# 扩展驱动

打开项目K210MV-APP，usrdev目录里都是用户扩展驱动，框架默认已经提供了几个扩展驱动以提供参考，复制其中一个驱动文件，比如 usrdev/lua\_helloword.cpp –> usrdev/lua\_firstdriver.cpp, **lua\_firstdriver.cpp**就是我们要实现的第一个扩展驱动，我们一下所有 更改都是基于这个文件，让我们体会到在microlua框架下扩展一个驱动是多么方便。

修改lua\_firstdriver.cpp代码，因为lua\_firstdriver.cpp代码最初来自lua\_helloword.cpp 我们先展现一下lua\_helloword.cpp的代码

Helloword 驱动代码说明

DEV\_HANDLE hDevHelloHandle = NULL;这一句是要声明一个全局设备句柄，也就是说这个驱动的全局句柄，我们可以把它改为**DEV\_HANDLE hDevGPIOLedHandle = NULL**

我们先看一下helloword函数，这个函数可以在lua脚本里面调用,这个函数如何在lua展现出来我们稍后再讨论这个问题。

int helloword\_fun(FUN\_HANDLE hFunHandle) {

printf("helloword\_fun\r\n");

return 0;

}

我们要控制LED需要一个函数来完成，我们把它命名为

**int switchled\_fun(FUN\_HANDLE hFunHandle) {**

**printf("switchled \_fun\r\n");**

**bool ret = arg\_toboolean(hFunHandle,2);**

**return 0;**

**}**

用它来控制led 就像 led(flase)或led(true) ，它需要一个boolean做为函数参数，我们把它改成这样,参数的索引值是2，为啥是2呢，因为1是这个模块本身，arg\_toboolean(hFunHandle,2);是从lua脚本中取出这个boolean值,不同类型的值对应取值函数不一样，有趣吧，脚本本质可以承载不同参数类型，不同参数数目，当然你也可以设置多个参数，但索引值是从2,3,4,5,6,7等开始的，这有点别扭，但lua原始就是这么干的,。

我们还发现一个函数，这个函数是干嘛的呢，因为开发者想要异步收到事件，比如串口数据来、定时器时间到了，这时就需要一个回调函数通知，通过SendModuleEvent(hDevHelloHandle);函数发送通知

int helloword\_event(FUN\_HANDLE hFunHandle) {

printf("helloword\_fun\r\n");

SendModuleEvent(hDevHelloHandle);

return 0;

}

通过这个函数来注册回调函数

int helloword\_callback(FUN\_HANDLE hFunHandle) {

printf("helloword\_callback\r\n");

arg\_tocallback(hFunHandle,2);

return 0;

}

这两个函数是配对出现的，但我们要实现的ledk控制其中SendModuleEvent(hDevHelloHandle);也可以单独在中断调用，比如硬件中断，例子代码是从lua调用这个函数，现实中往往不是这样的，大多数是从中断里调用的，对于大多是微控制器而然,这次led控制并不需要回调事件，所以我们忽略这两个函数

通常而然我们需要打印一下这个驱动信息,框架提供了一个这样的函数

int helloword\_tostring(FUN\_HANDLE hFunHandle) {

printf("helloword\_tostring\r\n");

char szinfo[1024] = { 0 };

snprintf(szinfo, sizeof(szinfo), "%s", getModuleName(hFunHandle));

pushstring\_ret(hFunHandle, szinfo,strlen(szinfo));

return 1;

}

我们把它改名为如下,这个函数可以把led控制状态信息打印出来，对于复杂的驱动我们需要打印的信息更多。

**int switchled\_tostring(FUN\_HANDLE hFunHandle) {**

**printf("switchled\_tostring\r\n");**

**char szinfo[1024] = { 0 };**

**snprintf(szinfo, sizeof(szinfo), "%s", getModuleName(hFunHandle));**

**pushstring\_ret(hFunHandle, szinfo,strlen(szinfo));**

**return 1;**

**}**

最后我们重点说明一下这个函数,这个函数体现了驱动和lua脚本之间的关联，

addModule("hello",helloword\_tostring,NULL);这个函数把驱动模块加入lua脚本里面

其中"hello"是驱动名，你可以在lua通过 board.usrdev(“hello”)来引用它,

addFuntion(hDevHelloHandle,"helloword",helloword\_fun);是把helloword\_fun加入这个驱动模块，通过

--这个是lua脚本

hello = board.usrdev(“hello”)

hello: helloword() 来引用它，其中"helloword"是脚本中引用的函数名

void helloword\_init() {

hDevHelloHandle = addModule("hello",helloword\_tostring,NULL);

assert(hDevHelloHandle != NULL);

addFuntion(hDevHelloHandle,"helloword",helloword\_fun);

addFuntion(hDevHelloHandle,"sendevent",helloword\_event);

addFuntion(hDevHelloHandle,"callback",helloword\_callback);

}

我们把这个函数改成这样

**void switchled \_init() {**

**hDevGPIOLedHandle = addModule("ledtest", switchled\_tostring,NULL);**

**assert(hDevGPIOLedHandle!= NULL);**

**addFuntion(hDevGPIOLedHandle,"led", switchled\_fun)；**

**}**

**最后我们在main.cpp initCustomDevice()函数里加入这个switchled \_init()初始化函数，以便在程序启动时调用它**

那么我们可以看到调用的lua代码是

Ledtest1 = board.usrdev(“ledtest”) –装载驱动

Ledtest1: led(true) --开led

Ledtest1: led(false) –-关led

print(Ledtest1) --打印驱动信息