# <u>SIXNET</u>®

### ISaGRAF 自学培训手册

介绍和安裝 ISaGRAF 概述 创建 ISaGRAF 项目 向 ISaGRAF 项目输出 SIXTAGS 使用快速梯形图编辑器 使用梯形图/功能块图编辑器 编译 ISaGRAF 应用 使用 I/O 仿真器 在 SIXNET Windows Run-time 方式运行应用 在 SIXTRAK 控制器中运行应用 使用在线调试器 用户自定义功能块

### 本章我们将要:

- 安装 ISaGRAF 软件
- 安装 SIXNET ISaGRAF Windows Run-time软件
- 复制需要的培训支持文件

### ISAGRAF 自我培训手册

本文档适用于熟悉SIXTRAK 硬件的用户。按步骤通过ISaGRAF IEC 1131-3编程,创建、编译、调试 仿真、及运行一个ISaGRAF应用。

本教程为模块结构,每章是一个有针对性的单独的文档。本教程有一定的连续性,所以我们推荐(但不要求)您完成本手册的每一章。

本手册适用的项目文件和 ISaGRAF Windows Runtime for SIXNET Control Room一起提供,自动 安装在正确的目录下。

## TRAINING.6PJ即 SIXTRAK Plant Floor 项目文件<br/>该项目包括一个 ST-GT-ETH-24P 控制器和 Virtual DI 8, Virtual DO 8, Virtual AI 8,<br/>及 Virtual AO 8 模块。本教程的运行不需要任何实际的硬件。如果您有不同于ST-<br/>GT-ETH-24P的其他SIXTRAK控制器,应当先对该项目文件作相应修改。

TRAIN\_SX.PIA 即 ISaGRAF 培训项目文件 本培训手册将一步一步地带领您创建一个完整的 ISaGRAF 项目。如果您想预先看 到完整的项目,可以用文档工具的恢复功能安装该文件到 ISaWIN目录。

SXON\_OFF.AIA 即 ON\_OFF 功能块 本培训手册将一步一步地带领您创建一个用户定义的功能块。如果您想预先看到完整的项目,可以用文档工具的恢复功能安装该文件到 ISaWIN目录。

### ISAGRAF 安装

ISaGRAF安装程序在Windows程序组加上 "ISaGRAF"程序组,并在\ISAWIN\EXE子目录下产生一个 初始化文件"ISA.ini"。按下列步骤安装ISaGRAF:

- **首先 →** 插入 SIXNET 软件光盘,光盘将自动启动。
- 然后 → 从主菜单选择安装
- **然后→** 选择ISaGRAF Workbench和下列在线指南完全安装:

安装程序会询问是否安装下列组件:

- ISaGRAF可执行程序
- 在线信息和帮助文件
- ISaGRAF标准库
- ISaGRAF 样例应用

推荐在首次安装时安装所有选项,个别组件也可后来重新安装。当所有的 ISaGRAF 文件复制后, ISaGRAF程序组声称, ISaGRAF主目录为"\ISAWIN"。子目录也自动建立。



### 安装SIXNET ISAGRAF WINDOWS RUN-TIME软件(ISARUN)

安装ISaGRAF Workbench后, Isarun自动安装。

**注意:** 为了与SIXNET I/O正确地通讯,必须安装SIXNET ISaGRAF Windows Run-time。Isarun 必须在Workbench安装后安装,重新安装ISaGRAF Workbench后,Isarun也应重新安装。

### 使用 ISAGRAF 在线帮助

在线帮助随ISaGRAF Workbench一起安装,包括以下主题:

- ISaGRAF语言参考手册
- 完整的用户指南 (对任何ISaGRAF工具)
- 标准库中元素的技术要点

### 使用 SIXNET 在线帮助

在线帮助随SIXNET ISaGRAF Windows Run-time一起安装。

### **ISaGRAF**概述

关于ISaGRAF的详细信息请参考

ISaGRAF 用户指南

1/7

### 本章我们将要:

- 概述ISaGRAF,及其结构和资源 •
- 定义基本的ISaGRAF术语
- 简述SIXNET ISaGRAF系统的设置步骤

### 本章开始前:

您应当已经阅读了本手册的介绍,并对控制程序有基本的了解。

### 什么是 ISAGRAF

ISaGRAF 是容易理解和使用的控制系统编程环境,它使得SIXTRAK I/O和 VersaTRAK RTUs 成为高 性能、低价格的控制器。ISaGRAF使用标准的工业PLC编程方法,不需要高级计算机语言或计算机硬 件专业知识就可设计出高性能的应用。

### ISAGRAF 项目和程序

一个ISaGRAF项目(project)是一组程序和函数的集合,构成一个完整的控制应用。每个程序控制应用 的一个特定部分。程序和函数在项目中按照其在ISaGRAF目标循环的位置分为四个部分。在同一个应 用中可以混合使用所有五种IEC 1131-3语言。

### IEC 1131-3 语言

₩ **梯形图(LD)** 是大多PLC支持的程序逻辑的类继电器 触点和线圈的经典表示方法。梯形图可在Quick Ladder编辑器中快速容易地建立,集合基本元素而形 成一个完整的程序。



功能块图(FBD) 是高端过程控制器与DCS系统的通用 0-0 **通**编程语言,把图形化的处理计算功能加入了 ISaGRAF。它可与梯形图元素一起使用以集合过程控 制与机器控制的功能。

SIXNET北京代表处

Tel: 010-62316290 Fax: 010-62316671 E-Mail: support@sixnetiot.com.cn

### ISaGRAF概述

2/7

顺序功能图(SFC) 是强大的工业流程图语言,将项目 组织为有顺序的(按步执行)操作。它使用简单的图形 表示不同的步(Step),并将这些步按布尔条件(变迁-Transition)链接起来,每步中的作用(action)由其他四 种语言详细描述。

	Start auto cycle	Manual press close	×
106	Net Manual_Mode and Tan_Cryste;	205	
20	Close the press Dise_Cond 33		
110	Wat for down link Dem Link;	210 Manually close press Geas_Grad;	-

结构文本(ST) 是类Pascal的计算机编程语言,它对于 编写复杂的操作很有效,常用于创建用户定义的功能 块和描述SFC的步及变迁。

**指令列表(IL)** 类似汇编语言,提供给熟悉Siemens陈 述列表编程的用户。

#### (\* extract left and right parts of the string \*)

right\_part := right ( animation , 1 ) ; left\_part := left ( animation , 9 ) ;

(\* invert left and right parts to modify string \*)

animation := ( right\_part + left\_part ) ;



### ISAGRAF 目标循环:

SIXNET ISaGRAF runtime 按照如下目标循环执行一个控制程序:



### ISAGRAF编程部分

如前所述,每个ISaGRAF控制项目由一系列程序、子程序和函数组成,按照上图的目标循环分作 四个不同部分。每部分可包含几个程序,也可根本没有。在ISaGRAF程序管理窗口,每部分有水平线 条分隔。

Beginning: 循环的程序不依赖于时间。本部分程序在输入扫描后的开始部分由系统执行。不能是SFC。

SIXNET北京代表处

Sequential: 遵守SFC规则和实现,依赖于一系列步和变迁。Sequential部分的程序支持程序间的平行处理编程和父子关系,必须是 SFC。

End: 与beginning部分程序有相同特点,在更新输出前的循环结束时执行。

Functions: 能够被其他三部分程序调用的子程序。函数不能是SFC。

- 🗄 ISaGRAF - A_SAMPLE - Programs 📃 🗖 🗖				
<u>File Make Tools Debug Options Help</u>				
▶ 四令田 ● □ □ ○ 本 ☆ ○ □ ○ ○ ☆ ○ ○				
Begin: En Scale En tankdemo				
Sequential: Plater				
End: Bafety				
Functions: 🕮 CoolDown				

### ISAGRAF字典

本手册常常引用ISaGRAF字典,该字典是内部变量、输入变量、输出变量以及定义的集合,在项 目程序中应用。

🔖 ISaGRAF - A_SAMPLE - Global booleans 📃 🗖 🔀						
<u>File E</u> dit <u>S</u> earch <u>O</u> ptions <u>H</u> elp						
	≙ 0 0 0	9 🖻 0	9 🖉 🕮 🖻 🖷 😽 🖪 🛃	8		
Name	Attrib.	Addr.	Comment			
Close Cmd	[internal]	0000	Command to close the press	<b>▲</b>		
Heat_Request	[internal]	0000	Command to apply power to the heater			
sim_mixer	[internal]	0000	Used by the simulator to turn the mixer			
OFF	[internal]	0000	Permanently False Variable			
Start_Cycle	(input)	0000	DI0:DI:1 (X0) Auto Cycle Start			
Down_Limit	(input)	0000	DI1:DI:2 (X1) Press Fully Closed			
Up_Limit [input] (		0000	DI2:DI:3 (X2) Press Fully Opened			
Safety_Gate	[input]	0000	DI3:DI:4 (X3) Safety Gate Closed			
Manual_Mode	[input]	0000	DI4:DI:5 (X4) On for Manual Mode			
Fill_Tank	[input]	0000	DI5:DI:6 (X5) On if Material is Needed			
Drain_Tank	[input]	0000	DI6:DI:7 (X6)			
DI8	[input]	0000	DI7:DI:8 (X7) also DO:8 (Y8)			
Press_Valve	[output]	0000	DO0:DO:1 (Y0) On to Close Press			
Main_Power	[output]	0000	DO1:			

输入和输出变量是I/O值。可以从SIXTAGS输出,以节省建立输入输出变量的时间。SIXTAGS输出在ISaGRAF字典中自动定义和组态I/O变量。

内部变量是用于计算和分析的逻辑名,既不是输入变量也不是输出变量。例如,可以建立一个量 程转换的温度变量,以便用工程单位表示原始的I/O值。

定义是一个常数的说明性名字,以便容易地维护项目。由于一些说明性名字在整个项目中使用, 如果需要改变数值,只需在字典中改变一次即可。

定义和变量可以在字典中创建,分作:

- Local (局部的),只用于一个程序
- Global (全局的),可用于单个项目的任何程序
- Common (通用的),可用于任何ISaGRAF项目



### 使用在线调试工具

🔍 ISaGRAF -	TRAINING	- Deb	ugger		
<u>File</u> <u>C</u> ontrol S	Spy <u>O</u> ptions	<u>H</u> elp			
@)))! ≫	6 6 66	Ö	# 🖾		
RUN al	owed=0		curre	nt=0 ma×imum=0	overflow=0

所有的ISaGRAF编辑器具有 "Debug" (调试) 命令,并提供一组观察目标站活动I/O数据的工具。每个I/O 点的状态和数值在程序逻辑中图示出来,并允许用户写入I/O值以测试应用。本手册后面的章节将详细学习调试器(Debugger)。

### 使用I/O仿真器

@ a_sample							
<u>File Opt</u>	ions <u>H</u> elp						
	0 Al0	1 A00	2 DI0	3 D00			
750	Press_Temp	81 [Tank_Level_Out	Start_Cycle	1 Press_Valve			
30	Hq	Temp_AO2	Down_Limit	2 Main_Power			
1200	ORP	Pressure_AO3	Dp_Limit	03 Heater_Power			
150	Zone_1_Temp	Flow_Rate_AO4	Safety_Gate	O4 Low_Temp_Alarm			
5	Conveyor_Speed	* * * *	Manual_Mode	95 High_Temp_Alarm			
30	Flow_Rate	* * * *	Fill_Tank	<b>6</b> DO6			
80	Tank_Level_1	* * * *	Drain_Tank	<b>07</b>			
75	J_Thermocouple			<b>●8</b> DO8 -			
•				▶ <i> </i> /;			

"Simulate"(仿真)命令可以仿真方式打开调试器(debugger)。仿真方式下,I/O仿真器窗口代替了链接外部I/O,使我们可以在连接到活动系统前测试程序。这个方便的开发工具节省了我们的时间,并减少了开始的故障。

程序日记

ISaGRAF包括一个与编辑的程序相联系的日记文件,编辑或修改程序时可以进行记录。可以由 "File / Diary"命令手动输入。如果在程序编辑器的 "Options"菜单选择了"Update diary"方式,则每次 程序保存前更新日记对话框会自动打开。

每次程序编译时,语法检查信息会自动记录到日记文件,并加上日期/时间标志。

📐 ISaGRAF - TRAINING:SAFETY - Diary 📃 🔳 💌				
<u>File Edit Options</u>	<u>H</u> elp			
≌ ≫ ⊑ ≾	8			
Name:	Scale	<b></b>		
Language:	FBD			
Creation date:	3/25/97 08:42:37			
Modification: 3/25/97 08:56:25				
Scale_temp engineering units (1-300 degrees)				
3/25/97 08:56:30: Verifying Scale No error detected				
<u>र</u>		▶ //		

### 使用文档特性(ARCHIVING)-- TRAINING项目

利用ISaGRAF文档(archive)工具可以保存ISaGRAF项目和重要文档或备份目录。

可将一个ISaGRAF项目所有的源代码、库元素和通用数据归档为一个完整的部分。文档工具的主窗口的**Options,Compression**(压缩)特性可用于减少备份文件的大小。ISaGRAF库(用户定义功能块,技术要点等)另外分别归档。

### 特别注意:

本手册相关的两个归档对象为:TRNGBKUP.PIA 和 SXON\_OFF.AIA。TRNGBKUP是完整的 ISaGRAF培训项目(training),SXON\_OFF是完整的ON\_OFF功能块。可使用Archive,Restore(恢 复)功能安装到ISaWIN目录下。

### 首先 →

打开ISaGRAF项目管理器,选择 Tools/Archive,选择Projects。

Archive主窗口左边Workbench栏是当前APL目录下的项目文件,右边 Archive栏是ARK目录下的已归档文

≢SIXNET ISaGRAF - Project Management				
<u>F</u> ile <u>E</u> dit <u>P</u> roject	<u>T</u> ools <u>O</u> ptions <u>H</u> elp			
🔓 🖾 🗋 💼 💼	<u>A</u> rchive	<u>P</u> rojects		
💼 creation	Libraries	<u>C</u> ommon data		
Ⅲ wy <mark>Ⅲ train_sx</mark>	Import IL program Taining program created by 3	self-study tutorial		

### ISaGRAF概述

6/7

### 件。

### 然后 →

从Archive栏选择train\_sx,按Restore 按钮,将其恢复到APL下。关闭之。

也可将当前项目文件备份(Backup按 钮)到要求目录下。

可按下Browse按钮,从弹出框中选取 文件和目录。

归档功能块:

