# LED

代码中将每个LED看做一个Led类的实例，创建Led实例的时候需要知道该Led的名字（LED的名字可以从设备树 或 “/sys/class/leds/”目录下查找）。

## Led构造函数

Led::Led(string ledName);

参数：

ledName：Led在系统中的名字

## LED初始化

int Led::ledInit();

返回值:

0 : 成功

1 : 失败

## 点亮LED

int Led::lightOn();

返回值:

0 : 成功

1 : 失败

## 关闭LED

int Led::lightOff();

返回值:

0 : 成功

1 : 失败

## LED闪烁

int Led::startBlink(int hz)

参数：

hz ： 每秒闪烁的次数

返回值:

0 : 成功

1 : 失败

int Led::startBlink(int delayOn, int delayOff)

参数：

delayOn：每个周期亮的时间(ms)

delayOff: 每个周期灭的时间(ms)

返回值:

0 : 成功

1 : 失败

## LED停止闪烁

int Led::stopBlink()

返回值:

0 : 成功

1 : 失败

# KEY

所有的按键都是通过Key类实现监听，创建Key实例需要知道Key在系统中的名字，可以通过设备树或者读取/sys/class/input/eventX/device/name来判断。

## Key构造函数

Key::Key(string name);

参数：

name：Key的name

## 注册按键事件回调函数

void Key::setKeyCB(KeyCB cb);

参数：cb 用户自定义按键处理函数

typedef void (\*KeyCB)(unsigned char keyCode, KeyStatus pressed);

pressed:

typedef enum{

KEY\_NONE\_STATUS = 0,

KEY\_PRESSED ,

KEY\_RELEASED

}KeyStatus;

keyCode:

#define KEY\_START 31

#define KEY\_UPDATE 22

#define KEY\_DESTROY 18

## 开启Key Monitor

pthread\_t Key::startKeyMonitor();

返回值：

>0： key monitor的线程ID

其他: 开启失败

## 停止Key Monitor

void Key::stopKeyMonitor();

此外还可以通过Key::execFlag = 0; 来停止Monitor

# UsbDetector

通过UsbDector的创建Monitor来监控usb和Sd卡的插拔，需要在创建实例的时候绑定输入端口、输出端口和sd卡的host在内存中的地址，该地址可以从设备树种查询或在各端口都有设备插入的时候在“/sys/class/block”目录执行ll查看。

## 构造函数

UsbDetector::UsbDetector(string inputHostAddr, string outputHostAddr, string sdcardHostAddr)

参数：

inputHostAddr：输入端口host的内存地址

outputHostAddr：输出端口host的内存地址

sdcardHostAddr：sd卡host的内存地址

## 注册设备插拔回调函数

void UsbDetector::setUsbDetectorCB(UsbDetectorCB cb);

参数：

cb ：设备插拔回调函数

typedef void (\*UsbDetectorCB)(vector<string> inputUsb,

vector<string> outputUsb,

vector<string> sdCard);

参数：

inputUsb：当前的inputUsb存储设备

outputUsb：当前的outputUsb存储设备

sdCard：当前的sd卡存储设备

## 注册USB monitor线程退出处理函数

void UsbDetector::setMonitorExitCb(UsbDetectorMonitorExitCb cb);

参数：

cb ：USB monitor 线程退出处理函数

typedef void (\*UsbDetectorMonitorExitCb)();

## 开始Usb detector monitor

pthread\_t UsbDetector::startMonitor()

返回值：

>0 : usb monitor的线程Id

其他：开启失败

## 停止Usb detector monitor

void UsbDetector::stopMonitor()

此外还可以通过UsbDetector::execFlag = 0; 来停止Monitor

# FPGA

fpga设备可以在“/dev”目录下查找“fpga-enc”，每次调用生成数据接口生成固定一页大小（sysconf(\_SC\_PAGE\_SIZE)）的数据块。

## 构造函数

Fpga::Fpga(string name)

参数：

name：FPGA的设备名，“/dev/fpga-enc”

## 初始化函数

unsigned char \* Fpga::init()

返回值：返回一个unsigned char指针指向FPGA生成数据的数据区，数据大小为一页

如果返回NULL，初始化失败

## 初始化秘钥：

int Fpga::setSeed(struct fpga\_seed \*seed)

参数seed：

struct fpga\_seed{

unsigned char \*seed;

int seed\_len;

};

seed.seed:指向秘钥地址

seed.seed\_len:秘钥长度，现在的限制是最大128个字节，长度限制可以在设备树中进行修改“max-seed-length”

返回值：

<0: 设置秘钥失败

## 生成数据：

int Fpga::genData()

返回值：

<0: 生成数据失败