

**AdvanTrol-ProV2.70**

# **SCX 语言使用手册**

# 目 录

SCX 语言 .....	1
1 概述 .....	1
1.1 功能特点 .....	1
1.2 SCX 语言程序开发流程及注意事项 .....	1
2 软件开发环境 .....	3
2.1 SCX 语言软件安装 .....	3
2.2 启动 SCX 语言软件开发环境 .....	3
2.2.1 SCX 语言程序的开发步骤 .....	3
2.3 SCX 语言软件窗口 .....	6
2.3.1 标题栏 .....	6
2.3.2 菜单栏 .....	6
2.3.3 工具栏 .....	6
2.3.4 状态栏 .....	7
2.3.5 源程序编译区 .....	7
2.3.6 编译信息显示窗口 .....	8
2.3.7 错误处理 .....	8
2.4 程序编译原理 .....	8
3 菜单命令和功能图标 .....	9
3.1 菜单命令 .....	9
3.1.1 文件菜单 .....	9
3.1.2 编辑菜单 .....	11
3.1.3 查看菜单 .....	12
3.1.4 编译菜单 .....	13
3.2 选项菜单 .....	13
3.2.1 窗口菜单 .....	14
3.2.2 帮助菜单 .....	15
3.3 功能图标 .....	15
4 编程规则 .....	16
4.1 SCX 语言特点 .....	16
4.2 代码说明 .....	16
4.3 程序生成步骤 .....	16
4.4 程序结构 .....	17
4.5 数据类型 .....	17
4.6 常数表示 .....	18
4.7 系统变量 .....	19

4.8 位号表示.....	19
4.9 标识符的定义规则.....	20
4.10 运算符.....	20
4.10.1 运算符优先级.....	20
4.11 运算表达式.....	21
4.12 语法结构.....	21
4.12.1 全局定义程序.....	21
4.12.2 赋值语句.....	22
4.12.3 条件语句.....	22
4.12.4 循环语句.....	24
4.13 函数和子程序.....	25
4.14 折线表.....	26
4.15 位号成员和控制模块的引用.....	26
4.15.1 位号.....	27
4.15.2 基本控制功能模块（BSC）.....	29
4.15.3 串级控制功能模块（CSC）.....	31
4.16 库函数.....	34
4.17 注释.....	36
4.18 关键字.....	36
5 编程实例.....	37
5.1 浮点实例.....	37
5.2 函数和折线表实例.....	37
5.3 DGAP 模块实例.....	38
5.4 PID(比例积分微分)控制.....	39
5.5 联锁保护.....	41
5.6 网关卡与其它智能设备互联通信程序.....	42
5.6.1 Modbus-RTU 协议.....	43
5.6.2 HostLink-ASCII.....	47
5.6.3 自定义：用户通信协议开放（波特率 19200bps）.....	57
6 库函数.....	83
6.1 库函数目录.....	83
6.2 库函数介绍.....	86
6.2.1 半浮点计算模块.....	86
6.2.2 浮点计算模块.....	89
6.2.3 辅助计算模块.....	95
6.2.4 混合计算模块.....	106
6.2.5 控制模块.....	108

6.2.6 类型转换模块 .....	117
6.2.7 常用语句 .....	120
7 资料版本说明 .....	124

# SCX 语言



型号为 FW243X、XP243X、FW247 的控制器不支持 SCX 语言。

## 1 概述

### 1.1 功能特点

SCX 语言（软件名为 SCLang.exe）编程软件是 SUPCON WebField 系列控制系统控制站的专用编程语言。在工程师站完成 SCX 程序的调试编辑，并通过工程师站将编译后的可执行代码下载到控制站执行。SCX 语言属高级语言，语法风格类似标准 C 语言，除了提供类似 C 语言的基本元素、表达式等外，还在控制功能实现方面作了大量扩充。

SCX 语言编辑环境符合 Windows 环境下编辑器的设计准则。灵活易用、功能完善的在线帮助系统使得 SCX 程序的调试、编译得心应手。

SCX 语言编程软件具有以下特点：

- 提供 SCX 语言编程环境：SCX 语言软件是一个运行在中文 Windows NT/2000 操作系统下的应用软件，有良好的用户界面，用户可以非常方便地在 SCX 语言软件中编写程序，检查语法错误。SCX 语言软件和 SUPCON WebField 控制系统的其它软件紧密集成，可以和组态软件交换信息。
- 功能强大：除了提供 C 语言的基本元素，如表达式、选择语句、循环语句、多维数组、结构类型外，还提供丰富的函数库、专门的控制功能模块、位号数据类型等。
- 稳定性高：SCX 语言软件及其内置的 XAC 编译器所进行的双重词法和语法检查保证了 SCX 语言软件程序编译执行的高稳定性和高可靠性。
- 实时性强，可靠性高：控制站每个周期执行一次 SCX 语言程序，并且监视用户编制的程序实时运行状况，对于程序中的超时、死循环会自动进行报警。
- 易于使用：软件提供了灵活易用的集成化开发环境——符合 Windows 环境编辑器设计准则的程序编辑环境、功能完善的在线帮助系统、程序编译中的诊错定位功能、语法着色等。这些都给 SCX 程序的编辑、编译和调试提供了极大的方便。
- 易于维护：软件提供了中间 C 语言代码查看功能，可以看到 SCX 语言程序对应的 C 语言表达，可以有效排除程序中可能存在的“Bug”。

### 1.2 SCX 语言程序开发流程及注意事项

SCX 语言程序在 SUPCON WebField 系列控制系统工程师站的开发流程：

- 1、 在控制组态自定义控制方案对话框中，直接激活 SCX 语言软件编程环境，按 SCX 语言规

定的语法规则编写程序，编写完毕保存为扩展名为“.SCL”的文档文件；

- 2、 调用软件集成环境中的编译功能菜单，将编写的 SCL 程序文件编译生成目标代码文件（扩展名为“.OBJ”），如果编译有错误，将会在错误信息栏中显示相关信息，以方便改错；
- 3、 组态软件把生成的目标代码与系统中其他组态信息一同封装，供监控软件调用，监控软件根据组态文件中关于 SCX 语言程序的信息，将程序目标代码下载到指定控制站的指定地址空间内；
- 4、 下载到各控制站的程序由控制站主控制卡的调度程序负责调度运行，从而实现所设计的控制算法。

SCX 语言软件 SCLang.exe 运行注意事项：

- 应用程序 SCLang.exe 最多支持三层目录，每层目录名建议不要超过 8 个英文字符；
- 目录 kernel 必须与应用程序 SCLang.exe 在同一层目录下；
- 待编译的 SCX 程序文件的文件名建议不要超过 8 个英文字符。
- 在程序编译过程中如果编译进度条出现长时间没有响应或给出“内核程序调用失败”的信息时，应仔细检查 SCX 语言程序运行环境是否满足上述要求，如不满足应修改并重新启动计算机。

## 2 软件开发环境

SCX 语言软件开发环境是编辑和编译 SCX 语言程序的集成工作环境。用户可以在 SCX 语言软件的编辑环境中完成程序的编写、编辑，并保存成程序文件存档。通过编译系统可以编译、检查程序中的错误，观察生成代码的部分信息，并最终生成可执行的目标代码。

SCX 语言软件开发环境包括七大块：灵活的程序编辑环境、完善的文件处理功能、准确的错误定位诊断处理、可靠的中间代码检查、强大的编译功能、灵活便捷的在线帮助系统、友好的语法着色。

### 2.1 SCX 语言软件安装

- 1、 SCX语言软件是软件包AdvanTrol-Pro的组成部分，随软件包一同安装（安装步骤请参考相应安装手册）。

- 2、 鼠标的基本操作

鼠标是中文 Windows 操作系统和 AdvanTrol-Pro 的主要操作工具。鼠标有三个按键，本软件采用标准二键式，只有左键和右键有效。操作中将会遇到的鼠标标准操作有：

- 移动：在鼠标板上移动鼠标，不按任何键。
- 单击：按下鼠标某一键又立即弹起的过程，其间鼠标没有发生移动。以下若不特别说明，一般指对鼠标左键的操作。
- 选中：单击鼠标左键。
- 双击：连续而均匀的单击两次鼠标某一键。以下若不特别说明，一般指对鼠标左键的操作。



以上描述在后面的章节中会频繁出现，需要用户牢记。

### 2.2 启动 SCX 语言软件开发环境

为了保证 SCX 语言软件编译器能正确识别用户程序中所引用的位号，SCX 语言软件开发环境必须从组态软件中启动。同时，SCX 语言软件编译生成的目标代码也必须由组态软件调用并生成可下装的目标代码；单独启动 SCX 语言软件，将不能正常运行。



本节讲述 SCX 语言的编程环境，如果对 windows 系统的操作十分熟练，可以略过不看。

#### 2.2.1 SCX 语言程序的开发步骤

- 1、 创建程序文件
  - 启动组态软件，打开需要开发自定义语言算法的工程应用。单击控制站 \自定义控制方案，出现自定义控制算法设置对话框，如图 2-1 所示。

- 在自定义控制算法设置对话框中，选中某一具体控制站，并在 SCX 语言编程输入框中输入待编辑的程序文件名，单击编辑按钮，将弹出 SCX 语言软件的编辑环境，如图 2-2 所示。
- 在 SCX 语言软件的编辑环境中就可以输入程序代码了。程序代码输入完毕后，应保存成程序文件。

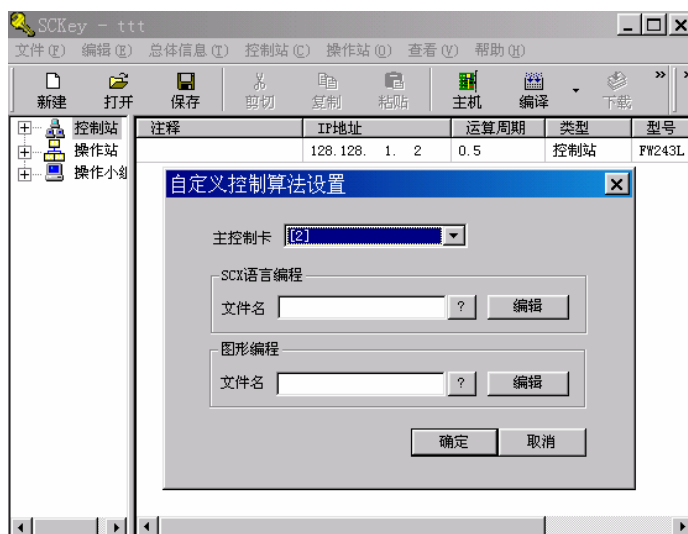


图 2-1 SCX 语言组态对话框

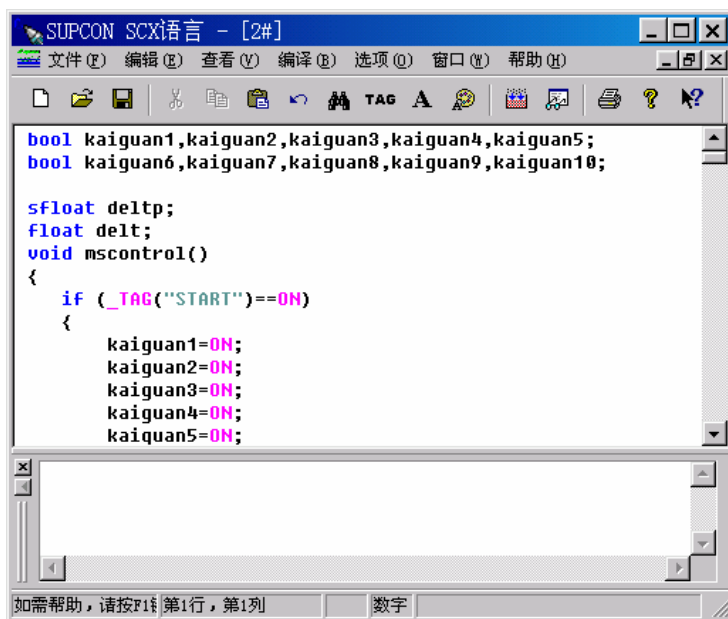


图 2-2 SCX 语言软件编辑环境

## 2、生成中间代码

单击编译\生成目标代码，将首先生成与工程 C 语言兼容的中间代码。中间代码文件的扩展名是“.C”，可以直接由 SCX 语言软件从文件菜单打开并查看生成的代码，如图 2-3 所示。

### 生成目标代码

当执行编译菜单的生成目标代码命令时，如果 SCX 语言程序正确，将生成可由控制站主控制卡



调度执行的目标代码。目标代码文件的扩展名是“.OBJ”，可以直接由 SCX 语言软件从查看菜单的目标代码查看命令查看，如图 2-4 所示。

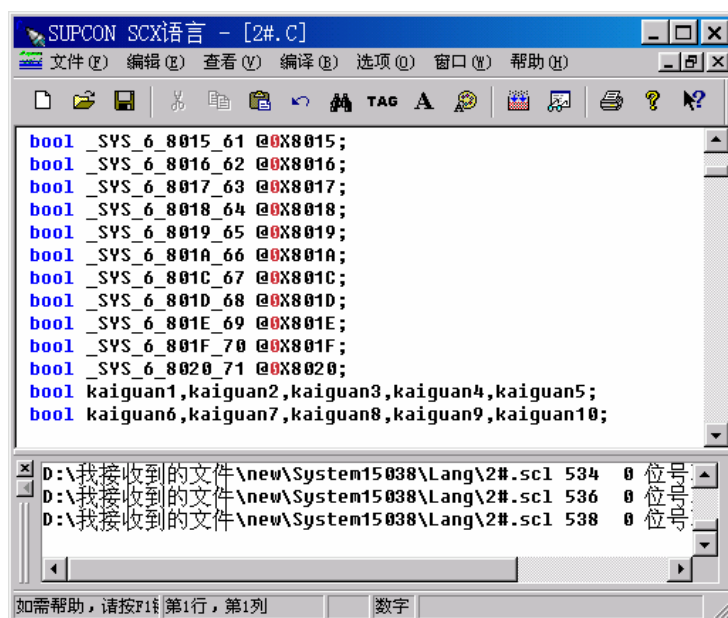


图 2-3 中间代码文件

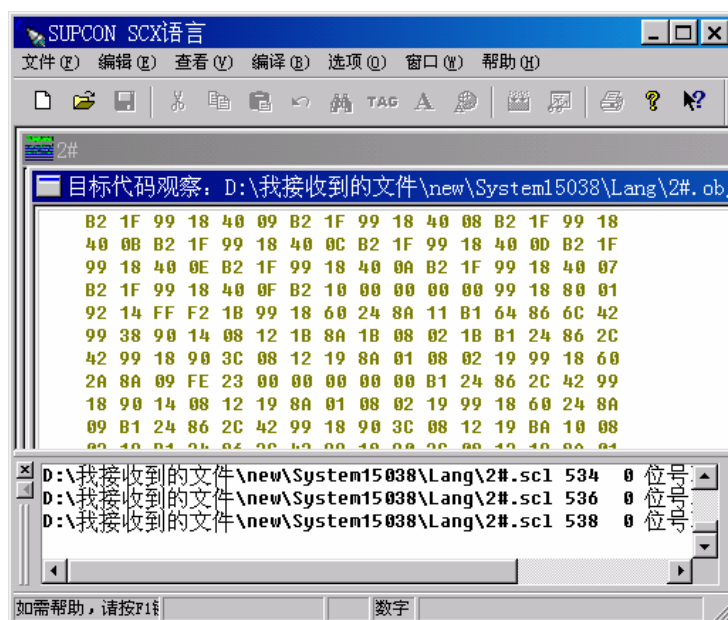


图 2-4 目标代码观察

### 3、下装并测试目标代码

在目标代码正确生成后，可以在组态软件内再次编译，和其它组态信息一起生成可下装的代码。该代码文件可以由组态软件下装到控制站的主控制卡并被调度运行，从而实现了指定的自定义控制算法。如何实现总体编译请参考组态软件的帮助，如何实现下装请参考组态软件的帮助。

通过监控软件的监控功能，可以测试自定义控制算法的正确性和有效性。

## 2.3 SCX 语言软件窗口

SCX 语言软件窗口按 Windows 标准编辑器设计，主要用于编辑和显示 SCX 语言代码文件及其编译信息。

窗口分标题栏、菜单栏、工具栏、状态栏、源程序编辑区、编译信息显示窗口、状态栏、光标位置指示及编译进度栏，如图 2-5 所示：

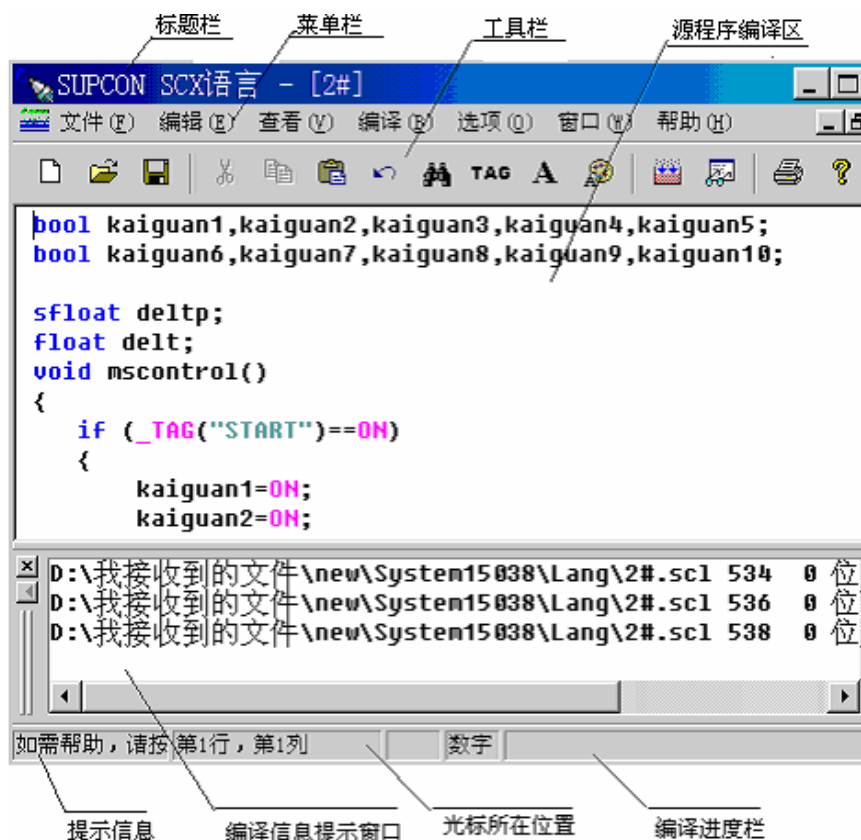


图 2-5 SCX 语言软件编辑窗口

### 2.3.1 标题栏

标题栏显示当前进行编辑的 SCX 语言程序文件名，并包含通常 Windows 应用软件的系统菜单及按钮。

### 2.3.2 菜单栏

菜单栏提供 SCX 语言软件编辑过程需要用到的一些菜单命令，分为文件菜单，编辑菜单，查看菜单，编译菜单，选项菜单，窗口菜单，帮助菜单等。具体各个菜单下的命令及功能详见第 3 章。

### 2.3.3 工具栏

工具栏显示于菜单栏的下面。工具栏所画的一些图标是菜单栏当中的部分比较常用的菜单命令，如新建文件，打开文件，保存文件等。这样使用者就可以很方便的使用部分常用命令。

### 2.3.4 状态栏

状态栏显示于 SCX 语言软件窗口的底部。要想显示或隐藏状态栏，可用选项菜单中的状态栏命令实现。

- 状态栏的左边区域描述了用鼠标掠过菜单时菜单项目的提示信息。同样地，在用鼠标按下工具栏上的按钮而没有释放前，这个区域仍显示工具栏的操作信息。在看了对工具栏按钮的操作信息后如果不希望执行此命令，可以在鼠标光标离开工具栏按钮后放开鼠标按钮。
- 状态栏的中间区域描述了在编辑环境中编辑 SCX 语言文档时光标位置指示，并指明下述哪些键被锁住，见下表所示。

表 2-1 按键锁定指示

指示器	描 述
大写	大写锁定键被锁住。
数字	数字锁定键被锁住。

- 状态栏的右边区域是编译进度指示。编译时，该区域显示编译进度信息。

### 2.3.5 源程序编译区

SCX 语言软件编辑环境是一个典型的多文档编辑环境，其特点：

#### 1、 灵活的编辑环境

通过编辑菜单、选项菜单的相关命令，可在编辑主窗口中进行 SCX 语言程序代码的文本编辑。在编辑区域编辑源程序时，状态栏的中间区域将时刻提示当前光标所在位置。编辑菜单、选项菜单的相关命令另见第三章。

在编辑窗口中单击鼠标右键时将弹出一个快捷菜单。它可以使得在编辑窗口中的操作变得灵活方便。弹出式菜单命令：

撤消——撤消上次的编辑操作。

剪切——从 SCX 语言文档中删除选定的内容并将其保留在剪贴板上。

复制——将 SCX 语言文档中已选定的数据复制到剪贴板上。

粘贴——将剪贴板的数据粘贴到文档中。

删除——从 SCX 语言文档中删除选定的内容。

全选——把编辑窗口中的所有内容全部选定。

#### 2、 完善的文件功能

通过文件菜单的命令，可完成 SCX 语言程序文档的文件功能，包括新建、打开、关闭、保存、打印等。

#### 3、 详实的帮助系统

在编写程序的过程中，如果想得到一个函数或关键字的帮助，可以在源程序输入窗口内输入该函数或关键字，然后将光标移动到该位置，按 F1 键即可获得联机帮助信息。

#### 4、 友好的语法着色

作为完善的高级语言软件，SCX 语言开发环境提供了友好的语法着色功能，以方便用户的使用。

语法着色将源代码按关键字、常数、注释、变量等语法成分按不同的颜色显示。

### 2.3.6 编译信息显示窗口

SCX 语言软件开发环境的编译信息输出窗口将显示编译过程产生的包括错误内容在内的所有信息。

在编译窗口中单击鼠标右键时将弹出一个快捷菜单。它可以使在编译窗口中的操作变得灵活方便。弹出式菜单中可用的命令：

- 复制——从 SCX 语言文档中将选定的数据复制到剪贴板上。
- 选择全部——把编译窗口中的所有内容全部选定。
- 隐藏——取消编译窗口。

### 2.3.7 错误处理

在错误输出栏内，直接双击错误信息，可以使编辑窗口中的光标自动移动到相应的源程序错误行。

不需要错误信息栏时，可以在该区域单击鼠标右键，通过弹出式菜单中的隐藏命令直接隐藏该窗口。

想保存错误信息时，请在错误信息窗口内单击鼠标右键，通过弹出式菜单中的选择全部命令选择所有内容并拷贝到任何可用的编辑器中并保存，当然也可以利用 SCX 软件的编辑环境。

## 2.4 程序编译原理

编译已编辑好的 SCX 语言程序文件，首先将生成扩展名为“.C”的中间代码。该中间代码文件可以直接由 SCX 语言软件从文件菜单打开并查看。如果 SCX 语言程序正确，接着将生成由控制站主控制卡调度执行的目标代码。目标代码文件（扩展名是“.OBJ”）可以直接由 SCX 语言软件从查看菜单的目标代码统计命令查看。

在目标代码正确生成后，还须在组态软件内和其它组态信息一起进行统一编译，生成的代码文件由组态软件下装到控制站运行，从而实现了指定的自定义控制算法。通过监控软件的运行，可以测试自定义语言算法的正确性和有效性。

## 3 菜单命令和功能图标

### 3.1 菜单命令

#### 3.1.1 文件菜单

文件菜单提供了以下命令，见下表。

表 3-1 文件菜单命令

命令	功 能 简 介	工具栏图标	快捷键
新建	建立一个新的 SCX 语言文件		Ctrl+N
打开	打开一个现存的 SCX 语言文件		Ctrl+O
关闭	关闭一个打开的 SCX 语言文件		
保存	用同样的文件名保存一个打开的 SCX 语言文件		Ctrl+S
另存为	用指定的文件名保存一个打开的 SCX 语言文件		
页面设置	设置打印时的页边距		
打印	打印当前 SCX 语言文件		Ctrl+P
打印预览	在屏幕上按被打印出的格式显示 SCX 语言文件		
打印设置	选择一个打印机以及打印机连接		
退出	退出 SCX 语言软件		Alt+F4

#### 1、 新建命令

建立一个新的 SCX 语言文件。须注意使用新建命令建立的文件没有组态信息。要使用组态信息，请从组态软件启动 SCX 语言编辑环境建立新文件。

#### 2、 打开命令

打开一个现存的 SCX 语言文件。须注意使用打开命令打开的文件没有组态信息。要使用组态信息，请从组态软件启动 SCX 语言编辑环境打开文件。

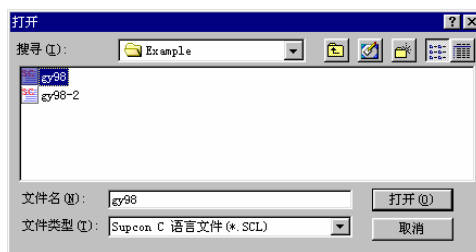



图 3-1 打开对话框

在打开对话框中，单击将打开的文件，然后单击打开按钮，即可打开该文件。

#### 3、 关闭命令

关闭当前正在操作的文档窗口。SCX 语言软件会建议在关闭该文档之前保存对文档所做的改动。如果没有保存而关闭了一个文档，将会失去自最后一次保存以来所做的所有改动。在关闭一个无标题的文档之前，SCX 语言软件会显示保存为对话框，建议命名和保存文档。也可以单击文档窗口上

