控制发送相关：

数据处理及组装

```
def float_to_uint(*dat):
    # 提取数据
    x = dat[0]          # 浮点值
    x_max = dat[1]  # 浮点值范围的最大值
    x_min = dat[2]  # 浮点值范围的最小值
    bits = dat[3]        # 位宽

    # 计算范围和偏移量
    span = x_max - x_min
    offset = x_min

    # 线性映射公式
    # return int((x - offset) * ((1 << bits) - 1) / span)
    return int((x - offset) * ((1 << bits)) / span)
```

val_sb_p：位置值；sb_p_max：sb_p 得最大值，self.sb_p_min：sb_p 最小值； bit_value：默认 16

val_sb_v：速度值

val_sb_kp：KP 值

val_sb_kd：KD 值

val_sb_t：力矩

```
operate_params = [
    float_to_uint(val_sb_p, self.sb_p_max.imum(), self.sb_p_min, bit_value),  # 角度 P
    float_to_uint(val_sb_v, self.sb_v_max, self.sb_v_min, bit_value),  # 角速度
    float_to_uint(val_sb_kp, self.sb_kp_max, self.sb_kp_min, bit_value),  # 刚度 Kp
    float_to_uint(val_sb_kd, self.sb_kd_max, self.sb_kd_min, bit_value),  # 阻尼 Kd
    float_to_uint(val_sb_t, self.sb_t_max, self.sb_t_min, bit_value),  # 扭矩 T
]
operate_params #对应的下面发送数据块
def to_send_operate_params(self, did=None, operate_params =None):
    dl = [0x0000,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x7FFF,
          0x0000]

    dl[31] = 0000

    motor_index = (did - 1) * 5 + 1
```

```
    for i, value in enumerate(operate_params):
        dl[motor_index + i] = value

    dl_bytes = []
    for i in range(len(dl)):
        dl_bytes.append(dl[i] & 0xFF)
        dl_bytes.append((dl[i] >> 8) & 0xFF)

    return dl_bytes #发送的数据帧
```

帧 ID： 0

帧长度：64

帧数据：dl_bytes

接收数据处理：

```
# 获取设备返回 位置、速度等值如何解析

def uint_to_float(x_int, *dat):
    x_max = dat[0]  # 浮点值范围的最大值
    x_min = dat[1]  # 浮点值范围的最小值
    bits = dat[2]       # 位宽

    span = x_max - x_min
    offset = x_min
    return float((x_int)*span/((float)((1<<bits))) + offset)
```

展示的可视值：uint_to_float(val, val_max, val_min, bit_value)

val: 速度、位置、力矩等值

val_max: 最大值

val_min: 最小值

bit_value：默认 16

如何获取设备返回的值

```
def data_frame2value(pData):
    position = pData[2] + (pData[3] << 8) # 位置
    speed = pData[4] + (pData[5] << 8)    # 速度
    torque = pData[6] + (pData[7] << 8)   # 力矩
    motorTemperature = pData[8] - 40      # 温度
    motorFaultRecv = pData[9]             # 状态
    motorVoltageRecv = pData[10]          # 电压状态

    motorEnableRecv = motorFaultRecv & 0x01  # '使能' if motorEnableRecv else '失能'
    motorOvervoltageRecv = (motorFaultRecv >> 1) & 0x01 # 过压故障 if motorOvervoltageRecv else '正常'
    motorOvercurrentRecv = (motorFaultRecv >> 2) & 0x01  #过流故障 if motorOvercurrentRecv else '正常'
    motorOverTemperatureRecv = (motorFaultRecv >> 3) & 0x01 # 过温故障 if motorOverTemperatureRecv else '正常'
```

```
    motorOverSpeedRecv = (motorFaultRecv >> 4) & 0x01  # 过速故障 if motorOverSpeedRecv else '正常'
    motorDualEncoderRecv = (motorFaultRecv >> 5) & 0x01 # 双编故障 if motorDualEncoderRecv else '正常'
    motorPreDriveRecv = (motorFaultRecv >> 6) & 0x01 # 预驱故障  if motorPreDriveRecv else '正常'

if can_id == motion_no: # 如果设备返回的 ID 和需要监测的设备 ID 相同
    if frame.len == 12: # 返回的数据帧是否是 12
        data_frame2value(pData) # pData 获取帧数据，并解析
```

易动智能