

**NicPro 用户手册**

**版本1.0**

**For Windows NT/2000/XP/7**

**NicSys®1000控制策略组态软件**

中核控制系统工程有限公司

http://www.cncs.bj.cn

**目录**

[1 NicPro概述 1](#_Toc406073610)

[1.1 NicPro特性 1](#_Toc406073611)

[1.2 系统运行环境 2](#_Toc406073612)

[1.3 NicPro安装 2](#_Toc406073613)

[1.3.1 程序安装 2](#_Toc406073614)

[1.3.2 运行NicPro 2](#_Toc406073615)

[1.3.3 运行NicPro仿真程序 2](#_Toc406073616)

[2 基本功能说明 1](#_Toc406073617)

[2.1 程序界面 1](#_Toc406073618)

[2.2 文件（File）菜单 1](#_Toc406073619)

[2.2.1 新建（N） 1](#_Toc406073620)

[2.2.2 打开（O） 1](#_Toc406073621)

[2.2.3 另存为（A） 2](#_Toc406073622)

[2.2.4 打印（Z） 2](#_Toc406073623)

[2.2.5 页面设置（U） 2](#_Toc406073624)

[2.2.6 保存（S） 2](#_Toc406073625)

[2.2.7 退出（X） 3](#_Toc406073626)

[2.3 在线（Online）菜单 3](#_Toc406073627)

[2.3.1 联机（O） 3](#_Toc406073628)

[2.3.2 脱机（F） 4](#_Toc406073629)

[2.3.3 下载（D） 4](#_Toc406073630)

[2.3.4 版本检查 5](#_Toc406073631)

[2.3.5 配置目标系统 5](#_Toc406073632)

[2.3.6 锁定控制器 6](#_Toc406073633)

[2.3.7 启动仿真程序 6](#_Toc406073634)

[2.4 编辑（Edit）菜单 6](#_Toc406073635)

[2.4.1 全部选择 6](#_Toc406073636)

[2.4.2 查找 6](#_Toc406073637)

[2.4.3 拷贝 7](#_Toc406073638)

[2.4.4 粘贴 7](#_Toc406073639)

[2.4.5 删除 7](#_Toc406073640)

[2.4.6 插入程序段 7](#_Toc406073641)

[2.4.7 建立连接 8](#_Toc406073642)

[2.4.8 插入外部块 8](#_Toc406073643)

[2.5 查看（View）菜单 8](#_Toc406073644)

[2.5.1 显示工程浏览器 8](#_Toc406073645)

[2.5.2 显示变量数据库 9](#_Toc406073646)

[2.5.3 显示查看输出窗口 10](#_Toc406073647)

[2.5.4 显示CPU 状态 11](#_Toc406073648)

[2.5.5 显示IO站状态 11](#_Toc406073649)

[3 NicPro编程语言 1](#_Toc406073650)

[3.1 概念 1](#_Toc406073651)

[3.2 功能块语言FBD 2](#_Toc406073652)

[3.2.1 放置功能块 2](#_Toc406073653)

[3.2.2 编辑功能块引脚 2](#_Toc406073654)

[3.3 NicC编程语言 3](#_Toc406073655)

[3.3.1 算法功能块和特殊功能块 3](#_Toc406073656)

[3.3.2 1499模式和1131模式 4](#_Toc406073657)

[3.3.3 语法规范 4](#_Toc406073658)

[3.3.4 基本功能块语法 5](#_Toc406073659)

[3.3.5 函数库 9](#_Toc406073660)

[3.3.6 编译器手册 11](#_Toc406073661)

[4 系统配置 1](#_Toc406073662)

[4.1 目标系统配置 1](#_Toc406073663)

[4.1.1 访问目标系统 1](#_Toc406073664)

[4.1.2 IP配置 1](#_Toc406073665)

[4.2 IO配置 2](#_Toc406073666)

[4.2.1 概述 2](#_Toc406073667)

[4.2.2 设备数据库 2](#_Toc406073668)

[4.2.3 SoftPLC的IO配置 4](#_Toc406073669)

[5 命令行界面组态程序f8config.exe 6](#_Toc406073670)

[5.1 概述 6](#_Toc406073671)

[5.2 SHELL命令 8](#_Toc406073672)

[5.3 工程管理 8](#_Toc406073673)

[5.3.1 open命令 8](#_Toc406073674)

[5.3.2 close命令 8](#_Toc406073675)

[5.3.3 save命令 8](#_Toc406073676)

[5.3.4 online命令 8](#_Toc406073677)

[5.3.5 offline命令 8](#_Toc406073678)

[5.3.6 export命令 9](#_Toc406073679)

[5.3.7 play命令 9](#_Toc406073680)

[5.4 功能块管理 9](#_Toc406073681)

[5.4.1 mount命令 9](#_Toc406073682)

[5.4.2 umount命令 9](#_Toc406073683)

[5.4.3 mv命令 9](#_Toc406073684)

[5.5 引脚管理 10](#_Toc406073685)

[5.5.1 exp命令 10](#_Toc406073686)

[5.5.2 uexp命令 10](#_Toc406073687)

[5.5.3 bind命令 10](#_Toc406073688)

[5.5.4 set命令 10](#_Toc406073689)

[5.5.5 lv命令 11](#_Toc406073690)

[5.5.6 invert命令 11](#_Toc406073691)

[5.6 连接管理 11](#_Toc406073692)

[5.6.1 link命令 11](#_Toc406073693)

[5.6.2 ulink命令 11](#_Toc406073694)

[5.7 版本管理 11](#_Toc406073695)

[5.7.1 ver命令 11](#_Toc406073696)

[5.8 调试支持 11](#_Toc406073697)

[5.8.1 p命令 11](#_Toc406073698)

[5.8.2 pp命令 11](#_Toc406073699)

[5.8.3 ls命令 12](#_Toc406073700)

[5.8.4 disasm命令 12](#_Toc406073701)

[5.8.5 error命令 12](#_Toc406073702)

[6 仿真程序fsimnt 13](#_Toc406073703)

[6.1 仿真器界面介绍 13](#_Toc406073704)

[6.1.1 设备部分 13](#_Toc406073705)

[6.1.2 内存部分 13](#_Toc406073706)

[6.2 仿真过程 14](#_Toc406073707)

[6.2.1 数字量输入仿真 14](#_Toc406073708)

[6.2.2 数字量输出仿真 14](#_Toc406073709)

[6.2.3 仿真过程简介 15](#_Toc406073710)

[7 监控系统的配置 16](#_Toc406073711)

[8 IEC基本功能块库 19](#_Toc406073712)

[8.1 数学库（arithmetic） 19](#_Toc406073713)

[8.1.1 ADD 【加法运算】 19](#_Toc406073714)

[8.1.2 SUB 【减法运算】 19](#_Toc406073715)

[8.1.3 MUL 【乘法运算】 20](#_Toc406073716)

[8.1.4 DIV 【除法运算】 20](#_Toc406073717)

[8.1.5 AND 【“与”运算】 21](#_Toc406073718)

[8.1.6 OR 【“或”运算】 21](#_Toc406073719)

[8.1.7 NOT 【“非”运算】 22](#_Toc406073720)

[8.1.8 XOR 【“异或”运算】 22](#_Toc406073721)

[8.1.9 INV 【求“反码”运算】 23](#_Toc406073722)

[8.1.10 NEG 【求“负”运算】 23](#_Toc406073723)

[8.2 逻辑库（logic） 24](#_Toc406073724)

[8.2.1 ANDn\_BOOL 【逻辑“与”】 24](#_Toc406073725)

[8.2.2 ORn\_BOOL 【逻辑“或”】 24](#_Toc406073726)

[8.2.3 NOT\_BOOL【逻辑“非”】 25](#_Toc406073727)

[8.2.4 XOR\_BOOL【逻辑“异或”】 25](#_Toc406073728)

[8.3 比较类（comparison） 26](#_Toc406073729)

[8.3.1 EQ【等于】 26](#_Toc406073730)

[8.3.2 GE【大于或等于】 26](#_Toc406073731)

[8.3.3 GT【大于】 27](#_Toc406073732)

[8.3.4 LE【小于或等于】 27](#_Toc406073733)

[8.3.5 LT【小于】 28](#_Toc406073734)

[8.3.6 NE【不等于】 28](#_Toc406073735)

[8.4 双稳态类（bistable） 29](#_Toc406073736)

[8.4.1 RS【RS触发器】 29](#_Toc406073737)

[8.4.2 SR【SR触发器】 29](#_Toc406073738)

[8.5 转换类（converter） 30](#_Toc406073739)

[8.5.1 B2\*\*\* 【字节类型转换】 30](#_Toc406073740)

[8.5.2 C2\*\*\* 【字符类型转换】 30](#_Toc406073741)

[8.5.3 DBL2\*\*\* 【双精度类型转换】 31](#_Toc406073742)

[8.5.4 DT2\*\*\* 【日期类型转换】 31](#_Toc406073743)

[8.5.5 DW2\*\*\* 【双字型转换】 32](#_Toc406073744)

[8.5.6 F2\*\*\* 【浮点类型转换】 32](#_Toc406073745)

[8.5.7 I2\*\*\* 【整型转换】 32](#_Toc406073746)

[8.5.8 S2\*\*\* 【短整型转换】 33](#_Toc406073747)

[8.5.9 W2\*\*\* 【字型转换】 33](#_Toc406073748)

[8.5.10 SCALING【量程转换】 34](#_Toc406073749)

[8.6 定时器类（timers） 34](#_Toc406073750)

[8.6.1 TOFF 【延时消去定时器】 34](#_Toc406073751)

[8.6.2 TON 【延时启动定时器】 35](#_Toc406073752)

[8.6.3 TON\_P【带时间暂停功能的延时启动定时】 36](#_Toc406073753)

[8.6.4 TP 【脉冲定时器】 37](#_Toc406073754)

[8.6.5 TIM\_TOL【时间累计器】 38](#_Toc406073755)

[8.7 计数器类（Counter） 39](#_Toc406073756)

[8.7.1 CTUD【递增/递减计数器】 39](#_Toc406073757)

[9 常用特殊功能块 41](#_Toc406073758)

[9.1 IO模件类（FCS） 41](#_Toc406073759)

[9.1.1 DI111 【16通道数字量输入模块】 41](#_Toc406073760)

[9.1.2 DO111 【16通道数字量输出模块】 42](#_Toc406073761)

[9.1.3 AI111 【高精度模拟量输入模块】 43](#_Toc406073762)

[9.1.4 AO111 【模拟量（4~20mA）输出模块】 44](#_Toc406073763)

[9.1.5 DI112 【事件记录顺序（SOE）模块】 45](#_Toc406073764)

[9.1.6 AI121 【8通道热电偶/mV输入模块】 47](#_Toc406073765)

[9.1.7 AI131 【热电阻输入模块】 48](#_Toc406073766)

[9.1.8 PI111 【双通道脉冲频率输入模块】 50](#_Toc406073767)

[9.2 网络通讯类 51](#_Toc406073768)

[9.2.1 WGFLOAT【网络浮点数广播】 51](#_Toc406073769)

[9.2.2 WGINT【网络整型数广播】 51](#_Toc406073770)

[9.2.3 WGBOOL【网络布尔量广播】 52](#_Toc406073771)

[9.2.4 RGFLOAT【网络浮点数读取】 53](#_Toc406073772)

[9.2.5 RGINT【网络整型数读取】 53](#_Toc406073773)

[9.2.6 RGBOOL【网络布尔量读取】 54](#_Toc406073774)

[9.3 常用驱动级（drivers） 55](#_Toc406073775)

[9.3.1 MTR\_LGE【马达驱动1】 55](#_Toc406073776)

[9.3.2 MTR\_LTE【马达驱动2】 56](#_Toc406073777)

[9.3.3 D\_AS\_VAL【电动门驱动1】 58](#_Toc406073778)

[9.3.4 D\_DS\_VAL【电动门驱动2】 60](#_Toc406073779)

[9.3.5 D\_DT\_VAL【电动门驱动3】 62](#_Toc406073780)

[9.4 程序处理类（MCS） 63](#_Toc406073781)

[9.4.1 AVE4W【四输入端加权平均值】 63](#_Toc406073782)

[9.4.2 BALANCE【二路输出平衡模块】 64](#_Toc406073783)

[9.4.3 COMP\_DB【带死区和磁滞的比较器】 65](#_Toc406073784)

[9.4.4 DECIS\_D【三路输入表决器】 66](#_Toc406073785)

[9.4.5 DEV\_HL【偏差报警】 67](#_Toc406073786)

[9.4.6 HLALM【带死区的高低限报警】 68](#_Toc406073787)

[9.4.7 HLALM\_T【带死区及延时的高低限报警】 68](#_Toc406073788)

[9.4.8 K\_SQRT【加权平方根】 69](#_Toc406073789)

[9.4.9 LEADLAG【超前滞后模块】 70](#_Toc406073790)

[9.4.10 LIMIT【限幅输出模块】 73](#_Toc406073791)

[9.4.11 LIMV【一阶速度限制器】 74](#_Toc406073792)

[9.4.12 MULDIV\_W【三输入加权乘除法器】 75](#_Toc406073793)

[9.4.13 PID 【PID调节器】 76](#_Toc406073794)

[9.4.14 SUM\_W【三输入加权加法器】 80](#_Toc406073795)

[9.4.15 Totalizer【积分累加块】 81](#_Toc406073796)

[9.4.16 ABS\_F【浮点数绝对值】 81](#_Toc406073797)

[9.4.17 RMP【斜坡函数】 82](#_Toc406073798)

[9.4.18 SCALING\_X【量程变换】 83](#_Toc406073799)

[9.4.19 TWOSEL【二选一选择器】 83](#_Toc406073800)

[9.4.20 THRSEL【三选一选择器】 84](#_Toc406073801)

[9.4.21 LAG1【一介滞后滤波器】 86](#_Toc406073802)

[9.5 扩展类（Extend） 87](#_Toc406073803)

[9.5.1 FLOW\_FIX【带开方及小信号切除的流量转换功能块】 87](#_Toc406073804)

[9.5.2 POLYNOM【多项式功能块】 88](#_Toc406073805)

[9.5.3 FUN12【多折线功能块】 88](#_Toc406073806)

[9.5.4 OHVFLW\_N【饱和蒸汽流量转换】 89](#_Toc406073807)

[9.6 控制类（Control） 90](#_Toc406073808)

[9.6.1 D\_SEL\_TCK【数字量跟踪】 90](#_Toc406073809)

[9.6.2 G\_SEL\_TCK【模拟量跟踪】 90](#_Toc406073810)

[9.6.3 ES\_MA\_EN【增强型模拟量操作器】 91](#_Toc406073811)

[9.6.4 SS\_MA\_EN【标准模拟量操作器】 92](#_Toc406073812)

[9.6.5 SFT【无扰切换模块】 93](#_Toc406073813)

[9.6.6 MS\_ID【脉冲型模拟输出操作器】 94](#_Toc406073814)

[9.6.7 FIRST【首出控制模块】 94](#_Toc406073815)

[9.6.8 MOV\_B【开关信号传递】 95](#_Toc406073816)

[9.6.9 MOV\_F【模拟信号传递】 96](#_Toc406073817)

[9.6.10 MODE\_SEL【模式选择】 96](#_Toc406073818)

[附录一 NicC语言语法 98](#_Toc406073819)

# NicPro概述

## NicPro特性

***实时操作系统支持***

目前，对于某种类型的任务，使用图形用户界面已成为一种基本需求。鉴于这个原因，NicPro是作为MS Windows 的应用程序而被设计的。具有非常好的移植特性，目前已经开发出在PC/WINDOWS NT、PC/RT-LINUX、PC/VXWORKS、PC/FREEDOS等操作系统下执行的NicSys®控制系统内核。其中RTLINUX和VXWORKS都是著名的实时操作系统，在这些实时操作系统上运行的NicSys®控制器内核能够很好的利用系统的调度特征，使得控制程序的执行更为平滑、可靠。

PC用户只要具有对Windows技术和鼠标操作的基本知识就可以方便的进行学习和操作。此外，所有普通显示器、显卡和打印机都可以在MS Windows下使用。作为用户，您将因此不再受限于特定的硬件配置。

***国际标准IEC 61131／61499***

NicPro按照国际标准规定IEC 61131／61499为有效的系统配置，采用FBD与NicC语言相结合的方式，提供统一的编程环境。在61131标准中，功能块图（Function Block Diagram）方式是一种图形化的程序设计模式，每个功能块图形表示一个算法，这些算法包括了简单的数学函数，也包括很多复杂的控制逻辑。算法的输入、输出参数可以很直观的在图形上表现出来。

在结构化文本方面，NicPro软件采用兼容Ansi-C99标准的NicC编程语言作为算法的载体，利用C语言的表达能力，提升了算法的表达能力。更重要的一点在于，由于采用了兼容性极好的C语言，可以非常方便的将历史上所有用C语言编写的算法移植到任何一个NicPro控制程序中。采用C语言作为控制程序的编程语言，这在同类产品中也是处于领先地位的。

***编程时相对控制器（CP）独立***

开发NicPro的指导原则是，系统配置的大多数步骤，尤其是程序创建，都是在不依赖于要编程的控制器（CP）的情况下设计的。

***方便的GUI操作方式***

NicPro组态软件运行在基于Windows的平台上，充分挖掘了Windows多年发展的图形化用户界面的功能。整个程序被分成与逻辑结构相对应的若干区段。NicPro配置工具允许功能块对象可以轻易地以图形形式选定、放置或移动。在FBD编辑器（功能块图/功能块语言）中，可行性测试在各块放置相互连接时发生。未经认可的连接，例如那些在不同数据类型之间的连接，在配置过程中就已被删除。在第一次成功的运行程序以后，程序可以在图形模式下通过移动连接、块以及文本进行优化，以改善显示。

在功能块图编程方式下，插入、删除、拷贝、粘贴、连接、查看属性等操作的方式，操作热键的布局，菜单项目的取舍，图标符号等等都是精心设计的，设计过程中着重考虑了目标用户的体验，具有很高的实用价值。

***在环仿真***

NicPro还拥有一个32位的仿真器窗口仿真控制器。借助于仿真器，不需要使用实际硬件即可对程序功能进行测试。

***网络支持***

现代控制系统离不开网络的支持，NicPro软件的运行利用了三个网络，一个是组态程序所在的计算机和NicSys®控制器内核之间的网络连接，称为控制网络；一个是NicSys®控制器内核和NicSys®现场IO控制设备之间的网络连接，称为IO网络；另外一个是冗余的主备控制器之间的网络连接，称为热备网络。如果说NicSys®控制系统是一个活的生命体，那么控制、IO和热备网就构成了NicSys®控制系统的循环系统，使其之间的数据交换顺畅自如。

***联机帮助***

NicPro还拥有联机帮助系统，可对常见功能块语法、函数库及项目管理进行注释、搜索和打印。

## 系统运行环境

软件：

* Windows 2000（包括Professional、Server、Advanced Server版）
* Windows XP（Home、Professional版）

硬件：

* CPU：最低Pentium II 233，建议PIII 450或以上
* 内存：最低32M，建议64M或以上
* 硬盘：软件安装要求硬盘上有不少于60M的剩余空间。
* 至少一个网络接口设备（速度在10M以上，推荐100M），为了支持网络冗余，必须有两个或者以上网络接口设备。

## NicPro安装

### 程序安装

NicPro应用程序为绿色软件。NicPro安装程序包可以用光盘或移动存储介质提供给终端用户，通过拷贝方式进行安装。

### 运行NicPro

运行NicPro软件可以通过以下途径：

双击NicPro安装目录中的conductor.exe进入其编程界面。

如下图所示：

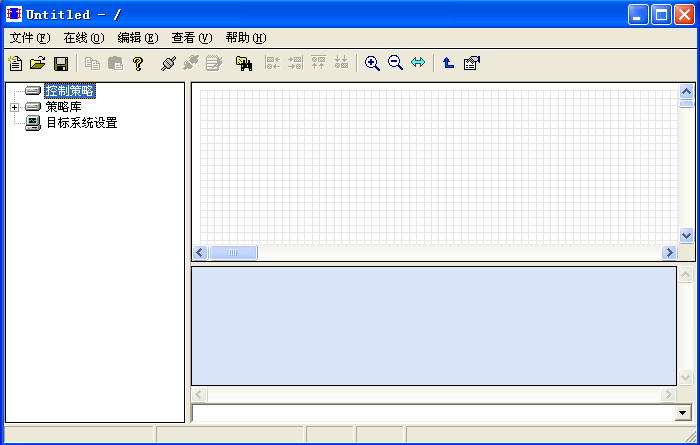


图1.3.1 NicPro编程界面—conductor.exe

### 运行NicPro仿真程序

若需要对控制策略程序进行仿真测试，在进行仿真联机时需先打开（运行）NicPro仿真器程序：双击NicPro安装目录中的fsimnt.exe打开仿真器界面，也可在NicPro编程界面通过菜单“在线→启动仿真程序”打开NicPro仿真器。

如下图所示：

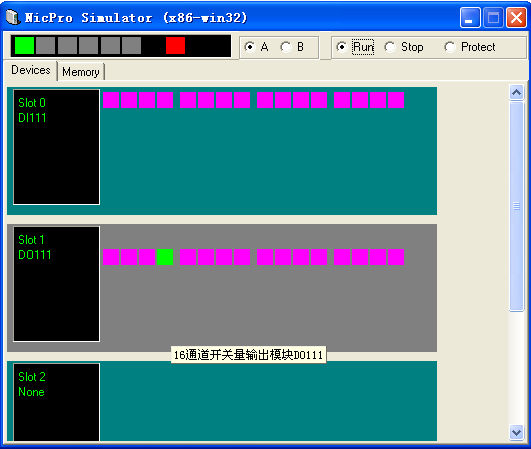


图1.3. 2 NicPro仿真器界面

# 基本功能说明

## 程序界面

运行conductor.exe后，出现如下界面：

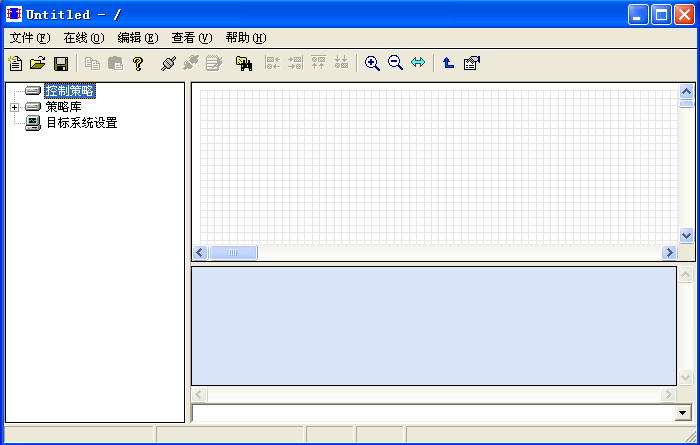


图2.1.1 NicPro编程界面—conductor.exe

## 文件（File）菜单

文件（File）菜单包括新建、打开、另存为、打印、页面设置、保存和退出等，如图2.2.1所示：

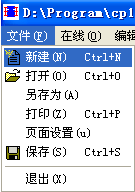


图2.2.1 文件菜单

### 新建（N）

可建立一个新的程序文件，默认文件名和格式为Untitled.ce。可以通过“另存为”或“保存”操作更改程序文件名和文件类型。NicPro软件支持的文件类包括“\*.ce”和“\*.txt”。

