

Target Definition Builder

Software release 4.20

July 2003

Target Definition Builder

Printing History 1st printing — December 21, 2001
 2nd printing — May 31, 2002
 3rd printing — October 31, 2002
 4th printing — July 31, 2003

© Copyright 1999-2003: ICS Triplex ISaGRAF Inc

All rights reserved. No portion of this work may be reproduced in any form or by any means, without the prior written permission of ICS Triplex ISaGRAF Inc.

Table of Contents

Target Definition Builder	1
Declaring and Describing Objects	2
Target	4
Array	6
Structure	7
Function	8
Function Block	9
Simple Device	11
Complex Device	13
Conversion Function	15
Network Driver	16
OEM Parameters	18
Functions of FB Parameters	20
Pin ID	22
Defining On-line Information for Objects	23
Selecting Objects in a Set	26
Building an Exchange File	27
Generating "C" Source Code	29
Other Tools	30
Importing ISaGRAF 3.3 Libraries	31
Naming Conventions	32
Copyright	33

Target Definition Builder

The Target Definition Builder is a tool dedicated to integrators. It allows the integrator to build the description of an add-on for the ISaGRAF PRO target software. It can also generate a text description file that will be imported in the database of a Workbench template or project, and templates of "C" source code for each add-on.

You can describe targets (main definition and options of the embedded software), complex data types (such as defined in IEC languages), "C" functions, function blocks and conversion functions, I/O devices or network drivers for IXL communication and/or data binding.

Declaring and Describing Objects

Declaring objects

Objects are listed in the tree on the left of the document window. The tree shows a classification according to various object types:

- Target (main definitions).
- User defined data types:
 - Array
 - Structure
- Embedded POU (Program Organization Units) written in "C":
 - Function
 - Function Block
- I/O devices:
 - Simple Device
 - Complex Device
 - Conversion Function
 - Network Driver

To declare new objects, use the commands from the **Insert** menu. A default name is assigned to each new object.

To rename objects, click on their name or use the command from the **Edit** menu. When renaming an object, you can add a single line comment between (* and *) delimiters after its name. Naming of objects should fit to "C" or IEC language naming rules.

The **Insert / HSD Local Network** command inserts the description of the standard HSD driver, used for local binding between resources of the same configuration. This driver is based on the portable ISaGRAF PRO system layer, and is generally kept when the target software is ported to a new platform. The HSD driver then must be included in the full description of the new target.

Describing objects

The description of a selected object is displayed in the right area of the document window, in the **Details** panel. To change the description of the selected object, double click on its name in the tree.

The description of an object is entered through a multiple page dialog box. The information to be entered for an object depends on its type.

Target

A "Target" groups main definition and implementation options for the embedded ISaGRAF PRO software. Defining a target is required when porting the target on a new operating system. It is not required to have the full description of an existing target to define new components such as POUs or IO devices.

The following pieces of information must be entered for a target:

- Code Style and Model
- Data Model
- Virtual Addresses
- Implementation Options
- Extended Resource Parameters
- On Line Info

Code Style and Model

You must define here the style of code possibly generated by the Workbench for the target. It can be TIC (target independent code) or "C" source code (to be compiled and linked with the basic code of the virtual machine). You must select at least one code style for the target.

The Code Model can be "Medium" or "Large". In the "Medium" model, all TIC instructions and virtual addresses are stored on 16 bits. In the "Large" model, they are stored on 32 bits. The "Large" model allows applications with more than 64KB of data to run in the virtual machine. The definition of the code model must fit to options used when compiling the target software.

Data Model

Information entered in this pane should fit to the conventions of the target "C" compiler, processor and operating system. The compiler uses the definition of the data model to align IEC variables and structure items according to the target conventions. If wrong, some mismatch may happen when accessing in "C" an IEC variable, or some wrong addressing may lead to a hardware fault in the processor.

Virtual Addresses

The virtual machine works with a collection of predefined "system" variables. The VA (Virtual Address) of each system variable must be defined for the target, according to definitions done in the target software. It is generally not required to change these addresses. The VA of the first application variable enables the ISaGRAF PRO Linker to allocate space for system variables.

You also need to define the reserved size for On Line Change. This is the memory size, *allocated for each resource*, where downloaded changes will be stored.

