SCADAPack E Target 5
Function Block Reference

Documentation

Table of Contents

I SCADAPack E Target 5 Function Block Reference

Schneider Electric

Documentation

Version: 8.05.4

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed. Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

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

SCADAPack Workbench and SCADAPack E Smart RTU are not suitable for controlling safety-critical systems. SCADAPack Workbench and SCADAPack E Smart RTU are not tested for, nor have approval for use in, the control of safety-critical systems. Safety-critical systems should be controlled by an approved safety-critical platform that is independent of SCADAPack Workbench and SCADAPack E Smart RTU.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not control safety-critical systems with SCADAPack Workbench and SCADAPack E Smart RTU.

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

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 Overview

This document describes Function Blocks and Functions supported by the SCADAPack E Smart RTU with the SCADAPack Workbench.

These function blocks or functions access to SCADAPack E Smart RTU configuration and I/O data in the form of point properties and attributes, and access to SCADAPack E Smart 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 Function Block Diagram or Structured Text languages.

The manual is arranged as follows:

- Point Attribute Function Blocks 12 presents the function blocks that return or set point properties.
- Real Time Clock Function Blocks 65 describes function blocks which provide access to the real time clock.
- DNP3 Communication Function Blocks 68 describes function blocks that interface with the SCADAPack E Smart RTU peer-to-peer communication facilities. Peer function blocks transfer data directly between the point databases of two peer DNP3 devices.
- Serial Port User Communication Functions 84 describes function blocks to set or retrieve serial port properties on the SCADAPack E Smart RTU.
- Miscellaneous Function Blocks 107 describes miscellaneous function blocks.
- TCP/IP Interface Functions 127 describe function blocks interface to TCP/IP communication and configuration services.
- Alarm Group Functions & Function Block 134 describes the Summary Alarm functions and function blocks. These provide a mechanism for grouping individual 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.
- File System Interface Functions & Function Blocks 141 describes interfaces to the SCADAPack E Smart RTU file system to allow manipulation of files.

Assumed Knowledge

Exposure to SCADAPack Workbench is recommended.

Target Audience

- Systems Engineers
- Commissioning Engineers
- Maintenance Technicians

References

- Workbench Help

- SCADAPack E Technical Reference Manuals.

4 Function Blocks

SCADAPack Workbench provides function blocks and functions for the SCADAPack E Smart RTU.

4.1 Point Attribute Function Blocks

These function blocks set or return the property or attribute of a point stored in the point 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 Smart 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 Smart 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 Workbench Technical Reference* manual.

4.1.1 Read and Write Point Values

Get database point value (getpntxx) and set database point value (setpntxx) functions and function blocks are used to interface IEC 61131-3 logic with the point database.

These functions require significantly more processing time than accessing points through I/O Devices. See SCADAPack Workbench I/O Device reference manual. Impact on IEC 61131-3 Resource performance should be considered when using these functions instead of I/O devices.

getpntxx functions and function blocks read database point values in to IEC 61131-3 variables, in various formats.

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

setpntxx function and function blocks write IEC 61131-3 variable values in to database point variables, in various formats.

- setpntb 20 (set point binary)
- setpntf 21 (set point floating point)
- setpntsl 22 (set point signed long)
- setpntss 23 (set point signed short)
- setpntus 25 (set point unsigned short)

4.1.1.1 getpntb

Get value of binary point

Description

The getpntb function returns the 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
Point	DINT	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 The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OUTPUT_POINT	BOOL	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
GETPNTB	BOOL	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 20

4.1.1.2 getpntc

Get value of counter point

Description

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

Arguments

Inputs	Type	Description
Point	DINT	Address of a database counter point number to read. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.

Outputs	Type	Description
Value	UDINT	32-bit unsigned integer value of counter point if found, otherwise 0

4.1.1.3 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
Point	DINT	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 The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OUTPUT_POINT	BOOL	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
GETPNTF	REAL	Current Engineering Value if found, otherwise 0.0

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

See Also

setpntf 21

4.1.1.4 getpntsl

Get value of analog point as a 32-bit signed 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.

The name of this function block is identical to the Target 3 function block. The L in the name refers to a Target 3 long integer (32-bit signed value). It should not be confused with the Target 5 data type LINT which refers to a 64-bit signed value.

Arguments

Inputs	Type	Description
Point	DINT	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 The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OUTPUT_POINT	BOOL	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
GETPNTSL	DINT	Current Integer Value if found, otherwise 0

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

See Also

setpntsl 22

4.1.1.5 getpntss

Get value of analog point as signed short integer

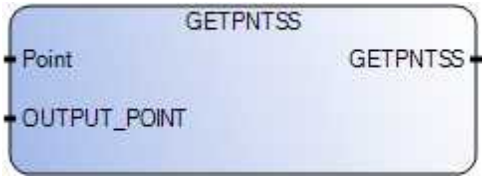
Description

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
Point	DINT	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 The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OUTPUT_POINT	BOOL	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
GETPNTSS	INT	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 23

4.1.1.6 getpntus

Get value of analog point as unsigned short integer

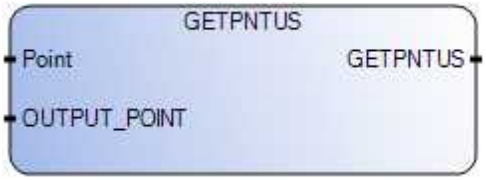
Description

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
Point	DINT	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 The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OUTPUT_POINT	BOOL	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
GETPNTUS	UINT	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 25

4.1.1.7 setpntb

Set value of binary point

Description

The setpntb function writes the BOOL 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
Point	DINT	Address of a binary point number. Point may be a binary output, derived binary point or system binary point. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
VALUE	BOOL	Value to write to the binary point.

Outputs	Type	Description
SETPNTB	BOOL	TRUE if the value was successfully set. FALSE for other cases.

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

See Also

getpntb 14

4.1.1.8 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
Point	DINT	Address of a binary point number. Point may be a binary output, derived binary point or system binary point. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
VALUE	REAL	Value to write to the binary point.

Outputs	Type	Description
SETPNTF	BOOL	TRUE if the value was successfully set. FALSE for other cases.

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

See Also

getpntf 15

4.1.1.9 setpntsl

Set value of analog point as a 32-bit signed 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.

The name of this function block is identical to the Target 3 function block. The L in the name refers to a Target 3 long integer (32-bit signed value). It should not be confused with the Target 5 data type LINT which refers to a 64-bit signed value.

Arguments

Inputs	Type	Description
Point	DINT	Address of an analog point number. Point may be an analog output, derived analog point or system analog point. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
VALUE	DINT	Value to write to the analog point.

Outputs	Type	Description
SETPNTSL	BOOL	TRUE if the value was successfully set. FALSE for other cases.

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

See Also

getpntsl 17

4.1.1.10 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
Point	DINT	Address of an analog point number. Point may be an analog output, derived analog point or system analog point. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
VALUE	INT	Value to write to the analog point. The value written to the point is clamped in the range -32768 to 32767

Outputs	Type	Description
SETPNTSS	BOOL	TRUE if the value was successfully set. FALSE for other cases.

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

See Also

getpntss 18

4.1.1.11 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
Point	DINT	Address of an analog point number. Point may be an analog output, derived analog point or system analog point. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
VALUE	UINT	Value to write to the analog point. The value from the point is clamped in the range 0 to 65535

Outputs	Type	Description
SETPNTUS	BOOL	TRUE if the value was successfully set. FALSE for other cases.

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

See Also

getpntus 19

4.1.2 Read Point Attributes

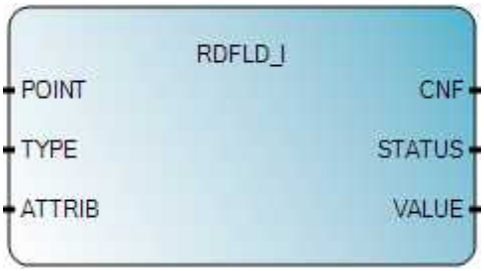
Read point field function blocks

4.1.2.1 rdfld_i

Read point attribute of type integer

Description

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



Arguments

Inputs	Type	Description
POINT	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
TYPE	DINT	Point Data Type argument can be one of the defined words below or an integer value corresponding to the data type. Defined Word / 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	DINT	Desired point attribute (property) argument can be a defined word or an integer value corresponding to the attribute. See tables below for defined words and their corresponding numeric values.

Outputs	TYPE	DESCRIPTION
CNF	BOOL	Confirm valid or invalid status Possible Values Meaning TRUE Confirm Valid Status FALSE
STATUS	DINT	Status of Read Request Possible Values Meaning -1 Unknown Status Code 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	DINT	Depends on 'Attrib' input parameter.

rdfld_i Function Attributes (attrib) for Point Properties

Attribute (Defined Word)	Equivalent Numeric Value	Description
Qlty	100	Point Quality
Fail	101	Point Operation Unsuccessful
IOF	102	IO Not Responding
ISA	103	IEC 61131-3 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

rdfld_i Function Attributes (attrib) for Integer (Analog) Points

Attribute (Defined Word)	Equivalent Numeric Value	Description
RoRPN	17	Rate Of Rise Point Number
RoFPN	18	Rate Of Fall Point Number

Attribute (Defined Word)	Equivalent Numeric Value	Description
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

rdfld_i Function Attributes (attrib) for Digital (BOOL) Points

Attribute (Defined Word)	Equivalent Numeric Value	Description
Inv	11	Invert Point State
AState	12	Alarm Active State
AClrTm	14	Alarm Clear Time Deadband

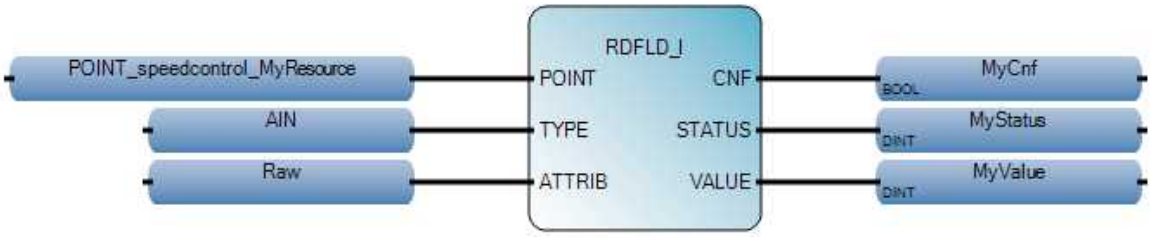
PTm	15	Output Pulse Time
DebTm	16	Debounce Time
DblStPt	54	Double Status Point Number

rdfld_i Function Attributes (attrib)

Attribute (Defined Word)	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 defined words in the tables above are reserved and should be used exclusively to retrieve point properties using this function block. User variables which duplicate these defined words but are not being used in the same context will generate errors during the compilation.

An IEC 61131-3 Function Block Diagram example of RDFLD_I is illustrated below. The current integer value (Attrib = Raw) of analog input point labeled POINT_speedcontrol_MyResource (analog input point 1) has been obtained. A variable or defined word POINT_speedcontrol_MyResource contains the point address of analog input point labeled ‘speedcontrol’. Alternatively, the numeric point address of the analog input can also be used.



IEC 61131-3 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_i defined in the program dictionary.