的关闭按钮  来关闭文档。

#### 4、 保存命令

如果当前文件已存在，则用同样的文件名保存。如果该文件是新文件，则用法和另存为命令一样。参见另存为命令。

#### 5、 另存为命令

用指定的文件名保存一个打开的 SCX 语言文件。选中另存为命令后，弹出如图 3-2 所示的对话框：

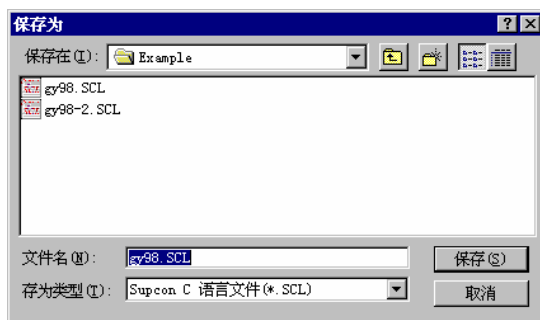


图 3-2 另存为对话框

除了保存按钮外，该对话框的各部件的用途与打开对话框的相似。选择好文件名后、或在文件名框内键入文件名后，单击保存按钮就以所选文件名保存该文件，如果该文件已存在，SCX 语言软件会提示该操作会覆盖原文件，可以决定是否取消此次存盘操作。

#### 6、 页面设置命令

设置打印时的页边距。选中页面设置命令后，弹出如图 3-3 所示的对话框：



图 3-3 页面设置对话框


修改好上、下、左、右的页边距后，单击确认按钮确认该操作，或单击取消按钮取消本次操作。

#### 7、 打印命令

将当前正在操作的 SCX 语言文件打印出来。选中打印命令后，显示如图 3-4 所示的对话框：



图 3-4 打印对话框

在打印对话框中，可以指明要打印的页码起止范围、份数、打印机名称，以及其它打印机设置选项。设置好各项参数后，单击确定按钮将文件送到指定的打印机，或单击取消按钮取消本次打印操作。应该注意的是，如果单击工具栏上的打印按钮，则不显示上述对话框，而将该文件按打印机当前设置的参数值全部直接打印出来。

#### 8、 打印预览命令

预先观察将当前文件打印到打印机的实际效果。


#### 9、 打印设置命令

设置打印机参数。选中打印设置命令后，显示如图 3-5 所示的对话框。



图 3-5 打印设置对话框

#### 10、 退出命令

选中退出命令或单击应用程序控制菜单上的关闭按钮退出 SCX 语言环境。在退出 SCX 语言环境之前，SCX 语言软件会提示保存尚未保存过的文档。

双击应用程序的控制菜单按钮也可以退出 SCX 语言环境。如图 3-6 所示：

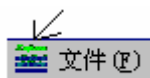







图 3-6 控制菜单按钮

### 3.1.2 编辑菜单

编辑菜单提供了以下命令，见表 3-2。

表 3-2 编辑菜单命令

命 令	功 能 简 介	工具栏图标
撤消	撤消上一步编辑操作	
剪切	从文档中删除数据并将其移到剪贴板上	
复制	从文档中将数据复制到剪贴板上	
粘贴	从剪贴板上将数据粘贴到文档中	
查找	在编辑区查找指定的字符串	
替换	用字符串替换指定的字符串	
到指定行	把光标移动到指定行的首字符	
选择全部	把编辑框内所有内容选定	

#### 1、 撤消命令

撤消上一步编辑操作。该命令能否起作用将取决于上一步所执行的操作。如果无法撤消上一步操作，菜单上的撤消命令变成灰色，表示禁止使用状态。

#### 2、 剪切命令

将当前被选取的数据从文档中删除并放置于剪贴板上。如当前没有数据被选取时，则此命令不可用。

#### 3、 复制命令

将被选取的数据复制到剪切板上。如当前无数据被选取时，则此命令不可用。

#### 4、 粘贴命令

将剪贴板中的内容复制到文档的指定插入点处。如果剪贴板是空的，菜单上的粘贴命令变成灰色，呈禁止使用状态。

#### 5、 查找命令

查找当前编辑文档中指定的字符串。从当前光标位置起，如果查找到该字符串首次出现的位置，光标将移动到查找到的位置。用 CTRL+F 快捷键可重复查找，直至查找结束。

#### 6、 替换...命令

用指定的字符串替换文档中搜索到的另一字符串。

#### 7、 到指定行...命令

将屏幕滚动到指定行，光标也移动到该行的首字符处。

#### 8、 选择全部命令


将当前活动文档中所有文本选定。

### 3.1.3 查看菜单

查看菜单提供了以下命令，见表 3-3。



表 3-3 查看菜单命令

命 令	功 能 简 介	工具栏图标
源码统计信息	统计 SCX 语言文档的有关信息。	
目标代码查看	统计 SCX 语言文档编译后生成的目标代码的有关信息。	
全 屏	按全屏方式查看编辑区	

#### 1、 源码统计信息命令

查看当前激活的 SCX 语言文档的有关信息，包括文件创建时间、修改时间、当前访问时间、文件大小等。

如果当前文档为空，没有存盘或没有被激活，该命令将显示空文件，提示存盘或呈禁止状态。若当前文档符合查看要求，则 SCX 语言软件将弹出一对话框显示相应的内容。

#### 2、 目标代码统计命令

以文本方式查看当前激活的 SCX 语言文档编译生成的目标代码统计信息和目标代码。

如果当前文档为空，没有存盘或没有被激活，该命令将显示空文件，提示存盘或呈禁止状态。若当前文档符合查看要求，则 SCX 语言软件将创建一新子窗口显示相应的内容。


#### 3、 全屏命令

按全屏方式显示编辑区域，可以扩大编辑视野。当不需要全屏显示的时候，可以按 ESC 键直接恢复到原来的显示状态。全屏后“全屏”显示命令将自动改变为“恢复”显示命令。按“恢复”显示命令将恢复原来的显示。

### 3.1.4 编译菜单

编译菜单提供了以下命令，见表 3-4。

表 3-4 编译菜单命令

命令	功能简介	工具栏图标
生成目标代码	编译当前 SCX 语言文件，生成可下载的目标代码	

#### 1、 生成目标代码命令



编译当前激活的 SCX 语言文档并生成可下装的目标代码。组态软件将把生成的目标代码和其它目标代码封装成监控软件可下装到控制站主控制卡的可执行代码。

如果编译有错误，编译完成后，可以在编译信息框里看到错误信息，每个错误占一行，指明错误所在行以及原因。如果编译有致命的错误，编译将不产生目标代码文件，原有的目标代码文件会被删除。如果编译通过，还需在组态软件进行总体编译时编译该源程序文件。

## 3.2 选项菜单

选项菜单提供了以下命令，见表 3-5。

表 3-5 选项菜单命令

命令	功能简介	工具栏图标
工具栏	显示或隐藏工具栏	
状态栏	显示或隐藏状态栏	
输出栏	显示错误信息输出窗口	
设置	设定 SCX 语言编辑环境	
改变字体	改变 SCX 语言编辑窗口的字体	
改变颜色	改变 SCX 语言编辑窗口的背景颜色	

#### 1、 工具栏命令

显示和隐藏工具栏。工具栏包括了 SCX 语言软件中一些较常用命令的按钮，如文件打开。在工具栏被显示时，一个打勾记号出现在该命令的左边。

#### 2、 状态栏命令

显示和隐藏状态栏。状态栏描述了被选取的命令或被按下的工具栏按钮，以及键盘的锁定状态将要执行的操作。当状态栏被显示时，在该命令的左边会出现一个打勾记号。

#### 3、 错误输出栏命令

激活 SCX 语言软件开发环境的编译信息输出窗口。在该窗口将显示编译过程产生的包括错误内容在内的所有信息。

#### 4、 设置命令

设置编辑环境的制表长度和位号输入对话框是否显示等。该设置将被登记到注册表中使得环境保持该设置。

#### 5、 改变字体命令

设置编辑字体。

#### 6、 改变颜色命令

设置编辑环境背景颜色。该设置将被登记到注册表中使得环境保持该设置。

### 3.2.1 窗口菜单

窗口菜单提供了以下命令。这些命令能在应用程序窗口中安排多个文档的多个视图，见表 3-6。

表 3-6 窗口菜单命令

命令	功能简介
层叠	按重叠方式安排窗口
平铺	按互不重叠平铺方式安排窗口
排列图标	安排已最小化窗口的图标
窗口 1, 2, ...	转到指定的窗口

#### 1、 层叠命令

按相互重叠形式来安排多个打开的窗口。

#### 2、 平铺命令

按互不重叠形式来安排多个打开的窗口。


#### 3、 排列图标命令

在主窗口的底部安排被最小化的窗口的图标。如果在主窗口的底部有一个打开的窗口，则有可能会看不见某些或全部图标，因为它们在这个文档窗口的下面。

### 3.2.2 帮助菜单

帮助菜单提供使用这个应用程序的帮助，并提供以下的操作命令，见表 3-7。

表 3-7 帮助菜单命令

命令	功能简介	工具栏菜单
今日技巧	可以得到一些使用该软件的技巧	
SCX 语言帮助索引	提供了帮助的主题索引	
联机帮助系统	提供了 SCX 语言软件的帮助	
如何使用帮助	教您如何使用 SCX 语言软件的帮助系统	
关于 SCLang.exe...	显示这个应用程序的版权、版本号等信息	

#### 1、 今日技巧命令

用此命令可以查看 SCX 语言软件使用过程中的各种技巧，这些技巧将帮助您更有效更快地完成工作。

#### 2、 SCX 语言帮助索引命令

此命令将弹出帮助窗口。通过帮助窗口，可跳到关于使用 SCX 语言软件各类指令的帮助信息以及各种不同类型参考资料，或通过将要查询的信息输入到索引框内，即可获得在线帮助。

一旦获得了帮助信息，在任何时候如果想要回到初始状态，都可以通过单击目录或索引按钮来实现。

#### 3、 联机帮助系统命令

直接进入 SCX 语言软件的联机帮助系统，在联机帮助系统里可以直接查看所需的帮助信息。


#### 4、 如何使用帮助命令

得到关于如何使用帮助的命令。


#### 5、 关于命令

显示 SCX 语言软件的版权通告和版本号码等信息。

## 3.3 功能图标

图标：

单击该按钮，可以在编辑区光标所在位置根据用户的选择自动插入位号。

图标：

单击该按钮，可以对菜单栏各命令、工具栏各按钮、编辑区提供在线帮助。

## 4 编程规则

### 4.1 SCX 语言特点

➤ 开放资源多

用户编写程序时，可以引用 SUPCON WebField 控制系统的各类仪表信息、测量值、输出值，并且可以改变允许操作的参数。

➤ 可使用多种数据类型

有字节型、整数型、长整数型、浮点型和半浮点型五种数据类型。

➤ 是一种专用于编写控制算法的高级语言

有函数的概念，提供复杂表达式计算、条件判断、循环等语句。在程序中可以使用在组态时定义的各种位号进行计算和操作。

提供了二维折线表和二维折线表，并提供对折线表进行操作的函数。

提供面向控制工程的模块，如单回路 PID 模块、串级 PID 模块，减少了工程师的工作量。

提供常用的库函数，如平方、开方、绝对值、比率限制、高选、低选和折线表计算等函数，方便用户编程。

➤ 实时运行

通过 SCX 语言编程实现的控制算法编译后生成目标文件和其它组态信息联编后下装到控制站的主控卡，每个控制周期主控卡执行一次该目标文件。

### 4.2 代码说明

SCX 语言软件涉及的代码有如表 4-1 所示几种类型：

表 4-1 SCX 语言软件涉及的代码

名称	简名	扩展名	属性	举例
源代码	SCX	SCL	文本	Test.SCL
中间代码	C	C	文本	Test.C
目标代码	OBJ	OBJ	二进制	Test.OBJ
HEX 代码	HEX	HEX	二进制	Test.Hex
位号文件	TAG	TAG	文本	~tmp.tag

### 4.3 程序生成步骤

SCX 语言程序的具体开发过程包括如以下几步内容：

准备工作，这里主要指准备待开发应用的各类资料，包括组态内容、位号定义、算法要求等

创建源程序

编写全局定义程序，包括宏定义、全局变量定义、折线表定义

编写子函数

局部变量定义

编写算法代码

编写折线表

编写主函数

局部变量定义

编写算法代码

编写折线表

调用子函数

存盘，主要指保存源程序到硬盘上。

编译，主要指编译源程序并改正语法错误。

联编，主要指在组态软件中进行再次编译，和其它组态信息一起生成可下装到控制站主控制卡的代码文件。

下装并调试

## 4.4 程序结构

```

global declaration           //全局定义程序（注 1、2）
main( )                      //主函数入口
{                             //大括号{}必须分别单行书写，缺一不可(以下同)
statements sequence         //语句序列
}
[return-type /*（注 4）*/ funcn /*子函数名*/ ([parameter list/*（注 3）*/])
//子函数入口
{                             //[]中的内容可写，也可不写(以下同)
statements sequence         //语句序列
}

```

注 1：全局定义程序包括宏定义、全局变量定义、函数头定义和折线表定义。

注 2：每行“//”后，或“/\*”、“\*/”间的文字为程序注释，用于加入说明文字。

注 3：为子函数的参数类型定义表。该表中可以只定义一个参数，也可以定义多个参数。

注 4：为子函数的返回值类型定义。每个子函数只有一个返回值。

## 4.5 数据类型

在 SCX 语言中，有 5 种基本数据类型的数值，见表 4-2。

表 4-2 基本数据类型

类型	关键字	字节数	表示范围
半浮点型	Sfloat	2	-7.9997 ~ +7.9997
开关型	Bool	1	0 ~ 255
浮点型	Float	4	$\pm 1.175490351\text{E}-38 \sim 3.402823466\text{E}+38$
长整型	Long [int]	4	-2147483648 ~ 2147483647
整数	Int	2	-32768 ~ +32767
数组	变量[下标]	同类型	
结构	struct 结构名	不定	
累积型	structAccum	8	

说明：

半浮点型：用于位号的赋值和运算。

开关型：即无符号字节型，用于开关位号的赋值和运算。开关型约定如下：

触点闭表示 ON，

触点开表示 OFF。

浮点型：用于浮点运算（注意：由于控制站运算速度的限制，建议只使用少量浮点运算）。

长整型：用于需要长整数的赋值和运算，如计数器，定时器等。

整型：用于整数运算。

数组：下标只能用整型常数、整型变量或简单表达式，不能用数组或函数。

数组声明：类型 变量[下标范围];

如：

int temp[20];//表示 20 个整数，起始下标为 0

变量不能直接在声明中赋值，变量不能同名，变量区分大小写。

在 SCX 语言中，有 2 种扩展数据类型

百分型：采用十二位小数的定点数，规定如下：

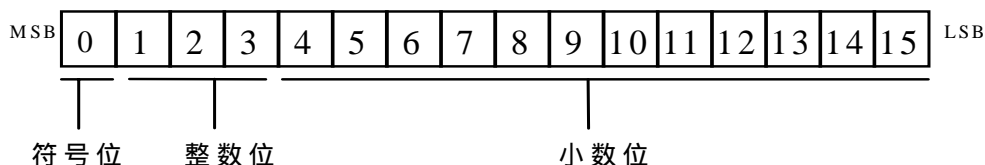


图 4-1 百分型数据类型

12 位小数的定点数，运算精度  $2^{-12}=0.024\%$ ；

即：小于 0.024% 的数被看作 0。

SUPCON WebField 系列控制系统规定，所有输入信号被转化成 0.000~1.000 的数值。（实际上是 0.00~0.9997 的四舍五入）

时间型：SUPCON WebField 系列控制系统时间型数据的标准单位为 0.1 秒，即一个码表示 0.1 秒，所以十六位无符号二进制数能表示的最大时间为 6553.5 秒，除系统变量 Timerm[n]和 Timers[n]输出的时间单位分别为分、秒外，Timerms[n]的时间单位为 0.1 秒。

## 4.6 常数表示

常数直接由数值表示。

在表达式中，常数类型必须同表达式参数类型相同，如：

int temp\_1, temp\_2;

float temp\_3, temp\_4;

sfloat temp\_5, temp\_6;

temp\_1 = 3;

temp\_2 = temp\_1\*2;//2, 3 表示整数

temp\_3 = 0.234;

temp\_4 = temp\_3\*2;//0.234, 2 表示浮点数

temp\_5 = 2.5f;  
temp\_6 = temp\_5\*2.0f; //2.5f, 2f 表示半浮点数  
开关型常数为 ON、OFF :  
开关类型变量为字节型，只能被赋值为开关型常数。

## 4.7 系统变量

系统有一些预先定义的变量用于控制信息的交换处理等，用户也可以访问这些变量。见表 4-3。

表 4-3 系统预定义变量

变量名	范围	用户权限	描 述
Timerm[n]	n=0 ~ 255	可读可写	第 n 号可随机访问的分定时器
Timers[n]	n=0 ~ 255	可读可写	第 n 号可随机访问的秒定时器
Timerms[n]	n=0 ~ 255	可读可写	第 n 号可随机访问的 100 毫秒定时器
g_bsc[n]	n=0~63	可读可写	第 n 号可随机访问的单回路控制模块
g_csc[n]	n=0~63	可读可写	第 n 号可随机访问的串级回路控制模块

## 4.8 位号表示

在 SCX 程序中引用位号时，需用\_TAG ( " " ) 把组态时定义的位号括起来，如\_TAG("PI-101a")。位号也可以当做数组来使用，如\_TAG("PI-101a")[1]表示位号 PI-101a 的下一个位号。用户可以自己参考组态中的定义直接在 SCX 语言源程序中合适的位置引用位号，也可以通过工具条上“TAG”命令选择位号并插入到指定的位置。

程序中引用位号时的注意事项：

所引用的位号不必预先在程序中定义，但必须是在组态文件中已经定义的。

位号引用规则：用户直接用位号标识代表位号值，可以进行取值和赋值操作。

不允许出现的字符集：“，”，{，}。

位号的第一字符只能是字母或下划线。

\_Tag 为关键字，不允许用户在非位号定义处使用。

位号中的空格不忽略，而前后的空格自动忽略。

( 与 “ 之间、 ” 与 ) 之间的空格忽略。

提供支持\_TAG("\*\*\*\*\*")[i]的书写形式，表示该位号后的第 i 个位号。

提供支持\_TAG("\*\*\*\*\*").MEMBER 的书写形式，表示该回路的成员变量。

提供支持若干关键字，如：main，if，else 等。

提供区分位号的码制/百分数制（1 码=1/4095 百分数）及其相互转换的函数。

提供区分位号的可读/可写功能。

位号字符数必须少于等于 10。

位号数组下标只能是变量、整数、简单表达式，不能用数组、函数和混合表达式。

合法位号举例：

\_TAG("abc"), \_TAG("ab c") , \_TAG("loop1").SV , \_TAG("loop")[i] , \_TAG("loop")[i].SV ,  
\_TAG("abc+1") , \_TAG("T1-101") [number+count]

其中，\_TAG("\*\*\*\*\*") 和 \_TAG("\*\*\*\*\*")表示同一位号。

## 4.9 标识符的定义规则

在 SCX 语言中，系统变量、用户自定义变量、函数、宏、标号等都需要定义其名称以供识别，这些名称统称为标识符。

标识符的命名必须满足下列条件：

- 以英文字母开头；
- 续以英文字母、数字或下划线( )；
- 字符长度最多为 32。

## 4.10 运算符

SCX 语言支持算术、逻辑运算、关系比较运算以及其他特殊运算。见表 4-4。

表 4-4 算术运算符

类 型	运 算 符	描 述
算 术 运 算	+	单目正
	-	单目负
	+	加
	-	减
	*	乘
	/	除
逻辑运算	NOT	逻辑非
	AND	逻辑与
	OR	逻辑或
关系比较运算	<	小于
	<=	小于等于
	>	大于
	>=	大于等于
	==	等于
	!=	不等于
其他运算	.	取模块或位号成员
	( )	括号
	=	赋值
	{ }	语句块



SCX 语言语法规定，逻辑运算可以和比较运算混合，但禁止算术运算和逻辑运算混合。

### 4.10.1 运算符优先级

SCX 语言的运算符的优先级按以下顺序从高到低排列(同一行运算符的运算优先级相同)：

{ } ( 该运算符只能单独一行 )



```

.
()
NOT
-(单目负) +(单目正)
* /
+ -
< <= > >= == !=
AND OR
=(赋值运算)

```

## 4.11 运算表达式

运算表达式由操作数（包括变量、常数及函数）和运算符号组成，可以根据运算优先级和各运算符的运算规则来计算出唯一的值。

运算表达式由以下规则产生：

opt1 为任意一个单目运算符，opt2 为任意一个二目运算符，oprn 为任意操作数(包括函数、模块成员的引用)。exp 为运算表达式，=表示定义。

```

exp = oprn ;
exp = opt1 oprn ;
exp = oprn opt2 oprn ;
exp = opt1 exp ;
exp = exp opt2 exp ;

```

在运算过程中，如果数据类型和运算要求的类型不一致，将导致程序出错。用户如需要进行混合运算，必须利用编译系统给定的函数转换类型。

在进行一些二目运算时，如+、-、\*、/，要求它们的操作数数据类型一致。如果它们的数据类型不一致，编译时 SCX 语言开发环境将给出错误信息。如果用户需要实现不同类型数据的混合运算时，必须利用语言提供的函数首先进行转换。每行只能写一个运算表达式。

## 4.12 语法结构

### 4.12.1 全局定义程序

全局定义程序中包括了宏定义、全局变量定义、函数头定义和折线表定义。在全局定义程序中定义的宏、全局变量和折线表，在程序各函数中都有效。

宏定义

语法：

```
#define 宏名 常数
```

解释：

定义一个常数。

举例：

```
#define Pi          3.14
```

```
#define LoopTime      160      //选择定时器序号
```



对程序进行编译时，程序中引用宏定义的地方，编译程序是先将相应的宏定义代替引用的宏，然后再进行编译。

变量声明

语法：

数据类型 变量名 1 ， 变量名 2 ， ... ， 变量名 n ；

数据类型 变量名[数组长度]；//数组声明

解释：

变量声明必须单行列出，不能在声明行对变量赋值。

数组只能是一维的。引用数组时，下标范围从 0 至数组长度减一。

举例：

```
sfloat    sA[60] ;                //表示共有 60 个半浮点数，起始下标为 0，末尾下标为 59
```

```
sfloat    sB[60] ;
```

```
sfloat    Ti210a,Ti210b ;          //变量 Ti210a 和 Ti210b 的数据类型都是半浮点型，
                                   //可以放在同一行进行数据类型定义
```



全局变量和局部变量的声明在语法上都是一致的。只是全局变量是定义在全局定义程序块中，在程序各函数中都有效；而局部变量是定义在某个具体函数中，只在该函数中有效。

#### 4.12.2 赋值语句

语法：

变量名 = 运算表达式；

举例：

```
temp_A = (temp_B + C) * D / temp_F ;
```

```
_TAG("DP205B")=ON ;              //将开关动作按钮置为 ON 状态
```

#### 4.12.3 条件语句

语法：

if 条件表达式 1

```
{
```

```
...
```

//语句序列 1

```
}
```

[[ else if 条件表达式 2

```
{
```

```
...
```

//语句序列 2

```
}
```

else if 条件表达式 3

```

{
...                               //语句序列 3
}
]
else
{
...                               //语句序列 4
}
]
...                               //语句序列 5

```

解释：

按如下步骤执行：

计算 if 后面的条件表达式 1，如果值非零，执行相应的语句序列 1，程序转入 if-else 块后，执行语句序列 5；如为零，转步骤 2；

如有 else if 部分(可以有多个 else if 分支)，计算 else if 后面的条件表达式 2，如果值非零，执行相应的语句序列 2，程序转入 if-else 块后，执行语句序列 5；如为零，如果存在下一个 else if 分支，判断执行此分支，否则转步骤 3；

执行 else 后的语句序列 4，程序转入 if-else 块后，执行语句序列 5。

举例：A、B、C 为三个不等数，求取它们的最大值，并赋给变量 X。

```

...
if (A > B)
{
x = A;
if (A > C)
{
x = A
}
else
{
x = C
}
}
else if (B > C)
{
x = B;
}
else
{
x = C;
}

```

#### 4.12.4 循环语句

for 语句

语法：

for (条件表达式)

{

...

//语句序列 1

}

...

//语句序列 2

其中条件表达式中所用的计数变量为 int 型变量。初值、终值、增量为运算表达式。

解释：

按以下步骤进行：

计算初值、终值和增量，增量缺省值为 1；

初值赋给计数变量；

比较计数变量和终值：如果计数变量 < 终值(此时增量 > 0)或计数变量 > 终值(此时增量 < 0)，则执行语句序列 1；否则，程序转入 for 块后的语句，开始执行语句序列 2；

计数变量加上增量；

转步骤 3。

举例：

```
int Stat[23];
```

```
int I;
```

```
for (I=0; I<23; I=I+1)
```

```
{
```

```
Stat[I] = I;
```

```
}
```

while 语句

语法：

while (条件表达式)

{

...

//语句序列 1

}

...

