U.P.M.A.C.S.

User Programmable Monitor, Alarm, and Control Software

version 6.2
Development System

DEVELOPER'S MANUAL

UPMACS Communications Inc.

Current for Development System v6.2.0

Table of Contents

THE U.P.M.A.C.S. DEVELOPMENT SYSTEM

Introduction

The U.P.M.A.C.S. Development System enables you to design and develop device drivers and station files for U.P.M.A.C.S., the monitor, alarm, and control software by UPMACS Communication Inc. This manual assumes that you have read the manual for the U.P.M.A.C.S. Operate System, or are otherwise familiar with it.

The U.P.M.A.C.S. Development System allows you to create and edit two types of files: U.P.M.A.C.S. station files (*.upmacs-station) files and U.P.M.A.C.S. device driver libraries (*.upmacs-drivers) files. To create U.P.M.A.C.S. image libraries (*.upmacs-images) files, use the U.P.M.A.C.S. Image Editor included with this package.

Station files are used by the Operate System to monitor and control a station. Device driver libraries are not used by the Operate System. They are used to store ready made device drivers that can then be added to a station file in the Development System.

You can edit one file at a time with the Development System. To edit more than one file at a time, run the Development System multiple times.

The formats for station files and device driver libraries are compatible. You can save any device driver library as a station file, and you can save any station file as a device driver library, as long as it contains only device drivers. This means that there is no need for separate "New Station File" and "New Device Driver Library" menu items. The type of a new file is first determined when you save it.

The U.P.M.A.C.S. Development System supports entering special characters like the degree symbol ("°") and the micro symbol ("µ") using special key combinations. See *Appendix C: Entering Special Characters* on page 181 for details.

HOW U.P.M.A.C.S. WORKS

Structure of an U.P.M.A.C.S. Station

The U.P.M.A.C.S. station files contain a complex database structure that describes the functionality of a station. There are several different types of top-level database objects, and each may contain one or more different kinds of embedded objects. Each top-level object has a tag and a name. The tag is used by other database objects, and by SCL programs, to refer to the object. The name is used to identify the object to the user.

There are six types of top-level objects in an U.P.M.A.C.S. database:

- Device drivers
- Serial ports
- Registers
- SCL programs
- Screens
- SABus requests

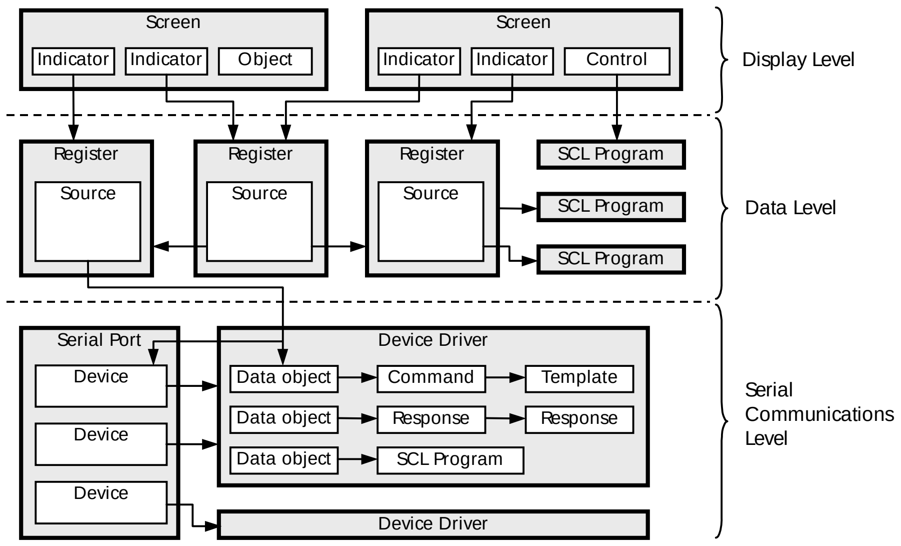
Some of these objects can contain other objects:

Device drivers contain data objects, commands, and device driver SCL programs

Serial ports contain devices

Screens contain static objects, indicators, controls, and labels

An U.P.M.A.C.S. station usually has a structure that looks something like this:



(The station shown here has no uplink port communication capabilities. The SABus Request objects are described under *Uplink Port Communications* on page 7.)

Serial Communication

Serial Communication Objects

A brief description of the objects used for serial communications is given below. The objects are described in more detail in their respective sections.

- Serial Ports

U.P.M.A.C.S. communicates with equipment via the serial ports of the computer. These ports include the built-in ports (COM1 and COM2), ports on a multiport serial extension card, as well as any special serial ports available to the computer (like ports on a network port server).

- The Polling Sequence

U.P.M.A.C.S. normally uses a sequence of polls to communicate with the equipment. Each poll sends a command to a piece of equipment, and waits for a response, if applicable. Once the response has been received, the next poll in the sequence is sent, and so on, until all polls have been sent. The polling sequence is then started over. Each serial port has exactly one polling sequence to service all equipment attached to it.

- Devices

Each serial port contains a number of devices. Each device represents one piece of equipment, like an up converter or an amplifier. It contains information about the equipment, as well as data objects to hold the information returned by the equipment.

- The Device Initialization Sequence

Each device may contain an initialization sequence. The initialization sequence consists of a series of commands sent to the equipment to initialize it.

- Device Drivers

Each device in a port uses a device driver to specify the behaviour of the equipment. There is one device driver for each type of equipment, but devices with the same characteristics (make, model) share a device driver. Device drivers are not tied to individual serial ports; devices on different ports can share the same device driver. Device drivers can have parameters that are specified by the device that uses the driver. Most drivers, for example, let the device specify a device address.

Device drivers can be stored in device driver libraries (*.upmacs-drivers files). See *Device Driver Libraries* on page 144 for details on using device driver libraries.

- Commands

Each device driver contains a number of commands. A command contains a description of the data that is sent to the computer, as well as a description of the data returned by the equipment, if any.

- Data Objects

Each device driver contains a number of data objects, containing the information returned by the equipment in response to a command. Data objects are used by register sources and SCL programs to access the data returned by the equipment.

The Polling Process

Most equipment sends data only in response to a command. U.P.M.A.C.S. will send commands to the equipment continuously, and evaluate the data returned. Which commands are sent to what piece of equipment on the port, and in what order, is determined by the polling sequence.

The polling sequence consists of a series of polls. Each poll in the polling sequence contains a reference to a command in a device. The command, in turn, contains a description of the format of the response data, if any.

Each poll is executed as follows:

- ➢ Send the command's data string to the equipment

- Wait for the response, if necessary
- Check the data received from the equipment against the command's response format
- If the data is OK, update all data objects that depend on the command
- Update all registers that depend on data objects whose values have changed

Some commands may not have a response. For polls with such commands, only the first step applies.

Port Access Synchronization

Since U.P.M.A.C.S. is multitasking, access to the serial ports must be synchronized between the different parts of the program, to prevent two commands from being sent to the same port at the same time.

Usually, port synchronization is done automatically by U.P.M.A.C.S. If an SCL program sends a command to a serial port, it has to wait until any polls that are currently being sent on the port have finished. The polling sequence is then interrupted until the SCL program has finished sending the command. There are two exceptions to this rule, however:

1. An SCL program might need to send several commands in a row without being interrupted. Usually, if an SCL program sends two commands to a port, a poll from the polling sequence will be sent in between to ensure that the port will continue to be serviced. An SCL program can, however, request exclusive access to a serial port, in which case no polls will be sent until the program has released the ports. See also *Serial Communication* in the *SCL Language Reference*.
2. It might be important that the polling sequence is not interrupted by an SCL program between two particular polls. This is usually the case if the first poll is a routing command to a routing device. You can specify that a command in the polling sequence is a routing command that routs a number of polls. U.P.M.A.C.S. will ensure that the routed commands will be sent immediately after the routing command, without being interrupted by an SCL program.

Storage and Processing of Information

U.P.M.A.C.S. uses the data received from device responses, as well as data from other sources to report certain states, or trigger certain actions. U.P.M.A.C.S. might display a certain output power as a result of a response received from a device, or it might switch a piece of equipment to standby when an alarm occurs.

Registers

All infomation in U.P.M.A.C.S. is stored in registers. The data contained in a register is called its value. Registers support a wide variety of methods for acquiring their data, as well as automatic controls, logging, and much more. U.P.M.A.C.S. has four types of registers:

- bistate registers
- digital registers
- analog registers
- string registers

- Bistate registers

Bistate registers contain data that can be expressed as ON or OFF. They can thus have two states, which is why they are called bistate registers. A bistate register might reflect the alarm condition of a piece of equipment, a switch position for a two-position switch, or the local-remote setting of a unit.

- Digital registers

Digital registers contain data that can be expressed as more than two states. A digital register can hold up two 2^{32} states, more than four billion. Each state is assigned a number between 0 and 4,294,967,295. A digital register might reflect the state of a three-position switch, or a channel selection between 1 and 10.

You can assign a name to each of a digital register's value, describing what the value means. The values of a register that holds the modulation type of a modem might be named "BPSK", "QPSK, "8PSK", and "16AQM", for example. This makes it easier to remember which value corresponds to which setting.

Since the states of digital registers are expressed as numbers, they can also be used to hold numerical data, as long as the range of values is from 0 to 4,294,967,295, and only integers are used. A digital register can thus be used as a counter, for example.

- Analog registers

Analog registers hold numerical data. The values have a practically unlimited range, and can include fractions and negative numbers. Analog registers also support calibration of data. An analog register will typically hold data like output power, temperature, or frequency.

Analog registers can also hold a whole set of numbers, rather than just a single one. You can use analog registers to hold such information as the trace of a spectrum analyzer or the history of some analog value.

- String registers

String registers hold text data or arbitrary binary data. A string register might contain a message received from equipment, or it might simply store some data for future reference.

Sources

Each register has a source. The source tells the register where the data it represents comes from. Most sources get the data from a data object in a device on a serial port. Other sources summarize the information contained in one or more other registers. There are a great number of different sources that get the data from different places.

SCL Programs

SCL programs are powerful tools used in many circumstances. The most common use of SCL programs is to perform actions if the user presses a control button or a command is received on an uplink port. SCL programs can also be used in register sources to decode the data received from a piece of equipment, or to perform actions when a message is received from a device. SCL programs can also be used to perform actions at startup, at regular intervals, or at scheduled times (e.g. on the 1st of every month).

Device drivers can also contain SCL programs. These programs are executed by regular SCL programs using the `DRVCALL` and `DRVRUN` SCL commands. See *Overview Of SCL Programs* in the *Developing Device Drivers* manual for more information on device driver programs.

User Interface

U.P.M.A.C.S. communicates with the user through a graphical user interface. The information in the registers is communicated to the user on the screen, and the user can click on buttons on the screen to trigger certain actions.

Screens

U.P.M.A.C.S. organizes its visual data in screens. A screen contains information about everything displayed in a single window or dialog. A screen can contain several types of objects:

- static objects
- indicators
- controls
- labels

Static Objects

Static objects always look the same, and never change their appearance. Static objects can be lines, splines, rectangles, ellipses, text, or images. Images are loaded from the image library.

There is a special kind of static object called a 3D object. 3D objects have an etched, raised, or sunken appearance.

Indicators

Indicators reflect the value of one or more registers. There are eight types of indicators:

- bistate indicators
- multistate indicators
- digital indicators
- analog indicators
- string indicators
- dials
- graphs
- X-Y position markers

Bistate indicators change the way they look depending on the ON/OFF state of any register. Bistate indicators can be lines, splines, rectangles, ellipses, text, or images. Certain properties like colour, fill style, and line thickness vary according to the alarm condition.

Multistate indicators are similar to bistate indicators, but they reflect the value of a digital register.

Digital indicators also reflect the value of a digital register, but they display the value directly as a number, in a variety of formats. The text properties, like bold and italic, as well as the colour, can change according to the ON/OFF state of the register.

Analog indicators are similar to digital indicators, but they display the value of an analog register.

String indicators are also similar to digital indicators, but they display the value of a string register as ASCII text.

Dials reflect the value of an analog register using a slider or coloured bar.

Graphs show the values of an analog register with a size of more than one value as a bar or line graph.

X-Y position markers are images that change their position within a region of the screen depending on the values of two analog registers. X-Y position markers can also change their appearance depending on the ON/OFF state of any register, similar to bistate indicators.

Controls

Controls are buttons that the user can press. When a control button is pressed, it can perform one of two actions:

- execute an SCL program, or
- show a screen

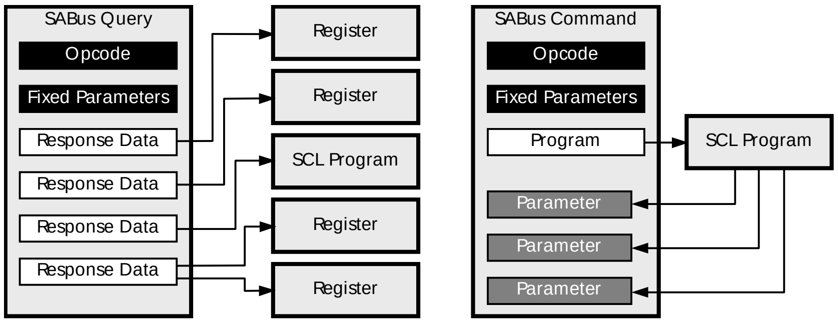
Labels

Labels are advanced objects. They are just like static text objects, with the added ability to include the title of a register in the text. This is useful for user configurable registers, where the user can set the title of a register from within the Operate System using the "Configure Data" dialog box.

Uplink Port Communications

Other systems can poll U.P.M.A.C.S. for register status and execute SCL programs over special serial ports called uplink ports. The uplink ports protocol is based on Scientific Atlanta's SABus protocol standard. The exact protocol is described in Appendix D on page 182.

There are two types of SABus Requests: Queries and Commands.



Queries

When U.P.M.A.C.S. receives a query, it automatically generates a response from the values of certain registers. You must specify which registers are used, and how the values are encoded. You can also specify an SCL program that supplies some or all of the data. The response is then sent automatically. Queries are typically used to let remote systems access status and alarm data.

Commands

When U.P.M.A.C.S. receives a command, it calls an SCL program to process the command. You can specify a list of parameters that need to be in the command packet; and U.P.M.A.C.S. will decode the commands and let you access them from within the SCL program. U.P.M.A.C.S. does not send a response to commands; the SCL program must generate and send the response itself. Queries are typically used to let remote systems perform certain actions, like switching states or setting parameters. You can, however, also use a command to provide information to a remote system, if you need complete control over the response data.

Designing an U.P.M.A.C.S. Station

Every station developer will have his own preferred way of designing a station. The following are guidelines to get a new user on his way.

The first step in developing an U.P.M.A.C.S. station database is defining the protocol to be used to communicate with the equipment. The protocol is defined in device drivers.

Step 1: Device Drivers

You need to create a device driver for each type of equipment with which you want U.P.M.A.C.S. to communicate. If you have equipment that supports daisy chaining on a single port, you do not have to write a separate driver for each device address. Device drivers can take parameters (like a device address), which are specified when the driver is used.

It is not always necessary to develop new device drivers for a new station file. You can import existing device drivers from device driver libraries.

See the *Developing Device Drivers* manual for information on how to develop device drivers.

Step 2: Serial Ports

Define one serial port for every physical port to which equipment is connected. Add a device for each piece of equipment on the port.

You must then specify the polling sequence. A port need not contain a polling sequence; if you do not create any polls, the port will simply remain idle until an SCL program sends a command.

Step 3: Registers

Define one register for every piece of information you want to display. If you need to keep track of additional data internally, also create registers for that purpose. Some register sources use SCL programs, in which case you must write the program before creating the register source.

Avoid using more that one register that contains the same piece of information.

Step 4: SCL Programs for Controls and SABus Commands

Write an SCL program for each type of action the user can take. If you have several pieces of identical equipment, you do not need to write separate control programs for each. Write one program, and use program arguments to specify which equipment the command is to be sent to.

Please remember that the user usually expects to see a confirmation dialog when he presses a button. Make sure to include a confirmation in your program. The confirmation box should tell the user exactly what will happen, and should warn him of potential unexpected results.

Programs are not automatically logged. If you desire logging of controls, you must include code to log the commands that are sent. Only log a command that was actually executed, if the user cancels a command, it is usually undesirable to log it.

Make all messages to the user clear, understandable, and unambiguous. Avoid convoluted sentences, avoid the passive voice, and avoid sentence fragments. Use simple grammar and proper punctuation. A good guideline for user messages is to use the same language you would use to speak to somebody in person. To test if a message is well written, ask yourself if this is how you would say it if you were standing face to face with the user.

If a third-party system needs to be able to poll U.P.M.A.C.S. via an uplink port, you must also create one SCL program for each action the third-party system can take. SABus commands, unlike user control commands, should not require a confirmation.

Step 5: Screens

Lastly, create the screens. You will usually want to provide a screen containing an overview of the entire station (usually a block diagram), which will be the screen visible initially. For large stations, you may also want to have overview screens for different sub systems.

If your station is very small, you can show all information on the main screen. For larger station, show detail screens for groups of equipment, or individual pieces of equipment. Provide buttons on the main screen and subsystem screens to switch between screens easily.

Screens come in two flavors: Windows and dialogs. Windows can be resized, tiled, maximized, minimized, and more than one window can be displayed at the same time, and the user can navigate between them. Dialogs are "modal," that is, they cannot be resized, maximized, or minimized, and the user cannot switch to a different screen when viewing a dialog screen.

Window screens are usually preferable because of their higher flexibility, but too many windows can become confusing to the user. Use dialog screens for equipment details, especially in large stations.

When designing your screens, keep the following things in mind:

- Avoid clutter. Leave sufficient space between objects. If information does not easily fit on one screen, use two screens.
- Avoid large screens. A screen should usually be entirely visible when maximized. If you use large screens, try to arrange them so that scrolling is required in one direction only, preferably the vertical.
- Only use a custom background colour of the screen in special situations. Using a custom background colour keeps the user from setting his preferred background colour in Windows' *Display* control panel.
- Use normal black 14pt Window font for most normal text. Avoid light or medium colours for text, as they are not easily readable. Avoid text smaller than 14pt.
- Arrange Objects in a logical fashion. Use clear and unambiguous labels. You can group objects using rectangles or lines. A rectangle around a piece of text is more effective in highlighting it than large or bold text.
- Try to provide error and masked states for all indicators. Otherwise, the indicator might disappear when equipment is disconnected, and the screen might look confusing.
- When you test your screens in the Operate System, look at them with and without control privileges. Make sure no superfluous objects remain visible when the control buttons are hidden, and that no important captions disappear.

Remember that the operator will be mostly looking at your screens. Working with even the best-designed earth station with the latest in equipment can become very frustrating if the M&C system is confusing or behaves inconsistently. Operators will usually spend more time looking at your screens than doing anything else.

Step 6: SABus Requests

> Note: It is usually not necessary to create SABus requests. You only need to create SABus requests if you require that a third-party system can poll U.P.M.A.C.S. via an uplink port.

Create an SABus query for every set of data that the remote system can request. Each request can contain one or more register values or alarm states, as well as other data generated using an SCL program. If the request needs parameters (a request for a switch position might specify the switch number, e.g.), you must either specify a separate request object for each possible combination of parameters, or use an SABus command instead of a query. The disadvantage of using commands instead of queries is that you must assemble and format the response data yourself, whereas U.P.M.A.C.S. does the work for you if you use a query.

Create an SABus command for every action that the third-party system can take. Unlike queries, it is not necessary to create a command for each possible combination of parameters that the command takes. You can create different command objects for different sets of parameters, but you can also specify additional parameters that U.P.M.A.C.S. will decode and place in SCL variables, and you can access the values of these parameters from within the command program.

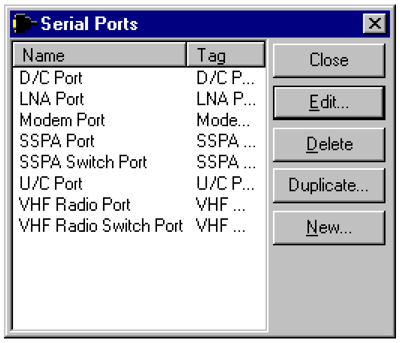
You can also create SABus commands for requests for information. This is necessary if you cannot or do not want to create a query for every possible set of parameters. Since commands can decode parameters and make their values available to you in their program, you can write an SCL program that determines which information should be included in the response based on those parameters, and constructs the response accordingly. The disadvantage of this method is that you have to collect and format the response data yourself. However, some of the work in assembling the response is done for you: you do not have to worry about the start and end characters, the device address and opcode, or the checksum. These “wrapper” elements of the response are added automatically by U.P.M.A.C.S., you only have to worry about the data. Please remember to check the “Allow execution without signing on” check box in the program’s properties window, or the third-party system will not be able to use this request unless it is signed on to U.P.M.A.C.S.

DATABASE OBJECTS

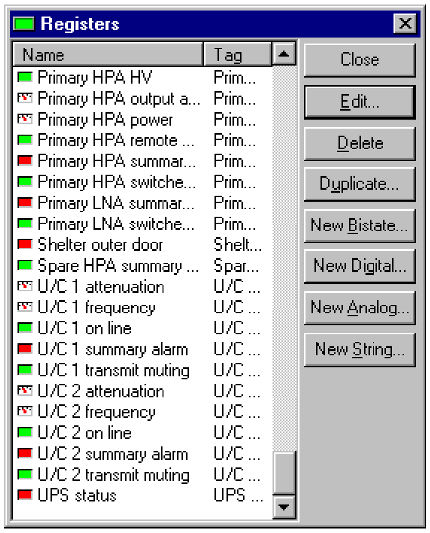
Viewing and Editing Objects

The Object Windows

The Development System provides a window for every type of object. The window contains a list of all objects of that type in the station, as well as a number of buttons to add, delete, and edit objects.



Name	Tag
D/C Port	D/C P...
LNA Port	LNA P...
Modem Port	Mode...
SSPA Port	SSPA ...
SSPA Switch Port	SSPA ...
U/C Port	U/C P...
VHF Radio Port	VHF ...
VHF Radio Switch Port	VHF ...



Name	Tag
🟩 Primary HPA HV	Prim...
📈 Primary HPA output a...	Prim...
📈 Primary HPA power	Prim...
🟩 Primary HPA remote ...	Prim...
🟥 Primary HPA summar...	Prim...
🟩 Primary HPA switche...	Prim...
🟥 Primary LNA summar...	Prim...
🟩 Primary LNA switche...	Prim...
🟥 Shelter outer door	Shelt...
🟥 Spare HPA summary ...	Spar...
📈 U/C 1 attenuation	U/C ...
📈 U/C 1 frequency	U/C ...
🟩 U/C 1 on line	U/C ...
🟥 U/C 1 summary alarm	U/C ...
🟩 U/C 1 transmit muting	U/C ...
📈 U/C 2 attenuation	U/C ...
📈 U/C 2 frequency	U/C ...
🟩 U/C 2 on line	U/C ...
🟥 U/C 2 summary alarm	U/C ...
🟩 U/C 2 transmit muting	U/C ...
🟥 UPS status	UPS ...

To show or hide the window, use the appropriate selection from the "View" menu. There will be a checkmark next to all the windows that are visible.

You can change the size of the window in the usual way, i.e. by grabbing an edge with the mouse and dragging it.

The lists show both name and tag of the object. You can resize the two columns by dragging the edges of the column headers with the mouse. Click on the "Name" header to sort the objects by name, click on the "Tag" header to sort them by tag.

The register window shows little icons next to the register to tell you what type it is:

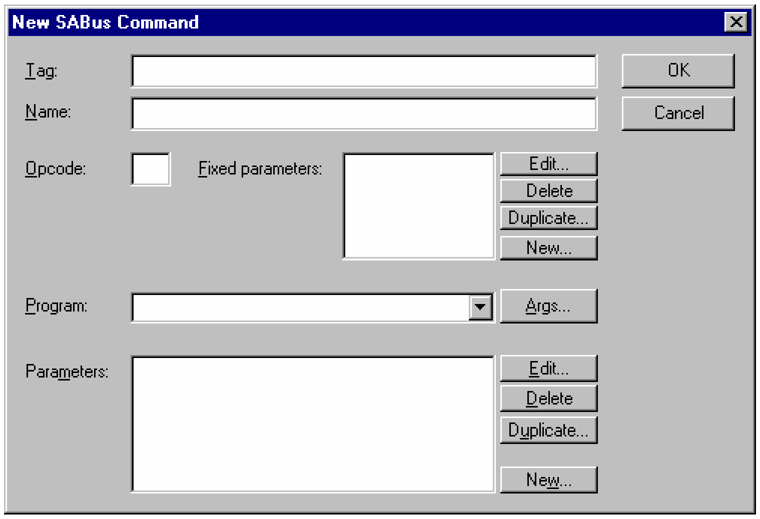
🟩	Bistate Register (status)
🟥	Bistate Register (alarm)
▣	Digital Register
📈	Analog Register
A	String Register

Creating New Objects

To create a new object, either select the appropriate item from the "New" menu, or press the "New…" button in the window for the object type. The New Object dialog will appear. Enter properties for the new object, and press OK.

The register dialog has four buttons for creating registers, one for each type of register. The SABus request dialog has two buttons for creating requests, one for queries and one for commands. The New menu, however, has only one selection for registers and one for requests. If you select "Register…" or "SABus Request…" from the "New" menu, you will be asked to choose the type of register or request to create.

This is what the New SABus Command dialog looks like:



The "Tag" and "Name" fields appear in all of the dialogs.

The tag is a unique identifier for the object, and no two objects of the same type can have the same tag. You cannot press the OK button until you have entered a tag for the object. The tag is used by SCL Programs, and by other objects, to access the new object. You can never change the tag of an object.

The name of the object is used to identify the object to the user. Even though it is possible to have two objects with the same name, you should assign a unique name for each object to avoid confusion. The name of an object can be changed at a later time, if need be.

Usually, there is no reason for the tag and name to be different. If you want the name of the object to be the same as the tag, just leave the name field blank. The Development System will use the tag as a name if you don't specify a name.

Another way to create an object is to duplicate an existing object. Select the object you wish to duplicate, and press the "Duplicate…" button. This will also pop up the New Object dialog, but all the fields will already be filled in with values from the original object. Since all database objects need to have unique tags, you must change at least the tag of the new object before pressing OK.

If you create a screen or SCL program, you specify the properties in the New Object dialog. Once you press OK, a window will appear that lets you place graphic objects on the screen or edit the program code.

Editing Objects

To change an object after you have created it, select it in the object window, and press the "Edit…" button. You can also just double-click on the object. The Edit Object dialog will appear. The Edit Object dialog looks exactly like the New Object dialog, except that you cannot change the tag field.

If you edit a screen or an SCL program, no dialog will appear. The Edit button will simply show the window used to place graphical objects or edit the program code. To change the properties of a screen or a program, select it and press the "Properties…" button. You can also select "Properties…" from the "Edit" menu when viewing the screen or program.