An equivalent Structured Text implementation 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 these variables are de*)
(* BOOL           CNF           *)
(* DINT           STATUS        *)
(* REAL           Value         *)
(* FB instances   rdfld i inst  *)
rdfld i inst( POINT speedcontrol MyResource, 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 *)
```

4.1.2.2 rdfld_r

Read point attribute of type real

Description

Point database attributes and property fields that return real (float) values may be read 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 SCADAPack E Smart RTU updates the Value variable from the specified point database field for the specified point and point type.

Arguments

Inputs	Type	Description
POINT	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
TYPE	DINT	Point Data Type argument can be one of the defined words below or an integer value corresponding to the data type. Defined Word — 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	DINT	Desired point attribute (property) argument can be an defined word or an integer value corresponding to the attribute. See Table 6.5 35 for defined words and their corresponding numeric values.

Outputs	TYPE	DESCRIPTION
CNF	BOOL	Confirm valid or invalid status Possible Values Meaning TRUE Confirm Valid Status FALSE
STATUS	DINT	Status of Read Request Possible Values Meaning -1 Unknown Status Code 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 bloc 8 Point is locked 12 Database is locked 18 Data Processor Unavailable
VALUE	REAL	Depends on 'Attrib' input parameter.

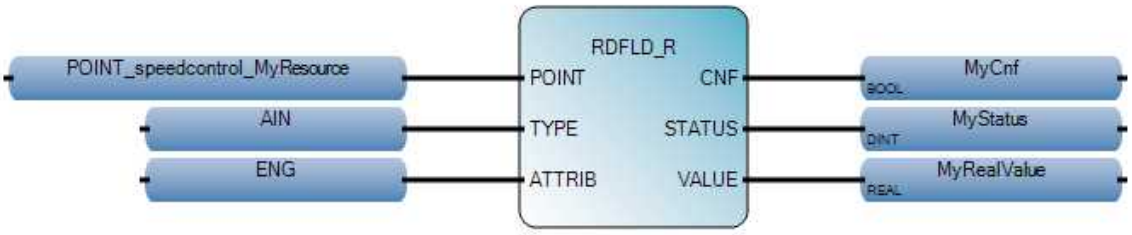
rdfld_r Function Attributes (attrib)

Attribute (Defined Word)	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
LimZero	41	Zero Threshold Limit
EvDev	50	Even _Deviation
LimOR	33	Over Range Limit
LimUR	34	Under Range Limit

The defined words in Table 6.5 35 are reserved and should be used exclusively to retrieve point properties using this function block. User defined variables which duplicate these defined words but are not being used in the same context will generate errors during the compilation.

An IEC 61131-3 Function Block Diagram example of RDFLD_R is illustrated in Figure 6.4 36 below. The current floating point or real value of analog input point labeled POINT_speedcontrol_MyResource (analog input pont 1) has been obtained. A variable or defined word POINT_speedcontrol_MyResource contains the point address of analog input point labeled ‘speedcontrol’. Alternatively, the numeric point

address of the analog input can also be used.



IEC 61131-3 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 36 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 fo*)
(* BOOL           CNF           *)
(* DINT           STATUS        *)
(* REAL           Value         *)
(* FB instances   rdfld r inst *)
rdfld r inst( POINT speedcontrol MyResource, 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 *)
```

4.1.3 Read Common Point Attributes

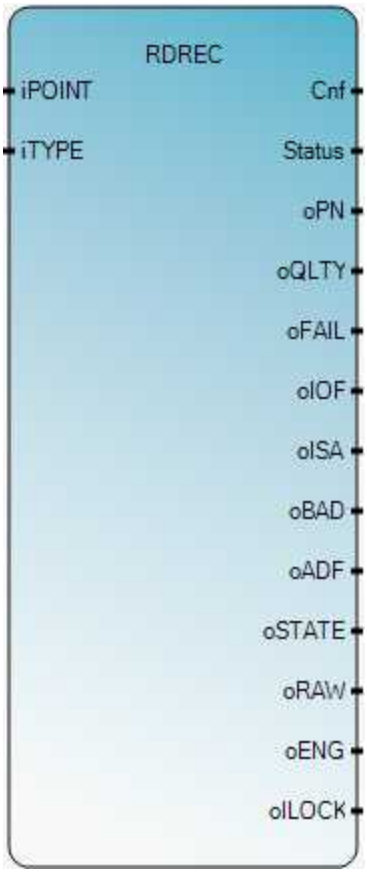
Function blocks to read common fields from a point record

4.1.3.1 rdrec

Read point attribute of type real

Description

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



Arguments

Inputs	Type	Description
iPOINT	DINT	Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
iTYPE	DINT	Point Data Type argument can be one of the defined words

		below or an integer value corresponding to the data type. Defined Word / 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

Outputs	TYPE	DESCRIPTION
Cnf	BOOL	Confirm valid or invalid status Possible Values / Meaning TRUE / Confirm Valid Status FALSE
Status	DINT	Status of Read Request Possbile Values / Meaning -1 / Unknown Status Code 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

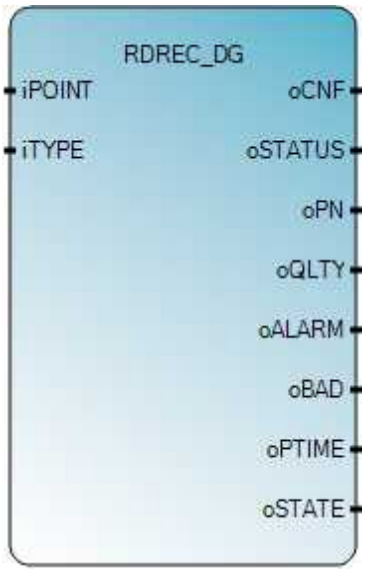
		18 Data Processor Unavailable
oPN	DINT	Point Number
oQLTY	DINT	Point Quality
oFAIL	BOOL	Point Operation Unsuccessful
oIOF	BOOL	IO not Responding
oISA	BOOL	IEC 61131-3 Controlled
oBAD	BOOL	Point is Bad
oADF	BOOL	A/D Reference Invalid
oSTATE	BOOL	Point State
oRAW	DINT	Raw Point Value
oENG	REAL	Engineering (REAL) Value
oILOCK	BOOL	Control Interlock Active

4.1.3.2 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.



Arguments

Inputs	Type	Description
iPOINT	DINT	Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
iType	DINT	Point Data Type argument can be one of the defined words below or an integer value corresponding to the data type. Defined Word — Equivalent Numeric Value — Comment DIN — 1 — Digital input point DOUT — 2 — Digital output point

Outputs	TYPE	DESCRIPTION
oCNF	BOOL	Confirm valid or invalid status Possible Meaning

		Values TRUE Confirm Valid Status FALSE
oSTATUS	DINT	Status of Read Request Possbile Values: Meaning -1: Unknown Status Code 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	DINT	Point Number
oQLTY	DINT	Point Quality
oALARM	BOOL	Point in Alarm
oBAD	BOOL	Point is Bad
oPTIME	DINT	Output Pulse Time
oSTATE	BOOL	Point State

IEC 61131-3 Structured Text prototypes take on the following form:

Prototype: rdrec_dg_inst (point, type)

complete_confirm := rdrec_dg_inst.ocnf

return_status := rdrec_dg_inst.pstatus

Point_number := rdrec_dg_inst.oPN …….

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.

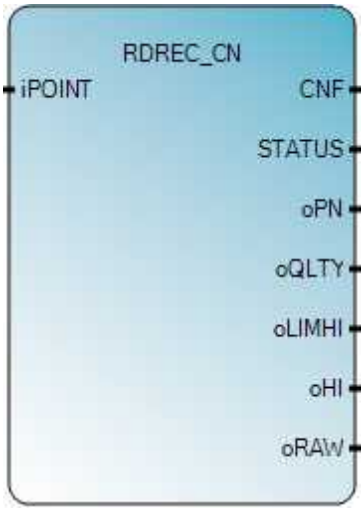
```
(*   Code Snippet Begins Here                        *)
(*   Ensure dictionary has the following variables *)
(*   BOOL                  cnf                       *)
(*   DINT                  status                    *)
(*   DINT                  point number              *)
(*   DINT                  point quality             *)
(*   BOOL                  alarm state               *)
(*   BOOL                  bad point                 *)
(*   DINT                  pulse tme                 *)
(*   BOOL                  point state               *)
(*   FB instances       rdrec dg inst                *)
rdrec dg inst( POINT 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 *)
```

4.1.3.3 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.



Arguments

INPUTS	TYPE	DESCRIPTION
iPOINT	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.

OUTPUTS	TYPE	DESCRIPTION
CNF	BOOL	Confirm valid or invalid status Possible Values — Meaning TRUE — Confirm Valid Status FALSE
STATUS	DINT	Status of Read Request Possible Values — Meaning -1 — Unknown Status Code 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	DINT	Point Number
oQLTY	DINT	Point Quality
oLIMHI	UDINT	Counter high limit
oHI	BOOL	Counter limit exceeded
oRAW	UDINT	Raw counter value

IEC 61131-3 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

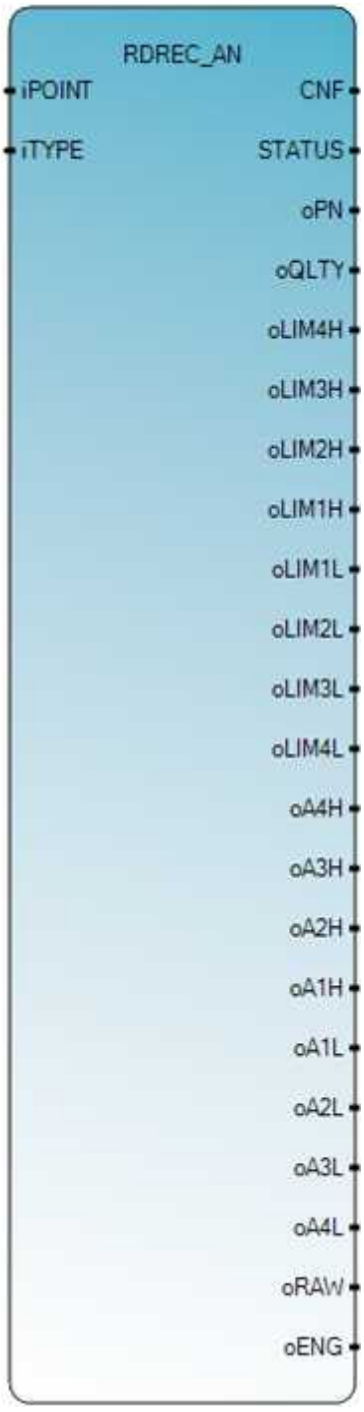
where rdrec_cn_inst is an instance of the function block rdrec_cn.

4.1.3.4 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.



Arguments

INPUTS	TYPE	DESCRIPTION
iPOINT	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a

		point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
iTYPE	DINT	Point data type argument can be one of the defined words below or an integer value corresponding to the data type. Defined Word / Equivalent Numeric Value / Comment AIN / 3 / Analog input point AOUT / 4 / Analog output point

OUTPUTS	TYPE	DESCRIPTION
CNF	BOOL	Confirm valid or invalid status Possible Values / Meaning TRUE / Confirm Valid Status FALSE
STATUS	DINT	Status of Read Request Possbile Values / Meaning -1 / Unknown Status Code 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	DINT	Point Number
oQLTY	DINT	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	BOOL	Alarm Limit Transgress 4H
oA3H	BOOL	Alarm Limit Transgress 3H
oA2H	BOOL	Alarm Limit Transgress 2H
oA1H	BOOL	Alarm Limit Transgress 1H
oA1L	BOOL	Alarm Limit Transgress 1L
oA2L	BOOL	Alarm Limit Transgress 2L
oA3L	BOOL	Alarm Limit Transgress 3L
oA4L	BOOL	Alarm Limit Transgress 4L
oRAW	DINT	Raw value
oENG	REAL	Floating point value

4.1.4 rdstring

Read a string point

Description

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

Arguments

INPUTS	TYPE	DESCRIPTION
Point	DINT	Point Address can be a numeric Point Address, a variable containing the DNP Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
Req	BOOL	Request to read a point Possible Values — Meaning TRUE — Read point enabled FALSE — Read point function disabled

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Confirm valid or invalid status Possible — Meaning

OUTPUTS	TYPE	DESCRIPTION
		Values TRUE Confirm Valid Status FALSE
Status	DINT	Status of Read Request Possbile Values: Meaning -1: Unknown Status Code 0: Success 1: Point does not exist
Value	STRING	Return string message
Length	DINT	The number of characters read into the STRING variable from the string point's current value

A sample function block implementation of rdstring is illustrated below. It reads the string system point 50100 where the name of the first IEC 61131-3 Resource is stored. The resulting name is stored in the variable application_name upon a successful read operation. The read operation is triggered by setting the BOOL variable read_string to TRUE.

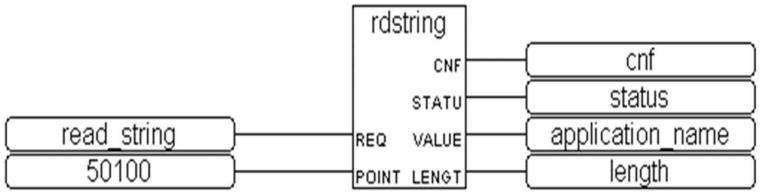


Figure 6.13: rdstring Function Block Example

A structured text implementation of rdstring is listed below. In this example, variable APP_1 contains the point address of the resource running on resource 1. This is the system point number for the first resource name.

```
Read String := TRUE;
APP 1 := 50100;
r string( Read String, 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;
```

4.1.5 Digital Output Controls

Digital output points on the SCADAPack E Smart 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 IEC 61131-3 Resource.

- A IEC 61131-3 Resource may choose to switch a BOOL Output variable on and off with timing through logic. In this case the Resource controls the output point directly. Exact output timing will be dependent on the consistency of the Resource scan rate
- The rtucrob 52 function block may be used to activate an output. In this case the Resource instructs the SCADAPack E Smart RTU point processing task to activate the output point, using the processing tasks' timing, rather than the IEC 61131-3 Resource timing. In general this will produce more accurate pulse timing.

If the digital output point is on a remote DNP3 device through the Data Concentrator, then the pulse command is initiated on the remote device using a DNP3 CROB control. It is recommended that the rtucrob 52 function block be used in this case as it provides flexibility in setting CROB parameters for interoperability with the remote DNP3 device.

4.1.5.1 rtucrob

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

- Control to an output point on the local SCADAPack E Smart 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 Smart RTU Data Concentrator facility.

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

Description

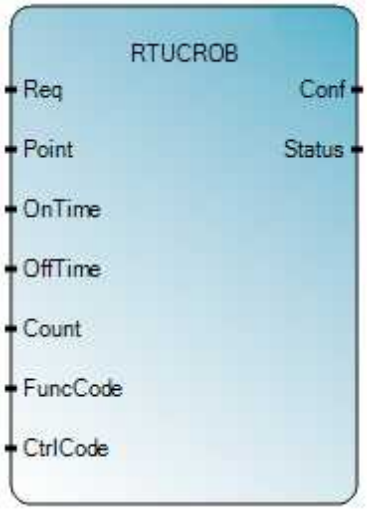
The function block will execute when the req input value transitions from FALSE to TRUE. req needs to be set to FALSE before each subsequent control is issued from this instance of the function block.