### 然后 →

打开ISaGRAF项目管理器,选择 Tools/Libraries。

### 然后 →

从弹出框Libraries的下拉框选择 Function Blocks。

从菜单栏选择Tools/Archive。

### 然后 →

弹出框Function Blocks的左栏为以归 档的功能块,右栏为目前可用的功能 块。

从左栏选取sx0n\_off功能块,按下 Restore按钮,即可将已归档功能块 恢复出来。

也可将当前功能块备份(Backup按 钮)到要求目录下。

可按下Browse按钮,从弹出框中选 取文件和目录。

Workbench	Archive	
reation	rfmath 🔺	Backup
rain_sx	rfmatrix	
'Y	rfrotate	<u>R</u> estore
	rfscope	
	rfsfc	Close
	rfshow	
	rfstack	Help
	rfstring	
	rftmrfh	
	scsdemo	
	sorting	
Archive location —		



📩 SIXNET ISaGRAF	- Libraries	_ 🗆 ×
<u>F</u> ile <u>E</u> dit <u>T</u> ools	s <u>O</u> ptions <u>H</u> elp	
Function blocks	] 🗅 🖻 💼 🖶 🕾 📟 🚝 🛛	
O configurations	- ON_OFF	
IO boards	ption: - ON/OFF switch with deadband	
Functions Function blocks	age: - Function Block Diagram	
C functions	m	
C function blocks	on date: - 01/01/97	
	ד: - New ISaGRAF User	

Archive - Function bl	ocks	×
Workbench	Archive	
sxon_off	sxon_off	Back <u>u</u> p
		<u>R</u> estore
		<u>C</u> lose
		Help
		Compress
Archive location		
D:\ISAWIN\ARK\		Browse

### ISAGRAF 作为一个SIXNET项目的一部分

SIXNET提供了包含ISaGRAF Workbench在内的一个相互协作的Windows软件工具,这些工具包括Plant Floor, Control Room, SixTags等,以共享组态数据、简化工作。下面是创建实现SIXNET共享资源数据库的系统的步骤:使用ISaGRAF调试工具测试并下装程序。

### 步骤 1: I/O硬件组态

使用Plant Floor 软件对I/O站组态,给每个I/O点分配位号名,并设置系统特性。Plant Floor 包括test I/O等工具,以便启动后查找故障。

工具: SIXNET Plant Floor (ST-SPF)

### 步骤 2: 创建Windows共享资源数据库

如果要将I/O站链接到基于Windows 的应用程序(包括在计算机上运行的ISaGRAF),运行 Control Room IOmap 以创建共享资源数据库 (DLL);如果从Plant Floor启动IOmap,则项目将 被打开,IOmap数据库自动建立。

工具: SIXNET Control Room (ST-SCR)

### 步骤 3: 将I/O位号定义输出给ISaGRAF项目

利用SixTags,将Plant Floor I/O位号定义输出给ISaGRAF。参考本手册输出位号章节。

工具: Sixtags tag dictionary utility

### 步骤 4: 编写ISaGRAF 项目

按本手册说明创建您的 ISaGRAF 项目文件。

工具: ISaGRAF Workbench

### 步骤 5: 仿真调试该项目

使用ISaGRAF I/O 仿真器测试项目。参考本手册撌褂肐/O仿真器所陆凇\_

### 工具: ISaGRAF Workbench

### 步骤 6: 使您的系统在线运行

使用调试工具下载并测试程序。也可使用SIXNET Control Room 和 Plant Floor软件的强有力诊断工具。

工具: ISaGRAF Workbench 和 SIXNET Control Room 及 Plant Floor

关于ISaGRAF项目管理器的详 细信息,参考ISaGRAF 用户指

南

1/3

### 本章您将要:

- 回顾ISaGRAF项目管理器元素
- 学习归档功能
- 创建ISaGRAF培训项目(Training)

### 本章开始前,您应当:

完成了本手册的 安装和介绍 和 ISaGRAF概述 章。

### ISAGRAF项目管理窗口:

一个ISaGRAF项目是一组程序(程序,子程序,函数,等等)的集合,用于控制一个过程。一个项目对应 于在一个目标控制器中运行的一个完整的过程。缺省情况下,ISaGRAF项目存储在:\ISAWIN\APL 目 录下。

ISaGRAF项目管理窗口布局与说明如下:



ISaGRAF项目管理窗口分作两部分,上面部分是项目列表,下面部分是项目说明。项目说明描述项目 创建的日期和时间,作者名,内容和目的等。

在项目管理窗口可以:打印项目,保存修改历史,归档和恢复项目,编辑项目说明等等。参见 ISaGRAF用户手册的相关章节。

### 本手册创建的项目:

本手册将帮助您建立一个简单的应用,使用ISaGRAF Workbench监视马达的运行。

相应的SIXTRAK Plant Floor 文件: TRAINING.6pj,包含创建ISaGRAF项目所需的所有的I/O组态。使用的I/O有:

Start:	DI0	启动马达
Stop:	DI2	停止马达
Safety_Gate:	DI3	防止马达在危险条件下运行的安全门
Motor:	DO0	马达状态
High_Alarm:	DO1	马达温度过高报警
Motor_Temp:	AI0	马达温度

TRAINING.6pj也可能包含本手册步需要的位号。这是为您进一步开发应用所需准备的。

### 开始:

双击	ISaGRAF - Project Management         File       Edit         Edit       Return         Image: Select project group         New project         Bename         Copy         Delete         Archive         Exit         Date of creation : 5/5/96         Version number       : 1 - ISaGRAF V3.03         Description       : A sample program for use with the SIXTRAK sales
然后 → 将程序名定义为 TRAINING. 然后 → 点击 OK.	Create new project     X       Name:     TRAINING       IO configuration:     OK       (none)     Image: Cancel

### 创建ISaGRAF项目

### 3/3

然后 →

点击 **重** 项目说明图标(**Project Descriptor**),在对话框中输入你的 信息。

选择 **File**, **Save** 及 **Exit** (保存, 然 后退出)。

📐 ISaGRAF - TRAINING - Project descriptor	_ 🗆 ×
<u>File Edit Options H</u> elp	
▲ ₩ ■ ≤ ■	
Reference : TRAINING	<b></b>
Author : New ISaGRAF User	
Date of creation : 1/12/97	
Version number : 1 - ISaGRAF 3.20-B11	
Description : Training Project	
т	
L	
	_
7	

### 然后 →

将项目管理窗口最小化。

关于 Sixtags 的详细信息,请参

考在线帮助

2

1/5

### 本章我们将要:

- 使用从SIXNET Sixtags 工具,从Training.6pj项目 将标签输出给ISaGRAF项目
- 确认标签限制
- 重新打开ISaGRAF TRAINING项目,确认输出成功

### 本章开始前,您应当:

完成了**创建ISaGRAF项目**章, ISaGRAF TRAINING项目已被建立。对SIXTRAK硬件及其组态工具应有所了解。

### SIXTAGS工具 -- 一个共享标签数据库:

在SIXNET Plant Floor组态文件(.6pj)中,创建了模块及I/O标签名,使用Sixtags应用工具的输出功能,标签名可方便地输出到其他的Windows应用,如*Citect, Intellution FIX,*.CSV格式,当然也包括ISaGRAF。

📓 SIXNET I/O Tag Dictionary - TRAINING.6PJ				
<u>F</u> ile <u>E</u> dit	<u> ⊻</u> iew <u>D</u> efine <u>H</u> elp			
<u>E</u>	🖻 🔩 Pr. 🕄 🔍 🎽			
	SIXNET I/O Tag Name	Tag Description	-	
1	Motor_Temp	Motor temperature in degrees		
2	Motor	Motor status		
3	High_Alarm	Alarm if motor temperature reaches 200		
4	Start	Start motor		
5	Stop	Stop motor		
6	Safety	Disables motor to run if unsafe		
7				
For Help	o, press F1			

记录在ISaGRAF输出文件的标签包含Plant Floor中组态的I/O标签名和模拟I/O量程值。利用输出功能,可减少或取消在不同应用程序中建立和维护标签名的要求。

### 开始:

从Sixtrak程序组中启动SIXNET Sixtags工 具。 然后 →

选择File, Open, Project, 然后选 TRAINING.6pj。

File Open		? ×
File <u>n</u> ame: training.6pj samples.6pj stdemo.6pj temp.6pj template.6pj test.6pj training.6pj training.6pj training.6pj training.vert	Eolders: c:\sixtrak c:\ sixtrak excel isa96~1 release	OK Cancel <u>H</u> elp N <u>e</u> twork
List files of <u>type:</u> Project Files (*.6pj)	Dri <u>v</u> es:	

### 然后 →

选择Define, Tags 和Links。

### 然后 →

选择Define, Restrict Names。

### 然后 →

选择ISaGRAF Restriction,点击OK

这个特性是确认所选择标签名是否符合ISaGRAF (IEC 1131-3)标准。

这时,Safety Gate标签变为黄色,表示它不符合标签名限制。

SIXNE	T I/O Tag Dictionary - TRAINING.6PJ	
File Edit	View Define Help Bach Dan El Man	
	SIXNET I/O Tag Name	Tag Description
1	Start	Start motor
2	Stop	Stop motor
3	Safety Gate	Disables motor if unsafe to run
4	Motor	Motor status
5	High_Alarm	Alarm if motor temp is too high
6	Motor_Temp	Motor temperature
7	Core_Temp	Temperature at core of tank
8	Heat_Valve	Heat release valve
9		
10		×
For Hel	p, press F1	Active

<b>1/</b> 0 Ta	🖬 1/0 Tag Definitions 🛛 🕅 🕅					
<u>W</u> indow	<u>E</u> dit <u>V</u> iew	<u>D</u> efine <u>H</u> elp				
	Station	Autoload		Station Name	I/O Type	Register 🔺
	Prefix	Update Stations				Address
1		<u>R</u> estrict Names		ET_Gateway	Discrete In	0
2		Insert Underscores	Ì	ET_Gateway	Discrete In	1
3		Safety Gate		ET_Gateway	Discrete In	2

		Restrict Tags	Windo	~		×		
		Restriction	Types:			_		
		Custom I	Restric	tions	<u>U</u>	K I		
		@ ISaGBA	- F Bestri	ictions	<u>C</u> an	cel		
			ncəu	ctions				
					<u><u>H</u>e</u>	lp		
		Restriction	s:					
		Duplicat	e Tao I	Names No	t Allowed			
					N			
		M <u>Spaces</u>		John Char	ag Names			
				upna cnai				
		IM <u>U</u> pper/L	ower L	ase are E	quivalent			
		📕 🕅 Limit <u>T</u> ag	g Name	s to	Charac	ters		
		Limit Pre	fix Nan	nes to 8	Charac	ters		
		🛛 🕅 Limit Pre	fix+Tag	g to 16	Charac	ters		
( <b>170 T</b> a /indew	Edit View D	lefine Help						×
	Station	I/O Tag Na	me	Stati	on Name	1/0 T	уре	Register 🔺
_	Prefix			<b>FT 0</b> .			<u>.</u>	Address
2		Start Ston		EI_Gatew FT Gatew	ray rav	Discrete	ln In	1
3		Safety_Gate		ET_Gatew	/ay	Discrete	In	2
4		Motor		ET_Gatew	/ay	Discrete	Out	0
5		High_Alarm Motor Temp		EI_Gatew ET Gatew	ray rav	Discrete Analog In	Out	1
7		Core_Temp		ET_Gatew	/ay /ay	Analog In		1
8		Heat_¥alve		ET_Gatew	/ay	Analog O	ut	0
9								
		<u>A</u> utoNa	ne	Add <u>N</u>	ext	<u>C</u> lose	1	2
ISaG	RAF Expo	ort Options						×
_ CI	ation							
- 21	auun.						<u>o</u> k	
∣⊙	All Sixtag	JS						
		e:	Cato			7	<u>C</u> anc	el
10	Unly Une	station:		way	<u></u>			
$  \circ$	All 170 in				-	1 🗆	<u>H</u> elp	,
1~							dit IO	man
<u>P</u> r	oject: —							
IS	aGRAF A	pplications Di	rector	V:				
С	:\ISAWIN			-	-	1	Brow	se
IS	aGRAF P	rojects:	_			<u>B</u> u	ın ISa(	GRAF
TRAINING								
<b>∆</b> c	lvanced F	eature: ——						
	Export In	puts as ISaGI	RAF O	utputs				

### 然后 →

双击Safety Gate 标签,将它改为 Safety\_Gate,关掉该窗口。

∐ir

然后 →

选择 File, Save 保存。

### 然后 →

选择File,Export,ISaGRAF。

在弹出的对话框中,选择要输出的标 签、项目所在的目录、项目名 (**TRAINING**)。如右图。

点击 **OK**。

ISaGRAF输出窗口允许选择所要输出 的I/O标签,可选择project, station, 或 lomap。详细信息请参考 SIXTRAK在线帮助。

### 向ISaGRAF项目输出Sixtags

4/5

这时,Sixtags确认所输出的标签数。 然后 → 点击 OK。

然后 →

退出 (Exit) Sixtags。

### 然后 →

最大化ISaGRAF项目管理窗口。

### 然后 →

双击项目名(training)图标,打开程 序管理窗口。

ISaGRAF - Project Management       File       Edit       Project       Options       Help       Emile       D       Emile       Emile       D       Emile       Emile	
a_sample       training       isa96       rfarray       rfbool       rfcjfb	•
Reference : TRAINING Author : New ISaGRAF User Date of creation : 1/12/97 Version number : 1 - ISaGRAF 3.20-B11 Description : Training Project	×

Please remember to remake the Workbench Project.

SIXNET I/O Tag Dictionary

**!**}

ISaGRAF Export Found :

2 Analog Inputs 1 Analog Outputs 3 Digital Inputs

2 Digital Outputs

### 然后 →

选择File, Dictionary。

### 然后 →

使用工具栏上 🔤 按钮,显示布尔I/O 变量 (离散).

🔖 ISaGRAF - TRAIN	NING - Global I	booleans		х	
<u>File E</u> dit <u>S</u> earch <u>O</u>	ptions <u>H</u> elp				
	≌ 0 0 0	9 0	20020304	3	
Name	Attrib.	Addr.	Booleans		
Start	[input]	0000	Diologianis motor	-	
Stop	[input]	0000	DI1:Stop motor		
Safety_Gate	[input]	0000	DI2:Disables motor if unsafe to run		
Motor	[output]		DO0:Motor status		
High Alarm	[output]	0000	DO1:Alarm if motor temp is too high		
				$\mathbf{v}$	
Start (* DI0:Start motor *) @0000 [input] (Off,On)					

### 向ISaGRAF项目输出Sixtags

5/5

### 然后 →

使用工具栏上按钮,显示整形/实型 I/O变量 (模拟).

字典应包含所有输出的标签。

然后 →

关闭 Dictionary.

