【行业技术点解析】
印刷机收放卷张力控制思路及计算方法

印刷机收放卷张力控制
张力控制思路及计算方法

刘柏严

Liu Baiyan

Technical Expertise

B&R Industrial Automation (China) Co., Ltd.

E-Mail：baiyan.liu@br-automation.com

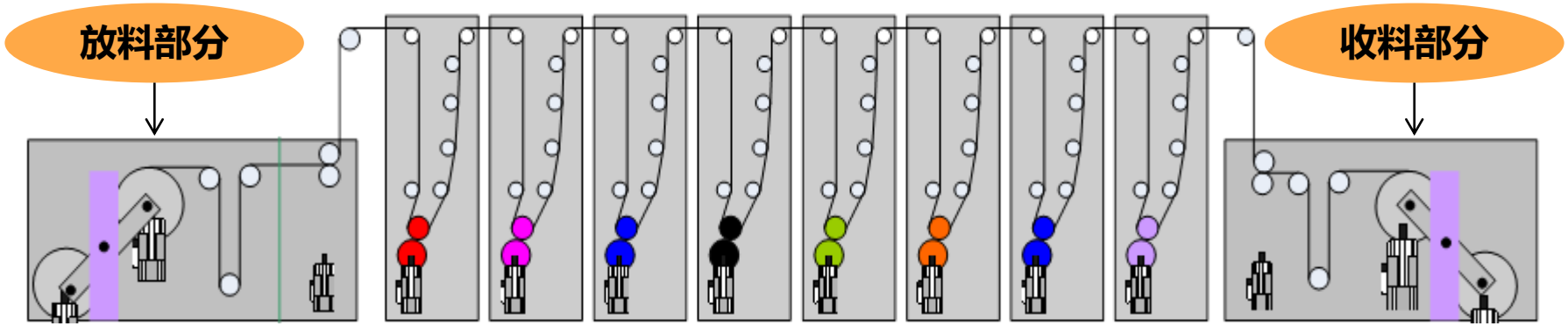
www.br-automation.com

20221221

印刷机收放卷张力控制思路及计算方法

演讲内容

1. 介绍收放卷基本结构
2. 主要控制原理及开发思路
   - 需求点，关键指标
   - 开发任务分配
   - 函数拆分及功能设计
3. 两个函数例子
   - 计算细节
   - 库函数编写方法



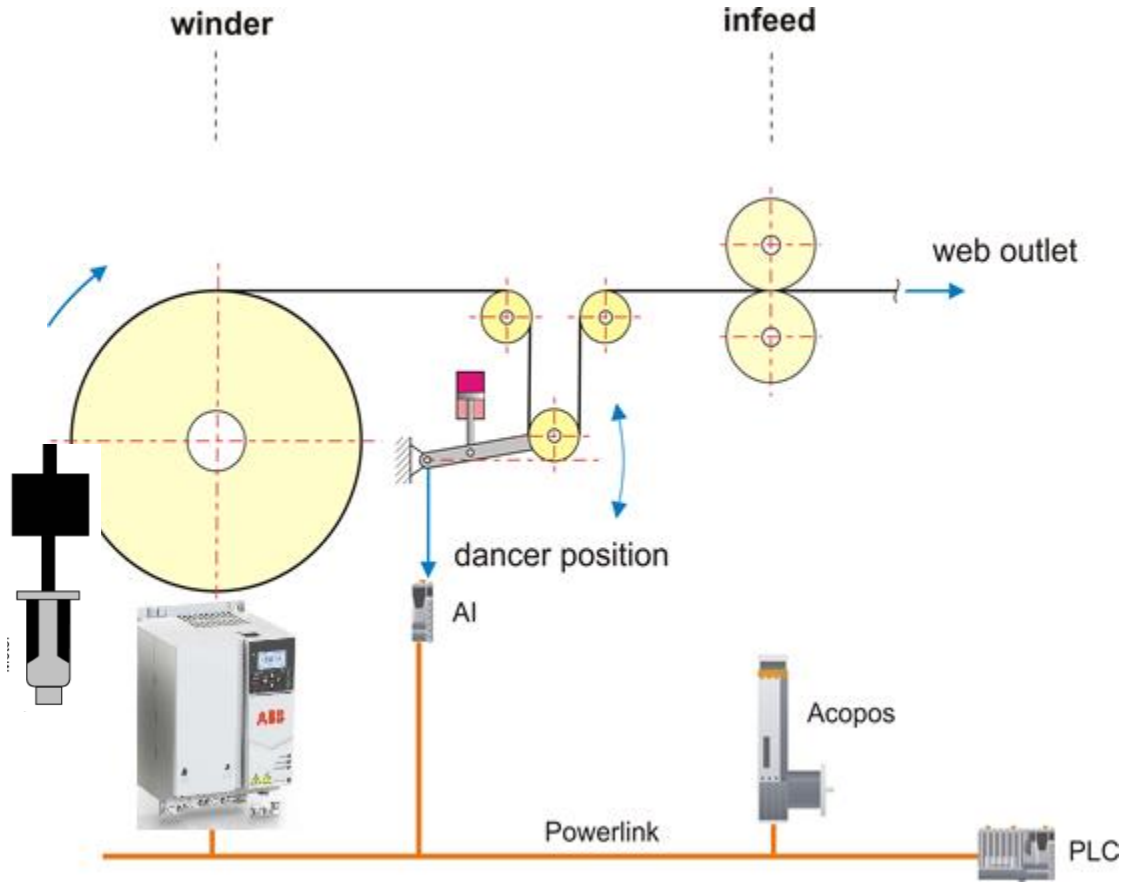
印刷机收放卷张力控制思路及计算方法

收放卷基本结构

- 放卷，收卷 由变频器驱动异步电机
  - 电机有编码器反馈接到变频器（方向都有可能）
  - 电机通过减速齿轮驱动皮带带动料卷（减速比非整除界面输入）
- 摆臂通过模拟量输入给PLC（中位可能偏移，方向可能反）

- ABB变频器的型号为ACS380，带外接的编码器作位置反馈，ABB变频器可以同时返回圈数值+单圈内的位置值，一圈内的位置值为1*10^9。大约10-20ms给出一个新位置。