The SCADAPack E Smart RTU will not send a control as instructed by this 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 Resource, or uses the point's Output Pulse Time attribute if the onTime is zero.
- The offTime duration is specified by the Resource. 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 the program, the digital output pulse is controlled by the Data Processor Task and is independent of the Resource 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 there is a status code returned or when the DNP3 CROB request is sent to the remote device.



Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Request to pulse an output point Possible Values — Meaning TRUE — pulse initiated on transition to TRUE FALSE — pulse point function disabled
Point	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.
OnTime	Time	Pulse On duration (ms timer format)
OffTime	Time	Pulse Off duration (ms timer format)
Count	DINT	Number of pulse cycles in the pulse train
FuncCode	DINT	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	DINT	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

<table>
<tr><th>OUTPUTS</th><th>TYPE</th><th>DESCRIPTION</th></tr>
<tr><td>Conf</td><td>BOOL</td><td>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.</td></tr>
<tr><td>Status</td><td>DINT</td><td>Status of Request. This should only be read if the conf output is TRUE.

Possible Values — Meaning

-1 — Generic status code

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 SCADAPack E Smart RTU.</td></tr>
</table>

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

4.1.6 Database Point Attributes

These function blocks set database point attributes

- setatr_i 56
- setatr_r 61

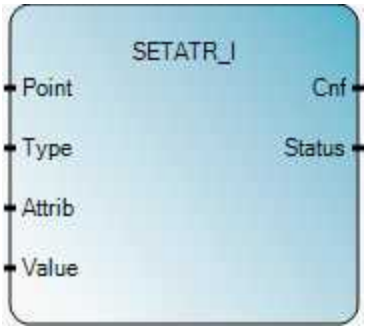
4.1.6.1 setatr_i

Set integer point attributes

Description

Point database attributes represented by integer values may be set using 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 SCADAPack E Smart RTU updates the specified point database field for the specified point number and point type from the value of a variable.



Arguments

<table>
<tr><th>INPUTS</th><th>TYPE</th><th>DESCRIPTION</th></tr>
<tr><td>Point</td><td>DINT</td><td>Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address.

The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the SCADAPack E Target 5 Technical Reference for details.</td></tr>
<tr><td>Type</td><td>DINT</td><td>point data type argument can be one of the defined words below or an integer value corresponding to the data type.

<table>
<tr><th>Defined Word</th><th>Equivalent Numeric Value</th><th>Comment</th></tr>
<tr><td>DIN</td><td>1</td><td>Digital input point</td></tr>
<tr><td>DOUT</td><td>2</td><td>Digital output point</td></tr>
<tr><td>AIN</td><td>3</td><td>Analog input point</td></tr>
<tr><td>AOUT</td><td>4</td><td>Analog output point</td></tr>
<tr><td>CIN</td><td>5</td><td>Counter input point</td></tr>
</table>
</td></tr>
<tr><td>Attrib</td><td>DINT</td><td>Desired point attribute (property) argument can be an defined word or an integer value corresponding to the attribute – see Table 6.6 58.</td></tr>
<tr><td>Value</td><td>DINT</td><td>Desired field value</td></tr>
</table>