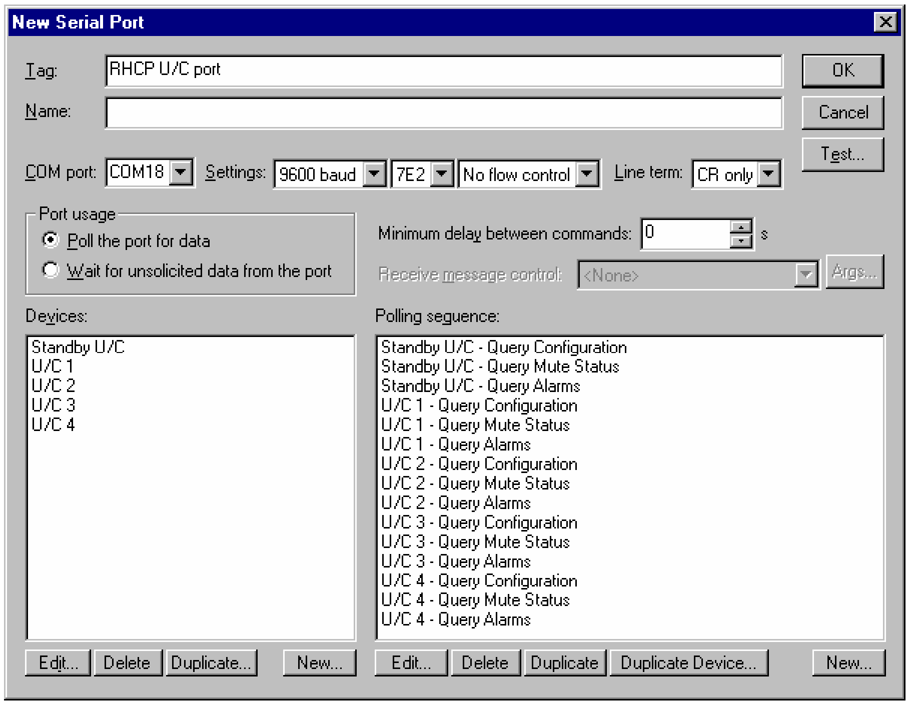
Deleting Objects

To remove an object from the station, select it and press the "Delete" button. You cannot delete objects that are used by other objects; you must delete the other object first. Be careful, deleting an object can never be undone!

Serial Ports

Serial ports provide information about communications settings of a port, and about what equipment is connected to which port. They also contain the polling sequence to be used on that port.

The New Serial Port Dialog



Tag: RHCP U/C port
Name:
COM port: COM18 Settings: 9600 baud 7E2 No flow control Line term: CR only

Port usage
[✓ Poll the port for data]
Wait for unsolicited data from the port

Minimum delay between commands: 0 s
Receive message control: <None>

Devices:
Standby U/C
U/C 1
U/C 2
U/C 3
U/C 4

Polling sequence:
Standby U/C - Query Configuration
Standby U/C - Query Mute Status
Standby U/C - Query Alarms
U/C 1 - Query Configuration
U/C 1 - Query Mute Status
U/C 1 - Query Alarms
U/C 2 - Query Configuration
U/C 2 - Query Mute Status
U/C 2 - Query Alarms
U/C 3 - Query Configuration
U/C 3 - Query Mute Status
U/C 3 - Query Alarms
U/C 4 - Query Configuration
U/C 4 - Query Mute Status
U/C 4 - Query Alarms

- Tag:
Enter the tag by which the port is identified. Each port must have a unique tag.
- Name:
Enter the name of the serial port. Leave this field blank if you want to use the tag as name.
- COM Port:
Select the hardware serial port here. You can also enter the name of a port that does not appear in the list directly. You must enter the complete name of the port, including the "COM." Some serial ports may not be called "COM1," "COM2," etc. If you have a hardware port with a different name, e.g. "PORT-A1," just enter that name.
- Settings:
Select the baud rate here, character format, and flow control to use for the port.

The character format consists of the number of data bits, the parity, and the number of stop bits. 8N1 stands for **8** data bits, **N**o parity, **1** stop bit. 7E2 stands for **7** data bits, **E**ven parity, **2** stop bits, etc.

RTS/DTS and DTR/DSR are hardware flow control; XON/XOFF is software flow control.

- Line term:

Specify the line termination for the equipment connected to the port. The line termination is not used with device commands. It is used only if the user pops up a terminal for the port, or if you use the `SENDSTR` SCL Command. If the equipment attached to the port has no line termination, this selection is without consequence.

- Port usage:

Select "Poll the port for data" to use a polling sequence for equipment that responds to commands. Select "Wait for unsolicited data from the port" if the equipment will send data out of its own accord.

Please note that so far, only legacy device drivers support waiting for unsolicited data from a port.

- Delay between commands:

Some poorly designed equipment will loose data or lock up if a second command is sent immediately after it responded to the first. If your equipment needs a delay between its response and the next command, enter that delay here, in seconds. You can enter fractions of a second.

- Receive message control:

This field is used for legacy devices only. See *Serial Ports* in *Appendix E: Legacy Objects* on page 198 for details.

- Devices:

Shows a list of all the devices attached to the port. Press the "Edit…", "Delete", "Duplicate…", and "New…" buttons to add, delete, or edit devices.

Any devices that have an initialization sequence will be initialized in the order shown here, where the topmost device is initialized first. You can change the order of the devices by grabbing them with the mouse and dragging them to a new position.

See *Devices* on page 15 for a description of the New Device dialog.

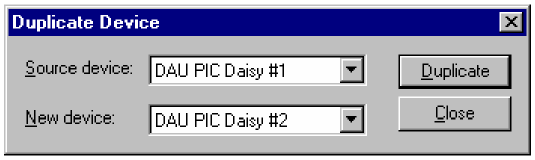
- Polling sequence:

This field is only avaliable if the port is configured to poll equipment. It shows the sequence of commands used to poll the equipment. Press the "New…" button to add a new poll. Press the "Delete" button to delete the selected poll. Press the "Edit…" button to edit the poll's command and routing value. You can use the "Duplicate" button to duplicate a single poll.

See *The Polling Sequence* on page 18 for a description of the New Poll dialog.

The polls will be sent in the order shown. You can change the order of the polls by grabbing them with the mouse and dragging them to a new position.

You can use the "Duplicate Device…" button to duplicate all polls of a single device. When you press this button, you will be prompted for a source device and a new device:


Duplicate Device
Source device: DAU PIC Daisy #1
New device: DAU PIC Daisy #2

The old and new devices must use the same device driver.

- The "Test…" button:

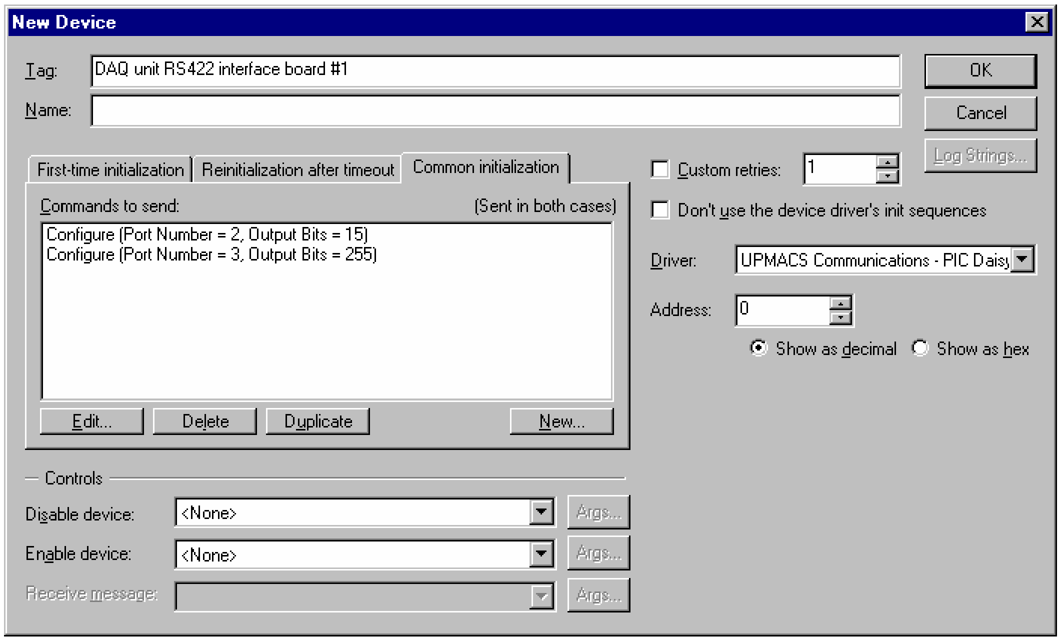
Press this button to test the port. See *Testing Serial Ports* on page 142 for a description of the Test Serial Port dialog.

Devices

Devices provide information about a piece of equipment connected to a serial port. The device specifies which driver to use, and how to initialize the equipment.

You can also specify automatic controls to be executed when the device is enabled or disabled. Disabling a device from the Devices dialog in the Operate System or from an SCL program automatically masks all registers associated with the device, but it is sometimes necessary to perform additional maintenance or to mask additional registers. The automatic controls allow you to do this.

The New Device Dialog



Tag: DAQ unit RS422 interface board #1
Name:
Commands to send: (Sent in both cases)
Configure (Port Number = 2, Output Bits = 15)
Configure (Port Number = 3, Output Bits = 255)
☐ Custom retries: 1
☐ Don't use the device driver's init sequences
Driver: UPMACS Communications - PIC Daisy
Address: 0
[✓ Show as decimal] Show as hex
Disable device: <None>
Enable device: <None>

- Tag:
Enter the tag by which the device is identified. Each device in a port must have a unique tag.
- Name:
Enter the name of the device. Leave this field blank if you want to use the tag as name.
- First-time initialization / Reinitialization after timeout / Common initialization:
Select which initialization commands should be displayed in the initialization sequence list. These tabs only affect which commands are displayed, not which type of initialization the device has. All devices always have all three types of initialization sequence (though some of them may be empty).

See *The Device Initialization Sequences* below for details.

- Commands to send:
Shows a list of all the commands in the initialization sequence in the order in which they will be sent. Use the tabs to select which initialization sequence is shown in the list.

Use the buttons to edit, delete, duplicate, and add new commands. You can grab the commands with the mouse and drag them to a new position in the list to change their order.

See *The Device Initialization Sequences* below for details.

- Custom retries:

If your device needs a different number of retries than the default specified in the driver, check the appropriate box, and enter the new number of retries.

The retry value is the number of times a command will be resent if there is a timeout. If you specify 1 in this field, commands will not be resent if there is a timeout. If you specify 3, U.P.M.A.C.S. will try to send each command up to three times before logging a timeout.

- Don’t use the device driver’s init sequence:

Check this box to skip the initialization commands defined in the device driver. If you check this box, only the initialization commands defined in this dialog will be sent.

- Driver:

Select the device driver that the device uses. Specify values for all the driver’s parameters immediately below. In the sample dialog, the device uses a driver that has one digital parameter called “Address”.

If you do not specify a value for a particular template parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the driver has any parameters of type digital that don’t have value names, you can select the way you want to enter the parameter values here. Select “Show as decimal” to enter the values in decimal, select “Show as hex” to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the driver has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Controls:

Select the automatic controls for enabling and disabling the device here, as well as the receive message control for ports that wait for unsolicited data. The program arguments are shown in parentheses after the names of the programs, but you only select the programs from the lists, not the arguments. To change the arguments, use the “Args…” buttons. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

Disable device:
Select the SCL program to be executed when the device is disabled. This control is also executed on startup if the device is initially disabled.

Enable device:
Select the SCL program to be executed when the device is enabled. This control is *not* executed on startup if the device is initially enabled.

The Device Initialization Sequences

The initialization sequence of a serial device consists of a number of command that are sent to initialize the equipment. There are three types of commands in the initialization sequence:

- First-time initialization commands that are sent when the port is first opened
- Reinitialization commands that are sent when the device needs to be reinitialized after a timeout or error
- Common initialization commands that are sent in both cases

Normally, the initialization sequence is defined in the device driver, but you can specify initialization commands to be sent in addition to those specified in the driver. The device also has the option to ignore the initialization sequence defined in the driver.

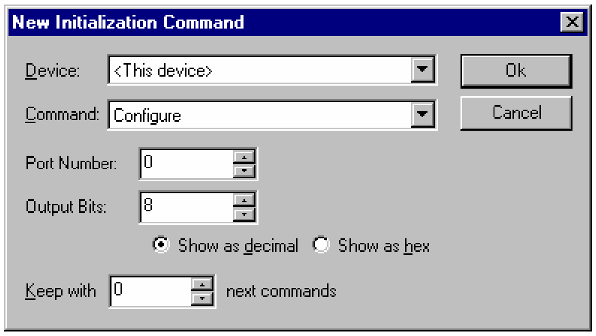
When the port is first opened, the initialization commands are sent in the following order:

1. First-time initialization commands specified in the device driver, in the order in which they appear in the driver's initialization sequence
2. Common initialization commands specified in the device driver, in the order in which they appear in the driver's initialization sequence
3. First-time initialization commands specified in the device, in the order in which they appear in the list
4. Common initialization commands specified in the device, in the order in which they appear in the list

When a timeout occurs, or when an error occurs that requires reinitialization, the initialization commands are sent in the following order:

1. Reinitialization commands specified in the device driver, in the order in which they appear in the driver's initialization sequence
2. Common initialization commands specified in the device driver, in the order in which they appear in the driver's initialization sequence
3. Reinitialization commands specified in the device, in the order in which they appear in the list
4. Common initialization commands specified in the device, in the order in which they appear in the list

The New Initialization Command Dialog



Device: <This device>
Command: Configure
Port Number: 0
Output Bits: 8
Show as decimal
Keep with 0 next commands

- Device:
Select the device to send the command to. Select <this device> to send the command to the device you are currently editing.
- Command:
Select the command to send. Specify values for all the command parameters immediately below. In the sample dialog, the command has two digital parameter called "Port Number" and "Output Bits".

If you do not specify a value for a particular command parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the command has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the command has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Keep with next:

Sometimes, it is necessary to treat a block of several commands as if it was a single command. If you specify a number here, this poll and as many polls that follow it as you specify, will be treated as an indivisible block of commands. This means that either all of the commands are sent, or none. If any of the devices of any of those commands are disabled, or need reinitializing, the whole block will be skipped, rather than just that one command.

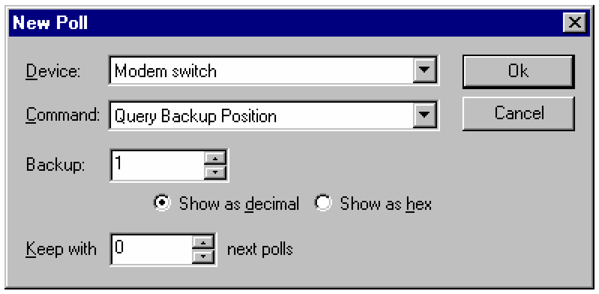
Usually, this field is used for commands to routing devices that will not work properly if the commands that they are supposed to route are not actually sent. For routing commands to routers enter the number of polls routed here.

The Polling Sequence

The polling sequence of a serial port consists of a number of commands that are sent to the equipment to retrieve data from it. The commands (polls) in the polling sequence are sent one after the other, until all commands have been sent. The polling then starts over with the first command in the sequence.

Usually, the polling sequence can be interrupted at any time if an SCL control needs to send a command to the equipment. You can, however, group polls into indivisible blocks that will never be interrupted. If a control needs to access the serial port, it must wait until the entire block of polls has been sent. If you group polls into a block, either all of the polls are sent, or none. If any of the devices of any of those polls are disabled, or need reinitializing, the whole block will be skipped, rather than just that one poll. This is useful for commands to routing equipment that will not work correctly if the polls they are supposed to route are not actually sent.

The New Poll Dialog



Device: Modem switch
Command: Query Backup Position
Backup: 1
[✓ Show as decimal] Show as hex
Keep with 0 next polls

- Device:

Select the device to send the command to.

- Command:

Select the command to send. Specify values for all the command parameters immediately below. In the sample dialog, the command has one digital parameter called "Backup".

If you do not specify a value for a particular command parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the command has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the command has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Keep with next:

If you specify a number here, this poll and as many polls that follow it as you specify, will be treated as an indivisible block of polls.

Usually, this field is used for commands to routing devices that will not work properly if the commands that they are supposed to route are not actually sent. For routing commands to routers enter the number of polls routed here.

Registers

Registers are used to hold information in U.P.M.A.C.S. All data that you wish to display, or that you wish to store for other purposes, will be stored in registers.

There are four types of registers:

- bistate registers
- digital registers
- analog registers
- string registers

Common Characteristics

All of the register types share certain characteristics. This section describes the things that are common to all types of registers.

- The ON/OFF state:

Each register has an ON/OFF state. How that state is determined depends on the register type. The ON/OFF state is used in four ways:

Display:
All types of indicators change the way they look according to the ON/OFF state of the register.

Logging:
The ON/OFF state can be logged. U.P.M.A.C.S. may log default log strings, or you can specify custom log strings.

Automatic controls:
U.P.M.A.C.S. can execute an SCL program every time the register changes state. This is termed an automatic control, as opposed to a manual control, which is triggered by the user by pressing a control button on a screen.

Alarms:
The ON state of a register can trigger an alarm.

- Alarm level:

Each register has an alarm level that determines the way the ON/OFF state is handled. There are four alarm levels:

No alarm level assigned:
The ON/OFF state does not trigger an alarm, and is not logged by default. Use this level for registers where the ON/OFF state is not used, or where it is only used internally. If you do not assign a level to a register, you do not have to provide a way to determine the ON/OFF state.

Status:
The ON/OFF state does not trigger an alarm, but is logged by default. Use this level for registers whose ON/OFF state represents a mode, setting, or other status. If you assign the Status level to a register, you must provide a way to determine the ON/OFF state.

Alarm:
The ON state triggers an alarm, and is logged by default. Use this level for registers whose ON state represents an alarm. If you assign the Alarm level to a register, you must provide a way to determine the ON/OFF state.

Latching alarm:
The ON state triggers an alarm, and is logged by default. The alarm is also latched, which means that it will not clear until it is acknowledged. Use this level for registers whose ON state represents an alarm, and if you want the alarm to remain active even if the alarm condition clears. If you assign the Latching alarm level to a register, you must provide a way to determine the ON/OFF state.

- Error state:

If a register has no value, it is in the error state. A registers will be in the error state until it has been updated. For registers with sources that use serial data buffers, the register will be in the error state until the buffer has been filled by a poll. A register will also be in the error state if the data is inaccessible in some way, e.g. because of a serial communications failure, or because a register it uses to get data from contains no data.

If a register is in the error state, it has no value or ON/OFF state. The value and state are not accessible to other registers or SCL programs, and they will not be displayed. You can define special ways of representing the error state for indicators that would show the register’s value.

- Masked state:

A register can be masked to disable it. A masked register is said to be in the masked state, and it has no value, no ON/OFF state, and no error state. The value and state are not accessible to other registers or SCL programs, and they will not be displayed. You can define special ways of representing the masked state for indicators that would show the register’s value.

Registers can be masked in one of three ways:

Manual masking:
The user can mask a register from within the U.P.M.A.C.S. Operate System. A user will usually do this to disable individual “nuisance” alarms that are not functioning properly or are otherwise irrelevant.

Auto masking:
Some register sources can automatically mask the register when the object that the value is taken from is disabled. Register sources that use data from a serial device, for example, will auto mask their register if an operator disables the device.

Internal masking:
SCL programs can mask and unmask registers internally using the `INTMASK` and `INTUNMASK` commands described in the SCL Programming Language Help. You will usually want to internally mask registers that relate to equipment or features that have been disabled, or to data that is not applicable in a certain situation. You might, for example, mask the tracking signal level reading on an antenna controller unit if satellite tracking has been disabled.

> Note: you can also simulate manual masking from within an SCL program using the MASK and UNMASK commands. This is not the same as masking internally: it is treated the same as manual masking by the user. If you mask a register manually, the user can unmask it. If you mask it internally, he cannot.

Each of these three ways of masking is maintained separately. Masking and unmasking the register manually does not affect the auto or internal mask state; setting and clearing the internal mask state from an SCL program does not affect the manual or auto mask states, and so on. A register is in its masked state if it has been masked either manually or automatically or internally.

- Logging:

You can configure a register to write a message to the log every time its value changes. By default, values are not logged. To log the values of a register, you must provide custom log strings. It is usually desirable to log the values of all registers that contain settings of equipment, but not those that contain readings like temperature, etc. Since the values of such readings change constantly, logging them would flood the log file with thousands of messages about slight changes of the same value.

The ON/OFF state of a register can also be logged. Registers with an alarm level provide default log string as follows:

Status:	*Register name* on / *Register name* off
Alarm:	*Register name* / *Register name* clear

where *Register name* is the name of the register. If an alarm is acknowledged, the following is logged:

Register name acknowledged

Manual masking of a register can also be logged. The default log strings are as follows:

Register name masked / *Register name* unmasked

You can provide your own log strings, or you can prevent any of the default log string from being logged.

- Automatic controls:

SCL programs can be attached to the register that are executed automatically whenever the register's value changes. Automatic controls can also be executed when the register goes into the ON or OFF state, when it is acknowledged, or when it is masked or unmasked manually.

- User definable registers:

Registers can be designated as being user definable. This simply means that their name and possibly some other settings can be configured by the user from within the U.P.M.A.C.S. Operate System. This feature is usually used to provide spares for the purpose of future expansion, or to provide custom labeling of transmit and receive chains.

To provide spares, define registers for them, and call them "Spare 1", "Spare 2", etc., or something similar, and make the registers user definable. The operator can then change the name from within the U.P.M.A.C.S. Operate System to reflect the meaning of the register once it will be used.

- Hiding registers:

You can hide a register's indicators from the screen. Unlike masking, the hidden state does not disable the functionality of the register; it merely hides it from the user. A hidden register still has an ON/OFF state, an error state, and a masked state, and it still triggers alarms and writes messages to the log. Registers can only be hidden and unhidden from within SCL programs.

It is usually preferable to mask a register internally rather than hide it, as masking will disable the register's functionality as well. Hiding registers, however, can be useful in some cases:

Displaying alternate forms of the same data:
You might want to allow the user to display output power in Watts or dBm. In this situation, you can create two registers, one containing the power in Watts, and one in dBm. You can then hide one of the two registers and display only the other.

Displaying different sets of information in the same area of a screen:
You might want to display different sets of information in the same area of a screen. For example: On an antenna control system screen, you might provide a set of summary alarms, like a system alarm, a motion alarm, and a tracking alarm. Underneath, you could have an alarm details box and a set of selection buttons, where the user can choose to view the individual alarms that belong for each summary alarm.

In this case, you would provide a set of indicators for each set of alarms, all occupying the same space on the screen. If the user presses the "View motion alarms" button, you can then unhide the motion alarms, and hide all other alarms. This can save space and reduce screen clutter.

Providing for different station configurations:
You might want to use the same station database for several almost identical earth stations. You can provide for subtle differences in the stations by hiding some registers on some stations. For example, one of the station might have an extra air conditioning unit. In that case, you could provide support for hiding the unit on the other systems.

If you decide to hide equipment, please keep in mind that hiding the equipment from the screen does not disable it. If you are hiding equipment or alarms that are not present, make sure to disable the appropriate devices and internally mask any extra registers, or timeout and error messages will be logged in the log file.

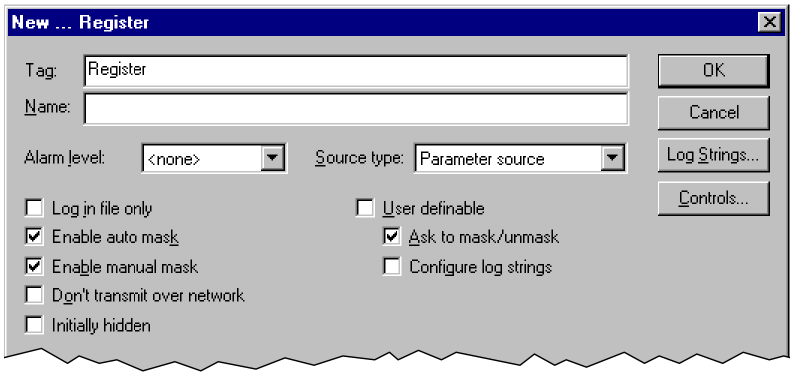
It is usually undesirable to hide equipment that may be added in the future. If you need room for future expansion, the equipment that is not present should be masked, not hidden.

- Sources:

Each register has a data source. The source specifies where the value of the register is to be taken from, and how. See *Sources* on page 41 for a description of the different register sources.

The New Register Dialogs

There are four different new register dialogs, one for each type of register. All four New Register dialogs share the following fields, however:



Tag: Register
Name:
Alarm level: <none>
Source type: Parameter source
[ ] Log in file only
[✓] Enable auto mask
[✓] Enable manual mask
[ ] Don't transmit over network
[ ] Initially hidden
[ ] User definable
[✓] Ask to mask/unmask
[ ] Configure log strings

The list below describes only those fields that appear in all four types of New Register dialogs. For the remaining fields, see the section on the appropriate register type.

- Tag:

Enter the tag by which the register is identified. Each register must have a unique tag. Two registers cannot have the same tag, even if they are of different types.

- Name:

Enter the name of the register. Leave this field blank if you want to use the tag as name.

- Alarm level:

Select the alarm level for the register. If you select a level other than <none>, you will have to specify a way to determine the ON/OFF state. Only bistate registers have a built-in mechanism. See the specific New Register dialogs for details.

Only select “Status” if your register has an ON/OFF state. If you only use values, not the ON/OFF state, select <none>.

- Log in file only:

Check this box if you want the log messages to be written to the log file only, and not to the log window.

- Enable auto mask:

Check this box to enable the source to automatically mask the register.

- Enable manual mask:

Check this box to enable the operator to manually mask the register from within the Operate System. Clearing this check box does not prevent SCL programs from manually masking the register, so you can still provide controls to manually mask and unmask the register, even if this check box is cleared.

- Don't transmit over network:

Check this box to prevent U.P.M.A.C.S. from transmitting this register across the network when someone connects to the station. This means that the register will only appear on the local computer, not on any remote computers. Use this flag for registers that are part of a network map, for example.

- Initially hidden:

Check this check box if you want the register to be hidden when the station is loaded. Registers can only be unhidden from within an SCL program. Please remember that hiding a register does not disable it, the register is still updated and logged, automatic controls are executed, and alarms are triggered.

- Source type:

Select the type of the register's data source. See Sources on page 41 for a description of the individual source types. The settings for the source are shown at the bottom of the dialog, under the heading "Source".

- User definable:

Check this box to allow the user to change the name of the register and other settings from within the U.P.M.A.C.S. Operate System.

- Ask to mask unmask:

This check box is only available for user definable registers. If you check this box, the register will be manually masked originally, and must be unmasked by the operator. The operator will be asked if he wants to unmask the register when he configures it, and if he wants to mask it if he reverts to the unconfigured state.

- Configure log strings:

This check box is only available for user definable registers. Check this box to allow an operator to specify his own log strings from within the Operate System.

- The "Log Strings…" button:

Press this button to specify custom log stings, or to prevent default log strings from being logged.

See *Register Log Strings* below for a description of the Edit Register Log Strings dialog.

- The "Controls…" button:

Press this button to specify automatic controls for the different events.

See *Automatic Controls* on page 26 for a description of the Edit Automatic Controls dialog.

Register Log Strings

You can specify custom log strings for different register states and values.

The percentage sign ("%") character has a special meaning in register log strings. It will be replaced by the name of the register. If the register is called "HPA 2" then

```
% in remote mode
```

will be logged as:

```
HPA 2 in remote mode
```

To write a percentage sign ("%") to the log, put two percentage signs ("%%") in the log string:

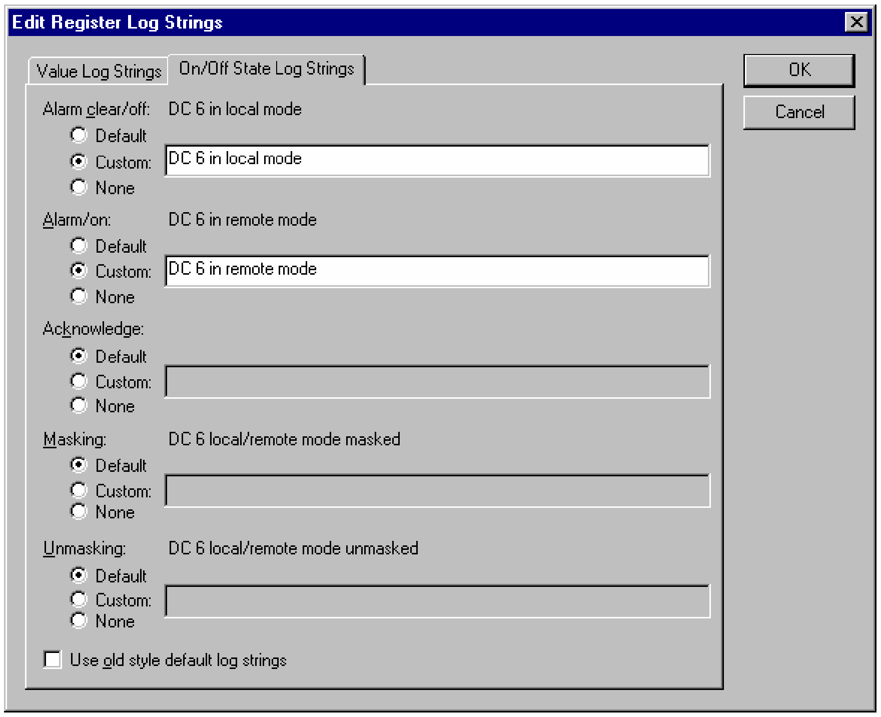
```
100%% premium orange juice
```

will be logged as:

```
100% premium orange juice
```

Press the "Log Strings…" button in the New Register dialog to pop up the Edit Register Log Strings dialog.