//语句序列 2

解释：

按以下步骤进行：

计算条件表达式的值；

如果值为真，执行语句序列 1，重复步骤 1；如为假，程序转入 while 块后的语句，开始执行语句序列 2。

举例：

```
int D;
```

```
D = 10;
```

```
while (D >= 5)
```

```
{
D = D - 1 ;
}
```

### 4.13 函数和子程序

函数或子程序是一个独立的程序块，具有某一特定的功能。

定义函数或子程序

[返回类型] 函数或子程序名([参数列表])

```
{
...
}
```

其中，参数列表形式如下：

数据类型 参数 1 , ... , 数据类型 参数 n

此时，参数 1 ~ 参数 n 称为形式参数，简称形参。

调用函数或子程序

引用函数或子程序时，若该函数或子程序定义了形参，则必须传入相应的实际参数（简称实参）。函数或子程序只能作为单独的一句语句调用，实现子程序的功能。如果有返回值，则实现了函数的功能，可以在等式的右边引用（不能作为运算表达式的一个操作数据项）。

举例

```
void TT_201B(void)
{
int A , B , C ;
Initialize ;
...
C = Max(A , B) ;
}
...
Initialize()
{
...
}
...
int Max(int A , int B)
{
If (A > B)
{
return A ;
}
else
{
return B ;
}
```

```
}
}
```



组态软件规定 SCX 语言组态时，自定义控制程序入口函数名为 `main()`。

## 4.14 折线表

系统定义有 32 个二维 10 段折线表和 32 个一维 16 段折线表，用户可以改变这些折线表的定义。

二维折线表定义语句

`#F_XY` 序号

```
{
X 值半浮点常数列表；
Y 值半浮点常数列表；
}
```

其中，序号是一个无符号的整型常数，取值范围为 0 ~ 31。

浮点常数列表是一个浮点常数的序列，常数之间用逗号隔开，如 “ 1.0f, 2.0f, 3.0f ”。

其中，X 常数序列必须递增。

一维折线表定义语句

`#F_X` 序号

```
{
半浮点常数列表；
}
```

其中，序号是一个整型常数，取值范围为 0 ~ 31。

折线表计算

调用 `FXY`、`FX` 进行计算。参见库函数部分。

在程序中，还可以调用函数 `getfx`、`setfx`、`getfxy_x`、`setfxy_x`、`getfxy_y` 和 `setfxy_y` 引用或改变折线表的局部数据。参见库函数。

## 4.15 位号成员和控制模块的引用

在程序中用 `_TAG( " " )` 把组态时定义的位号括起来以实现位号的引用，如 `_TAG("PI-101a")`。位号也可以当做数组来使用，如 `_TAG("PI-101a") [1]` 表示位号 PI-101a 的下一个位号（在组态时决定了位号的次序）。

SCX 语言提供两种控制模块：单回路控制模块和串级回路控制模块。位号数据和控制模块有许多数据成员和函数成员，供读写数据和调用运算功能。取模块成员用 “ . ” 操作符，不用该操作符时，表示取它的缺省成员。位号和控制模块的成员详见表 4-5、表 4-7 和表 4-8。

## 4.15.1 位号

表 4-5 模入位号数据成员列表

成员	说 明	成员类型
FLAG	信号质量码	Int
PV	信号值(缺省成员*)	Sfloat
ENA	报警使能开关 (ON 自动/OFF 手工输入)	Bool
MPV	手/自动 (OFF 关/ON 开)	Bool
SUM0	累积值 12 位小数+4 位无符号整数	Sfloat
SUM1	16 位无符号整数累积值次高位	Int
SUM2	16 位无符号整数累积值高位	Int
FILTER	滤波常数 (0.1 秒为单位的整型)	Int
CUT	小信号切除 (14 位)	Sfloat**
HH	报警上限 (14 位)	Sfloat**
HI	报警上限 (14 位)	Sfloat**
LO	报警下限 (14 位)	Sfloat**
LL	报警下下限 (14 位)	Sfloat**
DZ	报警死区 (14 位)	Sfloat**

模出量、开入量和开出量位号只有一个成员 PV。不用操作符“.”时，表示取该位号的缺省成员 PV，如，\_TAG("pv-101a")=\_TAG("pv-101a").PV。

以上六个成员是特殊的半浮点数类型，它们共有 14 位，也就是在普通半浮点数后再续以 6 位小数，以此提高数据精度。

## 1、质量码 (Flag) 的使用方法：

由于 SCX 语言并没有位操作指令，而质量码 (Flag) 采用位定义，所以可以通过质量码与某个成员的相关数值进行“与”操作的方法来获得该信号是否处于报警状态，如上限报警的相关数值为 1。

表 4-6 模入位号数据成员 Flag (位号质量码) 的定义

质量码的位	说明	位格式	相关数值
D2=0	D0 上限报警	1——报警	1
		0——不报警	
	D1 下限报警	1——报警	2
		0——不报警	
D2=1	D0 上上限报警	1——报警	5
		0——不报警	
	D1 下下限报警	1——报警	6
		0——不报警	
D3	偏差报警	1——报警	8
		0——不报警	
D4			
D5			
D6			
D7			
D8	模块故障	1——故障，伪信号	256
		0——无故障	
D9	速度报警	1——报警	512
		0——不报警	

D10	信号手动 / 自动输入	1-自动 (测量信号)	1024
		0-手动 (设定信号)	
D11	超量程报警*	1——报警	2048
		0——不报警	
D12	时限报警	1——报警	4096
		0——不报警	
D13			
D14			
D15			

若测量点与变送器之间的线路短路或断路，导致信号超量程，这又可称为开路报警。如输入为型电流信号，量程为 (4 ~ 20) mA，若输入线路短路，则为 0mA，那通过质量码的 D11 位就可以判断出来 (详见应用举例)。

例如，如果你需要判断某个模入信号是否处于上上限报警，首先将这个模入信号的成员 Flag 与成员 HH 的相关数值 (详见表 4-6) 进行“与”操作，即屏蔽无关位，只留下 FLAG 中的相关位 (D2D0)；然后将“与”操作完成后的数值与该相关数值 5 比较判断是否相等。如果相等，则表明此信号处于上上限报警状态；否则此信号没有处于上上限报警状态。

## 2、应用举例：利用质量码进行报警判断。

```
void ALARM()                                //模拟量输入信号，位号为 FI-OLD
{
if (andint(_TAG("FI-OLD").FLAG, 5) == 5) //判断是否上上限报警 (相关数值为 5)
{
    _TAG("HH-ALM") = ON ;                //自定义位号存储上上限报警信息
}                                         //ON：FI-OLD 处于上上限报警(以下同)
else
{
    _TAG("HH-ALM") = OFF ;
}
if (andint(_TAG("FI-OLD").FLAG, 1) == 1) //判断是否上限报警 (相关数值为 1)
{
    _TAG("HI-ALM") = ON ;
}
else
{
    _TAG("HI-ALM") = OFF ;
}
if (andint(_TAG("FI-OLD").FLAG, 2) == 2) //判断是否下限报警 (相关数值为 2)
{
    _TAG("LO-ALM") = ON ;
}
else
{
    _TAG("LO-ALM") = OFF ;
}
```



```
}
if (andint(_TAG("FI-OLD").FLAG, 6) == 6) //判断是否下下限报警(相关数值为 6)
{
    _TAG("LL-ALM") = ON ;
}
else
{
    _TAG("LL-ALM") = OFF ;
}
if (andint(_TAG("FI-OLD").FLAG, 2048) == 2048) //判断是否输入超量程或开路报警
// ( 相关数值为 2048 )
{
    _TAG("OVER-ALM") = ON ;
}
else
{
    _TAG("OVER-ALM") = OFF ;
}
}
```

#### 4.15.2 基本控制功能模块 (BSC)

单回路 PID 控制模块(BSC)实现单回路控制，其功能逻辑如图 4-2 所示，它包含一个标准 PID 控制单元和各种不同的数据成员，可进行功能扩展。

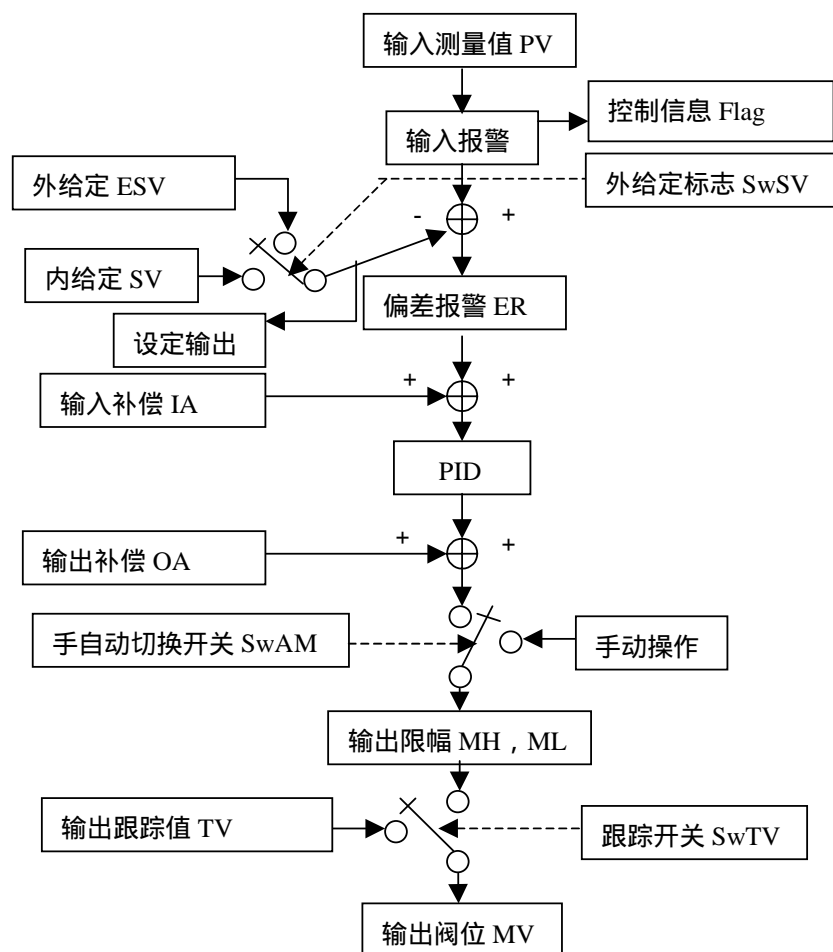


图 4-2 BSC 模块结构图

## 1、 BSC 函数成员

表 4-7 BSC 数据成员列表

成员	说明	数据类型	补充说明	属性
PV	输入测量值	sfloat		R
Flag	控制信息	bool		
SwSV	给定值切换开关	bool	ON：外给定 OFF：内给定	RW
ESV	外给定值	sfloat		RW
SV	内给定值	sfloat		RW
ER	偏差报警值	sfloat		
IA	输入补偿	sfloat		RW
KP	比例常数	sfloat	$2 \times Kc = 1/P$	RW
TI	积分时间	int	0.1 秒为单位	RW
TD	微分时间	int	0.1 秒为单位	RW
OA	输出补偿	sfloat		RW
SwAM	手动 / 自动切换开关	bool	ON：自动 OFF：手动	RW
MH	输出限幅上限	sfloat		RW

ML	输出限幅下限	sfloat		RW
SwTV	输出跟踪开关	bool	ON：输出跟踪 OFF：输出不跟踪	RW
TV	输出跟踪值	sfloat		RW
SwDT	微分方式切换开关	bool	ON：dPV/dt OFF：dErr/dt	RW
SwNeg	控制作用切换开关	bool	ON：反作用 OFF：正作用	RW
KV	可变增益	sfloat		
TS	控制周期	int		
MV	输出阀位	sfloat		RW

根据数据成员中的不同给定值（表 4-7），运行 RUN 函数可以实现效果不同的单回路控制。SUPCON WebField 系列控制系统中，每个控制站最多有 64 个 BSC 模块，同一序号的 BSC 模块只能使用一次。

运行控制模块函数：sfloat bsc(sfloat pv, int n)；

参数说明

pv：被控量

n：模块序号 0～63

返回值：返回输出阀位值。

BSC 模块的使用方法请参考库函数。

2、 举例

...

sfloat a；

...

a = bsc(pv, 0)；

...

#### 4.15.3 串级控制功能模块（CSC）

串级 PID 控制模块(CSC)实现串接控制，其功能逻辑如图 4-3 所示，它包含两个标准 PID 控制单元和各种不同的数据成员，可进行功能扩展。

1、 CSC 函数成员：

表 4-8 CSC 数据成员列表

成 员	说 明	数据类型	补充说明	属性
ExPV	外环测量值	sfloat		R
Flag	控制信息	bool		
SwSV	外环的外给定开关	bool	ON：外给定 OFF：内给定	RW
ExESV	外环的外给定值	sfloat	内外给定都反映实际给定值	RW
ExSV	外环的内给定值	sfloat		RW
ExER	外环的偏差报警	sfloat		
ExIA	外环的输入补偿	sfloat		RW
ExKP	外环的比例系数	sfloat	$2 \times K_c = 1/P$	RW

ExTI	外环的积分时间	Int	0.1 秒为单位	RW
ExTD	外环的微分时间	Int	0.1 秒为单位	RW
ExSwDT	外环的微分方式	Bool	ON : dPV/dt OFF : dErr/dt	RW
ExSwNeg	外环的控制作用切换开关	Bool	ON : 反作用 OFF : 正作用	RW
ExTS	外环控制周期	Int	赋值为 1 即控制周期为 0.1S, 最短控制周期为 0.1S	RW
ExKV	外环可变增益	sfloat		
ExMV	外环控制量	sfloat	ExMV=InSV-ExOA	RW
ExOA	外环的输出补偿	sfloat		RW
InPV	内环测量值	sfloat		R
InSV	内环给定值	sfloat	内环内给定	RW
SwCas	串级 / 单回路切换开关	bool	ON : 串级 OFF : 单回路	
InER	内环的偏差报警	sfloat		
InIA	内环的输入补偿	sfloat		RW
InKP	内环的比例系数	sfloat	$2 \times K_c = 1/P$	RW
InTI	内环的积分时间	int	0.1 秒为单位	RW
InTD	内环的微分时间	int	0.1 秒为单位	RW
InOA	内环的输出补偿	sfloat		RW
SwAM	手动 / 自动切换开关	bool	ON : 自动 OFF : 手动	RW
MH	输出限幅上限	sfloat	*	RW
ML	输出限幅下限	Sfloat		RW
SwTV	输出跟踪开关	bool	ON : 输出跟踪 OFF : 输出不跟踪	RW
TV	输出跟踪值	sfloat		RW
InSwDT	内环的微分方式	bool	ON : dPV/dt OFF : dErr/dt	RW
InSwNeg	内环的控制作用切换开关	bool	ON : 反作用 OFF : 正作用	RW
InTS	内环控制周期	int	赋值为 1 即控制周期为 0.1S, 最短控制周期为 0.1S	RW
InKV	内环可变增益	sfloat		
InMV	内环控制量	sfloat	InMV 为最终输出	RW

主控制卡在运行串级 PID 控制模块 CSC 时, 先运行手/自动操作, 然后运行输出限幅, 最后运行输出跟踪。

CSC 的结构形式基本上和 BSC 模块相同。根据数据成员中的不同给定值 (表 4-8), 运行 RUN 函数可以实现效果不同的串接控制。SUPCON WebField 系列控制系统中, 每个控制站有 64 个 CSC 模块, 同一序号的 CSC 模块只能使用一次。

运行控制模块函数: sfloat csc(sfloat pvEx, sfloat pvIn, int n);

参数说明

pvEx: 外环被控量

pvIn: 内环被控量

n: 模块序号 0 ~ 63

返回值

返回输出阀位值。

CSCX 模块的使用方法请参考库函数。

## 2、 举例

...

sfloat a ;

...

a = csc(pvEx,pvIn,0) ;

...

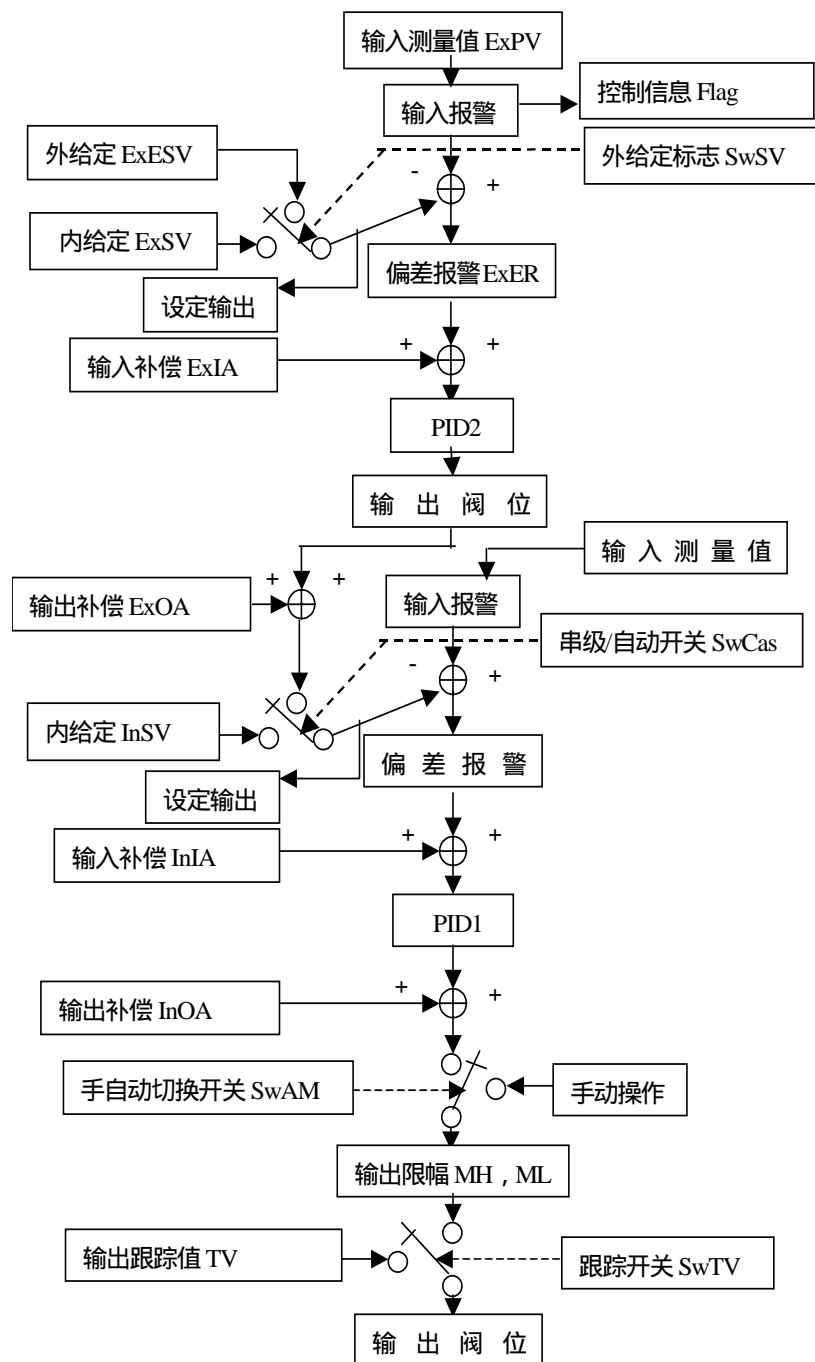


图 4-3 CSC 模块结构图

## 4.16 库函数

SCX 语言还提供库函数供用户调用。见表 4-9，具体的使用见附录。

表 4-9 SCX 语言库函数

函数名	返回值	函数类型	功能说明
sfadd	sfloat	半浮点计算模块	半浮点加法
Sfadd2	sfloat	半浮点计算模块	半浮点加法
sfsb	sfloat	半浮点计算模块	半浮点减法
Sfsb2	sfloat	半浮点计算模块	半浮点减法
sfmul	sfloat	半浮点计算模块	半浮点乘法
sfdv	sfloat	半浮点计算模块	半浮点除法
sfabs	sfloat	半浮点计算模块	半浮点求绝对值
sfsqr	sfloat	半浮点计算模块	半浮点求平方
sfsqrt	sfloat	半浮点计算模块	半浮点求平方根
Sin	float	浮点计算模块	正弦函数
Asin	float	浮点计算模块	反正弦函数
Sinh	float	浮点计算模块	工程正弦函数
Cos	float	浮点计算模块	余弦函数
acos	float	浮点计算模块	反余弦函数
cosh	float	浮点计算模块	工程余弦函数
Tan	float	浮点计算模块	正切函数
Atan	float	浮点计算模块	反正切函数
Atan2	float	浮点计算模块	反正切函数 2
tanh	float	浮点计算模块	工程正切函数
eval_poly	float	浮点计算模块	多项式求值
Pow	float	浮点计算模块	幂函数
Log	float	浮点计算模块	自然对数函数
log10	float	浮点计算模块	对数函数
Exp	float	浮点计算模块	指数函数
sqrt	float	浮点计算模块	求平方根函数
fabs	float	浮点计算模块	求绝对值函数
floor	float	浮点计算模块	按输入浮点值返回最大整数
Ceil	float	浮点计算模块	按输入浮点值返回最小整数
Vlm	sfloat	辅助计算模块	速度限制模块
hsl	sfloat	辅助计算模块	高选择器
Mav	sfloat	辅助计算模块	移动平均函数
Hal	bool	辅助计算模块	高报警模块
Ded	sfloat	辅助计算模块	纯滞后模块
ledlag	sfloat	辅助计算模块	一阶超前滞后模块
Led	sfloat	辅助计算模块	一阶超前模块

Hlim	sfloat	辅助计算模块	高限模块
Fxy	sfloat	辅助计算模块	二维折线表插值
Fx	sfloat	辅助计算模块	一维折线表插值
Msw	sfloat	辅助计算模块	多路切换模块
Getfx	sfloat	辅助计算模块	取一维折线表的数据项
Setfx	sfloat	辅助计算模块	写一维折线表的数据项
getfxv_x	sfloat	辅助计算模块	取二维折线表 X 轴的数据项
setfxv_x	sfloat	辅助计算模块	写二维折线表 X 轴的数据项
getfxy_y	sfloat	辅助计算模块	取二维折线表 Y 轴的数据项
setfxy_y	sfloat	辅助计算模块	写二维折线表 Y 轴的数据项
Sw	sfloat	辅助计算模块	开关切换模块
Llim	sfloat	辅助计算模块	低限模块
Lag	sfloat	辅助计算模块	一阶滞后模块
Lal	bool	辅助计算模块	低报警模块
Lsel	sfloat	辅助计算模块	低选择器
Kfdivk	int	混合计算模块	整数的半浮点除法
fkmulk	int	混合计算模块	半浮点整数乘法
kkdivf	sfloat	混合计算模块	整数除法
Fkdivf	sfloat	混合计算模块	半浮点整数除法
fkmulf	sfloat	混合计算模块	半浮点整数乘法
dgap2	sfloat	控制模块	二位式控制模块
dgap3	sfloat	控制模块	三位式控制模块
Csc	sfloat	控制模块	串级控制模块
Bsc	sfloat	控制模块	PID 控制模块
SteamComp	sfloat	控制模块	温压补偿函数
AddAccum	structAccum	控制模块	累积加
SubAccum	structAccum	控制模块	累积减
ConvertAccum	float	控制模块	累积转换成浮点
ConverToAccum	structAccum	控制模块	浮点转换成累积
CompAccum	int	控制模块	累积量比较
TotalAccum	structAccum	控制模块	累积加半浮点
Andint	int	控制模块	整型与
andlong	long	控制模块	长整型与
Orint	int	控制模块	整型或
Orlong	long	控制模块	长整型或
Getbit	bool	控制模块	取位值
getsfloat	sfloat	控制模块	取半浮点
Getint	int	控制模块	取整型值

getfloat	float	控制模块	取浮点值
Setbit	long	控制模块	设位值
Setsfloat	long	控制模块	设半浮点值
Setint	long	控制模块	设整型值
Setfloat	long	控制模块	设浮点值
Getmsg	long	控制模块	取控制站通讯数据
Sendmsg	void	控制模块	发数据
Norm	int	类型转换模块	归一化函数
Itosf	sfloat	类型转换模块	整数转换到半浮点
Sftoi	int	类型转换模块	半浮点转换到整数
Sftof	float	类型转换模块	半浮点转换到浮点
Ftosf	float	类型转换模块	浮点转换到半浮点
Itof	float	类型转换模块	整数转换到浮点
Denorm	sfloat	类型转换模块	反归一化
Ftoi	int	类型转换模块	浮点转换到整数

## 4.17 注释

SCX 程序中，可以用“//”或“/\* \*/”表示注释。每行“//”后或“/\*”与“\*/”间的文字为程序注释，用于加入说明文字或注释掉无用的程序行。

举例：

```
B=E-F ;           /* 求差 */
A = B + C + D ;    // 求和
```

## 4.18 关键字

以下列出 SCX 语言用到的关键字，用户定义的标识符不可以和它们相同。

```
main
bool    float    int    long    sfloat    void
if      else
for     while
break   continue goto    return
#define #F_X    #F_XY
ON      OFF      NOT     OR      AND
```

以下各标识符为系统变量名，但不是关键字：

```
g_bsc[] g_csc[] g_accum[] g_backup[]
g_msg[] timers[] timerm[] timerms[]
```