- 控制目标 – 控制变频器速度使摆臂居中



印刷机收放卷张力控制思路及计算方法

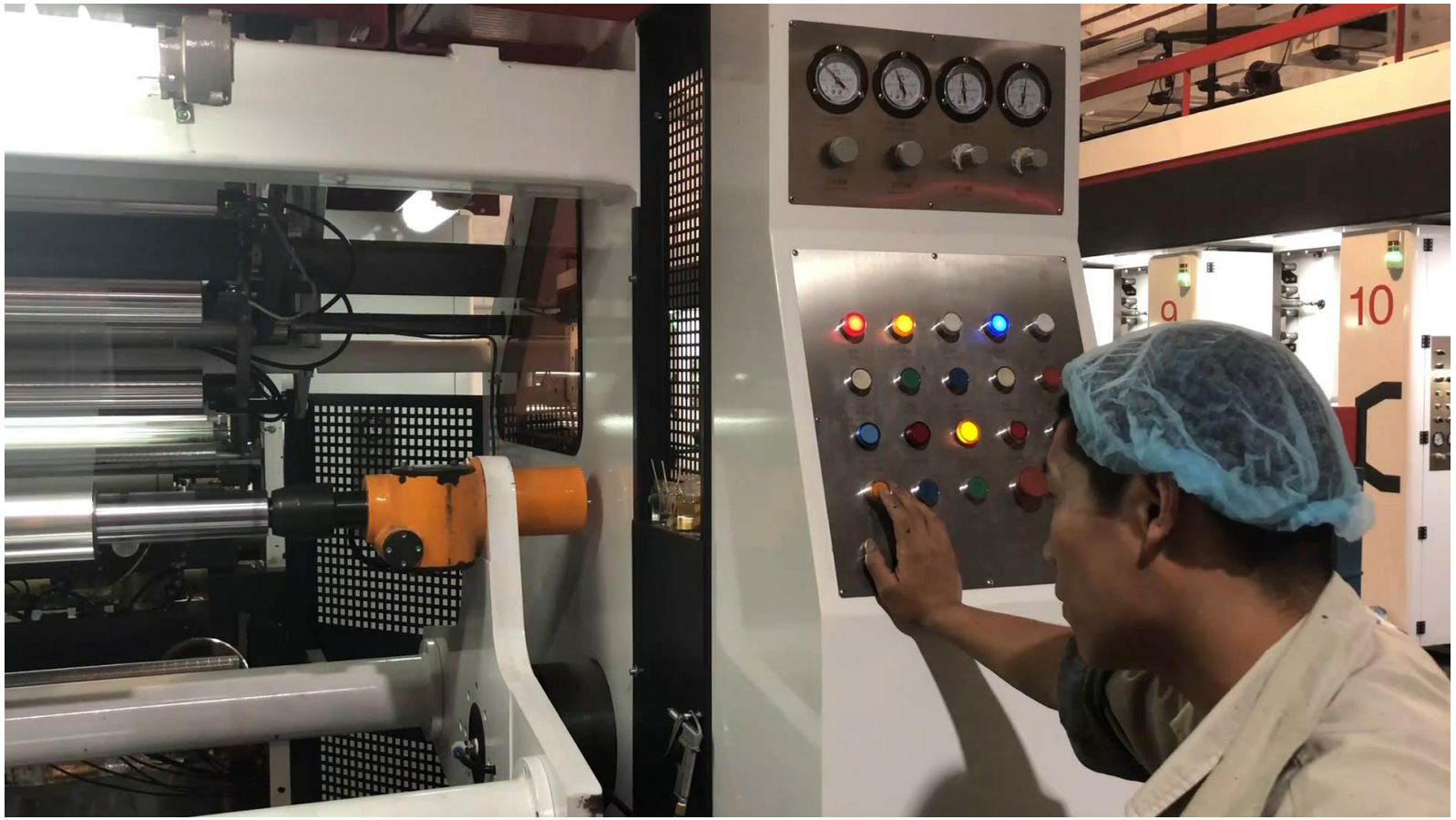
收放卷基本结构

印刷机收放卷张力控制思路及计算方法

收放卷预驱动视频

关键指标

- 旧轴剩余材料
- 新旧轴切换材料重叠长度



印刷机收放卷张力控制思路及计算方法

裁切动作视频

关键指标

- 旧轴剩余材料
- 新旧轴切换材料重叠长度

印刷机收放卷张力控制

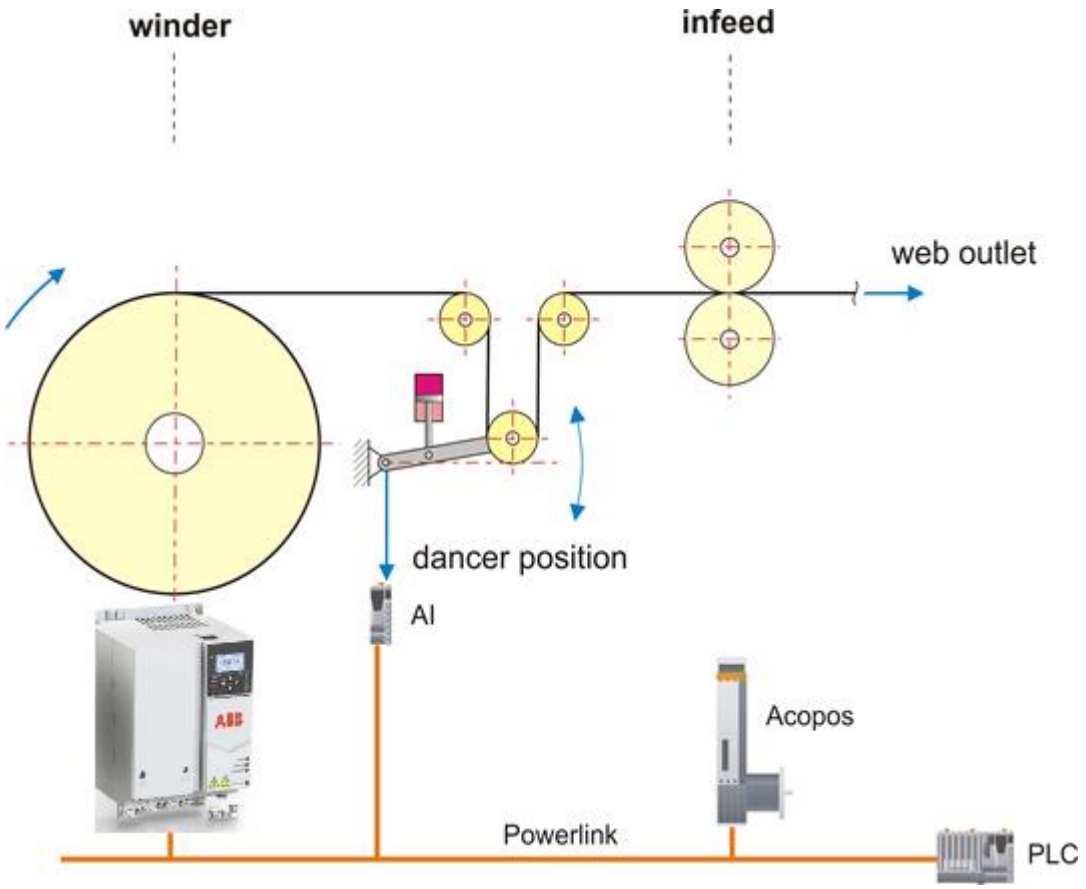
收放卷基本工艺要求

根据以下工艺要求：

- 卷径没有传感器，需要计算。
- 初始卷径自动模式时： 张力投入后，变频器拉紧摆臂，摆臂居中后根据吸收料长计算出初始卷径。
- 初始卷径可能不准，料膜联动运行（开机后），卷径进入计算模式，此时可能发生几次跳变。
- 换料裁切时如果知道胶带位置有助于减少接头长度。减少接头长度可以降低张力干扰，大大降低裁切废品（新旧轴切换材料重叠长度）
- 剩余时间最好可以计算便于实现全自动裁切(解决人工操作 旧轴剩余材料)

有几个需要考虑的点：

- 卷径突变怎么办?
- PID 输出给线速度还是角速度
- PID 是否分段
- 如果计算胶带位置，减速比不是整数切不能整除
- 在跨越一个设定相位给个输出
- 变频器位置跳的有点厉害



印刷机收放卷张力控制
卷径计算的两种思路

- 无位置反馈时，根据pid输出，缓慢增加或者减小卷径。
  - 步长需要测试，变化过快可能会造成抖动。
  - 卷径不会突变，但是初始卷径不准时，过渡时间会偏长。

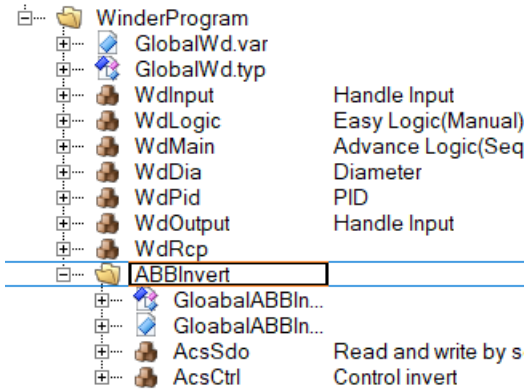
- 有位置反馈时，可以根据走料长度和收放卷转动圈数直接计算。
  - 计算速度快慢可控，和真实卷径偏差比较小
  - 计算直径和PID无关
  - 需要考虑摆臂吸收的料长
  - 需要适当滤波