The Edit Register Log Strings Dialog



Alarm clear/off: DC 6 in local mode
Default [✓ Custom:] None — DC 6 in local mode

Alarm/on: DC 6 in remote mode
Default [✓ Custom:] None — DC 6 in remote mode

Acknowledge:
[✓ Default] Custom: None

Masking: DC 6 local/remote mode masked
[✓ Default] Custom: None

Unmasking: DC 6 local/remote mode unmasked
[✓ Default] Custom: None

☐ Use old style default log strings

The Edit Register Log Strings dialog has two sections, one for editing the strings logged when the value changes, and one for editing the strings logged when the ON/OFF state or masked state changes. Use the “ Value Log Strings” / “On/Off State Log Strings” tabs to select which settings to display These tabs only affect which settings are displayed, not which log strings the register has. All registers (except bistate registers) have both types of log strings.

Bistate registers do not have a value and hence no value log strings. These tabs are not present if you are editing the log strings of a bistate register.

The fields below only describe the settings for the ON/OFF state log strings. See *Digital Value Log Strings*, *Analog Value Log Strings*, and *String Value Log Strings* on pages 29, 34, and 39, respectively, for details on editing value log strings.

- Specifying the log strings

For each type of log string, you can select how it should be logged. Select “Default” to log the default log string described in Registers under Logging. Select “Custom:” and enter a custom message in the text box to log a custom log string. Select “None” to log nothing. If there is no default log string, selecting “None” is the same as selecting “Default”.

A sample of the log string, with %-symbols replaced with the register name, is shown for each state.

- Fault clear/off:

Specify the log string for the OFF state of the register.

- Fault/on:

Specify the log string for the ON state of the register.

- Acknowledge:

Specify the string to log when an alarm is acknowledged. This string is not used for registers with alarm levels other than Alarm and Latching alarm.

- Masking:

Specify the string to log when the register is manually masked.

- Unmasking:

Specify the string to log when the register is manually unmasked.

- Use old style default log strings:

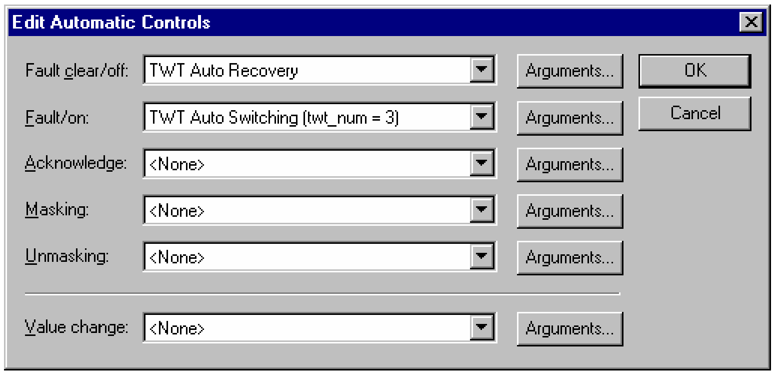
Older versions of U.P.M.A.C.S. used to include the word "alarm" in some of the default log strings for alarms and latching alarms. This is no longer the case. Any alarm registers you created with an earlier version of the U.P.M.A.C.S. Development System will have this check box checked to indicate that they will use the old-style logging that includes the word "alarm." You should not check this check box on newly created registers; this feature is only available so that alarms in older databases will still be logged as before. You should include the word "alarm" in the title of the register when you create alarms.

Automatic Controls

You can specify automatic controls for different register states and values. The automatic control for a particular state will be executed when the register goes into that state.

Press the "Controls…" button in the New Register dialog to pop up the Edit Automatic Controls dialog.

The Edit Automatic Controls Dialog


Edit Automatic Controls

Fault clear/off: TWT Auto Recovery

Fault/on: TWT Auto Switching (twt_num = 3)

Acknowledge: <None>

Masking: <None>

Unmasking: <None>

Value change: <None>

The Edit Automatic Controls dialogs for the different types of register have different fields for specifying automatic controls to be executed when the value of the register changes. See *Bistate Value Controls*, *Digital Value Controls*, *Analog Value Controls*, and *String Value Controls* on pages 28, 31, 37, and 41, respectively, for details.

- Specifying the control programs and arguments

Select the SCL programs you want to be executed when the register goes into the different states. The program arguments are shown in parentheses after the names of the programs, but you only select the programs from the lists, not the arguments. To change the arguments, use the "Args…" buttons. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

- Fault clear/off:

Select the SCL program to be executed when the register goes into the OFF state.

- Fault/on:

Select the SCL program to be executed when the register goes into the ON state.

- Acknowledge:

Select the SCL program to be executed when the register is acknowledged. This control is not used for registers with alarm levels other than Alarm and Latching alarm.

- Masking:

Select the SCL program to be executed when the register is manually masked.

- Unmasking:

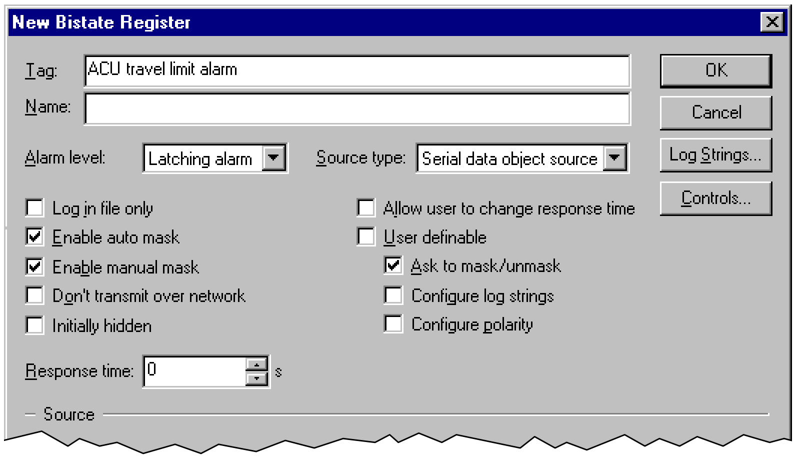
Select the SCL program to be executed when the register is manually unmasked.

Bistate Registers

Bistate registers do not have a value separate from the ON/OFF state. All the information contained in a bistate register is expressed by its ON/OFF state. Use a bistate register for data that can be expressed as two mutually exclusive states, like alarm/alarm clear, local/remote mode, or HV on/off.

Bistate registers have a response time. The response time is the minimum amount of time a state change has to last in order to be reflected by the register. If the register reflects an alarm, for example, and the response time is 1s, then the alarm will only be registered if it persists for 1s. If the alarm is cleared within 1s, the register will not go into its ON state. This is useful to eliminate erroneous alarms due to relay chatter or noise.

The New Bistate Register Dialog


New Bistate Register

Tag: ACU travel limit alarm

Name:

Alarm level: Latching alarm

Source type: Serial data object source

☐ Log in file only
☑ Enable auto mask
☑ Enable manual mask
☐ Don't transmit over network
☐ Initially hidden
☐ Allow user to change response time
☐ User definable
☑ Ask to mask/unmask
☐ Configure log strings
☐ Configure polarity

Response time: 0 s

The list below describes only those fields that are specific to the New Bistate Register dialog. For the remaining fields, see *The New Register Dialogs* on page 23.

- Configure polarity:

This check box is only available for user definable registers. Check this box to allow an operator to invert the ON/OFF states of the register from the Operate System. This is useful for spares on data acquisition units that might be connected to normally open or normally closed alarm contacts in the future.

- Response time:

Enter the response time, in seconds. This is the minimum time a change in the ON/OFF state must last in order to be registered. You can enter fractions of a second. Enter 0 to register state changes immediately.

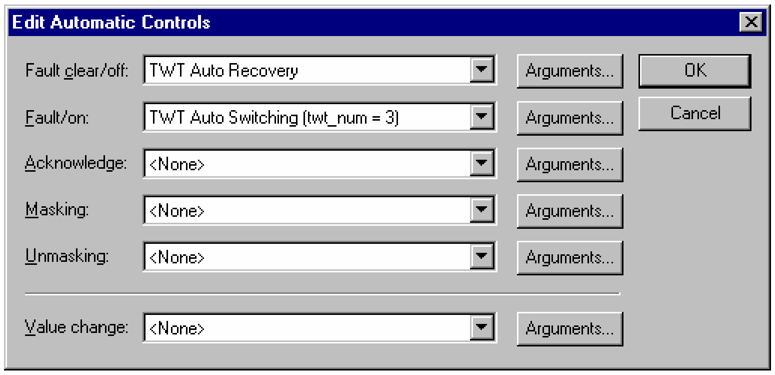
- Allow user to change response time:

Check this box to allow the operator to adjust the response time of the register from within the Operate System.

Bistate Value Controls

You can specify a control to be executed every time the value of a bistate register changes. Since the value of a bistate register is the same as its state, this control will be executed when the register goes into its OFF state and when it goes into its ON state.

The Edit Automatic Controls Dialog for Bistate Registers



Fault clear/off: TWT Auto Recovery
Fault/on: TWT Auto Switching (twt_num = 3)
Acknowledge: <None>
Masking: <None>
Unmasking: <None>
Value change: <None>

The list below only shows the fields for value controls. For the other fields, see *Register Log Strings* on page 24.

- Value change:

Select the SCL program you want to be executed when the value changes. The program arguments are shown in parentheses after the names of the program, but you only select the program from the list, not the arguments. To change the arguments, use the "Arguments…" buttons. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

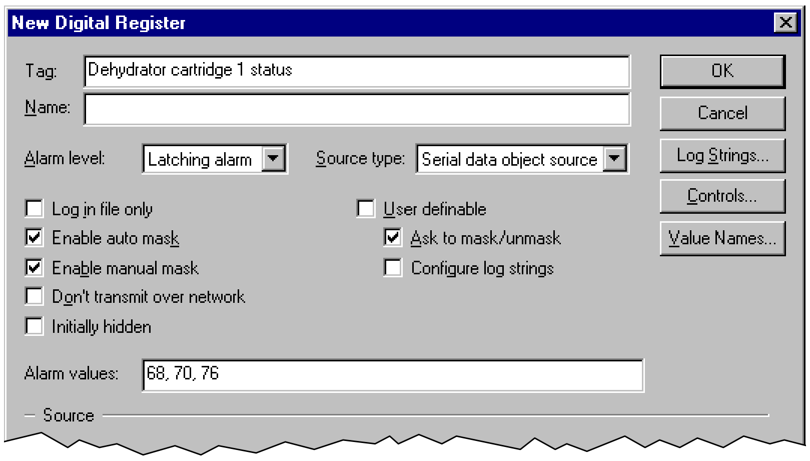
Digital Registers

The value of a digital register is an integer between 0 and 4,294,967,295. Use digital registers for data that can be expressed as three or more discreet states, like modulation type, selected satellite, or local / remote front panel / computer mode. You can also use digital registers to contain a number, as long as the number has no fractions, and will not exceed the range of a digital indicator. This includes channel numbers and counters, for example.

You can specify a name for different values of a digital register. The value names can be used in value log strings, in digital indicators, and in SCL programs. They are also shown in the pop-up list used for selecting the states values of multistate indicators. You should use value names if a digital parameter represents a number of different settings or choices (e.g. local / remote / remote front panel mode), rather than an actual number.

The ON/OFF state of a digital register is determined by its alarm values. The register will be in the ON state if the value is one of the alarm values you specified; otherwise it will be in the OFF state.

The New Digital Register Dialog



Tag: Dehydrator cartridge 1 status
Name:
Alarm level: Latching alarm
Source type: Serial data object source
[ ] Log in file only
[✓] Enable auto mask
[✓] Enable manual mask
[ ] Don't transmit over network
[ ] Initially hidden
[ ] User definable
[✓] Ask to mask/unmask
[ ] Configure log strings
Alarm values: 68, 70, 76

The list below describes only those fields that are specific to the New Digital Register dialog. For the remaining fields, see *The New Register Dialogs* on page 23.

- Alarm values:

Enter all the values of the register that trigger the ON state. Separate the values with commas. You must specify at least one alarm value if you have selected an alarm level other than <none>.

- The “Value Names…” button:

Press this button to assign names to the values of the register.

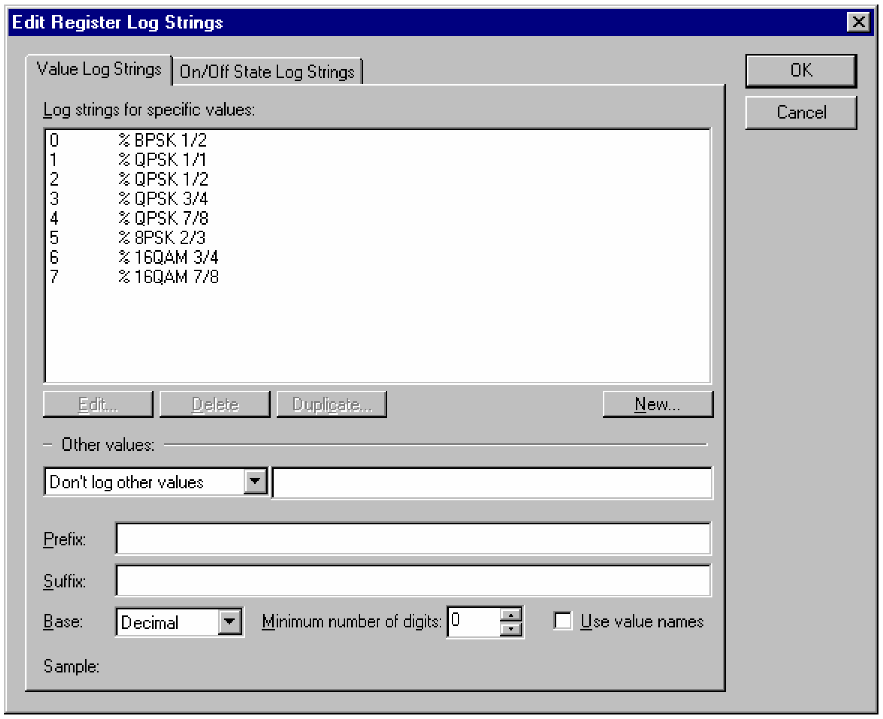
If you do not specify any value names for a register that has a serial data object source, and the object the register gets its value from has itself value names, then the register’s values will automatically have the same names as the data object’s. If you do specify value names in the register, the data object’s value names will not be used.

Digital Value Log Strings

You can specify a log string for different values of a digital register. You can also specify a single log string to be used for every value of the register for which you did not specify a log string.

The log string for every other value actually writes the value, or the value’s name, to the log. A prefix string is prepended to the value, and a suffix string is appended, before it is logged.

The Edit Register Log Strings Dialog for Digital Registers



0	% BPSK 1/2
1	% QPSK 1/1
2	% QPSK 1/2
3	% QPSK 3/4
4	% QPSK 7/8
5	% 8PSK 2/3
6	% 16QAM 3/4
7	% 16QAM 7/8

Other values: Don't log other values

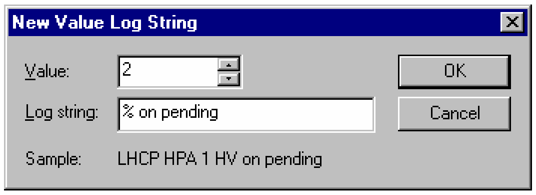
Base: Decimal

Minimum number of digits: 0

The list below only shows the fields for value controls. For the other fields, see *Register Log Strings* on page 24.

- Log strings for specific values:

Shows a list of all the values that will be logged, together with their log strings. Use the buttons to create, delete, duplicate, or edit log strings.



Value: 2

Log string: % on pending

Sample: LHCP HPA 1 HV on pending

The New Value Log String dialog

Enter the value for which you want the string to be logged, and the string you want to log. You can look at the sample to see what the log string will look like with %-symbols replaced by the register name.

- Other Values:

Select which other values to log.

Don't log other values:
Select this option if you want only those values logged that appear in the "Log strings for specific values" list.

Log all other values:
Select this option if you would like to log all values.

Log only on state values:
Select this option if you would like to log only the values you listed under "Alarm values" in the register dialog.

Log only off state values:
Select this option if you would like to log only the values you didn't list under "Alarm values" in the register dialog.

Log the following values:
Select this option if you would like to log only specific values. Enter the values to log, separated by commas, in the edit field on the right.

Log all values except:
Select this option if you would like to log all but a number of specific values. Enter the values you don't want to log, separated by commas, in the edit field on the right.

Note that values listed under "Log strings for specific values" will always use the log string you specified there, and never the one for other values.

- Prefix:

Enter the value prefix here. The prefix is prepended to the value before logging it. %-symbols in the prefix will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Suffix:

Enter the value suffix here. The suffix is appended to the value before logging it. %-symbols in the suffix will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Base:

Select the numerical base that values are to be shown in.

- Minimum digits:

Select the minimum number of digits to show. If the value has less digits than is specified here, it is padded with zero to the required number of digits.

- Use value names:

Check this box to log the names of values rather than the values themselves. Values that have no name will always be logged as numbers.

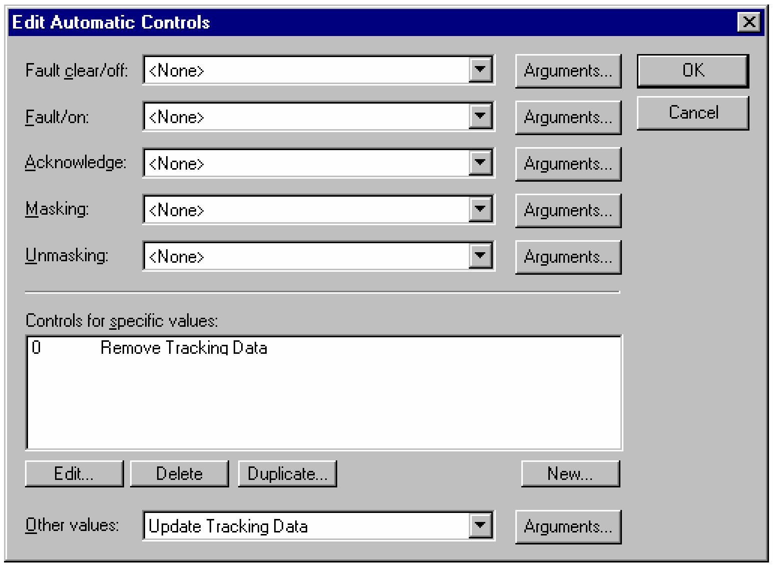
- Sample:

Shows a sample of how values will be logged. The value shown here is the same as that displayed by digital indicators.

Digital Value Controls

You can specify an automatic control to be executed when a digital register assumes certain values. You can also specify a single control to be executed for every value of the register for which you did not specify a different control.

The Edit Automatic Controls Dialog for Digital Registers

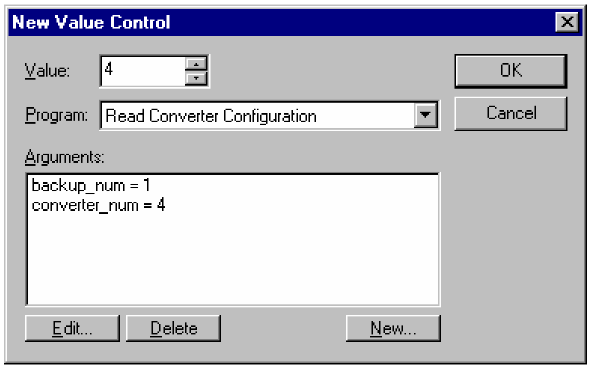


The list below only shows the fields specific to digital registers. For a description of the other fields, see *Automatic Controls* on page 26.

- Controls for specific values:

Shows a list of all the values that have automatic controls defined, together with the programs that will be executed.

Use the buttons to create, delete, duplicate, or edit controls.



The New Value Control dialog

Enter the value for which you want the control to executed, select the program from the list, and specify the arguments. See *Specifying Arguments for SCL Programs* on page 67 for more information.

- Other Values:

Select the SCL program you want to be executed for all values you didn't specify in the "Controls for specific values" list. The program arguments are shown in parentheses after the names of the program, but you only select the program from the list, not the arguments. To change the arguments, use the "Arguments…" buttons. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

Analog Registers

The value of an analog register is a number. The number can contain fractions, and has a virtually unlimited range. Use analog registers for things like frequencies, power levels, meter readings, Eb/N0 values, etc.

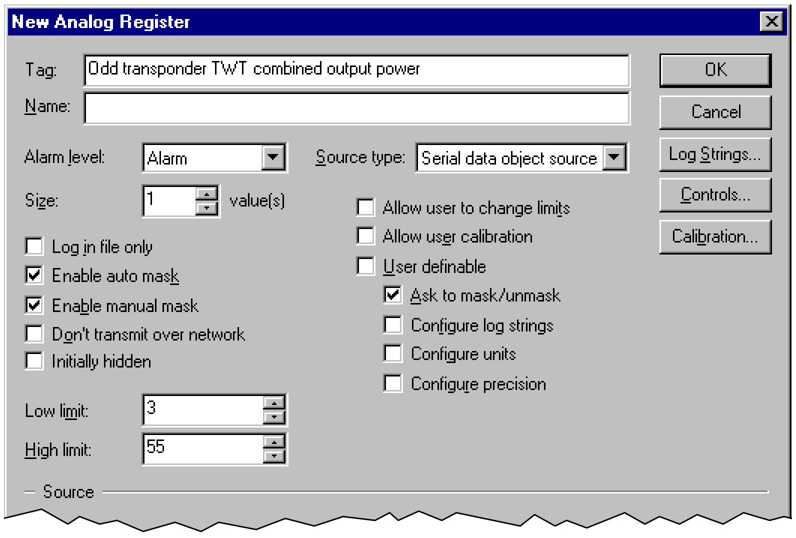
The ON/OFF state of an analog register is determined by a low and high limit. The register is in the OFF state if the value lies within the limits and in the ON state if the value lies outside.

You can configure an analog register to hold more than just one single value. An analog register can have a size of more than one value, which means it holds a whole set of values, numbered from 1 to the size of the register. You can access the values individually, you can display the greatest and least of the values, or you can display all of the values as a bar or line graph. Use analog registers with a size greater than 1 value to hold any data with more than one data point, e.g. a spectrum analyzer's trace.

A special application of analog registers with more than one value is showing the development over time (history) of another analog value. To create a history for a register called "Receive signal strength", for example, create a second register called "Receive signal strength history" (or something similar) with a size of 100 or so values. The exact number of values depends on how many data points you whish to remember. Then, write an SCL program that gets the value of "Receive signal strength" using the `ANAVAL` SCL function and then sets the value of "Receive signal strength history" using the `SETANAVAL` SCL command. Configure that program to run every time you want a new data point added, either at fixed intervals, or at fixed times. See also SCL Programs and SCL Program Scheduling for details on how to do this.

The values of analog registers can be calibrated from within the Operate System.

The New Analog Register Dialog



Tag: Odd transponder TWT combined output power
Name:
Alarm level: Alarm
Source type: Serial data object source
Size: 1 value(s)
- [ ] Log in file only
- [x] Enable auto mask
- [x] Enable manual mask
- [ ] Don't transmit over network
- [ ] Initially hidden
- [ ] Allow user to change limits
- [ ] Allow user calibration
- [ ] User definable
  - [x] Ask to mask/unmask
  - [ ] Configure log strings
  - [ ] Configure units
  - [ ] Configure precision

Low limit: 3
High limit: 55

The list below describes only those fields that are specific to the New Analog Register dialog. For the remaining fields, see *The New Register Dialogs* on page 23.

Size:
Enter the number of values the register has. For a single value, enter 1. Please note that you cannot configure the size of a register that has a serial data object source. Registers with serial data objects sources always have the same size as the data object that they get their value from.

Allow user to change limits:
Check this box if you want to allow the operator to change the high and low limits from within the U.P.M.A.C.S. operate system.

Allow user calibration:
Check this box if you want to allow the user to calibrate the value from within the U.P.M.A.C.S. Operate System. You should usually check this box for values that need calibration.

Configure units:
This check box is only available for user definable registers. Check this box to allow an operator to specify the units (e.g. Watts, MHz, °C) of the data from within the operate system. The units do not influence the functionality of the register, but they are displayed in the suffix of analog indicators. See Analog Indicators for details on displaying the value of analog registers.

Configure precision:
This check box is only available for user definable registers. Check this box to allow an operator to specify the number of digits after the decimal point shown in analog indicators for this register. See Analog Indicators for details on displaying the value of analog registers.

Low limit, high limit:
Enter the low and high limits here. Leave a field blank if you do not want the corresponding limit. You must specify at least one limit if you have selected an alarm level other than <none>.

The "Calibration…" button:
Press this button to add or remove calibration points, and to configure calibration curve smoothing.

See *Analog Calibration Settings* on page 37 for a description of the Calibration Data dialog.

Analog Value Log Strings

You can specify a string to be written to the log file every time the value of an analog register changes. The log string writes the register value to the log. A prefix string is prepended to the value, and a suffix string is appended, before it is logged.

You can specify a minimum change between values that are logged. If you specify a minimum change, the value will not be logged until it has changed by at least the amount specified from the last value logged.

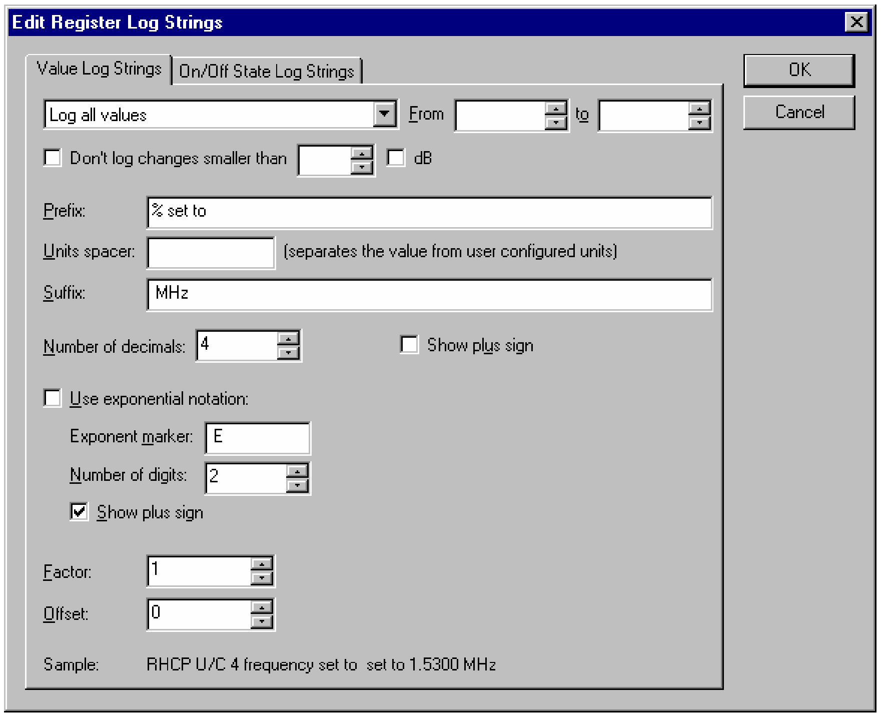
You can modify the register's value before logging it. The number logged is calculated from the value of the register as follows:

logged value = register value · factor + offset

To display the value unaltered, use a factor of 1 and an offset of 0.

