SCADAPack E ISaGRAF Function Block Reference

Schneider Electric

Documentation

Table of Contents

I SCADAPack E ISaGRAF Function Block Reference

1 Technical Support

Support related to any part of this documentation can be directed to one of the following support centers.

Technical Support: The Americas

Available Monday to Friday 8:00am – 6:30pm Eastern Time

Toll free within North America	1-888-226-6876
Direct Worldwide	+1-613-591-1943
Email	TechnicalSupport@controlmicrosystems.com

Technical Support: Europe

Available Monday to Friday 8:30am – 5:30pm Central European Time

Direct Worldwide	+31 (71) 597-1655
Email	euro-support@controlmicrosystems.com

Technical Support: Asia

Available Monday to Friday 8:00am – 6:30pm Eastern Time (North America)

Direct Worldwide	+1-613-591-1943
Email	TechnicalSupport@controlmicrosystems.com

Technical Support: Australia

Inside Australia	1300 369 233
Email	au.help@schneider-electric.com

2 Safety Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡	The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.
⚠	This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠DANGER
DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠WARNING
WARNING indicates a potentially hazardous situation which, if not avoided, **can result** in death or serious injury.

⚠CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, **can result** in minor or moderate injury.

CAUTION
CAUTION used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage..

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠CAUTION
EQUIPMENT OPERATION HAZARD
• Verify that all installation and set up procedures have been completed. • Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.

• Remove tools, meters, and debris from equipment.
Failure to follow these instructions can result in injury or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and grounds, except those grounds installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove ground from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

3 Preface

Scope

This manual describes in details all the custom function blocks provided with the SCADAPack E RTU ISaGRAF installation.

Purpose

The purpose of this document is to describe the custom function blocks provided with the SCADAPack E ISaGRAF installation.
This manual is intended to be used in conjunction with the *SCADAPack E ISaGRAF Technical Reference* and *SCADAPack E ISaGRAF I/O Connection Reference* manuals.

Assumed Knowledge

Familiarity with the ISaGRAF Workbench is strongly recommended.

Target Audience

- Systems Engineers
- Commissioning Engineers
- Maintenance Technicians

References

- *SCADAPack E Configuration Reference* manual
- ICS Triplex ISaGRAF Manuals
- *SCADAPack E ISaGRAF Technical Reference* manual
- *SCADAPack E I/O Connection Reference* manual

4 Overview

This document describes Function Blocks and Functions supported by the SCADAPack E RTU processors when using the ISaGRAF IEC61131-3 environment.

ISaGRAF refers to these as "C Functions" and "C Function Blocks" within the ISaGRAF Libraries and Archive.

These function blocks or functions provide an ISaGRAF application access to RTU configuration and I/O data in the form of point properties and attributes, access to RTU facilities such as serial ports, point data objects, peer-to-peer DNP3 communication, real time clock and additional control functions.

Functions can be implemented using the IEC 61131-3 Function Block Diagram or Structured Text languages.

The manual is arranged as follows:

- Section ***SCADAPack E Function Blocks*** 11 is sub-divided as follows:
  - Section ***Point Attribute Function Blocks*** 12 presents the function blocks that can be used to return or set RTU point properties
  - Section ***Real Time Clock Function Blocks*** 77 describes function blocks which provide an ISaGRAF application access to the RTU’s real time clock.
  - Section ***DNP3 Communication Function Blocks*** 80 describes ISaGRAF function blocks that interface with the SCADAPack E RTU’s peer-to-peer communication facilities. These function blocks transfer data between an ISaGRAF variable and a remote RTU point database.
  - Section ***DNP3 Queued Communication Function Blocks*** 100 describes ISaGRAF function blocks that interface with the SCADAPack E RTU’s peer-to-peer communication facilities. Queued peer function blocks transfer data directly between the point databases of two peer RTUs.
  - Section ***Serial Port User Communication Functions*** 117 describes ISaGRAF function blocks which can be used to set or retrieve serial port properties on the SCADAPack E RTU.
  - Section ***Miscellaneous Function Blocks*** 127 describes several miscellaneous function blocks including the PID_AL function block that provides PID control with output limiting and anti-integral windup protection.
  - Section ***TCP/IP Interface Functions*** 164 describe function blocks that provide interfaces to RTU TCP/IP communication and configuration services.
  - Section ***Alarm Group Functions & Function Block*** 173 describes the ISaGRAF ‘Summary Alarm’ functions and function blocks. These provide a mechanism for grouping individual RTU Digital point states (alarms) in to a named alarm group, and configuring an alarm output if any of the points in the group change to the alarm state.
  - Section ***RTU File System Interface Functions & Function Blocks*** 181 describes the ISaGRAF interfaces to the RTU's file system to allow manipulation of RTU files

5 SCADAPack E Function Blocks

The function blocks and functions listed in the following sections are provided by Schneider Electric as facilities in the ISaGRAF workbench library. They may be used in SCADAPack E RTU ISaGRAF applications as detailed in the following sections.

5.1 Point Attribute Function Blocks

These function blocks set or return the property or attribute of a point stored in the RTU database. Point properties or attributes that can be set or retrieved using the SCADAPack E Configurator can also be accessed using the function blocks presented in this subsection of the manual.

Physical and user points within the SCADAPack E RTU (and some system points) have both **Property** and **Attribute** fields associated with them. The distinction between these field types within a point is particularly needed for point processing.

A “**Property**” field of a point represents a physical or derived quantity, describing a particular aspect of the real time condition of a point.

An “**Attribute**” field of a point dictates how the SCADAPack E RTU should manipulate or present a particular aspect of a point. In terms of data processing, some attributes describe how some point properties are derived. Multiple point attributes may impact a point property. Similarly, multiple point properties may be impacted by a single attribute. For more information on point properties and attributes, consult the *SCADAPack E Configuration Technical Reference* manual.

- ***getpnt & setpnt*** 13
- ***rdfld*** 27
- ***rdrec*** 40
- ***rdstring*** 54
- ***rtupulse & rtupuls2***
- ***setatr*** 69

5.1.1 getpnt & setpnt

Get database point value (getpntxx) and Set database point value (setpntxx) functions and function blocks are used to interface ISaGRAF IEC61131-3 logic with the SCADAPack E point database.

getpntxx functions and function blocks read database point values in to ISaGRAF variables, in various formats.

setpntxx function and function blocks write ISaGRAF variable values in to database point variables, in various formats.

These functions require significantly more processing time than accessing points through ISaGRAF I/O boards. See *SCADAPack E ISaGRAF I/O Connection* reference manual. Impact on an ISaGRAF application's performance should be considered when using these functions instead of I/O boards.

- **getpntb** 14 (get point binary)
- **getpntc** 15 (get point counter)
- **getpntf** 17 (get point floating point)
- **getpntsl** 18 (get point signed long)
- **getpntss** 19 (get point signed short)
- **getpntus** 21 (get point unsigned short)

- **setpntb** 22 (set point binary)
- **setpntf** 23 (set point floating point)
- **setpntsl** 24 (set point signed long)
- **setpntss** 25 (set point signed short)
- **setpntus** 26 (set point unsigned short)

5.1.1.1 getpntb

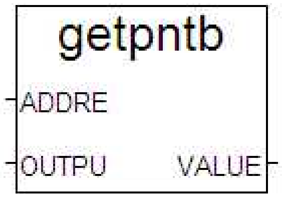
getpntb

Get value of binary point

Description

The **getpntb** function returns the boolean state of a specified database binary point. If the point is found and its current value is ON then TRUE is returned, otherwise FALSE is returned.

This function provides a method for accessing binary database points.



Arguments

Inputs	Type	Description
address	Integer	Address of a database binary point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a binary input, binary output, derived binary point or system binary point
output_point	Boolean	Set to TRUE for the **address** to refer to a binary output point. Set to FALSE for the **address** to refer to a binary input point, derived binary point or system binary point.

Outputs	Type	Description
value	Boolean	TRUE if value of the binary point is ON FALSE if value of the binary point is OFF or does not exist

prototype: value := GETPNTB (point_num, is_output_point);

See Also

setpntb 22

5.1.1.2 getpntc

getpntc

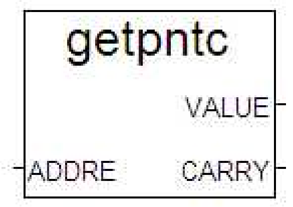
Get value of counter point

Description

The **getpntc** function block returns the integer value of a specified database counter point. If the point is found the current count value is returned, otherwise 0 is returned.

ISaGRAF treats integer values as 32-bit signed. The database treats counter values as 32-bit unsigned, therefore the value returned by this function block is limited to (2^31 - 1). If the internal value is in excess of (2^31 - 1) then the carry flag will be set and the value returned will be in the range 0 to (2^31 - 1).

This function provides a method for accessing database counter points.



Arguments

Inputs	Type	Description
address	Integer	Address of a database binary point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a binary input, binary output, derived binary point or system binary point

Outputs	Type	Description
value	Integer	31-bit signed integer value of counter point if found, otherwise 0 Database counter point is in 32-bit unsigned format. ISaGRAF variable is in 32-bit signed format. If the database counter point is greater than (2^31 - 1) then **carry** flag is set and the value returned is (count - 2^31 - 1)
carry	Boolean	TRUE if value of the counter point is greater than (2^31 -1) FALSE if value of the counter point is lessthan or equal to (2^31 - 1)

prototype:

getpntc_inst (POINT)

```
counts :=  getpntc_inst.VALUE
inc_next_ctr := getpntc_inst.CARRY
```

5.1.1.3 getpntf

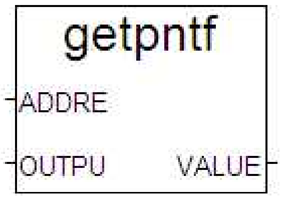
getpntf

Get value of analog point as floating point (real)

Description

The **getpntb** function returns the engineering value of a specified database analog point. If the point is found then the current value is returned, otherwise 0.0 is returned.

This function provides a method for accessing analog database points.



Arguments

Inputs	Type	Description
address	Integer	Address of database analog point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a analog input, analog output, derived analog point or system analog point
output_point	Boolean	Set to TRUE for the **address** to refer to an analog output point. Set to FALSE for the **address** to refer to an analog input point, derived analog point or system analog point.

Outputs	Type	Description
value	Real	Current Engineering Value if found, otherwise 0.0

prototype: value := GETPNTF (point_num, is_output_point);

See Also

setpntf 23

5.1.1.4 getpntsl

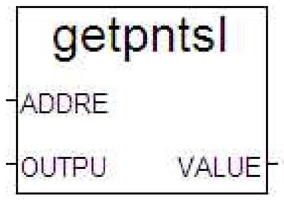
getpntsl

Get value of analog point as signed long integer

Description

The **getpntsl** function returns the integer value of a specified database analog point. If the point is found then the current value is returned, otherwise 0 is returned.

This function provides a method for accessing analog database points.



Arguments

Inputs	Type	Description
address	Integer	Address of database analog point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a analog input, analog output, derived analog point or system analog point
output_point	Boolean	Set to TRUE for the **address** to refer to an analog output point. Set to FALSE for the **address** to refer to an analog input point, derived analog point or system analog point.

Outputs	Type	Description
value	Integer	Current Integer Value if found, otherwise 0

prototype: value := GETPNTSL (point_num, is_output_point);

See Also

setpntsl 24

5.1.1.5 getpntss

getpntss

Get value of analog point as signed short integer

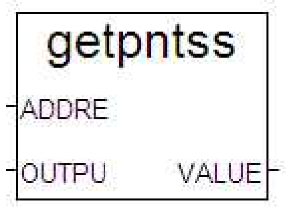
Description

The **getpntss** function returns the integer value of a specified database analog point. If the point is found then the current value is returned, otherwise 0 is returned.

If the current integer value of the analog point is > 32767, then the output value is clamped to 32767

If the current integer value of the analog point is < -32768, then the output value is clamped to -32768

This function provides a method for accessing analog database points.



Arguments

Inputs	Type	Description
address	Integer	Address of database analog point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a analog input, analog output, derived analog point or system analog point
output_point	Boolean	Set to TRUE for the **address** to refer to an analog output point. Set to FALSE for the **address** to refer to an analog input point, derived analog point or system analog point.

Outputs	Type	Description
value	Integer	Current Integer Value if found, otherwise 0 The value from the analog point is clamped in the range -32768 to 32767

prototype: value := GETPNTSS (point_num, is_output_point);

See Also

setpntss 25

5.1.1.6 getpntus

getpntus

Get value of analog point as unsigned short integer

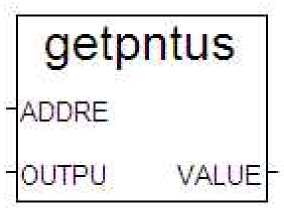
Description

The **getpntus** function returns the integer value of a specified database analog point. If the point is found then the current value is returned, otherwise 0 is returned.

If the current integer value of the analog point is > 65535 , then the output value is clamped to 65535

If the current integer value of the analog point is < 0, then the output value is clamped to 0

This function provides a method for accessing analog database points.



Arguments

Inputs	Type	Description
address	Integer	Address of database analog point number to read. In conjunction with the **output_point** parameter, it refers to a unique database point. Point may be a analog input, analog output, derived analog point or system analog point
output_point	Boolean	Set to TRUE for the **address** to refer to an analog output point. Set to FALSE for the **address** to refer to an analog input point, derived analog point or system analog point.

Outputs	Type	Description
value	Integer	Current Integer Value if found, otherwise 0 The value from the analog point is clamped in the range 0 to 65535

prototype: value := GETPNTUS (point_num, is_output_point);

See Also

setpntus 26

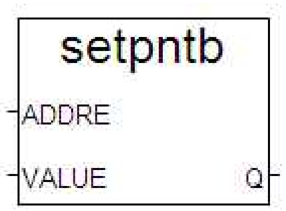
5.1.1.7 setpntb

setpntb

Set value of binary point

Description

The **setpntb** function writes the boolean state of **value** to the specified database binary point's current value. If the point is found and the value is successfully written, the output variable is set TRUE; otherwise it is set FALSE. If the point is not found, nothing is done and the output is set FALSE.



Arguments

Inputs	Type	Description
address	Integer	Address of a binary point number. Point may be a binary output, derived binary point or system binary point.
value	Boolean	Value to write to the binary point.

Outputs	Type	Description
Q	Boolean	TRUE if the value was successfully set. FALSE for other cases.

prototype: setok := SETPNTB (point_num, req_state);

See Also

getpntb 14

5.1.1.8 setpntf

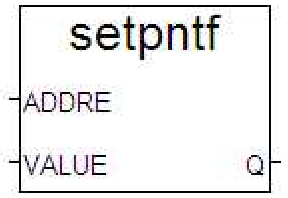
setpntf

Set value of analog point as floating point (real)

Description

The **setpntf** function writes the **value** to the specified database analog point's Current Engineering Value. If the point is found and the value is successfully written, the output variable is set TRUE; otherwise it is set FALSE. If the point is not found, nothing is done and the output is set FALSE.

This function can set the value of physical analog outputs, derived analogs and system analog points.



Arguments

Inputs	Type	Description
address	Integer	Address of a binary point number. Point may be a binary output, derived binary point or system binary point.
value	Real	Value to write to the binary point.

Outputs	Type	Description
Q	Boolean	TRUE if the value was successfully set. FALSE for other cases.

prototype: setok := SETPNTF (point_num, req_state);

See Also

getpntf 17

5.1.1.9 setpntsl

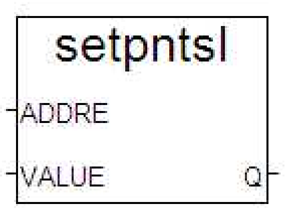
setpntsl

Set value of analog point as signed long integer

Description

The **setpntsl** function writes the **value** to the specified database analog point's Current Integer Value. If the point is found and the value is successfully written, the output variable is set TRUE; otherwise it is set FALSE. If the point is not found, nothing is done and the output is set FALSE.

This function can set the value of physical analog outputs, derived analogs and system analog points.



Arguments

Inputs	Type	Description
address	Integer	Address of an analog point number. Point may be an analog output, derived analog point or system analog point.
value	Integer	Value to write to the analog point.

Outputs	Type	Description
Q	Boolean	TRUE if the value was successfully set. FALSE for other cases.

prototype: setok := SETPNTSL (point_num, req_state);

See Also

getpntsl 18

5.1.1.10 setpntss

setpntss

Set value of analog point as signed short integer

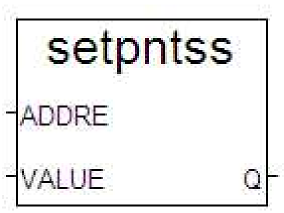
Description

The **setpntss** function writes the **value** to the specified database analog point's Current Integer Value. If the point is found and the value is successfully written, the output variable is set TRUE; otherwise it is set FALSE. If the point is not found, nothing is done and the output is set FALSE.

If the input value is > 32767, then the point value will be clamped to 32767

If the input value is < -32768, then the point value will be clamped to -32768

This function can set the value of physical analog outputs, derived analogs and system analog points.



Arguments

Inputs	Type	Description
address	Integer	Address of an analog point number. Point may be an analog output, derived analog point or system analog point.
value	Integer	Value to write to the analog point. The value written to the point is clamped in the range -32768 to 32767

Outputs	Type	Description
Q	Boolean	TRUE if the value was successfully set. FALSE for other cases.

prototype: setok := SETPNTSS (point_num, req_state);

See Also

getpntss 19

5.1.1.11 setpntus

setpntus

Set value of analog point as unsigned short integer

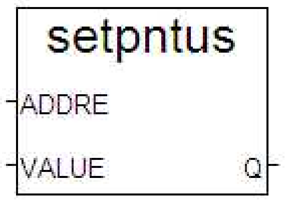
Description

The **setpntus** function writes the **value** to the specified database analog point's Current Integer Value. If the point is found and the value is successfully written, the output variable is set TRUE; otherwise it is set FALSE. If the point is not found, nothing is done and the output is set FALSE.

If the input value is > 65535, then the point value will be clamped to 65535

If the input value is < 0, then the point value will be clamped to 0

This function can set the value of physical analog outputs, derived analogs and system analog points.



Arguments

Inputs	***Type***	***Description***
address	Integer	Address of an analog point number. Point may be an analog output, derived analog point or system analog point.
value	Integer	Value to write to the analog point. The value from the point is clamped in the range 0 to 65535

Outputs	***Type***	***Description***
Q	Boolean	TRUE if the value was successfully set. FALSE for other cases.

prototype: setok := SETPNTUS (point_num, req_state);

See Also

getpntus 21

5.1.2 rdfld

Read point field function blocks

- *rdfld_i* 28
- *rdfld_r* 35

5.1.2.1 rdfld_i

Read point attribute of type integer

Description

The rdfld_i function reads an RTU point attribute or property and returns the value in a 32 bit integer variable.

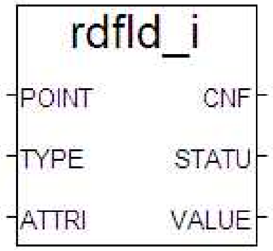


Figure 6.1: rdfld_i Function Block

Inputs	Type	Description
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Type	Integer	RTU Point Data Type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Comment** DIN / 1 / Digital input point Dout / 2 / Digital output point AIN / 3 / Analog input point AOUT / 4 / Analog output point CIN / 5 / Counter input point
Attrib	Integer	Desired point attribute (property) argument can be an ISaGRAF reserved keyword or an integer value corresponding to the attribute. See ***Table 6.1*** 29 to ***Table 6.4*** 33 for keywords and their corresponding numeric values.

Outputs	TYPE	DESCRIPTION
CNF	Boolean	Confirm valid or invalid status **Possible Values** / **Meaning** TRUE / Confirm Valid Status FALSE

Status	Integer	Status of Read Request **Possible Values** — **Meaning** -1 Unknown Return Error 0 Success 1 Point does not exist 2 Bad point type 3 Unknown attribute for this point 4 Bad value for this attribute 5 Invalid attribute for this function block 8 Point is locked 12 Database is locked 18 Data Processor Unavailable
Value	Integer	Depends on 'Attrib' input parameter.

Table 6.1: rdfld_i Function Attributes (attrib) for Point Properties

Attribute (ISaGRAF Keyword)	**Equivalent Numeric Value**	**Description**
Qlty	100	Point Quality
Fail	101	Point Failed
IOF	102	IO Not Responding
ISA	103	ISaGRAF Controlled
Ilock	104	Remote Control Interlock Enabled
State	105	Current State
Alarm	106	Binary Point is in Alarm (this attribute is used with Binary points only).
A4H	114	Alarm Limit Transgress 4H
A3H	113	Alarm Limit Transgress 3H
A2H	112	Alarm Limit Transgress 2H
A1H	111	Alarm Limit Transgress 1H
A1L	110	Alarm Limit Transgress 1l
A2L	109	Alarm Limit Transgress 2l
A3L	108	Alarm Limit Transgress 3l
A4L	107	Alarm Limit Transgress 4l
Raw	115	Current Integer Value
RoRise	117	Rate Of Rise Exceeded
RoFall	118	Rate of Fall Exceeded
NC	119	No Change Detected
ORange	120	Over Range

URange	121	Under Range
ADF	122	Ad Reference Check
HI	123	High Limit Exceeded
CntRZ	53	Counter Reset to Zero on Power Up
PObj	3	DNP Static Object Type
DigDbActive	128	Digital Dead-band Timer Active
AnaDbActive	129	Analog Dead-band Timer Active

Table 6.2: rdfld_i Function Attributes (attrib) for Integer (Analog) Points.

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description
RoRPN	17	Rate Of Rise Point Number
RoFPN	18	Rate Of Fall Point Number
NCPN	19	No Change Point Number
CexPN	20	Counter Exceeded Point Number
Rmin	29	Raw Min
Rmax	30	Raw Max
RoCTm	38	Rate Of Change Time
NCTm	39	No Change Time
Ev4H	49	Limit Event Enable 4h
Ev3H	48	Limit Event Enable 3h
Ev2H	47	Limit Event Enable 2h
Ev1H	46	Limit Event Enable 1h
Ev1L	45	Limit Event Enable 1l
Ev2L	44	Limit Event Enable 2l
Ev3L	43	Limit Event Enable 3l
Ev4L	42	Limit Event Enable 4l
LimHi	51	Counter High Limit
CntDev	52	Counter Change Deviation
EvDev	50	Event Deviation
EvDevUnsol	65	Event Deviation Unsolicited Enable
EvDevTp	97	Event Deviation Type

Table 6.3: rdfld_i Function Attributes (attrib) for Digital (Boolean) Points

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description
Inv	11	Invert Point State

AState	12	Alarm Active State
AClrTm	14	Alarm Clear Time Deadband
PTm	15	Output Pulse Time
DebTm	16	Debounce Time
DblStPt	54	Double Status Point Number

Table 6.4: rdfld_i Function Attributes (attrib)

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description
AInh	7	Alarm Inhibit
ATm	10	Alarm Time Deadband
Bad	6	Point Is Bad
IlockPN	4	Remote Control Interlock Point
IlockTm	5	Interlock Alarm Timeout
PrId	9	Profile Id
TInh	8	Trend Inhibit
PClassMA	66	Point Data Class (All Masters)
PClassM1	67	Point Data Class Master 1
PClassM2	68	Point Data Class Master 2
PClassM3	69	Point Data Class Master 3

The keywords in ***Table 6.1*** 29 to ***Table 6.4*** 33 are reserved and should be used exclusively to retrieve point properties using this function block. User defined variables which duplicate these keywords but are not being used in the same context will generate errors during the compilation.

The PClass attribute (value 2) is deprecated. It is replaced by the PClassM1 attribute. The attribute was changed in firmware 7.39 with the addition of multi-master functionality, but was not changed in ISaGRAF until version 7.71.

An IEC61131-3 Function Block Diagram example of RDFLD_I is illustrated in ***Figure 6.2*** 33 below. The current integer value (Attrib = Raw) of analog input point labeled z_speedcontrol (analog input point 1) has been obtained.

The preprocessor has been configured to create a dictionary variable z_speedcontrol of type 'defined word. The variable z_speedcontrol contains the point address of analog input point labeled 'speedcontrol'. Alternatively, the numeric point address of the analog input can also be used.

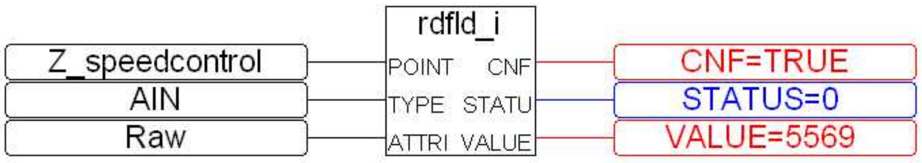


Figure 6.2: rdfld_i Function Block Diagram Example

IEC61131-3 Structured Text prototypes take on the following form:

```
prototype:  rdfld_i_inst (POINT, TYPE, ATTRIB)
            complete_confirm :=  rdfld_i_inst.CNF
            return_status := rdfld_i_inst.STATUS
            return_value := rdfld_i_inst.VALUE
```

where rdfld_i_inst is an FB instance of the function block rdfld_ defined in the program dictionary.

An equivalent Structured Text implementation of the function block diagram in ***Figure 6.2*** 33 is listed below. This code will store the outputs of the function block rdfld_i in the variables CNF, STATUS and VALUE.

```
(*              Code Begins Here              *)
(*  Ensure dictionary has the following
    variables defined:                        *)
(*  Boolean             CNF                   *)
(*  integer             STATUS                *)
(*  Real                Value                 *)
(*  FB instances        rdfld_i_inst          *)
rdfld_i_inst(z_speedcontrol, AIN, Raw);
if (rdfld_i_inst.CNF) then
    CNF := rdfld_i_inst.CNF;

    STATUS := rdfld_i_inst.status;

    VALUE := rdfld_i_inst.value;

end_if;
(* Code Ends Here *)
```

5.1.2.2 rdfld_r

Read point attribute of type real

Description

Point database attributes and property fields that return real (float) values may be read by ISaGRAF application using the function block “RDFLD_R”.

Attributes and properties may be read from the point database. The format of this function block is the same as “RDFLD_I” except that the “Value” field in the case of “RDFLD_R” is a Real (float) value, and in the case of “RDFLD_I” is an Integer value. The description below illustrates the purpose, inputs and outputs of the RDFLD_R function block. Each time the function block is called, the RTU updates the ISaGRAF value variable from the specified point database field for the specified RTU point and point type.

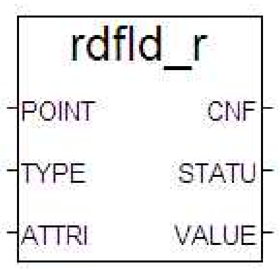


Figure 6.3: rdfld_r Function Block

Inputs	Type	Description
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of DNP point address defines
Type	Integer	RTU Point Data Type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** | **Equivalent Numeric Value** | **Comment** DIN | 1 | Digital input p DOUT | 2 | Digital output AIN | 3 | Analog input AOUT | 4 | Analog outpu CIN | 5 | Counter input
Attrib	Integer	Desired point attribute (property) argument can be an ISaGRAF reserved keyword or an integer value corresponding to the attribute. See ***Table 6.5*** 36 for keywords and their corresponding numeric values.

Outputs	TYPE	DESCRIPTION
CNF	Boolean	Confirm valid or invalid status **Possible Values** / **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possible Values** / **Meaning** -1 — Unknown Return Error 0 — Success 1 — Point does not exist 2 — Bad point type 3 — Unknown attribute for this poi 4 — Bad value for this attribute 5 — Invalid attribute for this functio 8 — Point is locked 12 — Database is locked 18 — Data Processor Unavailable
Value	Integer	Depends on 'Attrib' input parameter.

Table 6.5: rdfld_r Function Attributes (attrib)

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description
Emin	31	Engineering Min
Emax	32	Engineering Max
Eng	116	Current Engineering Value
Lim4H	28	Engineering Limit 4H
Lim3H	27	Engineering Limit 3H
Lim2H	26	Engineering Limit 2H
Lim1H	25	Engineering Limit 1H
Lim1L	24	Engineering Limit 1L
Lim2L	23	Engineering Limit 2L
Lim3L	22	Engineering Limit 3L
Lim4L	21	Engineering Limit 4L
LimRise	35	Rate Of Rise
LimFall	36	Rate Of Fall
LimNC	37	No Change
AclrVal	40	Alarm Clear Value Deadband

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description
LimZero	41	Zero Threshold Limit
EvDev	50	Even _Deviation
LimOR	33	Over Range Limit
LimUR	34	Under Range Limit

The keywords in ***Table 6.5*** 36 are reserved and should be used exclusively to retrieve point properties using this function block. User defined variables which duplicate these keywords but are not being used in the same context will generate errors during the compilation.

An IEC61131-3 Function Block Diagram example of RDFLD_R is illustrated in ***Figure 6.4*** 37 below. The current floating point or real value of analog input point labeled z_speedcontrol (analog input pont 1) has been obtained.

The preprocessor has been configured to create a dictionary variable z_speedcontrol of type 'defined word. The variable z_speedcontrol defines the point address of analog input point labeled 'speedcontrol'. Alternatively, the numeric point address of the analog input can also be used.

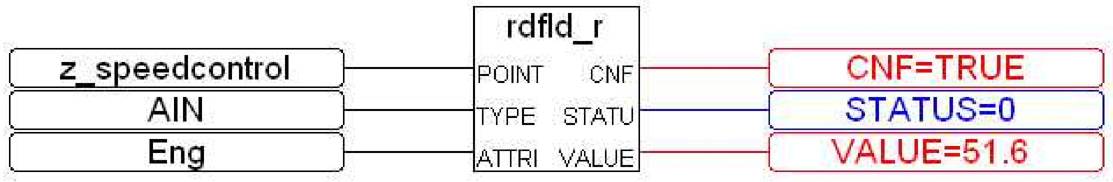


Figure 6.4: rdfld_r Function Block Diagram Example

IEC61131-3 Structured Text prototypes take on the following form:

```
prototype:  rdfld_r _inst (POINT, TYPE, ATTRIB)
            complete_confirm :=  rdfld_r_inst.CNF
            return_status := rdfld_r_inst.STATUS
            return_value := rdfld_r_inst.VALUE
```

where rdfld_r_inst is an instance of the function block rdfld_r defined in the program dictionary.

