目录

[附录：C语言编写规范 1](#_Toc11706)

[摘要 2](#_Toc7693)

[第一章 NOTE 10](#_Toc18062)

[1.1 碎笔录 10](#_Toc27977)

[1.2三环框图 20](#_Toc12880)

[1.3 驱动器调试 22](#_Toc10693)

[第二章 ST7工程笔记 24](#_Toc22836)

[2.1 ST7旧上位机通讯协议 42](#_Toc3377)

[2.2 ST8通讯协议 53](#_Toc24401)

[2.3 高创驱动器USB协议 61](#_Toc16347)

[2.4 辨识功能 62](#_Toc30184)

[第三章 Kinco上位机 65](#_Toc30578)

[第四章 ST6&ST7上位机 67](#_Toc18357)

[4.1 note 67](#_Toc22645)

[4.2 更新计划 68](#_Toc24843)

[第五章 C#语言 69](#_Toc6266)

[5.1 C#语言 70](#_Toc14727)

[5.2 VS组件 77](#_Toc3970)

[5.2.1 C#语言部分 77](#_Toc29850)

[5.2.2 VS控件学习 83](#_Toc6372)

[5.3 WPF 86](#_Toc22342)

[5.3.1 工程note 87](#_Toc26774)

[5.3.2 请教问题 88](#_Toc9393)

[5.3.3 问题解决记录 89](#_Toc24555)

[第六章 驱动器测试工装 91](#_Toc24714)

[第七章 伺服电机、步进电机、FOC 94](#_Toc1269)

[附录1：调试问题汇总 95](#_Toc11598)

[附录2：life mark 101](#_Toc6693)

# 附录：C语言编写规范

1. 能用switch的，永远不要用if else if else if else(超过2个if优先选择switch)；
2. 能使用指针的，永远不要直接传值；
3. 所用的函数默认都写成static类型；
4. 所有的变量，需要写成全局变量的，前面要用g\_来标识；
5. 尽量把同类型，有关联的，变量，组织起来放在一个结构体中；

# 摘要

任务：

本周任务：

1. 完善上位机需求书，制作三环简略框图----√
2. 看松下手册，重点是各个参数定义-----√
3. 旧ST7上位机采取波形时，会出现卡顿现象？----根据测试推测，是旧上位机数据处理的问题，驱动器应该是正常。
4. 数据存储62.5us。----目前驱动器支持
5. 性能评估：将示波器函数放入主循环中，在打开示波器状态下，会影响整个机械系统性能多少？
6. 在数据接收和处理那一块 考虑-----数据接收和处理方面还好
7. 针对DMA，明白其原理及占用时间；----√
8. 看松下P220页的参数说明；-----√
9. 电机参数表的对接和规划，调试参数界面的参数规划
10. 明天尝试开发板DMA功能，熟悉相关函数，study！-----√
11. 我比较在意两点：数据接收和处理那一块的DMA；-----√
12. 示波器再深入理解
13. 应用功能的IO，灵活处理
14. 将新协议进行完善，和modbus保持一致-----√

Assignment：

1. 界面规划，新+旧-----×
2. 破解高创驱动器上位机的通讯协议
3. 旧版本上位机字典文档整理------√
4. 新版本上位机字典------√
5. 示波器采集上限，提高-----×
6. BMS的上位机格式，及压缩方式
7. RAM的内存空间计算

示波器的结构体上限修改会报该错误：

the destination for compressed initializer batch "P2-1" is placed at an address that is dependent on the size of the batch

SAR1400

原因：RAM内存超了，内存空间不够

滤波作用，误码率

1. 滤波作用
2. 错误字典
3. **示波器功能-------√**
4. **多段运行：代码+字典-----√**
5. **思考上位机通讯协议的加密和压缩**
6. **错误码的查找和保存**
7. **C#语言学习。**
8. **收集万达生产那方面的需求。----版本号，BootLoader**
9. **找到 状态标志。**

**提示：**

**上位机那边，把高创的can，ethercat的某些文件添加了。**

**ST8过于依赖软件的问题。**

**Z相热量累积误差错误**

**2022.7.4**

**1.学习C#和VS组件---50%--今天下午看完**

**2.理解固件升级的协议代码，并制作说明书。**

**3.教吴当学代码；**

**那个测试人员的工作内容：**

1. **测试新固件**
2. **会用新工具—上位机，测脉冲**
3. **使用手册---ST7上位机，驱动器**

**今天下死命令：组件工程，要学完；**

**然后看字典部分，对示波器的参数进行确认；**

组件工程完成了，剩下的是熟悉组件功能，资源管理器，方法，关键字。

要能够有底气的在今天日报中，说自己有一定程度的了解

**提建议，技术支持的 现场问题汇总锦囊（可每周安排一次会议讨论，出一版后召开会议发布—销售-技术支持-生产）在推销产品，解决客户现场问题时更高效。**

且该文档可同步应用到上位机中

没有文档也不利用传承，教授其他人员

新员工入职计划表

关于新员工培训方面，石工和尹工部门都有自己的新员工培训计划表

顶多看完字符串这章节，剩下的看代码

结束符

帮二楼电机生产解决问题

我遇到了瓶颈，学习新的语言，学不进去，感觉能接受但是却让我有压力，是学习习惯导致的，还是说状态问题，自己考虑的东西有些多？也都不好说。

慢慢接受，循序渐进，找到状态，具备良好的学习和思考的习惯。

-------------------------------------------------------------------

1. 上位机示波器功能，00发送，11停止，数据可变
2. 老化箱老化驱动器
3. 整理电机ID信息
4. 固件升级

[https://www.iar.com/device-search/#!?tab=devices](https://www.iar.com/device-search/" \l "!?tab=devices)

2022.8.1

重要任务：

1. 快速把新手教程过一遍----√
2. 研究示波器，计算等相关问题-----√
3. 看新上位机代码
4. 修改示波器底层代码，完成连续采集波形

前瞻任务：

1. 示波器通道能建立新参数
2. 通讯协议新增 烧写固件的功能码，能跳转到BootLoader区
3. 通讯协议新增 驱动器复位功能码。
4. ST8测试工装。

向广谱提建议，给技术支持培训，如何使用ST7上位机，如何借助工具排查问题。

今天任务：看手册引脚图，测试上位机，寻找跳点问题，发USB驱动，看源代码文档。文档今天必须看一遍！

2022.8.8

1.完善P22组参数的底层

2.和刘凤对接

3.看扭力扳手文件

4.把上位机源代码过一遍

80 55 DB 47 80 56 DB 47

**00 00 00 00 44 71 DB 47**

80 55 DB 47 80 56 DB 47

80 55 DB 47 80 56 DB 47

验证P22组的参数

学习C#语言

2022.8.15

1. 错误代码调通，即P22组参数调通
2. 看扭力扳手文件
3. 调试上位机

辨识界面，快速上手，尽快做出来：

1. 捋清SCUT上位机算法
2. 熟悉示波器
3. Dll等各种库的移植

Ps.一个很好的成长学习的机会。现在真他妈给了一个大机遇，有12年老程序员教上位机代码，有华理研究生教算法，这不通吃？！

2022.8.25

1. 上位机bug之，示波器sleep时间修改
2. 上位机解决问题的过程录屏视频
3. ST6上位机的使用说明书
4. 辨识模式的上位机与下位机交互流程说明文档
5. 驱动器工装
6. 上位机新功能开发：实时波形，示波器通道倍数
7. ST8说明书+modbus协议指导
8. ST7说明书+modbus协议指导
9. 验证USB转modbus串口

2022.8.29

看完最后一个录屏----√

参数加载进入驱动器后加个提示----√

和蔡泽凯沟通-----√

查找到保存参数的个数在哪个文件

无logo上位机打包------√

Modbus波特率测试-9600------√

辨识功能示波器通道改造----√

数字IO功能的状态显示----

预备测试文档，关于modbus和新上位机

**本阶段任务罗列：**

测试0530 ST7版本，及modbus

解决使能过程中，辨识后会掉线

解决下文罗列的上位机bug

新增驱动器底层的功能

看辨识功能上下位机的算法

**上位机遗留问题：**

**数字IO，延时时间实时显示-----√**

**保存参数时的各种提示----√**

**使能状态显示；-----√**

**ST6和ST7驱动器上位机版本区别----√**

**示波器放大倍数-----√**

**上位机显示驱动器的各个版本号（FPGA,ARM）-----√**

**在安装包中添加辨识功能所需要的dll库----√**

**有些错误无法用上位机清除**

**底栏的模式不准确**

**使用示波器的光标后，会出现卡顿现象**

**保存参数的速度过慢，可采用多个数据读取的数据帧**

**右上角的**

本周任务：

1. 完善驱动器说明书
2. 将P18.19设置为0 的操作，去掉！
3. 测量数据，有时会漏掉一个，检查
4. 改之前没改完的bug
5. 考虑测试工装
6. 重点问题：示波器和掉线处理机制。
7. 考虑单位切换问题
8. 错误清除后，还会显示1，并且错误历史记录里，没有数据
9. 状态栏的，报错变红指示灯；

本月任务：

1.改上位机bug，故障处理

2.中科兴复测，国产407测试落实

3.ST6，ST7，ST8手册，学习ST8和高创

4.学习总线老化测试和使用。

5.测试工装搞定

6.modbus测试及培训

7.上位机更新电机ID型号

8.上位机英文版

今日任务：

* ST6/ST7说明书----搞定√
* modbus测试及培训----搞定√
* 上位机更新电机ID型号---搞定√
* Ethercat-欧姆龙和西创的

运行模式，脉冲控制模式，电流环标定模式

任务汇总：

1. ST8驱动器的测试工装+ABZ回零测试（中）----√
2. 上位机改名称，生成的工程（简）---√
3. 驱动器说明书检查细节，新增ST5驱动器（简）
4. 规划上位机新增ST5驱动器，要与驱动器底层配合；（简）
5. 制作12月3号讲解驱动器知识的PPT（难）
6. 惯量识别方案的学习（难）
7. 配合陷波功能的调研和方案讨论（简）---√
8. 辨识功能移植；

Monitor\_tools

今天搞完回零+垂直电机回零

周三搞完工装

惯量学习+PPT---周四

上位机+说明书---周五

1. 驱动器说明书检查细节，新增ST5驱动器；  
   2. 规划上位机新增ST5驱动器，要与驱动器底层配合；  
   3. 制作12月3号讲解驱动器知识的PPT；  
   4. 惯量识别方案的学习。将上位机的辨识功能中的惯量辨识做到下位机里，蔡泽凯协助。  
   5. 修改上位机bug：

上位机bug及功能汇总：

模式切换保持一致

离线模式的参数读取

全部参数界面，某些参数无法正常显示（文件导入后）

全部参数界面，必要的参数详细描述；

示波器采样点时间的单位显示

示波器的数据查看，有时会有1-2个通道未显示

示波器选择放大倍数后，参数会无法显示

第一次导入参数会有bug，数据错位

功能新增：

连续波形显示

可手动添加电机型号

可手动添加示波器参数

单位转换

双语言选择

ST6波形图坐标倍数问题，参考ST8驱动器

读写参数速度加快

任务规划：

1. 陷波参数设置页面---√
2. 陷波参数下发功能实现---√
3. 陷波参数计算 ---预计明天上午搞定
4. 上位机波特图页面
5. 刚性等级表
6. 安排陈怡娴测试
7. 操作说明书中加入电流环标定
8. 看两个代码：parser和demo
9. 修改旋转加速度
10. 制作刚性等级表

伯德图存在问题：

1. 坐标轴显示
2. 文字说明
3. 背景网格，透明度降低
4. 放大（鼠标和画矩形框）
5. 对称坐标

参考图例：

Stacked column side by side

着手解决两个问题：

1. 下面轴的图形要显示出来
2. 可以支持放大

ABZ和ST6，ST7工装

2023.1.4号 任务

1. 修改未实例化对象的bug----√
2. 全部参数界面的离线数据查看-----√
3. 全部参数界面的，参数详细描述的填写

惯量辨识的速度环增益----√

默认增益问题-----√

上位机参数详细描述

单位换算处理

工装

额外工具学习：matlab，git

1. 看论文
2. 整理问题
3. 看上位机界面设计：控制器界面，回零界面，fft，示波器放大倍数问题；

要解决的上位机bug：

1. 电机 ID号，恢复出厂设置后，重新连接会保存。建议：触发重新读取一遍数据；
2. 全部参数的详细描述；
3. FFT傅里叶变换；
4. 单位显示；
5. 连接界面，显示驱动器的版本号
6. 控制器界面设计---待审核；
7. 控制模式与右边小窗口显示保持一致
8. 示波器部分：放大倍数，光标显示的数据；
9. 在一键辨识界面 点动速度窗口，在左边给出推荐点动速度。

增量式位置反馈更新2

闷头做不如有计划的做

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 任务 | 起始时间 | 截止时间 | 持续时间 | 完成度 |
| ST7bug修复，惯量辨识 | 1.31 | 2.1 | 2天 | √ |
| ST6通讯测试，扰动补偿，惯量辨识，陷波功能测试 | 2.22 | 3.10 | 13天 |  |
| 上位机bug：参数描述，IO状态灯，控制模式切换，FFT分析 | 2.2 | 2.8 | 5天 | 控制模式√  示波器倍数√  参数描述×  IO状态灯×  FFT分析 |
| 示波器问题：倍数，光标显示，参数同步 | 2.9 | 2.10 | 2天 | 示波器倍数√ |
| 工装完善：ST5，ST6，ST7，ABZ | 2.13 | 2.28 | 12天 |  |
| 工装测试 | 3.1 | 3.3 | 3天 |  |

写文档；

看DD电机的惯量辨识

理解单圈和多圈编码器区别

该有的都有，理解框架，理解代码构成。

要有明确的目标，和解决问题的思维导向，不要陷进代码逻辑的漩涡中。提炼出有用的东西

**共同解决的问题：**

1. 实际运行速度与指令速度方向相反----√
2. 在STP-MT-H075B30W08N1V7电机和SAN-SDM160-024I2A20N电机上每次失能再使能后采集的位置波形图都不一样，除了断电重启后的第一次使能外示波器显示的位置和上位机底部状态栏显示的位置当前值也不一样。---原理问题，现象没错也不用修改，位置当前值的实时性需要考虑

**驱动器问题：**

1. 驱动器屏幕界面无P06组参数
2. 当P00.01控制模式为速度模式和P03.00速度指令来源为数字速度时，无论设置目标速度为何值每次使能都以断电前设置的目标速度运行，目标速度只在使能时更改生效；
3. 运行速度和模拟量给定值方向相反，10v模拟量给定控电机运行实际速度为-2650rpm
4. 在速度模式下模拟量给定速度控制、力矩限制时无法力矩限制。

**惯量辨识问题：**

1. 惯量识别后直线电机运行声音较大，随着辨识基准等级的增大而增大，当增大转矩指令滤波时有改善
2. LMZ-04b06030D11-皮带运行启停有速度冲击
3. 速度环增益有较大的偏差
4. 在DDR电机上惯量辨识及增益自整定后以100rpm及以上的速度运行会有异响，转矩指令滤波偏小，且随着辨识基准等级的增大而减小。

1.发送通道数据

2.注册channel

3.实时显示---验证ST8的底栏与示波器

获取最后点位数据集

复制chart到表格

1. 惯量辨识，使能状态下，不能进行动作

明天上午，总结：

1.惯量辨识---√

2.使能显示---√

3.工装规划。

模拟量

内部力矩给定控制

脉冲测试的波形图

**版本区分**

**硬件流控制，流控**

**梳理 402状态字和控制字**

**高低速脉冲切换，需要断电问题。**

**本周任务：**

1. **工装完成**
2. **完善文档，整理文档。包括手册。**
3. **有空看看 端口，连接功能，掉线**
4. **Linux，git**
5. **报错机制要优化（脉冲，IO）**

**说明文档，IO功能部分；**

**学习怡娴测试的内容；**

**任务：**

**1.生产测试工装（ST6 & ST7 & ST8），功能测试**

**2.陷波滤波器**

**3.上位机参数描述，看驱动器掉线部分尝试解决，ID号显示不准确**

**4.FFT分析**

**5.完善说明书，驱动器及相关功能说明（IO信号，平滑，）**

**工装进度，就差ST7的高速脉冲，测试不准确。**

**对比分析ST7和ST8的脉冲数设置。**

**已解决：**

**1.检查线路(还差一个短路测试线路)--驱动器会正常报错**

**2.调试条件下运行，看是否有问题---没问题**

**3.更换工装板---没问题**

**4.把 .h中声明的函数，确认一遍，无效的删除，尤其是485.h。----√**

**待解决：**

**5.错误代码读取功能，ST8待完善，ST7待补充**

**6.思考为什么要多循环一遍，脉冲数才对应上？**

**7.工装，显示报错，驱动器故障显示，异常判断。**

**8.脉冲测试时，ST8的脉冲来和回都很准确，ST7来和回差异很大。但最终往复结果是没有偏差的。**

**9.打短路时，ST7没有报错。有的时候会报Er010。ST8一直报错。**

**10.加入版本号**

**上电，恢复出厂设置，IO**

**自学：**

**Git，ethercat，Linux，http，云端互联网**

**皮带，测试**

**前馈，滤波，EtherCat**

**晚上搞GitHub，gitee不行的话，撤退**

**看泽凯发的论文，把资料上传到Git**

**待解决：**

**1.错误代码读取功能，ST8待完善，ST7待补充**

**2.思考为什么要多循环一遍，脉冲数才对应上？**

**3.工装，显示报错，驱动器故障显示，异常判断。**

**4.脉冲测试时，ST8的脉冲来和回都很准确，ST7来和回差异很大。但最终往复结果是没有偏差的。**

**未来事件：**

1. **上位机显示 平滑，Ethercat模式进入；**
2. **测试工装**
3. **使用手册，相关说明书**
4. **修改通讯协议**

**驱动器后台计算问题----√**

**惯量为0异常，不进入计算----√**

**固件升级工具的弹窗问题----√**

**讲课PPT----√**

**单位%------√**

**控制器界面刷新---√**

**固件更新按钮去掉---√**

**转矩和速度指令滤波选择去掉----√**

**上位机体验：**

1. **辨识功能界面的弹窗**
2. **IO功能，使能/失能变灰**
3. **控制器界面的单位问题**
4. **左下角参数更新**
5. **点动，调试，左右**
6. **【驱动器连接】界面，可以选择连接和断开；**
7. **底栏的【工作模式】起作用**
8. **脉冲控制模式下 ，可以选择截停和导通。**

**任务：**

**导师表格**

**工装**

**Modbus地址更改**

**待解决：**

1. **关闭短路测试，无效？----√**
2. **驱动器断电，继电器动作会导致芯片死机**
3. **脉冲不准确**

**状态字：**

**拔掉编码器线，上电，是先0x231失能，再报错0x218.**

**没有动力线，上使能不报错。**

**电源线，改！**

**脉冲数不对，要确认。**

**白噪声实验**

**------------------------------------------------------------------**

**1.03功能码 读功能有问题。---无问题**

1. **减速机结构验证**
2. **上位机界面使用优化，陈林的建议（9条）**
3. **示波器采集，会出现曲线翻头情况。（偶然会触发，待排查）**
4. **英文版本的上位机**
5. **更新电机ID表**

**现在工装还存在的问题：**

**ST7,脉冲测试不准确。**

**新改的短路待测试。----√**

**各种限制，停止，回零之类的---√**

**加入识别驱动器系列的标志**

**平行控制算法**

**看今天测试结果，优化**

**ABZ电机问题：（1）更新上位机；（2）找远伦协助；（3）**

**工装加入版本检查**

**上位机左下角更新参数，待验证---确实有问题，待修改**

**模拟量测试界面制作---√**

**皮带测试---√**

**上位机发布一版---√**

**ST7模拟量测试今天搞定**

**ST8模拟量学习**

**把蔡泽凯提供的控制器书籍必须安排到进程里**

结束符

# 第一章 NOTE

## 1.1 碎笔录

MAC（Media Access Control Address），媒体存取控制位址，也称局域网地址

**EEPROM的全称是“电可擦除可编程只读存储器”**

EEPROM 一般是用作存储程序运行时 要掉电不丢失的数据

FLASH 一般是用来存储程序的；

FPGA 是一种芯片，FPGA可通过改变芯片内部电路结构，满足产品需求。一般CPU编程是改变寄存器配置完成的，芯片逻辑功能并不能改变。

FPGA在抗干扰和速度上有很大优势，后来因代码实现麻烦，把麻烦的算法，CPU提交给FPGA，FPGA 计算完成后把结果返回给CPU

智能手机芯片，比如高通/Ti的芯片，一个ARM核控制整体运算，一个DSP处理语音编解码，一个GPU负责图像运算，一个基带和天线处理模块负责通信，再加一些其他模块。

学习软件 modbus poll

学习EtherCat

离散量输入 只写，1位，单个比特

线圈 读写，1位

输入寄存器 只读，16位，16比特

保持型寄存器 读写，16位

最常用的功能码： 03 04 06 10

0x03 读多个保持寄存器

0x04 读输入寄存器

0x06 写单个保持寄存器

0x10 写多个保持寄存器

目前关于Modbus比较权威的官方标准文档有两个：

1.modbus\_application\_protocol\_specification\_v1.1b3.pdf

2.基于Modbus协议的工业自动化网络规范 GB-T19582.1-2008.pdf

注意一点：写保持寄存器，需要区分0x06写单个寄存器和0x10写多个寄存器，而读保持寄存器不区分读单个和读多个，当需要读单个保持寄存器时，也是使用0x03指令，指定读取数量为1。

|  |  |  |
| --- | --- | --- |
| 功能描述 | 功能码 | 异常功能码 |
| 读线圈 | 0x01 | 0x81 |
| 读多个寄存器 | 0x03 | 0x83 |
| 写单个寄存器 | 0x06 | 0x86 |

协议功能、通信内容、通信规则、通信数据格式

地址域，功能码，数据，差错校验码（协议数据PDU）

master 端发送数据请求消息

slave 端接收到正确消息后发送数据到master端响应请求

master端也可以直接发消息修改slave端的数据

modbus RTU/ASCII 采用二进制数据方式直接传输数据

modbus TCP 采用16进制方式串联一起，以TCP码形式进行数据传送

modbus PLUS 是高速令牌传递网络

TCP---Transport Control Protocol

IP-----Internet Protocol

ASCII 模式采用LRC校验

RTU 模式采用16位CRC校验

PDU 简单协议数据单元

ADU 应用数据单元

音响旋转按钮结构

主站从站

学习EtherCat

看 埃斯顿代码和书籍

VB

报销单：

（1）打印，按照预览75%。

（2）在企业微信-审批-报销申请-附上打印的文件

（3）填写报销单罗列事项，签字并提交

刚刚客户调试经验总结：2022.3.30

1. 先沟通，判断问题出现在哪里
2. 调节参数和各种增益指数
3. 若过程中发现异常，比如今天遇到的使能，电机不使能，优先看线路插头是否正常；若声音过响可降低机械增益；摆动幅度过大不稳定可增加电气增益；
4. 一个调试细节：在电机使能状态稳定下，可直接调节参数和增益观察现象。循序渐进。

**EtherCAT是一种实时以太网技术**，由一个主站设备和多个从站设备构成。主站一般使用倍福公司开发的TwinCAT软件作为主站控制器，从站使用从站控制器（ESC），如专用集成芯片ET1100或ET1200

EtherCAT的数据帧结构包括：

14字节的以太网帧头，2字节的EtherCAT头，44-1498字节的EtherCAT数据，4字节的帧校验序列。

a.以太网帧头中包含6字节的目的地址，6字节的源地址和2字节的帧类型。

b.目的地址是接收方的MAC地址，源地址是发送方的MAC地址，帧类型为0x88A4。----MAC，Media Access Control Address，媒体存取控制位址，局域网地址

c.EtherCAT头包括三部分：11字节的EtherCAT数据长度（所有子报文的数据长度的总和），1字节的保留位，4字节的类型（类型固定为1，表示和EtherCAT从站通讯）。

d.每个子报文包含3部分：10字节的子报文头，最多1486字节的数据，2字节的WKC（工作计数器，记录子报文被从站操作的次数）。

e.其中子报文头可分为7个部分

|  |  |
| --- | --- |
| 名称 | 含义 |
| 命令 | 寻址方式及读写方式 |
| 索引 | 帧编码 |
| 地址区 | 从站地址 |
| 长度 | 报文数据区长度 |
| R | 保留位 |
| M | 后续报文标志 |
| 状态位 | 中断到来标志 |

