



深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

EM03DE-E4 编码器模块

用户手册

Version 2.3

2020年7月

©Copyright 2020Leadshine Technology Co., Ltd.

All Rights Reserved.

本手册版权归深圳市雷赛控制技术有限公司所有，未经本公司书面许可，任何人不得翻印、翻译和抄袭本手册中的任何内容。

本手册中的信息资料仅供参考。由于改进设计和功能等原因，雷赛公司保留对本资料的最终解释权，内容如有更改，恕不另行通知。

修改记录

修改日期	版本	修改说明		拟制人
		原来内容	更新内容	
20190720	V2.1		增加模块连接 BASIC 控制器的例程和连接控制卡例程 增加对象字典数据有效范围	产品部
20190920	V2.2		修改成使用 PDO 读取编码器	产品部
20200720	V2.3		增加 FIFO 说明	产品部



调试机器要注意安全！用户必须在机器中设计有效的安全保护装置，在软件中加入出错处理程序。否则所造成的损失，雷赛公司没有义务或责任负责。

目录

第 1 章 产品概述.....	4
1.1 产品简介.....	4
1.2 产品特点.....	4
1.3 技术规格.....	4
1.4 安装使用.....	6
第 2 章 产品外观及硬件接线.....	7
2.1 产品外观.....	7
2.2 接口分布及引脚定义.....	8
2.2.1 J3 电源接口.....	9
2.2.2 IN、OUT 接口定义.....	9
2.2.3 J201 接口定义.....	10
2.2.4 S301 接口定义.....	10
2.3 接口电路.....	11
2.3.1 编码器信号输入接口.....	11
2.3.1.1 可接收的编码器信号类型.....	11
2.3.1.2 编码器信号输入接口电路.....	12
2.3.2 高速位置锁存输入信号接口.....	13
2.3.3 高速位置比较输出信号接口.....	13
第 3 章 指示灯定义及说明.....	15
3.1 指示灯定义.....	15
3.2 指示灯闪烁规则.....	15
3.3 指示灯状态.....	16
第 4 章 对象字典.....	18
4.1 设备参数.....	18
4.2 通用参数.....	18
4.2.1 编码器参数设置.....	18
4.2.2 编码器值读取.....	19
4.2.3 锁存器参数设置.....	20
4.2.4 锁存器状态读取.....	21
4.2.5 比较器参数设置.....	24
4.2.6 比较器状态读取.....	27
4.2.7 OUT 输出.....	28
第 5 章 使用指南.....	29
5.1 IEC 示例.....	29
5.1.1 EtherCAT 主从站连接.....	29

5.1.1.1 创建工程	29
5.1.1.2 添加从站设备	30
5.1.1.3 设置 EtherCAT 主站参数	32
5.1.1.4 变量映射	32
5.1.2 高速计数功能使用	34
5.1.2.1 设置编码器相关参数	34
5.1.2.2 规划编码器接收的运动轨迹	37
5.1.2.3 运行结果	38
5.1.3 高速锁存功能使用	39
5.1.3.1 单次锁存功能	39
5.1.3.2 连续锁存功能	43
5.1.4 高速比较功能使用	46
5.2 BASIC 示例	52
5.2.1 EtherCAT 主从站连接	52
5.2.1.1 EtherCAT 主站的添加及配置	52
5.2.1.2 模块的添加	53
5.2.1.3 映射模块扩展 PDO	57
5.2.2 高速计数功能使用	59
5.2.3 高速锁存功能使用	63
5.2.3.1 单次锁存功能	63
5.2.3.2 连续锁存功能	67
5.2.4 高速比较功能使用	69
5.3 控制卡示例	72
5.3.1 EtherCAT 主从站连接	72
5.3.1.1 添加模块设备描述文件	73
5.3.1.2 扫描从站	73
5.3.1.3 映射模块扩展 PDO	74
5.3.2 高速计数功能使用	75
5.3.3 高速锁存功能使用	78
5.3.3.1 单次锁存功能	78
5.3.3.2 连续锁存功能	82
5.3.4 高速比较功能使用	83

第 1 章 产品概述

1.1 产品简介

雷赛 EM03DE-E4 模块是一款基于高性能、高可靠性的 EtherCAT 总线编码器模块，具有 3 路 5V 差分编码器信号输入接口、4 路高速输入接口、3 路高速输出接口。输入输出接口均采用光电隔离和滤波技术，可以有效隔离外部电路的干扰，以提高系统的可靠性。

EM03DE-E4 模块主要用于与雷赛公司支持 EtherCAT 总线通讯的控制器和控制卡配套使用。

1.2 产品特点

- ① 4 路高速输入：提供光电隔离、抗干扰滤波；
- ② 3 路高速输出：提供光电隔离、抗干扰滤波；
- ③ 3 路编码器信号输入：支持 5V 差分/单端输入；
- ④ 内部 24V 隔离电源，具有直流滤波器；
- ⑤ 塑壳安装，按压式接线端子

1.3 技术规格

EM03DE-E4 编码器模块的主要规格指标如下：

表 1.1 规格指标

编码器接口输入特性	
编码器输入端子排	直插按压式
编码器组数	3 组 (EA+EB+EZ)
输入类型	差分输入、单端输入
差分最小压差	2.5V DC
端口承受电压范围	0~7V
脉冲频率范围	脉冲方向：0~4M Hz； AB 相：0~2M Hz（四倍频前）
高速输入接口特性	

IO 端子排	直插
输入通道数	4 路
指示灯	无
输入类型	低电平输入有效
输入电压	21~27V DC
额定输入电压	24V DC
最大连续电压	30V DC
浪涌	35V DC, 500ms
导通电流	3.5mA 以上/5V 以下
关断电流	1.5mA 以下/19V 以上
光隔离	500V AC, 1 Minute
隔离组数	4 组, 单独隔离/通道
高速输出接口特性	
IO 端子排	直插
输出通道数	3 路
指示灯	无
输出类型	漏型输出, 低电平有效
负载电压	5~24V DC
输出电流	300mA/通道
漏电流	最大 8uA/通道
浪涌电流	2A, 100ms
光隔离	500V AC, 1 Minute
隔离组数	3 组, 单独隔离/通道
运行环境	
环境温度	水平安装: 0 ~ 55 ° C
	垂直安装: 0 ~ 45 ° C
相对湿度	95%无凝结
运输/存储环境	
运输/存储温度	-20 ~ 70 ° C
自由落体 EN60068-2-32	0.3 m, 5 次, 产品包装
相对湿度	95%无凝结
电磁兼容性	

静电放电 EN 61000-4-2	±8 kV, 对不导电表面的空气放电 ±4 kV, 对暴露导电表面的接触放电
快速瞬变脉冲 EN 61000-4-4	±2 kV, 5 kHz, 到交流和直流系统电源的耦合网络 ±2 kV, 5 kHz, 到 I/O 的耦合网络

1.4 安装使用

EM03DE-E4 模块采用底板定位孔的方式安装，安装尺寸如图 1.1 所示。

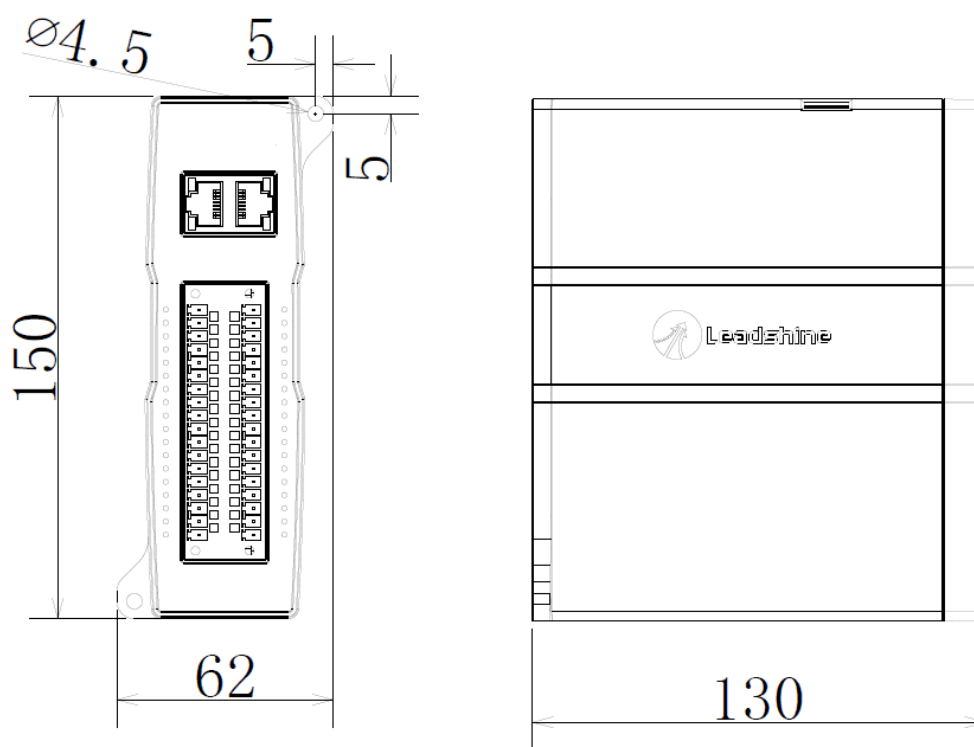


图 1.1 安装底板尺寸图

第 2 章 产品外观及硬件接线

2.1 产品外观

雷赛 EM03DE-E4 EtherCAT 总线编码器模块提供 3 路编码器输入、4 路高速输入接口、3 路高速输出接口等，产品外观如图 2.1 所示。



图 2.1 EM03DE-E4 EtherCAT 高速计数模块外观图

2.2 接口分布及引脚定义

雷赛 EM03DE-E4 EtherCAT 总线编码器模块硬件接口分布如图 2.2 所示，其接口定义如表 2.1 所示。

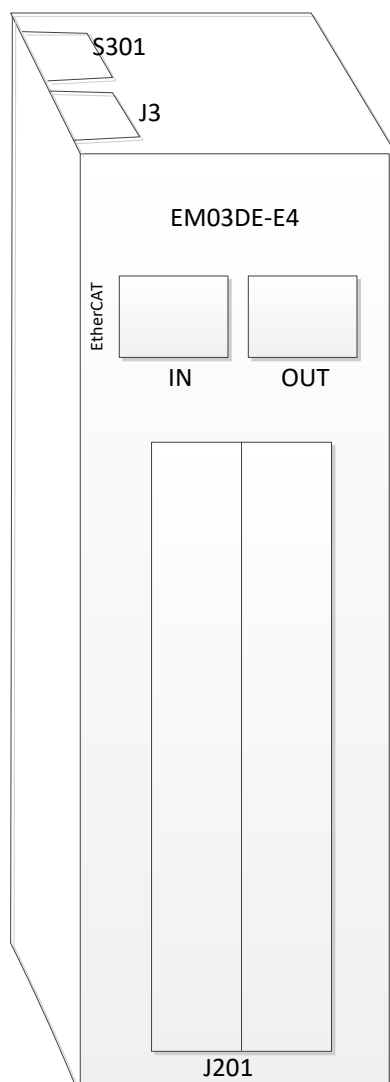


图 2.2 EM03DE-E4 模块硬件接口分布图

表 2.1 接口功能简述

名称	功能介绍
J3	直流 24V 电源输入
IN	EtherCAT0 总线接口 (in)
OUT	EtherCAT1 总线接口 (out)
J201	IO、编码器端口
S301	初始电平设置拨码

2.2.1 J3 电源接口

J3 为 24V 电源输入接口，标有 24V 的端子接+24V，标有 0V 的端子接外部电源地。PE 为外壳地接口。

2.2.2 IN、OUT 接口定义

接口 IN、OUT 是 EtherCAT 总线接口，采用 RJ45 端子，其引脚号和信号对应关系见表 2.2 所示：

表 2.2 接口 X1、X2 引脚号和信号关系表

IN 信号	信号描述	OUT 信号	信号描述	说明
1	TD+	1	TD+	发送信号+
2	TD-	2	TD-	发送信号-
3	CT	3	CT	中心抽头
4	NC	4	NC	保留
5	CT	5	CT	中心抽头
6	RD+	6	RD+	接收信号+
7	RD-	7	RD-	接收信号-
8	GND	8	GND	内部地

2.2.3 J201 接口定义

J201 接口表示 3 路编码器输入、4 路高速输入（IN0-IN3）、3 路高速输出（OUT0-OUT2），对应的引脚分布如下图表 2.3 所示：

表 2.3 J201 接口定义

序号	功能	序号	功能
1	EA0+	2	EA1+
3	EA0-	4	EA1-
5	EB0+	6	EB1+
7	EB0-	8	EB1-
9	EZ0+	10	EZ1+
11	EZ0-	12	EZ1-
13	D5V	14	D5V
15	DGND	16	DGND
17	保留	18	保留
19	EA2+	20	DIO
21	EA2-	22	DI1
23	EB2+	24	DI2
25	EB2-	26	DI3
27	EZ2+	28	EGND
29	EZ2-	30	OUT0
31	D5V	32	OUT1
33	DGND	34	OUT2
35	保留	36	EGND

2.2.4 S301 接口定义

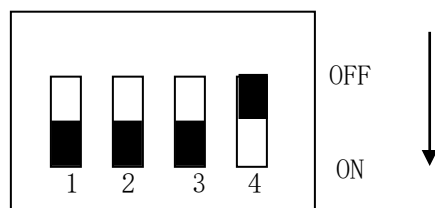


图 2.3 拨码示意图

S301 接口功能保留。

2.3 接口电路

2.3.1 编码器信号输入接口

2.3.1.1 可接收的编码器信号类型

EM03DE-E4 模块支持 2 种类型的信号输入：脉冲/方向信号和 A/B 相 4 倍频信号。

(1) 脉冲/方向信号输入

在此模式下 EA 端口接收脉冲信号；EB 端口接收方向信号。

(2) A/B 相 4 倍频信号输入

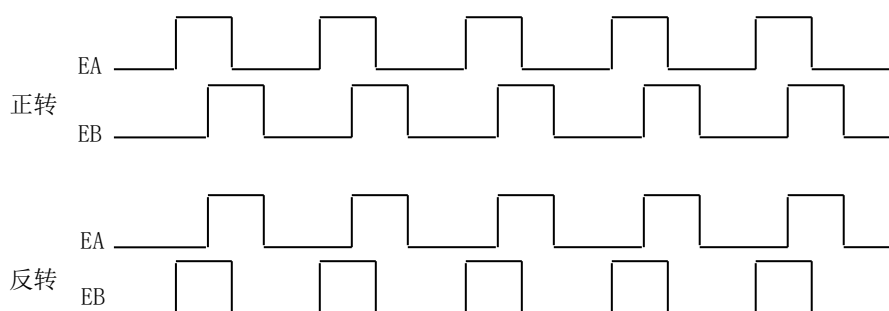


图 2.4 A/B 相正交信号

在这种模式下，EA 脉冲信号超前或滞后 EB 脉冲信号 90 度，而这种超前或滞后代表电机的运转方向。如图 2.4 所示，当 EA 信号超前 EB 信号 90° 时，被视为正转；当 EB 信号超前 EA 信号 90° 时，被视为反转。

本模块默认的计数模式为 4 倍频模式。

4 倍频计数：EA、EB 信号的上升沿和下降沿都参与了触发计数，故将一个脉冲周期分为四份。

例如：如果使用的编码器为 2500 线，即电机转一周反馈的 EA、EB 脉冲数都为 2500 个。本模块的计数值为 10000。

2.3.1.2 编码器信号输入接口电路

如果使用差分输出的编码器,输入信号的正端接 EA+(或 EB+, EZ+)端,负端接 EA-(或 EB-, EZ-)端。如图 2.5 所示。

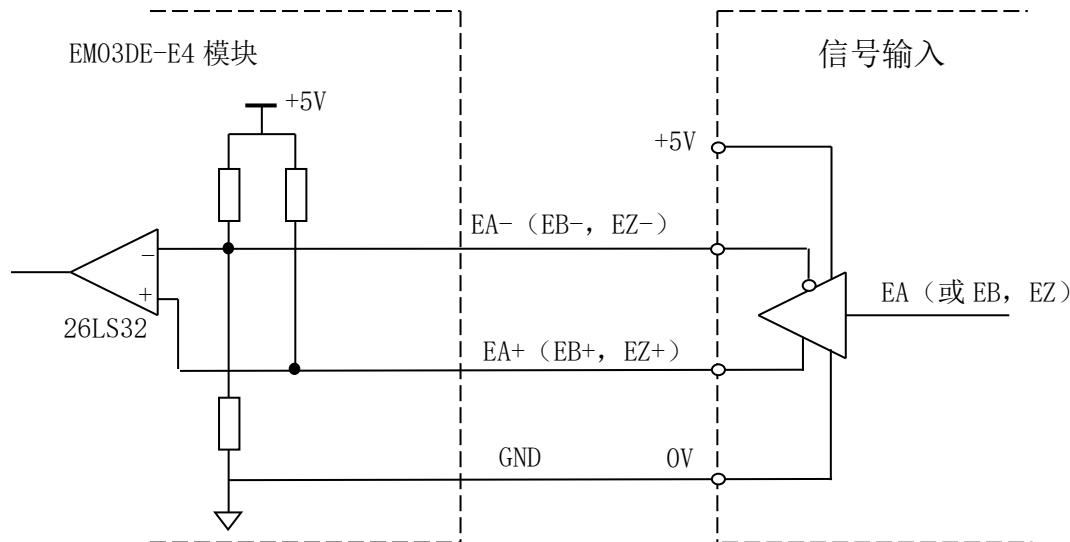


图 2.5 差分输出编码器接线原理图

如果使用集电极开路输出的编码器,则编码器输出信号接 EA+(或 EB+, EZ+)端,而 EA-(或 EB-, EZ-)端悬空。如图 2.6 所示。

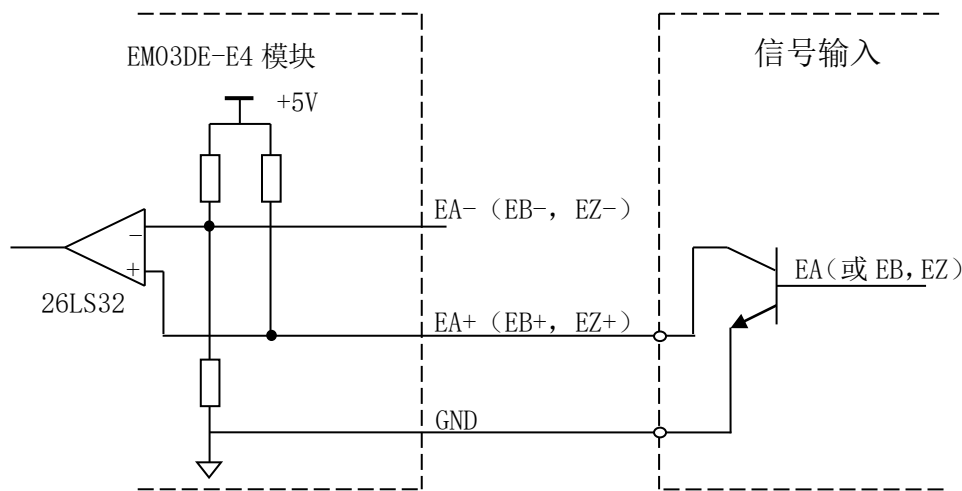


图 2.6 集电极开路输出的编码器接线原理图

注意:

- 1) 编码器等脉冲输入信号的 EA+、EA-、EB+、EB- 和 EZ+、EZ- 的差分信号电压差必须高于 3.5V, 小于 5V, 且输出电流不应小于 6mA。

2) 需要将输入设备的地线和模块的 DGND (5V 数字地) 连接。

2.3.2 高速位置锁存输入信号接口

EM03DE-E4 为用户提供 4 路高速数字输入接口，用于位置锁存、通用输入。其接口电路加有光电隔离元件，可以有效隔离外部电路的干扰，以提高系统的可靠性。高速数字输入接口接线图如图 2.7 所示：

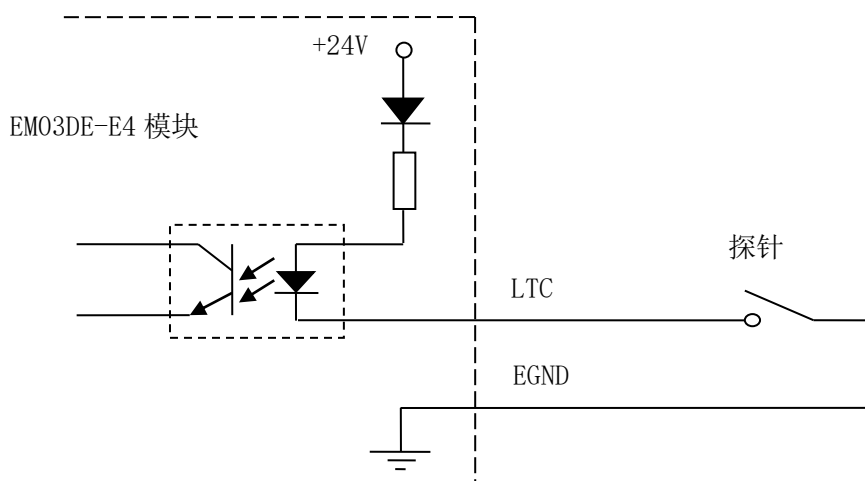


图 2.7 高速输入接线图

2.3.3 高速位置比较输出信号接口

EM03DE-E4 共有 3 个高速位置比较器，每个高速位置比较器均配有 1 个硬件位置比较输出接口。通过软件使能后，可分别设置比较模式以及关联编码器，当编码器寄存器内数值满足触发条件时，硬件自动在 CMP 端口上输出一个开关信号。模块通用数字输出信号控制常用元器件的接法如图 2.8 所示：