An equivalent Structured Text implementation of the function block diagram in ***Figure 6.4*** 37 is listed below. This code will store the outputs of the function block rdfld_r in the variables CNF, STATUS and VALUE.

```
(*              Code Begins Here              *)
(*   Ensure dictionary has the following      *)
     variables defined:
(*   Boolean            CNF                   *)
(*   integer            STATUS                *)
(*   Real               Value                 *)
(*   FB instances            rdfld_r_inst            *)
rdfld_r_inst(z_speedcontrol, AIN, Eng);
if (rdfld_r_inst.CNF) then
    CNF := rdfld_r_inst.CNF;

    STATUS := rdfld_r_inst.STATUS;
    VALUE := rdfld_r_inst.VALUE;

end_if;
(* Code Ends Here *)
```

5.1.3 rdrec

Function blocks to Read common fields from a point record

- *rdrec* 40
- *rdrec_dg* 44
- *rdrec_cn* 47
- *rdrec_an* 49

5.1.3.1 rdrec

Read point attribute of type real

Description

The RDREC function block reads the most common attributes for point types. If an attribute is not defined for the point in question, a value of zero is returned.

The description below illustrates the purpose, inputs and outputs of the RDREC function block.

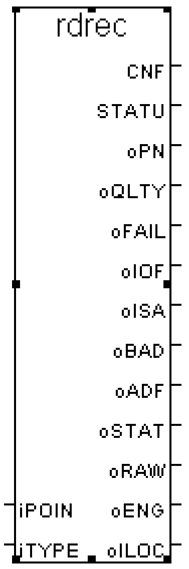


Figure 6.5: rdrec Function Block

Inputs	Type	Description
iPOINT	Integer	RTU Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
iType	Integer	RTU Point Data Type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Comment** DIN / 1 / Digital input p DOUT / 2 / Digital output AIN / 3 / Analog input AOUT / 4 / Analog outpu CIN / 5 / Counter input

Outputs	TYPE	DESCRIPTION
CNIF	Boolean	Confirm valid or invalid status **Possible Values** / **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possbile Values** / **Meaning** -1 — Unknown Return Error 0 — Success 1 — Point does not exist 2 — Bad Point Type 3 — Unknown attribute for this point 4 — Bad Value for this attribute 5 — Invalid Attribute for this function block 8 — Point is locked 12 — Database is locked 18 — Data Processor Unavailable
oPN	Integer	Point Number
oQlty	Integer	Point Quality
oFail	Boolean	Point Failed
oIOF	Boolean	ISaGRAF Controlled
oBAD	Boolean	Point is Bad
oADF	Boolean	A/D Reference Error
oSTATE	Boolean	Point State
oRAW	Integer	Raw Point Value
oENG	Real	Engineering (Real) Value
oIlLOC	Boolean	Contorl Interlock Active

An IEC61131-3 Function Block Diagram example of RDFLD_R is illustrated in below.

The preprocessor has been configured to create a dictionary variable z_speedcontrol of type ‘defined word. The variable z_speedcontrol contains the point address of analog input point labeled ‘speedcontrol’. Alternatively, the numeric point address of the analog input can also be used.

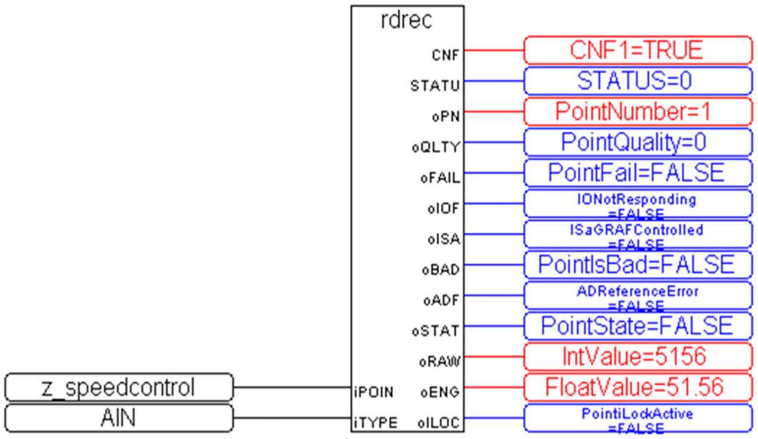


Figure 6.6: rdrec Function Block Example

IEC61131-3 Structured Text prototypes take on the following form:

```
prototype:  rdrec_inst (POINT, TYPE)
            complete_confirm :=  rdrec_inst.CNF
            return_status := rdrec_inst.STATUS
            Point_number := rdrec_inst.oPN .......
```

where rdrec_inst is an instance of the function block rdrec defined in the program dictionary.

An equivalent Structured Text implementation of the function block diagram in ***Figure 6.6*** 42 listed below. This code will store the outputs of the function block rdrec in the variables CNF, STATUS, PointNumber, PointQuality, etc…

```
(*                    Code Snippet Begins Here                *)
(*    Ensure dictionary has the following variables defined: *)
(*    Boolean                       CNF                       *)
(*    Boolean                       PointFailed               *)
(*    integer                       STATUS                    *)
(*    Integer                       PointNumber               *)
(*    Integer                       PointQuality              *)
(*    Integer                       IntValue                  *)
(*    Real                          FloatValue                *)
(*    FB instances             rdfld_inst                     *)
rdrec_inst(z_speedcontrol, AIN);
if (rdrec_inst.CNF) then
     CNF := rdrec_inst.CNF;
     STATUS := rdrec_inst.status;

     PointNumber := rdrec_inst.oPN;
```

```
        PointQuality := rdrec_inst.oQlty;

        PointFailed := rdrec_inst.oFail;

        IntValue := rdrec_inst.oRaw;

        FloatValue := rdrec_inst.oEng;

            end_if;
(* Code Ends Here *)
```

5.1.3.2 rdrec_dg

Read attributes for digital points

Description

The rdrec_dg function block reads attributes for digital (binary) points. Return Values not existing for a point type will return 0.

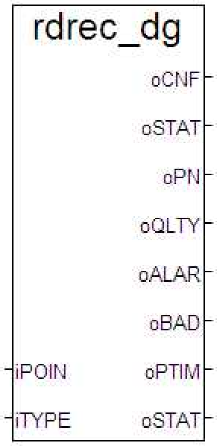


Figure 6.7: rdrec_dg Function Block

Inputs	Type	Description
iPOINT	Integer	RTU Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
iType	Integer	RTU Point Data Type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Commen** DIN / 1 / Digital input p DOUT / 2 / Digital output

Outputs	TYPE	DESCRIPTION
oCnf	Boolean	Confirm valid or invalid status **Possible Values** / **Meaning** TRUE / Confirm Valid Status FALSE

oStatus	Integer	Status of Read Request **Possbile Values** — **Meaning** -1 — Unsuccessful 0 — Success 1 — Point does not exist 2 — Bad Point Type 3 — Unknown attribute for this point 4 — Bad Value for this attribute 5 — Invalid Attribute for this function block 8 — Point is locked 12 — Database is locked 18 — Data Processor Unavailable
oPN	Integer	Point Number
oQlty	Integer	Point Quality
oAlarm	Boolean	Point Failed
oBad	Boolean	ISaGRAF Controlled
oPtime	Timer	Point is Bad
oState	Boolean	A/D Reference Error

IEC61131-3 Structured Text prototypes take on the following form:

Prototype:

```
rdrec_dg_inst (point, type)
complete_confirm :=  rdrec_dg_inst.cnf

return_status := rdrec_dg_inst.status

Point_number := rdrec_dg_inst.PN …….
```

where rdrec_dg_inst is an instance of the function block rdrec_dg defined in the program dictionary.

A sample Structured Text implementation is listed below.

```
(*                    Code Snippet Begins Here                  *)
(*   Ensure dictionary has the following variables defined:     *)
(*   Boolean                      cnf                           *)
(*   integer                      status                        *)
(*   Integer                      point_number                  *)
(*   Integer                      point_quality                 *)
(*   Boolean                      alarm_state                   *)
(*   Boolean                      bad_point                     *)
(*   Integer                      pulse_tme                     *)
(*   Boolean                      point_state                   *)
(*   FB instances             rdrec_dg_inst                     *)
rdrec_dg_inst(z_DIN1, DIN)
if (rdrec_dg_inst.ocnf) then
    cnf := rdrec_dg_inst.ocnf;

    status := rdrec_dg_inst.ostatus;
    point_number :=  rdrec_dg_inst.opn;

    point_quality := rdrec_dg_inst.oqlty;
    alarm_state := rdrec_dg_inst.oalarm;
    bad_point := rdrec_dg_inst.obad;
    pulse_time := rdrec_dg_inst.optime;
    point_state := redrec_dg_inst.ostate;

end_if;
(* Code Ends Here *)
```

5.1.3.3 rdrec_cn

Read attributes for counter points

Description

The rdrec_cn function block reads attributes for counter points. The return value for a point type that does not exist is 0.

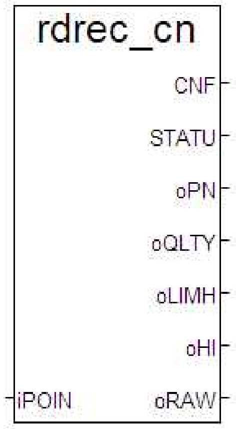


Figure 6.9: rdrec_cn Function Block

INPUTS	TYPE	DESCRIPTION
iPoint	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Confirm valid or invalid status **Possible Values** — **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possbile Values** — **Meaning** -1 — Unsuccessful 0 — Success 1 — Point does not exist 2 — Bad Point Type 3 — Unknown attribute for this point 4 — Bad Value for this attribute 5 — Invalid Attribute for this function block 8 — Point is locked 12 — Database is locked 18 — Data Processor Unavailable
oPN	Integer	Point Number
oQlty	Integer	Point Quality
oLim1h	Real	Counter high limit
oHi	Boolean	Counter limit exceeded
oRaw	Integer	Raw counter value

IEC61131-3 Structured Text prototypes take on the following form:

Prototype:
```
rdrec_cn_inst (point, type)
complete_confirm :=  rdrec_cn_inst.cnf

return_status := rdrec_cn_inst.status

Point_number := rdrec_cn_inst.opn

Point_quality  := rdrec_cn_inst.oQlty
```

where rdrec_cn_inst is an instance of the function block rdrec_cn defined in the program dictionary.

5.1.3.4 rdrec_an

Read attributes for analog (integer) input points

Description

The RDREC_AN function block reads attributes for analogue points. The return value for a point type that does not exist is 0.

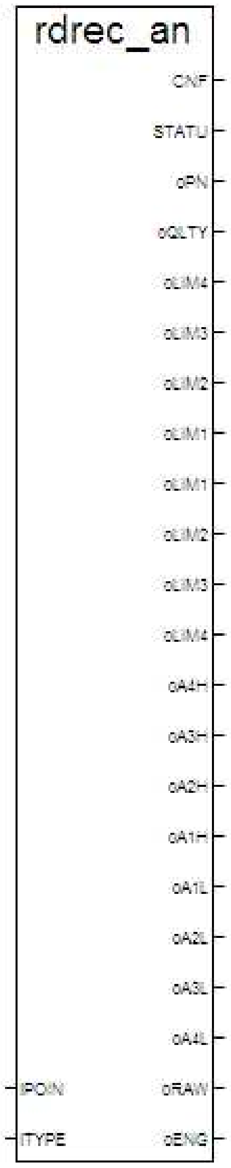


Figure 6.10: rdrec_an Function Block

INPUTS	TYPE	DESCRIPTION

iPOINT	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
iTYPE	Integer	RTU Point data type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Comment** AIN / 3 / Analog input point AOUT / 4 / Analog output point

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Confirm valid or invalid status **Possible Values** / **Meaning** TRUE / Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possbile Values** / **Meaning** -1 / Unknown Return Error 0 / Success 1 / Point does not exist 2 / Bad Point Type 3 / Unknown attribute for this point 4 / Bad Value for this attribute 5 / Invalid Attribute for this function block 8 / Point is locked 12 / Database is locked 18 / Data Processor Unavailable
oPn	Integer	Point Number
oQlty	Integer	Point Quality
oLim4h	Real	Engineering Limit 4H
oLim3h	Real	Engineering Limit 3H
oLim2h	Real	Engineering Limit 2H
oLim1h	Real	Engineering Limit 1H
oLim1l	Real	Engineering Limit 1L
oLim2l	Real	Engineering Limit 2L
oLim3l	Real	Engineering Limit 3L

oLim4l	Real	Engineering Limit 4L
oA4h	Boolean	Alarm Limit Transgress 4H
oA3h	Boolean	Alarm Limit Transgress 3H
oA2h	Boolean	Alarm Limit Transgress 2H
oA1h	Boolean	Alarm Limit Transgress 1H
oA1l	Boolean	Alarm Limit Transgress 1L
oA2l	Boolean	Alarm Limit Transgress 2L
oA3l	Boolean	Alarm Limit Transgress 3L
oA4l	Boolean	Alarm Limit Transgress 4L
oRaw	Integer	Raw value
oEng	Real	Floating point value

An IEC61131-3 Function Block Diagram example of rdrec_an is illustrated in below.

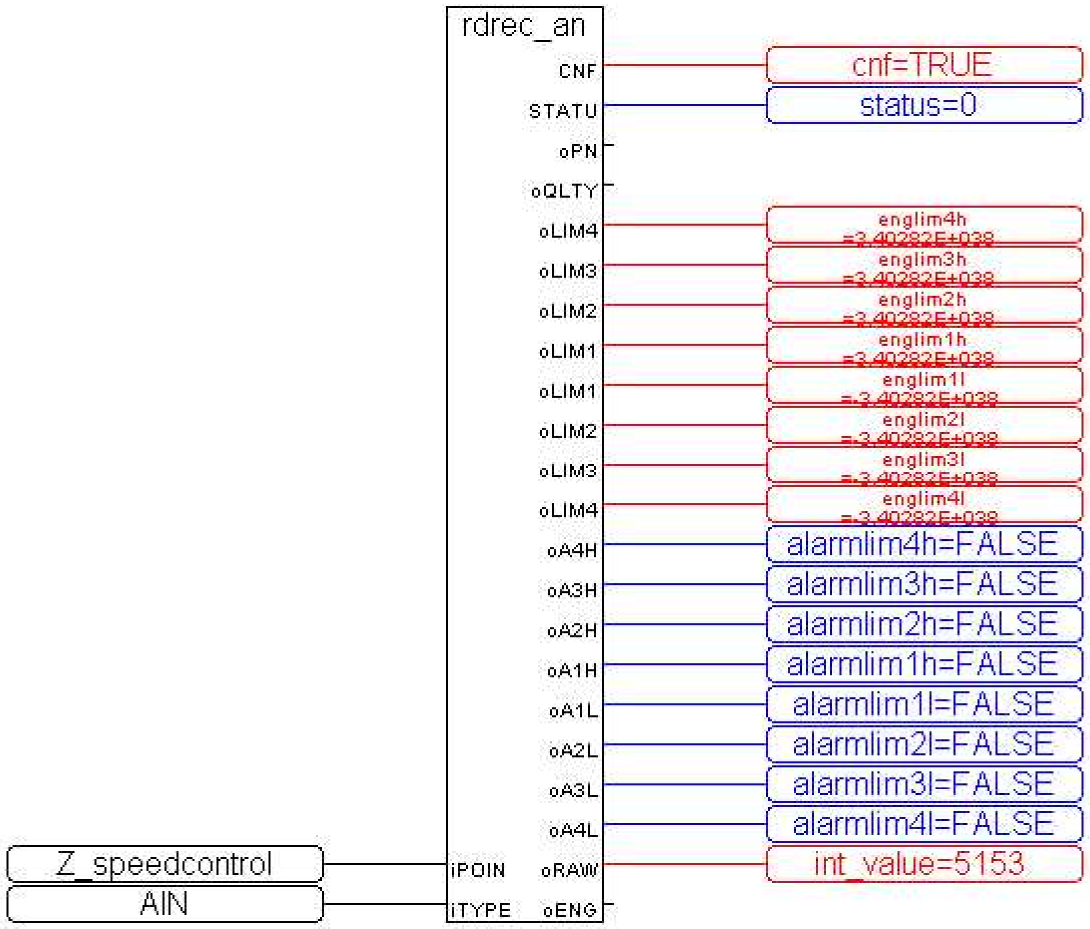


Figure 6.11: rdrec_an Function Block Example

IEC61131-3 Structured Text prototypes take on the following form:

prototype:

```
rdrec_an_inst (point, type)
complete_confirm :=  rdrec_an_inst.cnf

return_status := rdrec_an_inst.status

Point_number := rdrec_an_inst.pn ……
```

where rdrec_an_inst is an instance of the function block rdrec_an defined in the program dictionary.

A Structured equivalent Text sample implementation of the rdrec_an function block in ***Figure 6.10*** 49 is listed below.

```
(*              Code Snippet Begins Here                *)
(*   Ensure dictionary has the following variables      *)
     defined:
(*   Boolean            cnf                             *)
(*   integer            status                          *)
(*   Integer            point_number                    *)
(*   Integer            point_quality                   *)
(*   Real               eng_limit_4H                    *)
(*   Real               eng_limit_1L                    *)
(*   Boolean            alarm_limit_tran_4H             *)
(*   Boolean            alarm_limit_tran_1L             *)
(*   Integer            current_int_value               *)
(*   Real                     current_real_value        *)
rdrec_an_inst(z_speedcontrol, AIN);
if (rdrec_an_inst.cnf) then
     cnf := rdrec_an_inst.cnf;

     status := rdrec_an_inst.status;
     point_number :=  rdrec_an_inst.opn;

     point_quality := rdrec_an_inst.oqlty;
     eng_limit_4H := rdrec_an_inst.olim4h;
     eng_limit_1L := rdrec_an_inst.olim1l;
     alarm_limit_tran_4H := rdrec_an_inst.oa4H;
     alarm_limit_tran_1L := rdrec_an_inst.oa1L;
     current_int_value  := rdrec_an_inst.oraw;
     current_real_value  := rdrec_an_inst.oeng;
end_if;
(* Code Ends Here *)
```

5.1.4 rdstring

Read a string point

Description

Reads a string point or message from the RTU system string map. The return value for a point that does not exist is 0.

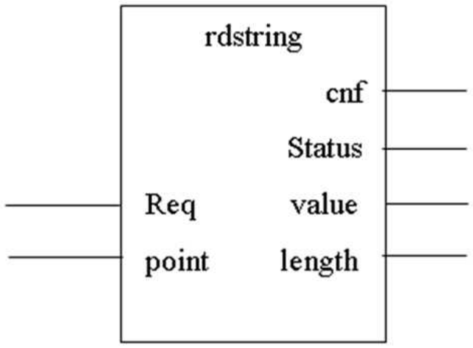


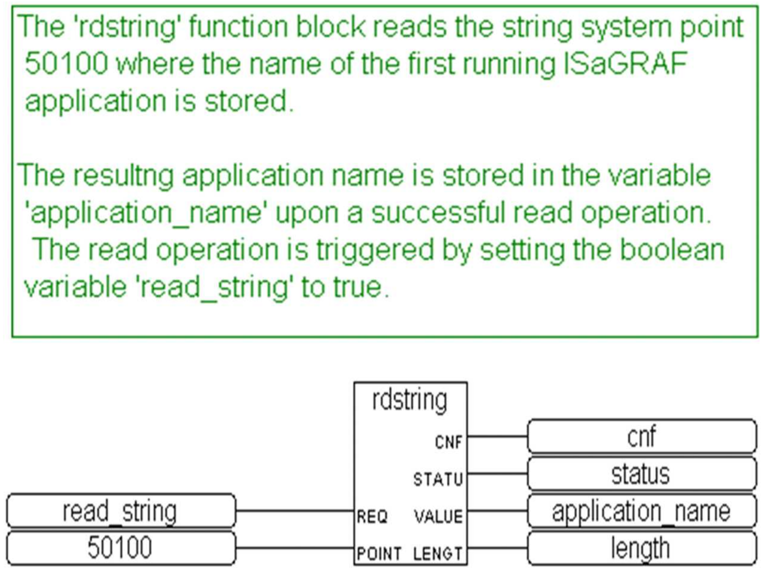
Figure 6.12: rdstring Function Block

INPUTS	TYPE	DESCRIPTION
point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Req	Boolean	Request to read a point **Possible Values** — **Meaning** TRUE — Read point enabled FALSE — Read point function disabled

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Confirm valid or invalid status **Possible Values** — **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possbile Values** — **Meaning** -1 — Unsuccessful

OUTPUTS	TYPE	DESCRIPTION
		0 Success 1 Point does not exist
Value	Msg	Return string message
Length	Integer	The number of characters read into the message variable from the string point's current value

A sample function block implementation of rdstring is illustrated below;


The 'rdstring' function block reads the string system point 50100 where the name of the first running ISaGRAF application is stored.

The resultng application name is stored in the variable 'application_name' upon a successful read operation. The read operation is triggered by setting the boolean variable 'read_string' to true.

Figure 6.13: rdstring Function Block Example

A structured text implementation of rdstring is listed below. In this example, variable ISaGRAF_APP_1 contains the point address of the ISaGRAF application name running on kernel 1. This is the system point number for the first ISaGRAF application name.

```
Read_String := TRUE;
ISaGRAF_APP_1 := 50100;
r_string(Read_String, ISaGRAF_APP_1);
IF(r_string.CNF) THEN
    Read_String := FALSE;
    IF(r_string.STATUS = SUCCESS) THEN
        ReadStringValue := r_string.VALUE;
        valid_string := TRUE;
```

```
          ELSE
                    valid_string := FALSE;
          END_IF;
END_IF;
```

5.1.5 Digital Output Controls

Generate a control on a digital output

Description

Digital output points on the SCADAPack E RTU can be activated for a predetermined length of time. There are two mechanisms for creating pulses on the digital output ports under control of the ISaGRAF application.

- An ISaGRAF application may choose to switch a *Boolean Output* variable on and off with timing through logic

  In this case the ISaGRAF application controls the output point directly. Exact output timing will be dependent on the consistency of the ISaGRAF application scan rate

- The RTU’s ISaGRAF ***rtupulse*** 58, ***rtupuls2*** 62 or ***rtucrob*** 65 function block may be used to activate an output

  In this case the ISaGRAF application instructs the RTU's point processing task to activate the output point, using the processing tasks' timing, rather than the ISaGRAF application timing. In general this will produce more accurate pulse timing.

If the digital output point is on a remote DNP3 device through the SCADAPack E Data Concentrator, then the pulse command is initiated on the remote device using a DNP3 CROB control. It is recommended that the ***rtucrob*** 65 function block be used in this case as it provides the most flexibility in setting CROB parameters for interoperability with the remote DNP3 device.

- ***rtupulse*** 58
- ***rtupuls2*** 62
- ***rtucrob*** 65

5.1.5.1 rtupulse

The function block ‘**rtupulse**’ sends a control message to output points. These may be:

- Control to an output point to the local RTU, turning it on for the configured length of time, then turning it off
- Control using a DNP3 CROB (control request object block - object group 12 variation 1) to an output point mapped to a remote DNP3 device by the SCADAPack E RTU's Data Concentrator facility.

There cannot be an ISaGRAF Boolean Output Board definition for the physical digital output being controlled (i.e. No ISaGRAF RTU output board whose address range corresponds to the address of the output point)

This function block causes the RTU to control local digital outputs in a similar fashion to the DNP3 CROB object. If the point is a remote point controlled by the data concentrator facility of the SCADAPack E RTU, a DNP3 CROB is sent using a Pulse On / NULL command. If other DNP3 CROB controls are required by the remote device, use the ***rtucrob***65 command.

- The function block will execute when the ‘**req**’ input value goes from FALSE to TRUE. ‘**req**’ needs to be set to FALSE before each subsequent pulse execution from this instance of the function block.
- When this function block is executed, the RTU turns ON the output point for the duration of the “**ptime** ” parameter of the function block.

If “**Ptime**” is zero, the output point is turned ON for the time duration configured in the point’s *Output Pulse Time* attribute. This field may initially have been configured through an ISaGRAF application using the setatr_i function block with the attribute set to *Ptm*, or provided as part of the point's configuration (e. g. through SCADAPack E Configurator).

SCADAPack ES physical digital outputs return **CONF** and **STATUS** parameters at the completion of the pulse or pulse train. Other SCADAPack E RTUs return status at the initiation of the pulse or pulse train.

The RTU will not pulse the digital point as instructed by this ISaGRAF function block if the digital point was not found, was read-only (eg. physical input point), if a remote control interlock point was active for this point, or if a pulse is currently being executed on this point.

This function block causes the RTU to control digital outputs in a similar fashion to the DNP3 CROB object. The main differences are:

- The pulse duration time is specified by the ISaGRAF application, or uses the point's *Output Pulse Time* attribute if ISaGRAF specifies a time of zero.
- Only a single pulse is generated. Once initiated by ISaGRAF, the digital output pulse is controlled by the RTU’s Data Processor Task and is independent of the ISaGRAF application cycle time.

TIP:	The ISaGRAF User Application may read the output point's "Output Pulse Time" attribute using the "rdfld_i" function block and use this to derive or range check the "rtupulse" function block's "ptime" parameter. For example, on a valve, the minimum useful pulse width is likely to be that specified in the "Output Pulse Time" point attributes.

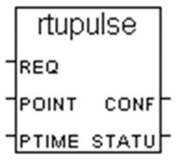


Figure 6.16: rtupulse Function Block

INPUTS	TYPE	DESCRIPTION
req	Boolean	Request to pulse a point **Possible Values** — **Meaning** TRUE — pulse initiated on transition to TRUE FALSE — pulse point function disabled
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Ptime	Timer	Pulse duration (ms timer format)

OUTPUTS	TYPE	DESCRIPTION
conf	Boolean	Confirms that Status is ready **Possible Values** — **Meaning** TRUE — Pulse has completed or pulse request is unsuccessful. TRUE indicates the status output is ready to read. FALSE — Pulse has not been completed or REQ = false.
Status	Integer	Status of Request **Possible Values** — **Meaning** -1 — Unsuccessful 0 — Output pulse has completed executing. 1 — Pulse not yet completed, or Digital Point Not Found (This value changes to 1 as the pulse command is initiated and returns to 0 after the pulse has been generated. If this value stays at 1, the point was not found). 2 — Unsuccessful: remote interlock active 3 — Unsuccessful: pulse already executing Status values of -1, 0 or 1 may be returned when controlling points on a remote device. Other status codes may be returned if controlling an output point on the local RTU.

In the sample function block diagram below, digital output point 5 has been programmed to pulse ON for 5 seconds when DIN1 transitions to TRUE.

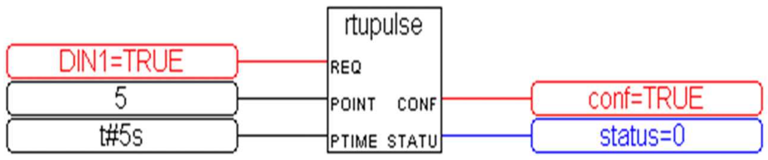


Figure 6.17: rtupulse Function Block Example

In the Structured text sample below, digital output point 6 has been programmed to pulse ON for 10 seconds when SW2 transitions to TRUE. The variables SW2, conf and status have to be defined in the program dictionary.

```
(* Code starts here *)

rtupulse_inst(SW2, 6, t#10s);
```

```
if (rtupulse_inst.conf) then
    conf := rtupulse_inst.conf;
    status := rtupulse_inst.status;
end_if;
```

5.1.5.2 rtupuls2

The function block ‘**rtupulse**’ sends a control message to output points. These may be:

- Control to an output point to the local RTU, pulsing it on and off for the configured times, for the configured number of cycles
- Control using a DNP3 CROB (control request object block - object group 12 variation 1) to an output point mapped to a remote DNP3 device by the SCADAPack E RTU's Data Concentrator facility.

There cannot be an ISaGRAF Boolean Output Board definition for the physical digital output being controlled (i.e. No ISaGRAF RTU output board whose address range corresponds to the address of the output point)

This function block causes the RTU to control local digital outputs in a similar fashion to the DNP3 CROB object. If the point is a remote point controlled by the data concentrator facility of the SCADAPack E RTU, a DNP3 CROB is sent using a Pulse On / NULL command. If other DNP3 CROB controls are required by the remote device, use the ***rtucrob*** 65 command.

The pulse characteristics are as follows:

- The **pulse on** duration is specified by the ***Ptime*** parameter on the function block, or is preset in the output point's *Output Pulse Time* attribute if ***Ptime*** has a value of zero. The *Output Pulse Time* attribute may have been initially configured through an ISaGRAF application using the setatr_i function block with the attribute set to Ptm, or provided as part of the point's configuration (e.g. through SCADAPack E Configurator).
- The **pulse off** duration is specified by the ***offtime*** parameter. If an offtime value of 0 is set 10 ms is used as the default pulse off duration.
- The number of pulses generated is determined by the ***count*** variable which is limited to a maximum value of 255. Once initiated by ISaGRAF, the digital output pulse is controlled by the RTU’s Data Processor Task or forwarded to the remote device for processing, and is independent of the ISaGRAF application cycle time.

The function block will execute when the ***req*** input value transitions from FALSE to TRUE. ***req*** needs to be set to FALSE before each subsequent pulse execution from this instance of the function block.

The function blocks returns ***CONF*** and ***STATUS*** parameters as follows:

- at the completion of executing the pulse train for SCADAPack ES local points
- at initiating the pulse train for local points on other controllers
- when a DNP3 CROB request is sent to a remote device (for control of a remote point)

The RTU will not pulse the digital point or send a control as instructed by this ISaGRAF function block if the digital point was not found, was read-only (eg. physical input point), if a remote control interlock point was active for this point, or if a pulse is currently being executed on this point.