P1631

一个是空间矢量，一个是定转子磁链空间矢量

进行合成 三相绕组共同产生的基波合成磁动势

空间矢量方程

定子、转子、空间矢量方程、ABC轴系表示的电压矢量方程。

V6

V6E

V7电机 绝对值17位

V7E

埃斯顿代码FOC部分罗列：

两个公式

电流调节器1

电流调节器2

电流前馈

对相处理

电流闭环磁场定位

异步拖动

SVPWM

InputFilter

SDRAM 同步动态随机存取内存

前后台系统，指外部事件的响应在中断里面完成，事件的处理还是在轮询系统中完成；中断称为前台，main函数里面的循环称为后台

**轮询系统---前后台系统---多线程系统**

主要体现的是系统的**实时性**是否提高

在多线程系统中，根据程序的功能，把程序主体分割成一个个独立的，无限循环且不能返回的小程序，这个小程序称之为线程。每个线程都是独立，互不干扰的，且具有优先级，由**操作系统做调度管理，**

|  |  |  |  |
| --- | --- | --- | --- |
| **模型** | **事件响应** | **事件处理** | **特点** |
| 轮询系统 | 主程序 | 主程序 | 轮询响应事件，轮询处理事件 |
| 前后台系统 | 中断 | 主程序 | 实时响应事件，轮询处理事件 |
| 多线程系统 | 中断 | 线程 | 实时响应事件，实时处理事件 |

为了之后工作需要，RTOS系统的学习可以提上日程。

lOT ---The Internet of things---物联网

合成 旋转磁动势F

模拟量采样，ADC采样进行模数转换

MCU 内部有内存、处理器、flash等其他模块

MPU 只有处理器，需要搭建外设才能构成系统

CPU 中央处理器

数据处理 DSP

图形显示 GPU

片上外设 SoC

当某个事件发生时，才调用对应函数，称之为 **事件驱动**。

Static 全局变量只能本文调用

Static 函数只能本文调用，强调私有性

Volatile 编译不能被优化

基于链表理解调度机制，知道内部机制，队列的使用

链表 是物理存储单元上非连续的、非顺序的存储结构，比如结构指针的定义。

汇编指令 读 load—LDR 读寄存器

LDR RO,[adde]

\_\_HAL\_TIM\_CLEAR\_IT(&TIM2\_Handler,TIM\_IT\_CC1|TIM\_IT\_UPDATE); //清除中断标志位

U8 u16 u32 以及uint\_t 8 u\_int\_t16 区别哪里？

电容触摸按键，利用定时器计数，复用一个GPIO口当做电容。当定时器要捕获值时，先对该电容释放电量并清除计数值，然后开始获取定时器的计数值并返回，根据多次获取的返回值 取值触发“按键功能”（也就是触发条件）控制LED灯亮灭。

内存保护，还是要初始化（MPU\_Region\_InitTypeDef）和调用保护函数（MPU\_Set\_Protection）。利用三个按键初始写，读，保护操作。

RTC（Real-Time Clock），指实时时钟芯片。

对日期，时间进行配置和初始化，在芯片中有专门开辟函数给这一个功能

SDRAM，就是读取内存空间大小

usmart，通过usmart调用单片机里面的函数，实现对LCD显示和LED以及延时的控制。

RNG 随机数发生器

VB：

2021版本：<https://www.microsoft.com/zh-cn/download/details.aspx?id=50722>

2012版本：<https://www.microsoft.com/en-us/download/details.aspx?id=7030>

安装教程：<https://www.toutiao.com/i7055285064481128974>

Permission 许可证，同意

今天俺就看摄像头，内存管理，U盘，鼠标键盘这四个工程，之后全部看核心算法。

对代码的视野，联想。

多看别人代码，对架构和逻辑，相关地址，函数等有自己的理解。

摄像头，涉及到的各种图像处理，内存管理包含的内存开辟和释放。

如果不借鉴别人代码，自己可能永远都不知道外面的代码，超出自己认知的架构有多么大。

USB U盘开始引入官方文件，协助处理（usbh\_msc.c usbh\_msc\_bot.c usbh\_msc\_scsi.c），再加入初始化配置和U盘读扇区和IO操作。

由框架，逻辑（※※）---🡪细节处理（※※※） ---即大量思考、理解

实现方式---🡪最终呈现效果

易语言，抱闸，锁轴，Verilog

电机振动，不锁轴，飞车，噪声大，限位到位有误，丢脉冲，

增益值大小意义。

抱闸：当电梯厢处于静止且马达处于失电状态下防止电梯再移动的机电装置。在某些控制形式中，它会在马达断电时刹住电梯。

梳理框架：

1. 告一段落

机械元件在转矩作用下都会产生一定程度的扭转变形，故转矩有时也称为扭矩。

X轴和Z轴电机的到位 信号

存在响应效率低的地方：

1. 驱动器上电时间。
2. Mosbus传输数据初始化复位时间 Modbus\_ready();
3. 为什么每个驱动器上电初始化时间不一致？问广谱
4. IO检测，使能信号，IO模块响应时间。
5. 使能后，响应时间，如何判断
6. 短路测试，驱动器上电断电，使能，所需时间
7. 若遇到错误信号如何清除，比如 电角度错误，FF1F

以上问题，解决的核心在于 modbus数据传输，用402状态字，探针方式，以便于获取驱动器状态。比如 伺服准备好。

尤其是在电角度方面，可采用清除错误，多次使能的方式。

但是402控制字和402状态字，我所知甚少。

**ARM定时器区别：**

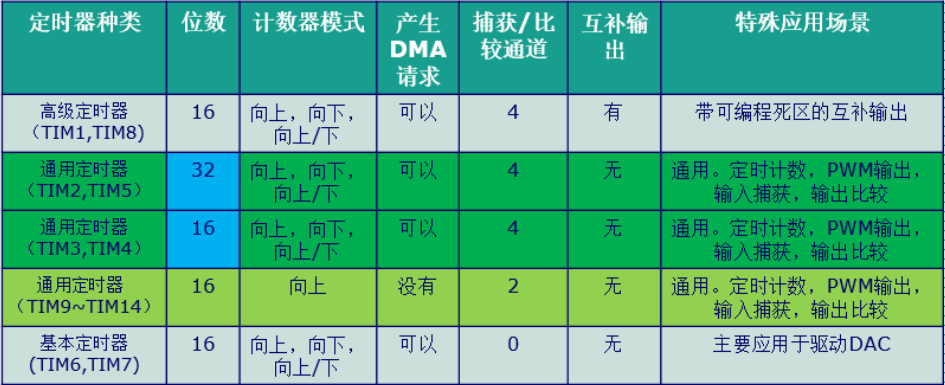
**（1）通用定时器：**

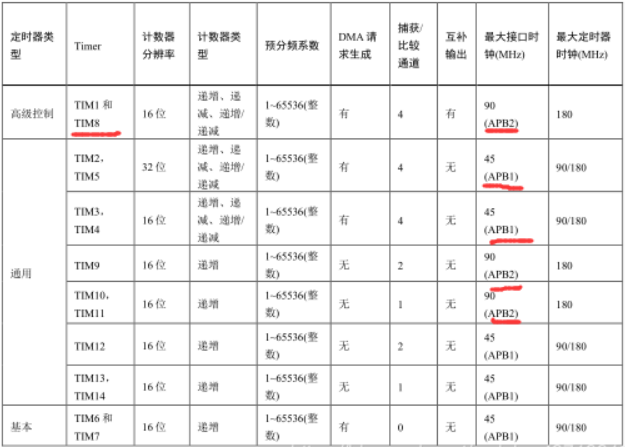
TIM2-TIM5：16位可编程预分频器（0~65535），4个独立通道，可输入捕获，输出比较，PWM生成，单脉冲模式输出。

TIM9-TIM14:：16位可编程预分频器（0~65535），2个独立通道，可输入捕获，输出比较，PWM生成，单脉冲模式输出。

**（2）基本定时器：**

16位可编程预分频器（0~65535），可触发DAC同步电路，产生中断和DMA请求

16位定时器和32位定时器，自动重装载值都是在0~65535之间



IAR Systems\Embedded Workbench 7.5

国有企业 事业单位？？

16位定时器和32位定时器 区别在于精度

在同一时间中 处理二进制的位数叫字长。

通常处理字长为8位数据的CPU叫8位CPU，32位CPU就是在同一时间内处理字长为32位的二进制数据

磁编 ST8 电流环频率 32k

光编 ST8 电流环频率 40K

BLDC 无刷直流电机

永磁体，就是加入稀土等材料后可以保持长期磁性的物质

线圈绕组，经过交变电流后，会产生变化的磁场，从而推动电机的转子进行旋转。

闭环，反馈，22us, 65us。

D轴和Q轴

杨工，小茂

1. 未开放字典整理。
2. ST8上位机说明手册整理
3. 看ST7上位机代码，三部分（按键，协议的两部分）
4. 看公众号和网址

SVPWM 是空间矢量脉宽调制

在int 16类型中，0xFFFF表示的是-1

0x8000表示的是-32768 即 到

220\*=311 380\*=537V

IQmath库

驱动器的驱动电路是H桥

三相永磁同步电机的驱动电路 输出 三相电压、、，相位之间相差120°

Clark变换源于驱动电路的三相，变成静止坐标系αβ的过程。

永磁同步电机的坐标系变换：Clark变换、Park变换

**Clark变换：**由静止三相坐标系ABC变换到静止坐标系αβ的过程称之为Clark变换。两轴间相位差为90°

**Park变换：**本质是静止坐标系αβ乘以一个旋转矩阵，从而得到dq坐标系（也叫交直变换）

Park变换的旋转矩阵指 2轴定子坐标系转换到2轴转子坐标系；

ABC轴-----αβ坐标系------dq坐标系

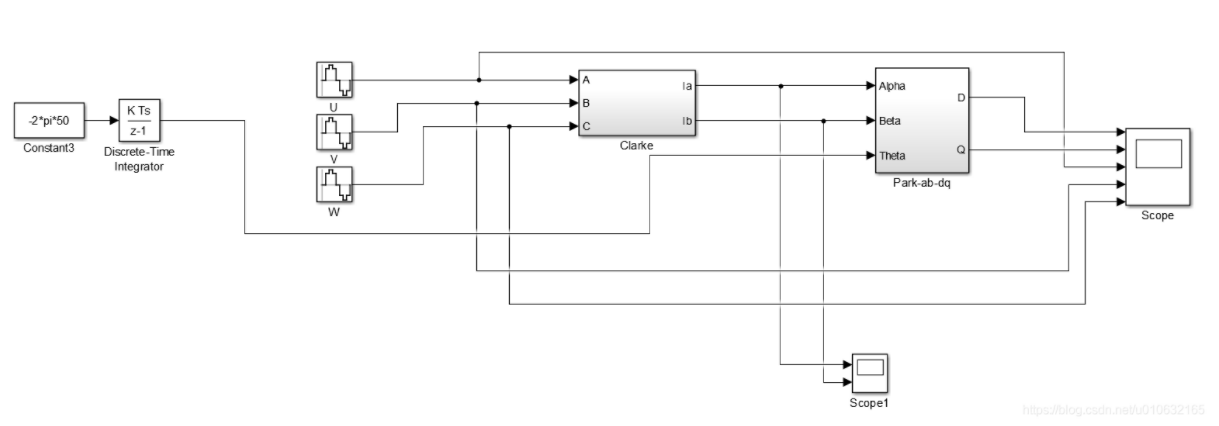
Stator定子 rotor 转子

αβ坐标系是随时间变化的正弦波形

dq坐标系是 相对于定子来说是旋转的坐标系，转速的角速度和转子旋转的角速度相同

d轴方向与转子磁链方向重合，又叫直轴；

q轴方向与转子磁链方向垂直，又叫交轴；



FOC电流采样方案：

电流采样关键主要是硬件的设计和采样点的设置。 最后会进行电流重构。

同时要注意窗口时间

**RS232和RS485区别：**

****

MDMA\_CxNDTR 寄存器**包含通道停止时剩余数据项的数目，**这样软件便可以确定通道

中断前已传输了多少数据项。

V6和V7电机的区别：

1. 内部构造，一个是贴装式，一个是内嵌式；
2. 旋转功率，V7的所能承载的功率更大；

PMC 指生产管制与物料控制。

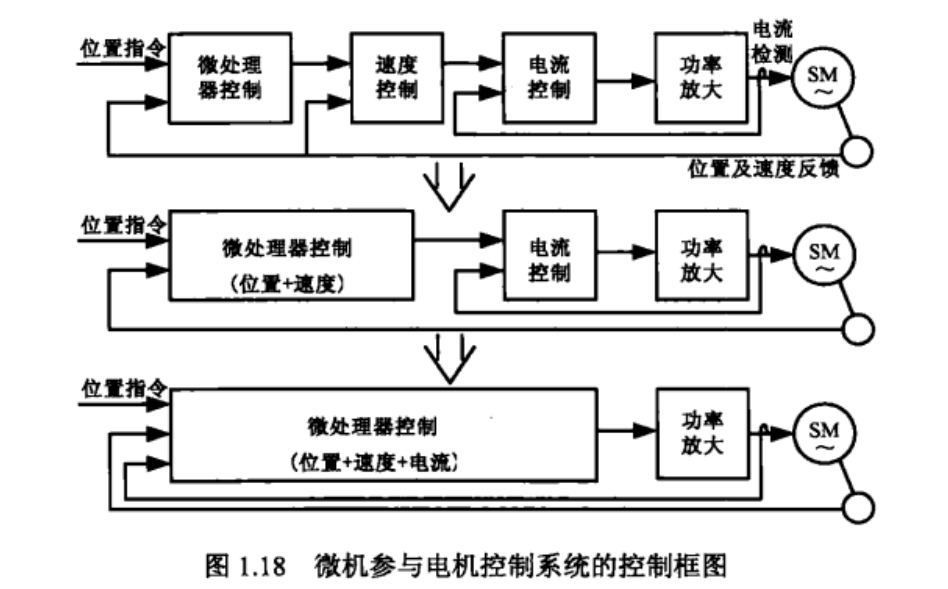
电池线--多圈电机的---雷赛的定义

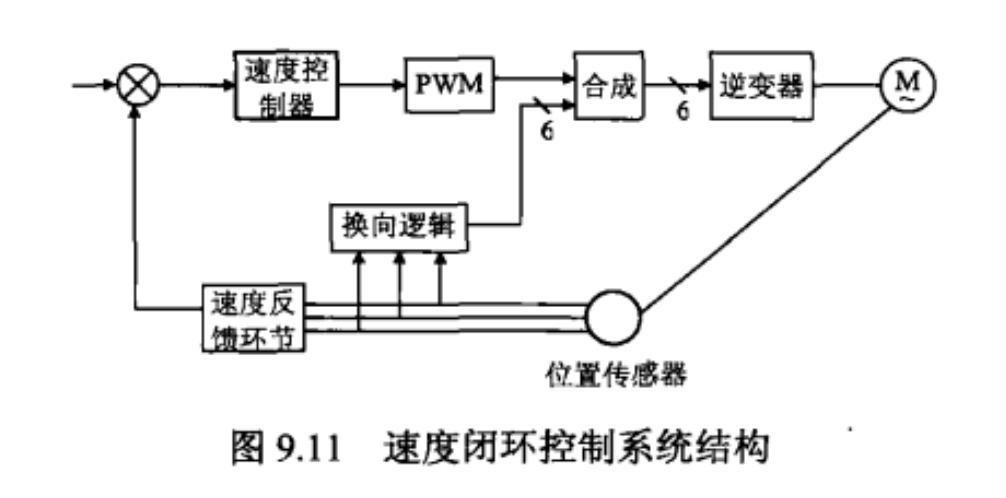
1波特即指每秒传输1个符号。

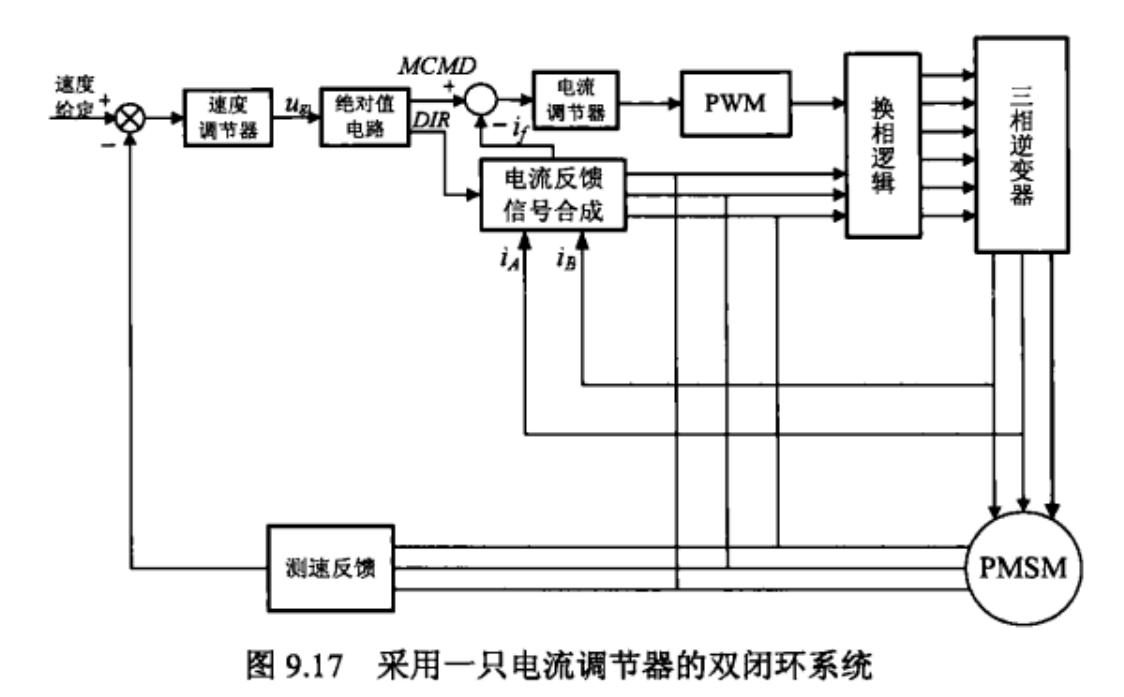
UART每秒钟传送240个字符，每个字符格式包含10位（1个起始位，一个停止位，8个数据位），这时的波特率为240Bd，波特率为10\*240个/秒=2400bps。

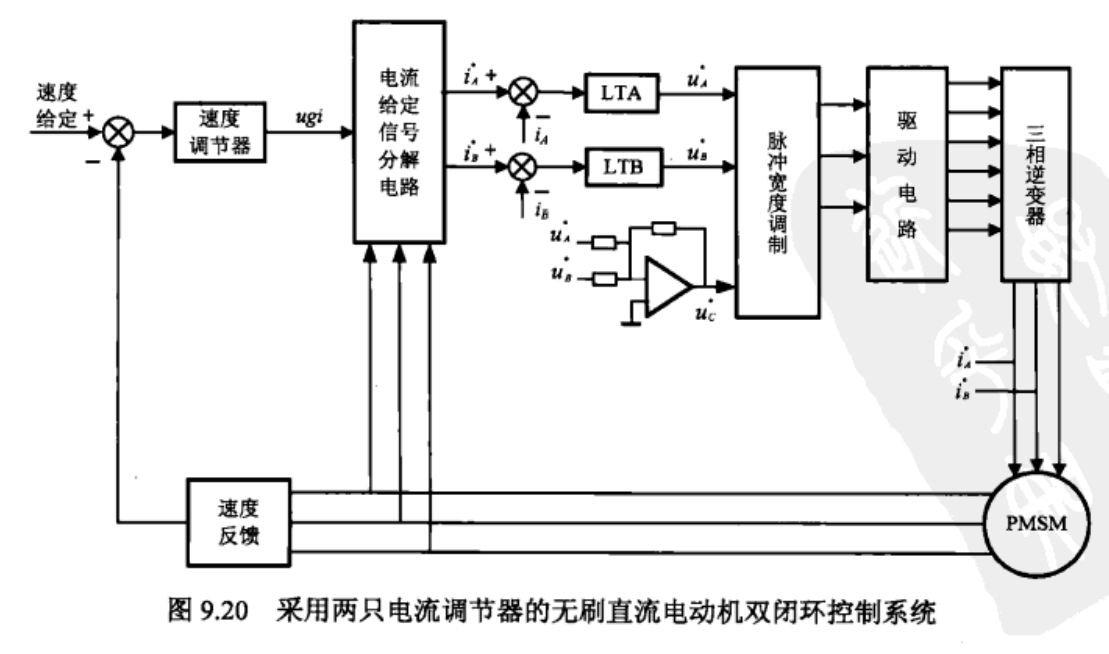
## 1.2三环框图

将用于ST7上位机的三环：电流环，速度环，位置环。









个人理解：

**电流环：**

位置指令输入---🡪微处理器控制---🡪速度控制---🡪电流控制---🡪功率放大-🡪SM

---------------------------------------------------

速度环：

输入---🡪速度控制器---🡪PWM---🡪合成---🡪逆变器---🡪M

-----速度反馈🡨------------- 位置传感器🡨------

## 1.3 驱动器调试

P220

Pr00.03 设定实时自动调整有效时的响应性：

设定值变高，则速度响应性变高，伺服刚性也提高，但容易产生振动

Pr00.04 惯量比

惯量比=（负载惯量/转子惯量）\*100%

Pr1.11 速度前馈滤波器

逐渐提高速度前馈增益，。在一定速度状态下动作中的位置误差，根据速度前馈增益的数值会变小，如一下公式：

位置偏差=指令速度/位置环增益\*（100-速度前馈增益）/100

**注意：和速度前馈一样，虽然转矩前馈滤波器的常数越大动作音就会越小，但加速度变化点的位置偏差就越大。**

**Pr1.13 转矩前馈滤波器**

提高转矩前馈增益，因为一定加减速时的位置偏差可以接近0。

**Pr2.00 自适应滤波器模式设定**

设定自适应滤波器推定的共振频率和推定后的动作。

**Pr2.01 第1陷波频率**

即设定第1陷波滤波器的中心频率，设定值为5000时，陷波滤波器的功能无效。

**Pr2.03 第1陷波深度选择**

即设定值第1陷波滤波器的中心频率深度；设定值越大，陷波深度越浅，相位延迟变小。

**Pr2.20 制振频率**

设定抑制负载尖端振动的制振控制频率。测量负载尖端振动的频率，单位0.1Hz

第1**位置环增益：决定位置控制系统的响应性**。提高位置环增益，可以缩短定位时间。但是设定值过大会引起振动。

第1**速度环增益：决定速度环的响应性**。为提高位置环增益来提高伺服系统整体的响应性，需要加大速度环增益值。但过大会引起振动。

第1**速度环积分时间常数：设定值越小，停止时的偏差越快接近于0**。

第1**速度检出滤波器：**速度检出后，可设定低通滤波器（LPF）的时间常数为6个阶段（0~5）。设定值大则时间常数也大，虽然**可降低电机噪音，但响应性也会下降。**

第1**转矩滤波器**时间常数：**可抑制因扭曲共振发生的振动。**

**速度前馈增益：**内部位置指令中计算后的速度指令与此参数的比率相乘的值加算到来自位置控制处理的速度指令中。

**速度前馈滤波器：**设定速度前馈输入相关的一次延迟滤波器的时间常数。速度前馈滤波器设定为50（0.5ms）的状态下，逐渐提高速度前馈增益，速度前馈生效。一定速度状态下动作中的位置偏差，根据速度前馈增益的数值有如下公式变小；

位置偏差[指令单位] = 指令速度/ 位置环增益×（100-速度前馈增益）/100.

