抱闸问题（2）

好吧，昨天调试了一上午的X1、X2，居然忘了还有旋转电机不能转的情况了😐。最后原因是陈亮发现抱闸线接错了。

今天主要任务就是理解：

- X1、X2点动的时候，出现其他刹车灯亮的问题
  - 是否出现了其他电机使能的问题 - 检查点动控制寄存器
- X1、X2位置移动的时候，电机运动的逻辑

其实就是理解p100s型号的电机，其对应的点动和位置移动逻辑。

p100s点动逻辑

前进按钮做了什么

点动逻辑要看函数 `actionJog()` 。以大屏为例，前进移动的按钮设计如下

```
connect(m_action_upRightFront_bt, &QPushButton::pressed, this,
&MotorCtrl::on_actionUpRightFrontBt_pressed);
connect(m_action_upRightFront_bt, &QPushButton::released, this,
&MotorCtrl::on_actionUpRightFrontBt_released);
```

前进按钮按下去的时候，会触发槽函数 `on_actionUpRightFrontBt_pressed()` 执行。我们看下对应函数逻辑

```
void MotorCtrl::on_actionUpRightFrontBt_pressed()
{
    if (!checkMotorConfigurationValidity()) {
         QLOG_ERROR()<<QString("MotorCtrl: 前进按钮长按失效");
         return;
    }
    // [点动模式] 移动
    if(m_runMode_chb->isChecked()){
        if(!isRunning()){
            m_allDone =false;
            actionJog(1);
        }
    }
}
```

其逻辑为：如果当前为点动模式且并未存在电机运行中，则执行 `actionJog(1)` 控制电机移动。

复杂的点动逻辑

查看点动逻辑 `actionJog(int)` （已经重构）

```
void MotorCtrl::actionJog(int dir)
{
    if (!canStartJog()) return;

    // 获取电机ID
    int motorID = getMotorID();
    m_jog_done = false;

    disableAllbt(dir);
    logJogStart(motorID, dir);

    // 预启动电机
    if (!applyJogConfig(motorID, dir)) return;
    if (!isPositionWithinLimit(motorID, dir)) return;

    // 点动控制伺服使能
    enableJogServo(motorID, dir);

    m_jog_dir = dir;
    m_checkWoker = new QThread;
    connect(m_checkWoker, &QThread::started, this,
&MotorCtrl::checkJog);
    connect(m_checkWoker, &QThread::finished, this,
&MotorCtrl::on_threadFinished);
    m_checkWoker->start();
}
```

我们将目光聚焦在 `enableJogServo()` 函数，其中存在刹车的逻辑。

刹车逻辑

X1、X2点动移动的时候，刹车部分想要实现的逻辑：

- 如果是X1、X2电机点动，则需要将其他电机的DO刹车端口屏蔽掉（将P3-21设置为0）
- 执行点动操作
- 恢复X1、X2电机外DO刹车端口正常输出

广播修改DO端口

现在想实现的是，通过广播修改P3-21（DO2）寄存器的值。思路是：

- 实现函数 `clearBrakeDOConfig()` ：将P3-21寄存器改为0
- 实现函数 `clearBrakeDOConfig()` ：将P3-21寄存器改为8（电磁制动器）
- （首先确保P3-21的值不为0）尝试在X1、X2电机组界面中调用 `clearBrakeDOConfig()` 发广播，此时查询伺服的 `P3-21` 寄存器是否为0。
  - 如果是0，则证明广播可行，可以进行下一步
  - 如果不是0，则证明广播修改P3-21不可行，需要循环处理。

当前Z2电机的P3-21值为8，代表刹车功能。

P3-31寄存器

P3-31 寄存器是用于设置**虚拟输入端子状态值**的参数，其作用是：

> **P3-31：虚拟输入端子状态值（00000000~11111111）**
> 通过设置此寄存器的每个位（共 8 位），可模拟输入端子的电平状态，实现“虚拟输入”，以便在无实际硬件接线的情况下进行测试或逻辑控制。

配合参数 P3-30 使用

P3-31 的生效方式依赖于参数 `P3-30` 的设置：

- `P3-30 = 0`：使用 DI1~DI4 物理输入，**P3-31 不生效**；
- `P3-30 = 1`：只使用 P3-31 虚拟输入，**可用 8 个虚拟输入**（DI1~DI8）；
- `P3-30 = 2`：混合使用物理输入（DI1~~DI4）和虚拟输入（DI5~~DI8）。

是否可以用于“点动控制”？

是的，在**速度模式且参数 `PA22=5`** 时，点动控制可以通过如下信号实现：

- `JOGP`（正向寸动）
- `JOGN`（反向寸动）

这些信号可以通过实际接线输入，也可以通过虚拟输入实现。例如：

- 设置 `P3-39` = 22（JOGP）
- 设置 `P3-40` = 23（JOGN）

- 然后将 `P3-30 = 1或2`（启用虚拟输入模式）
- 再通过设置 `P3-31` 相应位为1，模拟按下按钮，就可以实现点动功能。

`applyJogConfig()` 其主要逻辑为：

```
//  进行点动配置
bool ret = jogCfg(motorID, dir*DbCtrl::m_servoMotor_tb[motorID].dir);
if(!ret){
    ...
    return false;
}
```