每一程序行只能有一个关键字。其中：

if, else, void, while, main 行不能有分号；

break, return, continue 行必须跟分号；

除 for 语句小括号内有且必须有两个分号，其它语句每行最多只能有一个分号存在。



## 5 编程实例

本章给出一些 SCX 语言典型应用程序实例，包括网关卡与其它智能设备互联的通信程序供编程时参考。



通过 SCX 语言编程实现的控制算法程序编译后生成的目标文件和其它组态信息联编后下装到控制站的主控卡，每个控制周期主控卡执行一次该目标文件。

### 5.1 浮点实例

```

x = 0.75
y = 5x3 + log3.2x2 - 3.02.5x + 10.0
main()                                //定义主函数名为 main
{
float a[4], y;                        //数组 a 共有 4 个单元，每个单元均定义为浮点型
//变量 y 定义为浮点型
a[0] = 10.0;                          //赋值数组 a 的第一个单元为 10.0
a[1] = pow(3.0, 2.5);                 //计算 3.02.5 并赋值给数组 a 的第二个单元
a[2] = log(3.2);                      //计算 3.2 的自然对数值并赋值给数组 a 的第三个单元
a[3] = 5.0;                           //赋值数组 a 的第四个单元为 5.0
y = eval_poly(0.75, a, 4);            //计算多项式并赋值给变量 y
}

```

### 5.2 函数和折线表实例

```

#F_X 1                                //定义序号为 1 的一维折线表
{
0f,0.05f,0.1f,0.15f,0.2f,0.25f,0.3f,0.35f,0.4f,0.45f,0.5f,0.55f,0.6f,0.65f,0.7f,0.75f,0.8f
}
#F_XY 1                                //定义序号为 1 的二维折线表
{
0f, 0.05f, 0.1f, 0.15f, 0.2f, 0.25f, 0.3f, 0.35f, 0.4f, 0.45f, 0.5f           //X 轴向
0f, 0.05f, 0.1f, 0.15f, 0.2f, 0.25f, 0.3f, 0.35f, 0.4f, 0.45f, 0.5f           //Y 轴向
}
main()                                //定义主函数名为 main
{
sfloat x1, x2;                        //定义变量 x1, x2 为半浮点型
bool HAlert, LAlert;                  //定义变量 HAlert, LAlert 为布尔型
HAlert = hal(_TAG("TI001"), 0.9f, 0.05f, 0); //对位号 TI001 进行超高限报警，
//高限为 90%

```

```

LAlart = lal(_TAG("TI001"), 0.1f, 0.05f, 0); //对位号 TI001 进行超低限报警 ,
//低限为 10%
x1 = fxy(_TAG("TI001"), 1); //对位号 TI001 进行二维折线表插值计算
x1 = vlm(x1, 0.1, 0.1, 0); //对变量 x1 进行速度限幅处理
x2 = ded(x1, 20, 50, 15); //对变量 x1 进行纯滞后处理，纯滞后 100 秒
}

```

### 5.3 DGAP 模块实例

```

main() //定义主函数名为 main
{
    sfloat dgap2err, dgap3err; //定义变量 dgap2err, dgap3err 为半浮点型
    sfloat dgap2mv; //定义变量 dgap2mv 为半浮点型
    sfloat dgap3temp; //定义变量 dgap3temp 为半浮点型
    dgap2err = _TAG("dgap2sv") - _TAG("dgap2pv"); //计算给定值与测量值的偏差
    //并赋值给变量 dgap2err
    _TAG("dgap2out") = dgap2(dgap2err, _TAG("dgap2hys"), 0);
    //对偏差 dgap2err 进行二位式调节，输出阀位值给位号 dgap2out
    if(_TAG("dgap2out") > 0.5f)
    {
        dgap2mv = 1.0f; //输出大于 50%，将阀全开
    }
    else
    {
        dgap2mv = 0; //输出不大于 50%，将阀全关
    }
    dgap2mv = ded(dgap2mv, 5, 10, 0); //纯滞后处理，纯滞后 5 秒
    _TAG("dgap2pv") = lag(dgap2mv, 300, 0); //一阶滞后处理，时间常数为 30 秒
    dgap3err = _TAG("dgap3sv") - _TAG("dgap3pv"); //计算给定值与测量值的偏差
    //并赋值给变量 dgap3err
    _TAG("dgap3out") = dgap3(dgap3err, _TAG("dgap3zone"), _TAG("dgap3hys"), 0);
    //对偏差 dgap3err 进行三位式调节，输出阀位值给位号 dgap3out
    if(_TAG("dgap3out") > 0.75f)
    {
        _TAG("dgap3xxx") = _TAG("dgap3xxx") + _TAG("dgap3dx") / 2.0f;
        //输出大于 75%
    }
    else if(_TAG("dgap3out") < 0.25f)
    {
        _TAG("dgap3xxx") = _TAG("dgap3xxx") + _TAG("dgap3dx") / 2.0f;
        //输出小于 25%
    }
}

```

```

    }
    if(_TAG("dgap3xxx")>1.0f)
    {
        _TAG("dgap3xxx")= 1.0f;           //进行高限限幅
    }
    if(_TAG("dgap3xxx")<0.0f)
    {
        _TAG("dgap3xxx")= 0.0f;           //进行低限限幅
    }
    dgap3temp = ded(_TAG("dgap3xxx"), 5, 10, 1); //进行纯滞后处理，纯滞后 5 秒
    _TAG("dgap3pv")= lag(dgap3temp, 300, 1); //进行一阶滞后处理，
                                           //时间常数为 30 秒
}

```

## 5.4 PID(比例积分微分)控制

PID 控制是人们在长期的理论研究和生产实践中总结出来的经典控制方案，其特点是方案简单、功能灵活、对过程模型不敏感、鲁棒性好。在化工、制药、冶金、精细化工等工业过程中有 90% 以上的控制回路采用的是 PID 控制。SCX 语言能够对 PID 回路进行灵活的设置，下面的程序是某缩合工序的应用实例。

缩合过程是精细化工行业常见的一步反应过程，将小分子的化学键打开，重新聚合成大分子或者高分子产物。在缩合工序中为了提高收率需要尽量抑制同分异构体的生成，也就是要抑制副反应，因此对温度控制的要求比较高。本例要求缩合釜的温度要沿着一条给定的温度曲线来变化。由于在升温、保温和降温的过程中对象特性的差异，为了保证好的曲线拟合效果我们采用了分段变 PID 参数的控制方法，在升温、保温和降温过程分别赋给回路以不同的 PID 参数值，并且使该参数能够在实时监控软件中在线整定。

```

main()                                //定义主函数名为 main
{
    bool    Flag;                      //缩合保温和升温标志
    g_bsc[0].SwDT=ON;                  //设置微分先行开关
    //*****
    //          缩合温度控制程序区，采用分段变 PID 参数控制
    //*****
    if (_TAG("TA-STATE")==0)          //手动阶段
    {
        g_bsc [0].SwAM=OFF;             //回路 0 设置为手动
        timers[16]=0;                   //升温定时器清零
        timers[17]=0;                   //恒温定时器清零
    }
    else if(_TAG("TA-STATE")==1)       //全速降温阶段

```

```

{
    g_bsc [0].SwAM=ON;           //回路 0 设置为自动
    g_bsc [0].SV=0.164f;         //回路0 设定值设为18 (量程 0 ~110 )
    timers[16]=0;                //升温定时器清零
    timers[17]=0;                //恒温定时器清零
}
else if(_TAG("TA-STATE")==2)    //保温阶段，加酸
{
    g_bsc [0].SwAM=ON;           //回路 0 设为自动
    g_bsc [0].SV=0.164f;         //设定值为 18
    Flag=OFF;                    //标志设为保温
    timers[16]=0;                //升温定时器清零
    timers[17]=0;                //恒温定时器清零
}
else if(_TAG("TA-STATE")==3)    //提温阶段
{
    g_bsc [0].SwAM=ON;           //回路 0 设为自动
    Flag=ON;                     //标志设为升温
    if (timers[16]>60)            //判断升温定时器是否大于 60 秒
    {
        g_bsc [0].SV= g_bsc [0].SV+0.0018f; //若是，设定值增加 0.2
        timers[16]=0;            //升温定时器清零
    }
    //本段程序目的是使该回路设定值每分钟升高 0.2
    if (g_bsc [0].SV>0.309f)      //判断回路 0 的设定值是否超过 34
    {
        _TAG("TA-STATE")=4; //若是，切换至恒温静置阶段
    }
    timers[17]=0;                //恒温定时器清零
}
else if(_TAG("TA-STATE")==4)    //恒温静置阶段，保持 3 小时
{
    if (timers[17]<10800)         //判断恒温静置是否超过 3 小时
    {
        g_bsc [0].SwAM=ON;       //若否，回路 0 设为自动
        g_bsc [0].SV=0.309f;     //设定值为 34
    }
    else                          //若超过 3 小时
    {
        _TAG("TA-STATE")=5; //切换到 “ 停止 ”
    }
}

```

```

    }
    Flag=OFF;           //标志设为保温
    timers[17]=0;       //恒温定时器清零
}
else                   //停止阶段
{
    g_bsc [0].MV=0.0f;   //阀门全关
}
if (_TAG("TI201A")>0.364f)
{
    g_bsc [0].SV=_TAG("TI201A")-0.01f;
    g_bsc [0].TI=2000;
}
//*****
if (Flag==OFF)         //当标志为保温时
{
    g_bsc [0].KP=_TAG("T201BWP")*5.0f; //将保温阶段 PID 参数赋给该回路
    g_bsc [0].TI=_TAG("T201BWI");
    g_bsc [0].TD=_TAG("T201BWD");
}
else                   //当标志为升温时
{
    g_bsc [0].KP=_TAG("T201TWP")*5.0f; //将升温阶段 PID 参数赋给该回路
    g_bsc [0].TI=_TAG("T201TWI");
    g_bsc [0].TD=_TAG("T201TWD");
}
_TAG("TV201A")=bsc(_TAG("TI201A"),0); //运行单回路控制模块,回路号为0
}

```

## 5.5 联锁保护

联锁保护是过程控制中常见的一种控制方法，在过程控制系统在某些参数超出正常的变动范围时，紧急关停某些设备或者关闭或打开某些阀门。下面的例子表示当各种计量槽液位达到 100%时，关闭相应的物料泵。各计量槽顶部装有音叉，液位满信号是以开关量的形式送给控制系统。

```

main()                 //定义主函数名为 main
{
    //限位动作程序
    if(_TAG("LS205")==ON) //判断干苯计量槽是否已满
    {

```

```

_TAG("DP210B")=OFF;           //若是，关闭两台干苯泵
_TAG("DP210A")=OFF;
    }
    if(_TAG("LS204A")==ON)      //判断废酸计量槽是否已满
    {
if(_TAG("J_l iansuo")==0)
{
_TAG("DP202A")=OFF;           //若是，关闭东废酸泵
}
else
{
_TAG("DP202B")=OFF;           //若否，关闭西废酸泵
}
    }
    if(_TAG("LS203A")==ON)      //判断发烟酸计量槽是否已满
    {
_TAG("DV202")=OFF;           //若是，关闭发烟酸液下泵
    }
    if(_TAG("LS202")==ON)       //判断精醛计量槽是否已满
    {
_TAG("DV201")=OFF;           //若是，关闭精醛液下泵
    }
if(_TAG("LS201")==ON)          //判断氯苯计量槽是否已满
    {
_TAG("DP201A")=OFF;           //若是，关闭两台氯苯泵
_TAG("DP201B")=OFF;
    }
}
}

```

## 5.6 网关卡与其它智能设备互联通信程序

网关卡是通信接口单元的核心，是 SCnet 网络节点之一。在 SCnet 中网关卡处于与主控卡同等的地位。它解决了 SUPCON WebField 系列控制系统与其他厂家智能设备的互联问题。其作用是将用户智能设备的数据通过通信的方式联入 SUPCON WebField 系列控制系统中，通过 SCnet II 网络实现数据在 SUPCON WebField 系列控制系统中的共享。

网关卡的通信驱动程序可以通过 SCX 语言编写实现，由组态软件下载到网关卡中运行。SCX 语言提供了使用方便的串行通信库函数，Modbus 协议库函数，HostLink 协议库函数。

网关卡已实现通信的协议如下：

- Modbus-RTU                   （二进制）

- HostLink-ASCII (二进制)
- Mitsubishi FX2 系列
- 自定义：用户通信协议开放，波特率 19,200bps

实现与符合上述协议的智能设备，用户只需要直接调用 SCX 语言内相关的库函数即可。

### 5.6.1 Modbus-RTU 协议

背景：通信协议 Modbus，波特率 4,800bps，数据格式为 8 位数据位，无校验。

下位机为新加坡 TANK Reader，地址 201，要读 13 个罐的液位和温度值，每个罐的温度包括 3 个点温度值和一个平均温度。用的是 readholdingreg 命令，分 5 次读共 65 个位号。

硬件配置：1 块网关卡

互联方案：直接连接在 RS232 口上，其拓扑结构如下：

背景：通信协议 Modbus，波特率 4,800bps，数据格式为 8 位数据位，无校验。

下位机为新加坡 TANK Reader，地址 201，要读 13 个罐的液位和温度值，每个罐的温度包括 3 个点温度值和一个平均温度。用的是 readholdingreg 命令，分 5 次读共 65 个位号。

硬件配置：1 块网关卡

互联方案：直接连接在 RS232 口上，其拓扑结构如下：

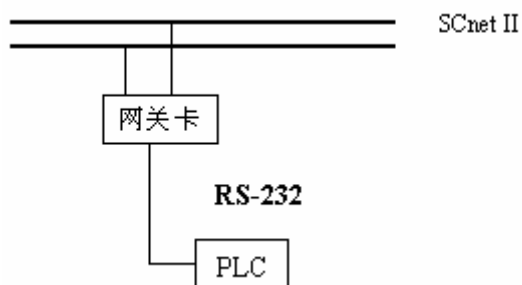


图 5-1 互联方案

SCX 语言程序如下：

```

int phase; //加上相位，使 SCX 语言程序在一个周期内只执行一条 readholdingreg
int error; // 语句，出错后还要停两个周期不做收发，一般一个周期内只能连
main()    //续执行三条收发语句，建议一个周期内只执行一条，周期可改。
{
    int i;
    int a[26];
    long aa[13];
    sfloat lev[13];
    float f;
    long l,m,n;
    if((phase<0) OR (phase > 4))

```

```

{
    phase = 0;
}
_TAG("add") = _TAG("add") + 1; // 检验 SCX 语言程序是否在运行
setcomm(4800,0);
if(phase == 0 AND error == 0)////////
{
    setdelaytime(300);
    _TAG("return0")= readholdingreg(201,4304,26,a); // 读液位,
    if(_TAG("return0") == 0) // 2 个寄存器存放一个液位, 34304 为
    {
        // 起始寄存器的地址
        for(i = 0; i < 13; i = i+1)
        {
            l = 0;
            m = setint(l,a[2*i],1);
            n = setint(l,a[2*i+1],0);
            aa[i] = m + n;
            l = aa[i];
            f = ltof(l);
            f = f * 0.0001; // 单位由 0.1mm 变为 1m
            f = f * 0.091; // 0~11—0~1 线性平移量程变为 (0 米 ~ 11 米)
            lev[i] = ftof(f); // 液位用 sfloat 位号组态
        }
        _TAG("level1") = lev[0];
        ... // 此处省略 11 条赋值语句
        _TAG("level13") = lev[12];
    }
    else
    {
        error = 2;
    }
}
if(phase == 1 AND error == 0)
{
    setdelaytime(300);
    _TAG("return3")= readholdingreg(201,5008,13,a); // 读温度 3,
    if(_TAG("return3") == 0) // 一个寄存器存放一个温度
    {
        for(i = 0; i < 13; i = i+1)
        {
            f = itof(a[i]);
            f = f * 0.1;

```



```

        f = f * 0.00286 + 0.143; // 温度量程 ( -50 ~ 300 )
        lev[i] = ftosf(f);
    }
    _TAG("temp103") = lev[0];
    ... //此处省略 11 条赋值语句
    _TAG("temp4203") = lev[12];
}
else
{
    error = 2;
}
}
if(phase == 2 AND error == 0)
{
    setdelaytime(300);
    _TAG("return2")= readholdingreg(201,5040,13,a); //读温度 2 ,
    if(_TAG("return2") == 0 )
    {
        for(i = 0;i < 13;i= i+1)
        {
            f = itof(a[i]);
            f = f * 0.1;
            f = f * 0.00286 + 0.143;
            lev[i] = ftosf(f);
        }
        _TAG("temp102") = lev[0];
        ... //此处省略 11 条赋值语句
        _TAG("temp4202") = lev[12];
    }
    else
    {
        error = 2;
    }
}
}
if(phase == 3 AND error == 0)
{
    setdelaytime(300);
    _TAG("return1")= readholdingreg(201,5072,13,a); //读温度 1 ,
    if(_TAG("return1") == 0 )
    {
        for(i = 0;i < 13;i= i+1)
        {
            f = itof(a[i]);

```

```

        f = f * 0.1;
        f = f * 0.00286 + 0.143;
        lev[i] = ftosf(f);
    }
    _TAG("temp101") = lev[0];
    ... //此处省略 11 条赋值语句
    _TAG("temp4201") = lev[12];
}
else
{
    error = 2;
}
}
if(phase == 4 AND error == 0)
{
    setdelaytime(300);
    _TAG("return4")= readholdingreg(201,5200,13,a); //读温度 ,
    if(_TAG("return4") == 0 ) //平均值
    {
        for(i = 0;i < 13;i= i+1)
        {
            f = itof(a[i]);
            f = f * 0.1;
            f = f * 0.00286 + 0.143;
            lev[i] = ftosf(f);
        }
        _TAG("tempavg01") = lev[0];
        ... //此处省略 11 条赋值语句
        _TAG("tempavg42") = lev[12];
    }
    else
    {
        error = 2;
    }
}
phase = phase + 1;
if(error < 0 OR error > 2 )
{
    error = 0;
}
if(error > 0)
{
    error = error -1;
}

```

```

    }
}

```

### 5.6.2 HostLink-ASCII

背景：通信协议 HostLink，波特率 19200bps，数据格式为 1 个起始位，8 个数据位，1 个停止位，偶校验。

OMRON PLC 将需要传递给 SUPCON WebField 控制系统的数据存放在 DM 区，具体存放规定如下：

表 5-1 DM 区分配

DM 区地址	存放的数据	备 注
2000~2019	模拟量输入	共 20 个字,每个字表示一个模拟量 0~4096 表示( 4~20 ) mA 的信号
2050~2053	开关量输入	每位存放一个开关量
2060~2064	开关量输出	每位存放一个开关量
2100~2139	SOE 事件存储	每个 SOE 记录由两个字构成：第一个字高 8 位为事件发生时间“分”，低 8 位为发生时间“秒”，第二个字高 10 位为事件发生时间“毫秒”，低 6 位为事件发生事件“SOE 点位号”，
2140	SOE 事件读取标志位	当有 SOE 事件产生时，PLC 将该字写为 FFFF，DCS 读完其中数据后，将该位写为 0000

由 PLC 负责 SOE 事件处理并记录 SOE 实时点；SUPCON WebField 控制系统采集 PLC 的 SOE 数据且负责 SOE 事件的显示和记录。

硬件配置：1 块网关卡（现场有一台 OMRON PLC）

互联方案：直接连接在 RS232 口上，采用点对点的方式连接。其拓扑结构如下：

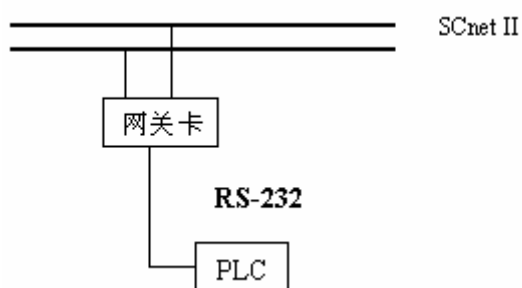


图 5-2 互联方案

SCX 语言程序如下：

```

int AIbuf[43];
int DIbuf[6]; //DI及DO数据合在一起
//int DObuf[5];
int SOEbuf[41];
//int SOEreset[2];
int hh[30];
int send[3];
int oldsoe40;

```

```

//*****

void PLCTONGXUN()//PLC通信程序
{
    int i;
    int j;
    int k;
    int Temp;
    sfloat shuju;
    long Templong;
    long templong1,templong2,templong3,templong4,templong5,
    templong6,templong8;
    send[0] = 0;
    send[1] = 0;

    _TAG("ComCount") = _TAG("ComCount") + 1;

    setcomm(9600,5); //设置波特率为9600，8位数据位，偶校验

    if (_TAG("ComCount") > 10)
    {
        _TAG("ComCount") = 0;
    }
    //*****模拟量前30个数据的
    读取
    if ((_TAG("ComCount") == 0) OR (_TAG("ComCount") == 1)) //读模拟量，连续
    读两次
    {
        setdelaytime(300);//延时300ms
        Temp = readdata(0,2000,30,AIbuf);
        //0为下位机地址，2000为下位机首个数据的地址位，30为读取数据个数，
        数据缓存到AIBUF[30]中
        if (Temp == -1)//通信失败
        {
            Temp = 5;
        }
        _TAG("AComErr") = Temp;
    }
    if (_TAG("ComCount") == 2)
    {
        if (_TAG("AComErr") == 0)//通信正常时，将AIBUF[I]中的数据写入自定义
        数据中。
        {

```

```

        for (i = 0 ; i < 30 ; i = i + 1 )
        {
            _TAG("AI")[i] = denorm(AIbuf[i]);
        }
    }
}

//*****混合数据
if ((_TAG("ComCount") == 3) OR (_TAG("ComCount") == 4)) //读模拟量，连续
读两次
{
    setdelaytime(300);//延时300ms
    Temp = readdata(0,2030,26,hh);
    //0为下位机地址，2000为下位机首个数据的地址位，30为读取数据个数，
数据缓存到hh[30]中
    if (Temp == -1)//通信失败
    {
        Temp = 5;
    }
    _TAG("AComErr") = Temp;
    _TAG("DComErr") = Temp;
    _TAG("DOComErr") = Temp;
}
if (_TAG("ComCount") == 5)
{
    if (_TAG("AComErr") == 0)//通信正常时，将hh[I]中的数据写入自定义数据
中。
    {
        for (i = 0 ; i < 26 ; i = i + 1 )//共有25个数据，前20个为模拟量数据，后
四个为DI数据，最后为DO数据。
        {
            if ( i < 20 )//前为20个模拟量数据
            {
                _TAG("AI30")[i] = denorm(hh[i]);
            }
            if((i >= 20) AND (i < 26))//后面为6个组合的DI及DO数据，数据
序号连续排布。
            {
                k = i - 20;
                Templong = setint(0,hh[i],0);
                for (j = 0; j < 16; j = j + 1)
                {
                    _TAG("DI")[k*16+j] = getbit(Templong,j);
                }
            }
        }
    }
}

```

```

//解开数据，将开关量数据放于自定义数据中
    }
    }
}
//模拟信号显示
for (i = 0; i < 8; i = i + 1)
{
    _TAG("FK21002")[i] = _TAG("AI1")[i];
    shuju = _TAG("AI1")[i];
    if ((shuju >= 0.95f) AND (shuju <= 0.05f))
    {
        _TAG("FK21002")[i] = _TAG("AI1")[i + 10];
    }
}
for (i = 0; i < 8; i = i + 1)
{
    _TAG("TK21019")[i] = _TAG("AI20")[i];
    shuju = _TAG("AI20")[i];
    if ((shuju >= 0.95f) AND (shuju <= 0.05f))
    {
        _TAG("TK21019")[i] = _TAG("AI20")[i + 10];
    }
}
_TAG("TK21043") = _TAG("AI40");
shuju = _TAG("AI42");
if ((shuju >= 0.95f) AND (shuju <= 0.05f))
{
    _TAG("TK21043") = _TAG("AI43");
}

}
//读开关量输入，连续读两次
// if ((_TAG("ComCount") == 3) OR (_TAG("ComCount") == 4) )
// {
//     setdelaytime(100);
//     Temp = readdata(0,2050,5,Dlbuf);
//     if (Temp == -1)
//     {
//         Temp = 5;
//     }
//     _TAG("DIComErr") = Temp;
// }

```

```

// if (_TAG("ComCount") == 5)
// {
//     if (_TAG("DIComErr") == 0)
//     {
//         for (i = 0; i < 5; i = i + 1)
//         {
//             Templong = setint(0,Dlbuf[i],0);
//
//             for (j = 0; j < 16; j = j + 1)
//             {
//                 _TAG("DI")[i*16+j] = getbit(Templong,j);
//             }
//         }
//     }
// }

// //读开关量输出，连续读两次
// if ((_TAG("ComCount") == 6) OR (_TAG("ComCount") == 7) )
// {
//     setdelaytime(300);
//     Temp = readdata(0,2055,1,DObuf);
//     if (Temp == -1)
//     {
//         Temp = 5;
//     }
//     _TAG("DOComErr") = Temp;
// }
// if (_TAG("ComCount") == 8)
// {
//     if(_TAG("DOComErr") == 0)
//     {
//         for (i = 0; i < 1; i = i + 1)
//         {
//             Templong = setint(0,DObuf[i],0);
//             for (j = 0; j < 16; j = j + 1)
//             {
//                 _TAG("DO17")[i*16+j] == getbit(Templong,j);
//             }
//         }
//     }
// }

//读SOE数据，分两批读，每批读两次
if (_TAG("ComCount") == 6 OR _TAG("ComCount") == 7)
{