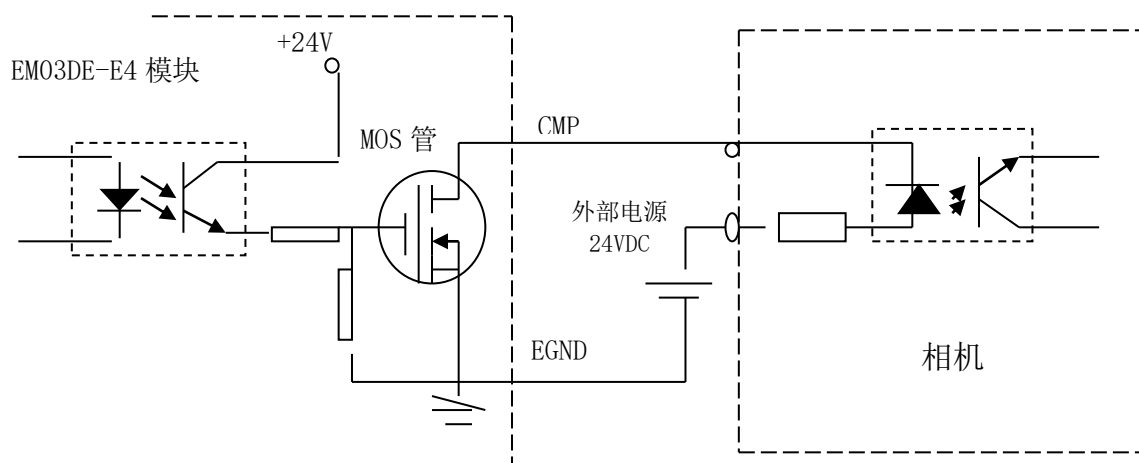


图 2.8 高速位置比较输出接线图

第 3 章 指示灯定义及说明

3.1 指示灯定义

EM03DE-E4 模块的指示灯包括连接/状态灯 (L/A)、运行灯 (RUN)、报警灯 (ERROR)。如图

3.1 所示：



图 3.1 EM03DE-E4 网口形态

其中 L/A 为网络连接/状态灯，RUN 为 RUN 灯，ERR 为 ERROR 灯。

3.2 指示灯闪烁规则

所有指示灯的闪烁都遵循如图 3.2 所示的闪烁规则。

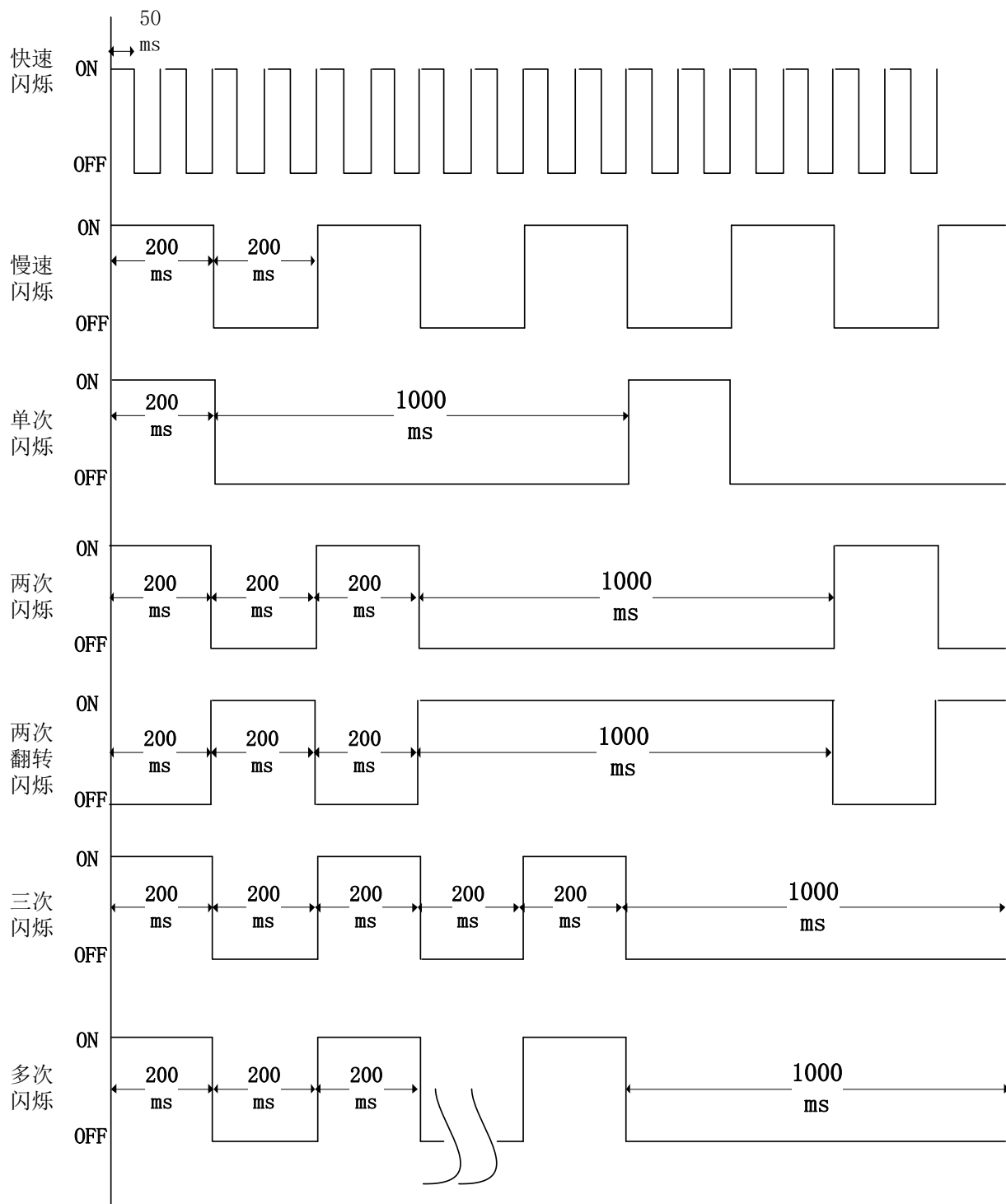


图 3.2 指示灯闪烁规则

3.3 指示灯状态

L/A 灯闪烁状态及所代表的含义如表 3-1 所示：

表 3-1 L/A 灯闪烁状态及含义

指示灯状态	状态描述	要求
常亮	端口打开	必备
快速闪烁	端口打开	必备
常灭	端口关闭	必备
两次翻转闪烁	端口关闭(模式需要手动打开)	可选
单次闪烁	本地 PHY 自动协商错误	可选
两次闪烁	远端 PHY 自动协商错误	可选
三次闪烁	位置 PHY 自动协商错误	可选

RUN 灯闪烁状态及所代表的含义如表 3-2 所示:

表 3-2 RUN 灯闪烁状态及含义

指示灯状态	连接状态	要求
常灭	设备处在初始化状态	必备
慢速闪烁	设备处在与操作状态	必备
单次闪烁	设备处在安全操作状态	必备
常亮	设备处在操作状态	必备
快速闪烁	设备正在启动, 还没进入到初始化状态或者设备处在 bootstrap 状态, 正在下载固件	可选

ERROR 灯闪烁状态及所代表的含义如表 3-3 所示:

表 3-3 ERROR 灯闪烁状态及含义

指示灯状态	连接状态	要求
常亮	典型通讯错误或者应用控制出错	可选
多次闪烁	保留	必备
三次闪烁	保留	必备
两次闪烁	应用程序看门狗超时	必备
单次闪烁	由于本地错误, 从站设备自动改为 EtherCAT 状态	必备
慢速闪烁	通用配置错误	必备
快速闪烁	启动错误	可选
常灭	正常通信	必备

第 4 章对象字典

4.1 设备参数

索引	子索引	名称	数据类型	访问属性	描述
1000H	00H	Device type	Unsigned32	r	Device type and profile (设备类型) 初始值: 0x0FFF0192
1001H	00H	Error register	Unsigned8	r	Error register (错误寄存器) 初始值: 0x00
1008H	00H	Device name	Vis String8	r	Manufacturer's designation 初始值: EM03DE-E4
1009H	00H	Hardware version	Vis String8	r	Hardware version 初始值: 2.1
100AH	00H	Software version	Vis String8	r	Software version 初始值: 2.1
1018H		Identity		r	(设备信息)
	00H	Largestsub-index	Unsigned8	r	Largest sub-index supported » 04h
	01H	Vendor ID	Unsigned32	r	Vendor ID 初始值: 0x00004321
	02H	Product code	Unsigned32	r	Product code 初始值: 0x01500033
	03H	Revision	Unsigned32	r	Revision number 初始值: 0x17060920
	04H	Serial number	Unsigned32	r	Serial number 初始值: 0x00000010

4.2 通用参数

4.2.1 编码器参数设置

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
6000H	00H	Encoder0 设置				
	01H	Encoder0_SetMode	Unsigned8	0-1	r/w	编码器 0 设置模式 0: AB 相 4 倍频模式

						1: 脉冲/方向模式 默认值: 0
	02H	Encoder0_SetABPhase	Unsigned8	0-1	r/w	编码器 0 设置 AB 相位 0: 负方向 1: 正方向 默认值: 1
	03H	Encoder0_SetVal	Signed32	-2147483648~ 2147483647	r/w	编码器 0 设置值 (设置编码器当前的值)
6001H	00H	Encoder1 设置				
	01H	Encoder1_SetMode	Unsigned8	0-1	r/w	编码器 1 设置模式 0: AB 相 4 倍频模式 1: 脉冲/方向模式 默认值: 0
	02H	Encoder1_SetABPhase	Unsigned8	0-1	r/w	编码器 1 设置 AB 相位 0: 负方向 1: 正方向 默认值: 1
	03H	Encoder1_SetVal	Signed32	-2147483648~ 2147483647	r/w	编码器 1 设置值 (设置编码器当前的值)
6002H	00H	Encoder2 设置				
	01H	Encoder2_SetMode	Unsigned8	0-1	r/w	编码器 2 设置模式 0: AB 相 4 倍频模式 1: 脉冲/方向模式 默认值: 0
	02H	Encoder2_SetABPhase	Unsigned8	0-1	r/w	编码器 2 设置 AB 相位 0: 负方向 1: 正方向 默认值: 1
	03H	Encoder2_SetVal	Signed32	-2147483648~ 2147483647	r/w	编码器 2 设置值 (设置编码器当前的值)

4.2.2 编码器值读取

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
TxPDO0	0x1A00	: Encoder Val				
6100H	00H					
	01H	Encoder0_Val	Signed32	-2147483648~ 2147483647	r	编码器 0 值
	02H	Encoder1_Val	Signed32	-2147483648~ 2147483647	r	编码器 1 值
	03H	Encoder2_Val	Signed32	-2147483648~ 2147483647	r	编码器 2 值

4.2.3 锁存器参数设置

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
6200 H	00H	Ltc0_Set				
	01H	Ltc0_Clear	Unsigned8	1	r/w	锁存器 0 状态清除 1:清除 (清除状态标记和锁存寄存器值)
	02H	Ltc0_SetMode	Unsigned8	0-1	r/w	设置工作模式: 0: 单次锁存 1: 连续锁存
	03H	Ltc0_SetFollow	Unsigned8	0-2	r/w	设置锁存器 0 锁存逻辑 0: 电平上升沿 IN0 状态 TRUE → FALSE 1: 电平下降沿 IN0 状态 FALSE → TRUE 2: 任意沿锁存
	04H	Ltc0_SetFilterTime	Unsigned32	0-65535	r/w	设置滤波时间, 单位 us。 最小 0us 最大 65535us
6201 H	00H	Ltc1_Set	Unsigned8		r/w	
	01H	Ltc1_Clear	Unsigned8	1	r/w	锁存器 1 状态清除 1:清除 (清除状态标记和锁存寄存器值)
	02H	Ltc1_SetMode	Unsigned8	0-1	r/w	设置工作模式: 0: 单次锁存 1: 连续锁存
	03H	Ltc1_SetFollow	Unsigned8	0-2	r/w	设置锁存器 1 锁存逻辑 0: 电平上升沿 IN1 状态 TRUE → FALSE 1: 电平下降沿 IN1 状态 FALSE → TRUE 2: 任意沿锁存
	04H	Ltc1_SetFilterTime	Unsigned32	0-65535	r/w	设置滤波时间, 单位 us。 最小 0us 最大 65535us
6202 H	00H	Ltc2_Set				
	01H	Ltc2_Clear	Unsigned8	1	r/w	锁存器 2 状态清除 1:清除 (清除状态标记和锁存寄存器值)
	02H	Ltc2_SetMode	Unsigned8	0-1	r/w	设置工作模式: 0: 单次锁存 1: 连续锁存
	03H	Ltc2_SetFollow	Unsigned8	0-2	r/w	设置锁存器 2 锁存逻辑 0: 电平上升沿 IN2 状态 TRUE → FALSE

						1: 电平下降沿 IN2 状态 FALSE->TRUE 2: 任意沿锁存
	04H	Ltc2_SetFilterTime	Unsigned32	0-65535	r/w	设置滤波时间, 单位 us。 最小 0us 最大 65535us
6203 H	00H	Ltc3_Set				
	01H	Ltc3_Clear	Unsigned8	1	r/w	锁存器 3 状态清除 1:清除 (清除状态标记和锁存寄存器值)
	02H	Ltc3_SetMode	Unsigned8	0-1	r/w	设置工作模式: 0: 单次锁存 1: 连续锁存
	03H	Ltc3_SetFollow	Unsigned8	0-2	r/w	设置锁存器 3 锁存逻辑 0: 电平上升沿 IN3 状态 TRUE-> FALSE 1: 电平下降沿 IN3 状态 FALSE->TRUE 2: 任意沿锁存
	04H	Ltc3_SetFilterTime	Unsigned32	0-65535	r/w	设置滤波时间, 单位 us。 最小 0us 最大 65535us

4.2.4 锁存器状态读取

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
TxPD01 0x1A01 Latch						
6300H	00H	Ltc0				
	01H	HighSpeed_IN0	Unsigned8	0-1	r	读取高速输入 0 状态 默认显示值: FALSE (电平状态为 24V)
	02H	Ltc0_Finished	Unsigned8	0-1	r	锁存器 0 锁存完成标志 TRUE: 锁存完成
	03H	Ltc0_Encoder0Val	Signed32	-2147483648~2147483647	r	锁存器 0 锁存编码器 0 值
	04H	Ltc0_Encoder1Val	Signed32	-2147483648~2147483647	r	锁存器 0 锁存编码器 1 值
	05H	Ltc0_Encoder2Val	Signed32	-2147483648~2147483647	r	锁存器 0 锁存编码器 2 值
6301H	00H	Ltc1				
	01H	HighSpeed_IN1	Unsigned8	0-1	r	读取高速输入 1 状态 默认显示值: FALSE (电平状态为 24V)

	02H	Ltc1_Finished	Unsigned8	0-1	r	锁存器 1 锁存完成标志 TRUE: 锁存完成
	03H	Ltc1_Encoder0Val	Signed32	-2147483648~2147483647	r	锁存器 1 锁存编码器 0 值
	04H	Ltc1_Encoder1Val	Signed32	-2147483648~2147483647	r	锁存器 1 锁存编码器 1 值
	05H	Ltc1_Encoder2Val	Signed32	-2147483648~2147483647	r	锁存器 1 锁存编码器 2 值
TxPD02 0x1A02 Latch						
6302H	00H	Ltc2				
	01H	HighSpeed_IN2	Unsigned8	0-1	r	读取高速输入 2 状态 默认显示值: FALSE (电平状态为 24V)
	02H	Ltc2_Finished	Unsigned8	0-1	r	锁存器 2 锁存完成标志 TRUE: 锁存完成
	03H	Ltc2_Encoder0Val	Signed32	-2147483648~2147483647	r	锁存器 2 锁存编码器 0 值
	04H	Ltc2_Encoder1Val	Signed32	-2147483648~2147483647	r	锁存器 2 锁存编码器 1 值
	05H	Ltc2_Encoder2Val	Signed32	-2147483648~2147483647	r	锁存器 2 锁存编码器 2 值
6303H	00H	Ltc3				
	01H	HighSpeed_IN3	Unsigned8	0-1	r	读取高速输入 3 状态 默认显示值: FALSE (电平状态为 24V)
	02H	Ltc3_Finished	Unsigned8	0-1	r	锁存器 3 锁存完成标志 TRUE: 锁存完成
	03H	Ltc3_Encoder0Val	Signed32	-2147483648~2147483647	r	锁存器 3 锁存编码器 0 值
	04H	Ltc3_Encoder1Val	Signed32	-2147483648~2147483647	r	锁存器 3 锁存编码器 1 值
	05H	Ltc3_Encoder2Val	Signed32	-2147483648~2147483647	r	锁存器 3 锁存编码器 2 值
SDO						
6310H	00H	Ltc0				
	01H	Ltc0_FIFO_Encoder0Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 0 锁存编码器 0 值的个数
	02H	Ltc0_FIFO_Encoder0Val	Signed32	-2147483648~2147483647	r	FIFO 模式下, 读取锁存器 0 锁存编码器 0 值
	03H	Ltc0_FIFO_Encoder1Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 0 锁存编码器 1 值的个数
	04H	Ltc0_FIFO_Encoder2Val	Signed32	-2147483648~2147483647	r	FIFO 模式下, 读取锁存器 0 锁存编码器 2 值

		coder1Val		~2147483647		锁存编码器 1 值
	05H	Ltc0_FIFO_En coder2Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 0 锁存编码器 2 值的个数
	06H	Ltc0_FIFO_En coder2Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 0 锁存编码器 2 值
6311H	00H	Ltc1				
	01H	Ltc1_FIFO_En coder0Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 1 锁存编码器 0 值的个数
	02H	Ltc1_FIFO_En coder0Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 1 锁存编码器 0 值
	03H	Ltc1_FIFO_En coder1Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 1 锁存编码器 1 值的个数
	04H	Ltc1_FIFO_En coder1Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 1 锁存编码器 1 值
	05H	Ltc1_FIFO_En coder2Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 1 锁存编码器 2 值的个数
	06H	Ltc1_FIFO_En coder2Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 1 锁存编码器 2 值
6312H	00H	Ltc2				
	01H	Ltc2_FIFO_En coder0Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 2 锁存编码器 0 值的个数
	02H	Ltc2_FIFO_En coder0Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 2 锁存编码器 0 值
	03H	Ltc2_FIFO_En coder1Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 2 锁存编码器 1 值的个数
	04H	Ltc2_FIFO_En coder1Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 2 锁存编码器 1 值
	05H	Ltc2_FIFO_En coder2Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 2 锁存编码器 2 值的个数
	06H	Ltc2_FIFO_En coder2Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 2 锁存编码器 2 值
6313H	00H	Ltc3				
	01H	Ltc3_FIFO_En coder0Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 3 锁存编码器 0 值的个数
	02H	Ltc3_FIFO_En coder0Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 3 锁存编码器 0 值
	03H	Ltc3_FIFO_En coder1Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 3 锁存编码器 1 值的个数
	04H	Ltc3_FIFO_En coder1Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 3 锁存编码器 1 值
	05H	Ltc3_FIFO_En coder2Num	Signed32	0-256	r	FIFO 模式下, 读取锁存器 3 锁存编码器 2 值的个数

	06H	Ltc3_FIFO_En coder2Val	Signed32	-2147483648 ~2147483647	r	FIFO 模式下, 读取锁存器 3 锁存编码器 2 值
--	-----	---------------------------	----------	----------------------------	---	--------------------------------

4.2.5 比较器参数设置

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
6400H	00H	Cmp0_Set 比较器 0 设置				
	01H	Cmp0_Clear	Unsigned8	1	r/w	比较器 0 清除缓冲区及比较状态 1:清除
	02H	Cmp0_SetMode	Unsigned8	0-5	r/w	设置比较器0工作模式: “000”: 关闭 “001”: 等于 “010”: 小于 “011”: 大于 “100”: fifo “101”: linear
	03H	Cmp0_Encoder_Sel	Unsigned8	0-2	r/w	选择编码器通道 可选择通道: 0、1、2
	04H	Cmp0_Out_Logic	Unsigned8	0-1	r/w	设置比较器 0 输出逻辑: “0”: 条件成立输出低电平 回读输出状态为 TRUE “1”: 条件成立输出高电平 回读输出状态为 FALSE
	05H	Cmp0_Out_timer	Unsigned32	0- 85899345	r/w	设置比较器 0 输出逻辑持续时间
	06H	Cmp0_AddData_Val	Signed32	-2147483648 ~2147483647	r/w	比较器 0 添加比较点(值)
	07H	Cmp0_Linear_Number	Unsigned32	0-65535	r/w	比较器 0 采用线性比较, 设置比较点数量
	08H	Cmp0_Linear_Interval	Signed32	-2147483648 ~2147483647	r/w	比较器 0 采用线性比较, 设置比较点增量

						值
6401H	00H	Cmp1_Set 比较器 1 设置				
	01H	Cmp1_Clear	Unsigned8	1	r/w	比较器 1 清除缓冲区及比较状态 1:清除
	02H	Cmp1_SetMode	Unsigned8	0-5	r/w	设置比较器1工作模式: “000”: 关闭 “001”: 等于 “010”: 小于 “011”: 大于 “100”: fifo “101”: linear
	03H	Cmp1_Encoder_Sel	Unsigned8	0-2	r/w	选择编码器通道 可选择通道: 0、1、2
	04H	Cmp1_Out_Logic	Unsigned8	0-1	r/w	设置比较器 1 输出逻辑: “0”: 条件成立输出低电平 回读输出状态为 TRUE “1”: 条件成立输出高电平 回读输出状态为 FALSE
	05H	Cmp1_Out_timer	Unsigned32	0- 85899345	r/w	设置比较器 1 输出逻辑持续时间
	06H	Cmp1_AddData_Val	Signed32	-2147483648 ~2147483647	r/w	比较器 1 添加比较点(值)
	07H	Cmp1_Linear_Number	Unsigned32	0-65535	r/w	比较器 1 采用线性比较, 设置比较点数量
	08H	Cmp1_Linear_Interval	Signed32	-2147483648 ~2147483647	r/w	比较器1采用线性比较, 设置比较点增量值
6402H	00H	Cmp2_Set 比较器 2 设置				
	01H	Cmp2_Clear	Unsigned8	1	r/w	比较器 2 清除缓冲区及比较状态 1:清除
	02H	Cmp2_SetMode	Unsigned8	0-5	r/w	设置比较器2工作模式:

						“000”：关闭 “001”：等于 “010”：小于 “011”：大于 “100”：fifo “101”：linear
	03H	Cmp2_Encoder_Sel	Unsigned8	0-2	r/w	选择编码器通道 可选择通道：0、1、2
	04H	Cmp2_Out_Logic	Unsigned8	0-1	r/w	设置比较器 2 输出逻辑： “0”：条件成立输出低电平 回读输出状态为 TRUE “1”：条件成立输出高电平 回读输出状态为 FALSE
	05H	Cmp2_Out_timer	Unsigned32	0- 85899345	r/w	设置比较器 2 输出逻辑持续时间
	06H	Cmp2_AddData_Val	Signed32	-2147483648 ~2147483647	r/w	比较器 2 添加比较点 (值)
	07H	Cmp2_Linear_Number	Unsigned32	0-65535	r/w	比较器 2 采用线性比较，设置比较点数量
	08H	Cmp2_Linear_Interval	Signed32	-2147483648 ~2147483647	r/w	比较器 2 采用线性比较，设置比较点增量值
6403H	00	Cmp buff0 control				
	01	enable	Unsigned8		r/w	0:禁止；1:使能
	02	clear	Unsigned8		r/w	0:无动作；1:清除ARM及FPGA缓存
	03	Add data	Signed32		r/w	添加数据点
6404H	00	Cmp buff1 control				
	01	enable	Unsigned8		r/w	0:禁止；1:使能
	02	clear	Unsigned8		r/w	0:无动作；1:清除ARM及FPGA缓存
	03	Add data	Signed32		r/w	添加数据点
6405H	00	Cmp buff2 control				
	01	enable	Unsigned8		r/w	0:禁止；1:使能
	02	clear	Unsigned8		r/w	0:无动作；1:清除ARM及FPGA缓存
	03	Add data	Signed32		r/w	添加数据点

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

4.2.6 比较器状态读取

索引	子索引	名称	数据类型	数据有效范围	访问属性	描述
TxPD03 0x1A03 Cmp						
6500H	00H	Cmp0				
	01H	HighSpeedOut0_Read	Unsigned8	0-1	r	回读取高速输出 0 状态 默认显示值: FALSE (电平为 24V)
	02H	Cmp0_FIFO_Exist	Unsigned16	0-32	r	比较器 0 当前缓冲区 剩余点数
	03H	Cmp0_Finished_Number	Unsigned16	0-65535	r	比较器 0 已经完成点 数
	04H	Cmp0_Current_CmpData	Signed32	-2147483648 ~2147483647	r	比较器 0 当前正在执 行比较点值
6501H	00H	Cmp1				
	01H	HighSpeedOut1_Read	Unsigned8	0-1	r	回读取高速输出 1 状 态 默认显示值: FALSE (电平为 24V)
	02H	Cmp1_FIFO_exist	Unsigned16	0-32	r	比较器 1 当前缓冲区 剩余点数
	03H	Cmp1_Finished_Number	Unsigned16	0-65535	r	比较器 1 已经完成点 数
	04H	Cmp1_Current_CmpData	Signed32	-2147483648 ~2147483647	r	比较器 1 当前正在执 行比较点值
6502H	00H	Cmp2				
	01H	HighSpeedOut2_Read	Unsigned8	0-1	r	回读取高速输出 2 状 态 默认显示值: FALSE (电平为 24V)
	02H	Cmp2_FIFO_exist	Unsigned16	0-32	r	比较器 2 当前缓冲区 剩余点数
	03H	Cmp2_Finished_Number	Unsigned16	0-65535	r	比较器 2 已经完成点 数
	04H	Cmp2_Current_CmpData	Signed32	-2147483648 ~2147483647	r	比较器 2 当前正在执 行比较点值
6503H	00H	Cmp buffer				

	01H	Cmp buff0 en	Unsigned8		r	Cmp buff0 使能状态
	02H	Cmp buff1 en	Unsigned8		r	Cmp buff0 使能状态
	03H	Cmp buff2 en	Unsigned8		r	Cmp buff0 使能状态
	04H	Cmp buff0 space	Unsigned16		r	Cmp buff0 剩余空间
	05H	Cmp buff1 space	Unsigned16		r	Cmp buff1 剩余空间
	06H	Cmp buff2 space	Unsigned16		r	Cmp buff2 剩余空间

4.2.7 OUT 输出

当比较模式设为关闭模式时，OUT 口做基本输出口。请注意，此对象字典已经添加为 PDO，不能进行 SDO 的操作，可以直接操作映射后的变量来进行输出口的的控制。

索引	子索引	名称	数据类型	数据有效范围		访问属性	描述
		RxPDO0 0x1600 OUT					
6600H	00H						
	01H	OUT_Val	Unsigned8	0-7		r/w	OUT 输出

对应输出口：1-OUT 口有输出，0-OUT 口无输出

OUT2	OUT1	OUT0	设置 OUT_Val 值
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

第 5 章使用指南

5.1 IEC 示例

下面将讲述 EtherCAT 计数模块 EM03DE-E4 与 SMC606 (IEC) 控制器配套使用，主要演示编码器计数功能、高速锁存功能和高速比较功能，主要步骤如下：

- (1) 将控制器的 EtherCAT 口与扩展模块的 EtherCAT 口使用网线连接，连接 24V 电源。
- (2) 打开 SMC IEC Studio 软件，与控制器通讯，新建工程 (StandProject)，然后在工程中添加设备 (EtherCAT Master)，添加成功之后在主站下面添加从站设备 EM03DE-E4 模块。
- (3) 编写各部分功能模块代码。
- (4) 配置从站设备的映射。
- (5) 编译并下载到控制器，运行程序。

具体操作如下所示。

5.1.1 EtherCAT 主从站连接

5.1.1.1 创建工程

新建工程，并在设备栏添加 EtherCAT 主站 EtherCAT Master，如图 5.1 所示；

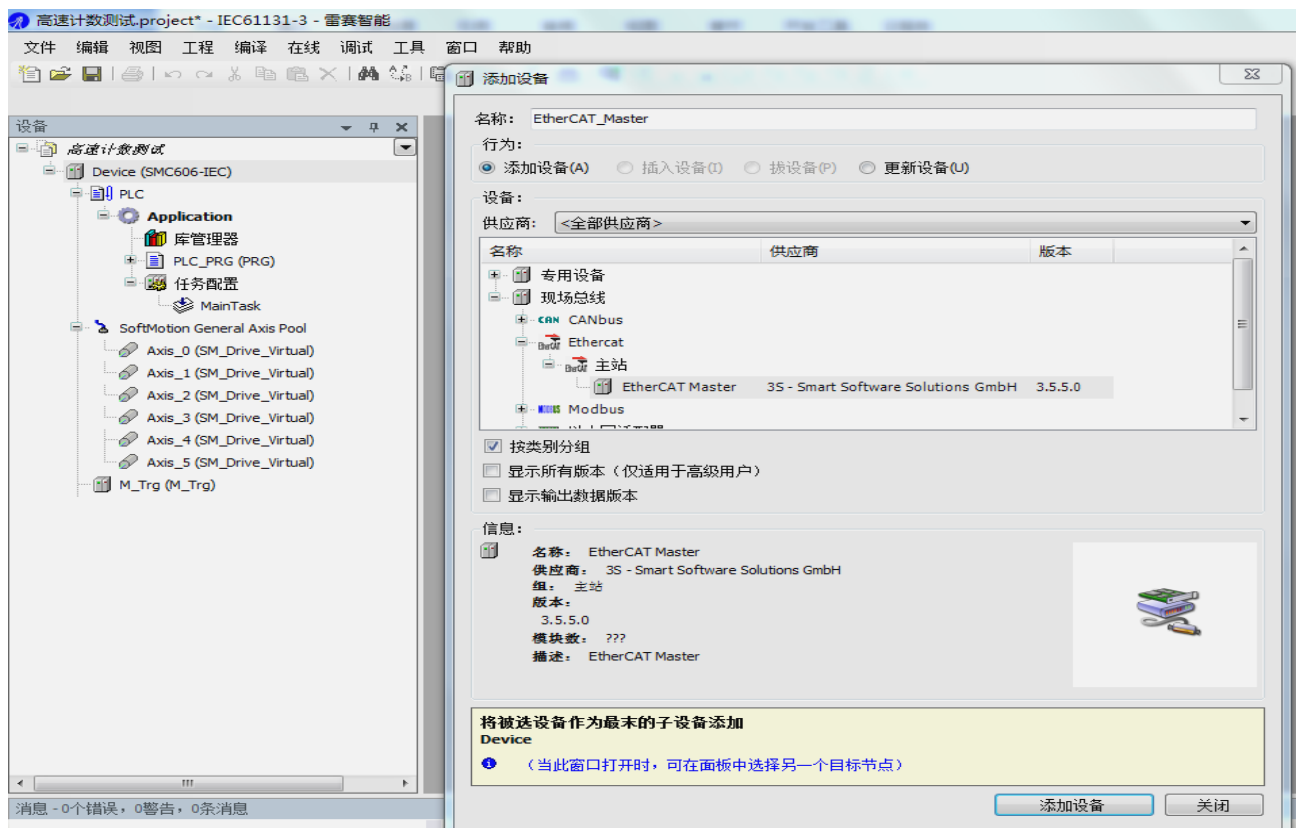


图 5.1 添加 EtherCAT 主站

5.1.1.2 添加从站设备

在 EtherCAT Master 目录树下，添加 EM03DE-E4 模块（请先将 EM03DE-E4 模块的设备描述文件添加到设备库中）如图 5.2，添加模块后的目录树结构如图 5.3 所示。

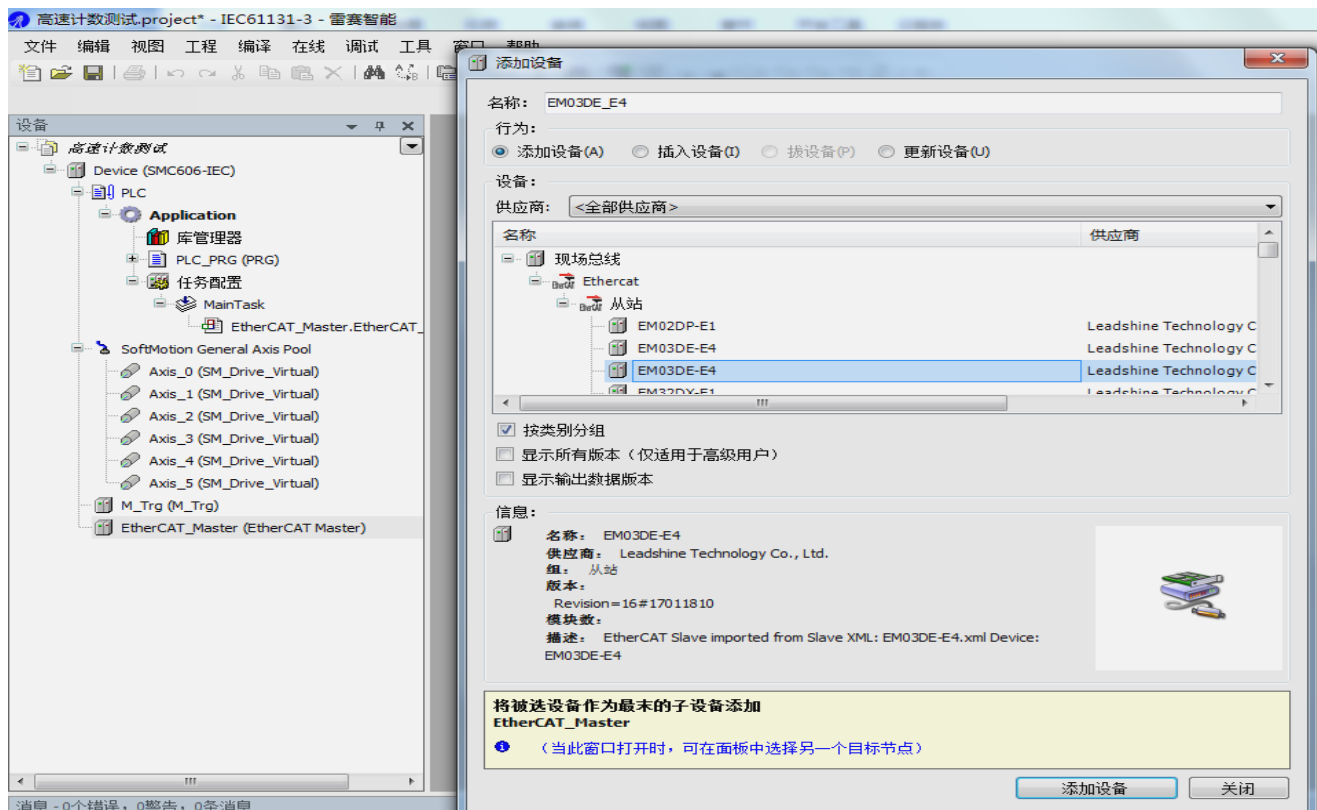


图 5.2 添加 EM03DE-E4 模块

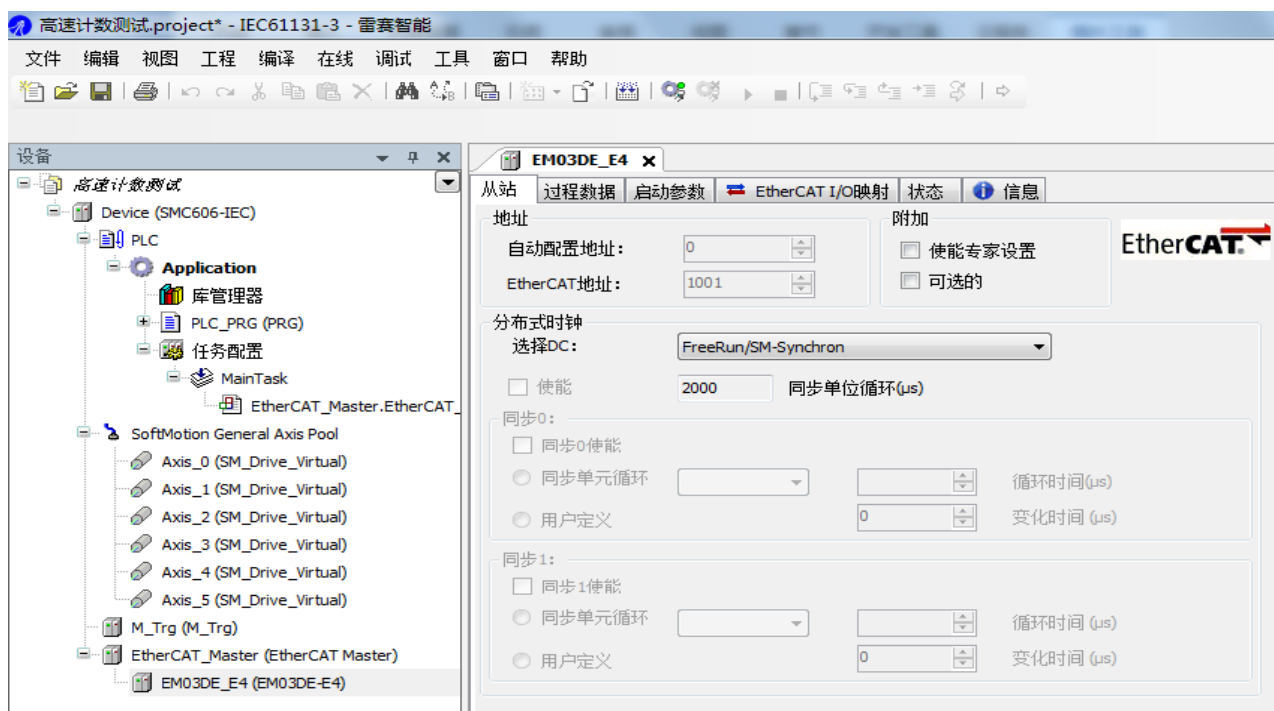


图 5.3 设备树

5.1.1.3 设置EtherCAT主站参数

双击左侧的树形目录的“EtherCAT Master”，并点击“主站”，根据名称选择网络，网络名称选择默认即可，如图 5.4 所示参数。

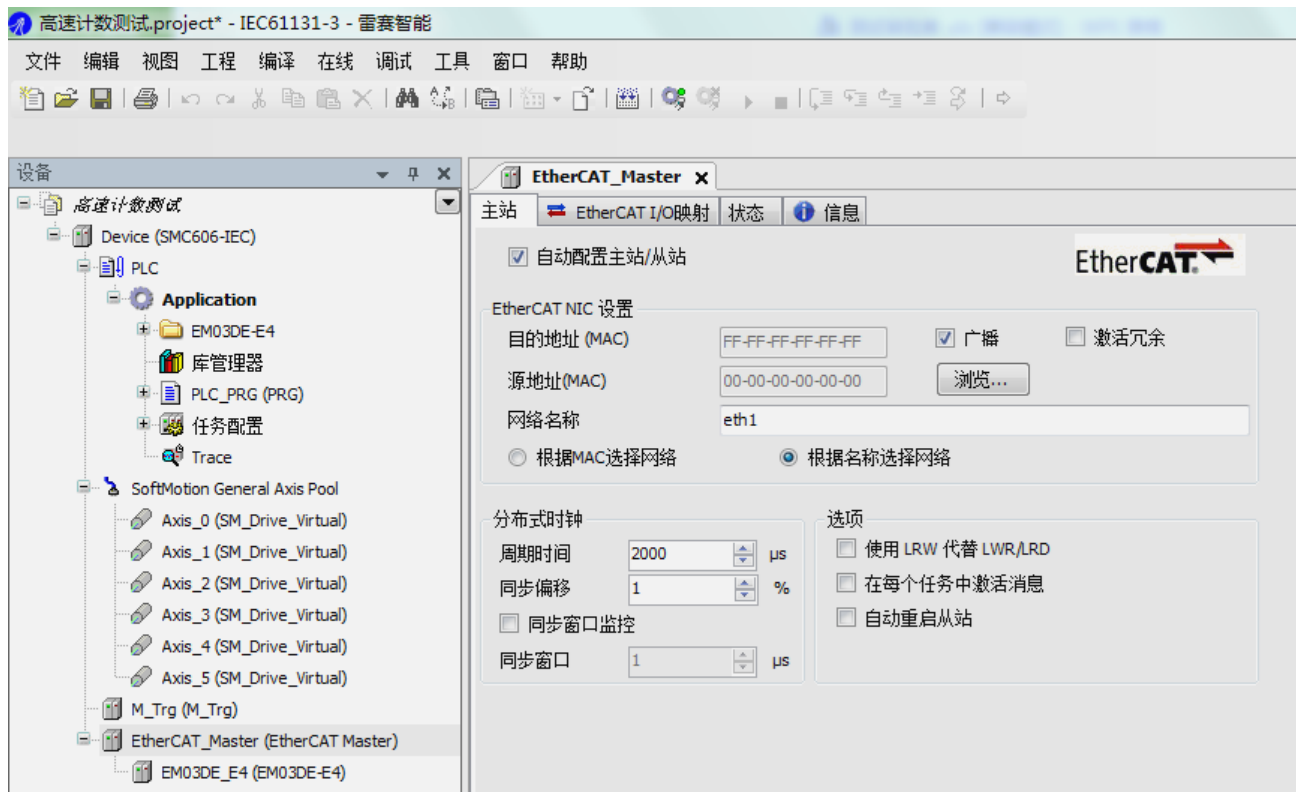


图 5.4 设置 EtherCAT 主站参数

5.1.1.4 变量映射

在配置映射关系前，必须在程序中先定义需要配置映射的变量，本例程中定义的变量请参考例程的源代码。

双击左侧的树形目录的“EM03DE-E4”，并点击“EtherCAT I/O 映射”，将程序中定义的编码器计数值、高速锁存输入、高速比较输出进行映射，映射如图 5.5 所示。