继续查看函数 `jogCfg()`

```
if(!m_motorRegs->writeReg16(motorID, MotorRegs::P3_31,
MotorRegs::JogMode)){
    QLOG_ERROR()<<QString("电机%1写入P3_31=%2失
败").arg(DbCtrl::s_motorNameList.at(motorID)).arg(MotorRegs::JogMode)
;
    return false;
}
quint16 act = MotorRegs::JogRunInc;
if(dir<0){
    act = MotorRegs::JogRunDec;
}
if(!m_motorRegs->writeReg16(motorID, MotorRegs::P3_31, act)){
    QLOG_ERROR()<<QString("电机%1写入P3_31=%2失
败").arg(DbCtrl::s_motorNameList.at(motorID)).arg(act);
    return false;
}
```

根据这个推测 `P3_31` 寄存器的功能是

- **第一次写入是“设定模式”**（告诉控制器我们要进入 Jog 控制模式 `JogMode`）；
- **第二次写入是“执行具体动作”**（触发 Jog 正向点动 `JogRunInc` 或 Jog 负向点动 `JogRunDec`）；

现在让我很疑惑的是，p100s的寄存器 `P3_31` 到底是什么功能？此时需要查看手册理解。

根据手册，P3-31寄存器要想使用，需要P3-30寄存器配置为 `1` 或 `2`：

- `P3-30 = 1`：只使用 P3-31 虚拟输入，**可用 8 个虚拟输入**（DI1~DI8）；
- `P3-30 = 2`：混合使用物理输入（DI1~~DI4）和虚拟输入（DI5~~DI8）。

那么点动之前是否配置了P3-30为 `1` 或 `2`？

在 `BaseConfig` 类的初始化电机寄存器参数的时候，初始化了此寄存器值为 `2` 。且其他地方并无更改此寄存器的值。

```
m_p100scfgs.append(qMakePair(MotorRegs::P3_30, 2));
```

下面逐步分析，给 `JogMode` 和正向点动 `JogRunInc` 或 Jog 负向点动 `JogRunDec` 的设置问题。

```
    enum JogAct{
        JogMode = 2,
        JogRunInc =6,
        JogRunDec =10
    };
```

JogMode

```
m_motorRegs->writeReg16(motorID, MotorRegs::P3_31,
MotorRegs::JogMode)
```

分析此代码的含义。在分析之前，要知道P3-31代表的寄存器都是哪些

> P3-31 代表的输入 IO 数为 8 个,对应参数 P3-38~P3-45

当P3-31设置为值 `2` 的时候。代表的是二进制数 `0000 0010` ，代表的寄存器为P3-39的值为 `1` 。P3-39代表的是

```
虚拟 IO 输入 DI2 功能
```

即寄存器P3-39所代表的DI2使能，而P3-39设置的值为 `16` ，对应的DI功能选项为

定义值	符号	功能	功能解析
16	CMODE	复合模式控制	当 PA-4 设置为 3，4，5 时，处于混合控制模式，可通过此输入端子可切换控制模式: (1)PA-4 为 3 时,CMODE OFF，为位置模式；CMODE ON，则为速度模式; (2)PA-4 为 4 时,CMODE OFF，为位置模式；CMODE ON，则为转矩模式; (3)PA-4 为 5 时,CMODE OFF，为速度模式；CMODE ON，则为转矩模式。

此时设置的PA-4值为 `3` ，是位置控制模式。则DI2输出为高电平的时候，则为速度模式。

那么确实将P3-31设置为2的时候，对应DI2端口输出为高电平，此操作会让电机处于速度模式（即点动控制）

JogRunInc

JogRunInc = 6，当P3-31设置为值 `6` 的时候。代表的是二进制数 `0000 0110` 。其含义为:

- `P3-39 = 1` -> DI2输出高电平 ->速度模式
- `P3-40 = 1` -> DI3输出高电平

P3-40代表的是

```
1 | 虚拟 IO 输入 DI3 功能
```

P3-40设置的DI3功能值为 `22`

```
1 | m_p100scfgs.append(qMakePair(MotorRegs::P3_40, 22));
```

对应的DI功能选项为

定义值	符号	功能	功能解析
22	JOGP	正向寸动	速度模式下，PA22=5 时，此信号接通，电机正方向寸动，速度受 PA21 设置。

PA-22寄存器初始化为 `5`

```
m_p100scfgs.append(qMakePair(MotorRegs::PA22, 5));
```

值 `5` 代表的意思为：IO端子控制点动操作。

所以此时如果伺服使能的话，就可以执行运动了。但是此时伺服使能寄存器并未置 `1` 高电平，所以电机不转。