```

```

        setdelaytime(300);
        Temp = readdata(0,2100,30,SOEbuf);
        for (i = 0 ; i < 20 ; i = i + 1 )           //测试SOE记录
        {
            _TAG("SOE")[i] = SOEbuf[i];
        }
        if (Temp == -1)
        {
            Temp = 5;
        }
    }
    if (_TAG("ComCount") == 8 OR _TAG("ComCount") == 9)
    {
        setdelaytime(300);
        Temp = readdata(0,2130,11,SOEbuf);
        for (i = 0 ; i < 11 ; i = i + 1 )           //测试SOE记录
        {
            _TAG("SOE30")[i] = SOEbuf[i];
        }
        if (Temp == -1)
        {
            Temp = 5;
        }
        _TAG("SOEComErr") = Temp;
    }

    //_TAG("SOE") = _TAG("SOE1_1"); //(* 测试*)
    //_TAG("SOE1") = _TAG("SOE1_2"); //(*测试*)
    //_TAG("SOE2") = _TAG("SOE2_1");测试
    //_TAG("SOE3") = _TAG("SOE2_2");测试
    //判断SOE区状态标志，是否有SOE事件产生
    if (_TAG("ComCount") == 10 )
    {
        //    if (_TAG("SOEComErr") == 0)//有SOE事件时的处理,将0写入数据区。
        //    {
            if (_TAG("SOE40") >= 65533)//产生SOE事件时为FFFF ( 既65535 )
            {
                //    SOEbuf[40] = 0;
                //    SOEreset[0] = 0;
                //    setdelaytime(100);
                //    Temp = writedata(0,2140,1,send); //清SOE记录
                _TAG("LIANSUO") = ON;
            }
        }
    }

```



```

else
{
    _TAG("LIANSUO") = OFF;
}
for (i = 0; i < 20; i = i + 1) //事件处理
{
    //读SOE第一个字节，分为秒及分
    templong1 = itol(_TAG("SOE")[i*2]);
    templong3 = andlong(templong1,15); //秒的个位
    templong5 = andlong(templong1,240) / 16; //秒的十位
    _TAG("SOE1_M")[i] = ltoi(templong5) * 10 + ltoi(templong3);
//SOE时间的秒

    _TAG("SOE1_M")[i] = ltoi(templong3); //SOE时间的秒
    templong3 = andlong(templong1,3840) / 256; //分的个位
    templong5 = andlong(templong1,61440) / 4096; //分的十位
    _TAG("SOE1_F")[i] = ltoi(templong5) * 10 + ltoi(templong3);
//SOE时间的分

    //读SOE第二个字节，分为毫秒及位号
    templong2 = itol(_TAG("SOE")[i*2 + 1]);
    templong4 = andlong(templong2,960) / 64; //十毫秒的个位
    templong6 = andlong(templong2,15360) / 1024; //十毫秒的十位
    templong8 = andlong(templong2,49152) / 16384; //十毫秒的百位
    //SOE时间的毫秒
    _TAG("SOE1_MS")[i] = ltoi(templong8) * 1000 + ltoi(templong6)
* 100 + ltoi(templong4) * 10;
    templong4 = andlong(templong2,15);
    templong6 = andlong(templong2,48) / 16;
    _TAG("SOE1_W")[i] = ltoi(templong6) * 10 + ltoi(templong4);
//SOE事件的动作原因

    //取事件发生的秒
    //Temp = SOEbuf[2*i];
    _TAG("SOE1_s")[4*i] = Temp AND 256;
    //取事件发生的分
    //Temp = SOEbuf[2*i] AND -32512; //(FF00)
    _TAG("SOE1_m")[4*i+1] = Temp/256;
    //取事件发生的毫秒
    //Temp = SOEbuf[2*i+1];
    //Temp = Temp AND -32704; //(FFC0)
    _TAG("SOE1_ms")[4*i+2] = Temp/64;
    //取事件发生的位号
    //Temp = SOEbuf[2*i+1];

```

```

        _TAG("SOE1")[4*i+3] = Temp AND 64;
    }

//    }
}
if (_TAG("START1") != OFF) //复位
{
    for (i = 0; i < 40; i = i + 1)
    {
        _TAG("SOE")[i] = 0;
    }
}
_TAG("OLDLIANSUO") = _TAG("LIANSUO");
//PLC通信程序完成

}
//*****

//*****

void BAOJING()//报警程序
{
    //丙烯流量报警
    if (_TAG("DI") == ON)
    {
        if (_TAG("FK21002") < _TAG("FK21002L1"))
        {
            _TAG("FK21002L1A") = ON;
        }
        else
        {
            _TAG("FK21002L1A") = OFF;
        }
    }
    else
    {
        _TAG("FK21002L1A") = OFF;
    }
    if (_TAG("DI2") == ON)
    {
        if (_TAG("FK21002") < _TAG("FK21002L7"))
        {
            _TAG("FK21002L1A") = ON;
        }
    }
}

```

```
        else
        {
            _TAG("FK21002L1A") = OFF;
        }
    }
else
{
    _TAG("FK21002L1A") = OFF;
}
//空气流量报警
if(_TAG("DI4") == ON)
{
    if (_TAG("FK21007") > _TAG("FK21007H1"))
    {
        _TAG("FK21007H1A") = ON;
    }
    else
    {
        _TAG("FK21007H1A") = OFF;
    }
}
else
{
    _TAG("FK21007H1A") = OFF;
}
if(_TAG("DI6") == ON)
{
    if (_TAG("FK21007") > _TAG("FK21007H7"))
    {
        _TAG("FK21007H1A") = ON;
    }
    else
    {
        _TAG("FK21007H1A") = OFF;
    }
}
else
{
    _TAG("FK21007H1A") = OFF;
}
//MP流量报警
if(_TAG("DI8") == ON)
{
    if (_TAG("FK21011") < _TAG("FK21011L1"))
```

```
        {
            _TAG("FK21011L1A") = ON;
        }
        else
        {
            _TAG("FK21011L1A") = OFF;
        }
    }
    else
    {
        _TAG("FK21011L1A") = OFF;
    }
    if(_TAG("DI10") == ON)
    {
        if (_TAG("FK21011") < _TAG("FK21011L7"))
        {
            _TAG("FK21011L1A") = ON;
        }
        else
        {
            _TAG("FK21011L1A") = OFF;
        }
    }
    else
    {
        _TAG("FK21011L1A") = OFF;
    }
}
//*****

main()
{
    //*****
    //PLC通信
    PLCTONGXUN();
    //*****

    //*****
    //报警
    BAOJING();
    //*****
}
```

//SOE数据为两字节的数据，共16位，其中第一个字节中前8个位为分数数据（前4位为分的十位，后4位为分的个位），后8位为秒数据（前4位为秒的十位，后4位为秒的个位）  
 //第二个字节中前10个位为毫秒数据（前2位为毫秒的百位，中4位为分的十位，后4位为分的个位），后6位为SOE事件触发条件（前2位为位号的十位，后4位为位号的个位）  
 //主控制卡控制周期为0.5秒  
 //AI，DI，DO 数据为 ASCII 码，SOE 为 BCD 码

### 5.6.3 自定义：用户通信协议开放（波特率 19200bps）

#### ➤ 网关卡与 S7-200 PLC 通信

背景：此项目中网关卡主要是与水泥计量系统的皮带称进行相关数据互相传送。其外部设备为西门子的 S7-200 PLC，接口为 RS-485，协议为自定义协议，编码方式为 BCD 码。采用主从问答方式，SUPCON WebField 系列控制系统为主，PLC 为从；通信速率：9,600 bps，8 位数据位，无校验。

读命令：AA 0 2 BB（4 字节）= 4 个 INT 型字符；  
 写命令：AA 1 3 1 99 99 BB（7 个字节）；  
 第三方 PLC 响应为：  
 CC 1 3F 99 99 99 99 99 99 99 99 DD（12 字节）；  
 状态为：  
 bit7 始终为 0  
 bit6 起停状态，1 起动；0 停止  
 bit5 备妥状态，1 已备妥；0 未备妥  
 bit4 就地中控状态，1 中控；0 就地  
 bit3 荷载上限报警状态，1 报警；0 正常  
 bit2 荷载下限报警状态，1 报警；0 正常  
 bit1 速度上限报警状态，1 报警；0 正常  
 bit0 速度下限报警状态，1 报警；0 正常  
 称号范围为：1~4  
 设定值范围为：0.00~99.99  
 瞬时量范围：0.00~99.99  
 累计量范围为：0.00~999999.99

此次通信，由于西门子的 PLC 系统没有加 CP341 模块，两个系统又不能相互支持对方的协议，只能采用自定义协议。

硬件配置：1 块网关卡，1 个 ADAM 模块

互联方案：采用 RS485 接口方式，其拓扑结构为：

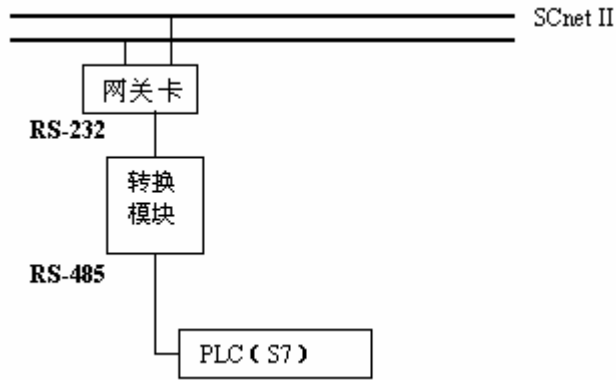


图 5-3 互联方案

SCX 语言程序如下：

```

bool FAN[12];
bool aa[8];
int i;
int phase;
int error;
bool mid01;
int send1[4];
int send2[7];

int buf[12];
int buf1[12];
int buf2[12];
int buf3[12];
long num1[4];
long num2[4];
long y,pdo,pdo1;
long mod[3],state2;
float bb,cc,dd,ee,ff;
int b,c,d,e,f,g,h,s,k;
bool a,PD01,PD02,PD03,PD04,LSTPD01,LSTPD02,LSTPD03,LSTPD04;
sfloat LSTVI1,LSTVI2,LSTVI3,LSTVI4;
main()
{
  //*****报表相关参数*****为报表产生标准1分钟
  if(timers[50]>=60)
  {
    _TAG("DSM-1") =ON;
    timers[50]=0;
  }
  else
  {
    _TAG("DSM-1")=OFF;
  }
}

```

```

    /***报表相关参数完***/
    state2=getmsg(2,0);

    PD01=getbit(state2,0);
    PD02=getbit(state2,1);
    PD03=getbit(state2,2);
    PD04=getbit(state2,3);
    if((phase<0) OR (phase > 7))
    {
        phase = 0;
    }
    //读通信数据,
    _TAG("add") = _TAG("add") + 1; //指示自定义运行状态
    setcomm(9600,0);                //读取速率

    if(phase == 0 AND error ==0 )
    {
        setdelaytime(200);           //等待时间
        send1[0] = 170;
        send1[1] = 0;
        send1[2] = 2;
        send1[3] = 187;

        _TAG("return0") = read(4,12,send1,buf); //readcoil
        _TAG("aaa") =OFF;

        if(_TAG("return0")==0 )
        {
            if(buf[0] == 204 AND buf[11]==221)
            {
                if( buf[1] == 2)
                {
                    //for(i=0;i<2;i=i+1)
                    //{
                        b = buf[2];
                        num1[0] =itol(b);
                    //}
                    g_msg[1]=num1[0];
                    //sendmsg(1);
                    for (i=0;i<8;i=i+1)
                    {
                        aa[i]=getbit(num1[0],i);
                    }
                    b=andint(buf[3],240)/16;
                    c=andint(buf[3],15);
                }
            }
        }
    }

```

```

        d=andint(buf[4],240)/16;
        e=andint(buf[4],15);

bb=(i tof(b)*10.0+i tof(c)+i tof(d)*0.1+i tof(e)*0.01)/3.0;
    if (aa[4]==OFF)
    {
        _TAG("SET011")=f tosf(bb);
    }
    b=andint(buf[5],240)/16;
    c=andint(buf[5],15);
    d=andint(buf[6],240)/16;
    e=andint(buf[6],15);

bb=(i tof(b)*10.0+i tof(c)+i tof(d)*0.1+i tof(e)*0.01)/3.0;

        _TAG("VI_011")=f tosf(bb);
        b=andint(buf[7],240)/16;
        c=andint(buf[7],15);
        d=andint(buf[8],240)/16;
        e=andint(buf[8],15);
        f=andint(buf[9],240)/16;
        g=andint(buf[9],15);
        h=andint(buf[10],240)/16;
        s=andint(buf[10],15);

bb=i tof(b)/0.00001+i tof(c)/0.0001+i tof(d)/0.001+i tof(e)*100.0+i tof(f)*10.0;
        bb=bb+i tof(g)+i tof(h)*0.1+i tof(s)*0.01;
        _TAG("ACCU012")=ConvertToAccum(bb);
    }
    }
else
    {
        _TAG("return0") = 1;
    }
}

_TAG("STA-0141")= aa[0];
_TAG("STA-0131")= aa[1];
_TAG("STA-0121")= aa[2];
_TAG("STA-0111")= aa[3];
_TAG("P-01AM1") = aa[4];
_TAG("P-01RD1") = aa[5];
_TAG("P-01RN1") = aa[6];

if (_TAG("return0")==0)
{

```



```

    }
    else
    {
        error = 2;
    }
}
if(phase == 1 AND error ==0 )
{
    //有外操作时改变暂存区数据
    setdelaytime(200);           //等待时间

    if(_TAG("P-01AM1")==ON)
    {
        a=PD01; //启停命令编码
        y=setbit(0,a,0);
        //设定值编码
        cc=sftof(_TAG("SET011"))*3.0; //100.0为量程,半浮点数转成浮点
数
        dd=cc/10.0; //取10位上的数
        b=ftoi(dd-0.5);
        c=ftoi(cc-0.5);
        d=c-b*10; //取个位上的数
        ee=(cc-itoi(c))*100.0; //取小数点后一位数
        e=ftoi(ee-0.5);
        ff=ee/10.0; //取小数后第二位数
        f=ftoi(ff-0.5);
        g=e-f*10;
        pdo=setint(0,g,0); //写0.01位
        mod[0]=itoi(f); //写0.1位
        for(i=0; i<4; i=i+1)
        {
            aa[i]=getbit(mod[0],i);
        }
        for(i=0; i<4; i=i+1)
        {
            k=i+4;
            a=aa[i];
            pdo=setbit(pdo,a,k);
        }
        pdo1=setint(0,d,0); //写个位

        mod[1]=setint(0,b,0); //写十位
        for(i=0; i<4; i=i+1)
        {
            aa[i]=getbit(mod[1],i);
        }
        for(i=0; i<4; i=i+1)
        {

```

```

        k=i+4;
        a=aa[i];
        pdo1=setbit(pdo1,a,k);
    }
    b=getint(pdo1,0);
    c=getint(pdo,0);
    //写命令
    send2[0]=170;
    send2[1]=1;
    send2[2]=2;
    send2[3]=ltoi(y);
    send2[4]=b;
    send2[5]=c;
    send2[6]=187;
//    wait(100);
    _TAG("return0") = read(7,12,send2,buf);    //readcoil
    if(_TAG("return0")==0)
    {
        error=0;
    }
    else
    {
        error=2;
    }
    if(buf[0]!=204 AND buf[11]!=221)
    {
        _TAG("return0")=1;
    }
}
}

```

//熟料通信

```

if(phase == 2 AND error ==0 )
{
    setdelaytime(200);                //等待时间

    send1[0] = 170;
    send1[1] = 0;
    send1[2] = 1;
    send1[3] = 187;

    _TAG("return0") = read(4,12,send1,buf);    //readcoil
    if(_TAG("return0")==0 )
    {
        if(buf[0] == 204 AND buf[11]==221)
        {

```

```

        if( buf[1] == 1)
        {
            //_TAG("read1")=buf[3];
            //_TAG("read2")=buf[4];
            //for(i=0;i<2;i=i+1)
            //{
                b = buf[2];
                num1[0] =i toI(b);
            //}
            g_msg[0]=num1[0];
            //sendmsg(1);
            for (i=0;i<8;i=i+1)
            {
                aa[i]=getbit(num1[0],i);
            }
            b=andint(buf[3],240)/16;
            c=andint(buf[3],15);
            d=andint(buf[4],240)/16;
            e=andint(buf[4],15);

            bb=(i tof(b)*10.0+i tof(c)+i tof(d)*0.1+i tof(e)*0.01)/40.0;
            if(aa[4]==OFF)
            {
                _TAG("SET031")=f tosf(bb);
            }
            b=andint(buf[5],240)/16;
            c=andint(buf[5],15);
            d=andint(buf[6],240)/16;
            e=andint(buf[6],15);

            bb=(i tof(b)*10.0+i tof(c)+i tof(d)*0.1+i tof(e)*0.01)/40.0;

            _TAG("VI_031")=f tosf(bb);
            b=andint(buf[7],240)/16;
            c=andint(buf[7],15);
            d=andint(buf[8],240)/16;
            e=andint(buf[8],15);
            f=andint(buf[9],240)/16;
            g=andint(buf[9],15);
            h=andint(buf[10],240)/16;
            s=andint(buf[10],15);

            bb=i tof(b)*100000.0+i tof(c)*10000.0+i tof(d)*1000.0+i tof(e)*100.0+i tof(f)*10
            .0;

            bb=bb+i tof(g)+i tof(h)*0.1+i tof(s)*0.01;
            _TAG("ACCU032")=ConvertToAccum(bb);
        }
    }

```

```

        else
        {
            _TAG("return0") = 1;
        }
    }

    _TAG("STA-0341")= aa[0];
    _TAG("STA-0331")= aa[1];
    _TAG("STA-0321")= aa[2];
    _TAG("STA-0311")= aa[3];
    _TAG("P-03AM1") = aa[4];
    _TAG("P-03RD1") = aa[5];
    _TAG("P-03RN1") = aa[6];

    if (_TAG("return0")==0)
    {

    }
    else
    {
        error = 2;
    }
}

if(phase == 3 AND error ==0 )//(PD02 !=LSTPD02 OR _TAG("SET031")!=LSTV12))

{
    //有外操作时改变暂存区数据
    setdelaytime(200);           //等待时间

    if(_TAG("P-03AM1")==ON)
    {
        a=PD02;//启停命令编码
        y=setbit(0,a,0);
        //设定值编码
        cc=sftof(_TAG("SET031"))*40.0;//100.0为量程，半浮点数转成浮
        点数

        dd=cc/10.0;//取10位上的数
        b=ftoi(dd-0.5);
        c=ftoi(cc-0.5);
        d=c-b*10;//取个位上的数
        ee=(cc-itoi(c))*100.0;//取小数点后一位数
        e=ftoi(ee-0.5);
        ff=ee/10.0;//取小数后第二位数
        f=ftoi(ff-0.5);
        g=e-f*10;
        pdo=setint(0,g,0);//写0.01位
    }
}

```

```

mod[0]=itol(f); //写0.1位
for(i=0; i<4; i=i+1)
{
    aa[i]=getbit(mod[0], i);
}
for(i=0; i<4; i=i+1)
{
    k=i+4;
    a=aa[i];
    pdo=setbit(pdo, a, k);
}
pdo1=setint(0, d, 0); //写个位

mod[1]=setint(0, b, 0); //写十位
for(i=0; i<4; i=i+1)
{
    aa[i]=getbit(mod[1], i);
}
for(i=0; i<4; i=i+1)
{
    k=i+4;
    a=aa[i];
    pdo1=setbit(pdo1, a, k);
}
b=getint(pdo1, 0);
c=getint(pdo, 0);
//写命令
send2[0]=170;
send2[1]=1;
send2[2]=1;
send2[3]=ltoi(y);
send2[4]=b;
send2[5]=c;
send2[6]=187;
// wait(100);
_TAG("return0") = read(7, 12, send2, buf); //readcoil
if(_TAG("return0")==0)
{
    error=0;
}
else
{
    error=2;
}
if(buf[0]!=204 AND buf[11]!=221)
{
    _TAG("return0")=1;
}

```

```

    }
    // LSTVI2=_TAG("SET031");
    // LSTPD02=PD02;
}
//石灰石
if(phase == 4 AND error ==0 )
{
    setdelaytime(200);           //等待时间

    send1[0] = 170;
    send1[1] = 0;
    send1[2] = 3;
    send1[3] = 187;

    _TAG("return0") = read(4,12,send1,buf);    //readcoil
    if(_TAG("return0")==0 )
    {
        if(buf[0] == 204 AND buf[11]==221)
        {
            if( buf[1] == 3)
            {
                //_TAG("read1")=buf[3];
                //_TAG("read2")=buf[4];
                //for(i=0;i<2;i=i+1)
                //{
                    b = buf[2];
                    num1[0] =itol(b);
                //}
                g_msg[2]=num1[0];
                //sendmsg(1);
                for (i=0;i<8;i=i+1)
                {
                    aa[i]=getbit(num1[0],i);
                }
                b=andint(buf[3],240)/16;
                c=andint(buf[3],15);
                d=andint(buf[4],240)/16;
                e=andint(buf[4],15);

                bb=(itof(b)*10.0+itof(c)+itof(d)*0.1+itof(e)*0.01)/4.0;
                if(aa[4]==OFF)
                {
                    _TAG("SET021")=ftosf(bb);
                }
                b=andint(buf[5],240)/16;
                c=andint(buf[5],15);
                d=andint(buf[6],240)/16;
                e=andint(buf[6],15);
            }
        }
    }
}

```

```

bb=(i tof(b)*10.0+i tof(c)+i tof(d)*0.1+i tof(e)*0.01)/4.0;

    _TAG("VI_021")=f tof(bb);
    b=andint(buf[7],240)/16;
    c=andint(buf[7],15);
    d=andint(buf[8],240)/16;
    e=andint(buf[8],15);
    f=andint(buf[9],240)/16;
    g=andint(buf[9],15);
    h=andint(buf[10],240)/16;
    s=andint(buf[10],15);

bb=i tof(b)/0.00001+i tof(c)/0.0001+i tof(d)/0.001+i tof(e)*100.0+i tof(f)*10.0;
    bb=bb+i tof(g)+i tof(h)*0.1+i tof(s)*0.01;
    _TAG("ACCU022")=ConvertToAccum(bb);
}
}
else
{
    _TAG("return0") = 1;
}
}

_TAG("STA-0241")= aa[0];
_TAG("STA-0231")= aa[1];
_TAG("STA-0221")= aa[2];
_TAG("STA-0211")= aa[3];
_TAG("P-02AM1") = aa[4];
_TAG("P-02RD1") = aa[5];
_TAG("P-02RN1") = aa[6];

if (_TAG("return0")==0)
{

}
else
{
    error = 2;
}
}
if(phase == 5 AND error ==0 )
{
    //有外操作时改变暂存区数据
    setdelaytime(200);                //等待时间

    if(_TAG("P-02AM1")==ON)

```

```

{
    a=PD03; //启停命令编码
    y=setbit(0,a,0);
    //设定值编码
    cc=sftof(_TAG("SET021"))*4.0; //4.0为量程,半浮点数转成浮点数
    dd=cc/10.0; //取10位上的数
    b=ftoi(dd-0.5);
    c=ftoi(cc-0.5);
    d=c-b*10; //取个位上的数
    ee=(cc-itoi(c))*100.0; //取小数点后一位数
    e=ftoi(ee-0.5);
    ff=ee/10.0; //取小数后第二位数
    f=ftoi(ff-0.5);
    g=e-f*10;
    pdo=setint(0,g,0); //写0.01位
    mod[0]=itoi(f); //写0.1位
    for(i=0; i<4; i=i+1)
    {
        aa[i]=getbit(mod[0],i);
    }
    for(i=0; i<4; i=i+1)
    {
        k=i+4;
        a=aa[i];
        pdo=setbit(pdo,a,k);
    }
    pdo1=setint(0,d,0); //写个位

    mod[1]=setint(0,b,0); //写十位
    for(i=0; i<4; i=i+1)
    {
        aa[i]=getbit(mod[1],i);
    }
    for(i=0; i<4; i=i+1)
    {
        k=i+4;
        a=aa[i];
        pdo1=setbit(pdo1,a,k);
    }
    b=getint(pdo1,0);
    c=getint(pdo,0);
    //写写命令
    send2[0]=170;
    send2[1]=1;
    send2[2]=3;
    send2[3]=ltoi(y);
    send2[4]=b;
    send2[5]=c;
}

```



```

        send2[6]=187;
    //    wait(100);
    _TAG("return0") = read(7,12,send2,buf);    //readcoil
    if(_TAG("return0")==0)
    {
        error=0;
    }
    else
    {
        error=2;
    }
    if(buf[0]!=204 AND buf[11]!=221)
    {
        _TAG("return0")=1;
    }
}
}
//煤灰计量通信
if(phase == 6 AND error ==0 )
{
    setdelaytime(200);                //等待时间

    send1[0] = 170;
    send1[1] = 0;
    send1[2] = 4;
    send1[3] = 187;

    _TAG("return0") = read(4,12,send1,buf);    //readcoil
    if(_TAG("return0")==0 )
    {
        if(buf[0] == 204 AND buf[11]==221)
        {
            if( buf[1] == 4)
            {
                //_TAG("read1")=buf[3];
                //_TAG("read2")=buf[4];
                //for(i=0;i<2;i=i+1)
                //{
                    b = buf[2];
                    num1[0] =itol(b);
                //}
                g_msg[3]=num1[0];
                //sendmsg(1);
                for (i=0;i<8;i=i+1)
                {
                    aa[i]=getbit(num1[0],i);
                }
                b=andint(buf[3],240)/16;
            }
        }
    }
}