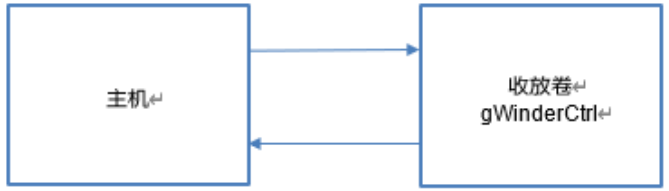
主要控制原理及思路
确定程序任务分配

强分割 独立任务

- 1个全局结构体与主机交互（如需增加全局结构体必须少）
- 参数配方自己保存

- 最终任务分配如下图

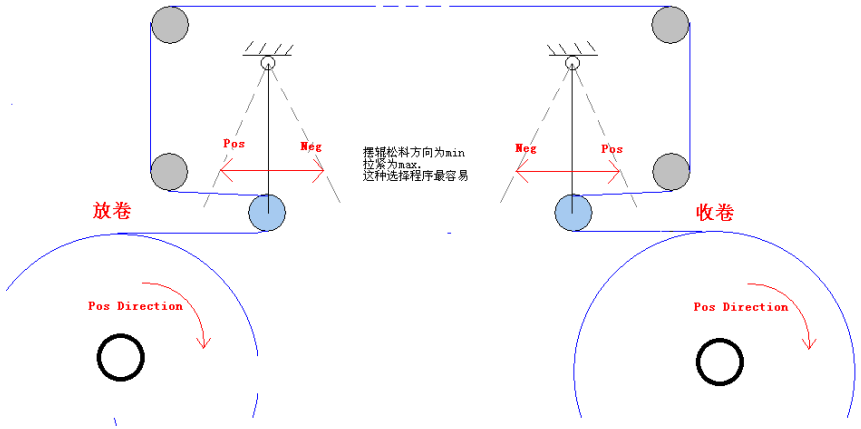




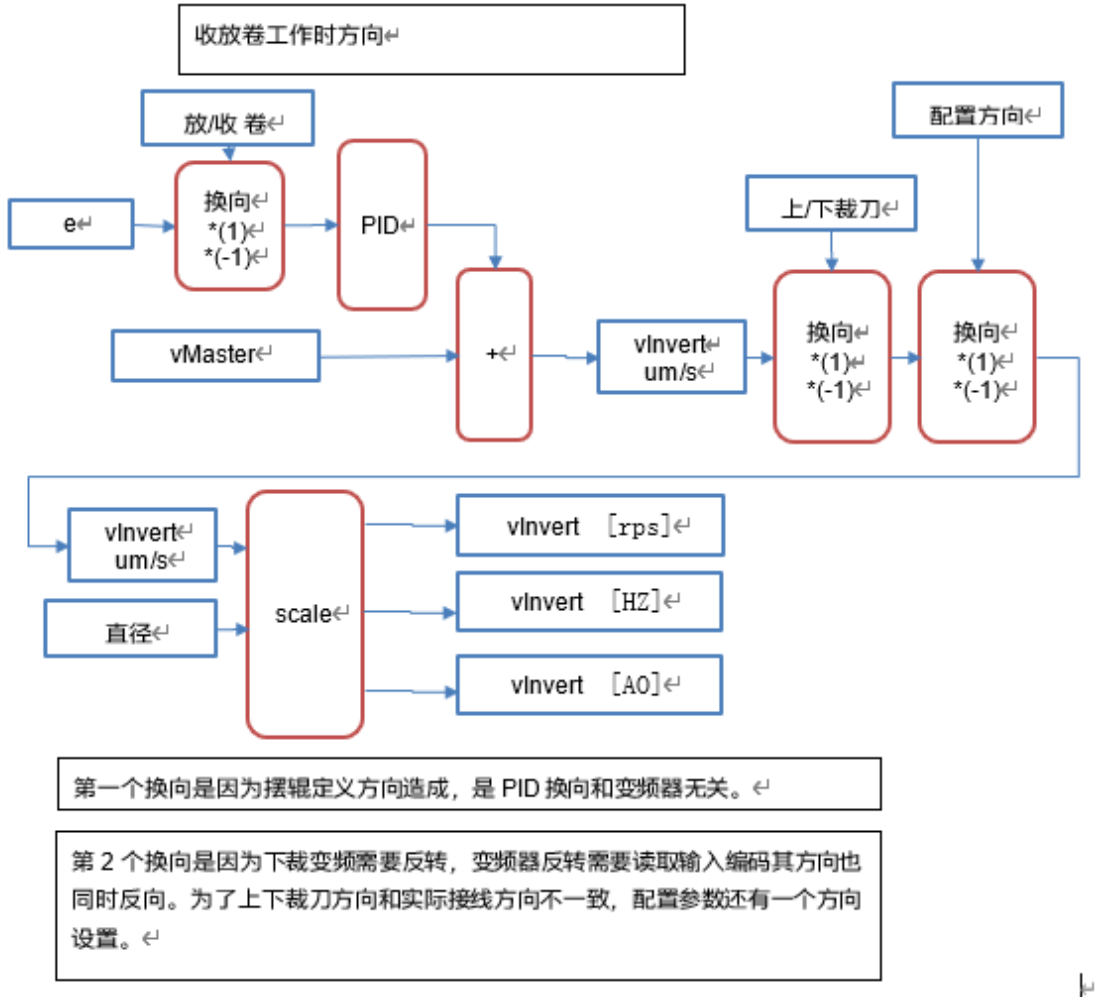
主机与收放卷 交互数据		
主机 –〉收放卷		
变量名	类型	说明
		张力投入
		主机状态 – 停机，空转，联动。
		膜料线速度
		引入压辊压
		引出压辊压
		起始张力
		结束张力
		起始张力对应直径
		结束张力对应直径
收放卷 –〉主机		
	Int	收卷当前张力
		放卷工作轴
		放卷工作轴直径
		收卷工作轴
	Int	收卷直径 mm
	Int	变频器报警
	Int	报警轴
	Int	报警号
	Bool	放卷裁切信号

主要控制原理及思路

开发前准备 方向定义



- 定义好方向后所有函数按照规定方向进行输入输出编写



主要控制原理及思路

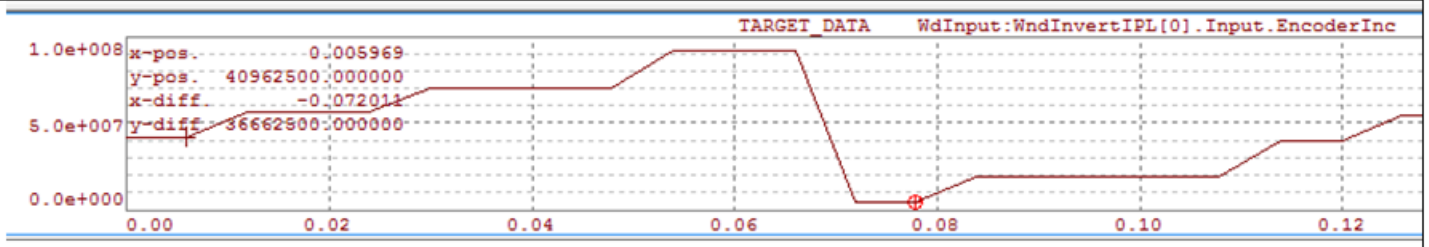
WdFbIPL编码器处理（标定+差值）

- 变频器给的位置信息有两个变量，一个是圈数，一个是当前圈的位置0-100000000
- 变频器的位置Iomapping 刷新速度大约10ms，有时可能20ms。