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Confirm status ready to write Possible Values — Meaning TRUE — Confirm Valid Status FALSE
Status	DINT	Status of Read Request Possbile Values — Meaning -1 — Unknown Status Code 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
```

Valid Attributes (attrib) for setatr_i Function Block

Attribute (Defined Word)	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

Attribute (Defined Word)	Equivalent Numeric Value	Description	Point Type where this field applies
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 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
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

Attribute (Defined Word)	Equivalent Numeric Value	Description	Point Type where this field applies
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 0 = event disabled 1 = event enabled	Analog
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

Attribute (Defined Word)	Equivalent Numeric Value	Description	Point Type where this field applies
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

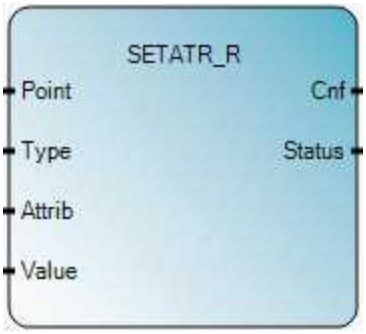
4.1.6.2 setatr_r

Set real point attributes

Description

point database attributes represented by real (floating point) values may be set using 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 SCADAPack E Smart RTU updates the specified point database field for the specified point number and point type from the value of a variable.



Arguments

INPUTS	TYPE	DESCRIPTION
Point	DINT	Point Address can be a numeric Point Address, a variable containing the Point Address or a defined word referencing a point address. The I/O Device Pre-Processor generates Defined Words for variables associated with points in the RTU database. See the

<table>
<tr><td></td><td></td><td>SCADAPack E Target 5 Technical Reference for details.</td></tr>
<tr><td>Type</td><td>DINT</td><td>point data type argument can be one of the defined words below or an integer value corresponding to the data type.

Defined Word | 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</td></tr>
<tr><td>Attrib</td><td>DINT</td><td>Desired point attribute (property) argument can be an defined word or an integer value corresponding to the attribute – see Table 6.7 63.</td></tr>
<tr><td>Value</td><td>REAL</td><td>Desired field value</td></tr>
</table>

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Confirm status ready to write Possible Values — Meaning TRUE — Confirm Valid Status FALSE
Status	DINT	Status of Read Request Possible Values — Meaning -1 — Unknown Status Code 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

Valid Attribute (attrib) Values for setatr_r

Attribute (Defined Word)	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

4.2 Real Time Clock Function Blocks

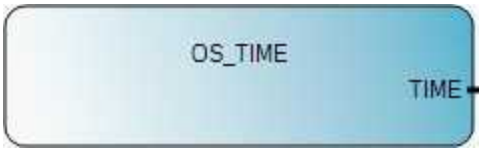
The following function blocks provide a IEC 61131-3 Resource access to the SCADAPack E Smart RTU real time clock.

4.2.1 os_time

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

Description

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



Arguments

OUTPUT	TYPE	DESCRIPTION
TIME	DINT	operating system time (UTC) in seconds since 00:00:00, 1970

4.2.2 loc_time

Return local time since midnight, in milliseconds

Description

This function block returns the local time. If the real time clock is operating in UTC mode, the LOCAL TIME OFFSET FROM UTC system point is applied prior to returning the data. In addition, the SCADAPack E Smart 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" timer format (e.g.. t#23h59m59s999ms).

Arguments

OUTPUTS	TYPE	DESCRIPTION
TIME	Time	local time in milliseconds since midnight

4.2.3 timedate

Return local time and date

Description

This function block returns the local time and date. Return parameters are of type integer. If the real time clock is operating in UTC mode, the LOCAL TIME OFFSET FROM UTC system point is applied prior to returning the data. In addition, the SCADAPack E Smart 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.

Arguments

OUTPUTS	TYPE	DESCRIPTION
HOUR	DINT	Current hour in 24 hour format (0-23)
MINUTE	DINT	Current minute (0-59)
SECOND	DINT	Current second (0-59)
DAY	DINT	Current day (1-7 where 1 = Sunday)
DATE	DINT	Current date (1-31)
MONTH	DINT	Current month (1-12)
YEAR	DINT	Current year (2000-2099)

4.3 DNP3 Peer Communication Function Blocks

The Peer function blocks transfer data directly between the point databases of two peer devices.

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 destination device (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 destination device (or Executed) by the function block “peer_wrx” using that queue.

Two “queue lists” are created for each IEC 61131-3 Resource, one for Read requests and the other for Write requests. These queue lists hold any number of named “point request queues”. For example, a program 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.

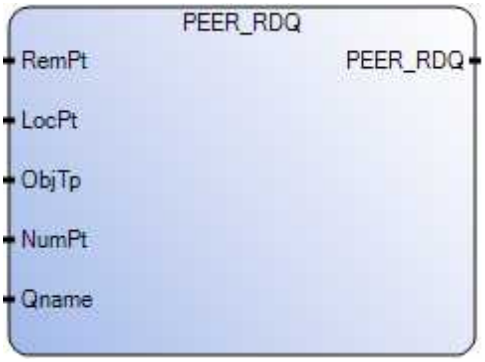
The queues and “queue lists” persist while the Resource is running, but will be cleared when the Resource is stopped or the SCADAPack E Smart RTU is Cold Reset. Points queued using “peer_rdq” need not be re-queued unless the Resource is stopped.

4.3.1 peer_rdq

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

Description

This function adds a specified number of DNP points, to be read from a remote peer and written to a local device, 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 72. The “peer_rdq” function would be typically executed only at IEC 61131-3 Resource startup.



Arguments

INPUTS	TYPE	DESCRIPTION
RemPt	DINT	DNP point number in the destination device to read from valid values are 0-65534.
LocPt	DINT	DNP point number in the local device to write needs to be a valid physical output tor derived point.
ObjTp	DINT	DNP data object to read from peer device. 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	DINT	Number of consecutive points to add to queue. Valid entries are 1 – 65534.
Qname	STRING	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.

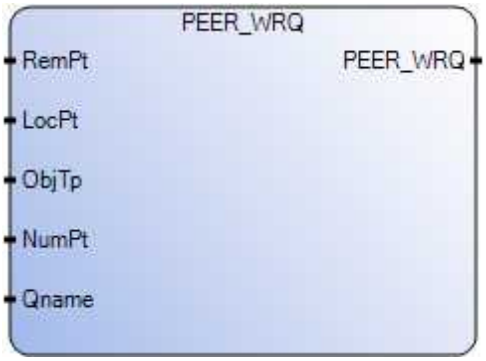
OUTPUTS	TYPE	DESCRIPTION
PEER_RDQ	BOOL	TRUE for successful FALSE for not successful

4.3.2 peer_wrq

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

Description

This function adds a specified number of DNP points, to be copied from the local device onto a Remote device, 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 Resource startup.



Arguments

INPUTS	TYPE	DESCRIPTION
RemPt	DINT	DNP point number in the destination device to write to. Valid values are 0-65534.
LocPt	DINT	DNP point number in the local device where the current value or state is read from needs be a valid input point type.
ObjTp	DINT	DNP data object to write to peer device. The following values are valid for this function: CROB_DirOp AnaOut_32_DirOp AnaOut_16_DirOp AnaOutFloat_DirOp
NumPt	DINT	Number of consecutive points to add to queue. Valid entries are 1 – 65534.
Qname	STRING	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
PEER_WRQ	BOOL	TRUE for successful FALSE for not successful

4.3.3 peer_rdx

Execute queued DNP Read requests

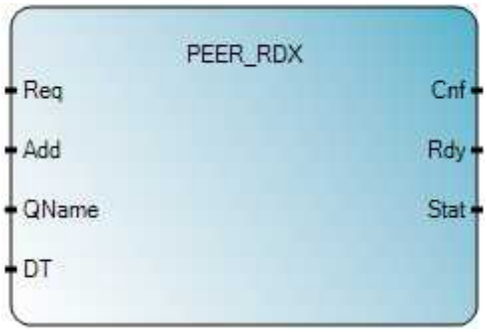
Description

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.

The returned values or states of the remote points are written directly to the local 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 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 a status code. Also see Status Codes 79.

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



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Rdy	BOOL	Data is ready
Stat	DINT	Status = 0 indicates a successful read Status = 255 indicates an outstanding DNP request See Status Codes 79 for other status codes

4.3.4 peer_wrx

Execute queued DNP Write requests

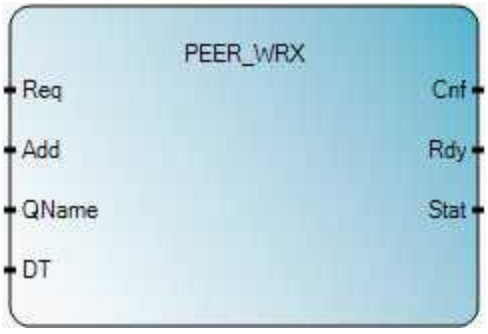
Description

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 points in the Write queue will not have any properties changed.

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.

The returned values or states of the remote points are written directly to the local 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 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 a status code. Also see Status Codes 79.



Arguments

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

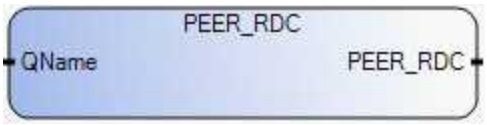
OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise
Rdy	BOOL	Data is ready
Stat	DINT	Status = 0 indicates a successful write (control) Status = 255 indicates an outstanding DNP request See Status Codes 79 for other status codes

4.3.5 peer_rdc

Clear a Read request queue

Description

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



Arguments

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

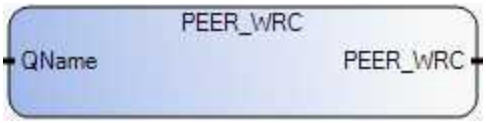
OUTPUTS	TYPE	DESCRIPTION
PEER_RDC	BOOL	TRUE for success otherwise FALSE.

4.3.6 peer_wrc

Clear a Write request queue

Description

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



Arguments

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

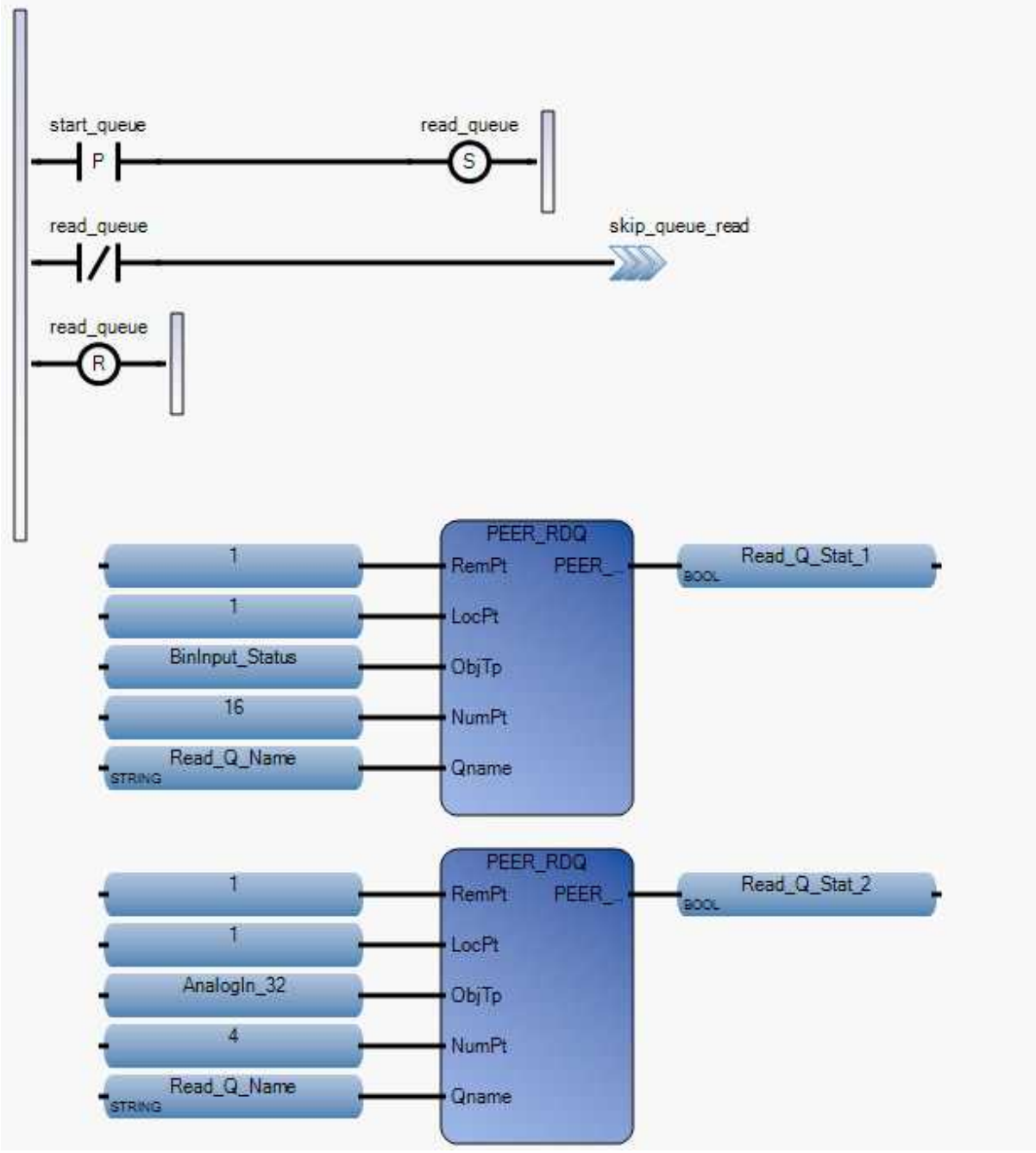
OUTPUTS	TYPE	DESCRIPTION
PEER_WRC	BOOL	TRUE for success, otherwise FALSE.

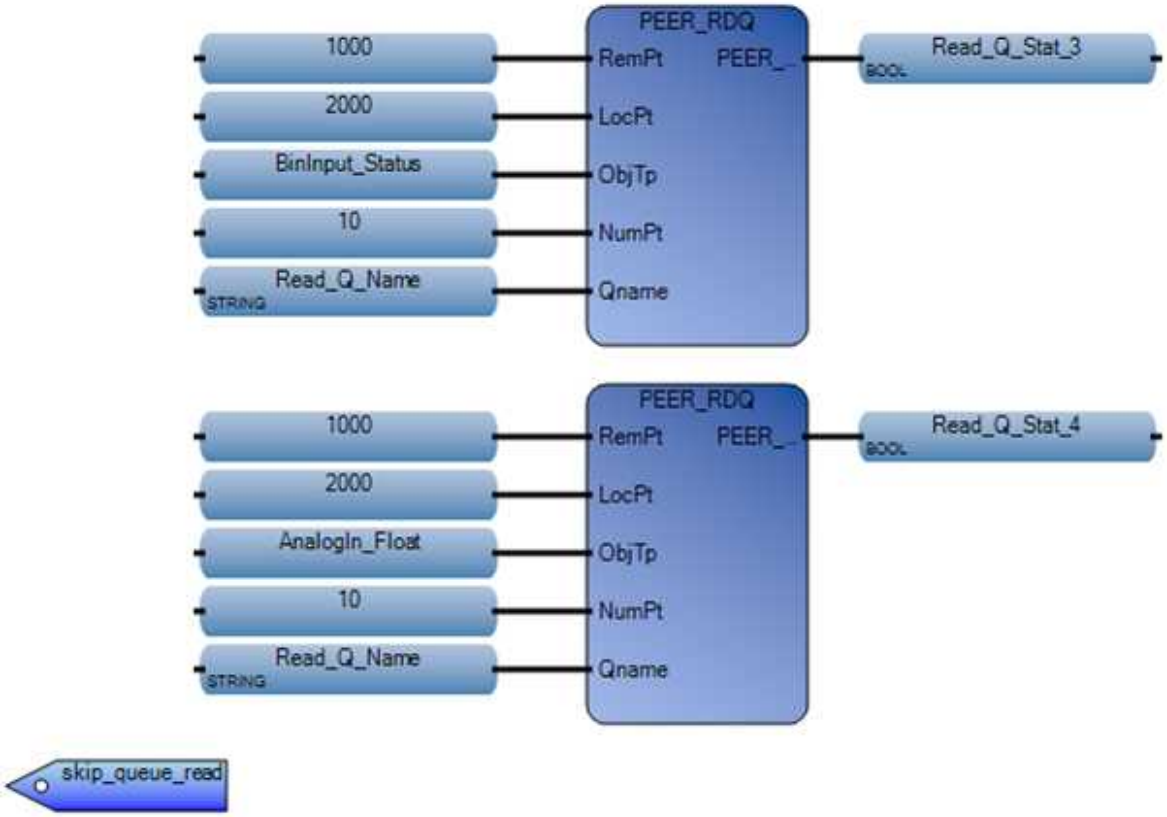
4.3.7 Peer Read Example Program

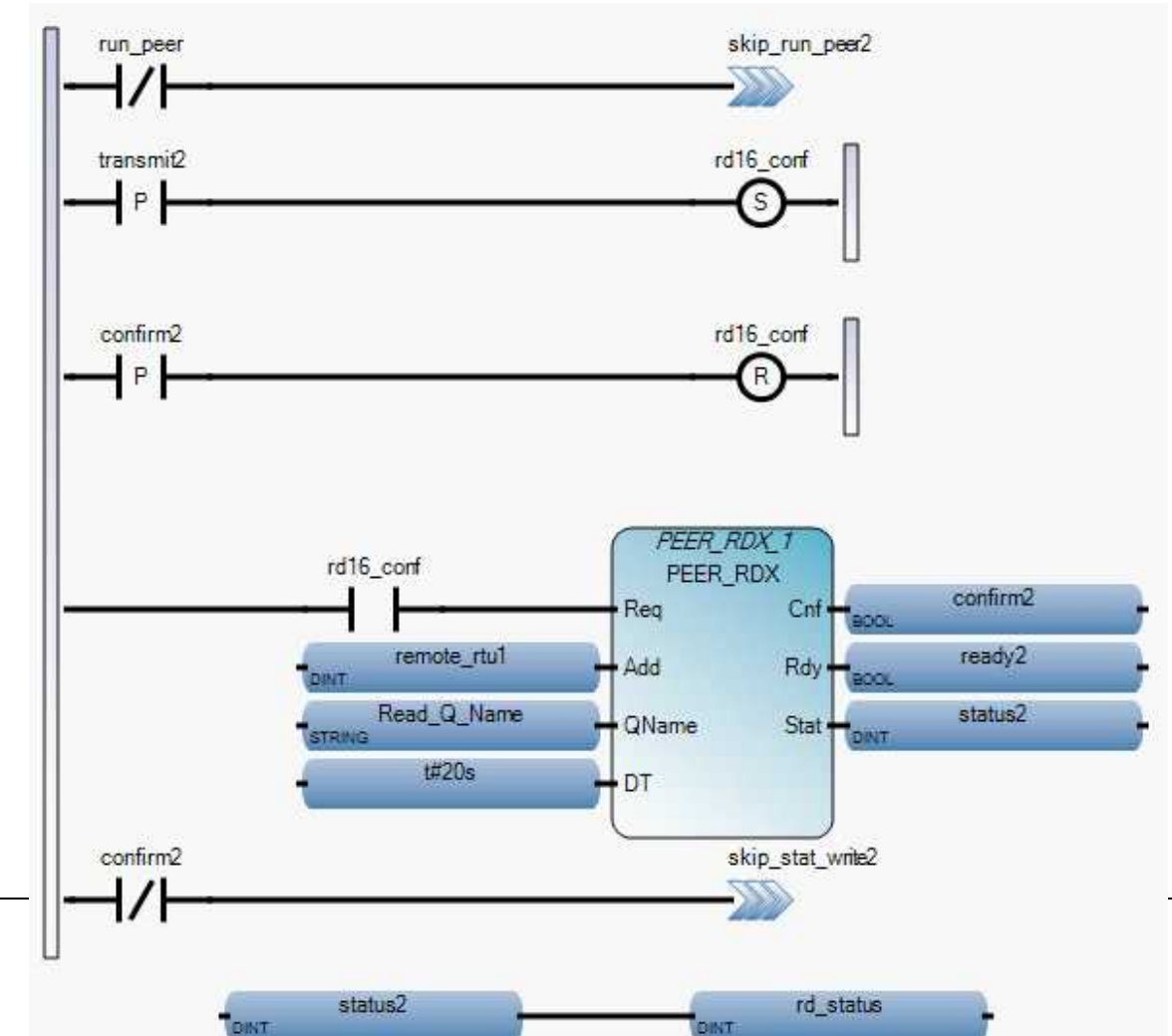
This example shows how a Peer Read test program might be implemented. Setting the Boolean variable “start queue” TRUE causes four Read Queue functions to be executed once only. Following that, setting the “run_peer” variable TRUE will cause the “peer_rdx” to send a DNP Peer read request. The “peer_rdx” 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.

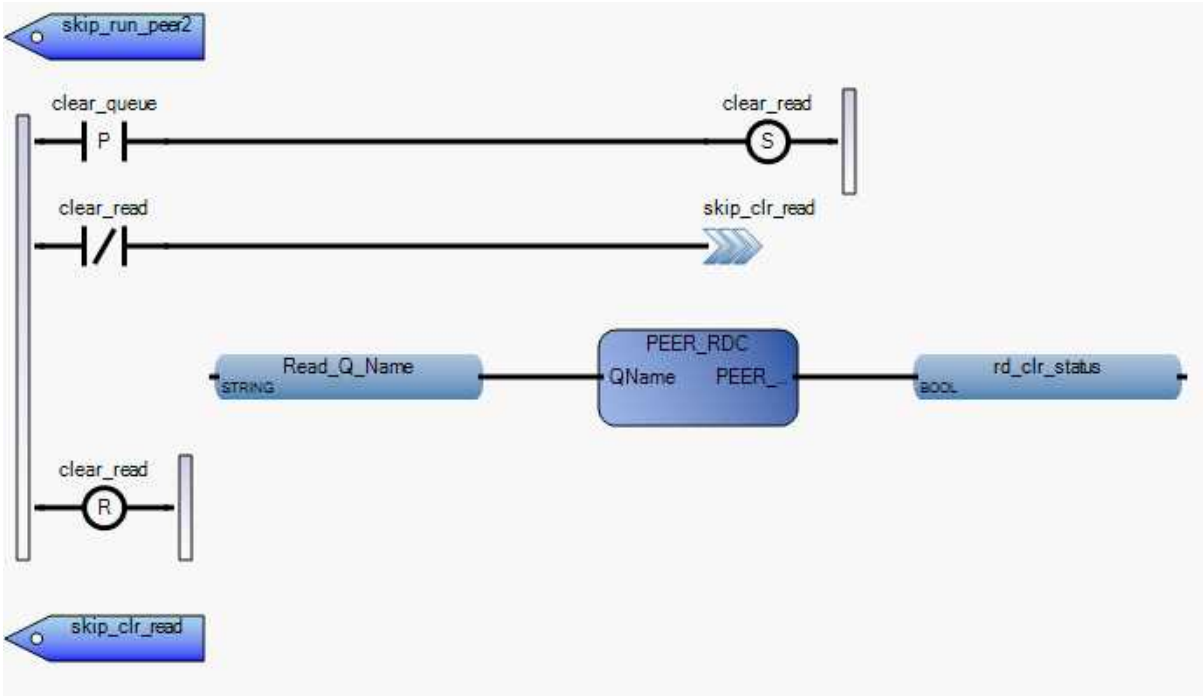
Setting the “clear_queue” Boolean variable TRUE causes the “peer_rdc” function to execute once. This

will clear every Read request in the named queue.




PEER_RDQ
1000
RemPt
PEER_
Read_Q_Stat_3
BOOL
2000
LocPt
BinInput_Status
ObjTp
10
NumPt
Read_Q_Name
STRING
Qname
PEER_RDQ
1000
RemPt
PEER_
Read_Q_Stat_4
BOOL
2000
LocPt
AnalogIn_Float
ObjTp
10
NumPt
Read_Q_Name
STRING
Qname
skip_queue_read


run_peer
skip_run_peer2
transmit2
P
rd16_conf
S
confirm2
P
rd16_conf
R
rd16_conf
PEER_RDX_1
PEER_RDX
Req
Cnf
confirm2
BOOL
remote_rtu1
DINT
Add
Rdy
ready2
BOOL
Read_Q_Name
STRING
QName
Stat
status2
DINT
t#20s
DT
confirm2
skip_stat_write2
status2
DINT
rd_status
DINT



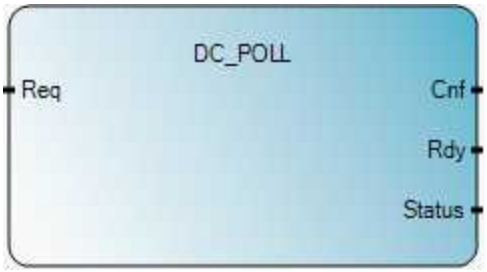
4.3.8 dc_poll

Force an Integrity Poll to configured remote devices.

Description

The IEC 61131-3 Resource can force the Data Concentrator or master device to send a DNP3 Integrity poll (Class 1,2,3,0 poll) to configured remote devices in order to update the local database with a current image of remote device point values. The DC_POLL Function Block is used to send this request.

The function block operates regardless if the Data Concentrator 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 Data Concentrator is disabled, IED responses will not generate local events in the Data Concentrator.



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm: indicates completion of request TRUE => Request Completed FALSE => Request not completed
Rdy	BOOL	Status of data TRUE => Data ready FALSE => Data not ready
Status	DINT	When CNF is TRUE and RDY is FALSE this indicates a table data access status code is active Status = 0 indicates a successful poll Status = 255 indicates an outstanding DNP request See Status Codes 79 for other status codes

4.3.9 Status Codes

These Status Codes are 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	Incorrect control formatting
64	Control operation not supported
65	Control queue full
66	Check control hardware
127	Request could not be added to queue
130	Request not licensed
251	Invalid ObjectType parameter
252	Invalid ObjectType code
253	Point does not exist
254	Invalid ObjectType for this function block
255	Request in progress

4.4 PLC Communications Control Function Blocks

These function blocks provide an interface to disable and enable Read and Write communications to PLC Devices.

4.4.1 mbusctrl

Control Communications to Modbus Device

Description

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

Every I/O device 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.

Arguments

INPUTS	TYPE	DESCRIPTION
Address	DINT	Modbus address of the device to control. An address of 0 will enable/ disable communication to every device on the specified port
Port	DINT	The serial port number that the device is connected to