工具栏图标 

快捷键 Ctrl+N

### 打开（O）

可直接在NicPro编程界面中打开一个已有的工程组态文件，包括“\*.ce”或“\*.txt”格式。

工具栏图标 

快捷键 Ctrl+O

### 另存为（A）

可通过“另存为”操作修改当前文件名，或通过该操作修改当前已打开文件的属性，如“\*.ce”格式文件改为“\*.txt”格式文件，或将“\*.txt”格式文件改为“\*.ce”格式文件。“另存为”操作执行后编辑器中当前文件名和格式为“另存为”操作的目标文件名及各式。

### 打印（Z）

打印当前编程窗口，打印页面设置由“页面设置”命令完成。

快捷键 Ctrl+P

### 页面设置（U）

设置打印页面属性以及添加系统打印机。如下图2.2.2和图2.2.3所示：

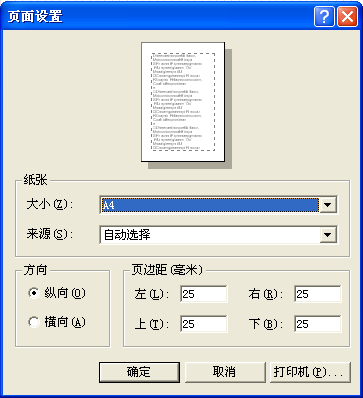


图2.2.2修改页面设置

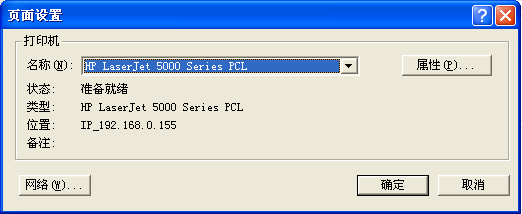


图2.2.3添加或设置系统打印机

### 保存（S）

保存NicPro编程环境当前已打开的程序文件。对于新建程序文件，则已“另存为”命令方式要求用户输入保存目标文件名以及文件格式。默认文件格式“\*.ce”。如图2.2.4所示：

工具栏图标 

快捷键 Ctrl+S

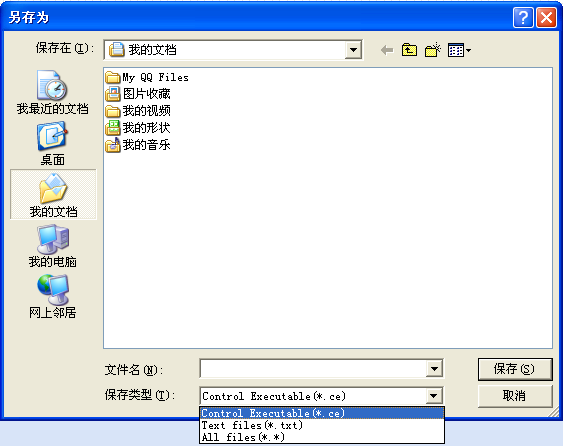


图2.2.4 保存新建文档

### 退出（X）

关闭NicPro编程界面，执行此命令，若当前编程环境中已打开或新建的程序文件未被保存，退出时将提示用户保存当前程序文件。如图2.2.5所示：

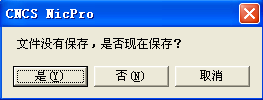


图2.2.5 退出程序提示

## 在线（Online）菜单

在线（Online）菜单包括联机、脱机、下载、版本检查、配置目标系统、启动仿真程序和锁定控制器等。



图2.3.1 在线菜单

### 联机（O）

将NicPro编程环境连接到目标系统（如CP控制器或NicPro仿真器）进行在线编辑或监视。进行该操作时首先弹出“目标系统选择”对话框进行目标系统选择，操作如下：

一 目标系统为NicSys®CP控制器

联机应注意满足如下条件：

1. 与目标系统控制器是否已建立网络连接；
2. 目标系统控制器是否已上电；
3. 目标系统地址和当前使用的计算机通讯端口是否在同一个IP网段上；

达到上述条件，只需在“目标系统地址”框内填入目标系统的IP地址，如图2.3.2所示，点击“联机”即完成与目标系统的联机，同时NicPro编程界面背景则以“灰色”显示，目标系统进入在线监视状态，此时只要当前已打开的程序文件与目标系统的程序版本一致，便可对目标系统进行在线编辑或修改。

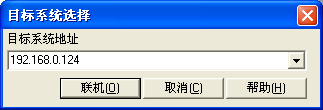


图2.3.2 目标系统选择

二 目标系统为NicPro仿真器

联机应满足如下条件：

1. NicPro仿真器程序已启动；

达到上述条件，只需在“目标系统地址”框内填入本地计算机主机名（Localhost）或计算机名（如LYZH）或（127.0.0.1）即完成与目标系统的联机。

*注意：*

1. *对于目标系统是NicSys®CP控制器，联机时应确认当前打开的程序文件和目标系统程序版本一致（可通过“版本检查”命令来检查），否则即使联机成功后也只能对目标系统的部分项目进行监视而无法进行在线编程或修改。*
2. *在线编程模式下，对在编程环境中的程序任一微小改动，即使从修改后又撤消修改，也将引起目标系统控制器的程序版本变化，此时若目标系统控制器处于热备在线状态则热备在线将离线。*

工具栏图标 

快捷键 F5

### 脱机（F）

该操作断开与控制器或仿真器的连接。

工具栏图标 

快捷键 F9

### 下载（D）

该操作将组态程序下载到NicSys®CP控制器或NicPro仿真器中。下载操作提示对话框如下图所示：

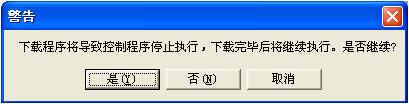


图2.3.3 下载操作提示

注意：

1. **下载操作将导致CP控制器程序终止执行；**
2. **程序下载过程中，系统内所有输出状态将全部被复位（“0”状态）。**

进行下载操作时应检查控制器是否处于锁定状态，同时要检查是否有其他NicPro程序对该控制器进行在线组态，否则此操作无法进行。如图2.3.4所示：

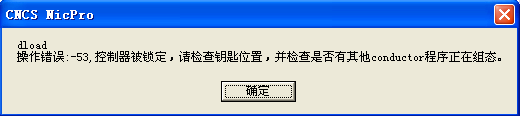


图2.3.4 无效操作报警

“解除控制器锁定”操作参见2.3.6节。

### 版本检查

该操作检查控制器程序版本与NicPro编程窗口程序是否一致。

使用NicPro软件每次组态后都会导致控制程序版本的变化，如果目标系统上执行的程序的版本和本地程序不一致，则在线组态功能不会成功。

要使两者的版本一致，只能使用完全下装。NicSys®系统不支持远程系统和本地程序之间的比较和增量同步功能。

### 配置目标系统

设置NicSys®控制器系统的各个I／O设备类型和参数。在工程浏览器窗口又称“目标系统配置”。Nicsys内核目标系统配置包括IO的设备配置和内存配置两部分，其中IO设备配置是保证控制器内核数据和IO模块正确交换的关键。

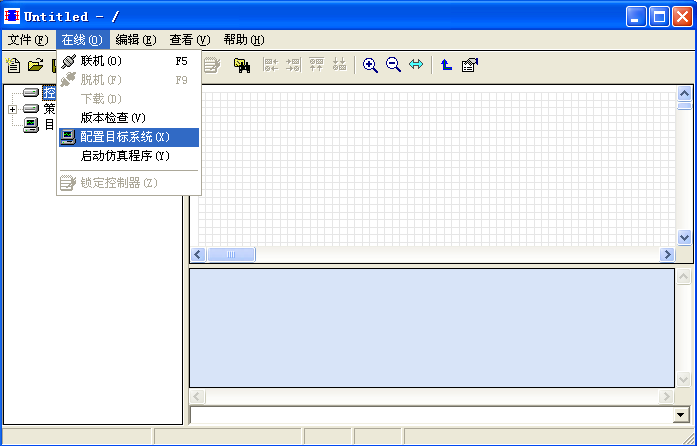


图2.3.5（1）配置目标系统

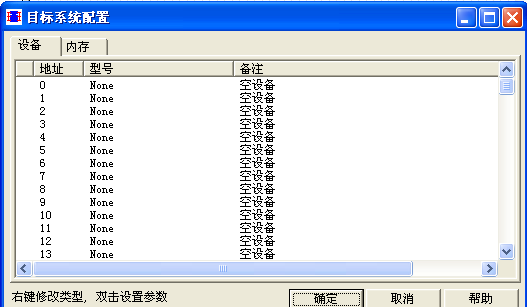


图2.3.5（2） 目标系统配置

NicSys®系统目前支持一个控制器最多带127个IO设备，编号从0到126，127为保留地址。连接在同一个控制器上的IO设备地址不能重复，在使用CNCS提供的I/O设备时，系统配置的设备地址和I/O通讯模块的拨码地址要相对应。

目标系统配置中包含CNCS提供的所有I/O模块类型及参数配置选项。详细说明参见相关模块的技术说明。

### 锁定控制器

该操作将锁定或解锁控制器。“锁定控制器”状态下禁止对控制器进行任何操作，包括编辑、修改或下载。

工具栏图标  控制器锁定状态

 控制器解锁状态

### 启动仿真程序

该操作启动NicPro仿真器程序。

## 编辑（Edit）菜单

编辑（Edit）菜单包括全部选择、查找、拷贝、粘贴、删除、插入程序段、建立连接和插入外部块等命令。



图2.4.1 编辑菜单

### 全部选择

该操作选中当前程序编辑窗口的所有对象。

快捷键 Ctrl+A

### 查找

“查找”功能将列出被查找变量在程序中出现的所有位置。通过该操作可以跟踪程序中变量的使用情况。如下图所示：

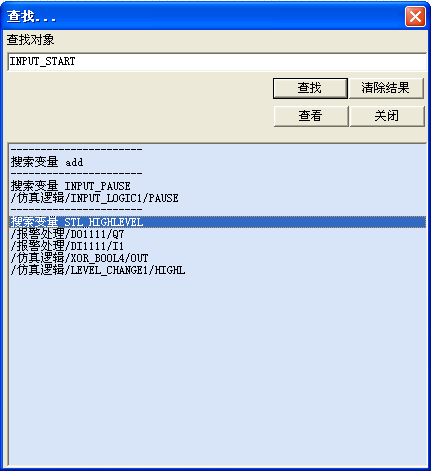


图2.4.2 变量查找

在“查找”对话框内选择搜索结果内容双击或点击“查看”按钮可以跳至该变量被引用的程序段位制。

点击“清除结果”按钮清除搜索结果。

工具栏图标 

快捷键 F3

### 拷贝

该操作复制编程窗口中被选中的对象，变量和功能块间的连接线不能作为对象被复制。

工具栏图标 

快捷键 Ctrl+C

### 粘贴

该操作将“拷贝”操作的结果粘贴到当前编程窗口。在粘贴操作中功能块之间的连接线以及功能块的输出引脚将被屏蔽。

工具栏图标 

快捷键 Ctrl+V

### 删除

该操作删除编程窗口中所选中的对象，不可逆过程，应慎重使用。

### 插入程序段

该操作在程序浏览器中插入一个新的程序段。

如图2.4.3将在程序窗口创建一个新的程序段。右键点击可以修改其属性以及进行一些基本的操作。

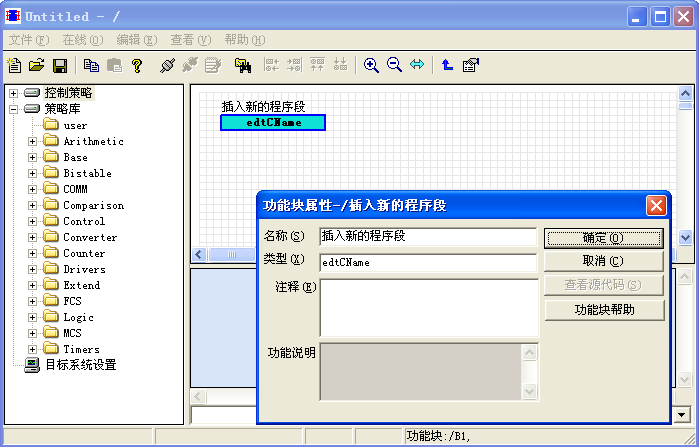


图2.4.3 插入新的程序段

快捷键 Ctrl+K

### 建立连接

该操作建立功能块之间的连接。

选定可以逻辑相连的引脚，点击该命令即可在他们之间建立连接。多条连接之间的接点用一个实心圆表示。

快捷键 Ctrl+L

### 插入外部块

该操作插入一个用户自己编写或定义的功能块。

后缀名为 \*.blk。例如：user文件夹下的自定义功能块。如图2.4.4



图2.4.4 插入外部块

## 查看（View）菜单

查看（View）菜单包括显示工程浏览器、变量数据库、查看输出窗口、CPU 状态和IO站状态5个命令。

### 显示工程浏览器

显示或隐藏工程浏览器。如图所示：

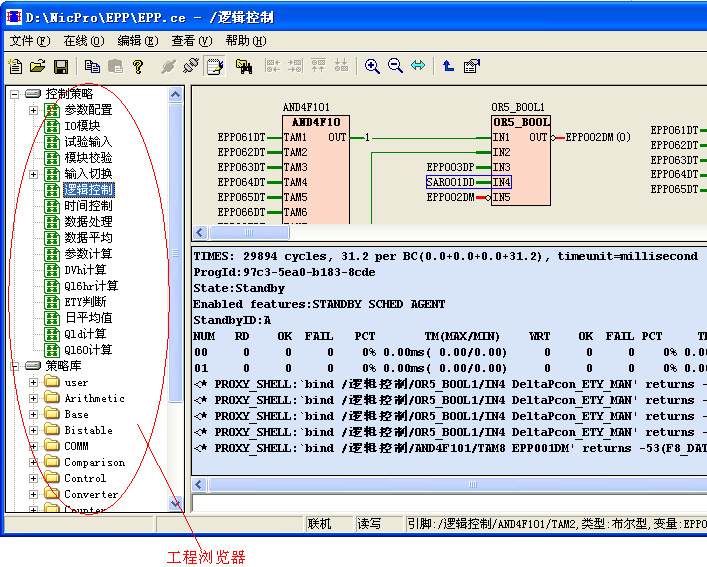


图2.5.1 工程浏览器（左侧）

### 显示变量数据库

显示控制器变量数据库。

其快捷键为“F8”键，由图可见变量数据库包括变量名称、变量类型、引用次数、当前值、初始值和注释组成。数据库的下方还可进行变量搜索，只要输入关键字，点击如图2.5.2，则可以查找出所有带有该关键字的变量。

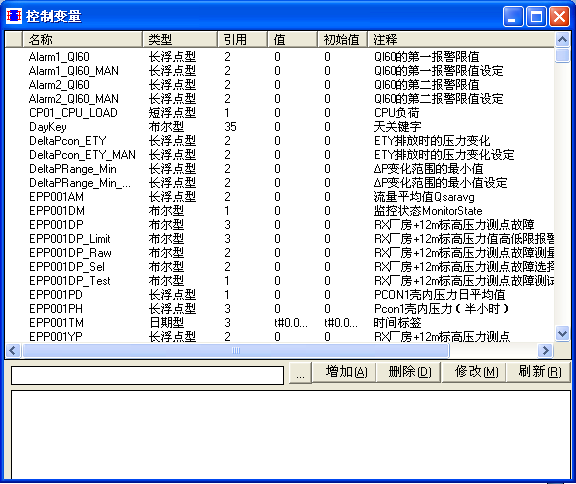


图2.5.2 变量数据库

**变量数据库操作：**

A）查询

在查找输入框中输入需查找内容的可知部分，点击“”按钮，在结果栏将列出所有相关的查询结果：如下图2.5.3(1)所示：



图2.5.3(1) 数据库“查询”操作

B）增加

该操作在变量数据库增加新变量。如图2.5.3(2)：

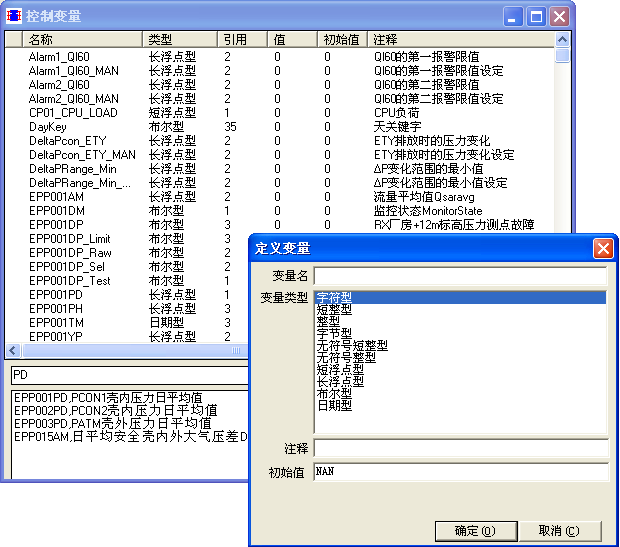


图2.5.3(2) “增加”数据库变量操作

C）删除

删除数据库变量，该操作不可逆。

D）修改

修改数据库变量名称及描述，该操作不可修改数据库变量的数据类型。

E）刷新

刷新数据库变量显示。

### 显示查看输出窗口

显示或隐藏右下方输出窗口及操作命令写入窗口。

### 显示CPU 状态

在输出窗口中显示CPU在线状态。如图2.5.4所示。

### 显示IO站状态

在输出窗口中显示IO 站此刻状态。如图2.5.4所示联机状态下在输出窗口显示的CPU与IO状态的画面。

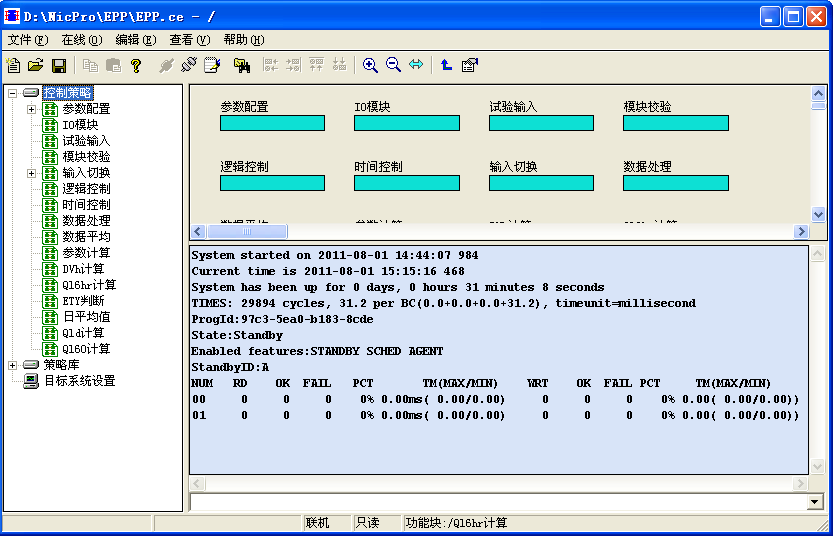


图2.5.4 联机状态下的CPU 及I/O信息

# NicPro编程语言

## 概念

***控制程序***

用户用NicSys®控制器专用的编程语言NicC编制的程序，NicC编译后成为二进制的CE格式文件（Control Executable）。在PC中，NicC文件的后缀为“\***.**f”，功能块文件的后缀为“\***.**blk”。控制程序下载到控制器之后，可以实现特定的工业过程控制功能，控制程序由功能块组成，功能块是控制程序的基本组成单位。

***功能块网络***

控制程序中包含多个功能块，这些功能块的引脚之间可以建立连接表示数据或者事件的流动，这样就组成一个网络，成为功能块网络。

***引脚***

每个功能块和其他功能块交互必须通过引脚，即每个功能块有若干引脚，每个功能块可以通过这些引脚和别的功能块交换数据，参见IEC-61499。NicC支持的数据引脚类型包括：有/无符号字符、短整型、长整型、浮点型、长浮点型和日期/时间型。

***输入、输出引脚***

将事件或者数据引入到功能块的引脚称为输入引脚，将事件或者数据导出功能块的引脚称为输出引脚。

***基本功能块***

由数据引脚、程序组成的有机体。

***连接***

将功能块网络中的功能块的输入引脚和输出引脚相互链接，则构成连接，在控制程序运行期间，数据将在连接制定的路径上流动。

***控制组态***

编制控制程序的过程有两种方式：图形组态和文本组态。文本组态利用NicC进行，图形组态在GUI环境下进行。

***组态语言***

编写NicSys®控制器程序采用的语言，即NicC，其原文件后缀通常为“\***.**f”，NicC采用C语言的一个子集。

***图形组态***

图形组态是类似做电路图的一种组态方式。

## 功能块语言FBD

### 放置功能块

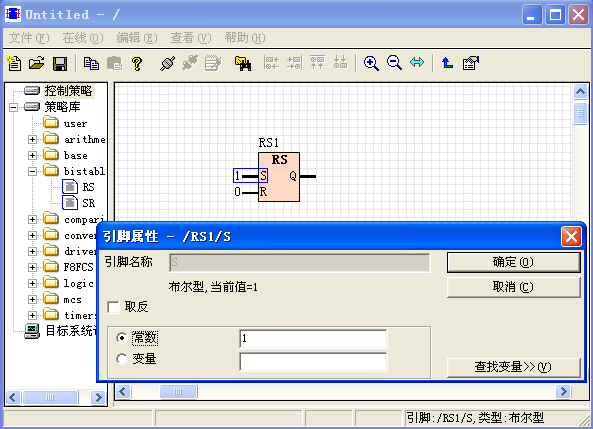
点击工程浏览器的策略库，在相应目录中找到所需功能块，点击左键拖动或双击功能块，即可将该功能块添加至右边的编程窗口。