### 然后 →

关闭Training项目。

现在,您已为建立ISaGRAF应用做好 了准备。

A M M M M M M M M M M M M M M M M M M M					
♥▥◳▥▯▯◓▯◓▯					
Minnent					
Integers/Reals lotor_Temp					
10:Motor temperature					
11:Temperature at core of tank					
00:Heat release valve					
<b>v</b>					
scale_temp (* Scaled value of Motor_Temp *)					

梯形图编辑器的详细信息请参考

ISaGRAF用户手册的相关章节。

1/12

### 本章我们将要:

- 学习Quick Ladder (快速梯形图) 工具
- 创建一个简单的梯形图程序
- 建立一个内部变量 Scale\_temp
- 保存并确认这个程序
- 向这个程序将入一个功能块安全特性

### 开始本章前, 您应当:

已经完成了本手册的向ISaGRAF项目输出位号章节,并且有较好的梯形图逻辑编程基本概念。

### 梯形图编程基础:

一个梯形图程序表示为由接点(离散输入)和线圈(离散开关输出)组成的一个阶梯列表。下面是梯 形图(LD)的基本元件:

- ▶ 阶梯头 (左栏杆) 每个阶梯从左栏杆开始,它表示撜鏀状态。阶梯可有一个逻辑名(Label),用于跳转指示。当放置一 个接点时,左栏杆自动产生。
- **----** 接点

接点是按一个布尔变量状态而修改的布尔数据流,变量名显示在接点符号上面。ISaGRAF 支持下面的 接点类型:

▲▶.....直接接点(常开接点)

╋╋

- ╋╋
- ₩.....下降沿触发接点
- **-()** 线圈

线圈表示一个动作,由其左边的链接状态决定。符号上显示出变量名。ISaGRAF支持下面的线圈类型:

- **{}**.....直接线圈

- ●.....下降沿检测线圈

- 2/12
- **む** 功能块 在一个 LD 程序中,一个块表示一个函数、一个功能块、一个子程序、或一个操作符。其第一个输入和 输出参数(开关量)总是连接到阶梯上。
- ➡ 阶梯尾(右栏杆)一个阶梯的结尾,当放置一个线圈时,右栏杆自动产生。
- 跳转符号
  跳转符号指向一个具有标号的目标阶梯,位于一个阶梯的末端,当该阶梯为真时,执行跳转。
- 返回符号
   返回符号位于一个阶梯的末端,当该阶梯为真时,表示该程序必须停止。(扫描结束条件)

### HFOH 快速梯形图编辑器限制:

ISaGRAF快速梯形图编辑器不允许在线圈的右边插入接点或线圈。如果几个输出由同一个阶梯控制,相关的线圈应平行画出。



*详细信息请参考快速梯形图编辑器在线帮* 助。

### 梯形图编程实例

本例中,我们将建立一个基本的梯形图:



\_ 🗆 ×

x

<u>0</u>K

Cancel

<u>H</u>elp

\_ 🗆 ×

•

### 开始:

从ISaGRAF项目管理器窗口中,打开 - 🗄 ISaGRAF - TRAINING - Programs File Make Tools Debug Options Help TRAINING项目。(双击图标 <u>E</u>dit Return s The mode and set of the set of 📷 training 🔾 **D**ictionary Parameters Diary 然后 → <u>N</u>ew Program comment text 从Training编程窗口,选择 File, <u>R</u>ename/Move Arrange programs New. <u>С</u>ору Copy to other project Delete Import from library Export to library E<u>x</u>it 然后 → New Program Location in hierarchy tree: 填写程序名: RunMotor, Name: RunMotor Language: • Beginning of cycle (top level) 选择 Quick LD 作为编程语言, C Seguential (top level) ○ <u>s</u>fc C End of cycle (top level) 选择 Beginning of cycle 作为其在 ○ <u>F</u>BD/LD O Function 层次树中的位置。 • Quick <u>L</u>D C Function block  $\bigcirc S\underline{I}$ C Chil<u>d</u> of: 点击 OK。 O]L 2 关于程序层次详细信息,参考本手册 ISaGRAF概述章节,或ISaGRAF用户指 南。 SagRAF - TRAINING - Programs 然后 → <u>File Make Tools Debug Options Help</u> 双击 🚾 RunMotor 编程图标,进入 RunMotor QLD (快速梯形图编程)环境。 Begin: Begin: RunMotor (Ladder Diagram)

### 快速梯形图编辑器环境:

ISaGRAF快速梯形图编辑器(Quick LD)可以使用键盘和鼠标容易地建立梯形图程序。它按照逻辑栅格自动地连接和布置阶梯。

本培训假定大多数编程由鼠标完成。关于功能键和键盘的用法请参考ISaGRAF用户指南。



位置指示器

### 5/12

### 然后 →

点击 <u>F2-1 H-</u> 图标,插入第一个阶梯。

注: ISaGRAF 自动将左栏杆放置到 (1,1)点,并自动插入接点、线圈 和右栏杆。

(点击 🗱 图标可加上栅格).

# Image: IsaGRAF - TRAINING:RUNMOTOR - Quick LD Program File Edit Inset Options Help Image: Imag

### 然后 →

双击接点小格(1,1),以给接点分配一 个变量。

### 然后 →

选择 Stop 并点击 OK.

(如果没有显示出布尔变量,点击按 钮 🔲 即可列出。)

### 然后 →

点击 **[?**],将 **Stop** 设为常闭变量 接点 (**↑⊦**)。





### 然后 →





6/12

### 然后 →

将Start变量分配给它(2,1)。



插入平行接点,选择小格(2,1),然后 点击 4 4 5 图标,一个平行接 点将自动放置到小格(2,2)。

### 然后 →

双击小格(2,2),将变量Motor分配到 该接点。

从Select Variable窗口,选择Motor 并点击OK.

### 然后 →

双击小格(3,1),将Motor变量分配到 线圈。

### 然后 →

从Select Variable窗口,选择Motor 并点击OK.

程序的基本轮廓就完成了。快速梯形 图编辑器已完成了所有的连接。



7/12

注意:小格(0,0)显示的(\*\*)是一个 注释栏,可以双击它输入简单的程序 说明。

### 然后 →

双击小格(0,0)处的(\* \*),写入: Simple command to turn on the motor.

### 然后 →

点击 **OK.** 

🕮 ISaGRAF - TRAINING:RUNMOTOR - Quick LD Program 📃 🖬 🛛							
<u>File Edit Insert Options H</u>	elp						
🗈 🖆 📈 💷 🛠 🔟 🕯	8 * 6 6	₽\$- {?}-	Q 🗉 🏢	8			
		+		•		Ŧ	
(* <u>*</u> )							
-12	Stop	<sup>⁺</sup> Si	tart	+	Motor	+	
│		I	<b>—</b>		_< >		
+ +		t Me	otor	t		+	
							-
		-		-		*	ĿĒ
F2: <b>-1 F</b> -1 <b>F</b> -3: -1 <b>F</b> -1 <b>F</b> -1 <b>F</b> -4: <b>T</b>	T F5: -OH	F6: <b></b>	F7:-] [-	F8: T		×F9:-return>	
					U		
pos=0,0							
Bung Commont							$\sim$
nung comment							
Simple command to	turn on the	motor.					
4							
	04			•	-1		

### 然后 →

然后 **→** 点击 **≧** 图标,保存该程序。

这时,日记工具窗口出现,它允许您 在每次修改程序时做出记录,请写入 适当信息,点击 **OK**.

### 然后 →

点击 **OK.** 

Update diary for program runmotor	×
Created program to turn motor on.	
	<u>C</u> ancel
<u>र</u>	Þ

8/12

然后 →	🗣 ISaGRAF - TRAINING - Code Generator
3.21	<u>File Options Edit H</u> elp
点击 🎿 (确认)图标,检查程序错	Verifying variables declarations
误,如有错误,则用红字表示出来。	Verifying RunMotor
双击该字,可进入出错处。	No error detected
	Resetting temporary files
然后 →	Building configuration data Linking object files
当程序被确认并无错时,退出代码生 <sup>成要</sup>	Relocating object files Post-compiling code No every datasted
//ATH 0	No error detected
关闭确认窗口,返回 RunMotor 程	No error detected.

序。

如果确认时有错误,双击该红色错误 信息,ISaGRAF将带您到出错处。修 改差错,重新保存和确认之。

## 程序中加入功能块:

现在我们要利用加入功能块实现安全性能:监视马达温度,当其过热(高于**200**℃)时,发出报警声,并关闭马达。

### 然后 →

在 **RunMotor** 程序中选择小格 (0,4)。

### 然后 →

点击 📴 🕂 图标。

### 然后 →

双击功能块中心,将弹出功能块列 表。

≣⊞ ISaGF	RAF - TRAINING:R	UNMOTOR - Quick	k LD Program			_ 🗆 ×
<u>F</u> ile <u>E</u> dit	<u>Insert Options</u>	elp				
		8 > 6 1	₽1?} Q.E	•		
(† Oinsula	<b>.</b>	•• **	+	Ŧ	-	Ţ,
(* Simple	command to turn o Stop	n the motor. *) Start	+ Motor	+	+	+
۲	F <sup>+</sup> −1/F	Motor		+	+	+
۲	+	+ + +	+	+	+	+
(* *)	+	Ĭ	<b>I</b>	+	+	+
F	+		Ţ	+ <	+	+
F	+	Ļ	1	+	+	+ 💌
F2: <b>-  [-]</b>	F3: -] F4: -]			E-10- F8: T		×F9:- <u>return</u> >
pos=2,4			43			

Do you want to exit the Code Generator now ?

<u>Continue</u>

E<u>x</u>it

9/12

然后 →

从功能块列表中选择 >= 功能块并 点击 OK 。

注意: En (使能) 自动加在功能块左上角,在梯形图程序中,功能块必须接受一个布尔输入,以激活(使能)它。En为真时,功能块激活。





由于 **Motor\_Temp** 原始值为0-32767,而我们报警是以200℃,所 以应创建一个内部变量(scale\_temp) 以将Motor\_Temp表示为工程单位。



🗞 ISaGRAF - TRAINING - Global integers/reals - 🗆 × <u>File Edit Search Options Help</u> 🖷 🛰 🗈 💰 ≙ ○ ◯ ໟ ๒ 屁 ◊ ഈ ഈ ㅌ | 8 Integers/Reals Motor Temp linput.integer1 0000 Core\_Temp Heat\_Valve [input,integer] loooo Al1:Temperature at core of tank [output,integer] 0000 AO0:Heat release valve Motor\_Temp (\* Al0:Motor temperature \*) @0000 [input,integer] Unit=deg

选择 Edit, New.

10 / 12

### 然后 →

按下列参数填写弹出窗口: Name: scale temp

Name: scale\_temp Comment: Scaled value of Motor\_Temp Unit: Deg Attribute: Internal Format: Integer

然后 →

点击 Store, 然后 Cancel。

选择 File, Exit 及 Save。

### 然后 →

双击小格(1,5)。会显示出布尔(离散) 值来。

Integer/Real \	/ariable				×
Name:	scale_temp		Netwo	rk Address:	
Comment:	Scaled value	of Motor_Temp			
Unit:	Deg		Conversion:	(none)	
Attributes C In <u>t</u> ern C Input	al	Format © Integer © <u>R</u> eal	(standard)	<u> </u>	<u>S</u> tore
C <u>O</u> utpu	ıt				<u>N</u> ext
C Const	<u>a</u> nt	Initial value: 「R <u>e</u> tain	0		Previous



### 然后 →

点击 🔤 图标,以显示整形/实型变 量。

### 然后 →

选择**scale\_temp**变量,点击**OK。**将 它分配到功能块的第一个输入上。

### 然后 →

双极小格 **(1,6)**.

应显示出整形/实型变量。

然后 →

写入 200. 点击OK.

Scope: (Glo	bal) 🔽 🖸 🖄 🖾 Integer/Real 💌
200	
Core_Temp	Al1:Temperature at core of tank
Heat_Valve	A00:Heat release valve
Motor_Temp	AIO:Motor temperature
scale_temp	Scaled value of Motor_Temp

### 11 / 12

现在,功能块作用是: ∋曬ISaGRAF - TRAINING:RUNMOTOR - Quick LD Prog \_ 🗆 × <u>File Edit Insert Options Help</u> 如果 scale\_temp 等于或大于 200, **影** (?) QE 则输出为真。 Stop Start Motor 4 V İ 4 F Motor -+ 然后 → \*) en 双击阶梯的线圈格(4,4)。 scale\_tempin1 200 E9 «F9:-(return) pos=4,4 然后 → Select variable X 点击 🔤 图标,显示出布尔变量。 🖪 🖾 🖄 🖾 Boolean Scope: (Global) • • High\_Alarm D01:Alarm if motor temp is too high 然后 → High\_Alarm D01:Alarm if motor temp is too hig DOO:Motor status Motor 选择High\_Alarm并点击OK,将其分 Safety\_Gate D12:Disables motor if unsafe to run 配给块的输出。 Start DIO:Start motor Stop DI1:Stop motor <u>0</u>K <u>C</u>ancel ﷺ ISaGRAF - TRAINING:RUNMOTOR - Quick LD Progra 然后→ \_ 🗆 × <u>File Edit Insert Options H</u>elp 光标移到Stop接点(1,1)上。 -然后 → (\* Simple command to turn on the motor \*) . Start Motor 4 F 点击图标 F2:-] [--] [-在Stop左面插 Motor 4 1 入一个接点。 <u>(</u>\* \*) High\_Alarm >= scale\_temp-in1 + 200 pos=1,1 Stop: DI1:Stop motor

### 12 / 12

# 然后 → Select variable 双击这个新的接点。 Scope: [[ilobal] 然后 → High\_Alarm, 点击 OK. 然后 → 小选择常闭接点给 点击 ① 图标,选择常闭接点给 High\_Alarm DO High\_Alarm. Motor DO Safety DI2 Statt DI0 Stop DI1 Stop DI1 Motor DO Safety DI2 Statt DI0 Stop DI1 /

### 然后 →

保存并确认该程序(Save / Verify)

### 然后 →

在程序确认为无错后,退出代码生成器。(**Exit**)

关闭确认窗口,并关闭**RunMotor**程 序窗口。

现在,我们将准备学习下一章 **快速梯** 形图/功能块图编辑器"

0.1	
Select va	mable X
Scope:	(Global)
High_Al	arm
High_Ala	arm D01:Alarm if motor temperature reaches 200
Safety	DI2:Disables motor to run if unsafe
Start Stop	D10:Start motor D11:Stop motor
	<u> </u>
⊒⊯ ISaGRAF	- TRAINING:RUNMOTOR - Quick LD Program
Eile Edit Ins	set Options Help
(* Simple co	
	High_Alarm Stop + Start + Motor_On +
F	Mator_On + +
(* *)	Rung Comment
	Monitor temperature, sound alarm and turn off motor.
r I	
ŀ	* * * * *
F	* * * * *
F2: <b>-] []</b> [- F3	
pos=0,3	
🐔 ISaG	BAF - TRAINING - Code Generator
File Op	otions <u>E</u> dit <u>H</u> elp
Verifyir	ng variables declarations
Verifyir	ng runmotor
No erro	
Co	de Generator
	No error detected.
	To you want to exit the Lode Generator now ?
	<u> </u>

关于编辑器的详细信息,请参考

用章节或在线帮助

ISaGRAF用户手册的LD/FBD编辑器使

1/10

### 本章我们将要:

- 学习梯形图/功能块图(LD/FBD)编辑器
- 使用LD/FBD编辑器建立两个程序
- 保存和确认程序

### 开始本章前, 您应当:

完成了本手册的概述章,以及快速梯形图编辑器使用章。

### 功能块图编程基础:

一个功能块图(FBD)描述了输入变量和输出变量间的一个函数,执行不同操作的功能块表示为一个图 形集,功能块图由名称、输入参数、输出参数组成为一个长方形图符,如:



输入和输出变量由连接线连到块上,并应与要求的数据类型一致(Boolean, integer, real, etc.)。输入连 到左边,可以是常数、内部变量、输入变量、输出变量;输出连到块的右边,应为内部或输出变量。



与梯形图一样,功能块图总是从左向右、从上到下解释。可以将功能块图与梯形图一起使用

### 关于功能块图及梯形图编程的详细信息,请参考 ISaGRAF 语言参考手册

### 实际练习:

本练习中,我们将建立两个简单的功能块图程序。

第一个程序(Scale),将原始模拟输入值 Motor\_Temp (1 - 32767)按比例转换为工程单位值(1 - 300℃),结果赋给内部变量 scale\_temp(在QLD章中创建)。

第二个程序(Safety),是一个安全控制,检查目标状态以确保控制安全。

### SCALE程序开始:

从Training程序窗口,选择 File, New.