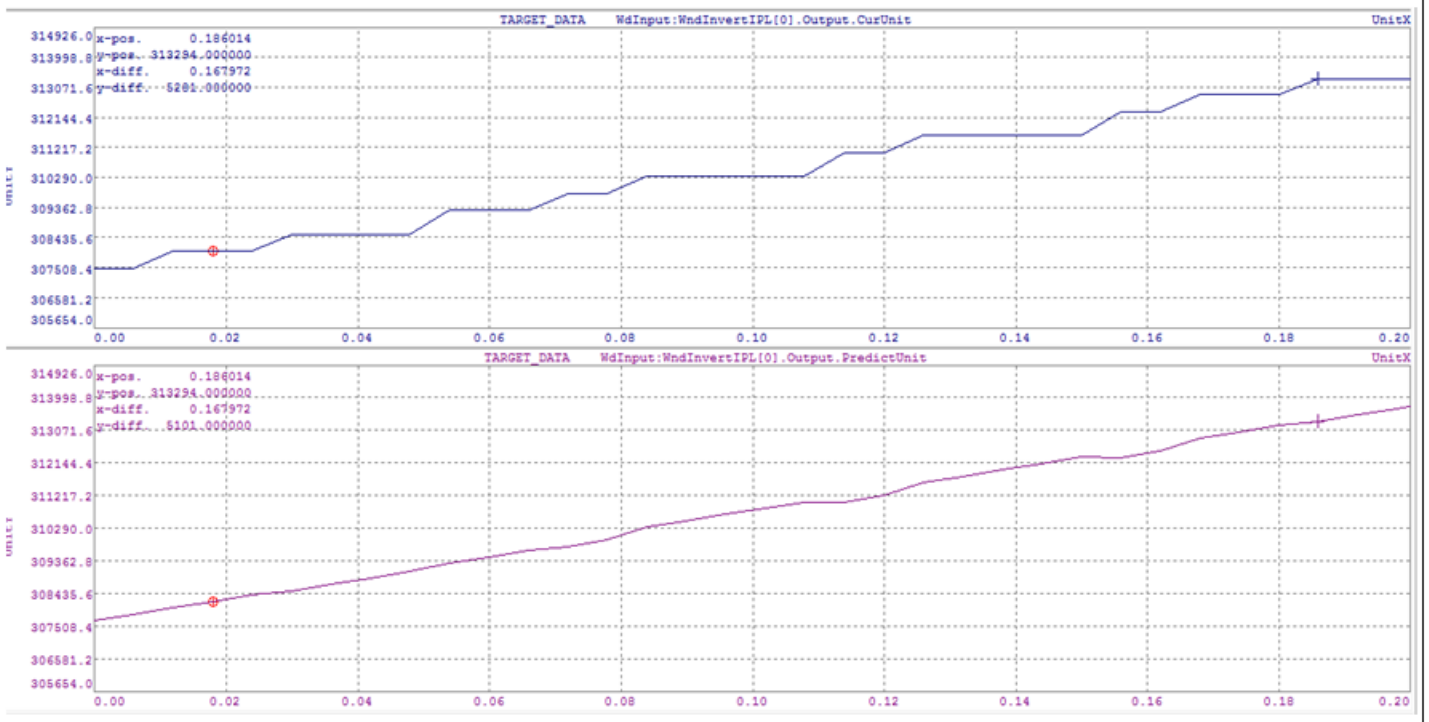
- 所以第一步设计一个函数具有标定及差值功能，方便其他函数处理
  - 标定是重新标定每圈unit
  - 差值为了保证程序每个周期计算都有新的数据，有利于平滑的计算卷径

变频器 IO pos 0-100000000 (0-1 亿)

1 变频器位置更新不确定 1-4 个任务周期 （6ms）



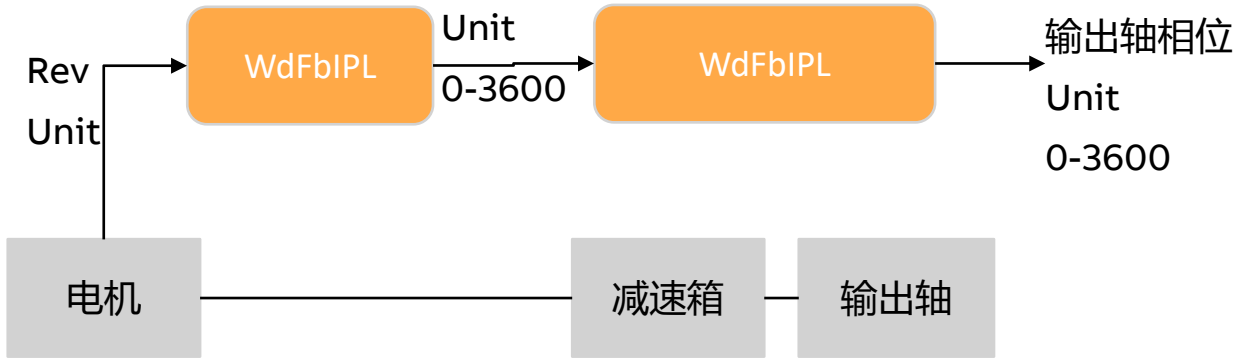
2 位置插值函数结果 – 比不插值稳定 速度 500rpm



主要控制原理及思路

WdFbEncToRev过减速机后信号处理（非整除减速比 + 设定相位的跨相位信号输出）

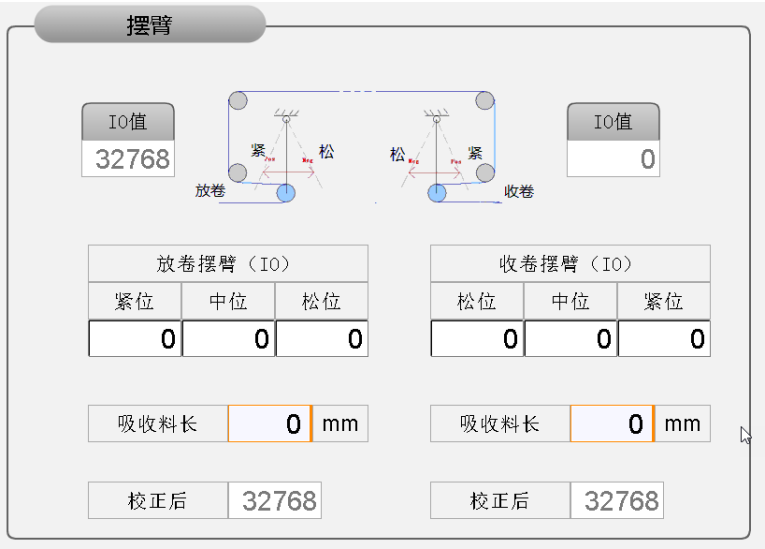
- 编码器经过差值后，仍然是电机位置。
- 为了实现连续转一段时间不丢失准确相位，需要处理经过非整除减速比的情况。
- 裁切时需要再一个设定相位输出信号，所以这个功能也在此函数中进行考虑。
- 输出轴相位方便进行卷径计算