[指令单位/s] [1/s] [%]

第1增益，第2增益 区别

uint32\_t NDTR;

\_\_HANDLE\_\_)->Instance)->NDTR;

ST8驱动器：

0x6041---61443---0xF003 未使能条件下的数据为0x231

使能成功后的数据为0x637

下使能，状态字先为0x631，再变为0x231

报错情况下是0x218

报错清除后是0x250

复位后的数据为0x250

旋转电机的三大应用：皮带，丝杆，直驱

机械增益：到位时间

速度波动带宽：速度观测器带宽

结束符

# 第二章 ST7工程笔记

如何使用IAR工程

主要任务目标：

1. 完善modbus协议
2. RS232协议完善

安装软件时，选择以管理员身份打开，目的：为了给安装软件更高的权限，避免电脑自带杀毒软件删除软件关键文件。

我们的驱动器 使用的协议是多摩川。

FSMC：可变静态存储控制器

FPGA

先进入中断，然后扫描，根据条件判断

Modbus 参数返回字节数计算 //两种计算方式，一种返回值，一种定义形参

Modbus 参数校验码获取

CRC校验码，套4层结构体

需要咨询的问题，1.485协议时间规划；2.需要哪些参数，应该是需要对应地址和数据类型函数建立。

RS232 GND RXD TXD

RS485 A B两端的差分信号，它指发的TTL电平信号

Modbus 指的是一种数据结构，即协议

模块里应该是有485芯片，RS232转485的芯片，要把对应的引脚接好

ABZ读头 没有极对距

ST8编码器反馈，选择AB头，且z相误差消除应该是要取消的。目前ABZ头，只支持AB。

判断读头是否有问题，需要安装到电机跑一跑才知道。

Modbus协议定义了一个控制器能认识的消息结构，比如 01 03 00 35 00 02

保持型寄存器 读写 16位

输入寄存器 只读

线圈 读写

在Isr 文件中，数据状态机中断，DataStateSystemManage()；

参数ID存放在eeprom中。

0x30 00 14

01 03

步科官网，型号MT4414T

DIO h8

RS485.c UART4 DMA

RS485\_Modbus.c USART2 DMA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对象ID | 对象名称/描述 | 类型 | M/O | 种类 |
| 0x00 | 厂商名称 | ASCII字符串 | 强制的 | 基本 |
| 0x01 | 产品代码 | ASCII字符串 | 强制的 |
| 0x02 | 主要修订本 | ASCII字符串 | 强制的 |
| 0x03 | VendorUrl | ASCII字符串 | 可选的 | 规则 |

Unsigned char 和 char：

区别：char是有符号的，最高位为符号位；unsigned cahr是无符号的。

在输出以及数据赋值运算时，无符号的数值是不会有区别的，但有符号的，计算机还要进行额外计算，进行为扩展，会出现一些问题。

if(Addr >=0 && Addr <= 4132) //0x1024 为什么是这个取值范围？

大连榕树

0x5A 数据06 E0 00 00 ST7和ST8是不区分大小写

0x59 数据00 00 06 E0

U8 256

U16 65535

U32

**MODBUS协议的 掉帧率，误码率 检测**

485丢包原因：情况1，发送数据过于简单，主机发送完数据后，还未切换到接收状态，从机不需要做过多复杂处理，即回复过快，导致主机接收数据丢失。当主机切换到接收状态时，已经丢失部分数据。

DMA指，不由CPU来控制过程的操作方式，CPU只负责开始和结束的管理，中间过程不需要介入

明确工作计划：

1. 这周，即4月17日之前，测试modbus协议正确性，分配地址空间
2. 下轴，即4月24日之前，搞定modbus协议
3. 下下周，即4月30日，交付成果

这周计划，读懂协议函数，有哪些处理函数，通信是如何实现的，今天内容，明白通信的机制；

有精力的话，把MP157的单片机知识，学习。原神，拿到行秋后，可以不玩了。

一切都在好转，保持好心态

有一些细节处理，不太理解，无论是从获取（1）数据的字节长度，（2）还是对协议的解析

当然最终case 的功能码03 06 10，是完全可用的

**现在问题出现在，没有足够的地址分配和数据，也就是在case 9中的 03 06 10**

**Modbus 03H操作， RamDataModbus[48];**

**Modbus 06H操作， RamDataModbus[Set\_EEPROM.SetDataId]**

现在我需要知道**整个modbus模块**，包括 初始化，接收，发送，协议数据格式。

重要的是modbus整个模块，因为以后可能就交给我负责，必须整明白。

然后要理清楚，**servoalldata的协议单，与03H定义的case 区别和联系**。

Modbus 使用的是 **DMA\_Stream5**

Generator

EtherCAT 版本 比如402状态字，是固定死的

阈值：来源于函数的定义域，是临界值，指一个效应能够产生的最低值或最高值

阀值：是工程学术语，值“到达安全极限”，“到达发生某种事件的极限”

闲着没事，开始看 Ethercat

**03H ReturnParameter = RamDataModbus2[80];**

**06H RamDataModbus[Set\_EEPROM.SetDataId]= (Val>=0 && Val<=1000)?Val:0;**

**unsigned char rxbuffer2[134]; //modbus使用的缓存数组**

**unsigned char txbuffer2[134];**

**unsigned short ParamaterIDmap(unsigned short ReadID);**

**磁编和光编**

镜像是什么

多摩川协议

Z相，回零，消除误差

磁栅，光栅，信号的来源

通讯式，通讯方式

绝对式读头，有极距

绝对 位置是唯一的

增量式 以上电时，为零点

寻相，找零

角度尺

高云

注释： Ctrl+k

取消注释： Ctrl+shift+k

按键显示模块TM1629B

1. 03H和06H的Switch case 赋值好弄，就是如何把这个值真正写入MCU中，在哪可以获取到？----写入是存放在管道，通过main函数中写入
2. 另外，逻辑分析仪捕捉到的脉冲是零散的，有明显的干扰？如何理解---所以电路加互感电阻之类的，减少干扰。

面向对象，就是把特征值封装在 一个结构体中，调用功能模块

面向过程，流控制和指针

EEPROM

FLASH

Modbus模块部分，我特码要自己改造！

**一个16进制的值 a=0xFFFF，(a&0xff)结果为0x00FF，然后<<8，表示左移8位，**

**即0x00ff变成0xff00---该方式被否决了，其实是标准正确的**

板子尺寸问题：

原因在于，机器微割时，有毛刺没处理好

**03H**

**06H 这两个问题很大**

驱动器只在电流环的方式下运行，位置环与速度环不运行，我们称之为驱动器运行于力矩模式。（对于同一个电机，其产生的力矩T=电流I\*力矩常数Kt。所以给定了电流就等于给定了力矩）

转子位置（电角度）

一个很重要思想，管道，把数据放到管道里，while（1）里面一直存

偏移地址

数据分两路：一路用作显示，一路用作电机驱动使用。

**ServoDataArea[P06\_44].Data.all**

**无论是 二进制，十进制，十六进制，在程序中都是二进制，可直接加减运算**

\*(pData++)

**实时写入，非实时写入=上电生效**

**Rw\_u 可读可写，非立即有效**

**int RamDataModbus[100]; ---06H**

**int RamDataModbus2[100]; ---03H**

**SevroParameterReg[MapId].EffectTime ----键值数据表**

**RamDataModbus[Set\_EEPROM.SetDataId] -----06H写入数据缓存数组**

**Data\_Int32.all -----06H写入数据，的赋值对象**

**ServoDataArea[MapId].Data.all -----06H写入数据，case2的赋值对象**

**EepromData[MapId].Data.all -----06H写入数据，case4的赋值对象**

**ModbusFramesRx.FrameStructReg.StartAddr -----modbus起始地址**

**问题点集合：**

**P00.17 异常，出现4字节，低位为1**

**P00.18 可正常，不过之后的键值，连续读下一位**

**P00.19 正常**

**P00.22 可正常，连续读下一位**

**P00.23 正常**

**有几个问题：**

1. **多读两个字节，原因在哪----标准485协议是读单个字节，即一帧8位**
2. **数据长度，返回4字节数据，即10字节长度，出现问题----已找到原因在哪，是原代码已经固定死只能读/写8字节长度，不够灵活满足不了需求，在考虑解决方案**
3. **06组，返回值少一个数据，即CRC校验码-----已解决**
4. **06组，返回值，修改值若超过限定值会返回该非法值----已修复**

**直线电机，是12位，后面几位舍弃了**

**旋转电机，是17位**

**从P03组开始有问题。**

**算了，先把所有参数搞好再调试具体某一参数的问题。**

**每周的工作完成后，一定要拷贝一份工程做记录，避免意外**

**SVN GIT**

**低压驱动器：**

**电流环，速度环跑的不太行。**

**位置模式最大速度到不了700，速度模式最大速度到不了900**

**06H short ParameterId = 0,Val = 0; 这句定义，可能关系到字节长度**

**位置环和速度环 增益，参数比较小；400 500**

**速度积分时间 比较大 10000**

**转矩指令滤波 只有150**

**现在有两个问题：**

1. **逻辑分析仪 乱码情况----引脚飞线要接在芯片那端，如果在用户端口那里存在很多干扰，电路的部分功能是滤除干扰**
2. **定义并且使用了，程序却提示 未使用？**
3. **有一个尴尬问题，手册和ST7字典协议，描述存在矛盾情况。---已汇报给广谱**

**各种字典已经完成，剩下调试21 20段的协议**

**其实我对程序中的很多结构体名字 不是很熟悉。---现在好多了，要适应命名方式**

**微分超前，预测前面几次的数值**

**中断向量具备两个属性：抢占属性、响应属性。编号越小，优先级越高**

**抢占指打断其它中断的属性，会出现嵌套中断。抢占属性由PreemptionPriority 配置。**

**响应指在两个抢占优先级相同时，会优先处理响应优先级高的中断SubPriority 参数配置。**

**情况1：正常执行A事件，则不能被抢占优先级相同的B打断**

**情况2：若抢占优先级相同的A、B中断同时到达，才会比较响应优先级，内核会优先考虑级别高的。**

**响应优先级，也成子优先级。**

**数字量指**

**模拟量指 不能人手控制，采取单片机电路获取数值的方式，这种获取值叫模拟量**

**任务：**

1. **主要是485通讯的实时性记录----完成**
2. **管道程序写入，滤波处理，ADC采样处理**
3. **测试工装，代码优化。**
4. **毛刺处理。**
5. **体验感+保护措施+运算量**

**末端抖动，末端补偿**

**电磁声 由电感造成**

**现在困惑：**

**1.对程序，产品，个人定位**

**现在生效情况：**

1. **断电生效----已经说明，按照重新上电才生效**
2. **上电状态就可生效**
3. **使能状态下生效---速度，增益**

伺服电机 使能有异常声响，速度环增益过大！

C=3

3a+^3b+3=0 B(-3^3，0)

PcCom 上位机文件夹

P21.05 检测ADC采样 U相电流有效值

才意识到一件事，ST7驱动器的上位机 是RS232通信，采用的帧头是 A5 5A

链表是什么？

位域：

指把一个字节中的二进制划分为几个不同的区域，把几个不同的对象用一个字节的二进制位域来表示。且一个位域必须存储在同一个字节中，不能跨两个字节

数据状态机：

通信扫描---🡪 判断数据帧头（A5 5A）---🡪传输数据---🡪case条件判断---🡪case 01跳转函数

写都是用管道思想写入数据

可能自己也想疯狂一把。

今天下午的任务是什么呢：

1. 看上位机代码，框架，实现逻辑。

char FirstByte;

char SecondByte;

char ThirdByte;

char FourthByte; 这四个定义是什么意思呢？

每个人都有自己的难处，做好自己，做最好的自己！

今晚流程：7点半走人，回去理发，然后调整状态氪金抽刀！

给平平抽烟绯，氪！

仅此而已，以后就做个零氪玩家，玩腻了调整状态，晚上回去练操！

广东政务服务网深圳市

既然选择了电机，驱动器领域，那就要踏踏实实的做好

typedef struct {

unsigned char DATA\_FourthByte;

unsigned char DATA\_ThirdByte;

unsigned char DATA\_SecondByte;

unsigned char DATA\_FirstByte;

}BIT32DATA;

ST7驱动器的 RS232协议：A5 5A 00 02

串口屏：A5 5A B6 F0 A0 A0 A0 A1

A5 5A 00

看了几篇文章，我也要努力加油，踏踏实实做好自己的事情。

Subnod 子节点

Nod 节点

可以选择发一个朋友圈

共用体占用的内存等于最长成员占用的内存，且同一时刻只能保存一个成员的值，若对新成员赋值，就会把原来的成员的值覆盖掉。

位域，n: 4，代表变量n只占用4个； 超出部分被截去

ST7上位机数据帧： 5个字节

帧头（2）+站号（1）+数据长度（1）+操作字（1）+数据（length）+CRC（2）

Union 称作共用体，是因为union中所有变量共用一段内存，存储在相同的内存地址中，其中某个值被赋值后，其他变量也会共用这个值。

共用的值，从下面往上赋值

16 typedef union {

17 struct {

18 int byte1 : 8; // [0:7]

19 int byte2 : 8; // [8:15]

20 int byte3 : 8; // [16:23]

21 int byte4 : 8; // [24:31]

22 }data\_bits;

23 int data;

24 }test\_union;

小端对齐：低地址存放低位数据，高地址存放高位数据

大端对齐：与之相反

加载数据：ServoDataRefresh.c ---🡪 DataLoadToServo();

S曲线参数计算---这个查！---没查到

位置环FIR滤波器参数计算

电子齿轮比参数计算

速度环PI参数计算

今天的主要任务是看上位机通信，关于修改值写入，读取操作，执行顺序还没看明白。

明天的话，看工装，把ST7/ST8的测试工装整合并完善。

有一件事没想清楚：

为什么显示屏，是B0结尾？B0代表的意义是什么呢？

答：例程是0x30，也就是阿拉伯数字0，ASCII表。

草，USB线买错了！

今天需要解决的问题：

1. 理解ST7上位机工作架构
2. 上位机如何读取，修改值

我需要考虑的一件事：

ST7上位机通信协议是什么？

ST7上位机是怎么实现：读取数据，修改参数，示波器

ST7上位机代码逻辑是什么？

通信协议：A5 5A 0/1 帧长度 操作字 数据 CRC

其中：**帧长度**，**操作字**，**数据** 具体指什么？怎么操作的？

操作字：**00（通讯检查）**，**01（读操作）**，**02（写操作）**，**06（示波器）**

pictrlabs

Pctrlpos\_float

型号： 脉冲型，模拟量（力控），总线型

**OS 猜测指的是示波器**

**Sampling 采样**

**Sample 采样**

**Simple 简单的，简明的**

**他喵的，采样（sample）这个单词都写错了，草，全是这样命名的。**

**研究好现在ST7上位机和示波器这两种功能的实现，然后看看S型曲线，顺带研究测试工装代码架构。**

**按照我的思想来看：**

**按键功能实现的简单，又紧急的事件和参数**

**上位机实现的是更为复杂的多任务功能，包括示波器，伯德图，参数写入的界面优化。**

**最主要的问题是，上位机通讯功能我并不了解，怎么实现通讯的？**

**写我能懂一半，今天应该能把写功能看明白，但是读和示波器功能，这两个我是没有一点头绪，就硬看？**

**DATA\_FirstByte-----DATA\_FourthByte 这四个字节类型，是什么意思。**

**那个包子不适合我，以后尽量不吃了**

**今天的任务：**

**诺，既然看了上位机代码，汇总结果：**

1. **上位机通讯和modbus一样，是放在数据状态机中断中轮询。这个TIM2定时器的时钟频率----在servo\_peripheral.c中，500us中断一次。**
2. **数据传输协议：A5 5A** **站号(0/1）帧长度 操作字 数据 CRC**

**操作字：包括00（通讯检查）**，**01（读操作）**，**02（写操作）**，**06（示波器）**

1. **关于协议详细理解：主要用到了 union(共用体)，操作字用到了位域。**
2. **对于代码中的DATA\_FirstByte-----DATA\_FourthByte 应该是与上位机内部协议定义相关，但没有说明文档，所以是未知，需要自己开辟？**

**一个很重要的问题，在公司中被影响的习惯，尤其是思考和输出的习惯。**

**这个问题很重要。**

**示波器功能获取数据，是一串打包发送的。**

**材质特性 热胀冷缩，过孔距离太短，一般0.22-0.23**

**Bja 5个miun 孔与孔的外边缘距离5个miun**

**既然代码不想看，那么就给自己放松一下，看C#语言和QT界面制作，俺要制作一个软件来玩。**

**算了，今天下午继续 磨代码，肯代码，缕清思路**

**按照现在情况，驱动器肯定是要做的，并且要做好，貌似没有退路**

**那么依照我现在了解到的情况，可以从这几点突破：**

1. **上位机，目前显示的版本还是18年，4年前的。**
2. **相关算法的优化，比如S形曲线>T形曲线**
3. **各种隐形bug，如末端补偿，回零，支持ABZ读头，modbus协议完善。**
4. **开发新的驱动器ST7E，目前了解到的也就是移植性能更好的电流环，其他性能会提升具体指哪些方面，提升的指数暂时未知。**

**其实，对于电流环，速度环，位置环，这三环我了解的并不多（优劣，关系的比较）**

**还有一点是对于效率和时间的把握，也就是频率，现在ST7驱动器的电流环频率是8K，但是迪巨他们是能做到16K, 32K。**

**霍尔库仅限于操作外设**

**FOC运算**

**经验：**

1. **会上位机代表新奇的东西**
2. **每次晚上回去后，必须把窗户关掉**
3. **以后可以尝试早上步行去公司。**

**也就是说，要对代码很熟悉，足够熟悉**

**滤清代码：**

**首先是主函数：各种初始化，版本号确定，EEPROM写入扫描**

**再之后是状态机：**

**（1）ADC采集---TIM5--100us。采集项目：主板类型，模拟速度命令，模拟转矩命令，母线电压，温度采样**

**（2）看门狗**

**（3）主循环中断**

**（4）电机状态机中断**

**（5）数据状态机中断：modbus通讯，上位机通讯，错误信号扫描，参数加载**

**（6）按键中断---已屏蔽，失效**

**（7）显示中断**

**还有未知但存在的：**

**电流环，速度环，位置环，S型曲线，滤波，截断误差补偿，平滑，电流环频率，力矩控制，驱动器的保护措施，读头（通讯式，ABZ读头）**

**下一步的战略：**

**针对这么多的功能和内容，提炼出精华！**

**出大问题，现在脑子对很多事情容易记不住！喜欢看笔记。**

**VS visual studio**

**要明确任务：**

**先把整个框架记录好，然后对部分功能进行理解，提炼精华和细节，学习**

**想认真看代码，却看不进去，没有人带领着？自己没有入门掌握代码阅读的技巧？**

**串口中断，定时器中断，外部中断，联合中断？**

采用定时器中断：精密ADC采样，看门狗，电机状态，数据状态机（modbus，上位机，错误显示），液晶显示屏

外部中断：主循环

**Modbus数据是通过 DMA\_usart 方式接收，将数据映射到一个数组中，通过**

**Alignment 排成直线**

**数据长度的翻头变换，超过限制长度，则进行一次翻头，**

**T型曲线**

**电机驱动电路 H桥**

**傅里叶变换代码：**

**void fft(cp \*a,int n,int inv)**

**{**

**int bit=0;**

**while ((1<<bit)<n)bit++;**

**fo(i,0,n-1)**

**{**

**rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));**

**if (i<rev[i])swap(a[i],a[rev[i]]);//不加这条if会交换两次（就是没交换）**

**}**

**for (int mid=1;mid<n;mid\*=2)//mid是准备合并序列的长度的二分之一**

**{**

**cp temp(cos(pi/mid),inv\*sin(pi/mid));//单位根，pi的系数2已经约掉了**

**for (int i=0;i<n;i+=mid\*2)//mid\*2是准备合并序列的长度，i是合并到了哪一位**

**{**

**cp omega(1,0);**

**for (int j=0;j<mid;j++,omega\*=temp)//只扫左半部分，得到右半部分的答案**

**{**

**cp x=a[i+j],y=omega\*a[i+j+mid];**

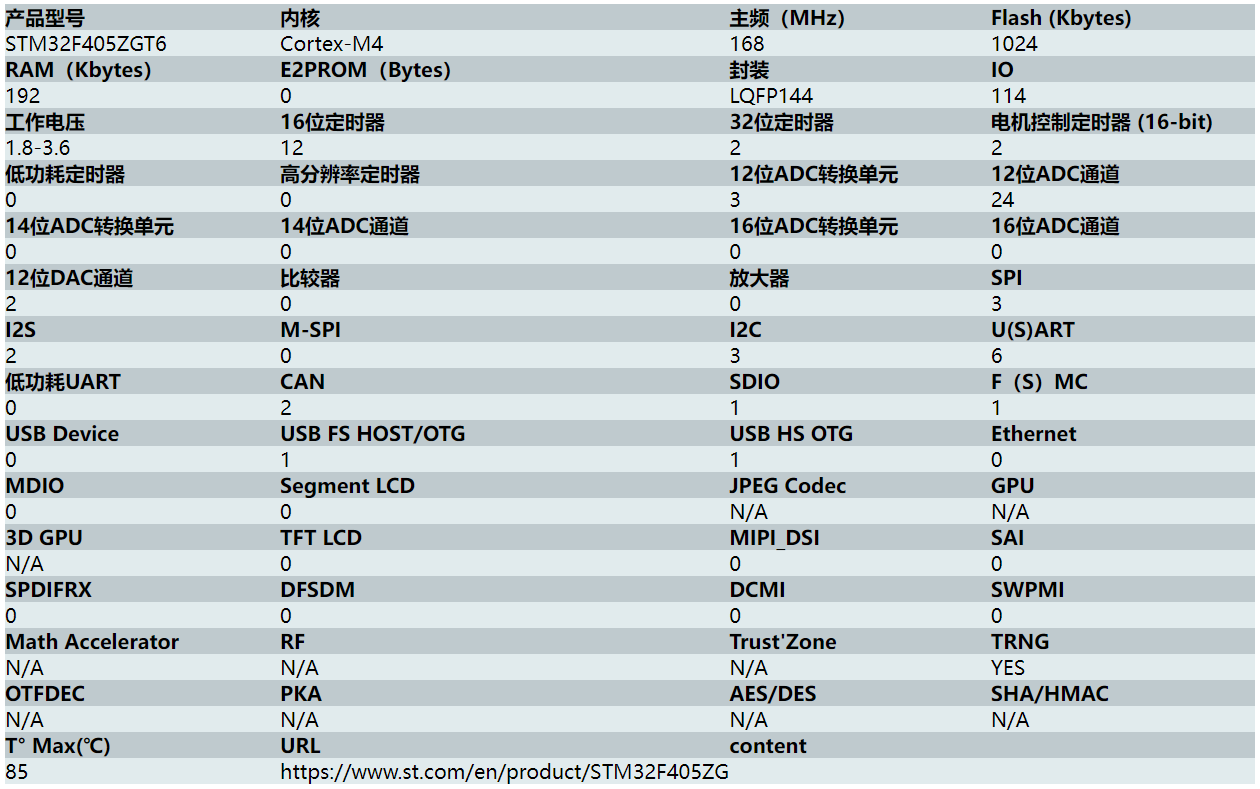
**a[i+j]=x+y,a[i+j+mid]=x-y;//这个就是蝴蝶变换什么的**