Note: The values of analog registers with a size of more than one value cannot be logged.

The Edit Register Log Strings Dialog for Analog Registers


Edit Register Log Strings

Value Log Strings | On/Off State Log Strings

Log all values — From: — to:

☐ Don't log changes smaller than ☐ dB

Prefix: % set to

Units spacer: (separates the value from user configured units)

Suffix: MHz

Number of decimals: 4 ☐ Show plus sign

☐ Use exponential notation:

Exponent marker: E

Number of digits: 2

☑ Show plus sign

Factor: 1

Offset: 0

Sample: RHCP U/C 4 frequency set to set to 1.5300 MHz

The list below only shows the fields for value log strings. For the fields for the ON/OFF-state log strings, see *Register Log Strings* on page 24.

- Selecting which values to log:

Select which values to log at the top right of the dialog.

Don't log any values:
Select this option if you don't want any values to be logged.

Log all values:
Select this option if you would like to log all values.

Log only on state values:
Select this option if you would like to log only values outside the register's low and high limit.

Log only off state values:
Select this option if you would like to log only values within the register's low and high limit.

Log values within or on the following limits:
Select this option if you would like to log only a specific range of values. Enter the limits in the "From" and "to" fields. Values that are exactly on the limit will be logged if you select this option.

Log values outside or on the following limits:
Select this option if you would like to log all values except a specific range of values. Enter the limits in the "From" and "to" fields. Values that are exactly on the limit will be logged if you select this option.

Log values strictly within the following limits:
Select this option if you would like to log only a range of values. Enter the limits in the "From" and "to" fields. Values that are exactly on the limit will not be logged if you select this option.

Log values strictly outside the following limits:
Select this option if you would like to log all values except a certain range of values. Enter the limits in the "From" and "to" fields. Values that are exactly on the limit will not be logged if you select this option.

- Don't log changes smaller than:

Check this box to prevent logging if the value changes by less than a certain amount. A new value will not be logged until it differs by at least the number you enter here from the last value logged. If you check the "dB" box, the change you enter is a relative change in dB, not an absolute one.

- Prefix:

Enter the value prefix here. The prefix is prepended to the value before logging it. %-symbols in the prefix will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Units spacer:

Enter a spacer to place between the value and any units string the user may have configured for the register. This string is only used for user configurable registers that have the "configure units" check box checked. If the user configured a units string for the register, the spacer, followed by the units will be inserted between the value and the suffix.

%-symbols in the separator will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Suffix:

Enter the value suffix here. The suffix is appended to the value before logging it. %-symbols in the suffix will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Number of decimals:

Enter the number of digits to appear after the decimal point. The number will be rounded to the specified number of decimals, or padded with zeros to the right, as necessary.

- Show plus sign:

Check this box if you want a plus sign to be prepended to positive numbers.

- Use exponential notation:

Check this box to use exponential (scientific) notation for the value.

- Exponent marker:

Enter the exponent marker here. The exponent marker is placed between the mantissa and the exponent. Usually, the exponent marker should be " E" or " e".

- Number of digits:

Enter the number of digits you want to show in the exponent.

- Show plus sign:

Check this box if you want a plus sign to be prepended to positive exponents.

- Factor:

Enter the factor with which the register value is to be multiplied before logging it. The factor is applied before the offset.

- Offset:

Enter the offset that is to be added to the register value before logging it. The offset is applied after the factor.

- Sample:

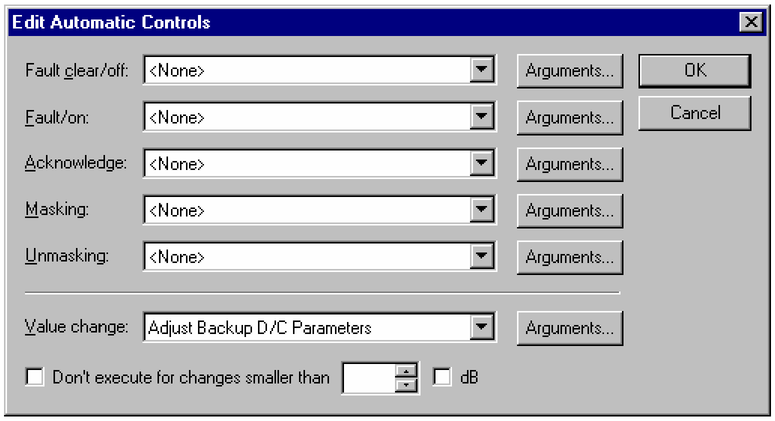
Shows a sample of how values will be logged. The value shown here is the same as that displayed by analog indicators.

Analog Value Controls

You can specify a control to be executed every time the value of an analog register changes. You can specify a minimum change between values. If you specify a minimum change, the control will not be executed until the register’s value has changed by at least the amount specified from the last time the control was executed.

Note: The values of analog registers with a size of more than one value cannot trigger a control.

The Edit Automatic Controls Dialog for Analog Registers


Edit Automatic Controls

Fault clear/off: <None>

Fault/on: <None>

Acknowledge: <None>

Masking: <None>

Unmasking: <None>

Value change: Adjust Backup D/C Parameters

☐ Don't execute for changes smaller than ☐ dB

The list below only shows the fields for value controls. For the other fields, see *Register Log Strings* on page 24.

- Value change:

Select the SCL program you want to be executed when the value changes. The program arguments are shown in parentheses after the names of the program, but you only select the program from the list, not the arguments. To change the arguments, use the “Arguments…” buttons. See Specifying Arguments for SCL Programs for a description of the *Specifying Arguments for SCL Programs* on page 67.

- Don’t execute for changes smaller than:

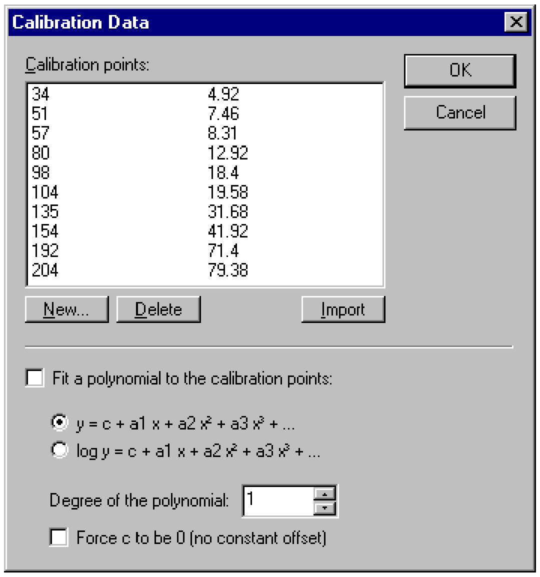
Check this box to prevent execution if the value changes by less than a certain amount. The control will only be executed once the value differs by at least the number you enter here from the value when the control was executed last. If you check the “dB” box, the change you enter is a relative change in dB, not an absolute one.

Analog Calibration Settings

You can add some default calibration points to an analog register from within the Development System. Usually there is no need to do this, since any analog value will have to be calibrated from within the Operate System.

You can specify a polynomial curve that is to be fitted to the calibration points.

The Calibration Data Dialog



Calibration points:

34	4.92
51	7.46
57	8.31
80	12.92
98	18.4
104	19.58
135	31.68
154	41.92
192	71.4
204	79.38

☐ Fit a polynomial to the calibration points:

● $y = c + a1\ x + a2\ x^2 + a3\ x^3 + \ldots$
○ $\log y = c + a1\ x + a2\ x^2 + a3\ x^3 + \ldots$

Degree of the polynomial: 1

☐ Force c to be 0 (no constant offset)

- Calibration points:

Shows all the calibration points currently defined. Raw data is shown on the left, calibrated data on the right. Use the "New…" and "Delete" buttons to add and delete calibration points.

Use the "Import" button to import calibration points from a calibration that you performed from within the Operate System. To do this, do the following:

- Create the register and check the "Allow user calibration" check box
- Save the database and load it into the U.P.M.A.C.S. Operate System. Do not close the Development System.
- Calibrate the value
- Go back to the Development System and bring up the Calibration Data dialog for the register. You do not need to reload the database.
- Press the "Import" button in the Calibration Data dialog.

The calibration points you added in the Operate System will appear in the list. Press OK to accept them, or modify them first.

- Fit a polynomial to the calibration points:

Check this box to use a polynomial curve fitted to the calibration points to calculate the calibrated value. If you leave this box blank, the values will be calculated by interpolating or extrapolating from the two nearest calibration points.

$y = c + a1\ x + a2\ x^2 + a3\ x^3 + \ldots$:
Select this option if the calibrated value depends directly on the raw value.

$\log y = c + a1\ x + a2\ x^2 + a3\ x^3 + \ldots$:
Select this option if the calibrated value depends exponentially on the raw value.

- Degree of the polynomial:

Select the degree of the polynomial. The degree is the highest exponent used. To fit a straight line, specify a degree of 1. To fit a parabola, specify 2.

- Force c to be 0 (no constant offset):

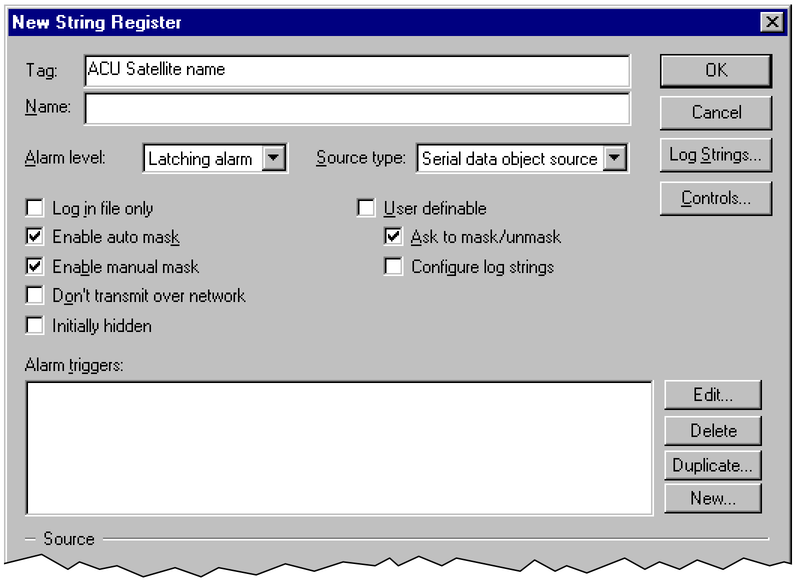
Check this box to force the constant offset c to be 0 in the equation you chose. This means that the calibrated value will be 0 if the raw value is 0, or the logarithm of the calibrated value will be 0 (and calibrated value will be 1), depending on the equation you chose.

String Registers

The value of a string register is arbitrary binary data, including text. You can use string registers for things like satellite names, equipment descriptions, etc..

The ON/OFF state of a string register is determined using a number of alarm triggers. Each trigger is a regular expression. If the value of the register matches any of the triggers, the register goes into its ON state.

The New String Register Dialog



Tag: ACU Satellite name
Name:
Alarm level: Latching alarm
Source type: Serial data object source
☐ Log in file only
☑ Enable auto mask
☑ Enable manual mask
☐ Don't transmit over network
☐ Initially hidden
☐ User definable
☑ Ask to mask/unmask
☐ Configure log strings
Alarm triggers:

The list below describes only those fields that are specific to the New String Register dialog. For the remaining fields, see *The New Register Dialogs* on page 23.

- Alarm Triggers:

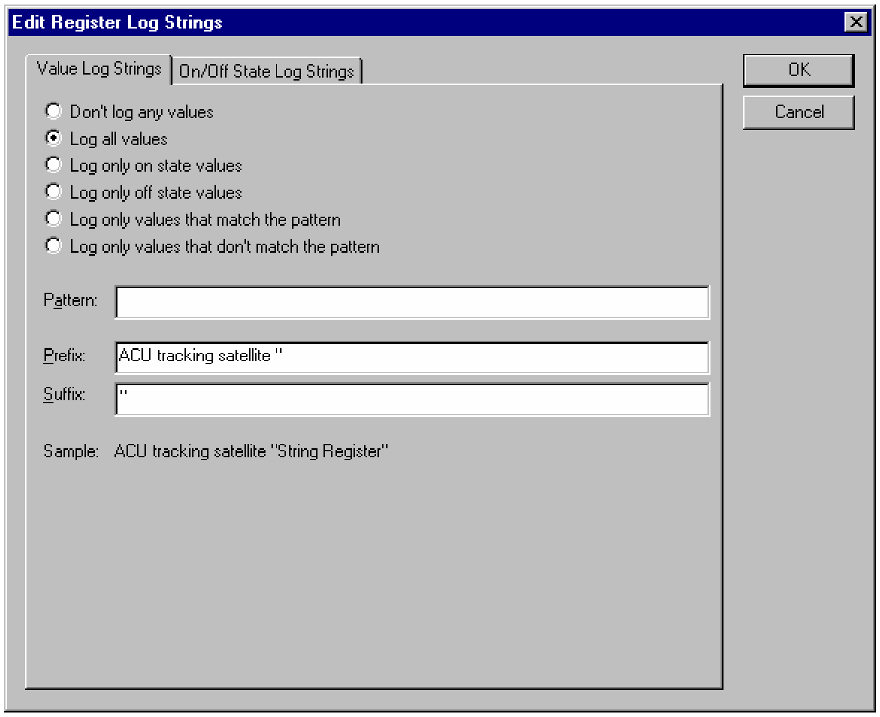
Shows a list of all alarm triggers. Use the buttons to create, delete, and edit alarm triggers. The register goes into the ON state if its value matches any of the triggers.

See *Appendix A: Regular Expressions* on page 171 for details.

String Value Log Strings

You can specify a string to be written to the log file every time the value of a string register changes. The log string writes the register value to the log. A prefix string is prepended to the value, and a suffix string is appended, before it is logged.

The Edit Register Log Strings Dialog for String Registers



Don't log any values [✓ Log all values] Log only on state values Log only off state values Log only values that match the pattern Log only values that don't match the pattern

Pattern:

Prefix: ACU tracking satellite "

Suffix: "

Sample: ACU tracking satellite "String Register"

The list below only shows the fields for value log strings. For the fields for the ON/OFF-state log strings, see *Register Log Strings* on page 24.

- Don't log any values:
Select this option if you don't want any values to be logged.
- Log all values:
Select this option if you would like to log all values.
- Log only on state values:
Select this option if you would like to log only values outside the register's low and high limit.
- Log only off state values:
Select this option if you would like to log only values within the register's low and high limit.
- Log only values that match the pattern:
Select this option if you would like to log only those values that match a regular expression pattern.
- Log only values that don't match the pattern:
Select this option if you would like to log only those values that do not match a regular expression pattern.
- Pattern:
If you selected "Log only values that match the pattern" or "Log only values that don't match the pattern", enter the regular expression for the pattern here. See *Appendix A: Regular Expressions* on page 171 for details on regular expressions.
- Prefix:
Enter the value prefix here. The prefix is prepended to the value before logging it. %-symbols in the prefix will be replaced with the register's name, as described in *Register Log Strings* on page 24.

- Suffix:

Enter the value suffix here. The suffix is appended to the value before logging it. %-symbols in the suffix will be replaced with the register’s name, as described in *Register Log Strings* on page 24.

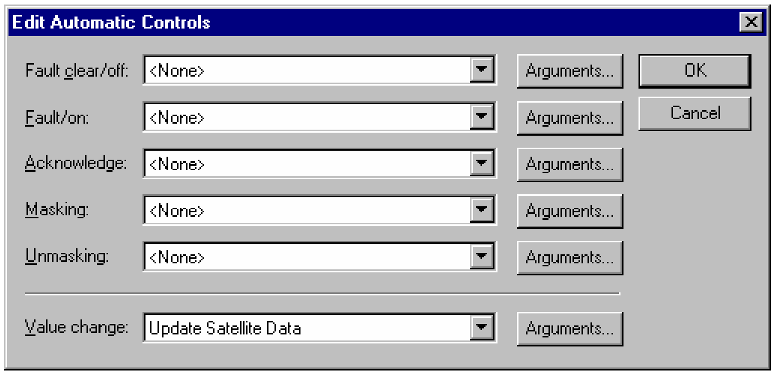
- Sample:

Shows a sample of how values will be logged. The value shown here is the same as that displayed by string indicators.

String Value Controls

You can specify a control to be executed every time the value of a string register changes.

The Edit Automatic Controls Dialog for String Registers



Fault clear/off: <None>
Fault/on: <None>
Acknowledge: <None>
Masking: <None>
Unmasking: <None>
Value change: Update Satellite Data

The list below only shows the fields for value controls. For the other fields, see *Register Log Strings* on page 24.

- Value change:

Select the SCL program you want to be executed when the value changes. The program arguments are shown in parentheses after the names of the program, but you only select the program from the list, not the arguments. To change the arguments, use the “Arguments…” buttons. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

Sources

Every register has a data source. The source specifies where the data for the register is to be taken from, and how. You select the type of source the register uses in the New Register dialogs. The settings for the source do not appear in separate dialogs; they are shown at the bottom of the New Register dialogs, under the heading of “Source”.

Each type of register has a distinct set of source types to choose from. There are four types of sources that all registers share:

- Serial Data Object Sources
- Summary Sources
- Remote Register Value Sources
- Parameter Sources

Bistate registers can have the following additional source types:

- Bit Mask Sources
- Timeout Sources
- Ping Result Sources
- Grand Summary Sources
- Remote Station Alarm Sources

Digital registers can have the following additional source types:

- Thresholds Sources
- Bit Collection Sources

Analog registers can have no additional source types.

String registers can have the following additional source types:

- Filter Sources

Serial Data Object Sources

Serial data object sources take the register's value from a data object of a serial device.

The value of a bistate data object can be inverted before it is used for the ON/OFF state of a bistate register. This means that the register's value will be ON if the object's value is OFF and vice versa.

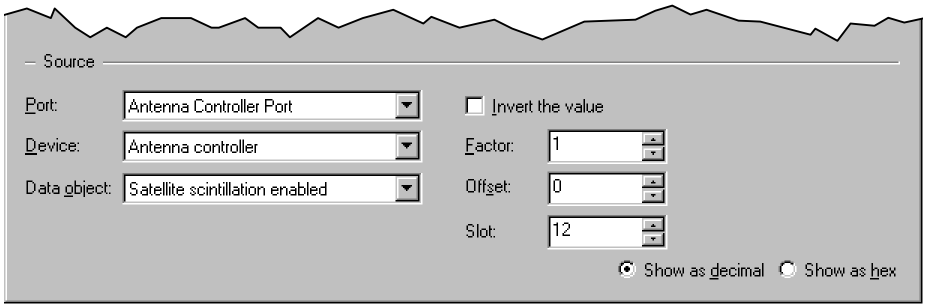
The value of an analog data object can be modified using factor and an offset before it is used as the value of an analog register. The number logged is calculated from the value of the register as follows:

register value = object value · factor + offset

To use the object's value unaltered, use a factor of 1 and an offset of 0.

Use serial data object sources for registers that contain the value of a serial data object.

The Serial Data Object Source Dialog



Source

Port: Antenna Controller Port

Device: Antenna controller

Data object: Satellite scintillation enabled

☐ Invert the value

Factor: 1

Offset: 0

Slot: 12

[✓ Show as decimal] Show as hex

- Port:

Select the serial port that the device is attached to.

- Device:

Select the device.

- Data object:

Select the data object. The data object must have the same type as the register, except for analog registers, which can use digital data objects.

Specify values for all the data object's parameters on the right, below the "Invert the value" box or "Factor" and "Offset" fields, if applicable. In the sample dialog, the source uses a data object that has one digital parameter called "Slot".

If you do not specify a value for a particular data object parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the data object has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the data object has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Invert the value:

Check this box if you want a bistate register to contain the inverse of the data object's value. The register will be ON if the value is OFF and vice versa. This box is only present for bistate registers.

- Factor:

Enter the factor with which the data object's value is to be multiplied before using it for an analog register. The factor is applied before the offset.

This field is only present for analog registers.

- Offset:

Enter the offset that is to be added to the data object's value before using it for an analog register. The offset is applied after the factor.

This field is only present for analog registers.

- Slot:

This is a parameter of the sample object. The parameters for the actual data object you selected will appear here instead.

Summary Sources

Summary sources take information from a number of other registers. Summary sources provide a default implementation for each type of register. The default implementation for summary sources does the following:

Bistate registers:
The register will be in the ON state if any of the summarized registers are in the ON state. The register will be in the OFF state if all of the summarized registers are in the OFF state.

Digital and analog registers:
The register will contain the number of summarized registers that are in the ON state.

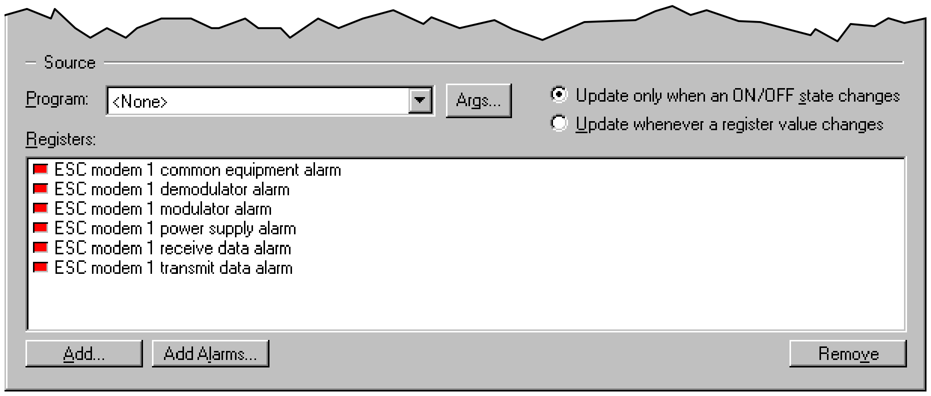
String registers:
The register will contain one line of text for each register that is in the ON state. The line will contain the name of the register that is in the ON state.

You can also provide an SCL program to do your own handling. See *Programs for Sources, Checksums, and SABus Response Data* in the *SCL Language Reference* for details.

If all the registers that are being summarized are in the error state, the register will go into its error state as well. If all register are masked, the register will be auto masked.

Use summary sources for registers whose value is determined from the value of one or more other registers.

The Summary Source Dialog



Program: <None>

(●) Update only when an ON/OFF state changes
( ) Update whenever a register value changes

Registers:
- 🟥 ESC modem 1 common equipment alarm
- 🟥 ESC modem 1 demodulator alarm
- 🟥 ESC modem 1 modulator alarm
- 🟥 ESC modem 1 power supply alarm
- 🟥 ESC modem 1 receive data alarm
- 🟥 ESC modem 1 transmit data alarm

- Program:

Select the SCL program that is to do the evaluation. Select <None> to use the default implementation. The program arguments are shown in parentheses after the name of the program, but you only select the program from the lists, not the arguments. To change the arguments, use the "Arguments…" button. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

- Update only when an ON/OFF state changes:

Select this option to update the register only if the ON/OFF state of one of the registers changes, not when the value changes without affecting the ON/OFF state. This option is only available if you specify a program to evaluate the registers.

- Update only whenever a register value changes:

Select this option to update the register whenever the value of a register changes, even if it does not affect that register's ON/OFF state. This option is only available if you specify a program to evaluate the registers.

- Registers:

Shows a list of all registers that this register summarizes. The register will be updated whenever one of the registers it depends on changes value.

Use the "Add…" and "Remove" buttons to add and remove registers. The "Add Alarms…" button is also used to add registers to the list. If you use the "Add Alarms…" button, you will be asked to select the registers from a list that shows only registers with Alarm or Latching alarm levels.

Remote Register Value Sources

Remote register value sources take the register's value from a register on a remote computer that is also running U.P.M.A.C.S.. The remote register must be of the same type as the local register, except for bistate registers, which can get the ON/OFF state of any type of remote register.

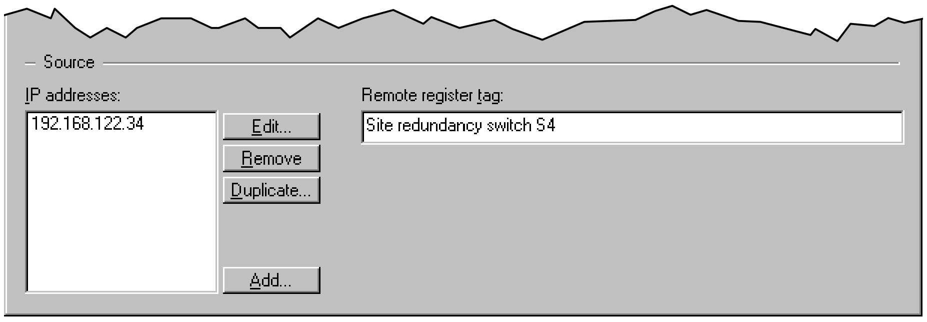
Remote register value sources support multiple redundant backup computers. If the main computer fails, or if a connection cannot be established with it, the source will try to connect to all the backup computers in the order in which you specified them, until a successful connection has been established. If the source is connected to a backup, and that backup goes off line, it will try to reconnect to the primary. If the primary is still not available, all backups are tried again, in order.

The source will auto mask the register if the remote register has been masked (automatically, manually, or internally).

The remote computer must be running U.P.M.A.C.S. v6.0 or later for network sources to work. You can use remote station alarm sources to get alarms from older stations.

Use remote register value sources to implement network overview screens that show data from several different stations.

The Remote Register Value Source Dialog



IP addresses: 192.168.122.34

Remote register tag: Site redundancy switch S4

- IP addresses:

Shows the IP addresses for all the computers you specified. The top address in the list is the primary, all the backups are listed beneath it in order. You can grab and drag the computer names to change their order. Use the buttons to add, edit, and remove addresses.

- Remote register tag:

Enter the tag of the remote register, as defined in the station file of the remote computer.

Parameter Sources

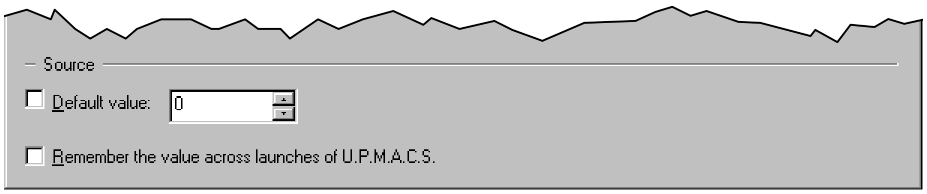
Parameter sources do not get the register's value from anywhere. The value of registers with parameter sources must be set via SCL programs.

Parameter sources support a default value. The register will be set to the default value when the station is opened. If you do not specify a default value, the register will remain in its error state until you set its value from within an SCL program.

Parameter sources can also provide the facility to remember the value of the register across launches of U.P.M.A.C.S.. If you enable this feature, U.P.M.A.C.S. will remember the value the register had when the station is closed, and set it to the same value when it is reopened. In this case, the default value (if any) is used only the very first time the station is loaded.

Use parameter sources for user settings, or for any other data that you maintain yourself rather than getting it from the equipment.

The Parameter Source Dialog



☐ Default value: 0

☐ Remember the value across launches of U.P.M.A.C.S.

- Default value:

Check this box to specify a default value for the register. Specify or select the default value to the right.

Show as text / Show as hex: (not shown)
For string registers, you can select the way you want to enter the default value here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Remember the value across launches of U.P.M.A.C.S.:

Check this box to store the register's value when the station is closed, and restore it when it is reopened.

Bit Mask Sources

Bit mask sources use one or more bits from a digital serial data object to determine the ON/OFF state of a bistate register.