```

```

        c=andint(buf[3],15);
        d=andint(buf[4],240)/16;
        e=andint(buf[4],15);

        bb=(itof(b)*10.0+itof(c)+itof(d)*0.1+itof(e)*0.01)/10.0;
        if(aa[4]==OFF)
        {
            _TAG("SET041")=ftosf(bb);
        }
        b=andint(buf[5],240)/16;
        c=andint(buf[5],15);
        d=andint(buf[6],240)/16;
        e=andint(buf[6],15);

        bb=(itof(b)*10.0+itof(c)+itof(d)*0.1+itof(e)*0.01)/10.0;

        _TAG("VI_041")=ftosf(bb);
        b=andint(buf[7],240)/16;
        c=andint(buf[7],15);
        d=andint(buf[8],240)/16;
        e=andint(buf[8],15);
        f=andint(buf[9],240)/16;
        g=andint(buf[9],15);
        h=andint(buf[10],240)/16;
        s=andint(buf[10],15);

        bb=itof(b)/0.00001+itof(c)/0.0001+itof(d)/0.001+itof(e)*100.0+itof(f)*10.0;
        bb=bb+itof(g)+itof(h)*0.1+itof(s)*0.01;
        _TAG("ACCU042")=ConvertToAccum(bb);
    }
}
else
{
    _TAG("return0") = 1;
}
}

_TAG("STA-0441")= aa[0];
_TAG("STA-0431")= aa[1];
_TAG("STA-0421")= aa[2];
_TAG("STA-0411")= aa[3];
_TAG("P-04AM1") = aa[4];
_TAG("P-04RD1") = aa[5];
_TAG("P-04RN1") = aa[6];

if (_TAG("return0")==0)
{

```

```

    }
    else
    {
        error = 2;
    }
}
if(phase == 7 AND error ==0 )
{
    //有外操作时改变暂存区数据
    setdelaytime(200);           //等待时间

    if(_TAG("P-03AM1")==ON)
    {
        a=PD04; //启停命令编码
        y=setbit(0,a,0);
        //设定值编码
        cc=sftof(_TAG("SET041"))*10.0; //100.0为量程，半浮点数转成浮
        点数

        dd=cc/10.0; //取10位上的数
        b=ftoi(dd-0.5);
        c=ftoi(cc-0.5);
        d=c-b*10; //取个位上的数
        ee=(cc-itoi(c))*100.0; //取小数点后一位数
        e=ftoi(ee-0.5);
        ff=ee/10.0; //取小数后第二位数
        f=ftoi(ff-0.5);
        g=e-f*10;
        pdo=setint(0,g,0); //写0.01位
        mod[0]=itoi(f); //写0.1位
        for(i=0; i<4; i=i+1)
        {
            aa[i]=getbit(mod[0],i);
        }
        for(i=0; i<4; i=i+1)
        {
            k=i+4;
            a=aa[i];
            pdo=setbit(pdo,a,k);
        }
        pdo1=setint(0,d,0); //写个位
        mod[1]=setint(0,b,0); //写十位
        for(i=0; i<4; i=i+1)
        {
            aa[i]=getbit(mod[1],i);
        }
        for(i=0; i<4; i=i+1)
        {

```

```

        k=i+4;
        a=aa[i];
        pdo1=setbit(pdo1,a,k);
    }
    b=getint(pdo1,0);
    c=getint(pdo,0);
    //写命令
    send2[0]=170;
    send2[1]=1;
    send2[2]=4;
    send2[3]=ltoi(y);
    send2[4]=b;
    send2[5]=c;
    send2[6]=187;
    //wait(100);
    _TAG("return0") = read(7,12,send2,buf);    //readcoil
    if(_TAG("return0")==0)
    {
        error=0;
    }
    else
    {
        error=2;
    }
    if(buf[0]!=204 AND buf[11]!=221)
    {
        _TAG("return0")=1;
    }
}

}

phase = phase +1;
if(error < 0 OR error >2)
{
    error = 0;
}
if(error >0)
{
    error = error -1;
}

sendmsg(4);

}

```

➤ 网关卡与三菱、华光 PLC 及单片机系统通信

将三菱的 PLC 一套、华光的 PLC 一套、以及单片机系统一套连接至我们的系统。其中，华光的 PLC 其协议格式已经提供。

硬件配置：1 块网关卡，1 个 ADAM 模块

互联方案：采用 RS485 接口方式

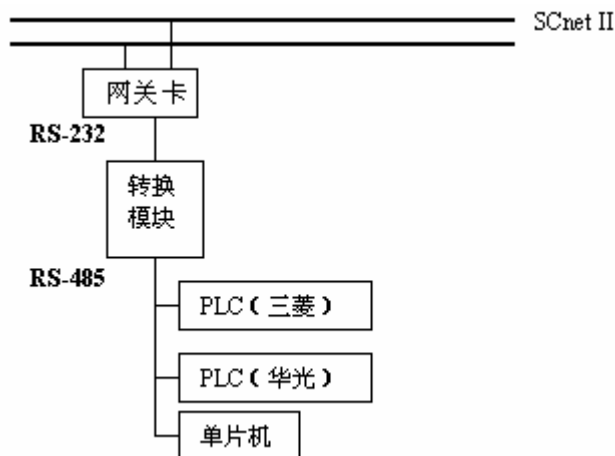


图 5-4 互联方案

SCX 语言程序如下：

```

////汉江过滤PLC通信
int SendBuf[20]; //发送缓冲区
int ReceBuf[100]; //接收缓冲区

int _dtoa(int Data); //1~16变为'0'~'9' 'A'~'F'
void dtoa(int Data,int Pos); //数字ASCII化,放到SendBuf中Pos位置
int _atod(int Data); // '0'~'9' 'A'~'F' 变为1~16
long atod(int Pos); //4字节还原

int CheckSum(int Length); //计算校验和,源串在SendBuf中,数据长度Length
int Mod(int a,int b); //取模

int Mod(int a,int b) //取模
{
    int temp;
    temp=a/b;
    temp=a-temp*b;
    return temp;
}

void dtoa(int Data,int Pos)
{ //数字ASCII化,放到SendBuf中Pos位置

```

```

    int temp;
    Data=Mod(Data,256);
    temp=Data/16;//取高4位
    SendBuf[Pos]=_dtoa(temp);
    temp=Data-(temp*16);//取低4位
    SendBuf[Pos+1]=_dtoa(temp);
}

int CheckSum(int Length)//计算校验和，源串在SendBuf中，数据长度Length
{//跳过首字(SOH)，因PLC首字不参与校验和的运算
    int i;
    int sum;
    sum=0;
    for(i=1;i<Length;i=i+1)
    {
        sum=sum+SendBuf[i];
    }
    sum=Mod(sum,256);
    return sum;
}

int _dtoa(int Data)//1~16变为'0'~'9' 'A'~'F'
{
    int temp;
    Data=Mod(Data,16);
    if(Data<10) //小于10时为0~9
    {
        temp=Data+'0';
        return temp;
    }
    temp=Data-10+'A'; //大于9时为A~F
    return temp;
}

int _atod(int Data)//'0'~'9' 'A'~'F' 变为1~16
{
    int r;
    r='9';
    if(Data>r)

```

```
{
    r=Data-'A'+10;
    return r;
}
r=Data-'0';
return r;
}

long atod(int Pos)//4字节还原
{
    long temp2;
    int intl, inth;
    Pos=Pos+5;
    intl=_atod(ReceBuf[Pos+3])+_atod(ReceBuf[Pos+2])*16;
    inth=_atod(ReceBuf[Pos+1])+_atod(ReceBuf[Pos])*16;
    temp2=setint(0,intl+inth*256,0);
    return temp2;
}

main()
{
    int r;
    long temp3;
    setcomm(9600,0);
    setdelaytime(400);
    SendBuf[0]=5;
    SendBuf[1]='0';
    SendBuf[2]='0';
    SendBuf[3]='F';
    SendBuf[4]='F';
    SendBuf[5]='W';
    SendBuf[6]='R';
    SendBuf[7]='0';
    SendBuf[8]='Y';
    SendBuf[9]='0';
    SendBuf[10]='0';
    SendBuf[11]='0';
    SendBuf[12]='0';
    SendBuf[13]='0';
```

```
SendBuf[14]='7';
dtoa(CheckSum( 15),15);
r=read(17,36,SendBuf,ReceBuf); //发送并接收
if(r!=0)
{
    _TAG("comm_plc1")=OFF;
    return;
}
_TAG("comm_plc1")=ON;

temp3=atod(0);
_TAG("y000")=getbit(temp3,0);
_TAG("y001")=getbit(temp3,1);
_TAG("y002")=getbit(temp3,2);
_TAG("y003")=getbit(temp3,3);
_TAG("y004")=getbit(temp3,4);
_TAG("y005")=getbit(temp3,5);
_TAG("y006")=getbit(temp3,6);
_TAG("y007")=getbit(temp3,7);

_TAG("y010")=getbit(temp3,8);
_TAG("y011")=getbit(temp3,9);
_TAG("y012")=getbit(temp3,10);
_TAG("y013")=getbit(temp3,11);
_TAG("y014")=getbit(temp3,12);
_TAG("y015")=getbit(temp3,13);
_TAG("y016")=getbit(temp3,14);
_TAG("y017")=getbit(temp3,15);

temp3=atod(4);
_TAG("y020")=getbit(temp3,0);
_TAG("y021")=getbit(temp3,1);
_TAG("y022")=getbit(temp3,2);
_TAG("y023")=getbit(temp3,3);
_TAG("y024")=getbit(temp3,4);
_TAG("y025")=getbit(temp3,5);
_TAG("y026")=getbit(temp3,6);
_TAG("y027")=getbit(temp3,7);
```



```
_TAG("y030")=getbit(temp3,8);  
_TAG("y031")=getbit(temp3,9);  
_TAG("y032")=getbit(temp3,10);  
_TAG("y033")=getbit(temp3,11);  
_TAG("y034")=getbit(temp3,12);  
_TAG("y035")=getbit(temp3,13);  
_TAG("y036")=getbit(temp3,14);  
_TAG("y037")=getbit(temp3,15);
```

```
temp3=atod(8);  
_TAG("y040")=getbit(temp3,0);  
_TAG("y041")=getbit(temp3,1);  
_TAG("y042")=getbit(temp3,2);  
_TAG("y043")=getbit(temp3,3);  
_TAG("y044")=getbit(temp3,4);  
_TAG("y045")=getbit(temp3,5);  
_TAG("y046")=getbit(temp3,6);  
_TAG("y047")=getbit(temp3,7);
```

```
_TAG("y050")=getbit(temp3,8);  
_TAG("y051")=getbit(temp3,9);  
_TAG("y052")=getbit(temp3,10);  
_TAG("y053")=getbit(temp3,11);  
_TAG("y054")=getbit(temp3,12);  
_TAG("y055")=getbit(temp3,13);  
_TAG("y056")=getbit(temp3,14);  
_TAG("y057")=getbit(temp3,15);
```

```
temp3=atod(12);  
_TAG("y060")=getbit(temp3,0);  
_TAG("y061")=getbit(temp3,1);  
_TAG("y062")=getbit(temp3,2);  
_TAG("y063")=getbit(temp3,3);  
_TAG("y064")=getbit(temp3,4);  
_TAG("y065")=getbit(temp3,5);  
_TAG("y066")=getbit(temp3,6);  
_TAG("y067")=getbit(temp3,7);
```

```
_TAG("y070")=getbit(temp3,8);
```

```
_TAG("y071")=getbit(temp3,9);  
_TAG("y072")=getbit(temp3,10);  
_TAG("y073")=getbit(temp3,11);  
_TAG("y074")=getbit(temp3,12);  
_TAG("y075")=getbit(temp3,13);  
_TAG("y076")=getbit(temp3,14);  
_TAG("y077")=getbit(temp3,15);
```

```
temp3=atod(16);  
_TAG("y100")=getbit(temp3,0);  
_TAG("y101")=getbit(temp3,1);  
_TAG("y102")=getbit(temp3,2);  
_TAG("y103")=getbit(temp3,3);  
_TAG("y104")=getbit(temp3,4);  
_TAG("y105")=getbit(temp3,5);  
_TAG("y106")=getbit(temp3,6);  
_TAG("y107")=getbit(temp3,7);
```

```
_TAG("y110")=getbit(temp3,8);  
_TAG("y111")=getbit(temp3,9);  
_TAG("y112")=getbit(temp3,10);  
_TAG("y113")=getbit(temp3,11);  
_TAG("y114")=getbit(temp3,12);  
_TAG("y115")=getbit(temp3,13);  
_TAG("y116")=getbit(temp3,14);  
_TAG("y117")=getbit(temp3,15);
```

```
temp3=atod(20);  
_TAG("y120")=getbit(temp3,0);  
_TAG("y121")=getbit(temp3,1);  
_TAG("y122")=getbit(temp3,2);  
_TAG("y123")=getbit(temp3,3);  
_TAG("y124")=getbit(temp3,4);  
_TAG("y125")=getbit(temp3,5);  
_TAG("y126")=getbit(temp3,6);  
_TAG("y127")=getbit(temp3,7);
```

```
_TAG("y130")=getbit(temp3,8);  
_TAG("y131")=getbit(temp3,9);
```

```

_TAG("y132")=getbit(temp3,10);
_TAG("y133")=getbit(temp3,11);
_TAG("y134")=getbit(temp3,12);
_TAG("y135")=getbit(temp3,13);
_TAG("y136")=getbit(temp3,14);
_TAG("y137")=getbit(temp3,15);

temp3=atod(24);
_TAG("y140")=getbit(temp3,0);
_TAG("y141")=getbit(temp3,1);

_TAG("y150")=getbit(temp3,8);
_TAG("y151")=getbit(temp3,9);
_TAG("y152")=getbit(temp3,10);
_TAG("y153")=getbit(temp3,11);
_TAG("y154")=getbit(temp3,12);
_TAG("y155")=getbit(temp3,13);
_TAG("y156")=getbit(temp3,14);
_TAG("y157")=getbit(temp3,15);

}
///Koyo PLC 通信
main()
{
    int r;
    long Data;
    int SendBuf[2];//发送
    int ReceBuf[200];//接收
    int InitFlag;

    setcomm(1200,0);
    setdelaytime(400);

    SendBuf[0]=1;
    SendBuf[1]=1;
    r=read(1,5,SendBuf,ReceBuf);
    if(r==0)
    {
        Data=setint(0,ReceBuf[2],0);
    }
    else
    {
        Data=0;
    }
}

```

```

    }
    _TAG("yn11")=getbit(Data,0);
    _TAG("yn12")=getbit(Data,1);
    _TAG("yn13")=getbit(Data,2);
    _TAG("yn14")=getbit(Data,3);
    _TAG("yn15")=getbit(Data,4);
    _TAG("yn16")=getbit(Data,5);
    _TAG("yn17")=getbit(Data,6);
    _TAG("yn18")=getbit(Data,7);

//*****
    SendBuf[0]=2;
    SendBuf[1]=2;
    r=read(1,5,SendBuf,ReceBuf);
    if(r==0)
    {
        Data=setint(0,ReceBuf[2],0);
    }
    else
    {
        Data=0;
    }

    _TAG("yn21")=getbit(Data,0);
    _TAG("yn22")=getbit(Data,1);
    _TAG("yn23")=getbit(Data,2);
    _TAG("yn24")=getbit(Data,3);
    _TAG("yn25")=getbit(Data,4);
    _TAG("yn26")=getbit(Data,5);
    _TAG("yn27")=getbit(Data,6);
    _TAG("yn28")=getbit(Data,7);

//*****
    SendBuf[0]=3;
    SendBuf[1]=3;
    r=read(1,5,SendBuf,ReceBuf);
    if(r==0)
    {
        Data=setint(0,ReceBuf[2],0);
    }
    else
    {
        Data=0;
    }

    _TAG("yn31")=getbit(Data,0);
    _TAG("yn32")=getbit(Data,1);
    _TAG("yn33")=getbit(Data,2);

```

```

        _TAG("yn34")=getbit(Data,3);
        _TAG("yn35")=getbit(Data,4);
        _TAG("yn36")=getbit(Data,5);
        _TAG("yn37")=getbit(Data,6);
        _TAG("yn38")=getbit(Data,7);
//*****

    SendBuf[0]=4;
    SendBuf[1]=4;
    r=read(1,5,SendBuf,ReceBuf);
    if(r==0)
    {
        Data=setint(0,ReceBuf[2],0);
    }
    else
    {
        Data=0;
    }

    _TAG("yn41")=getbit(Data,0);
    _TAG("yn42")=getbit(Data,1);
    _TAG("yn43")=getbit(Data,2);
    _TAG("yn44")=getbit(Data,3);
    _TAG("yn45")=getbit(Data,4);
    _TAG("yn46")=getbit(Data,5);
    _TAG("yn47")=getbit(Data,6);
    _TAG("yn48")=getbit(Data,7);

}

```

### ➤ 网关卡与上海衡器总厂的 XK3190-A1+称重仪的通信

背景：此项目的网关卡是完成 SUPCON WebField 控制系统与上海衡器总厂的 XK3190-A1+称重仪的通信，采用自定义协议，编码方式为 ASCII 码。不采用主从通信方式，下位机不接受主从方式。通信速率：9600，8 位数据位，无校验。

编码格式：DCS 系统发送：地址（1），波特率（9600），方式（连续）；

对方返回数据格式：1 2 3 4 5 6 7 8 9 10 11 12

1：开始设备码；2：正负；3~8：数据位；9：小数点；10~11：校验；

12：接收标志；

硬件配置：1 块网关卡

互联方案：直接连接在 RS232 口上，其拓扑结构如下：

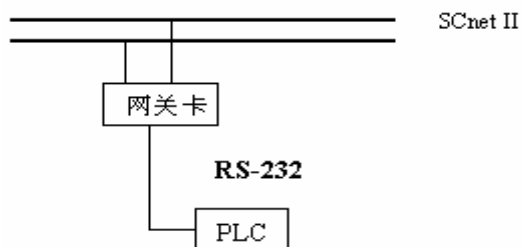


图 5-5 互联方案

SCX 语言程序如下：

```

main()
{
    int send[6];
    int buf[12];
    int a[12];
    int i,l;
    bool b[4];
    float f;
    _TAG("add")=_TAG("add")+1;
    setcomm(9600,0);    //9600波特率，无校验
    setdelaytime(300); //等待时间300毫秒
    send[0]=50;        //开始
    send[1]=49;        //1号站
    send[2]=68;        //读净重
    send[3]=48;        //校验
    send[4]=48;        //校验
    send[5]=51;        //结束

    _TAG("return")=read(4,12,send,buf);    //通用读命令
    for(i=0;i<9;i=i+1)
    {
        a[i]=buf[i];
        a[i]=andint(a[i],15);    //与掉高4位。
    }
    f=itof(a[2])*10000.0+itof(a[3])*1000.0+itof(a[4])*100.0+itof(a[5])*
    10.0+itof(a[6])+itof(a[7])*0.1;    //仪表只能保留一位
        //小数点
        //加和求重量。
    if(buf[0]==2)    //如果返回是2，就是正确的。
    {
        _TAG("S04")=f;    //送到显示
    }
    else
    {
        f=_TAG("S04");    //返回错误拒收。
    }
}

```

## 6 库函数

### 6.1 库函数目录

#### 1、 半浮点计算模块

函数名	返回值	功能说明
sfadd	sfloat	半浮点加法
sfsb	sfloat	半浮点减法
sfadd2	sfloat	半浮点加法
sfsb2	sfloat	半浮点减法
sfmul	sfloat	半浮点乘法
sfdiv	sfloat	半浮点除法
sfabs	sfloat	半浮点求绝对值
sfsqr	sfloat	半浮点求平方
sfsqrt	sfloat	半浮点求平方根

#### 2、 浮点计算模块

函数名	返回值	功能说明
fabs	float	求绝对值函数
floor	float	按输入浮点值返回最大整数
ceil	float	按输入浮点值返回最小整数
sin	float	正弦函数
asin	float	反正弦函数
sinh	float	工程正弦函数
cos	float	余弦函数
acos	float	反余弦函数
cosh	float	工程余弦函数
tan	float	正切函数
atan	float	反正切函数
atan2	float	反正切函数 2
tanh	float	工程正切函数
eval_poly	float	多项式求值
pow	float	幂函数
log	float	自然对数函数
log10	float	对数函数
exp	float	指数函数
sqrt	float	求平方根函数

#### 3、 辅助计算模块

函数名	返回值	功能说明
hal	bool	高报警模块
lal	bool	低报警模块
hlim	sfloat	高限模块
llim	sfloat	低限模块
hsel	sfloat	高选择器
lsel	sfloat	低选择器
fx	sfloat	一维折线表插值
getfx	float	取一维折线表的数据项
setfx	float	写一维折线表的数据项
fxxy	sfloat	二维折线表插值
getfxxy_x	float	取二维折线表的 X 轴的数据项
getfxxy_y	float	取二维折线表的 Y 轴的数据项
setfxxy_x	float	写二维折线表的 X 轴的数据项
setfxxy_y	float	写二维折线表的 Y 轴的数据项
ded	sfloat	纯滞后模块
led	sfloat	一阶超前模块
lag	sfloat	一阶滞后模块
ledlag	sfloat	一阶超前滞后模块
mav	sfloat	移动平均函数
msw	sfloat	多路切换模块
sw	sfloat	开关切换模块
vlm	sfloat	速度限制模块

#### 4、混合计算模块

函数名	返回值	功能说明
fkdivf	sfloat	半浮点整数除法
fkmulf	sfloat	半浮点整数乘法
fkmulf	int	半浮点整数乘法
kfdivk	int	整数的半浮点除法
kkdivf	sfloat	整数除法

#### 5、控制模块

函数名	返回值	功能说明
dgap2	sfloat	二位式二状态控制模块
dgap3	sfloat	二位式三状态控制模块
csc	sfloat	串级控制模块
bsc	sfloat	PID 控制模块
SteamComp	sfloat	温压补偿函数



AddAccum	structAccum	累积加
SubAccum	structAccum	累积减
ConvertAccum	float	累积转换成浮点
ConverToAccum	structAccum	浮点转换成累积
CompAccum	int	累积量比较
TotalAccum	structAccum	累积加半浮点
andint	int	整型与
andlong	long	长整型与
orint	int	整型或
orlong	long	长整型或
getbit	bool	取位值
getsfloat	sfloat	取半浮点
getint	int	取整型值
getfloat	float	取浮点值
setbit	long	设位值
setsfloat	long	设半浮点值
setint	long	设整型值
setfloat	long	设浮点值
getmsg	long	取控制站通讯数据
sendmsg	void	发数据

## 6、 类型转换模块

函数名	返回值	功能说明
norm	int	归一化函数
denorm	sfloat	反归一化函数
itosf	sfloat	整数转换到半浮点
sftoi	int	半浮点转换到整数
ftosf	sfloat	浮点转换到半浮点
sftof	float	半浮点转换到浮点
itof	float	整数转换到浮点
ftoi	int	浮点转换到整数

## 7、 常用语句

break  
continue  
define  
for  
goto  
if ... else  
main

```

return
timerm
timers
temerms
void
while
g_bsc
g_csc

```

## 6.2 库函数介绍

### 6.2.1 半浮点计算模块

#### 1、 sfadd (半浮点数加法)

语法：

```
y= sfadd ( x1,x2 );
```

参数 x1、 x2 的数据类型为半浮点数。

返回值 y 为 x1、 x2 的和值，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。当两数和值超出该范围时，则限制在此范围内，即

x1+x2<-7.9997 时，返回值为-7.9997，

x1+x2>7.9997 时，返回值为 7.9997。

举例：

```

...
sfloat a,b,c;
a = sfadd ( 7.0f,2.0f ) ;// a = 7.9997f
b = sfadd ( 0.35f,0.25f ) ;// b = 0.6f
c = sfadd ( -0.3f,0.9f ) ;// c = 0.6f

```

参考条目：

```
sfsb sub sfsmul sfsdiv sfabs sfsqr sfsqrt
```

#### 2、 sfsb (半浮点数减法)

语法：

```
y= sfsb ( x1,x2 );
```

被减数 x1，减数 x2 的数据类型为半浮点数。

返回值 y 为 x1 与 x2 的差值，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。当两数差值超出该范围时，则限制在此范围内，即

x1-x2<-7.9997 时，返回值为-7.9997，

x1-x2>7.9997 时，返回值为 7.9997。

举例：

```

...
sfloat a,b,c;

```

```
a = sfsb ( 0.9f,0.8f ) ;//a = 0.1f
b = sfsb ( 0.9f,-0.8f ) ;// b = 1.7f
c = sfsb ( 7.0f,-6.0f ) ;// c = 7.9997f
```

...

参考条目：

sfadd sfmul sfddiv sfabs sfsqr sfsqrt

### 3、 sfmul ( 半浮点数乘法 )

语法：

```
y= sfmul ( x1,x2 ) ;
```

参数 x1、 x2 的数据类型为半浮点数。

返回值 y 为 x1、 x2 的积值，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。当两数积值超出该范围时，则限制在此范围内，即

$x1 \times x2 < -7.9997$  时，返回值为-7.9997，

$x1 \times x2 > 7.9997$  时，返回值为 7.9997。

举例：

...

```
sfloat a,b,c;
```

```
a = sfmul ( 0.6f,0.5f ) ;// a = 0.3f
```

```
b = sfmul ( 2.0f,5.0f ) ;// b = 7.9997f
```

```
c = sfmul ( 2.0f,-3.0f ) ;// c = -6.0f
```

...

参考条目：

sfadd sfsb sfddiv sfabs sfsqr sfsqrt

### 4、 sfddiv ( 半浮点数除法 )

语法：

```
y= sfddiv ( x1,x2 ) ;
```

被除数 x1，除数 x2 的数据类型为半浮点数。

返回值 y 为  $x1/x2$  的商值，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。当两数商值超出该范围时，则限制在此范围内，即

$x1/x2 < -7.9997$  时，返回值为-7.9997，

$x1/x2 > 7.9997$  时，返回值为 7.9997。

说明：

当 0/0 时，返回值为 0；

当负数/0 时，返回值为-7.9997；

当正数/0 时，返回值为 7.9997。

举例：

...

```
sfloat a,b,c;
```

```
a = sfddiv ( 0.5f,0.1f ) ;// a = 5.0f
b = sfddiv ( 3.0f,0.1f ) ;// b = 7.9997f
c = sfddiv ( -3.0f,5.0f ) ;// c = - 0.6f
```

...

