系统代码修改规范

	部门	系统代码修改规范
文件状态：	版本	0.2
[√] 草稿	作者	杨贵平
[ ] 正在修改	完成时间	**20170718**
[ ] 正式发布	审核	
	审核时间	
密级状态：绝密( ) 秘密( ) 内部资料(√) 公开( )		

修改记录:

版本	修订者	时间	说明
0.1	杨贵平	2017/07/18	基本版本, 初步的修改要求

一. 背景及问题:

系统代码因为主要考虑基于原厂提供的公版代码做修改， 且涉及自主修改，公版 patch， 多人维护交叉修改等问题。为统一格式， 方便代码识别，增加可读性。特别定义此规范。

二. 思路和方法:

规范主要是针对系统代码修改的，分成三部分， 分别是针对 TAB 处理， 针对回车换行的处理， 修改的识别三个方面做出明确的定义要求。

另外，针对新加的代码，尽量按标准写. 虽然不是强制要求，但还是要尽量严谨一些.

1. 针对 TAB 处理

代码中，行对齐的 TAB 键， 要换成空格， 不允许直接使用 TAB 键。
原因是不同的编辑器对 TAB 的解释都不一样， 造成对齐的混乱。
所以在使用 UE 编辑代码之后，记得使用"格式/转换制表符为空格"
这个在原厂的代码中可能本身就有这样的问题. 修改时碰到了，就一并转换过来.

2. 针对回车换行的处理：

因为系统代码一般是在 linux 服务器上编译的。 换行符全部要使用 unix 的换行符。大家一般都是在 windows 下编辑修改代码的， 所以很容易出现 windows 换行符。
出现 windows 换行符时，在 git diff 时， 可以看到行尾出现^M 的字符. 这个可以做为检查修改的方法.

```
diff --git a/drivers/input/sensors/sensor-dev.c b/drivers/input/sensors/sensor-dev.c
index d7536fc..4107539 100755
--- a/drivers/input/sensors/sensor-dev.c
+++ b/drivers/input/sensors/sensor-dev.c
@@ -1810,7 +1810,7 @@ int sensor_probe(struct i2c_client *client, const struct i2c_device_id *devid)

                        pdata->orientation[6] = 0;
                        pdata->orientation[7] = 0;
-                       pdata->orientation[8] = -1;
+                       pdata->orientation[8] = 1;^M
                        break;
```

使用 vim 编辑器时也会显示这种^M 符号.
所以使用 UE 修改代码之后，记得转换成 unix 格式.

提交代码之前，尽量检查一下. 特别是原厂的代码中也可能存在这种问题，修改过程中碰到了一并转换过来.

3. 修改的识别:

增加修改的识别标识，用于告诉其它人，这个地方是谁,为什么修改，修改的开始和结束位置.
这个的目的方便其它的针对公版代码对比查问题时，方便区分哪些是我们自己做的修改，哪些是公版更新的.

对比合并时方便有针对性的处理.

建议的识别:
单行:
//steven: commens for change.

多行:
// steven: commens for changes.
{codes added}
// steven: end

这两条都适用 C/C++, JAVA 代码. 同样也适用于脚本的修改.

其它特殊修改说明:
1. 如果修改的地方是其它人已经修改过的地方, 可以不另外增加识别. 直接修改, 在 git 提交信息中说明即可.
2. 自行修改代码, 识别必需加. 防止别人合并代码时, 把你的修改合并掉了.
3. 合并原厂提供的 patch, 如果是单个问题的 patch, 修改的文件/内容不多, 必需加.
   如果是批量的 patch 合并, 修改的文件很多, 内容也很多. 可以不加. 但在提交的 git 信息中要说明清楚.