映射完成后选择变量更新方式为 Enables 2 (always in bus cycle task)，如图 5.6 所示。

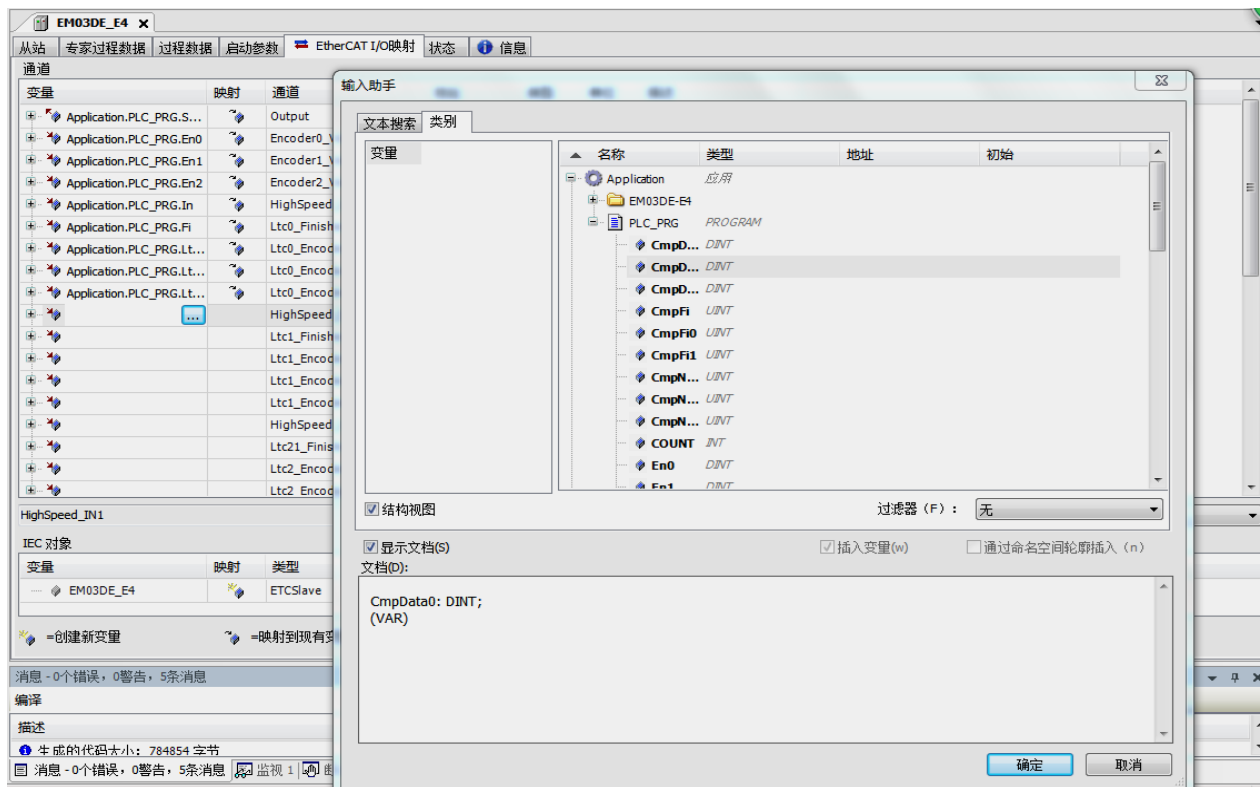


图 5.5 变量映射

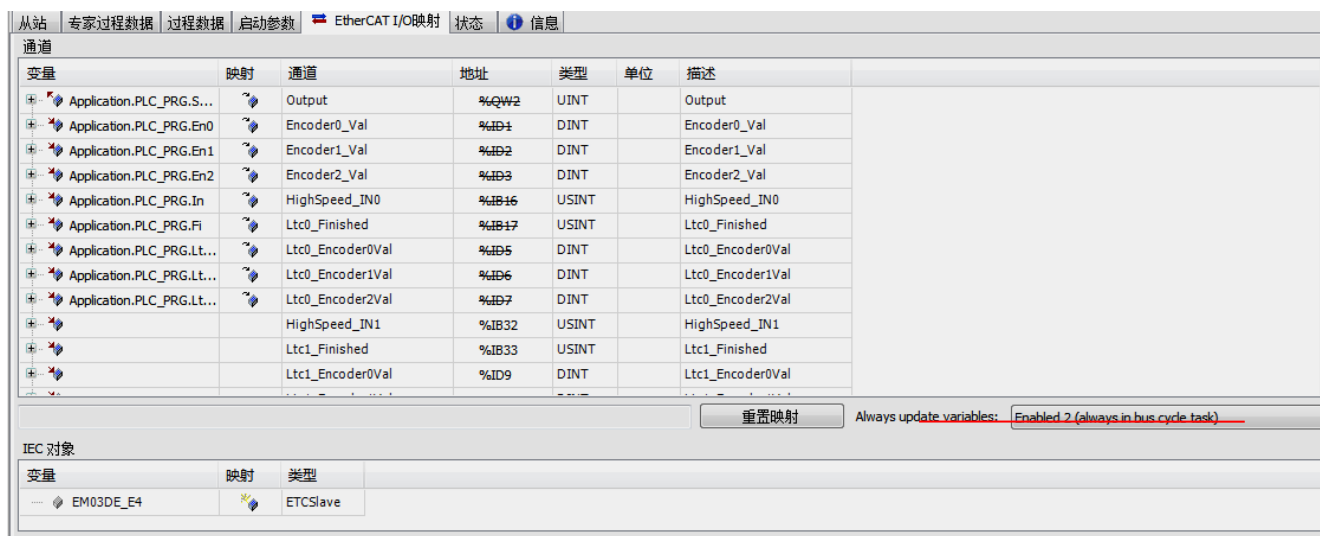


图 5.6 变量映射完成

5.1.2 高速计数功能使用

5.1.2.1 设置编码器相关参数

完成上述步骤后在程序中修改变量 Test_index 的值即可实现对高速计数器、高速锁存器以及高速比较器的选择，从而进行相应的参数设置。

设置参数采用 SDO 功能块，如图 5.7 所示。

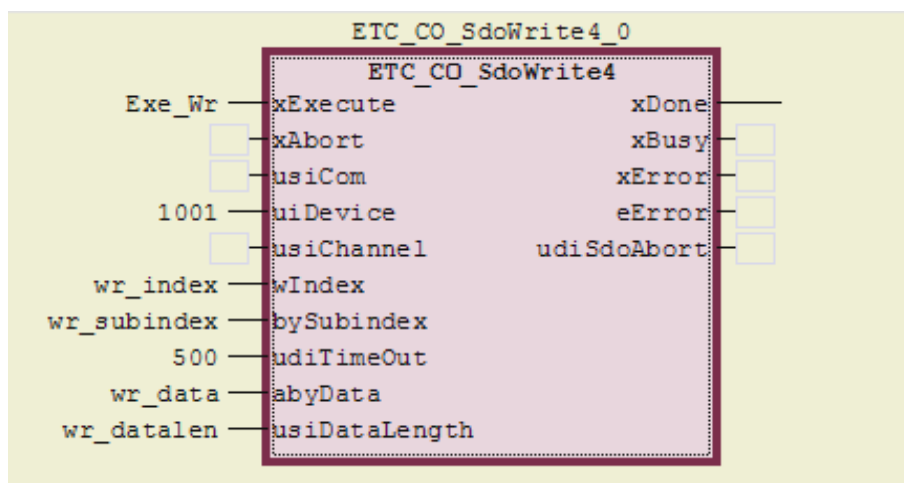


图 5.7 SDO 设置功能块

编码器参数设置步骤如下：

a. 编码器通道选择：

编码器 0 的对象字典索引为 16#6000。

编码器 1 的对象字典索引为 16#6001。

编码器 2 的对象字典索引为 16#6002。

b. 设置编码器计数模式：

计数模式的对象字典子索引为 1。

编码器有两种工作模式：0-AB 相模式；1-脉冲方向模式；

示例代码如图 5.8 所示。

```

// 设置编码器的参数
CASE ystate OF
0:
    Exe_Wr:=FALSE;
1:
    //Set Mode: 0-AB相模式; 1-脉冲方向模式
    wr_data[1]:=0; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6000;
    wr_subindex:=1;
    wr_dataLen:=1;
    Exe_Wr:=TRUE;
    ystate:=2;
2:
    IF ETC_CO_SdoWrite4_0.xDone THEN
        Exe_Wr:=FALSE;
        ystate:=21;
    END_IF

```

图 5.8 设置编码器计数模式

c. 设置编码器计数方向:

计数方向的对象字典子索引为 2。

计数方向有效值：0-负方向；1-正方向；

示例代码如图 5.9 所示。

```

21:
    //Set AB Phase: 0-负方向; 1-正方向
    wr_data[1]:=1; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6000;
    wr_subindex:=2;
    wr_dataLen:=1;
    Exe_Wr:=TRUE;
    ystate:=22;
22:
    IF ETC_CO_SdoWrite4_0.xDone THEN
        Exe_Wr:=FALSE;
        ystate:=31;
    END_IF

```

图 5.9 设置编码器方向

d. 设置编码器初始值:

编码器初始值的对象字典子索引为 3。

编码器初始值有效范围：-2147483648 至 2147483647（有符号的 32 位值）

示例代码如图 5.10 所示。

```
31:
    //Set Encoder Val;
    wr_data[1]:=0; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6000;
    wr_subindex:=3;
    wr_dataLen:=4;
    Exe_Wr:=TRUE;
    istate:=32;
32:
    IF ETC_CO_SdoWrite4_0.xDone THEN
        //Exe_Wr:=FALSE;
        istate:=100;
    END_IF
100:
    ;
END_CASE
```

图 5.10 编码器设置值

完成上述步骤后，即完成对编码器参数配置的代码编写。

编码器 0 的计数模式设置为 AB 相，正方向计数，编码器初始值为 0。

5.1.2.2 规划编码器接收的运动轨迹

向编码器发送 AB 相脉冲，加减速度 2M，以 2M 的速度进行恒速相对运动，相对运动距离为 4M。

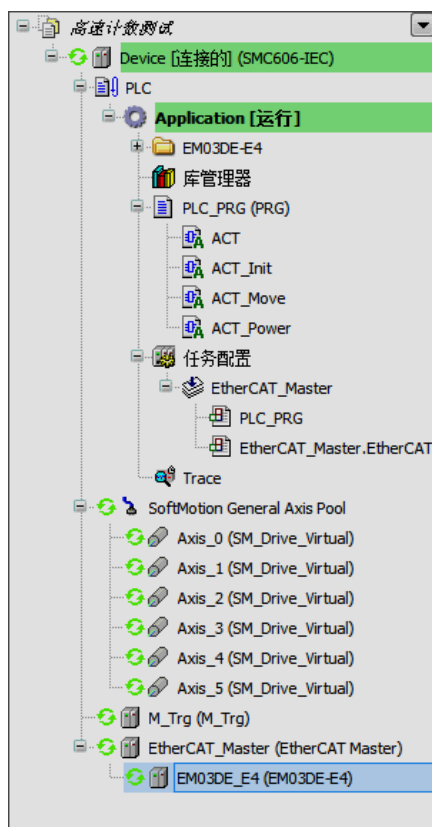


图 5.11

注意：运动程序与 EtherCAT 程序必须放在同一个任务中，在本例程中任务配置如图 5.11。

5.1.2.3 运行结果

完成上述步骤后，程序运行结果如下：

从站 专家过程数据 过程数据 启动参数 在线 EtherCAT I/O映射 状态 信息									
通道									
变量	映射	通道	地址	类型	当前值	准备值	单位	描述	
Application.PLC_PRG.S...		Output	%QW2	UINT	0				Output
Application.PLC_PRG.En0		Encoder0_Val	%ID1	DINT	16000000				Encoder0_Val
Application.PLC_PRG.En1		Encoder1_Val	%ID2	DINT	0				Encoder1_Val
Application.PLC_PRG.En2		Encoder2_Val	%ID3	DINT	0				Encoder2_Val

图 5.12 映射编码器实际计数值

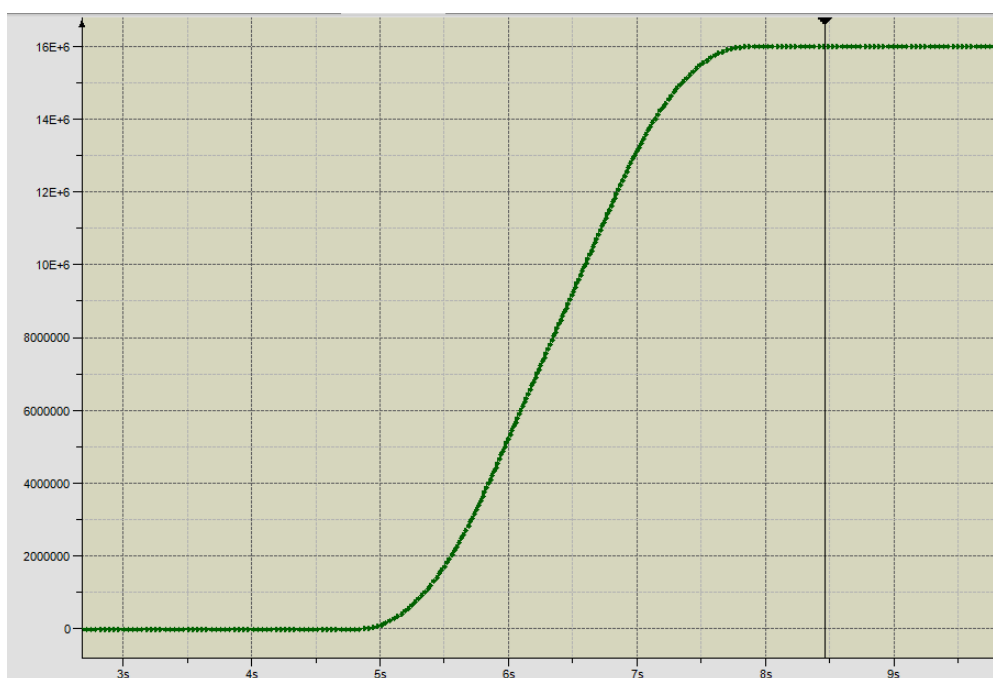


图 5.13 trace 实际数值

由于编码器设置为 AB 模式进行计数，即四倍频计数，由图 5.12 与图 5.13 可以得到，编码器的计数为 16M。

5.1.3 高速锁存功能使用

高速锁存功能是在高速计数功能的基础上运行的（高速计数功能使用参考 5.1.2 节）。

5.1.3.1 单次锁存功能

锁存器参数设置步骤如下：

a. 锁存器通道选择：

锁存器 0 的对象字典索引为 16#6200。

锁存器 1 的对象字典索引为 16#6201。

锁存器 2 的对象字典索引为 16#6202。

锁存器 3 的对象字典索引为 16#6203。

b. 清除锁存器的锁存标志：

清除锁存标志的对象字典索引为 1。

清除锁存标志的有效值为 1。

清除锁存标志的示例代码如图 5.14 所示。

```
// 清除锁存状态
CASE irstate OF
0:
    Exe_Wr:=FALSE;
1:
    //clear ltc; 清楚锁存状态
    wr_data[1]:=1; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6200;
    wr_subindex:=1;
    wr_dataLen:=1;
    Exe_Wr:=TRUE;
    irstate:=2;
2:
    IF ETC_CO_SdoWrite4_0.xDone THEN
        Exe_Wr:=FALSE;
        irstate:=3;
    END_IF
```

图 5.14 清楚锁存状态

c. 设置锁存器的锁存模式：

锁存模式的对象字典索引为 2。

锁存器有两种工作模式：0-单次锁存；1-连续锁存。

设置锁存模式的示例代码如图 5.15 所示。

```

3:
//Ltc0_SetMode; 0: 单次锁存 1: 连续锁存
wr_data[1]:=0; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6200;
wr_subindex:=2;
wr_datalen:=1;
Exe_Wr:=TRUE;
istate:=4;
4:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=5;
END_IF

```

图 5.15 锁存器模式

d. 设置锁存器的锁存逻辑：

设置锁存逻辑的对象字典索引为 3。

锁存逻辑有三种：0-电平上升沿；1-电平下降沿；2-任意沿锁存。

设置锁存逻辑的示例代码如图 5.16 所示。

```

5:
//Ltc0_Setfollow; 0-电平上升沿锁存; 1-电平下降沿锁存 ; 2: 任意沿锁存
wr_data[1]:=2; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6200;
wr_subindex:=3;
wr_datalen:=1;
Exe_Wr:=TRUE;
istate:=6;
6:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=7;
END_IF

```

图 5.16 设置锁存逻辑

e. 设置锁存器的滤波时间:

设置滤波时间的对象字典子索引为 4。

设置滤波时间，单位 us。最小 0us~最大 65535us。

设置滤波时间的示例代码如图 5.17 所示。

```
7:
    //Ltc0_SetFilterTime; 设置滤波时间, 单位us. 最小0us最大65535us
    wr_data[1]:=0; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6200;
    wr_subindex:=4;
    wr_dataLen:=4;
    Exe_Wr:=TRUE;
    istate:=8;
8:
    IF EIC_CO_SdoWrite4_0.xDone THEN
        //Exe_Wr:=FALSE;
        istate:=100;
    END_IF
100:
    ;
END_CASE
ACT_SDO();
```

图 5.17 设置滤波时间

完成上述步骤后，即完成对锁存器参数配置的代码编写。

锁存器 0 的锁存模式为单次锁存，锁存逻辑为任意沿锁存，滤波时间设置为 0。编码器设置与 5.1.2 设置相同。

f. 运行结果：完成以上步骤后，程序运行结果如下：

Application.PLC_PRG.In	HighSpeed_IN0	%IB16	USINT	1	HighSpeed_IN0
Application.PLC_PRG.Fi	Ltc0_Finished	%IB17	USINT	1	Ltc0_Finished
Application.PLC_PRG.Lt...	Ltc0_Encoder0Val	%ID5	DINT	14605924	Ltc0_Encoder0Val
Application.PLC_PRG.Lt...	Ltc0_Encoder1Val	%ID6	DINT	14605924	Ltc0_Encoder1Val
Application.PLC_PRG.Lt...	Ltc0_Encoder2Val	%ID7	DINT	14605924	Ltc0_Encoder2Val

图 5.18 单次锁存的锁存值

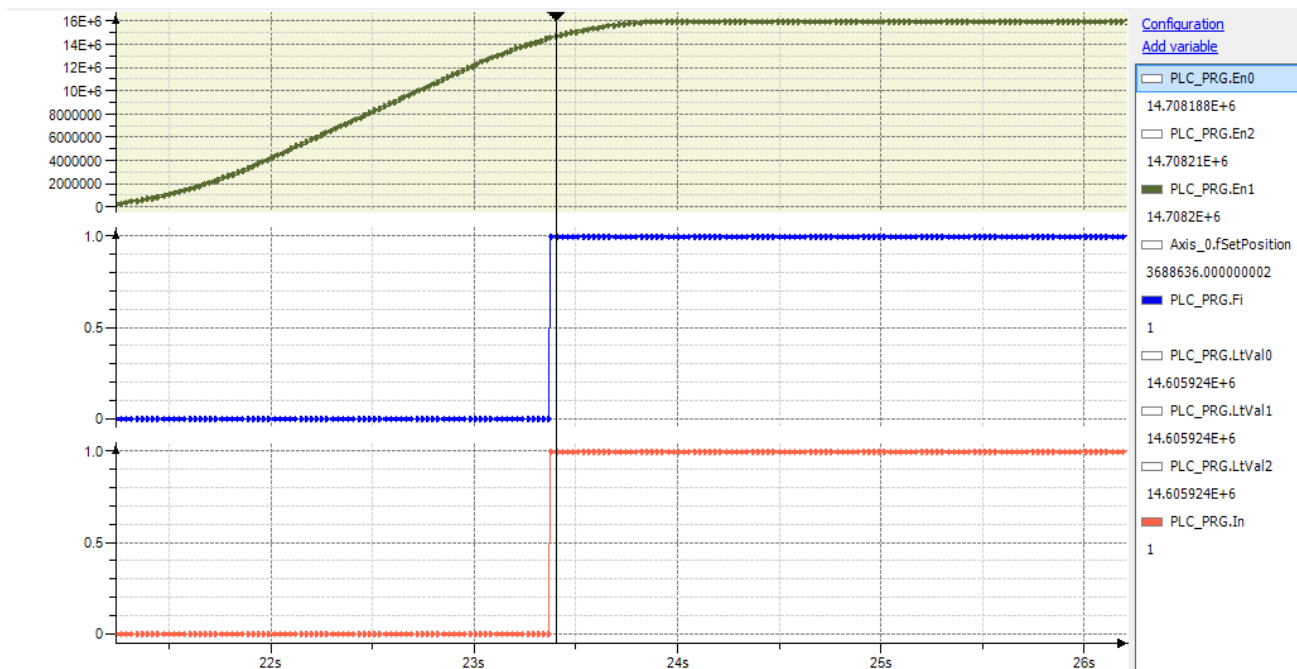


图 5.19 单次锁存 trace 曲线

5.1.3.2 连续锁存功能

锁存器参数设置步骤： 见单次锁存功能 5.1.3.1，需将步骤 c 设置为连续锁存模式。

锁存器连续锁存状态读取设置：

读取参数采用 SDO 功能块，如图 5.20：

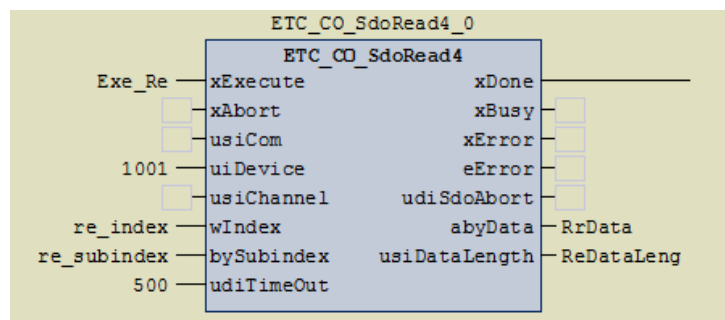


图 5.20 SDO 读取功能块

a. 读取锁存器的选择：

读取锁存器 0 的对象字典索引为 16#6310。

读取锁存器 1 的对象字典索引为 16#6311。

读取锁存器 2 的对象字典索引为 16#6312。

读取锁存器 3 的对象字典索引为 16#6313。

b. 锁存器的状态读取：

读取锁存编码器 0 值的个数-对象字典子索引为 1。

读取锁存编码器 0 值-对象字典子索引为 2。

读取锁存编码器 1 值的个数-对象字典子索引为 3。

读取锁存编码器 1 值-对象字典子索引为 4。

读取锁存编码器 2 值的个数-对象字典子索引为 5。

读取锁存编码器 2 值-对象字典子索引为 6。

读取锁存器状态的示例代码如图 5.21、图 5.22 所示。

```

//读取个数
CASE istate_read OF
0:
    Exe_Re:=FALSE;
1:
    re_index:=Test_index;
    re_subindex:=Test_subindex;
    Exe_Re:=TRUE;
    istate_read:=2;
2:
    IF ETC_CO_SdoRead4_0.xDone THEN
        Exe_Re:=FALSE;

        //读取当前锁存的数据个数
        LchNum:=Pack_ByteToDINT(RrData);
        IF LchNum>0 THEN
            istate_read:=10;
            Lch_ValPos :=0;
        ELSE
            istate_read:=100;
        END_IF
    END_IF
END_IF

```

图 5.21 锁存值个数读取

```

10:
    //读取锁存的数据
    Exe_ReVal:=TRUE;
    re_index:=Test_index;
    re_subindexVal:=Test_subindex+1;
    istate_read:=11;
11:
    IF ETC_CO_SdoRead4_2.xDone THEN
        Exe_ReVal:=FALSE;

        Lch_ValBuff[Lch_ValPos]:=Pack_ByteToDINT(RrDataVal);
        Lch_ValPos:=Lch_ValPos+1;

        IF Lch_ValPos>=LchNum THEN //读取完所有数据
            istate_read:=100;
        ELSE //继续读取下一个
            istate_read:=10;
        END_IF
    END_IF

100:
;
END_CASE

```

图 5.22 锁存值读取

完成上述步骤后，即完成锁存器状态读取配置的代码编写。

读取锁存器 0 锁存编码器 0 值的个数以及编码器 0 值。

c. 运行结果：完成以上步骤后，程序运行结果如下：

LchNum	DINT	9
Lch_ValBuff	ARRAY [0..100] OF..	
Lch_ValBuff[0]	DINT	153396
Lch_ValBuff[1]	DINT	1420300
Lch_ValBuff[2]	DINT	2546698
Lch_ValBuff[3]	DINT	5582804
Lch_ValBuff[4]	DINT	7198110
Lch_ValBuff[5]	DINT	10399592
Lch_ValBuff[6]	DINT	12014210
Lch_ValBuff[7]	DINT	14568590
Lch_ValBuff[8]	DINT	15372109
Lch_ValBuff[9]	DINT	0