### 按下列参数创建Scale程序:

然后 →

Name : Scale

Language: FBD/LD

Location: Beginning of Cycle

### 点击 OK.

说明:在ISaGRAF目标循环中,为使 RunMotor程序使用scale temp变量 的当前值, Scale程序应放到 RunMotor程序的前面。

### 然后 →

选择 File, Arrange Programs.

选择 Scale 并点击 Up.

点击 OK.

### 然后→

双击 Scale 程序。

当移动鼠标时,可观察屏幕左下角的 位置指示。

### 然后 →

选择 Options 并确保选中 FBD toolbar 和 Auto input.

### 然后 →

选择 Options, Layout 并确保选中所 有的 Workspace 和 Document 选 项。



\_\_ Program co<u>m</u>ment tex Rename/Move Arrange program

Copy to other project

Import from library Export to library

Сору

Delete



<u>Up</u>

Down

将1~32767的原始值按比例转换为整形值1~300:

<u>原始值x 300</u> = 比例转换值

32767

Scale程序将原始的Moter\_Temp值转换为工程单位 (℃):

### <u>Motor\_Temp x 300</u> = scale\_temp

32767



点击工具栏上 1 按钮。

### 然后 →

双击小格 (12,2).

	Ē	S	aG	RA	F -	TF	łΑΙ	NII	NG	:50	AI	E ·	- F	BD	/LI	) P	rog	jrar	n												-		х
	<u>F</u> ile		<u>E</u> di	it	Opt	tion	s į	Hel	p																								
100000000		đ	2		7	<u>_</u>	Q •			1	₽	9	<	Ē	6		Q			9	3												
SCOUNCE IN	ню	]	R	3 8					Ø	۶ <b>۲</b>	9	ORET		) <b>+</b> [	13	Ŧ	ŀ	Ð		Ð	1	ga	in				•						
Γ		÷																		ᇩ						÷					÷		
																				Ir	ser	t fu	nct	ion	blo	ćk							
L			÷	÷					÷	÷			·		·			÷	·	·		·	÷	÷	÷	·	·	·	÷	÷	÷		
L			÷	·									·					÷	·			·	÷	÷	÷	·	·		÷	÷	÷		
L			÷								÷		·									·		÷		·	·						
L			÷								·													÷		·	·						
			÷																														
	•	•	·	·	•	•			•	·	•	•	•		•	•	·	·	·			•	•	·	·	•	•		•	·	·	·	

### 然后 →

在功能块选择窗口,选择 MUL 功能 块。

注意:输入变量的数目可选,缺省为 2。

### 然后 →

点击INFO看该块的结构,及输入/输 出参数数据类型。

### 然后 →

点击Note看该块的详细信息。

Select func	tion block		×
Block:	MUL		ĸ
	mid 🔺		
Number of	min	<u>C</u> ar	ncel
	mien		
🗹 <u>D</u> isplay	Msa 🚽	<u>I</u> n	ifo 📗
	MUL		

Parameters - 'x'	×
- 00) ← Z III 01 ← Z … 02 ← Z IN 50 → Z Q	<u>D</u> K <u>N</u> ote

### 然后 →

将鼠标放到功能块的第一个输入脚左 边。

此时IN (Integer/real) 出现在屏幕左 下角位置标志的右边, 表示该输入所 要求的类型是实型/整型。

300 I S	GaGR	AF - TR	AINI	NG:SC/	ALE -	FBD/LI	) Progra	m		-		
<u>F</u> ile	<u>E</u> dit	Options	: <u>H</u> e	lp								
Đ	ò	X 🗉	¢.	8		E 2	] Q	<u> </u>	ĩ			
ЧЮ				1	) RET		₽₽	E	1 gai	in		
											. 🔺	
				· [		*					.	
				+								
	•			· -[				- ·		•		
•	<u> </u>	·									.▼ ▶	
pos=	11,3		IN (I	nteger/re	al)							
Sele	ct va	riable										X
Sco	ope:	(Globa	al)		•		20) 0	🖾 In	teger	/Real		<b>_</b>
Mo	tor_T	emp										
AIO	:Moto	or tempe	eratu	e								
Cor	e_Te	mp .	AI1:T	empera	ature a	at core	of tank					
Hea	at_Va	lve .	AO0:	Heat re	lease	valve						
Mo	tor_l	emp	AIU:N	lotor te	mpera	ature						

然后 →

然后 →

在变量选择窗点击 🔤 图标,选择 变量 Motor\_Temp, 点击OK。

### 然后 →

连接变量 MOTOR\_TEMP 到功能块 第一个输入:

点击 📑 图标(画连接线),在变量 右侧按下鼠标左键,将鼠标拖到MUL 功能块的第一个输入脚。



### 使用梯形图/功能块图编辑器

5/10

### 然后 →



### 然后 →

在变量选择窗口直接写入缺省变量 300. 点击 OK.

### 然后 →

选择 **按**扭,将变量连接到**MUL** 功能块的另一输入脚。

该功能块的输出为变量 Motor\_temp 与 300 的乘积。该输出应被32767 除,才可返回比例后的值。

### 然后 →

点击 **正** 按钮加入**DIV**(除法)功能块。

### 然后 →

不从弹出框, 而是使用下拉菜单选择 功能块。

### 然后 →

选择DIV功能块,并点击小格 (20,3) 一次。

elect variable	
Scope: (Glo	bal) 🗾 🗖 🖂 🖄 🕅 Integer/Real
300	
k ₽	
Core_Temp	Al1:Temperature at core of tank
Heat_Valve	A00:Heat release valve
Motor_Temp	AIO:Motor temperature
	Scaled value of Motor, Temp



### 使用梯形图/功能块图编辑器

6/10

DIV功能块用第二个输入去除第一个 输入。

然后 →

使用 按钮连接MUL功能块的输出到DIV块的第一个输入,小格(16,4) 连接到 (20,4).

### 然后 →



然后 →

在变量选择窗口, 输入 32767.

### 然后 →

点击 **OK**.

聽 ISaGRAF - TRAINING:SCALE - FBD/LD Program	_ [ 0]
<u>File Edit Options Help</u>	
▶ ▶ ▼ ● ◆ ■ ◆ ■ ◆ ■ ◆ ■ ◆	
₩ ╘ॎॎॻॻ ◈♥? Ё ७ + 1 ₪	
Motor_Temp]	/ 
o I	
Select variable	>
Scope: (Global) ▼	🔤 Integer/Real 🔹 🔹






#### 使用梯形图/功能块图编辑器

7/10

#### 然后 →

点击 🔲 按钮和小格 (28,5).

#### 然后 →

在变量选择窗口,选择 scale\_temp. 点击 **OK**.

#### 然后 →

使用按钮 将 scale\_temp 连接 到DIV功能块的输出,即(24,5) 连到 (28,5).

#### 然后 →

点击**匪**按钮插入程序注释,光标移 至(1,0)。

#### 然后 →

保持鼠标左键按下,并将鼠标拖至小格 (20,1),然后放开。

# **然后→**

写入注释: 揝cales motor temperature to value between 1 and 300"

点击 **OK**.

Select va	riable					x
Scope:	(Global)	-		3 🔤 Int	eaer/Rea	
scale te						
Scaled v	value of Motor 1	emn				
Core Te	mp Al1:Tem	perature at	core of ta	nk		
Heat_Va	alve AOO:Hea	at release v	alve			
Motor_T	emp AlO:Moto	or temperatu	ire or Tomp			
SCOIC_IC	mp scaled (			7		
≋ISaGRAF -	TRAINING:SCALE - FB	D/LD Program				
ile <u>E</u> dit <u>O</u> pt	ions <u>H</u> elp		9			
		10 VIII 13 7 10 10		-		
چ <del> ا</del> ارا	sert comment					
		*				
	Motor_Temp		Dr	/		
L		32	767	<u>ه</u> م	• •	scale_temp
os=3,0						
E <b>ISaGRAF</b> - ile Edit Opt	TRAINING:SCALE - FB	D/LD Program				
) <b>(</b> ) ()		1601	6			
Ю 🔓 🖽		[ଜ주관] 8		<u>-</u>		
'			+'			
	Motor_Temp	_ *	Ē	/		
	300	<u> </u>	767	2 0		scale temn
					u	
1						
os=18,1						
Jommen						
Scales	motor tempera	tuie 1-300	deg			ОК
						Cancel
						Cancer



### 然后 →

保存并确认该程序。

### 然后 →

当确认无错误后,退出代码生成器。 关闭确认窗口及**Scale**程序窗口。



如果您没有完成本手册使用快速梯形图编辑器章,请参考本章第8页如何确认程序。

#### SAFETY程序:

本程序目的是:检查在写入任何输出前,接点Safety\_Gate是关闭的。如果Safety\_Gate是开的,马达就关上。把此程序放到ISaGRAF目标循环结束部分,这样即使RunMotor程序执行ON命令时,也不对马达操作。

#### 然后 →

选择:	File,	New	Program
-----	-------	-----	---------

- 程序名: Safety
- 语言: FBD/LD
- 位置: End of cycle

#### 然后 →

打开Safety程序,并确保Options菜 单下的Auto-input选项选中。

点击 <u>+K</u> 图标,显示出梯形图工 具。

#### 然后 →

选择 **ト** 图标 (左栏杆)并将其放 置到小格(2,3): 点击小格。

Name: Safety Language: C SFC C EBD/LD C Quick LD C SI C IL	Location in hierarchy tree: © Beginning of cycle (top level) © Seguential (top level) © End of cycle (top level) © Function © Function block © Chil <u>d</u> of:	<u>D</u> K Cancel <u>H</u> elp
- ISaGRAF - TRAINI	NG - Programs	
File Make Tools Det	oug Options Help IEI mil  ॐ ‰ ដ⊳   com Նյα t≤,	\$4 & ⊒ °2
Begin: କଇଅ ଜାଙ୍କ	Scale RunMotor	
End: 🔁	Safety	
interstand State - TRAINING:SAF File Edit Options Help	ETY - FBD/LD Program	
14 X98 M&	*16 0.111 6	
71 🕅 🖬 🖉 🐔	፼ ፟ችቅ⊕ ዞ ነነላቀቀ ቋ	()()()()()()) -
		<u> </u>

#### 使用梯形图/功能块图编辑器

#### 9/10

#### 然后 →

选择 **小** 图标(常闭接点),点击 位置 (4,4)。

然后 →

从字典中选择**Safety\_Gate**变量(布 尔类型)。

然后 →

选择 (R) 图标(复位线圈),点击 位置 (9,4)。

然后 →

从字典中选择**Motor**变量(布尔类型)。

Select vari	able
Scope:	(Global) 🗾 및 🏳 🖄 🖽 Boolean 🗾
Safety_G	ate
D12:Disab	les motor if unsafe to run
High_Alar	m D01:Alarm if motor temp is too high
Motor	DOO:Motor status
Safety_G	ate DI2:Disables motor if unsafe to run
Start	DIO:Start motor
Stop	DI1:Stop motor