Implementation Options

Options defined here must fit to compiling options defined for target integration. The text "end of line" character set is used for downloading of text files.

It is not recommended to change the maximum number of parameters for functions and function blocks, as it may lead to not portable IEC applications.

Other optional features correspond to compiling flags in the ISaGRAF PRO target software. Removing them can be required to reduce the code size of the virtual machine.

Extended Resource Parameters

Resource OEM defined parameters are sent to the virtual machine with the code of the resource. They allow customizing each particular instance of the ISaVM task. They are entered in a list of OEM parameters.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Array

Complex data types must be declared in the Target Definition Builder if used in the definition of other objects (POUs, conversions, I/O devices...).

An "array" data type is a multiple dimension array and can be based on a simple data type, or another array or structure declared in the target definition. In case of an array of STRINGS, the maximum length of the string must be specified (between 1 and 255).

The dimensions of the array are entered according the IEC syntax:

```
[ MinIndex .. MaxIndex , MinIndex .. MaxIndex ... ]
```

Dimensions cannot be less than 0.

Warning: Data types have a global scope within a template or project. Collision may happen if other type names are defined somewhere else. Refer to the **ISaGRAF PRO** User's Guide to know the list of reserved IEC keywords and names.

Structure

Complex data types must be declared in the Target Definition Builder if they are used in the definition of other objects (POUs, conversions, I/O devices...).

Warning: Data types have a global scope within a template or project. Collision may happen if other type names are defined somewhere else. Refer to the **ISaGRAF PRO** User's Guide to know the list of reserved IEC keywords and names.

To enter the structure items:

- You must enter for each item a valid name, a data type, and optionally a comment text.
- If data type is STRING, you must enter a maximum length (between 1 and 255).
- Select an item in the list and change its description in the controls below the list.
- Select "... " choice in the list to append a new item.

You can also use the following commands through push buttons:

- **Enter:** move list selection to the following item. "Enter" is generally used to append consecutive items at the end of a structure.
- **Insert:** insert a new item before the selected one.
- **Remove:** remove the selected item from the list.
- **Up and Down:** move the selected item in the list (used to arrange items in the structure).

Function

A function is a program organization unit (POU) written in "C" language and integrated to the embedded virtual machine. A function calculates a result using values of its input parameters.

The following pieces of information must be entered for a function:

- Calling Prototype
- On Line Info

Warning: A function name must be unique for a target within a project database. The same name cannot be used for a function and a function block. Refer to the **ISaGRAF PRO** User's Guide to know the list of reserved IEC keywords and names.

Calling Prototype

The prototype defines all the input parameters and the return value of the function. Prototype is entered in a list of parameters.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Function Block

A function block is a program organization unit (POU) written in "C" language and integrated to the embedded virtual machine. A function block has input and output parameters, plus local data duplicated for each instance of the block. Parameters and local data are stored in a structure.

The following pieces of information must be entered for a function block:

- Calling Prototype and Local Data
- Options
- On Line Info

Warning: A function block name must be unique for a target within a project database. The same name cannot be used for a function and a function block. Refer to the **ISaGRAF PRO** User's Guide to know the list of reserved IEC keywords and names.

Calling Prototype and Local Data

The prototype defines all the parameters and the local data of the function block. Prototype is entered in a list of parameters.

Options

You can choose to perform initialization for each instance of the block used in your project. The initialization choices are available from the Options menu:

- Yes, initialize instances
- No, don't call instance initialization

These options only make changes to the text file that is imported in the Workbench by writing both initialization functions (FblInstInit, FblInstExit) in the source file, even when the "NO" choice is checked. The initialization functions are used to initialize local variables for each instance of a function block. Note that the body of these functions could be empty. The kernel only calls these functions when the imported text file was created with the "YES" choice.

For function blocks not requiring initialization, not calling the initialization functions saves memory on the target.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Simple Device

A simple I/O device is a part of an I/O driver that pilots one group of channels with same data type and same direction (input or output). A simple I/O device can be designed for standalone usage, or to be a part of a complex I/O device.

The following pieces of information must be entered for a simple I/O device:

- I/O Driver Identification
- Definition of I/O Channels
- Parameters
- On Line Info

I/O Driver Identification

You must enter a package name and a driver name for the device. These names are used for dynamic link with the I/O driver when the virtual machine launches an application. Simple I/O devices which are parts of a complex device must have same package and driver names.