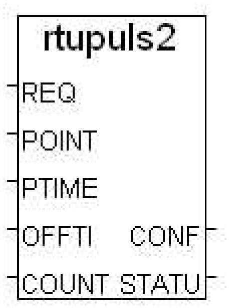


Figure 6.18: rtupuls2 Function Block

INPUTS	TYPE	DESCRIPTION
req	Boolean	Request to pulse an output point **Possible Values** – **Meaning** TRUE – pulse initiated on transition to TRUE FALSE – pulse point function disabled
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Ptime	Timer	Pulse On duration (ms timer format)
offtime	Timer	Pulse Off duration (ms timer format)
Count	Integer	Number of pulse cycles in the pulse train

OUTPUTS	TYPE	DESCRIPTION
conf	Boolean	Confirm that Status is ready **Possible Values** / **Meaning** TRUE: Pulse has completed or pulse request is unsuccessful. TRUE indicates the Status output is ready to read. FALSE: Pulse has not been completed or REQ = false.
Status	Integer	Status of Request **Possbile Values** / **Meaning** -1: Unsuccessful 0: Output pulse has completed executing. 1: Pulse not yet completed, or Digital Point Not Found (This value changes to 1 as the pulse command is initiated and returns to 0 after the pulse has been generated. If this value stays at 1, the point was not found). 2: Unsuccessful: remote interlock active 3: Unsuccessful: pulse already executing Status values of -1, 0, or 1 may be returned when controlling points on a remote device. Other status codes may be returned if controlling an output point on the local RTU.

prototype: RTUPULS2(REQ, POINT, PTIME, OFFTIME, COUNT);

5.1.5.3 rtucrob

The function block ‘**rtucrob**’ sends a control message to output points. These may be:

- Control to an output point to the local RTU
- Control using a DNP3 CROB (control request object block - object group 12 variation 1) to an output point mapped to a remote DNP3 device by the SCADAPack E RTU's Data Concentrator facility.

There cannot be an ISaGRAF Boolean Output Board definition for the physical digital output being controlled (i.e. No ISaGRAF RTU output board whose address range corresponds to the address of the output point)

The function block will execute when the ***req*** input value transitions from FALSE to TRUE. ***req*** needs to be set to FALSE before each subsequent control is issued from this instance of the function block.

The RTU will not send a control as instructed by this ISaGRAF function block if the digital point was not found, was read-only (eg. physical input point), if a remote control interlock point was active for this point, or if a pulse is currently being executed on this point.

- The **onTime** duration is specified by the ISaGRAF application, or uses the point's *Output Pulse Time* attribute if ISaGRAF specifies a time of zero.
- The **offTime** duration is specified by the ISaGRAF application. If an offtime value of 0 is set 10 ms is used as the default pulse off duration.
- The number of pulses generated is determined by the **count** variable which is limited to a maximum value of 255. Once initiated by ISaGRAF, the digital output pulse is controlled by the RTU’s Data Processor Task and is independent of the ISaGRAF application cycle time.
- The **funcCode** parameter specifies which DNP3 function code is used to send the CROB control code (see table below)
- The **ctrlCode** parameter specifies which CROB control code is used (see table below)

The function blocks returns the ***CONF*** and ***STATUS*** parameters if the control was not successfully sent or when the DNP3 CROB request is sent to the remote device.

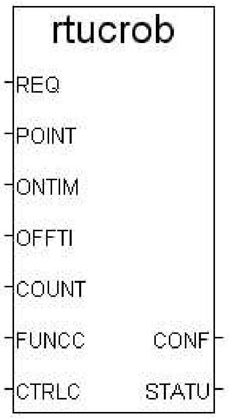


Figure 6.67: rtucrob Function Block

INPUTS	TYPE	DESCRIPTION
req	Boolean	Request to pulse an output point **Possible Values** — **Meaning** TRUE — pulse initiated on transition to TRUE FALSE — pulse point function disabled
point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
onTime	Timer	Pulse On duration (ms timer format)
offTime	Timer	Pulse Off duration (ms timer format)
count	Integer	Number of pulse cycles in the pulse train
funcCode	Integer	The function code to send if the point is on a remote DNP3 device. Use one of the following defines: CROB_DirOp — Direct Operate CROB_SelOp — Select Before Operate CROB_DONA — Direct Operate, No Acknowledgement
ctrlCode	Integer	The control code to send if the point is on a remote DNP3 device. Use one of the following defines: CROB_PulseOn CROB_PulseOff CROB_LatchOn CROB_LatchOff CROB_Trip CROB_Close CROB_Clear

OUTPUTS	TYPE	DESCRIPTION
conf	Boolean	Confirm that Status is ready **Possible Values** — **Meaning** TRUE — Pulse has completed or pulse request is unsuccessful. TRUE indicates the Status output is ready to read. FALSE — Pulse has not been completed or REQ = false.
Status	Integer	Status of Request. This should only be read if the *conf* output is TRUE. **Possible Values** — **Meaning** -1 — Unsuccessful 0 — Output pulse has completed executing. 1 — Pulse not yet completed, or Digital Point Not Found (This value changes to 1 as the pulse command is initiated and returns to 0 after the pulse has been generated. If this value stays at 1, the point was not found). 2 — Unsuccessful: remote interlock active 3 — Unsuccessful: pulse already executing 4 — Invalid *funcCode* input. Status values of -1, 0,1 or 4 may be returned when controlling points on a remote device. Other status codes may be returned if controlling an output point on the local RTU.

prototype: RTUCROB(REQ, POINT, ONTIME, OFFTIME, COUNT, FUNCCODE, CTRLCODE);

5.1.6 setatr

Set database point attributes

- *setatr_i* 70
- *setatr_r* 74

5.1.6.1 setatr_i

Set integer point attributes

Description

RTU point database attributes represented by integer values may be set by an ISaGRAF application using ISaGRAF function block “setatr_i”.

The tables below describe the inputs and outputs of the setatr_i function block. Each time the function block is called, the RTU updates the specified point database field for the specified RTU point number and point type from the value of an ISaGRAF variable.

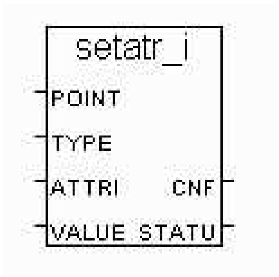


Figure 6.19: setatr_i Function Block

INPUTS	TYPE	DESCRIPTION
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Type	Integer	RTU point data type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Comment** DIN / 1 / Digital input point DOUT / 2 / Digital output point AIN / 3 / Analog input point AOUT / 4 / Analog output point CIN / 5 / Counter input point
Attrib	Integer	Desired point attribute (property) argument can be an ISaGRAF reserved keyword or an integer value corresponding to the attribute – see ***Table 6.6*** 71.
Value	Integer	Desired field value

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Confirm status ready to write **Possible Values** — **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possbile Values** — **Meaning** -1 — Unknown Return Error 0 — Success 1 — Information not found 2 — Bad Point Type 3 — Unknown attribute for this point 4 — Bad value for this attribute 5 — Invalid attribute for this point

Structured text prototypes take on the following format:

```
SETATR_I _INST(POINT, TYPE, ATTRIB,VALUE)

complete_confirm :=  SETATR_I_NST.CNF

return_status := SETATR_I_INST.STATUS
```

Table 6.6: Valid Attributes (attrib) for setatr_i Function Block

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description	Point Type where this field applies
PClassMA	66	Point Data Class (All Masters)	All
PClassM1	67	Point Data Class Master 1	All
PClassM2	68	Point Data Class Master 2	All
PClassM3	69	Point Data Class Master 3	All
PObj	3	DNP3 Static Object Type	All
IlockPN	4	Remote Interlock Point	Output points only
IlockTm	5	Interlock Alarm timeout	Output points only
Bad	6	Point is bad	All
AInh	7	Alarm Inhibit 0 = alarm enable 1 = alarm disable	All
TInh	8	Trend Inhibit 0 = trend enable 1 = trend disable	All

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description	Point Type where this field applies
PrId	9	Profile Id	All
ATm	10	Alarm Time Deadband	All
Inv	11	Invert Point State	Digital points only
AState	12	Alarm Active State	Digital points only
AClrTm	14	Alarm Clear Time Deadband	all
PTm	15	Output Pulse Time	Digital output only
DebTm	16	Debounce Time	Digital input only
RoRPN	17	Rate Of Rise Point Number	Analog
RoFPN	18	Rate Of Fall Point Number	Analog
NCPN	19	No Change Point Number	Analog
CexPN	20	Counter Exceeded Point Number	Counter
CntRZ	53	Counter Reset to Zero on power up 0 = retain previous value 1 = Reset to zero	Counter
Rmin	29	Raw Min	Analog
Rmax	30	Raw Max	Analog
RoCTm	38	Rate Of Change Time	Analog
NCTm	39	No Change Time	Analog
Ev4H	49	Limit Event Enable 4h 0 = event disabled 1 = event enabled	Analog
Ev3H	48	Limit Event Enable 3h 0 = event disabled 1 = event enabled	Analog
Ev2H	47	Limit Event Enable 2h 0 = event disabled 1 = event enabled	Analog
Ev1H	46	Limit Event Enable 1h 0 = event disabled 1 = event enabled	Analog
Ev1L	45	Limit Event Enable 1l 0 = event disabled 1 = event enabled	Analog
Ev2L	44	Limit Event Enable 2l 0 = event disabled 1 = event enabled	Analog
Ev3L	43	Limit Event Enable 3l	Analog

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description	Point Type where this field applies
		0 = event disabled 1 = event enabled	
Ev4L	42	Limit Event Enable 4l 0 = event disabled 1 = event enabled	Analog
LimHi	51	Counter High Limit	Counter
CntDev	52	Counter Change Deviation	Counter
AclrVal	40	Alarm Clear Value Deadband	Analog
LimZero	41	Zero Threshold Limit	Analog
LimOR	33	Over Range Limit	Analog
LimUR	34	Under Range Limit	Analog
EvDev	50	Event Deviation	Analog
EvDevUnsol	65	Event Deviation Unsolicited enable	Analog
EvDevTp	97	Event Deviation Type	Analog

The PClass attribute (value 2) is deprecated. It is replaced by the PClassM1 attribute. The attribute was changed in firmware 7.39 with the addition of multi-master functionality, but was not changed in ISaGRAF until version 7.71.

5.1.6.2 setatr_r

Set real point attributes

Description

RTU point database attributes represented by real (floating point) values may be set by an ISaGRAF application using ISaGRAF function block “setatr_r”.

The table below describes the inputs and outputs of the setatr_r function block. Each time the function block is called, the RTU updates the specified point database field for the specified RTU

point number and point type from the value of an ISaGRAF variable.

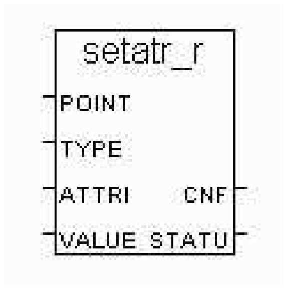


Figure 6.20 setatr_r Function Block

INPUTS	TYPE	DESCRIPTION
Point	Integer	RTU Point Address can be a numeric Point Address, a variable containing the Point Address or a define referencing a point address. See ISaGRAF 3 Pre-Processor for automatic generation of point address defines
Type	Integer	RTU point data type argument can be one of the ISaGRAF reserved keywords below or an integer value corresponding to the data type. **ISaGRAF Keyword** / **Equivalent Numeric Value** / **Comment** DIN / 1 / Digital input point DOUT / 2 / Digital output point AIN / 3 / Analog input point AOUT / 4 / Analog output point CIN / 5 / Counter input point
Attrib	Integer	Desired point attribute (property) argument can be an ISaGRAF reserved keyword or an integer value corresponding to the

		attribute – see ***Table 6.7*** 76.
Value	Real	Desired field value

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Confirm status ready to write **Possible Values** / **Meaning** TRUE — Confirm Valid Status FALSE
Status	Integer	Status of Read Request **Possible Values** / **Meaning** -1 — Unknown Return Error 0 — Success 1 — Information not found 2 — Bad Point Type 3 — Unknown attribute for this point 4 — Bad value for this attribute 5 — Invalid attribute for this point

Table 6.7: Valid Attribute (attrib) Values for setatr_r

Attribute (ISaGRAF Keyword)	Equivalent Numeric Value	Description	Point Type where this field applies
Emin	31	Engineering Min	Analog
Emax	32	Engineering Max	Analog
Lim4H	28	Engineering Limit 4H	Analog
Lim3H	27	Engineering Limit 3H	Analog
Lim2H	26	Engineering Limit 2H	Analog
Lim1H	25	Engineering Limit 1H	Analog
Lim1L	24	Engineering Limit 1L	Analog
Lim2L	23	Engineering Limit 2L	Analog
Lim3L	22	Engineering Limit 3L	Analog
Lim4L	21	Engineering Limit 4L	Analog
LimRise	35	Rate Of Rise	Analog
LimFall	36	Rate Of Fall	Analog
LimNC	37	No Change	Analog
EvDev	50	Event Deviation	Analog

5.2 Real Time Clock Function Blocks

The following function blocks provide an ISaGRAF application access to the SCADAPack E RTU real time clock.

In addition to these functions, the SCADAPack E RTU supports ISaGRAF's standard "day_time" function returning the RTU clock in several string formats.

For more information on this standard function, see the *ISaGRAF Workbench Language Reference* manual. The "day_time" function returns the local RTU time, adjusted for Summer Time (see TIMEDATE description of the time-zone modifier for more information).

- ***os_time & loc_time*** 78
- ***timedate*** 79

5.2.1 os_time & loc_time

os_time

Return RTU time in seconds since 00:00:00, 1st Jan 1970

Description

This function block returns the current time (UTC) as used by the RTU operating system. The return parameter is an integer analog variable, in 'Seconds since 00:00:00, 1st Jan 1970. This function block does not adjust for UTC offsets or daylight saving time. I.e. it returns standard RTU time from the RTU's real time clock.

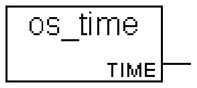


Figure 6.21: os_time Function Block

OUTPUT	TYPE	DESCRIPTION
Time	Intege	RTU operating system time (UTC) in seconds since 00:00:00, 1970

loc_time

Return RTU local time since midnight, in milliseconds

Description

This function block returns the RTU local time. If the RTU's real time clock is operating in UTC mode, the RTU's *LOCAL TIME OFFSET FROM UTC* system point is applied prior to returning the data. In addition, the RTU examines a system point called *TIME ZONE MODIFIER* – binary system point 50302, and adjusts for summer time by adding 1 hour (if the point is set) prior to presenting the time to the user. The return parameter is a timer variable in "ms since midnight" ISaGRAF timer format (e.g.. t#23h59m59s999ms).

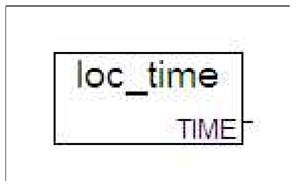


Figure 6.22: loc_time Function Block

OUTPUTS	TYPE	DESCRIPTION
Time	Timer	RTU local time in milliseconds since midnight

5.2.2 timedate

Return RTU local time and date

Description

This function block returns the RTU local time and date. Return parameters are of type integer. If the RTU's real time clock is operating in UTC mode, the RTU's *LOCAL TIME OFFSET FROM UTC* system point is applied prior to returning the data. In addition, the RTU examines a system point called *TIME ZONE MODIFIER* – binary system point 50302, and adjusts for summer time by adding 1 hour (if the point is set) prior to presenting the time to the user.

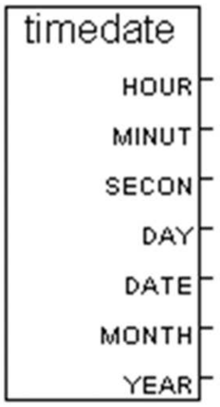


Figure 6.23: timedate Function Block

OUTPUTS	TYPE	DESCRIPTION
Hour	Integer	Current hour in 24 hour format (0-23)
Minute	Integer	Current minute (0-59)
Second	Integer	Current second (0-59)
Day	Integer	Current day (1-7 where 1 = Sunday)
Date	Integer	Current date (1-31)
Month	Integer	Current month (1-12)
Year	Integer	Current year (2000-2099)

5.3 DNP3 Peer Communication Function Blocks

The following section details ISaGRAF function blocks that interface with the SCADAPack E RTU Peer-to-Peer communication facilities. Communication between RTU devices uses DNP3 protocol. This section details function blocks for transfer of simple data types for ISaGRAF variables and DNP3 points.

Section ***DNP3 Queued Communication Function Blocks***100 details queued communication function blocks for RTU database point data between RTU devices.

Each DNP3 Communication Function Block described in this section is implemented for one of two broad communication types:

RD	data read
WR (SET)	data write

These operate on three types of DNP objects and corresponding ISaGRAF variable types:

BIN	binary objects	(corresponding to ISaGRAF Boolean variables)
ANA	analog objects (signed integers)	(corresponding to ISaGRAF Integer variables)
FLT	floating point objects	(corresponding to ISaGRAF Real variables)

A number of function blocks are defined, each with a fixed number of point parameters for reading or writing that number of points. In this documentation, generic references "xx" are made referring to a number of points on a particular function block. For example RD16ANA reads 16 analog objects. Where the number of points read or written is desired to be flexible, or a different number than that available with these fixed function blocks, use ***DNP3 Queued Communication Function Blocks***100.

The function blocks perform data transfer in one of two ways, depending on the value of the ObjectType parameter (see below).

- Local RTU data access (current point state or value in this RTU)
- DNP data communications with Peer RTU node

The RTU processing of user requests from a DNP3 ISaGRAF communication function block is limited to around 20 simultaneous requests. It is recommended that no more than this number of requests from an ISaGRAF application is generated within a single ISaGRAF scan. This allows the DNP3 driver to process the queued ISaGRAF requests.

Decreasing the rate of the DNP3 ISaGRAF communication block requests to fit these constraints should have little affect on the performance of the requested communications, particularly communication with peer DNP3 devices. The SCADAPack E RTU will process only a single outstanding communication request per DNP3 RTU port. Other DNP3 requests made at around the same time are processed in order, when each previous request is completed on that channel.

The DNP communication function blocks take the following general parameters:

Calling (Input) Parameters

REQ (boo)	Data transfer communication request is initiated on the rising edge of this input. This user should keep this input asserted until the CNF output parameter is asserted (see below)
DNPnode (ana)	DNP peer node with which communication occurs (only applicable for DNP peer communications. Set value to *0* when doing Local RTU data access)
ObjectType (ana)	DNP3 data object requested from peer RTU node, or *Local_RTU_Data* for access to RTU data
Index (ana)	Data index of first DNP data object accessed by this block
DT (timer)	Transaction time-out time in ISaGRAF timer format when communicating with peer DNP node (not applicable when doing Local RTU data access. Should be set to *0*)
SDx	Send data parameters (only present on Write function blocks) transferred by this function block when the REQ line is asserted (rising edge). The number of parameters (and hence the number of data objects transferred) depends on the number fixed for the function block (e.g. WR8ANA has parameters SD0..SD7 and writes 8 analog objects)

Return (Output) Parameters

CNF (boo)	Data transfer completion confirm: asserted by the function block to indicate completion of the request
RDY (boo)	Data ready asserted by the function block in conjunction with CNF to indicate successful completion of the transfer operation. RDx parameters (if any) will be valid at the time of the rising edge of RDY. If CNF is activated and RDY is not activated, the data transfer was unsuccessful
STATUS (ana)	Data transfer status: when CNF is active and RDY inactive this output parameter indicates a status code for the unsuccessful data transfer. Status = 255 while there is an outstanding DNP request. See **Status Codes** 99 for more information.
RDx	Read data parameters (only present of Read function blocks), are valid when the RDY parameter is active. The quantity of RDx parameters depends on the function block type (e.g. RD16BIN has Boolean parameters RD0…RD15 when 16 data objects are read)

The following tables describe the allowed Object Types for each of the communication function blocks.

Schneider Electric '**common.eqv**' file contains definitions for these parameters.

Table 6.8: Read Function Block Object Types

Function Block	**RDxxBIN**	**RDxxANA**	**RDxxFLT**
Object Types Supported	Local_RTU_Data	Local_RTU_Data	Local_RTU_Data
	BinaryInput BinInput_Status BinOutput_Stat	AnalogIn_32 AnalogIn_16 AnaIn_32_NoFlag AnaIn_16_NoFlag	AnalogIn_Float AnaOutFloat_Stat

		AnaOut_32_Stat AnaOut_16_Stat	
		BinCounter_32 BinCounter_16 BinCtr_32_NoFlag BinCtr_16_NoFlag	
		FrozCounter_32 FrozCounter_16 FrzCtr_32_NoFlag FrzCtr_16_NoFlag	
ISaGRAF Variable Type Received	BOOLEAN	INTEGER	REAL

When Local_RTU_Data is used as the object type, no communication with a peer device occurs, rather the local RTU database is read. Other object types generate peer communication.

Table 6.9: Write Function Block Object Types

Function Block	WRxxBIN	WRxxANA	WRxxFLT
Object Types Supported	Local_RTU_Data	Local_RTU_Data	Local_RTU_Data
	CROB_DirOp CROB_SelOp CROB_DONA	WriteTimeAndDate WriteLANTime	AnOutFloat_DirOp AnOutFloat_SelOp AnOutFloat_DONA
		AnaOut_32_DirOp AnaOut_32_SelOp AnaOut_32_DONA AnaOut_16_DirOp AnaOut_16_SelOp AnaOut_16_DONA	
ISaGRAF Variable Type Transmitted	BOOLEAN	INTEGER	REAL

When Local_RTU_Data is used as the object type, no communication with a peer device occurs, rather the local RTU database is written. Other object types generate peer communication.

5.3.1 rdxxbin

Read DNP3 binary points from the local or peer RTU address map

Description

This series of function blocks reads current value data from local RTU DNP binary points, or generates a DNP3 read request to a peer node for DNP3 binary objects. *xx* in the function block name refers to the number of objects to read and can be either 1, 4, 8 or 16. The valid DNP object indexes that can be read from a peer DNP device are dependent on the peer device. The SCADAPack E RTU generates DNP3 start/stop range qualifiers (00 & 01) in requests to peer devices. Consult the DNP3 device manufacturer's device profile for more information on support for these qualifiers and point ranges.

Peer Read function blocks perform Application Layer retries as configured in the SCADAPack E Configurator 'Appl. Layer Attempts' field (default = 2). This means that for a single trigger of this function block, subsequent attempts could be made if the requests are unsuccessful. The output parameters (CFN, RDY, and Status) will only be updated with the timeout status after all Application Layer Attempts have been performed (i.e. attempts x DT (timeout)).

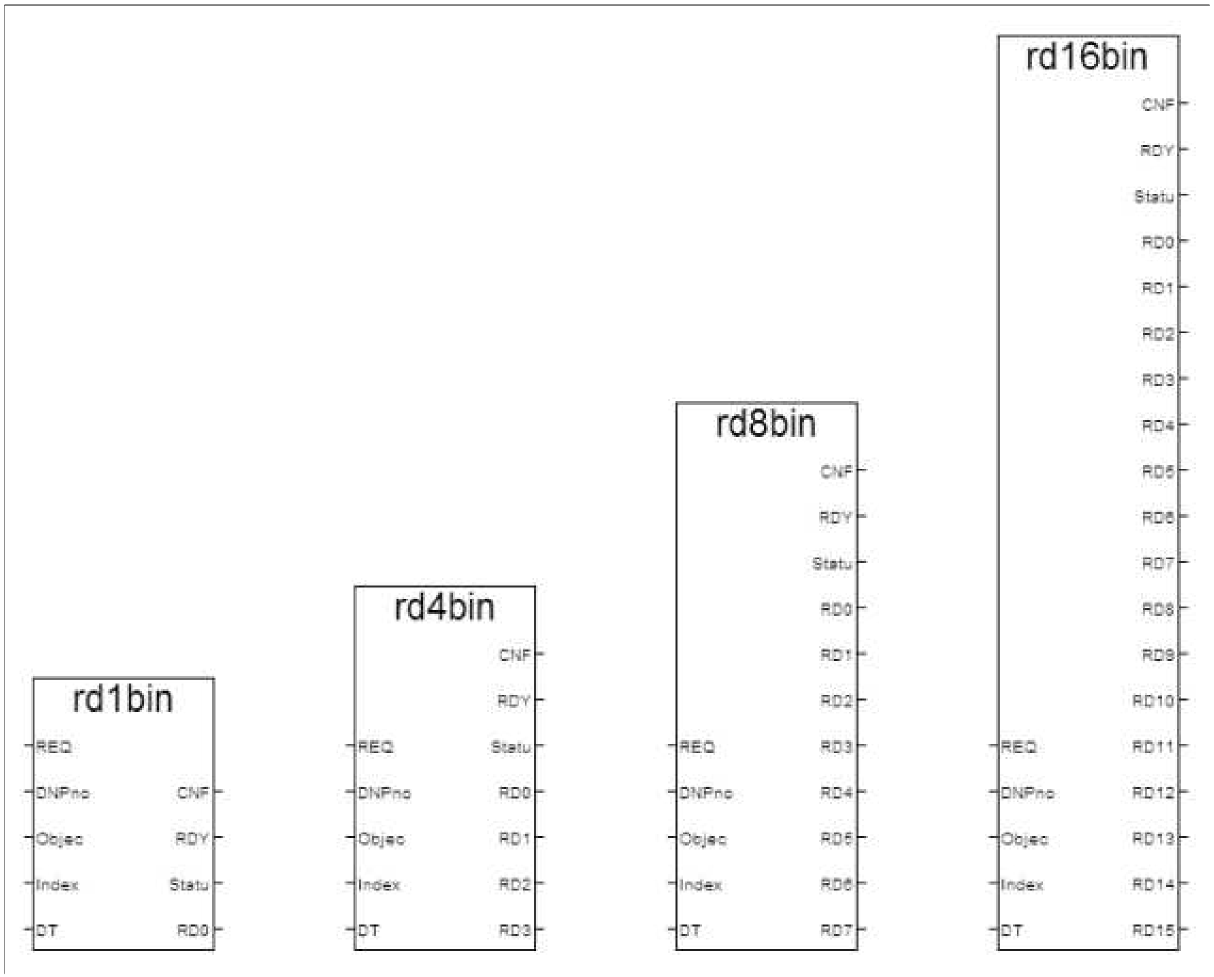


Figure 6.24: rdxxbin Function Blocks

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge.
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to read from peer RTU. For peer RTU read, the following values are valid for this function block: BinaryInput BinInput_Status BinOutput_Sta
Index	Integer	Starting index of DNP data object to read (consecutive data objects will be read starting with this one).
dt	Integer	Transaction time-out (peer RTU request only).

OUTPUTS	TYPE	DESCRIPTION

Cnf	Boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Integer	Data ready
		TRUE => Data Ready
		FALSE => Data not ready

Status	Integer	When CNF is TRUE and RDY is FALSE, this indicates a table data access problem (status code) Status = 0 indicates a successful read Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes
RDx	Boolean	Binary data outputs are valid when DNF and RDY are TRUE. Quantity of RDi parameters depends on the function block.

5.3.2 rdxxana

Read DNP3 analog points from the local or peer RTU address map

Description

This series of function blocks reads current integer value data from local RTU DNP analog points, or generates a DNP3 read request to a peer node for DNP3 integer analog objects. xx in the function block name refers to the number of objects to read. The valid DNP object indexes that can be read from a peer DNP device are dependent on the peer device. The RTU generates DNP3 start/stop range qualifiers (00 & 01) in requests to peer devices. Consult the DNP3 device manufacturer's device profile for more information.

Peer Read function blocks perform Application Layer retries as configured in the SCADAPack E Configurator 'Appl. Layer Attempts' field (default = 2). This means that for a single trigger of this function block, subsequent attempts could be made if the requests are unsuccessful. The output parameters (CFN, RDY, and Status) will only be updated with failure status after all Application Layer Attempts have been performed (i.e. attempts x DT (timeout)).

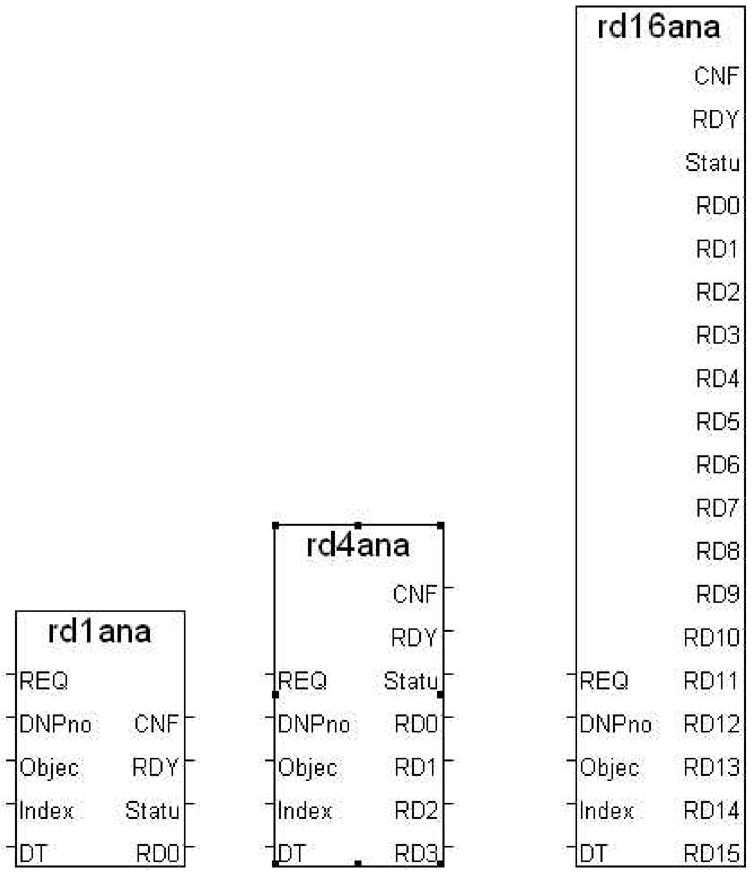


Figure 6.25: rdxxana Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge.
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to read from peer RTU. For peer RTU read, the following values are valid for this function block: AnalogIn_32 BinCounter_16 AnalogIn_16 BinCtr_32_NoFlag AnaIn_32_NoFlag BinCtr_16_NoFlag AnaIn_16_NoFlag FrozCounter_32 AnaOut_32_Stat FrozCounter_16 AnaOut_16_Stat FrzCtr_32_NoFlag FrzCtr_16_NoFlag BinCounter_32
Index	Integer	Starting index of DNP data object to read (consecutive data objects will be read starting with this one).
dt	Integer	Transaction time-out (peer RTU request only).