En_RD	BOOL	When TRUE, PLC device input devices (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	BOOL	When TRUE, data changes on PLC device output devices (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	BOOL	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

4.4.2 mtcpctrl

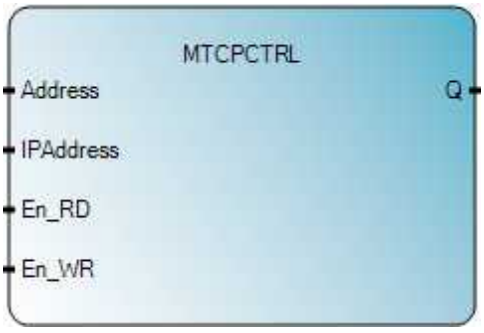
Control Communications to Modbus/TCP or Modbus RTU in TCP Device

Description

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

Every I/O device 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.



Arguments

INPUTS	TYPE	DESCRIPTION
Address	DINT	Modbus address of the device to control. An address of 0 will enable/ disable communication to every device on the specified port
IPAddress	STRING	The IP address that the device is connected to
En_RD	BOOL	When TRUE, PLC device input devices (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	BOOL	When TRUE, data changes on PLC device output devices (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	BOOL	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

4.4.3 df1ctrl

Control Communications to DF/1 Device

Description

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

Every I/O device 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.

Arguments

INPUTS	TYPE	DESCRIPTION
Address	DINT	DF/1 address of the device to control. An address of 0 will enable/disable communication to every device on the specified port
Port	DINT	The serial port number that the device is connected to
En_RD	BOOL	When TRUE, PLC device input devices (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	BOOL	When TRUE, data changes on PLC device output devices (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	BOOL	FALSE if the Address or Port inputs are incorrect; otherwise TRUE

4.5 Serial Port User Communication Functions

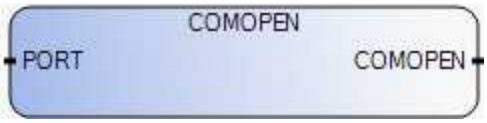
These function blocks provide an interface to setup serial port parameters on the SCADAPack E Smart RTU.

4.5.1 comopen

Open a serial port for use

Description

This function is used to open a serial port for use within a 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 port “Function” configuration needs to be set to “ISaGRAF-User” using the SCADAPack E Configurator before the port can be used within an IEC 61131-3 program. Multiple ports may be configured for ISaGRAF-User and opened for User access by the IEC 61131-3 Resource.



Arguments

INPUTS	TYPE	DESCRIPTION
PORT	STRING	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

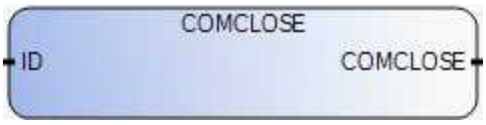
COMOPEN	DINT	Id of serial port. Return value is 0 if port could not be opened.

4.5.2 comclose

Close a serial port

Description

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



Arguments

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

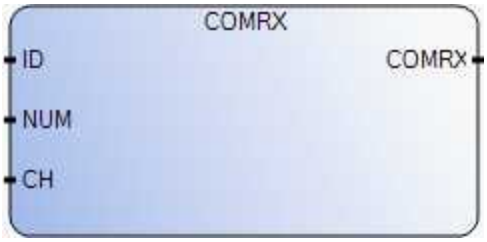
OUTPUTS	TYPE	DESCRIPTION
COMCLOSE	BOOL	TRUE if operation is successful. Else FALSE.

4.5.3 comrx

Read characters from a serial port

Description

This function receives a string of ASCII characters from an open 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:



Arguments

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

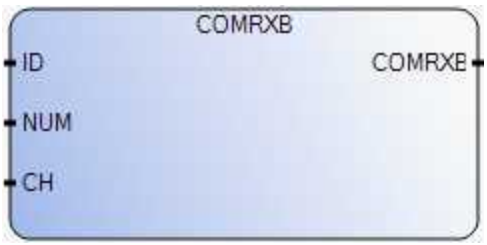
OUTPUTS	TYPE	DESCRIPTION
COMRX	STRING	Character string read from serial port

4.5.4 comrxb

Read binary characters from serial port and display in ASCII

Description

This function receives an array of binary characters from an open 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).



Arguments

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

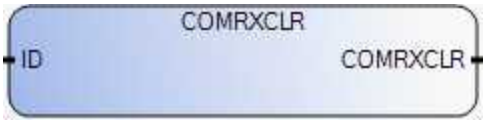
OUTPUTS	TYPE	DESCRIPTION
COMRXB	STRING	Character string representation of the binary protocol from serial port

4.5.5 comrxclr

Clear serial port receive (RX) buffer

Description

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



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
COMRXCLR	BOOL	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.

4.5.6 comtx

Write characters to a serial port

Description

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

Arguments

INPUTS	TYPE	DESCRIPTION
ID	DINT	ID of serial port to close (ID from previous comopen)
IN	STRING	Character string to write to the serial port
LEN	DINT	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
COMTX	BOOL	TRUE if operation successful.

4.5.7 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 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.

Arguments

INPUTS	TYPE	DESCRIPTION
ID	DINT	ID of serial port to close (ID from previous comopen)
IN	STRING	Character string to convert to binary and write to serial port
LEN	DINT	Number of characters to write. (2* the number of binary characters to write).

OUTPUTS	TYPE	DESCRIPTION
COMTXB	BOOL	TRUE if operation successful.

4.5.8 getport

Read a DNP3 RS-232 line state

Description

The GETPORT 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.

Arguments

Inputs	Data type	Notes
REQ	BOOL	Data Read request: initiate data transfer when asserted (rising edge)
PORT	UINT	DNP3 Serial Port Number
PARAM	UINT	Which DNP serial port hardware line to read. Use one of the following defines: CTS, DCD, DSR These are defined in the Global Data Types - Defined Words tab.

Outputs	Data type	Notes
CNF	BOOL	Data transfer confirm: indicates completion of request
STATUS	UINT	Transaction status value: Success = 0, Invalid Port = 1, Invalid Param. = 2
LSTATE	BOOL	The state of the serial port hardware line. True if the hardware line is asserted.

4.5.9 setport

Set a DNP3 port RS-232 line state

Description

The SETPORT 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.

Arguments

Inputs	Data Type	Notes
REQ	BOOL	Data Write request: initiate data transfer when asserted (rising edge)
PORT	UINT	DNP3 Serial Port Number
PARAM	UINT	Which DNP serial port hardware line to control. Use one of the following defines are valid: RTS, DTR These are defined in the Global Data Types - Defined Words tab.
LSTATE	BOOL	The new state of the serial port hardware line. True if the hardware line is to be asserted.

Outputs	Data Type	Notes
CNF	BOOL	Data transfer confirm: indicates completion of request
STATUS	UINT	Transaction status value: Success = 0, Invalid Port = 1, Invalid Param. = 2

4.6 Process Function Blocks

These function blocks interact with physical processes. They include Proportional-Integral-Derivative (PID) control, flow accumulation and flow totalizing features.

4.6.1 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.

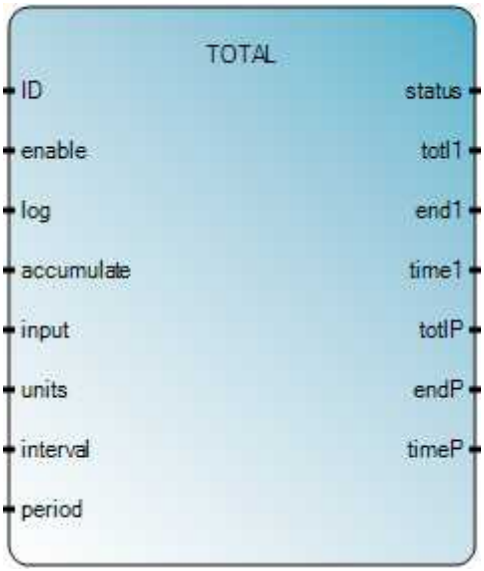
Stored data is retained when the program is stopped, and when the controller is powered off or reset.

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 each IEC 61131-3 Resource.

The total function block appears as follows.



Arguments

Inputs	Type	Description
ID	DINT	Identifies the internal totalizer (1 to 32).
enable	BOOL	Enables accumulation and creates log. FALSE to TRUE transition = create and initialize log TRUE = enable accumulation FALSE = delete log and clear outputs
log	BOOL	Log Data FALSE to TRUE = log data
accumulate	BOOL	Accumulate flow ON = accumulate flow
input	REAL	Rate Input
units	DINT	Input rate period 0 = seconds 1 = minutes 2 = hours 3 = days
interval	DINT	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	DINT	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	DINT	Status 0 = normal 1 = invalid ID 2 = invalid Units input 3 = invalid Interval input 4 = invalid Period input
totl1	REAL	Total for period 1
end1	DINT	Time at end of period 1 Seconds from January 1, 1970
time1	DINT	Accumulation time for period 1 Time in seconds
totlP	REAL	Total for specified period
endP	DINT	Time at end of specified period Seconds from January 1, 1970
timeP	DINT	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.

4.6.2 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.

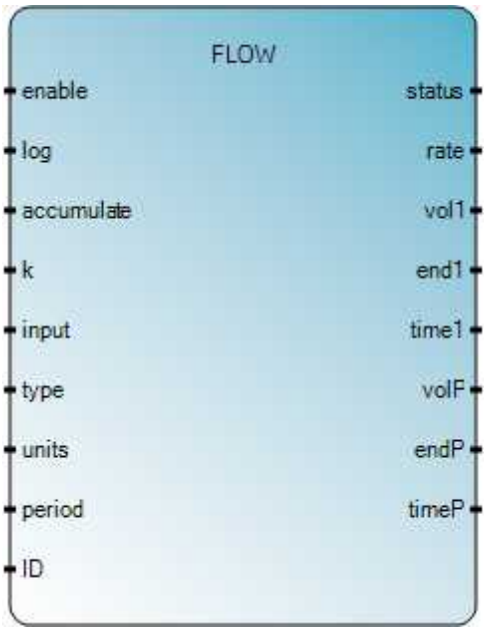
When the accumulator enabled input is ON, accumulation and rate calculations are enabled. When the input is OFF, accumulators, stored data and the outputs are set to zero.

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 each IEC 61131-3 Resource.

The flow accumulator function block flow appears as follows.



Arguments

Inputs	Type	Description
enable	BOOL	TRUE = enable accumulation FALSE = clear outputs
log	BOOL	Log Data FALSE to TRUE transition = log data
accumulate	BOOL	Accumulate flow TRUE = accumulate flow
k	REAL	K factor F needs to be a positive value.
input	DINT	Pulse Input
type	DINT	Input type 0 = 32 bit running counter 1 = 32 bit difference
units	DINT	Rate period 0 = seconds 1 = minutes 2 = hours 3 = days
period	DINT	Specified record to be displayed (1 to 35).
ID	DINT	Identifies the internal flow accumulator (1 to 32).

Outputs	Type	Description
status	DINT	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	DINT	Time at end of period 1 Seconds from January 1, 1970
time1	DINT	Flow time for period 1 Time in seconds
volP	REAL	Volume for specified period
endP	DINT	Time at end of specified period Seconds from January 1, 1970
timeP	DINT	Flow time in specified period Time in seconds

Notes

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.

When a Resource is stopped and restarted, the flow function block will calculate the elapsed time from the last sample (before the Resource was stopped) until new the start time (after the Resource is started). This time is added to the flow time.

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.

4.6.3 pidd

Discrete Output PID Function Block

Description

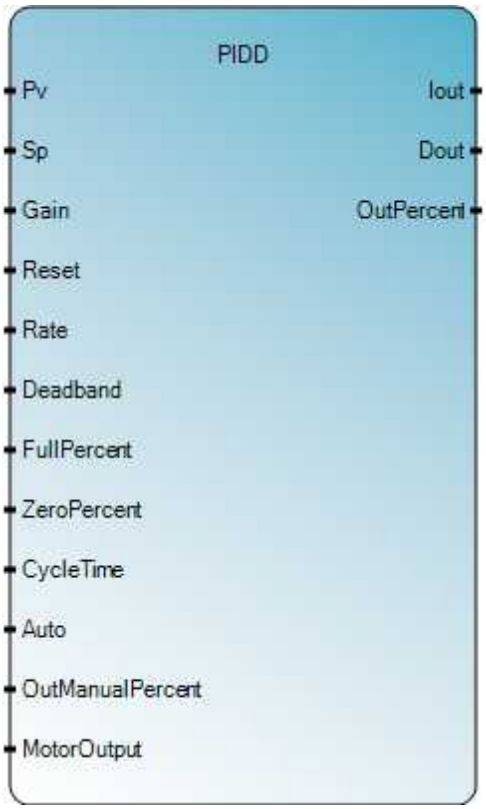
The pidd function block performs a PID algorithm and controls two discrete outputs. The output is a duty cycle of a BOOL value. The duty cycle depends on the value of the output. The increase output, iOut, is cycled when the outputPercent is greater than zero. The decrease output, dOut, is cycled when 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 IEC 61131-3 Resource, the sampling period is the period of the IEC 61131-3 Resource.

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.
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 IEC 61131-3 Resource, the scan time of the IEC 61131-3 Resource becomes the PID cycleTime.
Auto	BOOL	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

Inputs	Type	Description
		function block is in the manual mode.
MotorOutput	BOOL	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	BOOL	The increase output. This output cycles when outPercent is greater than zero.
Dout	BOOL	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
Ti = 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 103

4.6.4 pida

Analog Output PID Function Block

Description

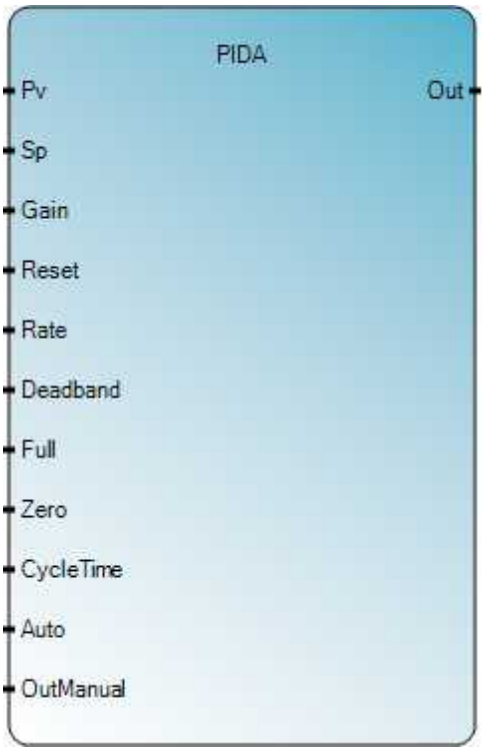
The pida function block performs a PID algorithm and calculates an analog output.

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 IEC 61131-3 Resource, the sampling period is the period of the IEC 61131-3 Resource.

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 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

Inputs	Type	Description
		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 IEC 61131-3 Resource, the scan time of the IEC 61131-3 Resource becomes the PID cycleTime.
Auto	BOOL	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
Ti = 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 Aso

pidd 99

4.7 Miscellaneous Function Blocks

These miscellaneous function blocks provide a variety of control and communication functions.

4.7.1 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 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 65). If a zero value is entered for the time input parameter, the 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.

Hint:	If IEC 61131-3 logic is to be used to write value changes to the specified point before forcing the event use the DNP3 communication function blocks to update the point values (where the ObjectType input is set to Local_RTU_Data), as opposed to using output devices. The operating system will then process the value update before the "force event" request.

Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Force events request. This invokes a request to force a event on the specified point when asserted (rising edge).
Index	DINT	This input specifies the point number of the configuration point.
Type	DINT	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	DINT	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 is specified as UTC time. If a 0 value is entered, the operating system will timestamp the event with the current time.

OUTPUTS	TYPE	DESCRIPTION
Sts	DINT	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.

4.7.2 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 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 65).