Definition of I/O Channels

You must select a data type and a direction for the channels of the device. You don't need to specify a maximum length in case of STRING data type.

You also need to define the number of I/O channels. If your driver handles it, the number of channels can be variable. In this case, the user will be able to adjust it when selecting the device in the I/O wiring tool. If your driver cannot support variable number of channels, enter the same value for "from" and "to" limits.

Parameters

OEM defined parameters are sent to the virtual machine with the description of the I/O. They allow special settings for each instance of the device. They are entered in a list of _OEM parameters.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Complex Device

A complex I/O device is a part of an I/O driver that pilots several simple I/O devices. Simple devices must be checked in the same set.

The following pieces of information must be entered for a complex I/O device:

- I/O Driver Identification
- List of Simple I/O Devices
- Parameters
- On Line Info

I/O Driver Identification

You must enter a package name and a driver name for the device. These names are used for dynamic link with the I/O driver when the virtual machine launches an application. Simple I/O devices which are parts of a complex device must have same package and driver names.

List of Simple I/O Devices

Available I/O devices are listed in the upper list. Only devices with same package and driver names are visible. The contents of the complex devices are displayed in the other list. Use the **Add** and **Remove** commands to add or remove a device to or from the complex device. The **Move Up / Down** commands allow you to change the order of the simple devices for rearranging the complex device.

Parameters

OEM defined parameters are sent to the virtual machine with the description of the I/O. They allow special settings for each instance of the device. They are entered in a list of OEM parameters.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Conversion Function

Conversion functions are "C" written functions that can be applied as filters to I/O channels.

The following pieces of information must be entered for a conversion function:

- Data Type
- Supported Conversion
- On Line Info

Data Type

This is the data type supported by the function. The Workbench will check that the conversion is used only for I/O channels with the corresponding data type.

Supported Conversion

You must specify whether your function supports conversion of input channels, or output channels, or both. The corresponding operations have to be supported in the "C" implementation of the conversion.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

Network Driver

Network drivers can be used either for IXL client / server exchanges, or for variable binding (real time data exchange among resources).

All OEM parameters are entered in a list of OEM parameters. The following pieces of information must be entered for a network driver:

- Supported Exchanges
- Remote Settings
- Connection Settings
- Resource Binding Settings
- Variable Binding Settings
- On Line Info

Supported Exchanges

You must specify whether your driver supports IXL communication, or variable binding, or both. It should obviously support at least one type of exchanges.

Remote Settings

Network OEM defined parameters for remote exchanges are sent to the configuration for settings of remote exchanges among configurations.

Connection Settings

Connection OEM defined parameters describe the connection of a configuration to the network. They are used for remote binding and for exchanges between the Workbench and remote configurations.

Resource Binding Settings

Network OEM defined parameters for resources are sent to the virtual machine with the code of each resource. They allow special setting of the binding among resources.

Variable Binding Settings

Network OEM defined parameters for bound variables are sent to the virtual machine with each binding link. They allow special setting of each particular variable binding.

On Line Info

You can define one of two types of on-line information that is launched by the Workbench when help is required by the user.

OEM Parameters

You must enter for each parameter a valid name, an access mode, a data type, a default value and optionally a comment text. Select an item in the list and change its description in the controls below the list. Select "... " choice in the list to append a new item.

Access mode can be:

- **User defined:** the user will have to enter a value for the parameter in the Workbench. Default is actually used as a "default value" in this case.
- **Read only:** the user can see the parameter value, but cannot change it. Default is the constant value of the parameter in this case.
- **Hidden:** the user cannot see the value of the parameter. Default is the constant value of the parameter in this case.

Data type can be:

- **Word:** signed 16 bit integer
- **Long:** signed 32 bit integer
- **Word hexa:** unsigned 16 bit integer entered in hexadecimal format
- **Long hexa:** unsigned 32 bit integer entered in hexadecimal format
- **Bool:** true/false value stored on one byte (only lowest bit is used)
- **Char:** single character
- **String:** array of 16 bytes containing a null terminated string
- **Float:** single precision 32 bit floating value

You can also use the following commands through push buttons:

- **Enter:** move list selection to the following item. "Enter" is generally used to append consecutive items at the end of a structure.
- **Insert:** insert a new item before the selected one.