图 5.23 锁存值个数与锁存值

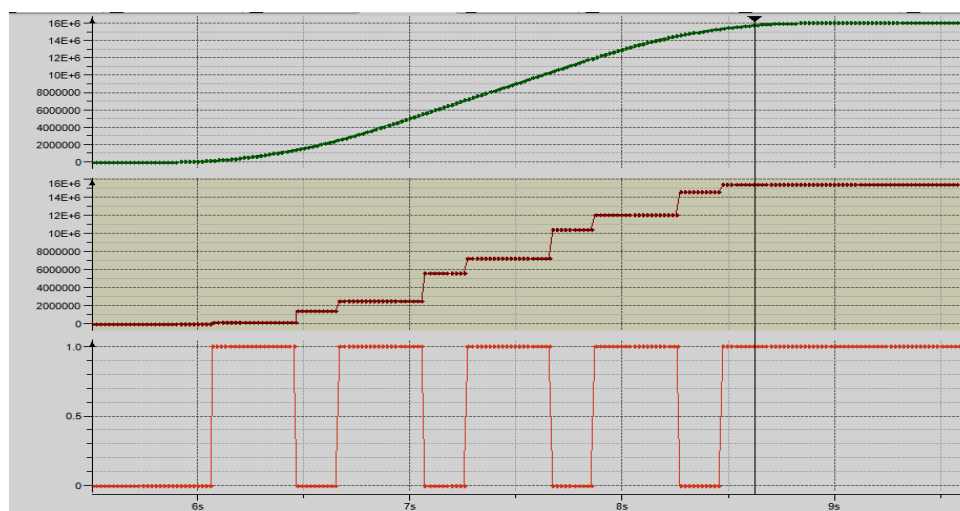


图 5.24 trace 实际数值

编码器设置为 AB 模式进行计数，即四倍频计数，由图 5.23 与图 5.24，编码器的计数为 16M，锁存器设置的是任意沿锁存模式，锁存值个数为 9。

5.1.4 高速比较功能使用

高速比较功能是在高速计数功能的基础上运行的（高速计数功能使用参考 5.1.2 节）。

比较器参数设置步骤如下：

a. 比较器通道选择：

比较器 0 的对象字典索引为 16#6400。

比较器 1 的对象字典索引为 16#6401。

比较器 2 的对象字典索引为 16#6402。

b. 清除比较器的比较状态：

清除比较状态的对象字典索引为 1。

清除比较状态的有效值为 1。

设置清除比较状态的示例代码如图 5.25 所示。

```
// 设置比较器
CASE  istate OF
0:   Exe_Wr:=FALSE;
1:
    //clear ; 清除比较状态
    wr_data[1]:=1; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
    wr_index:=16#6400;
    wr_subindex:=1;
    wr_dataLen:=1;
    Exe_Wr:=TRUE;
    istate:=2;
2:
    IF ETC_CO_SdoWrite4_0.xDone THEN
        Exe_Wr:=FALSE;
        istate:=21;
    END_IF
```

图 5.25 清除缓冲区以及比较状态

c. 设置比较器的比较模式:

比较模式的对象字典子索引为 2。

比较器有六种工作模式：0-关闭，1-等于，2-小于，3-大于，4-FIFO（队列），5-Liner（线性）

备注：当比较器工作模式为 0-关闭时，输出口味普通输出口。

设置比较模式的示例代码如图 5.26 所示。

```

21:
//Set Mode: 0:关闭 1:等于 2:小于 3:大于 4:FIFO 5:Liner
wr_data[1]:=3; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=2;
wr_dataLen:=1;
Exe_Wr:=TRUE;
istate:=22;
22:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=31;
END_IF

```

图 5.26 比较器比较模式

d. 选择编码器通道:

选择编码器通道的对象字典子索引为 3。

编码器通道有三个：0-编码器 0；1-编码器 1；2-编码器 2。

选择编码器通道的示例代码如图 5.27 所示。

```

31:
//Set Encoder_Sel; 编码器通道选择: 0, 1, 2
wr_data[1]:=0; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=3;
wr_dataLen:=1;
Exe_Wr:=TRUE;
istate:=32;
32:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=41;
END_IF

```

图 5.27 编码器通道

e. 设置比较器输出逻辑:

比较器输出逻辑的对象字典子索引为 4。

比较器输出逻辑有两种：“0”-条件成立输出低电平，显示状态为 1（TRUE）；

“1”-条件成立输出高电平，显示状态为 0（FALSE）。

设置输出逻辑的示例代码如图 5.28 所示。

```

41:
//Set Out_Logic; 0--条件成立显示状态为1; 1--条件成立显示状态为0
wr_data[1]:=1; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=4;
wr_datalen:=1;
Exe_Wr:=TRUE;
istate:=42;
42:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=51;
END_IF

```

图 5.28 比较器输出逻辑

f. 设置比较器输出电平时间:

比较器输出电平时间的对象字典子索引为 5；

电平时间的有效范围为：0 至 85899345（无符号的 32 位值）单位：us

设置输出电平时间的示例代码如图 5.29 所示。

```

51:
//Set Out_timer; 有效电平时间
wr_data[1]:=6; wr_data[2]:=8; wr_data[3]:=5; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=5;
wr_datalen:=4;
Exe_Wr:=TRUE;
istate:=52;
52:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=61;
END_IF

```

图 5.29 比较器输出电平有效时间

注意： 比较器输出电平有效时间只对于比较模式 4--队列比较与比较模式 5--线性比较有效。

g. 添加比较值点:

添加比较点的对象字典子索引为 6。

比较点的有效范围：-2147483648 至 2147483647（有符号的 32 位值）

添加比较点的示例代码如图 5.30 所示。

```

61:
//Set AddData_Val; 添加比较点(值)
wr_data[1]:=16#20; wr_data[2]:=16#A1; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=6;
wr_datalen:=4;
Exe_Wr:=TRUE;
istate:=62;
62:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=63;
END_IF
63:
//Set AddData_Val ; 添加比较点(值)
wr_data[1]:=16#20; wr_data[2]:=16#A1; wr_data[3]:=16#8; wr_data[4]:=0;
wr_index:=16#6400;|
wr_subindex:=6;
wr_datalen:=4;
Exe_Wr:=TRUE;
istate:=64;
64:
IF ETC_CO_SdoWrite4_0.xDone THEN
    //Exe_Wr:=FALSE;
    istate:=100;
END_IF

```

图 5.30 添加比较点(值)

h. 设置比较点数量:

设置比较点数量的对象字典子索引为 7。

比较点数量的有效范围：0 至 0-65535（无符号的 32 位值）

设置比较点数量的示例代码如图 5.31 所示。

```

//=====
81:
//Set Linear_Number; 采用线性比较, 设置比较点数量
wr_data[1]:=3; wr_data[2]:=0; wr_data[3]:=0; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=7;
wr_datalen:=4;
Exe_Wr:=TRUE;
istate:=82;
82:
IF ETC_CO_SdoWrite4_0.xDone THEN
    Exe_Wr:=FALSE;
    istate:=83;
END_IF

```

图 5.31 设置比较点数量

注意：设置比较点数量只在比较模式 5—线性比较时有效。

i. 设置比较点增量值：

设置增量值的对象字典子索引为 8。

比较点增量值的有效范围：-2147483648 至 2147483647（有符号的 32 位值）

设置比较点增量值的示例代码如下所示。

```

83:
//Set Linear_Interval; 采用线性比较, 设置比较点增量值
wr_data[1]:=6; wr_data[2]:=9; wr_data[3]:=5; wr_data[4]:=0;
wr_index:=16#6400;
wr_subindex:=8;
wr_dataLen:=4;
Exe_Wr:=TRUE;
istate:=84;

84:
IF ETC_CO_SdoWrite4_0.xDone THEN
    //Exe_Wr:=FALSE;
    istate:=100;
END_IF

//=====
100:
;
END_CASE

ACT_SDO();

```

图 5.32 设置比较点增量值

注意：设置比较点增量只在比较模式 5—线性比较时有效。

完成上述步骤后，即完成对比较器参数的配置的代码编写。

比较器 0 的锁存模式设置上升沿锁存，选择大于比较模式，比较输出逻辑设置为 1，比较通道为编码器 0。编码器的设置与 5.1.2 设置相同。

j. 运行结果：完成以上步骤后，程序运行结果如下：

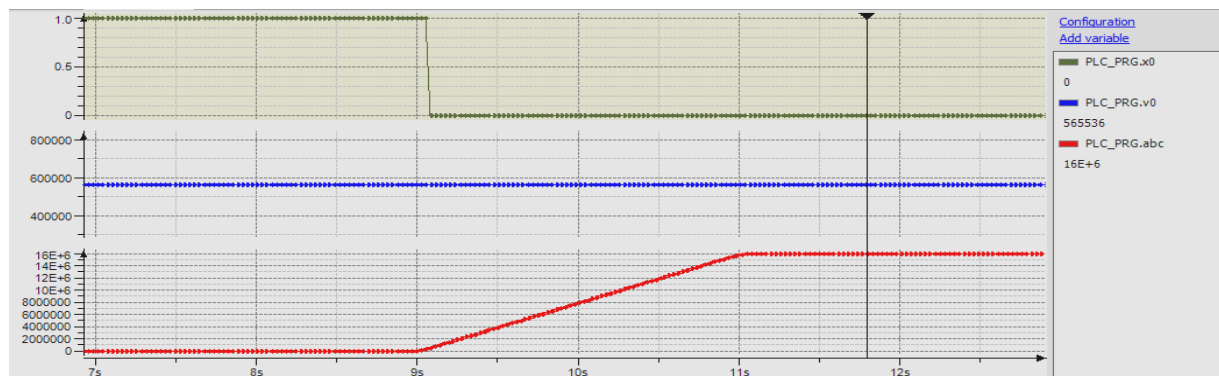


图 5.33 trace 实际计数值，比较值

编码器设置为 AB 模式进行计数, 即四倍频计数, 由图 5.33 可以得到, 编码器的计数为 16M, 比较器设置模式为大于模式, 输出逻辑设置为 1, 即条件成立显示模式为 0, 比较值为 565536。

5.2 BASIC 示例

下面将讲述 EtherCAT 计数模块 EM03DE-E4 与 BAC332E 控制器配套使用，主要演示高速锁存功能和高速比较功能，主要步骤如下：

- (1) 将控制器的 EtherCAT 口与扩展模块的 EtherCAT 口使用网线连接，连接 24V 电源。
- (2) 打开 SMC Basic Studio 软件，与控制器通讯，新建工程（StandProject），然后在工程中添加设备（添加 EtherCAT 设备），添加成功之后在主站下面添加从站设备 EM03DE-E4 模块编写各部分功能模块代码。
- (3) 配置从站设备的映射。
- (4) 编译并下载到控制器，运行程序。

5.2.1 EtherCAT 主从站连接

5.2.1.1 EtherCAT主站的添加及配置

打开 SMC BASIC STUDIO 编程软件之后，需要新建一个工程（详细建立工程过程请参考《BAC332E 用户使用手册》）。在该工程中会自动添加 EtherCAT 主站。主站的参数除了通讯周期时间之外，其他的参数不需要用户配置，保持默认即可。连接上控制器之后，在左侧“设备”栏，双击“EtherCAT_0”即可以看到主站的相关信息，如图 5.34 所示：

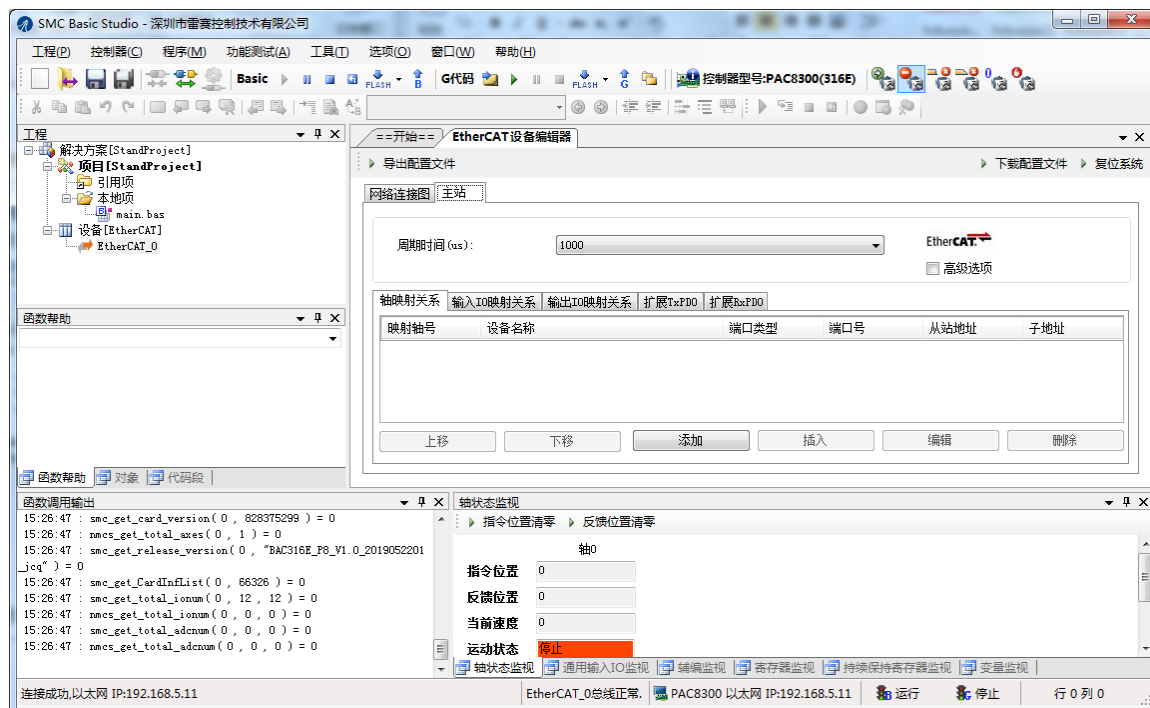


图 5.34 主站信息

5.2.1.2 模块的添加

在 SMC BASIC STUDIO 编程软件中，可以手动添加从站模块和自动扫描从站模块。在添加从站之前，必须保证设备库中有对应的模块设备描述文件，具体操作请参考《BAC332E 用户使用手册》里“安装设备描述文件”章节。

(1) 手动添加

在“工程”栏的目录里，选中主站“EtherCAT_0”，然后点击鼠标右键，选择“添加从站”在弹出的窗口中找到对应的设备描述文件，如图 5.35 所示：

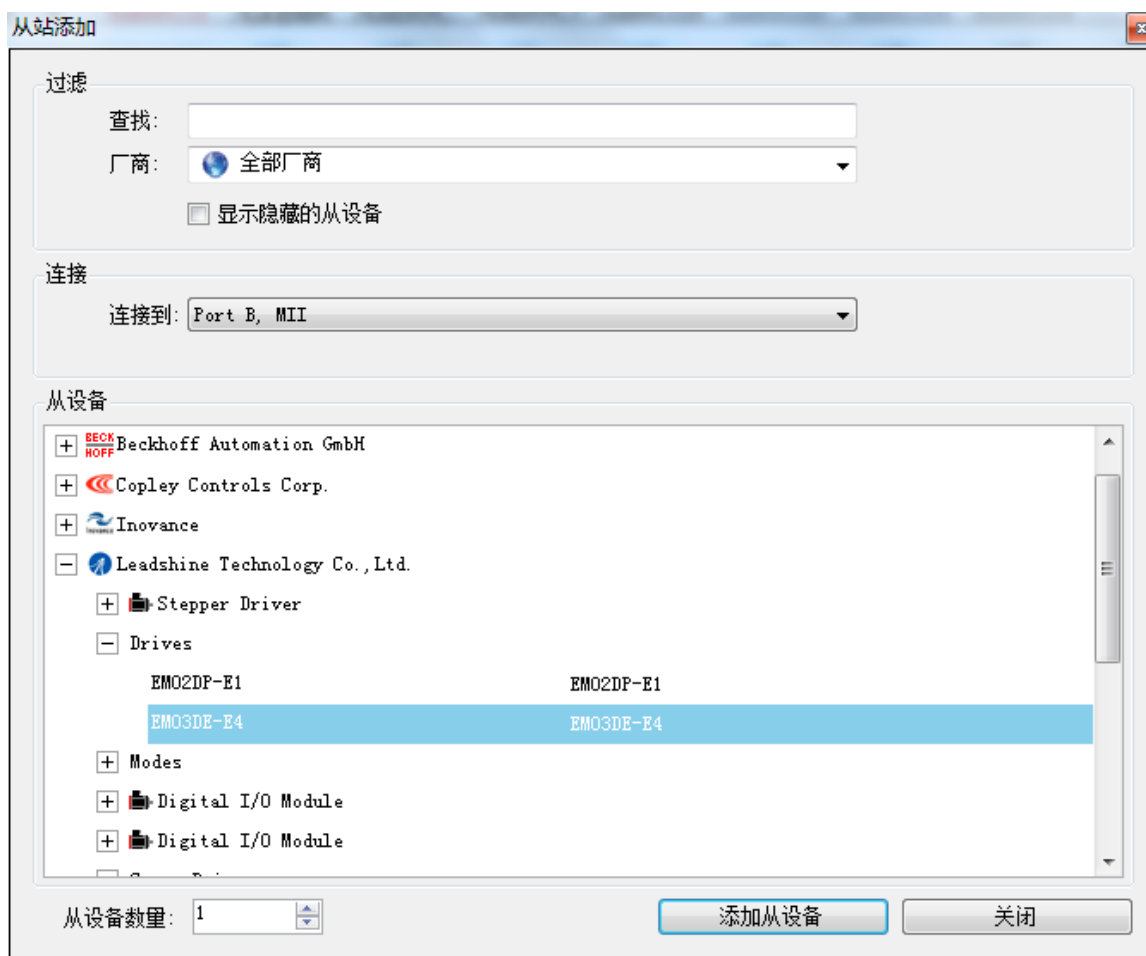


图 5.35 手动添加从站

然后选择“添加从设备”，在左侧“工程”目录下可以找到添加成功的模块。

(2) 自动扫描

在“工程”栏的目录里，选中主站“EtherCAT_0”，然后点击鼠标右键，选择“扫描设备”，扫描成功后会提示是否下载对应的配置文件，同时主站目录下会出现扫描到的从站模块，如图 5.36 所示