The state is calculated as follows:

- A number is taken from a digital serial data object
- The number is XORed with an XOR mask
- The number is ANDed with an AND mask

The register is set to OFF if the result is 0, and to ON if the result is not zero, or vice versa (depending on the polarity)

The XOR mask inverts all the bits in the number that are 1 in the XOR mask. The AND mask sets all bits that are 0 in the AND mask to 0. To look at a number of bits do the following:

- Set all the bits that are of interest to you in the AND mask
- Set all the bits that need to be inverted (1s become 0s and 0s become 1s) in the XOR mask

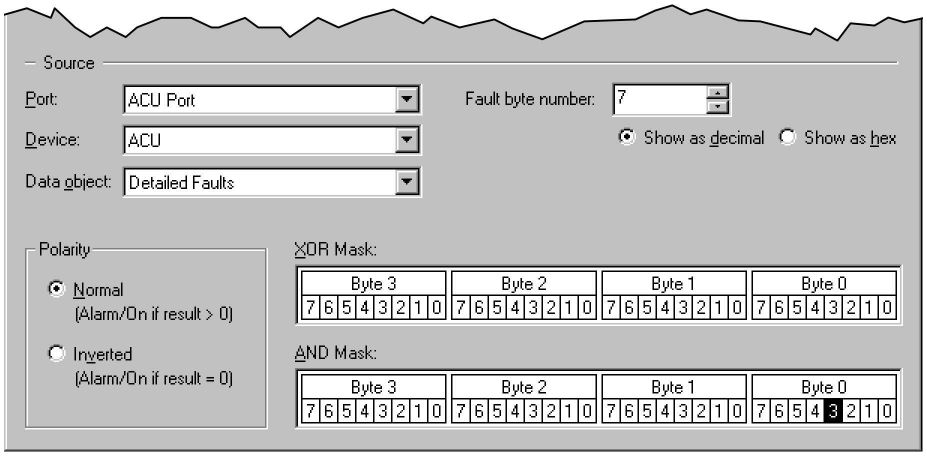
Example:

number (in binary):	0 0 1 0 0 0 1 0	0 0 1 1 0 1 1 0	0 0 0 1 1 0 1 1	1 1 0 1 1 0 0 1
apply XOR mask:	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 1 0 1 0
result:	0 0 1 0 0 0 1 0	0 0 1 1 0 1 1 0	0 0 0 1 1 0 1 1	1 1 0 1 0 0 1 1
apply AND mask:	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 1 1 1 1
result:	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 1

The result is 00000011 00000000 00000000 00000000= 50331648.

Use bit mask sources for bistate registers whose state depends on the value of one or more bits in a digital data object.

The Bit Mask Source Dialog



Source

Port: ACU Port

Device: ACU

Data object: Detailed Faults

Fault byte number: 7

[✓ Show as decimal] Show as hex

Polarity

[✓ Normal (Alarm/On if result > 0)]

Inverted (Alarm/On if result = 0)

XOR Mask:

Byte 3	Byte 2	Byte 1	Byte 0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

AND Mask:

Byte 3	Byte 2	Byte 1	Byte 0
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 [✓ 3] 2 1 0

- Port:

Select the serial port that the device is attached to.

- Device:

Select the device.

- Data object:

Select the data object. Specify values for all the data object's parameters on the right. In the sample dialog, the source uses a data object that has one digital parameter called "Fault byte number".

If you do not specify a value for a particular data object parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the data object has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the data object has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

- AND mask:

Select the bits you want to use. The least significant bit and byte are shown on the right. Black bits will be used; white bits will be masked out to 0s. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

- Polarity:

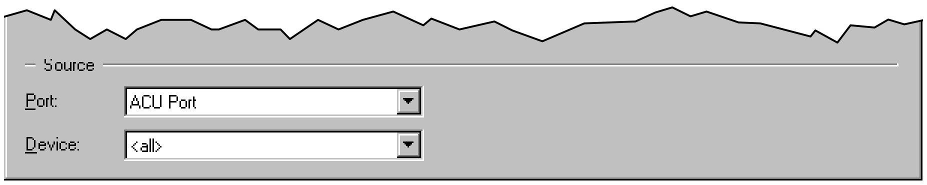
Select the polarity of the register. An alarm can either be triggered by any result greater than 0 (normal polarity), or by the result 0 only (inverted polarity).

Timeout Sources

Registers with timeout sources are set to the ON state if a serial command sent to a device times out. If the device is communicating normally, the register is set to OFF.

Use timeout sources to implement communication state indicators and alarms.

The Timeout Source Dialog



Port: ACU Port

Device: <all>

- Port:

Select the serial port that the device is attached to.

- Device:

Select the device. If you select <all>, the indicator will be set to ON if any of the devices attached to the port time out.

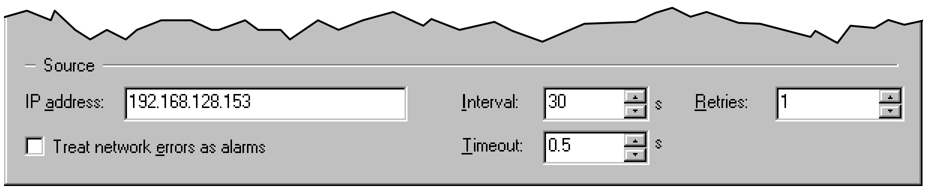
Ping Result Sources

Ping result sources use the result of a ping network query. The register will be OFF if the ping was successful, or ON if the ping timed out.

Usually, the register will go into its error state if a network error other than a ping timeout occurs (no network present, unreachable host, etc.), but the source can be configured to set the register to the ON state on any kind of network error.

Use this source type to determine whether a network device, like a computer, router, print server, etc. is functioning properly.

The Ping Result Source Dialog


Source
IP address: 192.168.128.153
Interval: 30 s
Retries: 1
☐ Treat network errors as alarms
Timeout: 0.5 s

- IP address:

Enter the IP address of the network device you would like to ping.

- Treat network errors as alarms:

If you leave this check box blank, the register will go into its error state if a network error other than a ping timeout occurs. If you check this box, the register will go into its ON state. A network error other than a timeout usually indicates that something is wrong with the networking on the computer that U.P.M.A.C.S. is running on, rather than with the network device you are trying to ping.

- Interval:

Enter the interval between pings. If you enter 30s, the device will be pinged every 30s. The interval must be at least 0.5s longer than the timeout multiplied by the retries. This is necessary so that the total amount of time needed to determine that a device is not working is shorter than the interval.

- Timeout:

Enter the ping timeout in s. This is the time that the network device is allowed to take to respond. If the register tends to go into its ON state even though the network device is working correctly, increase the timeout or the retries (see below).

- Retries:

Enter the number of times U.P.M.A.C.S. tries to ping the device. If the register tends to go into its ON state even though the network device is working correctly, increase the retries or the timeout (see above).

Grand Summary Sources

Grand summary sources summarize the alarm state of all alarms in the station. The register will be ON if any other register with alarm level Alarm or Latching alarm is ON.

Use grand summary sources for global summary alarms, for example to represent the local station on a network map.

Grand summary sources cannot be edited, as they always behave the same. There is no Grand Summary Source dialog.

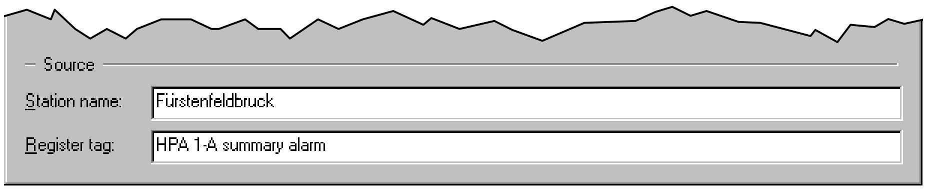
Remote Station Alarm Sources

Remote station alarm sources use the ON/OFF state from a register on a remote station that U.P.M.A.C.S. is connected to. If you do not specify a register, the remote station alarm source will act like a grand summary source for a remote station. Remote station alarm sources are provided to get alarm information from remote stations running versions of U.P.M.A.C.S. prior to v6.0. For stations that run U.P.M.A.C.S. v6.0 or newer, use remote register value sources.

If U.P.M.A.C.S. is not connected to the remote station, the register will be auto-masked.

Use remote station alarm sources to implement network overview screens that show data from stations that run versions of U.P.M.A.C.S. prior to v6.0.

The Remote Station Alarm Source Dialog



Station name: Fürstenfeldbruck

Register tag: HPA 1-A summary alarm

- Station name:

Enter the name that the station will be connected as. This name must be the same as the name entered in the "Connect as:" field of the Connect To dialog in the U.P.M.A.C.S. Operate System.

- Register tag:

Enter the tag of the register whose ON/OFF state you wish to reflect. Leave this field blank to show a grand summary alarm for all registers in the remote station. If you leave this field blank, the register will be ON if any register on the specified station with alarm level Alarm or Latching alarm is ON.

Thresholds Sources

Thresholds sources set the value of a digital register according to the value of a digital or analog serial data object and a set of thresholds. You can specify a value that the register should take if the object's value is above a certain threshold. You must also specify a bottom value for the register. The bottom value is the value the register should have if the number lies below all the thresholds.

Example:

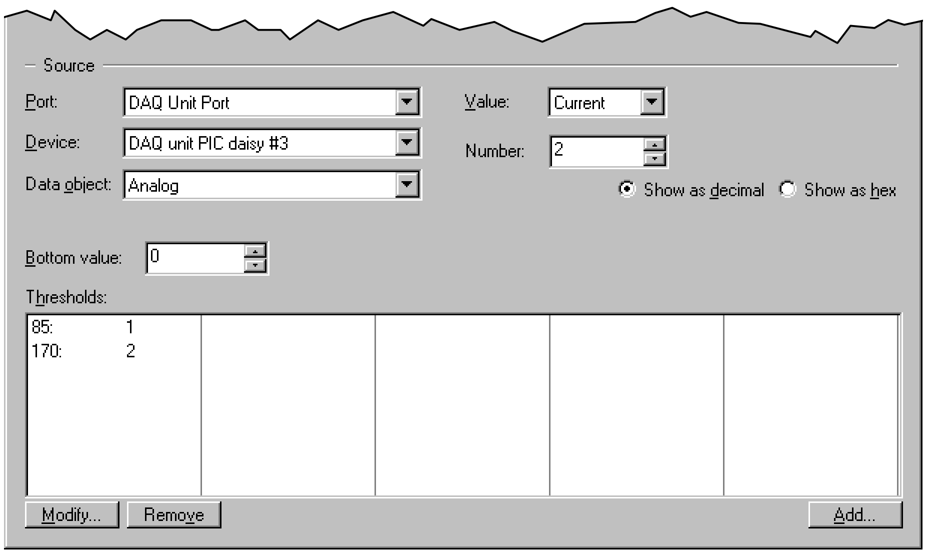
The sample dialog shows the following thresholds and values:

Bottom value:	0
Threshold at 85:	1
Threshold at 170:	2

The register will have value 0 for numbers less than 85, value 1 for numbers of at least 85 but less than 170, and value 2 for numbers of at least 170.

Use thresholds sources for registers whose value reflects a set of states represented as different analog levels on an input of a data acquisition unit.

The Thresholds Source Dialog



- Port:

Select the serial port that the device is attached to.

- Device:

Select the device.

- Data object:

Select the data object. Specify values for all the data object’s parameters on the right, below the “Value” field. In the sample dialog, the source uses a data object that has one digital parameter called “Number”.

If you do not specify a value for a particular data object parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex:
If the data object has any parameters of type digital that don’t have value names, you can select the way you want to enter the parameter values here. Select “Show as decimal” to enter the values in decimal, select “Show as hex” to enter the values in hexadecimal.

Show as text / Show as hex: (not shown)
If the data object has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Value:

For analog data objects, select which value to use.

Current:
Select this option for data objects with a size of one value, or to use the value with the greatest index of a data object with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the data object’s values.

Lowest:
Select this option to use the lowest of all the data object’s values.

This option is not available if you selected a digital data object rather than an analog one.

- Bottom value:

Enter the value that the register should take if the number lies below any of the thresholds.

- Thresholds:

Lists all the thresholds and their corresponding values. Use the buttons to add, remove, or modify thresholds and their values.

Bit Collection Sources

Bit collection sources use bits from one or more digital serial data objects to determine the value of a digital register. The bits are taken from one or more bit sections, and the values of those sections are ORed together. You can then invert bits in the result using an XOR mask.

The value of each section is determined as follows:

- A number is taken from a digital serial data object
- The number is bit rotated to adjust the position of the relevant bits
- Any bits that need to be inverted are inverted using an XOR mask
- The relevant bits are masked out using an AND mask

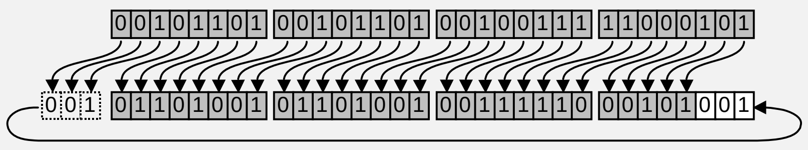
All the sections are then ORed together, and the XOR mask of the bit collection source is applied.

Bit rotation explained:

A bit rotation left moves all bits in a value left. The topmost bits are moved to the bottom, into the spot left vacant when the bottom bits moved left.

A bit rotation right moves all bits in a value right. The bottom most bits are moved to the top, into the spot left vacant when the top bits moved right.

A bit rotation left by 3 looks like this:



You can also specify a value map for the source. Sometimes, it is not possible to manipulate the bits so that you will get the desired values for the desired states. Other times, different results correspond to the same value. A value map will allow you to substitute other values for values calculated by the source. Normally, the result of the calculation described above is used directly as the value of the register. If the result is 2, the value of the register will also be 2. If, however, you would prefer the value of the register to be 4 if the result is 2, then you could specify a value map that maps a source value of 2 to a register value of 4. You can specify register values to be substituted for any number of source values.

Use bit collection sources for digital registers whose value depends on a number of bits from a number of digital data objects.

Examples

There are two basic ways of using bit collection sources.

- Using a number of adjacent bits from a single data object:

Sometimes, the state of a piece of equipment is represented by a number of bits in a status byte. Bits 3 and 4 in a byte might represent a local/computer/remote front panel mode, for example. The two bits might be 00, 01, and 10, depending on the mode. To turn this into a number between 0 and 2, construct one bit section as follows:

number (in binary):	1 1 0 0 1 0 0 1
rotate right by 3 bits:	1 1 1 0 0 1 0 0
	0 1 1 1 0 0 1 0
	0 0 1 1 1 0 0 1
apply XOR mask:	0 0 0 0 0 0 0 0
result:	0 0 1 1 1 0 0 1
apply AND mask:	0 0 0 0 0 0 1 1
result:	0 0 0 0 0 0 0 1

The result of this section is then treated with the global XOR:

result of the section:	0 0 0 0 0 0 0 1
apply global XOR:	0 0 0 0 0 0 0 0
result:	0 0 0 0 0 0 0 1

The value of the register will thus depend on the value of the two bits alone.

- Using bits from different data objects, or using non-adjacent bits:

Sometimes, the bits that represent the value you desire are not right next to each other in the same data object. You might want to turn a standby / transmit state and a warming / ready state of an HPA into a single object, but the states are represented by different bits in different data objects. Let's assume the high bit is bit 0 in one data objects, and the low bit is bit 2 in another data object. You will need two bit sections, as follows:

The first section turns bit 0 of one data object into bit 1 of the result:

number (in binary):	0 1 1 0 1 0 1 1
rotate left by 1 bit:	1 1 0 1 0 1 1 0
apply XOR mask:	0 0 0 0 0 0 0 0
result:	1 1 0 1 0 1 1 0
apply AND mask:	0 0 0 0 0 0 1 0
result:	0 0 0 0 0 0 1 0

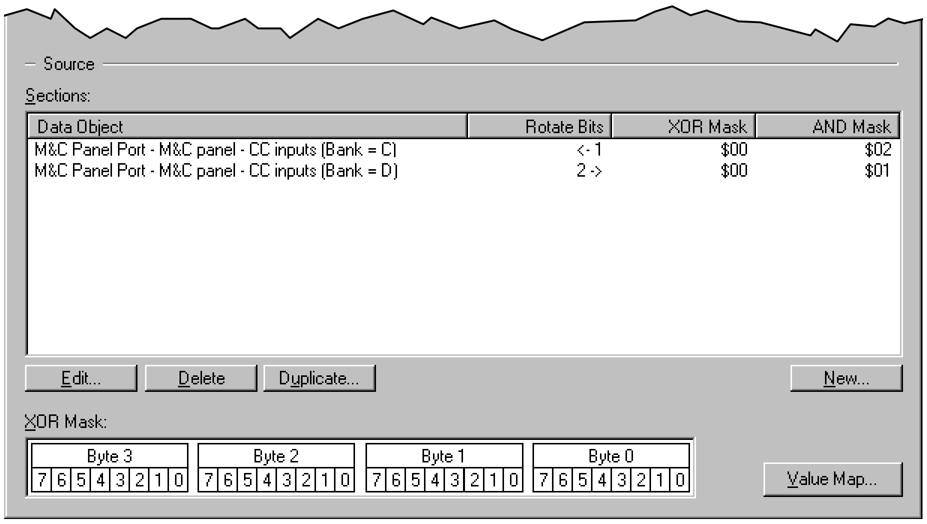
The second section turns bit 2 of the other data object into bit 0 of the result:

number (in binary):	1 1 0 1 1 0 0 1
rotate right by 2 bits:	1 1 1 0 1 1 0 0
	0 1 1 1 0 1 1 0
apply XOR mask:	0 0 0 0 0 0 0 0
result:	0 1 1 1 0 1 1 0
apply AND mask:	0 0 0 0 0 0 0 1
result:	0 0 0 0 0 0 0 0

The result of the sections are ORed together and then treated with the global XOR:

result section 1:	0 0 0 0 0 0 1 0
result section 2:	0 0 0 0 0 0 0 0
ORed together:	0 0 0 0 0 0 1 0
apply global XOR:	0 0 0 0 0 0 0 0
result:	0 0 0 0 0 0 1 0

The Bit Collection Source Dialog



Data Object	Rotate Bits	XOR Mask	AND Mask
M&C Panel Port - M&C panel - CC inputs (Bank = C)	<- 1	$00	$02
M&C Panel Port - M&C panel - CC inputs (Bank = D)	2 ->	$00	$01

- Sections:

Lists the data objects, rotate values, and XOR and AND masks of all the sections. Use the buttons to create, delete, or edit sections. Use the "Duplicate…" button to duplicate the selected section.

See *Bit Sections* on page 54 for a description of the New Bit Section dialog.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

- The "Value Map" button:

Click this button to edit the value map. The value map allows you to substitute different values for the values calculated by the source.

Bit Sections

Bit sections are used in bit collection sources. A bit collection source uses the results of one or more bit sections, ORed together.

The result of a bit section is calculated as follows:

- A number is taken from a digital serial data object
- The number is bit rotated to adjust the position of the relevant bits
- Any bits that need to be inverted are inverted using an XOR mask

➢ The relevant bits are masked out using an AND mask

The XOR mask inverts all the bits in the number that are 1 in the XOR mask. The AND mask sets all bits that are 0 in the AND mask to 0. To look at a number of bits do the following:

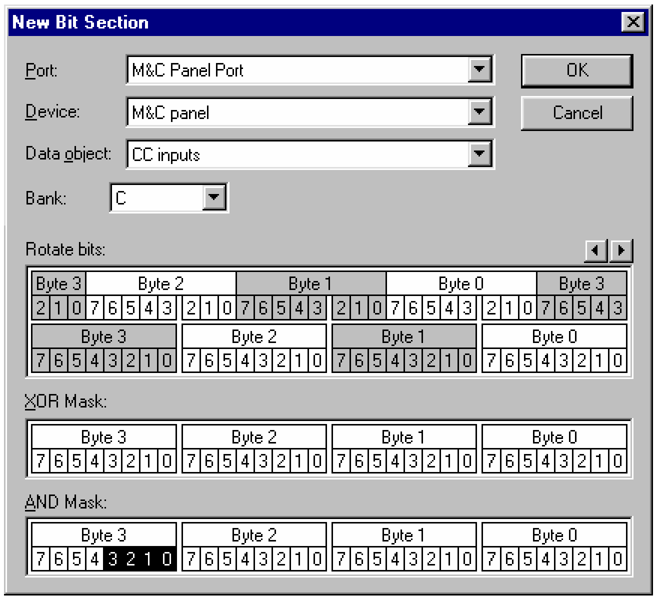
➢ Bit rotate the number left or right to move the bits to the desired location within the result
➢ Set all the bits that are of interested to you in the AND mask
➢ Set all the bits that need to be inverted (1s become 0s and 0s become 1s) in the XOR mask

Example:

number (in binary):	1 1 0 1 1 1 1 0	1 1 0 0 1 0 0 1	0 0 0 1 0 0 0 1	1 0 1 1 0 0 0 0
rotate left by 5 bits:	1 0 1 1 1 1 0 1	1 0 0 1 0 0 1 0	0 0 1 0 0 0 1 1	0 1 1 0 0 0 0 1
	0 1 1 1 1 0 1 1	0 0 1 0 0 1 0 0	0 1 0 0 0 1 1 0	1 1 0 0 0 0 1 1
	1 1 1 1 0 1 1 0	0 1 0 0 1 0 0 0	1 0 0 0 1 1 0 1	1 0 0 0 0 1 1 0
	1 1 1 0 1 1 0 0	1 0 0 1 0 0 0 1	0 0 0 1 1 0 1 1	0 0 0 0 1 1 0 1
	1 1 0 1 1 0 0 1	0 0 1 0 0 0 1 0	0 0 1 1 0 1 1 0	0 0 0 1 1 0 1 1
apply XOR mask:	0 0 0 0 1 0 1 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
result:	1 1 0 1 0 0 1 1	0 0 1 0 0 0 1 0	0 0 1 1 0 1 1 0	0 0 0 1 1 0 1 1
apply AND mask:	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
result:	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

The result is 00000011 00000000 00000000 00000000= 50331648.

The New Bit Section Dialog


New Bit Section

Port: M&C Panel Port

Device: M&C panel

Data object: CC inputs

Bank: C

- Port:

Select the serial port that the device is attached to.

- Device:

Select the device.

- Data object:

Select the data object. Specify values for all the data object's parameters immediately below. In the sample dialog, the bit section uses a data object that has one digital parameter called "Bank".

If you do not specify a value for a particular data object parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex (not shown):
If the data object has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex (not shown):
If the data object has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Rotate bits:

Specify the number of bits to rotate to the left or right. Press the ◀▶ arrows on the top right until the bits align the way you would like them.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte. The masks are applied after the bit rotation.

- AND mask:

Select the bits you want to use. The least significant bit and byte are shown on the right. Black bits will be used; white bits will be masked out to 0s. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte. The masks are applied after the bit rotation.

Filter Sources

Filter sources use the value of a string serial data object as the value of the string register, if and only if it matches one of a number of patterns you specify. If the data object's value does not match any of the patterns, the register's value remains unchanged.

Filter sources are used to cache data that matches a certain pattern. Some equipment, for example, logs messages about events in a message queue and allows you to retrieve the messages in order. In such a case, a filter source lets you remember the last message that referred to a certain piece of information, like for example a certain alarm. If the message in the buffer is overwritten with something that does not refer to that particular alarm, the filter source will ignore the new message, and the register will retain the last relevant one. You can thus attach a number of filter sources to the message buffer; each filtering out messages pertaining to a different alarm. The last message about each alarm will then always be available, even if it has been overwritten in the equipment's message buffer. You can then display the message directly, trigger an alarm based on the message using an alarm trigger, or use a summary source to process the information further.

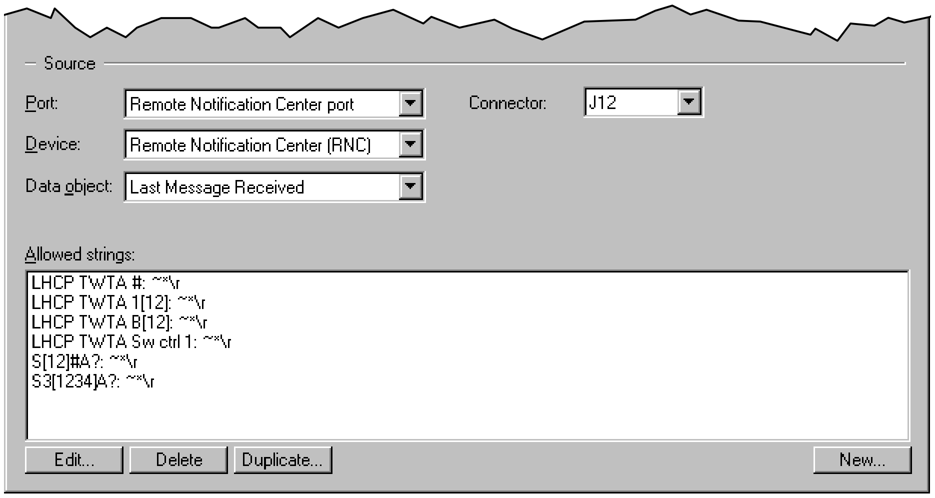
Example:

An LNA controller has only a summary alarm bit, but writes a message to a queue every time an alarm occurs or clears. The message contains the time and date of the state change, the alarm name, and the word "set" or "clear" to describe the new state. In this case, you could create a register for each type of alarm. Each register will have a filter source that filters out only messages containing the alarm name of the alarm the register represents, and an alarm trigger that triggers an alarm if the value contains the word "set".

Each alarm will then trigger when a message arrives saying that the alarm has set. All other messages are then ignored, until a message arrives that says the alarm has cleared. The register's value will then be updated, and the alarm will clear, because the value no longer contains the word "set".

You could also attach another register to each alarm via a summary source, displaying the time the alarm occurred or cleared. Since the register's value is only updated when a message pertaining to its alarm arrives, the date and time of the message stored in the register value will always be the date and time of the last change.

The Filter Source Dialog



Source

Port: Remote Notification Center port

Connector: J12

Device: Remote Notification Center (RNC)

Data object: Last Message Received

Allowed strings:

```
LHCP TWTA #: ~*\r
LHCP TWTA 1[12]: ~*\r
LHCP TWTA B[12]: ~*\r
LHCP TWTA Sw ctrl 1: ~*\r
S[12]#A?: ~*\r
S3[1234]A?: ~*\r
```

- Port:

Select the serial port that the device is attached to.

- Device:

Select the device.

- Data object:

Select the data object. Specify values for all the data object's parameters immediately below. In the sample dialog, the source uses a data object that has one digital parameter called "Connector".

If you do not specify a value for a particular data object parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex (not shown):

If the data object has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex (not shown):
If the data object has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Allowed strings:

Lists regular expressions describing patterns that the data must match. The value of the register will only be changed if the data matches one of the allowed strings.

See *Appendix A: Regular Expressions* on page 171 for details.

Use the buttons to create, delete, and edit the expressions.