主要控制原理及思路

WdFbDancer 摆臂信号处理（滤波 + 偏移 + scale ）

- 信号滤波 LCRMovAvgFlt
- 中位偏移
- 速度计算
- Scale To Percent
- 吸收料长



I/O	parameter	datatype	description
In	PosLimit	Dint	摆辊正极限值
In	NegLimit	Dint	摆辊负向极限值
In	MiddleValue	Int	摆辊中位值
In	MaterialLen	Dint	吸收料长，摆辊从正极限到负极限之间的料长
In	AnalogInput	Int	摆辊当前值
In	Enable	Bool	使能
Out	DancerValue	Dint	经过滤波处理后的摆辊位置值
Out	InPosMiddleAndStable	Bool	摆辊在中位并且稳定
Out	CurMaterialLength	Dint	吸收料长
Out	DancerSpeed	Dint	摆辊当前速度
Out	InPosLimit	Bool	摆辊在正限位
Out	InPosMiddle	Bool	摆辊在中位
Out	InNegLimit	Bool	摆辊在负限位
Out	StandStill	Bool	摆辊稳定，通过摆辊的速度进行判断的
Out	DancerPercent	Int	摆辊的位置百分比，从负极限到正极限对应-100—100；

主要控制原理及思路

PID的输入和输出定义

PID 的输入和输出定义：

输入 - 偏差为摆臂误差

输出选择：

1.可以考虑加在线速度上，这样后续输出时需要通过卷径转成角速度。

2. 可以选择加在角速度上，这样不用通过卷径转化。但此方式会造成相同偏差时，不同直径时输出的响应不一致。容易引起误解，虽然也可以后续通过增加直径分段来处理，但这样容易混淆概念，不易理解（此方式如果不分段，适应性极大降低）。

较好的方式是选择 1.以确保相同条件PID结果一致。

主要控制原理及思路

PID 什么时候分段分段

对于一个阶跃响应，我们可以调整一组PID参数。

在收放卷系统，卷径发生变化后惯量会极大的发生变化。（如果大小卷直径相差5倍，惯量和直径是4次方关系，会产生625倍关系）

负载以及其惯量发生很大变化会造成阶跃响应发生较大变化，阶跃响应时间加长，容易震荡。

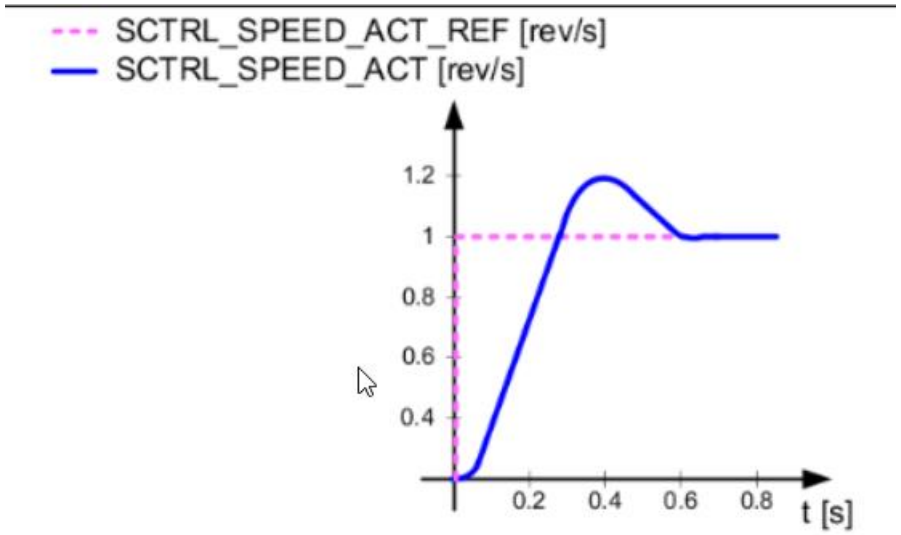
为了保证控制稳定，可以采用以下两种办法：

1.加强系统响应刚性，尽量维持阶跃响应曲线。如果是伺服，加强三环。

2.分段PID，减少闭环控制，防止震荡。此时系统响应会变慢。

收放卷这里是变频器控制，采用方法2。

如果是伺服控制可以考虑1或1,2结合。



主要控制原理及思路
卷径突变处理

输出的计算过程 角速度=线速度/周长

vSpeedInvertRPS = (vMasterLineSpeed+vPidLineSpeed)/(PI*Dia)

当卷径Dia从Dia_0变化为Dia_1时，为了保证输出vSpeedInvertRPS不变对vPidLineSpeed进行同时跳变处理,使得下式相等。

(vMasterLineSpeed+vPidLineSpeed_0)/(PI*Dia_0) = (vMasterLineSpeed+vPidLineSpeed_1)/(PI*Dia_1)

vPidLineSpeed_1 = vMasterLineSpeed*(Dia_1- Dia_0)/Dia_0 + vPidLineSpeed_0*Dia_1/Dia_0

将pid输出vPidLineSpeed_1设置为vMasterLineSpeed*(Dia_1- Dia_0)/Dia_0 + vPidLineSpeed_0*Dia_1/Dia_0 可以保持输出不变