ጬISaGRAF - TR/	INING:SAF	ETY - FBD/LI	) Program						_ 🗆
<u>File E</u> dit <u>O</u> ptions	<u>H</u> elp								
1≧ 🖄 🖳	Ջ 🖬 🖗	* 🗈 👌	Q	8					
n 🕅 🗆	■ Ø♥	유민구	₽⊦	41-14-1	Nf fpf	ŧ	() ()	(R) (S)	H
								8	
								Insert	reset coil
Safety_	Gate								
	F								
	1								
Select variable		-							x
Select variable Scope: (Glo	bal)	<u> </u>		20	B	oole	an		×
Gelect variable Scope: (Glo Motor	ibal)	<u>•</u>		20	B	oole	an		×
Gelect variable Scope: [Glo Motor D00:Motor st	ibal) atus	<u> </u>		20	B	oole	an		×
Select variable Scope: (Gio Motor D00:Motor st High_Alarm Motor	atus DO1:Al	▼ arm if moto	) 🗇 (	⊇ ⊘ s too hi	🖼 B gh	oole	an		×
Select variable Scope: (Glo Motor DOO:Motor st High_Alarm Motor Safety_Gate	obal) atus D01:Al D00:M D12:Dis	arm if moto otor status sables moto	) 🗖 (	s too hi	🖼 B gh	oole	an		×
Select variable Scope: [Gld Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop	atus D01:Al D00:M D12:Dis D10:Sta D11:Sta	arm if moto otor status sables motor art motor op motor	or temp i	고 🔗 s too hij	B gh	oole	an		×
Select variable Scope: [Glo Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop	atus DO1:Al DO2:M DI2:Dis DI0:Sta DI1:Sta	arm if moto otor status sables moto art motor op motor	or temp i	s too hi	gh	oole	an		
Select variable Scope: [Gid Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop	atus D01:Al D02:Mi D12:Dis D10:Sta D11:Sta	arm if moto otor status sables moto art motor op motor ETY - FBD/LD	) 「」 or temp i or if unsa	s too his	gh un	oole	an		×
Select variable Scope: [Gik Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop	atus DO1:Al DO2:Mi DI2:Dis DI0:Sta DI1:Sta NING:SAF Help ₹ I I I I I I I I I I I I I I I I I I I	arm if moto otor status sables motor art motor op motor ETY - FBD/LD	or temp i or if unsa	s too hi afe to ru	gh un	oole	an		
Select variable Scope: [Giu Motor DO0:Motor st High_Alarm Motor Safety_Gate Start Stop Start Stop	atus DO1:Al DO2MI DI2:Dis DI0:Sta DI1:Sta NINCESAF Help ₹. ■ � ₹	arm if moto otor status sables moto art motor op motor ETY-FBD/LC ■ ■ ■ →	or temp i or if unsa	s too hi afe to ru	명h In		an () ()	R ()	
Select variable Scope: [Gir Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop Start Stop	atus DO1:Al DO1:Al DI2:Dis DI0:Sta DI1:Sta DI1:Sta Help ₹ <u>H</u> ⊕ ₹ 9 <b>■</b> Ø ₹9	arm if moto otor status sables moto art motor op motor ETY - FBD/LD 응 같 :	or temp i or if unsa	s too hi s too hi afe to ru = + 1 + + + +	gh In		aan	<u>@</u> (9)	
Select variable Scope: [Git Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop Ele Edt Options Ele Edt Options M M () Stop	atus D01:Al D02:Dis D10:Sta D11:Sta D11:Sta Help 2 III - Sta Help 2 III - Sta	arm if moto otor status sables moto art motor op motor ETY - FBD/LD ETY - FBD/LD Craw	or temp i or if unsa Program	s too hig afe to ru	gh Jn	oole	aan	<u>e</u> ()	
Select variable Scope: [Gik Motor DOO:Motor st High_Alarm Motor Safety_Gate Start Stop	atus D01:Al D02:Mi D12:Dis D10:Sta D11:Stc UNING:SAF Help R M @ 20 Sate	arm if moto otor status sables motor art motor op motor ETY - FBD/LD ETY - FBD/LD ETY - FBD/LD ETY - FBD/LD ETY - FBD/LD ETY - FBD/LD ETY - FBD/LD	or temp i or if unsa	s too hi afe to n	gh In	oole =	ean	R ()	



选择 图标 (连接) 将接点和 线圈连起来。

#### 然后 →

现在您的程序逻辑按如下工作:

如果**Safety\_Gate**是关闭的,则开关 保持关闭。常闭接点上不通过电源, 没有动作。

如果**Safety\_Gate**是打开的,电源通 过马达线圈并复位它。由于**Safety**程 序在每个执行循环的最后执行,任何 在完成循环更新输出前的马达动作将 被阻止。

#### 然后 →

保存并确认该程序。

#### 然后 →

确认无错误后,退出代码生成器。 关闭确认窗口,关闭**Safety**程序窗口。

通過 IS File	aGRAF - TRAINI	NG:SAFETY	- FBD/LD Pro	ogram				_[	×
न्तः गः			부루亜	F	4 F 4/F 4MF 4PF	₽	() () ()	(s)	+
	_								
	Safety_Gate	Motor R							
	•								
									•
<b>↓</b> pos='	17,10								•



#### 开始本章前,您应当:

已完成*使用Quick Ladder编辑器和使用LD/FBD编辑器*章节,已创建了 ISaGRAF TRAINING 项目的程序。

#### 编译ISAGRAF应用:

Make菜单的命令用于运行代码生成器,及输入选项和附加数据。运行命令前必须正确设置目标代码生成选项。

生成目标代码前,所有未经确认的程序将先进行语法差错检查。ISaGRAF包含一个增量编译器,它对 上次编译过的、未修改的程序不再进行编译。



#### 编译ISaGRAF应用

2/4

从ISaGRAF Training程序窗口: 选择 Make, Compiler Options.

Code generation parameters	×
Targets:	
SIMULATE: Workbench Simulator	Select
ISA68M: Run-time for SIXNET gateway	<u></u>
ISA86M: Run-time for Windows (3.1, 95 or NT)	Unselect
V Use embedded SEC engine	
Uptimizer:	Default
Run two optimizer passes	
Evaluate constant expressions	
Suppress unused labels	
☑ Optimize variable copying	
Coptimize expressions	
Suppress unused code	
Optimize arithmetic operations	ок 🛌
☑ Optimize boolean operations	
Build binary decision diagrams (BDDs)	<u>C</u> ancel

注意编译器选项窗口的三个目标选项:

- SIMULATE: Workbench Simulator 在ISaGRAF的仿真器上运行(不要求硬件)。这是您在办公 室调试程序的好办法。
- ISA68M: Run-time for SIXNET gateway 在VersaTRAK或SIXTRAK可编程控制器中运行程序。
- ISA86M: Run-time for Windows (3.1, 95 or NT) 使用SIXNET Control Room 和 ISaRUN runtime 软件,在基于Windows 的 PC 上运行程序。

每个选项将按照其目标处理器编译程序。ISaGRAF代码生成器可在同一次编译时产生所有的三种版本 代码。

#### 然后 →

使用Select按钮选择所有三个选项: SIMULATE, ISA68M 和 ISA86M。

#### 然后 →

选择所有的优化选项。

#### 选择 **OK**.

会通知您下次编译时所有程序将被确 认。点击 **OK。** 

现在将可以在所有三种目标中运行您的应用。

de ge	neration parameters
arget	s:
SIM	ULATE: Workbench Simulator
ISA	68M: Run-time for SIXNET gateway
ISA	36M: Run-time for Windows (3.1, 95 or NT)
MU	se embedded SFL engine
Dptin	izer:
E B	un two ontimizer passes
	uni uno optimizzo passes
	valuate constant expressions
le ∎ La n	appress unused labels
MU	primize variable copying
M	ptimize expressions
IN S	uppress unused code
<b>I</b> 0	ptimize arithmetic operations
₩0	ptimize boolean operations
E B	uild hinaru decision diagrame (RDDs)

#### 然后 →

#### 选择 Make, Make Application.

ISaGRAF代码生成器将确认程序,无误后编译之。

#### 然后 →

选择 Exit.

#### 然后 →

关闭training程序窗口。

	-:-ISaGRAF - TRAINING - Programs
	<u>File Make Iools Debug Options H</u> elp
	B Make application MIA III W K III & IIII  & IIIII & IIIII & IIIII & IIIIII
	Beg Touch
	End: Application run time Options
	Compiler options
	History of modifications
	<u>R</u> esources
🗧 ISa	GRAF - TRAINING - Code Generator 📃 🔲 🗙
<u>F</u> ile <u>O</u>	<u>)</u> ptions <u>E</u> dit <u>H</u> elp
Verify	ing variables declarations
	5
Reset	ting temporary files
	<b>.</b> . ,
Buildi	ing configuration data
Linkin	ng object files
Reloc	ating object files
Post-r	compiling code
Noer	ror detected
Noer	ror detected
	Code Generator
	No error detected.
	Do you want to exit the Code Generator now ?
	Eui) Continue
	Exit <u>C</u> ontinue

#### 使用密码保护TRAINING项目

ISaGRAF为保护应用提供了16个保护级别。密码级别从0至15,0是最高级。

从ISaGRAF项目管理窗口:

选择Training程序(选中但不打开).

然后 →

📨 ISaGRAF - Project Management \_ 🗆 × File Edit Project Options Help È 🖽 Project descriptor 10 贝 틒 History 💷 a\_sa ٠ Print 🕫 train 🎟 Heat Import <u>IL</u> program 💷 rfarr: <u>S</u>et password 🏢 rfbars rfhaal

选择 Project, Set Password.

Data p	protection - 'training'			×
0	SuperUser	8	Admin	<u>0</u> K
1	Executive	9		Cancel
2	PlantManagement	10		
3	Level_One_Engineer	11		
4	Level_Two_Engineer	12		
5	Level_Three_Engineer	13	Accounts	
6	Level_Four_Engineer	14	Shipping	
7	Level_Five_Engineer	15	FrontOffice	<u>F</u> ree
Crea	ite new program			Level 7 🔺
Mov	e program in hierarchy			Level 1
Verif	iy program			Level 6 💻
Mak	e application code	Level 1		
Touc	ch/compact (delete gener	ated co	de)	Level 0
Conv	version tables			Level 7 💌

在相应密码级别号码的右边文本框,按您的要求写入密码。

窗口下半部分是项目的元素列表 (如: 创建新应用,编译应用,移动程序等)。选择各元素,然后点击 要求的级别,可分配访问级别。

选择元素,然后点击 FREE 按钮,可去掉保护密码。退出ISaGRAF,然后重新打开项目管理器,密码 才可生效时。

注: TRAINING应用不需要密码保护。

#### 然后 →

选择Cancel,退出密码窗口。

## 使用 I/O 仿真器

关于ISaGRAF 仿真器的详细信息,

参考 ISaGRAF 在线帮助或用户手册

1/5

#### 本章我们将要:

- ISaGRAF仿真器概述
- 运行I/O仿真器调试RunMotor程序
- 使用仿真器调试Safety程序

#### 本章开始前,您应当:

已完成了本手册的使用梯形图/功能块图编辑器章节。

#### I/O仿真器:

ISaGRAF I/O仿真器是一个完整的ISaGRAF目标系统,支持ISaGRAF标准特性和CJ International和 SIXNET发布的的所有标准库函数和功能块。仿真器支持和ISaGRAF调试器的完全通信,所以仿真时 具有所有的调试功能。*如果在上次编译后做过修改,则运行仿真器前应重新编译。* 

2

@ a_s	ample			
<u>File</u>	<u>)</u> ptions <u>H</u> elp		_	
	0	1	2	3
750	Press_Temp	81 Tank_Level_Out	1 Start_Cycle	1 Press_Valve
30	pH	Temp_AO2		2 Main_Power
1200	ORP	Pressure_AO3	Up_Limit	O3 Heater_Power
150	Zone_1_Temp	Flow_Rate_AO4	Safety_Gate	O4 Low_Temp_Alarm
5	Conveyor_Speed	* * * *	Manual_Mode	95 High_Temp_Alarm
30	Flow_Rate		Fill_Tank	<b>6</b> DO6
80	Tank_Level_1		Drain_Tank	07 DO7
75	J_Thermocouple		<b>1 8</b> DI8	<b>●8</b> DO8 -
		-	-	• //

### 开始:

打开Training项目。

然后 →

从ISaGRAF Training 程序窗口:

#### 选择 Debug, Simulate.

当仿真器启动时,调试器, Training 程序,及仿真器窗口以调试方式打 开。

- 🗄 ISaGRAF - TRAINING - Programs 📃 🗖 🗵								
<u>F</u> ile <u>M</u> ak	e <u>T</u> ools	De <u>b</u> ug	<u>O</u> ptions	<u>H</u> elp				
	🕹 111	<u>D</u> ebi	ug	1 X Iv		រៃ 🖾	¥₫	\$ <u>⊡</u> \$
Begin:	(	<u>S</u> imu	ilate	1				
End:	<u>(</u>	Link	setup -• <b>y</b>	]				

调试器窗口 →	Q_ISaGRAF - TRAINING   File Control   File Control   ▶ N   ▶ N   ▶ N   ▶ N	G-Debugger 18 <u>H</u> elp 23	_	
调试方式下的程序窗口 →	ISaGRAF - TRAINING     File   Iools   Options   Help     Im   Im   Iools   Iools   Iools     Begin:   Imit So   Iools   Iools   Iools     End:   Iools   Iools   Iools   Iools   Iools	G - Debug programs G - X4 Calle unMotor afety		
ISaGRAF I/O 仿真器 →	Begin: Scale (Function Blo training File Options Help 0 Alo 0 Motor Temp	1 AO0 Heat Valve	2 Di0	3 D00 1 Motor
窗口与右边所示不同,选择 确保之选中:Color Variable names, Always			Stop 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	2 High_Alarm

然后 →

如果您的 Options, display, on top.



@ training			
<u>File Options H</u> elp			
0 Al0	1 A00	2 DI0	3 D00
0 Motor_Temp 0 Core_Temp	Heat_Valve	Start Stop	01 Motor High_Alarm
		Safety_Gate	<u>●3</u> ●4
•			▶ //

显示出Training项目的各个I/O点

- Analog Inputs: (最左边所示)点击输入点数值框,输入原始数值。也可使用[上下箭头]按钮增加 或减少数值。
- Analog Outputs: (本例没有)输出值以红色数字表示。
- Discrete Inputs: (右边第二栏) 鼠标左键就象一个触发开关 按下为真(亮), 弹起为假(灭)。鼠标右 键就象一个按钮 按钮按下时输入为真。

Discrete Outputs: (最右边栏) 由红色LED表示的输出。输出为真时LED亮。

#### 使用仿真器测试应用:

#### 开始**:**

左击 **Safety\_Gate** LED, 使能马达操 作。

#### 然后 →

右击 Start 按钮启动马达。注意现在 马达LED为亮(真)。

#### 然后 →

右击 Stop 按钮。注意现在马达LED 为灭(假)。

#### 然后 →

重新右击 Start 按钮。

#### 然后 →

双击 RunMotor 程序。

双击 Scale 程序。

双击 Safety 程序。

la training				х
<u>File Options H</u> elp				
0 AlO	1 A00	2 DI0	3 D00	<b>^</b>
0 Motor_Temp	Heat_Valve	Start	Motor	
		3 Safety_Gate		
	0		<b>0</b> 4	•
i training File Options Help				×
	1	2 DI0	3 D00	-
0 Motor_Temp	Heat_Valve	Start	1 Motor	
0 Core_Temp	0	Safety Cate	High_Alarm	
				-
1			•	//
-S-ISaGRAF - TRAINING	i - Debug programs		_ [ ]	×
File Loois Uptions Help	- 184			
Begin: Path Sc	⇒ x∾   ale			
Ru Bu	inMotor			
End: End:	fety			
End: Safety (Function Bloo	ok Diagram)			

#### 然后 →

将这些窗口改变大小和位置以便可以同时观察到。

记住,本应用的High\_Alarm函数:

如果 scale\_temp >= 200, 则 High\_Alarm = True 且 Motor = False.

ISaGRAF仿真器不显示内部变量,所以我们只能使用原始温度变量 Motor\_Temp测试。原始温度高限是 21845.

然后 →

点击 Motor\_Temp 区。



#### 然后 →

写入 21840. 使用 🚔 键将

Motor\_Temp值增加至21845.

注意当温度值到达 **21845** 时,变量 **High\_Alarm** LED变亮,而 **Motor** LED 变灭。

#### 然后 →

使用 🙀 键将Motor\_Temp值减少到 21845以下。则High\_Alarm LED变 灭。

#### 然后 →

右击Start按钮,重新启动马达。

training			
<u>File Options H</u> elp			
0 Al0	1 AO0	2 DI0	3 🔺
21845 Motor_Temp	0 Heat_Valve	<b>1</b> Start	Motor 2 High_Alarm
	<b></b>	Safety_Gate	



#### 然后 →

在仿真器上设置 Safety\_Gate 为 False, 然后右击 Start.