双击模块，则弹出功能块属性编辑窗口。可以在该窗口下完成对功能块名称的修改和对该功能块添加注释。

若存在该功能块\*.blk文件对应的\*.f文件，点击属性编辑窗口右侧的“查看源代码（S）”，还可以查看该功能块的NicC源程序。

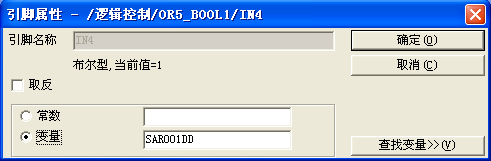
### 编辑功能块引脚

双击功能块的任一引脚，如图3.2.1（1），则弹出引脚属性编辑窗口。通过引脚属性的设定，可以将相应类型的常数值赋给引脚。例如本图中，将该引脚赋了布尔值1。

****

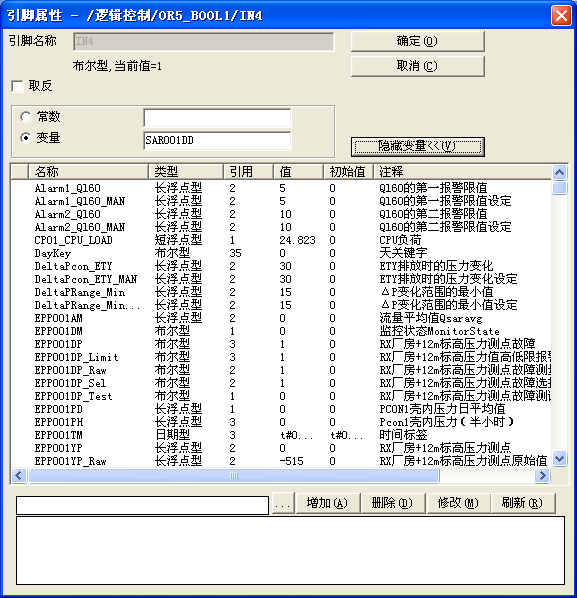
**图　3.2.1（1）功能块引脚属性**

若需要将引脚同某个变量联系起来，则点击变量按钮，在空白框填入变量名。如图3.2.1（2）。



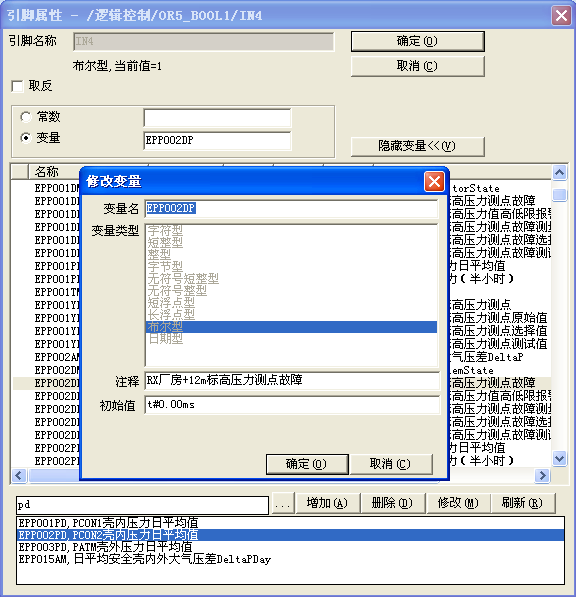
**图　3.2.1（2）引脚变量名填入**

若想对变量进行查找，可以通过引脚属性窗口中，点击右下角的“查找变量>>（V）”。如图3.2.1（3），在下方的空白框进行变量的查找、增加、删除、修改和刷新。此功能与菜单栏中“查看——>变量数据库”的效果相同。



**图　3.2.1（3）查找变量**

查找到需要的变量时，双击数据库中的相应变量（用灰色底色显示），可以看到该变量的各种信息。如：名称、类型、注释和初始值。并可以在该窗口对除了类型以外的信息进行修改。如图3.2.1（4）。



**图　3.2.1（4）变量信息修改**

## NicC编程语言

### 算法功能块和特殊功能块

基本功能块可以分为两种：算法功能块和特殊功能块。算法功能块是用户用来实现控制算法的功能块。特殊功能块是用来实现特殊功能的功能块。

目前支持的特殊功能块包括面向NicSys®控制系统的CPU、通信站、IO模块等。其中，CPU代表一个控制器，通信站（Station）代表一个IO站，而IO模块将IO设备上寄存器的数据导入到控制程序中供用户使用。

算法块和特殊块都可以用NicC编程语言编写。

### 1499模式和1131模式

基本功能块由两种工作模式：1499模式、1131模式。1499模式是NicSys®系统推荐使用的工作模式，控制器内核对这种工作模式的功能块提供优化的支持。在这种模式下，功能块包含事件和数据引脚。1131模式的功能块只包含数据引脚。功能块的工作模式在源代码中制定，并且一旦指定就不可能更改。

1311模式的功能块必须在源代码中指定attribute（autoscan）属性，例如下面是1131模式的延时关闭功能块。

fblock TOFF;

attribute(autoscan);

uuid{c87d-88f6-0d60-fdb4};

var

S : bool;in;

T : date;in;

Q : bool;out;

ET: date;out;

implementation

static bool old\_S;

static date due\_T;

void main()

{

date now;

now = gettime();

if(!S){

if(old\_S){

// negative edge detected

due\_T = now + T;

ET = 0;

}else if(now > due\_T){

Q = 0;

ET = T;

}else{

Q = 1;

ET = T + now - due\_T;

}

}else{

Q = 1;

ET = 0;

}

old\_S = S;

}

### 语法规范

关键字：***fblock，uuid，attrib，comments，var，event，timer，integer，bool，in，out，invoke，char，short，byte，word，dword，float，double，implementation，cblock，uses，links，exports，end* 等*。***

除功能块类型名外，所有关键字和标识符均区分大小写。

NicC采用C语言作为算法的编程语言，所以在功能块程序中的标识符除了不能使用NicC的关键字以外，也不能使用任何C语言的关键字。

以“#”开头的行（“#”前面不能有其他字符）为注释行，编译时会被忽略。

### 基本功能块语法

先看一个例子。

fblock PI111;

attribute(autoscan);

uuid{0d52-e32e-1914-badf};

var

ADDR : word;in; //模块地址

F1 : float;out;

F2 : float;out;

implementation

int main()

{

int \* pVal;

pVal = (int\*)io\_mem(ADDR);

if(!pVal){

return -1;

}

if(\*pVal == 0){

F1 = 0.;

}else{

F1 = 60. / \*pVal \* 10000000.;

}

pVal++;

if(\*pVal == 0){

F2 = 0.;

}else{

F2 = 60. / \*pVal \* 10000000.;

}

return 0;

}

基本功能块程序由如下部分组成：

**头部** + **UUID说明** + **编译注释** + **引脚说明** + **算法说明**等。

***头部格式***

**fblock PI111；**

用来声明功能块的类名（见上面例中的“fblock PI111”），在图形组态环境下，fblock声明的类名会显示在功能块图形内部。如图3.3.1（1）。

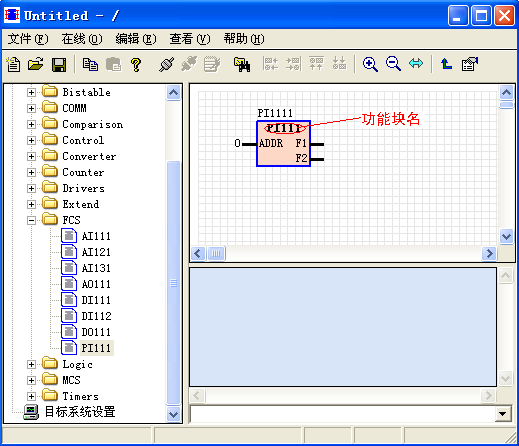


图3.3.1（1）功能块类名

***UUID声明格式***

**uuid{1976 – 0524 – 00e0 – c06d }；**

这里定义的UUID是功能块的类UUID，如果一个控制程序多次引用了同一类功能块，则这些功能块具有相同的类。同类的功能块提供相同的外部接口（即：具有相同数量和类型的引脚，在每个输入事件上执行相同的动作）。

对于IO特殊功能块，***uuid***有特殊含义，实际上，必须保证这里的***uuid***和设备描述信息中的***uuid***一致，才能正确使用IO设备中的数据。

获取***uuid***有两种方法：

1．可以在安装目录下点击f8***uuid.exe***文件，则在剪切板内自动存储了个***uuid***序列，将其粘贴在程序编辑界面即可获得。如图3.3.1（2）；

2．从开始菜单→运行→cmd窗口，进入NicPro所在目录，写入“***f8uuid***”，则可在窗口中看到生成的***uuid***序列。如图3.3.1（3）。

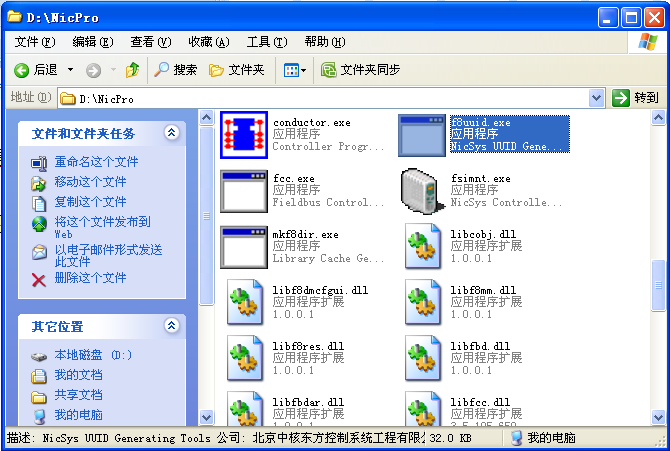


图3.3.1（2）f8uuid

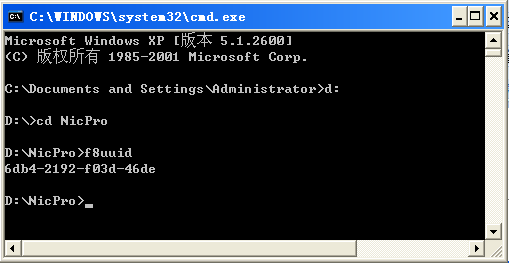


图3.3.1（3）

***编译注释***

编译注释是用户指定编译器生成到目标文件中的信息，和注释不同，注释指那些以“# ”开头的行，会被编译器丢弃，而编译注释则会被编译器记录到资源文件中。编译注释的格式如下：

comments

version=”1.0” ；

author=”Zander” ；

编译注释中可以书写任意的等式，等式左边是编译变量名，右边是变量的值，注意值必须用双引号。

***引脚声明格式***

<名称> ：<类型>< in | out >；

例子：

r ：event；in；

名称可以是任何一个标识符。合法的类型包括：布尔型***bool，***字符型***char，***短整型***short，***整型***integer，***字节型***byte，***字型***word，***双字型***dword，***日期/事件型***date。***其中应注意的是，***integer***和***dword***的长度为8个字节。

***声明算法***

算法由函数组成，函数的写法和C语言的语法一致。例如：

void pidCycle()

{

}

和：

void fft(double \* x, int N)

{

}

等都是合法的函数。

***自定义类型***

可以在算法中使用自定义类型数据，来实现复杂的功能块。自定义类型数据的声明和使用同**ANSI-C**语言。例如下面程序中的***pid\_struct***就是一个自定义类型的数据。

fblock pid;

var

clk : event; in, invoke(on\_clk);

pv : float;in;

sp : float;in;

pout:float;out;

implementation

struct pid\_struct{

float p;

float i;

float d;

float pv;

float sp;

float diff;

float t\_step;

};

static struct pid\_struct pid;

static void do\_pid\_step(const pid\_struct \* t)

{

　　// do some processing

}

static void on\_clk(void)

{

　　do\_pid\_step(&pid);

}

***规定和限制***

NicC在语言规范上，其表达式、流程控制、函数调用等各方面格式都和C语言相同，但限制在于：

（1）全局变量必须用**static**属性声明；

（2）只能调用NicPro函数库，NicPro函数库包含C语言运行函数库的一个子集和若干面向工业过程控制的增强函数；

（3）除**main**函数外，函数必须用**static**声明。

用户可以定义两个特殊的函数：**on\_start**和**on\_stop**，前者在功能块启动的时候被自动调用，后者在功能块停止或者动态删除的时候被自动调用，可以在其中执行一些初始化或者反初始化的工作。

### 函数库

***数学函数库***

***• double gettime();***

　 返回自1600年1月1日至今的时间，以秒为单位。

***• int year(double time);***

　 返回当前年号。

***• int month(double t);***

返回当前月份。

***• int day(double t);***

返回当前日期。

***• void halt(u16 eventId);***

停机函数，停止控制程序执行，如果在功能块执行中发现严重错误（例如被０除），可以调用这个函数中止控制算法的运行。停机事件会记录在控制器内核的日志中，可以使用NicPro的命令行组态工具读取这个日志。

***• int abs(int);***

　求整数的绝对值。

***• double acos(double);***

　反余弦函数。

***• double asin(double);***

　反正弦函数。

***• double atan(double);***

　反正切函数

***• double atan2(double x, double y);***

　返回ｘ／ｙ的反正切值。

***• double cos(double);***

　余弦函数。

***• double cosh(double);***

　双曲余弦函数。

***• double exp(double);***

　ｅ为底的指数函数。

***• double fabs(double);***

　返回浮点数的绝对值。

***• double fmod(double a, double b);***

　浮点数除法的余数，返回ａ除以ｂ的余数。

***• double log(double);***

　以ｅ为底的对数。

***• double log10(double);***

　以１０为底的对数。

***• double pow(double x, double power);***

　幂函数。

***• double sin(double);***

　正弦函数。

***• double sinh(double);***

　双曲正弦函数。

***• double tan(double);***

　正切函数。

***• double tanh(double);***

　双曲正切函数。

***• double sqrt(double);***

　平方根函数。

***过程控制库***

***• int sched(int delay,PIN\*e,u8 schedType);***

　 请求在输出事件引脚e上延时delay毫秒后调度一个事件，如果delay=0，则调度请求立刻进入系统的事件调度队列。schedType是个枚举常数：SCHED\_T\_ONESHOT或者SCHED\_T\_PERIODIC，前者表示一次触发，后者表示周期性触发。此函数返回一个整型值，程序可以在随后调用cancel取消这个调度（如果还没有生效）。

***• f8\_bool cancel(int schedId,PIN\*);***

取消在系统事件调度队列中和输出事件引脚e相关的事件，程序必须保证schedId是调用sched返回的，否则会导致控制器停机。另外pin参数也必须是当初调用sched时传递的同样的值，否则调用会失败（但不会停机）。如果成功的取消了调度，则函数返回TRUE，否则返回FALSE，这仅仅发生在如果指定的事件已经被调度的情况下。

### 编译器手册

编译器程序为fcc.exe，用法为：

fcc.exe –i：<功能块源文件名>

例如：

fcc –i：cod.f

编译后将生成二进制的功能块文件（\*.blk），功能块文件由两部分组成：CE流和资源流。CE（Control Executable）流是CE格式的NicPro可执行程序，可以下装到控制器中运行；资源流包括了符号信息、调试信息、编译注释信息等等，组态时会用到。

例子：

d:\proj\target\bin>fcc -i:F8Cib\f8\ads1711.f

F8Cib\f8\ads1711.f

Compiling ...

Ok, 1888 RAM, 1636 ROM(1200 text, 28 data, 408 pins), 4 bss, 1309 res, total 4833.

# 系统配置

## 目标系统配置

### 访问目标系统

用NicPro软件对目标系统进行组态是访问NicSys®系统最主要的方式，除此之外，还可以使用ftp和telnet方式，某些版本的控制器内核也支持web界面访问。升级控制器固件、控制器内核、修改内核参数等都只能用ftp方式进行，没有图形界面。

### IP配置

NicSys®系统的网络配置是手工进行的，没有图形组态界面。需要设定的内容包括，两个和上位机通讯的以太网的100Mbps网卡的IP地址、掩码，两个和IO网通讯的10Mbps网卡的IP地址。这通过修改控制器上的初始化文件startup.sh来实现。

startup.sh在控制器上电的时候执行一次，因此修改完毕之后必须重新启动控制器（重新上电）才会生效。

具体操作如下：

1. 从控制器上下载“/c/startup.sh”
2. 在PC机上编辑startup.sh
3. 把startup.sh上载到控制器的“/c”目录

其中下载必须用命令行版本的ftp客户端，cuteftp、ie自带的ftp客户端均不能正常工作。下载过程如下，采用匿名ftp，密码为空。

D：\tmp>ftp 192.168.0.209

Connected to 192.168.0.209.

220 VxWorks (VxWorks5.5) FTP server ready

User (192.168.0.209：(none))： ftp

331 Password required

Password：

230 User logged in

ftp> binary

200 Type set to I, binary mode

ftp> get /c/startup.sh

200 Port set okay

150 Opening BINARY mode data connection

226 Transfer complete

ftp： 收到300字节，用时0.01 seconds 20.00 Kbytes/sec.

ftp> bye

221 Bye...see you later

上载过程如下：

D：\tmp>ftp 192.168.0.209

Connected to 192.168.0.209.

220 VxWorks (VxWorks5.5) FTP server ready

User (192.168.0.209：(none))： ftp

331 Password required

Password：

230 User logged in

ftp> binary

200 Type set to I, binary mode

ftp> cd /c

200 Port set okay

ftp> put startup.sh

200 Port set okay

150 Opening BINARY mode data connection

226 Transfer complete

ftp：发送300字节，用时 0.01 seconds 20.00Kbytes/sec.

ftp> bye

221 Bye...see you later

标注startup.sh文件的内容如下，配置的时候只能修改ip地址部分，其他不能修改，否则会造成控制器不能正常工作。下面的例子中，配置和上位机接口的两个网卡地址分别为192.168.0.209和192.168.1.209。

shellInitHook

initInterface "rtl 0 192.168.0.209 f8 0xffffff00"

initInterface "rtl 1 192.168.1.209 f8 0xffffff00"

initInterface "ene 0 192.168.10.209 f8 0xffffff00"

initInterface "ene 1 192.168.11.209 f8 0xffffff00"

localexec "/d/f8kernel.out","ex\_main\_x"

## IO配置

### 概述

IO是NicSys®控制系统和物理世界进行信息交换的地方，IO系统支持多种IO设备，包括面向FCS的多种分布式IO设备，以及面向SoftPLC的IO板卡等。

### 设备数据库

NicPro维护一个设备数据库，组态的时候用户从这个数据库中选择IO模块的类型。设备数据库通过修改config目录下的NicSys-devices.inf完成，图4.2.1显示了这个文件的位置和格式。这个文件没有图形界面的维护工具，只能手工修改，修改可以用任何文本编辑器进行。



**图　4.2.1 设备数据库文件**

NicSys-devices.inf描述一个IO设备的参数特征，例如IO通道个数，如果在softplc环境下使用，还制定了驱动程序文件名称等等，举例说明。

例1：在设备数据库中加入NicSys®专用IO模块的信息

[DI111]

description=16通道开关量输入模块DI111

n\_di=16

id=4d44-3231-0032-0000

[DI112]

description=事件顺序记录(SOE)模块DI112

n\_ai=20

ai\_width=1

n\_ao=1

ao\_width=1

id=4d44-3231-0033-0000

[DO111]

description=16通道开关量输出模块DO111

n\_do=16

id=4d44-3232-0032-0000

[AI111]

description=高精度模拟量输入模块AI111

n\_ai=8

ai\_width=2

n\_ao=8

ao\_width=1

id=4d44-3233-0031-0000

例2：在设备数据库中加入ADS-1711Softplc设备的描述信息

[ADS1711]

；设备名称

description=研华1711-PCI多功能卡

；模拟量（AI）输入通道个数

n\_ai=16

；AI输入通道数值字节宽度

ai\_width=2

；AO通道个数

n\_ao=2

；AO通道数值字节宽度

ao\_width=2

；DI通道个数

n\_di=16

；DO通道数值字节宽度

n\_di =16

；设备标示符

id=4c57-b05f-eb22-1711

；驱动程序名称

PlcDriver=libads1711\_device.dll

需要说明的是，设备描述信息的n\_ai、ai\_width、n\_di、n\_di等参数必须和实际设备一致，也必须和设备特殊功能块一致，id参数必须和设备的特殊功能块源代码中指定的uuid属性一致，这样才能保证访问设备上的IO数据。设备的id必须和实际设备中编码的id一致，否则不能读写该IO设备。

### SoftPLC的IO配置

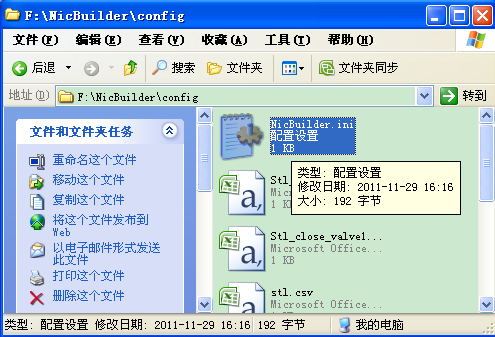
SoftPLC的IO子系统原理如图4.2.2（1）所示。由IO专用功能块、控制器内核和控制器IO驱动程序三层逻辑结构组成。编辑好的程序通过控制器内核的处理，达到驱动IO设备的目的。实现了软硬件的统一。

IO专用功能块

Nicsys控制器内核

Nicsys控制器IO驱动程序

**图　4.2.2（1）SoftPLC的IO子系统原理图**



**图　4.2.2（2）设备的驱动程序配置文件**

为了在SoftPLC中使用一个IO设备，首先需要将该设备的驱动程序加入控制器内核，通过修改“NicBuilder．ini”完成，图4.2.2（2）显示了其位置和格式。目前还没有提供图形界面的工具，只能手工修改这个文件。NicBuilder．ini通知控制器内核在启动的时候加载哪些设备的驱动程序。这两个文件修改完毕后，必须重新启动控制器内核才能生效。

下面的程序段为在NicBuilder．ini中制定Nicsys控制器内核在启动的时候加载ADS1711设备驱动：

[softplc]

devices=ads1711

ads1711.type=actech.ads1711

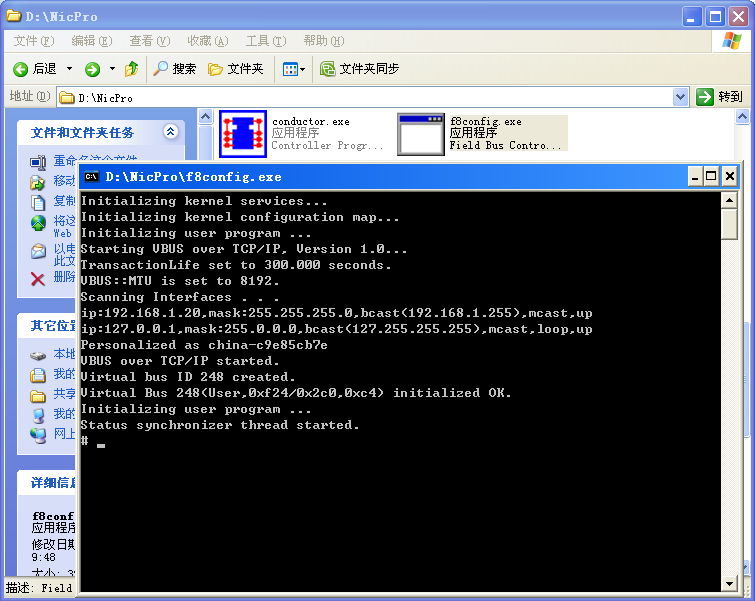
ads1711.config=

每种IO设备一旦配置成功，就可以在控制程序中使用和该IO设备对应的块输入输出数据。这通过一个和设备相关的特殊功能块来实现，和ADS1711对应的特殊功能块参见附录2。ADS1711功能块上有个ADDR引脚，这个引脚的值必须和在IO配置中指定的设备地址一致。

# 命令行界面组态程序f8config.exe

## 概述

位于NicPro软件安装目录下的f8config.exe是一个字符界面的控制逻辑组态程序，如图5.1.1。



**图　5.1.1 字符界面的组态程序**

f8config通过shell的方式组态，控制器的shell是一个用来对控制器或者控制程序进行在线、离线组态的文本接口，shell的工作原理是接受一个字符串，并执行该字符串指定的动作。f8config.exe程序利用了这个接口实现了一个命令行的组态程序。

在很多时候命令行界面组态是一种更有效的组态方式，尤其是其支持批处理功能，可以快速的建立变量数据库、快速复制类似逻辑等等。

用“help”命令可以显示支持的所有命令，如下：

d:\proj\target\bin>f8config

# ?

q ： exit simulator.

help： show available commands.

