驱动程序启动过程

1. 概述

Nuttx 驱动程序主要包括两种：1、字符驱动，2、块驱动。他们都是建立在 Nuttx 伪文件系统的基础上的。对文件操作句柄的操作实现对设备的读写及相关控制操作。同时，每个字符和块驱动需要调用 register_driver()函数进行注册，并传输在伪文件系统中的位置和初始化文件操作句柄。在注册以后，用户可以采用标准的驱动操作(open(),close(),read(),write(), 等)对设备进行操作。

2. 驱动程序的注册流程

在本程序中，驱动主要采用字符驱动机制。register_driver() 函数的实体在 Nuttx 操作系统源文件夹下的 fs/fs_registerdriver.c 中。函数功能，在文件系统中注册一个文件节点。函数声明：

register_driver(FAR const char *path, FAR const struct file_operations *fops,
mode_t mode, FAR void *priv)。

其中：PATH 表示需要创建的文件地址。Fops 表示文件操作句柄结构，里面包含对设备的操作函数接口。Mode 表示 inmode priviledges。Priv 表示与节点相关的私有用户数据。

返回：0---表示成功 ，负值表示注册失败，并对应与错误信息。EINVAL 表示地址操作失败。EEXIST 表示已经存在节点。 ENOMEM 表示分配内存失败。

在 PX4 的程序中设备的注册是在初始化函数中实现的。通过驱动程序类的结构，初始化也是这样层层深入的。

对于特定设备驱动的注册是在对应设备驱动函数的 Start()函数中进行。以 H5883 驱动为例进行分析，具体过程如下：

1、在 shell 文件（ ROMF/init.d/rcs ） 中， 通过执行 sh /etc/init.d/rc.sensors 调用 shell 文件 rc.sensors.在该 shell 文件中调用形如：lsm303d start；的形式启动相应设备，跳转到上述 start（）函数中。

2、相应的设备对象---g_dev 会在相关设备的.cpp 的初始进行定义，通过该对象可以调用相应的设备操作。

3、Start()函数中采用 g_dev->init()的形式调用相应设备的初始化函数。

4、在对应的 init()函数中进行初始 I2C，并在 I2C 初始化中初始化 CDev 初始化，在此初始化中执行 Dvice 初始化，并在其中执行中断注册函数 register_interrupt(_irq, this)，在该中断函数中首先检查是否有注册中断的位置，然后，把本设备的中断号和中断函数入口赋给中断调度表。其中，中断号只在直接以 CDev 为父类的子类里面定义。

5、为 report 数据分配内存。

6、注册该类驱动 register_class_devname(MAG_DEVICE_PATH); MAG_DEVICE_PATH 为该设备在为文件系统中的地址。其实体在 CDev.cpp 中。其实质是内嵌一个 register_driver()函数，实现驱动的注册。

7、注册完成后，将定义的数据存储结构体按字节赋值为 0---memset(&zero_report, 0, sizeof(zero_report)) .然后创建一个“topic”节点，并且“publish”初始值。如果成功则表示对象创建成功，否则创建失败。

3. 总体执行流程如下所示：

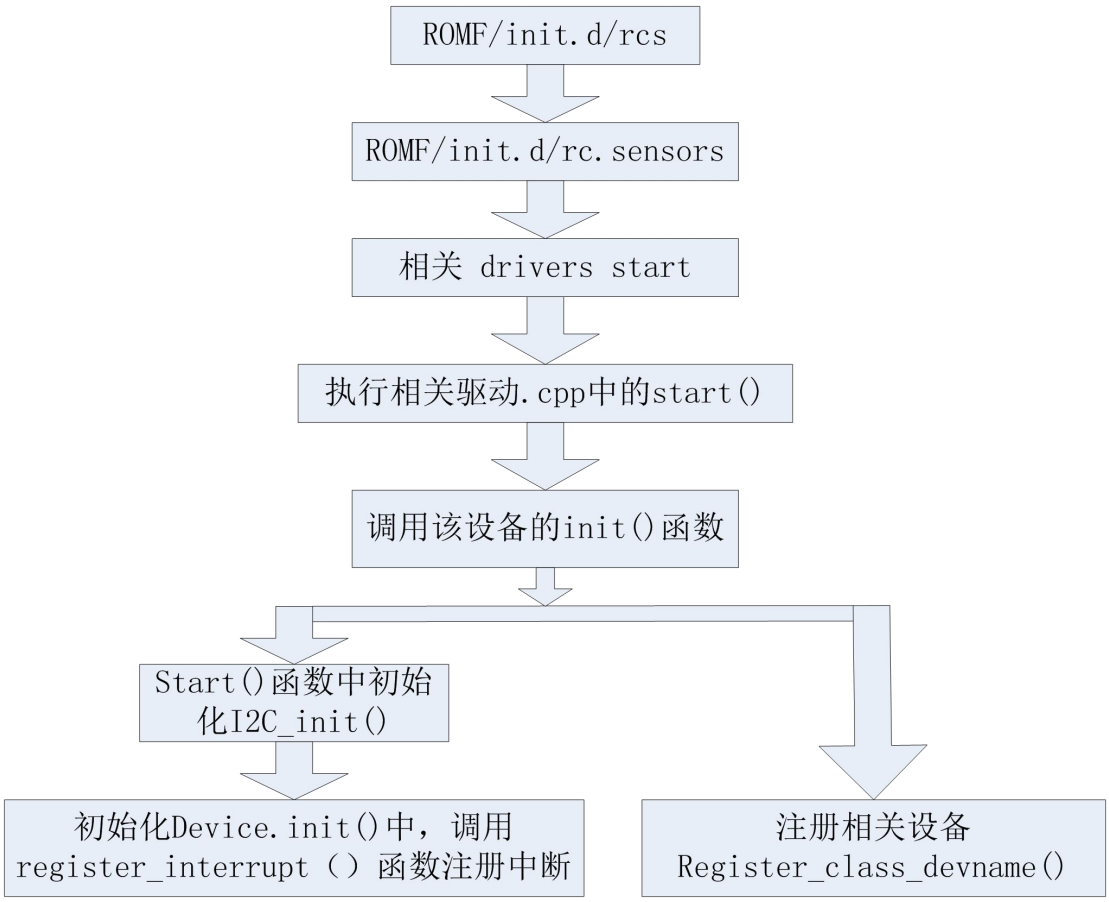

图 1、执行流程图