- If a zero value is entered for the TimeSec input parameter, the 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.

Hint:	If IEC 61131-3 logic 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 output devices. The operating system will then process the value update before the "force event" request.

Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Force events request. This invokes a request to force a event on the specified point when asserted (rising edge).
Index	DINT	This input specifies the point number of the configuration point.
Type	DINT	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	DINT	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 operating system will timestamp the event with the current time and the msTime input is ignored.
msTime	DINT	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	DINT	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.

4.7.3 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 SCADAPack E Smart RTU can record a value and timestamp for a point when a DNP3 event is generated. This information is then available 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 resource scans, only the recent value and timestamp will be available to the 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.

Arguments

INPUTS	TYPE	DESCRIPTION
Point	DINT	specifies the point number of the analog point.
Mode	DINT	0 = MODE_RTC: The timestamps presented represent the time at which the function block outputs are updated. Using this mode along with the elevated resource task priority (see Section rtuparam 115) 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. The SCADAPack E Smart RTUchecks 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 point database

		using the points CURRENT ENGINEERING VALUE property. STRING string containing converted value.
Time	Time	The time-stamp at which the current value occurred (milliseconds since midnight). The SCADAPack E Smart RTU 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	DINT	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 = Unsuccessful

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.

4.7.4 cmd_exec

Execute a Command Line Command

Description

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

Arguments

Inputs	Data Type	Notes
iReq	BOOL	Command Execute request: initiate command when asserted (rising edge)
iCmd	STRING	Command to be Executed (See below for acceptable command list)

Outputs	Data Type	Notes
oRdy	BOOL	Command execute confirm: indicates completion of request
oSts	DINT	Transaction status value: Success = 0 Incomplete Command = 1 Command Not Found = 2 Command Unsuccessful = 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 information
COPY	Duplicate a file
DEL, DELETE	Delete a file
REN, RENAME	Rename a file
RESTART	Restart facilities
GETCONFIG	Generate an configuration file
FILEDIAG	Enable/Disable File Diagnostics
SYSDIAG	Enable/Disable System File Diagnostics
PLCDIAG	Enable/Disable PLC File Diagnostics

4.7.5 rtuparam

Modify a specific RTU parameter

Description

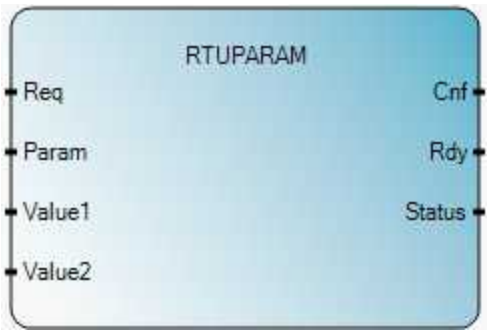
This function block provides the ability to modify SCADAPack E Smart RTU operational parameters.

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

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

When the SCADAPack E Smart 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 SCADAPack E Smart 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 SCADAPack E Smart RTU was configured. The global defined word "USE_RTU_DEFAULT" is also provided for this purpose.



Arguments

INPUTS	TYPE	DESCRIPTION

Req	BOOL	Data transfer request initiated on rising edge.
Param	DINT	Name of parameter to modify
Value1	DINT	Parameter dependent value. See Table 1 below 117
Value2	DINT	Parameter dependent value. See Table 1 below 117

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Rdy	BOOL	Data transferred successfully.
Status	DINT	0 for success when Cnf and Rdy are TRUE. Otherwise see status codes in Table 2 below 122.

Table 1: 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 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 SCADAPack E Smart RTU
APPL_EVENT_TO	5	Secs -1 = return to configured value	0 = Master 1 1 = Master	Changes DNP event confirm timeout to the appropriate DNP3 Master session.

PARAM	PARAM Value*	Value1	Value2	Comment
			1 2 = Master 2 3 = Master 3	
MY_DNP_ADDRESS	6	New DNP Address = 0 to 65529 -1 = return to configured value		Changes the following DNP3 addresses: • Slave address when communicating with Master 1 • DNP address when sending Peer requests • DNP address when sending 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 SCADAPack E Smart 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	0 = Master 1 1 = Master 1 2 = Master 2	Changes the DNP enabled Unsolicited Event Classes “on-line” without a restart. This functionality may be useful in a dual-redundancy change-over situation

PARAM	PARAM Value *	Value1	Value2	Comment
		Class 1,2 = 6 Class 1,3 = 10 Class 2,3 = 12 -1 = return to configured value	3 = Master 3	
SYS_ERR_CODE	10	100-999		User status code number sent to Status Code system analog point 50020
ISA_TASK_PRI	11	0 = Normal Priority 1 = High_Priority		Increases this Resource task's priority in the operating system. Requires that the IEC 61131-3 Resource be configured with "Trigger cycles" in the Run Time Options' Cycle Timing settings. Also requires the IEC 61131-3 Resource actually scans faster than the configured Cycle Timing.
DISCONNECT_PORT	12	0 = Port0 1 = Port1 2 = Port2 3 = Port3 4 = Port4		Requests that the serial port driver task associated with the supplied port number 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.

PARAM	PARAM Value *	Value1	Value2	Comment
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 IEC 61131-3 Resource. 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 DNP3 Technical Reference for a description of these modes.

PARAM	PARAM Value *	Value1	Value2	Comment
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”.
		2 = Sampler		Restarts Sampler task. Refer to the 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 status 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 port number. 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 STRING_TO_IP 132 function.

++ On SCADAPack 300E controllers, PORT_DISABLE on Port0 disables the USB Peripheral Port. 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 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 a IEC 61131-3 logic 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 2: RTUPARAM Status Codes

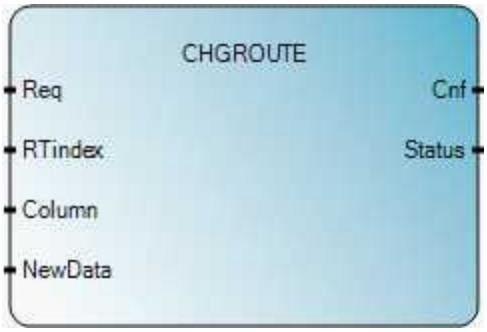
RTUPARAM STATUS	DESCRIPTION
-1	Unsuccessful
0	Operation successful
1	Information not found
2	Bad Point Type
3	Unknown attribute for this point
4	Bad value for this attribute
5	Invalid attribute for this point
6	Invalid argument

4.7.6 chgroute

Modify entries in the DNP3 Routing Table

Description

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



Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Data transfer request initiated on rising edge.
RTindex	DINT	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	DINT	Route Table Column number (0-7). See Table 1 123
NewData	DINT	New value for route table entry.

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Status	DINT	0 for success when Cnf is TRUE. Otherwise see Status Codes 79

Table 1: 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

4.7.7 chgrtnum

Change DNP3 Routing Table Connect Number String

Description

This function block provides the ability to modify the “Connect No.” string in the DNP3 Routing Table of the SCADAPack E Smart 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 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).

Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Data transfer request initiated on rising edge.
Dest	DINT	Destination DNP node address to search (0-65535)

Dial	STRING	Connect No. (message string) to transfer to the matching route entry. E.g. Dial number, IP address

OUTPUTS	TYPE	DESCRIPTION
Cnf	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise.
Status	DINT	0 for success when Cnf is TRUE. Otherwise see Status Codes 79

4.7.8 chgrtprt

Change DNP3 Routing Table port number

Description

This function block provides the ability to modify the destination port number in a DNP3 Routing Table. The CHGRTPRT function block operates in a similar way to the 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 (DINT analog).

Arguments

INPUTS	TYPE	DESCRIPTION
Req	BOOL	Data transfer request initiated on rising edge
Dest	DINT	Destination DNP node address to search (0-65535)
Port	DINT	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	BOOL	Data transfer confirm TRUE indicates completion of request. FALSE, otherwise
Status	DINT	0 for success when Cnf is TRUE. Otherwise see Status Codes 79

4.8 TCP/IP Interface Functions

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

- ip_add 127
- ip_del 129
- ip_cycgw 130
- ip_ping 131
- string_to_ip 132
- ppp_echo 133

4.8.1 ip_add

Add an IP Routing Table Entry

Description

This function provides an interface to the TCP/IP facilities in the SCADAPack E Smart 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 these route entry types, listed in order of descending implementation:

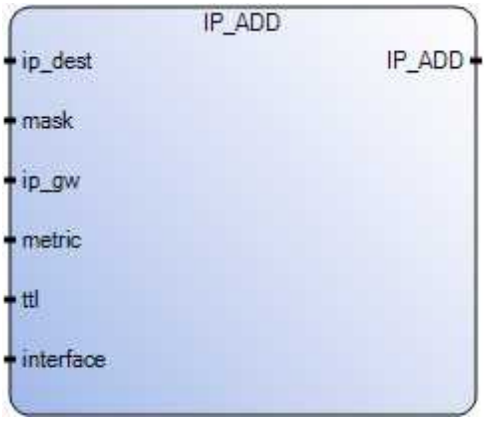
- DEFAULT GATEWAY

- 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 the SCADAPack E Smart RTU is restarted or powered off.



Arguments

INPUTS	TYPE	DESCRIPTION
ip_dest	STRING	Destination IP Address for route

mask	STRING	Destination Subnet Mask for route
ip_gw	STRING	Gateway host IP Address for route
metric	DINT	route metric (cost) for added route. (0 = use default interface metric)
ttl	TIME	Time to Live route entry. (t#0 = static route: no lifetime)
interface	DINT	Number of IP interface (E.g. 0=Port0, 1=Port1, etc.).

OUTPUTS	TYPE	DESCRIPTION
IP_ADD	DINT	Indicates status code when adding to IP Table. 0 = success; For other status and non-execution codes, refer to SCADAPack E RTU TCP/IP Status (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*.

4.8.2 ip_del

Delete an IP Routing table entry

Description

This function interfaces to the TCP/IP facilities in the SCADAPack E Smart 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 configuration, command line ROUTE ADD command or 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 a restart or power on.

Arguments

INPUTS	TYPE	DESCRIPTION
ip_dest	STRING	Destination IP Address for route
mask	STRING	Destination Subnet Mask for route

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

4.8.3 ip_cycgw

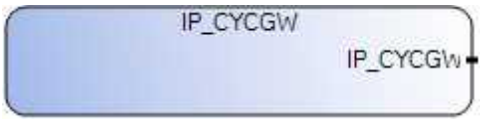
Cycle default IP gateway route

Description

This function interfaces to the TCP/IP facilities in the SCADAPack E Smart 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 SCADAPack E Smart RTU is restarted or powered on.



Arguments

OUTPUTS	TYPE	DESCRIPTION
IP_CYCGW	DINT	Indicates status code when cycling between default gateway IP routes. 0 = success

4.8.4 ip_ping

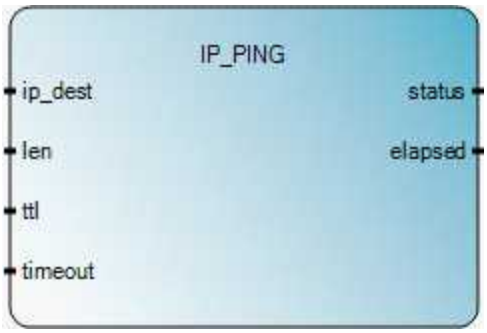
Ping a remote IP node

Description

This function block interfaces to the TCP/IP PING Client facilities in the SCADAPack E Smart 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 ip_ping function block could be used by a program executing in the second IEC 61131-3 Resource, so as not to slow the performance of a main control programs executing in the first IEC 61131-3 Resource.

This function block executes synchronously with the Resource execution scan, and may significantly increase the IEC 61131-3 Resource scan rate, particularly if the remote IP hose does not reply to the PING request.



Arguments

INPUTS	TYPE	DESCRIPTION
ip_dest	STRING	Destination IP address of remote host
len	DINT	Number of bytes to send in ping request. (needs to be less than 1500)
ttl	TIME	Time To Live for ping packet on IP network. (t#0 = default TTL)
timeout	TIME	Time to wait for remote IP host to respond

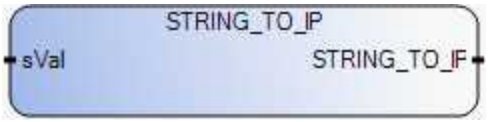
OUTPUTS	TYPE	DESCRIPTION
status	DINT	Indicates status code for Ping request. 0 = success
elapsed	TIME	Elapsed time between Ping Response and Ping Request. Valid when STATUS = 0

4.8.5 string_to_ip

Convert a string to an IP address

Description

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



Arguments

INPUTS	TYPE	DESCRIPTION
sVal	STRING	TCP/IP address as a string in dotted decimal format.

OUTPUTS	TYPE	DESCRIPTION

STRING_TO_IP	DINT	TCP/IP address as an DINT 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, string_to_ip('192.168.0.218'), 0);
END_IF;
```