? ： show available commands.

open ： open project.

create： create project

close ： open project.

save ： save project

online ： go online

offline ： go offline

uload ： upload configuration

dload ： download configuration

fix ： fix project

ver ： show local and target version

mount ： insert a block into an task.

umount ： delete a block from an task.

cp ： copy a block to a new position

mv ： move or rename a block

lib ： save block in library

reset ： reset controller.

start ： start controller

stop ： stop controller

break ： pause controller

continue ： continue controller

suspend ： suspend controller

resume ： resume controller

link ： make connection

ulink ： delete connection

exp ： export a pin

uexp ： unexport a pin

bind ： set pin binding

ubind ： remove pin memory binding.

invert ： invert boolean pin

p ： print pin value

pp ： print pin value

ls ： list subblocks.

u ： unassembly block

mmap ： show memory map

d ： dump exchange memory

vres ： dump resource

error ： show error message for error number.

whouses ： find out who uses the variable

whoowns ： find out who owns the address

log ： show or clear the target log buffer

set ： set variable

uset ： unset variable

lv ： list variables

modify ： modify variable

device ： view/modify device configuration

memsize ： view/modify memory section sizes

memusage ：view memory section usage

play ： play script commands

place ： place block or links

export ： export project in text format

test ： test

<\* PROXY\_SHELL：‘?’ returns 0(F8\_SUCCESS) \*>

#

## SHELL命令

Shell命令分为工程管理、功能块管理、引脚管理、连接管理、版本管理、调试支持等几个部分。下面对他们一一进行介绍。

## 工程管理

### open命令

含义：打开工程

语法：open<工程文件名>

说明：文件名可以带后缀（\*．ce），也可以省略。打开一个工程会自动关闭当前打开的工程，工程打开后自动进入离线状态。

### close命令

含义：关闭工程

语法：close

说明：关闭工程后，系统进入离线状态，并且控制程序数据库和资源数据库都为空。

### save命令

含义：保存工程

语法：save[<工程文件名>]

说明：如果是一个新建的工程，则必须输入工程文件名，否则可以省略。如果指定一个新的文件名，则将保存作为一个新的工程文件，也就是相当于“save as”。

### online命令

含义：进入联机状态

语法：online[<目标名>]

说明：这个命令将使工程和一个目标系统建立连接关系，进入联机状态后，所有组态操作将同时在操作系统和本地工程上进行。第二次执行这个命令时可以省略目标名参数，shell会使用上次的目标名。目标名的格式和目标系统类型以及目标系统使用的组态协议有关，目前只支持一种协议即TCP/IP，目标地址就是目标系统的IP地址。

### offline命令

含义：进入脱机状态

语法：offline

说明：这个命令断开和目标系统的连接关系，脱机状态中，所有组态命令都只在本地工程上进行。

### export命令

export命令将选定的程序段导出为文本形式，这些文本实际上就是为了重建这些程序段所必须执行的命令，可以随后用play命令回访。

export命令的语法如下：

# export /?

Options：

-of：<string> output file name

-p：<string> root block path

-config include configuration

-source include source

-? | -help show this screen

其中“-source”表示把组态内容导出，“-config”表示将配置信息导出。

### play命令

将一个文件中的内容作为组态命令逐行执行。典型用法是可以执行由export命令导出声称的文本文件。

## 功能块管理

### mount命令

含义：安装功能块

语法：mount path=<挂接点>

name=<实例名> type=<类型名>[rw]

说明：控制程序由功能块组成，mount命令就是向控制工程中添加功能块的指令。挂接点是一个路径表达式。类似Windows系统中文件目录一样的字符串，例如：/task\_1/blinker/light，路径表达式由永“/”分割的功能块名组成，每个“/”分割形成的部分表示功能块的不同层次。注意挂接点指向的功能块必须是一个非只读的组合功能块。

“rw”参数用来指定是否以只读方式插入功能块，基本功能块插入后都是只读的，组合功能块缺省也是只读的。组合功能块有两种插入方式：只读和非只读方式。用只读方式插入的功能块不能修改其结构，用读写方式插入的功能块可以随后修改其结构。只读方式可以进行版本跟踪，即如果只读块的模版修改了，则可以通过分析工程更新控制程序中的同类型的功能块。读写功能块插入后就和其模版断开联系了，即如果模版修改，则无法更新插入的功能块。

新插入的功能块名称不能和挂接点下其他功能块重名。

### umount命令

含义：拆卸功能块

语法：umount <功能块路径名>

说明：从控制程序中删除一个功能块。

### mv命令

含义：移动功能块

语法：mv <路径名1> <路径名2>

说明：路径名2必须指向一个现存的组合功能块，将路径名1指向的功能块挂接到路径名2指定的位置，如果在新的位置出现重名，则命令失败。

注意：和此功能块相关的连接关系都会被清除。

## 引脚管理

### exp命令

含义：导出一个引脚

语法：exp <名称> <引脚路径>

说明：将引脚路径指向的引脚导出为父功能块的一个引脚。

### uexp命令

含义：删除导出引脚

语法：uexp <引脚路径>

说明：将引脚上原有的连接关系都会被删除。

### bind命令

含义：设置引脚的绑定状态

语法：bind <引脚路径><绑定表达式>

说明：此命令只对DI和DO类型的引脚有效，对EI和EO无效。

每个引脚在任何一个时刻都处于两种状态之一：连接状态和绑定状态。连接状态即引脚上连接了其他引脚，如果不是处于连接状态，当引脚处于绑定状态时，可以：

1. 为引脚指定一个立即数，只对DI有效；
2. 为引脚指定一个内存变量。参见set指令。

视引脚的数据类型，立即数有不同的格式，如下：

***。布尔型***

true，false；

***。整型（包括char / short / integer / byte / word / dword）***

一个整型表达式，例如13，55，-7等；

***。浮点型（包括float和double）***

一个浮点表达式，例如-3.5，1.2e-13等；

***。日期型***

可以用ms作单位，也可以用s作单位，其意义分别为毫秒和秒，例如“t#1000ms”或者“t#5s”等；

### set命令

set和user指令分别用来创建和删除内存变量，set指令的格式为：

set type=<> name=<> [comment=<>] [scope=<>]

例如：

set type=integer name=tmrOpenTime scope=/#1阳床/再生逻辑 comment="再生时间"

### lv命令

列出所有内存变量。

### invert命令

含义：设置或者取消布尔类型引脚的逻辑非标志

语法：invert <引脚路径>

说明：对于bool类型的引脚，可以在数据锁存进入功能块或者锁存输出功能块时附加一个逻辑非操作，invert就是用来设置或者取消这种功能。

## 连接管理

### link命令

含义：在两个引脚之间创建连接

语法：link <引脚路径1> <引脚路径2>

说明：引脚类型必须匹配，否则会失败；源引脚和目的引脚的顺序无关紧要，但如果目的引脚上已经有连接，则命令失败，即一个目的引脚同时只能接受一个来源的数据，引脚上建立了连接会自动进入连接状态，如果曾经指定过绑定，则绑定信息自动丢失。

### ulink命令

含义：删除连接

语法：ulink <目标引脚路径名>

说明：删除连接到指定引脚上的连接，引脚必须是个输入类型的引脚（即DI或者EI），因为输出类型的引脚上可能会有多个连接。

## 版本管理

### ver命令

含义：比较目标系统和本地工程的版本

语法：ver

说明：只能在连接状态下执行，将打印出目标系统和本地系统的版本值（是一个8字节的整数，用xxxx – xxxx – xxxx – xxxx的格式显示）。

## 调试支持

### p命令

含义：打印功能块或者引脚的值

语法：p <功能块路径名>—<引脚路径名>[…]

说明：p命令会显示引脚的类型和值，如果指定的是一个功能块，则会显示其所有引脚的值和类型，另外会显示该功能块的所有实例数据的值和类型。

### pp命令

含义：连续显示引脚的值

语法：pp <引脚路径名>[…]

说明：以500ms为间隔，持续20s，打印引脚上的值，在联机状态下可以用来观察引脚值上的变化情况。

### ls命令

含义：列出某一个组合功能块下面所有的子功能块

语法：ls <组合功能块路径名>

说明：ls命令会显示所有子功能块的uuid、class\_uuid、class\_name和instance\_name。

### disasm命令

含义：反汇编功能块

语法：disasm <功能块路径名>

说明：这个命令可以用来查看功能块的源代码。注意：当前版本不支持显示基本功能块的实现部分（即implementation部分）

### error命令

含义：查看错误代码的含义

语法：error <错误代码>[…]

说明：显示错误代码代表的意义，例如“err-19”会显示“f8\_vertion\_mismatch”

# 仿真程序fsimnt

## 仿真器界面介绍

如图6.1.1为仿真器NicPro Simulator（x86-win32）的界面。

上面一条为状态栏：左边为对应板卡状态的七个显示灯，中间A、B为主站和备战之间的切换，右边Run、Stop和Protect显示控制器的三种运行状态——运行、停止和保护状态。

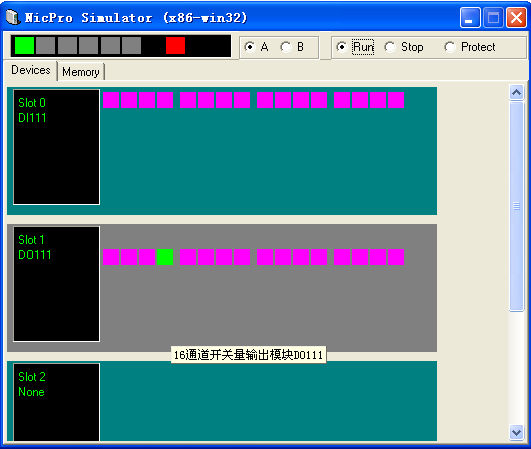


图　6.1.1 NicPro Simulator界面

界面的下方分为两部分，设备（Devices）和内存（Memory）。

### 设备部分

左边的黑色长方块表示各个板卡的槽位，用slot x表示。

卡件槽位同组态过程中“目标系统配置”中的地址一致。槽位下方显示了该板卡的类型，如DI111、DO111、AI111等，如果槽位被配置为“空设备”则显示“none”

如需要显示某个卡件，双击黑框内部，会弹出一个卡件地址窗口，在其中填入所查看的卡件地址，如“1”槽位的卡件，则该槽位即显示为“slot 1”，如图6.1.2。

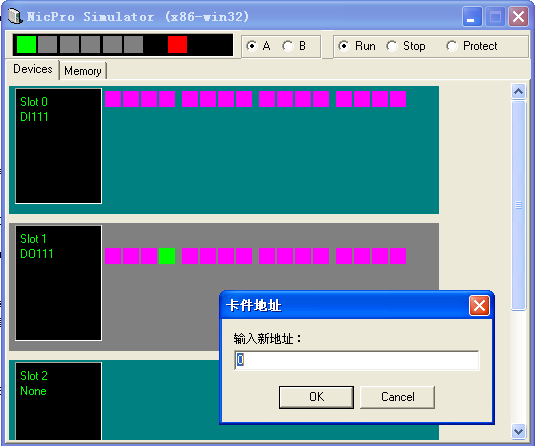


图　6.1.2 输入卡件地址查看卡件

### 内存部分

如图6.1.3，点击“Memory”键即切换为内存地址，可以看到它随着时间在不停的刷新。

每行显示十六个字节数，若需要访问某一地址数据，可以在“ADDRESS”框中输入地址，点击“goto”，则可以转到该地址所在的段位进行查看。

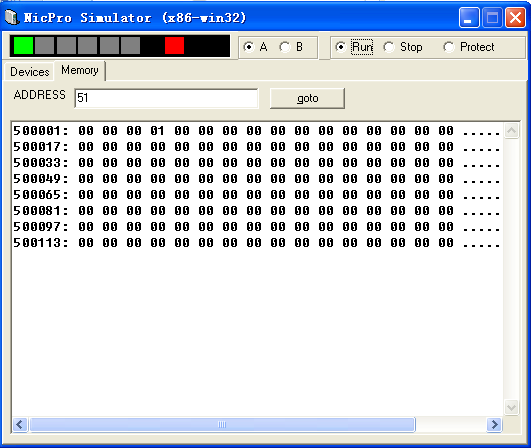


图　6.1.3 内存地址

## 仿真过程

打开仿真器窗口，联机并装载程序，即可在线对所编程序进行仿真。

下面介绍几种常用的仿真方法：

### 数字量输入仿真

假设slot 0 中的卡件为DI111，即16位开关量输入模块，则下装后仿真界面显示为如图6.2.1。

模拟点击仿真器中的小方块，使其在红色与绿色之间转化（红色表示0，绿色表示1），可以看到改变输入值时，模块上对应的引脚也相应地发生变化。

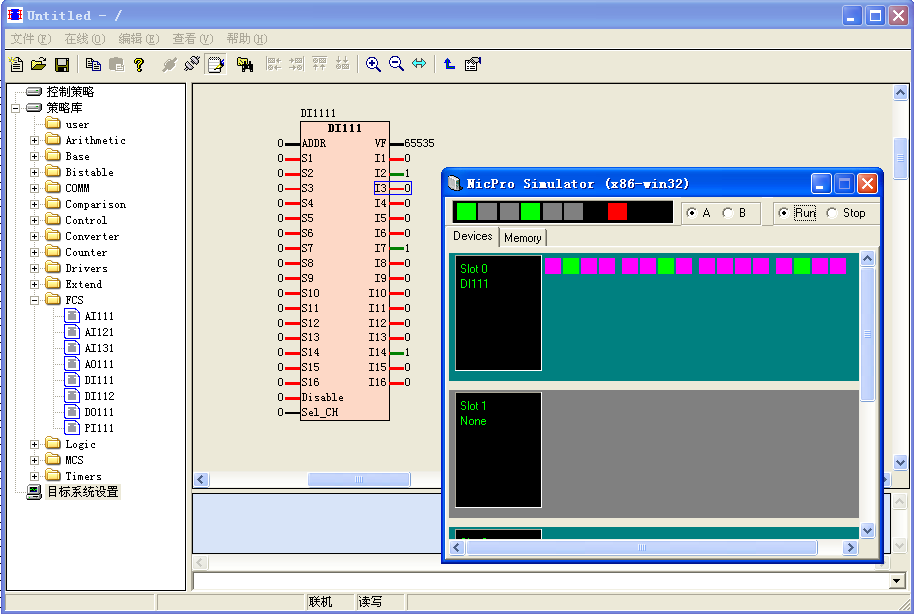


图　6.2.1 数字量输入仿真

### 数字量输出仿真

假设slot 18 中的卡件为DO111，即16位开关量输出模块，则下装后仿真界面显示为如图6.2.2。

当我们在模块上对某个引脚取反，则对应的仿真器小方块颜色发生变化，说明输出值也发生了相应了改变。

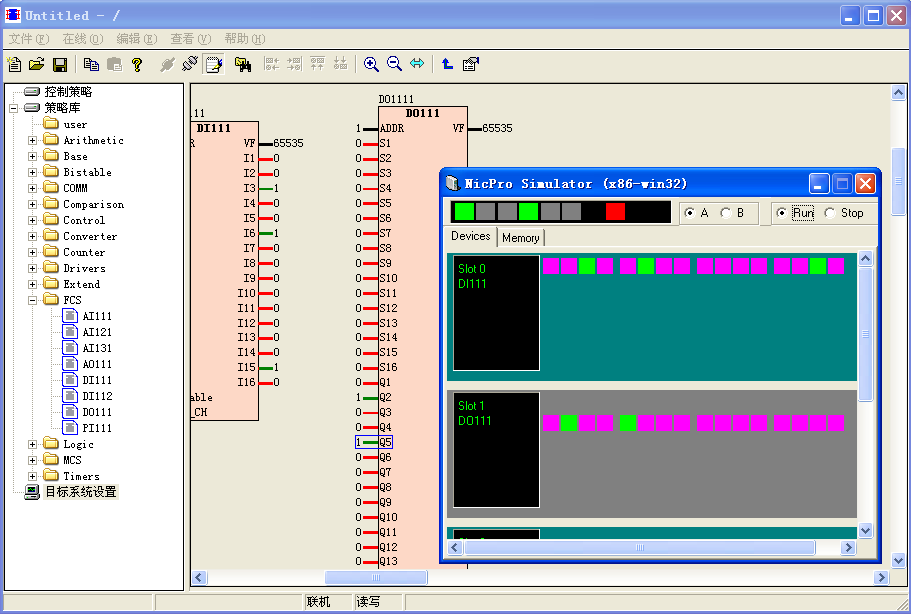


图　6.2.2 数字量输出仿真

### 仿真过程简介

在实际操作中，我们可以将两台电脑相连，其中一台运行画面，另一台运行仿真器，这样就可以方便快捷的进行数据处理，达到对整个程序在线仿真的效果，帮助工程人员更好的差错和编程。

# 监控系统的配置

在NicBuilder系统中添加一个接口设备，就可以在NicBuilder系统中访问的实时数据。程序中定义的变量会自动进入NicBuilder的实时数据库，不需要单独组态。如图7.1点击NicBuilder目录下的NicBuilder.exe工业监控组态程序。



图　7.1 NicBuilder工业监控组态

如图7.2，弹出的NicBuilder工业监控组态窗口。点击“刷新”，可见左边的导航栏中出现“+”，一直展开可以看到“IO设备”项，点击“添加新设备”，出现图7.3。

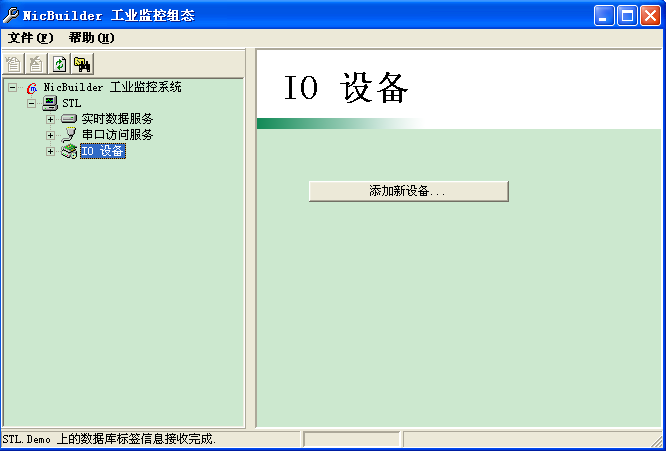


图　7.2 NicBuilder实时数据库组态

选择设备类型，点击“生产厂商”栏中“北京中核东方控制系统工程有限公司”，会在设备型号栏出现所有中核控制自主研发的设备名称。

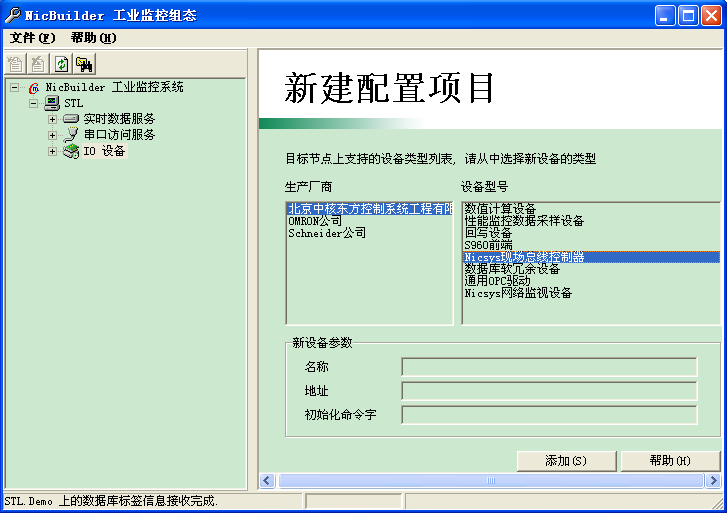
****

图　7.3 选择设备类型

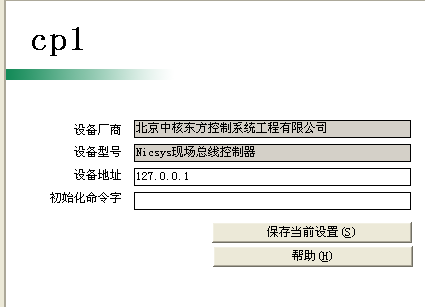


图　7.4 设备参数配置

下一步对Nicsys设备接口的参数进行配置，我们可以填写和修改“设备地址”和“初始化命令字”，如图7.4，修改后点击“保存当前设备（S）”。

启动NicBuilder服务程序后，Nicsys变量自动写入NicBuilder实时数据库中，如图7.5。



图7.5 NicBuilder实时数据库中的变量

# IEC基本功能块库

## 数学库（arithmetic）

### ADD 【加法运算】

该功能块实现加法运算，将任意数字类型组（ANY\_NUM）数据的输入值相加，并将赋值结果输出。输入值得数据类和输出值得数据类型必须是相同的。可以使用一些特殊函数处理不同类型的数据。