SCL Programs

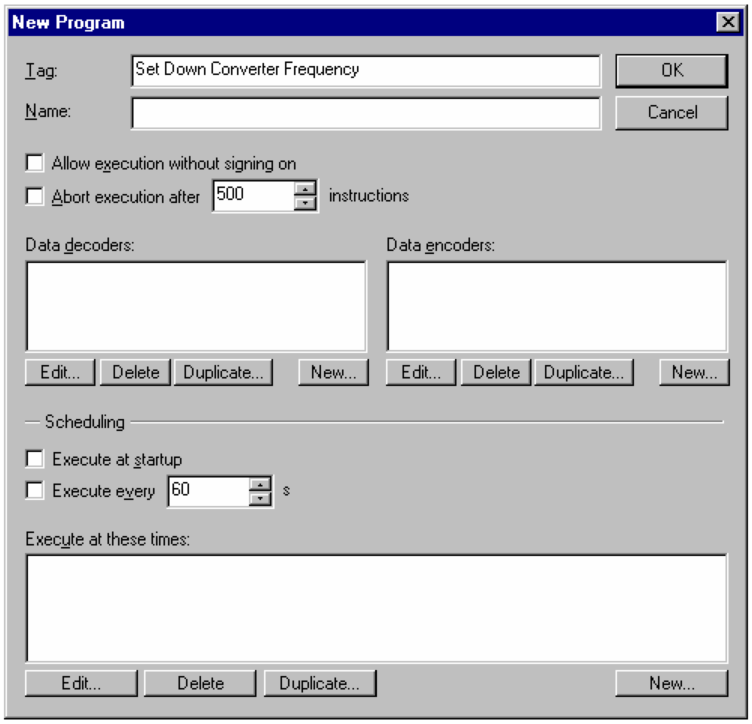
SCL programs are used as manual and automatic controls, in SABus commands, and in SABus processor response data. For a description of the SCL language, see the *SCL Language Reference*.

In addition, SCL programs can be configured to run automatically when a station is loaded, or at regular intervals. You can also specify special times and dates at which a program will be executed.

Once you have created an SCL program, an SCL Program Editor window will appear so that you can let you edit the code.

If you select a program in the Programs window, and press the "Edit…" button, the editor window will appear rather than the Edit Program dialog. To change the properties of a program, select it and press the "Properties…" button. You can also select "Properties…" from the "Edit" menu when viewing the program's code.

The New Program Dialog



Tag: Set Down Converter Frequency
Name:
Abort execution after 500 instructions
Execute every 60 s

- Tag:

Enter the tag by which the program is identified. Each program must have a unique tag.

- Name:

Enter the name of the program. Leave this field blank if you want to use the tag as name.

- Allow execution without signing on:

If you check this box, this SCL program can be executed even by a user that does not have Control Privileges, when no user is signed on, or if someone is signed on from a different location. This means that control buttons that use this program will always be visible, and SABus commands that use this program will never be rejected with the 'USR' error message.

Please remember that the operator will usually assume that he has exclusive control of the station when he is signed on. You should only check this box for programs that do nothing but provide information to the operator, never for programs that change settings or control equipment. Otherwise, it is possible for several users at several different locations to attempt to perform the same action at the same time, usually with unexpected or confusing results.

- Abort execution after … instructions:

Check this box if you want to guard the program against endless loops. The program will be aborted after the specified number of SCL commands or assignments have been executed. Programs used in processor and summary sources have a built-in instruction limit of 500 instructions. Check this box to specify a greater or smaller limit.

- Data decoders:

Shows a list of all data decoders defined for this program. use the buttons, to add, delete, and edit decoders.

See ***Error! Reference source not found.*** on page **Error! Bookmark not defined.** for details.

- Data encoders:

Shows a list of all data encoders defined for this program. use the buttons, to add, delete, and edit encoders.

See ***Error! Reference source not found.*** on page **Error! Bookmark not defined.** for details.

- Execute at startup:

Check this box to execute the program as soon as the station is loaded.

- Execute every … s:

Check this box to execute the program at regular intervals. Specify the interval in seconds. You can enter fractions of a second. If the program takes longer to execute than the interval specified, a second instance of the program is started.

- Scheduled execution times:

Shows a list of times and dates at which the program will be executed automatically. Use the buttons to create, delete, or edit the scheduled times.

See *SCL Program Scheduling* below for a description of the Schedule Execution Time dialog.

SCL Program Scheduling

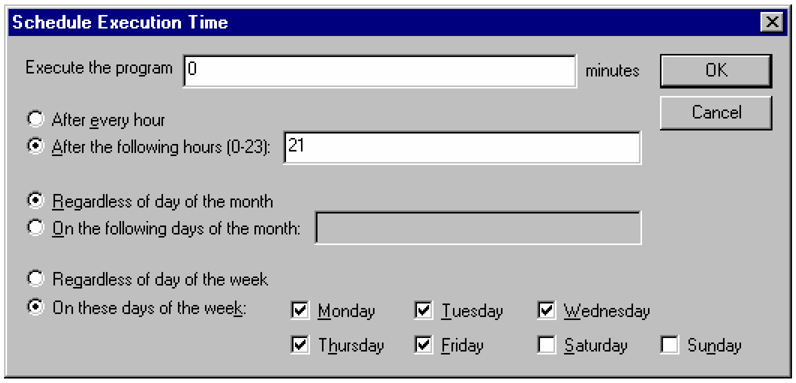
You can specify special times and dates at which to execute an SCL program automatically. You can, for example, execute a program automatically:

- 5 minutes after the hour, every hour
- at 5:03, 7:03, and 14:03 every day
- at 9:00 every Monday and Thursday
- at 17:00 every 1st and 15th of the month

- midnight on Halloween

Press the "New…" or "Edit" buttons beneath the list of Scheduled execution times in the New Program dialog to pop up the Scheduled Execution Time dialog.

The Scheduled Execution Time Dialog


Schedule Execution Time

Execute the program 0 minutes

( ) After every hour
(●) After the following hours (0-23): 21

(●) Regardless of day of the month
( ) On the following days of the month:

( ) Regardless of day of the week
(●) On these days of the week: [✓] Monday [✓] Tuesday [✓] Wednesday [✓] Thursday [✓] Friday [ ] Saturday [ ] Sunday

- Execute the program … minutes:
Enter the minutes after the hour that you want the program executed. To execute the program at 5:03, enter 3 here. You can specify several values, separated by commas. To execute the program at 5:00 and 5:30, enter "0, 30".

- After every hour:
Select this radio button if you want to execute the program the specified number of minutes after every hour.

- After the following hours (0-23):
Select this radio button if you want to execute the program the specified number of minutes after specific hours only. Enter the hours, in 24-hour time, after which the program is to be executed. To execute the program at 5:03, enter 5 here. You can specify several values, separated by commas. To execute the program at 9:30 and 14:30, enter "9, 14".

- Regardless of the day of the month:
Select this radio button if you want to execute the program regardless of the day of the month. You should select the button if you want to execute the program every day, or if you want to execute it on specific days of the week.

- On the following days of the month:
Select this radio button if you want to execute the program on specific days of the month only. Enter the days of the month on which you want the program to be executed. You can enter more than one value, separated by commas. To execute the program on the 1st and 15th of every month, enter "1, 15".

If you select this option and the "On these days of the week" option (see below), the program will be executed only on days that are both the correct day of the month and the correct day of the week. E.g., selecting the 13th of the month and Friday will execute the program only on Friday the 13th.

- Regardless of the day of the week:
Select this radio button if you want to execute the program regardless of the day of the week. You should select the button if you want to execute the program every day, or if you want to execute it on specific days of the month.

- On these days of the week:

Select this radio button if you want to execute the program on specific days of the week only. Select the days of the week on which you want the program to be executed by checking the appropriate check boxes. You can check any number of boxes.

If you select this option and the "On the following days of the month" option (see above), the program will be executed only on days that are both the correct day of the month and the correct day of the week. E.g., selecting the 13^{th} of the month and Friday will execute the program only on Friday the 13^{th}.

The SCL Program Editor

SCL Program Editor windows are used to edit the code of an SCL program. The window will automatically appear when you create a program. To open an SCL editor window for a program once you have closed it, select the program in the Programs window, and press the "Edit…" button, or double-click on the program's name.

Syntax Colouring

The SCL Program Editor uses different colours to display different kinds of SCL language components. See *SCL Syntax Colouring* on page 66 for details.

Insert/Overwrite Mode

Usually, text you type will be inserted into the code at the cursor position without removing any other text. You can switch the editor to Overwrite mode, however, by pressing the Insert key. If you then type something, the text will overwrite the existing code character by character. Press the Insert key again to return to Insert mode. As long as the editor is in overwrite mode, the three letters "OVR" appear in the status bar, near the lower right hand corner of the Development System Window.

The Selection Margin

The selection margin is a grey stripe along the left side of the editor window. Click in the selection margin to select an entire line of SCL code.

Drag-And-Drop Editing

The SCL Program Editor supports drag-and-drop editing. To move a section of text from one place in the code to another, select it, and then click on it and hold the mouse button down. Drag the mouse until the dotted insertion cursor is at the desired location, and let go of the mouse button. The selected text will be moved to the new location.

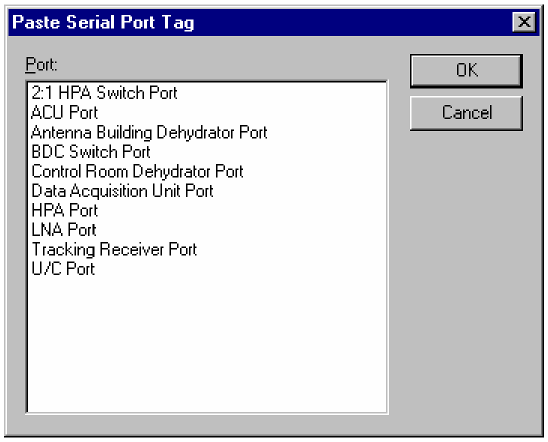
To move a copy of the text, without removing the original, hold down the Ctrl key when you release the mouse button.

Paste Special

Many SCL commands require you to enter the tag of a database object. The SCL Program Editor allows you to paste the tag of certain objects, enclosed in double quotes (`""`), using the Paste Special feature.

To paste a database object tag, select one of the options under "Paste Special" in the "Edit" menu:

- Paste Port Tag:

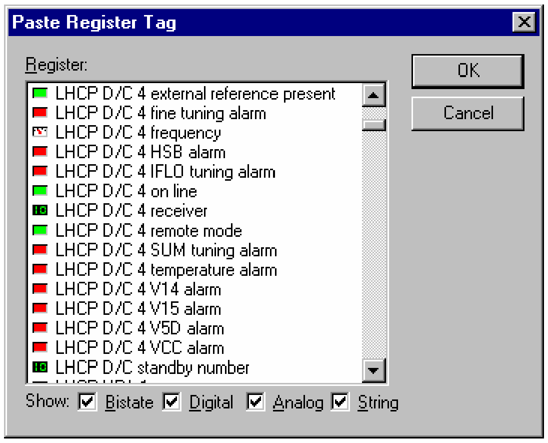


2:1 HPA Switch Port
ACU Port
Antenna Building Dehydrator Port
BDC Switch Port
Control Room Dehydrator Port
Data Acquisition Unit Port
HPA Port
LNA Port
Tracking Receiver Port
U/C Port

The Paste Serial Port Tag dialog.

Select this option to paste the tag of a serial port, enclosed in double quotes ("").

- Paste Register Tag:



LHCP D/C 4 external reference present
LHCP D/C 4 fine tuning alarm
LHCP D/C 4 frequency
LHCP D/C 4 HSB alarm
LHCP D/C 4 IFLO tuning alarm
LHCP D/C 4 on line
LHCP D/C 4 receiver
LHCP D/C 4 remote mode
LHCP D/C 4 SUM tuning alarm
LHCP D/C 4 temperature alarm
LHCP D/C 4 V14 alarm
LHCP D/C 4 V15 alarm
LHCP D/C 4 V5D alarm
LHCP D/C 4 VCC alarm
LHCP D/C standby number

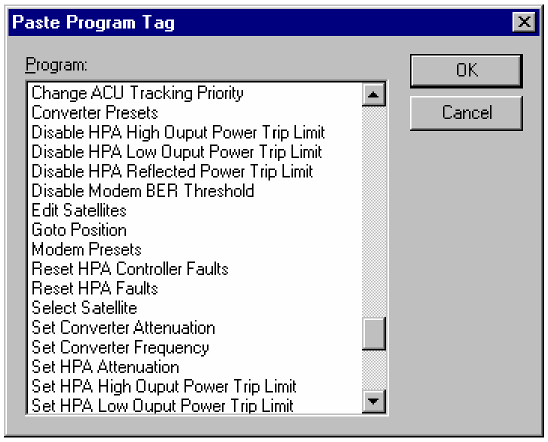
Show: [✓ Bistate] [✓ Digital] [✓ Analog] [✓ String]

The Paste Register Tag dialog.

Select this option to paste the tag of a register, enclosed in double quotes ("").

Select the register whose tag you want to paste from the list and press OK, or double-click on it. You can select which types of registers to display using the "Show" check boxes at the bottom of the dialog. The registers are marked with the same icons as in the register window to show their type.

- Paste Program Tag:

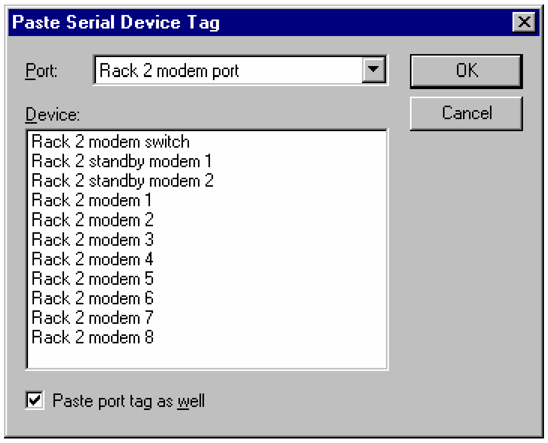


Change ACU Tracking Priority
Converter Presets
Disable HPA High Ouput Power Trip Limit
Disable HPA Low Ouput Power Trip Limit
Disable HPA Reflected Power Trip Limit
Disable Modem BER Threshold
Edit Satellites
Goto Position
Modem Presets
Reset HPA Controller Faults
Reset HPA Faults
Select Satellite
Set Converter Attenuation
Set Converter Frequency
Set HPA Attenuation
Set HPA High Ouput Power Trip Limit
Set HPA Low Ouput Power Trip Limit

The Paste Program Tag dialog.

Select this option to paste the tag of another SCL program, enclosed in double quotes (`""`).

- Paste Device Tag:



Port: Rack 2 modem port

Rack 2 modem switch
Rack 2 standby modem 1
Rack 2 standby modem 2
Rack 2 modem 1
Rack 2 modem 2
Rack 2 modem 3
Rack 2 modem 4
Rack 2 modem 5
Rack 2 modem 6
Rack 2 modem 7
Rack 2 modem 8

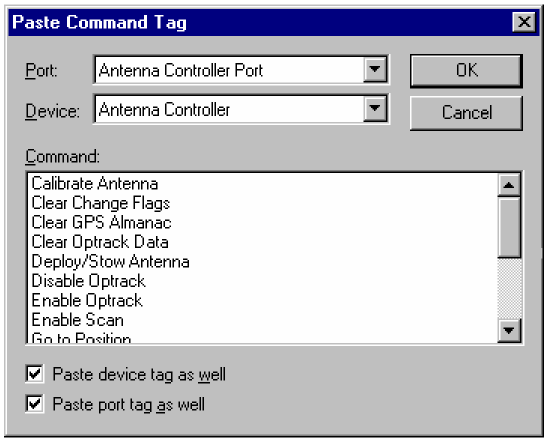
☑ Paste port tag as well

The Paste Device Tag dialog.

Select this option to paste the tag of a serial device, enclosed in double quotes (`""`).

Select the port that the device is attached to in the "Port" field, and highlight the device whose tag you want to paste. If you check the "Paste port tag as well" check box, the port tag will be pasted before the driver tag, separated by a comma.

- Paste Command Tag:



Port: Antenna Controller Port
Device: Antenna Controller
Command:
Calibrate Antenna
Clear Change Flags
Clear GPS Almanac
Clear Optrack Data
Deploy/Stow Antenna
Disable Optrack
Enable Optrack
Enable Scan
Go to Position

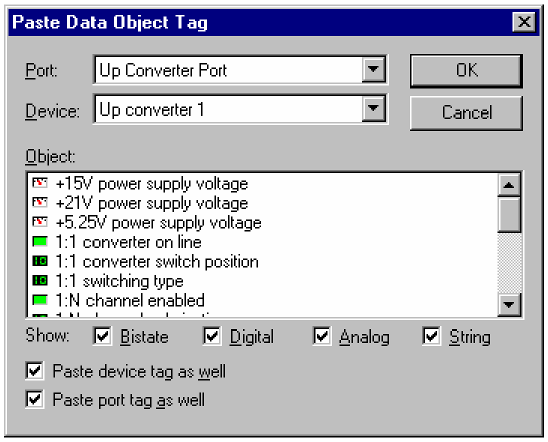
☑ Paste device tag as well
☑ Paste port tag as well

The Paste Command Tag dialog.

Select this option to paste the tag of a serial device command enclosed in double quotes (" ").

Select the port that the command's device is attached to in the "Port" field, select the device in the "Device" field, and highlight the command whose tag you want to paste. If you check the "Paste device tag as well" check box, the device tag will be pasted before the command tag, separated by a comma. If you check the "Paste port tag as well" check box, the port tag will be pasted before the device tag, also separated by a comma.

- Paste Data Object Tag:



Port: Up Converter Port
Device: Up converter 1
Object:
+15V power supply voltage
+21V power supply voltage
+5.25V power supply voltage
1:1 converter on line
1:1 converter switch position
1:1 switching type
1:N channel enabled

Show: ☑ Bistate ☑ Digital ☑ Analog ☑ String

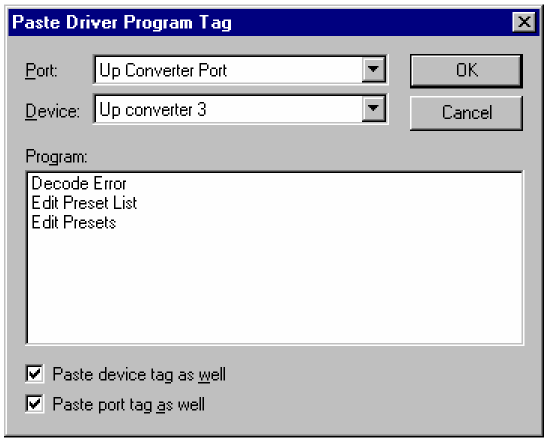
☑ Paste device tag as well
☑ Paste port tag as well

The Paste Data Object Tag dialog.

Select this option to paste the tag of a serial data object enclosed in double quotes (" ").

Select the port that the data object's device is attached to in the "Port" field, select the device in the "Device" field, and highlight the data object whose tag you want to paste. If you check the "Paste device tag as well" check box, the device tag will be pasted before the data object tag, separated by a comma. If you check the "Paste port tag as well" check box, the port tag will be pasted before the device tag, also separated by a comma.

- Paste Driver Program Tag:



Port: Up Converter Port
Device: Up converter 3
Program:
Decode Error
Edit Preset List
Edit Presets

☑ Paste device tag as well
☑ Paste port tag as well

The Paste Driver Program Tag dialog.

Select this option to paste the tag of a device driver program enclosed in double quotes (`""`).

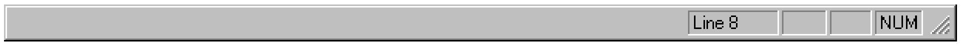
Select the port that the program's device is attached to in the "Port" field, select the device in the "Device" field, and highlight the program whose tag you want to paste. If you check the "Paste device tag as well" check box, the device tag will be pasted before the program tag, separated by a comma. If you check the "Paste port tag as well" check box, the port tag will be pasted before the device tag, also separated by a comma.

Undo/Redo

The SCL Program Editor supports multiple Undos and Redos. Select "Undo" from the "Edit" menu to undo something you did, select "Redo" to redo the last thing you undid.

Line Numbers

When the cursor is in an SCL Program Editor window, the number of the line the cursor is currently in appears in the status bar. The status bar is located at the bottom of the U.P.M.A.C.S. Development System window.



If you want to go to a specific line, select "Go To Line…" from the "Edit" menu, and enter the desired line number in the Go To Line dialog.

The line numbers in the status bar and the Go To Line editor refer to the lines of text in the SCL program editor, not the SCL line numbers used for jumps and subroutines.

Bookmarks

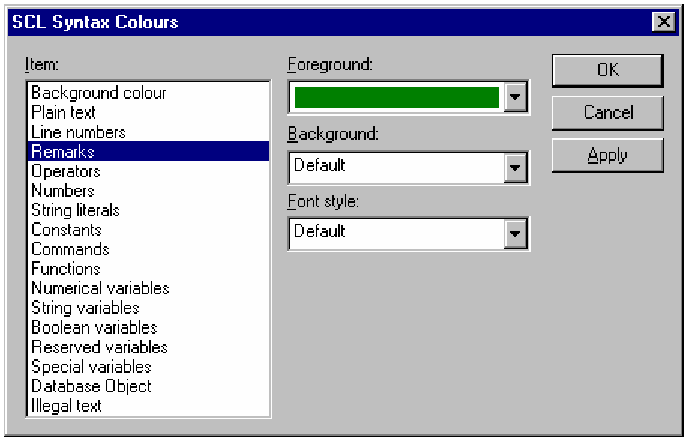
You can mark certain lines in an SCL Program Editor window with a bookmark. To set or remove a bookmark, move the cursor to the line you want to bookmark, and select "Toggle Bookmark" from the "Edit" menu. Once you bookmark a line, a red square appears next to it in the window. To remove all bookmarks, select "Clear All Bookmarks" from the "Edit" menu.

To move the cursor to a bookmark, use the "Previous Bookmark" and "Next Bookmark" items in the "Edit" menu.

SCL Syntax Colouring

The SCL Program Editor uses different colours to display different kinds of SCL language components. To change the colours used for the different language components, select “SCL Syntax Colours…” from the “Edit” menu.

The SCL Syntax Colours Dialog



- Item:
Select the item for which you want to change the colours.

▪ Select	▪ To change
Background colour	Background colour of the SCL editor windows
Plain text	Colours for normal text
Line numbers	Colours for line numbers
Remarks	Colours for remarks (comments)
Operators	Colours for operators such as `+`, `-`, `*`, `AND`, `OR`, etc.
Numbers	Colours for numbers
String literals	Colours for strings in double quotes (`""`)
Constants	Colours for names of constants like `PI` and `RET$`
Commands	Colours for names of commands
Functions	Colours for names of functions
Numerical variables	Colours for names of numerical variables
String variables	Colours for names of string variables
Boolean variables	Colours for names of Boolean variables
Reserved variables	Colours for names of reserved variables like `TIME$` and `USR$`
Special variables	Colours for the special variables `RESULT`, `RESULT$`, `RESULT%`, `GLRESULT`, and `MASKRESULT%`
Database objects	Colours for tags of database object in double quotes (`""`)
Illegal text	Colours for illegal characters and misplaced text

- Foreground:
Select the colour of the text for the current item.

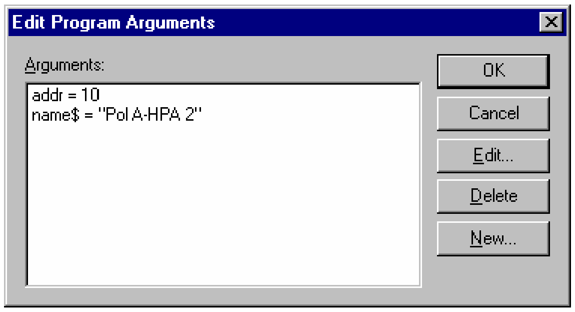
- Background:
Select the background colour for the current item.

- Text style:
Select the text style for the current item.

Specifying Arguments for SCL Programs

Whenever an SCL program is used, you can specify arguments for it. The arguments simply consist of variables that are initialized to values other than their default value. See *Program Arguments* in the *SCL Language Reference* for details.

The Edit Program Arguments Dialog



```
addr = 10
name$ = "Pol A-HPA 2"
```

- Arguments:
Shows a list of the arguments currently defined.
- The “Edit…” button:
Press this button to change the name or value of the selected argument.
- The “Delete” button:
Press this button to delete the selected argument.
- The “New…” button:
Press this button to add a new argument.

SCL Data Decoders and Encoders

You can define one or more data decoders and encoders for an SCL program. These decoders and encoders are used to construct or interpret SCL strings using special SCL commands and functions. See *Decoding and Encoding Data* in the *SCL Language Reference* for details.

A decoder tells the SCL program how to parse a value from a data string. An encoder tells the program how to write a value to a data string. Each decoder and each encoder has a decoder or encoder number, which is used to tell the SCL function or command which decoder or encoder to use.

There are five types of decoders:

- Numerical decoders (floating point number)
- Numerical decoders (unsigned integer)
- Numerical decoders (set of strings)
- String decoders
- Boolean decoders

There are also the equivalent five types of encoders:

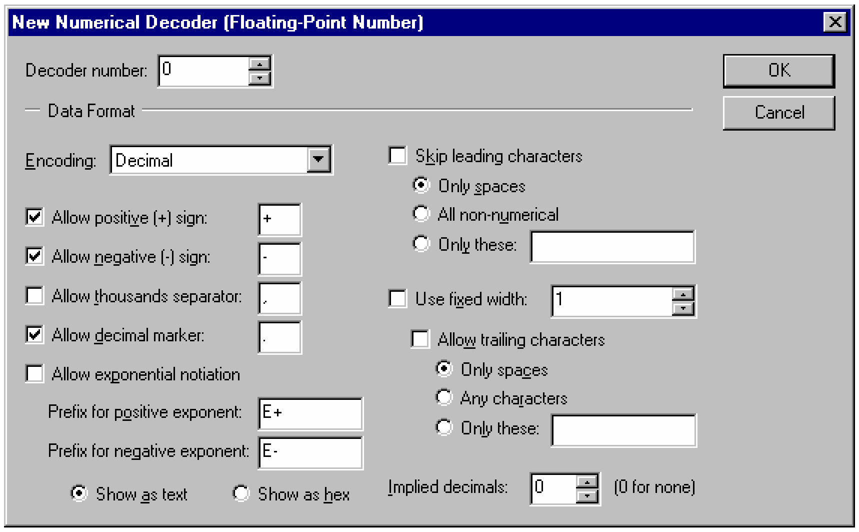
- Numerical encoders (floating point number)
- Numerical encoders (unsigned integer)
- Numerical encoders (set of strings)
- String encoders
- Boolean encoders

Boolean decoders and encoders are used for values that can be expressed as either true or false (or on and off), similar to the values of bistate registers.

Numerical Decoders (Floating-Point Number)

Floating-point number decoders get a numerical value with virtually unlimited range from a data string. Floating-point number decoders support decimal fractions as well as exponential (scientific) notation.

The New Numerical Decoder (Floating-Point Number) Dialog


New Numerical Decoder (Floating-Point Number)

Decoder number: 0

Data Format

Encoding: Decimal

[✓] Allow positive (+) sign: +
[✓] Allow negative (-) sign: -
[ ] Allow thousands separator: ,
[✓] Allow decimal marker: .
[ ] Allow exponential notation
Prefix for positive exponent: E+
Prefix for negative exponent: E-
[✓ Show as text] Show as hex

[ ] Skip leading characters
[✓ Only spaces] All non-numerical Only these:
[ ] Use fixed width: 1
[ ] Allow trailing characters
[✓ Only spaces] Any characters Only these:
Implied decimals: 0 (0 for none)

- Decoder number:

Enter the decoder number by which the decoder is identified. Each decoder in an SCL program must have a unique decoder number.

- Encoding:

Select the encoding method that the number is encoded in. U.P.M.A.C.S. supports the following encoding methods:

Byte:
A single byte (character) in the command string is used to represent the number as an 8-bit value. If you select signed byte encoding, the most significant bit of the byte will be interpreted as the sign in the standard way ($FF is -1, $FE is -2, etc.). The byte encoding method requires a fixed width of 1.

Multibyte:
Two or more bytes (characters) in the command string are used to represent the number as a 16, 24, or 32-bit value. You can choose between lo-hi and hi-lo byte ordering. If you select lo-hi byte ordering, the least significant byte (the lower 8 bits) must appear in the response first; if you selct hi-lo byte ordering, the most significant byte (the upper 8 bits) must appear first. If you select signed multibyte encoding, the most significant bit of the most significant byte will be interpreted as the sign in the standard way (for 16-bit values $FFFF is -1,$FFFE is -2, etc.). The multibyte encoding method requires a fixed width of 2, 3, or 4.