**}**

**}**

**}**

**}**

****

**图 产品405型号**

****

**图 产品H743型号**

**大端对齐和小端对齐：**

**大端对齐：高地址存放低字节，低地址存放高字节；如0x12345678，采用大端对齐是0x78563412。优势，第一个字节是符号位，方便判断正负。**

**小端对齐：高地址存放高字节，低地址存放低字节；如0x12345678，采用小端对齐是0x12345678。优势，较容易截取字节。**

**需要敲定，电机加减速，速度模式的最大速度限制**

**报错信息整理（29个）：**

**Er001**

**Er002**

**Er003**

**Er004**

**Er005**

**Er006**

**Er007**

**Er009**

**Er010**

**Er011**

**Er013**

**Er015**

**Er020**

**Er021**

**Er022**

**Er024**

**Er025**

**Er026**

**Er027**

**Er043**

**Er044**

**Er045**

**Er046**

**Er047**

**Er049**

**Er050**

**Er056**

**Er086**

**Er087**

|  |  |  |  |
| --- | --- | --- | --- |
| **报警代码** | **名称** | **处理方式** | **能否复位** |
| **Err.003** | **FPGA通讯故障** | **断电重启** | **可以复位** |
| **Err.005** | **电机与驱动器匹配错误** |  | **不可复位** |
| **Err.006** | **EEPROM参数异常** |  | **可以复位** |
| **Err.007** | **主板参数异常** |  | **不可复位** |
| **Err.010** | **硬件过流故障** |  | **不可复位** |
| **Err.013** | **绝对值编码器异常** |  | **可以复位** |
| **Err.020** | **过电压** | **加制动电阻** | **可以复位** |
| **Err.021** | **欠电压** | **检查是否是220v供电压** | **可以复位** |
| **Err.024** | **速度偏差过大** |  | **可以复位** |
| **Err.025** | **电机寻相错误** |  | **可以复位** |
| **Err.026** |  |
| **Err.027** |  |
| **Err.043** | **位置偏差过大** |  | **可以复位** |
| **Err.044** | **主回路缺相** |  | **可以复位** |
| **Err.045** | **动力线故障** |  | **不可复位** |
| **Err.047** | **电机过载** |  | **可以复位** |
| **Err.049** | **驱动器过热** |  | **可以复位** |

**V6驱动器抄写合川的驱动器。**

**virtual amplifier 虚拟放大器**

**rotary 旋转**

**linear 直线的**

**incremental 增加的，逐步的**

1. 上位机
2. 说明书/操作手册
3. 技术支持的培训
4. 硬件目前来说是能接受，关键是软件代码这块（效率+质量？）
5. 现场问题文档汇总
6. 测试流程的规范，测试人员
7. 研发进度的保证和跟进，这点非研发人员不太确定有信心和能力跟得上。
8. 解决办法，人员的修正-----领导+研发人员
9. 摆烂

直线要能接受6倍冲击

旋转不用考虑那么多冲击

**DD的过载能力高**

**直线的冲击高**

**ST7驱动器-单方向旋转步骤：**

1. **填写参数**

* **P18.01 密码为2**
* **P18.19为0**
* **P18.25为136**
* **P00.01为1**
* **P00.05为2**
* **P01.02设为8800**
* **P03.36设为50。指速度是50rpm。**

1. **断电重启**
2. **将P04.11设为1，DD电机即可运转**
3. **上电自动运转模式：P04.40设为510**
4. **恢复出厂设置：P20.06设置为1.**

**ST7驱动器-单方向旋转步骤：**

1. **检查P18.19为0，P18.25为73，P00.01为1，P00.05为2，P03.36设为1500**
2. **断电重启**
3. **将P04.11设为1，伺服电机即可运转。P04.11设为0，停止运转**

**调试步骤结束。**

**ST7驱动器-往复旋转步骤：**

1. **填写参数**
   * **P18.01 密码为2**
   * **P18.19为0**
   * **P18.25为136**
   * **P00.04设为100**
   * **P00.05为2**
   * **P01.02设为8800**
   * **P08.02和P08.05设为25。指速度是25rpm。**
2. **断电重启**
3. **将P04.11设为1，DD电机即可运转**
4. **上电自动运转模式：P04.40设为510**
5. **恢复出厂设置：P20.06设置为1.**

**P00 Group 基本设定**

**P01 Group 增益调整**

**P02 Group 振动抑制**

**P03 Group 速度转矩控制**

**P04 Group IO端子功能**

**P05 Group 研发参数1**

**P06 Group 研发参数2**

**P07 Group 研发参数3**

**P08 Group 多段运行控制**

**P09 Group 限定参数**

**P18 Group 电机参数**

**P19 Group 电流环参数**

**P20 Group 恢复出厂设置**

**P21 Group 状态监控**

**针对示波器的共域，做大端对齐/小端对齐实验**

**可惜不能调试，要不然就可以查看一下是示波器缓存数组的数据问题，还是通讯协议端的问题。**

**直线：1um的分辨率，2mm的周期 ST7,11位 ST8,12位 2048或4096 2mm即转一圈即对应2000个脉冲数 ps.2048/2000**

**旋转：2000/131072\*360=机械角度 ps.发送2000的脉冲数**

**电气角度，伺服电机4对极，即12个定子，12/3=4。即4组UVW。26214.4**

直线电机：

绝对式读头—脉冲数/2048

AB读头---脉冲数/2000

机械角度是“空间几何角度”，一圈恒等于360°

电角度是根据电流完成一个完整的周期变化来定义的，与电动机绕组的磁极数有关。

**对于伺服电机(servo motor)：**

**要取决于编码器位数，现有伺服电机类型是17位和23位，即**

**脉冲数/131072 \* 360°=机械角度（deg）**

**脉冲数/8388608 \*360°=机械角度（deg）**

**对于直线电机（Linear motor）：**

**取决于读头和电机极距，现有的编码器读头类型分为 通讯式读头和ABZ读头，即**

**通讯式读头：脉冲数/1024 =物理单位（mm）**

**ABZ读头：脉冲数/1000 = 物理单位（mm）**

**极对数 和 极距 涉及到电气角度**

**131072单位是 count/2π**

**2048 单位是 count/2mm**

**注： 通讯式读头 1mm对应1000count**

**1mm对应1024count**

驱动器报警处理：驱动器先停止，上位机可200ms做探针，确认是否出错，没有实时性的高要求。

**专攻通讯连接和示波器**

**可以尝试把所有的代码进行注释，强迫自己联系、理解**

**共用体 EEPROM\_DATA**

**驱动器报错记录和单位换算，这是我要做得事。**

**关于数据类型我并没有搞清楚，int float char double 结构体嵌套**

**Char 占1个字节，字符指针数据类型**

**short 占2个字节，短整型数据**

**Int 占4个字节，基本整型**

**Long 占4个字节，长整型**

**Float 单精度实型**

**Double 双精度实型**

**487 是正常脉冲模式**

**511 是ethercat模式**

**P19.23 设为0，脉冲模式**

**P19.23 设为1，ethercat模式**

**P06.46** 屏蔽多圈编码器错误1

**P19.09** 清除多圈电池错误标志

**0 1 4 5 8 9**

**采集数量**

IsScopeRunning

bool continuousRecordMode = DriveContext.Instance.ControlsContainer.ContainedScopeChart.ContinuousRecordMode;

速度环积分时间和转矩指令滤波时间：

刚性越强，二者值越小，时间就越短，刚性就越强。

速度环积分时间 P01\_02 ：刚性越强，速度环积分时间越小（上位机显示）

转矩指令滤波 P01\_04 ：刚性越强，转矩指令滤波越小（上位机显示）

伺服400W 23位 ID号是33

伺服750W17位多圈 ID号是43

**inc是include的简称,src是source的简称**

**402状态字：报错，使能，失能，位置到达，伺服准备好，使能准备好**

**结束符**

## 2.1 ST7旧上位机通讯协议

**在ST7驱动器IAR工程中，上位机发送数据Tx1\_Buffer，数组赋值完成后，使能 DMA1\_Stream4，然后会自己通过DMA方式发送数据，并没有用到for循环结构逻辑。**

**用到串口，只不过使能发送后，该DMA会自动将数组里面的数据发送出去。**

**突然意识到了，上位机，其他参数都是单个读取，但是示波器的话，请求一次，我可以返回好多个数据，也就是示波器波形的采样点。随时间进行读取，也就是坐标轴。**

**然后离线分析，就是停止其他数据发送，包括实时波形图，提高数据传输频率，让数据更准确！**

**上位机：DMA1\_Stream2----------UART4->DR 外设到存储器 UART4\_RX\_BUF**

**DMA1\_Stream4----------UART4->DR 存储器到外设 Tx1\_Buffer RS485.c**

**Mdobus：DMA1\_Stream5--------USART2->DR**

**DMA失能，则禁用DMA，传输的数据停止。当再次被启用时，DMA将从暂停的地方继续传输数据。 DMA硬件流控制，数据接收的数组已在初始化时定义完成。**

**只要剩余传输数据量不是0，而且DMA是启动状态，那么就会发生数据传输。**

**DMA进行数据传输的必要条件：**

1. **剩余传输数据量大于0；**
2. **DMA通道传输使能；**
3. **通道上DMA数据传输有事件请求；**

**一个是参数与ST8上位机不一致，一个是示波器是否继续用**

**除了底层字典，其他都要变，包括示波器**

**每个采样点，大约2ms.**

**BECKHOFF**

**EK1110**

**CX9020-0112**

**通讯异常：**

**1.干扰**

**2.速度太快，运算过大，时间被挤压，数据无法传输。**

**端子：故障，清除，功能复用，5个输入5个输出**

**泓凯，能否请教一下上位机示波器接收数据处理是什么情况？**

1. **上位机参数获取----按照字典读取**
2. **每种参数的处理，分组？还是重复读取？----由每帧的数据结构决定，可设置不同功能码**
3. **读取数据的频率，多少赫兹---待测试**

**就比如说示波器的 位置给定误差，赋值为错误代码。这个能找到代码准确位置处ma**

**ST7上位机：**

**功能码： 10 20 21 60 61**

**10 读取所有数据 case 1**

**20 修改单个数据 case 2**

**21 恢复出厂设置**

**60 采集通道赋值 case6，case0示波器通道赋值**

**61 示波器采集数据 case6，case1 示波器采集**

**自定义：07 00 xx xx**

**00 待定**

**01 读取所有**

**02 写单个/多个**

**03 示波器**

**04离线分析**

**读取参数：07 00 02 10 00 05 E2 C0**

**写入参数：07 00 06 20 00 15 00 00 03 84 65 6D**

**A5 5A 00 07 20 08 00 58 02 00 00 89 2A**

**示波器功能：07 00 02 61 01 00 5A 50 A5 5A 00 03 61 01 00 21 8E**

**07 00 06 21 00 15 00 00 03 84 65 6D**

**07 00 02 10 00 05 E2 C0**

**07 00 02 10 00 0A E2 C0**

**07 00 02 10 00 0F E2 C0**

**07 00 02 10 00 14 E2 C0**

**07 00 02 10 00 19 E2 C0**

**07 00 08 60 0B 02 03 04 05 06 00 96 02 4B**

**示波器：**

**07 00 08 60 01 02 03 04 05 06 00 96 7D CB**

**07 01 01 60 01 3C 28**

**07 00 02 61 01 00 5A 50**

**07 01 32 61 01 00 80 45 FF 47 00 00 00 00 80 45 FF 47 00 00 00 00 00 00 00 00 00 00 00 00 80 44 FF 47 00 00 00 00 80 44 FF 47 00 00 00 00 00 00 00 00 00 00 00 00 21 19**

**07 01 2A 10 00 09 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 0C 00 00 00 01 00 00 00 02 00 00 00 00 00 00 00 00 00 00 0C 80 00 00 27 10 6E 90**

**07 01 2A 10 00 13 00 00 7F FF 00 00 27 10 00 00 7F FF 00 00 27 10 00 00 0E 10 00 00 27 10 00 00 00 00 00 00 00 00 00 00 00 01 00 98 96 80 6E 33**

**上位机协议：读取数据(33) //这里有多少条读数据帧，则上位机发多少个数据帧**

**A5 5A 00 03 10 05 00 73 55**

**0A 00 76 A5**

**0F 00 75 F5**

**14 00 7F 05**

**19 00 7B 95**

**1E 00 79 A5**

**23 00 69 35**

**28 00 6E 05**

**2D 00 6D 55**

**32 00 65 65**

**37 00 66 35**

**3C 00 61 05**

**41 00 40 55**

**46 00 42 65**

**4B 00 46 F5**

**50 00 4C 05**

**55 00 4F 55**

**5A 00 4A A5**

**5F 00 49 F5**

**64 00 5A C5**

**69 00 5E 55**

**6E 00 5C 65**

**73 00 55 35**

**78 00 52 05**

**7D 00 51 55**

**82 00 10 A5**

**87 00 13 F5**

**8C 00 14 C5**

**91 00 1D 95**

**A5 5A 00 03 10 96 00 1F A5**

**07 01 18 10 0A 00 FF 7F 00 00 FF 7F 00 00 80 96 98 00 F4 01 00 00 84 03 00 00 08 07 00 00 xx xx**

**CRC-USB校验码**

**5A A5 00 03 10 05 00 73 55**

**帧头 控制模式 惯量比 脉冲指令 脉冲串 一圈所需 电子齿轮比**

**5A A5 01 1B 10 05 00 00 00 00 00 01 00 00 00 02 00 00 00 00 00 00 00 80 0C 00 00 FF 7F 00 00 FD 9A**

**A5 5A 00 03 10 0A 00 76 A5**

**帧头 电子齿轮比 分母 偏差阀值 位置环1 速度环1 积分1**

**5A A5 01 1B 10 0A 00 FF 7F 00 00 FF 7F 00 00 80 96 98 00 F4 01 00 00 BC 02 00 00 08 07 00 00 7C 36**

**积分时间1 速度滤波1 转矩指令1 位置2 速度2 积分2**

**5A A5 01 1B 10 0F 00 08 07 00 00 02 00 00 00 2C 01 00 00 90 01 00 00 C8 00 00 00 B8 0B 00 00 07 6C**

**10 读取所有数据 case 1**

**20 修改单个数据 case 2**

**21 恢复出厂设置**

**60 采集通道赋值 case6，case0示波器通道赋值**

**61 示波器采集数据 case6，case1 示波器采集**

**00 待定**

**10 读取所有**

**11 读取单个界面数据**

**20 写单个**

**21 写多个**

**60 示波器-采集通道赋值**

**61 示波器-采集数据**

**62离线分析**

**A5 5A 帧头**

**00 站号**

**07 数据长度**

**20 功能码**

**01 00 字典序列号**

**F4 01 修改的数据**

**00 00 也是修改的数据，数据长度扩展**

**58 E3 校验码**

**4 2 4 2 =12**

**帧头+字典+数据+CRC 数据长度len=字典+数据**

**07 01 04 20 87 00 01 00 00 00 25 49**

**帧头 站号 帧长 功能码 字典 数据 CRC**

**A5 5A 00 07 20 01 00 01 00 00 00 3B 3E**

**A5 5A 01 07 20 87 00 01 00 00 00 25 49 P2001**

**A5 5A 00 07 20 08 00 BC 02 00 00 BF DA 位置环增益**

**07 00 06 20 00 15 00 00 02 58 6C 6D**

**修改控制模式：（1）**

**帧头 站号 帧长 功能码 字典**

**A5 5A 00 07 20 00 00 01 00 00 00 3B 3E**

**5A A5 01 07 20 00 00 01 00 00 00 3B 3E**

**修改惯量比：（500）**

**A5 5A 00 07 20 01 00 F4 01 00 00 58 E3**

**5A A5 01 07 20 01 00 F4 01 00 00 58 E3**

**修改第一段位置：（98888880）**

**A5 5A 00 07 20 45 00 B0 EC E4 05 58 61**

**5A A5 01 07 20 45 00 B0 EC E4 05 58 61**

**修改第一段速度：(800)**

**A5 5A 00 07 20 46 00 20 03 00 00 CF A4**

**5A A5 01 07 20 46 00 20 03 00 00 CF A4**

**修改第一段等待时间：（500）**

**A5 5A 00 07 20 47 00 F4 01 00 00 56 45**

**5A A5 01 07 20 47 00 F4 01 00 00 56 45**

**恢复出厂设置：**

**A5 5A 00 07 21 DB 01 01 00 00 00 04 D5**

**5A A5 01 07 21 DB 01 01 00 00 00 04 D5**

**示波器功能：**

**这6位是示波器参数 采样间隔**

**A5 5A 00 09 60 00 01 04 05 03 02 96 00 B1 92-------（开头）**

**5A A5 01 01 60 BF 68(不变)**

**长**

**A5 5A 00 03 61 01 00 21 8E---------------------（数据）**

**5A A5 01 33 61 01 00 80 03 B8 47 00 01 B8 47 00**

**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

**03 B8 47 80 02 B8 47 00 00 00 00 00 00 00 00 00**

**00 00 00 00 00 00 00 6C D3**

**A5 5A 00 03 61 03 00 20 EE**

**5A A5 01 33 61 03 00 80 03 B8 47 80 01 B8 47 00**

**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80**

**03 B8 47 80 01 B8 47 00 00 00 00 00 00 00 00 00**

**00 00 00 00 00 00 00 12 4C**

**位置给定：0**

**位置反馈：1**

**位置偏差：2**

**编码器Reg：3**

**速度给定：4**

**速度反馈：5**

**速度观测：F**

**速度偏差：6**

**U相电流：7**

**V相电流：8**

**W相电流：9**

**Q轴电流：A**

**D轴电流：B**

**电流给定：C**

**母线电压：D**

**指令输出：E**

**位置给定误差：F**

**TestFlag：10**

**TestFloat：11**

**TestFloat1：11**

**TestFloat2：12**

**示波器功能：位置给定、位置反馈、速度给定、速度反馈、Q轴电流、电流给定**

**A5 5A 00 03 61 01 00 21 8E**

**5A A5 01 33 61 01 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 DF 62(无)**

**5A A5 01 33 61 01 00 00 7E E8 C6 80 CA A1 C7 00 FC CF 3E E2 92 D0 3E 00 26 60 3B 80 46 FD 47 00 FE 86 C6 80 10 82 C7 00 A2 C0 3E B7 8E C1 3E 80 F9 F6 BB C0 F1 03 C8 78 96(有)**

**A5 5A 00 03 61 03 00 20 EE**

**5A A5 01 33 61 03 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 99(无)**

**5A A5 01 33 61 03 00 00 A0 E3 C5 00 0E 4A C7 00 27 AE 3E 86 7B AF 3E 00 31 3C BC E0 35 AB C8 00 80 1C 44 00 54 16 C7 00 C6 98 3E 11 C1 98 3E C0 91 86 BC 80 B7 3D C8 D4 39(有)**

**A5 5A 00 03 61 05 00 23 4E**

**5A A5 01 33 61 05 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6A D5**

**A5 5A 00 03 61 07 00 22 2E**

**A5 5A 00 03 61 09 00 26 4E**

**……**

**A5 5A 00 03 61 FD 02 35 8E**

**A5 5A 00 03 61 FF 02 E0 2F**

**A5 5A 00 03 61 01 03 61 8F**

**……**

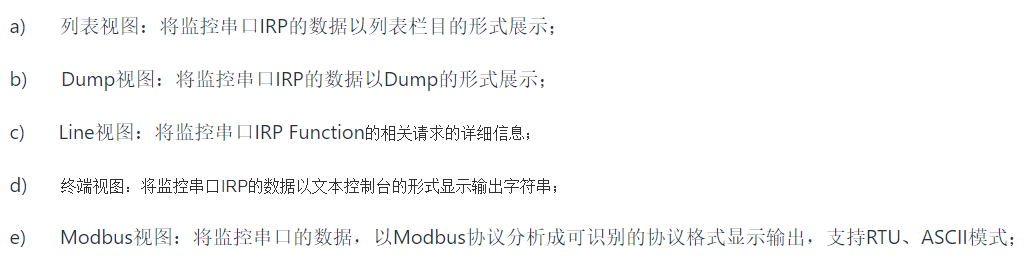
**A5 5A 00 03 61 FF 03 21 EF**

**5A A5 01 33 61 FF 03 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 4B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 D8**

**7.1ms采集一次数据**

**主循环中断，优先级7----isr\_Simpling**

**在这个函数里，示波器采集，间隔10ms采集一次**

****

如何把EEPROM 与ARM芯片连接起来，某个引脚初始化，还有FPGA

**旧波器采集数据：**

**发送与接收时间间隔：1.55ms 8ms**

**接收与下次发送间隔：6.27ms**

**两次发送时间间隔：12.7ms**

**两次接收时间间隔：9.11ms**

**发送时间长度：762.53us**

**接收时间长度：4.92ms**

示波器的连续采集，底层保持不变。当缓存区的值取完后，上位机再发一次采集6个通道的指令，然后赋值到缓存区，实现连续采样。

离线采样只是把连续采样分为一个。

至于描绘的图形。把图形放大后曲线应该还要尽量保持平滑，避免折线。

思路：在示波器的图形构造时，将数据布置的紧密一些减少采样点之间的时间间隔。即同样一块区域，采样点增加。

V6E电机 磁编17位-----ID=30

**canopen协议**

**0 位置给定 PCtrl\_Pos\_float.Ref1\_Pos**

**1 位置反馈 PosGeneartor.PosOut**

**2 位置偏差 PCtrl\_Pos\_float.Fdb1\_Pos**

**3 编码器REG Qep.Reg**

**4速度给定 Speed\_Ctrl\_float.Ref**

**5 速度反馈 Speed\_Ctrl\_float.Fdb**

**6 速度观测**

**7 速度偏差 Speed\_Ctrl\_float.Err**

**8 U相电流 AdcSample.Current\_ADCU**

**9 V相电流 AdcSample.Current\_ADCV**

**10 W相电流 AdcSample.Current\_FdbW**

**11 Q轴电流 AdcSample.Current\_FdbQ**

**12 D轴电流 AdcSample.Current\_FdbD**

**13电流给定 TorqueRef.CurrentQ**

**14母线电压 AdcSample.DcBusVoltage**

**15 指令输出**

**位置给定误差**

**新示波器通道：**

1 位置规划值 **PCtrl\_Pos\_float.Ref1\_Pos**

2 位置当前值 **PosGeneartor.PosOut**

3 目标位置

4 位置误差 **PCtrl\_Pos\_float.Fdb1\_Pos**

5 速度规划值 **Speed\_Ctrl\_float.Ref**

6 速度当前值 **Speed\_Ctrl\_float.Fdb**

7 速度观测器实际速度

8 速度误差 **Speed\_Ctrl\_float.Err**

9 电流给定 TorqueRef.CurrentQ

10 电流当前值 **AdcSample.Current\_FdbQ**

11 电流误差

12 D轴电流规划值

13 D轴电流实际值 **AdcSample.Current\_FdbD**

14 U相电流 **AdcSample.Current\_ADCU**

15 V相电流 **AdcSample.Current\_ADCV**

16 W相电流 **AdcSample.Current\_FdbW**

17 Z相位检测偏差

18 力矩规划值

19 力传感器当前值

20 力规划值

21 力控误差

22 Alpha轴电压实际值

23 Alpha轴电压规划值

24 Beta轴电压实际值

25 Beta轴电压规划值

26 编码器REG Qep.Reg

27 脉冲输入计数 Pulse\_float.Reg0

28 母线电压 AdcSample.DcBusVoltage

**新示波器通道计算方式：**

1 位置规划值 \*1.0f

2 位置当前值 \*1.0f

3 目标位置

4 位置误差 \*1.0f

5 速度规划值 \*3000.0f 或 \*1000f **//伺服电机乘3000，直线电机乘1000**

6 速度当前值 \*3000.0f 或 \*1000f  **//伺服电机的速度单位是rpm/s**

7 速度观测器实际速度

8 速度误差 \*3000.0f 或 \*1000f  **//直线电机的速度单位是mm/s**

9 电流给定

10 电流当前值 \*参数f -----该参数见下文附图（P18.18）

11 电流误差

12 D轴电流规划值

13 D轴电流实际值 \*参数f -----该参数见下文附图（P18.18）

14 U相电流 \*参数f -----该参数见下文附图（P18.18）

15 V相电流 \*参数f -----该参数见下文附图（P18.18）

16 W相电流 \*参数f -----该参数见下文附图（P18.18）

17 Z相位检测偏差

18 力矩规划值

19 力传感器当前值

20 力规划值

21 力控误差

22 Alpha轴电压实际值

23 Alpha轴电压规划值

24 Beta轴电压实际值

25 Beta轴电压规划值

26 编码器REG \*1.0f

27 脉冲输入计数

28 母线电压 \*0.125f



IQmax

Cmdconmmunication.cs

Recorderdate.cs

Recorder\_new.cs

Chardataparser.cs

**5种模式：位置模式、速度模式、脉冲控制模式、电流环标定模式、多段运行模式**

* **位置模式：P00.01为0，P00.05为2，P08.00为2**
* **速度模式：P00.01为1，P00.05为2**
* **脉冲控制模式：P00.05为0**
* **多段运行模式：P00.01为0，P00.05为2。**
* **电流环标定模式：P18.19为3或4。为3是进行电流环标定，为4是按照标定参数驱动电机，都是属于电流环标定模式下。**