这样需要在设计PID的时候考虑一个命令可以改变pid输出，由于比例部分是一直计算，所以修改积分部分比较方便。

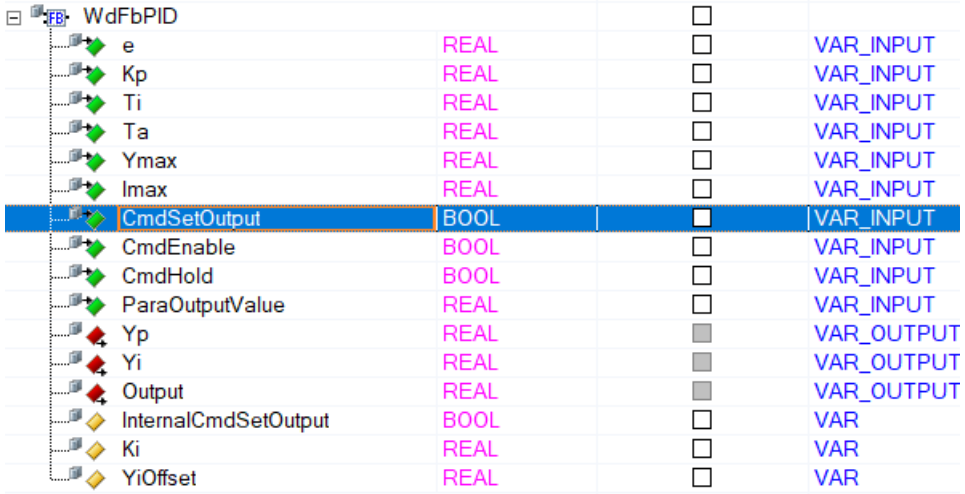
（*单独增加一个变量也是可以的。）

主要控制原理及思路

WdFbPID PID函数（有积分突变接口）

需要考虑的点：

- 包含一个Hold命令
- 包含一个设定总输出命令。（内部修改积分部分）



主要控制原理及思路

WdFbDiaThick 计算收放卷的卷径，料膜厚度，剩余时间

需要考虑的点：

- 基于位置的直径计算
- 直径滤波 （均值滤波）
- 厚度计算
- 剩余时间计算

- 后续还可以考虑的点
- 同时基于pid进行计算卷径
- 外部干扰时使用观测输出
- ……

I/O	parameter	datatype	description
In	WdFbdiaThcikCmd_typ		
	Enable	Bool	功能块使能
	InitThickness	Bool	初始化料膜厚度，厚度在参数中设置
	Hold	Bool	暂停计算
In	WdFbDiaThcikInput_typ		
	MasterPosition	Dint	[um]主机/主轴位置（印刷机机上通常是牵引），单位一定是 um
	MasterSpeed	Dint	[um/s]主机/主轴速度
	WdAxisUnit	Dint	收放卷的角度位置值，经过功能块 WdFbEncToRev 处理后的角度位置累加值
	WdDancerPos	Dint	摆辊的位置，进过功能块 WdFbDancer 滤波以后的摆辊位置值
In	WdFbDiaThcikPara_typ		
	AxisUnitPerCycle	Dint	收放卷每圈的位置值，和功能块的 WdFbEncToRev 的参数一致，当前程序的值为 3600
	AxisDirection	Dint	0—增长 1 – 减少
	MinChangeRollerDia	Dint	自动裁切时设置的裁切直径
	ChangeABInitDia	Dint	换卷时的新轴初始卷径，如果该值为 0，功能块会也会自动计算卷径，但是计算的时间较长
	InitThickness	Dint	料膜厚度，可以手动设定改制
	WInderType	Dint	收放卷类型，0-放卷 1-收卷
	MinDiameter	Dint	最小卷径[um]，单位要转成 um
	DancerWebLength	Dint	摆辊正负极限之间的料长 单位转为 um
	DancerRange	Dint	正负极限模拟量范围 一般情况下为正极限值–负极限值
Out	WdFbDiaThickOutput_typ		
	DancerStable	Bool	摆辊稳定状态
	ActDiameter	Dint	实际卷径，基于编码器位置计算的结果 单位[um]
	MaterialThick	Dint	计算的材料厚度 单位[um]
	TimeToMinDiaMs	Dint	剩余自动裁切的时间单位[ms]，用于程序内部计算
	TimeToMinDiaMinute	Dint	剩余自动裁切的分钟[minute]，用于程序界面显示
	TimeToMinDiaSecond	Dint	剩余自动裁切的秒数[s]，用于程序界面显示
	Error	Dint	报错，参数设置错误时可能会有报错
	UpdateDiamter	Bool	卷径更新标志位
	UpdateLeftTime	Bool	剩余时间更新标志位
	ThickInitReady	Bool	材料厚度计算完成标志位

函数举例

WdFbIPL编码器输入处理（标定+差值）

接口设计

- 命令结构体
- 输入结构体
- 参数结构体
- 内部数据结构体
- 输出结构体

Name	Type	& Ref...	Repli...	Value	Description [1]
WdFbIPLCmd_typ			☑		
Enable	BOOL	☐	☑		
WdFbIPLInput_typ			☑		
EncoderInc	DINT	☐	☑		编码器脉冲 0 -- EncoderRound For example 0-1000000
EncoderRound	DINT	☐	☑		编码器圈数 需要dint 类型范围-2147483648 _2147483647
WdFbIPLPara_typ			☑		
EncoderIncRev	DINT	☐	☑		编码器每圈脉冲inc
UnitRev	DINT	☐	☑		输出每圈个数unit (与编码器一圈对一圈)
Mode	USINT	☐	☑		0 - 使用脉冲和位置两个输入，允许脉冲跳变 1 - 使用脉冲输入，
WdFbIPLInternal_typ			☑		
FunRTInfo	RTInfo	☐	☑		
CycT	DINT	☐	☑		
EnableOld	BOOL	☐	☑		
EncodrIncOld	DINT	☐	☑		
EncodrIncDelta	DINT	☐	☑		
EncodrIncRoundOld	DINT	☐	☑		
EncodrIncRoundDelta	DINT	☐	☑		
EncodrIncDeltaTotal	DINT	☐	☑		
CycCount	DINT	☐	☑		
CycTime	DINT	☐	☑		
CycSpeed	REAL	☐	☑		
SignalPredict	DINT	☐	☑		
UnitHalf	DINT	☐	☑		
UnitScaled	DINT	☐	☑		
UnitScaledOld	DINT	☐	☑		
UnitDeltaTotal	DINT	☐	☑		
UnitDelta	DINT	☐	☑		
UnitTolerance	DINT	☐	☑		
UnitSum	DINT	☐	☑		
UnitSumOld	DINT	☐	☑		
ScaleFactor	REAL	☐	☑		
WdFbIPLOutput_typ			☑		
CurUnit	DINT	☐	☑		[unit]类型范围-2147483648 _2147483647 输入刷新时输出刷新
PredictUnit	DINT	☐	☑		[unit]类型范围-2147483648 _2147483647 每个周期刷新
Error	DINT	☐	☑		如果必须要的参数设置错误，有报警

函数举例

WdFbIPL编码器输入处理

使能处理

- Enable == 0 时，清输出及EnableOld
- Enable上升沿，清Internal结构体，检查参数范围，默认设置，部分内部参数计算，部分内部Old变量初始化。
- 正文程序需要的内部real类型系数在上升沿处理，提高效率
- 如有错误，返回

RTInfo

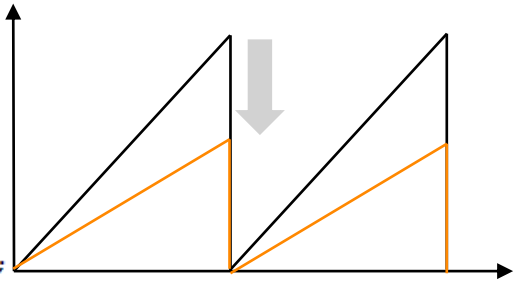
- 获取任务周期，方便代码计时。
- 方便使用者可以放在任何任务周期中调用

```
/*****************************************************************主使能*****************
if(inst->Cmd.Enable == 0)
{
    /*Clear output*/
    inst->Output.CurUnit = 0;
    inst->Output.PredictUnit = 0;
    inst->Output.Error = 0;

    inst->Internal.EnableOld = 0;
    return;
}
/****************初始化 执行一次****************/
if(inst->Cmd.Enable == 1 && inst->Internal.EnableOld == 0)
{
    /*Clear buffer*/
    memset(&inst->Internal,0,sizeof(inst->Internal));
    inst->Internal.EnableOld = 1;
    /*Get cycle time in us*/
    inst->Internal.FunRTInfo.enable = 1;
    RTInfo(&inst->Internal.FunRTInfo);
    inst->Internal.CycT = inst->Internal.FunRTInfo.cycle_time;

    /***************默认参数处理*****************/
    if(inst->Para.EncoderIncRev == 0)       inst->Output.Error = 1; /*防止除零错误*/
    if(inst->Para.UnitRev == 0)             inst->Output.Error = 1; /*防止除零错误*/

    inst->Internal.UnitHalf = inst->Para.UnitRev/2;
    inst->Internal.ScaleFactor = (REAL)inst->Para.UnitRev/inst->Para.EncoderIncRev;
    /***************Old处理*****************/
    inst->Internal.EncodrIncRoundOld = inst->Input.EncoderRound;
    inst->Internal.UnitScaledOld = inst->Internal.UnitScaled;
    inst->Internal.UnitSumOld = inst->Internal.UnitSum;

    inst->Internal.CycCount = 0;
    inst->Internal.CycTime = 0;
    return;
}
if(inst->Output.Error == 1) return;
```