注意 Motor 停止运行。

这表明目标循环(target cycle)的重要性。尽管在开始(Beginning)程序中具备启动马达的所有条件,但 Safety程序(End)可以在ISaGRAF 写入输出前捕获不安全条件。

#### 然后 →

设置 Safety\_Gate 为 True,马达又可以运行了。



#### 然后 →

从仿真器窗口选择 File, Exit, 退出 仿真方式。

## 在SIXNET Windows Run-time方式运行应用

关于 ISaRUN -- SIXNET Windows

Run-time -- 的更多信息,请参考

SIXTRAK Control Room 软件的在线

1/9

#### 本章我们将要:

- 使用SIXNET Windows Run-time运行您的应用
- 设置 PC/PLC 链接选项
- 使用VersaTRAK I/O仿真器仿真 I/O

#### 本章开始前,您应当:

已经完成了本手册的**编译应用**和**使用仿真器**章节,您的系统必须已经安装SIXNET Control Room软件和SIXNET ISaGRAF Run-time软件。

2

帮助

本例不需要 SIXTRAK硬件, VersaTRAK I/O 仿真器软件将仿真实际 I/O。

#### IOMAP 和 SIXNET WINDOWS RUN-TIME (ISARUN):

由SIXNET ISaGRAF支持的SIXNET ISaRUN run-time软件是在计算机存储器中的ISaGRAF项目和 SIXTRAK I/O数据库或IOmap间的接口。

SIXTRAK项目所建立的Iomap是数据库管理器,提供Windows应用和实际I/O的接口。SIXTRAK Iomaps由SIXNET Control

Room Iomap工具创建。

lomap主窗口包含一个用户组态 表格,可创建与远程站交换的 I/O类型和数量。lomap使用这 个信息在DLL (动态连接库)设置 数据数组合查询远程站的扫描任 务。ISaGRAF和其他 Windows 程序可通过DLL访问Control Room数据库中的I/O。

关于Iomap和ISaRUN的更多信息 请参考SIXNET在线帮助。

			I/O Type Tag Name	St	tation Tag Name	Register Count	Update Action		
1	Di	scret	e In	SIXTRA	< Demo	8	Read Scan		
2	Di	scret	e Out	SIXTRA	< Demo	8	Write Scan		
3	An	alog	In	SIXTRAK	< Demo	16	Read Scan		
4	An	alog	Out	SIXTRA	< Demo	8	Write Scan		
5	1	Stati	on Definitions						X
6	$\underline{W}$	(indow	<u>E</u> dit <u>D</u> efine <u>H</u> elp						
7			C4-4 T-		Davidara	Casting	Cardina	Chatria	
8			Name	19	Device	Number	Туре	Flag	
<u> </u>		1	Local Computer		None		IOmap		
11		2	SIXTRAK Demo		Default	0	SIXTRAK		
12		3							
For		4							
		5							
		6							
		7							
		8							•
	Default Device: Ethernet 🗾 <u>C</u> lose <u>H</u> elp								
	Allow Duplicate Station Numbers   Auto Load     Auto Load   Sort								
		Each the sl	station requires a u ation's name in othe	inique tag er program	name of up to 20 o s. Select Auto Loa	characters. Id on the De	This tag name m efine menu to loa	nust match ad station	▲ ▼

#### 开始**:**

双击 图标,打开 Control Room IOmap 软件。

### 然后 →

选择 File, New, IOmap.

然后 →

选择 TRAINING.6pj 项目文件。

点击 **OK**.

然后 →

将lomap命名为 **Training** , 点击 **OK**.

**注意:**在设备窗口,缺省选中**DDE**服 务器。

#### 然后 →

选择 Yes,以便在TRAINING.6pj项 目使用站定义和I/O寄存器。

Control Room自动执行这个步骤。

使用在线帮助查看选项说明。

#### 然后 →

选中 Run an ISaGRAF program, 然 后点击 ISaGRAF Setup.

#### 然后 →

#### 选择 Run ISaGRAF.

在ISaRUN runtime方式运行应用由三种方法:第一种是启动ISaRUN,然后启动ISaGRAF应用;第二种是直接从I/Omap运行已经编译的应用.

#### 然后 →

从Workbench选择 Start Project.

选择 **OK.** 

File Open	x
Set IOmap Name	<u>o</u> k
Enter 10map Name: <u>O</u> K	<u><u>C</u>ancel</u>
Litaining <u>C</u> ancel	<u>H</u> elp
Select an existing IOmap name or assign a new Aname to this IOmap Definition to identify it in the	N <u>e</u> twork
trn1296.6pi	
List Files of <u>Type:</u> Drives:	
Project Files (*.6pj)	-
IOmaps in Project:	

Select Devices Window	×					
✓ Include DDE Server	DDE Setup					
₩ <u>B</u> un an ISaGRAF Program	ISaGRAF Setup					
De <u>f</u> ault COM Device:	NULL					
Run 10map <u>N</u> ow						
<u>V</u> iew IOmap Se	etup					
	<u>C</u> ancel <u>H</u> elp					

#### 在 SIXNET Windows Run-time方式运行应用

3/9

### 然后 →

选择 Null作为缺省通信设备 (Default COM Device)。

这样,就可以没有外部 I/O,而在计 算机上运行了。

然后 →

#### 选择 Run IOmap Now.

将会提示您保存组态:选择 OK.

ISaGRAF File Options	×
Project:	<u>O</u> K <u>C</u> ancel <u>H</u> elp <u>B</u> rowse
C:\ISAWIN\APL	

Select Devices Window	×
✓ Include DDE Server	DDE Setup
₩ <u>R</u> un an ISaGRAF Program	ISaGRAF Setup
De <u>f</u> ault COM Device:	NULL
Run 10map <u>N</u>	ow
<u>⊻</u> iew 10map Se	etup
5년 교험 ISaRun	
About	
	5 <sub>IEC</sub> 1131
ISaGRAF Project: No Application	
Active IOmap: No Active IOma	p
Run-Time Status: No Warnings	
Stop Runtim	le
🛛 🏭 sca 🛛 🗱 Six	閉路 ISa

ISaRUN runtime将自动启动并最小化,并通知你组态已改变。

然后 →

选择 **OK.** 

然后 →

当站定义窗口出现时,选择 Close.

选择 OK, 然后关闭 IOmap.

现在,由三个任务以最小化方式运行:

Control Room Scan Task, DDE Server, 和 ISaRun Windows Runtime.

### 在 SIXNET Windows Run-time方式运行应用

4/9

<b>然后 →</b> 从TRAINING程序管理窗口,选择 Debug, Link Setup.	SaGRAF - TRAINING - P     File   Make   Icols   Debug   D     Ein   Imil 谷 (D):   Debug   D     Begin:   Imil Imil Imil Imil Imil Imil Imil Imil	rograms □ X ptions Help e tup
然后 →	PC-PLC link parameters	×
在通信口旁的下拉条选择 SIXNET.	Target Slave Number:	1 <u>0</u> K
然后 →	Communication port:	SIXNET Cancel
点击 Setup.	Control	<u>S</u> etup
	l ime out (seconds):	
	neules.	
· 伏后 →	Communication S	etup 🗙
选择 Windows Run-time.	Target System	
点击 <b>OK</b> .		Run-time Cancel
点击 <b>OK</b> .	O Sixtrak <u>G</u>	ateway <u>H</u> elp
	Communicatio	m to Station-
	© <u>S</u> ingle Ga	ateway Mode
	C <u>N</u> etwork	Mode
	Device:	COM1
	S <u>t</u> ation Name:	ET_Gateway
	Select Sixn	et Project <u>Fi</u> le
然后 →	🔍 ISaGRAF - TRAINING -	Debugger
从主菜单选择 <b>DeBug</b> , <b>DeBug</b> .	File Control Spy Options	Help A Marine 1991
	No application	
你将看到调试器窗口出现,但没有应用。		

### 然后 →

SIXNET北京代表处

选择 File, Download 并选择 Download × **ISA86M Windows Runtime.** ISA68M: Run-time for SIXNET gateway ISA86M: Run-time for Windows (3.1, 95 or NT) 然后 → Application symbols 点击 Download. 观察下载过程,并注意状态条变为 Run. **Download** Cancel - ISaGRAF - TRAINING - Debug programs  $\rightarrow$ - 🗆 × <u>File Tools Options Help</u> 调试器也将程序管理窗口以调试方式 B m ⊗ m v ⊑ ¥ 打开。本手册也学习调试器的使用。 🕮 Scale Begin: RunMotor End: 🖭 Safety Begin: Scale (Function Block Diagram)

#### 用VERSATRAK I/O将仿真I/O链接到 IOMAP

现在应用在SIXNET ISaRun Windows run-time下运行。可以象在I/O仿真器一样观察应用状态。

在程序管理-调试方式窗口双击图标 打开 **RunMotor** 程序。

#### 然后 →

在程序管理-调试方式窗口双击图标 打开 Scale 程序。

🔍 ISaGRA	F - TRAINING	- Debugger			
<u>File</u> <u>C</u> ontro	I Sp <u>y O</u> ptions	<u>H</u> elp			
@))) ≈	• • • •	🔅 👫 🖾			
RUN	allowed=0	curre	nt=0 r	naximum=1	overflow=0
-⊜ISaGRA	F - TRAINING	- Debug progr	ams		
<u>File T</u> ools	Options <u>H</u> elp				
	8 💷 Vo 🗠	\$ ¥			
Begin:	🖭 Sc	ale			
	HIN Ru	nMotor			
End:	🖭 Sa	fety			
Begin: Sca	le (Function Bloc	ck Diagram)			

VersaTRAK I/O Simul... 💌

6/9

VersaTRAK I/O仿真器是一个调试应 用时仿真实际I/O的工具。



**∏** Y3

选择: Motor\_Temp, Start, Stop, Safety\_Gate, Motor, High\_Alarm,

然后点击 Apply.

Apply

<u>C</u>ancel

<u>H</u>elp

### 在 SIXNET Windows Run-time方式运行应用

7/9

#### 然后 →

将RunMotor和Scale程序窗口和仿真 器窗口大小和位置调整,以便可同时 看到。

#### 然后 →

左击 **Safety\_Gate** LED, 使能马达操 作。

#### 然后 →

右击 Start按钮启动马达。

注意:现在,仿真其中**Motor LED** 变亮。

在RunMotor程序中观察 Start接点变为ON (由蓝色变为红色).

#### 然后 →

在仿真器中,向右拉**Motor\_temp** 滑 块。

在Scale程序观察相应数值的变化。

注意:当 scale\_temp 值达到200度 以上时, High\_Alarm 变为 ON 而 Motor 变为 OFF.

然后 →

继续测试您的应用。





#### 然后 →

完成后,关闭 SIXNET I/O 仿真器。

#### 然后 →

关闭调试器窗口。

#### 进行程序修改

ISaGRAF应用将继续在ISaRUN run-time下运行。您可以关闭ISaGRAF调试器、Control Room Iomap 然后继续编辑ISaGRAF程序。如果改变了程序,应当重新编译并向SIXNET Windows Runtime下载新的版本。

本例假定改变温度由200到250时, High\_Alarm 变为 ON。

#### 开始**:**

双击RunMotor 程序图标打开 RunMotor程序。

#### 然后 →

双击整型输入200,将其改为250.

然后 →

Save, Verify 该程序。

#### 然后 →

退出 RunMotor程序。



从程序管理窗口选择 Make, Make Application.

#### 然后 →

如果编译是没有错误,退出代码生成器。

Exit

Continue

#### 在 SIXNET Windows Run-time方式运行应用

9/9

#### Warning: different applications X 然后 → 从程序管理窗口,选择 DeBug, Workbench application: OK Version: 5 DeBug. Date: 1/12/97 22:45:15 你将得到提示信息:目标应用与当前 CRC (symbols): 250407222 编译的应用版本不同。 Target application: 然后 → Version: 4 1/12/97 22:15:14 Date: 点击 OK. CRC (symbols): 250407222 🔍 ISaGRAF - TRAINING - Debugger \_ 🗆 × NEXT $\rightarrow$ <u>File Control Spy Options H</u>elp Stop application 0 🖱 🖾 从调试器窗口,选择 File, Update Start application current=0 num=407 overflow=f Download Application. Update application Bealize undate Get version number E<u>x</u>it 然后→ Download updated application X 选择选项 ISA86M: Run-time for ISA68M: Run-time for SIXNET gateway ISA86M: Run-time for Windows (3.1, 95 or NT) Windows (3.1, 95, NT). Application symbols 然后 → 选择 NOW. Update 然后 → O Later • Now 选择 Download. 您将看到下载新的 应用。 <u>D</u>ownload <u>C</u>ancel 按以前的说明打开SIXNET I/O仿真器 并确认程序的改变。.

然后 →

完成后,关闭调试器。

关于调试应用的详细信息,请参考使用调试器章。

详细信息请参考 SIXNET

ISaGRAF 帮助文件

1/6

#### 本章我们将要:

- 使用一个SIXNET可编程控制器运行应用
- 使用VersaTRAK I/O仿真器测试应用
- 设置PC/PLC链接选项

#### 开始本章前,您应当:

已经完成了ISaGRAF Training项目的创建、编译及使用仿真器章节。

完成本章您需要一个SIXNET或 VersaTRAK可编程控制器,不需要其他I/O硬件。如果您想使用ST-Demo演示单元,只需改变将Virtual I/O改为实际 I/O,并向控制器链接和下载组态。

本例使用以太网控制器(ST-GT-ETH-24P),通过计算机COM1口连接到控制器的Plant Floor (RS232)口。如果您的硬件设置不同,只需在Plant Floor项目中做很小的修改即可。下列步骤反映了适当的控制器。

#### 开始**:**

从 TRAINING程序管理窗口选择 Debug, Link Setup.

- 🗄 ISaGRAF - TR/	AINING - Program	\$		_ 🗆 ×
<u>F</u> ile <u>M</u> ake <u>T</u> ools	De <u>bug</u> Dptions	<u>H</u> elp		
6 6 8 10	<u>D</u> ebug	‱ m)	ρ⊑ ¥	ર 🛄 ટૈ
Begin: (	<u>S</u> imulate			
End'	Link setup			
	-a Jaicty N			
L <u> </u>				

#### 然后 →

在通信口选项的下拉框,选择 SIXNET.