4.8.6 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 Smart RTU. The ppp_echo function sends an LCP ECHO command to the nominated PPP interface on an SCADAPack E Smart RTU.

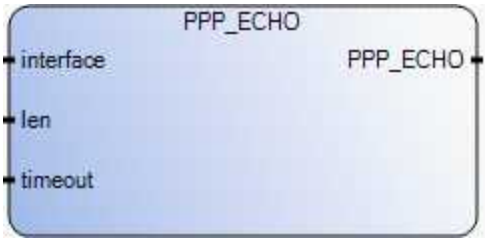
A successful status (0) indicates PPP link “UP” state on the serial interface.

PPP_ECHOmay return a timed-out status (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 Smart RTU, see the TCP/IP Technical Reference Manual.

This function executes synchronously with the Resource execution scan, and may significantly increase the IEC 61131-3 Resource 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 a program executing in the second IEC 61131-3 Resource, so as not to slow the performance of a main control application executing in the first IEC 61131-3 Resource.



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
PPP_ECHO	DINT	Indicates status code for PPP ECHO operation. 0 = success

4.9 Alarm Group Functions & Function Block

Alarm functions and function blocks provide a mechanism for grouping individual 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 points.

These interfaces are handled through independent functions and function blocks. As detailed below, the point numbers registered for alarm summaries need not necessarily have to be imported in to the IEC 61131-3 Resource through an input device.

- almadd 135
- almproc 136
- almload 140
- almclr 141

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

For information on alarm processing see the Data Processing Technical Reference manual.

4.9.1 almadd

Add a point to an alarm group

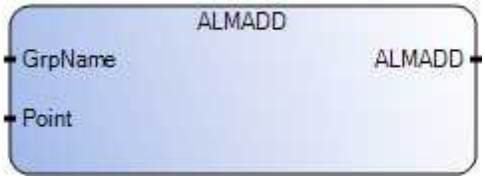
Description

Creating and configuring the Alarm Group is performed by a call to the almadd Function.

Typically the function calls to configure an alarm group are executed during the start-up phase of a IEC 61131-3 Resource. Once executed at startup, this code no longer needs to be executed while the IEC 61131-3 Resource 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 status code 3.



Arguments

INPUTS	TYPE	DESCRIPTION
GrpName	STRING	INPUTS of alarm group (a string)
Point	DINT	Digital Input or User Point number.

OUTPUTS	TYPE	DESCRIPTION
ALMADD	DINT	Indicates status code status: -1 = Internal Status Code 0 = Success 1 = Point does not exist 2 = Invalid Group 3 = Point already exists in Group

Structured Text Example

This function dynamically creates an alarm group and dynamically adds points to the alarm groups. Apart from upper memory limitations of the SCADAPack E Smart RTU 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.

```
IF (almadd('ProtectionFault', POINT_P_PHA_TRIP) = 0) THEN
      Status1 := almadd('ProtectionFault',POINT_P_PHB_TRIP);
      Status2 := almadd('ProtectionFault',POINT_P_PH_TRIP);
END_IF;
```

4.9.2 almproc

Process a summary alarm group

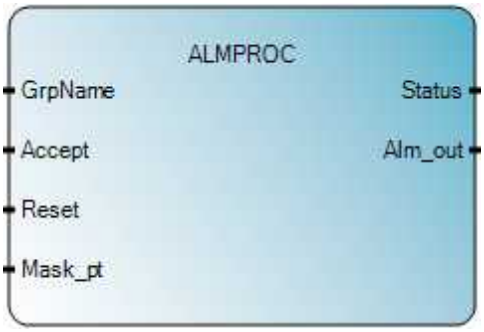
Description

Processing (execution) of the Alarm Group is performed by a call to the ALMPROC Function Block.

Once configured in a one-off execution of IEC 61131-3 logic using calls to the almadd function block instance (as described in Section almadd 135 above), a call to an almproc function block would typically occur each scan. The rate of execution could be controlled through additional logic if required, but optimal responsiveness is achieved through execution each 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 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 IEC 61131-3 Resource 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 140 below.



Arguments

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

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

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

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 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.

Point Attribute	Description	Effect on ALARM activation
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.
Alarm Inhibit	Inhibits the “Point-in-Alarm” property being activated, regardless of the input condition.	Stops the point going in to the 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 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 point database) and the 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 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 device 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 a status 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 a BOOL Output variable that updates a point in the point database. In this example it is assumed that the Oper_ACCEPT and Oper_RESET inputs are latched variables on input devices from 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 scan.

4.9.3 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 using the almload 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 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 Digital User points. If the MASK_PT input is zero, a status code is returned. The number of digital points read corresponds to the number of alarms in the Alarm Group.

Arguments

INPUTS	TYPE	DESCRIPTION
GrpName	STRING	String value identifying alarm group.
Mask_pt	DINT	digital user point number. 0 = No mask point

OUTPUTS	TYPE	DESCRIPTION
ALMLOAD	DINT	-1 = Internal Status Code 0 = Success 1 = Point does not exist 2 = Invalid group

4.9.4 almclr

Destroy an alarm group

Description

A previously created Alarm Group can be removed using the ALMCLR function. Calling this function frees 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. Alarm group functions and function blocks referencing this alarm group after an ALMCLR will result in unsuccessful status codes being set.

Arguments

INPUTS	TYPE	DESCRIPTION
GrpName	STRING	String value identifying alarm group.

OUTPUTS	TYPE	DESCRIPTION
ALMCLR	DINT	-1 = Internal Status Code 0 = Success 2 = Invalid group

4.10 File System Interface Functions

This section describes the functions and function blocks that support access to the SCADAPack E Smart RTU file system. Many of the functions and function blocks detailed in this section return an analog status value.

Functions that provide similar functionality to the command line interface for file system access are detailed in Section File System Management Functions 142.

The directory / drive information function blocks are detailed in Section Directory Information Function Blocks 152 and the file read / write functions are detailed in Section File Read / Write Functions 155.

Refer to Section File System Access Status Codes 162 for a list of file system status codes.

4.10.1 File System Access Functions

This functions allow IEC 61131-3 logic to manage the file system. Equivalent functionality provided by the SCADAPack E Smart RTU command line commands are indicated where applicable.

Function Name	Equivalent Command Line command	Purpose
F_DEL 142	DEL	Delete Files
F_DELTREE 143	DELTREE	Delete a directory and files
F_COPY 144	COPY	Copy a file
F_JOIN 145	JOIN or APPEND	Append files
F_REN 146	RENAME	Rename a file
F_MKDIR 147	MD	Make a directory
F_RMDIR 147	RMDIR	Remove a directory
F_CD 148	CD	Change current working directory
F_DSKSEL 149	DSKSEL	Select a Disk device
F_PWD 150		returns the Present Working Directory
F_DV_RDY 151		indicates if a disk Device is Ready

4.10.1.1 F_DEL

Delete a file from the file system

Description

The F_DEL function is used to delete the specified file from the file system from within a program.