OUTPUTS	TYPE	DESCRIPTION

Cnf	Boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Integer	Data ready
		TRUE => Data Ready
		FALSE => Data Not Ready

Status	Integer	When CNF is TRUE and RDY is FALSE, this indicates a table data access problem (status code) Status = 0 indicates a successful read Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes
RDx	Integer	Analog data. Outputs are valid when CNF and RDY are TRUE. Quantity of RDx parameters depends on the function block.

Reads may be performed using 16-bit or 32-bit analog objects.

5.3.3 rdxxflt

Read DNP3 floating points from the local or peer RTU address map

Description

This series of function blocks reads current floating point value data from local SCADAPack E RTU analog points, or generates a DNP3 read request to a peer node for DNP3 floating point analog objects. xx in the function block name refers to the number of objects to read. The valid DNP object indexes that can be read from a peer DNP device are dependent on the peer device. The RTU generates DNP3 start/ stop range qualifiers (00 & 01) in requests to peer devices. Consult the DNP3 device manufacturer's device profile for more information.

Peer Read function blocks perform Application Layer retries as configured in the SCADAPack E Configurator 'Appl. Layer Attempts' field (default = 2). This means that for a single trigger of this function block, subsequent attempts could be made if the requests are unsuccessful. The output parameters (CFN, RDY, and Status) will only be updated with failure status after all Application Layer Attempts have been performed (i.e. attempts x DT (timeout)).

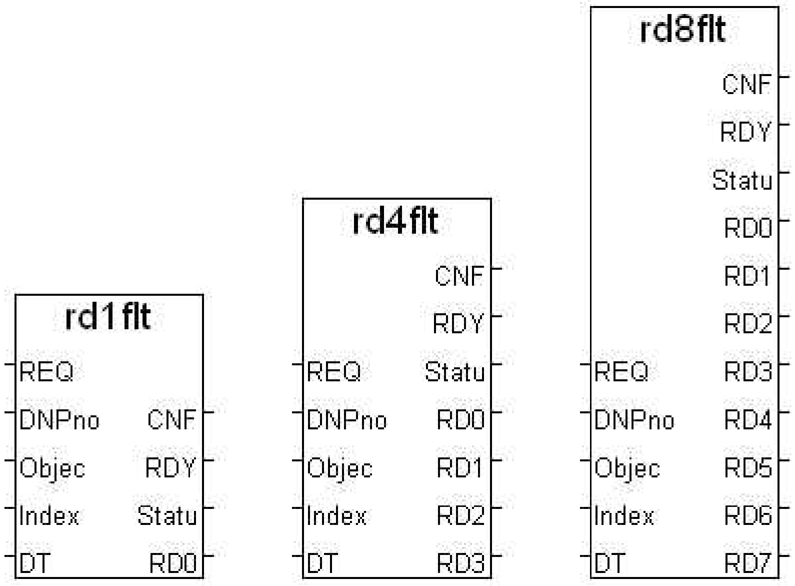


Figure 6.26: rdxxflt Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to read from peer RTU. For peer RTU read requests, the following values are valid for this function block: AnalogIn_Float AnaOutFloat_Stat
Index	Integer	Starting index of DNP data object to read (consecutive data objects will be read starting with this one)
dt	Integer	Transaction time-out (peer RTU request only.

OUTPUTS	TYPE	DESCRIPTION

Cnf	boolea	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Integer	Data ready
		TRUE => Data Ready
		FALSE => Data Not Ready

Status	Integer	When CNF is TRUE and RDY is FALSE this indicates a table data access problem (status code) Status = 0 indicates a successful read Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes
RDx	Integer	Analog data outputs are valid when CNF and RDY are TRUE. Quantity of RDx parameters depends on the function block.

5.3.4 wrxxbin

Write (Control) DNP3 binary points to local or peer RTU address space

Description

This series of function blocks writes current value ISaGRAF boolean data into local SCADAPack E RTU DNP binary points, or generates a DNP3 operate or write request to a peer node for DNP3 binary objects. xx in the function block name refers to the number of objects to send (fixed combinations available). The valid DNP object indexes that can be written to a peer DNP device are dependent on the peer device. For other DNP3 devices consult the device manufacturer’s documentation.

No Application Layer retries are performed on peer write function blocks. If the user requires retries then either configure Data Link retries (with 'Always' mode), or implement the retires in the ISaGRAF application.

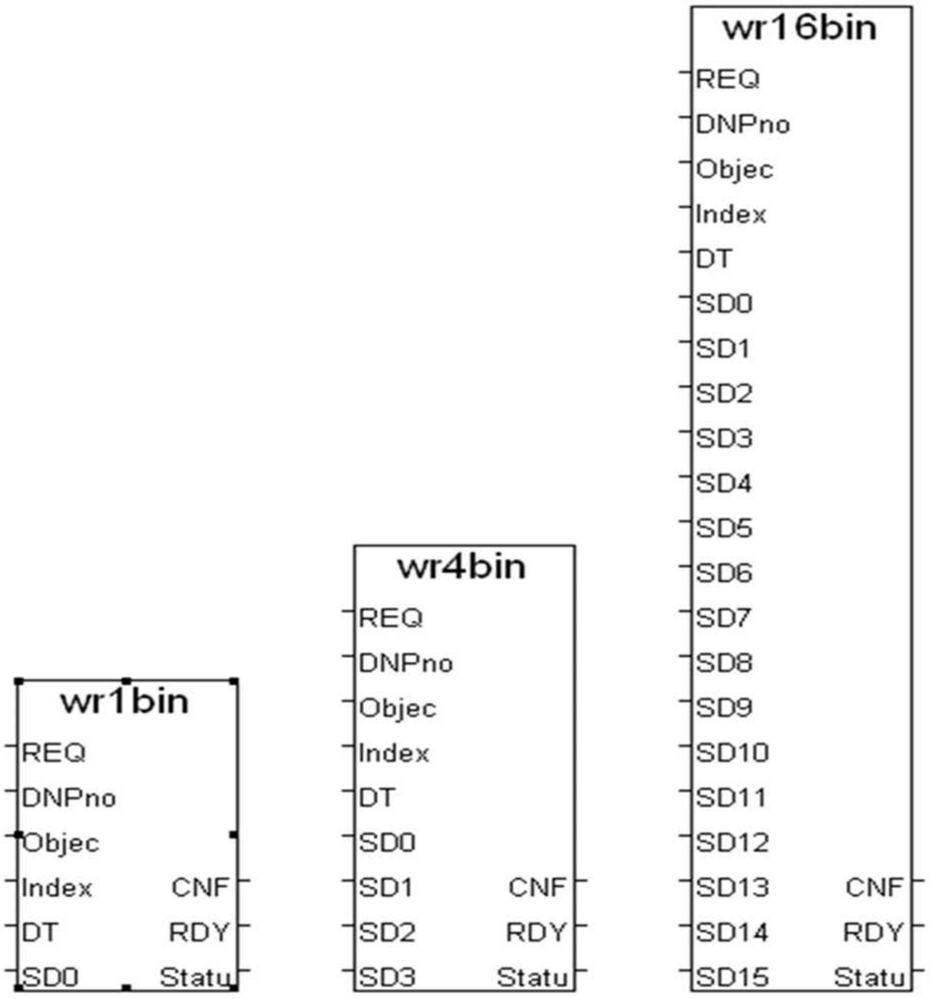


Figure 6.27: wrxxbin Function Block

INPUTS	TYPE	DESCRIPTION
Req	boolean	Data Transfer Request. Initiate data transfer request on rising edge
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to write to peer RTU. For peer RTU write requests, the following values are valid for this function block: BinaryOutput CROB_DirOp CROB_SelOp CROB_DONA
Index	Integer	Starting index of DNP data object to send (consecutive data objects will be sent starting with this one)
dt	Integer	Transaction time-out (peer RTU request only.
SDx		Send data parameters, valid when RDY is TRUE. Quantity of SDx parameters depends on the function block. SDx data is sent when REQ is activated (rising edge).

OUTPUTS	TYPE	DESCRIPTION

Cnf	boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Boolean	Status of data transfer
		TRUE => Data transfer successful
		FALSE => Data transfer unsuccessful

Status	Integer	When CNF is TRUE and RDY is FALSE this indicates a table data access problem (status code) Status = 0 indicates a successful write Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

BinaryOutput ObjectType uses DNP3 *Write* function and provides an efficient method of changing remote binary data. This operation does not provide any direct feedback of the write operation result from the remote node, and is not an implementation required or recommended by the DNP3 Subset Definitions.

CROB_DirOp (Control Relay Output Block) uses DNP3 *Direct Operate* function which provides more secure control and feedback of the result at the remote node. It is a requirement of the DNP3 Subset Definition, but is much less efficient than a *Binary Output Write.* Direct Operate is single phase (request/ response).

CROB_SelOp (Control Relay Output Block) uses DNP3 *Select* and *Operate* functions which provide the most secure control and feedback available in DNP3 but is much less efficient than a *Binary Output Write*. Select Operate is dual phase (select request/select response, operate request/operate response).

CROB_DONA (Control Relay Output Block) uses DNP3 *Direct Operate No Acknowledge* function. This is an insecure control operation as there is no feedback or confirmation from the remote node. Although required by the DNP3 Subset Definitions, **the use of Direct Operate No Acknowledge (DONA) is not recommended for peer communication.**

5.3.5 wrxxana

Write (Control) DNP3 analog data to local or peer RTU address space

Description

This series of function blocks writes current value ISaGRAF integer data into local SCADAPack E RTU DNP analog points, or generates a DNP3 operate or select/operate request to a peer node for DNP3 integer analog objects. xx in the function block name refers to the number of objects sent. The valid DNP object indexes that can be written to a peer DNP device are dependent on the peer device. For other DNP3 devices consult the device manufacturer's documentation.

No Application Layer retries are performed on peer write function blocks. If the user requires retries then either configure Data Link retries (with 'Always' mode), or implement the retries in the ISaGRAF applications.

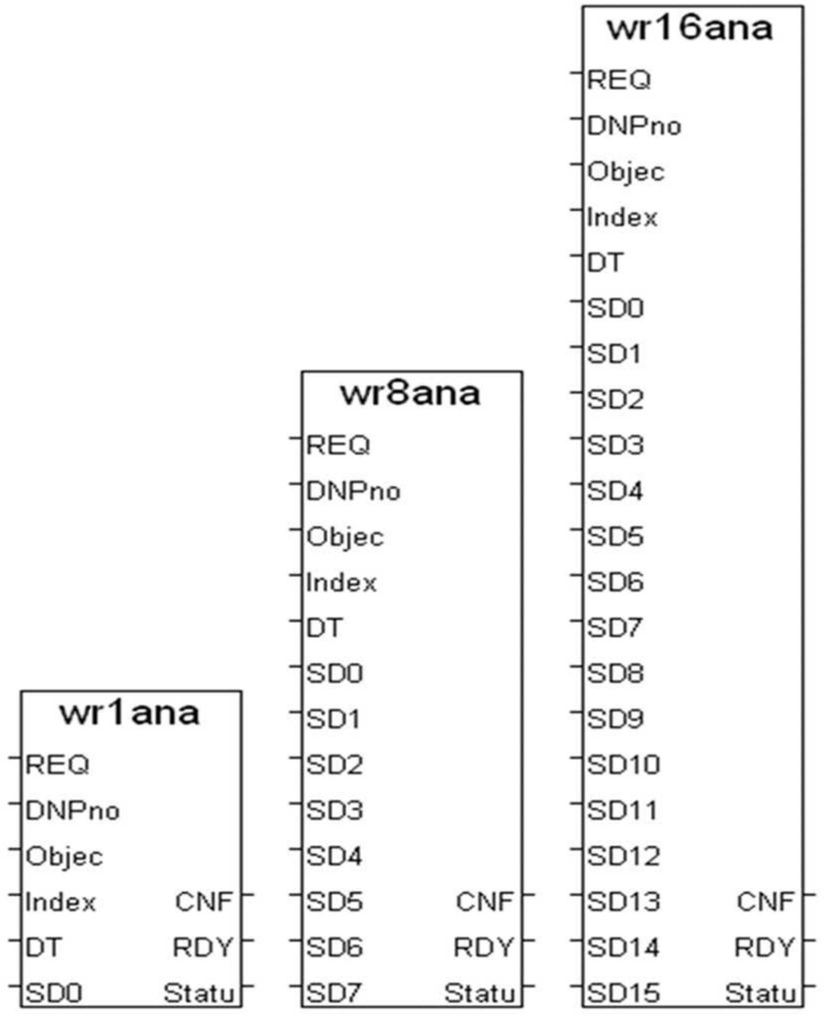


Figure 6.28: wrxxana Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge.
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to write to peer RTU. For peer RTU write requests, the following values are valid for this function block: AnaOut_32_DirOp AnaOut_16_DirOp AnaOut_32_SelOp AnaOut_16_SelOP AnaOut_32_DONA AnaOut_16_DONA WriteTimeAndDate WriteLANTime
Index	Integer	Starting index of DNP data object to write (consecutive data objects will be written starting with this one).
dt	Integer	Transaction time-out (peer RTU request only).
SDx		Send data parameters, valid when RDY is TRUE. Quantity of SDx parameters depends on the function block. SDx data is sent when REQ is activated (rising edge).

OUTPUTS	TYPE	DESCRIPTION

Cnf	Boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Boolean	Status of data transfer
		TRUE => Data transfer successful
		FALSE => Data transfer unsuccessful

Status	Integer	When CNF is TRUE and RDY is FALSE, this indicates a table data access problem (status code) Status = 0 indicates a successful write Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

The that *WriteTimeAndDate* Object Type ignores data presented in the SDx parameters. Rather, the RTU real time clock is sent to the peer RTU. CNF, RDY & STATUS parameters indicate completion of operation as normal.

DNP3 *Direct Operate* functions provides secure control and feedback of the result at the remote node. Direct Operate is single phase (request/response).

DNP3 *Select* and *Operate* functions provide the most secure control and feedback available in DNP3.

Select Operate is dual phase (select request/select response, operate request/operate response).

DNP3 *Direct Operate No Acknowledge* function is an insecure control operation as there is no feedback or confirmation from the remote node. Although required by the DNP3 Subset Definitions, **the use of Direct Operate No Acknowledge (DONA) is not recommended for peer communication.**

5.3.6 wrxxflt

Write DNP3 floating point data into local or peer RTU address space

Description

This series of function blocks writes current value ISaGRAF floating point data into local SCADAPack E RTU DNP analog points, or generates a DNP3 operate or select/operate request to a peer node for DNP3 floating point analog objects. xx in the function block name refers to the number of objects to send. The valid DNP object indexes that can be written from a peer DNP device are dependent on the peer device. For other DNP3 devices consult the device manufacturer’s documentation.

No Application Layer retries are performed on peer write function blocks. If the user requires retries then either configure Data Link retries (with 'Always' mode), or implement the retries in the ISaGRAF applications.

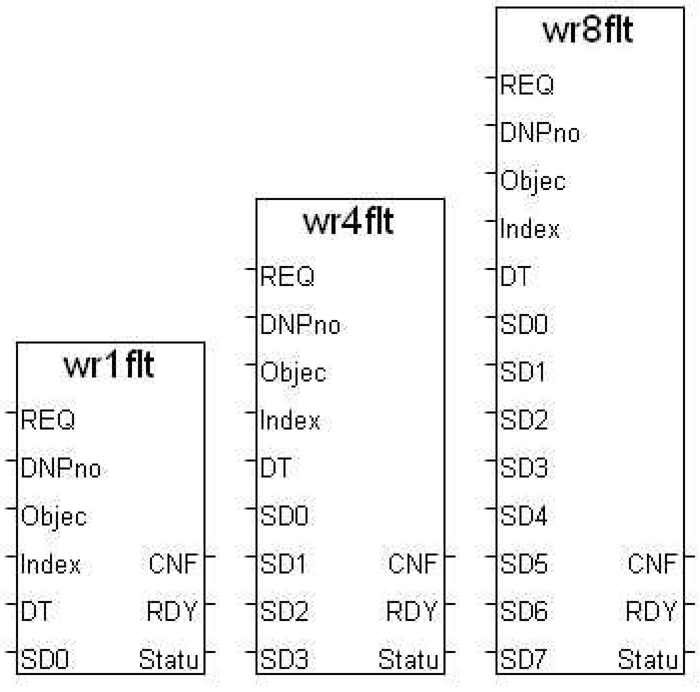


Figure 6.29: wrxxflt Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge.
DNPnode	Integer	DNP Node address (peer RTU request only). Set value to 0 when doing Local RTU data access.
ObjectType	Integer	Local_RTU_Data or DNP data object to write to peer RTU. For peer RTU write, the following values are valid for this function block: AnOutFloat_DirOP AnOutFloat_SelOP AnOutFloat_DONA
Index	Integer	Starting index of DNP data object to write (consecutive data objects will be written starting with this one).
dt	Integer	Transaction time-out (peer RTU request only).
SDx		Send data parameters, valid when RDY is TRUE. Quantity of SDx parameters depends on the function block. SDx data is sent when REQ is activated (rising edge).

OUTPUTS	TYPE	DESCRIPTION

Cnf	Boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Boolean	Status of data transfer
		TRUE => Data transfer successful
		FALSE => Data transfer unsuccessful

Status	Integer	When CNF is TRUE and RDY is FALSE, this indicates a table data access problem (status code) Status = 0 indicates a successful write Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

DNP3 *Direct Operate* functions provides secure control and feedback of the result at the remote node. Direct Operate is single phase (request/response).

DNP3 *Select* and *Operate* functions provide the most secure control and feedback available in DNP3. Select Operate is dual phase (select request/select response, operate request/operate response).

DNP3 *Direct Operate No Acknowledge* function is an insecure control operation as there is no feedback or confirmation from the remote node. Although required by the DNP3 Subset Definitions, **the use of Direct Operate No Acknowledge (DONA) is not recommended for peer communication.**

5.3.7 dc_poll

Force an Integrity Poll to all configured IED's

Description

The ISaGRAF application needs to be able to force the Data Concentrator or master RTU to issue a DNP3 Integrity poll (Class 1,2,3,0 poll) to configured IED's in order to update the local database with a current image of IED point values. The DC_POLL Function Block is used to send this request:

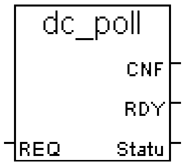


Figure 6.30: dc_poll Function Block

INPUTS	TYPE	DESCRIPTION
Req	boolean	Data Transfer Request. Initiate data transfer request on rising edge

OUTPUTS	TYPE	DESCRIPTION

Cnf	boolean	Data transfer confirm: indicates completion of request
		TRUE => Request Completed
		FALSE => Request not completed

Rdy	Boolean	Status of data
		TRUE => Data ready
		FALSE => Data not ready

Satus	Integer	When CNF is TRUE and RDY is FALSE this indicates a table data access problem (status code) Status = 0 indicates a successful poll Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

The function block operates regardless if the DCons. is currently disabled or not. The dc_poll function block will raise its CNF output when the IED's polled have responded or timed out.

If the dc_poll request is issued while the DCons. is disabled, IED responses will **not** generate local events in the DCons.

5.3.8 Status Codes

Table 6.10: Status Codes Returned by Peer Function Blocks

DNP STATUS	DESCRIPTION
0	Operation successful
8	DNP timeout
9	Bad Read
10	Bad Operate
11	Operation Canceled
50	Bad Function IIN set
51	Object Unknown IIN set
52	Out of Range IIN set
53	Control Already Executing
61	Control Arm Timeout expired
62	Control Select not received
63	Control formatting error
64	Control operation not supported
65	Control queue full
66	Control hardware problem
127	Request could not be added to queue
130	Request not licensed
252	Invalid *ObjectType* code
253	Point does not exist
254	Invalid *ObjectType* for this function block
255	Request in progress

5.4 DNP3 Peer Queued Communication Function Blocks

A second series of ISaGRAF DNP3 Peer function blocks can be used for more sophisticated communication regimes than the transfer of simple data types as described in the function blocks in Section ***DNP3 Communication Function Blocks*** 80. Whereas the simpler peer function blocks transferred data between ISaGRAF variables and remote RTU DNP points, the Queued Peer function blocks transfer data directly between the point databases of two peer RTUs. These queued function blocks provide enhanced functionality. For example:

- If the application programmer has six Integer Analog, one Floating Point Analog and two Binary states to read from a Peer RTU, he may use as many as four simple data type function blocks to achieve this. These four function blocks, if triggered at the same time, would cause four DNP Read request fragments to be generated and queued for transmission. As the second and subsequent fragments are not be sent until the target RTU has processed and responded to the previous fragment, delays due to network latencies are four times longer than if the request could have been combined into one DNP fragment.
- The simple "rdxxana" function blocks allow the application programmer to read 1, 4, 8 or 16 consecutive analog points from the target, depending on which block he chooses. This tends to either force the application programmer to be conscious of either "packing" his target point numbers close together or of having to use more Peer function blocks if they are not close together. This maybe sufficient for new applications, but could be inefficient if adding extra Peer points to an existing application.
- The simple "rdXY" function block does not transfer any point quality data from the target RTU to the ISaGRAF application. This quality data is returned in the "Flag" octet of certain DNP Object types and gives additional information such as "Over-Range" and "A/D Reference error" for example. It is useful to be able to make this quality information available to ISaGRAF programmers.

The Queued Peer function blocks operate in quite a different manner than the simpler peer function blocks in that Peer Read or Write point requests will be queued by the functions "peer_rdq" and "peer_wrq", but executed by the functions "peer_rdx" and "peer_wrx". I.e. a request built up using one or more "peer_rdq" calls to the same queue is sent to the target RTU (or Executed) by the function block "peer_rdx" using that queue. A request built up using one or more "peer_wrq" calls to the same queue is sent to the target RTU (or Executed) by the function block "peer_wrx" using that queue.

Two "queue lists" are created for each ISaGRAF target kernel application, one for Read request and the other for Write requests. These queue lists hold any number of named "point request queues". For example, an application programmer may use an instance of the "peer_rdq" function to send 'Read' requests into a queue named "Lane Cove West A" and another instance of "peer_rdq" to send 'Read' requests into another queue named of "North Head STP". Two separate queues would be created, each of which are executed independently.

The queues and "queue lists" persist while the ISaGRAF application is running, but will be cleared when the ISaGRAF application is stopped or the RTU is Cold Reset. I.e. Points queued using "peer_rdq" need not be re-queued unless the ISaGRAF application is stopped.

5.4.1 peer_rdq

Adds a range of remote DNP points into a named Read Request queue

Description

This ISaGRAF function adds a specified number of DNP points, to be read from a remote peer and written to a local RTU, into a queue for later execution. The DNP point numbers will be added to a specified queue if present or a new queue will be created with the given name. The final *Read* request is executed by the peer_rdx function block, see Section **peer_rdx** 104. The "peer_rdq" function would be typically executed only at application startup.

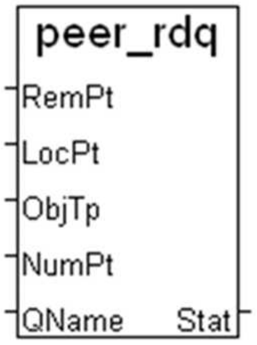


Figure 6.31: "peer_rdq" Function

INPUTS	TYPE	DESCRIPTION
RemPt	Integer	DNP point number in the target RTU device to read from valid values are 0-65534.
LocPt	Integer	DNP point number in the local RTU to write needs to be a valid physical output tor derived point.
ObjTp	Integer	DNP data object to read from peer RTU. The following values are valid for this function: BinaryInput BinInput_Status BinOutput_Stat BinCounter_32 inCounter_16 BinCtr_32_NoFlag BinCtr_16_NoFlag AnalogIn_32 AnalogIn_16 AnaIn32_NoFlag AnaIn_16_NoFlag AnalogIn_Float AnaOut_32_Stat AnaOut_16_Stat AnaOutFloat_Stat
NumPt	Integer	Number of consecutive points to add to queue. Valid entries are 1 – 65534.
Qname	Message	String name of queue, max 50 characters, spaces OK.

The "LocPt" or Local Point parameter should refer to a DIGITAL_OUT_TYPE, ANALOG_OUT_TYPE or COUNTER_TYPE point type, depending on which "ObjTp" DNP Object Type is specified.

OUTPUTS	TYPE	DESCRIPTION
Stat	Boolean	TRUE for successful FALSE for not successful

5.4.2 peer_wrq

Adds a range of DNP points on the local RTU into a named Write queue

Description

This ISaGRAF function adds a specified number of DNP points, to be copied from the local RTU onto a Remote RTU, into a queue for later execution. The DNP point numbers will be added to a specified queue if present or a new queue will be created with the given name. The final *Write* request is executed by the peer_wrx function block, See ***4.4.4 - peer_wrx***. The “peer_wrq” function would be typically executed only at application startup.

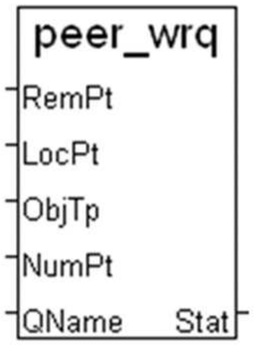


Figure 6.32: "peer_wrq" Function

INPUTS	TYPE	DESCRIPTION
RemPt	Integer	DNP point number in the target RTU device to write to. Valid values are 0-65534.
LocPt	Integer	DNP point number in the local RTU where the current value or state is read from needs be a valid RTU input point type.
ObjTp	Integer	DNP data object to write to peer RTU. The following values are valid for this function: CROB_DirOp AnaOut_32_DirOp AnaOut_16_DirOp AnaOutFloat_DirOp
NumPt	Integer	Number of consecutive points to add to queue. Valid entries are 1 – 65534.
Qname	Message	String name of queue, max 50 characters, spaces OK.

The “LocPt” or Local Point parameter should refer to a DIGITAL_IN_TYPE or ANALOG_IN_TYPE point type, depending on which “ObjTp” DNP Object Type is specified.

OUTPUTS	TYPE	DESCRIPTION
Stat	Boolean	TRUE for successful FALSE for not successful

5.4.3 peer_rdx

Execute queued DNP Read requests

Description

The interface for this ISaGRAF function block is similar to the simple “rd*xx*” family of function blocks. When triggered with a rising edge on the REQ input, the function block will look for a matching queue name with the one supplied by the user. If found, it will iterate the queue in order to build up one or more DNP request fragments. Consecutive point numbers will be packed into the DNP request fragment in an efficient manner.

Unlike the simple ISaGRAF Peer function blocks (which read into ISaGRAF variables), the returned values or states of the remote points are written directly to the local RTU points specified when the point request was queued with “peer_rdq”. Where the user has queued a point request using a DNP Object type that supports status flags, the response status flags will be written to the local RTU points.

The local points will contain valid data (and flags) when CNF & RDY are TRUE. The Analog “*Stat*” output variable will contain Zero if the transaction was successful, otherwise an error code. Also see **Status Codes** 99.

Should the DNP request be unsuccessful, the ISaGRAF application programmer may choose to set the "Point is Bad" property on local RTU points in that named Read queue. This will cause the RTU's Data Processor to also set each point's "Point is Failed" property. A subsequent successful transaction could clear the "Point is Bad" property.

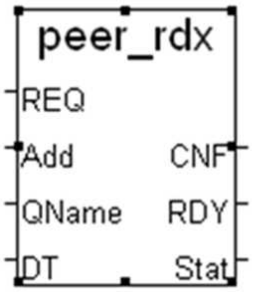


Figure 6.33: "peer_rdx" Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge
Add	Integer	DNP target device address. Valid range is 0-65534.
QName	Message	String name of the queue, max 50 characters, spaces OK.
DT	Timer	Transaction time-out (peer RTU request only.

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Rdy	Boolean	Data is ready
Stat	Integer	Status = 0 indicates a successful read Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

5.4.4 peer_wrx

Execute queued DNP Write requests

Description

The operation of this function block is very similar to the “peer_rdx” function block. It triggers a DNP “Direct Operate” control for the points queued using the “peer_wrq” function. The local RTU points in the Write queue will **not** have any properties changed.

The interface for this ISaGRAF function block is similar to the simple “wr*xx*” family of function blocks. When triggered with a rising edge on the REQ input, the function block will look for a matching queue name with the one supplied by the user. If found, it will iterate the queue in order to build up one or more DNP request fragments. Consecutive point numbers will be packed into the DNP request fragment in an efficient manner.

Unlike the simple ISaGRAF Peer function blocks (which write from ISaGRAF variables), the returned values or states of the remote points are written directly to the local RTU points specified when the point request was queued with “peer_wrq”. Where the user has queued a point request using a DNP Object type that supports status flags, the response status flags will be written to the remote RTU points.

The remote points will contain valid data (and flags) when CNF & RDY are TRUE. The Analog “*Stat*” output variable will contain Zero if the transaction was successful, otherwise an error code. Also see **Status Codes** 99.

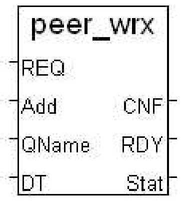


Figure 6.34: "peer_wrx" Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data Transfer Request. Initiate data transfer request on rising edge
Add	Integer	DNP target device address. Valid range is 0-65534
QName	Message	String name of the queue, max 50 characters, spaces OK
DT	Timer	Transaction time-out (peer RTU request only)

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise
Rdy	Boolean	Data is ready
Stat	Integer	Status = 0 indicates a successful write (control) Status = 255 indicates an outstanding DNP request See **Status Codes** 99 for other status codes

5.4.5 peer_rdc

Clear a Read request queue

Description

The “peer_rdc” ISaGRAF functions allow the application programmer to clear every point in a named queue. The functions will return TRUE if the specified queue is found and its points are removed successfully.

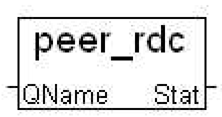


Figure 6.35: "peer_rdc" Function

INPUTS	TYPE	DESCRIPTION
QName	Message	String name of the queue, max 50 characters, spaces OK.

OUTPUTS	TYPE	DESCRIPTION
Stat	Boolean	TRUE for success otherwise FALSE.

5.4.6 peer_wrc

Clear a Write request queue

Description

The “peer_wrc” ISaGRAF functions allow the application programmer to clear every point in a named queue. The functions will return TRUE if the specified queue is found and its points are removed successfully.