**功能块图**



图8.1.1 加法运算功能块

**公式**

ANY\_NUM：OUT = IN1 + IN2

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 加数1

IN2 ANY\_NUM 加数2

OUT ANY\_NUM 和

**运行错误**

运行时如果出现下列情况，发出错误信息：

* 输出值的范围越限

### SUB 【减法运算】

该功能块实现减法运算。被减数IN1减去减数IN2，结果被赋值输出。

ANY\_NUM类型数据能够被处理。输入值和输出值的数据类型必须是相同的。可以使用一些特殊函数处理不同类型的数据。

**功能块图**



图8.1.2 减法运算功能块

**公式**

OUT = IN1 – IN2

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM，TIME 被减数

IN2 ANY\_NUM，TIME 减数

OUT ANY\_NUM，TIME 差

**运行错误**

运行时如果出现下列情况，发出错误信息：

* 输出值的范围越限

### MUL 【乘法运算】

该功能块实现乘法运算。输入值相乘的结果被赋值输出。

输入值得数据类和输出值得数据类型必须是相同的。可以使用一些特殊函数处理不同类型的数据。

**功能块图**



图8.1.3乘法运算功能块

**公式**

OUT = IN1 × IN2

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 被乘数

IN2 ANY\_NUM 乘数

OUT ANY\_NUM 积

**运行错误**

运行时如果出现下列情况，发出错误信息：

* 输出值的范围越限

### DIV 【除法运算】

该功能块实现除法运算。将输入IN1的值除以输入IN2的值后，结果赋值输出。

输入输出的数据类型必须相同，可以使用一些特殊函数处理不同类型的数据。当除法的数据类型是ANY\_INT组类型时，产生的结果如果有小数部分，则小数点后面的数将被截除，例如：7 ÷ 3 = 2；-7 ÷ 3 = -2。

**功能块图**



图8.1.4 除法运算功能块

**公式**

OUT = IN1 ÷ IN2

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 被除数

IN2 ANY\_NUM 除数

OUT ANY\_NUM 商

**运行错误**

运行时如果出现下列情况，发出错误信息：

* IN2 = 0

### AND 【“与”运算】

该功能块实现“与”运算功能。将输入端连接的两组数据进行“与”操作后将结果返回到输出。AND操作是逐位执行的。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.5 与运算功能块

**公式**

OUT = IN1 & IN2

ANY\_NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 输入1

IN2 ANY\_NUM 输入2

OUT ANY\_NUM 输出

### OR 【“或”运算】

该功能块实现“或”运算功能。将输入端连接的两组数据进行“或”操作后将结果返回到输出。OR操作是逐位执行的。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.6 或运算功能块

**公式**

OUT = IN1 OR IN2

ANY\_NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 输入1

IN2 ANY\_NUM 输入2

OUT ANY\_NUM 输出

### NOT 【“非”运算】

该功能块实现“非”运算功能。将输入端的数据进行“非”操作后将结果返回到输出。NOT操作是逐位执行的。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.7 非运算功能块

**公式**

OUT = NOT IN

ANY\_ NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

**参数描述**

参数 数据类型 意义

IN ANY\_ NUM 输入数据

OUT ANY\_ NUM 已逐位反相的数据输出

### XOR 【“异或”运算】

该功能块实现“异或”运算功能。将输入端连接的两组数据进行“异或”操作后将结果返回到输出。XOR操作是逐位执行的。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.8 异或运算功能块

**公式**

OUT = IN1 XOR IN2

ANY\_NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

**参数描述**

参数 数据类型 意义

IN1 ANY\_NUM 输入1

IN2 ANY\_NUM 输入2

OUT ANY\_NUM 输出

### INV 【求“反码”运算】

该功能块实现求反码运算功能。将输入端的数据进行求反码操作后将结果返回到输出。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.9 求反运算功能块

**公式**

OUT = ~ IN

ANY\_ NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

**参数描述**

参数 数据类型 意义

IN ANY\_ NUM 输入数据

OUT ANY\_ NUM 输出

### NEG 【求“负”运算】

该功能块实现求负运算功能。将输入端的数据进行求取其负值操作后将结果返回到输出。ANY\_NUM类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.1.10 求负运算功能块

**公式**

OUT = －IN

ANY\_ NUM包括BYTE、CHAR、DWORD、INTEGER和SHORT。

***参数描述***

参数 数据类型 意义

IN ANY\_ NUM 输入数据

OUT ANY\_ NUM 输出

## 逻辑库（logic）

### ANDn\_BOOL 【逻辑“与”】

该功能块实现逻辑“与”功能。在输入端连接的位序列进行“与”操作后将结果返回到输出。AND操作是逐位执行的。ANY\_BIT类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。功能块输入端的数量可以从２增加到８。

**功能块图**



图8.2.1 逻辑与运算功能块

**公式**

OUT = IN1 & IN2 & … & INn　　　（n <= 8）

**参数说明**

参数 数据类型 意义

IN1 ANY\_BIT 输入1

IN2 ANY\_BIT 输入2

INn ANY\_BIT 输入n

OUT ANY\_BIT 输出

### ORn\_BOOL 【逻辑“或”】

该功能块实现逻辑“或”功能。在输入端连接的位序列进行“或”操作后将结果返回到输出。OR操作是逐位执行的。ANY\_BIT类型组类型的数据能够被该功能块处理。

所有输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。功能块输入端的数量可以从２增加到８。

**功能块图**



图8.2.2 逻辑与运或算功能块

**公式**

OUT = IN1 OR IN2 OR … OR INn　　　（n <= 8）

**参数说明**

参数 数据类型 意义

IN1 ANY\_BIT 输入1

IN2 ANY\_BIT 输入2

INn ANY\_BIT 输入n

OUT ANY\_BIT 输出

### NOT\_BOOL【逻辑“非”】

该功能块实现逻辑“非”功能。在输入端连接的位序列逐位执行的“非”操作后将结果返回到输出。ANY\_BIT类型组类型的数据能够被该功能块处理。

输入的数据类型必须和输出数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**

图8.2.3 逻辑非运算功能块



**公式**

OUT = NOT IN

**参数说明**

参数 数据类型 意义

IN ANY\_BIT 输入位序列

OUT ANY\_BIT 已逐位反相的位序列输出

### XOR\_BOOL【逻辑“异或”】

该功能块实现逻辑“异或”功能。输入的位串按“异或”逻辑处理，并将结果赋给输出端。输入的位串逐位执行“异或”。 该功能块处理ANY\_BIT类型组数据。

输入端的数据类型必须和输出端的数据类型一致。可以使用一些专用的函数进行不同类型数据的处理。

**功能块图**



图8.2.4 逻辑异或运算功能块

**公式**

OUT = IN1 XOR IN2

**参数说明**

参数 数据类型 意义

IN1 ANY\_BIT 输入位串1

IN2 ANY\_BIT 输入位串2

OUT ANY\_BIT 输出位串

## 比较类（comparison）

### EQ【等于】

该功能块实现“等于”功能。如果所有的输入量都相等，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.1 等于功能块

**公式**

如果： （IN1 = IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

### GE【大于或等于】

该功能块实现“大于或等于”功能。如果所有的输入量都相等或以值递减方式排列，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.2 大于或等于功能块

**公式**

如果：（IN1 ≥ IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

### GT【大于】

该功能块实现“大于”功能。如果所有的输入量以值递减方式排列，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.3 大于功能块

**公式**

如果：（IN1 ＞ IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

### LE【小于或等于】

该功能块实现“小于或等于”功能。如果所有的输入量都相等或以值递增方式排列，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.4 小于或等于功能块

**公式**

如果：（IN1 ≤ IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

### LT【小于】

该功能块实现“小于”功能。如果所有的输入量以值递增方式排列，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.5 小于功能块

**公式**

如果：（IN1 ≤ IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

### NE【不等于】

该功能块实现“不等于”功能。如果IN1与IN2不相等，输出被置“1”，否则输出为“0”。

所有的输入量的数据类型必须是相同的。可以使用特定的函数来处理不同类型的数据。

**功能块图**



图8.3.6 不等于功能块

**公式**

如果：（IN1 ≠ IN2）

则： OUT = 1

否则： OUT=0

ANY\_NUM包括BYTE、CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN1 ANY\_ NUM 输入1

IN2 ANY\_ NUM 输入2

OUT BOOL 输出

## 双稳态类（bistable）

### RS【RS触发器】

该功能块实现RS触发器功能，其优先级为“复位优先”。

当输入R变为“1”时（R ＝ 1），输出Q变为“0”（Q ＝ 0）。即使输入R回到“0”，该状态仍然保持,此时若当输入S变为“1”（S ＝ 1）时，输出Q1变回“1”。如果输入R和S同时变为“1”，由于R的优先输入将使输出Q1置“0”。

当该功能块第一次被调用时，输出Q1的初始状态为“0”。

**功能块图**



图8.4.1 RS触发器功能块

**参数说明**

参数 数据类型 意义

S BOOL 置位输入

R BOOL 复位输入（优先）

Q BOOL 输出

### SR【SR触发器】

该功能块实现SR触发器功能，其优先级为“置位优先”。

当输入S变为“1”时（S ＝ 1），输出Q变为“1”（Q ＝ 1）。即使输入S回到“0”，该状态仍然保持,此时若当输入R变为“1”（R ＝ 1）时，输出Q1变回“0”。如果输入R和S同时变为“1”，由于S的优先输入将使输出Q1置“1”。

当该功能块第一次被调用时，输出Q1的初始状态为“0”。

**功能块图**



图8.4.2 SR触发器功能块

**参数说明**

参数 数据类型 意义

S BOOL 置位输入（优先）

R BOOL 复位输入

Q BOOL 输出

## 转换类（converter）

### B2\*\*\* 【字节类型转换】

该功能块将一个BYTE数据类型的输入转换为一个ANY\_NUM(char、double、date、dword、float、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

注意：EFB功能块的转换严格按照IEC的规则执行。因为EFB是作为类函数被实现的，因而将有一些非逻辑的转换，例如：BYTE\_TO\_DATE。因此，必须注明被传送到输出字的最高关键字的输入字节模式。

当将BYTE类型数据转换为ANY\_NUM组类型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

**功能块图**



图8.5.1 字节类型转换功能块

ANY\_NUM包括CHAR、DATE、DOUBLE、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN BYTE 输入值

OUT ANY\_NUM 输出值

### C2\*\*\* 【字符类型转换】

该功能块将一个CHAR数据类型的输入转换为一个ANY\_NUM(byte、double、date、dword、float、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将CHAR类型数据转换为ANY\_NUM组型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

**功能块图**



图8.5.2 字符类型转换功能块

ANY\_NUM包括BYTE、DATE、DOUBLE、DWORD、FLOAT、INT、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN CHAR 输入值

OUT ANY\_NUM 输出值

### DBL2\*\*\* 【双精度类型转换】

该功能块将一个DOUBE数据类型的输入转换为一个ANY\_NUM(byte、char、date、dword、float、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将DOUBLE类型数据转换为BYTE或WORD类型时，输入值的最小关键位被传送到输出。

**功能块图**



图8.5.3 双精度类型转换功能块

ANY\_NUM包括BYTE、DATE、CHAR、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN DOUBLE 输入值

OUT \*\*\* 输出值

### DT2\*\*\* 【日期类型转换】

该功能块将一个DATE数据类型的输入转换为一个ANY\_NUM(byte、char、double、dword、float、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将DOUBLE类型数据转换为BYTE或WORD类型时，输入值的最小关键位被传送到相应的输出。

**功能块图**



图8.5.4 日期类型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DWORD、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN DATE 输入值

OUT \*\*\* 输出值

### DW2\*\*\* 【双字型转换】

该功能块将一个DWORD数据类型的输入转换为一个ANY\_NUM(byte、char、double、date、float、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将DWORD类型数据转换为BYTE或WORD类型时，输入值的最小关键位被传送到相应的输出。

**功能块图**



图8.5.5 双字型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DATE、FLOAT、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN DWORD 输入值

OUT \*\*\* 输出值

### F2\*\*\* 【浮点类型转换】

该功能块将一个FLOAT数据类型的输入转换为一个ANY\_NUM(byte、char、double、date、dword、integer、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将FLOAT类型数据转换为ANY\_NUM组类型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

**功能块图**



图8.5.6 浮点类型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DATE、DWORD、INTEGER、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN FLOAT 输入值

OUT \*\*\* 输出值

### I2\*\*\* 【整型转换】

该功能块将一个INTEGER数据类型的输入转换为一个ANY\_NUM(byte、char、double、date、dword、float、short、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将INTEGER类型数据转换为ANY\_NUM组类型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

**功能块图**



图8.5.7 整型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DATE、DWORD、FLOAT、SHORT和WORD。

**参数说明**

参数 数据类型 意义

IN INTEGER 输入值

OUT \*\*\* 输出值

### S2\*\*\* 【短整型转换】

该功能块将一个SHORT数据类型的输入转换为一个ANY\_NUM(byte、char、double、date、dword、float、integer、word)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将SHORT类型数据转换为ANY\_NUM组类型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

**功能块图**



图8.5.8 短整型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DATE、DWORD、INTEGER、FLOAT和WORD。

**参数说明**

参数 数据类型 意义

IN SHORT 输入值

OUT \*\*\* 输出值

### W2\*\*\* 【字型转换】

该功能块将一个WORD数据类型的输入转换为一个ANY\_NUM (byte、char、double、date、dword、float、integer、short)类型的数据输出。可以使用特定的函数来处理不同类型的数据。

当将WORD类型数据转换为ANY\_NUM类型时，输入量的位模式被传送到输出的最小关键位，而位数不足时，输出的最高关键位被置“0”。

当将WORD类型数据转换为BYTE类型数据时，输入值得最小关键位被传送到输出。

**功能块图**



图8.5.9 字型转换功能块

ANY\_NUM包括BYTE、DOUBLE、CHAR、DATE、DWORD、INTEGER、SHORT和FLOAT。

**参数说明**

参数 数据类型 意义

IN WORD 输入值

OUT \*\*\* 输出值

### SCALING【量程转换】

该功能块将浮点型（FLOAT型）输入端数据转换为指定量程的数据输出。该块常用于模拟输入信号的工程单位量程转换。

**功能块图**



图8.5.10 量程转换功能块

**参数说明**

参数 数据类型 意义

IN FLOAT 输入值，范围0～1

MIN FLOAT 量程下限

MAX FLOAT 量程上限

OUT FLOAT 输出值

**公式**

OUT = IN × (Max - Min) + Min

## 定时器类（timers）

### TOFF 【延时消去定时器】

该功能块实现延时消去定时器功能。输入ET的初始状态在该功能块第一次被调用时为“0”。

**功能块图**



图8.6.1 延时消去定时器功能块

**参数说明**

参数 数据类型 意义

IN BOOL 启动延时

PT TIME 预设延时时间

Q BOOL 输出

ET TIME 内部时间

**时序图**

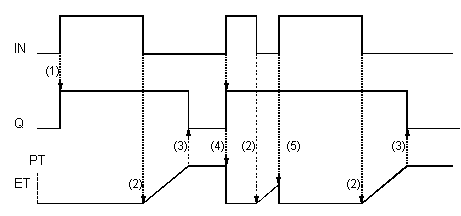


图8.6.2 延时消去定时器时序图

如上图所示，选取5个数据点，并得出如下说明：

* 当IN = 1时，Q = 1；
* 如果IN由1变为0，内部定时器ET将启动；
* 当内部定时时间ET = PT，Q = 0；
* 如果IN = 1且Q = 1，内部定时器将被停止或复位；
* 如果在内部定时器时间ET在达到PT之前，IN 跳变为1，不论输出Q是否回“0”，内部定时器ET将停止或复位而。

### TON 【延时启动定时器】

该功能块实现延时启动定时器功能。输入ET的初始状态在该功能块第一次被调用时为“0”。

**功能块图**



图8.6.3 延时启动定时器功能块

**参数说明**

参数 数据类型 意义

IN BOOL 启动延时

PT TIME 预设延时时间

Q BOOL 输出

ET TIME 内部时间

**时序图**

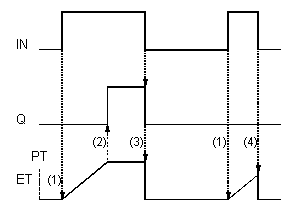


图8.6.4 延时启动定时器时序图

如上图所示，延时启动定时器有如下特点：

* 如果IN从0跳变为1时，内部定时器ET将启动，并且一旦当内部定时时间ET达到设定值 PT，Q = 1；
* 当IN = 0，Q=0时，内部定时器将被停止或复位；
* 如果在内部定时器时间ET在达到PT之前IN = 0，内部定时器ET将停止或复位而不论输出Q是否回“1”。

### TON\_P【带时间暂停功能的延时启动定时】

该功能块实现延时启动定时器功能。输入ET的初始状态在该功能块第一次被调用时为“0”。

**功能块图**



图8.6.5 带时间暂停功能的延时启动定时器功能块

**参数说明**

参数 数据类型 意义

S BOOL 启动延时指令

Pm BOOL 延时暂停指令

T DATE 预设延时时间

Q BOOL 电平输出

P BOOL 脉冲输出

ET DATE 延时进行时间

Tm DATE 延时剩余时间

**时序图**



图8.6.6 带时间暂停功能的延时启动定时器时序图

如上图所示，带时间暂停功能的延时启动定时器有如下特点：

* 如果S = 1，内部定时器ET、Tm将启动；
* 一旦当内部定时时间ET达到设定值 T，Q = 1；并且P端输出一个正向沿脉冲，宽度为一个程序执行周期；
* 如果在S = 1期间在Pm端收到一个高电平信号（Pm = 1），内部定时器ET、Tm处于保持状态，Pm信号撤销后恢复计时；
* 如果S = 0且Pm = 0，内部定时器将被复位，ET = 0，Tm = T；输出Q = 0；
* 如果在内部定时器时间ET在达到设定值T之前S = 0，内部定时器ET将复位。

### TP 【脉冲定时器】

该功能块实现延时消去定时器功能。输入ET的初始状态在该功能块第一次被调用时为“0”。

**功能块图**



8.6.7 脉冲定时器功能块

**参数说明**

参数 数据类型 意义

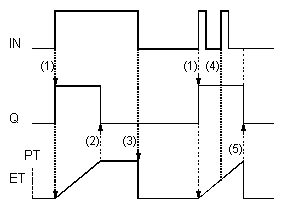
IN BOOL 启动延时

PT TIME 预设延时时间

Q BOOL 输出

ET TIME 内部时间

**时序图**



8.6.8 脉冲定时器时序图

如上图所示，脉冲定时器有如下特点：

* 当IN = 1，则Q = 1，内部定时器ET将启动，当内部定时时间ET达到设定值 PT，Q = 0（独立于输入IN）；
* 如果IN = 0，且Q=0，内部定时器将被停止或复位；如果内部定时器时间ET没有达到设定值PT，内部定时器时间不受输入IN的影响；
* 如果在内部定时器时间ET达到设定值PT且IN = 0，内部定时器ET将停止或复位且Q = 0。

### TIM\_TOL【时间累计器】

该功能块实现时间累计功能。

**功能块图**



8.6.9 时间累计器功能块

**参数说明**

参数 数据类型 意义

S BOOL 时间累计指令

R BOOL 累计时间清零指令

P BOOL 累计暂停指令

OUT DATE 累计时间输出

**功能说明**

1. 当S=1时启动时间累计，S=0时时间累计暂停；
2. 当R端输入一个正向脉冲时，累计输出时间OUT被清零，而无论S、P输入状态是0或1；
3. 当P=1时累计时间输出暂停，P=0时恢复时间累计。

## 计数器类（Counter）

### CTUD【递增/递减计数器】

该功能块实现递增或递减计数器。计数值为INTEGER型。

在输入R给一个“1”信号，将使一个“0”值被赋给CV输出。在输入LD给一个“1”信号，将使PV值被赋给CV输出。在CU输入端的每一个“0→1”的跳变，CV的值将加1。在CD端的每一个“0→1”的跳变，将使CV的值减1。

如果在CU和CD输入端都有一个连续的“1”信号，输入CU（递增计数器）优先。

如果在R和LD输入端都有一个连续的“1”信号，输入R（递增计数器）优先。

当CV≥PV时，输出QU被置“1”。

当CV≤0时，输出QD被置“1”。

**功能块图**



图8.7.1 递增/递减计数器

**参数说明**

参数 数据类型 意义

CU BOOL 递增触发器输入（上升沿）

CD BOOL 递减触发器输入（上升沿）

R BOOL 复位

LD BOOL 加载数据

PV INTEGER 预设值

QU BOOL 递增输出

QD BOOL 递减输出

CV INTEGER 计数器输出（实际值）

# 常用特殊功能块

## IO模件类（FCS）

### DI111 【16通道数字量输入模块】

从现场采集来的数字信号通过该功能块（对应相应位置的数字信号输入I/O模块）输入并传递给控制器控制程序进行运算和处理。

强制状态下（Disable = 1）可通过被选择的S1~S16引脚改变DI111功能块I1~I16状态，而不受I/O输入状态影响。

**功能块图**

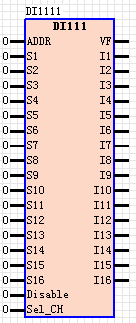


图9.1.1 DI111

**参数说明**

参数 数据类型 意义

ADDR WORD 离散信号输入模块地址

S1~S16 BOOL 强制输入，对应Disable及Sel\_CH设置时有效

Disable BOOL 强制输入使能（1=强制输入）

Sel\_CH WORD Disable=1时强制通道选择，为0时无强制通道

VF WORD 保留字

I1~I16 BOOL I/O通道状态及强制输入状态

**强制输入**

强制状态下（Disable=1）的通道选择设置（Sel\_CH），Sel\_CH为一个16位的WORD型输入（二进制表达为0000 0000 0000 0000~1111 1111 1111 1111），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输入，为3时表示第1、2通道为强制输入，Sel\_CH=0无强制输入。