函数举例

WdFbIPL编码器输入处理

- 单圈Inc转化为单圈Unit
  - DINT = REAL*DINT
  - 无积累误差，有浮点误差
- 计算增量（经常需要用到的计算）
  - DINT_Delta = DINT_A – DINT_A_OLD
- 两种模式
  - 模式1：输入为圈数+单圈位置
  - 模式2：输入为DINT全范围



```
inst->Internal.UnitScaled = inst->Internal.ScaleFactor * inst->Input.EncoderInc;
/*计算增量*/
inst->Internal.UnitDelta = inst->Internal.UnitScaled - inst->Internal.UnitScaledOld;
inst->Internal.UnitScaledOld = inst->Internal.UnitScaled;

inst->Internal.EncodrIncRoundDelta = inst->Input.EncoderRound - inst->Internal.EncodrIncRoundOld;
inst->Internal.EncodrIncRoundOld = inst->Input.EncoderRound;
/*根据模式计算总增量*/
if(inst->Para.Mode == IPL_MODE_USE_ROUND)
{
    inst->Internal.UnitDeltaTotal = inst->Internal.EncodrIncRoundDelta * inst->Para.UnitRev + inst->Internal.UnitDelta;
}
else
{
    if(inst->Internal.UnitDelta > inst->Internal.UnitHalf)
    {
        inst->Internal.UnitDeltaTotal = inst->Internal.UnitDelta - inst->Para.UnitRev;
    }
    else if(inst->Internal.UnitDelta < -inst->Internal.UnitHalf)
    {
        inst->Internal.UnitDeltaTotal = inst->Internal.UnitDelta + inst->Para.UnitRev;
    }
    else
    {
        inst->Internal.UnitDeltaTotal = inst->Internal.UnitDelta;
    }
}
```

函数举例

WdFbIPL编码器输入处理

分步计算及输出

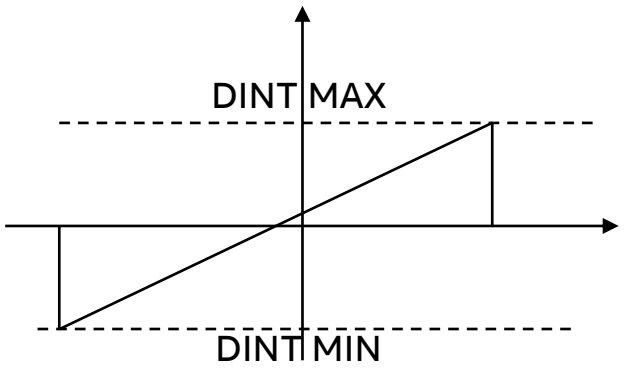
- 计算总Unit
- 计算速度，每周期增量
- 简单预测
- 输出（累加型）
  - CurUnit 未经过差值
  - PredictUnit经过差值

```
/*****1 - unit sum cal***********/
inst->Internal.UnitSum += inst->Internal.UnitDeltaTotal;

/*****2 - unit speed cal***********/
inst->Internal.CycCount += inst->Internal.UnitDeltaTotal;
inst->Internal.CycTime ++;
if(inst->Internal.CycCount > inst->Para.UnitRev || inst->Internal.CycCount < -inst->Para.UnitRev || inst->Internal.CycTime > 100)
{
    inst->Internal.CycSpeed = inst->Internal.CycCount / inst->Internal.CycTime;
    inst->Internal.CycCount = 0;
    inst->Internal.CycTime = 0;
}
/*****3 - unit predict cal*********/
if(inst->Internal.CycSpeed == 0)    inst->Internal.SignalPredict = inst->Internal.UnitSum;
else                                inst->Internal.SignalPredict = inst->Internal.SignalPredict + inst->Internal.CycSpeed;
//inst->Internal.UnitTolerance = inst->Internal.UnitSum - inst->Internal.SignalPredict;
if(inst->Internal.UnitSum != inst->Internal.UnitSumOld)
{
    inst->Internal.SignalPredict = inst->Internal.UnitSum;
}
inst->Internal.UnitSumOld = inst->Internal.UnitSum;

/*****4 - output handle*********/
inst->Output.CurUnit = inst->Internal.UnitSum;
inst->Output.PredictUnit = inst->Internal.SignalPredict;
```

函数举例
WdFbIPL编码器输入处理

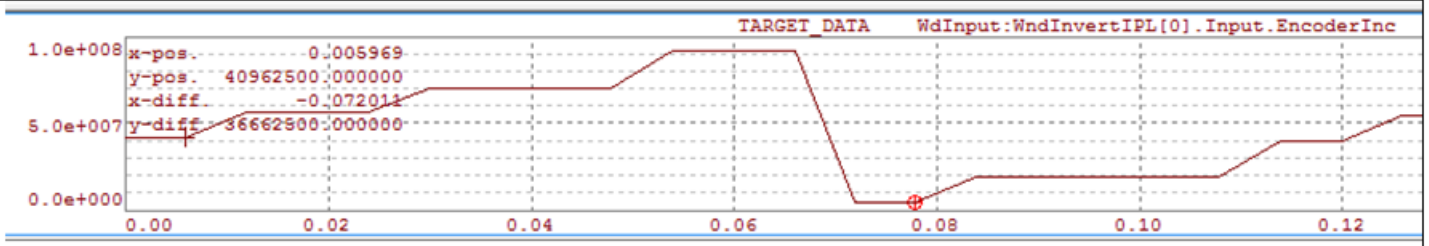
分步计算及输出

- 计算总Unit
- 计算速度，每周期增量
- 简单预测
- 输出（累加型防止丢圈）
  - CurUnit 未经过差值
  - PredictUnit经过差值

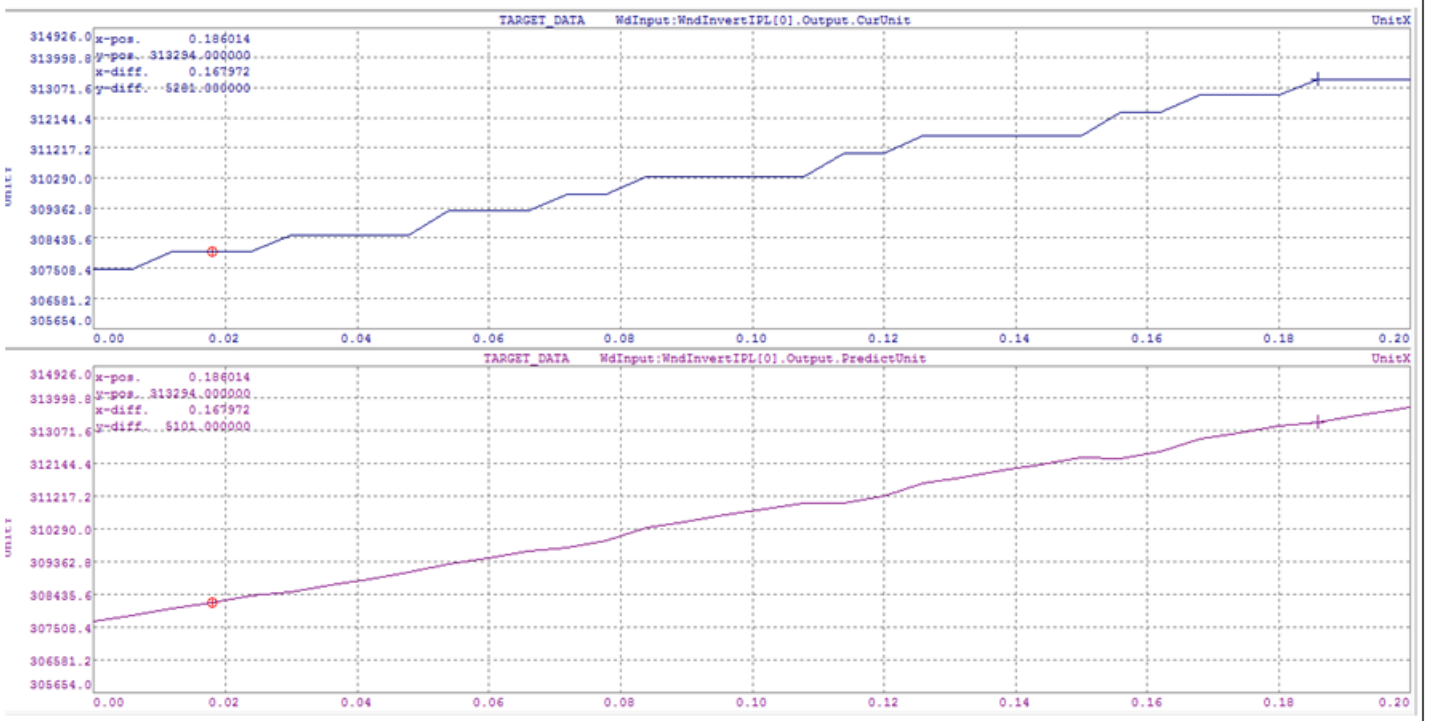


变频器 IO pos 0-100000000 (0-1 亿)