这个通信口应当总是SIXNET – 不管 您是使用 ISaRUN run-time 还是可编 程控制器。

PC-PLC link parameters		×
Target Slave Number:		<u>D</u> K
communication port.		<u>L</u> ancei
Control		<u>S</u> etup
Time out (seconds):	2	
Retries:	1	

#### 在SIXTRAK可编程控制器中运行应用

<u>o</u>k

<u>C</u>ancel

X

**Communication Setup** 

Target System

œ

Windows Run-time

Sixtrak <u>G</u>ateway

2/6

#### 然后 →

点击 Setup. 在通信设置的目标系统 中选取 SIXTRAK Programmable Gateway.

然后 →

点击 **OK**.

点击 **OK.** 

#### 然后 →

选择 Debug, Debug.

选择 File, Download,并选择 ISA68M: Run-time for SIXNET gateway. 将会进行下载,下载进程 显示在屏幕上方的状态栏上。

现在,您的应用已经下载到 SIXNET 可编程控制中了。

您可使用 ISaGRAF 调试工具观察应用的 运行。参见使用调试器章。

#### 12 <u>H</u>elp Communication to Station Single Gateway Mode Network Mode Passthru Mode COM1 $\mathbf{T}$ Device: Station Name: ET\_Gateway • Select Sixnet Project File Q ISaGRAF - TRAINING - Debug - I × <u>File Control Spy Options Help</u> ⊕ )) >> ) >> ) >> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) <>> ) < RUN allowed=0 current=0 overflow=0 num=1 - ISaGRAF - TRAINING - Debug programs \_ 🗆 × <u>File Tools Options Help</u> B m ⊗ m w ⊑ ¾ 🖭 Scale Begin: া RunMotor End: 🖭 Safety Begin: Scale (Function Block Diagram) Download X ISA68M: Run-time for SIXNET gateway ISA86M: Run-time for Windows (3.1, 95 or NT) Application symbols <u>Cancel</u> <u>D</u>ownload - I × Q ISaGBAF - TRAINING <u>File Control Spy Options Help</u> RUN allowed=0 overflow=0 current=0 -::ISaGRAF - TRAINING - Debug program \_ 🗆 X File Tools Options Help Begin: 🖭 Scale 🗝 RunMotor B Safety End:

#### 然后 →

在调试方式下,从程序管理器双击程 序图标打开RunMotor程序。

#### 然后 →

在调试方式下,从程序管理器双击程 序图标打开**Scale**程序。

х

-

•

VersaTRAK I/O 仿真器工具允许您即 📡 VersaTRAK 170 Simul... 💌 S. 12 使在没有实际I/O情况下,将其发送 到SIXNET 控制器。 - | <u>Þ</u>l Motor\_Temp 然后→ P Start 双击 🕤 打开 VersaTRAK I/O 仿真 P Simvtrak Stop 器。 P Safety\_Gate 然后 → Motor 选择 📳 (总在最前). High\_Alarm 🛐 (设置)图标。 点击 设置屏幕的右栏是仿真器使用项目选 VersaTRAK Simulator Setup Active I/O 项。 Directory: C:\SIXTRAK AX0 Motor\_Temp Project File: TRAINING.6PJ □ AX1 Core\_Temp 然后→ □ AX2 Training Station: 选择 Training.6pj 项目文件。 ∏ AX3 Target I/O Location ▼X0 Start • VersaTBAK 选择 Training 站。 C Control Room 10map 🖂 X1 Stop 设置目标 I/O 位置为 VersaTRAK. ▼X2 Safety\_Gate **Computer Com Port: □** X3 Com Port: - I 选择适当的计算机通信口(如COM CEthernet ∏X4 Control Room 1)。 ∏ X5 VersaTRAK Com Port ∏ X6 设置 VersaTRAK 作为主Plant Floor • Main Serial Port C Plant Floor Port E X7 通信口。 C User Port 1 V0 Motor C User Port 2 设置屏幕的左栏是该站所有可用I/O C Ethernet V1 High\_Alarm 点列表。 TY2 Low\_Alarm **□** Y3 Apply <u>C</u>ancel <u>H</u>elp

#### 然后 →

选择: Motor\_Temp, Start, Stop, Safety\_Gate, Motor, High\_Alarm,

再选择 Apply.

#### 在SIXTRAK可编程控制器中运行应用

4/6

#### 然后 →

调整 RunMotor、Scale程序窗口和仿 真器窗口的尺寸和位置,以便可同时 观察到。

#### 然后 →

在仿真器窗口左击**Safety\_Gate**,使能马达操作。

#### 然后 →

在仿真器窗口右击 **Start** , 启动马达。

注意:此时仿真器中 **Motor LED** 变 亮 (**True**)。

如果您有一个ST-Demo演示单元,则 ST-DO-DCI-08 的第一个指示灯变 亮。

#### 然后 →

在仿真器中,向右拖动 **Motor\_temp** 滑条。

在Scale程序窗口观察scale\_temp 值,当它升到 200 度以上时:

High\_Alarm 指示灯变亮。

如果您有ST-Demo演示单元,则**ST-**DO-DCI-08 的第二个指示灯变亮。

马达状态灯变灭。

#### 然后 →

继续测试你的应用。



完成后,关闭 SIXNET I/O 仿真器。

#### 然后 →

关闭调试器窗口。





#### 进行程序修改

即使您关闭了ISaGRAF调试器并关闭了计算机,ISaGRAF应用仍然继续在控制器中运行。

如果您继续编辑Training项目,并想实现这个改变,您需要重新编译,并将新版本下载到SIXNET控制器中。在TRAINING程序管理窗口进行这些步骤。

#### 开始**:**

双击RunMotor图标,打开RunMotor jॠ ISaGRAF - TRAINING:RUNMOTOR - Quick LD Program \_ 🗆 X 程序。 <u>File Edit Insert Options H</u>elp 然后 → Simple command to turn on the motor. \*) Stop High\_Alarm Start Motor H -1 \ F < >双击整型输入 200, 将其改为 250. Motor 4 1 然后→ Monitor temperature, sound alarm, turn off motor \*) >= 保存、确认程序。 en scale temp in 1 然后 → 退出 RunMotor程序。 Select variable 🗖 🗠 👌 🚾 Integer/Real Scope:  $\mathbf{T}$ (Global) Ŧ 250 Core\_Temp Al1:Temperature at core of tank Heat\_Valve A00:Heat release valve Motor\_Temp AI0:Motor temperature scale\_temp Scaled value of Motor\_Temp - U X ISaGRAF - TRAINING - Code Generator 然后 → <u>File Options Edit H</u>elp Verifying variables declarations 从程序管理窗口选择 Make, Make Application. Verifying runmotor No error detected 然后退出代码生成器。 Code Generator X No error detected. Do you want to exit the Code Generator now ? Exit <u>C</u>ontinue

#### 在SIXTRAK可编程控制器中运行应用

6/6

#### 然后 →

从程序管理窗口选择 **DeBug**, **DeBug**.

您将得到提示信息,说明目标中运行 的应用与刚编译的应用版本不同。

然后 →

点击 **OK**.

#### 然后 →

从调试器窗口选择 File, Update Application.

然后 →

选择选项 ISA68M: Run-time for SIXNET gateway.

然后 →

选择立即下载 - NOW.

#### 然后 →

选择 Download. 您将看到下载过程。

#### 然后 →

使用VersaTRAK I/O仿真器测试改变 后的程序。

完成后,关闭调试器和仿真器。

关于调试应用,请参考使用调试器章节。

blications	x
on:	<u> </u>
5	
1/12/97 22:45:15	.0
250407222	
4	
1/12/97 22:15:14	
250407222	
	on: 5 1/12/97 22:45:15 250407222 4 1/12/97 22:15:14 250407222

🔍 ISaGRAF - TRAININ	G - Debugger			_ 🗆 ×
File Control Spy Option	ns <u>H</u> elp			
Stop application	) () <b>m</b>			
<u>D</u> ownload	current=0	ma×imum=407	overflow=0	
Update application	Į			
<u>R</u> ealize update	ľ			
Get version number				
Exit				

Download u ISA68M: F ISA86M: F Application	pdated application Run-time for SIXNET Run-time for Window n symbols	gateway s (3.1, 95 or NT)	×
Update -	O <u>L</u> ater	© Now	
	<u>D</u> ownload	<u>C</u> ancel	



#### 本章开始前,您应当:

已经完成了ISaGRAF TRAINING项目程序的创建和编译,以及本手册撌褂梅抡嫫鲾章。

#### 使用图形调试器:

ISaGRAF调试器允许您观察在三种目标系统的控制应用: ISaRUN Windows run-time, SIXTRAK 控制器,或 ISaGRAF 仿真器。 使用这个强有力的工具,您可以节省大量的开发时间。调试方式下可以 查看模拟值,I/O状态变量,扫描时间等。

<b>Q</b> , 19	GaGRAF	- TR	AINING	- Debi	ıgger		
<u>F</u> ile	<u>C</u> ontrol	Spy	Options	<u>H</u> elp			
	)) (©	•	H DF	Ô	#1 🖾		
RUI	s I	illow	red=0		curre	nt=0 maximum=1	overflow=0

启动时,调试器(debugger)自动以 调试方式打开程序管理窗口。调试方 式下,可打开ISaGRAF应用的任何程 序,但是所有的编辑命令被禁止。

调试方式下,程序元素(步,变迁, 变量...)以图形或数值显示当前值, 以便容易地跟踪应用的执行。

🔍 ISaGRAF -	TRAINING - Del	ougger		
<u>File</u> <u>C</u> ontrol	Sp <u>y O</u> ptions <u>H</u> elj			
@)))  ≫	> N >> 😚	#1. 🖾		
RUN a	llowed=0	current=0	maximum=2	overflow=0
	•CISaGRAF ·	TRAINING - Debug	programs	
	<u>File T</u> ools <u>O</u>	ptions <u>H</u> elp		
	🖹 🖬 😔	`ko t≅, ≱v		
	Begin:	🖭 Scale		
	=	RunMotor		
	Ena:	eren Satety		
	Begin: Scale	(Function Block Diagram	)	

#### 目标状态和循环定时

🔍 ISaGRAF	- TR/	AINING	- Deb	ugger					- 0	×
<u>File</u> <u>C</u> ontrol	Spy	<u>Options</u>	<u>H</u> elp							
@ ))( ≫	•	H	Ô	#1 🖾						
RUN a	illow	ed=0		curre	nt=0	maxim	um=110	overflow=0		

这是一个调试器主窗口和控制面板 (在调试菜单)。它显示目标应用的全局状态,和执行循环时间的信息。

#### 可能显示的目标状态:

Logging:	正在与目标系统建立通信
Disconnected:	不能与目标系统建立通信
No Application:	与目标系统的通信正常,但目标中没有ISaGRAF应用
Application active:	与目标系统的通信正常,目标中ISaGRAF应用已激活
Run:	目标应用以实时方式运行
Stop:	目标应用以循环到循环方式(Cycle to Cycle)运行
BreakPoint:	由于遇到断点,目标应用以循环到循环方式运行
Fatal Error:	由于发生严重错误,目标应用故障

#### 显示的循环时间信息:

Allowed:	程序扫描循环计时,0=以最大速率运行(连续)
Current:	上次完整执行循环计时
Maximum:	应用启动后检测到的最大循环
Overflow:	检测到的超过允许时间的执行循环数

所有时间以毫秒计,在仿真方式下不显示时间值。

#### 更多信息请参考ISaGRAF用户手册

#### 关于循环计时:

一般地,程序在目标中以实时方式运行。这意味着执行循环计时由程序条件触发。通过从调试器窗口选择Control, Cycle to Cycle,循环可由用户控制一个一个执行 (用Control, Execute one cycle 命令)。如果在调试时使用Cycle to Cycle方式,可以使用Control, Change cycle timing命令设置允许时间,以毫秒计。

#### 在线修改

ISaGRAF提供在线修改特性,允许用户在过程运行时修改应用。这在中断过程会危及生产或安全的场

SIXNET北京代表处

合是需要的。这个功能必须非常小心地使用。

在ISaGRAF程序管理窗口使用**Debug**菜单命令启动调试器。调试器启动时,先确认在目标系统的应用 是否与最新编译的项目版本一致。

如果您已经修改应用并发出 **Make** 命 令,**ISaGRAF**将通知您新的版本与目 标运行的应用不匹配。

ISaGRAF不自动下载新的版本,请在 调试器窗口使用Download 或 Update 命令下在新的版本。

<b>K</b> 7 10:43:51 297
7 10:43:51
7 10:43:51 <sup>1</sup> 0 297
297
201
7 09:48:43
297

在线修改时,可从调试器窗口选择 File, Update命令重新编译和下载。

必须选择目标系统,然后可选择立即 更新或以后更新。

系统在线时一些修改不能做,这时, 应当停止应用,然后从菜单选择 File, Download命令。

关于在线修改的详细信息,	请参考	
ISaGRAF用户手册		

Download ( ISA68M: I ISA86M: Applicatio	updated application Run-time for SIXNE Run-time for Windo In symbols	n ET gateway ows (3.1, 95 or NT)	×
- Update -	C <u>L</u> ater	© <u>N</u> ow	
	Download	<u>C</u> ancel	

#### 使用监视(SPY)菜单元素:

调试器的监视菜单提供两个便于使用的工具,以便调试时观察 I/O 数据的活动:列出变量(List Variables)和跟踪变量(Trace Variables)。

#### 开始前,应当:

在Windows runtime下运行Training应用,并按照**撛***赦***indows Runtime运行应用** *注*触疺ersaTRAK I/O 仿真器。



#### 使用在线调试器

5/9

#### 然后 →

在VersaTRAK仿真器上左击 Safety\_Gate使能马达操作。

#### 然后 →

右击Start启动马达。

注意:此时变量列表窗口中Motor为 ON.



×

<u>File Edit Options Help</u>

🗅 🖹 🖆 🐐 🗄 🛰

🕱 ISaGRAF - TRAINING:[untitled] -... 💶 🗖

Off

On

🗑 🗄 👪 🔶

•

Start

Stop

Motor\_Temp

9

9

۵

#### 然后 →

在仿真器中使用滑动条增加马达温度 (原始值) Motor\_Temp.

#### 然后 →

在变量列表窗口观察变换后的温度值 scale\_temp.

注意: 当scale\_temp达到200度时, High Alarm变为ON 而 Motor 变为 **OFF**。

#### 然后 →

在变量列表窗口左击 Motor 选择 之。

#### 然后 →

点击 按钮选择 Safety\_Gate.

#### 然后 →

使用VersaTRAK仿真器将 Safety\_Gate变为OFF.