**注意：数字信号输入模块DI111必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### DO111 【16通道数字量输出模块】

控制器通过数字信号输出块DO111（对应相应位置的数字信号输出I/O模块）将控制程序运算结果输送到现场执行设备，从而实现现场或过程控制设备的程序控制。

强制状态下（Disable = 1）可通过被选择的S1~S16引脚改变离散量I/O输出状态，而不受Q1~Q16状态影响。

**功能块图**

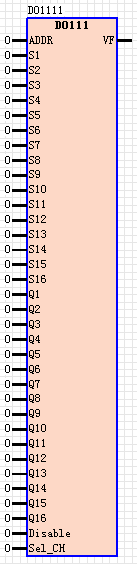


图9.1.2 DO111

**参数说明**

参数 数据类型 意义

ADDR WORD 离散信号输出模块地址

S1~S16 BOOL 强制输出，对应Disable及Sel\_CH设置时有效

Q1~Q16 BOOL 离散信号输出点（控制程序运算值输出）

Disable BOOL 强制输出使能（1=强制输出）

Sel\_CH WORD Disable=1时强制通道选择，为0时无强制通道

VF WORD 保留字

**强制输出**

强制状态下（Disable=1）的通道选择设置（Sel\_CH），Sel\_CH为一个16位的WORD型输出（二进制表达为0000 0000 0000 0000（值0）~1111 1111 1111 1111（值255）），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输出，为3时表示第1、2通道为强制输出，Sel\_CH=0无强制输出。

**注意：数字信号输出模块DO111必须在NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### AI111 【高精度模拟量输入模块】

从现场采集来的模拟信号（包括4~20mA、0~5V、0~10V、-5V~5V、-10V~10V）通过该功能块（对应相应位置的模拟量输入I/O模块AI111）输入并传递给给控制器控制程序进行运算和处理。在A1~A8引脚输出一个0~1范围的无量纲数据，可进行量程转换得到需要的工程（过程）测量值。

**功能块图**

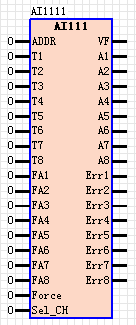


图9.1.3 AI111

**参数说明**

参数 数据类型 意义

ADDR WORD 模拟信号输入模块地址

T1~T8 BYTE 模拟输入信号类型及量程配置，见下表。

FA1~FA8 FLOAT 通道1~通道8强制输入

Force BOOL 强制输入开关：1=强制；0=正常

Sel\_CH BYTE Force=1时强制通道选择，为0时无强制通道

VF BYTE 通道断线诊断字

A1~A8 FLOAT 模拟输入信号转换值，范围0~1。

Err1~Err8 BOOL 通道1~通道8故障报警输出，1=故障报警

|  |  |
| --- | --- |
| **T1~T8**  **参数值** | **对应信号类型及量程** |
| 0 | 电流型 4 ~ 20mA |
| 1 | 电压型 0 ~ 5V |
| 2 | 电压型 -5V ~ +5V |
| 3 | 电压型 0 ~ 10V |
| 4 | 电压型 -10V ~ +10V |

**量程转换：**

模拟信号（包括4~20mA、0~5V、0~10V、-5V~5V、-10V~10V）采集后，在An端输出一个0~1范围的无量纲数据，需经过SCALING功能块进行量程转换后才能得到需要的工程（过程）测量值。

**断线诊断：**

对于4~20mA输入信号，在测量回路电流小于1.2mA，模块将判断该测量回路处于断线状态，并在VF端输出报警。定义如下：



图9.1.4 AI111断线诊断

**强制输入**

强制状态下（Force=1）的通道选择设置（Sel\_CH），Sel\_CH为一个8位的BYTE型输入（二进制表达为0000 0000（值0）~1111 1111（值255）），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输入，为3时表示第1、2通道为强制输入，Sel\_CH=0无强制输入。

**注意：模拟信号输入模块AI111必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### AO111 【模拟量（4~20mA）输出模块】

AO111功能块将过程控制参数转换为模拟4~20mA信号输出到模拟量输出I/O模件通道，实现对现场设备的连续控制。

**功能块图**

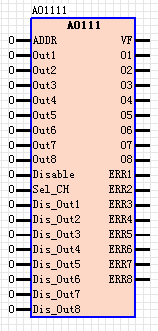
****

图9.1.5 AO111

**参数说明**

参数 数据类型 意义

ADDR WORD 模拟信号输出I/O模块地址

Out1~Out4 FLOAT 模拟信号输出通道（控制程序运算值输出），范围0~1对应4~20mA

Disable BOOL 强制输出使能（1=强制输出）

Sel\_CH WORD Disable=1时强制通道选择，为0时无强制通道

Dis\_out1~Dis\_out4 FlLOAT 强制输出，对应Disable及Sel\_CH设置时有效

VF BYTE 保留字

O1~O8 FLOAT 输出DA值监视，范围0~65535。

**强制输出**

强制状态下（Disable=1）的通道选择设置（Sel\_CH），Sel\_CH为一个8位的BYTE型输入（二进制表达为0000 0000（值0）~1111 1111（值255）），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输出，为3时表示第1、2通道为强制输出，Sel\_CH=0无强制输出。

**注意：模拟信号输入模块AO111必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### DI112 【事件记录顺序（SOE）模块】

事件顺序记录（SOE）模块为12通道4×3按组隔离，可输入无源或有源开关量信号，触发脉冲分辨率1ms。

通常情况下，该模块可作为开关量输入模块使用。若启用其SOE功能，必须为该模块提供GPS通讯对时信号，对时采用RS485接口方式。将输入转换为可处理的BOOL型数据和带时间标识的SOE事件序列。

**功能块图**

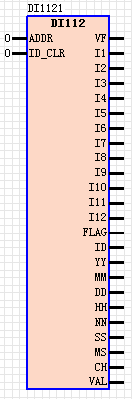


图9.1.6 DI112

**参数说明**

参数 数据类型 意义

ADDR WORD 模块地址

ID\_CLR WORD SOE事件读取指令

VF WORD 保留字

I1~I16 BOOL I/O通道状态

FLAG BYTE SOE事件标志

ID BYTE SOE事件指针

YY WORD ID指针对应的SOE事件时标：年

MM WORD ID指针对应的SOE事件时标：月

DD WORD ID指针对应的SOE事件时标：日

HH WORD ID指针对应的SOE事件时标：时

NN WORD ID指针对应的SOE事件时标：分

SS WORD ID指针对应的SOE事件时标：秒

MS WORD ID指针对应的SOE事件时标：毫秒

CH BYTE ID指针对应的SOE事件产生的I/O通道

VAL BYTE ID指针对应的SOE事件状态（上升沿或下降沿）

**功能说明**

* 1. SOE事件读取指令由上位控制软件（PMC）产生，因此要读取SOE事件记录需在ID\_CLR端定义一个指令变量并在上位监控系统引用（详细说明参考《PMC用户手册》）。
  2. FLAG = 1表示SOE模块内有SOE事件记录存在，

**注意：**

首次使用SOE事件输入模块必须要有GPS对时信号对SOE模块进行时标初始化校正，否则，SOE模块将无法产生正确的SOE事件信息。

SOE模块寄存器最多能够存储32条SOE事件，因此，应及时读取SOE模块中的SOE事件，以免造成SOE事件丢失。

**注意：模拟信号输入模块DI112必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### AI121 【8通道热电偶/mV输入模块】

该功能块用于热电偶（T/C）输入I/O模块AI121的通道配置和读数，以及I/O通道诊断信息、冷端补偿、输入强制等。模块输出为已进行冷端补偿的物理量值（摄氏温度（℃））。

**功能块图**

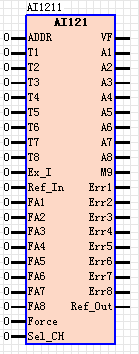


图9.1.7 AI121

**参数说明**

参数 数据类型 意义

ADDR WORD T/C输入模块地址

T1~T8 BYTE 通道1~通道8输入热电偶分度号(本版支持K型热电偶，T=0)

Ex\_I BOOL 1=外部冷端补偿；0=内部冷端补偿

Ref\_In FLOAT 外部冷端补偿输入

FA1~FA8 FLOAT 通道1~通道8强制输入

Force BOOL 强制输入开关：1=强制；0=正常

Sel\_CH BYTE Force=1时强制通道选择，为0时无强制通道

VF BYTE 通道诊断字

A1~A8 FLOAT 通道1~通道8带冷端补偿物理量（温度）输出

Err1~Err8 BOOL 通道1~通道8故障报警输出，1=故障报警

Ref\_Out FLOAT 冷端补偿值输出，Ex\_I =1为外部冷端补偿值；Ex\_I =0为内部冷端补偿值

**故障诊断**

热电偶输入I/O模块故障诊断字在VF端输出，并在Err1~Err8输出报警。定义如下：



图9.1.8 AI121断线诊断

**强制输入**

强制状态下（Force=1）的通道选择设置（Sel\_CH），Sel\_CH为一个8位的BYTE型输入（二进制表达为0000 0000（值0）~1111 1111（值255）），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输入，为3时表示第1、2通道为强制输入，Sel\_CH=0无强制输入。

**注意：8通道热电偶/mV输入模块AI121必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### AI131 【热电阻输入模块】

该功能块用于热电阻（RTD）输入I/O模块AI131的通道配置和读数，以及I/O通道诊断信息、输入补偿、输入强制等。模块输出为物理量值（摄氏温度（℃））。

热电阻（RTD）输入I/O模块支持二线制、三线制、四线制热电阻输入，Conductor软件系统提供RTD分度选择及线路补偿修正计算。

**功能块图**

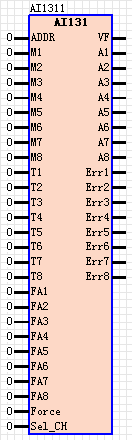


图9.1.9 AI131

**参数说明**

参数 数据类型 意义

ADDR WORD T/C输入模块地址

M1~M8 BYTE 各通道接线方式设置

T1~T8 BYTE 通道1~通道8输入热电阻分度号

FA1~FA8 FLOAT 通道1~通道8强制输入

Force BOOL 强制输入开关：1=强制；0=正常

Sel\_CH BYTE Force=1时强制通道选择，为0时无强制通道

VF BYTE 通道诊断字

A1~A8 FLOAT 通道1~通道8物理量（温度）输出

Err1~Err8 BOOL 通道1~通道8故障报警输出，1=故障报警

**分度号**

T1~T8分别对应1~8通道热电阻输入传感器分度号，AI131热电阻输入I/O模块支持的热电阻分度号如下：

|  |  |
| --- | --- |
| **T值** | **对应分度号** |
| 0 | Pt100 |
| 1 | Cu50 |
| 2 | Cu100 |

**断线诊断**

热电阻输入I/O模块故障诊断字在VF端输出，并在Err1~Err8输出报警。定义如下：



图9.1.10 AI131断线诊断

**强制输入**

强制状态下（Force=1）的通道选择设置（Sel\_CH），Sel\_CH为一个8位的BYTE型输入（二进制表达为0000 0000（值0）~1111 1111（值255）），不同的取值分别对应不同的强制通道或通道组合，例如值为1时表示第1通道为强制输入，为3时表示第1、2通道为强制输入，Sel\_CH=0无强制输入。

**补偿（修正）**

补偿一般用于热电阻测量方式或线路误差修正计算，C1~C8引脚为修正值输入，补偿方式下被直接加到对应通道在A1~A8输出。

**注意：热电阻输入模块AI131必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

### PI111 【双通道脉冲频率输入模块】

双通道脉冲频率输入模块，用于接近开关或电磁传感器等开关量的计数处理。

**功能块图**

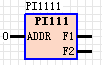


图9.1.11 PI111

**参数说明**

参数 数据类型 意义

ADDR WORD 脉冲输入I/O模块地址

F1 FLOAT 通道1脉冲实时频率（转速）输出

F2 FLOAT 通道2脉冲实时频率（转速）输出

**注意：双通道脉冲频率输入模块PI111必须在 NicPro编程环境中的目标系统配置中进行配置后相应功能块才能正常工作。**

## 网络通讯类

### WGFLOAT【网络浮点数广播】

该功能块将本地控制器节点的浮点型（FLOAT型）实时数据广播到NicSys®系统的控制网络上，以供该网络上的其他控制器读取相应的实时数据，本地控制器不能够读取该实时广播数据。

**功能块图**



图 9.2.1 网络浮点数广播功能块

**参数说明**

参数 数据类型 意义

SADDR WORD 节点（控制器）块地址（0~31）

TADDR WORD 数据源地址（0~1023）

IN1~IN8 FLOAT 广播数据发送通道

OUT1~OUT8 FLOAT 广播数据监视

**块地址（SADDR）**：块地址（SADDR）是指控制器内可引用的WGFLOAT、WGINT、WGBOOL等网络实时数据广播块的地址编码，地址范围0~31。同一控制器内的块地址（SADDR）必须保证唯一。

**数据源地址（TADDR）**：数据源地址（TADDR）指控制网络数据广播地址。一个NicSys®系统控制网络中最多可广播1024个块（WGFLOAT、WGINT、WGBOOL）的实时数据。在控制网络中，必须保证WGFLOAT、WGINT、WGBOOL广播块数据源地址的唯一性，即网络中所有广播块的源地址必须为0~1023，且不能有重复地址。

### WGINT【网络整型数广播】

该功能块将本地控制器节点的整型（INTEGER型）实时数据广播到控制网络上，以供该网络上的其他控制器读取相应的实时数据，本地控制器不能够读取该实时广播数据。

**功能块图**



9.2.2 网络整型数广播模块

**参数说明**

参数 数据类型 意义

SADDR WORD 节点（控制器）块地址（0~31）

TADDR WORD 数据源地址（0~1023）

IN1~IN8 INTEGER 广播数据发送通道

OUT1~OUT8 INTEGER 广播数据监视

\*\* SADDR、TADDR参数说明同9.2.1节。

### WGBOOL【网络布尔量广播】

该功能块将本地控制器节点的布尔型（BOOL型）实时数据广播到控制网络上，以供该网络上的其他控制器读取相应的实时数据，本地控制器不能够读取该实时广播数据。

**功能块图**



9.2.3 网络布尔量广播模块

**参数说明**

参数 数据类型 意义

SADDR WORD 节点（控制器）块地址（0~31）

TADDR WORD 数据源地址（0~1023）

IN1~IN32 BOOL 广播数据发送通道

\*\* SADDR、TADDR参数说明同9.2.1节。

### RGFLOAT【网络浮点数读取】

该功能块将控制网络上广播的浮点型（FLOAT型）实时数据读取到本地控制器，以供本地控制器引用。

**功能块图**



9.2.4 网络浮点数读取功能块

**参数说明**

参数 数据类型 意义

TADDR WORD 网络上数据源地址（0~1023）

OUT1~OUT8 FLOAT 广播数据读取通道

\*\* TADDR参数说明同9.2.1节。

### RGINT【网络整型数读取】

该功能块将控制网络上广播的整型（INTEGER型）实时数据读取到本地控制器，以供本地控制器引用。

**功能块图**



9.2.5 网络整型数读取功能块

**参数说明**

参数 数据类型 意义

TADDR WORD 网络上数据源地址（0~1023）

OUT1~OUT8 INTEGER 广播数据读取通道

\*\* TADDR参数说明同9.2.1节。

### RGBOOL【网络布尔量读取】

该功能块将控制网络上广播的布尔型（BOOL型）实时数据读取到本地控制器，以供本地控制器引用。

**功能块图**



图9.2.6 网络布尔量读取功能块

**参数说明**

参数 数据类型 意义

TADDR WORD 网络上数据源地址（0~1023）

OUT1~OUT32 BOOL 广播数据读取通道

\*\* TADDR参数说明同9.2.1节。

## 常用驱动级（drivers）

### MTR\_LGE【马达驱动1】

该功能块实现对一类马达设备的启/停控制。

特点：

* 驱动故障闭锁输出指令
* 指令宽度可调
* 状态验证输出

**功能块图**



9.3.1 马达驱动功能块

**参数说明**

参数 数据类型 意义

L\_R BOOL 远程/就地切换

AM BOOL 自动/手动切换

KO BOOL 操作员合闸指令

AO BOOL 自动合闸指令

PO BOOL 联锁合闸指令

FO BOOL 允许合闸条件

LO BOOL 禁止启动

RUN BOOL 已运行反馈

DIS BOOL 设备故障

STOP BOOL 已停止反馈

LC BOOL 禁止停止

FZ BOOL 允许跳闸条件

PC BOOL 保护跳闸指令

PZ BOOL 联锁跳闸指令

AC BOOL 自动跳闸指令

KC BOOL 操作员分闸指令

MPP BOOL 故障及超时确认

FIX BOOL 检修状态

MT BOOL 1=验证状态反馈启动；０=无验证状态反馈启动

TON DATE 合闸超时时间

TOFF DATE 分闸超时时间

STRENO BOOL 合闸条件不满足

STRCMD BOOL 合闸命令

MOTRDIS BOOL 设备驱动故障

STPCMD BOOL 分闸命令

STPENO BOOL 分闸条件不满足

DTrip BOOL 故障跳闸（逻辑判断）

Status INTEGER 设备状态

ETon DATE 启动命令执行时间监视

EToff DATE 停止命令执行时间监视

### MTR\_LTE【马达驱动2】

该功能块实现对二类马达设备的启/停控制。

**功能块图**



图9.3.2（1） 马达驱动功能块

**参数说明**

参数 数据类型 意义

L\_R BOOL 远程/就地切换

AM BOOL 自动/手动切换

KO BOOL 操作员合闸指令

AO BOOL 自动合闸指令

PO BOOL 联锁合闸指令

FO BOOL 允许合闸条件

RUN BOOL 已运行反馈

DIS BOOL 设备故障

STOP BOOL 已停止反馈

FZ BOOL 允许跳闸条件

PC BOOL 保护跳闸指令

PZ BOOL 联锁跳闸指令

AC BOOL 自动跳闸指令

KC BOOL 操作员分闸指令

T1 DATE 合闸超时时间

T2 DATE 分闸超时时间

STRENO BOOL 合闸条件不满足

STRCMD BOOL 合闸命令

MOTRDIS BOOL 设备驱动故障

STPCMD BOOL 分闸命令

STPENO BOOL 分闸条件不满足

ET1 DATE 启动命令执行时间监视

ET2 DATE 停止命令执行时间监视

**逻辑图**



图9.3.2（2）马达驱动功能块逻辑图

### D\_AS\_VAL【电动门驱动1】

该功能块适用于带阀位反馈的电动阀门的开/关控制。

**功能块图**



图9.3.3 电动门驱动功能块

**参数说明**

参数 数据类型 意义

AM BOOL 自动/手动模式

KO\_S BOOL 操作员开短指令

KO\_L BOOL 操作员开长指令

AO BOOL 自动开指令

PO BOOL 联锁开指令

DPO BOOL 超驰开指令

KO\_FB BOOL 禁止操作员开指令

LO BOOL 禁止开指令

FO BOOL 允许开条件

OPN BOOL 开行程反馈

AI FLOAT 阀位反馈信号

MaxR FLOAT 位置信号上限

MinR FLOAT 位置信号下限

CLS BOOL 关行程反馈

MPP BOOL 中停/故障复位指令

FZ BOOL 允许关条件

LC BOOL 禁止关指令

KC\_FB BOOL 禁止操作员关指令

PC BOOL 联锁及保护关指令

AC BOOL 自动关指令

KC\_L BOOL 操作员关长指令

KC\_S BOOL 操作员关短指令

PS BOOL 设备或电源故障

TD DATE 驱动超时设定

ONENO BOOL 打开条件不满足

ONCMD BOOL 打开（1）命令

OFFCMD BOOL 关闭条件不满足

OFFENO BOOL 关闭（1）命令

TIMOUT BOOL 驱动超时故障

ETD DATE 开/关命令执行时间监视

### D\_DS\_VAL【电动门驱动2】

该功能块适用于普通型电动阀门的开/关控制。

**功能块图**



图9.3.4（1） 电动门驱动功能块

**参数说明**

参数 数据类型 意义

AM BOOL 自动/手动模式

KO BOOL 操作员开指令

AO BOOL 自动开指令

PO BOOL 联锁开指令

KO\_FB BOOL 禁止操作员开指令

LO BOOL 禁止开指令

FO BOOL 允许开条件

OPN BOOL 开行程反馈

MPP BOOL 中停/故障复位指令

LC BOOL 禁止关指令

KC\_FB BOOL 禁止操作员关指令

FZ BOOL 允许关条件

CLS BOOL 关行程反馈

PC BOOL 联锁及保护关指令

AC BOOL 自动关指令

KC BOOL 操作员关指令

PS BOOL 设备或电源故障

TD DATE 驱动超时设定

ONENO BOOL 打开条件不满足

ONCMD BOOL 打开（1）命令

OFFCMD BOOL 关闭条件不满足

OFFENO BOOL 关闭（1）命令

TIMOUT BOOL 驱动超时故障

ETD DATE 开/关命令执行时间监视

**逻辑图**



图9.3.4（2） 电动门驱动功能块逻辑图

### D\_DT\_VAL【电动门驱动3】

该功能块适用于带中停功能电动阀门的开/关控制。

**功能块图**



图9.3.5 电动门驱动功能块

**参数说明**

参数 数据类型 意义

AM BOOL 自动/手动模式

KO BOOL 操作员开指令

AO BOOL 自动开指令

PO BOOL 联锁开指令

OPN BOOL 开反馈

FO BOOL 允许开条件

KO\_FB BOOL 禁止操作员开指令

LO BOOL 禁止开指令

MPP BOOL 中停/故障复位指令

LC BOOL 禁止关指令

KC\_FB BOOL 禁止操作员关指令

FZ BOOL 允许关条件

CLS BOOL 关反馈

PC BOOL 联锁及保护关指令

AC BOOL 自动关指令

KC BOOL 操作员关指令

DIS BOOL 设备故障

L\_R BOOL 1=就地；0=远方

TS BOOL 1=脉冲指令；0=指令宽度由TD决定

TP DATE 脉冲指令宽度

TD DATE 驱动超时设定

ONENO BOOL 打开条件不满足

ONCMD BOOL 打开（1）命令

OFFCMD BOOL 关闭条件不满足

OFFENO BOOL 关闭（1）命令

MPPT BOOL 中停/复位输出

TIMOUT BOOL 驱动超时故障

STU INTEGER 设备状态

ETD DATE 开/关命令执行时间监视

## 程序处理类（MCS）

### AVE4W【四输入端加权平均值】

**功能块图**



9.4.1 四输入端加权平均值模块

**参数说明**

参数 数据类型 意义

IN1 FLOAT 输入值1

IN2 FLOAT 输入值2

IN3 FLOAT 输入值3

IN4 FLOAT 输入值4

K1 FLOAT 加权系数1

K2 FLOAT 加权系数2

K3 FLOAT 加权系数3

K4 FLOAT 加权系数4

OUT FLOAT 加权平均值

W\_SUM FLOAT 加权和

K\_SUM FLOAT 系数和

ERR BOOL 系数错误

**算法描述**

K\_SUM = k1 + k2 + k3 + k4

W\_SUM = k1\*IN1 + k2\*IN2 + k3\*IN3 + k4\*IN4

若K\_SUM = 0

ERR = TRUE；

否则

ERR = FALSE；

OUT = W\_SUM / K\_SUM；

### BALANCE【二路输出平衡模块】

**功能块图**



9.4.2 二路输出平衡模块

**参数说明**

参数 数据类型 意义

X FLOAT 输入量

DB FLOAT 差值输入

TR1 FLOAT 第一路跟踪量

TS1 FLOAT 第一路跟踪切换开关

TR2 FLOAT 第二路跟踪量

TS2 FLOAT 第二路跟踪切换开关

YH FLOAT Y1和Y2的上限

YL FLOAT Y1和Y2的下限

Y1 FLOAT 第一路平衡值输出

Y2 FLOAT 第二路平衡值输出

ERR FLOAT 上下限错误

**算法描述**

1. 如果TS1 = 1，TS2 = 1，Y1 = TR1，Y2 = TR2；
2. 如果TS1 = 1，TS2 = 0，Y1 = TR1，Y2 = 2 \* X – Y1；
3. 如果TS1 = 0，TS2 = 1，Y1 = 2 \* X – Y2，Y2 = TR2；
4. 如果TS1 = 0，TS2 = 0，Y1 = X – DB，Y2 = X – DB；
5. 如果YH < YL，Y1 = 0，Y2 = 0，ERR = 1；
6. 如果Y1 > YH，Y1 = YH；如果Y1 < YL，Y1 = YL；
7. 如果Y2 > YH，Y2 = YH；如果Y2 < YL，Y2 = YL；