**多段运行模式和位置模式共用前两段，若修改前两段参数，二者是等同的。大于2段，则是多段运行。关于模式设定上有bug，后续会将这块底层完善。**

**关于电机ID号：**

**若P18.19为1，则读取ID号是在P19.25。为1是自动读取电机ID，不可修改，只读**

**若P18.19为0，则读取ID号是在P18.25。为0是在P18.25输入ID号，可读可写**

运行的前提是电机ID号正常，才可进行之后的运动控制，若ID不正确，则运动错误会报错。

P18.19的功能：

0：正常模式，电机ID由参数表读取，需要设定电机P18.25电机ID，可正常运行

1：正常模式，电机ID由编码器读取，驱动器自动读取编码器电机 ID，可正常运行，P19.25 可查看读取到的电机 ID；

3：标定模式，电机ID由P18和P19组参数决定，设定完成之后重启，进入电流环标定模式；

4：正常模式，电机ID由P18和P19组参数读取，在模式3下标定电流环参数后，该参数设定为4，重新上电，由模式3标定好的P18和P19参数运行电机。

先判断P18.19的值是否为3或4，若是，则上位机的工作模式跳转到4-电流环标定模式；

若不是，则判断P00.01 P00.05 P08.00 这三个参数确定驱动器工作在 位置，速度，脉冲，多段运行这四个模式中 的哪一个。

对于“连接ID”界面，ID号的读取，先判断P18.19，若为0，则读取P18.25的值进行显示；若为1，则读取P19.25的值进行显示。

电流 \* P18.18组设定的额定电流

烧录固件，连接出现卡死现象

1. 字节转换；
2. 通道赋值的代码分类；

00 01 04 05 03 02

00 01 04 05 0C 0A

很多参数的倍数或系数，以标1值为基准。也是相对单位制的一种。

Flash属于eeprom，电可擦除的ROM。Flash是按扇区操作，eeprom是按字节操作。

## 2.2 ST8通讯协议

**ST8上位机通讯协议：**

**请求设置串口内部接收缓冲区的大小---4096**

**请求将串口配置为在发生一组指定的等待事件中的任何一个事件之后通知客户端**

**请求设置串口设备上的波特率---921600**

**请求设置串口RTS（发送请求）---Set RTS**

**请求设置串口DTR（数据终端准备就绪）---Set DTR**

**请求设置串口线路控制寄存器（LCR）---字长8**

**请求设置串口驱动程序用于握手流控制的特殊字符**

**请求设置串口握手流控制的配置**

**IRP---IO请求包**

**40 6C 60 40 1E 35**

**40 64 60 40 F4 60**

**40 78 60 40 1F 35**

**位置规划值：40 62 60**

**位置误差趋势图：40 F4 60**

**速度当前值：40 6C 60**

**速度误差趋势图：40 1E 35**

**电流误差趋势图：40 1F 35**

**电流当前趋势图：40 78 60**

**ADC1当前值趋势图：40 8B 35**

**ADC2当前值趋势图：40 8C 35**

**Alpha轴电压实际值：40 F7 35**

**Alpha轴电流当前值：40 47 36**

**Beta轴电压实际值：40 F8 35**

**Beta轴电压规划值：40 FA 35**

**Beta轴电流当前值：40 48 36**

**D轴电流实际值：40 F6 35**

**D轴电流规划值：40 F3 35**

**U相电压趋势图：40 7B 35 01**

**U相电流趋势图：40 7A 35 01**

**V相电压趋势图：40 7B 35 02**

**W相电压趋势图：40 7B 35 03**

**W相电流趋势图：40 7A 35 03**

**Z相位检测偏差：40 68 35**

**估计负载力等效的转矩电流趋势图：40 8E 35**

**位置当前值：40 64 60**

**位置当前原始值：40 3B 36**

**位置规划值：40 62 60**

**位置误差：40 F4 60**

**内环位置误差：40 73 35**

**内环当前位置：40 72 35**

**内环当前速度：40 74 35**

**内环速度误差：40 75 35**

**内部位置规划值：40 FC 60**

**力传感器当前值：40 0D 36**

**力控误差：40 8A 35**

**力矩规划值：40 74 60**

**力规划值：40 E9 35**

**当前模式：40 61 60**

**控制字：40 40 60**

**母线电压：40 79 60**

**电机热量累计值：40 A5 35**

**电机转子惯量：40 16 35**

**电流当前值：40 78 60**

**电流误差趋势图：40 1F 35**

**脉冲输入计数：40 F0 35**

**辨识状态：40 08 30**

**速度当前值：40 6C 60**

**速度观测器实际速度：40 04 36**

**速度规划值：40 6B 60**

**速度误差趋势图：40 1E 35**

**驱动器热量累计值：40 ED 35**

**实时波形图：**

**40 6C 60 00 00 00 00 00**

**40 1E 35 00 00 00 00 00**

**40 64 60 00 00 00 00 00**

**40 F4 60 00 00 00 00 00**

**40 78 60 00 00 00 00 00**

**40 1F 35 00 00 00 00 00**

**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 19**

**40 41 36 00 00 00 00 00**

**40 42 36 00 00 00 00 00**

**06 0A 0A 22 72 82 02 02 02 FC**

**06 0E 02 02 F2 02 02 03 02 FE**

**离线分析：**

**(1)位置当前值，位置规划值**

**FD 01 00 58 00 00 00 00 00 00 00 00**

**FD 01 00 59 00 00 00 00 00 00 00 00**

**FD 01 00 5A 00 00 00 00 00 00 00 00**

**(2)位置当前值**

**FD 01 00 3A 00 00 00 00 00 00 00 00**

**FD 01 00 3B 00 00 00 00 00 00 00 00**

**FD 01 00 3C 00 00 00 00 00 00 00 00**

**FD 01 00 3D 00 00 00 00 00 00 00 00**

**(3)速度规划值，速度当前值，速度误差，位置规划值，位置当前值，目标位置**

**FD 05 00 58 00 00 00 00 00 00 00 00**

**FD 05 00 59 00 00 00 00 00 00 00 00**

**FD 05 00 5A 00 00 00 00 00 00 00 00**

**FD 03 00 3D**

**FD 03 00 3E**

**FD 06 00 59**

**FD 05 00 5A 00 38 B0 51 EF FF B0 51 EF FF DC 52**

**EF FF DC 52 EF FF DC 52 EF FF DC 52 EF FF DC 52**

**EF FF 10 5B EF FF 10 5B EF FF 10 5B EF FF 10 5B**

**EF FF 10 5B EF FF EB 50 EF FF E8 50 EF FF 00 00**

**C6 FD 05 00 5B 00 38 E8 50 EF FF E8 50 EF FF E8**

**50 EF FF 10 5B EF FF 10 5B EF FF 10 5B EF FF 10**

**5B EF FF 10 5B EF FF BC 4F EF FF BC 4F EF FF BC**

**4F EF FF BC 4F EF FF BC 4F EF FF F0 57 EF FF 00**

**00 A5**

**上位机：**

**07 xx xx**

**Hex 是由对应机器语言码或常量数据的16进制编码数字组成，由intel创建**

**数据终端准备好DTR（Data Terminal Ready）；请求发送RTS（Request To Send）。**

**清除发送CTS（Clear To Send）；数据准备好DSR（Data Set Ready）。**

**ANSI编码：这些使用了2个字节来表示一个字符的各种延伸编码方式，称为ANSI编码。在简体中文系统下，ANSI编码代表GB2312，在日文操作系统下，ANSI编码代表JIS编码。不同ANSI编码之间互不兼容。**

**CRC循环冗余校验，有常用21个标准CRC参数模型。**

**流控制 可以控制数据传输的进程，防止数据的丢失。如台式与单片机之间的通讯，若接收端缓冲区已满，继续发送来的数据就会丢失，流控的概念，当接收端数据处理速度不够时，就发出“不再接收”的信号，发送端就停止发送，直到收到“可以继续发送”的信号再发送数据。**

1. **示波器参数**
2. **提高采样频率**

**云闪付**

**本地宝-深圳消费补贴**

**缕缕思路：**

**现在已经能够确定ST7上位机协议，ST8示波器协议，但ST8上位机其他按钮协议还未知；**

**还有两个地方需要理解，不过可以直接在工程中做修改。**

**关于示波器采集效率问题，参数修改影响不大；RS232通信速率提高，921600，不知可不可以**

1. **ST7字典有效问题。**
2. **上位机验证问题。自己可以通过串口助手发送，但实际验证不好确认**

**ST8通讯时间：（130字节）**

**发送与接收间隔：903us**

**两次数据间隔：4.7ms**

**发送数据时间长度：774.5us**

**接收数据时间长度：702us**

**ST7通讯时间：（32字节）**

**发送与接收间隔：4.1ms**

**两次数据间隔：2.0ms**

**发送数据时间长度：2.8ms**

**接收数据时间长度：753.92**

**误码率：**

**7---4597（13791）---0.15%（舍）**

**9---19405（58216）---0.046%**

**24---60787（182363）---0.04%**

**3---3576（10701）---0.08%**

**平均误码率：0.055%**

**&rxbuffer和rxbuffer区别在哪，为什么会报错**

**两点不规范：**

1. **共用体union和结构体的结合组成的共域顺序不对，导致代码理解的过程曲折**
2. **帧长度的制定，不标准**
3. **CRC校验也不是标准格式，无法通过标准格式校验。**

**目前上位机问题：**

**1.当电机运转速度很快，如3000rpm时，示波器采样无法正常运行**

**A5 5A 00 07 20 08 00 BC 02 00 00 BF DA**

p存放的是地址，而\*p是让程序去那个地址取出数据。

&等于\*\*p；

每个变量都有自己的存储空间，那么指针变量也有自己的存储空间。

Int p; //当然就是直接的赋值了 p=1;

Int \*p; //定义指针变量，这个特殊性就决定了会有存储\*p的空间地址，即&p取得是指针变量p的地址。 \*p=&a[1]; 想要赋值给指针

**重新烧写RAM的hex文件，驱动器内部参数不变**

**示波器采样期间，停止采样后，驱动器也停止发送数据 √**

**Bootloader：是嵌入式系统上电后执行的第一段代码，然后初始化开发板上主要硬件（时钟，内存，硬盘），并加载到内存；然后跳转到用户的应用程序（即操作系统）并运行。**

**Boot：**

**Loader：（1）从硬盘指定的地址加载kernel到内存指定的地址；（2）跳转到内存kernel所在的地址，执行。**

1. **让ST8的上位机通过ST7驱动器显示波形图**
2. **再仔细分析一下数据间隔及长度**
3. **统计错误数据帧的个数----bingo**
4. **修改波特率---------bingo**
5. **USB握手掉线---参考高创上位机**
6. **底层波特率字典的赋值操作**

**还有两个问题解决：**

1. **波形图实时采集的协议，需要增加。**
2. **示波器采集错误时，显示---误码率0.055%**

**反复提到的两个问题：**

1. **传输速率，提高效率**
2. **降低误码率**
3. **丰富模块**
4. **USB掉线情况，错误数据返回**
5. **波特率自适应调整。**

**解决方法：**

**多交流**

**进度把控**

**版本1：（正常）**

**发送与接收时间间隔：1.81ms**

**两次发送时间间隔：7.11ms**

**两次接收时间间隔：5.03ms**

**发送时间长度：771.19us**

**接收时间长度：2.84ms**

**版本1对比：（正常-示波器62.5us的数据采集）**

**发送与接收时间间隔：1.94ms**

**两次发送时间间隔：9.34ms**

**两次接收时间间隔：5.50ms**

**发送时间长度：745.2us**

**接收时间长度：4.93ms**

**版本2：（屏蔽modbus）**

**发送与接收时间间隔：3.43ms**

**两次发送时间间隔：8.17ms**

**两次接收时间间隔：7.13ms**

**发送时间长度：762.53us**

**接收时间长度：2.81ms**

**版本3：（TIM2---50us）**

**发送与接收时间间隔：负**

**两次发送时间间隔：4.64ms**

**两次接收时间间隔：2.56ms**

**发送时间长度：762.53us**

**接收时间长度：2.85ms**

**版本4：（波特率460800）**

**发送与接收时间间隔：2.24ms**

**两次发送时间间隔：4.78ms**

**两次接收时间间隔：5.28ms**

**发送时间长度：184.09us**

**接收时间长度：711.98us**

**版本5：（波特率921600）**

**发送与接收时间间隔：1.93ms**

**两次发送时间间隔：5.08ms**

**两次接收时间间隔：4.64ms**

**发送时间长度：103.92us**

**接收时间长度：351.01us**

**ST8：（921600）**

**发送与接收时间间隔：756.86us**

**两次发送时间间隔：5.68ms**

**两次接收时间间隔：5.29ms**

**发送时间长度：796.18us**

**接收时间长度：703.09us**

**目前有这样的问题：**

1. **修改为 TIM2—50us，USB掉线。正常是500us。**

**Hx58110**

需重写fputc()这个c标准库函数（ printf()在c标准库函数实质是一个宏，实际是调用fputc()函数）

**6个通道实时显示，能想到的是，当获取数量为3-FF时，再次赋值通道，然后返回数据**

**就看赋值通道---到数据返回时间是多久，会不会产生较大间隔误差。**

**协议还有个问题，华南理工他们做的上位机部分与原先协议一致，但原先协议有几处配置不标准；**

**跳脱现有代码框架，构思如何提高执行效率，设想架构。**

**我今天又又又犯了个错，电机编码器类型，旋转的好说。但是！直线的，要选12，编码器分辨率是2^(12)\*20/2=40960**

**DMA测试总结：**

**（1）根据测试结果，DMA应该是传输过程不影响主循环的**

**（2）\_\_HAL\_DMA\_GET\_COUNTER(&UART1TxDMA\_Handler);//通道停止时剩余数据的数量。**

**无论是广谱开放参数，还是更改协议，要放在一起，同一个版本下**

**旧版本可以不支持**

**高创驱动器：**

1. **看它的上位机协议是怎样的形式，是否要像ST8隐藏每个按钮对应的协议**
2. **示波器采集的协议格式**

**编码器错误**

**高创运动200mm 200mm/s**

## 2.3 高创驱动器USB协议

**41 43 54 49 56 45 3C 42 43 3E 0D**

**41 43 54 49 56 45 3C 42 43 3E 0D 0A 30 3C 33 30 3E 0D 0A 2D 2D 3E**

**4D 4F 54 4F 52 4E 41 4D 45 3C 42 32 3E 0D**

**4D 4F 54 4F 52 4E 41 4D 45 3C 42 32 3E 0D 0A 22 53 57 4C 2D 44 53 2D 53 32 3C 38 45 3E 0D 0A 2D 2D 3E**

**4D 49 50 45 41 4B 3C 42 37 3E 0D**

**4D 49 50 45 41 4B 3C 42 37 3E 0D 0A 32 32 2E 30 30 30 20 5B 41 5D 3C 33 42 3E 0D 0A 2D 2D 3E**

**4F 50 4D 4F 44 45 3C 43 34 3E 0D**

**4F 50 4D 4F 44 45 3C 43 34 3E 0D 0A 38 3C 33 38 3E 0D 0A 2D 2D 3E**

**5C 30 0D**

**2D 2D 3E**

**5C 30 3C 38 43 3E 0D**

**5C 30 3C 2D 2D 3E**

**56 3C 35 36 3E 0D**

**56 3C 35 36 3E 0D 0A 30 2E 30 30 30 20 5B 6D 6D 2F 73 5D 3C 34 32 3E 0D 0A 2D 2D 3E**

**64 72 69 76 65 6E 61 6D 65 3C 42 42 3E 0D**

**64 72 69 76 65 6E 61 6D 65 3C 42 42 3E 0D 0A 22 3C 32 32 3E 0D 0A 2D 2D 3E**

**61 64 64 72 3C 39 42 3E 0D**

**61 64 64 72 3C 39 42 3E 0D 0A 30 30 3C 36 30 3E 0D 0A 2D 2D 3E**

**69 6E 66 6F 3C 41 43 3E 0D**

## 2.4 辨识功能

需要请教的问题：

示波器掉线问题是如何处理解决的

发送数据协议，以及缓存区域在哪里

运动轨迹规划：环路选择

**切换至速度环，若串口打开则对模式缓存区进行赋值**

**数据采集模式切换：mode=0，表示STQ采集，mode=1表示sulab采集**

**辨识计算（ IdentifyCal ）:**

1. **等待模型辨识结果；**
2. **从文件中读取辨识数据---该数据文件推测是由MATLAB计算得到**
3. **从数据栏中读取，判断是单项数据还是多项数据：**
4. **单项数据，判断是哪组数据，校验后放置txtdata文件中**
5. **两组数据，判断是否都是sin函数，否则提醒要重新进行计算**

环路切换：

1. 速度环
2. 位置环
3. 相对位置
4. 绝对位置
5. 速度模式
6. 位置模式
7. 默认环路

模型辨识：一种是加载文件里面的数据进行辨识，一种是默认本次运动作为辨识数据。

辨识计算：

1. 默认是从文件中读取
2. 数据栏读取：a.单项数据；b.多项数据（即两组参数）

一般的辨识法，是利用返回的数据进行最小二乘辨识，加速度要滤掉

同时要写入模型参数

**P07组排序**

**数据类型分类**

**上位机--->下位机：**

**修改参数的数据：int型**

**示波器的参数地址和通道：char型**

**下位机--->上位机：**

**单个或多个参数传输：int型数据 ServoDataArea**

**示波器通道数据传输：float型数据**

**备注：若传递参数为float型，可通过移位或乘个倍数赋值给int型，记得文档说明该倍数**

**char和uint8\_t都是8位，一般可以理解为没什么区别，用uint8\_t更加健壮。**

**（\*a）.b等价于a->b**

**一般情况下用“.”，只需要声明一个结构体，就可以用结构体名.加域名引用。**

**用“->”，则要声明一个结构体的指针，要手动开辟一个结构体的内存，把返回的指针给声明的结构体指针才能正使用“->”。**

**速度环的时间：125us，也就是两次电流环会有一次速度环中断。**

**数据传输3种：**

**1.P07组参数，正常读写**

**2.长序列，P07.10和P07.18，伪随机序列和转矩波动补偿序列，需要04功能码读写数据。**

**3.长数据帧：**

**在三角波辨识过程，需要400个点（电流给定和速度反馈）；**

**在转矩波动补偿过程，需要4000个点（电流给定，速度反馈，位置反馈，编码器值）**

**伪随机辨识过程，需要8192个点（速度反馈）**

**回限位启用标志位：第一位 转矩波动补偿**

**规划器启用标志位：第三位 三角波，伪随机**

**0：既不运动，也不采集；**

**1：采集和运动**

**2：只运动**

**Char string float byte int 之间关系和转化**

**问题：**

1. **理解公式，伯德图，根轨迹**
2. **我需要做得工作是什么？**
3. **最小二乘法，matlab建立模型**
4. **If a b**
5. **在底层如何插入**
6. **如何采集iq**
7. **如何实现交互；**
8. **刚性等级又是如何设计；**

**伺服系统衡量着其对变化的输入指令的响应速度，决定了它能够以多快的速度响应位置、速度或扭矩方面的参数变化。**

**对于伺服的控制回路，内侧控制环的响应带宽一般会是外侧控制环的5-10倍。**

**惯量比指的是负载惯量与电机转子惯量的比值。**

**直线：0.2-0.5 m**

**旋转：0.02**

**Speed\_Ctrl\_float.Ref = 1 则为3000rpm**

**3000rpm =>50r/s =>100pi rad/s**

**电机ID 33 空载 10000倍 0.02距离 3m/s^2 -690**

**负载 100 倍 0.02距离 1m/s^2 -22**

**负载 100 倍 0.01距离 1m/s^2 -8(转矩波动电流)**

**966-1247**

**3504 4208 3910**

**3473 3450 3865**

**3468 3457 3834**

**运行速度推荐：**

**伺服电机---50~100rpm**

**直线电机---10 rpm**

**DD电机---10rpm**

1. **扫频，电机无寻相**
2. **上电使能，IO功能**

## 2.5 惯量辨识

1.遗留问题

（1）生效逻辑，每次重新上电要修改刚性等级时，需要重新进行惯量辨识

（2）惯量等级低时无法启动，速度增益为0；等级高时速度环发散，电机会剧烈振动。这两种极端情况是否要处理？

（3）默认增益的确认。

Speed\_Ctrl\_float.MotorIntertia //电机惯量基值 0.00007

Speed\_Ctrl\_float.RatedCurrentAmplitude //电机额定电流幅值

Speed\_Ctrl\_float.MotorIntertiaRate //惯量比

Speed\_Ctrl\_float.TorqueUnit //电机转矩系数

rw\_u 立即生效

rw\_S 立即生效

Rwru 不会显示，也不起作用，断电后生效

5 131072

10 240 20 7900 0 35649

15 700 70 2800 0 5710

20 2020 210 900 0 1523

23 3680 390 500 0

25 4900 520 400 0

Speed\_Ctrl\_float.MotorIntertia 0.000007

Pdf比pi好

0.947 0.038

丝杆：20kg 822%

10kg 545

一键辨识 默认增益 ：速度环增益342

存在两个问题：

1. 为什么一键辨识的结果，惯量比为0
2. 而惯量辨识生成惯量比很大

会议总结：

问题存在：

1. 电流环频率
2. 内部指令的缺陷，加速没问题，减速段会有有两个跳点
3. 减速阶段会调整两次，导致整定时间变长

未来规划：

1. 用ST6驱动器
2. 换新的读头，高性能的读头

测试规范：

测试要求，步骤，数据

验证一键辨识的惯量辨识和最后控制器设计，推荐的参数对比

0420 ST7新固件

1. 恢复出厂设置，ID号设置33，重启。惯量辨识，修改模式。**直接使能，飞车**。
2. 恢复出厂设置，ID号设置33，重启。惯量辨识，修改模式。**重新上电，使能正常**