然后 →

试着用仿真器启动马达。

由于Safety\_Gate是一个故障检测 器,您将不能启动马达。

#### 然后→

SIXNET北京代表处



VersaTRAK I	/0 Simul 🗙		
•	Þ		
Motor_Temp			
۵			
Start	😭 ISaGRAF - TRAINING:[u	intitled] - List of variables	
0	<u>File Edit Options H</u> elp		
Stop	🗅 🖹 🚔 🖷 🐭		
6	Safety_Gate	Off	
Safety Gate	Motor	Off	
ouroty_dute	High_Alarm	Off	
	scale_temp	152 Deg	
Motor	<end list="" of=""></end>		
High_Alarm			

在变量列表窗口双击 Safety\_Gate, 选择Lock.

这个特性用于在扫描周期锁住输入变量,变量锁住时,ISaGRAF将不写入变量。

### 然后 →

在变量列表窗口双击 Safety\_Gate, 选择ON.

**注意:** VersaTRAK仿真器中该值没有 变化 – 实际的Safety\_Gate寄存器读 数为**OFF**.

ISaGRAF现在覆盖实际的I/O寄存器,显示Safety\_Gate为ON.

#### 然后 →

在仿真器上右击 Start.

在变量列表和仿真器种, Motor 都为 ON.

#### 然后 →

双击 Safety\_Gate选择**Unlock**(解 锁).

观察已锁住的变量表或对所有变量解 锁,请从调试器窗口选择 Control, Unlock all IO Variables。

#### 然后 →

从变量列表窗口选择 File, Save 保存 变量表。

#### 然后 →

将变量表命名为 Motor, 选择 OK.

#### 然后 →

退出变量列表窗口.

Write bo	oolean variable		×
	variable S	afety_Gate	
<u>o</u> [	Off	On	1
	ock Uni	ock	ancel
Write bo	oolean variable		×





Save list as	×
MOTOR	ОК
MOTOR	
	<u>L</u> ancel

#### 继续前,应当:

Training应用仍然以Windows runtime方式运行,VersaTRAK I/O仿真器为打开状态。将Safety\_Gate 变为ON、将Motor变为OFF、将Motor\_Temp复位为0.



#### 使用在线调试器

#### 8/9

#### 然后 →

点击选中变量 scale\_temp. 使用设置量程按钮 **↓** ,选择量程 为 1-300.



#### 然后 →

使用 VersaTRAK 仿真器启动Motor.

注意:将Motor 变为ON 将立即在跟踪 图上反映为一个上升沿。

#### 然后 →

在仿真器重用滑动条增加马达温度值 Motor\_Temp.

**注意**: 当 scale\_temp 达到 200时, Motor 变为 OFF, 跟踪图立即显示 一个下降沿。

**可同上部分一样锁定输入 – 双击变量,** 选择Lock,然后重新双击变量,写入要 求值。

#### 然后 →

使用 File, Save 命令保存这组跟踪变 量,将其命名为Motor.

退出变量跟踪窗口。

#### 使用调试器的其他工具观察您的应用:

可使用变量字典(**Dictionary**)观察 运行应用的变量的当前状态。


## 使用在线调试器

9/9



在程序中双击变量名可锁住或写入变量

## 然后 →

退出调试器。

#### 本章我们将要:

- 创建一个ISaGRAF功能块
- 编辑这个功能块
- 在LD/FBD编辑器中的功能块菜单确认之

#### 开始本章前, 您应当:

已经完成了本手册梯形图和功能块图编辑器的使用章节和I/O仿真器的使用章节。

## 关于用户自定义功能块

ISaGRAF允许用户使用<u>ISaGRAF Libraries</u>中的工具创建自定义的功能块。可使用标准的ISaGRAF 快速梯形图、梯形图/功能块图、结构化文本、或指令表程序编辑器创建*用户自定义功能块*,不能使 用顺序功能表编辑器创建。

功能块的编译和任何ISaGRAF程序或函数一样,使用代码生成器。一个库函数或功能块可以有局部变 量和局部定义字。编译后,用户自定义功能块就可在LD/FBD图形编辑器中的功能块窗口被选择了。功 能块的归档应使用 Archive, Function Blocks 命令。

Parameters - 'ON_OFF'		×
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	IT D	<u>O</u> K <u>C</u> ancel
Name: DEADBAND Mode © Call © R <u>e</u> turn	C <u>B</u> oolean & A <u>n</u> alog C <u>B</u> eal C <u>T</u> imer C <u>M</u> essage	<u>I</u> nsert Delete <u>A</u> rrange

上图左上方显示块的参数,有调用参数和返回参数。函数和功能块可有多达**32**个参数(输入或输出)。 一个功能块总是有且只有一个返回参数,它必须与该功能块同名。

## 关于 ON\_OFF 功能块

我们将创建一个功能块,它接收输入变量 (PV) 并判断其值是否在一个范围内,以决定输出的真假。由参数 HIGHPOINT 决定范围的高限,而由 (HIGHPOINT - DEADBAND) 决定范围的低限。

# 如果 PV > HIGHPOINT 则 ON\_OFF = TRUE 如果 PV < (HIGHPOINT - DEADBAND) 则 ON\_OFF = FALSE 如果 HIGHPOINT > PV > (HIGHPOINT - DEADBAND) 则 ON\_OFF = 当前值

注意:如果您没有完成使用本手册的LD/FBD 编辑器章,请先完成。



#### 选择 Function Block Diagram 作为 编辑语言。

然后 →

## 选择 **OK**

将出现功能块参数设置窗口。

注意: 在块右边的返回参数

模式设置为 Return.

缺省情况下, ISaGRAF设置类型为模 拟, 由于 ON\_OFF 块要求返回一个 真/假值:

将类型设置为布尔型。

ON\_OFF块有三个输入变量(或调用 变量): PV, HIGHPOINT, 和 DEADBAND.

#### 调用变量名与返回变量名:

- 不超过16个字符
- 首字符必须为字母,其余可以是字
- 母、数字或下划线。
- 不区分大小写。

### 然后 →

点击 **Insert** 按钮三次,插入三个新的 调用参数。

Parameters - 'ON_OFF'		×
	LOFF	<u>O</u> K <u>C</u> ancel
Name: ON_OFF Mode Call © R <u>e</u> turn	Type	<u>I</u> nsert <u>D</u> elete <u>A</u> rrange

Parameters - 'ON_OFF'		×
ON_OFF 00 01 02 03 → <u>r</u> ON_01 04	FF	<u>O</u> K <u>C</u> ancel
Name: Tyr Mode C Ca <u>l</u> l C R <u>e</u> turn	pe C <u>B</u> oolean CA <u>n</u> alog C <u>R</u> eal C <u>T</u> imer C <u>M</u> essage	<u>Insert</u> <u>D</u> elete <u>A</u> rrange

<u>0</u>K

4 / 11

**注意**: 输入将以在表中出现的相同顺 序出现在功能块上。

### 然后 →

点击名字区,向第一个参数写入名字 **PV**。

PV是块的输入,将模式选择为 Call. 将类型选择为 Analog.

## 然后 →

点击输入01旁的小框,选择下一个参数。

## 然后 →

点击名字区,写入HIGHPOINT.

将模式选为 Call.

将类型选为 Analog.

Name: $PV$ $O_4$ $O_$	F C <u>B</u> oolean © A <u>n</u> alog C <u>R</u> eal C <u>T</u> imer C <u>M</u> essage	<u>I</u> nsert <u>D</u> elete <u>A</u> rrange
Parameters - 'ON_OFF' ON_OFF 00 ← 2 PV 01 ← 2 HIGHP 02 03 → ON_OF 04	OINT F	∑ <u>Ω</u> K <u>C</u> ancel
Name: HIGHPOINT	De C <u>B</u> oolean C <u>An</u> alog C <u>R</u> eal C <u>I</u> imer C <u>M</u> essage	<u>I</u> nsert Delete <u>A</u> rrange

Parameters - 'ON\_OFF'

ON\_OFF

00

加上最后一个调用参数:

### 然后 →

点击输入**02**旁的小框,选择下一个参数。

在名字区写入**DEADBAND.** 

将模式选为 Call.

将类型选为 Analog.

现在,ON\_OFF块的参数如右图所示。

Parameters - 'ON_OFF'		×
00 ← 2 PV 01 ← 2 HIGHPOIN 02 ← 2 DEADBAN 03 → 1 ON_OFF 04	IT D	OK Cancel
Name: DEADBAND Type - Mode © Call © Return	C <u>B</u> oolean ← A <u>n</u> alog C <u>R</u> eal C Timer C <u>M</u> essage	<u>I</u> nsert <u>D</u> elete <u>A</u> rrange

要求变量列表的最后一个是返回参数, 点击排列(Arrange)按钮可自动将返回 参数排列到最下面。

#### 然后 →

选择 **OK.** 

5 / 11

## 然后 →

从ISaGRAF Libraries库窗口,选择 Edit, Technical Note.

这里,您可以将该块的功能、调用参数、返回参数、作者和其他相关信息 整理为文档。

### 然后 →

如右图所示,写入相关信息,然后保 存并退出这个技术笔记编辑器。

#### 然后 →

在ISaGRAF Libraries窗口,双击 ON\_OFF 块程序,可打开熟悉的 LD/FBD编辑器。

确保功能块工具条显示出来,并确保 选择了选项菜单中的 verify Auto Input(检验自动输入)。

#### 首先,我们将设置:当PV大于 HIGHPOINT时,ON\_OFF 为真

然后 →

选择 按钮,在位置(4,3)放置一个输入变量。

### 然后 →

出现选择变量窗口时,使用下拉条选择 **ON\_OFF**.

点击模拟变量选择按钮,确保模拟变 量显示出来。

### 然后 →

选择PV,点击OK





6 / 11



大于块的右边放置置位线圈。



## 然后 →

选择变量窗口出现时,点击[Boolean 按钮],显示出布尔变量。

选择ON\_OFF,点击 OK.

然后 →

选择 与置位线圈之间画连接线。

现在,我们将设置:当PV 小于 HIGHPOINT-DEADBAND 时,设置 ON\_OFF 为假。

然后 →

<u>의</u> 使用 具条。

然后 →

选择 🛄 按钮,在位置(4,10)放置

输入变量 HIGHPOINT。

# 然后 →

选择变量窗口出现时,点击[Analog

按钮],显示出模拟变量。

## 选择 DEADBAND.

### 选择 **OK**

然后 →

从工具条选择 **正** 按钮,使用下 拉菜单选择 **SUB**(减法)功能块。 将该块放置到PV右侧,位置(14,8)

## 然后 →

选择	]+[	按钮,	在 <b>DEADBAND</b>
和减	去块的	第二个结	谕入间画连接线。

-Jin	€ I S	GaG	RA	F -	ON,	_0F	F-	FBI	D7L	DP	rog	ram										_	
E	ile <u>E</u> dit <u>O</u> ptions <u>H</u> elp																						
ŀ	ю	ĥ	: [					2 K	9	2	1÷	Lia(	÷	·		13	SUB	-	]				
F			•	÷	·	÷			•		·	·					SR stackint	-	•		•	•	
·			•	·	·	·			÷		·	·					stsample		· ·		•	•	·
Ŀ																>	SUB		· ·				
ŀ	Γ							٠P	V						141		sxndate k	2	0	$N_{0}$	DEE		•
Ŀ	Г			•	Н	IG⊦	ΗP	ЫN	ITI						142		sxnhold			-(3	÷Η-		
ŀ					•	•	•							·	•		sxnonoff sxntime		·				
Ŀ																	system		1 -				•
		•	•	·		•	•	•	•	•	•	·	·	÷	·	÷	tan Tmr	-		•			

Īœ	ISa	GR/	۹F -	ON	_0F	FF -	FBI	D7L	DP	rog	ram														_ [	IX
Eile	• <u>E</u>	dit	<u>O</u> pl	ions	H	elp																				
	12	) (	¥		12	è	*	Ē	5	] [	2		8	5												
ЧK	9 [	R	E				2	۶¢	2	년	]ðį	₽	۰		1	s	UB			•						
·	•	•	•	•		•		•	•	•			·	·	•	·		•	·	·	•	·	·	•	•	· 🔺
·		•	•					·	·	•			·	·		·	•	•	·		•	·			•	•
·								÷					·		, n	>			·			·				
·	•	÷	÷	÷			٠P	V						141					·		0	N_	OF	F		
·	•	•	•	Η	IGł	ΗP	ЛЮ	IT						142			0		 			-(3	Ð			
·	•							•						•												
·																										
·																										
·										·		-														
·											141															
·	•	•		DE	EAE	DB.	AN	D	_	-	142			0	<b>-</b> .											
·										. '																
·																										
																										. 🗖
pos	s=9,0	)																								

ISaGRAF 允许使用上面的输入变量 HIGHPOINT作为减法块的输入。

### 然后 →

在 HIGHPOINT 和减法块的第一个输入间画连接线。

ISaGRAF自动为新块创建一个分支。



9 / 11

## 然后 →

从工具条选择 **正** 按钮,使用 下拉菜单选择 less than (小于) 功能

块。 将该块放置到 PV 输入的右边,位置 (20,6)

#### 然后 →

选择按钮 , 在 **SUB** 块的输出与 < 块的第二个输入间画连接线。

#### 然后 →

在**PV**与 **<** 块的第一个输入间画连接 线。

我们将使用一个复位(reset)线圈 给ON\_OFF的返回参数分配一个**假** 值。

## 然后 →

使用 **H** 融发按钮, 弹出 LD 工 具条。

## 然后 →

选择 [reset] 工具, 点击位置(26,8), 在**小于**块的右侧放置一个复位线圈。





### 然后 →

选择变量窗口出现时,点击[Boolean 按钮],显示出布尔变量。

选择 ON\_OFF 并点击 OK.

10 / 11



## 然后 →

选择按钮 , 在小于块的输出和复位线圈之间画连接线。

## 然后 →

选择 File, Save

# 然后 →

### 选择 Options, Compiler Options

确保选中所有三种编译目标。

## 编译器选项在本手册其他章节有详细 的讨论。

现在,确保编译器选项如右图所示,选择 **OK** 

# 然后 →

### 选择 File, Verify.

ON\_OFF 块现在被编译,并可以在 您的 ISaGRAF 项目中使用了。

退出代码生成器。

# 然后 →

退出程序和 ISaGRAF Libraries工具。

11 / 11

在 FBD 编辑器中寻找 ON\_OFF 功能 块:

打开 Scale 程序

在 FBD 工具条选择下拉框,选中 ON\_OFF

双击该块,将其放置到程序屏幕任何 地方。

选择 Info 和 Notes ,可看到该块的 说明。

选择 OK 并不保存退出程序窗口。

避避ISaGRAF - TRAINING:SCALE - FBD/LD Program	
<u>File E</u> dit <u>O</u> ptions <u>H</u> elp	
₩ ╠▥▥▥ ◈♥♡의 └₲락⊕ ◧	ON_OFF _
Scales motor temperature to dea 1-300           Motor_Temp]           300]           32767]	Negate not_mask odd ON_OFF Operate  or_mask  pow r_trig rand Real replace