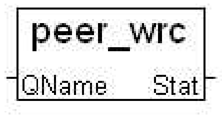
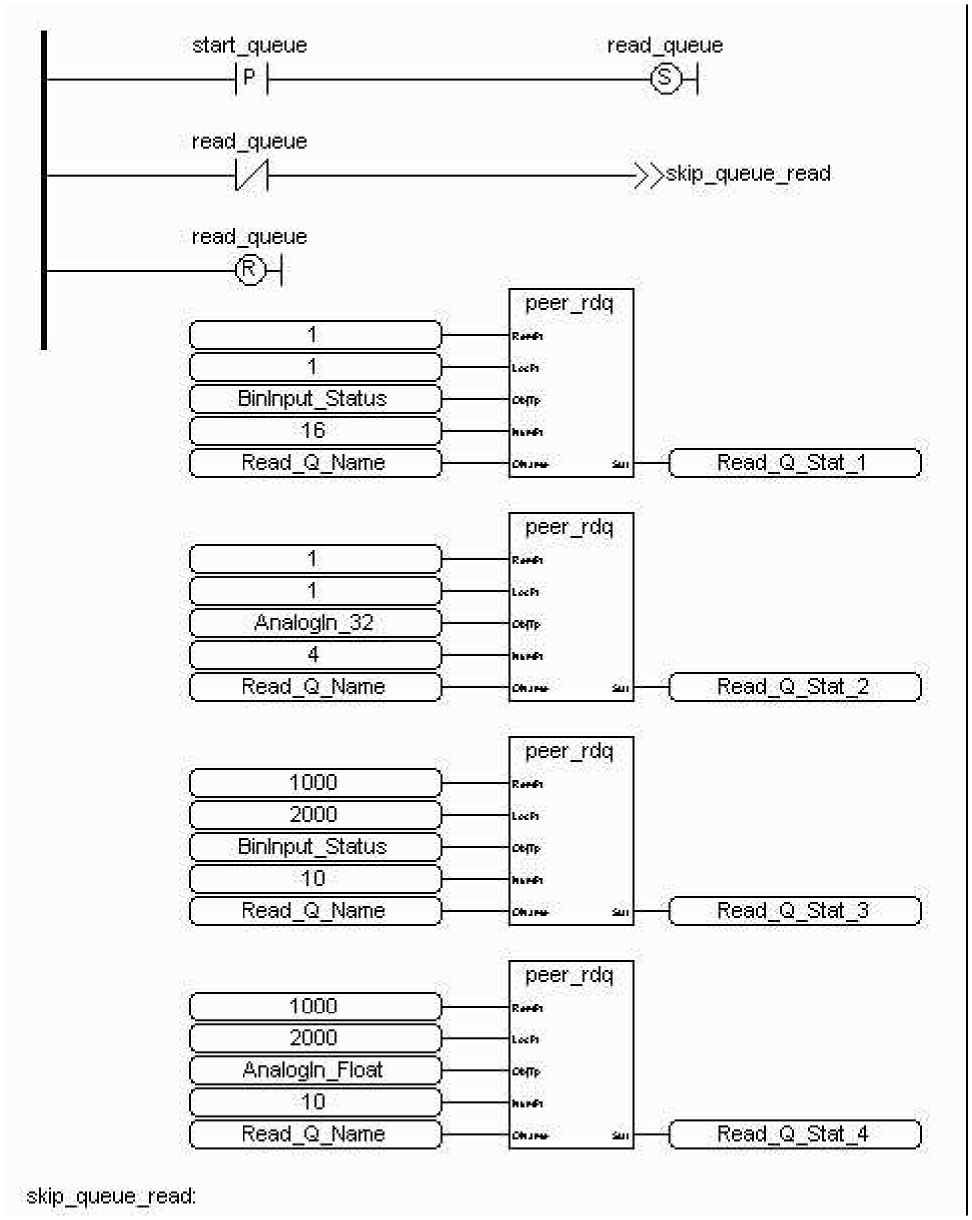


Figure 6.36: "peer_wrc" Function

INPUTS	TYPE	DESCRIPTION
QName	Message	String name of the queue, max 50 characters, spaces OK.

OUTPUTS	TYPE	DESCRIPTION
Stat	Boolean	TRUE for success, otherwise FALSE.

5.4.7 Queued Peer Read Example



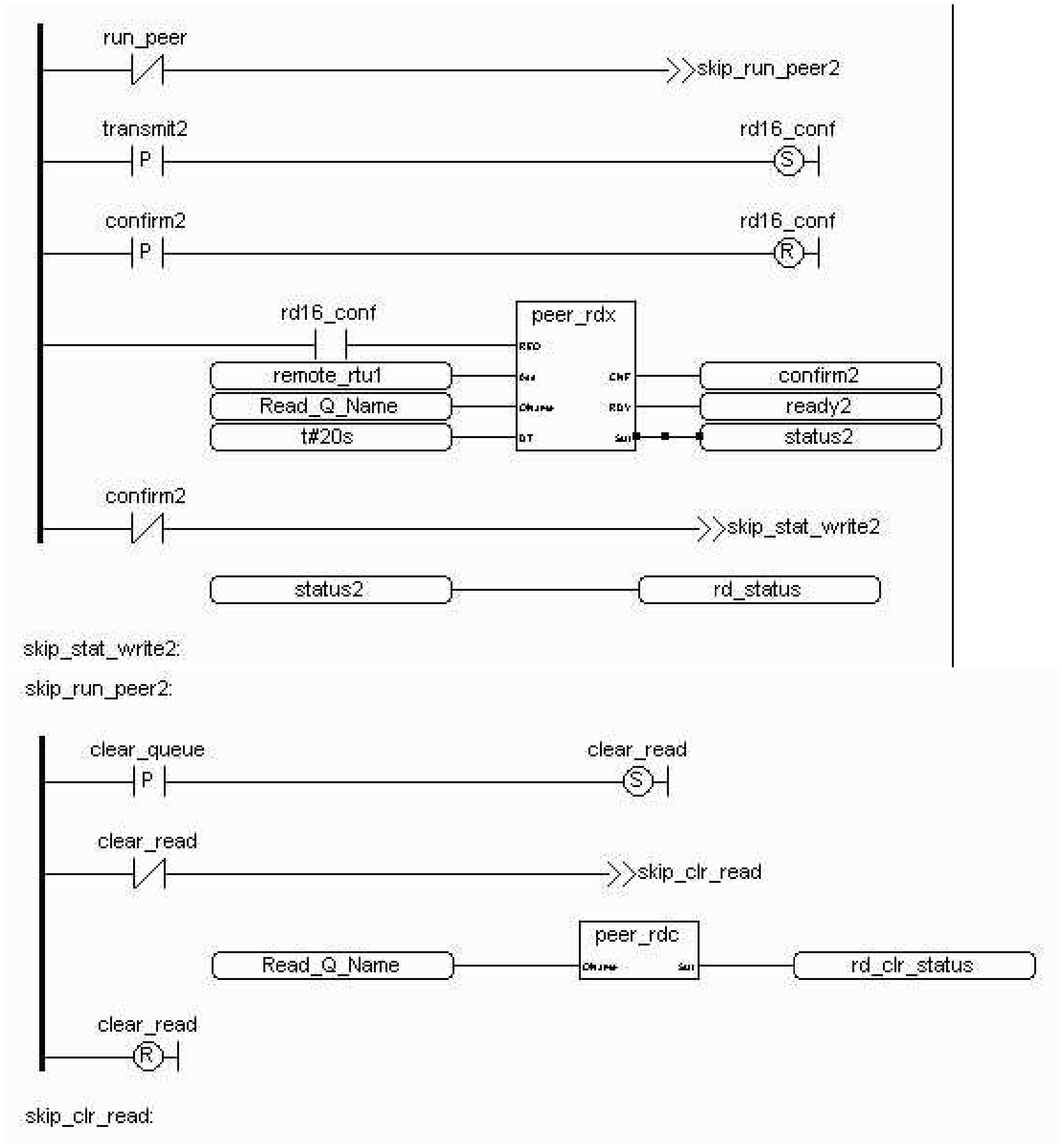


Figure 6.37: Example Peer Queued Read

Figure 6.37 110 shows an example of how a Peer Queued Read test program might be implemented. Setting the boolean variable “start queue” TRUE causes four Read Queue functions to be executed once only. Following that, setting the “run_peer” variable TRUE will cause the “peer_rdx” to issue a DNP Peer read request. The “peer_rdx” function block will be re-triggered by its Confirm (CNF) line going TRUE to indicate completion. This cycle will continue until “run_peer” is set FALSE.

Setting the “clear_queue” Boolean variable TRUE causes the “peer_rdc” function to execute once. This will clear every Read request in the named queue.

5.5 PLC Communications Control Function Blocks

These function blocks provide an interface to disable and enable Read and Write communications to PLC Devices using an ISaGRAF application:

- ***mbusctrl*** 113
- ***mtcpctrl*** 114
- ***df1ctrl*** 115
- ***koyoctrl*** 116

5.5.1 mbusctrl

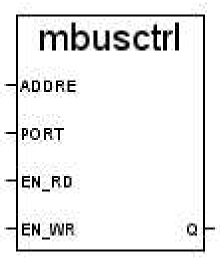
Control Communications to Modbus Device

Description

This function block allows enabling or disabling of communication with Modbus PLC devices configured using ISaGRAF PLC device I/O boards. Device communications setup by **mbusxxyy** I/O boards can be controlled using this function block.

Every I/O board defining the communication to the specified Modbus PLC device are affected by this function block.

A PLC device address of 0 will enable or disable communications to every device on the specified port.



mbusctrl Function Block

INPUTS	TYPE	DESCRIPTION
Address	Integer	Modbus address of the device to control. An address of 0 will enable/disable communication to every device on the specified port
Port	Integer	The serial port number that the device is connected to
En_RD	Boolean	When TRUE, ISaGRAF PLC device Input Boards (with the specified Modbus PLC device address) will be updated at the configured data update rate. When FALSE, data reads to the PLC Device will be disabled
En_WR	Boolean	When TRUE, data changes on ISaGRAF PLC device Output Boards (with the specified Modbus PLC device address) will be written to the device. When FALSE, data writes to the PLC Device will be disabled

OUTPUTS	TYPE	DESCRIPTION
Q	Integer	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

5.5.2 mtcpctrl

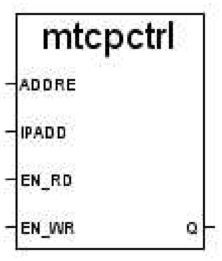
Control Communications to Modbus/TCP or Modbus RTU in TCP Device

Description

This function block allows enabling or disabling of communication with Modbus/TCP PLC devices configured using ISaGRAF PLC device I/O boards. Device communications setup by **mtcpxxyy** or **mrtpxxyy** I/O boards can be controlled using this function block.

Every I/O board defining the communication to the specified Modbus/TCP or Modbus RTU in TCP PLC device are affected by this function block.

A PLC device address of 0 will enable or disable communications to every device at the specified IP Address.



mtcpctrl Function Block

INPUTS	TYPE	DESCRIPTION
Address	Integer	Modbus address of the device to control. An address of 0 will enable/disable communication to every device on the specified port
IPaddr	Message	The IP address that the device is connected to
En_RD	Boolean	When TRUE, ISaGRAF PLC device Input Boards (with the specified Modbus/ TCP or Modbus RTU in TCP PLC device address) will be updated at the configured data update rate. When FALSE, data reads to the PLC Device will be disabled
En_WR	Boolean	When TRUE, data changes on ISaGRAF PLC device Output Boards (with the specified Modbus/TCP or Modbus RTU in TCP PLC device address) will be written to the device. When FALSE, data writes to the PLC Device will be disabled

OUTPUTS	TYPE	DESCRIPTION
Q	Integer	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

5.5.3 df1ctrl

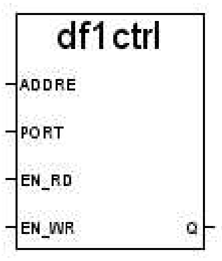
Control Communications to DF/1 Device

Description

This function block allows enabling or disabling of communication with DF/1 PLC devices configured using ISaGRAF PLC device I/O boards. Device communications setup by **df1xxyy** I/O boards can be controlled using this function block.

Every I/O board defining the communication to the specified DF/1 PLC device are affected by this function block.

A PLC device address of 0 will enable or disable communications to every device on the specified port.



df1ctrl Function Block

INPUTS	TYPE	DESCRIPTION
Address	Integer	DF/1 address of the device to control. An address of 0 will enable/disable communication to every device on the specified port
Port	Integer	The serial port number that the device is connected to
En_RD	Boolean	When TRUE, ISaGRAF PLC device Input Boards (with the specified DF/1 PLC device address) will be updated at the configured data update rate. When FALSE, data reads to the PLC Device will be disabled
En_WR	Boolean	When TRUE, data changes on ISaGRAF PLC device Output Boards (with the specified DF/1 PLC device address) will be written to the device. When FALSE, data writes to the PLC Device will be disabled

OUTPUTS	TYPE	DESCRIPTION
Q	Integer	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

5.5.4 koyoctrl

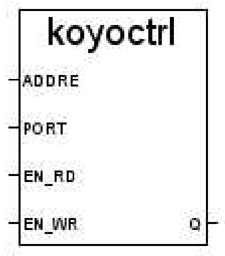
Control Communications to DirectNET Koyo Device

Description

This function block allows enabling or disabling of communication with DirectNET Koyo PLC devices configured using ISaGRAF PLC device I/O boards. Device communications setup by **koyoxxyy** I/O boards can be controlled using this function block.

Every I/O board defining the communication to the specified Koyo PLC device are affected by this function block.

A PLC device address of 0 will enable or disable communications to every device on the specified port.



koyoctrl Function Block

INPUTS	TYPE	DESCRIPTION
Address	Integer	DirectNET Koyo address of the device to control. An address of 0 will enable/ disable communication to every device on the specified port
Port	Integer	The serial port number that the device is connected to
En_RD	Boolean	When TRUE, ISaGRAF PLC device Input Boards (with the specified Koyo PLC device address) will be updated at the configured data update rate. When FALSE, data reads to the PLC Device will be disabled
En_WR	Boolean	When TRUE, data changes on ISaGRAF PLC device Output Boards (with the specified Koyo PLC device address) will be written to the device. When FALSE, data writes to the PLC Device will be disabled

OUTPUTS	TYPE	DESCRIPTION
Q	Integer	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

5.6 Serial Port User Communication Functions

These function blocks provide an interface to setup serial port parameters on the SCADAPack E RTU via an ISaGRAF application:

- ***comopen*** 118
- ***comclose*** 119
- ***comrx*** 120
- ***comrxb*** 121
- ***comrxclr*** 122
- ***comtx*** 123
- ***comtxb*** 124
- ***getport*** 125
- ***setport*** 126

5.6.1 comopen

Open a serial port for use

Description

This function is used to open an RTU serial port for use within an ISaGRAF program. The function returns a positive integer ID for the opened port if successful, or a zero if not successful. The ID is used by the *comrx, comtx, comclose* and *comrxclr* functions to identify which port to act upon, as more than one serial port may be in use. The RTU port “Function” configuration needs to be set to “ISaGRAF-User” using the SCADAPack E Configurator before the port can be used within an ISaGRAF program. Multiple ports may be configured for ISaGRAF-User and opened for User access by the ISaGRAF applications.

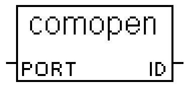


Figure 6.38: comopen Function

INPUTS	TYPE	DESCRIPTION
Port	Message	Serial Port to control valid values are: PORT 0 PORT 1 PORT 2 PORT 3 PORT 4 PORT is case insensitive (upper and lower case allowed), and a space before the port digit is optional (e.g. “PORT0” & “PORT 0” allowed).

OUTPUTS	TYPE	DESCRIPTION
Id	Integer	Id of serial port. Return value is 0 if port could not be opened.

5.6.2 comclose

Close a serial port

Description

This function is used to close an RTU serial port currently in use by an ISaGRAF program (ie. A port previous opened using the comopen function). The figure below shows the function with the following calling and return parameters:

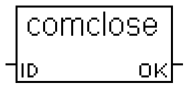


Figure 6.39: comclose Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	TRUE if operation is successful. Else FALSE.

5.6.3 comrx

Read characters from a serial port

Description

This function receives a string of ASCII characters from an open RTU serial port. Either a string length or a terminating character can be specified to indicate when to stop reading characters. The figure below shows the function with the following calling and return parameters:

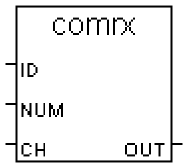


Figure 6.40: comrx Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)
Num	Integer	Number of characters to be read (0 = no minimum length to be read)
Ch	Integer	Watch for this character to terminate string (non–valid ASCII character >=256 disables a terminating character)

OUTPUTS	TYPE	DESCRIPTION
Out	Message	Character string read from serial port

5.6.4 comrxb

Read binary characters from serial port and display in ASCII

Description

This function receives an array of binary characters from an open RTU serial port and displays it as an ASCII string. This function is used for receiving binary protocols. Either an array length or a terminating character can be specified to indicate when to stop reading characters. The array length should be set as twice the number of binary characters that you wish to receive. For example, to receive the binary characters 0x32 0xF4 0x5D, you would set NumChars to be 6, and the received msg would be the string ‘32F45D’ (which has 6 characters). ***Figure 6.41***121 shows the function with the following calling and return parameters:

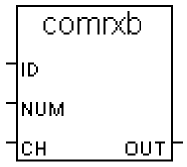


Figure 6.41: comrxb Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)
Num	Integer	2 * Number of characters to be read (0 = no minimum length to be read)
Ch	Integer	Watch for this character to terminate string (non–valid ASCII character >=256 disables a terminating character)

OUTPUTS	TYPE	DESCRIPTION
Out	Message	Character string representation of the binary protocol from serial port

5.6.5 comrxclr

Clear serial port receive (RX) buffer

Description

This function is used to clear an RTU's serial port receive buffer. It requires that the serial port has been opened using the comopen function.

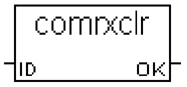


Figure 6.42: comrxclr Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	TRUE if characters were cleared from the buffer. FALSE if characters were not cleared (i.e. no characters available), or if the port wasn't open, or if the ID was invalid.

5.6.6 comtx

Write characters to a serial port

Description

This function writes a string of ASCII characters to an open RTU serial port.

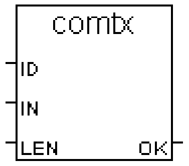


Figure 6.44: comtx Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)
In	Message	Character string to write to the serial port
Len	Integer	Number of characters to write. The LEN parameter also allows for the transmission of the NUL (ASCII 0) character, which is generally used to terminate a string of characters.

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	TRUE if operation successful.

5.6.7 comtxb

Convert a string of ASCII characters to binary and write to serial port

Description

This function converts a string of ASCII characters to binary and writes them out an open RTU serial port. The NumChars variable should be set to the number of ASCII characters to transmit (i.e. twice the number of binary characters that will be transmitted). That is, to transmit the binary characters 0x02 0xFD 0x6A then you would set TxMsg as '02FD6A' and NumChars as 6. ***Figure 6.45***124 shows the function with the following calling and return parameters:

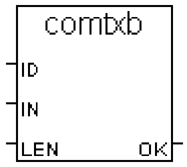


Figure 6.45: comtxb Function

INPUTS	TYPE	DESCRIPTION
Id	Integer	ID of serial port to close (ID from previous comopen)
In	Message	Character string to convert to binary and write to serial port
Len	Integer	Number of characters to write. (2* the number of binary characters to write).

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	TRUE if operation successful.

5.6.8 getport

Read a DNP3 RS-232 line state

Description

The GETPORT ISaGRAF function block reads a serial port hardware line state for a DNP3 serial port only.

This function is not supported on non-DNP3 ports. DNP3 driver ports (PPP, GPRS, 1xRTT, Hayes Modem) are also not supported.

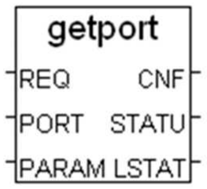


Figure 6.54: getport Function Block

Inputs:

Name	ISaGRAF data type	Notes
Req	Boolean	Data Read request: initiate data transfer when asserted (rising edge)
Port	Analog	DNP3 RTU Serial Port No.
Param	Analog	Which DNP serial port hardware line to read. Use one of the following defines: CTS, DCD, DSR (defined in the “common.eqv” file).

Outputs:

Name	ISaGRAF data type	Notes
CNF	Boolean	Data transfer confirm: indicates completion of request
Status	Analog	Transaction status value: Success = 0, Invalid Port = 1, Invalid Param. = 2
Lstate	Boolean	The state of the serial port hardware line. True if the hardware line is asserted.

5.6.9 setport

Set a DNP3 port RS-232 line state

Description

The SETPORT ISaGRAF function block sets a serial port hardware line state for a DNP3 serial port only.

This function is not supported on non-DNP3 ports. DNP3 driver ports (PPP, GPRS, 1xRTT, Hayes Modem) are also not supported.

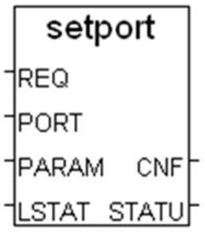


Figure 6.55: setport Function Block

Inputs:

Name	ISaGRAF data type	Notes
Req	Boolean	Data Write request: initiate data transfer when asserted (rising edge)
Port	Analog	DNP3 Serial Port Number
Param	Analog	Which DNP serial port hardware line to control. Use one of the following defines are valid: RTS, DTR (Defined in the "common.eqv" file).
Lstate	Boolean	The new state of the serial port hardware line. True if the hardware line is to be asserted.

Outputs:

Name	ISaGRAF data type	Notes
CNF	Boolean	Data transfer confirm: indicates completion of request
Status	Analog	Transaction status value: Success = 0, Invalid Port = 1, Invalid Param. = 2

5.7 Miscellaneous Function Blocks

The application programmer can use these miscellaneous function blocks to further enhance the capability of an ISaGRAF application:

- ***pida*** 128
- ***pidd*** 131
- ***pid_al*** 135
- ***flow*** 138
- ***total*** 140
- ***rea_msg*** 143
- ***gen_evt*** 145
- ***genmsevt*** 147
- ***ana_time*** 149
- ***cmd_exec*** 151
- ***rtuparam*** 152
- ***chgroute*** 159
- ***chgrtnum*** 162
- ***chgrtprt*** 163

5.7.1 pida

pida

Analog Output PID Function Block

Description

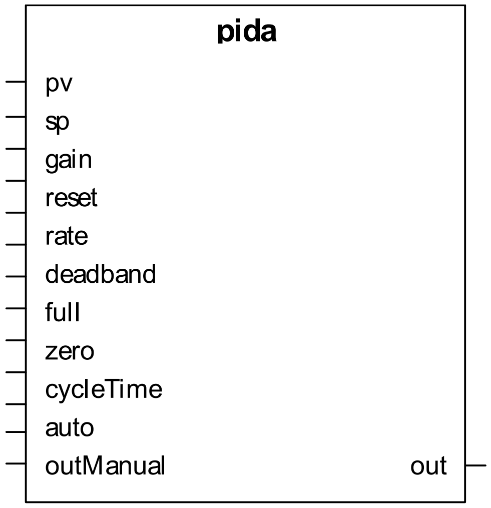
The **pida** function block performs a PID algorithm and calculates an analog output.

In automatic mode, the output is calculated using the PID algorithm. A new calculation is done at the rate specified by cycleTime.

In manual mode (auto = FALSE), the output is set to the value of the outManual input. The output is limited to the range set by full and zero.

If the cycleTime parameter is less than the cycle time of the ISaGRAF application program, the sampling period is the period of the application.

The analog output function block ***pida*** appears as follows.



Arguments

Inputs	Type	Description
pv	Real	The process value is used to calculate the process error in the PID algorithm. *error* = *pv* – *sp*
sp	Real	The setpoint value is used to calculate the process error in the PID algorithm. *error* = *pv* – *sp.*
gain	Real	The proportional gain. A positive value of gain configures a forward-acting PID controller and a negative value of gain configures a reverse acting controller.
reset	Real	The reset time, in seconds. This controls the reset gain (or magnitude of integral action) in a PI or PID controller. Valid range

Inputs	Type	Description
		is any value greater than 0. A value of 0 disables the reset action.
rate	Real	The rate time, in seconds. This controls the rate gain (or magnitude of derivative action) in a PD or PID controller. Valid range is any value greater than 0. A value of 0 disables the rate action.
deadband	Real	The setpoint deadband is used by the PID algorithm to determine if the process requires control outputs. If the absolute value of the error is less than the deadband, then the function block skips execution of the control algorithm. This permits faster execution when the error is within a certain acceptable range or deadband. Valid range is any value greater than 0.
full	Real	The full input is used in limiting the maximum output value of the pida function block. If the PID algorithm calculates an **out** quantity that is greater than the value stored in full the out quantity is set equal to the value stored in full. The full input should be greater than the zero input.
zero	Real	The zero input is used in limiting the minimum output value of the pida function block. If the PID algorithm calculates an **out** quantity that is less than the value stored in zero, the out quantity is set equal to the value stored in zero. The zero input should be less than the full input.
cycleTime	Time	The value of the PID algorithm execution period measured in seconds. Any value greater than or equal to 0.001 seconds (1 ms) may be specified. If the cycleTime specified is less than the scan time of the ISaGRAF application program, the scan time of the application becomes the PID cycleTime.
auto	Boolean	When set to TRUE to the **out** value is calculated using the PID algorithm. When set to FALSE the **out** value is set to the value of **outManual**.
outManual	Real	The value that the **out** is set to when the pida function block is in the manual mode.

Outputs	***Type***	***Description***
out	Real	PID algorithm output.

PID Velocity Algorithm

The PIDA function uses the velocity form of the PID algorithm. The velocity form calculates the change in the output and adds it to the previous output.

$$m_n = m_{n-1} + K\left[e_n - e_{n-1} + \frac{T}{T_i}e_n + \frac{R}{T}\left(p_n - 2p_{n-1} + p_{n-2}\right)\right]$$

where

$$e_n = s_n - p_n$$

and: *e* = error
s = setpoint
p = process value
K = gain
T = execution period
T_i = integral or reset time
R = rate gain
m = output

The above parameters are fully described below.

Setpoint

The **setpoint** is a floating-point value representing the desired value of the process value. The error value is the difference between the process value and the setpoint.
error = process value – setpoint (+/- deadband).

Process Value

The **process value** is a value that represents the actual state of the process being controlled.

Gain

The proportional (P) part of the PID algorithm is the **gain.** A positive value of gain configures a forward-acting PID controller and a negative value of gain configures a reverse acting controller.

Reset Time

The integral (I) part of the PID algorithm is the **reset time**. This value, in seconds, controls the reset gain (or magnitude of integral action) in a PI or PID controller. This is typically referred to as Seconds Per Repeat. From the equation above it is seen that the integral action of the PI or PID controller is a function of the reset time and the execution period (cycle time). A smaller reset time provides more integral action and a larger reset time provides less integral action. Valid range is any value greater than 0. A value of 0 disables the reset action.

Rate Gain

The derivative (D) part of the PID algorithm is the **rate time.** This value, in seconds, controls the rate gain (or magnitude of derivative action) in a PD or PID controller. From the equation above it is seen that the derivative action of the PD or PID controller is a function of the rate gain and the execution period (cycle time). A larger rate gain provides more derivative action and a smaller rate gain provides less derivative action. Valid range is any value greater than 0. A value of 0 disables the rate action.

Deadband

The **deadband** parameter is used by the PID algorithm to determine if the process requires the control outputs to be changed. If the absolute value of the error is less than the deadband, then the function block skips execution of the control algorithm. This stops changes to the output when the process value is near the setpoint and can reduce wear on the control elements. Valid range is any value greater than 0.The **setpoint** is a floating-point value representing the desired value of the process value.

Full

The **full** setting is used in limiting the maximum output value of the PIDA function. If the PID algorithm calculates an **output** quantity that is greater than the value stored in **full**, the output quantity is set equal to the value stored in **full**. The **full** setting should be greater than the **zero** setting.

Zero

The **zero** setting is used in limiting the minimum output value of the PIDA function. If the PID algorithm calculates an **output** quantity that is less than the value stored in **zero**, the **output** quantity is set equal to the value stored in **zero**. The **zero** setting should be less than the **full** setting.

Cycle Time

The **cycle time** is the floating-point value of the PID algorithm execution period measured in seconds. Any value greater than or equal to 0.001 seconds (1 ms) may be specified. If the cycle time specified is less than the scan time of the program, the program scan time becomes the PID cycle time.

Manual Mode

The **manual mode output** is the value that the **output** is set to when the PIDA function is in manual mode.

See Also pidd 131

5.7.2 pidd

pidd

Discrete Output PID Function Block

Description

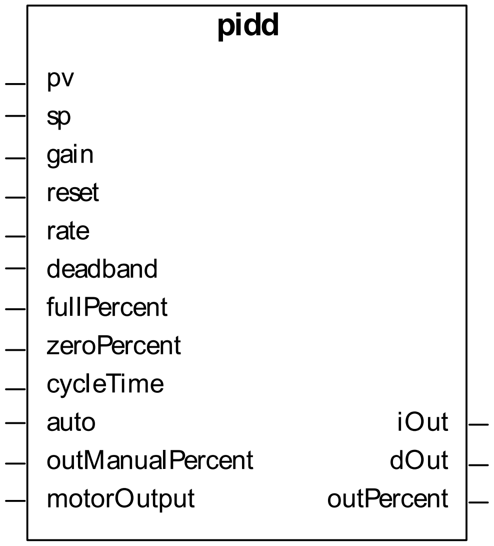
The **pidd** function block performs a PID algorithm and controls two discrete outputs. The output is a duty cycle of a Boolean value. The duty cycle depends on the value of the output. The increase output, iOut, is cycled when the outputPercent is greater than zero. The decrease output, dOut, is cycled when the outputPercent is less than zero.

In automatic mode, the output is calculated using the PID algorithm. A new calculation is done at the rate specified by cycleTime.

In manual mode (auto = FALSE), the outPercent is set to the value of the outManualPercent input. The output is limited to the range set by fullPercent and zeroPercent.

If the cycleTime parameter is less than the cycle time of the ISaGRAF application program, the sampling period is the period of the application.

The analog output function block ***pidd*** appears as follows.