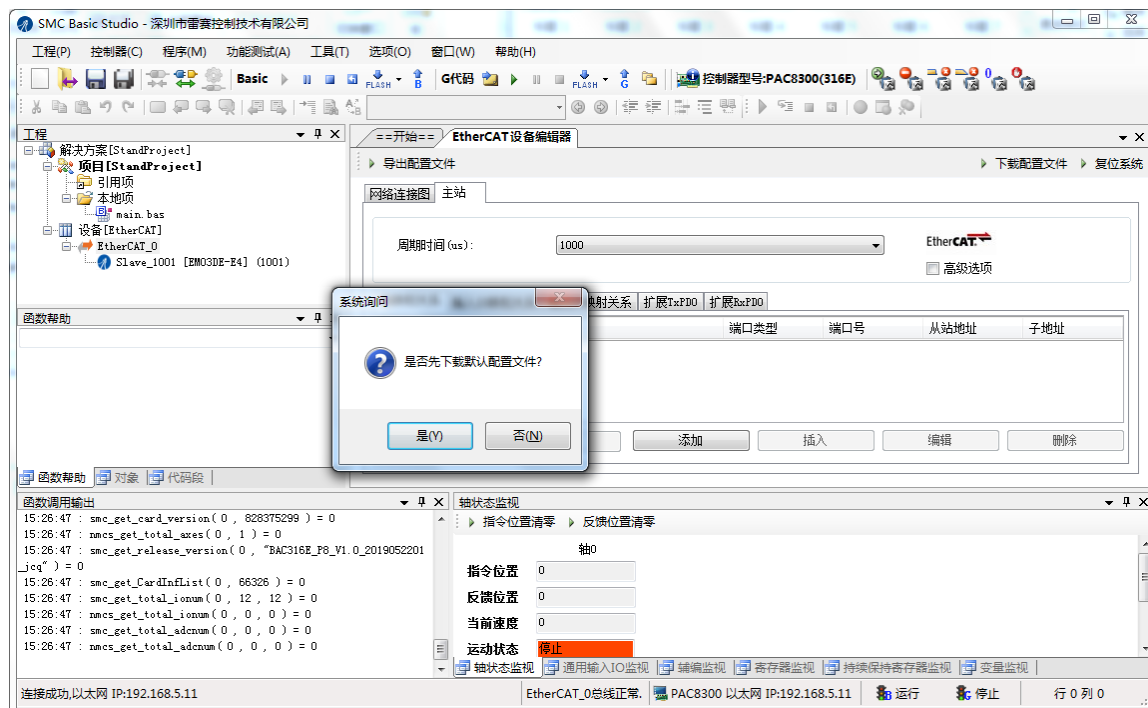


图 5.36 自动扫描添加

选择“是”；

下载成功后会重启系统，双击从站“Slave_1001 [EM03DE-E4] (1001)”，可以看到从站模块的信息，如图5.37所示

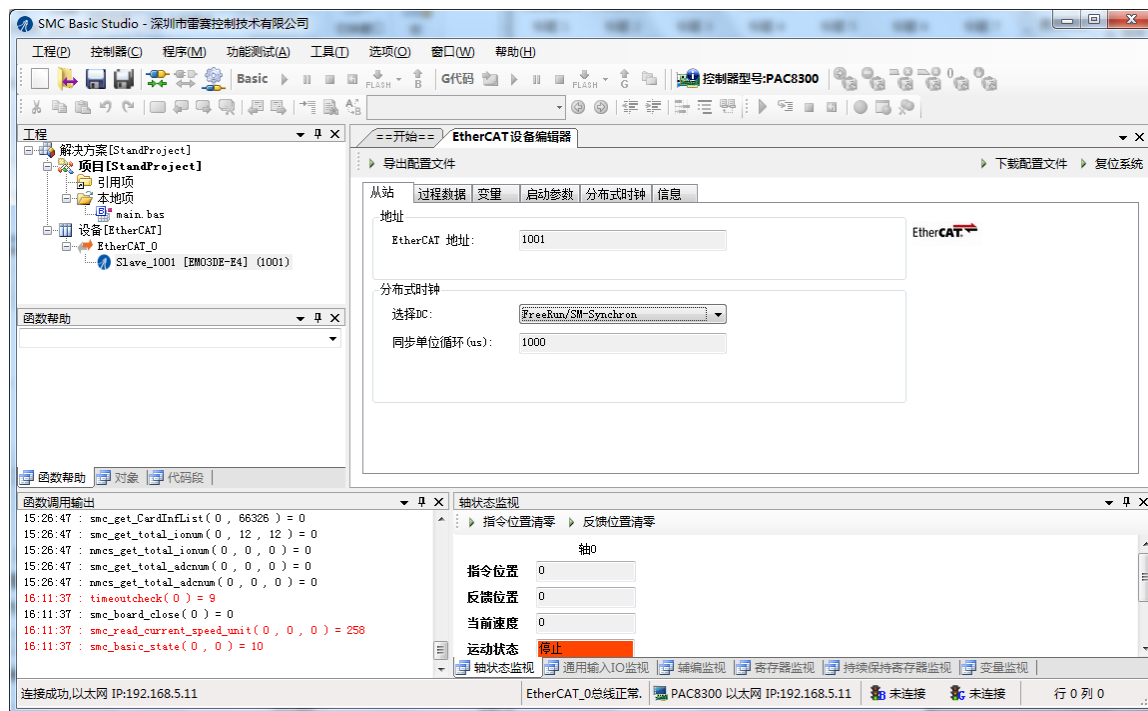


图 5.37 从站模块信息

在EtherCAT设备编辑器中，可以看到从站模块的所有信息，包括从站地址、同步时间周期、

PDO、时钟、模块信息等。从站的参数都是系统默认匹配的，不需要用户修改。如图5.38、5.39、5.40、5.41、5.42所示：



图 5.38 从站地址

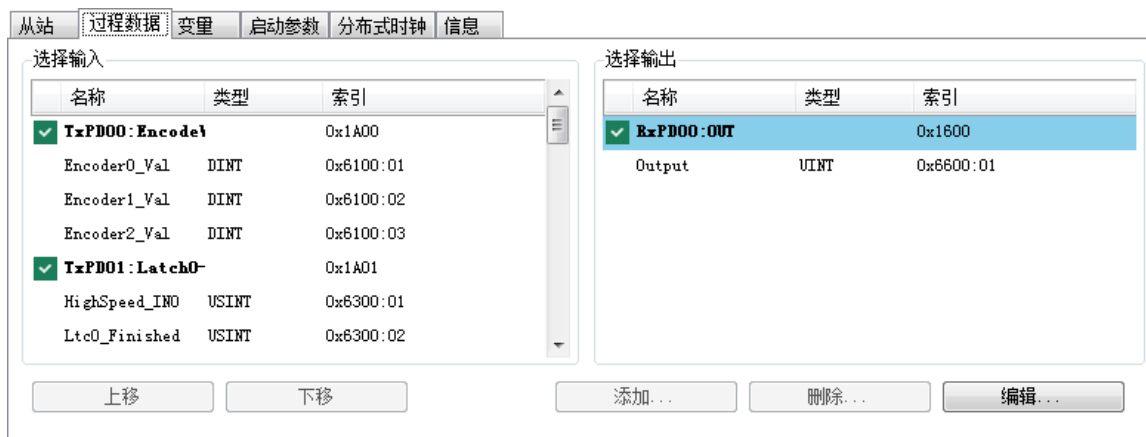


图 5.39 过程数据



图 5.40 变量

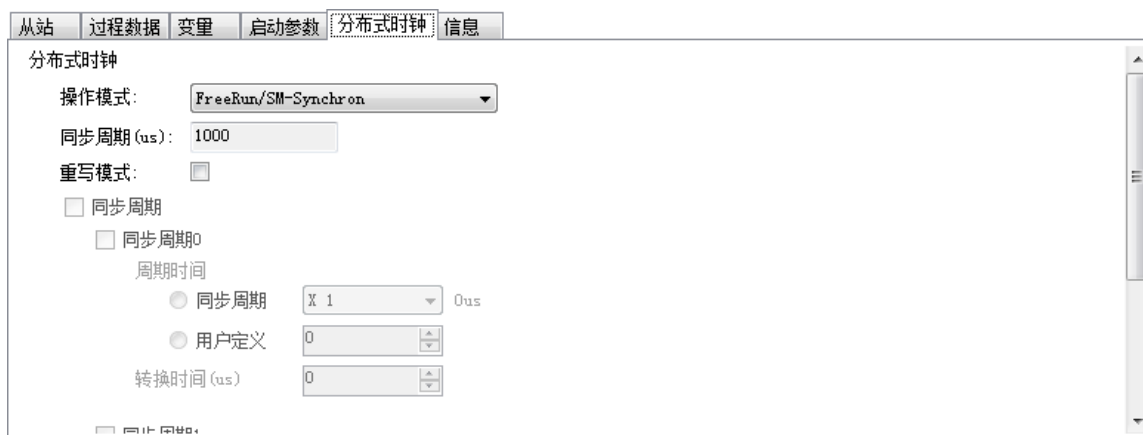


图 5.41 时钟

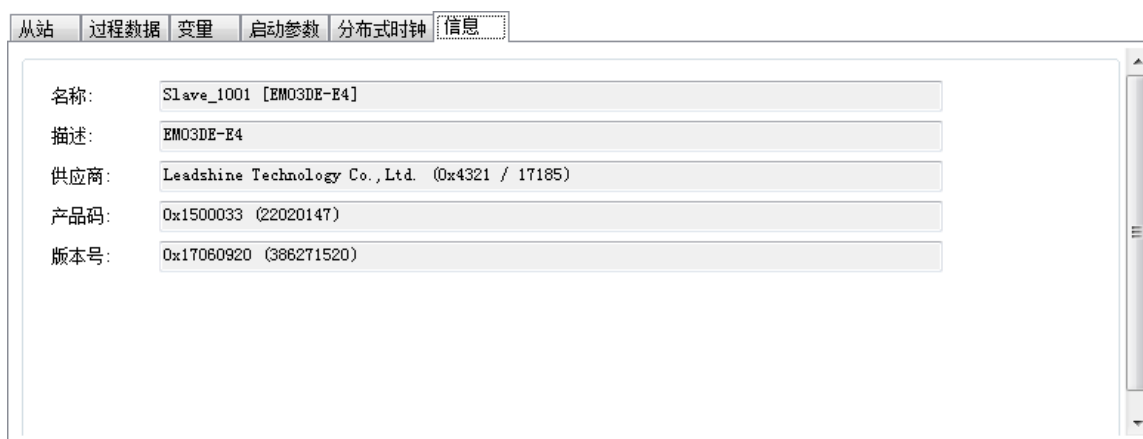


图 5.42 信息

至此，从站模块的添加已经完成。

5.2.1.3 映射模块扩展PDO

在 EtherCat 设备编辑器界面，选择“扩展 TxPDO”，点击“添加”，弹出“寄存器映射”窗口，在该窗口选择需要映射的变量，再点击确认“确认”。添加完需要的变量后，点击“下载配置文件”，完成变量映射。如图 5.43 所示。

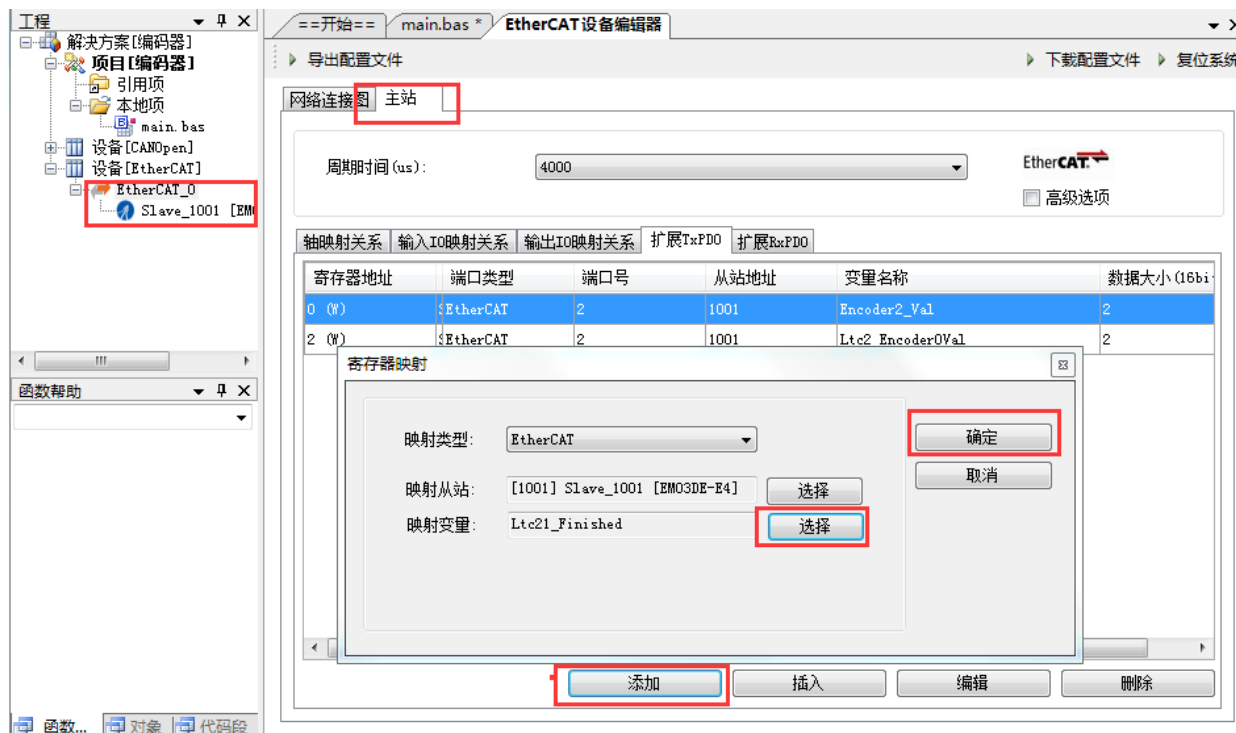


图 5.43 模块扩展 PDO 映射

5.2.2 高速计数功能使用

编码器参数设置步骤如下：

a. 编码器通道选择：

编码器 0 的对象字典索引为 16#6000 (24576)

编码器 1 的对象字典索引为 16#6001 (24577)

编码器 2 的对象字典索引为 16#6002 (24578)

b. 设置编码器计数模式：

计数模式的对象字典子索引为 1。

编码器有两种工作模式：0-AB 相模式；1-脉冲方向模式；

使用到的函数及其说明如图 5.44、5.45 所示。

NMCSSetNodeOd

语 法：short NMCSSetNodeOd(WORD PortNo,WORD NodeNum, WORD Index,WORD SubIndex,WORD ValLength,DWORD* Value)

描 述：设置从站对象字典

参 数：	PortNo	端口号 0-3
	NodeNum	节点号
	Index	索引
	SubIndex	子索引
	ValLength	值长度
	Value	主站值

返回值：错误代码

适用范围:全系列控制器

NMCSGetNodeOd

语 法：short NMCSGetNodeOd(WORD PortNo,WORD NodeNum, WORD Index,WORD SubIndex,WORD ValLength,DWORD* Value)

描 述：获取从站对象字典

参 数：PortNo 端口号 0-3
 NodeNum 节点号
 Index 索引
 SubIndex 子索引

 ValLength 值长度
 Value 主站值

返回值：错误代码

适用范围:全系列控制器

图 5.44

```
NMCSReadTxPDOExtra(WORD PortNum, WORD
address, WORD DataLen, DWORD* Value)
```

PortNum: 端口号, 0, 1表示CANOpen, 2, 3表示EtherCAT端口

address: 扩展PDO的首地址

DataLen: 数据长度, 按16bit计算, 最大值为2 (表示32bit数据)

Value: 数据值

图 5.45

示例代码如图 5.46 所示 (参考第 4 章对象字典表格, 写有 TxPDO 和 RxPDO 的对象在编程时使用对应的 PDO 操作函数: NMCSReadRxpdoExtra 和 NMCSReadTxpdoExtra。反之, 使用 nmcsetnodeod 和 nmcgetnodeod 函数)。

```

23 | .....编码器设置.....
24 |
25 | *编码器0的对象字典索引为16#6000 (24576)
26 | *编码器1的对象字典索引为16#6001 (24577)
27 | *编码器2的对象字典索引为16#6002 (24578)
28 | dim Index=24578      'Index 索引
29 |
30 | *Set Mode; 0-AB相模式; 1-脉冲方向模式,子索引为1
31 | dim Mvalue
32 | nmcssetnodeod( PortNo , NodeNum , Index , 1 , Vallength , 1)
33 | nmcsgetnodeod( PortNo , NodeNum , Index , 1 , Vallength , Mvalue )
34 | print "计数模式: ",Mvalue
35 | *Set AB Phase; 0-负方向; 1-正方向,子索引为2
36 | dim Pvalue
37 | nmcssetnodeod( PortNo , NodeNum , Index , 2 , Vallength , 0)
38 | nmcsgetnodeod( PortNo , NodeNum , Index , 2 , Vallength , Pvalue )
39 | print "AB相方向: "Pvalue
40 | *设置编码器初始值, -2147483648至2147483647,索引为3
41 | dim PO
42 | nmcssetnodeod( PortNo , NodeNum , Index , 3 , 32 , 0)
43 | nmcsgetnodeod( PortNo , NodeNum , Index , 3 , 32 , PO )
44 | print "编码器初始值: "PO

```

图 5.46 设置编码器计数模式、编码器方向、初始值

c. 设置编码器计数方向:

计数方向的对象字典子索引为 2。

计数方向有效值：0-负方向；1-正方向；

该方向，是指编码器计数模式为 AB 相时的计数方向；

示例代码如图 5.46 所示。

d. 设置编码器初始值:

编码器初始值的对象字典子索引为 3。

编码器初始值有效范围：-2147483648 至 2147483647（有符号的 32 位值）

示例代码如图 5.46 所示。

完成上述步骤后，即完成对编码器参数配置的代码编写。

编码器 2 的计数模式设置为脉冲+方向，编码器初始值为 0。

e. PDO 方式编码器值读取:

读取编码器值对象字典的索引为：6100H(24832)

子索引，编码器 0：01H

编码器 1：02H

编码器 2：03H

```

.....编码器值读取.....
'读取编码器值:索引: 6100H (24832), 编码器0值: 1H 编码器1值:2H 编码器3值:3H
dim Encodervalue, errcode
dim    ret, rtn

while Encodervalue<2000000000000000

    NMCSReadTxPDOExtra(2, 2, 2, Encodervalue )
    print "编码器的值: " Encodervalue |

wend

end sub

```

图 5.47 编码值读取

读取编码器 2 的值，在输出窗口打印：40000。

f. 运行结果：完成上述步骤后，程序运行结果如图 5.48 所示：

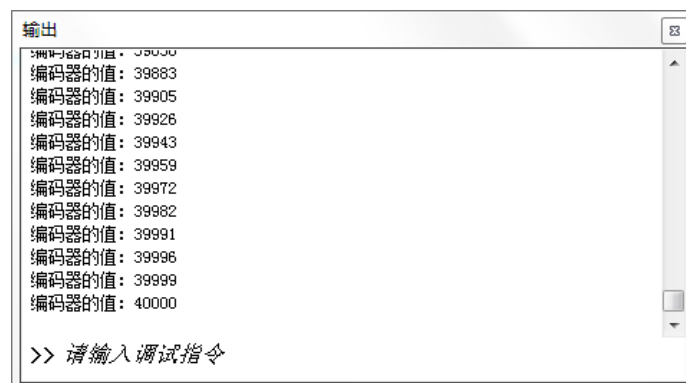


图 5.48 映射编码器实际计数值

5.2.3 高速锁存功能使用

5.2.3.1 单次锁存功能

a. 锁存器通道选择:

锁存器 0 的对象字典索引为 16#6200。

锁存器 1 的对象字典索引为 16#6201。

锁存器 2 的对象字典索引为 16#6202。

锁存器 3 的对象字典索引为 16#6203。

b. 清除锁存器的锁存标志:

清除锁存标志的对象字典索引为 1。

清除锁存标志的有效值为 1。

清除锁存标志的示例代码如图 5.49 所示。

```

176 | .....锁存参数设置.....
177 |
178 | 锁存器0的对象字典索引为16#6200(25088)
179 | 锁存器1的对象字典索引为16#6201(25089)
180 | 锁存器2的对象字典索引为16#6202(25090)
181 | 锁存器3的对象字典索引为16#6203(25091)
182 | dim IndexL=25090
183 | 清除锁存标志的对象字典索引为1, 有效值为:1
184 | dim clear
185 | nmcssetnodeod( PortNo , NodeNum , IndexL , 1 , Vallength , 1)
186 | nmcsgetnodeod( PortNo , NodeNum , IndexL , 1 , Vallength , clear )
187 | print clear

```

图 5.49 清楚锁存状态

c. 设置锁存器的锁存模式:

锁存模式的对象字典索引为 2。

锁存器有两种工作模式：0-单次锁存；1-连续锁存。

设置锁存模式的示例代码如图 5.50 所示。

```

189 | 锁存模式的对象字典索引为2 : 0-单次锁存; 1-连续锁存。
190 | dim Lmode
191 | nmcssetnodeod( PortNo , NodeNum , IndexL , 2 , Vallength , 0)
192 | nmcsgetnodeod( PortNo , NodeNum , IndexL , 2 , Vallength , Lmode )
193 | print Lmode

```

图 5.50 锁存器模式

d. 设置锁存器的锁存逻辑:

设置锁存逻辑的对象字典索引为 3。

锁存逻辑有三种：0-电平上升沿；1-电平下降沿；2-任意沿锁存。

设置锁存逻辑的示例代码如图 5.51 所示。

```

194 | '设置锁存逻辑的对象字典索引为3 : 0-电平上升沿; 1-电平下降沿; 2-任意沿锁存
195 | dim Lcmode
196 | nmcssetnodeod( PortNo , NodeNum , IndexL , 3 , Vallength , 1)
197 | nmcsgetnodeod( PortNo , NodeNum , IndexL , 3 , Vallength , Lcmode )
198 | print Lcmode

```

图 5.51 设置锁存逻辑

e. 设置锁存器的滤波时间:

设置滤波时间的对象字典索引为 4。

设置滤波时间，单位 us。最小 0us~最大 65535us。

设置滤波时间的示例代码如图 5.52 所示。

```

199 | '设置滤波时间的对象字典索引为4 最小0us~最大65535us
200 |
201 | dim Ftime
202 | nmcssetnodeod( PortNo , NodeNum , IndexL , 4 , Vallength , 1)
203 | nmcsgetnodeod( PortNo , NodeNum , IndexL , 4 , Vallength , Ftime )
204 | print Ftime

```

图 5.52 设置滤波时间

完成上述步骤后，即完成对锁存器参数配置的代码编写。

锁存器 2, 锁存模式为: 单次锁存, 锁存逻辑为下降沿锁存, 滤波时间设置为 1us。编码器设置与 5.2.2 设置相同。

f. 单次锁存值读取:

'锁存器 0 状态读取的对象字典索引为 16#6300H (25344)

'锁存器 1 状态读取的对象字典索引为 16#6301 (25345)

'锁存器 2 状态读取的对象字典索引为 16#6302 (25346)

'锁存器 3 状态读取的对象字典索引为 16#6303 (25347)

使用到的函数及其说明如图 5.53 所示

NMCSReadTxpdoExtra
NMCSReadTxpdoExtra(WORD PortNo,WORD address,WORDDataLen,DWORD* Value)

功能：读扩展 txpdo

参数： ConnectNo 链接号： 0-7 号，默认值 0

PortNum: 端口号， 0,1 表示 CANOpen, 2, 3 表示 EtherCAT 端口

address: 扩展 PDO 的首地址

DataLen: 数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）

Value: 数据值

返回值:错误码

图 5.53

单次锁存值读取的示例代码如图 5.54 所示。

```

211 IndexLR=25346
212
213 WHILE 1
214
215 '读取高速输入0状态默认显示值: FALSE (电平状态为24V) 子索引01H
216 dim HighSpeed_IN2
217 nmcsreadtxpdoextra( PortNo , 5, 1 , HighSpeed_IN2 )
218 print "HighSpeed_IN2:"HighSpeed_IN2
219 '锁存器0锁存完成标志,TRUE: 锁存完成 子索引:02H
220 dim Ltc2_Finished
221 nmcsreadtxpdoextra( PortNo , 4, 1 , Ltc2_Finished )
222 print "Ltc2_Finished:"Ltc2_Finished
223 dim Ltc2_Encoder2Val
224 nmcsreadtxpdoextra( PortNo , 6, 2 , Ltc2_Encoder2Val )
225 print "Ltc2_Encoder2Val:"Ltc2_Encoder2Val
226 if Ltc2_Finished=1 then
227     print Ltc2_Encoder2Val
228     goto min()
229 endif
230
231 WEND