- **Remove:** remove the selected item from the list.
- **Up and Down:** move the selected item in the list (used to arrange items in the structure).

Functions of FB Parameters

Parameters are displayed in three different lists:

- input parameters on the left
- local data within the block (for function blocks only)
- output parameters on the right

For each parameter, you must enter a valid name, a data type, and optionally a pin ID (short text to be displayed within the box in FBD and LD) and comment text. For the STRING data type, you must enter a maximum length between 1 and 255. Select an item in a list and change its description in the controls below the list. Select "... " choice in a list to append a new item.

You can also use the following commands through push buttons:

- **Enter:** move list selection to the following item. "Enter" is generally used to append consecutive items at the end of a structure.
- **Insert:** insert a new item before the selected one.
- **Remove:** remove the selected item from the active list.
- **Up and Down:** move the selected item in the active list (used to arrange items in the structure).

Name must fit to IEC language naming rules. This is not required for the pin ID.

Hidden Parameters

Hidden parameters are input parameters. These are declared when a C element is defined and are not shown in the FBD editor. Hidden parameters can only be assigned simple type values entered by the user through a dialog box. You can only set values for hidden parameters from the FBD editor when the function or function block is inserted in a calling program. All hidden parameters must have a valid default value which is applied if the user does not change it. Hidden parameters must be the last inputs.

To assign a value to a hidden parameter

1. In the FBD editor, insert the C function or C function block element having hidden parameters for which to assign values.
2. Double-click the element, then in the Select Blocks dialog, select the Parameters tab.
3. Assign the desired values to the parameters.

Pin ID

The "Pin ID" of a function or function block parameter is the short text displayed inside the block rectangle, when the function or function block is used in FBD or LD. The Pin ID is also called "**short name**" in ISaGRAF PRO documents.

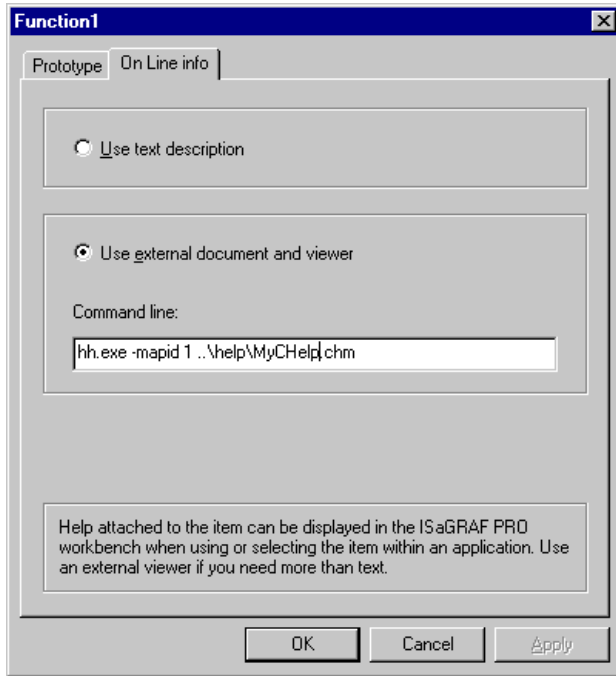
Defining On-line Information for Objects

You define on-line information for objects such as targets, functions and function blocks, simple devices and complex devices, conversion functions, and network drivers. Data types (arrays and structures) do not have on-line information. This on-line information is called from within the application when the user clicks Help and can be one of two types:

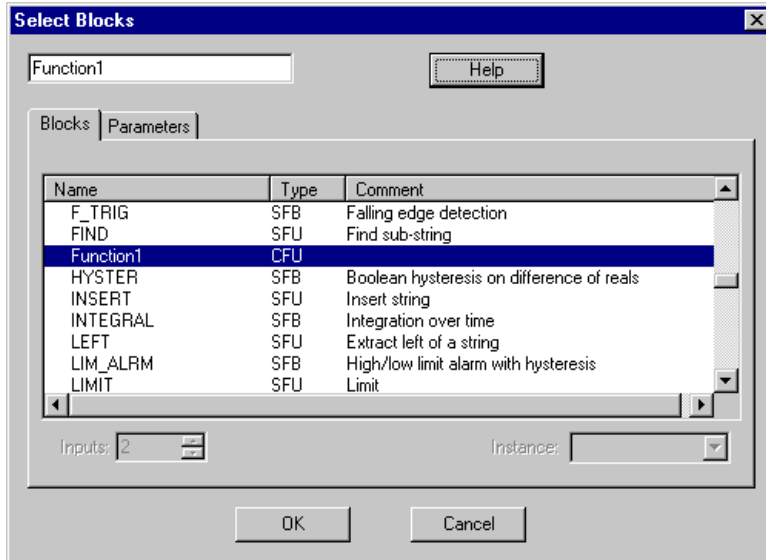
- A multi-line text description, entered in the Target Definition Builder, appearing in a basic comment window
- A separate document using an external viewer. For example, a HTML page contained in a CHM file.

To enter a text description, select the object on the tree and activate the second pane (Description) on the right of the document window where you can directly enter the text.

When using an external viewer, you must enter the complete command line (including arguments) that must be launched by the Workbench when help is required by the user.



For C functions and C function blocks, this on-line information is available from the Select Blocks dialog when the function or function block is selected from the list and the user clicks Help:



Selecting Objects in a Set

The Target Definition Builder works on a "set" when building exchange or "C" files. A "set" groups some of the objects defined in the document. You can define several sets. The name of the active set is displayed in the drop down list upon the object tree.

To add or remove objects to or from the active set, use the **Check** or **Remove Check** commands of the Build menu. You can also click on checkboxes shown in the object tree to add or remove objects.

To select the active set, use the drop down list or run the **Build / Sets** command. When using the menu command, the open dialog box can be used to create new sets, rename or delete existing sets. There must always be at least one set defined.

A set must be consistent. A consistency check is performed before any generation of exchange or "C" file. You can run a consistency check when you want by running the **Build / Verify** command.

Notes:

- A set cannot contain more than one target. It is optional to check a target in a set. If not, you're will be prompted to enter a target name when building an exchange file.
- When a complex I/O device is checked in a set, all the simple I/O devices referred to in the complex device must be checked too.

Building an Exchange File

The **Build / Send** to file command stores the definition of the selected set in an exchange file. The exchange file can then be imported in a project template or database, using the **File / Import** command of the Workbench. You need to enter the following information when building the exchange file:

- specify the name of the target to be created or updated in ISaGRAF PRO.
- enter a description for the created or modified target .
- select the pathname of the generated exchange file.
- enter a version number and a note to be embedded in the exchange file.

When sending a target definition file to a project or template database, you need to specify one of two options:

- **Fully redefine the target and all of its components.** This option is used to define a new target. When using this option on an already defined target (in the ISaGRAF PRO project database), it is possible to change some of the target's parameters. However, all elements previously attached to the target (C functions, IO drivers) are deleted, therefore, you need to redefine the set of components that you want to be available for this target.
- **Update or add components to the target.** This option creates some new elements or modifies already existing elements attached to a target but it does not make any modification on the target itself (Memory models, OEM parameters... are not modified).

Specifying a Target Name

The exchange file can be used to fully define a target. This is reserved for declaration of new targets when the ISaGRAF PRO embedded software is ported on another operating system. In this case, the target is the one checked in the active set.

The exchange file can also be used to add definitions to an existing target. This is required for adding new objects (POUs, I/O drivers...) integrated with the ISaGRAF PRO embedded software. In this case the file will not erase the current definition of the target when imported in a project database.

For updating a target, it is not mandatory to have a target checked in the active set. If not, you have to enter the name of the target you want to upgrade. If a target is checked in the set, you can use its name for upgrading, but its details are not stored in the exchange file when "update" mode is selected.

Entering a Description

Most of the case, when the exchange file describes a target upgrade, you may need to change the comment and multiple line description of the target in a project database. This is not required for a full target definition.

Selecting the Exchange File Pathname

Exchange file generally has the .TXT extension. The multiple line text description of the selected objects are stored in separate .TXT files in the same folder. (The embedding of description in the same file is not supported by the ISaGRAF PRO Import tool.) It is recommended to send exchange files in a separate folder. If several files are created, you must keep them together on the same folder for importation in ISaGRAF PRO. If you move them to another folder, you will need to adjust pathnames in the main files (these are absolute pathnames). When build is complete, the list of created file is displayed in a box.