Arguments

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

OUTPUTS	TYPE	DESCRIPTION
F_DEL	DINT	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 IEC 61131-3 logic using F_CD function (see Section F_CD Function 148).

The function returns a 0 if the delete request was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to Section File System Access Status Codes 162 for the range of possible status codes and their descriptions.

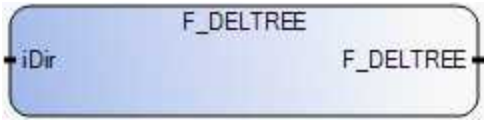
4.10.1.2 F_DELTREE

Delete a directory and its contents

Description

The F_DELTREE function is used to remove the specified directory and files in that directory. Subdirectories and files below the specified directory are also deleted.

The function returns a 0 if the delete tree request was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to Section File System Access Status Codes 162 for the range of possible status codes and their descriptions.



Arguments

INPUTS	TYPE	DESCRIPTION
iDir	STRING	Directory to delete

OUTPUTS	TYPE	DESCRIPTION
F_DELTREE	DINT	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.

4.10.1.3 F_COPY

Copy a file

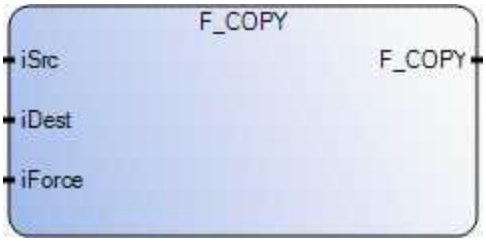
Description

The F_COPY function is used to copy the specified source file to a new file in the file system whose name is specified as the destination filename.

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 IEC 61131-3 logic using F_CD function (see Section F_CD Function 148).

The function returns a 0 if the copy request was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to Section File System Access Status Codes 162 for the range of possible status codes and their descriptions.



Arguments

INPUTS	TYPE	DESCRIPTION
iSrc	STRING	Source file to be copied
iDest	STRING	Filename for the new copied file
iForce	BOOL	Forces copy over existing files

OUTPUTS	TYPE	DESCRIPTION
F_COPY	DINT	Status of Delete Request

4.10.1.4 F_JOIN

Append a file to another file

Description

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 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.

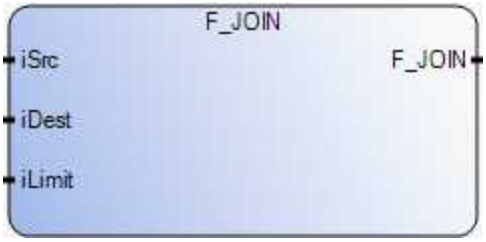
The iLimit argument specifies the maximum size of the destination file. If this argument is zero, the source file(s) will be unconditionally appended to the destination file.

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 IEC 61131-3 logic using F_CD function (see F_CD Function 148).

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

The function returns a 0 if the join request was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status 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 status code. In this specific case the function returns 0.



Arguments

INPUTS	TYPE	DESCRIPTION
iSrc	STRING	Source file(s) to be appended. The string may include a wildcard specification, e.g. SF*
iDest	STRING	File name to which the source file(s) is appended.
iLimit	UDINT	‘Optional’ maximum size for the destination file (0 = no limit)

OUTPUTS	TYPE	DESCRIPTION
F_JOIN	DINT	Status of Request

4.10.1.5 F_REN

Rename a file

Description

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

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 iOld and iNew 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 IEC 61131-3 logic using F_CD function (see F_CD Function 148).

The function returns a 0 if the rename request was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.

Arguments

INPUTS	TYPE	DESCRIPTION
iOld	STRING	Existing filename
iNew	STRING	New filename

OUTPUTS	TYPE	DESCRIPTION
F_REN	DINT	Status of Request

4.10.1.6 F_MKDIR

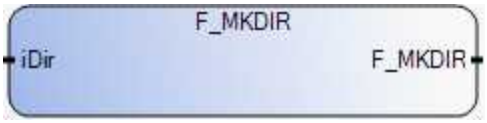
Create a directory

Description

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 IEC 61131-3 logic using F_CD function (see F_CD Function 148). The F_MKDIR function supports creation of directories on other drives, i.e. different to the current drive.

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 unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.



Arguments

INPUTS	TYPE	DESCRIPTION
iDir	STRING	Directory to create

OUTPUTS	TYPE	DESCRIPTION
F_MKDIR	DINT	Status of Request

4.10.1.7 F_RMDIR

Remove a directory

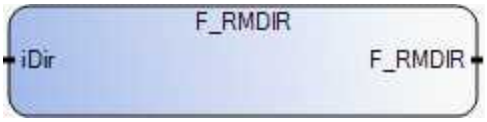
Description

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 IEC 61131-3 logic using F_CD function (see F_CD Function 148). The F_RMDIR function supports removal of

directories on other drives, i.e. different to the current drive.

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 an unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.



Arguments

INPUTS	TYPE	DESCRIPTION
iDir	STRING	Directory to remove

OUTPUTS	TYPE	DESCRIPTION
F_RMDIR	DINT	Status of Request

4.10.1.8 F_CD

Change the current working directory

Description

The F_CD function is used to change the current working directory within a given drive. If the 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.

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 unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.

Arguments

INPUTS	TYPE	DESCRIPTION
iDir	STRING	Target Directory

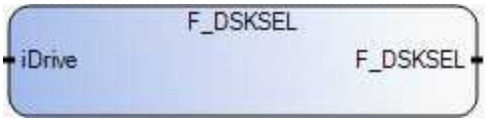
OUTPUTS	TYPE	DESCRIPTION
F_CD	DINT	Status of Request

4.10.1.9 F_DSKSEL

Change the working drive device

Description

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



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
F_DSKSEL	DINT	Status of Request. 0 = Success. See File System Access Status Codes 162 for the range of possible status codes and their descriptions

The following disk devices are available:

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. log files, sampler files, etc)
E:	External CompactFLASH Drive	up to 2 GB	-	External drive available for user data (not available on SCADAPack 300E controllers)
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 controllers)

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 148.

4.10.1.10 F_PWD

Display the current working directory

Description

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

Arguments

OUTPUTS	TYPE	DESCRIPTION
F_PWD	STRING	Present (current) Working Directory.

4.10.1.11 F_DV_RDY

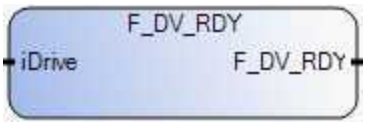
Determine if drive is ready

Description

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:

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.



Arguments

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

OUTPUTS	TYPE	DESCRIPTION
F_DV_RDY	BOOL	Status of Request

4.10.2 Directory Information Function Blocks

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

- FINDFILE Function Block 152
- DIR_INFO Function Block 154

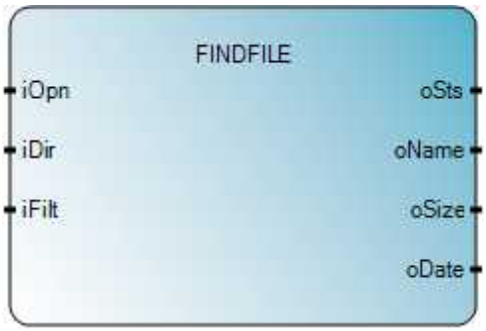
4.10.2.1 FINDFILE

Search for a file

Description

The FINDFILE function block searches 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”, or the “oldest file”. 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”. 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.

All instances of the FINDFILE function block in the IEC 61131-3 Resource 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 a Resource.



Arguments

INPUTS	TYPE	DESCRIPTION
iOpn	DINT	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).
iDir	STRING	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	STRING	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.

OUTPUTS	TYPE	DESCRIPTION
oSts	DINT	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 unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.
oName	STRING	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	UDINT	specifies the size of the detected file in bytes and is only valid if the oSts output indicates success, i.e. 0.
oDate	UDINT	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.

4.10.2.2 DIR_INFO

Read directory information

Description

The DIR_INFO function block reads 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

Arguments

INPUTS	TYPE	DESCRIPTION
iDir	STRING	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.
iFilt	STRING	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.

OUTPUTS	TYPE	DESCRIPTION
oSts	DINT	specifies which status of the directory information search. The function block oSts value returns 0 if the information search was successful. If an unsuccessful operation was detected, a negative DINT value is returned. Refer to File System Access Status Codes 162 for the range of possible status codes and their descriptions.
oNum	UDINT	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	UDINT	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	UDINT	specifies the total number of bytes available in the file system. The presented value is only valid if the oSts output indicates success.

4.10.3 File Read / Write Functions

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

- F_WOPEN 156
- F_ROPEN 157

4.10.3.1 F_WOPEN

Open a file in write mode

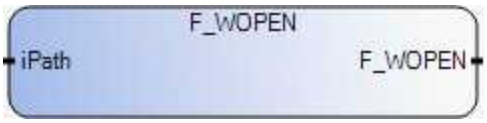
Description

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 iPATH 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 F_WOPEN 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.



Arguments

INPUTS	TYPE	DESCRIPTION
iPath	STRING	Filename to open in write mode (may include path)

OUTPUTS	TYPE	DESCRIPTION
F_WOPEN	DINT	File ID (file handle)

4.10.3.2 F_ROPEN

Open a file in read mode

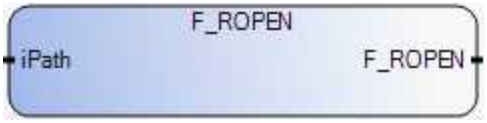
Description

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 iPATH 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 F_ROPEN 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.



Arguments

INPUTS	TYPE	DESCRIPTION
iPath	STRING	Filename to open in read mode (may include path)

OUTPUTS	TYPE	DESCRIPTION
F_ROPEN	DINT	File ID (file handle)

4.10.3.3 F_CLOSE

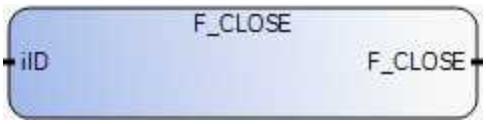
Close an open file

Description

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

The iID 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 F_CLOSE return parameter presents a BOOL status for the file close request. TRUE is returned is the file close is OK, otherwise FALSE is returned.



Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
F_CLOSE	BOOL	Status of file close request

4.10.3.4 F_EOF

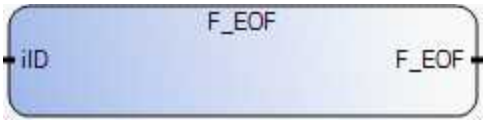
Test for end of file

Description

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

The iID 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 F_EOF return parameter presents a BOOL 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.



Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
F_EOF	BOOL	Status of request

4.10.3.5 FA_READ

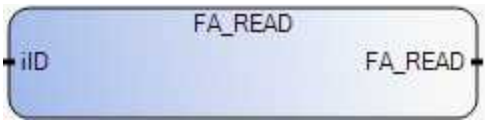
Read analog values from a binary file

Description

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 iID 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 FA_READ return parameter returns the DINT analog value read from the file.



Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
FA_READ	DINT	DINT value read from file

4.10.3.6 FA_WRITE

Write analog value to binary file

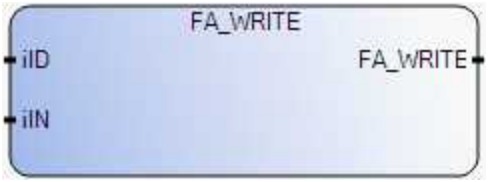
Description

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 iID 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 FA_WRITE return parameter returns the status of the file write request. TRUE is returned if the file write was successful, otherwise FALSE is returned.



Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)
iIN	DINT	DINT value to be written to file

OUTPUTS	TYPE	DESCRIPTION
FA_WRITE	BOOL	Status of file write request

4.10.3.7 FM_READ

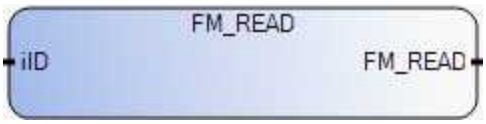
Read a string from a binary file

Description

The FM_READ function reads STRING 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 iID 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 FM_READ return parameter returns the string read from the file.



Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)

OUTPUTS	TYPE	DESCRIPTION
FM_READ	STRING	Variable read from file

4.10.3.8 FM_WRITE

Write a string to a binary file

Description

The FM_WRITE function writes a STRING 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 iID 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 FM_WRITE return parameter returns the status of the file write request. TRUE is returned if the file write was successful, otherwise FALSE is returned.

Arguments

INPUTS	TYPE	DESCRIPTION
iID	DINT	File ID (file handle)
iIN	STRING	STRING variable to be written to file

OUTPUTS	TYPE	DESCRIPTION
FM_WRITE	BOOL	Status of file write request

4.10.4 File System Status Codes

Many file system functions and function blocks return an DINT status code. The status code values are detailed in the following table.

Status Value	Description
0	Success
-1	Unknown Status Code
-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	Could not create working buffer

-1010	Could not write 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