**存在问题：**

1. **正弦波进行时，会震荡，进行惯量辨识 惯量比很大。**

**一个是速度**

**一个是路程**

**关于上位机，需要蔡泽凯确认的问题：**

**1.获取电机信息，identificationStep1.xaml.cs和TuningParams.cs重复的函数，能否合并。**

**结束符**

# 第三章 Kinco上位机

说明手册

**14 G070 MT4414T（工程兼容性—P39）**

**上位机 2051地址，实际是0802地址，差1**

**01 03 08 02 00 02 67 AB**

**1 接收 485+ A**

**5 GND**

**6 发送 08 02 00 02 67 A8**

**现在屏幕这一块的问题：**

**1.显示有异常**

**2.发送数据 是05寄存器数量，应该是1-2个，没找到设置值在哪**

**3.修改值有问题，也就是用“数值显示元件”修改寄存器内部值？**

**4.存在一定的延时性？明明已经修改参数值，却还是上次的值进行发送。**

**地址应该是4X，如果是3X则不会发送数据**

**连接失败的原因，是没有数据传输所以显示，连接没有响应，并不是线路接错。**

**现在还是没有搞清楚，发送的寄存器数量，这个是怎样判断的。**

**搞了半天，我对多寄存器读写功能理解的不全，修改了驱动器代码**

**连接失败的原因，可能是驱动器那边死机了**

**11个卡死**

**排查问题思路：**

**1.了解整个程序框架（次要）；**

**2.准确定位问题点，比如这次通讯不能超过10个，要准确定位到这个问题点**

**3.了解程序框架也很重要，但希望能尽快解决问题。**

**程序理解：**

**1.状态机（TIM2），一直在不断的产生中断，进行扫描---优先级为9**

**2.扫描内容包括 Modbus通信，通信扫描，错误扫描，参数加载**

**3.通信扫描 DMAcount = DMA\_GetCurrDataCounter(DMA1\_Stream5);**

**貌似是通过这行代码，实现数据获取。**

**4. len值代表？**

**主观能动性**

**好吧，问题找到了，但为什么？**

**ADC采样，这个点很重要，ADC模块**

**三个问题：**

**1.一圈所需脉冲数----这个可能不是即时生效，必须断电重启后才能有效。**

**2.HMI的数据范围---首先驱动器只支持8位数据，其次HMI数据处理 高8位和低8位相反。**

**3.理解代码两个点，buff和for循环数据填充---buff也就是在获取地址后，可增加的容错性，for循环是依次赋值连续地址，但实际HMI屏只获取前2个字节数据，后面舍弃？**

**4.先试试ST7波特率效果---可实现**

**5.试试ST8驱动器效果---果然一个字典可以，多个元件就不行**

**6.位置的返回值，要不要除以1024**

**任务：**

1. **实时性标注，冲突处标注**
2. **ST7和ST8对比**

**Rxbuff2[20]，对应10个元件**

**Rxbuff2[22]，对应12个元件**

**Rxbuff2[25]，对应13个元件**

**滤波器原理，加恒平均**

**多摩川协议**

**480 0E10**

# 第四章 ST6&ST7上位机

我是傻逼工程

## 4.1 note

**C#是微软公司开发出来的面向对象编程语言，继承了C/C++的许多方面的特点，同时去掉了一些C/C++的复杂特性。**

**S曲线模块 微分+微分**

**跟着小麦大叔走，上位机，S曲线，其他技能点**

**mainwindow.h文件是我们自己创建的，引入时用" "双引号括起来；**

**QApplication 是 Qt 提供给我们的，引入时用<>括起来。**

**Qt 中所有可视化的元素都称为控件**

**QT安装及初步使用结束，等待后续任务安排。**

Widget 小部件，小工具

Geometry 几何，也就是位置及大小，包括框位置和框大小

Sizeincrement 控制尺寸变化时的步进尺寸

Palette 调色板

Cursor 光标

mouse tracking 鼠标跟踪（速度）

tablet tracking 平板电脑跟踪

window title 窗口标题

window opacity 窗口透明度

tool tip duration 工具提示时间

layoutDirection 布局的方向

autofillbackground 自动填充背景颜色

style sheet 样式表

locale 地点（事情发生的）

window file path 窗口文件路径

[inputMethodHints](javascript:;)  控件的输入模式的提示

window modality 窗口模式

mainwindow 主窗口

frameshadow 框架阴影值

plain 朴素，单色的

main() 函数中就必须调用 **exec() 函数**，它的功能就是使程序能够**持续不断地接收各种事件。**

在 Qt 中，对信号做出的响应动作就称为槽。

Alignment 表示对其方式

Pixmap 表示Qlabel控件内的显示图片

Indent 保存QLabel控件内文本的缩进量

Margin 保存QLabel控件中内容与边框之间的距离，默认为0

Wordwrap 保存QLabel控件内文本的换行

QLabel控件中的一些函数

|  |  |
| --- | --- |
| **成员方法** | **功 能** |
| hide() | 隐藏文本框。 |
| clear() | 清空 QLabel 控件内所有显示的内容。 |
| setToolTip(QString) | 设置信息提示，当用户的鼠标放在QLabel 文本框上时会自动跳出文字。 |
| setToolTipDuration(int) | 设置提示信息出现的时间，单位是毫秒。 |
| setStyleSheet(QString) | 设置 QLabel 文本框的样式。 |
| setGeometry(int x, int y, int w, int h) | 设置 QLabel 文本框的位置 (x, y) 以及尺寸 (w, h)。 |

Icon 保存按钮左侧要显示的图标

Iconsize 保存按钮左侧图标的尺寸

Flat 初始状态下，按钮是否显示边框。默认属性是flase，表示按钮带有边框。

AutoDefault 当用户按下Enter回车键时，是否触发点击按钮的事件。

数字IO，信号电平和每个事件的高低电平进行与，若电平相同则亮绿灯，无论高电平还是低电平

## 4.2 更新计划

1. 在连接界面增加版本信息

2. 数字IO界面，可以变为实时显示，将两个参数值放入不断循环监控序列中

3. 【全部参数】界面的参数描述

4. 电机ID号有时会显示不准确；

5. 上位机的记忆功能，参数保存至C盘；

6. FFT和一键辨识，扫频功能；

7. 各个参数界面的更新；

# 第五章 C#语言

实时

namespace 里面包含了一系列的类

*Console.WriteLine();* 是一个定义在 *System* 命名空间中的 *Console* 类的一个方法。该语句会在屏幕上显示消息 "Hello World"。

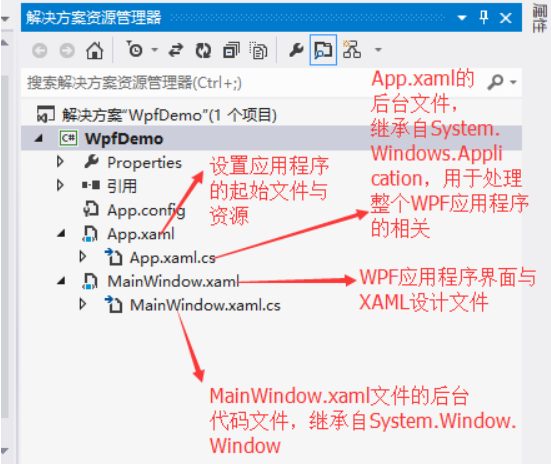
***console.ReadKey();*** 是针对用户，使得程序会等待一个按键动作，防止程序在屏幕运行完，快速关闭

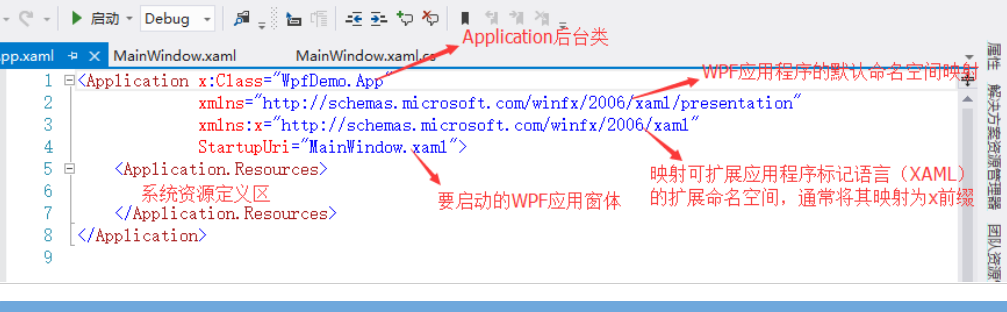
***Console.ReadLine();*** 用于接收来自用户的输入，并把它存储到一个变量中。如：num=Convert.ToInt32(Console.ReadLine());

WPF基于矢量绘图，因此产生的图形界面能够支持各种分辨率的显示。

WPF引入一种新的XAML语言来开发界面，使用XAML语言将界面开发以及后台逻辑开发很好的分开，降低了前后台开发的耦合度。

Utils常用工具





Xaml中的命名空间中的“网址”，是遵循xaml解析器标准的命名规则，不是真正的网址（在浏览器中无法打开）。

看连接部分的if (!GRID\_Main.Dispatcher.CheckAccess())

{

GRID\_Main.Dispatcher.Invoke(delegate

{

afterConnectionActions();

});

}

获取文件的路径：

xmlFilePath = FilePathUtils.getPathOfAllUsers() + "CommandsHD.xml";

**Try catch 用法：**

**先执行try中的代码，如果抛出异常会有catch捕获并执行，没有发生异常catch捕获也会执行，但是代码结果会被忽略掉。**

**Try 语句能测试代码块中的错误**

**Catch 语句能允许处理错误**

**Throw 语句能允许创建自定义错误（抛出错误）**

**Finally 语句能在try和catch之后，执行代码，无论结果如何**

byte[] bytes = BitConverter.GetBytes(s);

Array.Reverse(bytes);  //bytes转换为倒序（反转），可实现大端小端的转换

**array.GetLength(0)得到的是数组array第一维的长度为3**

**array[0].GetLength(0)得到的是数组array第二维的长度为5**

定义变量的同时赋予数值和先定义后赋值有什么区别:

定义时赋值和定义后赋值，区别是文件的大小不同，定义后赋值会稍微大一点。其他一致。

定义时赋值和定义时不赋值，文件大小基本保持一致。

定义时赋值和定义后赋值的 .text相同，而定义后不赋值，执行的语句会少一条，这个会在 .text体现。

定义直接赋值和定义后马上赋值在这里还是一样的。而不赋值的时候，少了一条movl $10, -4(%rbp)，这句话的含义就是这里给int 开辟栈空间并赋值10.

## 5.1 C#语言

特点：面向对象，面向组件

.Net 框架是一个平台，该框架能编写Windows，web应用程序，Web服务

方法首字母通常是大写，Java是小写

程序的执行是从main开始的

类名称可以和文件名不一样

C#的基本语法：

**Using 关键字：**

在任何C#程序中，第一条语句都是using system；

Using 关键字指包含的命名空间，一个程序可包含多个using语句。

**Class 关键字：**

用于声明一个类

**C#中的注释：**

多行注释-----/\* \*/

单行注释-----//

**成员变量**---类似于变量

**成员函数（方法）**-----类似于函数

**C#关键字**-----不可使用，系统默认

**C#数据类型：**

**值类型（value types）：**源自**System.ValueTypes**中，包含int，char，float

**引用类型（reference types）：**指对变量的引用。引用类型有object、dynamic、string。

1. 当一个值类型转换为对象类型时，则被称为**装箱**；而当一个对象类型转换为值类型时，则被称为**拆箱**。如object obj; obj=100; //装箱
2. 动态类型-dynamic。存储任何类型的值在动态数据类型变量中，类型检查是在运行时发生的。

声明动态类型的语法：dynamic <variable\_name> = value; Dynamic d=20;

注：对象类型变量的类型检查是在**编译时**发生的；

动态类型变量的类型检查是在**运行时**发生的；

指针类型（pointer types）

注意：值类型初始有默认值，引用类型初始没有默认值，需要自己定义；

**C#类型转换**

隐式类型转换：默认以安全方式转换，不会导致数据丢失。如从小的整数类型转换为大的整数类型，从派生类转换为基类。

显示类型转换：强制类型转换，会造成数据丢失。比如double 1234.56转为int

（1）int a=(int)b; (2)int a = Convert.ToInt32(b);

**C#变量：**

整数类型：sbyte、byte、short、ushort、int、uint

浮点型：float、double

十进制类型

布尔类型：

空类型：可为空值的数据类型

**C#常量：**

常量是使用const关键字定义，如：

const <data\_type> <constant\_name> = value; ------const int a=100;

**C#运算符：**

Console.WriteLine("a==b:{0}", a == b); //{0}会切换第一个数a，{1}会切换第二个数b.

Console.WriteLine("我是一个自定义方法a:[0],b:[1]:",a,b);

foreach (int i in 30) 是for的加强版

**C#的定义方法**

定义一个方法时，从根本上说是在声明它的结构的元素。在C#中，定义方法的语法如下：<Access Specifier> <Return Type> <Method Name>(Parameter List) {Method Body}

（1）Access Specifier：访问修饰符，这个决定了变量或方法对于另一个类的可见性。

参数传递：

值参数

引用参数

输出参数：可以返回多个值

public void mymethod2(ref int a)

public void mymethod2(out int a)

注：ref可以把参数的数值传递进函数，但是out就是要把参数清空。Out进去后参数的值就会变空所以必须初始化一次。Ref是有进有出，out是只出不进。

**C#的可空类型（NUllable）**

Nullable<short> na = null;

**?单问号**用于对int，double，bool等无法直接赋值为null的数据类型进行null的赋值，意思是这个数据类型是nullable类型的。

如：int?i=3; 等同于 Nullable<int>I = new Nullable<int>(3);

**??双问号**用于判断一个变量在为null的时候返回一个指定的值。为空取代

**C#数组**

int[] arr = new int[10];

int[] arr2 = new int[5] {1,5,33,456,2 }; //第一种

int[] arr3 = { 1, 5, 33, 456, 2 }; //第二种

arr[0] = 5; //下标赋值

**C#字符串**

通过给string变量指定一个字符串 string a = "hansj98789~!gf";

通过使用string类构造函数 string b = new string(new char[] {'a','b','w' });

通过使用字符串串联运算符 string c = "afv" + a;

public static string Contat(string str(),string str1); //连接两个string对象

public bool Contains(string value); //判断指定value是否在字符串中

如：bool v=a.Contains(b);

public bool EndsWith(string value); //判断value是否是字符串的结尾值

public bool Equals(string value); //判断两个对象是否相同

public static string Format(string format, Object arg0); //把指定字符串替换为对象的字符串表达形式。如：string a = "hansj{0}98789~{1}!gf";

string vl = string.Format(a,"你好","世界");

**C#结构体：**

Prop

**C#枚举**

是一组命名整型常量

**C#什么是类**

类的定义是以关键字class开始，后跟类的名称。类的主体，包含在一对花括号内。

类的构造函数是类的一个特殊的成员函数，当创建类的新对象时执行。

有参构造器，无参构造器

类的静态成员：

用static关键字把类成员定义为静态的。当我们声明一个类成员为静态时，意味着无论有多少个类的对象被创建，只会有一个该静态成员的副本。

分为：静态变量，静态函数（方法）

调用方式：类名.静态成员名称 类名.静态函数名称

Car car1 = new Car("比亚迪",120);

car1.SetName("奔驰");

Console.WriteLine(car1.Name);

在静态方法里，不能调用动态属性或变量。

**析构函数：**

**类的析构函数 是类的一个特殊的成员函数，当类的对象超出范围时执行。**

**析构函数的名称是在类的名称前加上一个波浪形（~）作为前缀，它不返回值，也不带任何参数。析构函数用于在结束程序（比如关闭文件、释放内存等）之前释放资源。析构函数不能继承或重载。**

访问修饰符：

可以定义一个类成员的范围和可见性。

* + **Public**：所有对象都可以访问；
  + **Private**：对象本身在对象内部可以访问；（仅自己有权限）
  + **Protected**：只有该类对象及其子类对象可以访问（仅自己和子孙有权限）
  + **Internal**：同一个程序集的对象可以访问；（仅家族人士有权限）
  + **Protected internal**：访问限于当前程序集或派生自包含类的类型；（仅家族人士或自己和子孙有权限）

类默认修饰符是internal，能被public修饰，其他不可。

若修改为public后，点住alt+enter键，即可替换更新信息。

**重载**

**针对的是方法。**

特点：在同一类或接口里面；

函数名称相同，参数列表不同 返回类型可以相同也可以不相同

比如：public void Run(){} public void Run(string name){}

public int Run(string name, int a)

public Car Run(string name, int a, int b)

{

return new Car(); }

**重写**

多态是同一个行为具有多个不同表现形式或形态的能力。

多态性意味着有多重形式。

特点：（1）在子类里面；（2）函数名称相同，参数列表相同，返回类型相同，唯一不同的就是函数里面具体的代码。

如下面：TVSet是父类，BwTVSet和ColorTVSet是子类，也称派生类

TVSet tVSet1 = new BwTVSet();

TVSet tVSet2 = new ColorTVSet();

tVSet1.Play();

tVSet2.Play();

属性：（1）定义成员变量：private <data\_type> <variable\_name>

定义属性： public <data\_type> <prop\_name> {get{return <variable\_name>};

Set{<variable\_name>= value};}

（2）public <data\_type> <prop\_name> {get;set;}

例程：第（1）种private int money;

public int Money { get => money; set => money=value; }

第（2）种 public int Money { get; set; }

**C#接口**

接口定义了属性、方法和事件，这些都是接口的成员。接口只包含了成员的声明。成员的定义是派生类的责任。抽象类在某种程度上与接口类似，但是它们大多是由基类的派生类。

实现接口的快捷键：Alt+enter 实现接口

（1）接口本身不实现任何功能；

（2）抽象类不能直接实例化；可以使用其派生类。

如：Animal animal = new Pig(); //实例化只能使用其派生类，接口类型的animal不行

void run(); //不能实现功能，即扩充{}，否则报错

C#正则表达式：

正则表达式 是一种匹配输入文本的模式。

string content = "Warning! \u0007"; //报警字符 \u0007

//正则的类

Match match = Regex.Match(content,"\a");

if (match.Success)

{

Console.WriteLine("匹配");

Console.WriteLine(match.Value);

}

//匹配手机号

string telPattern = @"1[3-9]\d{9}";

**C#异常处理：**

异常是在程序执行期间出现的问题，C#中的异常是对程序运行时出现的特殊情况的一种响应。

常见异常类：

|  |  |
| --- | --- |
| 异常类 | 描述 |
| System.IO.IOException | 处理I/O错误 |
| System.IndexOutOfRangeException | 处理当方法指向超出范围的数组索引时生成的错误 |
| System.OutOfMemoryException | 处理空间内存不足生成的错误 |
| System.StackOverflowException | 处理栈溢出生成的错误 |

如：using System.Text.RegularExpressions;

using System.Windows;

try {

//可能存在的异常码

//第一种 空指针异常

/\*Car car = null;

car.SetName("");\*/

//第二种 运行时异常

/\*int a = 0;

Console.WriteLine(10/a);\*/

//第三种 数组越界

int[] arr = { 8,9};

Console.WriteLine(arr[1]);

}

catch (Exception ex) {

//捕获异常码

//throw;

Console.WriteLine(ex.Message); //抛出异常

MessageBox.Show(ex.Message); //弹窗式

}

Finally 代码块最终都会执行，不因为return而结束。

**文件IO流操作**

This-----代表对象本身 它是通过实例化出来的，在非静态方法里面使用（）

Base-----代表父类对象本身 它是通过实例化出来的 在非静态方法里面使用

C#文件的输入和输出

当打开文件进行读写时，它变成一个流（stream），从根本上说，流是通过通信路径传递的字节序列。有两个主要的流：输入流和输出流。**输入流用于从文件读取数据**（读操作），**输出流用于向文件写入数据**（写操作）。

**操作：** 文件创建，删除，写入数据，读取数据

FileInfo

FileStream 这两者区别

-----------------------分----------割------------线----------------------

可空类型： int？num=1;

Null合并运算符（??）：如果第一个操作数为null，则返回第二个操作数，否则返回第一个。 例如： A=B??C //如果B为空则将C赋值给A，如果B不为空则将B赋值给A。

函数和Java一样具有**构造函数**和**带参函数**，且在这多了一个**析构函数。**

**析构函数以 ~ 作为前缀，不需要任何修饰符，不返回值，也不带参数，不能继承或重载。大致意思是用来释放资源的，适当时候回收。**

继承是面向对象设计的最重要概念之一，一个类继承自另一个类。比如：class A{ } class B:A{ }.B继承自A，B称之为子类/派生类，A可以称为基类/父类。 ----派生类和基类，子类和父类。 B拥有访问A的能力，形象点说就是子类可以拿到父类的财产，父类拿不到子类的财产。

C#不支持多重继承，C++是支持多重继承。

多态是面向对象程序的核心功能之一，是同一个行为具有多个不同表现的能力。函数重载，动态重载（abstract，virtual）

**接口使用interface**关键字声明，和类class的声明差不多，默认public。

**命名空间namespace** 是方便管理，比如同一个文件夹下不能有同名的文件，不同文件夹下可以有相同名的文件。 也可以命名空间套娃操作，如：using namespace.namespace

异常：try，catch，finally，throw

**Try：**一个try块标识了一个将被激活的特定的异常的代码块，后面跟一个或多个catch块。

**Catch：**程序 通过异常处理程序 捕获异常，catch关键字表示异常的捕获。

**Finally：**finally块用于执行给定的语句，不管异常是否被抛出都会执行，比如：如果打开一个文件，不管是否出现异常文件都要被关闭。

**Throw：**当问题出现时，程序抛出一个异常。使用throw关键字来完成。

**反射：**

**通俗的讲就是可以访问一个类的所有属性和方法，且能够调用。**与public的区别：一个是正常写代码，知道逻辑和内容；另一个反射是访问不清楚的类，比如第三方框架的某个类/方法不对外开放，而自己又需要使用就用反射把她抛出来。

优点：（1）反射提高了程序的灵活性和扩展性；（2）降低耦合性，提高自适应能力；（3）允许程序创建和控制任何类的对象，无需提前知道目标类。

缺点：（1）性能问题：使用反射是一种解释操作，用于字段和方法接入时要远慢于直接代码，普通程序不建议使用；（2）使用反射会模糊程序内部逻辑；反射会绕过源代码的逻辑技术，会比相应的直接代码更复杂。

**委托：**

使用关键字 delegate来修饰，必须用new关键字来创建

官方描述：表示对具有特定参数列表和返回类型的方法的引用。通过委托，可以将方法视为可分配给变量并可作为参数传递的实体。类似于C中的“函数指针”概念，但委托是面向对象且类型安全的。

扩散理解：委托就是包一个外壳，操作外壳就是调用壳内的内容。就好像老板让你搬砖，你委托给另一个人让她搬砖，这就是委托。

委托中还能通过加减号来实现多个，称之为多播。只有相同类型的委托可以合并。

---------------------------------------------------------------------

string[] str1 = new string[4] { "1","2","3","A"};

int len1 = listBox1.Items.Count;

if (!int.TryParse(text, out int index))

public List<AbstractCommand\_New> CmdSet\_New

{

get

{

return ListsContainer\_CDHD.CmdSet\_New;

}

}

Indexof 如果找到该字符，则为 value 的[索引](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/indexof/_blank)位置；否则如果未找到，则为 -1。

text.Trim() 移除当前 String 对象中的所有前导空白字符和后置空白字符

Substring 返回从指定的字符位置开始且具有指定的长度的 字符串

Find()   检索与指定匹配的第一个元素

FindAll()   检索与指定匹配的所有元素

Lambda表达式：

p => p.Age > 6).ToList(); //所有Age>6的Person的集合

p => p.Age == 1); //Age=1的单个people类

p => p.Name.Contains("年龄")).ToList(); //所有Name包含年龄的Person的集合

text = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData);

