前言：

这里主要介绍Gui与系统的关系，如果要设计裸机流程，则进行一些简单改造即可

本Gui是以事件调度为模型，需要依赖一些基础的数据结构组件，Gui对操作系统的依赖度不高，你需要有多线程接口以及基本的同步原语（互斥锁和信号量）即可

嵌入式Gui通常需要有三个线程，Gui主事件调度线程，Disp异步送显线程，输入设备事件投递线程（输入设备事件投递也可能是中断服务例程中），最终三个线程通过同步原语保持着恰当的调度节奏，完成事件调度与绘制送显

我们需要有一个桥梁去构建三个线程的关联关系，通常我们将这个叫做事件管理器，将线程中的事件，通过传递到事件管理器之中，通过Gui主事件调度，通过信号量去触发对应的响应

例：一个Touchpad产生了一个屏幕报点事件，向Gui的事件调度管理器报告一个屏幕报点事件（传递Tp报点），并且传递一个信号量，收到该信号量则激活Gui的主事件调度，如果产生重绘，则进一步产生一个送显事件（传递帧缓冲区），并且传递一个信号量，收到该信号量则激活Disp的异步送显流程，将帧缓冲区的数据通过驱动传递给屏幕，完成Disp刷新

以下是Gui的相关线程构建情况：

/\*@brief scui refr子线程

\*/

static APP\_THREAD\_GROUP\_HANDLER(app\_thread\_scui\_refr\_routine)

{

while (true) {

/\*@brief scui 屏幕刷新回调接口

\*/

void app\_dev\_gui\_disp\_scui\_flush(scui\_surface\_t \*suface);

scui\_surface\_refr\_routine(app\_dev\_gui\_disp\_scui\_flush);

app\_delay\_ms(10); // 等待刷新完毕

}

}

这是送显相关的线程，通过向底层驱动回调投递一个送显画布，完成送显，该例程内部会等待信号量，只有绘制结束后，才会产生送显信号量，所以该动作由送显信号量激发，并且送显结束后进入到下一个信号量的等待中

**备注：双Buffer线程异步可以通过上面的scui\_surface\_refr\_routine去获得参考实现**

/\*@brief scui draw子线程

\*/

static APP\_THREAD\_GROUP\_HANDLER(app\_thread\_scui\_draw\_routine)

{

while (true) {

scui\_engine\_wait();

scui\_engine\_execute();

}

}

这是主调度线程，它会等待到来引擎的一个事件，并且事件会产生一个信号量，消耗该信号量完成一次事件调度，然后进入到下一个信号量的等待中

/\*@brief scui 输入设备回调接口

\*/

void app\_dev\_gui\_ptr\_scui\_read(scui\_indev\_data\_t \*indev\_data);

void app\_dev\_gui\_enc\_scui\_read(scui\_indev\_data\_t \*indev\_data);

void app\_dev\_gui\_key\_scui\_read(scui\_indev\_data\_t \*indev\_data);

/\* \*/

scui\_indev\_data\_t indev\_data;

/\* 事件派发 \*/

app\_dev\_gui\_ptr\_scui\_read(&indev\_data);

scui\_indev\_notify(&indev\_data);

app\_dev\_gui\_enc\_scui\_read(&indev\_data);

scui\_indev\_notify(&indev\_data);

app\_dev\_gui\_key\_scui\_read(&indev\_data);

scui\_indev\_notify(&indev\_data);

这是输入设备的事件调度，它会周期的产生pointer（指针类型，TP报点）,encode（编码器事件）,key（按键） 事件，这个处理流程会计算生成Gui所需要的输入事件，并且向主调度事件管理器发送对应的输入设备被翻译后的事件