Arguments

Inputs	Type	Description
pv	Real	The process value is used to calculate the process error in the PID algorithm. *error* = *pv* – *sp*
sp	Real	The setpoint value is used to calculate the process error in the PID algorithm. *error* = *pv* – *sp*
gain	Real	The proportional gain. A positive value of gain configures a forward-acting PID controller and a negative value of gain configures a reverse acting controller.
reset	Real	The reset time, in seconds. This controls the reset gain (or magnitude of integral action) in a PI or PID controller. Valid range is any value greater than 0. A value of 0 disables the reset action.
rate	Real	The rate time, in seconds. This controls the rate gain (or magnitude of derivative action) in a PD or PID controller. Valid range is any value greater than 0. A value of 0 disables the rate action.

Inputs	Type	Description
deadband	Real	The setpoint deadband is used by the PID algorithm to determine if the process requires control outputs. If the absolute value of the error is less than the deadband, then the function block skips execution of the control algorithm. This permits faster execution when the error is within a certain acceptable range or deadband. Valid range is any value greater than 0.
fullPercent	Real	The full scale output limit in percent of cycleTime.
zeroPercent	Real	The zero scale output limit in percent of cycleTime.
cycleTime	Time	The value of the PID algorithm execution period measured in seconds. Any value greater than or equal to 0.001 seconds (1 ms) may be specified. If the cycleTime specified is less than the scan time of the ISaGRAF application program, the scan time of the application becomes the PID cycleTime.
auto	Boolean	When set to TRUE to the output value is calculated using the PID algorithm. When set to FALSE the output value is set to the value of **outManualPercent** .
outManualPercent	Real	The value that the output is set to when the pida function block is in the manual mode.
motorOutput	Boolean	When set to TRUE **iOut** and **dOut** are off when error is within the deadband. When set to FALSE **iOut** and **dOut** act normally.

Outputs	Type	Description
iOut	Boolean	The increase output. This output cycles when **outPercent** is greater than zero.
dOut	Boolean	The decrease output. This output cycles when **outPercent** is less than zero.
outPercent	Real	PID algorithm output.

PID Velocity Algorithm

The PIDD function uses the velocity form of the PID algorithm. The velocity form calculates the change in the output and adds it to the previous output.

$$m_n = m_{n-1} + K\left[e_n - e_{n-1} + \frac{T}{T_i}e_n + \frac{R}{T}\left(p_n - 2p_{n-1} + p_{n-2}\right)\right]$$

where

$$e_n = s_n - p_n$$

and:	*e* = error
	s = setpoint
	p = process value
	K = gain
	T = execution period
	T_i = integral or reset time
	R = rate gain
	m = output

The above parameters are fully described below.

Setpoint

The **setpoint** is a floating-point value representing the desired value of the process value. The error value is the difference between the process value and the setpoint.
error = process value – setpoint (+/- deadband).

Process Value

The **process value** is a value that represents the actual state of the process being controlled. See the Function Variables section above for the registers to use for the process value input to the PIDD.

Gain

The proportional (P) part of the PID algorithm is the **gain.** A positive value of gain configures a forward-acting PID controller and a negative value of gain configures a reverse acting controller.

Reset Time

The integral (I) part of the PID algorithm is the **reset time**. This value, in seconds, controls the reset gain (or magnitude of integral action) in a PI or PID controller. This is typically referred to as Seconds Per Repeat. From the equation above it is seen that the integral action of the PI or PID controller is a function of the reset time and the execution period (cycle time). A smaller reset time provides more integral action and a larger reset time provides less integral action. Valid range is any value greater than 0. A value of 0 disables the reset action.

Rate Gain

The derivative (D) part of the PID algorithm is the **rate time.** This value, in seconds, controls the rate gain (or magnitude of derivative action) in a PD or PID controller. From the equation above it is seen that the derivative action of the PD or PID controller is a function of the rate gain and the execution period (cycle time). A larger rate gain provides more derivative action and a smaller rate gain provides less derivative action. Valid range is any value greater than 0. A value of 0 disables the rate action.

Deadband

The **deadband** parameter is used by the PID algorithm to determine if the process requires the control outputs to be changed. If the absolute value of the error is less than the deadband, then the function block skips execution of the control algorithm. This stops changes to the output when the process value is near the setpoint and can reduce wear on the control elements. Valid range is any value greater than 0.

Full Scale Output

The **full%** setting is the full-scale output limit in percent of **cycle time**. For example if the cycle time is 10 seconds and the full% value is 100 then the maximum duty cycle of the output% is 100 percent or 10 seconds.

When the zero% value is 0 or greater then the increase output is turned on for the duty cycle of the output%. When the zero% value is less than zero the decrease output is turned on for the duty cycle of the output% when the output% is negative.

Zero Scale Output

The **zero%** setting is the zero scale output limit in percent of **cycle time**. When the zero% value is 0 or greater then the increase output is turned on for the duty cycle of the output%. When the zero% value is less than zero the decrease output is turned on for the duty cycle of the output% when the output% is negative.

Cycle Time

The **cycle time** is the floating-point value of the PID algorithm execution period measured in seconds. Any value greater than or equal to 0.001 seconds (1 ms) may be specified. If the cycle time specified is less than the scan time of the program, the program scan time becomes the PID cycle time.

Manual Mode Output

The **manual mode output%** is the value that the **output%** is set to when the PIDD function is in manual mode.

Motor Output

When the **motor output** is set to **enabled** the **increase** and **decrease** outputs are de-energized when error is within the deadband. When the **motor output** is set to **disabled** the **increase** and **decrease** outputs operate continuously based on the **output%**.

See Also pida 128

5.7.3 pid_al

PID regulator with output limiting and integral wind-up prevention

Description

This function block provides PID (Proportional-Integral-Derivative) control to ISaGRAF with output limiting and integral wind-up prevention. The figure below shows the function block with the following calling and return parameters:

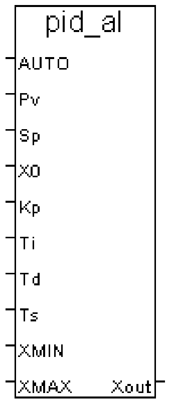


Figure 6.46: pid_al Function Block

INPUTS	TYPE	DESCRIPTION
Auto	Boolean	PID Automatic mode when TRUE PID Manual mode when FALSE AUTO mode needs to be set to FALSE at initialization In Manual mode, the following occurs: Xout = X0 Integral = (X0*Ti)/Kp; where Ti and Kp and Integral Time constants and Proportional constants respectively.
PV	Real	Process Variable –output value from process
SP	Real	Set Point –desired value of process variable
X0	Real	Manual mode output adjustment value
Kp	Real	Proportionality constant
Ti	Real	Integral Time constant
Td	Real	Derivative Time Constant
Ts	Timer	Sampling period in ms. This value needs to be greater than the ISaGRAF target scan rate.
Xmin	Real	Minimum limit on process output command value
Xmax	Real	Maximum limit on process output command value. Should be greater than Xmin.

OUTPUTS	TYPE	DESCRIPTION
Xout	Real	Process command value. Xout is limited within Xmin and Xmax. The Integral term is held constant when Xout reaches Xmin or Xmax. Xout = X0 in manual mode.

5.7.4 flow

flow

Non-Volatile Flow Accumulator

Description

The ***flow*** function accumulates flow from pulse type devices such as turbine flow meters.

When the accumulate input is ON the function accumulates volume and calculates the flow rate.

When the Log Data input goes from OFF to ON, the accumulated volume is saved in the history registers. Older history is pushed down and the oldest record is discarded. The Log Data input needs to be triggered at least once every 119 hours (at the maximum pulse rate of 10kHz). Otherwise the volume accumulator will overflow, and the accumulated volume will not be accurate.

The accuracy of the FLOW block improves with longer periods between logging data. For greatest accuracy avoid logging small periods of time.

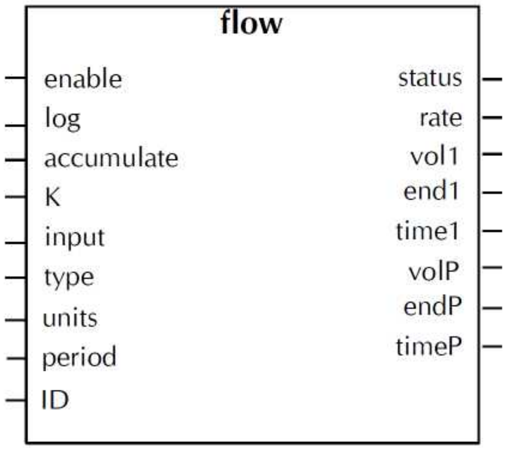
When the accumulator enabled input is ON, accumulation and rate calculations are enabled. When the input is OFF, accumulators, stored data and the outputs are set to zero.

The function reads and accumulates the number of pulses, and divides by the K factor to calculate the total volume. This is done on each scan of the controller logic. The function calculates the flow rate in engineering units based on the K-factor provided. The rate updates once per second if there is sufficient flow. If the flow is insufficient, the update slows to as little as once every ten seconds to maintain resolution of the calculated rate.

Stored data is retained when the program is stopped, and when the controller is powered off or reset.

32 ***flow*** function blocks can be added to the controller. These are shared between the two ISaGRAF kernels. The same ID cannot be used in both kernels.

The flow accumulator function block ***flow*** appears as follows.



Arguments

Inputs	Type	Description

enable	Boolean	TRUE = enable accumulation
		FALSE = clear outputs

log	Boolean	Log Data FALSE to TRUE transition = log data
accumulate	Boolean	Accumulate flow TRUE = accumulate flow
K	Real	K factor F needs to be a positive value.
input	Integer	Pulse Input
type	Integer	Input type 0 = 32 bit running counter 1 = 32 bit difference
units	Integer	Rate period 0 = seconds 1 = minutes 2 = hours 3 = days
period	Integer	Specified record to be displayed (1 to 35).
ID	Integer	Identifies the internal flow accumulator (1 to 32).

Outputs	**Type**	**Description**
status	Integer	Status 0 = normal 1 = invalid **K** input 2 = invalid **Period** input 3 = invalid **Type** input 4 = invalid **Units** input 5 = pulse rate is too low for accurate flow rate calculation 6 = invalid **ID**
rate	Real	Flow rate
vol1	Real	Volume for period 1
end1	Integer	Time at end of period 1 Seconds from January 1, 1970
time1	Integer	Flow time for period 1 Time in seconds
volP	Real	Volume for specified **period**
endP	Integer	Time at end of specified **period** Seconds from January 1, 1970
timeP	Integer	Flow time in specified **period** Time in seconds

The ***flow*** function block stored data is retained when the program is stopped, power is cycled or the controller is reset. The stored data is cleared when the controller is initialized or when the enable input is OFF.

The maximum pulse rate is 10 kHz. Measuring the accumulated pulses and dividing by the K factor calculates the accumulated flow. The K factor cannot be changed while an accumulation is in progress. Only change the K factor while accumulation is stopped.

5.7.5 total

total

Non-Volatile Totalizer

Description

The ***total*** function block reads a rate input and integrates it over the sample interval to accumulate a total. Up to 32 total function blocks can be added to a program.

When the accumulate input is ON the function accumulates the total. The time1 and timeP outputs report the totals.

When the enable input is ON, accumulation is enabled. When the enable input is OFF, the accumulated values and the logged data are deleted.

When the log input goes from OFF to ON, the accumulated total is saved in the history records. Older history is pushed down and the oldest record is discarded.

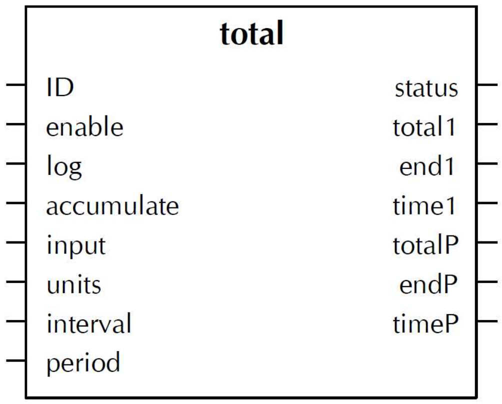
Stored data is retained when the program is stopped, and when the controller is powered off or reset.

The accumulated total does not change if the period is changed.

The accumulated total is set to zero if the period is invalid.

32 ***total*** function blocks can be added to the controller. These are shared between the two ISaGRAF kernels. The same ID cannot be used in both kernels.

The ***total*** function block appears as follows.



Arguments

Inputs	Type	Description
ID	Integer	Identifies the internal totalizer (1 to 32).
enable	Boolean	Enables accumulation and creates log. FALSE to TRUE transition = create and initialize log TRUE = enable accumulation FALSE = delete log and clear outputs
log	Boolean	Log Data FALSE to TRUE = log data

accumulate	Boolean	Accumulate flow ON = accumulate flow
input	Real	Rate Input
units	Integer	Input rate period 0 = seconds 1 = minutes 2 = hours 3 = days
interval	Integer	The interval specifies the sample interval at which the rate input will be sampled. A sample is taken and a calculation performed if the time since the last sample is greater than or equal to the sample interval. Valid values are 1 to 65535 tenths of a second. The period over which readings are taken will vary, but will stay in the range –1 * (Expected Interval + Configured Sample Interval + Maximum time between ladder logic scans) <= **Actual Sampling Interval** <= +1 * (Expected Interval + Configured Sample Interval + Maximum time between ladder logic scans). As a result the total for individual periods may fluctuate despite a constant input.
period	Integer	Specified record to be displayed (1 to 35). The accumulated total does not change if the period is changed. The accumulated total is set to zero if the period is invalid.

Outputs	Type	Description
status	Integer	Status 0 = normal operation 1 = invalid **ID** 2 = invalid **Units** input 3 = invalid **Interval** input 4 = invalid **Period** input
total1	Real	Total for period 1
end1	Integer	Time at end of period 1 Seconds from January 1, 1970
time1	Integer	Accumulation time for period 1 Time in seconds
totalP	Real	Total for specified **period**
endP	Integer	Time at end of specified **period** Seconds from January 1, 1970
timeP	Integer	Accumulation time in specified **period** Time in seconds

Notes

The ***total*** function block stored data is retained when the program is stopped, power is cycled or the controller is reset. The stored data is cleared when the controller is initialized or when the enable input is OFF.

The accumulated value is a floating-point number. Floating-point numbers are approximations. If the accumulated value grows large, then low rate inputs will have little or no effect on the accumulated value and the accumulated value will not be accurate. Use the log input to save the accumulated value and start a new accumulation when the accumulated value grows large.

The log input needs to be triggered at a suitable rate or the accumulator will overflow, and the accumulated value will not be accurate.

5.7.6 rea_msg

Convert a Real variable to a Message variable

Description

The standard ISaGRAF library provides a “MSG” conversion function for converting between ISaGRAF variable types. The standard function, however, truncates real (float) values before converting to a message string. I.e. only the integer portion is converted to a message string. The *rea_msg* function converts the integer and fractional portions of the real (float) value to a message string.

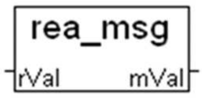


Figure 6.51: rea_msg Function

INPUTS	TYPE	DESCRIPTION
rVal	Integer	Read (floating point) value

OUTPUTS	TYPE	DESCRIPTION
mVal	Message	Message string containing converted value

5.7.7 gen_evt

Generate an event for a given point

Description

This function block provides a mechanism to force point events on appropriately configured configuration points. Typically events are generated for points on significant value changes, with the associated timestamp reporting the actual time of the value change. This mechanism allows events to be generated (forced) on the specified points as required. The value included in the forced event will be the current point database value of the point in question.

The *gen_evt* function block includes a *time* input parameter which allows a timestamp to be included in the forced DNP3 event. The *time* input parameter represents the "*number of seconds since 1970*". This *time* input parameter needs to coincide with how the RTU's real time clock is set, i.e. either UTC or Standard Time format. (This value can be obtained from the *os_time* function block as detailed in Section ***os_time*** 78). If a zero value is entered for the *time* input parameter, the RTU's operating system will timestamp the forced DNP3 event.

Events generated using the *gen_evt* function blocks are inserted into the DNP3 event buffer as ***Buffered*** events.

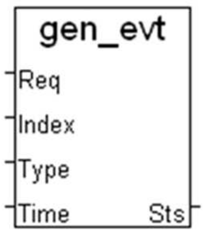


Figure 6.53: gen_evt Function Block

Hint:	If ISaGRAF is to be used to write value changes to the specified point **before** forcing the event use the DNP3 communication function blocks to to update the point values (where the *ObjectType* input is set to *Local_RTU_Data),* as opposed to using ISaGRAF output boards. The operating system will then process the value update **before** the *"force event"* request.

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Force events request. This invokes a request to force a event on the specified point when asserted (rising edge).
Index	Integer	This input specifies the point number of the configuration point.
Type	Integer	This input specifies the point type of the configuration point. Valid values for the **Type** input are listed as follows DIN or 1 (Digital Input) AIN or 3 (Analog Input) CIN or 5 (Counter Input)

Time	Integer	This input specifies the timestamp to be used in the forced point event. This input represents the number of seconds since January 1st, 1970 and must be specified as UTC time. If a 0 value is entered, the RTU's operating system will timestamp the event with the current RTU time.

OUTPUTS	TYPE	DESCRIPTION
Sts	Integer	Data transfer confirm indicates completion of request 0 = Success 1 = Point does not exist 2 = Bad Point Type -1 = Unsuccessful

NOTES:

In order for the *gen_evt* function block to successfully force events on the specified point (i.e. the configuration point specified by the ***Index*** and ***Type*** input parameters), the ***Point Data Class*** attribute of the configuration point needs to be configured as follows.

- Point Data Class : Set this attribute to Class 1, Class 2 or Class 3.
- Alarm Inhibit: The event will be forced on the point irrespective of the state of the *Alarm Inhibit* attribute. Therefore if events are only to be generated on the specified point using the *gen_evt* function block, then set the *Alarm Inhibit* attribute to YES which will stop normal value changes from generating events on the specified point.

5.7.8 genmsevt

Generate an event for a given point with millisecond time input

Description

This function block provides a mechanism to force point events on appropriately configured configuration points. Typically events are generated for points on significant value changes, with the associated timestamp reporting the actual time of the value change. This mechanism allows events to be generated (forced) on the specified points as required. The value included in the forced event will be the current point database value of the point in question.

The *genmsevt* function block includes *timeSec* and a *msTime* input parameters which allows a timestamp to be included in the forced DNP3 event.

The *TimeSec* input parameter represents the "*number of seconds since 1970*". This input parameter needs to coincide with how the RTU's real time clock is set, i.e. either UTC or Standard Time format. (This value can be obtained from the *os_time* function block as detailed in Section ***os_time*** 78).

- If a zero value is entered for the *TimeSec* input parameter, the RTU's operating system will timestamp the forced DNP3 event.
- If a negative value is entered for the TimeSec input parameter the forced DNP3 event is time stamped with the maximum supported time value: 19-JAN-2038 03:14:08.999.

The *msTime* input is the millisecond portion of the timestamp to be included in the forced DNP3 event. This parameter allows a more precise DNP3 timestamp to be generated. Values of 0 to 999 are valid.

The events generated using the *genmsevt* function blocks are inserted into the DNP3 event buffer as ***Buffered*** events.

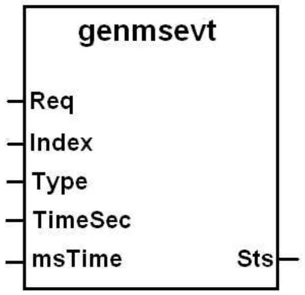


Figure 6.54: genmsevt Function Block

Hint:	If ISaGRAF is to be used to write value changes to the specified point **before** forcing the event, check that these point value updates are carried out using the DNP3 communication function blocks (where the *ObjectType* input is set to *Local_RTU_Data),* as opposed to using ISaGRAF output boards. The operating system will then process the value update **before** the *"force event"* request.

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Force events request. This invokes a request to force a event on the specified point when asserted (rising edge).

Index	Integer	This input specifies the point number of the configuration point.
Type	Integer	This input specifies the point type of the configuration point. Valid values for the **Type** input are listed as follows DIN or 1 (Digital Input) AIN or 3 (Analog Input) CIN or 5 (Counter Input)
TimeSec	Integer	This input specifies the timestamp to be used in the forced point event. This input represents the number of seconds since January 1st, 1970 and must be specified as UTC time. If a 0 value is entered, the RTU's operating system will timestamp the event with the current RTU time and the msTime input is ignored.
msTime	Integer	This input is the millisecond portion of the timestamp to be included in the forced DNP3 event. This parameter allows a more precise DNP3 timestamp to be generated. Values of 0 to 999 are valid. This input is ignored if the TimeSec input is set to 0.

OUTPUTS	TYPE	DESCRIPTION
Sts	Integer	Data transfer confirm indicates completion of request 0 = Success 1 = Point does not exist 2 = Bad Point Type 4 = Invalid ms Time -1 = Unsuccessful

NOTES:

In order for the *genmsevt* function block to successfully force events on the specified point (i.e. the configuration point specified by the ***Index*** and ***Type*** input parameters), the ***Point Data Class*** attribute of the configuration point needs to be configured as follows.

- Point Data Class : Set this attribute to Class 1, Class 2 or Class 3.
- Alarm Inhibit: The event will be forced on the point irrespective of the state of the *Alarm Inhibit* attribute. Therefore if events are only to be generated on the specified point using the *genmsevt* function block, then set the *Alarm Inhibit* attribute to YES which will stop normal value changes from generating events on the specified point.

5.7.9 ana_time

Read analog point float value with timestamp

Description

This function block provides an analog point's current engineering value, and the relevant timestamp corresponding to the reported value. In order to provide an accurate timestamp for the point, the RTU can record a value and timestamp for a point when a DNP3 event is generated. This information is then available to ISaGRAF through this function block.

The function block can be operated in one of 2 modes. The first mode ignores point events and presents current floating point value and current time. The second mode uses pointFloat Events to obtain greater accuracy timestamps. This second mode requires that the point in question be appropriately configured to generate point float events for any change of value as described at the end of this subsection.

If multiple point events are generated between ISaGRAF scans, only the recent value and timestamp will be available to the ISaGRAF function block. If the point is not configured as detailed below, the timestamp presented shall be the time at which the function block outputs are updated.

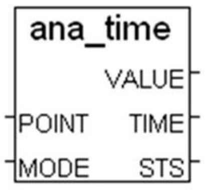


Figure 6.52: ana_time Function

INPUTS	TYPE	DESCRIPTION
Point	Integer	specifies the RTU point number of the analog point.
Mode	Integer	0 = MODE_RTC: The timestamps presented represent the time at which the function block outputs are updated. Using this mode along with the elevated ISaGRAF task priority (see Section ***rtuparam*** 152) can result in relatively constant time intervals between timestamps. The that point events are not used to derive timestamps in this mode. 1 = MODE_EVT: Point float event timestamps are used to update the function block timestamp output. In the absence of any point events, the timestamp presented, is the time at which the function block outputs are updated (same as for mode MODE_RTC). This mode produces timestamps of greater accuracy though the interval between timestamps can vary significantly.

OUTPUTS	TYPE	DESCRIPTION
Value	Real	The engineering value of the point. ISaGRAF checks to see if any events for the point have been generated since the last scan, and if

		so the recent event will provide the value to the function block output. If no events have generated since the last scan, the current value will be retrieved from the RTU point database using the points CURRENT ENGINEERING VALUE property. Message string containing converted value.
Time	Timer	The time-stamp at which the *current value* occurred (milliseconds since midnight). ISaGRAF checks to see if any events fro the point have been generated since the last scan, and if so the recent event will provide the timestamp to the function block output. If no events have generated since the last scan, the current time (as milliseconds since midnight) will be presented as the function block timestamp output. Note that the time is presented as Standard Time without local time or daylight savings correction.
Sts	Integer	Function block status values are indicated as follows: 0 = Success 1 = Point does not exist 6 = Invalid argument to function block (e.g. incorrect mode) -1 = Unknown error

NOTES:
In order for the ANA_TIME function block to provide accurate time stamping in MODE_EVT mode, it is necessary that the analog point be correctly configured to generate events as follows (refer to the *SCADAPack E Configurator User Manual* for further details):

- **Point Data Class:** Set this attribute to Class 1, Class 2 or Class 3.
- **DNP Static Object Type**: Set this attribute to Object 30 variation 5 or Object 40 variation 3.
- **Event Deviation** : Set to 0% so that an event is generated for ANY value change
- **Alarm Inhibit**: Set to NO.

5.7.10 cmd_exec

Execute an RTU Command

Description

The CMD_EXEC function block executes a subset of the RTU's command line functions. The command to execute is passed into the function block as a string and executed by the RTU when requested.

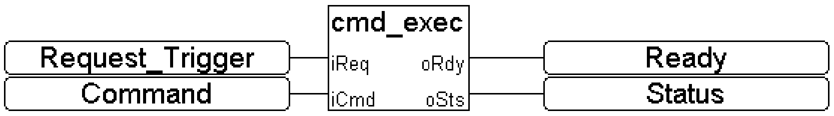


Figure 4.1: cmd_exec Function Block

Inputs:

Name	**ISaGRAF data type**	**Notes**
iReq	Boolean	Command Execute request: initiate command when asserted (rising edge)
iCmd	Message	Command to be Executed (See below for acceptable command list)

Outputs:

Name	**ISaGRAF data type**	**Notes**
oRdy	Boolean	Command execute confirm: indicates completion of request
oSts	Analog	Transaction status value: Success = 0 Incomplete Command = 1 Command Not Found = 2 Command Failed = 3

Restarting services will impact process control and RTU availability. Assess the impact prior to performing a restart operation.

WARNING

UNEXPECTED EQUIPMENT OPERATION

Evaluate the operational state of the equipment monitored and controlled by the SCADAPack E RTU prior to restarting services.

Failure to follow these instructions can result in death, serious injury or equipment damage.

The following commands will be accepted by the CMD_EXEC Function Block. For detailed information on these commands, refer to the ***SCADAPack E Operational Reference Manual,*** *Section Command Line & Diagnostics*.

APPEND, JOIN	Append file to end of another
CLEAR	Clear specified RTU information
COPY	Duplicate a file
DEL, DELETE	Delete a file
REN, RENAME	Rename a file
RESTART	Restart RTU facilities
GETCONFIG	Generate an RTU configuration file
FILEDIAG	Enable/Disable File Diagnostics
SYSDIAG	Enable/Disable System File Diagnostics
PLCDIAG	Enable/Disable PLC File Diagnostics

5.7.11 rtuparam

Modify a specific RTU parameter

Description

This function block provides an ISaGRAF application with the ability to modify RTU operational parameters.

The RTUPARAM function block operates in a similar way to the ISaGRAF peer communication function blocks described above, with respect to the REQ, CNF, RDY and STATUS parameters.

Changes to RTU operating parameters made with this function block are not permanently stored in the SCADAPack E RTUs Non-Volatile memory (NV-RAM).

When the SCADAPack E RTU is reset (and/or Warm reset in the case of DNP parameters), the parameter values revert to the previous value that was stored in NV-RAM when the RTU was configured.

Some parameters accept a value of "-1" (as indicated in the table below). Using this value reverts the parameter to the value that was stored in NV-RAM when the RTU was configured. The ISaGRAF ***common.eqv*** global define "USE_RTU_DEFAULT" is also provided for this purpose.

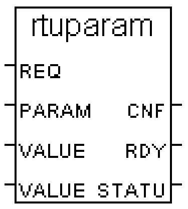


Figure 6.47: rtuparam Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data transfer request initiated on rising edge.

Param	Integer	Name of parameter to modify
Value1	Integer	Parameter dependent value. See ***Table 6.11*** 154
Value2	Integer	Parameter dependent value. See ***Table 6.11*** 154

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Rdy	Boolean	Data transferred successfully.
Status	Integer	0 for success when Cnf and Rdy are TRUE. Otherwise see error code in ***Table 6.11a*** 158.

Table 6.11: PARAM and VALUE fields for RTUPARAM

PARAM	PARAM Value*	Value1	Value2	Comment
MASTER_COMPORT	1	0 = Port0 1 = Port1 2 = Port2 3 = Port3 4 = Port4 5 = Ethernet 1 6 = Ethernet 2 10 = Port5 11 = Port6 12 = Port7 13 = Port8 -1 = return to configured value	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	Changes RTU communication port to which unsolicited messages are directed, for the appropriate DNP3 Master station
UNSOL_TX_DELAY	2	Secs -1 = return to configured value	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	Changes Unsolicited transmission delay to the appropriate DNP3 Master session
APPL_TO	3	Secs -1 = return to configured value		Changes DNP application layer timeout used for default Peer and Data Concentrator requests
COLD_RESTART	4			Restarts the RTU
APPL_EVENT_TO	5	Secs -1 = return to configured value	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	Changes DNP event confirm timeout to the appropriate DNP3 Master session.
MY_DNP_ADDRESS	6	New DNP Address = 0 to 65529 -1 = return to configured value		Changes the following RTU DNP3 addresses: • Slave address when communicating with Master 1 • DNP address when sending Peer requests • DNP address when sending

PARAM	PARAM Value*	Value1	Value2	Comment
				Data Concentrator requests Changes address "on-line" without a DNP3 Warm Restart or RTU restart.
ETH_IP_ADDRESS_1	7	Numeric Value of the new TCP/IP address** -1 = return to configured value		Changes the TCP/IP address for Ethernet Port 1 "on-line" without an RTU restart.
ETH_IP_ADDRESS_2 (for RTU models supporting dual Ethernet interfaces)	8	Numeric Value of the new TCP/IP address** -1 = return to configured value		Changes the TCP/IP address for Ethernet Port 2 "on-line" without an RTU restart.
UNSOL_CLASSES	9	No Classes = 0 Class 1,2,3 = 14 Class 1 = 2 Class 2 = 4 Class 3 = 8 Class 1,2 = 6 Class 1,3 = 10 Class 2,3 = 12 -1 = return to configured value	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	Changes the RTU's DNP enabled Unsolicited Event Classes "on-line" without a restart. This functionality may be useful in a dual-redundancy change-over situation
SYS_ERR_CODE	10	100-999		User error code number sent to RTU Error Code system analog point 50020
ISA_TASK_PRI	11	0 = Normal Priority 1 = High_Priority		Increases this ISaGRAF Kernel task's priority in the RTU operating system. Requires that the ISaGRAF application be configured with "Trigger cycles" in the ISaGRAF Application Run Time Options' Cycle Timing settings. Also requires the ISaGRAF application actually scans faster than the configured Cycle Timing.
DISCONNECT_PORT	12	0 = Port0 1 = Port1 2 = Port2		Requests that the serial port driver task associated with the supplied port number

PARAM	PARAM Value*	Value1	Value2	Comment
		3 = Port3 4 = Port4		disconnects its current connection (if it has one). Valid for PPP, GPRS, 1xRTT and Hayes Modem port modes only.
UNSOL_ALLOWED	13	0 = No Unsolicited Allowed 1 = Unsolicited Allowed -1 = return to configured value	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	Controls Unsolicited Response transmissions to the appropriate DNP3 Master session.
DISABLE_MASTER	14	0 = Enable Master 1 = Disable Master	0 = Master 1 1 = Master 1 2 = Master 2 3 = Master 3	A disabled Master session does not respond to any DNP3 messages from that master, nor generate any Unsolicited Responses until it is enabled again by the ISaGRAF application. May be used in conjunction with system points 63420,63421,63422 to set Master(s) to a disabled startup state, then later enable the Master(s) with this parameter
DISABLE_DCONS	15	0 = Data Concentrator Enabled 1 = Data Concentrator Disabled		When disabled, the DNP3 Data Concentrator does not generate any periodic polls (forced polls are ok) or confirm any Unsolicited responses from IEDs. #
PORT_CONF_MODE	16	0=Never 1=Sometimes 2=Always -1 = return to configured value	0 = Port0 1 = Port1 2 = Port2 3 = Port3 4 = Port4 5 = Ethernet 1 6 = Ethernet 2 10 = Port5 11 = Port6 12 = Port7 13 = Port8	Changes the link layer confirm mode of a serial or Ethernet port to the mode specified. See the *SCADAPack E DNP3 Technical Reference* for a description of these modes.
RESTART_SERVICE	17	1 = History		Appends trend sampler files to file called history. Refer to the Trend Sampler Manual for more information on “restart history”.