BCD:
The number is encoded using Binary Coded Decimal encoding. In BCD encoding, each nibble (hex digit) in a byte represents one decimal digit. The number 20,841,057 would be encoded as the byte (character) values hex 20 (32), hex 84 (132), hex 10 (16), and hex 57 (87). The BCD encoding method requires a fixed width.

Decimal, Hexadecimal, Binary, Octal:
The number is written out as a decimal, hexadecimal, binary, or octal number using ASCII characters. Hexadecimal encoding recognizes both capital and small letters ("A" to "F" and "a" to "f") as hex digits.

Do not confuse the byte/multibyte and binary encoding methods. The binary encoding expects the number to be written out as a series of ASCII characters 1 (hex 31) and 0 (hex 30), not using the individual bits of each byte.

- Allow positive (+) sign:

Check this box to allow a plus sign to denote positive numbers. The plus sign does not actually have to be the "+" character. You can specify any character you want.

This setting only applies to the decimal encoding.

- Allow negative (-) sign:

Check this box to allow a minus sign to denote negative numbers. If you leave this check box blank, all numbers will be positive. The minus sign does not actually have to be the "-" character. You can specify any character you want.

This setting only applies to the decimal encoding.

- Allow thousands separator:

Check this check box if you want digits to be separated into groups of three using a thousands separators (usually a comma). If you check this box, you must also specify the actual character used to group digits.

This setting only applies to the decimal encoding.

- Allow decimal marker:

Check this box to allow a decimal marker and decimals. If you leave this check box blank, all numbers will be whole numbers. If you leave this box checked, you can specify the character used as a decimal marker.

This setting only applies to the decimal encoding.

- Allow exponential notation:

Check this box to allow exponential (scientific) notation with the decimal encoding method.

- Prefix for positive / negative exponent:

Enter the exponent markers for positive and negative exponents here. The exponent markers must include the sign of the exponent. Usually, the positive exponent marker is "E+" or "e+", and the negative marker is "E-" or "e-". If the plus sign is omitted for positive exponents, specify "E" or "e" for the positive exponent prefix. The exponent prefixes can be any arbitrary data. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

Show as text / Show as hex:
Select the way you want to enter the exponent prefixes. See *Appendix B: Entering Binary Data* on page 179 for details.

- Skip leading characters:

Check this box to skip any characters that appear before the number in the data string. This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

Only spaces / All non-numerical / Only these:
Specify the characters that are allowed to appear before the number.

- Use fixed width:

Check this check box to require the number to be a fixed number of bytes (characters) long. Specify the number of bytes in the edit box.

You must check this box for the byte, multibyte, and BCD encoding methods.

Allow trailing characters:
Check this box to allow characters that are not part of the number to appear after it in the data string. This only applies to characters within the fixed width. All characters are always allowed to appear beyond the fixed width. Specify the characters that are allowed to appear after the number using the radio buttons.

This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

- Implied decimals:

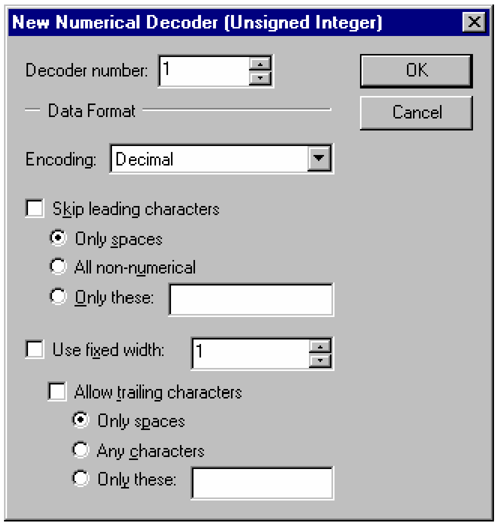
Specify the number of implied decimals here. Implied decimals are a way of encoding fractions without using a decimal point by multiplying it with a power of ten before encoding it. A number that is encoded using 3 implied decimals, for example, will be multiplied by 1000 before it is encoded. This will move the three decimals from the right to the left of the decimal point, and the decimal point will no longer be needed. 21.304, for example, will simply be encoded as 21304.

Please note that the implied decimals for the hexadecimal, octal, and binary encoding methods are decimal fractions, not hexadecimal, octal or binary fractions. In hexadecimal encoding with three implied decimals, 43.249 (which is~2B.3FC in hex) will appear as A8F1 (which is 43249 in decimal) rather than 2B3FC. In other words, the number is always multiplied by powers of 10, even if the encoding method uses base 16, 2 , or 8.

Numerical Decoders (Unsigned Integer)

Unsigned integer decoders get an integer value between 0 and 4,294,967,295 (hex FFFFFFFF) from a data string. The functionality of unsigned integer decoders is a sub-set of that of floating-point number decoders. Use a floating-point number decoder for values that can be less or greater than the range of an unsigned integer, for values that contain fractions, or for values that use exponential (scientific) notation.

The New Numerical Decoder (Unsigned Integer) Dialog


New Numerical Decoder (Unsigned Integer)

Decoder number: 1

Data Format

Encoding: Decimal

[ ] Skip leading characters
(●) Only spaces
( ) All non-numerical
( ) Only these:

[ ] Use fixed width: 1

[ ] Allow trailing characters
(●) Only spaces
( ) Any characters
( ) Only these:

- Decoder number:

Enter the decoder number by which the decoder is identified. Each decoder in an SCL program must have a unique decoder number.

- Encoding:

Select the encoding method that the number is encoded in. U.P.M.A.C.S. supports the following encoding methods:

Byte:
A single byte (character) in the command string is used to represent the number as an 8-bit value. Requires a fixed width of 1.

Multibyte:
Two or more bytes (characters) in the command string are used to represent the number as a 16, 24, or 32-bit value. You can choose between lo-hi and hi-lo byte ordering. If you select lo-hi byte ordering, the least significant byte (the lower 8 bits) must appear in the response first; if you selct hi-lo byte ordering, the most significant byte (the upper 8 bits) must appear first. The multibyte encoding method requires a fixed width of 2, 3, or 4.

BCD:
The number is encoded using Binary Coded Decimal encoding. In BCD encoding, each nibble (hex digit) in a byte represents one decimal digit. The number 20,841,057 would be encoded as the byte (character) values hex 20 (32), hex 84 (132), hex 10 (16), and hex 57 (87). The BCD encoding method requires a fixed width.

Decimal, Hexadecimal, Binary, Octal:
The number is written out as a decimal, hexadecimal, binary, or octal number using ASCII characters. Hexadecimal encoding recognizes both capital and small letters ("A" to "F" and "a" to "f") as hex digits.

Do not confuse the byte/multibyte and binary encoding methods. The binary encoding expects the number to be written out as a series of ASCII characters 1 (hex 31) and 0 (hex 30), not using the individual bits of each byte.

- Skip leading characters:

Check this box to skip any characters that appear before the number in the data string. This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

Only spaces / All non-numerical / Only these:
Specify the characters that are allowed to appear before the number.

- Use fixed width:

Check this check box to require the number to be a fixed number of bytes (characters) long. Specify the number of bytes in the edit box.

You must check this box for the byte, multibyte, and BCD encoding methods.

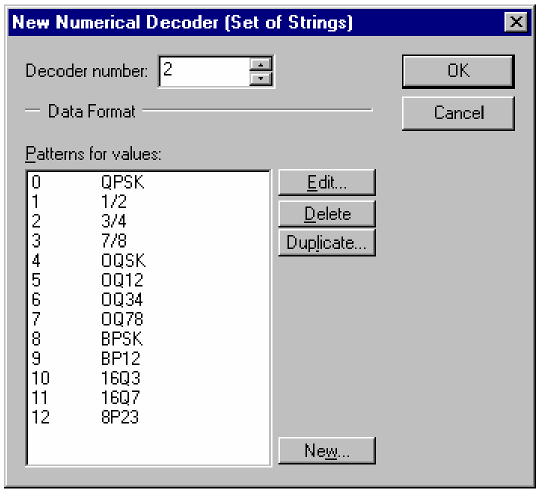
Allow trailing characters:
Check this box to allow characters that are not part of the number to appear after it in the data string. This only applies to characters within the fixed width. All characters are always allowed to appear beyond the fixed width. Specify the characters that are allowed to appear after the number using the radio buttons.

This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

Numerical Decoders (Set Of Strings)

Set of strings decoders get a number from the data string using a set of regular expressions. See *Appendix A: Regular Expressions* on page 171 for details on regular expressions.

The New Numerical Decoder (Set Of Strings) Dialog



Decoder number: 2

0	QPSK
1	1/2
2	3/4
3	7/8
4	OQSK
5	OQ12
6	OQ34
7	OQ78
8	BPSK
9	BP12
10	16Q3
11	16Q7
12	8P23

- Decoder number:

Enter the decoder number by which the decoder is identified. Each decoder in an SCL program must have a unique decoder number.

- Patterns for values:

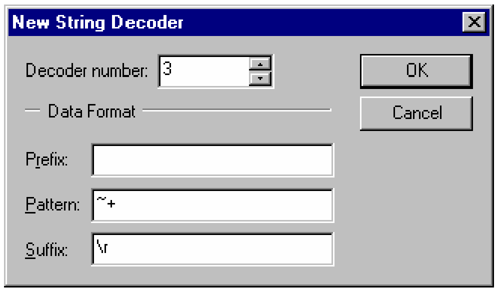
Shows a list of all values that you specified regular expressions for, together with the expression that describes the data string for that value. Use the buttons to edit, delete, duplicate, and add values.

String Decoders

String decoders get a string value from a data string. String decoders use regular expression patterns to recognize the string value. The data is divided into three sections: A prefix, the value, and a suffix. The prefix and suffix are discarded, and only the value is used.

See *Appendix A: Regular Expressions* on page 171 for details on regular expressions.

The New String Decoder Dialog



Decoder number: 3

Prefix:

Pattern: ~+

Suffix: \r

- Decoder number:

Enter the decoder number by which the decoder is identified. Each decoder in an SCL program must have a unique decoder number.

- Prefix:

Enter a regular expression that describes any data that appears before the parameter in the response. Any data that matches the prefix will be discarded.

- Pattern:

Enter a regular expression that describes the value here.

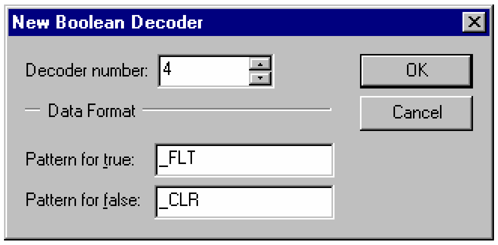
- Suffix:

Enter a regular expression that describes any data that appears after the parameter in the response. Any data that matches the suffix will be discarded.

Boolean Decoders

Boolean decoders get a Boolean value from a data string. A Boolean value can be either true or false.

The New Boolean Decoder Dialog



Decoder number: 4

Pattern for true: _FLT

Pattern for false: _CLR

- Decoder number:

Enter the decoder number by which the decoder is identified. Each decoder in an SCL program must have a unique decoder number.

- Pattern for true:

Enter a regular expression for the true value. The result will be true if this expression is found at the start of the data.

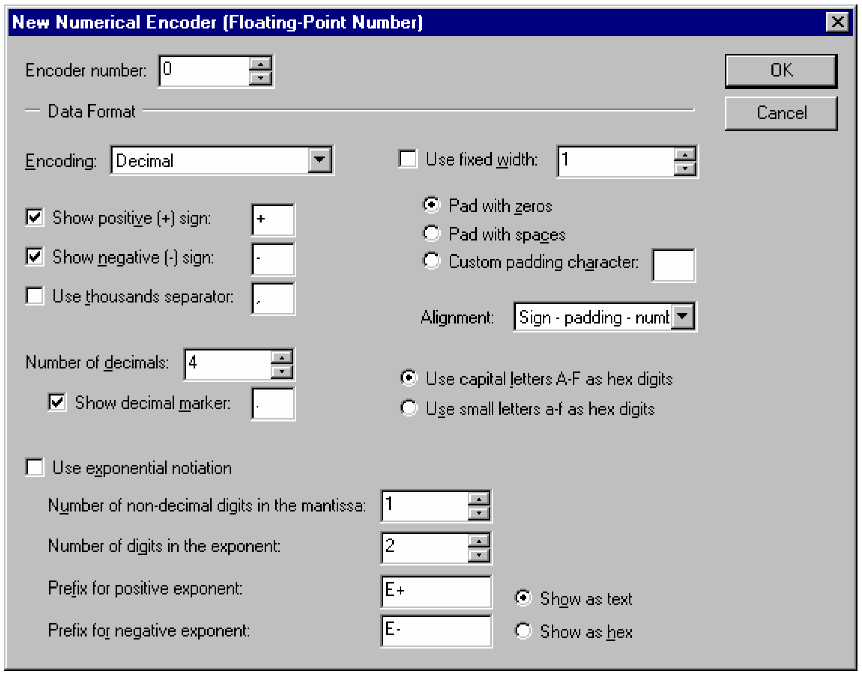
- Pattern for false:

Enter a regular expression for the false value. The result will be false if this expression is found at the start of the data.

Numerical Encoders (Floating-Point Number)

Floating-point number encoders write a numerical value with virtually unlimited range to a data string. Floating-point number encoders support decimal fractions as well as exponential (scientific) notation.

The New Numerical Encoder (Floating-Point Number) Dialog


New Numerical Encoder (Floating-Point Number)

Encoder number: 0

Data Format

Encoding: Decimal

[✓] Show positive (+) sign: +

[✓] Show negative (-) sign: -

[ ] Use thousands separator: ,

Number of decimals: 4

[✓] Show decimal marker: .

[ ] Use fixed width: 1

(●) Pad with zeros

( ) Pad with spaces

( ) Custom padding character:

Alignment: Sign - padding - numt

(●) Use capital letters A-F as hex digits

( ) Use small letters a-f as hex digits

[ ] Use exponential notiation

Number of non-decimal digits in the mantissa: 1

Number of digits in the exponent: 2

Prefix for positive exponent: E+

Prefix for negative exponent: E-

(●) Show as text

( ) Show as hex

- Encoder number:

Enter the encoder number by which the encoder is identified. Each encoder in an SCL program must have a unique encoder number.

- Encoding:

Select the encoding method for the number. U.P.M.A.C.S. supports the following encoding methods:

Byte:
A single byte (character) in the command string is used to represent the number as an 8-bit value. If you select signed byte encoding, the most significant bit of the byte will be used to denote the sign in the standard way (-1 is $FF, -2 is $FE, etc.). If you select unsigned byte encoding, the sign of negative numbers will be ignored, i.e. -20 will be encoded the same as 20. The byte encoding method requires a fixed width of 1.

Multibyte:
Two or more bytes (characters) in the command string are used to represent the number as a 16, 24, or 32-bit value. You can choose between lo-hi and hi-lo byte ordering. If you select lo-hi byte ordering, the least significant byte (the lower 8 bits) will be sent first; if you select hi-lo byte ordering, the most significant byte (the upper 8 bits) will be sent first. If you select signed multibyte encoding, the most significant bit of the most significant byte will be used to denote the sign in the standard way (for 16-bit values -1 is $FFFF, -2 is $FFFE, etc.). If you select unsigned multibyte encoding, the sign of negative numbers will be ignored, i.e. -20 will be encoded the same as 20. The multibyte encoding method requires a fixed width of 2, 3, or 4.

BCD:
The number is encoded using Binary Coded Decimal encoding. In BCD encoding, each nibble (hex digit) in a byte represents one decimal digit. The number 20,841,057 would be encoded as the byte (character) values hex 20 (32), hex 84 (132), hex 10 (16), and hex 57 (87). The BCD encoding method requires a fixed width.

Decimal, Hexadecimal, Binary, Octal:
The number is written out as a decimal, hexadecimal, binary, or octal number using ASCII characters.

Do not confuse the byte/multibyte and binary encoding methods. The binary encoding writes the number out as a series of ASCII characters 1 (hex 31) and 0 (hex 30), it does not encode it using the individual bits of each byte.

- Show positive (+) sign:

Check this check box if you want a plus sign to be shown for positive values. The plus sign does not actually have to be the "+" character. You can specify any character you want.

This setting only applies to the decimal encoding.

- Show negative (-) sign:

Check this check box if you want a minus sign to be shown for negative values. The minus sign does not actually have to be the "-" character. You can specify any character you want.

This setting only applies to the decimal encoding.

- Use thousands separator:

Check this check box if you want digits to be separated into groups of three using a thousands separators (usually a comma). If you check this box, you must also specify the actual character used to group digits.

This setting only applies to the decimal encoding.

- Number of decimals:

For the decimal encoding, this field allows you to specify the number of digits shown after the decimal point. If you do not want decimal digits or a decimal point, enter 0 here.

For all other encodings, this field allows you to specify the number of implied decimals. Implied decimals are a way of encoding fractions without using a decimal point by multiplying it with a power of ten before encoding it. A number that is encoded using 3 implied decimals, for example, will be multiplied by 1000 before it is encoded. This will move the three decimals from the right to the left of the decimal point, and the decimal point will no longer be needed. 21.304, for example, will simply be encoded as 21304.

You can use implied decimals with the decimal encoding method as well. Simply set the required number of decimals in the "Number of decimals" field, and uncheck the "Show decimal marker" check box described below. No decimal point will then be shown.

Please note that the implied decimals for the hexadecimal, octal, and binary encoding methods are decimal fractions, not hexadecimal, octal or binary fractions. In hexadecimal encoding with three implied decimals, 43.249 (which is~2B.3FC in hex) will be written as A8F1 (which is 43249 in decimal) rather than 2B3FC. In other words, the number is always multiplied by powers of 10, even if the encoding method uses base 16, 2 , or 8.

Show decimal marker:
Clear this check box if you do not want a decimal point to be shown (implied decimals). If you leave this box checked, you can specify the character to be used as a decimal marker. The decimal marker does not have to be a period, you can use any character you like. This setting only applies to the decimal encoding.

- Use fixed width:

Check this check box if you want the resulting string to be a fixed number of bytes (characters) long. Specify the number of bytes in the edit box.

You must check this box for the byte, multibyte, and BCD encoding methods.

Pad with zeros / Pad with spaces / Custom padding character:
Specify the character that is to be used to pad the number to the fixed width if it is too short. Only applies to the decimal, hexadecimal, binary, and octal encodings.

Alignment:
Choose the alignment of the data within the fixed width. You can choose to have any padding characters added before the sign, between the sign and the number, or after the number. If the sign is not shown, the Sign-padding-number and Padding-sign-number alignments are the same. This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

- Use capital letters A-F / small letters a-f as hex digits:

Select whether capital or small letters should be used for hex digits. Only applies to the hexadecimal encoding.

- Use exponential notation:

Check this box to use exponential (scientific) notation with the decimal encoding method.

Number of non-decimal digits in the mantissa:
Enter the number of digits that should appear on the left of the decimal point of the mantissa (the part of the number that is not the exponent). Note that this setting does not necessarily reflect the number of digits that actually appear before the decimal point. It is merely used to determine the range of the mantissa. If you specify a value of 0, the mantissa will always be between 0 and 1. If you specify 1, it will be either 0, or between 1 and 10, etc.. The number 13895.468, for example, will be shown as 0.13895468E+5 with 0 mantissa digits, as 1.38954680E+4 with one digit, and as 138.95468000E+2 with three digits. The number 0 will always be shown as 0.00000000E+0, regardless of the number of mantissa digits specified. If you want to force 0 to be shown as 000.00000000E+0, you must specify a fixed width, and "0" as the padding character.

Number of digits in the exponent:
Enter the number of digits shown in the exponent. Unlike the number of digits in the mantissa digits, the exponent will always have exactly the number of digits specified here.

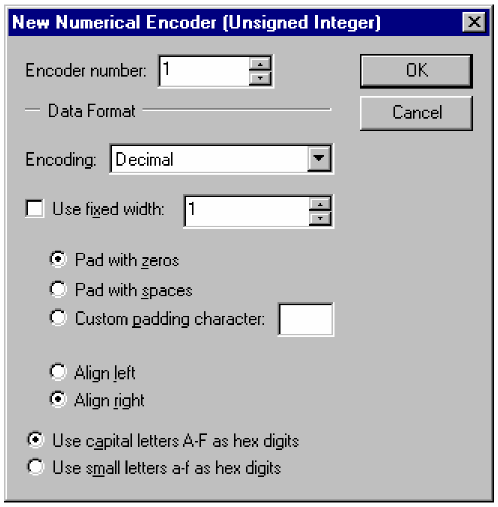
Prefix for positive / negative exponent:
Enter the exponent markers for positive and negative exponents here. The exponent markers must include the sign of the exponent. Usually, the positive exponent marker is "E+" or "e+", and the negative marker is "E-" or "e-". If you want to omit the plus sign for positive exponents, specify "E" or "e" for the positive exponent prefix. The exponent prefixes can be any arbitrary data. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

Show as text / Show as hex:
Select the way you want to enter the exponent prefixes. See *Appendix B: Entering Binary Data* on page 179 for details.

Numerical Encoders (Unsigned Integer)

Unsigned integer encoders write an integer value between 0 and 4,294,967,295 (hex FFFFFFFF) to a data string. The functionality of unsigned integer encoders is a sub-set of that of floating-point number encoders. Use a floating-point number encoder for values that can be less or greater than the range of an unsigned integer, for values that contain fractions, or if you want to use exponential (scientific) notation.

The New Numerical Encoder (Unsigned Integer) Dialog



Encoder number: 1

Encoding: Decimal

☐ Use fixed width: 1

(●) Pad with zeros
( ) Pad with spaces
( ) Custom padding character:

( ) Align left
(●) Align right

(●) Use capital letters A-F as hex digits
( ) Use small letters a-f as hex digits

- Encoder number:

Enter the encoder number by which the encoder is identified. Each encoder in an SCL program must have a unique encoder number.

- Encoding:

Select the encoding method for the number. U.P.M.A.C.S. supports the following encoding methods:

Byte:
A single byte (character) in the command string is used to represent the number as an 8-bit value. Requires a fixed width of 1.

Multibyte:
Two or more bytes (characters) in the command string are used to represent the number as a 16, 24, or 32-bit value. You can choose between lo-hi and hi-lo byte ordering. If you select lo-hi byte ordering, the least significant byte (the lower 8 bits) will be sent first; if you select hi-lo byte ordering, the most significant byte (the upper 8 bits) will be sent first. The multibyte encoding method requires a fixed width of 2, 3, or 4.

BCD:
The number is encoded using Binary Coded Decimal encoding. In BCD encoding, each nibble (hex digit) in a byte represents one decimal digit. The number 20,841,057 would be encoded as the byte (character) values hex 20 (32), hex 84 (132), hex 10 (16), and hex 57 (87). The BCD encoding method requires a fixed width.

Decimal, Hexadecimal, Binary, Octal:
The number is written out as a decimal, hexadecimal, binary, or octal number using ASCII characters.

Do not confuse the byte/multibyte and binary encoding methods. The binary encoding writes the number out as a series of ASCII characters 1 (hex 31) and 0 (hex 30), it does not encode it using the individual bits of each byte.

- Use fixed width:

Check this check box if you want the resulting string to be a fixed number of bytes (characters) long. Specify the number of bytes in the edit box.

You must check this box for the byte, multibyte, and BCD encoding methods.

Pad with zeros / Pad with spaces / Custom padding character:
Specify the character that is to be used to pad the number to the fixed width if it is too short. This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

Align left / Align right:
Select whether the number should be left or right aligned within the fixed width. If the number is left aligned, any padding characters will be added to the end of the number; if it is right aligned, they will be added to the beginning. This setting only applies to the decimal, hexadecimal, binary, and octal encodings.

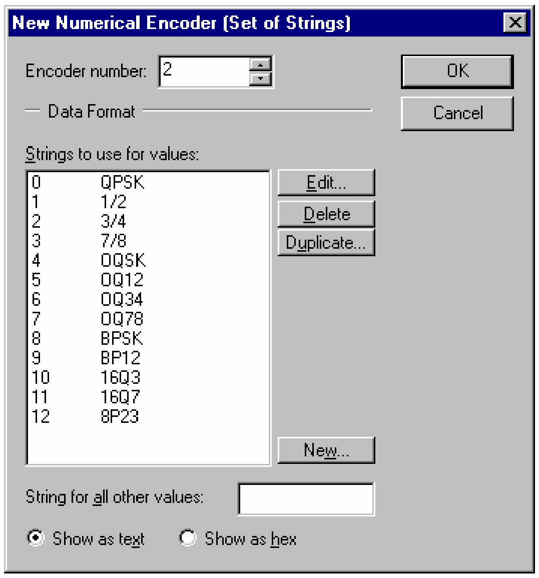
- Use capital letters A-F / small letters a-f as hex digits:

Select whether capital or small letters should be used for hex digits. This setting only applies to the hexadecimal encoding.

Numerical Encoders (Set Of Strings)

Set of strings encoders encode a number using a set of pre-defined data string. You explicitly specify the data to be used for the different values.

The New Numerical Encoder (Set Of Strings) Dialog



Encoder number: 2

Strings to use for values:

0	QPSK
1	1/2
2	3/4
3	7/8
4	OQSK
5	OQ12
6	OQ34
7	OQ78
8	BPSK
9	BP12
10	16Q3
11	16Q7
12	8P23

String for all other values:

[✓ Show as text] Show as hex

- Encoder number:

Enter the encoder number by which the encoder is identified. Each encoder in an SCL program must have a unique encoder number.

- Strings to use for values:

Shows a list of all values that you specified data strings for, together with their strings. Use the buttons to edit, delete, duplicate, and add values.

- String for other values:

Enter the data to use for all values that do not appear in the list above. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

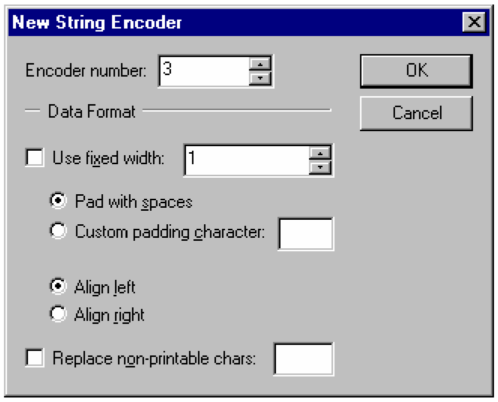
- Show as text / Show as hex:

Select the way you want to enter the data strings. See *Appendix B: Entering Binary Data* on page 179 for details.

String Encoders

String encoders write a string value to a data string.

The New String Encoder Dialog



Encoder number: 3

☐ Use fixed width: 1

● Pad with spaces

○ Custom padding character:

● Align left

○ Align right

☐ Replace non-printable chars:

- Encoder number:

Enter the encoder number by which the encoder is identified. Each encoder in an SCL program must have a unique encoder number.

- Use fixed width:

Check this check box if you want the resulting string to be a fixed number of bytes (characters) long. Specify the number of bytes in the edit box.

Pad with spaces / Custom padding character:
Specify the character that is to be used to pad the string to the fixed width if it is too short.

Align left / Align right:
Select whether the string should be left or right aligned within the fixed width. If the number is left aligned, any padding characters will be added to the end of the string, if it is right aligned, they will be added to the beginning.

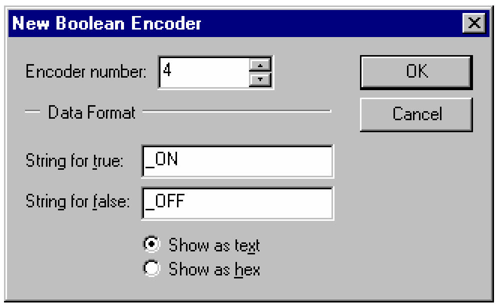
- Replace non-printable chars:

Check this check box if you want all non-printable characters (characters other than ASCII $20-ASCII $7E) in the string replaced. Enter the character that you would like to replace them with in the box provided.