```

图 5.54 单次锁存值读取

g. 运行结果：完成以上步骤后，程序运行结果如图 5.55 所示：

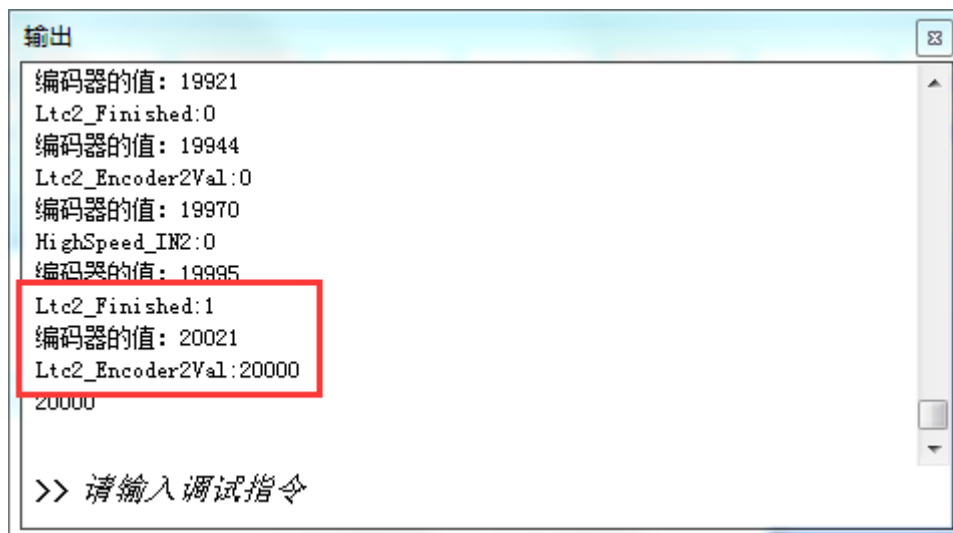


图 5.55 单次锁存的锁存值

该示例的高速锁存功能是用同一模块高速比较输出给锁存输入信号实现的，所以，锁存编码器为 2，锁存值：20000。

5.2.3.2 连续锁存功能

锁存器参数设置步骤： 见单次锁存功能 5.2.3.1，需将步骤 c 设置为连续锁存模式。

a. 读取锁存器的选择：

读取锁存器 0 的对象字典索引为 16#6310。

读取锁存器 1 的对象字典索引为 16#6311。

读取锁存器 2 的对象字典索引为 16#6312。

读取锁存器 3 的对象字典索引为 16#6313。

b. 锁存器的状态读取：

读取锁存编码器 0 值的个数-对象字典子索引为 1。

读取锁存编码器 0 值-对象字典子索引为 2。

读取锁存编码器 1 值的个数-对象字典子索引为 3。

读取锁存编码器 1 值-对象字典子索引为 4。

读取锁存编码器 2 值的个数-对象字典子索引为 5。

读取锁存编码器 2 值-对象字典子索引为 6。

读取锁存器状态的示例代码如图 5.56 所示。

```

246 dim IndexLRL=25362
247 i=0
248
249 while 1
250     ret1=SMCReadoutBit(0)
251     if ret1=0 then
252         '读取锁存器2锁存编码器2值的个数
253         dim Ltc2_FIF2_EncoderONum
254         nmcsgetnodeod( PortNo , NodeNum , IndexLRL , 5 , 32 , Ltc2_FIF2_EncoderONum )
255         print "Ltc2_FIF2_EncoderONum:"Ltc2_FIF2_EncoderONum
256         '读取锁存器2锁存编码器2值
257         dim Ltc2_FIFO_Encoder2Val,array1(4)
258         nmcsgetnodeod( PortNo , NodeNum , IndexLRL , 6 , 32 , Ltc2_FIFO_Encoder2Val )
259         print "Ltc2_FIFO_Encoder2Val:"Ltc2_FIFO_Encoder2Val

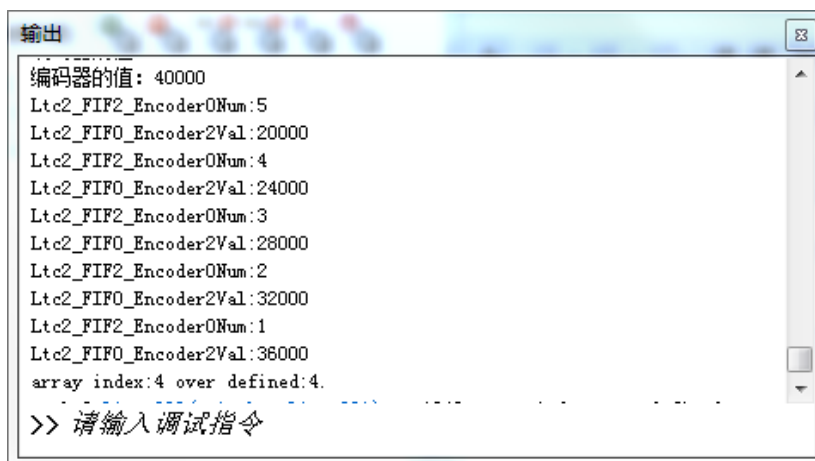
```

图 5.56 锁存值个数、锁存值读取

完成上述步骤后，即完成锁存器状态读取配置的代码编写。

读取锁存器 2，锁存编码器 2，锁存值的个数以及锁存值。

c. 运行结果：完成以上步骤后，程序运行结果图 5.57 所示：



```
输出
编码器的值: 40000
Ltc2_FIFO_Encoder0Num:5
Ltc2_FIFO_Encoder2Val:20000
Ltc2_FIFO_Encoder0Num:4
Ltc2_FIFO_Encoder2Val:24000
Ltc2_FIFO_Encoder0Num:3
Ltc2_FIFO_Encoder2Val:28000
Ltc2_FIFO_Encoder0Num:2
Ltc2_FIFO_Encoder2Val:32000
Ltc2_FIFO_Encoder0Num:1
Ltc2_FIFO_Encoder2Val:36000
array index:4 over defined:4.
>> 请输入调试指令
```

图 5.57 连续锁存值

该示例的高速锁存功能是用同一模块高速比较输出给锁存输入信号是实现的，所以，锁存编码器为 2，锁存个数：5，锁存值为：20000, 24000, 28000, 32000, 36000，如图 5.57 所示。

5.2.4 高速比较功能使用

高速比较功能是在高速计数功能的基础上运行的（高速计数功能使用参考 5.2.2 节）。

比较器参数设置步骤如下：

a. 比较器通道选择：

比较器 0 的对象字典索引为 16#6400（25600）

比较器 1 的对象字典索引为 16#6401（25601）

比较器 2 的对象字典索引为 16#6402（25602）

b. 清除比较器的比较状态：

清除比较状态的对象字典子索引为 1。

清除比较状态的有效值为 1。

设置清除比较状态的示例代码如图 5.58 所示。

```

75 dim IndexC=25602
76
77 '清除比较器缓冲区及比较状态 1:清除 子索引:01H
78 dim Cvalue
79 nmcssetnodeod( PortNo , NodeNum , IndexC , 1 , ValLength , 1)
80 nmcsgetnodeod( PortNo , NodeNum , IndexC , 1 , ValLength , Cvalue )
81 print "清零: " Cvalue

```

图 5.58 清除缓冲区以及比较状态

c. 设置比较器的比较模式：

比较模式的对象字典子索引为 2。

比较器有六种工作模式：0-关闭，1-等于，2-小于，3-大于，4-FIFO（队列），5-Liner（线性）

备注：当比较器工作模式为 0-关闭时，输出口味普通输出口。

设置比较模式的示例代码如图 5.59 所示。

```

82 '设置比较器工作模式: 0: 关闭,,1: 等于,,2: 小于,,3: 大于,,4: fifo,5: linear, 子索引:02H
83 dim CMvalue
84 nmcssetnodeod( PortNo , NodeNum , IndexC , 2 , ValLength , 5)
85 nmcsgetnodeod( PortNo , NodeNum , IndexC , 2 , ValLength , CMvalue )
86 print "比较模式: " CMvalue

```

图 5.59 比较器比较模式

d. 选择编码器通道：

选择编码器通道的对象字典子索引为 3。

编码器通道有三个：0-编码器 0；1-编码器 1；2-编码器 2。

选择编码器通道的示例代码如图 5.60 所示。

```

87 选择编码器通道可选择通道: 0、1、2      子索引: 03H
88  dim Evalue
89  nmcsetnodeod( PortNo , NodeNum , IndexC , 3 , ValLength , 2)
90  nmcgetnodeod( PortNo , NodeNum , IndexC , 3 , ValLength , Evalue )
91  print "编码器通道:" Evalue

```

图 5.60 编码器通道

e. 设置比较器输出逻辑:

比较器输出逻辑的对象字典子索引为 4。

比较器输出逻辑有两种：“0”-条件成立输出低电平，显示状态为 1（TRUE）；

“1”-条件成立输出高电平，显示状态为 0（FALSE）。

设置输出逻辑的示例代码如图 5.61 所示。

```

92 设置比较器输出逻辑：“0”：条件成立输出低电平,回读输出状态为TRUE  “1”：条件成立输出高电
93  dim Lvalue
94  nmcsetnodeod( PortNo , NodeNum , IndexC , 4 , ValLength , 0)
95  nmcgetnodeod( PortNo , NodeNum , IndexC , 4 , ValLength , Lvalue )
96  print "输出有效电平:" Lvalue

```

图 5.61 比较器输出逻辑

f. 设置比较器输出电平时间:

比较器输出电平时间的对象字典子索引为 5；

电平时间的有效范围为：0 至 0- 85899345（无符号的 32 位值）单位：us

设置输出电平时间的示例代码如图 5.62 所示。

```

97 设置输出有效电平时间      子索引: 05H
98  dim Tvalue
99  nmcsetnodeod( PortNo , NodeNum , IndexC , 5 , 32 , 10000)
100 nmcgetnodeod( PortNo , NodeNum , IndexC , 5 , 32 , Tvalue )
101 print "输出有效电平时间:" Tvalue

```

图 5.62 比较器输出电平有效时间

注意：比较器输出电平有效时间只对于比较模式 4--队列比较与比较模式 5--线性比较有效。

g. 添加比较值点:

添加比较点的对象字典子索引为 6。

比较点的有效范围：-2147483648 至 2147483647（有符号的 32 位值）

添加比较点的示例代码如图 5.63 所示。

```

102 '添加比较点(值)          子索引: 06H
103 dim P1
104 nmcssetnodeod( PortNo , NodeNum , IndexC , 6 , 32 , 20000)
105 nmcsgetnodeod( PortNo , NodeNum , IndexC , 6 , 32 , P1)
106 print "添加比较点:" P1

```

图 5.63 添加比较点(值)

h. 设置比较点数量:

设置比较点数量的对象字典子索引为 7。

比较点数量的有效范围: 0 至 65535 (无符号的 32 位值)

设置比较点数量的示例代码如图 5.64 所示。

```

119 '比较器采用线性比较时,设置比较点数量      子索引: 07H
120 dim N
121 nmcssetnodeod( PortNo , NodeNum , IndexC , 7 , 32 , 5)
122 nmcsgetnodeod( PortNo , NodeNum , IndexC , 7 , 32 , N )
123 print "比较点个数:" N

```

图 5.64 设置比较点数量

注意: 设置比较点数量只在比较模式 5—线性比较时有效。

i. 设置比较点增量值:

设置增量值的对象字典子索引为 8。

比较点增量值的有效范围: -2147483648 至 2147483647 (有符号的 32 位值)

设置比较点增量值的示例代码如图 5.65 所示。

```

124 '比较器0采用线性比较,设置比较点增量值      子索引: 08H
125 dim V
126 nmcssetnodeod( PortNo , NodeNum , IndexC , 8 , 32 , 4000)
127 nmcsgetnodeod( PortNo , NodeNum , IndexC , 8 , 32 , V )
128 print "比较点增量:" V

```

图 5.65 设置比较点增量值

注意: 设置比较点增量只在比较模式 5—线性比较时有效。

完成上述步骤后,即完成对比较器参数的配置的代码编写。

比较器 0, 比较模式: 5(线性比较), 比较输出逻辑: 0(低电平), 比较通道为编码器 2, 输出电平时间: 10ms, 比较点个数: 5, 比较值增量: 4000。编码器的设置与 5.2.2 设置相同

j. 运行结果: 完成以上步骤后,程序运行结果如图 5.66 所示(该图为实验测试仪器采集数据后绘制而成):

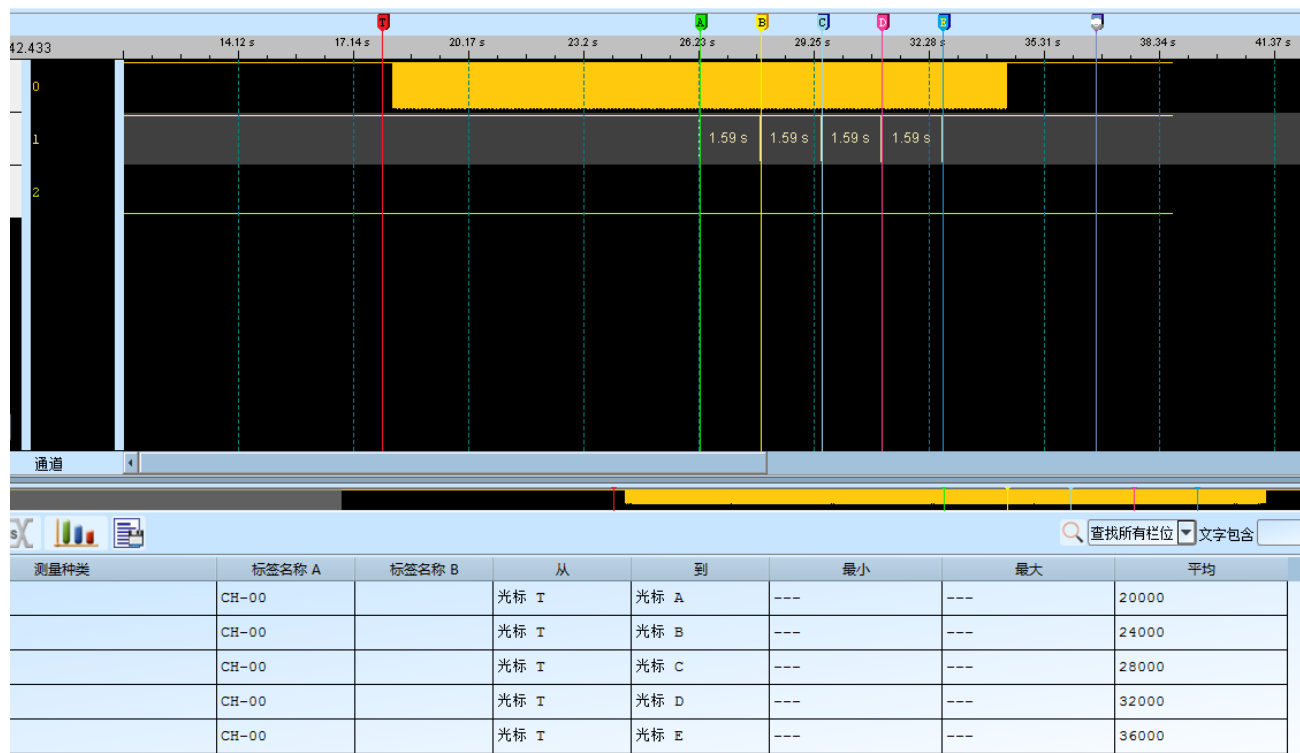


图 5.66 实际计数值，比较值

由图5.66可看出，比较器0，比较模式：5（线性比较），第一个比较点编码器值为：20000，比较增量为：40000，比较点分别为：20000, 24000, 28000, 32000, 36000。

5.3 控制卡示例

下面将讲述 EtherCAT 编码器模块 EM03DE-E4 与控制卡 DMC-E3032 的配套使用，主要演示编码器计数功能、高速锁存功能和高速比较功能，主要步骤如下：

- 1) 将控制器的 EtherCAT 口与扩展模块的 EtherCAT 口使用网线连接，连接 24V 电源。
- 2) 打开 motion3.0 软件，扫描控制卡，然后在总线配置界面扫描添加 EM03DE-E4 模块。
- 3) 手动添加映射模块的扩展 PDO。
- 4) 打开 VS2010，编写例程，实现模块功能。

5.3.1 EtherCAT 主从站连接

5.3.1.1 添加模块设备描述文件

打开 motion3.0 软件,扫描控制卡,然后在总线配置界面,鼠标右键单击“设备[EtherCat]”添加 EM03DE-E4 模块的设备描述文件,如图 5.67, 5.68 所示:



图 5.67

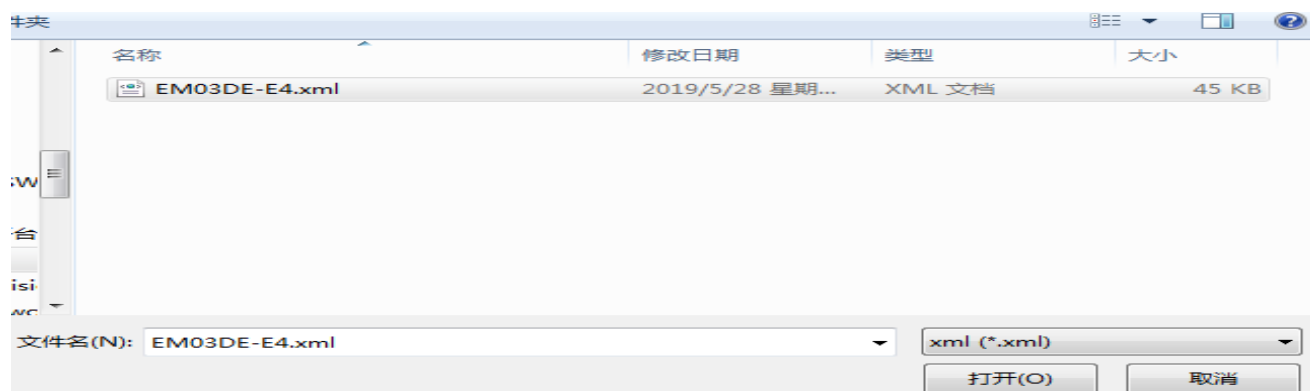


图 5.68 添加模块设备描述文件

5.3.1.2 扫描从站

添加模块的设备描述文件后,鼠标右键单击“EtherCat Master Unit”,扫描设备如图 5.69 所示:

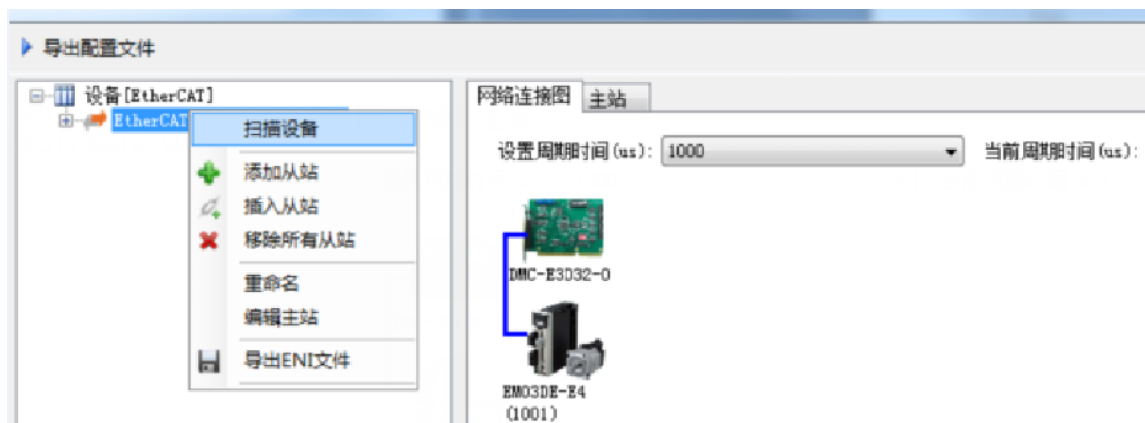


图 5.69 扫描模块

5.3.1.3 映射模块扩展PDO

在 EtherCat 总线配置界面的主站界面，选择“扩展 TxPDO”，点击“添加”，弹出“寄存器映射”窗口，在该窗口选择需要映射的变量，再点击确认“确定”。添加完需要的变量后，点击“下载配置文件”，完成变量映射，如图 5.70 所示。

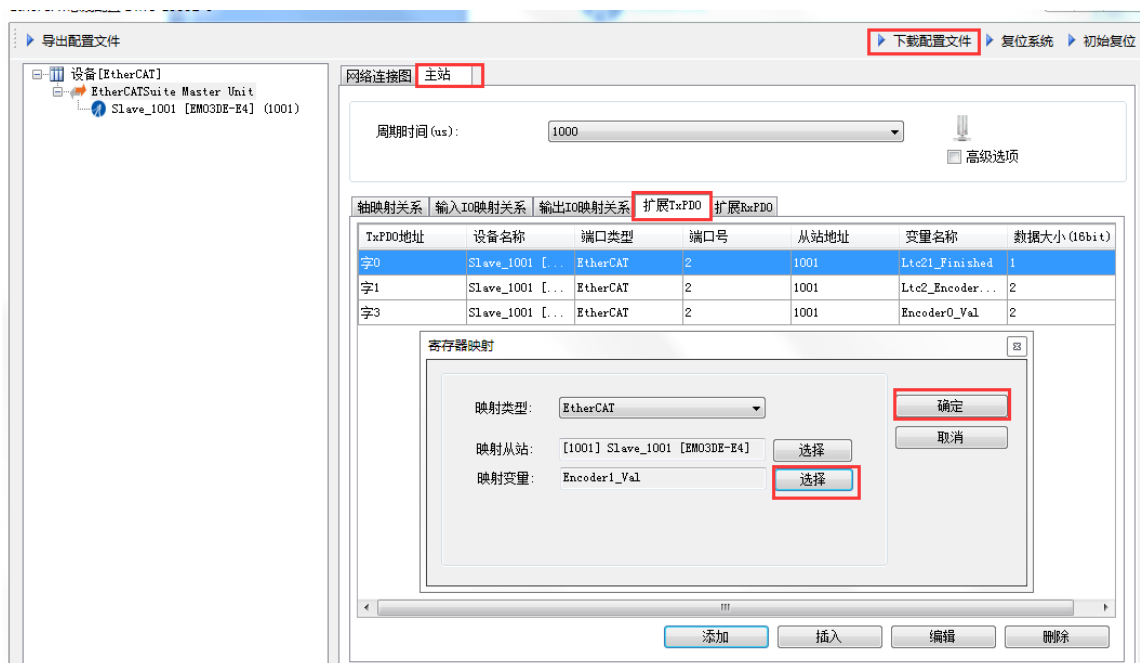


图 5.70 变量映射

5.3.2 高速计数功能使用

编码器参数设置步骤如下：

a. 编码器通道选择：

编码器 0 的对象字典索引为 16#6000 (24576)

编码器 1 的对象字典索引为 16#6001 (24577)

编码器 2 的对象字典索引为 16#6002 (24578)

b. 设置编码器计数模式：

计数模式的对象字典子索引为 1。

编码器有两种工作模式：0-AB 相模式；1-脉冲方向模式；

使用到的函数及其说明如图 5.71、5.72 所示

```
short nmc_set_node_od(WORD CardNo, WORD PortNum, WORD NodeNum, WORD Index, WORD
SubIndex, WORD ValLength, DWORD Value)
```

功 能：设置从站对象字典参数值

参 数：CardNo 控制卡卡号

PortNum EtherCAT 端口号，固定为 2

NodeNum 从站 EtherCAT 地址，第 i 个 EtherCAT 从站地址为 1000+i

Index 对象字典索引

SubIndex 对象字典子索引

ValLength 对象字典索引长度(单位：bit)

Value 对象字典索引参数值

返回值：错误代码

```
short nmc_get_node_od(WORD CardNo, WORD PortNum, WORD NodeNum, WORD Index, WORD
SubIndex, WORD ValLength, DWORD* Value)
```

功 能：读取从站对象字典参数值

参 数：	CardNo	控制卡卡号
	PortNum	EtherCAT 端口号，固定为 2
	NodeNum	从站 EtherCAT 地址，第 i 个 EtherCAT 从站为 1000+i
	Index	对象字典索引
	SubIndex	对象字典子索引
	ValLength	对象字典索引长度(单位：bit)
	Value	对象字典索引参数值

返回值：错误代码

图 5.71

```
short nmc_read_txpdo_extra(WORD CardNo, WORD PortNum, Word address, Word DataLen, int
* Value)
```

功 能：读取从站扩展有符号 TxPDO 值

参 数：	CardNo	控制卡卡号
	PortNum	EtherCAT 端口号，固定为 2
	address	扩展 PDO 的首地址
	DataLen	数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）
	Value	数据值