参考条目：

sfadd sfsb sfddiv sfabs sfsqr sfsqrt

## 5、 sfabs ( 半浮点数求绝对值 )

语法：

```
y= sfabs ( x ) ;
```

参数 x 的数据类型为半浮点数。

返回值 y 为 x 的绝对值，数据类型为半浮点数。

举例：

...

```
sfloat a,b;
```

```
a = sfabs ( 0.9f ) ;// a = 0.9f
```

```
b = sfabs ( -0.9f ) ;// b = 0.9f
```

...

参考条目：

sfadd sfsb sfddiv sfabs sfsqr sfsqrt

## 6、 sfsqr ( 半浮点数平方 )

语法：

```
y= sfsqr ( x ) ;
```

参数 x 的数据类型为半浮点数。

返回值 y 为 x 的平方值，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。当平方值超出该范围时，则限制在此范围内，即

$x^2 < -7.9997$  时，返回值为 -7.9997，

$x^2 > 7.9997$  时，返回值为 7.9997。

举例：

...

```
sfloat a,b;
```

```
a = sfsqr ( 4.0f ) ;// a = 7.9997f
```

```
b = sfsqr ( -0.8f ) ;// b = 0.64f
```

...

参考条目：

sfadd sfsb sfddiv sfabs sfsqr sfsqrt

## 7、 sfsqrt ( 半浮点数平方根 )

语法：

```
y= sfsqrt ( x );
```

参数 x 的数据类型为半浮点数。

返回值 y 为 x 的平方根值,数据类型为半浮点数。当  $x < 0.01$  时,进行小信号切除,返回值为 0。

举例：

```
...
```

```
sfloat a,b;
```

```
a = sfsqrt ( 0.81f );// a = 0.9f
```

```
b = sfsqrt ( 0.009f );// b = 0.0f
```

```
...
```

参考条目：

```
sfadd  sfsb  sfmul  sfdiv  sfabs  sfsqr
```

### 6.2.2 浮点计算模块

#### 1、 sin ( 正弦函数 )

语法：

```
y= sin ( x );
```

参数 x 为弧度值,数据类型为浮点数。求角度值的正弦 y,需要先转换为弧度值,然后再使用本公式求值。

如求  $45^\circ$  角的正弦,先转换为弧度值,约为 0.785,然后应用本公式求正弦值。

返回值 y 为 x 的正弦值,数据类型为浮点数。返回值范围： $-1 \leq y \leq 1$ 。

举例：

```
...
```

```
float a,b;
```

```
b=0.8;
```

```
a = sin ( b );// a = 0.7174
```

```
...
```

参考条目：

```
sinh  asin  cos  acos  cosh  tan  atan  atan2  tanh
```

#### 2、 asin ( 反正弦函数 )

语法：

```
y= asin ( x );
```

参数 x 的数据类型为浮点数,数值范围为 $-1.0 \leq x \leq 1.0$ ,当 x 超出该范围时,

$x < -1$  时,返回值为 $-\pi/2$ ,

$x > 1$  时,返回值为 $\pi/2$ 。

返回值 y 为 x 的反正弦值,数据类型为浮点数。返回值范围： $-\pi/2 \leq y \leq \pi/2$ 。

举例：

```
...
```

```
float a,b;
```

```
b=0.8;
```

```
a = asin ( b ) ;// a = 0.9273
```

...

参考条目：

sin sinh cos acos cosh tan atan atan2 tanh

### 3、 sinh ( 工程正弦函数 )

语法：

```
y= sinh ( x ) ;
```

参数 x 的数据类型为浮点数。

返回值 y 为 x 的工程正弦值，数据类型为浮点数。返回值为

$y=(e^x-e^{-x})/2$ 。

举例：

...

```
float a,b;
```

```
b=0.78;
```

```
a = sinh ( b ) ;// a = 0.8615
```

...

参考条目：

sin asin cos acos cosh tan atan atan2 tanh

### 4、 cos ( 余弦函数 )

语法：

```
y= cos ( x ) ;
```

参数 x 为弧度值，数据类型为浮点数。求角度值的正弦 y，需要先转换为弧度值，然后再使用本公式求值。

返回值 y 为 x 的余弦值，数据类型为浮点数。返回值范围： $-1 \leq y \leq 1$ 。

举例：

...

```
float a,b;
```

```
b=0.25;
```

```
a = cos ( b ) ;// a = 0.9689
```

...

参考条目：

sin asin sinh cosh acos tan atan atan2 tanh

### 5、 acos ( 反余弦函数 )

语法：

```
y= acos ( x ) ;
```

参数 x 的数据类型为浮点数，数值范围为 $-1.0 \leq x \leq 1.0$ 。

返回值 y 为 x 的反余弦值，数据类型为浮点数。返回值范围在 0 到 pi。当 x 超出该范围时：

$x < -1.0$ ，返回值为 3.1416;

$x > 1.0$  , 返回值为 0。

举例：

```
...
float a,b;
b=-0.6;
a = acos ( b ) ;// a = 2.2143
...
```

参考条目：

sin asin cos cosh tan atan atan2 tanh

## 6、 cosh ( 工程余弦函数 )

语法：

$y = \cosh ( x ) ;$

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  的工程余弦值，数据类型为浮点数。返回值为

$y = (e^x - e^{-x})/2$ 。

举例：

```
...
float a,b;
b=5.0;
a = cosh ( b ) ;// a = 74.2100
...
```

参考条目：

sin asin sinh cos acos tan atan atan2 tanh

## 7、 tan ( 正切函数 )

语法：

$y = \tan ( x ) ;$

参数  $x$  为弧度值，数据类型为浮点数。求角度值的正弦  $y$ ，需要先转换为弧度值，然后再使用本公式求值。

返回值  $y$  为  $x$  的正切值，数据类型为浮点数。

举例：

```
...
float a,b;
b=-1.47;
a = tan ( b ) ;// a = -9.8874
...
```

参考条目：

cos cosh acos sin sinh asin atan atan2 tanh

## 8、 atan ( 反正切函数 )

语法：

$y = \text{atan}(x);$

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  的反正切值，数据类型为浮点数。返回值范围： $-\pi/2$  到  $\pi/2$ 。

举例：

...

`float a,b;`

`b=1.0;`

`a = atan(b) ;// a = 0.7854`

...

参考条目：

`sin asin sinh cos acos cosh tan atan2 tanh`

## 9、atan2 (反正切函数)

语法：

$c = \text{atan2}(y,x);$

参数  $x, y$  的数据类型为浮点数，允许  $x = 0$ 。

返回值  $c$  为  $y/x$  的反正切值，数据类型为浮点数。返回值范围在  $-\pi$  到  $\pi$ 。相当于在复平面下  $x+yi$  的幅角。

举例：

...

`float a;`

`a = atan2(1.0,1.0) ;// a = 0.7854`

...

参考条目：

`sin asin sinh cos acos cosh tan atan tanh`

## 10、tanh (工程正切函数)

语法：

$y = \text{tanh}(x);$

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  的工程正切值，数据类型为浮点数。

举例：

...

`float a,b;`

`b=0.18;`

`a = tanh(b) ;// a = 0.17808`

...

参考条目：

`sin asin sinh cos acos cosh tan atan tanh2`

## 11、eval\_ploy (多项式加法函数)



语法：

`y= eval_ploy ( x,a[ ],n ) ;`

参数 `x` 为操作数，数据类型为浮点数；

参数 `a[ ]` 为参数数组，数据类型为浮点数；

参数 `n` 为数组元素个数，数据类型为整型。

返回值 `y` 为多项式的和值，数据类型为浮点数。返回值为

$$y=a_{n-1}X^{n-1}+a_{n-2}X^{n-2}+\dots+a_1X+a_0$$

举例：

...

`float a[4];`

`float y;`

`// y = 4x3+3x2+2x1+1`

`a[0] = 1.0;`

`a[1] = 2.0;`

`a[2] = 3.0;`

`a[3] = 4.0;`

`y = eval_poly ( 3.3,a,4 ) ;// y = 184.018`

...

参考条目：

`pow log log10 exp sqrt fabs`

## 12、 pow ( 幂函数 )

语法：

`z= pow ( x,y ) ;`

参数 `x` 为操作数，数据类型为浮点数。操作数 `x` 数值范围为 `x >= 0`。

参数 `y` 为指数，数据类型为浮点数。

返回值 `z` 为 `x` 的 `y` 幂次值，数据类型为浮点数。返回值为

$$z=x^y$$

当 `x < 0` 时，返回值为 0。

举例：

...

`float a,b,c;`

`a=5.4,b=7.8;`

`c = pow ( a,b ) ;// c = 516025.66`

...

参考条目：

`eval_ploy log log10 exp sqrt fabs`

## 13、 log ( 自然对数函数 )

语法：

`y= log ( x ) ;`

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  的自然对数值，数据类型为浮点数。返回值为

$$y = \log_e x$$

当  $x = 0$  时，返回值为 0。

举例：

...

float a,b;

b=4.75;

a = log ( b ) ;// a = 1.558145

...

参考条目：

eval\_ploy pow log10 exp sqrt fabs

#### 14、log10 (对数函数)

语法：

$y = \log_{10}(x)$  ;

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  以 10 为底时的对数值，数据类型为浮点数。返回值为

$$y = \log_{10} x$$

当  $x = 0$  时，返回值为 0。

举例：

...

float a,b;

b=0.6;

a = log10 ( b ) ;// a = - 0.22185

...

参考条目：

eval\_ploy pow log exp sqrt fabs

#### 15、exp (指数函数)

语法：

$y = \exp(x)$  ;

参数  $x$  的数据类型为浮点数。

返回值  $y$  为  $x$  的指数值，数据类型为浮点数。返回值为

$$y = e^x$$

举例：

...

float a,b;

b=0.235;

a = exp ( b ) ;// a = 1.26491

...

参考条目：

eval\_ploy pow log log10 sqrt fabs

## 16、sqrt (浮点数平方根)

语法：

y= sqrt ( x )；

参数 x 的数据类型为浮点数，数值范围为  $x \geq 0$ 。

返回值 y 为 x 的平方根值，数据类型为浮点数。

举例：

...

float a,b;

b=4.0;

a = sqrt ( b ) ;// a = 2.0

...

参考条目：

eval\_ploy pow log log10 exp fabs

## 17、fabs (浮点数求绝对值)

语法：

y= fabs ( x )；

参数 x 的数据类型为浮点数。

返回值 y 为 x 的绝对值，数据类型为浮点数。

举例：

...

float a,b;

b=-0.56;

a = fabs ( b ) ;// a = 0.56

...

参考条目：

eval\_ploy log log10 exp pow sqrt

### 6.2.3 辅助计算模块

#### 1、hal (高限报警)

语法：

y= hal ( x,hlimit,interval,n )；

参数 x 为输入值，数据类型为半浮点数；

参数 hlimit 为报警值，数据类型为半浮点数；

参数 interval 为磁环宽度，数据类型为半浮点数；

参数 n 为指定的序号，数据类型为整型，范围是 0~127。在同一程序中不允许用同一个序号调

用本函数。

返回值  $y$  的数据类型为布尔值。若输入值大于报警设定值，返回 ON，若小于报警设定值则返回 OFF 值。

报警采用磁环形式如图 6-1 示：

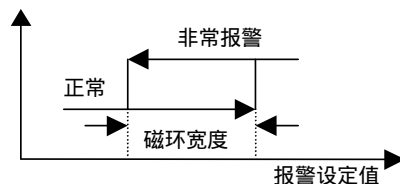


图 6-1 磁环报警

举例：

```
...
sfloat x;
bool c;
c = hal ( x, 0.9f, 0.05f, 0 );
```

...

参考条目：

lal

## 2、lal (低限报警)

语法：

```
y = lal ( x, llimit, interval, n );
```

参数  $x$  为输入值，数据类型为半浮点数；

参数  $llimit$  为报警值，数据类型为半浮点数；

参数  $interval$  为磁环宽度，数据类型为半浮点数；

参数  $n$  为指定的序号，数据类型为整型，范围是 0 ~ 31。在同一程序中不允许用同一个序号调用本函数。

返回值  $y$  的数据类型为布尔值。若输入值小于报警设定值，返回 ON，若大于报警设定值则返回 OFF 值。

报警采用磁环形式如图 6-2 示：

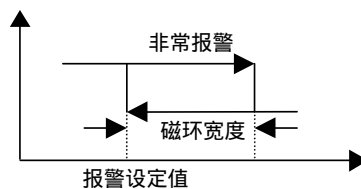


图 6-2 磁环报警

举例：

```
...
sfloat x;
bool b;
```

```
b = lal ( x,0.1f,0.05f,0 ) ;
```

```
...
```

参考条目：

```
hal
```

### 3、 hlim ( 高限器 )

语法：

```
y= hlim ( x,hlimit ) ;
```

参数 x 为输入值，数据类型为半浮点数。

参数 hlimit 为高限值，数据类型为半浮点数。

返回值 y 的数据类型为半浮点数。

若  $x < hlimit$ ，则返回 x；

若  $x \geq hlimit$ ，则返回 hlimit。

举例：

```
...
```

```
sfloat hlimit;
```

```
hlimit = 3.0f;
```

```
sfloat a, c,d;
```

```
a = 0.2f;
```

```
c = hlim ( a,hlimit ) ;// c = 0.2f
```

```
d = hlim ( 0.5f,0.1f ) ;// d = 0.1f
```

```
...
```

参考条目：

```
llim
```

### 4、 llim ( 低限器 )

语法：

```
y= llim ( x,llimit ) ;
```

参数 x 为输入值，数据类型为半浮点数。

参数 llimit 为低限值，数据类型为半浮点数。

返回值 y 的数据类型为半浮点数。

若  $x > llimit$ ，则返回 x；

若  $x \leq llimit$ ，则返回 llimit。

举例：

```
...
```

```
sfloat a,c;
```

```
sfloat llimit;
```

```
llimit = 0.5f;
```

```
a = 0.7f;
```

```
c = llim ( a,llimit ) ;// c = 0.7f
```

```
...
```

参考条目：

hlim

## 5、 hsel (高选择器)

语法：

```
y= hsel ( x1,x2 );
```

参数 x1、 x2 的数据类型为半浮点数。

返回值 y 为 x1 与 x2 中的较大值，数据类型为半浮点数。

举例：

```
...
sfloat a,b;
sfloat c;
a=0.3f;
b=0.2f;
c = hsel ( a,b ) ;// c = 0.3f
...
```

参考条目：

lsel

## 6、 lsel (低选择器)

语法：

```
y= lsel ( x1,x2 );
```

参数 x1、 x2 的数据类型为半浮点数。

返回值 y 为 x1 与 x2 中的较小值，数据类型为半浮点数。

举例：

```
...
sfloat a,b;
sfloat c;
a=0.7f;
b=0.5f;
c = lsel ( a,b ) ;// c = 0.5f
...
```

参考条目：

hsel

## 7、 fx (一维 16 段折线表折线插值计算)

语法：

```
y= fx ( input,n );
```

参数 input 为输入值，数据类型为半浮点数。

参数 n 为指定的折线表序号，范围是 0 ~ 15，数据类型为整型。

返回值 y 为用指定的一维 16 段折线表进行折线插值计算后的值，数据类型为半浮点数。

请参考折线表数据定义。

举例：

```
...
sfloat a,c;
c = fx ( a,3 );
```

...

参考条目：

getfx setfx

## 8、 getfx ( 取一维折线表的数据项 )

语法：

```
y= getfx ( index,n );
```

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0~31，数据类型为整型。

返回值 y 为指定的一维折线表的 index 号数据的值，数据类型为半浮点数。

请参考折线表数据定义。

举例：

```
...
sfloat c;
c = getfx ( 0,3 );//取 3 号折线表的 0 号数据
```

...

参考条目：

fx setfx

## 9、 setfx ( 写一维折线表的数据项 )

语法：

```
setfx ( d,index,n );
```

参数 d 为所写数据，数据类型为半浮点数。

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。

该函数没有返回值，只在函数内部进行一些操作。

请参考折线表数据定义。

举例：

```
...
sfloat d;
d=0.3f;
setfx ( d,0,10 );//把 10 号折线表的 0 号数据改为 d
```

...

参考条目：

fx    getfx

### 10、fxy (二维 10 段折线表折线插值计算)

语法：

y= fxy ( input,n );

参数 input 为输入值，数据类型为半浮点数。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。

返回值 y 为用指定的二维 10 段折线表进行折线插值计算后的值，数据类型为半浮点数。

请参考折线表数据定义。

举例：

...

sfloat a,c;

c = fxy ( a,1 );

...

参考条目：

getfxy\_x    getfxy\_y    setfxy\_x    setfxy\_y

### 11、getfxy\_x (取二维折线表的 x 轴的数据项)

语法：

y= getfxy\_x ( index,n );

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。

返回值 y 为指定的 n 号二维折线表的 x 轴的 index 号数据的值,数据类型为半浮点数。

请参考折线表数据定义。

举例：

...

sfloat c;

c = getfxy\_x ( 0,10 );//取 10 号二维折线表的 x 轴的 0 号数据的值

...

参考条目：

fxy    getfxy\_y    setfxy\_x    setfxy\_y

### 12、getfxy\_y (取二维折线表的 y 轴的数据项)

语法：

y= getfxy\_y ( index,n );

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。返回值 y 为指定的 n 号二维折线表的 y 轴的序号为 index 号数据的值,数据类型为半浮点数。

请参考折线表数据定义。



举例：

```
...
sfloat c;
c = getfxy_y ( 0,10 ) ;//10 号二维折线表的 y 轴的序号为 0 号数据
```

...

参考条目：

fxy getfxy\_x setfxy\_x setfxy\_y

### 13、setfxy\_x ( 写二维折线表的 x 轴的数据项 )

语法：

```
setfxy_x ( d,index,n ) ;
```

参数 d 为所写数据，数据类型为半浮点数。

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。

该函数没有返回值，只在函数内部进行一些操作。

请参考折线表数据定义。

举例：

```
...
sfloat d;
d=0.5f;
setfxy_x ( d,0,10 ) ;//写 10 号折线表的 0 号 x 数据
```

...

参考条目：

fxy getfxy\_x getfxy\_y setfxy\_y

### 14、setfxy\_y ( 写二维折线表的 y 轴的数据项 )

语法：

```
setfxy_y ( d,index,n ) ;
```

参数 d 为所写数据，数据类型为半浮点数。

参数 index 为数据项序号，数据类型为整型。

参数 n 为指定的折线表序号，范围是 0 ~ 31，数据类型为整型。

该函数没有返回值，只在函数内部进行一些操作。

请参考折线表数据定义。

举例：

```
...
sfloat d;
d=0.4f;
setfxy_y ( d,0,10 ) ;//写 10 号折线表的 0 号 y 数据
```

...

参考条目：

fx y getfxy\_x getfxy\_y setfxy\_x

## 15、ded (纯滞后模块)

语法：

$y = \text{ded}(x, n\text{TSample}, n\text{Num}, n);$

参数  $x$  为输入值，数据类型为半浮点数。

参数  $n\text{TSample}$  为采样时间常数，单位为 0.1 秒，数据类型为整型。

参数  $n\text{Num}$  为保留历史数据个数，最大为 50，数据类型为整型。

参数  $n$  为指定的序号，数据类型为整型，范围是 0~31。在同一程序中不允许用同一个序号调用本函数。

返回值  $y$  为对输入值  $x$  进行纯滞后处理后的值，数据类型为半浮点数。

本函数的功能是产生一个纯滞后，即现在的输出值为  $t = n\text{TSample} * n\text{Num}$  秒种以前的数值，换句话说，现在的输入值需  $t$  秒延迟后方能输出。例如纯滞后 100 秒，通常可设  $n\text{TSample}$  为 2 秒， $n\text{Num}$  为 50 个数据。 $n\text{TSample}$  最好设为系统控制周期的整数倍。

举例：

...

sfloat c,x;

$c = \text{ded}(x, 20, 50, 0);$  //采样周期 2 秒，50 个历史数据，纯滞后 100 秒。

...

参考条目：

led lag ledlag

## 16、led (一阶超前模块)

语法：

$y = \text{led}(x, t, n);$

参数  $x$  为被处理量，数据类型为半浮点数。

参数  $t$  为时间常数，单位为 0.1 秒，数据类型为整型。

参数  $n$  为指定的序号，数据类型为整型，范围是 0~31。在同一程序中不允许用同一个序号调用本函数。

返回值  $y$  为对某个量  $x$  进行一阶超前处理后的值，数据类型为半浮点数。

一阶超前是按下式进行处理： $X \times TS / (TS + 1)$ ，其中  $S$  为拉普拉斯算子， $T$  为时间常数。

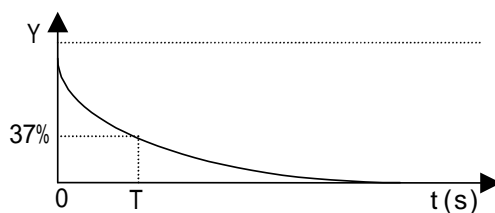


图 6-3 led 特性曲线

举例：

...

```
sfloat c,a;
c = led ( a,100,0 );
```

...

参考条目：

```
ded lag ledlag
```

## 17、lag（一阶滞后模块）

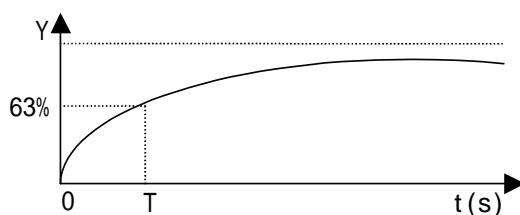


图 6-4 lag 特性曲线

语法：

```
y= lag ( x,t,n );
```

参数 x 为输入值，数据类型为半浮点数。

参数 t 为时间常数，单位为 0.1 秒，数据类型为整型。

参数 n 为指定的序号，数据类型为整型，范围是 0~31。在同一程序中不允许用同一个序号调用本函数。

返回值 y 为对输入值 x 进行一阶滞后处理后的值，数据类型为半浮点数。

一阶滞后是按下式进行处理： $[1/(TS+1)] \times X$ ，其中 S 为拉普拉斯算子，T 为时间常数，X 为输入值。

举例：

...

```
sfloat a,b;
int t,n;
a = lag ( b,t,n );
```

...

参考条目：

```
ded led ledlag
```

## 18、ledlag（一阶超前滞后模块）

语法：

```
y= ledlag ( x,ti,td,n );
```

参数 x 为被处理量，数据类型为半浮点数。

参数 ti 为滞后时间常数，单位为 0.1 秒，数据类型为整型。

参数 td 为超前时间常数，单位为 0.1 秒，数据类型为整型。

参数 n 为指定的序号，数据类型为整型，范围是 0~31。在同一程序中不允许用同一个序号调用本函数。

返回值 y 为对某个量 x 进行一阶超前滞后处理后的值，数据类型为半浮点数。

一阶超前滞后是按下式进行处理： $X \times (T_d \times S + 1) / (T_i \times S + 1)$ ，其中  $S$  为拉普拉斯算子。

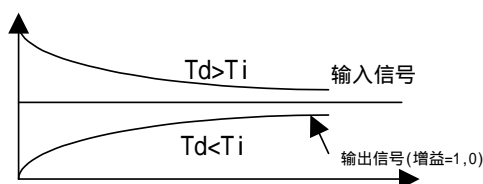


图 6-5 一级超前滞后阶跃响应

举例：

```
...
sfloat a,b;
b = ledlag ( a,100,50,0 );
```

...

参考条目：

ded led lag

## 19、mav (移动平均函数)

语法：

```
y= mav ( x,t,num,n );
```

参数  $x$  为输入新值，数据类型为半浮点数。

参数  $t$  为采样时间间隔设定值，数据类型为整型。

参数  $num$  为采样点数，取值范围为 1~8，数据类型为整型。

参数  $n$  指定的序号，数据类型为整型，范围是 0~31，在同一程序中不允许用同一个序号调用本函数。

返回值  $y$  为移动平均值，数据类型为半浮点数。

该指令对输入数据用指定的时间间隔采集指定的点数（最大为 8 点）对采样点求平均值，经过一个采样周期，采样一点新值就丢掉最前一点值，再求平均值。这就叫做移动平均值，由于记忆过去的的数据，所以最早的采样也可以参照。计算公式如下：

$$y = [X(T) + X(T-1) + \dots + X(T-m+1)] / m$$

$y$  为运算输出值

$X(T)$  为  $T$  时采样值

举例：

```
...
sfloat x,y;
y = mav ( x,20,6,0 );
```

...