### COMP\_DB【带死区和磁滞的比较器】

**功能块图**



图9.4.3 带死区和磁滞的比较器模块

**参数说明**

参数 数据类型 意义

IN1 FLOAT 输入1

IN2 FLOAT 输入2

DBAND FLOAT 死区限

HYST FLOAT 磁滞限

GREATER BOOL 输入1大于输入2

EQUAL BOOL 输入1等于输入2

LESS BOOL 输入1小于输入2

**算法描述**

1. 当DBAND <= HYST，若IN1 - IN2 > DBAND，GREATER = 1；若IN1 - IN2 = DBAND，EQUAL = 1；若IN1 - IN2 < DBAND，LESS = 1。
2. 若|IN1 - IN2| > DBAND - HYST，则GREATER = 0，EQUAL = 0，LESS = 0。

### DECIS\_D【三路输入表决器】

**功能块图**



图9.4.4 三路输入表决器模块

**参数说明**

参数 数据类型 意义

C1V FLOAT 测量通道1

C2V FLOAT 测量通道2

C3V FLOAT 测量通道3

C1Bad BOOL 测量通道1通道故障

C2Bad BOOL 测量通道2通道故障

C3Bad BOOL 测量通道3通道故障

V FLOAT 表决器输出

CH\_Bad BOOL 两个以上测量通道故障

**算法描述**

1. 表决器算法如下：

avge = (C1V + C2V + C3V) / 3.0；

aa = fabs(avge - C1V)；

bb = fabs(avge - C2V)；

cc = fabs(avge - C3V)；

if (aa >= bb) {

if (aa >= cc) {

V = (C2V + C3V) / 2.0；

}

else {

V = (C1V + C2V) / 2.0；

}

}

else {

if(bb >= cc) {

V = (C1V + C3V) / 2.0；

}

else {

V = (C1V + C2V) / 2.0；

}

}

}

CH\_Bad = C1Bad && C2Bad && C3Bad

该表决器输出偏差较小的两个输入值的平均在值。

### DEV\_HL【偏差报警】

该功能块实现PV、SP的偏差计算及越限报警输出。

**功能块图**



图9.4.5 偏差报警模块

**参数说明**

参数 数据类型 意义

PV FLOAT 测量值输入

SP FLOAT 设定值输入

SPV FLOAT 偏差设定输入

D BOOL 偏差越限报警输出

D1 BOOL 偏差为正值指示

D2 BOOL 偏差为负值指示

ERR FLOAT 偏差输出

**算法描述**

1. ERR = PV – SP；
2. 若|PV – SP| > SPV，则D = 1。
3. 若ERR > 0，则D1 = 1；若ERR < 0，则D2 = 1。

### HLALM【带死区的高低限报警】

带死区和故障闭锁的高低限值报警模块。

**功能块图**



9.4.6 带死区的高低限报警模块

**参数说明**

参数 数据类型 意义

X FLOAT 输入值

H FLOAT 报警上限

L FLOAT 报警下限

HDB FLOAT 上限报警死区

LDB FLOAT 下限报警死区

X\_Bad BOOL 故障闭锁输入

D1 BOOL 越上限报警

D2 BOOL 越下限报警

**算法描述**

1、若X > H，D1 = 1；X < H – HDB，D1 = 0；

2、若X < L，D2 = 1；X > L + LDB，D2 = 0；

3、若X\_Bad = 1，D1 = 0，D2 = 0。

### HLALM\_T【带死区及延时的高低限报警】

**功能块图**



图9.4.7 带死区及延时的高低限报警模块

**参数说明**

参数 数据类型 意义

X FLOAT 输入值

H FLOAT 报警上限

L FLOAT 报警下限

HDB FLOAT 上限报警死区

LDB FLOAT 下限报警死区

X\_Bad BOOL 故障闭锁输入

HTm DATE 上限报警延时

LTm DATE 下限报警延时

D1 BOOL 越上限报警

D2 BOOL 越下限报警

HEt DATE 上限报警延时指示

LEt DATE 下限报警延时指示

**算法描述**

1、若X > H，延时HTm后，D1 = 1；X < H – HDB，D1 = 0；

2、若X < L，延时LTm后，D2 = 1；X > L + LDB，D2 = 0；

3、若X\_Bad = 1，D1 = 0，D2 = 0。

### K\_SQRT【加权平方根】

**功能块图**



9.4.8 加权平方根模块

**参数说明**

参数 数据类型 意义

IN FLOAT 输入值

K FLOAT 加权系数

CUTOFF FLOAT 截断点

OUT FLOAT 开方结果

ERR BOOL 1：输入小于零或截断点小于零

**算法描述**

1. 如果IN < 0 或者CUTOFF < 0，ERR = 1且OUT = 0；
2. OUT = K × SQRT（IN）。

### LEADLAG【超前滞后模块】

该功能块实现带有时间滞后的PD组件（伴随低通滤波器）。

其具有以下一些特性：

* 可定义延时的D部件（微分）
* 手动，自动和中断操作模式

**功能块图**



图9.4.9 超前滞后模块

**参数说明**

参数 数据类型 意义

X FLOAT 输入变量

MAN BOOL 1：手动模式

HALT BOOL 1：中断模式

GAIN FLOAT 微分增益

LEAD DATE 微分时间常数

LAG DATE 延迟时间常数

YMAX FLOAT 输出上限

YMIN FLOAT 输出下限

YMAN FLOAT 手动操作量

Y FLOAT 输出

QMAX BOOL 越上限报警

QMIN BOOL 越下限报警

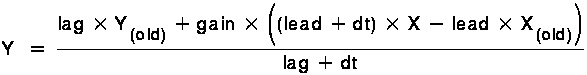
ERR BOOL 高低限错误报警

**算法描述**

传递函数：



计算公式：



其中

X（old）：前一周期的输出Y值

Y（old）：前一周期的输出Y值

dt：当前周期和前一周期的时间差

**参数说明**

功能块的参数分配是通过决定微分增益gain，微分时间常数lead和延迟时间常数lag等参数以期达到满意的结果。

对于一个非常短的采样时间和一个单位阶跃输入X（X从0跳变到1.0），输出Y跳变到一个新值gain\*lead/lag（理论值，事实上由于采样时间不可能无限的小，所以该值应比理论值略小），然后随着延迟时间常数的消失，输出Y将达到值gain\*1.0。

**操作模式**

通过参数halt和man的输入，有三种操作模式可供选择：

操作模式 man halt

自动模式 0 0

手动模式 1 0或1

中断模式 0 1

**一、自动模式**

自动模式下的功能块操作参考“参数描述”。

**二、手动模式**

手动模式下手动操作量YMAN被直接传送到输出Y。

**三、中断模式**

中断模式下输出Y值保持在最近一次的计算值，该值能够被应用户重写。

**举例**

下列的图示说明了功能块的阶跃响应的例子。

（1）当lead = lag时，功能块是一个乘数因子为gain的纯粹的乘法块。

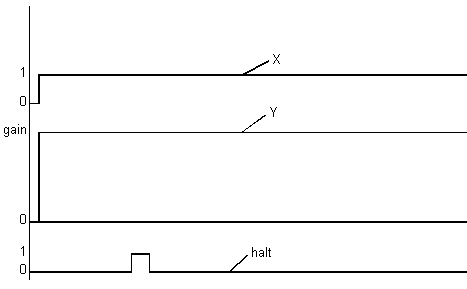


图9.4.9（1） 超前滞后模块的阶跃响应

（2）当lead/lag = 0.5，gain = 1时，输出Y跳变到终值的一半，接着，随着延迟时间常数lag的消失，最终Y值又回到终值gain\*X。

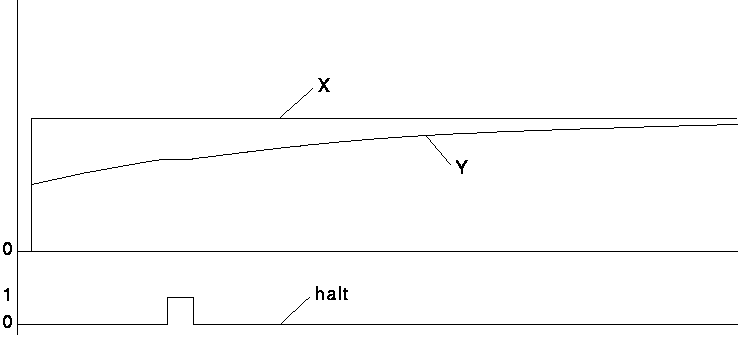


图9.4.9（2） 超前滞后模块的阶跃响应

（3）当lead/lag = 2，gain = 1时，输出Y跳变到终值的一倍，接着，随着延迟时间常数lag的消失，最终Y值又回到终值gain\*X。

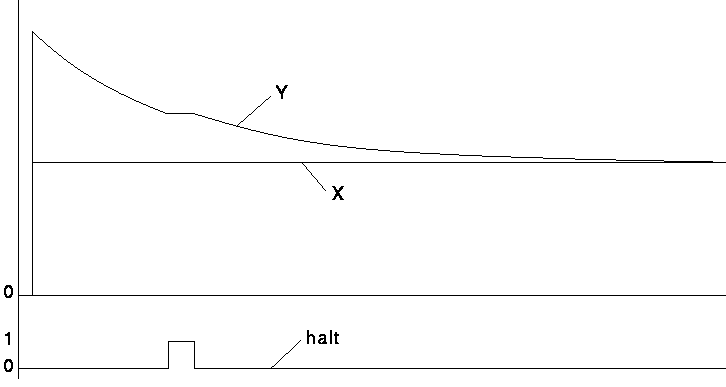


图9.4.9（3） 超前滞后模块的阶跃响应

**运行错误**

* 输入YMAN和X是无效的浮点数。

### LIMIT【限幅输出模块】

该功能块实现限幅输出。如果输入IN值在最小值MN和最大值MX之间，则功能块将输入值不变得传送到输出。如果输入IN小于最小值MN，则将最小值MN传送到输出；如果输入IN大于最大值MX，则将最大值传送到输出。

**功能块图**



图9.4.10 限幅输出模块

**参数说明**

参数 数据类型 意义

MIN FLOAT 输出下限

IN FLOAT 输入值

MAX FLOAT 输出上限

OUT FLOAT 输出

ERR BOOL 输出越限

**算法描述**

1、如果MIN ≤ IN ≤ MAX，OUT = IN

2、如果IN > MX，OUT = MAX，ERR = 1

3、如果IN < MN，OUT = MIN，ERR = 1

### LIMV【一阶速度限制器】

该功能块实现带控制变量限制的一阶速度积分器。

输入变量X值的变化率被预先设定的值RATE限制，输出Y则受限于YMAX和YMIN，该功能块适用于转速设备和感应电机的信号处理。

该功能块具有如下特性：

* 手动模式、中断模式和自动模式
* 自动模式下的控制变量限制

**功能块图**



图9.4.11 一阶速度限制器模块

**参数说明**

参数 数据类型 意义

MAN BOOL “1”为手动模式

HALT BOOL “1”为中断模式

X FLOAT 输入值

RATE FLOAT 最大速度变化率

YMAX FLOAT 控制上限

YMIN FLOAT 控制下限

YMAN FLOAT 手动控制输出值

Y FLOAT 输出值

QMAX BOOL 为“1”时输出Y越上限

QMIN BOOL 为“1”时输出Y越下限

对于输出Y的最大速度变化率RATE和输出上限YMAX和输出下限TMIN必须被定义。最大输出速度将限制每秒钟输出Y允许的变化量。根据最大速度变化率参数RATE可以计算出输出Y，当RATE＝0时，Y＝X。输出上下限YMAX和YMIN则使得输出Y维持在一个给定的范围内。因此， YMIN<=Y<=YMAX，当输出Y越限时，将通过QMAX或QMIN给出越限报警，而越限的部分将被截除。

* 当Y>=YMAX，QMAX=1
* 当Y<=YMIN，QMIN=1

**算法描述**

**功能块模式**

模式 MAN HALT

自动模式 0 0

手动模式 1 0或1

中断模式 0 1

**自动模式**

在自动模式下，Y的当前值通过计算并输出。

**手动模式**

在手动模式下，手动值YMAN被直接输送到Y输出端，而不受YMAX或YMIN的限制。

**中断模式**

在中断模式下，输出Y保持前一次计算值。输出不会被程序更新，而由用户直接重写。例如：

下图表达了LIMV功能块的动态属性。功能块将依据它的最大速度RATE跟随输入X跳变。在中断模式期间，输出Y将保持在其当前值且不变。输出Y的限制YMAX和YMIN将通过其相应的响应信号QMAX和QMIN来显示。

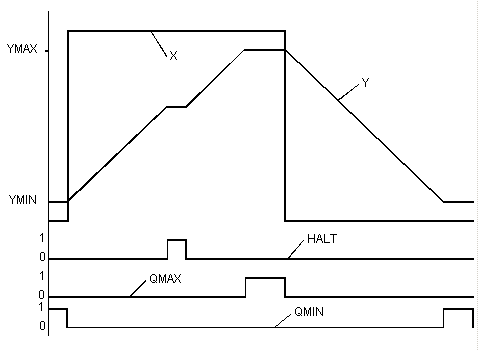


图9.4.11（1） 一阶速度限制器中断模式下的输出

**运行出错**

* 当YMAX<YMIN时，将产生错误。

### MULDIV\_W【三输入加权乘除法器】

MULDIV\_W功能块允许对3个输入变量进行加权乘法或除法运算。

**功能块图**



图9.4.12 一阶速度限制器模块

**参数说明**

参数 数据类型 意义

IN1 FLOAT 输入值1

IN2 FLOAT 输入值2

IN3 FLOAT 输入值3

K FLOAT 加权系数

c1 FLOAT 计算系数1

c2 FLOAT 计算系数2

c3 FLOAT 计算系数3

c3 FLOAT 计算系数4

OUT FLOAT 输出值

ERR FLOAT 除数为零，即计算系数3与输入值3之和为零

**算法描述**

OUT =（K \*（IN1 + c1）\*（IN2 + c2））/（IN3 + c3）+ c4

### PID 【PID调节器】

该功能块实现PID调节器。将产生一个基于设定值SP和过程变量SP的系统偏差ERR。系统偏差ERR将在操作变量Y产生一个变化。

该功能块具有如下一些特殊特性：

* 具有独立的GAIN、TI、TD设置的真正的PID调节器
* 手动、中断和自动模式
* 平滑的手动/自动切换
* 操作变量限制（调节器输出）
* 独立的P-portion（比例作用）、I-portion（积分作用）和D-portion（微分作用）
* 防饱和复位（anti windup reset）
* 防饱和校正（anti windup measure），仅在I-portion活动时
* 可定义延时的微分作用
* 微分作用可转换成过程变量SP或系统偏差ERR

功能块公式为 Y = YP + YI + YD + BIAS

其中

YP：P-portion，比例作用

YI：I-portion，积分作用

YD：D-portion，微分作用

BIAS：扰动

在计算出各个作用的总和后，受限的操作变量（调节器输出Y）将在一个有效的区间内变化：YMIN<=Y<=YMAX；并且依赖于EN\_P，EN\_I和EN\_D的输入，各部分的计算如下：

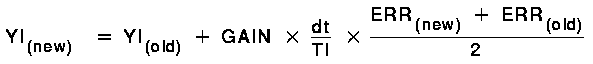
YP对于手动、自动和中断模式时

EN\_P = 1：YP = GAIN×ERR

EN\_P = 0：YP = 0

YI对于自动模式

EN\_I = 1：



EN\_I = 0：YI = 0

I-portion 作用依据梯形规则来构成。

YI对于手动和中断模式

EN\_I = 1：YI = Y – YP – BIAS

其中

Y：中断模式下的当前输出或手动模式下的输出YMAN。

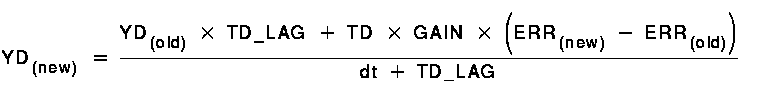
YP：比例作用

BIAS：扰动

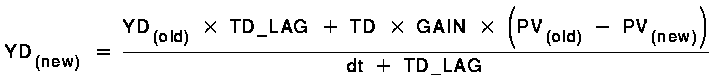
EN\_I = 0：YI = 0

YD对于自动模式

EN\_D = 1 且 D\_ON\_X = 0



EN\_D = 1 且 D\_ON\_X = 1



EN\_D = 0：YD = 0

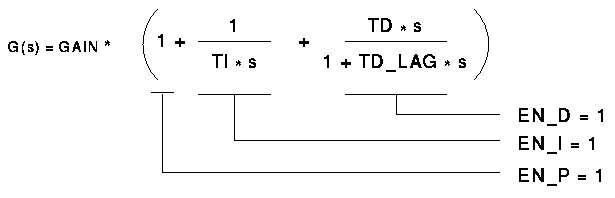
YD对于手动或中断模式

YD = 0

说明：

* ERR是系统偏差（SP – PV）
* ERR（new）是当前扫描阶段的系统偏差
* ERR（old）是前一扫描阶段的系统偏差
* PV（new）是当前扫描阶段的过程变量
* PV（old）是前一扫描阶段的过程变量
* dt 是当前扫描周期和前一扫描周期之差

下面的配置图表达了PID调节器的结构。通过设置比例带GAIN、积分时间TI和微分时间TD完成对功能块的纯PID参数设置。其传递函数：



通过EN\_P、EN\_I和EN\_D参数的选择，可以实现不同类型的调节器功能。

调节器类型 EN\_P EN\_I EN\_D

P调节器 1 0 0

PI调节器 1 1 0

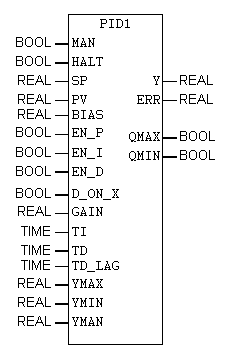
PD调节器 1 0 1

PID调节器 1 1 1

I调节器 0 1 0

而I-portion（积分作用）也可以通过将TI＝0而终止。

**功能块图**



**参数说明**

参数 数据类型 意义

MAN BOOL “1”为手动模式

HALT BOOL “1”为中断模式

SP REAL 设定值输入

PV REAL 过程变量

BIAS REAL 扰动输入

EN\_P BOOL 为“1”时进行比例作用

EN\_I BOOL 为“1”时进行积分作用

EN\_D BOOL 为“1”进行微分作用

D\_ON\_X BOOL 为“1”时微分作用依赖于过程变量，为“0”时微分作用依赖于系统偏差

GAIN REAL 比例带（比例增益）

TI TIME 积分时间

TD TIME 微分时间

TD\_LAG TIME 微分滞后时间

YMAX REAL 控制上限

YMIN REAL 控制下限

YMAN REAL 手动调节输出值

ERR REAL 系统偏差输出

Y REAL 操作变量（调节器输出）

QMAX BOOL 为“1”时输出Y达上限

QMIN BOOL 为“1”时输出Y达下限

**功能块模式**

通过MAN和HALT参数选择，可以实现三种不同的操作模式。

操作模式 MAN HALT

自动模式 0 0

手动模式 1 0或1

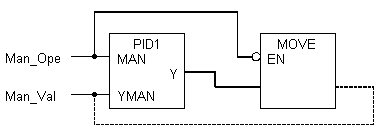
中断模式 0 1

**自动模式**

在自动模式下，以离散的PID运算法则确定由过程变量PV还是参数变量SP来决定调节器的输出Y。调节器输出受限于YMAX和YMIN。该限制同样适用于防饱和复位（anti windup reset）。