Enter a Version Number and Note

The version number and note entered in the last pane of the **Send to file** wizard will be written at the beginning of the exchange file. It is recommended to maintain version and note in order to keep track of exchange file versions.

Refer to the **ISaGRAF PRO** Development Kit user's guide for further information about the 'Import' procedure.

Generating "C" Source Code

The **Tools / Generate "C" code** command build templates of "C" source files for development and integration in the ISaGRAF PRO embedded software of the objects declared in the Target Definition Builder. The following steps are required before generating code:

- select a style of "C" source code template to be generated.
- select the objects defined in the active set to be described in "C" interfaces.
- select a destination folder and adjust the name generated "C" and "H" files.
- check whether you want extra information to be stored as comments in the source.
- check whether you want to generate DLLs for NT or libraries for other operating system.

When build is complete, the list of created file is displayed in a box. You can now add them to a "C" project or makefile, and start completing them.

Refer to the **ISaGRAF PRO** developer's guides for further information about "C" integration of components.

Other Tools

Other useful commands are available in the Tools menu:

- **references to type:** this command displays in the build bar all the objects based on a specified data type. It can be either a predefined simple data type or a user defined array or structure type.
- **references to I/O device:** this command displays in the build bar all the complex I/O devices using the specified simple I/O device.
- **undefined references:** this command displays in the build bar all the items referencing an unknown object. Undefined references may happen after deleting a data type.

Found objects are displayed in blue in the build bar. You can double click on a line to open the description of the object.

Importing ISaGRAF 3.3 Libraries

The Target Definition Builder can read libraries from ISaGRAF version 3.3x. Use the **Import 3.3 library** of the **Tools** menu. You have to select the folder where ISaGRAF 3.3 libraries are stored. This is the "LIB" sub-folder of the installed ISaGRAF 3.3x Workbench. Default installation is:

```
C:\ISAWIN\LIB
```

All elements from the library are imported: "C" functions and function blocks, I/O simple and complex devices and conversion functions.

The definition of ISaGRAF 3.3 elements is put in a new document. Use the **Save As** command to store it in a target definition file for ISaGRAF PRO, or use the **Set / Send To File** command for importation in an ISaGRAF PRO project or template database.

Notes:

- The "C" source code of POU's and I/O devices is not imported. You will have to import it manually from ISaGRAF and to adapt it according to new "C" interfaces defined for PRO.
- All imported complex and simple drivers have their package name and driver name set to "ISA33". You can change it if you want to use other names in your PRO implementation.

ISaGRAF 3.3 accepts both DINT and REAL data types for "Analog" simple I/O devices and conversion functions. The import tool turns such data types to DINT.

Naming Conventions

All data types, functions, function blocks and conversion functions must fit to IEC language naming conventions. The first character must be a letter or an underscore sign. Following characters can be letters, digits or underscore signs. There cannot be two underscores together in the name. The length of the name cannot exceed 128 characters.

All I/O devices and network drives must fit to "C" language naming conventions. The first character must be a letter or an underscore sign. Following characters can be letters, digits or underscore signs. The length of the name cannot exceed 128 characters.

All target names are limited to 15 characters. The first character must be a letter or an underscore sign. Following characters can be letters, digits, underscore sign or dash (minus) sign.

Names are not case sensitive. The Target Definition Builder automatically adjusts case when generating "C" source code.

Copyright

Information in these pages is subject to change without notice and does not represent a commitment on the part of ICS Triplex ISaGRAF Inc. No part of these pages may be reproduced in any form or by any means, electronic or mechanical, for any purpose without the express written permission of ICS Triplex ISaGRAF Inc.

© 1999-2003 ICS Triplex ISaGRAF Inc. All rights reserved.

Product or company names included in these pages are trademarks or registered trademarks of their respective holders.

All logos and links used in this guide are, to the best of our knowledge, included with the permission of the owner - if this is not the case, please let us know immediately.

Any changes made to documentation issued by ICS Triplex ISaGRAF Inc. without prior permission of ICS Triplex ISaGRAF Inc. (in writing) will void any responsibilities and liabilities normally associated with its contents.

