在Linux系统上，写一个DMA驱动需要完成一下5个步骤：

1.申请DMA通道

```
struct dma_chan *dma_request_channel(dma_cap_mask_t mask, dma_filter_fn filter_fn,
 void *filter_param);
```

在申请DMA通道之前至少需要确定外设ID（filter_param）和通道类型（mask），外设ID不用说根据数据手册或者pl330可以获得，通道类型一般设置为DMA_SLAVE或DMA_CYCLIC，如果外设ID为空，则dma_request_channel将返回满足mask的第一个DMA通道，当外设ID有指向时，dma_request_channel将在满足mask的空闲通道中寻找，知道找到和外设ID相符的通道。

在3.8的内核中，通过三星提供的以下API完成这一操作：sdd->dma->ch = sdd->ops->request(dma->dmach, &req);至少完成这一步之后，我们获取到一个DMA通道sdd->rx_data.ch，使用sdd->rx_data.ch就将我们的驱动程序和该通道关联在一起。

2.设置DMA通道传输参数

```
int dmaengine_slave_config(struct dma_chan *chan, struct dma_slave_config *config)
```

设置DMA通道方向，通道设备端的物理地址（如果使用DMA向SPI收发数据，则为SPI数据寄存器的物理地址），通道字节宽度等信息。

在3.8内核中，通过三星提供的以下API完成这一操作：sdd->ops->config(dma->ch, &config);这一步为申请到的管道配置方向，管道宽度，以及外设的物理地址，根据方向的不同，这里的外设物理地址有可能是源地址（DMA_DEV_TO_MEM），也有可能是目的地址（DMA_MEM_TO_DEV）

3.获取desc添加回调函数

在驱动函数中，将发送数据个数，通道方向，数据缓存的总线地址等参数赋值给scatterlist结构体，调用dmaengine_prep_slave_sg或dmaengine_prep_dma_cyclic获取desc，再将回调函数指针传给desc->callback，在DMA的API中，回调函数总是以DMA任务上下文的方式调用的，而与中断

上下文无关。

在3.8内核中，通过三星提供的以下API完成这一操作：sdd->ops->prepare(dma->ch, &sdd->rx_info);

4.递交配置好的通道

调用dmaengine_submit(（struct dma_async_tx_descriptor *)desc)，将desc提交到DMA驱动等待队列，通常第3和第4步都是在DMA驱动的prepare函数中实现的。

在3.8内核中，通过三星提供的以下API完成这一操作：sdd->ops->prepare(dma->ch, &sdd->rx_info);

5.开启通道，等待回调函数

```
void dma_async_issue_pending(struct dma_chan *chan);
```

调用dma_async_issue_pending激活挂起的等来队列，如果此时通道空闲则开始传输队列中的数据，传输结束后调用回调函数。

在3.8内核中，通过三星提供的以下API完成这一操作：sdd->ops->trigger(dma->ch);