CommonApplicationData             目录，它用作所有用户使用的应用程序特定数据的公共储存库。

## 5.2 VS组件

### 5.2.1 C#语言部分

常用快捷键：

Ctrl+K+D 快速对齐代码

Ctrl+J 快速弹出智能提示

Ctrl+K+C 注释所选代码

Ctrl+K+U 取消对所选代码的注释

F1 转到帮助文档

折叠冗余代码：#Region #EndRegion

文档注释： ///多用来解释类或方法

Eeprom flash bootloader区别

Try 语句是用来判断是否有异常

Catch 语句块捕捉异常，并进行处理；

Throw 是抛出异常；

当界面组件文字被缩进时，在组件的属性框中---AutoSize---True。即指定空间自动调整自身的大小以适应其内容。

粉色，类似于微信的收藏图标-----指定义的方法（函数）；

蓝色，类似于橡皮擦-----指定义的字段？

This.sendCommand(); //this 指什么意思？

**解决方案资源管理器：**

1. 解决方案
2. 项目：form1.cs 该窗体文件里包含 窗体的设计，代码的编辑
3. 引用：硬件驱动，数学工具包
4. Form.cs 源代码所在地；

添加引用，就是using在文件顶端添加头文件

工具箱：

一个意思，就是熟悉。不同组件对应的结构体不同，也就是结构体的套用

.dll 是指动态链接库文件

IDE-----Integrated Development Environment---集成开发环境

SDK-----Software Development Kit---软件开发工具包

通常main方法中不写具体代码，只用作实例化和方法调用。

Console 类表示控制台应用程序的标准输入流、输出流和错误流。

编写规范：（1）尽量使用接口，然后使用类实现接口，提高程序灵活性

（2）建议局部变量在最接近使用它的地方声明

（3）避免书写代码量过大的try……catch

（4）生成构建长的字符串时，要使用StringBuild类型，而不是string

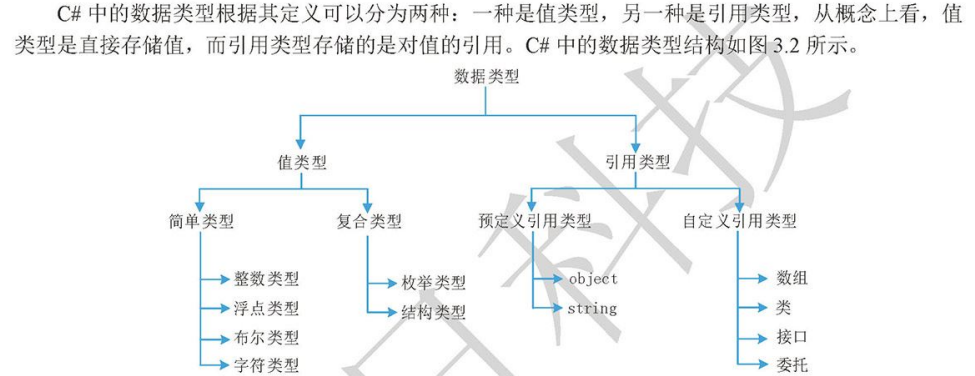
三种项目类型：控制台应用程序、Windows窗体应用程序，ASP.NET网站应用程序

控制台应用程序：没有独立窗口的程序，在命令行运行

Windows窗体应用程序：在计算机上运行的客户端应用程序。如QQ

ASP.NET网站应用程序：是可以通过web访问的应用程序，在浏览器打开即可，如百度，淘宝网

数据类型：



局部变量只有在当前定义的方法内有效，不能用于类的其它方法。局部变量的生命周期取决于方法。

Readonly常量是一种特殊的常量，也称动态常量。特殊之处在于，只能在**构造函数中赋值**。

在C#中，整数的默认类型是int型。

&&若左边错误直接输出结果； & 左边错误还会继续判断右边表达式； 结果一致。

&两边是表达式，输出结果是bool值； &两边是数字，则进行按位与运算并输出结果。

隐式类型转换 是不需要声明就能进行的转换；

decimal类型表示128位数据类型，它的精度可以达到28位。

Break 和continue 在嵌套循环中，只应用该关键字所在循环中。

一维数组的创建：（1）int[] arr; arr = new int[5]; //先声明，再为数组分配内存

(2) int[] month = new int[12]; //声明与数组分配内存同时进行

声明时没有分配内存，那么就使用关键字new分配内存

二维数组的创建：（1）int[ , ] myarr; int [ ][ ] myarr;

(2) int [ , ] a = new int[2,4]; //定义一个2行4列的数组

(3) int[][] a = new int[2][]; //定义一个2行的int类型二维数组

a[0] = new int[2]; //初始化二维数组的第1行有2个元素

a[1] = new int[3]; //初始化二维数组的第2行有3个元素

(4) int [,] myarr = new int[2,2];

myarr[0,0] = 0; //为二维数组的第1行第1列赋值

myarr[1,1] = 0; //为二维数组的第2行第2列赋值

**例子：**string s = String.Empty ; //定义空的字符串变量

s = "2,3";

string[] ss = s.Split(','); //拆分字符串，放入数组中

int one = int.Parse(ss[0]); //得到座位行数

int two = int.Parse(ss[1]); //得到座位列数

zuo[one, two] = "【已售】"; //标记售票状态

C#还支持不规则数组：

int[][] a = new int[3][];

a[0] = new int[5]; //第一行分配5个元素

a[1] = new int[3]; //第二行分配3个元素

a[2] = new int[4]; //第三行分配4个元素

foreach语句通常用来遍历集合，如foreach（string role in roles）

对数组进行排序：Arr.Sort() 方法用于对**一维**Array中的元素进行排序。

Arr.Reverse 方法用于反转**一维**Array中的元素**顺序**

IndexOf 方法返回的是搜索字符或字符串**首次出现**的索引位置；

LastIndexOf 方法返回的是搜索的字符或字符串**最后一次**出现的索引位置；

如： string str = "we are the world";

int size = str.IndexOf('e');

StartsWith方法用来判断字符串是否以指定的内容开始；

EndsWith 方法用来判断字符串是否以指定的内容结束；

如：bool result = str.StartsWith("梦想");

Equals方法主要用于比较两个字符串是否相同，相同返回true，否则为false；

如：public static bool Equals(string a,string b);

ToUpper 方法用来将字符串转换为**大写形式**； Console.WriteLine(str.ToUpper()); //大写

ToLower 方法用来将字符串转换为**小写形式**；

format 用来指定字符串所要格式化的形式

如：Console.WriteLine(String.Format("当前日期的短日期格式表示：{0：d}",strData));

String 本身不可变，只能赋值一次；

StringBuilder 值可变，可通过追加移除、替换或插入字符等方式修改

面向对象程序设计具备三大基本特征：封装、继承和多态。

**封装：**将对象的属性和行为封装起来的载体是类。

**继承：**实现重复利用，子类通过继承复用父类的属性和行为的同时，又添加了子类特有的属性和行为。

**多态：**将父类对象应用于子类的特征就是多态。

类的定义包括类头和类体两部分。类体中的成员包括：字段、属性、方法、构造函数等。

字段：就是程序开发中常见的常量/变量；

属性：描述状态信息；

构造函数：该函数名称与类名相同。每个类都有一个，若不定义则会有默认无参构造函数；

静态构造函数：只会在引用类之前执行一次，且不能带任何参数。

析构函数：主要用来释放对象资源，以类名加前缀“~”来命名。

析构函数和构造函数，主要用来对对象进行初始化和释放对象资源。一般来说，对象的生命周期从构造函数开始，以析构函数结束。

不能把类定义为 private、protected、protected internal类型，没有实际意义，这些修饰符只能用应用于成员。

方法的重载：指方法名相同，但参数的数据类型、个数或顺序不同的方法。编译器会判断在哪种情况下调用哪种方法。

在C#程序中，共享的变量或方法用static修饰，它们被称作静态变量和静态方法，**静态成员是属于类所有的**。

对象的创建： Test test = new Test();

将对象赋值为“null”时，会被销毁。

**Protected 表示受保护的成员**，只有父类和派生类（子类）才能访问该成员，外部代码不能访问

在实现类的继承时，子类的可访问性要低于或等于父类的可访问性。

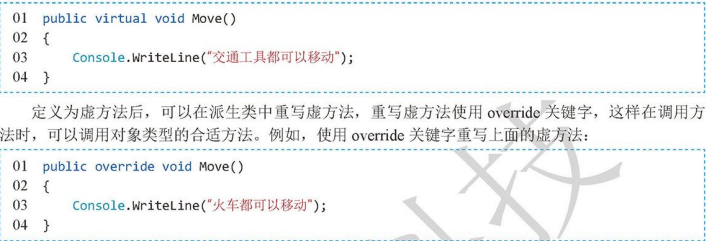
C#中只支持单继承，不支持多重继承；

**This关键字代表本类对象，base关键字代表父类对象**。

多态：

虚方法：在方法前面加上关键字virtual ，则称该方法为虚方法。

定义虚方法后，可在派生类中重写虚方法，需要使用override关键字



抽象，抽象方法不提供具体的方法，而是留下让派生类来重写实现。

接口，可以包含属性、方法、事件等； 只能定义成员，不能实现成员；

接口中定义的成员不允许加访问修饰符，因为接口成员永远是公共的；

接口中的成员不能声明为虚拟或者静态；

**组件笔记：**

（1）MessageBox.Show("请选择保存路径"); ---弹出窗口，提示

MessageBox.Show("未选择串口", "无法获取串口"); -----弹出窗口（内容，框名称）

（2）textBox\_path.Text = folderBrowserDialog1.SelectedPath; //设置保存路径 ---会弹窗口选择路径

（3）console.Read方法：返回值为int类型；

Console.ReadLine 方法：返回类型为string类型，可将控制台输入的任何类型存储为字符串类型数据。

Console.Write方法：输出后不换行，如：Console.Write（”hello world!”）;

Console.WriteLine方法：输出后换行

**重要例子：**Console.WriteLine((i+1)+ "月有" + day[i] + "天");

Console.Write(role+" "); //输出遍历到的元素

（4）int age = Int32.Parse(Console.ReadLine()); //将输入的年龄转换成int类型

（5）int y = Convert.ToInt32(x); //强制类型转换为32位

理解构造，逻辑的思想

对于《button》

对于《comboBox》

对于《textBox》

对于《label》

C和C#区别：

（1）static。在C中，static修饰的静态局部变量只执行一次，直到程序运行结束后才释放；修饰全局变量时，只能在本文件中访问，修饰函数时只能在本文件中调用。

在C#中，在类中任意调用。

**疑问：**

（1）.NET Framework组件，COM组件，WPF组件，通用Windows组件的区别

答：.NET Framework组件是全功能版类库的丰富子集，包括对XML Wed Services、Windows窗体、数据、绘图等。

COM组件，是一些小的二进制可执行文件，C，C#，C++都可以编写COM

WPF组件（Windows presentation foundation），主要负责图形显示

（2）有一个问题：每个组件对应的方法名，如何在属性栏中进行修改？？

答：在属性栏—外观：这里是修改按钮代表用户使用的名称

在属性栏—设计：这里是修改代码中标识该对象的名称

1. foreach关键字的用法？

Foreach语句是for语句的特殊简化版本。

Foreach的语句格式：for（元素类型(int) 元素变量x，遍历对象obj）{引用了x的语句}

Foreach没有索引，不能操作元素中的元素。

**例如：int arr[] = {7,10,1};**

**System.out.println(“一维数组中的元素分别为：”);**

**For(int x : arr ) { system.out.println(x); } //遍历该arr数组**

1. var……in…… 用法？
2. List<string> temp = new List<string>(); 这种创建方式？？

答：List是泛型

### 5.2.2 VS控件学习

更换窗体的图标：属性-Icon，更改图片即可

隐藏窗体的标题栏：需要将FormBorderStyle属性设置为None即可。

窗体跳转：在按钮调用： Form2 frm2 = new Form2();

Frm.show();

this.Hide(); //可以把当前窗体隐藏。

click单击事件：

消息提示类弹窗：MessageBox.Show("已经点击了该窗体！");

load加载事件：

if (MessageBox.Show("是否查看窗体！", "", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{ }

FormClosing（关闭）事件：

DialogResult dr = MessageBox.Show("是否关闭窗体", "提示", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (dr == DialogResult.Yes)

{

e.Cancel = false; //单击是 按钮，则关闭窗体

}

else

{

e.Cancel = true;

}

（3）MDI窗体，即多文档界面

起容器作用的窗体被称为“父窗体”，可放在父窗体中的其他窗体被称之为“子窗体”。

**见P225**。

（4）相关操作

添加控件有三种方式：a.在窗体上绘制控件；b.将控件拖曳到窗体上；c.以编程方式向窗体添加控件

WPF设计上位机界面，其文件后缀名是 **.xaml**。

**.cs** 是C#的源代码文件后缀名。

NotifyPropertyChanged 接口：向客户端发出某一**属性值已改的通知**。

控件和组件的区别：

1. 控件的基类是control，将其拖放到窗体中时，具有可视化的界面
2. 组件的基类是component，将其拖放到窗体时，不具有可视化的界面。

使用Command对象与数据源建立连接后，可以使用command对象对数据源执行查询、添加、删除和修改等各种操作，操作实现的方式可以使用SQL语句，也可以使用存储过程。

应用command对象调用存储过程，存储过程可以使管理数据库和显示数据库信息等操作变得非常容易，它是SQL语句和可选控制流语句的预编译集合，它存储在数据库内，在程序中可以通过command对象来调用，其执行速度比SQL语句快，同时还保证了数据的安全性和完整性。

SQL server数据库---> SqlConnection---> SqlCommand--->SqlDataReader--->win forms

dataSet 对象是将用户需要的数据从数据库中“复制”下来存储在内存中，用户对内存中的数据直接操作；

dataReader 对象则像一根管道，连接到数据库上，“抽”出用户需要的数据后，管道断开，所以用户在使用DataReader对象读取数据时，一定要保证数据库的连接状态是开启的。

public static void DoRef(ref int txt) //ref传递的是对象的引用，也可理解为值的地址

public static void DoNotRef(int txt) //传递的是值

委托，用于将方法作为参数传递给其他方法，事件处理程序就是通过委托调用的方法

C语言中“=>”的意思：lambda表达式，表示一个匿名函数，=>前面的是参数，后面的是函数体。你可以把它当作一个函数。举例说明：Func<int, int, int> Add = (x, y) => x + y;Console.WriteLine(Add(2, 3));

**生成的操作：**

**内容---不编译该文件，但将其包含在“内容”输出组中**

**编译---将该文件编译到生成输出中。此设置用于cs代码文件**

**嵌入的资源---将该文件作为DLL或可执行文件嵌入到主项目生成输出中。内容的读写和运行目录下的其他文件没有区分。**

复制到输出目录，是指生成的目录文件；不复制，指不会复制到输出目录中；若较新则复制和始终复制，指会在生成后复制到输出目录中；

生成操作，是指exe文件安装后的资源文件，选择嵌入的资源会在生成操作后生成一份文件。

频繁出现的两个问题：

索引超出了数组界限如何解决

未将对象引用到对象实例：该实例/组件 未加载

**8K频率带宽，会出现 叽叽的电流声**

失去焦点和获得焦点是一个鼠标行为，获得焦点是指当鼠标点击输入框使其处于编辑状态，这就是获得焦点。失去焦点是指当鼠标点击其他输入框或其他区域时刚开始的那个输入框就属于失去焦点状态。

## 5.3 WPF

Windows presentation foundation（简称WPF）

可扩展应用程序标记语言（XAML）

Xmlns:x=”” 关系是x:前缀映射支持作为xaml语言定义一部分的内部函数

WPF是一种将xaml用作语言的实现，并为定义了其对象的词汇。

CLR指的是公共语言运行库

System.Windows.FrameworkElement 此类表示所提供的WPF框架级别实现基于UIElement定义的WPF核心级别API

控件分类：内容控件，文本控件，条目控件，布局控件，其它控件。

WPF是微软提供的一种用来开发“桌面应用”的技术，必须会xaml语法

WPF会在“引用”里自动添加“**PresentationCore、PresentationFramework、WindowsBase**”三个核心驱动集

App.xaml 设置应用程序的起始文件与资源

WPF 应用程序界面与XAML设计文件

StartupUri 属性可以指定运行时的启动窗体

1. Dll库是如何引用的；
2. 通讯协议的发送和处理在哪里---不断的跳转到底层
3. 封装的模块更多是关于什么样的内容。
4. 如何将设计的界面封装成工具箱，当做器件使用----自动生成

.g.i.cs 是前端可视化界面与底层业务代码的连接文件。

控制器连接界面：

<FromControls:ControlItem x:Name="m\_Contro\_form"/>

在WPF项目中，编译时，

第一步是把xaml编译成baml（obj/Debug下）。根据xaml生成部分类即g.cs（obj/Debug下）文件；

第二步，生成程序集，把部分类合并，并把baml嵌入程序集中

x:Class="HelloWPF.MainWindow” HelloWPF是命名空间后面MainWindow是他的一个类，位置在xaml文件所属的.cs文件中

<Grid>标签，相当于添加了派生类，也就是做扩展；**xaml是用来定义MainWindow这个类的UI部分**。

前端是xaml文件，后端是C#语言，两者联系在一起是依靠xaml解析器。

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" //包含的是一组名称空间，简称“xaml语言空间”

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" //包含名称空间里的几乎所有的UI元素

WPF与WinForm最大的区别是“数据驱动界面”。包括为**界面准备的XAML**，**为底层数据准备的Dependency Property**，**为消息传递准备的Routed Event & Command**。

XAML+C# :把软件的UI设计和逻辑设计分开

1. xaml能不能用来写逻辑代码？ ---不能
2. Xaml文件能不能包含逻辑代码？---可以

Xaml是一种声明性语言，就是用来声明UI元素的，不能用来写逻辑代码；

Xaml文件里可以包含逻辑，可以使用<x:code>标签来嵌入一些逻辑代码

和信，安陆，高云

配置文件.dll文件中提供了大量的读写配置，便于调用和修改。

**同步发送---设备返回命令---读写功能的尝试---写入参数命令**

**经验：text是可以直接赋值的；DataContext是绑定获取的值、**

**说实在的，IO的实时显示，问题就卡在两方面，一个是text和datatext的区别；另一个是修改后的触发事件的设置。 问题都很好解决，关键是自己没有想到，对这些并没有清晰的规划**

**Textbox 实现多行文本框，密码框。有边框，可输入的控件；常用【文本改变事件TextChange】**

**TextBox的PlaceholderText属性，在空间获取到焦点前会一直显示灰色提示文字。**

**TextBlock 用来显示少量的数据。直接继承FrameworkElement，**

**而Label继承于ContentControl。**

**控制模式显示---存在问题，P18.19为4时也可以进入位置，速度模式，要判断。--解决√**

**Invoke 是同步更新，会阻塞所在工作者线程；**

**BeginInvoke 是异步更新，不会阻塞当前线程；**

**Invoke调用后将指定代码立即插入主线程中执行；而BeginInvoke调用后，发送消息给UI线程，相当于向Dispatcher队列中添加工作项，待之前UI线程任务完成后，再执行BeginInvoke中的内容。**

**lock 关键字可以用来确保代码块完成运行，而不会被其他线程中断。这是通过在代码块运行期间为给定对象获取互斥锁来实现的。**

计算函数：Math.Pow(a,b); 求a的b次幂。

### 5.3.1 工程note

（1）P18.19为0，则电流值的标幺值是根据电机信息表的参数设定系数

（2）给示波器的通道一个额定的放大倍数

第一步：

ParseAndAddToChart(null, fromFile: false, m\_RecordDataAllDrives);// 将数据添加到chart

第二步：

if (fromFile)

{

m\_ChartDataParser.ParseTextRecordingData(orig, ref showMoreErrors, out PointsToRecord, out RecordSampleInterval, ref NewUnitsFlag);

}

第三步：

处理byte数据||处理文件导入数据

第四步：

先判断是否为空

再将MakeSeriesOnChart(array, array2, fromFile: false);//将串数据放到chart中

if (recorderData.resultsData == null) { return;}

第五步：

foreach (FrameworkElement child2 in containedScopeChart.GRID\_Charts.Children)//数据放到XY轴中

to pan the chart. //来平移图表

Legend //图例，标记

zoomExtents //缩放至默认大小

Animation //动画

Append //附加

Annotations //注释

Wrapper //包装，封装

ExponentialCurve //指数曲线

sci\_chart:MouseWheelZoomModifier //鼠标能滚动放大

sci\_chart:ZoomExtentsModifier //双击能缩放至默认大小

sci\_chart:ZoomPanModifier //能拖动示波器图

sci\_chart:RubberBandXyZoomModifier //鼠标画矩形放大

sci\_chart:TooltipModifier //可以看到单个通道数据

sci\_chart:CursorModifier //提示工具，可看到所有通道数据

DrawMajorBands //绘制主网格的线

DrawMajorGridLines //绘制主要网格的线

DrawMinorGridLines //绘制小网格的线

<sci\_chart:SciChartSurface.XAxis>

<sci\_chart:NumericAxis x:Name="XAxis" AxisAlignment="bottom" AxisTitle="频率（hz）"/>

</sci\_chart:SciChartSurface.XAxis>

instance  //实例

嗯嗯

LegendModifier 绑定到数据源

SciChartLegend 显示数据

Public property:

[AreAnyTouchesCapturedWithin](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //获取到一个值，指示元素是否至少有一次被捕获(只读)

[AutoFitMarginalLabels](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisBase~AutoFitMarginalLabels.html) //设置一个值，当轴太长时是否将边缘标签移动来适应轴

[AutoTicks](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~AutoTicks.html) //设置值，是否自动计算刻度，默认是true

[BorderThickness](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //边框厚度

[ContentStringFormat](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //字符串格式化，获取/设置复合字符串

[Cursor](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //游标

[CursorTextFormatting](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~CursorTextFormatting.html) //获取光标上的字符串标签

[DrawLabels](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~DrawLabels.html) //如果为true，则为每个主网格绘制主标签

[DrawMajorBands](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~DrawMajorBands.html) //设置为true，则绘制主网格的填充区域

[FlowDirection](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //设置文本方向，可控制父级元素层的流动方向

[ForceCursor](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //设定一个值，强制UI界面遵循光标的

[GrowBy](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~GrowBy.html) //获取增长因子。比如GrowBy(0.1,0.2)，会增加min10%，max20%

[IsAxisFlipped](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisBase~IsAxisFlipped.html) //当前轴是否反转（只读）

[IsLogarithmicAxis](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis~IsLogarithmicAxis.html) //是否是对数轴（只读）

[IsMouseCaptured](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //是否将鼠标捕获到此元素

[IsStylusOver](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //获取到一个值，该元素是否被触控笔捕获

[MajorGridLineStyle](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~MajorGridLineStyle.html) //设置样式

[MajorTickLineStyle](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~MajorTickLineStyle.html)

[MinorGridLineStyle](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~MinorGridLineStyle.html)

[MinorsPerMajor](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~MinorsPerMajor.html)

[MinorTickLineStyle](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~MinorTickLineStyle.html) //设置样式

[MajorDelta](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.NumericAxis~MajorDelta.html) //主增量

[MinorDelta](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.NumericAxis~MinorDelta.html) //最小增量

[RenderSize](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //获取此元素的最终呈现大小

[ScientificNotation](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.NumericAxis~ScientificNotation.html) //数字格式

[Scrollbar](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisBase~Scrollbar.html) //滚动条

[ShowAxisOffsets](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis~ShowAxisOffsets.html) //显示轴偏移

[TextFormatting](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~TextFormatting.html) //设置此轴的文本标记字符串

[TickCoordinatesProvider](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~TickCoordinatesProvider.html) //获取实例，转换坐标轴的坐标

[TickProvider](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~TickProvider.html) //获取实例，计算轴

[TitleFontSize](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisBase~TitleFontSize.html) //标题字体大小

[TitleFontWeight](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisBase~TitleFontWeight.html) //标题字体权重

[TitleStyle](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~TitleStyle.html) //标题样式风格

[UseLayoutRounding](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.LogarithmicAxis.LogarithmicNumericAxis_members.html) //获取设置值，指示布局时是否将布局舍入应用于该元素的大小和位置

[VisibleRange](https://www.scichart.com/documentation/win/current/SciChart.Charting~SciChart.Charting.Visuals.Axes.AxisCore~VisibleRange.html) //可视化范围

### 5.3.2 请教问题

（1）上位机掉线问题是如何解决的

（3）有测试串口掉线数据时，其中有一帧返回18 46 C8 B8……这个返回值是怎么回事，按照我的协议应该没有这样的返回值

（4）System.Windows.Forms.Application.DoEvents();这句话起的是什么作用？多线程的一个界面更新理论，因为那个位置没有加延时，有这个语句能够使UI进行重绘。