通常，从自动模式向手动模式切换是不平滑的，因为在自动模式时输出Y可以是YMIN～YMAX之间的任何一个值，当向手动切换时，输出Y则被强制为YMAN值。尽管这样，从自动模式向手动模式切换仍然能够达到预期的平滑，这里有两种选择：

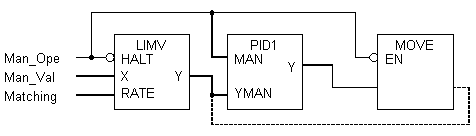
1、采用MOVE功能块函数，将YMAN的值加载到Y。



注：上图的表达只能近似的阐述该数据加载过程。因为，在NicPro中，不能够出现象上图中的输出直接连接到输入的这样一种连接环，编程时可以通过变量来代替这一回环。

当PID调节器在自动模式工作时（MAN＝0），MOVE功能块被执行，YMAN的值将跟踪调节器输出值Y，即YMAN＝Y，当调节器从自动切换到手动时，MOVE功能块的传送被中断，当在此之前Y＝YMAN，所以可以实现模式切换时的平滑过渡，而在手动模式下，你可以慢慢调整YMAN输入值。

2、如果不打算改变YMAN的值（由于它可能是一个固定值），则可以采用速度积分器（LIMV）来超前解决以实现平滑切换。



注：上图的表达只能近似的阐述该数据加载过程。因为，在Concept中，不能够出现象上图中的输出直接连接到输入的这样一种连接环，编程时可以通过变量来代替这一回环。

当PID调节器在自动模式工作时（MAN＝0），MOVE功能块被执行，YMAN的值将跟踪调节器输出值Y，即YMAN（PID1）＝Y（PID1），当调节器从自动切换到手动时，MOVE功能块的传送被中断，当在本执行周期内Y（PID1）＝YMAN（PID1），而在下一个执行周期开始，YMAN（PID1）才在LIMV功能块的作用下和实际的YMAN（Man\_Ope）输入值匹配，所以可以实现模式切换时的平滑过渡，而在手动模式下，你可以慢慢调整YMAN输入值。

**手动模式**

在手动模式下，YMAN值被直接送到调节器输出Y。当然，调节输出依然受限于YMAX和YMIN。只要内部变量匹配，积分部分能起作用，PI调节器能够从手动模式平滑的切换到自动模式。对调节器输出的限制也是对防饱和复位（anti windup reset）的限制。

**中断模式**

在中断模式下，控制输出Y保持中断模式开始时的调节输出值，功能块不再改变调节器输出Y（调节器仍然工作），此时，Y＝Y（old）。内部变量保持匹配，调节器比例作用和积分作用的和与调节器输出一致，并保持连续平滑输出。对调节器输出的限制也是对防饱和复位（anti windup reset）的限制。中断模式下的一个重要特征就是：通过外部操作器设置的调节器输出Y直接匹配调节器内部各部分。

在该模式下，微分作用被自动设置为“0”。

**运行出错**

* 当YMAX < YMIN时，系统运行出错。

### SUM\_W【三输入加权加法器】

该功能块实现三输入加权加法运算。

**功能块图**



9.4.14 三输入加权加法器模块

**参数说明**

参数 数据类型 意义

IN1 FLOAT 输入信号1

IN2 FLOAT 输入信号2

IN3 FLOAT 输入信号3

k1 FLOAT 加权系数1

k2 FLOAT 加权系数2

k3 FLOAT 加权系数3

c BOOL 修正系数

Y FLOAT 输出

**计算公式**

OUT = k1\*IN1 + k2\*IN2 + k3\*IN3 + c。

### Totalizer【积分累加块】

该功能块实现输入量的记分累积功能。

**功能块图**



9.4.15 积分累加块

**参数说明**

参数 数据类型 意义

IN FLOAT 输入信号（以小时变化率为单位）

M\_hold BOOL 保持

M\_rst BOOL 复位

OUT FLOAT 累积输出

**算法描述**

1. 当保持信号M\_hold由0→1时，输出OUT值保持不变，直至M\_hold由1→0后恢复累积操作。
2. 当复位信号M\_rst由0→1时，输出OUT被清零，直至M\_rst由1→0后从0开始累积操作。

### ABS\_F【浮点数绝对值】

**功能块图**



9.4.16 浮点数绝对值功能块

**参数说明**

参数 数据类型 意义

IN FLOAT 输入信号

OUT FLOAT 输出

**计算公式**

OUT = fabs（IN）。

### RMP【斜坡函数】

该功能块实现斜坡函数发生器功能。

**功能块图**



9.4.17 斜坡函数功能块

**参数说明**

参数 数据类型 意义

Base FLOAT 基点

End FLOAT 终点

Rate FLOAT 变化率，以秒为单位

Hold BOOL 保持

Rst BOOL 复位

Y FLOAT 斜坡发生器输出

D BOOL 发生器达到终点

**算法描述**

1. 当复位信号Rst由0→1时，输出Y = 基点值Base，D = 0；然后输出Y以速率Rate从基点趋向于终点，达到终点后D = 1，输出Y保持终点值不变。
2. 在输出Y趋向于终点值的变化过程中，若Hold由0→1，则输出Y保持不变，直到Hold由1→0，输出Y恢复以Rate速率从保持点趋向于终点。直至下一个Rst由0→1信号重新开始。

### SCALING\_X【量程变换】

该功能块实现带小信号截除合修正功能的量程变换。

**功能块图**



9.4.18 量程变换功能块

**参数说明**

参数 数据类型 意义

IN FLOAT 输入信号

Min FLOAT 变换量程下限

Max FLOAT 变换量程上限

Kc FLOAT 修正系数

Kt FLOAT 小信号截除比

OUT FLOAT 输出值

**量程变换公式**

OUT = IN \* (Max - Min) + Min + Kc；

说明：

* 1. 如果Min + (Max - Min) \* Kt > OUT，则OUT = Min；
  2. 如果OUT > Max，则OUT = Max。

### TWOSEL【二选一选择器】

该功能块实现二输入选择输出。

**功能块图**



9.4.19 二选一选择器模块

**参数说明**

参数 数据类型 意义

X1 FLOAT 输入信号1

X2 FLOAT 输入信号2

X1\_BAD BOOL 输入信号1品质坏

X2\_BAD BOOL 输入信号2品质坏

Mode INTEGER 选择方式：0=选均值，1=低选，2=高选

DB BOOL 偏差限值

Y FLOAT 选择后输出

X\_BAD BOOL 品质坏输出

Mode\_ERR BOOL 选择模式错误

**功能描述**

1、如果输入1品质坏（X1\_BAD = 1），输出Y = X2；

2、如果输入2品质坏（X2\_BAD = 1），输出Y = X1；

3、如果输入1和2品质均坏，输出Y保持，X\_BAD输出报警；

4、如果输入1和2品质均非坏，输出Y取值由Mode参数和DB参数决定；

如果二者间的偏差越限（|X2-X1| > DB），输出保持不变，且X\_BAD输出报警；

如果二者间偏差不越限，则输出由Mode决定为均值、低选或高选。

### THRSEL【三选一选择器】

该功能块实现三输入选择输出。

**功能块图**



9.4.20 三选一选择器

**参数说明**

参数 数据类型 意义

X1 FLOAT 输入信号1

X2 FLOAT 输入信号2

X3 FLOAT 输入信号3

X1\_BAD BOOL 输入信号1品质坏

X2\_BAD BOOL 输入信号2品质坏

X3\_BAD BOOL 输入信号3品质坏

Mode INTEGER 参考算法描述

DB BOOL 偏差限值

Y FLOAT 选择后输出

X\_BAD BOOL 品质坏输出

Mode\_ERR BOOL 选择模式错误

**算法描述**

1. 强制选择方式：Mode = 4，Y = X1；Mode = 5，Y = X2；Mode = 6，Y = X3；
2. 输入X1、X2、X3均为坏点的情况下，输出Y保持、X\_BAD = 1；
3. 任意两个输入为坏点，则输出等于另一个正常输入点；
4. 任意一个输入为坏点，按照下列方式输出：
   1. 如果另两个点的偏差大于DB，则输出Y保持、X\_BAD = 1；
   2. 如果另两个点的偏差不大于DB，则输出根据Mode = 0、1、2或3分别取平均值、低值、高值或平均值。
5. 如果输入X1、X2、X3均为好点，按下列方式选择：
   1. 如果两点间的偏差不越限，而另一点对这两点的偏差越限，则输出不越限两点的平均值；
   2. 如果两点间的偏常越限，而另一点对这两点的偏差不越线限，则输出取另一点值；
   3. 如果输入点间的偏差均越限，则输出保持Y不变、X\_BAD = 1；
   4. 如果输入点间的偏差均不越限，则输出根据Mode = 0、1、2或3分别取平均值、低值、高值或中值。

### LAG1【一介滞后滤波器】

该功能块实现一阶滞后滤波器功能。

该功能块包括自动模式、手动模式和自动模式等三种操作模式。

**功能块图**



9.4.21 一阶滞后滤波器

**公式**



其中：

X（old）：前一周期的输入值X；

Y（old）：前一周期的输出值Y；

dt：当前周期和前一周期的时间差。

**参数描述**

参数 数据类型 意义

MAN BOOL “1”为手动模式

HALT BOOL “1”为中断模式

X FLOAT 输入值

GAIN FLOAT 积分增益

LAG TIME 延时时间常数

YMAN FLOAT 手动控制输出值

Y FLOAT 输出值

增益GAIN和延时时间常数LAG必须被定义。其传递函数如下：



在一段延时后，输出Y跟随阶跃函数输入X（X：0→1）。值X\*GAIN可以近似表示为 exp（-t/LAG）。

**功能块模式**

模式 MAN HALT

自动模式 0 0

手动模式 1 0或1

中断模式 0 1

**自动模式**

在自动模式下，功能块操作参考上述的参数描述。

**手动模式**

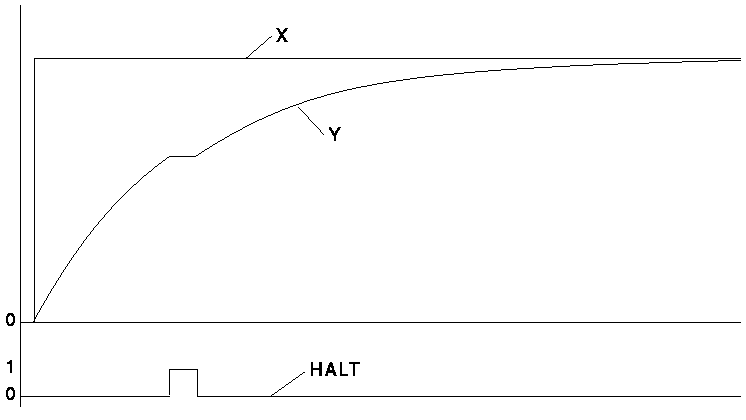
在手动模式下，手动值YMAN被直接输送到Y输出端。

**中断模式**

在中断模式下，输出Y保持最近的计算值。输出不会被程序更新，而由用户直接重写。

例如：

下图表达了LAG1的工作过程。当输出X跳变到一个新值后，输出Y则渐渐逼近新的X值。



9.4.22 一阶滞后滤波器中断模式曲线

## 扩展类（Extend）

### FLOW\_FIX【带开方及小信号切除的流量转换功能块】

该功能块实现常温液态差压型流量测量转换功能。

**功能块图**



9.5.1带开方及小信号切除的流量转换功能块

**参数描述**

参数 数据类型 意义

IN FLOAT 流量差压输入

Pmin FLOAT 差压下限

Pmax FLOAT 差压上限

Tmin FLOAT 流量下限

Tmax FLOAT 流量上限

Kt FLOAT 小流量截除比

OUT FLOAT 流量值输出

P FLOAT 流量对应差压值输出

**计算公式**

OUT = sqrt(IN \* (Pmax - Pmin) + Pmin) \* (Tmax - Tmin) / sqrt(Pmax - Pmin)

### POLYNOM【多项式功能块】

该功能块实现多项式计算功能。

**功能块图**



9.5.2 多项式功能块

**参数描述**

参数 数据类型 意义

X FLOAT 变量输入

C0~C5 FLOAT 多项式系数

Y FLOAT 多项式输出

**计算公式**

Y = C0 + C1\*X + C2\*X2 + C3\*X3 + C4\*X4 + C5\*X5

### FUN12【多折线功能块】

该功能块实现多折线分段计算功能。

**功能块图**



9.5.3 多折线功能块

**参数描述**

参数 数据类型 意义

X FLOAT 变量输入

X1、Y1 FLOAT 多折线起始点坐标

… FLOAT 多折线拐点坐标

X12、Y12 FLOAT 多折线终止点坐标

Y FLOAT 值输出

**计算公式**

Y = Y（n-1）+（X – X（n-1））/（Xn-X（n-1））\*（Yn-Y（n-1））

n = 2、3…12

### OHVFLW\_N【饱和蒸汽流量转换】

**功能块图**



9.5.4 饱和蒸汽流量转换模块

**参数描述**

参数 数据类型 意义

T FLOAT 过热蒸汽温度

P FLOAT 过热蒸汽压力

dP FLOAT 过热蒸汽流量差压信号

C FLOAT 修正系数

Fmax FLOAT 过热蒸汽额定流量

T0 FLOAT 额定工况过热蒸汽温度

P0 FLOAT 额定工况过热蒸汽压力

dPmax FLOAT 过热蒸汽流量最大差压信号

OUT FLOAT 过热蒸汽流量输出

## 控制类（Control）

### D\_SEL\_TCK【数字量跟踪】

该功能块实现数字（离散）信号检测与设定跟踪。

**功能块图**



9.6.1 数字量跟踪模块

**参数说明**

参数 数据类型 意义

G BOOL 跟踪开关

X BOOL 检测状态

MAN BOOL 设定状态

OUT BOOL 跟踪值输出

outc BOOL 跟踪值监视

**算法描述**

1. 当G = 1时，输出OUT = MAN；outc = MAN；
2. 当G = 0时，输出OUT = X；outc = X；
3. 用户输入变量与输出变量必须使用相同的变量名，才能实现用户输入跟踪功能。

### G\_SEL\_TCK【模拟量跟踪】

该功能块实现模拟信号测量与设定跟踪。

**功能块图**



9.6.2 模拟量跟踪模块

**参数说明**

参数 数据类型 意义

G BOOL 跟踪开关

X FLOAT 测量值输入

MAN FLOAT 设定值输入

OUT FLOAT 跟踪值输出

outc FLOAT 跟踪值监视

**算法描述**

1. 当G = 1时，输出OUT = MAN；outc = MAN；
2. 当G = 0时，输出OUT = X；outc = X；
3. 用户输入变量与输出变量必须使用相同的变量名，才能实现用户输入跟踪功能。

### ES\_MA\_EN【增强型模拟量操作器】

**功能块图**

、

9.6.3 增强型模拟量操作器

**参数说明**

参数 数据类型 意义

X FLOAT 输入量

OV FLOAT 超驰输入量

OS BOOL 超驰开关

MA BOOL 手自动切换

Ymax FLOAT Y上限

Ymin FLOAT Y下限

L BOOL 屏蔽操作(用于检修等状态)

YB FLOAT 偏置输入

Yman FLOAT 手动值给定

YURate FLOAT 增指令输入限速（手动状态有效）

YDRate FLOAT 减指令输入限速（手动状态有效）

Ex1 FLAOT 特定值输入1

Ex2 FLAOT 特定值输入2

Ex3 FLOAT 特定值输入3

Y FLAOT MA站输出，带上下限

y\_man FLOAT 指令跟踪输出端

MSAM FLOAT 状态输出：1：自动，0：手动

YBo FLOAT 偏置输出

ybo\_out FLOAT 当前偏置

Lock BOOL 检修状态

RateErr BOOL 限幅输入错误

**算法描述**

1. 超驰状态下（OS = 1），输出Y = OV，Yman指令通过y\_man外部返回跟踪；
2. 手动模式下，手动给定指令（Yman）受限幅（Ymax、Ymin）和限速（YURate、YDRate）输入限制，每次手动给定指令的变化幅度不应超过限速（YURate、YDRate），手动给定指令的范围不应超出限幅（Ymax、Ymin），输入越限将返回并保持上一次指令；
3. 检修状态下（LOCK = 1），该功能块操作被挂起；
4. 特定值输入在手动状态下优先，不受限速（YURate、YDRate）输入的限制。

### SS\_MA\_EN【标准模拟量操作器】

**功能块图**



9.6.4 标准模拟量操作器

**参数说明**

参数 数据类型 意义

X FLOAT 输入量

OV FLOAT 超驰输入量

OS BOOL 超驰开关

MA BOOL 手自动切换

Ymax FLOAT Y上限

Ymin FLOAT Y下限

L BOOL 屏蔽操作(用于检修等状态)

Yman FLOAT 手动值给定

OUT FLAOT MA站输出，带上下限

MSOUT FLOAT 指令跟踪输出端

MSAM FLOAT 状态输出：1：自动，0：手动

Lock BOOL 检修状态

RateErr BOOL 限幅输入错误

**算法描述**

1. 超驰状态下（OS = 1），输出Y = OV，Yman指令通过y\_man外部返回跟踪；
2. 手动模式下，手动给定指令（Yman）受限幅（Ymax、Ymin）输入限制，输入越限将返回并保持上一次指令；
3. 检修状态下（LOCK = 1），该功能块操作被挂起。

### SFT【无扰切换模块】

**功能块图**



9.6.5 无扰切换模块

**参数说明**

参数 数据类型 意义

X1 FLOAT 输入量1

X2 FLOAT 输入量2

Z BOOL 切换开关

DY1 BOOL 由X2向X1切换时的变化率（以每分钟的变化量）

DY2 FLOAT 由X1向X2切换时的变化率（以每分钟的变化量）

Y FLOAT 无扰切换输出

### MS\_ID【脉冲型模拟输出操作器】

**功能块图**



9.6.6 脉冲型模拟输出操作器

**参数说明**

参数 数据类型 意义

IN FLOAT 自动模式下输入信号

IC BOOL 手动增指令脉冲

DC BOOL 手动减指令脉冲

SW BOOL 手/自动方式切换

RATE FLOAT 输出指令Y变化率限制

YMAX FLOAT 输出指令上限

YMIN FLOAT 输出指令下限

YMAN FLOAT 手动给定/跟踪输入

Y FLOAT 输出指令

Yout FLOAT 输出指令跟踪

QMAX BOOL 输出指令达上限

QMIN BOOL 输出指令达下限

### FIRST【首出控制模块】

该功能块实现多路开关输入的首出记忆功能。

**功能块图**



9.6.7 首出控制模块

**参数说明**

参数 数据类型 意义

IN1 FLOAT 输入值1

Z1~Z16 BOOL #1至#16开关输入

Num INTEGER 当输入开关中个数大于等于该值，则D1置1

R BOOL 当输入无1时允许复位

Y INTEGER 第1个由0 → 1变化的输入端序号

Yp WORD 输入信号打包输出

D BOOL 首出指示

D1 BOOL Z1 +···+ Z16输出

**算法描述**

该算法从16个输入中计算出第一个从0变为1的开关量的序号，输入信号的优先级为前高后低。如果输入中有≥Num个1时，输出D1（n）=1，否则D1（n）=0。

### MOV\_B【开关信号传递】

该功能块实现BOOL型变量或值的传递。

功能块图



9.6.8 开关信号传递

**参数说明**

参数 数据类型 意义

IN BOOL 输入变量或值

OUT BOOL 输出变量或值

**算法描述**

OUT = IN。

### MOV\_F【模拟信号传递】

该功能块实现浮点数型（FLOAT）变量或值的传递。

**功能块图**



9.6.9 模拟信号传递

**参数说明**

参数 数据类型 意义

IN FLOAT 输入变量或值

OUT FLOAT 输出变量或值

**算法描述**

OUT = IN。

### MODE\_SEL【模式选择】

该功能块提供几种程序模式之间的切换控制。

**功能块图**



9.6.10 模式选择

**参数说明**

参数 数据类型 意义

Auto BOOL “自动模式”选择指令

Prog BOOL “程控模式”选择指令

Man BOOL “手动模式”选择指令

Auto\_S BOOL “自动模式”状态

Prog\_S BOOL “程控模式”状态

Man\_S BOOL “手动模式”状态

**功能描述（真值表）**



附录一 NicC语言语法

用BNF格式列出的NicC语言语法，其中implementation部分的语法同ansi-C的语法一致，在此不单独列出。

program -> TOK\_FBLOCK id ’；’ blk\_attributes uuid comments interface implementation

；

blk\_attributes ->

| TOK\_ATTRIBUTE ’(’ blk\_attribute\_list ’)’ ’；’

；

blk\_attribute\_list -> blk\_attribute

| blk\_attribute\_list ’,’ blk\_attribute

；

blk\_attribute -> id

；

uuid ->

| uuid\_const ’；’

；

uuid\_const -> TOK\_UUID

；

comments ->

| TOK\_COMMENTS comment\_list

；

comment\_list ->

| comment\_list comment

；

comment -> id ’=’ string ’；’

；

string -> TOK\_STRING

| string TOK\_STRING

；

interface ->

| TOK\_VAR var\_list

；

var\_list ->

| var\_list var\_def

；

var\_def -> id ’：’ type\_def ’；’ attribute\_part

；

type\_def -> TOK\_EVENT

| TOK\_INTEGER

| TOK\_FLOAT

| TOK\_BOOL

| TOK\_DATE

| TOK\_BYTE

| TOK\_WORD

| TOK\_DOUBLE

| TOK\_CHAR

| TOK\_SHORT

| TOK\_DWORD

；

attribute\_part ->

| attribute\_list ’；’

；

attribute\_list -> attribute

| attribute\_list ’,’ attribute

；

attribute -> TOK\_IN

| TOK\_OUT

| TOK\_TIMER ’(’ integer\_number ’)’

| TOK\_HIDDEN

；

integer\_number -> TOK\_NUMBER

| TOK\_HEXNUMBER

；

id -> TOK\_ID

；

implementation -> TOK\_IMPLEMENTATION

**以下为空白**