PARAM	PARAM Value*	Value1	Value2	Comment
		2 = Sampler		Restarts Sampler task. Refer to the *SCADAPack E Trend Sampler Reference* manual for more information on “restart sampler”.
		3 = Profile		Restarts the Profile task
		4 = TCP_Service		Restarts the TCP Service task (TCP serial port service).
		5 = Mask		Clears the system error code, reset reasons mask, and task watchdog mask system points
DISABLE_PORT	18	0 = Port0 / USB++ 1 = Port1 2 = Port2 3 = Port3 4 = Port4 5 = Ethernet 1 6 = Ethernet 2 10 = Port5 11 = Port6 12 = Port7 13 = Port8	0 = Enable Port 1 = Disable Port	Disables or Enables the operation of the specified RTU port number. **Note:** Active PSTN / GSM or TCP/IP connections are not automatically closed on the port. You can use the DISCONNECT_PORT parameter (described above) to close a connection prior to disabling the port if desired.

* The PARAM name are defined in Schneider Electric ***common.eqv*** file (global defines).

** The numeric value of a TCP/IP address may be obtained from the SCADAPack E ISaGRAF Function, “MSG_IP”. See ***msg_ip***171 function.

++ On SCADAPack 300E RTUs, PORT_DISABLE on Port0 disables the USB Peripheral Port on the RTU. This disables USB peripheral port activities. If a device is connected to the USB peripheral port either prior to, or after the port is disabled, the device will not be able to communicate when the port is enabled. The device needs to be disconnected and reconnected after the port is enabled to resume communication.

# If the Data Concentrator becomes disabled due to an ISaGRAF RTUPARAM request, local mapped points are not marked as “I/O Not responding”, as would normally be the case for points mapped to a non-responding IED. The “Data Concentrator Ready” (Binary System Point 50269) state is set to False.

When the Data Concentrator is re-enabled by ISaGRAF request, IED’s are marked internally as IED_NOT_FOUND. Their state will then be changed to IED_RUNNING or IED_COMMS_FAILED depending upon the result of the initial poll response.

When the Data Concentrator restart is complete, the “Data Concentrator Ready” (Binary System Point 50269) state will change to True.

Table 6.11a: RTUPARAM Status Codes

RTUPARAM STATUS	DESCRIPTION
-1	Unknown error
0	Operation successful
1	Information not found
2	Bad Point Type
3	Unknown attribute for this point
4	Bad value for this attribute
5	Invalid attribute for this point
6	Invalid argument

5.7.12 chgroute

Modify entries in the SCADAPack E RTU DNP3 Routing Table

Description

This function block provides an ISaGRAF application with the ability to modify entries in the SCADAPack E RTU DNP3 Routing Table. The *chgroute* function block operates in a similar way to the ISaGRAF peer communication function blocks described above, though the parameters that are modified are in the local RTU. The parameter meanings are as follows:

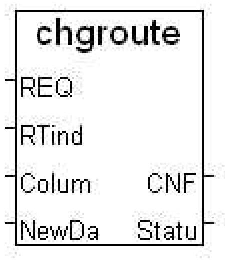


Figure 6.48: chgroute Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data transfer request initiated on rising edge.
RTindex	Integer	Route Table Row Index (0-49). The RTindex parameter defines the Route Table row in which the parameter is to be changed. The route table rows are configured by SCADAPack E Configurator and are indicated as Rows 1 to 50. Corresponding values for Rtindex are 0-49.
Column	Integer	Route Table Column number (0-7). See ***Table 6.12*** 160
NewData	Integer	New value for route table entry.

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Status	Integer	0 for success when Cnf is TRUE. Otherwise see **Status Codes** 99

Table 6.12: Column Parameters for chgroute

Column Value	DNP Route Table Column	Valid Column Values	Comment
0	Source Port	0 = Port0 / USB 1 = Port1 2 = Port2 3 = Port3 4 = Port4 5 = Ethernet1 6 = Ethernet2 10 = Port5 11 = Port6 12 = Port7 13 = Port8 254 = Any 255 = Table End	Source port of DNP3 message to be routed
1	Source Start	0-65535	Source DNP node Address Range
2	Source End	0-65535	
3	Dest Start	0-65535	Destination DNP node Address Range
4	Dest End	0-65535	
5	Dest Port	0 = Port0 / USB 1 = Port1 2 = Port2 3 = Port3 4 = Port4 5 = Ethernet1 6 = Ethernet2 10 = Port5 11 = Port6 12 = Port7 13 = Port8	Destination port to route DNP3 message to
6	Status	0 = Offline Static 1 = Online Static 256 = Offline Dynamic 257 = Online Dynamic 512 = Offline Fixed 513 = Online Fixed	Route Type
7	Lifetime	0-32767 secs	Lifetime of Dynamic Route for automatic

			status change from Online to Offline

5.7.13 chgrtnum

Change DNP3 Routing Table Connect Number String

Description

This function block provides an ISaGRAF application with the ability to modify the "*Connect No.*" string in the DNP3 Routing Table of the SCADAPack E RTU. This can change a PSTN or GSM Dial number string, IP address string, etc.

The *chgrtnum* function block operates in a similar way to the ISaGRAF *chgroute* function block described above.

This function block, though, takes a DNP Destination address as a parameter, and searches the route table for an entry whose destination address range includes the function block parameter value.

The first matching entry that is found has its *Connect No.* field updated by the function block DIAL parameter value (string).

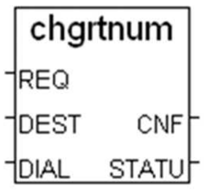


Figure 6.49: chgrtnum Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data transfer request initiated on rising edge.
Dest	Integer	Destination DNP node address to search (0-65535)
Dial	Message	Connect No. (message string) to transfer to the matching route entry. E.g. Dial number, IP address

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Status	Integer	0 for success when Cnf is TRUE. Otherwise see **Status Codes** 99

5.7.14 chgrtprt

Change DNP3 Routing Table port number

Description

This function block provides an ISaGRAF application with the ability to modify the destination port number in a SCADAPack E RTU DNP3 Routing Table. The CHGRTPRT function block operates in a similar way to the ISaGRAF CHGROUTE function block described above. This function block, though, takes a DNP Destination address as a parameter, and searches the route table for an entry whose destination address range includes the function block parameter value. The first matching entry that is found has its *Destination Port* updated by the function block PORT parameter value (integer analog).

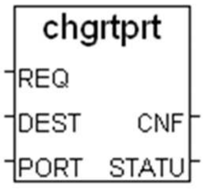


Figure 6.50: chgrtprt Function Block

INPUTS	TYPE	DESCRIPTION
Req	Boolean	Data transfer request initiated on rising edge
Dest	Integer	Destination DNP node address to search (0-65535)
Port	Integer	Destination Port number to transfer to the matching route entry. Port number values for this parameter are: 0=Port0 / USB 1=Port1 2=Port2 3=Port3 4=Port4 5=Ethernet1 6=Ethernet2 10=Port5 11=Port6 12=Port7 13=Port8

OUTPUTS	TYPE	DESCRIPTION
Cnf	Boolean	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise
Status	Integer	0 for success when Cnf is TRUE. Otherwise see **Status Codes** 99

5.8 TCP/IP Interface Functions

These function blocks provide interfaces to RTU TCP/IP communication and configuration services. For more information on IP routing with the SCADAPack E RTU, see the SCADAPack E *TCP/IP Technical Reference* manual:

- ***ip_add*** 165
- ***ip_del*** 167
- ***ip_cycgw*** 169
- ***ip_ping*** 170
- ***msg_ip*** 171
- ***ppp_echo*** 172

5.8.1 ip_add

Add an IP Routing Table Entry

Description

This function provides an interfaces to the TCP/IP facilities in the SCADAPack E RTU for adding an entry to the IP Routing Table. The *ip_add* function adds a routing entry to the IP Routing Table. A route entry access counter is incremented each time that the function is executed for this route. An *ip_del* function decrements the access counter and removes the route entry when the access counter is 0. Execute the *ip_add* function when the route is first added, only. Depending on the input parameter settings, add one of the following route entry types, listed in order of descending implementation:

- DEFAULT GATEWAY
- GATEWAY Route
- NETWORK Route
- HOST Route

The Function Block Interface and the associated Port Type assignments are shown in the table below:

Function Block Interface Number	Port Type
0	PPP on Serial Port 0
1	PPP on Serial Port 1
2	PPP on Serial Port 2
3	PPP on Serial Port 3
4	PPP on Serial Port 4
10	Ethernet 1
11	Ethernet 2

Route Entries added using this function are not preserved in NV RAM so are not retained if an RTU is restarted or powered off.

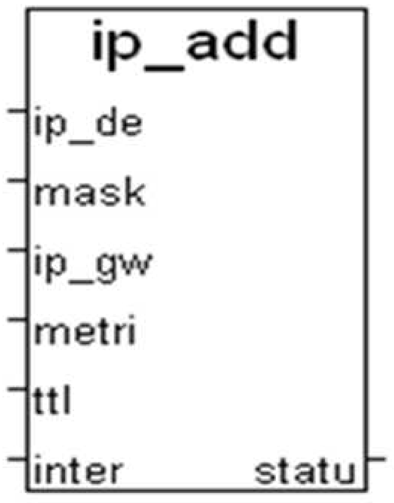


Figure 6.57: ip_add Function

INPUTS	TYPE	DESCRIPTION
Ip_dest	Message	Destination IP Address for route
Mask	Message	Destination Subnet Mask for route
IP_gw	Message	Gateway host IP Address for route
Metric	Integer	route metric (cost) for added route. (0 = use default interface metric)
TTL	Timer	Time to Live route entry. (t#0 = static route: no lifetime)
Interface	Integer	Number of IP interface (E.g. 0=Port0, 1=Port1, etc.).

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code when adding to IP Table. 0 = success; For other status and non-execution codes, refer to *SCADAPack E RTU TCP/IP Errors* (*Codes and Descriptions*) in the *TCP/IP Technical Manual*.

Further information about IP Route Types can be found in *IP Routing* (*IP Route Types*) in the *TCP/IP Technical Manual* and in *TCP/IP Folder* (*Advanced TCP/IP Property Page*) in the *Configurator User Manual*.

5.8.2 ip_del

Delete an IP Routing table entry

Description

This function interfaces to the TCP/IP facilities in the SCADAPack E RTU for deleting an entry to the IP Routing Table. The IP_DEL function removes a routing entry to the IP Routing Table. The route entry's access counter is decremented each time that the function is executed and the route entry is deleted when the access counter reaches 0. It is advised to execute the function only when it is required to delete a route.

The specified IP address and mask information needs to match those used to previously to add an entry to the route table. I.e. via static routes in the RTU's configuration, command line ROUTE ADD command or ISaGRAF's ip_add function.

Configured static route entries previously preserved in NV RAM are not permanently removed by the *ip_del* function. The permanent route entries will be restored upon an RTU restart or power on.

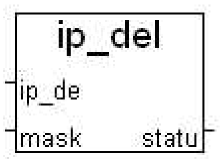


Figure 6.58: ip_del Function

INPUTS	TYPE	DESCRIPTION
Ip_dest	Message	Destination IP Address for route
Mask	Message	Destination Subnet Mask for route

OUTPUTS	TYPE	DESCRIPTION
Status	Intege	Indicates status code when adding to IP Table. 0 = success OR EINVAL: 2019 = An invalid parameter was provided to a TCP/IP function or function block. For example, if you define an IP address in single quotes (e.

OUTPUTS	TYPE	DESCRIPTION
		g., '172.168.1.100'), this syntax is invalid. Check the function or function block inputs. OR ERTNOTFOUND: 2036 = Routing table entry not found. This status is reported when the requested entry is not in the Routing Table.

5.8.3 ip_cycgw

Cycle default IP gateway route

Description

This function interfaces to the TCP/IP facilities in the SCADAPack E RTU for cycling default gateway entries in the IP Routing Table. The IP_CYCGW function cycles between DEFAULT GATEWAY route entries in the IP Routing Table. This is only applicable where multiple DEFAULT GATEWAY entries have been added to the IP Routing Table.

This operation of this function has no effect if there are no DEFAULT GATEWAY entries in the IP Routing Table, or if there is only a single DEFAULT GATEWAY entry.

The first DEFAULT GATEWAY entry found in the routing table is the active default gateway. This entry will be the initial active default gateway whenever the RTU is restarted or powered on.

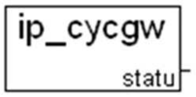


Figure 6.59: ip_cycgw Function

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code when cycling between default gateway IP routes. 0 = success

5.8.4 ip_ping

Ping a remote IP node

Description

This function block interfaces to the TCP/IP PING Client facilities in the SCADAPack E RTU. The *ip_ping* function block sends an ICMP ECHO request to the IP host specified by the *ip_dest* parameter. This tests IP (Network Layer) operations on the remote host.

It is suggested that the i*p_ping* function block could be used by an application executing in the second ISaGRAF Target Kernel, so as not to slow the performance of a main control application executing in the first ISaGRAF Target Kernel.

This function block executes synchronously with the ISaGRAF execution scan, and may significantly increase the ISaGRAF application scan rate, particularly if the remote IP hose does not reply to the PING request.

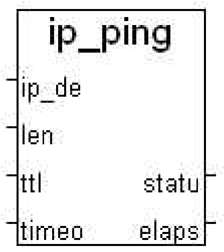


Figure 6.60: ip_ping Function Block

INPUTS	TYPE	DESCRIPTION
Ip_dest	Message	Destination IP address of remote host
Len	Integer	Number of bytes to send in ping request. (needs to be less than 1500)
TTL	Timer	Time To Live for ping packet on IP network. (t#0 = default TTL)
Timeout	Tmer	Time to wait for remote IP host to respond

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code for Ping request. 0 = success
Elapsed	Timer	Elapsed time between Ping Response and Ping Request. Valid when STATUS = 0

5.8.5 msg_ip

Convert a message type string to an IP address

Description

This function converts an ISaGRAF Message type containing a TCP/IP address in dotted decimal format into a numeric value.

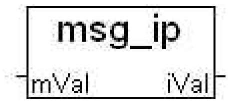


Figure 6.61: msg_ip Function

INPUTS	TYPE	DESCRIPTION
mVal	Message	TCP/IP address as a string in dotted decimal format.

OUTPUTS	TYPE	DESCRIPTION
iVal	Integer	TCP/IP address as an integer value.

Structured text example:

```
rtuparam_2 (FALSE, ETH_IP_ADDRESS_1, 0, 0);
IF (r_trig2.Q = TRUE) THEN
        rtuparam_2 (TRUE, ETH_IP_ADDRESS_1, msg_ip ('192.168.0.218'), 0);
END_IF;
```

5.8.6 ppp_echo

Check the status of a PPP link

Description

This function interfaces to the TCP/IP facilities for querying the status of a PPP link on a SCADAPack E RTU. The *ppp_echo* function sends an LCP ECHO command to the nominated PPP interface on an SCADAPack E RTU.

A successful status (0) indicates PPP link “UP” state on the serial interface.

PPP_ECHOmay return a timed-out error (Status = 2040) on a busy PPP link. Therefore it is advisable to check the link a few times after a timeout before being confident that connection on the PPP link is lost.

For more information on using PPP with the SCADAPack E RTU, see the *SCADAPack E TCP/IP Technical Reference* Manual.

This function executes synchronously with the ISaGRAF execution scan, and may significantly increase the ISaGRAF application scan rate, particularly if the peer PPP device does not reply to the LCP ECHO request.

It is suggested that the *ppp_echo* function could be used by an application executing in the second ISaGRAF Target Kernel, so as not to slow the performance of a main control application executing in the first ISaGRAF Target Kernel.

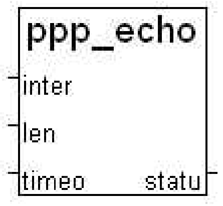


Figure 6.62: ppp_echo Function

INPUTS	TYPE	DESCRIPTION
Interface	Integer	Number of PPP port IP interface. (E.g. 0=Port0, 1=Port1, etc.)
Len	Integer	Number of bytes to send in LCP Echo. Needs to be less than 500.
Timeout	Timer	Time to wait for the peer PPP device to respond

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code for PPP ECHO operation. 0 = success

5.9 Alarm Group Functions & Function Block

The ISaGRAF ‘Summary Alarm’ functions and function blocks available in the SCADAPack E RTU provide a mechanism for grouping individual RTU Digital point states (alarms) in to a named alarm group, then provide a summary alarm output if any of the points in the group transition in to an alarm state. There are three aspects of the summary alarm functions that are provided through user interfaces:

- Configuration of points within a named alarm group
- Processing of the alarm group
- Transferring and loading the point “mask” values externally through RTU points.

These interfaces are handled through independent ISaGRAF functions and function blocks. As detailed below, the point numbers registered for alarm summaries need not necessarily have to be imported in to the ISaGRAF application through an input board.   However, if they are attached to I/O boards as ISaGRAF input variables, translation to the DNP3 point number can be provided for the point in the RTU database. The Schneider Electric ISaGRAF pre-processor can automatically build an ISaGRAF “Defines” list based on variables attached to RTU point I/O boards. These defines can then be used as named point numbers. See ISaGRAF 3 Pre-Processor for details.

- ***almadd*** 174
- ***almproc*** 176
- ***almload*** 179
- ***almclr*** 180

The SCADAPack E Data Processor sub-system supports the concept of point alarms, as a standard feature. Each binary point within the RTU database has a “Point-In-Alarm” property which is processed through these Summary Alarm interfaces.

For information on RTU alarm processing see the *SCADAPack E Data Processing Technical Reference* manual.

5.9.1 almadd

Add a point to an alarm group

Description

Creating and configuring the Alarm Group is performed by a call from ISaGRAF User Application code to the RTU ISaGRAF *almadd* Function.

Typically the ISaGRAF function calls to configure an alarm group are executed during the start-up phase of an ISaGRAF application. Once executed at ISaGRAF application startup, this code no longer needs to be executed while the ISaGRAF application is running. Where an alarm group identified by the string passed to GRPNAME does not exist, it will be created by the *almadd* function and the specified Point number will be added to the alarm group. A subsequent call to the *almadd* function whose GRPNAME has been previously created, will add the new point to the existing alarm group.

The function Output is used to indicate the success (or otherwise) of the creation of a named alarm group, and/or the successful (or otherwise) addition of the point to the named alarm group.

Attempting to add the same Point number twice to the same Alarm Group name will result in the *almadd* function returning an error code 3.

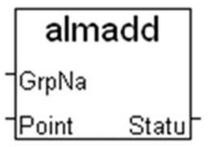


Figure 6.63: almadd Function

INPUTS	TYPE	DESCRIPTION
GrpName	Message	INPUTS of alarm group (a string)
Point	Integer	RTU Digital Input or User Point number.

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code status: -1 = Internal Error 0 = Success 1 = Point does not exist 2 = Invalid Group 3 = Point already exists in Group

Structured Text Example:

```
    IF (almadd(‘ProtectionFault’, Z_P_PHA_TRIP) = 0) THEN
        Status1 := almadd(‘ProtectionFault’,Z_P_PHB_TRIP);
        Status2 := almadd(‘ProtectionFault’,Z_P_PH _TRIP);
    END_IF;
```

The use of a point number “Z_…”. This could be a numeric value, or an ISaGRAF define generated by the Schneider Electric ISaGRAF Pre-Processor. This ISaGRAF function dynamically creates an alarm group and dynamically adds points to the alarm groups. Apart from upper memory limitations of the RTU’s CPU, there is no design limit to the number of alarm groups that can be created using this function block. Similarly there is no design limit to the number of points that can be added to an alarm group.

5.9.2 almproc

Process a summary alarm group

Description

Processing (execution) of the Alarm Group is performed by a call from ISaGRAF user application code to the ALMPROC Function Block.

Once configured in a one-off execution of ISaGRAF code using calls to the *almadd* function block instance (as described in Section ***almadd***174 above), a call to an *almproc* function block would typically occur each ISaGRAF scan. The rate of execution could be controlled through additional logic if required, but optimal responsiveness is achieved through execution each ISaGRAF scan.

The *almproc* function block takes as an input an alarm group name (as created and configured by calls to the ALMADD function block). The *almproc* function block summarizes the alarm states of the given alarms and asserts its *alm_out* output parameter when one of the input points is in alarm. In addition, an ACCEPT input masks the active alarms and clears the *alm_out* output parameter. An alarm condition that is cleared on a point (i.e. a point that goes out of the alarm state) automatically clears the mask for that alarm on the next ISaGRAF scan of the *almproc* function. A recurrence of the alarm, or occurrence of another new alarm, again sets the *alm_out* output parameter. An ACCEPT masks the active alarms and clears the *alm_out* output parameter.

An RTU restart or ISaGRAF application restart clears the internal alarm mask. If an *almload* function is not used prior to using *almproc* active alarms on the Alarm Group will cause regeneration of the *alm_out* condition. *almload* can be used to restore an alarm mask, thereby preserving the alarm mask and stopping regeneration of the group alarm. See Section ***almload***179 below.

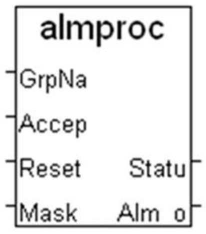


Figure 6.64: almproc Function Block

INPUTS	TYPE	DESCRIPTION
GrpName	Message	String value identifying alarm group.
Accept	Boolean	OFF to ON transition (rising edge) of this point accepts current alarms.
Reset	Boolean	OFF to ON (rising edge) transition of this point clears internal alarm asks and generates the ALARM output if required.
Mask_pt	Integer	RTU digital user point number. 0 = No mask point

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	Indicates status or error code status: -1 = Internal Error 0 = Success 1 = Point does not exist 2 = Invalid group 4 = Mask point does not exist
Alm_out	Boolean	Unaccepted new alarm has occurred. Activating the ACCEPT input tasks the current alarms and clears the ALARM output point.

RTU point database attributes are applied to the logic associated with Alarm activation as shown in the following table. See the *SCADAPack E Data Processing Technical Manual* for more information.

Table 6.13: RTU Alarm Point Attributes and Descriptions

Point Attribute	Description	Effect on ALARM activation
Invert State (Physical digital inputs)	When OFF, the physical input in an Energized state represents "ON" (active) state in the database. When ON, the physical input in an Energized state represents "OFF" (active) state in the database.	No direct effect (see Alarm Active State below)
Debounce Time (Physical digital inputs)	Duration (in ms) that the input needs to stay active before it is registered as being ON in the database.	Delays updating point state in the RTU Database after the input is active.
Alarm Active State	Indicates which state in the point database represents the "alarm" state of the point. When this attribute value = ON, the database point state being ON is the alarm condition.	Determines which point state causes activation of the "Point-in-Alarm" property.
Alarm Time Deadband	Duration (in Seconds) that the point needs to stay in the alarm state before the "Point-in-Alarm" property is activate.	Delays activation of the alarm property of the point.
Alarm Clear Time Deadband	Duration (in Seconds) that the point must stay out the alarm state before the "Point-in-Alarm" property is de-activated.	Delays clearing of the alarm property of the point.
	Inhibits the "Point-in-Alarm"	Stops the point going in to the

Point Attribute	Description	Effect on ALARM activation
Alarm Inhibit	property being activated, regardless of the input condition.	alarm state.

Transition of the RESET input parameter (from OFF to ON state) causes the alarm mask (internal for that alarm group) to be reset. Following clearing of the mask, any active alarm states previously masked will reactivate the ALM_OUT output.

'fleeting alarms' of very short duration may not annunciate via the ALM_OUT output if the ISaGRAF scan rate (or call rate to the ALMPROC function block) exceeds the duration of the alarm condition.

A short duration alarm that exceeds a point's alarm time deadband (configured in the RTU point database) and the ISaGRAF scan rate will activate the ALM_OUT output. The ALM_OUT output will remain active (until ACCEPTed) even if the input point alarm condition is no longer active.

The MASK_PT input parameter is used to allow the internal Function Block mask to be written out into consecutive RTU Digital User points. These User points can be read later by the **ALMLOAD** Function in order to load a new internal mask. This may be used to make the internal mask non-volatile or to transfer the internal mask values to another RTU via DNP3 Peer Function Blocks for redundancy purposes, for example. If the MASK_PT input is zero, this functionality is disabled. If the point supplied to the MASK_PT input does not exist for one or more points in the Alarm Group, the STATUS value will be set to an error code of 4. However, this does not otherwise affect processing of the Alarm Group.

The STATUS output parameter is used to indicate the success (or otherwise) of the processing of a named alarm group. The ALM_OUT output indicates the presence of a new alarm that has not yet been accepted.

A typical structured text example could be:

```
SumAlm_ProtFault('ProtectionFault', Oper_ACCEPT, Oper_RESET);

IF (SumAlm_ProtFault.Status = 0) THEN

    ProtFault_Alarm := SumAlm_ProtFault.ALM_OUT;

    OPERATE(Oper_ACCEPT, 0);

    OPERATE(Oper_RESET, 0);

END_IF;
```

where "SumAlm_ProtFault" is an Instance of an "ALMPROC" function block. "Oper_ACCEPT" is a Boolean variable being the operator 'Accept' input and "Oper_RESET" is a Boolean variable to clear the summary alarm mask. "ProtFault_Alarm" could be an ISaGRAF Boolean Output variable that updates a point in the RTU point database. In this example it is assumed that the Oper_ACCEPT and Oper_RESET inputs are latched variables on ISaGRAF input boards from RTU database points. The OPERATE function clears the variables and their point source in the database. If pulse points were to be used instead of latch points, the OPERATE function calls would be unnecessary. However, when using Pulse Points, the pulse duration of ACCEPT and RESET points needs to be long enough to significantly exceed the length of the ISaGRAF scan.

5.9.3 almload

Load a mask to an alarm group

Description

Loading of the Alarm Group internal mask from an external source can be performed by a call from ISaGRAF User Application code using the *almload* ISaGRAF Function.

The *almload* function takes as an input an alarm group name (as created and configured by calls to the ALMADD function block) and an RTU User Digital point number. When used for restoring the alarm mask state, this Digital point number is normally the same number that was supplied as the MASK_PT input parameter to the ALMPROC function block for the same alarm group name.

The MASK_PT input parameter is used to allow the internal *almproc* Function Block mask to be loaded by reading consecutive RTU Digital User points. If the MASK_PT input is zero, an error code is returned. The number of RTU digital points read corresponds to the number of alarms in the Alarm Group.

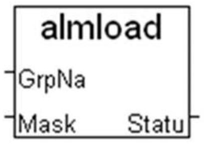


Figure 6.65: almload Function

INPUTS	TYPE	DESCRIPTION
GrpName	Message	String value identifying alarm group.
Mask_pt	Integer	RTU digital user point number. 0 = No mask point

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	-1 = Internal Error 0 = Success 1 = Point does not exist 2 = Invalid group

5.9.4 almclr

Destroy an alarm group

Description

A previously created Alarm Group can be removed using the ALMCLR ISaGRAF function. Calling this function frees RTU system resources associated with the named Alarm Group. After this function is called, the named alarm group no longer exists and does not perform alarm group functions. ISaGRAF alarm group functions and function blocks referencing this alarm group after an ALMCLR will result in status error codes.

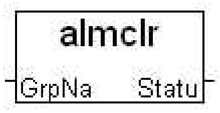


Figure 6.66: almclr Function

INPUTS	TYPE	DESCRIPTION
GrpName	Message	String value identifying alarm group.

OUTPUTS	TYPE	DESCRIPTION
Status	Integer	-1 = Internal Error 0 = Success 2 = Invalid group

5.10 RTU File System Interface Functions & Function Blocks

This section describes the functions and function blocks that support access to the RTU file system.

Many of the functions and function blocks detailed in this section return an analog status value.

ISaGRAF functions that provide similar functionality to the command line interface for file system access are detailed in Section ***File System Management Functions*** 182.

The directory / drive information function blocks are detailed in Section ***Directory Information Function Blocks*** 195 and the file read / write functions are detailed in Section ***ISaGRAF File Read / Write Functions*** 201.

Refer to Section ***File System Access Error Codes*** 209 for a list of file system status error codes.