返回值：错误代码

图 5.72

示例代码如图 5.73 所示（参考第 4 章对象字典表格，写有 TxPDO 和 RxPDO 的对象在编程时使用对应的 PDO 操作函数：nmc_read_rxpdo_extra 和 nmc_read_txpdo_extra。反之，使用 nmc_get_node_od 和 nmc_set_node_od 函数）。

```

309     {
310         //编码器0的对象字典索引为16#6000 (24576)
311         //编码器1的对象字典索引为16#6001 (24577)
312         //编码器2的对象字典索引为16#6002 (24578)
313         ushort PortNum = 2;
314         ushort nodenum=Convert.ToUInt16(textBox5.Text ); //从站ID
315         ushort index = Convert.ToUInt16(textBox17.Text ); //索引
316         int Mvalue= Convert.ToUInt16(textBox18.Text ); //编码器计数模式
317         int Mvalue0 = 3;
318         int Pvalue0 = 3;
319         //int P0 = 3;
320         int Pvalue= Convert.ToUInt16(textBox22.Text ); // AB Phase
321         // Set Mode; 0-AB相模式; 1-脉冲方向模式,子索引为1
322         LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 1, valuelength, Mvalue);
323         LTDMC.nmc_get_node_od(_CardID, PortNum, nodenum, index, 1, valuelength, ref Mvalue0);
324
325         // Set AB Phase; 0-负方向; 1-正方向,子索引为2
326         LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 2, valuelength, Pvalue);
327         LTDMC.nmc_get_node_od(_CardID, PortNum, nodenum, index, 2, valuelength, ref Pvalue0);
328     }

```

图 5.73 编码器计数模式、计数方向设置

c. 设置编码器计数方向:

计数方向的对象字典子索引为 2。

计数方向有效值: 0-负方向; 1-正方向; 仅对 AB 相有效。

示例代码如图 5.73 所示。

d. 设置编码器初始值:

编码器初始值的对象字典子索引为 3。

编码器初始值有效范围: -2147483648 至 2147483647 (有符号的 32 位值)

示例代码如图 5.74 所示

```

331     {
332         ushort PortNum = 2;
333         ushort nodenum = Convert.ToUInt16(textBox5.Text); //从站ID
334         ushort index = Convert.ToUInt16(textBox17.Text); //索引
335         int P0 = 3;
336
337         // 设置编码器初始值, -2147483648至2147483647, 子索引为3
338         LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 3, 32, 0);
339         LTDMC.nmc_get_node_od(_CardID, PortNum, nodenum, index, 3, 32, ref P0);
340     }
341

```

图 5.74 编码器初始值设置

完成上述步骤后, 即完成对编码器参数配置的代码编写。

e. PDO 方式编码器值读取:

读取编码器值对象字典的索引为: 6100H(24832)

子索引, 编码器 0: 01H

编码器 1: 02H

编码器 2: 03H

```

//编码器读取
//读取编码器值:索引: 6100H (24832), 编码器0值: 1H  编码器1值:2H  编码器2值:3H
ushort PortNum = 2;
uint value = 0;
//ushort nodenum = Convert.ToInt16(textBox5.Text); //从站ID
//ushort subindex = Convert.ToInt16((Convert.ToInt16(textBox36.Text) + 1)); //编码器0值: 1H  编码器1值:2H  编码器2值:3H
//LTDMC.nmc_get_node_od(_CardID, 2, 1001, 24832, 1, 32, ref value);
LTDMC.nmc_read_txpdo_extra(_CardID, PortNum, 0, 2, ref value);
textBox23.Text = value.ToString();

```

图 5.75

在图 5.76 的界面中设置编码器的参数，



图 5.76 编码器参数设置及读取

f. 运行结果:

编码器 0，计数模式设置为 AB 相，编码器初始值为 0，计数脉冲个数：266553

5.3.3 高速锁存功能使用

高速锁存功能是在高速计数功能的基础上运行的（高速计数功能使用参考 5.3.2 节）。

5.3.3.1 单次锁存功能

锁存器参数设置步骤如下：

a. 锁存器通道选择:

锁存器0的对象字典索引为16#6200 (25088)

锁存器1的对象字典索引为16#6201 (25089)

锁存器2的对象字典索引为16#6202 (25090)

锁存器3的对象字典索引为16#6203 (25091)

b. 清除锁存器的锁存标志:

清除锁存标志的对象字典子索引为 1。

清除锁存标志的有效值为 1。

清除锁存标志的示例代码如图 5.77 所示。

```

440      // 清除锁存标志的对象字典索引为1,      有效值为:1
441      ushort PortNum = 2;
442      ushort nodenum = Convert.ToUInt16(textBox5.Text); //从站ID
443      ushort index = Convert.ToUInt16(textBox31.Text);
444      int CL = 0;
445      LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 1, 8, 1);
446      LTDMC.nmc_get_node_od(_CardID, PortNum, nodenum, index, 1, 8, ref CL);
447

```

图 5.77 清楚锁存状态

c. 设置锁存器的锁存模式:

锁存模式的对象字典子索引为 2。

锁存器有两种工作模式：0-单次锁存；1-连续锁存。

设置锁存模式的示例代码如图 5.78 所示。

```

427      //锁存模式的对象字典子索引为2      : 0-单次锁存; 1-连续锁存
428      int Lmode= Convert.ToUInt16(textBox32.Text);
429      LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 2, 8, Lmode);
430      // 设置锁存逻辑的对象字典索引为3      : 0-电平上升沿; 1-电平下降沿; 2-任意沿锁存
431      int Lcmode= Convert.ToUInt16(textBox33.Text);
432      LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 3, 8, Lcmode);
433      // 设置滤波时间的对象字典索引为4      最小0us~最大65535us
434      int Ftime= Convert.ToUInt16(textBox34.Text);
435      LTDMC.nmc_set_node_od(_CardID, PortNum, nodenum, index, 4, 32, Ftime);

```

图 5.78 锁存模式、锁存逻辑、滤波时间设置

d. 设置锁存器的锁存逻辑:

设置锁存逻辑的对象字典子索引为 3。

锁存逻辑有三种：0-电平上升沿；1-电平下降沿；2-任意沿锁存。

设置锁存逻辑的示例代码如图 5.78 所示。

e. 设置锁存器的滤波时间:

设置滤波时间的对象字典子索引为 4。

设置滤波时间，单位 us。最小 0us~最大 65535us。

设置滤波时间的示例代码如图 5.78 所示。

完成上述步骤后，即完成对锁存器参数配置的代码编写。

f. 读取锁存值:

锁存器 0 状态读取的对象字典索引为 16#6300H (25344)

锁存器 1 状态读取的对象字典索引为 16#6301H (25345)

锁存器 2 状态读取的对象字典索引为 16#6302H (25346)

锁存器 3 状态读取的对象字典索引为 16#6303H (25347)

使用到的函数及其说明如图 5.79 所示

```
short nmc_read_txpdo_extra(WORD CardNo, WORD PortNum, Word address, Word DataLen, int
* Value)
```

功 能：读取从站扩展有符号 TxPDO 值

参 数：	CardNo	控制卡卡号
	PortNum	EtherCAT 端口号，固定为 2
	address	扩展 PDO 的首地址
	DataLen	数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）
	Value	数据值

返回值：错误代码

图 5.79

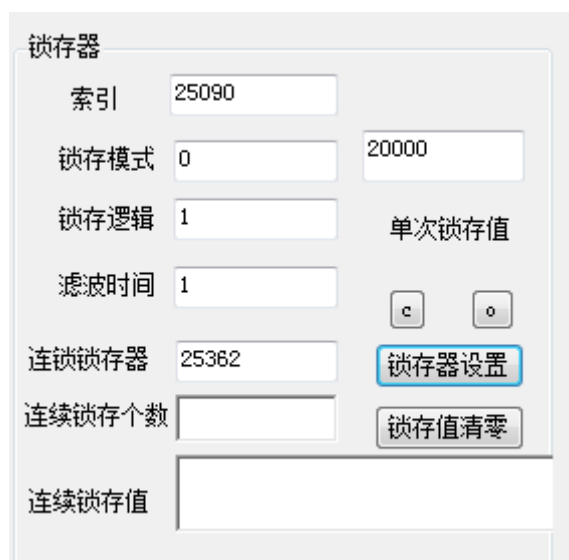
```

94 |         ushort index = 25346; // Convert.ToInt16(textBox37.Text);
95 |         LTDMC.nmc_get_node_od(_CardID, PortNum, nodenum, index, 2, 8, ref Lmode);
96 |         if (Lmode == 0)
97 |         {
98 |             LTDMC.nmc_read_txpdo_extra(_CardID, PortNum, 2, 1, ref Ltc0_Finished);
99 |             LTDMC.nmc_read_txpdo_extra(_CardID, PortNum, 0, 2, ref Ltc0_Encoder2Val);
100 |
101 |             if (Ltc0_Finished == 1)
102 |             {
103 |                 textBox35.Text = Ltc0_Encoder2Val.ToString();
104 |             }
105 |         }

```

图 5.80 锁存值读取

在图 5.81 的界面中设置锁存器的参数，



锁存器	
索引	25090
锁存模式	0
锁存逻辑	1
滤波时间	1
连锁锁存器	25362
连续锁存个数	
连续锁存值	

单次锁存值: 20000

按钮: 锁存器设置, 锁存值清零

图 5.81 锁存器参数设置及读取

锁存器 2，锁存模式：0（单次锁存），锁存逻辑：1（下降沿锁存），滤波时间设置为 1us。
编码器设置与 5.3.2 设置相同。

g. 运行结果：

该示例的高速锁存功能是用同一模块高速比较输出给锁存输入信号实现的，所以，锁存编码器为 2，锁存值：20000。

5.3.3.2 连续锁存功能

锁存器参数设置步骤： 见单次锁存功能 5.3.3.1，需将步骤 c 设置为连续锁存模式。

a. 读取锁存器的选择：

读取锁存器 0 的对象字典索引为 16#6310 (25360)

读取锁存器 1 的对象字典索引为 16#6311 (25361)

读取锁存器 2 的对象字典索引为 16#6312 (25362)

读取锁存器 3 的对象字典索引为 16#6313 (25363)

b. 锁存器的状态读取：

读取锁存编码器 0 值的个数-对象字典子索引为 1。

读取锁存编码器 0 值-对象字典子索引为 2。

读取锁存编码器 1 值的个数-对象字典子索引为 3。

读取锁存编码器 1 值-对象字典子索引为 4。

读取锁存编码器 2 值的个数-对象字典子索引为 5。

读取锁存编码器 2 值-对象字典子索引为 6。

读取锁存器状态的示例代码如图 5.82 所示。

```

70      int LFN = 0; //连续锁存值个数
71      int LFV = 0; //连续锁存值
72      ushort indxe = 25362; // Convert.ToInt16(textBox38.Text);
73      short ret = LTDMC.dmc_read_outbit(_CardID, 0);
74      if (ret == 0)
75      {
76          LTDMC.rmc_get_node_od(_CardID, PortNum, nodenum, indxe, 5, 32, ref LFN);
77          LTDMC.rmc_get_node_od(_CardID, PortNum, nodenum, indxe, 6, 32, ref LFV);
78          a++;
79          if (a <= 10)
80          {
81              richTextBox3.Text = richTextBox3.Text + LFN.ToString() + ",";
82              richTextBox2.Text = richTextBox2.Text + LFV.ToString() + ",";
83          }
84      }

```

图 5.82 锁存值个数、锁存值读取

完成上述步骤后，即完成锁存器状态读取配置的代码编写。

在图 5.83 中设置连续锁存参数：

锁存器	
索引	25090 25346
锁存模式	1 单锁锁存器
锁存逻辑	1 0
滤波时间	1 单次锁存值
连锁锁存器	25362 <input type="button" value="c"/> <input checked="" type="radio"/>
连续锁存个数	5, 4, 3, 2, 1 <input type="button" value="锁存器设置"/>
连续锁存值	20000, 60000, 100000, 140000, 180000 0, 0, 0, 0, 0, 0 <input type="button" value="锁存值清零"/>

图 5.83 连续锁存设置及读取

锁存器 2，锁存模式：1（连续锁存），锁存逻辑：下降沿锁存，滤波时间：1us

c. 运行结果：

该示例的高速锁存功能是用同一模块高速比较输出给锁存输入信号是实现的，所以，锁存编码器为 2，锁存个数：5，锁存值为：20000, 60000, 100000, 140000, 180000，如图 5.83。

5.3.4 高速比较功能使用

高速比较功能是在高速计数功能的基础上运行的（高速计数功能使用参考 5.3.2 节）。

比较器参数设置步骤如下：

a. 比较器通道选择：

比较器 0 的对象字典索引为 16#6400。

比较器 1 的对象字典索引为 16#6401。

比较器 2 的对象字典索引为 16#6402。

b. 清除比较器的比较状态：

清除比较状态的对象字典子索引为 1。

清除比较状态的有效值为 1。

设置清除比较状态的示例代码如图 5.84 所示。

```

409     {
410         ushort PortNum = 2;
411         ushort index = Convert.ToInt16(textBox24.Text);
412         ushort nodenum=Convert.ToInt16(textBox5.Text ); //从站ID
413         //清除比较器缓冲区及比较状态    1:清除    子索引:01H
414         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 1, 8, 1);
415     }

```

图 5.84 清除缓冲区以及比较状态

c. 设置比较器的比较模式:

比较模式的对象字典子索引为 2。

比较器有六种工作模式：0-关闭，1-等于，2-小于，3-大于，4-FIFO（队列），5-Liner（线性）

备注：当比较器工作模式为 0-关闭时，输出口为普通输出口。

设置比较模式的示例代码如图 5.85 所示。

```

347     ushort PortNum = 2;
348     ushort nodenum = Convert.ToInt16(textBox5.Text); //从站ID
349     ushort index = Convert.ToInt16(textBox24.Text);
350     int CM= Convert.ToInt16(textBox25.Text);
351     int CM0= 6;
352     // 设置比较器工作模式：0: 关闭,,1: 等于,2: 小于,,3: 大于,4: fifo,5: linear,    子索引:02H
353     LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 2, valuelength, CM);
354     LTDMC.rmc_get_node_od(_CardID, PortNum, nodenum, index, 2, valuelength, ref CM0);

```

图 5.85 比较器比较模式

d. 选择编码器通道:

选择编码器通道的对象字典子索引为 3。

编码器通道有三个：0-编码器 0；1-编码器 1；2-编码器 2。

选择编码器通道的示例代码如图 5.86 所示。

```

355     // 选择编码器通道可选择通道：0、1、2    子索引：03H
356     int CE= Convert.ToInt16(textBox26.Text);
357     int CE0= 6;
358     LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 3, valuelength, CE);
359     LTDMC.rmc_get_node_od(_CardID, PortNum, nodenum, index, 3, valuelength, ref CE0);

```

图 5.86 编码器通道

e. 设置比较器输出逻辑:

比较器输出逻辑的对象字典子索引为 4。

比较器输出逻辑有两种：“0”-条件成立输出低电平，显示状态为 1（TRUE）；

“1”-条件成立输出高电平，显示状态为 0（FALSE）。

设置输出逻辑的示例代码如图 5.87 所示。

```

360 // 设置比较器输出逻辑：0”：条件成立输出低电平，回读输出状态为TRUE “1”：条件成立输出高电平，回
361 int Lvalue= Convert.ToInt16(textBox27.Text);
362 int Lvalue0= 6;
363 LTDMC.nmc_set_node_od(CardID, PortNum, nodenum, index, 4, valuelength, Lvalue);
364 LTDMC.nmc_get_node_od(CardID, PortNum, nodenum, index, 4, valuelength, ref Lvalue0);

```

图 5.87 比较器输出逻辑

f. 设置比较器输出电平时间：

比较器输出电平时间的对象字典子索引为 5；

电平时间的有效范围为：0 至 85899345（无符号的 32 位值）单位：us

设置输出电平时间的示例代码如图 5.88 所示。

```

365 // 设置输出有效电平时间 子索引：05H
366 int Tvalue= Convert.ToInt16(textBox28.Text);
367 int Tvalue0= 6;
368 LTDMC.nmc_set_node_od(CardID, PortNum, nodenum, index, 5, 32, Tvalue);
369 LTDMC.nmc_get_node_od(CardID, PortNum, nodenum, index, 5, 32, ref Tvalue0);

```

图 5.88 比较器输出电平有效时间

注意： 比较器输出电平有效时间只对于比较模式 4—队列比较与比较模式 5—线性比较有效。

g. 添加比较值点：

添加比较点的对象字典子索引为 6。

比较点的有效范围：-2147483648 至 2147483647（有符号的 32 位值）

添加比较点的示例代码如图 5.89 所示。

```

370         if (CMO == 1 || CMO == 2 || CMO == 3)
371         {
372             // 添加比较点(值)           子索引: 06H
373             LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 20000);
374         }
375     }
376
377     if (CMO == 4)
378     {
379         // 添加比较点(值)           子索引: 06H
380         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 20000);
381         // 添加比较点(值)           子索引: 06H
382         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 50000);
383         // 添加比较点(值)           子索引: 06H
384         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 60000);
385         // 添加比较点(值)           子索引: 06H
386         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 80000);
387         // 添加比较点(值)           子索引: 06H
388         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 100000);
389     }
390     if (CMO == 5)
391     {
392         // 添加比较点(值)           子索引: 06H
393         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 6, 32, 20000);
394         // 比较器采用线性比较时, 设置比较点数量           子索引: 07H
395         int CN = Convert.ToInt16(textBox29.Text);
396         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 7, 32, CN);
397
398         // 比较器0采用线性比较, 设置比较点增量值           子索引: 08H
399         int CV = Convert.ToInt16(textBox30.Text);
400         LTDMC.rmc_set_node_od(_CardID, PortNum, nodenum, index, 8, 32, CV);
401     }

```

图 5.89 添加比较点(值)、设置比较点数量、增量值

h. 设置比较点数量:

设置比较点数量的对象字典子索引为 7。

比较点数量的有效范围: 0 至 65535 (无符号的 32 位值)

设置比较点数量的示例代码如图 5.89 所示。

注意: 设置比较点数量只在比较模式 5—线性比较时有效。

i. 设置比较点增量值:

设置增量值的对象字典子索引为 8。

比较点增量值的有效范围: -2147483648~2147483647 (有符号的 32 位值)

设置比较点增量值的示例代码如图 5.89 所示。

注意: 设置比较点增量只在比较模式 5—线性比较时有效。

完成上述步骤后, 即完成对比较器参数的配置的代码编写。

在图 5.90 中设置高速比较的参数。



图 5.90 参数设置

图 5.90 中设置的参数为：比较器 0，比较模式：5（线性比较），比较点数：5，比较值增量：40000，比较输出电平逻辑：0（低电平），输出电平有效时间：1ms，编码器通道：2，编码器的设置与 5.3.2 设置相同。

j. 运行结果：完成以上步骤后，程序运行结果如下（该图为实验测试仪器采集数据后绘制而成）：

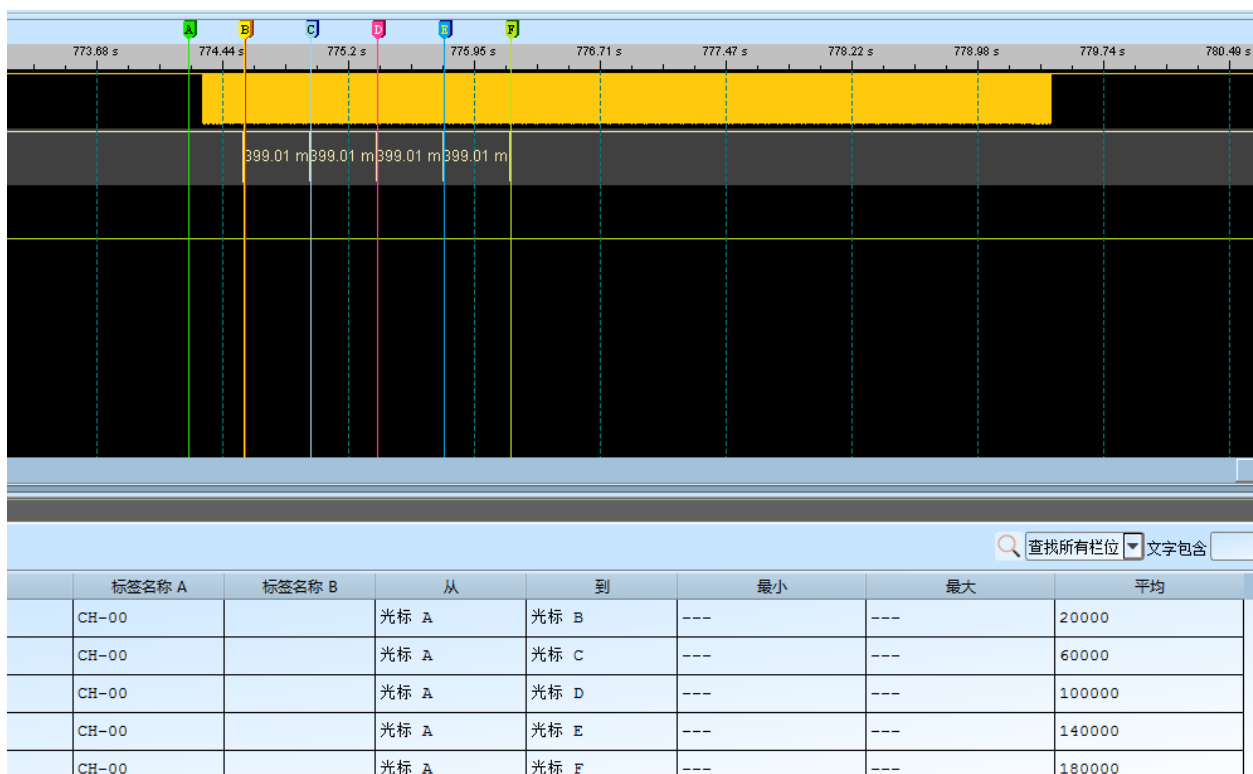


图 5.91 实际计数值，比较值

由图5.91可看出，比较器0，比较模式：5（线性比较），第一个比较点编码器值为：20000，比较增量为：40000，比较点分别为：20000, 60000, 100000, 140000, 180000。



深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

深圳市雷赛控制技术有限公司

地址：深圳市南山区学苑大道 1001 号南山智园 A 3 栋 9 楼

邮编：518052

电话：0755-26415968

传真：0755-26417609

Email: info@szleadtech.com.cn

网址: <http://www.szleadtech.com.cn>