参考条目：

msw sw vlm

## 20、msw (多路切换)

语法：

`y= msw ( switch,x0,x1,x2,x3,x4,x5 ) ;`

参数 switch 为切换选取序号，数据类型为整型；

参数 x0、 x1、 x2、 x3、 x4、 x5 为切换值，数据类型为半浮点数。

返回值 y 的数据类型为半浮点数。当 switch = 0 时，返回值为 x0；switch = 1 时，返回值为 x1；依次类推。

举例：

```
...
int switch;
sfloat a;
switch=2;
a = msw ( 2,0.0f,0.1f,0.2f,0.3f,0.4f,0.5f ) ;// a = 0.2f
...
```

参考条目：

mav sw vlm

## 21、sw（两路切换）

语法：

`y= sw ( switch,x1,x2 ) ;`

参数 switch 为切换开关，数据类型为布尔值。

参数 x1、 x2 为切换值，数据类型为半浮点数。

返回值 y 的数据类型为半浮点数。当 switch=OFF 时返回值为 x1；sw=ON 时返回值为 x2。

举例：

```
...
bool switch;
sfloat a;
switch = ON;
a = sw ( switch,0.0f,1.0f ) ;// a = 1.0f
...
```

参考条目：

mav msw vlm

## 22、vlm（速度限幅模块）

语法：

`y= vlm ( x,v1,v2,n ) ;`

参数 x 为输入值，数据类型为半浮点数，范围是 0~1。

参数 v1 为上升速度限制，单位 1/分，正数，数据类型为半浮点数。

参数 v2 为下降速度限制，单位 1/分，正数，数据类型为半浮点数。

参数 n 为指定的序号，数据类型为整型，范围是 0 ~ 127。在同一程序中不允许用同一个序号调用本函数。

返回值 y 为对输入值 x 进行速度限幅处理后的值，数据类型为半浮点数。

如图 6-6 所示，当输入变化率大于  $v1$  时，输出以  $v1$  速度变化，当输入速度小于  $v2$  时，输出以  $v2$  速度变化。

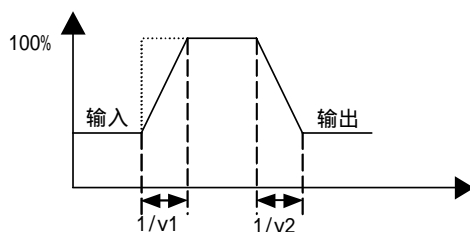


图 6-6 速度限幅响应

举例：

```
...
float a,c;
c = vlm ( a,0.8f,0.8f,0 );
```

...

参考条目：

max msw sw

## 6.2.4 混合计算模块

### 1、fkdivf (半浮点整数除法)

语法：

```
y= fkdivf ( x1,k );
```

参数  $x1$  的数据类型为半浮点数，参数  $k$  的数据类型为整型。

返回值  $y$  为  $x1$  除以  $k$  的商，实现对半浮点数的整数倍缩小，数据类型为半浮点数。返回值范围：

-7.9997  $y$  7.9997。

说明：

当  $0/0$  时，返回值为 0；

当负数/0 时，返回值为 -7.9997；

当正数/0 时，返回值为 7.9997。

举例：

```
...
sfloat a,b;
a = fkdivf ( -5.0f,-5 );// a = 1.0f
b = fkdivf ( 7.0f,10 );// b = 0.7f
```

...

参考条目：

fkmulf fkmulk kedivk kkdivf

### 2、fkmulf (半浮点整数乘法)

语法：

```
y= fkmulf ( x1,k );
```

参数  $x1$  的数据类型为半浮点数，参数  $k$  的数据类型为整型。

返回值  $y$  为  $x1$  乘以  $k$  的积，实现对半浮点数的整数倍放大，数据类型为半浮点数。返回值范围： $-7.9997 \leq y \leq 7.9997$ 。当两数积值超出该范围时，则限制在此范围内，即

$x1 \times k < -7.9997$  时，返回值为  $-7.9997$ ，

$x1 \times k > 7.9997$  时，返回值为  $7.9997$ 。

举例：

...

`sfloat a,b;`

`a = fkmulf ( 3.0f,10 ) ;// a = 7.9997f`

`b = fkmulf ( 0.05f,10 ) ;// b = 0.5f`

...

参考条目：

`fkdivf fkmulk kedivk kkdivf`

### 3、 fkmulk ( 半浮点整数乘法 )

语法：

`y= fkmulk ( x1,k ) ;`

参数  $x1$  的数据类型为半浮点数，参数  $k$  的数据类型为整型。

返回值  $y$  为  $x1$  乘以  $k$  的积，数据类型为整型。函数目的是将半浮点数放大，用整数来描述大小。

举例：

...

`sfloat a;`

`int b,c;`

`a = 0.1f;`

`b = fkmulf ( a,20 ) ;// b = 2`

`c = fkmulf ( -3.0f,20 ) ;// c = - 60`

...

参考条目：

`fkdivf fkmulf kedivk kkdivf`

### 4、 kfdivk ( 整数的半浮点除法 )

语法：

`y= kfdivk ( k,x1 ) ;`

被除数  $k$  的数据类型为整型，除数  $x1$  的数据类型为半浮点数。

返回值  $y$  为  $k$  除以  $x1$  的商，数据类型为整型。返回值范围： $-32767 \leq y \leq 32767$ 。

当  $0/0$  时，返回值为  $0$ ；

当负数  $/0$  时，返回值为  $-32767$ ；

当正数  $/0$  时，返回值为  $32767$ 。

举例：

...

```
int a,b;
a = kfdivk ( 50,0.5f ) ;// a = 100
b = kfdivk ( 10000,0.01f ) ;// b = 32767
```

...

参考条目：

fkdivf fkmulf fkmulk kkdivf

## 5、 kkdivf ( 整数除法 )

语法：

```
y= kkdivf ( k1,k2 ) ;
```

被除数 k1、除数 k2 的数据类型为整型。

返回值 y 为 k1 除以 k2 的商，数据类型为半浮点数。返回值范围：-7.9997 y 7.9997。函数目的是整数相除得半浮点数。

举例：

...

```
sfloat a,b;
a = kkdivf ( 30,50 ) ;// a = 0.6f
b = kkdivf ( 20,1 ) ;// b = 7.9997f
```

...

参考条目：

fkdivf fkmulf fkmulk kedivk

## 6.2.5 控制模块

### 1、 dgap2 ( 二位式二状态控制模块 )

语法：

```
y= dgap2 ( deviation,hysteresis,n ) ;
```

参数 deviation 为给定值与测量值间的偏差 SV-PV，数据类型为半浮点数。

参数 hysteresis 为死区间隙，数据类型为半浮点数。

参数 n 为模块序号 0~31，数据类型为整型。

返回值 y 为输出阀位值，数据类型为半浮点数。输出 100% 表示输出增加，输出 0 表示输出减少。

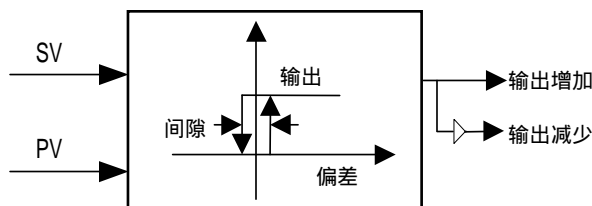


图 6-7 二状态调节

举例：

...

```
sfloat a,err;
a = dgap2 ( err,0.05,0 ) ;
```



...

参考条目：

dgap3 csc bsc

## 2、 dgap3 (二位式三状态控制模块)

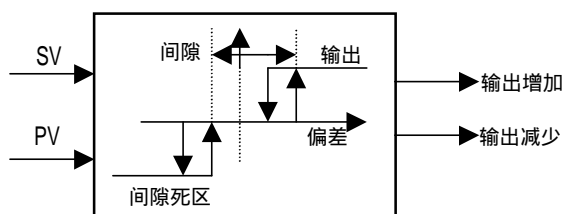


图 6-8 三状态调节

语法：

```
y= dgap3 ( deviation,zone,hysteresis,n );
```

参数 deviation 为给定值与测量值间的偏差 SV-PV，数据类型为半浮点数。

参数 zone 为间隙，数据类型为半浮点数。

参数 hysteresis 为死区间隙，数据类型为半浮点数。

参数 n 为模块序号 0~31，数据类型为整型。

返回值 y 为输出阀位值，数据类型为半浮点数。输出 100%表示输出增加，输出 0 表示输出减少，50%表示不增不减。

举例：

...

```
sfloat a,err;
```

```
a = dgap3 ( err,0.05,0.01,0 );
```

...

参考条目：

dgap2 csc bsc

## 3、 bsc (单回路 pid 控制模块)

语法：

```
y= bsc ( pv,n );
```

参数 pv 为被控量，数据类型为半浮点数。

参数 n 为模块序号 0~63，数据类型为整型。

模块各参数作用见 BSC 参数表。

返回值 y 为输出阀位值，数据类型为半浮点数。模块的结构请参考 BSC 结构图。

举例：

...

```
sfloat a,pv;
```

```
a = bsc ( pv,0 );
```

...

参考条目：

dgap2 dgap3 csc

#### 4、 csc ( 串级 pid 控制模块 )

语法：

y= csc ( pvEx,pvIn,n );

参数 pvEx 为外环被控量，数据类型为半浮点数。

参数 pvIn 为内环被控量，数据类型为半浮点数。

参数 n 为模块序号 0~63，数据类型为整型。

模块各参数作用见 [CSC 参数表](#)。

返回值 y 为输出阀位值,数据类型为半浮点数。模块的结构请参考 CSC 结构图。

举例：

...

sfloat a,pvEx,pvIn;

a = csc ( pvEx,pvIn,0 );

...

参考条目：

dgap2 dgap3 bsc

#### 5、 orint ( 16 位整数或 )

语法：

y= orint ( x1,x2 );

参数 x1、 x2 的数据类型为整数。

返回值 y 为 x1、 x2 的或值，数据类型为整数。

举例：

...

int a,b;

a = 3;//

b = orint(a,9);// b = 11

...

参考条目：

orlong, andint, andlong

#### 6、 andint ( 16 位整数与 )

语法：

y= andint ( x1,x2 );

参数 x1、 x2 的数据类型为整数。

返回值 y 为 x1、 x2 的与值，数据类型为整数。

举例：

...

```
int a,b;
a = 3;//
b = andint(a,9);// b = 1
```

...

参考条目：

andlong, orint, orlong

## 7、 orlong ( 32 位长整数或 )

语法：

```
y= orlong ( x1,x2 );
```

参数 x1、 x2 的数据类型为长整数。

返回值 y 为 x1、 x2 的或值，数据类型为长整数。

举例：

...

```
long a,b;
a = 3;//
b = orlong(a,9);// b = 11
```

...

参考条目：

orint, andint, andlong

## 8、 andlong ( 32 位长整数与 )

语法：

```
y= andlong ( x1,x2 );
```

参数 x1、 x2 的数据类型为长整数。

返回值 y 为 x1、 x2 的与值，数据类型为长整数。

举例：

...

```
long a,b;
a = 3;//
b = andlong(a,9);// b = 1
```

...

参考条目：

orlong, andint, orint

## 9、 getbit ( 从长整数中取 BOOL )

语法：

```
y= getbit(num,serial);
```

参数 num 的数据类型为长整数,serial 的数据类型为整数。Serial 的意思是取 num 中的哪一位，Serial 的取值为 0 到 31

返回值 y 数据类型为 BOOL。

举例：

```
...
long a;
bool y;
a = 3;
y = getbit(a,0); // y = ON
y = getbit(a,2); // y = OFF
...
```

参考条目：

setbit, getsfloat, getint, getlong, setsfloat, setint, setlong, setfloat, getfloat

## 10、getsfloat（从长整数中取半浮点数）

语法：

```
y= getsfloat(num,serial);
```

参数 num 的数据类型为长整数,serial 的数据类型为整数。Serial 的意思是取 num 中的高 16 位或是低 16 位，Serial 的取值为 0 或非 0，0 表示取低 16 位中的数据

返回值 y 数据类型为 sfloat。

举例：

```
...
long a;
sfloat y;
a = 4096;
y = getsfloat(a,0); // y = 1.0f
y = getsfloat(a,1); // y = 0.0
...
```

参考条目：

getbit, setbit, setsfloat, getint, getlong, setsfloat, setint, setlong, setfloat, getfloat

## 11、getint（从长整数中取整数）

语法：

```
y= getint(num,serial);
```

参数 num 的数据类型为长整数,serial 的数据类型为整数。Serial 的意思是取 num 中的高 16 位或是低 16 位，Serial 的取值为 0 或非 0，0 表示取低 16 位中的数据

返回值 y 数据类型为整数。

举例：

```
...
long a;
int y;
a = 4096;
y = getsfloat(a,0); // y = 4096
y = getsfloat(a,1); // y = 0
...
```

参考条目：

setint, getbit, get sfloat, getlong, setbit, setsfloat, setlong, setfloat, getfloat

## 12、getfloat (从长整数中取浮点数)

语法：

```
y= getfloat(num);
```

参数 num 的数据类型为长整数。

返回值 y 数据类型为浮点数。

举例：

```
...
long  a;
float y;
a = setfloat(10.0);
y = getfloat(a);  // y = 10.0
...
```

参考条目：

setbit, getbit, setsfloat, getsfloat, setint, getint, setlong, getlong, setfloat

## 13、setbit (向长整数中放 bool)

语法：

```
y= setbit(num,val,serial);
```

参数 num 的数据类型为长整数。Val 为 bool, serial 为整数类型。

返回值 y 数据类型为长整数型。

举例：

```
...
bool  a;
long y;
a = ON
y = setbool(0,a,0); //y = 1
y = setbool(y,a,1); // y = 3
...
```

参考条目：

getbit, setsfloat, getsfloat, setint, getint, setlong, getlong, setfloat, getfloat

## 14、setsfloat (向长整数中放 sfloat)

语法：

```
y= setsfloat(num,val,serial);
```

参数 num 的数据类型为长整数。Val 为 sfloat, serial 为整数类型。

返回值 y 数据类型为长整数型。

举例：

```
...
sfloat  a;
long y;
```

```
a = 1.0f
y = setfloat(0,a,0); //y = 4096
...
```

参考条目：

setbit, getbit, setsfloat, setint, getint, setlong, getlong, setfloat, getfloat

### 15、setint ( 向长整数中放 int )

语法：

```
y= setint(num,val,serial);
```

参数 num 的数据类型为长整数。Val 为 int,serial 为整数类型。

返回值 y 数据类型为长整数型。

举例：

```
...
int  a;
long y;
a = 1
y = setint(0,a,1); //y = 65536
...
```

参考条目：

setbit, getbit, setsfloat, getsfloat, getint, setlong, getlong, setfloat, getfloat

### 16、setfloat ( 向长整数中放 float )

语法：

```
y= setfloat(num,val);
```

参数 num 的数据类型为长整数。Val 为 float。

返回值 y 数据类型为长整数型。

举例：

```
...
float  a;
long y;
a = 1.0
y = setfloat(a); //
...
```

参考条目：

setbit, getbit, setsfloat, getsfloat, setint, getint, setlong, getlong, getfloat

### 17、getmsg ( 从其他控制站取消息 )

语法：

```
y = getmsg(nStation,serial);
```

参数 nStation 的数据类型为长整数。表示控制站地址

Serial 数据类型为 int , 表示控制站的第几个消息槽位。

返回值 y 数据类型为长整数型。

举例：

```
...
y = getmsg(8,0); //取 8 号控制站 0 槽位的数据
```

...

参考条目：

sendmsg

## 18、 sendmsg ( 设置向其他控制站发消息的数目 )

语法：

```
void sendmsg(size);
```

size 数据类型为 int , 表示要发多少条消息。

要发的消息通过全局数组 g\_msg 设置

举例：

```
...
long y;
y = setfloat(0.8);
g_msg[0] = y;
sendmsg(1); //发一个数据
```

...

参考条目：

getmsg

## 19、 AddAccum ( 累积量加法 )

语法：

```
y = AddAccum(x1,x2);参数 x1、 x2 的数据类型为累积量。
```

返回值 y 数据类型为累积量。

参考条目：

SubAccum, ConvertAccum, ConvertToAccum, CompAccum, TotalAccum,

## 20、 SubAccum ( 累积量减法 )

语法：

```
y = SubAccum(x1,x2);
```

参数 x1、 x2 的数据类型为累积量。

返回值 y 数据类型为累积量。

参考条目：

AddAccum, ConvertAccum, ConvertToAccum, CompAccum, TotalAccum,

## 21、 ConvertAccum ( 累积量转换成浮点量 )

语法：

```
y = ConvertAccum (x1);
```

参数 x1 的数据类型为累积量。

返回值 y 数据类型为浮点量。

参考条目：

AddAccum, SubAccum, ConvertToAccum, CompAccum, TotalAccum,

## 22、ConvertToAccum (浮点量转换成累积量)

语法：

y = ConvertToAccum (x1);

参数 x1 的数据类型为浮点量。

返回值 y 数据类型为累积量。

参考条目：

AddAccum, SubAccum, ConvertAccum, CompAccum, TotalAccum,

## 23、CompAccum (累积量比较运算)

语法：

y = CompAccum (x1,x2);

参数 x1、x2 的数据类型为累积量。

返回值 y 数据类型为整数。

X1 = x2     y = 0

X1 > x2     y = 1

X1 < x2     y = -1

参考条目：

AddAccum, SubAccum, ConvertAccum, ConvertToAccum, TotalAccum,

## 24、TotalAccum (累积量累积运算)

语法：

y = TotalAccum (x1,x2);

参数 x1 的数据类型为累积量。

参数 x2 的数据类型为半浮点数

返回值 y 数据类型为累积量。

举例：

y = TotalAccum( y, 0.5f); // y 将每秒加 0.5

参考条目：

AddAccum, SubAccum, ConvertAccum, ConvertToAccum, CompAccum

## 25、SteamComp (过热蒸气流量补偿)

语法：

y = SteamComp(F,T,P,DesignV);

参数 F 的数据类型为浮点数表示初始流量。

参数 T 的数据类型为浮点数表示实际温度

参数 P 的数据类型为浮点数表示实际压力

参数 DesignV 的数据类型为浮点数表示设计比容



返回值 y 数据类型为半浮点数表示补偿后的流量。

参考条目：

## 6.2.6 类型转换模块

### 1、 norm ( 归一化函数 )

语法：

y= norm ( x ) ;

参数 x 的数据类型为半浮点数，半浮点数 = 整数/4096。

返回值 y 为半浮点数 x 归一化后所得整数，数据类型为整型。函数目的是将半浮点数换算成整数形式显示。

举例：

```
...
sfloat a;
int b;
a = 0.5f;
b = norm ( a ) ;// b = 2048
```

...

参考条目：

denorm sftoi itosf sftof ftosf sftol ltosf ftoi itof

### 2、 denorm ( 反归一化函数 )

语法：

y= denorm ( x ) ;

参数 x 的数据类型为整型，整数值 = 半浮点数 × 4096。

返回值 y 为整数 x 反归一化后所得半浮点数，数据类型为半浮点数。当

x= - 4096 时，返回值为-1.0；

x=0 时，返回值为 0.0；

x=4096 时，返回值为 1.0；

x=32767 时，返回值为 7.9997。

举例：

```
...
sfloat a;
int b;
b=2048;
a = denorm ( b ) ;// a = 0.5f
```

...

参考条目：

norm sftoi itosf sftof ftosf sftol ltosf ftoi itof

### 3、 itosf ( 整数转换到半浮点 )

语法：

```
y= itosf ( x ) ;
```

参数 x 的数据类型为整型，数值范围为-8    x    8，

返回值 y 为整数 x 转换成的半浮点数，数据类型为半浮点数。返回值范围： - 7.9997    y  
7.9997。当 x 超出该范围时，

x<-8 时，返回值为- 7.9997，

x>8 时，返回值为 7.9997。

举例：

...

```
sfloat a,b;
```

```
a = itosf ( 6 ) ;// a = 6.0f
```

```
b = itosf ( 9 ) ;// b = 7.9997f
```

...

参考条目：

norm   denorm   sftoi   sftof   ftosf   sftol   ltosf   ftoi   itof

#### 4、 sftoi ( 半浮点数转换到整数 )

语法：

```
y= sftoi ( x ) ;
```

参数 x 的数据类型为半浮点数。

返回值 y 为半浮点数 x 经过四舍五入转换成的整数，数据类型为整型。

举例：

...

```
int a;
```

```
sfloat b;
```

```
b = 7.3f;
```

```
a = sftoi ( b ) ;// a = 7
```

...

参考条目：

norm   denorm   itosf   sftof   ftosf   sftol   ltosf   ftoi   itof

#### 5、 ftosf ( 浮点数转换到半浮点数 )

语法：

```
y= ftosf ( x ) ;
```

参数 x 的数据类型为浮点数。

返回值 y 为浮点数 x 转换成的半浮点数，数据类型为半浮点数。返回值范围： -7.9997    y  
7.9997。

举例：

...

```
sfloat a,b;
```

```
a = ftof ( 0.7 ) ;// a = 0.7f
b = ftof ( 23.4 ) ;// b = 7.9997f
```

...

参考条目：

norm denorm sftoi itosf sftof sftol ltosf ftoi itof

## 6、 sftof ( 半浮点数转换到浮点 )

语法：

```
y= sftof ( x ) ;
```

参数 x 的数据类型为半浮点数。

返回值 y 为半浮点数 x 转换成的浮点数，数据类型为浮点数。

举例：

...

```
float a;
sfloat b;
b = 6.5f;
a = sftof ( b ) ;// a = 6.5
```

...

参考条目：

norm denorm sftoi itosf ftof sftol ltosf ftoi itof

## 7、 itof ( 整数转换到浮点数 )

语法：

```
y= itof ( x ) ;
```

参数 x 的数据类型为整型。

返回值 y 为整数 x 转换成的浮点数，数据类型为浮点数。

举例：

...

```
float a;
int b;
b = 24;
a = itof ( b ) ;// a = 24.0
```

...

参考条目：

norm denorm sftoi itosf sftof ftof sftol ltosf ftoi

## 8、 ftoi ( 浮点数转换到整数 )

语法：

```
y= ftoi ( x ) ;
```

参数 x 的数据类型为浮点数。

返回值 y 为浮点数 x 经过四舍五入转换成的整数，数据类型为整型。

举例：

```
...
float a,b;
b=23.5;
a = ftoi ( b ) ;// a = 24
...
```

参考条目：

norm denorm sftoi itosf sftof ftosf sftol ltosf itof

## 6.2.7 常用语句

### 1、 break

该语句将使应用程序的执行从最近一层跳出到外一层并继续执行。

### 2、 continue

该关键字主要用于循环语句，当程序执行到该语句是，将跳过所有后续的语句从循环体头开始。

### 3、 define（宏定义）

语法：

#define 宏名（常数）

定义一个常数。



在程序编译时，程序中引用宏定义的地方，编译程序是先将相应的宏定义代替引用的宏，然后再进行编译。

举例：

```
...
#define Pi 3.14
...
```

### 4、 for（循环语句）

语法：

for（i 的初值,i 的终值,i 的增量）

```
{
...
}
```

其中计数变量 i 为 int 型变量。初值、终值、增量为运算表达式。

语句按以下步骤进行：

计算初值、终值和增量，增量缺省值为 1；

初值赋给计数变量；

比较计数变量和终值，如果计数变量 < 终值（增量 > 0）或计数变量 > 终值（增量 < 0），则

继续下一步，否则退出循环。

执行 For 后 “ { , } ” 语句之间的语句序列；

计数变量加上增量；

转第三步。

举例：

```
...
int Stat[23];
int i;
for ( i=0; i<23; i=i+1 )
{
    Stat[i] = i;
}
...
```

## 5、 goto

程序执行到该语句将跳转到标号指定的位置继续执行。

## 6、 if ... [else if ... ] else ... （条件语句）

语法：

if 条件表达式

```
{
```

```
...
```

```
}
```

[ [ else if 条件表达式

```
{
```

```
...
```

```
}
```

else if 条件表达式

```
{
```

```
...
```

```
}
```

```
]
```

else

```
{
```

```
...
```

```
}
```

按如下步骤执行：

- 计算 if 后面的条件表达式，如果值非零，执行相应的 Then 后面的语句序列，程序转入 if 块后的语句；如为零，转（2）；
- 如有 elseif 部分（可以有多个 elseif 分支），计算 elseif 后面的条件表达式，如果值非零，执

行后面的语句序列，程序转入程序块后的语句；如为零，如果存在下一 `elseif` 分支，判断执行此分支，否则转 (3)；

➤ 执行 `else` 后的语句序列，程序转入程序块后的语句。

举例：

```
...
if ( A > B )
{
x = A;
}
else
{
x = B;
}
if ( x < C )
{
x = C;
}
...
```

## 7、 `main`

`main ( )` 是 SCX 语言应用程序的入口函数，在每一个应用中必不可少。其在应用程序中的使用方法请参考应用程序框架和应用实例。

## 8、 `return`

该语句将导致应用程序从当前执行的函数中返回到调用程序中。

## 9、 `timerm`

第 `n` 号可随机访问的分定时器。

变量名 `timerm[n]`

这是系统预先定义，用于控制信息的交换处理等的变量，用户可以访问此变量，进行读写。

这里 `n` 的取值范围是 0 ~ 255。

## 10、 `timers`

第 `n` 号可随机访问的秒定时器。

变量名 `timers[n]`

这是系统预先定义，用于控制信息的交换处理等的变量，用户可以访问此变量，进行读写。

这里 `n` 的取值范围是 0 ~ 255。

## 11、 `void`

表明该函数没有返回值，只在函数内部进行一些操作。

## 12、 `while`

语法：

while 条件表达式

{

语句序列;

}

执行步骤如下：

- 计算条件表达式的值；
- 如果值为真，执行两大括号之间的语句，转上步；如为假，转入执行大括号后的一句。

举例：

...

int D;

D = 10;

while ( D >= 5 )

{

D = D - 1;

}

...

## 7 资料版本说明

表 7-1 版本升级更改一览表

资料版本号	更改说明
SCX 语言使用手册手册（V2.0）	适用于软件版本：AdvanTrol-Pro V2.65+SP05
SCX 语言使用手册手册（V2.1）	适用于软件版本：AdvanTrol-Pro V2.70