下个版本必须解决的问题：

1. 各个参数的单位
2. 数字IO功能
3. 辨识功能；
4. 示波器放大倍数
5. 掉线问题

要给楚坤无logo的上位机安装包

电流 10 0000

速度 1000

今天下午解决问题：

1. 控制模式界面的参数单位---定位
2. 数字io功能映射生成到安装包
3. 示波器放大倍数---定位
4. 启动界面 左上角---定位

**上位机遗留问题：**

**数字IO，延时时间实时显示----√**

**使能状态显示；-----√**

**ST6和ST7驱动器，**

**示波器放大倍数-----√**

**有些错误无法用上位机清除**

**底栏的模式不准确**

结束符

### 5.3.3 问题解决记录

1. 连线后，使能灯会自己亮

答：条件判断里，加入text判断，若为1则进入才会亮。这个解决方案是错误的。

在父.cs代码中，不要将电机使能/失能状态 放入到在线/掉线的条件中。

备注：完全是对代码逻辑不清楚。

2.状态栏，有的指示灯亮，有的指示灯不亮。

答：逻辑问题。已解决

3.如何使用父控件，获取到的值

private void ltlCurrent\_TextChangedEvent(object sender, TextChangedEventArgs e)

{

TextBox textBox = sender as TextBox;

## 5.4 git

Git本地的工作区域：

工作目录（working Directory），

暂存区（State/Index），

资源库（Repository）

远程git仓库（Remove Directory）

Git 的一般工作流程：

1. 在工作中添加，修改文件
2. 将需要进行的版本管理的文件放入暂存区域；
3. 将暂存区域文件提交到git仓库中；

因此，git管理的文件有三种状态：已修改（modified），已暂存（staged），已提交（committed）。

工作目录（WorkSpace）一般是希望git帮助管理的文件夹，可以是项目目录，也可以是空目录，建议不要有中文。

日常需要记住的6个命令：

Fetch/clone，push，pull，commit，checkout，add

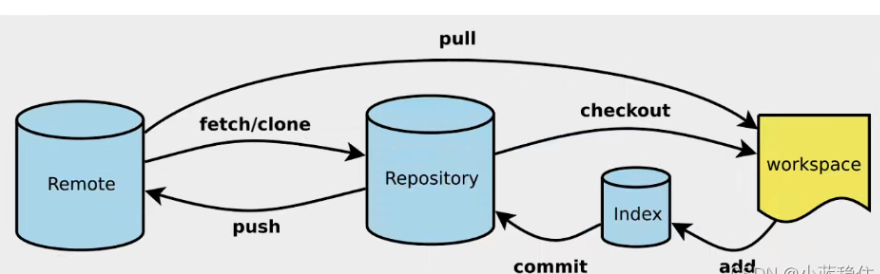
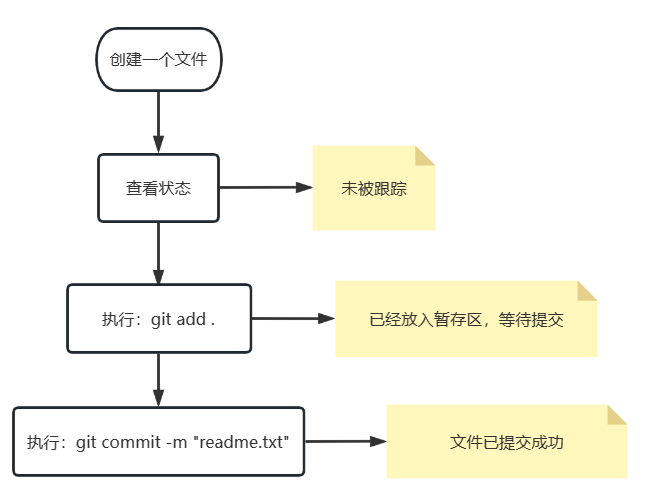


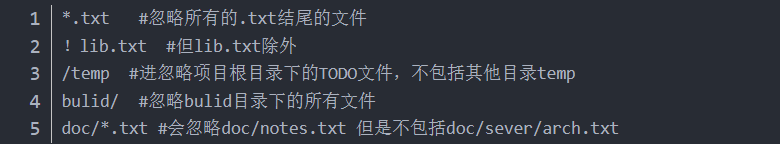
图5-4-1 常用git命令

git文件操作：

* Untracked：未跟踪，此文件在文件夹中，但并没有加入到git仓库，不参与版本控制。通过git add 状态变为staged。
* Unmodify：文件已经入库，未修改，即版本库中的文件快照内容与文件夹完全一致。这种类型的文件有两种去处：如果它被修改，则变为modified；如果使用git rm移出版本库，则成为untracked文件。
* Modified：文件已修改，仅仅是修改，并没有进行其他的操作，这个文件也有两个去处，通过git add可进入暂存staged状态，使用git checkout，则丢弃修改过，返回unmodify状态，这个git checkout即从库中取出文件，覆盖当前修改！
* Staged：暂存状态，执行git commit则将修改同步到库中，这时库中的文件和本地文件又变为一致，文件为****Unmodify****状态.执行****git reset HEAD filename****取消暂存，文件状态为****Modified****



**图5-4-1 文件提交流程**



**图5-4-2 忽略文件的规则**

git rm --cached 文件名称 移除Git仓库中对应的文件，保留工作区对应的文件

git rm -f 文件名称 同时移除本地仓库和工作区对应的文件

git log -2：查看最新两条历史

git log -2 --pretty=oneline：一行显示两条最新历史

git log -2 --oneline：与上个命令一样（对应提交唯一表识长度不一样，比上个命令更短）

git reflog --oneline：与上个命令一样（旧版本一行查看历史）

git log -2 --pretty=format:“%h | %an | %s”：%h提交的简写哈希值 %an作者名 %ar作者修订日期 %s提交说明

git reset --hard 跳转到指定历史

git add “文件名称”(指定文件放入暂存区)

git add .（表示当前所有文件放入暂存区）

git commit -m “add distributed” 提交文件

git commit -m “append GPL” 提交文件

git commit -m "文件描述" 指本次提交的说明

Git commit -m “commit message” 提交所有更新过的文件

git diff 查看修改内容，比较不同

git log --pretty=oneline 查看提交历史版本的精简版

git reset --hard HEAD^ 版本回退

git reset --hard a1684 指定回到某个版本（往前或往后）

cat readme.txt 查看文件内容

git commit -m “git tracks changes”

git diff HEAD -- readme.txt

git checkout -- readme.txt 撤回到最近一次git add或者git commit的状态

git remote -v 查看远程库信息

git remote rm origin 移除远程库

git push -u origin master 提交文件

git checkout -b dev 创建一个新的分支‘dev’,并切换

git branch dev 创建一个分支‘dev’

git checkout dev 切换到分支‘dev’

git switch -c dev 切换分支

git branch 查看当前分支

Git merge dev 将两个分支的成果合并，合并到当前分支中

git branch -d dev 删除dev分支

隐藏目录“.git”，是git的版本库

版本库存放了，包括stage和第一个分支master，以及指向master的指针HEAD。

结束符

# 第六章 驱动器测试工装

ST8有两个待优化：

（1）清除错误；

（2）使能寻相后的探针；

ST8驱动器编码器分辨率：2^12 /2 \*32（2^12 ÷2\*（极对距））

ST7 ST8的上电后得分探针形式 只需要03读功能码

STM32F1 90nm 72MHz M3

STM32F4 90nm 168MHz M4

STM32F7 90nm 216MHz M7

STM32H7 40nm 400MHz M7

让单片机通过wifi模块连接无线网并登录Aliyun的MQTT服务器并上传数据

**为什么对收到显示屏的数据进行处理后，值是EA=234？？？**

**答：是因为数据长度定义太短，1002=3EA**

**遇到一个奇怪的问题：**

**将keil的优化等级设置为0，就可以在调试模式下打断点，且功能恢复正常；否则工作不正常。难道优化等级会影响代码质量？**

目前占用时间长的是 复位和使能信号，也就是监控0x637,0x250,0x218

ST8驱动器：

0x6041---61443---0xF003 使能成功后的数据为**0x637**，0x237（ing）

运转过程中，使能标志位 0x1637

下使能，状态字先为0x631，再变为**0x231** 0x1231

报错情况下是**0x218**

报错清除后是0x250

上电重启 / 复位后的数据为**0x250**

**回零发送 0x1F（正在）**

**回零完成后 0x1637（完成）**

**回零模式下，报错状态字 0x2218**

**多段运行， 运转过程0x237**

**失能0x231**

**到达目标位置637**

0010 0001 1000 0x218 报错

0010 0011 0001 0x231 失能

0010 0011 0111 0x237 运转过程

0010 0101 0000 0x250 重启/复位

0110 0011 0001 0x631

0110 0011 0111 0x637 到达目标位置

4567

4+0=4

5+4\*16=69

6+69\*16=1110

7+1110\*16=17767

脉冲计数要准确，明天要能读取到计数值进行判断。----√

第二次检测无法通过---√

短路测试核对----√

URT4 ORE 溢出错误---定时器中断，发送modbus数据导致的串口接收溢出错误----√

加入看门狗

驱动器恢复出厂设置 试一下---√

脉冲计数，要读取count值----√

短路测试结果显示----√

在设备上运行

考虑ST6和ST7的功能测试，主要是modbus

加入看门狗

24577 0x4500 0：复位

一切的前提是modbus正常，相信它返回的数据是正常的。

Modbus poll 控制字 0xF002 61442 06失能 F使能

状态字 0xF003 61443

系统控制字 0x47B 1147

点击“ST8整机测试”---会获取ST8的标志位

点击“下一片”---在ST8界面，会获取到标志位：StartTest\_ST8，

---标志位会定位到ST8的测试函数和标志位 Test\_connectHMI

显示屏---测试结果

电机下压+驱动器通电

通过modbus写入辨识后的电机参数，并复位

版本核验（3A 6A 10A）

IO测试（驱动器准备好后，打开引脚继电器吸合，LED灯亮；

使能好后，另一半灯会亮，并对测试结果进行显示，

随之把IO检测的电平拉低关闭）

脉冲测试，先高速脉冲，再低速脉冲

短路测试，默认标志位为进行短路测试，可通过显示屏选择是否短路测试

教训及经验：

1. 无法正常写入modbus数据，是保存时间不够，已延长至2000ms。
2. 无法正常回零，是case 定义的define值不对，不能0x6066E
3. 无法正常捕捉到返回的数据，是中断函数中，没有形成循环，为step1重新赋值slave\_address。

如何快速排查问题，确定问题，找到解决问题的方法。

一个是设备实际情况问题，一个是基本知识问题，一个是个人细心程度的问题

**控制字：（8个）**

R reserved 保留

Oms operation mode specific 具体操作模式

H halt 停止

Fr fault reset 故障复位

Eo enable operation 使能操作

Qs quick stop 急停

Ev enable voltage 使能电压

So Switch on 接通，开启

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| r | | | | | | oms | h | fr | oms | | | eo | qs | ev | So |

其中oms(Bit4 5 6 9)在不同控制模式下定义不同：

**状态字：（12个）**

R reserved 保留

Oms operation mode specific 具体操作模式

Ila internal limit active

Rm remote

W warning 警告

Sod switch on disabled 失能

Qs quick stop 急停

Ve voltage enable 电压使能

F fault 故障，报错

Oe operation enabled 操作使能

Se switched on 接通，开启

Rso ready to switch on 准备接通（伺服准备好）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| r | | oms | | ila | oms | rm | r | w | sod | qs | ve | f | oe | se | Rso |

驱动器状态读取：（Bit0~3 BIt5 Bit6）

|  |  |  |
| --- | --- | --- |
| 状态字 | PDS状态 | |
| Xxxx xxxx x0xx 0000 | Not ready to switch on | 初始化未完成状态 |
| Xxxx xxxx x1xx 0000 | Switch on disabled | 初始化完成状态 |
| Xxxx xxxx x01x 0001 | Ready to switch on | 主电路电源OFF状态 |
| Xxxx xxxx x01x 0011 | Switch on | 伺服使能OFF/伺服准备状态 |
| Xxxx xxxx x01x 0111 | Operation enabled | 伺服使能ON |
| Xxxx xxxx x00x 0111 | Quick stop active | 急停 |
| Xxxx xxxx x0xx 1111 | Fault reaction active | 异常判断 |
| Xxxx xxxx x0xx 1000 | fault | 异常状态 |

Bit4 为1，指主电已上

Bit5 为0，表示

0000 0000 0000 0x000 初始化未完成

0010 0001 1000 0x218 报错

0010 0011 0001 0x231 失能

0010 0011 0111 0x237 使能

0010 0101 0000 0x250 重启/复位/报错清除

0110 0011 0111 0x637 到达目标位置

0001 0110 0011 0111 0x1637 动作完成

运转过程中，**动作完成标志** 0x1637

下使能，状态字变为**0x231**

使能成功后的数据为 0x237

报错情况下是**0x218**

上电重启 / 复位 / 报错清除 后的数据为**0x250**

**1.驱动器上电时，如果编码器未连接，ID号错误，错误参数直接导致的报错，或者硬件电路错误等之类的错误，显示屏，状态字应该先处于错误状态（0x218），而不是伺服准备好**

**2.上电时，存在报错条件，却能上使能，让电机处于使能状态。**

**3.上电后，能使能，发脉冲电机却不转，驱动器有接收到脉冲，脉冲数不对。感觉是驱动器使能条件未满足，却处于使能状态。无论显示屏还是状态字。**

CIA402

**ST7模拟量速度：**

P00.01 设置为1

P03.00 设置为0，速度指令来源

P05.03 系数

P05.09 模拟命令偏置

**ST7模拟量转矩：**

P00.01 设置为2

P05.00 设置为0，转矩指令来源

P05.06 模拟转矩命令转换系数,默认1

P05.09 模拟命令偏置

P03.00 数字速度

P03.36 目标速度

ST8模拟量测试：

电机参数导入

ADC1当前值，ADC2当前值

当前值模式：模拟量速度，模拟量力矩

参数设置-反馈参数-模拟量反馈设置：ADC1采样偏移，ADC2采样偏移

61449 0x6060 模拟量速度（22）模拟量转矩（23）

794 0x31A ADC1采样偏移值

796 0x31C ADC2采样偏移值

532 0x214 ADC1当前值

534 0x216 ADC2当前值

测试流程：

显示屏--

Modbus---

偏置值计算--

重新上电--

使能--

模拟量速度（转矩），实时显示速度值---

失能，测试通过

先制作显示屏，再写代码

地位寻

结束符

# 第七章 伺服电机、步进电机、FOC

步进电机是一种将电脉冲信号转换成相应 角位移 或 线位移的电动机，每输入一个脉冲信号，转子就转动一个角度或者前进一步，其输出的角位移或线位移与输入的脉冲数成正比，转速与脉冲频率成正比。因此，步进电机又称为脉冲电动机。

步进电机相当于一个完成数字模式转化的执行元件

步进电机无法直接在交流/直流电源上工作，需使用驱动电源（步进电机驱动器）。

伺服电机和步进电机的精度不一样。

步进电机一般按励磁方式分为磁阻式、永磁式和混磁式三种；按相数可分为单相，两相，三相和多相等形式。

# 附录1：调试问题汇总

B000 DD电机

B010 伺服电机

B020 直线电机

**绝对位置**，是设定多少值，电机就走到那个位置，这个值是目标值。

**相对位置**，相对当前位置，走多少距离，这个距离值就是输入的。

1. **V6E驱动器**

（1）V6驱动器，位置偏差过大不会报错

答：首先看引脚是否焊接：15端子----O3----定位完成。

位置偏差过大阀值，P00.19 是可设，可修改的。

（2）力矩限定模式

答：默认参数值：3 1 100

P0827 3力矩模式

P0829 1 10 100 报警时间 毫秒

P0830 100% 200% 输出电流百分比

1. ST6驱动器调试问题

**487 是正常脉冲模式**

**511 是ethercat模式**

**P19.23 设为0，脉冲模式**

**P19.23 设为1，ethercat模式**

1. **ST7驱动器调试问题**

（1）ST7驱动器正常启动，但使能后，运行不正常，电机ID也是正常的？

答：对于伺服电机来说，优先考虑速度环增益是否参数过大

（2）ST7驱动器驱动DD电机时，若运行咔咔，或不稳定？

答：优先看惯量比，一般自己测试DD电机的惯量比是500左右，之前认为空载跑1000是有问题的，先从500开始累加。

（3）旋转方向，脉冲方向

答：电机旋转方向：P00.00-----0/1

脉冲反馈方向：P00.16-----0/1

（4）电流环标定

答：如果标定现象 电机过于振动，P19.20 可升高频率给定，P19.22降低电流给定，电流减少。现象可能会弱一些。

P19.20频率越高声音越不一样，正常的标定频率是 100.

（5）小的DD电机参数

答：位置环100，速度环10000，速度积分时间10000，转矩指令滤波2000

1. 自动上电使能 P04.40 510；
2. 额定电流2.6A，想让驱动器超过1A就给出信号

解决方案：P18.18设定额定电流，P04.46设定转矩输出限定值，信号由IO端子的03发出。比如：额定电流2.6A，设定1A报警，则将P04.46设置为385（1/2.6=0.385），到达1A，则03会报警。

1. 选择1300W的电机，单圈17位5对极，选择ID号61

P02.00---10 P02.01---3

1. DD电机飞车问题
2. 电机没有锁零；b.最大速度输出限制；c.过速报警
3. 伺服电机，上使能直接飞车

解决方案：

1. ST7驱动器，23位编码器，上使能剧烈抖动

答：速度环积分时间1填写到10000

1. **ST8驱动器调试问题**

（1）ST8驱动器，上电后使能并没有抱闸，辨识不通过。

答：更换动力端子？

（2）ST8驱动器，参数正确，但是辨识不通过，使能跑有问题

答：第一种情况：电流环频率修改，40K—32k

（3）电机编码器类型

答：旋转的好说。但是！直线的，要选12，编码器分辨率是2^(12)\***20**/2=40960

ABZ的，比如增量式22位ABZ编码器，

编码器线数：2^(22)/4 =4194304/4=1048576

（4）回零开关如何设置

答：应用功能GPI设置回零开关（除input 1）（1）第一种是在设备控制-回零模式—选择33/34（编码器encoder回零）；（2）第二种是回零模式—选择7/8/9/10四种回零模式

（5）辨识时报错：电机过载

答：（1）看电流设置，是否过小。（2）ST8驱动器，电流环运行频率是32000Hz。

（6）脉冲控制模式的系数

答：脉冲输入滤波系数-----设置为最大脉冲输入速率的5倍-10倍

平滑系数-----越大轨迹越平滑

（7）DD电机一直有类似于电流声。

答：降低位数（软件23-20），或者降低转子惯量

（8）辨识过程一直报过速0x8400

答：第一种方法，选择好运动模式，把最大速度限制设定好，初始默认值是很小的，有时系统会判断该值而出错；第二种方法，把最大速度限制保护打开再辨识。

（9）脉冲控制模式，到达一定速度后跑不上去

答：脉冲输入滤波系数要加大，一百万的系数，每秒多少脉冲数。

V6驱动器：3（24V+） 4(EN) 5(ERR\_RST) 12(GND) 21(S\_ERR+) 22(S\_ERR-)

（10）ST8报错：驱动器累计热量过载

1. **高创驱动器**
2. **反馈类型：AB读头，选择“增量式编码器A/B 上电使能PHASEFIND初始化”**

**每转线数 8000LPP**

1. **moveincspeed1 1000**

**moveincspeed2 1000**

**en //使能**

**k //杀死进程**

**acc 10000 //设置加速度**

**dec 10000 //设置减速度**

**mb //激活运动缓冲器**

**movesmoothavg //位置指令移动平均滤波器**

**4．其它**

（1）高川板卡与ST7驱动器的IO高速/低速脉冲线

**高川板卡(13pin)：**1（GND） 2(ALM\_IN) 3(EN) 4(A+) 5(B+) 9(DIR)

10(GND脉冲) 11(PULSE-) 14(+24V) 15(ALM\_CLR) 17(A-) 18(B-) 22(DIR-) 23(PULSE+)

ST7驱动器：3(24V+) 4(EN) 5(ERR\_RST) 12(GND) 18(S\_ERR+) 19(S\_ERR-) 32(GND) 33(A+) 34(A-) 35(B+) 36(B-)

低速：25(PLS+) 26(PLS-) 27(DIR+) 28(DIR-)

高速：39(H\_PLS+) 40(H\_PLS-) 41(H\_DIR+) 42(H\_DIR-)

V6驱动器：3（24V+） 4(EN) 5(ERR\_RST) 12(GND) 21(S\_ERR+) 22(S\_ERR-) 33(信号GND) 36（A+） 37（A-） 38(B+) 39（B-）

只有低速：26（PLS+） 27（PLS-） 30（DIR+） 31（DIR-）

**高川---ST7驱动器IO端子：**

1-19 2-18 **9-27 11-26 23-25 18-35** 17-33 5-36 4-34

14-3 1-12 3-4 10-32 **22-28**（低速线）

1-19 2-18 **9-41 11-40 23-39 18-35** 17-33 5-36 4-34

14-3 1-12 3-4 10-32 **22-42**（高速线）

**高川---V6驱动器IO端子：**

**1-22 1-12 2-21 3-4 4-37 5-39 9-30 10-33 11-27 14-3 15-5 17-36 18-38 22-31 23-26** （低速线）

备注：

1. A+ A- B+ B- 脉冲反馈
2. ST8驱动器，编码器分辨率65536，设置一圈反馈脉冲数为65536，即一圈32mm，一个极对距。若设置1000rpm=1000mm/s，运行速度填写65536/32=2048p/mm。若设置5g加速度=50000mm/s^2，加减速度填写65536\*50000/32/1000000=102.4。

**重点：一圈反馈脉冲数/32=\*\*\*p/mm；**

（2）读头DB9---编码器DB15转接线

DB9 通讯式读头（母）**：**4(485+) 5(+5V) 8(485-) 9(GND)

DB9 ABZ读头（母）**：**2(Z+) 3(B+) 4(A+) 5(+5V) 6(Z-) 7(B-) 8(A-) 9(GND)

DB15 编码器公头**：** 1(B+) 2(B-) 3(485+) 4(485-) 8(A+) 9(A-) 10(Z+) 11(Z-) 14(+5V) 15(GND)

**DB9通讯式---DB15编码器公头：**

**4-3 5-14 8-4 9-15**

**DB9 ABZ读头---DB15编码器公头：**

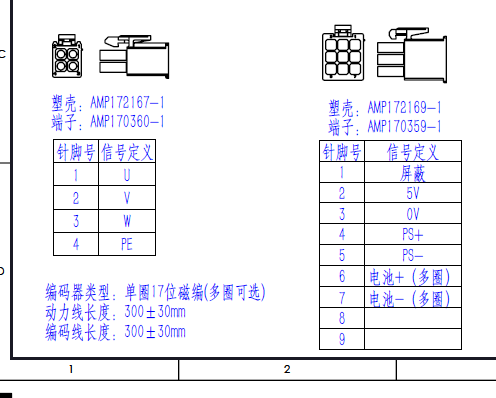
**2-10 3-1 4-8 5-14 6-11 7-2 8-9 9-15**

1. 电机的编码器头-----驱动器的编码器头

⊥

电机端口编码器：1（485+） 2（485-） 3（+5V） 4（GND） 5（屏蔽线）

驱动器DB15编码器：3（485+） 4（485-） 14（5V） 15（GND） 外壳（屏蔽线）

】



1. 增量式编码器和多圈区别

增量式编码器，上电后以上电位置为零点；单圈（多圈）编码器以实际位置为起点，比如上电时处于60°，那么就以60°为起点。

垂直L：信号+ 信号- 屏蔽线 GND +5V

5（+5V）9（GND）4（信号+）8（信号-）

1. 模拟量接线

3（+12V），12（GND）,21（REF2+），22（REF2-），29（REF1+），44（REF1-）

结束符

# 附录2：life mark

更何况高楼的每个房间把大家分割开，宅男宅女的出现也不是可以理解