5.10.1 File System Access Functions

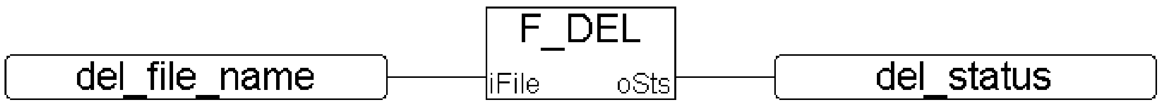
This section details the functions to allow ISaGRAF to manage the file system.

Equivalent functionality provided by the SCADAPack E command line commands are indicated where applicable.

ISaGRAF Function Name	Equivalent RTU Command Line command	Purpose
F_DEL 182	• DEL	Delete Files
F_DELTRE 184	• DELTREE	Delete a directory and files
F_COPY 185	• COPY	Copy a file
F_JOIN 186	• JOIN or APPEND	Append files
F_REN 187	• RENAME	Rename a file
F_MKDIR 188	• MD	Make a directory
F_RMDIR 189	• RMDIR	Remove a directory
F_CD 190	• CD	Change current working directory
F_DSKSEL 191	• DSKSEL	Select a Disk device
F_PWD 193		returns the Present Working Directory
F_DV_RDY 194		indicates if a disk Device is Ready

5.10.1.1 F_DEL

The F_DEL function is used to delete the specified file from the RTU file system from within an ISaGRAF program. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iFile	Message	Filename to delete (including path)

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Delete Request

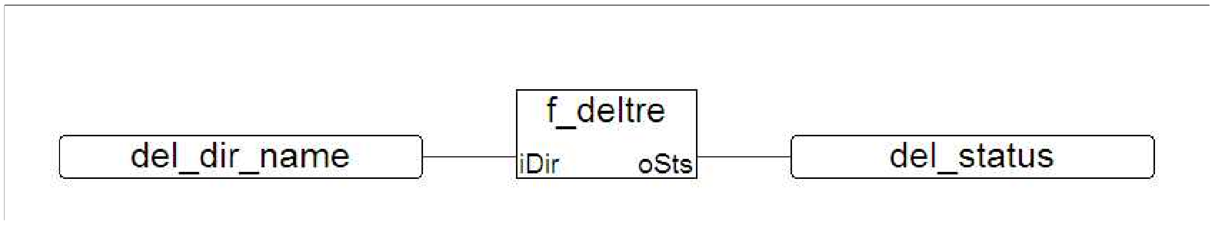
The **iFile** argument specifies the filename to delete, and is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iFile** argument is 255.

The **iFile** argument and a may include the complete path, e.g. “C:\sample.txt”. If only the filename is specified, the current working directory will be used to determine the complete path. The current working directory can be changed in ISaGRAF using **F_CD** function (see Section ***F_CD Function***190).

The function returns a 0 if the delete request was successful. If an error was detected, a negative integer value is returned. Refer to Section ***File System Access Error Codes***209 for the range of possible error codes and their descriptions.

5.10.1.2 F_DELTRE

The F_DELTRE function is used to remove the specified directory and files in that directory. Subdirectories and files below the specified directory are also deleted. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDir	Message	Directory to delete

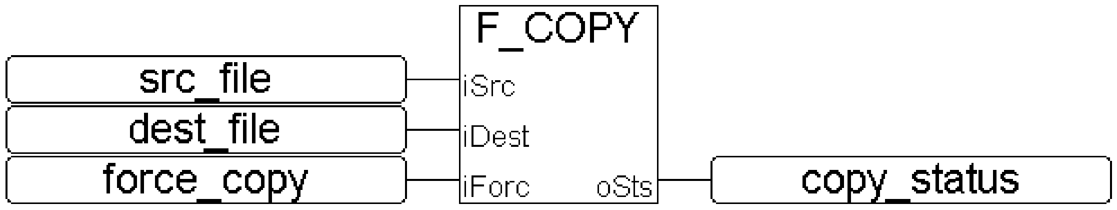
OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Delete Request

iDir is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iDir** argument is 255.

The function returns a 0 if the delete tree request was successful. If an error was detected, a negative integer value is returned. Refer to Section ***File System Access Error Codes***209 for the range of possible error codes and their descriptions.

5.10.1.3 F_COPY

The F_COPY function is used to copy the specified source file to a new file in the RTU file system whose name is specified as the destination filename. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iSrc	Message	Source file to be copied
iDest	Message	Filename for the new copied file
Force_copy	Message	Forces copy over existing files

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Delete Request

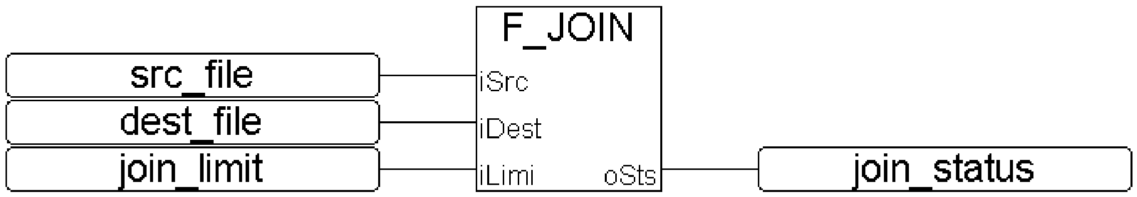
The **iSrc** argument specifies the filename of the file to be copied, and the **iDest** argument specifies the filename for the new copied file. Both of these arguments are case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iSrc** and **iDest** arguments is 255. The **iForce** argument allows existing destination files to be overwritten.

The **iSrc** and **iDest** arguments may include the complete path, e.g. “C:\sample.txt”. If only the filename is specified, the current working directory will be used to determine the complete path. The current working directory can be changed in ISaGRAF using **F_CD** function (see Section ***F_CD Function***190).

The function returns a 0 if the copy request was successful. If an error was detected, a negative integer value is returned. Refer to Section ***File System Access Error Codes***209 for the range of possible error codes and their descriptions.

5.10.1.4 F_JOIN

The F_JOIN function is used to append the specified source file to the specified destination file. The third argument to this function block allows the programmer to “optionally” specify a maximum size (in bytes) for the destination file. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iSrc	Message	Source file to be copied
iDest	Message	Filename for the new copied file
iLimit	Integer	‘Optional’ maximum size for the destination file (0 = no limit)

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request

The **iSrc** argument specifies the filename of the file(s) to be appended, and the **iDest** argument specifies the filename to which the source file is appended to. Both of these arguments are case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iSrc** and **iDest** arguments is 255.

The **iLimit** argument specifies the maximum size of the destination file. If this argument is zero, the source file(s) will be unconditionally appended to the destination file.

The **iSrc** and **iDest** arguments may include the complete path, e.g. “C:\sample.txt”. If only the filename is specified, the current working directory will be used to determine the complete path. The current working directory can be changed in ISaGRAF using **F_CD** function (see ***F_CD Function*** 190).

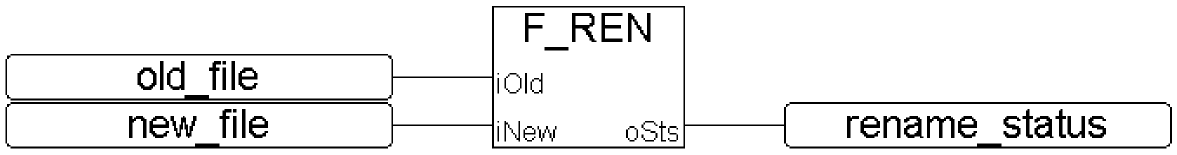
The **iSrc** filename may include a wildcard specification, e.g. SF*

The function returns a 0 if the join request was successful. If an error was detected, a negative integer value is returned. Refer to ***File System Access Error Codes*** 209 for the range of possible error codes and their descriptions.

The F_JOIN function can be used with wildcards and executed cyclically. In cases where there are no source files in the file system matching the **iSrc** filename, the F_JOIN function block intentionally does NOT return a file system error. In this specific case the function returns 0.

5.10.1.5 F_REN

The F_REN function is used to rename a specified file to a new filename in the RTU file system. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iOld	Message	Existing filename
iNew	Message	New filename

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request

The **iOld** argument specifies the existing filename of the specified file, and the **iNew** argument specifies the new filename for the specified file. Both of these arguments are case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iOld** and **iNew** arguments is 255.

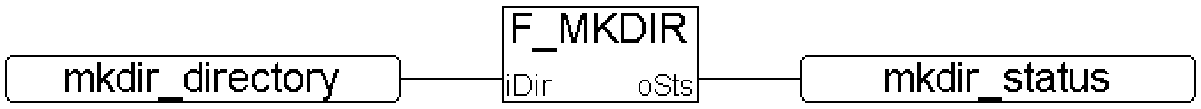
The **iSrc** and **iDest** arguments may include the complete path, e.g. “C:\sample.txt”. If only the filename is specified, the current working directory will be used to determine the complete path. The current working directory can be changed in ISaGRAF using **F_CD** function (see ***F_CD Function*** 190).

The function returns a 0 if the rename request was successful. If an error was detected, a negative integer value is returned. Refer to ***File System Access Error Codes*** 209 for the range of possible error codes and their descriptions.

5.10.1.6 F_MKDIR

The F_MKDIR function is used to make (create) directories, and requires a single argument which specifies the directory name (or complete path) to be created. If the directory name alone is specified, the directory is created as a subdirectory of the current working directory. A complete path specification allows directories to be created wherever required. The current working directory can be changed in ISaGRAF using **F_CD** function (see ***F_CD Function***190). The F_MKDIR function supports creation of directories on other drives, i.e. different to the current drive.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDir	Message	Directory to create

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request

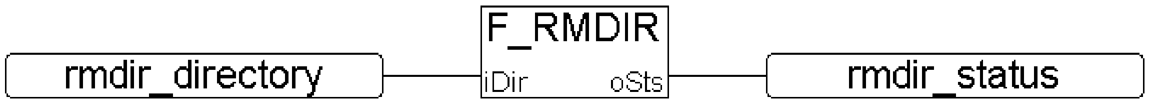
iDir is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iDir** argument is 255.

The function returns a 0 if the make directory request was successful. If an error was detected, a negative integer value is returned. Refer to ***File System Access Error Codes***209 for the range of possible error codes and their descriptions.

5.10.1.7 F_RMDIR

The F_RMDIR function is used to remove directories, and requires a single argument which specifies the directory name (or complete path) to be removed. If the directory name alone is specified, the directory needs to exist as a subdirectory of the current working directory. A full path specification allows directories to be removed wherever required. The current working directory can be changed in ISaGRAF using **F_CD** function (see ***F_CD Function***190). The F_RMDIR function supports removal of directories on other drives, i.e. different to the current drive.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDir	Message	Directory to remove

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request

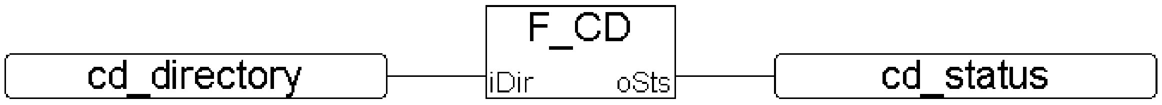
iDir is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iDir** argument is 255.

The function returns a 0 if the remove directory request was successful. If the operation was unsuccessful a negative integer value is returned. Refer to ***File System Access Error Codes***209 for the range of possible status codes and their descriptions.

5.10.1.8 F_CD

The F_CD function is used to change the current working directory within a given drive. If the target directory is directly below the current working directory, only the directory name is required, otherwise the complete path is required. An **iDIR** argument of "**..**" will change the working directory to one level above the current working directory.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDir	Message	Target Directory

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request

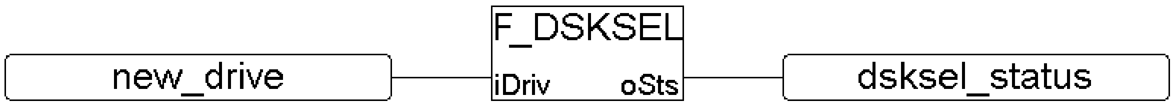
iDir is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iDir** argument is 255.

The function returns a 0 if the change working directory request was successful. If an error was detected, a negative integer value is returned. Refer to ***File System Access Error Codes***209 for the range of possible error codes and their descriptions.

5.10.1.9 F_DSKSEL

The F_DSKSEL function is used to change the between working drive devices on the RTU file system. After executing this function, the current working directory on the target drive will be selected (see notes below).

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDrive	Message	Target Drive (e.g. C: E: c: e:) Text is case insensitive

OUTPUTS	TYPE	DESCRIPTION
oSts	Integer	Status of Request. 0 = Success. See ***File System Access Error Codes*** 209 for the range of possible status codes and their descriptions

The following disk devices are available on SCADAPack E RTUs:

Drive	Type	Size (SCADAPack ES and SCADAPack ER)	Size (SCADAPack 300E)	Description
C:	Main Flash File System Drive	12 MB	7 MB	Main file system used by the operating system and available for user data. It is recommended that user files be placed in sub-directories below the root directory
D:	RAM Drive	2 MB	128 KB	Used for temporary storage by the operating system (e.g. RTU log files, sampler files, etc)
E:	External CompactFLASH Drive	up to 2 GB	-	External drive available for user data (not available on SCADAPack 300E RTUs)
F:	Internal Flash File System Drive	16 MB	-	Additional file system available for user files. It is recommended that

				user files be placed in sub-directories below the root directory (not available on SCADAPack 300E RTUs)

Notes when using sub-directories:

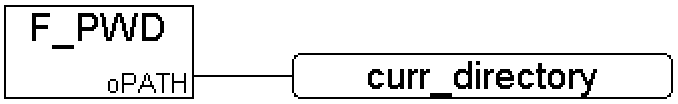
The current working directory on each drive is preserved prior to changing drives.

For example, If the current working directory is currently “C:\testdir”, and the last specified working directory on the “D drive” was “D:\targetdir”, the F_DSKSEL function with the argument “**D:**” would then result with “D:\targetdir” as the current working directory. Calling the F_DSKSEL function with the argument “**C:**” would then result with “C:\testdir” as the current working directory.See F_CD 190.

5.10.1.10 F_PWD

The F_PWD function is used to display the present (current) working directory for the ISaGRAF kernel task in which the function is called.

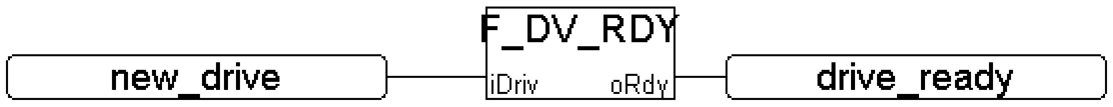
The figure below shows the function with the following return parameters. There are no calling parameters.



OUTPUTS	TYPE	DESCRIPTION
oPATH	Message	Present (current) Working Directory.

5.10.1.11 F_DV_RDY

The F_DV_RDY function is used to determine whether or not the specified drive is mounted and ready. The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
iDrive	Message	Target Drive to be checked (e.g. C: E:)

OUTPUTS	TYPE	DESCRIPTION
oRdy	Boolean	Status of Request

iDrive is case insensitive (upper and lower case allowed). The maximum number of characters allowed for the **iDrive** argument is 10.

The function returns TRUE if the specified drive is mounted and ready for use, otherwise FALSE in returned.

5.10.2 Directory Information Function Blocks

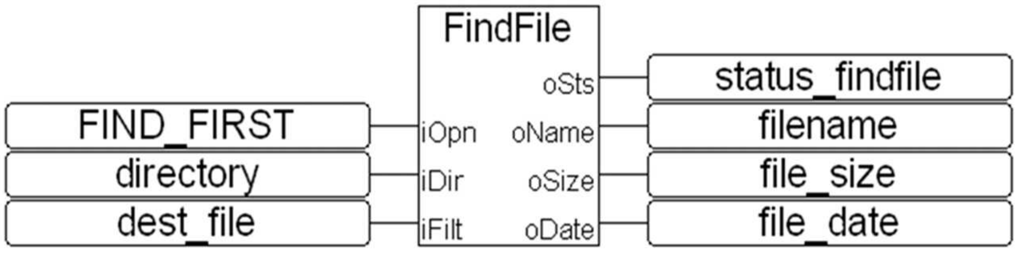
The functionality of the command line **DIR** command is provided by the following ISaGRAF function blocks

- ***FINDFILE Function Block*** 196
- ***DIR_INFO Function Block*** 199

5.10.2.1 FINDFILE Function Block

The **FINDFILE** function block allows the programmer to search a given directory for a specific file. The specific file details to be returned are determined by the specified file operation, i.e. “first file”, “next file”, “oldest file”, "first directory, or the "next directory". The function block also allows the programmer to specify the directory to search, and a filter as the search criteria, which is only required when searching for the “first file” or "first directory". In addition to the retrieved filename, this function block also returns size of the file in bytes and the file creation data in seconds since 1970.

Instances of the FINDFILE function block in the ISaGRAF target kernel share the same internal position data. This means that calling one instance of FILEFIND with the FIND_FIRST operation would affect the results from later calling a different instance with the FIND_NEXT operation. Only one FINDFILE function block can be active in each ISaGRAF target kernel.



Function Block Parameters:

The inputs to the **FINDFILE** function block are as follows:

iOpn (ANA) –	specifies which file operation to be carried out. The possible values are listed as follows:

0 = FIND_FIRST	This instructs the FindFile function block to search for the first file in the specified directory (defined in common.eqv as FIND_FIRST)
1 = FIND_NEXT	This instructs the FindFile function block to search for the next file in the specified directory. The filter for the search would have been specified in the previous "FIND_FIRST" call defined in common.eqv as FIND_NEXT).
2 = FIND_OLDEST	This instructs the FindFile function block to search for the oldest file in the specified directory. The filter for the "oldest" file search is specified by the Dir input parameter defined in common.eqv as FIND_OLDEST).
3 = FIND_FIRST_DIR	This instructs the FindFile function block to search for the first directory in the specified directory (defined in common.eqv as FIND_FIRST_DIR)
4 = FIND_NEXT_DIR	This instructs the FindFile function block to search for the next directory in the specified directory. The filter for the search would have been specified in the previous "FIND_FIRST_DIR" call defined in common.eqv as FIND_NEXT_DIR).

iDir (MSG) – specifies the directory for the file search. This argument is case insensitive (upper and lower case allowed) and the maximum number of characters allowed is 255. If no directory path is specified, the current working directory will be referenced for the search.

iFilt (MSG) – specifies the filter for the search criteria, e.g. “*” to search all files. This is not required for the FIND_NEXT calls as this would have been specified in the FIND_FIRST call. This argument is case insensitive (upper and lower case allowed) and the maximum number of characters allowed is 255. If no filter is specified then “*” will be used.

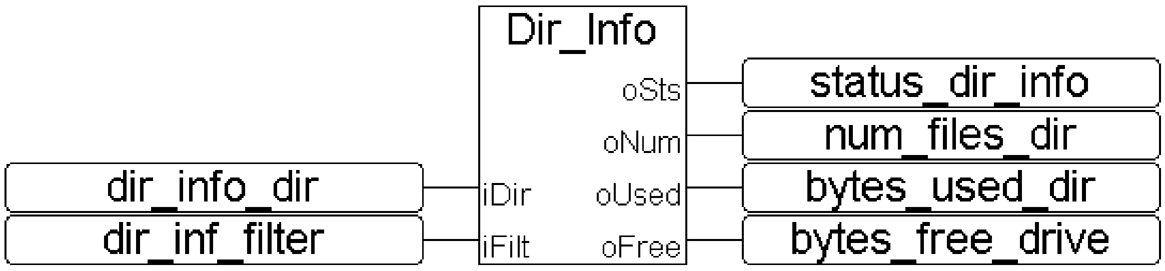
The outputs to the **FINDFILE** function block are as follows:

oSts (ANA)	–	specifies which status of the requested file operation. The function block **oSts** value returns 0 if the search was successful (according the specified search criteria). If an error was detected, a negative integer value is returned. Refer to ***File System Access Error Codes*** 209 for the range of possible error codes and their descriptions.
oName (MSG)	–	specifies the file name retrieved and is only valid if the **oSts** output indicates success, i.e. 0. The maximum number of characters allowed for the **oName** output is 255.
oSize (ANA)	–	specifies the size of the detected file in bytes and is only valid if the **oSts** output indicates success, i.e. 0.
oDate (ANA)	–	specifies the time and date stamp of the detected file in “seconds since 1970”, and is only valid if the **oSts** output indicates success, i.e. 0.

5.10.2.2 DIR_INFO Function Block

The **DIR_INFO** function block allows the programmer to determine the following information for a given directory

- number of files in the directory
- total bytes used in directory (according to specified filter)
- bytes available in the current working drive



Function Block Parameters:

The inputs to the **DIR_INFO** function block are as follows:

iDir (MSG) – specifies the directory for the information search. This argument is case insensitive (upper and lower case allowed) and the maximum number of characters allowed is 255. If no directory is specified, the current working directory will be referenced for the directory information search.

iFiltr (MSG) – specifies the filter for the **DIR_INFO** information search, e.g. “*” to include all files. This argument is case insensitive (upper and lower case allowed) and the maximum number of characters allowed is 255. If no filter is specified then “*” will be used.

The outputs to the **DIR_INFO** function block are as follows:

oSts (ANA) – specifies which status of the directory information search. The function block oSts value returns 0 if the information search was successful. If the operation was unsuccessful a negative integer value is returned. Refer to ***File System Access Error Codes***[209] for the range of possible status codes and their descriptions.

oNum (ANA) – specifies the number of files in the directory according to the search criteria, i.e. the iFiltr input. The presented value is only valid if the oSts output indicates success.

oUsed (ANA) – specifies the total number of bytes used by files in the directory that satisfy the search criteria, i.e. the iFiltr input. The presented value is only valid if the oSts output indicates success.

oFree (ANA) – specifies the total number of bytes available in the RTU file system. The presented value is only valid if the oSts output indicates success.

5.10.3 ISaGRAF File Read / Write Functions

This section describes the functions that provide both read and write access to files in the RTU file system and allow for transfer of both ASCII and binary data.

These functions are included in the standard ISaGRAF IEC61131-3 library.

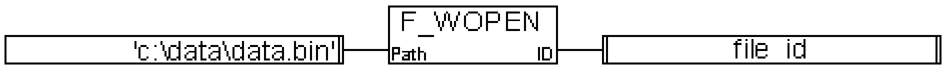
The functions are detailed in the following sections:

- *F_WOPEN* 202
- *F_ROPEN* 203
- *F_CLOSE* 204
- *F_EOF* 205
- *FA_READ* 206
- *FA_WRITE* 207
- *FM_READ* 208
- *FM_WRITE* 209

5.10.3.1 F_WOPEN Function

The F_WOPEN function opens the specified file in write mode. It is to be used with the FA_WRITE, FM_WRITE, and F_CLOSE functions.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
Path	Message	Filename to open in write mode (may include path)

OUTPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

The **PATH** argument specifies the filename to open in write mode.

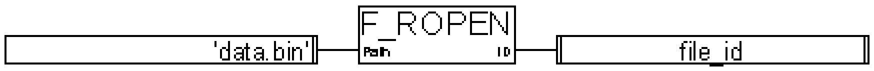
This argument may include the complete path, otherwise the current working directory shall be used to determine the complete path for the specified file. This argument is case insensitive (upper and lower case allowed), and the maximum number of characters allowed is 255.

The **ID** return parameter presents a file handle to be used for subsequent file accesses. A returned value of 0 indicates the specified file could not be opened in write mode.

5.10.3.2 F_ROPEN Function

The F_ROPEN function opens the specified file in read mode. It is to be used with the FA_READ, FM_READ, and F_CLOSE functions.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
Path	Message	Filename to open in read mode (may include path)

OUTPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

The **PATH** argument specifies the filename to open in read mode.

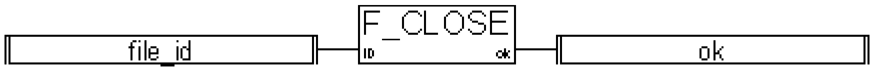
This argument may include the complete path, otherwise the current working directory shall be used to determine the complete path for the specified file. This argument is case insensitive (upper and lower case allowed), and the maximum number of characters allowed is 255.

The **ID** return parameter presents a file handle to be used for subsequent file accesses. A returned value of 0 indicates the specified file could not be opened in read mode.

5.10.3.3 F_CLOSE Function

The F_CLOSE function closes files that have been opened with F_WOPEN or F_ROPEN.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	Status of file close request

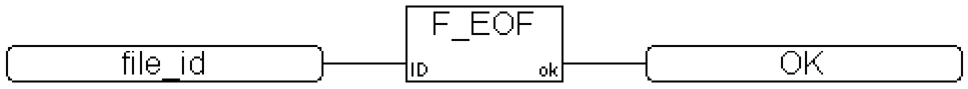
The **ID** argument specifies the file handle that was returned in calls to either F_WOPEN or F_ROPEN. A valid ID value for a currently open file is non-zero.

The **OK** return parameter presents a boolean status for the file close request. TRUE is returned is the file close is OK, otherwise FALSE is returned.

5.10.3.4 F_EOF Function

The F_EOF function tests whether the end of file has been reached.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
OK	Boolean	Status of request

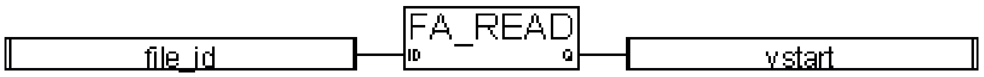
The **ID** argument specifies the file handle that was returned in calls to either F_WOPEN or F_ROPEN. A valid ID value for a currently open file is non-zero.

The **OK** return parameter presents a boolean status for the end of file test. TRUE is returned is the end of file has been reached in the last read or write procedure call, otherwise FALSE is returned.

5.10.3.5 FA_READ Function

The FA_READ function reads analog values from a binary file. It is to be used with the F_ROPEN and F_CLOSE functions. This function makes a sequential access to the file from the previous position. The first call after F_ROPEN reads the first 4 bytes of the file. Each call pushes the “read” pointer. To check whether the end of file is reached, the F_EOF function is used.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
q	Integer	Integer value read from file

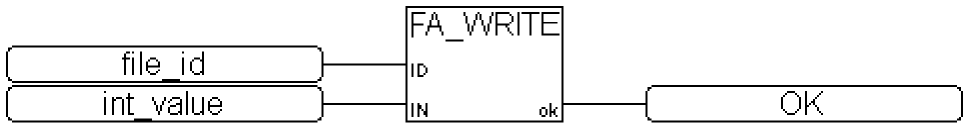
The **ID** argument specifies the file handle that was returned in the call to F_ROPEN. A valid ID value for a currently open file is non-zero.

The **Q** return parameter returns the integer analog value read from the file.

5.10.3.6 FA_WRITE Function

The FA_WRITE function writes analog values to a binary file. It is to be used with the F_WOPEN and F_CLOSE functions. This function makes a sequential access to the file from the previous position. The first call after F_WOPEN writes the first 4 bytes of the file. Each call pushes the “write” pointer.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)
IN	Integer	Integer value to be written to file

OUTPUTS	TYPE	DESCRIPTION
q	Integer	Status of file write request

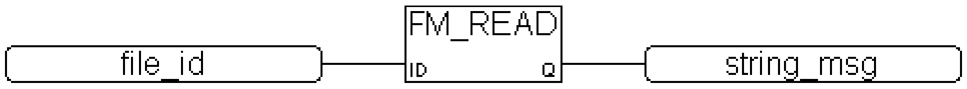
The **ID** argument specifies the file handle that was returned in the call to F_WOPEN. A valid ID value for a currently open file is non-zero. The **IN** argument represents the actual value that will be written (in 4 bytes) to the specified file.

The **OK** return parameter returns the status of the file write request. TRUE is returned if the file write was successful, otherwise FALSE is returned.

5.10.3.7 FM_READ Function

The FM_READ function reads MESSAGE values from a binary file. It is to be used with the F_ROPEN and F_CLOSE functions. This function makes a sequential access to the file from the previous position. The first call after F_ROPEN reads the first string of the file. Each call pushes the “read” pointer. A string is terminated by null (0), end of line (‘\n’) or return (‘\r’). To check whether the end of file is reached, the F_EOF function is used.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
q	Integer	Message variable read from file

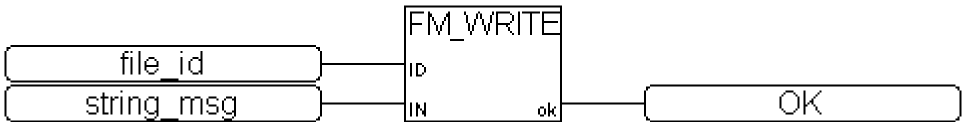
The **ID** argument specifies the file handle that was returned in the call to F_ROPEN. A valid ID value for a currently open file is non-zero.

The **Q** return parameter returns the string read from the file.

5.10.3.8 FM_WRITE Function

The FM_WRITE function writes a message variable to a binary file as a null terminated string. It is to be used with the F_WOPEN and F_CLOSE functions. This function makes a sequential access to the file from the previous position. The first call after F_WOPEN writes the first string to the file. Each call pushes the “write” pointer.

The figure below shows the function with the following calling and return parameters:



INPUTS	TYPE	DESCRIPTION
ID	Integer	File ID (file handle)
IN	Message	Message variable to be written to file

OUTPUTS	TYPE	DESCRIPTION
q	Integer	Status of file write request

The **ID** argument specifies the file handle that was returned in the call to F_WOPEN. A valid ID value for a currently open file is non-zero. The **IN** argument represents the actual string that will be written to the specified file.

The **OK** return parameter returns the status of the file write request. TRUE is returned if the file write was successful, otherwise FALSE is returned.

5.10.4 File System Status and Error Codes

Many of the RTU file system functions and function blocks return an integer status / error code. The status / error code values are detailed in the following table.

Table 6.14: File System Status / Error Codes

Status Value	Description
0	Success
-1	Unknown Error
-1000	Source name is illegal

-1001	Source file is in use
-1002	Source file does not exist
-1003	Invalid file pointer
-1004	Illegal position
-1005	Illegal destination name
-1006	Source file name in use
-1007	Invalid query structure
-1008	Destination file name in use
-1009	Error creating working buffer
-1010	Error writing to file
-1011	Could not create file
-1012	New file would exceed the maximum file size
-1013	Requested operation not supported
-1014	Directory in use
-1015	File or Directory does not exist
-1016	Invalid Path
-1017	File or Directory already exists