(io) b[5]

sd_clk(o) b[4]
sd_dat(io) b[3]
3: sd3
sd_cmd(io) b[20]
sd_clk(o) b[21]
sd_dat(io) b[22]

c. 初始化 SDIO，设置工作频率

调用 SdioControllerInit ()初始化 Sdio Controller 后通过 SdCardGetInfo()获取卡相关信息(如：支持的时钟频率)，再设置工作频率 SdioSetClk(..)

d. 收发数据

1. 发送读/写命令
2. 设置收发 buffer，指定目标 sector 号
3. 等待传输完成
4. 停止数据传输
5. 发送停止命令

示例代码：

```
SdioSendCommand(18/*CMD18_READ_MULTIPLE_BLOCK*/, 0, 20);
SdioStartReciveData(Sec, 512/*SD_BLOCK_SIZE*/);
while(!SdioIsDatTransDone());
SdioStartReciveData(&Sec[512], 512/*SD_BLOCK_SIZE*/);
while(!SdioIsDatTransDone());
SdioEndDatTrans();
SdioSendCommand(12/*CMD12_STOP_TRANSMISSION*/, 0, 20);
```

SDIO所有通信都是由Host端发出命令开始，先有Host端发出相应命令 SdioSendCommand(BYTE Cmd, DWORD Param, WORD TimeOut)，按照SD2.0协议中的命令配置相应参数，不同命令的流程不同（具体参见《SD2.0协议栈应用和说明文档》），再调用相应的数据收发函数完成数据传输。