1 变频器位置更新不确定 1-4 个任务周期 （6ms）



2 位置插值函数结果 – 比不插值稳定 速度 500rpm



函数举例

WdFbEncToRev过减速机后信号处理

使能处理

- 为零清输出

```
/*TODO: Add your code here*/
if(inst->Cmd.Enable == 0)
{
    inst->Cmd.SetZero = 0;
    /*Clear output*/
    inst->Internal.EncodrIncOld = inst->Input.EncoderIn
    inst->Internal.EnableOld = 0;

    inst->Output.CurAngle = 0;
    inst->Internal.UnitSum = 0;
    inst->Internal.EncoderSum = 0;
    return;
}
```

- 上升沿清内部数据
- 上升沿获取任务周期
- 上升沿默认参数处理
- 上升沿内部系数计算
  - 计算输入输出unit转化系数 IncToUnitRatio(REAL)
  - 计算减速比系数GearRatio (REAL)
  - 计算DINT总转化系数EncoderMinRes(用来比较)
  - 计算LREAL总转化系数TotalRatio（用来计算）
  - 计算输入数值范围EncRange及半圈范围EncoderIncHalf
  - 计算输出数值范围UnitRange及半圈范围Unit180
- 上升沿旧值初始化

- 目标：Inc * GearIn 转化到 Unit*GearOut

```
/****************初始化 执行一次****************/
if(inst->Cmd.Enable == 1 && inst->Internal.EnableOld == 0)
{
    /*Clear buffer*/
    memset(&inst->Internal,0,sizeof(inst->Internal));
    inst->Internal.EnableOld = 1;
    /*Get cycle time in us*/
    inst->Internal.FunRTInfo.enable = 1;
    RTInfo(&inst->Internal.FunRTInfo);
    inst->Internal.CycT = inst->Internal.FunRTInfo.cycle_time;

    /***************默认参数处理*****************/
    if(inst->Para.GearIn  == 0)                 inst->Output.Error = 1; /*防止除零错误*/
    if(inst->Para.GearOut == 0)                 inst->Output.Error = 1; /*防止除零错误*/
    if(inst->Para.EncoderIncMotorRev == 0)      inst->Output.Error = 1; /*防止除零错误*/
    if(inst->Para.UnitWebRev == 0)              inst->Output.Error = 1; /*防止除零错误*/

    if(inst->Output.Error == 1) return;

    inst->Internal.IncToUnitRatio = (REAL)inst->Para.EncoderIncMotorRev/inst->Para.UnitWebRev;
    inst->Internal.GearRatio = (REAL)inst->Para.GearIn / inst->Para.GearOut;
    inst->Internal.EncoderMinRes = inst->Internal.IncToUnitRatio * inst->Internal.GearRatio;
    if(inst->Internal.EncoderMinRes < 1) inst->Internal.EncoderMinRes = 1;
    inst->Internal.TotalRatio = inst->Internal.GearRatio * inst->Internal.IncToUnitRatio;

    inst->Internal.EncRange = inst->Para.EncoderIncMotorRev * inst->Para.GearIn;
    inst->Internal.UnitRange = inst->Para.UnitWebRev * inst->Para.GearOut;

    inst->Internal.EncodrIncHalf = inst->Para.EncoderIncMotorRev/2;
    inst->Internal.Unit180 = inst->Para.UnitWebRev/2;
    /***************Old处理*****************/
    inst->Internal.EncodrIncOld = inst->Input.EncoderInc;
    inst->Internal.UnitSumOld = inst->Internal.UnitSum;
    //inst->Internal.SignalPredictOld = inst->Internal.SignalPredict;
}
if(inst->Output.Error == 1) return;
```

函数举例

WdFbEncToRev过减速机后信号处理

- 零相位确认命令接口
- 根据输入方向计算增量（这里这样设计是因为输出方向与上下裁刀有关，程序正向编码器可能反向运行）
- 计算总累加输入量EncoderSum（可以理解为余）
- 当大于最小比较单位进行一次计算
  - 整数部分UnitDelta取出,小数部分没有取
  - 余数EncoderSum(LREAL)按照真实减

```
if(inst->Cmd.SetZero == 1)
{
    inst->Cmd.SetZero = 0;
    inst->Internal.EncodrIncOld = inst->Input.EncoderInc;
    inst->Internal.EncoderSum = 0;
    inst->Internal.UnitSum = 0;
}
/*Encoder handle*/
if(inst->Para.Direction == 0)   inst->Internal.EncodrIncDelta = inst->Input.EncoderInc - inst->Internal.EncodrIncOld;
else                            inst->Internal.EncodrIncDelta = -inst->Input.EncoderInc + inst->Internal.EncodrIncOld;
/ if(inst->Internal.EncodrIncDelta > inst->Internal.EncodrIncHalf)
/ {
/     inst->Internal.EncodrIncDelta = -inst->Para.EncoderIncMotorRev + inst->Internal.EncodrIncDelta;
/ }
/ if(inst->Internal.EncodrIncDelta < -inst->Internal.EncodrIncHalf)
/ {
/     inst->Internal.EncodrIncDelta = inst->Para.EncoderIncMotorRev + inst->Internal.EncodrIncDelta;
/ }
inst->Internal.EncodrIncOld = inst->Input.EncoderInc;

inst->Internal.EncoderSum += inst->Internal.EncodrIncDelta;

if(inst->Internal.EncoderSum > inst->Internal.EncoderMinRes || inst->Internal.EncoderSum < -inst->Internal.EncoderMinRes)
{
    inst->Internal.UnitDelta = inst->Internal.EncoderSum/inst->Internal.TotalRatio;
    inst->Internal.EncoderSum -= inst->Internal.UnitDelta * inst->Internal.TotalRatio;
}
else
{
    inst->Internal.UnitDelta = 0;
}
```

函数举例

WdFbEncToRev过减速机后信号处理

关于上面两行比较难以理解

举一个计算例子，减速比3.3333333（10/3)

Step1：电机编码器增加4 时，余数4.00000，负载侧增加1，余数重新计算0.666666

Step2：电机编码器增加4 时，余数4.66666，负载侧增加1，余数重新计算1.3333333

Step3：电机编码器增加3 时，余数4.333333，负载侧增加1，余数重新计算 1

。。。。。。

右图为测试结果：

- 转动1000圈
- 余数$8.88*10^{-16}$

Name	Type	Scope	Force	Value
EncoderSum	LREAL	local		8.8817841970012523E-16
TotalRatio	LREAL	local		3.3333333333333308
UnitDelta	DINT	local		0
outputUnit	DINT	local		0
outputUnitSum	DINT	local		300000
timesCont	DINT	local		1000000

- 实际Trace可以看出仍然后一点点real余数，如果想完全处里这块可以在输入超过其范围Inc * GearIn时将余数清零。
- 此函数没有对这部分处理。原因是高速几天也转不到1个unit误差。
- 收放卷功能预备裁切通常1分钟以内相位准确就足够。


B&R
A member of the ABB Group