1. 设置cdma，主要是为cdma control register

```
int stmotor_open(uint32_t motor_num)
{
        if (motor_num >= STEPPER_NUM_MOTORS)
                return -1;
        smot_step_set_blk_addr_irq(motor_num, stmotors[motor_num].stmotor_cdma_num);
        stmotor_active |= 1 << motor_num;
        return 0;

}
```

```
static void smot_step_set_blk_addr_irq(stmotor_id_t motor_id, uint32_t cdma_num)
{
        stmotors[motor_id].tx_dma_slave.vtype = MV61_VDMA_OWNED;
        stmotors[motor_id].tx_dma_slave.wr_delay = 0;
        stmotors[motor_id].tx_dma_slave.destendian = MV61_DMA_LITTLE_ENDIAN;
        stmotors[motor_id].tx_dma_slave.srcendian = MV61_DMA_LITTLE_ENDIAN;
        stmotors[motor_id].tx_dma_slave.flowctrl = MV61_DMA_MEMORY_TO_PERIPHERAL;
        stmotors[motor_id].tx_dma_slave.dest_pid = cdma_num;
        stmotors[motor_id].tx_dma_slave.dest_addr_inc = true;
        stmotors[motor_id].tx_dma_slave.src_addr_inc = true;
        stmotors[motor_id].tx_dma_slave.dest_width = MV61_DMA_XFER_WIDTH_32BIT;
        stmotors[motor_id].tx_dma_slave.src_width = MV61_DMA_XFER_WIDTH_32BIT;
        stmotors[motor_id].tx_dma_slave.data_unit_size = MV61_DMA_UNIT_SIZE_32BIT;
        stmotors[motor_id].tx_dma_slave.dest_burst = MV61_DMA_BURST1;
        stmotors[motor_id].tx_dma_slave.src_burst = MV61_DMA_BURST1;
        stmotors[motor_id].tx_dma_slave.dest_reg = (dma_addr_t)&(stmotors[motor_id].phy_stmotor_regs->PWM_T);
        stmotors[motor_id].tx_dma_slave.timebase = MV61_TIMEBASE_1MS;
        stmotors[motor_id].tx_dma_slave.timer = 0;
        stmotors[motor_id].tx_dma_slave.wrap = 24;

}
```

2. request cdma channel

```
stmotors[motor_id].stmotor_dma_chan = dma_request_channel(mask, filter, &(stmotors[motor_id].tx_dma_slave));
```

由于cdma driver实现的原因，必须使用这种形式，即filter callback和struct mv61_dma_slave(已经被初始化完毕)

3. 把要传输的data放置到sg中，并且进行virtual to physical mapping

```
        memcpy(list_entry->data_buffer, buf, count);
        sg_alloc_table(list_entry->sg_table, 1, GFP_KERNEL);
        sgl = list_entry->sg_table->sgl;

        sg_set_buf(sgl, list_entry->data_buffer, count);
        sgl = sg_next(sgl);

        len = dma_map_sg(stmotors[motor_id].stmotor_dma_chan->device->dev,
                         list_entry->sg_table->sgl,
                         1,
                         DMA_TO_DEVICE);
```

4. 准备dma传输，获得transfer descriptor

```
        tx_desc = stmotors[motor_id].stmotor_dma_chan->device->device_prep_slave_sg(
                stmotors[motor_id].stmotor_dma_chan,
                list_entry->sg_table->sgl,
                len,
                DMA_TO_DEVICE, DMA_PREP_INTERRUPT | DMA_CTRL_ACK, &len); /* &len is for context, which is not used*/
```

其实可以用dmaengine_prep_slave_single()，而不是象这样直接访问函数指针。

5. 设置cdma传输完毕后的callback

```
        tx_desc->callback = smot_step_motor_cdma_callback;
        tx_desc->callback_param = &stmotors[motor_id].stmotor_cdma_num;
```

6. 提交该transfer descriptor

```
cookie = dmaengine_submit(tx_desc);
```

7. 等待callback

```
stmotors[motor_id].stmotor_dma_chan->device->device_issue_pending(stmotors[motor_id].stmotor_dma_chan);
```

invoke dma_async_issue_pending() API可能更好。