Boolean Encoders

Boolean encoders write a Boolean value to a data string. A Boolean value can be either true or false.

The New Boolean Encoder Dialog



Encoder number: 4

String for true: _ON

String for false: _OFF

(● Show as text) ○ Show as hex

- Encoder number:

Enter the encoder number by which the encoder is identified. Each encoder in an SCL program must have a unique encoder number.

- String for true:

Enter the data to be sent if the value is ON. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- String for false:

Enter the data to be sent if the value is OFF. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Show as text / Show as hex:

Select the way you want to enter the data strings. See *Appendix B: Entering Binary Data* on page 179 for details.

Screens

U.P.M.A.C.S. organizes its visual data in screens. A screen contains information about everything displayed in a single window or dialog. A screen contains graphic objects, which determine what is shown in the window.

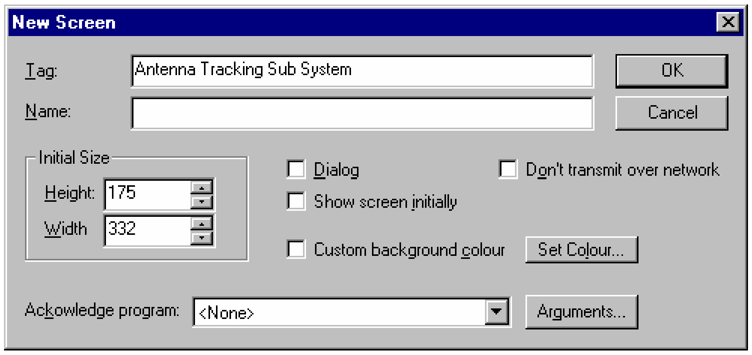
A screen can be displayed in a regular window, or in a dialog window. Regular windows can be sized, moved, tiled, minimized, maximized, and otherwise manipulated, whereas dialogs can only be moved, and not sized.

Once a station file has been loaded, a number of windows are opened showing certain screens. You can specify for each screen whether it should be shown initially in this fashion or not.

Once you have created a screen, a Screen Editor window will appear so that you can add graphical objects.

If you select a screen in the Screens window, and press the “Edit…” button, the editor window will appear rather than the Edit Screen dialog. To change the properties of a screen, select it and press the “Properties…” button. You can also select “Properties…” from the “Edit” menu when viewing the screen in an editor window.

The New Screen Dialog



Tag: Antenna Tracking Sub System
Name:
Height: 175
Width: 332
Ackowledge program: <None>

- Tag:

Enter the tag by which the screen is identified. Each screen must have a unique tag.

- Name:

Enter the name of the screen. Leave this field blank if you want to use the tag as name. The name of the screen appears in the window's title bar, as well as in the New Window dialog of the Operate System.

- Initial size:

Enter the height and width of the screen, in pixels. You can also change the size of the screen graphically within the Screen Editor.

- Dialog:

Check this box if you want the screen to appear in a dialog rather than a regular window.

- Show screen initially:

Check this box if you want a window containing this screen to appear as soon as the station is loaded. This option is not available for dialog screens.

- Don't transmit over network:

Check this box to prevent U.P.M.A.C.S. from transmitting this screen across the network when someone connects to the station. This means that the screen will only appear on the local computer, not on any remote computers. Use this flag for screens that are used as a network map, for example.

- Custom background colour:

Check this box to set a background colour for the screen. If you leave this check box blank, U.P.M.A.C.S. uses the background colour specified in the Display control panel. Press the "Colour…" button to set the colour.

- Acknowledge program:

Select an SCL program to execute when the user presses the Acknowledge button. The program arguments are shown in parentheses after the name of the program, but you only select the program from the lists, not the arguments. To change the arguments, use the "Arguments…" button. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

The Screen Editor

Screen Editor windows are used to edit the graphical objects contained in a screen. The window will automatically appear when you create a screen. To open an editor window for a screen once you have closed it, select the program in the Screens window, and press the "Edit…" button, or double-click on the screen's name.

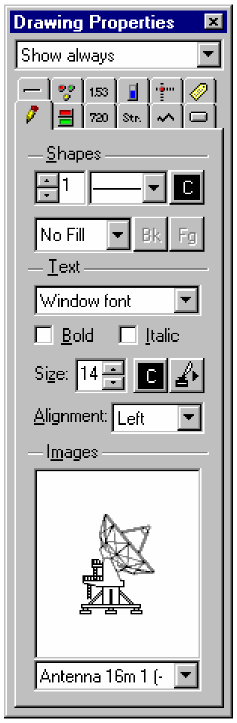
Creating Graphic Objects

Everything displayed on a screen is a graphic object. There are eight types of objects:

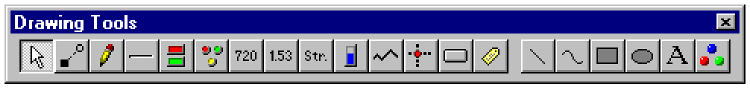
- static objects
- 3D objects
- bistate indicators
- multistate indicators
- digital indicators
- analog indicators
- string indicators
- dials
- graphs
- X-Y position markers
- controls (buttons)
- labels

Each object type is described in its own section.

To create a specific type of object, you must first set the drawing properties for the type of object you wish to draw. Make sure that no graphic objects are selected by clicking on a blank space on the screen before changing the properties. If any objects are selected, you will change the properties of those objects instead of the current drawing properties.



After setting the properties, choose the correct drawing tool from the "Draw" menu, or by pressing one of the Drawing Tool buttons.



To draw static objects, bistate indicators, or multistate indicators, you must also select a secondary drawing tool. The secondary drawing tool determines the shape of the object or indicator: line, rectangle, ellipse, etc.

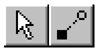
Once you have selected the tool or tools, click and drag in the editor window to create the graphic object.

There is a special shape known as a Bezier spline. A Bezier spline is a curved line, and has two end points and two "helper" points that define the curvature of the line. Static objects, as well as bistate and multistate indicators can have Bezier spline shape. To draw a Bezier spline, move the mouse pointer to the starting point of the curve, and click and drag to indicate the direction of the curved line at that point. Let go of the mouse button, and move the mouse pointer to the end point of the curve. Click and drag again to indicate the direction of the curve at the end point.

If you are drawing static text, click and drag to define the bounding box of the text. Once you release the mouse button, a dialog will pop up, asking you to enter the text. Enter the text you wish to display and press OK.

Selecting, Moving, and Resizing Objects

To move or resize objects, or to change their properties, you must select them. To select objects, and to move or resize them, you must choose the "Select" or "Reshape" drawing tools. Choose "Select" or "Reshape" from the "Draw" menu, or press the "Select" or "Reshape" Drawing Tool buttons:


The "Select" and "Reshape" Drawing Tool buttons

To select an object, click on it with the mouse. To select a group of objects, click anywhere where there is no graphic object, and drag to create a selection rectangle. When you release the mouse button, everything completely enclosed in the rectangle will be selected.

Once an object has been selected, the editor draws its edit handles. Edit handles are small squares of inverted screen surface. On a white screen, the edit handles appear as black squares. You can drag the object to move it, or drag an edit handle to resize the object. You can move the selected objects one pixel at a time using the cursor (arrow) keys.

Some objects (Bezier splines and some indicators) have special edit handles called reshape handles. These handles are hollow circles or triangles, rather than solid squares. Reshape handles are used to change some additional geometry of the objects, and are only visible if you have chosen the Reshape drawing tool.

You can add and remove single objects to/from the selection by clicking on them while holding down the Shift key. You can add and remove a group of objects by dragging a selection rectangle while holding down the Shift key. To select all objects, choose "Select All" from the "Edit" menu.

To deselect all objects, click on any blank space on the screen.

Locking the Graphics Tool

Once you have created an object, the drawing tool automatically changes to the Select tool or the Reshape tool to allow you to make additional changes to the object. If you want to create a series of objects without having to re-select the drawing tool every time, you can lock the tool. To lock a drawing tool, select "Lock Tool" from the "Draw" menu, or click on the Drawing Tool button a second time. A small picture of a padlock will appear on the button to indicate that the tool is locked. If the tool is locked, it will not revert to the Select or Reshape tool once you have created an object.

The Graphics Grid

The screen editor has a graphics grid to help you space objects evenly. The graphics grid consists of regularly spaced points, which may be invisible, or can be shown as dots. You can set a snap-to-grid option, causing all objects to be created, moved, and resized, to grid points.

To show or hide the grid points, select "Grid" from the "View" menu. To change the spacing of the grid, choose "Grid Size…" from the "View" menu. To move the grid points so that one is aligned with the upper left-hand corner of the screen, choose "Align Grid With Screen" from the "View" menu. To move the grid points so that one is aligned with the current selection, choose "Align Grid With Selection" from the "View" menu.

To switch grid snap on or off, select "Snap To Grid" from the "Draw" menu. While grid snap is on, all objects are created, moved, and resized to the nearest grid point.

Editing Object Properties

If you have selected one or more objects, the Drawing Properties window will show the properties of the selected objects. You can then change the properties of the selected objects in the Drawing Properties.

Some objects have text, and you can modify the text after you have created the object. For some types of objects, like controls and labels, the text is part of the drawing properties and appears in the Drawing Properties window. For those types of objects, you can edit the text in the Drawing Properties window as you would any other property. Some objects, however, have text that does not appear in the drawing properties. For those objects, you can edit the text one of two ways:

There are buttons in the Drawing Properties window that have a picture of a quill in an ink well on it. Press that button to edit the text. This will change the text for all selected objects of the same type.

If you double-click on an object that has text on it, a window will pop up that lets you edit the text. This will only affect one object at a time, the one you double-clicked on. No other objects are affected, even if they are selected.

Using Properties from Existing Objects to Create New Objects

You can enter the properties of the currently selected objects into the drawing properties used to create new objects by choosing "Get Properties From Selection" from the "Draw" menu. To draw an object that has the same drawing properties as an existing object, proceed as follows:

- Select the objects whose properties you want to use.
- Choose "Get Properties From Selection" from the "Draw" menu. This will change the current drawing properties to match the selected object's properties.
- Choose the drawing tool or tools for the object you wish to draw.
- Draw the object.
- Deleting and Copying Objects
- To delete all selected objects, press the Delete key.

You can cut and paste objects between screens. To cut or copy objects, select them, and then choose "Cut" or "Copy" from the "Edit" menu. To paste objects, Choose "Paste" from the "Edit" menu. Pasting objects will replace all currently selected objects. You can cut and paste from one screen to another, or within the same screen, but you cannot cut and paste objects from one station file to another. There is currently no support for copying object between station files.

You can duplicate objects by selecting them and choosing "Duplicate" from the "Edit" menu. The new objects will be slightly offset from the original ones. Another easy way to duplicate objects is to select them, and to click on them, drag them, or resize them while holding down the Ctrl key.

To make multiple, evenly spaced copies of an object, select it, and choose "Duplicate" or Ctrl-drag it to create the first duplicate. Move the duplicate to its new position, and then repeatedly choose "Duplicate" from the "Edit" menu (or use the Ctrl-D shortcut key). The new duplicates of the object will be spaced the same as the original and the first duplicate.

Modifying the Screen Area

You can resize or move the screen area by clicking and dragging the window outline in the screen editor. You can grab either edge or corner of the window to size it, or grab the hatched title bar area to move it.

Controlling the Way Indicators Are Displayed

Since the Development System does not actually poll equipment or process any data, the registers in the Development System do not have any values. This means that the indicators shown in the Screen Editor cannot reflect the actual values of equipment. You can control the way indicators are displayed, and the values they show.

- State shown by indicators:

To specify the state in which the indicators are to be displayed in the Screen Editor, use the indicator state pop-up menu on the appropriate page of the Drawing Properties window. The indicator state pop-up will show one of the following values:

- Off state
- On state
- Blink state
- Masked state
- Error state

Select the state in which indicators of the type corresponding to the page should be shown.

- Value shown by multistate indicators:

The state pop-up menu on the multistate indicator page does not have the On or Off state selection. Instead, the menu has options for different values, e.g. Value 0, Value 1, Value 25, etc. If you select one of these values, all multistate indicators will show that value.

- Value shown by digital, analog, and string indicators:

To specify the value that should be displayed in digital, analog, and string indicators, select "Indicator Values…" from the "View" menu. A dialog will pop up that allows you to enter values for each type of indicator.

- Value shown by dials, graphs, and X-Y position markers:

Dials, graphs, and X-Y position markers always show the same values. You cannot change the values shown by these types of indicators.

Manipulating Graphic Objects

Front-To-Back Ordering Of Objects

The objects on a screen have a front-to-back ordering. This means that some objects are "in front of" others, and cover all or part of them. When you create an object other than a control, the object is created as the frontmost object, and covers all other objects it overlaps. Controls are always created as the backmost object, so they don't cover any objects.

You can change the front to back ordering by using the following menu items of the "Arrange" menu:

- Raise:
Brings the selected objects one object further forward. You may not immediately see a change, as the object you are raising may not overlap the object in front of which it moved. If nothing appears to happen, repeatedly raise the object until you get the desired result.
- Lower:
Sends the selected objects one object further back. You may not immediately see a change, as the object you are lowering may not overlap the object behind which it moved. If nothing appears to happen, repeatedly lower the object until you get the desired result.
- Bring To Front:
Brings the selected objects all the way to the front, to hide all other objects that they overlap.
- Send To Back:
Sends the selected objects all the way to the back, to hide none of the objects that they overlap.

Grouping Objects

You can group a number of objects together to act as one object when moving and resizing. To group objects, select them, and choose "Group" from the "Arrange" menu. You can include a group of objects previously grouped in a group. To change the group back to its component objects, select "Ungroup" from the "Arrange" menu.

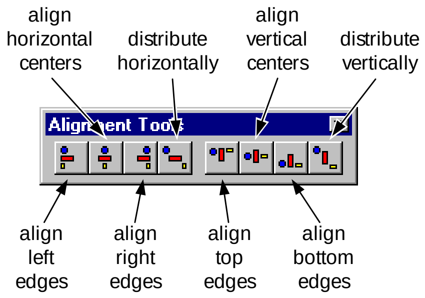
Flipping/Rotating Objects

You can flip (mirror) and rotate objects by using menu items in the "Arrange" menu. Please note that all selected objects are flipped and rotated individually about their own centers. To flip or rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be flipped or rotated.

Aligning Objects

You can align the current selection to the nearest grid point by choosing "Align With Grid" from the "Arrange" menu. All selected objects are aligned together as a group; their position relative to each other is not changed.

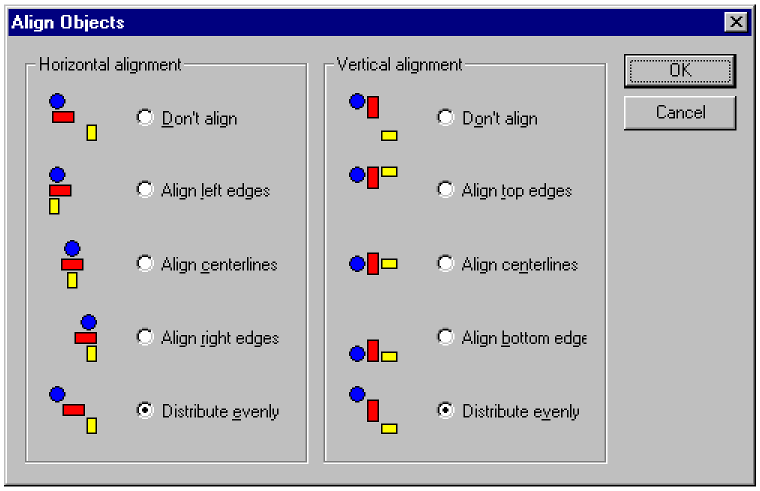
You can also align objects with each other, either by choosing "Align Objects…" from the "Arrange" menu or by pressing one of the Alignment Tool buttons.



The Alignment Tools window has eight buttons:

Align left edges:	Aligns the left edge of all objects with the leftmost object
Align horizontal centers:	Aligns the horizontal centers of all objects at the center of the selection
Align right edges:	Aligns the right edge of all objects with the rightmost object
Distribute horizontally:	Spaces the objects evenly along the horizontal axis
Align top edges:	Aligns the top edge of all objects with the topmost object
Align vertical centers:	Aligns the horizontal centers of all objects at the center of the selection
Align bottom edges:	Aligns the bottom edge of all objects with the bottommost object
Distribute vertically:	Spaces the objects evenly along the vertical axis

If you choose "Align Objects…" from the "Arrange" menu, the Align Objects dialog will appear.


Align Objects

Horizontal alignment
- ( ) Don't align
- ( ) Align left edges
- ( ) Align centerlines
- ( ) Align right edges
- (•) Distribute evenly

Vertical alignment
- ( ) Don't align
- ( ) Align top edges
- ( ) Align centerlines
- ( ) Align bottom edge
- (•) Distribute evenly

▪ Horizontal alignment:
Choose from one of the following alignment options for the horizontal axis:

Don't Align:	Don't change the horizontal position of the objects
Align left edges:	Aligns the left edge of all objects with the leftmost object
Align centerlines:	Aligns the horizontal centers of all objects at the center of the selection
Align right edges:	Aligns the right edge of all objects with the rightmost object
Distribute evenly:	Spaces the objects evenly along the horizontal axis

▪ Vertical alignment:
Choose from one of the following alignment options for the vertical axis:

Don't Align:	Don't change the vertical position of the objects
Align top edges:	Aligns the top edge of all objects with the topmost object

Align centerlines:	Aligns the vertical centers of all objects at the center of the selection
Align bottom edges:	Aligns the bottom edge of all objects with the bottommost object
Distribute evenly:	Spaces the objects evenly along the vertical axis

Images

Besides lines, Bezier splines, rectangles, ellipses, and text, you can display bitmap images in U.P.M.A.C.S. These bitmap images are loaded from an U.P.M.A.C.S. Image Library (*.upmacs-images file) created using the U.P.M.A.C.S. Image Editor.

To load an image library, select "Load Images…" from the "File" menu. You can then use images from that image library in your station. When you start the Development System, the last image library you used will be reloaded automatically.

When you save you station, the images you used (but not the ones you didn't use) are saved in the station file. When you load a station file, you can use the images in that station file as well as those in the image library. If, however, you load an image library after you loaded a station, the images in the station will be overwritten with new images from the image library. Images that are not in the image library will be replaced by the library's replacement image. You should therefore make sure that an image library contains all the necessary images for the current station file before loading it.

Since the images that are needed for the station are saved in the station file, it is not necessary to install the image library on the computer where the station will be used. Image libraries are only used by the Development System, not by the Operate System.

Note: Station files created with versions of U.P.M.A.C.S. prior to v6.0 do not contain the necessary images. If you want to edit an older station file, you must have the corresponding image library as well.

Static Objects

Static objects never change the way they look. Everything on a screen that does not depend on states or names of registers, is a static object. Static objects can have one of the following six shapes:

Lines:	Straight lines
Splines:	Curved lines, using two end points and two "helper" points
Rectangles:	Rectangles, hollow or filled
Ellipses:	Ellipses, hollow or filled. This includes circles.
Text:	Text
Images:	Images from the image library. See *Images* on page 88 for details on image libraries.

There are some special static objects called 3D objects because they have a raised or sunken look. These objects are discussed in *3D Objects* on page 91.

- Drawing static objects:

To draw a static object, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.

2. Go to the Object Properties page in the Drawing Properties window and set the properties of the new object. The tab for the Object Properties page is marked with the same icon as the "Objects" Drawing Tool button (see picture below).
3. Choose "Objects" from the "Draw" menu, or press the "Objects" Drawing Tool button:

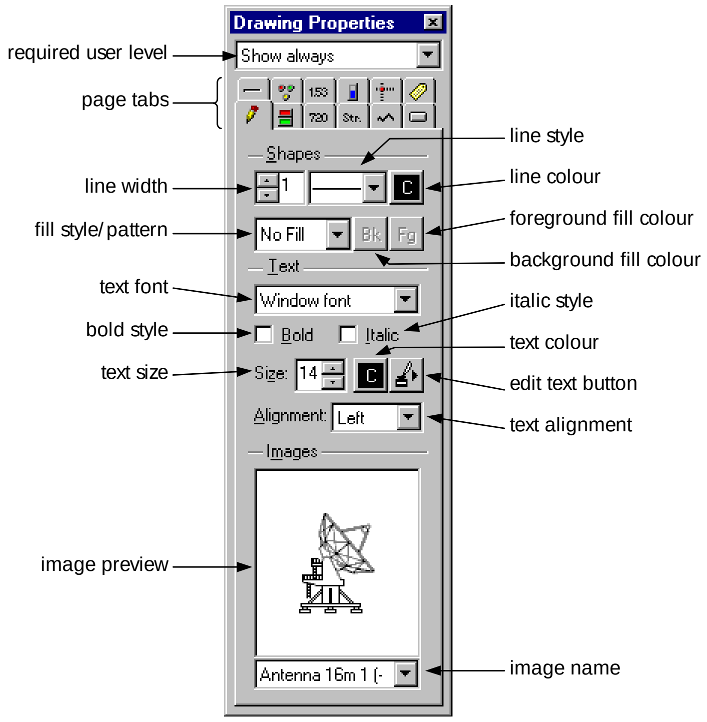
The "Objects" Drawing Tool button

If you want to draw more than one object, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.

4. Choose the secondary tool that corresponds to the shape (see above) that you want to draw.
5. Draw the object by clicking and dragging on the screen (Splines require two click-and-drag operations to be drawn).

If you selected the text shape, a dialog will appear. Enter the text in the edit field of the dialog and press OK.

The Object Properties Page



Show always
1
No Fill
Window font
Size: 14
Alignment: Left
Antenna 16m 1 (-

If no graphic object is currently selected, the Object Properties page shows the current drawing properties for static objects. If any graphic objects are selected, the page shows all the properties applicable to the selected static objects. Properties that are not applicable to any objects (such as line width for text objects) appear grayed. If a particular property is different for two selected objects (e.g. two lines with different line widths are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Line width:

Select the line width for lines and splines, and for the outline of rectangles and ellipses.

- Line style:

Select the line style (solid, dashed, dotted, etc.) for lines and splines, and for the outline of rectangles and ellipses.

- Line colour:

Select the colour for lines and splines, and for the outline of rectangles and ellipses.

- Fill style/pattern:

Select the fill style or fill pattern for rectangles and ellipses.

- Background fill colour:

If you selected a two-colour pattern as the fill style, select the background colour for the pattern here.

- Foreground fill colour:

If you selected “Solid colour” as the fill style, select the fill colour here. If you selected a two-colour pattern as the fill style, select the foreground colour for the pattern here.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here. If no graphic objects are selected, this button is wider and reads “Colour…” instead of just “C”.

- Edit text button:

This button is only visible if any static objects are selected. Press this button to edit the text of text objects.

- Alignment:

Select the alignment (justification) for text here.

- Image name:

Select the image for image objects here. The image will appear in the image preview field.

- Image preview:

This field shows the image you have selected in the image name field.

3D Objects

3D objects are special types of static object. Like static objects, they never change the way they look. 3D objects are objects whose colour depends on the colours specified in the *Display* control panel. They usually have a raised, sunken, or chiselled appearance. 3D objects come in the following shapes:

- Horizontal dividers
- Vertical dividers
- Group boxes
- Raised areas
- Sunken areas
- Recessed areas
- Cutout areas
- Window colour areas
- Dialog colour areas

All of these objects (except window colour areas) use the colours specified for 3D objects in the *Display* control panel.

- Drawing 3D objects:

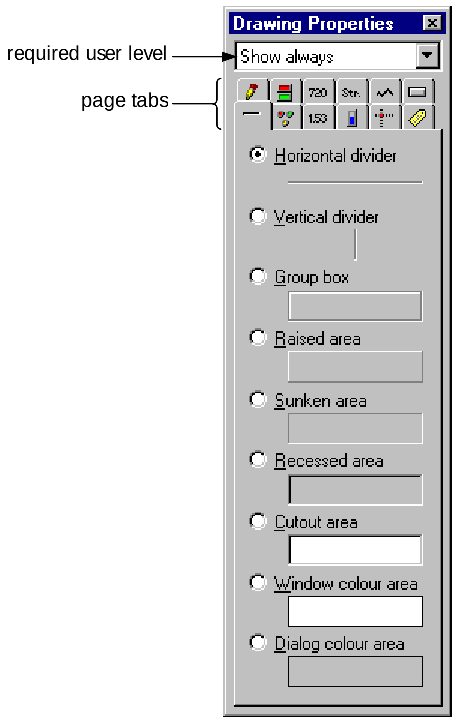
To draw a 3D object, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the 3D Object Properties page in the Drawing Properties window and set the properties of the new object. The tab for the 3D Object Properties page is marked with the same icon as the "3D Objects" Drawing Tool button (see picture below).
3. Choose "3D Objects" from the "Draw" menu, or press the "3D Objects" Drawing Tool button:

   The "3D Objects" Drawing Tool button

   If you want to draw more than one object, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.
4. Draw the object by clicking and dragging on the screen.

The 3D Object Properties Page



(●) Horizontal divider
( ) Vertical divider
( ) Group box
( ) Raised area
( ) Sunken area
( ) Recessed area
( ) Cutout area
( ) Window colour area
( ) Dialog colour area

If no graphic object is currently selected, the 3D Object Properties page shows the current drawing properties for 3D objects. If any graphic objects are selected, the page shows the shape of the selected 3D objects. If no 3D objects are among the selected objects, the shapes appear grayed. If two selected labels have different shapes, none of the shapes will be selected.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Horizontal divider:

Select this option to draw a horizontal divider. Dividers are lines with a chiselled appearance.

- Vertical divider:

Select this option to draw a vertical divider. Dividers are lines with a chiselled appearance.

- Group box:

Select this option to draw a group box. Group boxes are rectangles with a chiselled appearance.

- Raised area:

Select this option to draw a raised area. Raised areas have the same colour as the background of dialogs, and they are shaded to appear raised above the rest of the screen.

- Sunken area:

Select this option to draw a sunken area. Sunken areas have the same colour as the background of dialogs, and they are shaded to appear sunken into the rest of the screen.

- Recessed area:

Select this option to draw a recessed area. Recessed areas have the same colour as the background of dialogs, and they are shaded to appear recessed into the rest of the screen. Recessed areas appear recessed more deeply than sunken areas.

- Cutout area:

Select this option to draw a cutout area. Cutout areas have the same colour as the background of regular windows, and they are shaded to appear recessed back from the rest of the screen.

- Window colour area:

Select this option to draw a window colour area. Window colour areas have the same colour as the background of regular windows.

- Dialog colour area:

Select this option to draw a dialog colour area. Dialog colour areas have the same colour as the background of dialogs.

Indicators

Indicators are graphic objects that change their appearance to represent the state or value of a register.

The State of an Indicator

The state of the register associated with an indicator controls some visual aspects of it, like line width or text colour. This is termed the state of the indicator.

- The different states:

The state of an indicator depends on the ON/OFF state and error and masked state of the register. It can be one of the following:

➢ On state:	The register is in alarm or ON state
➢ Off state:	The register is in alarm clear or OFF state
➢ Error state:	The register contains no data
➢ Masked state:	The register has been masked

There is one additional state for registers with Alarm or Latching alarm levels. If an alarm register is in its alarm state, and has not been acknowledged, its indicators flash to indicate the fault. The flashing is created by alternating the appearance of the indicator between the On state and a special state, the Blink state:

➢ Blink state:	Used to create flashing for unacknowledged alarms

Multistate indicators do not have the On or Off state. Multistate indicators reflect the state and value of a digital register. Each value of the register corresponds to a state of the indicator, and these states replace the On and Off state. If a multistate indicator has an alarm register associated with it, it flashes between the Blink state and the state that corresponds to the register's current value.

Graphs do not have any states at all. They look the same in the On and Off state, and are invisible in the Error and Masked state.

- How states are displayed:

The state of an indicator determines the drawing properties of the graphic object. Drawing properties include colours, line width and style, fill style, font, and text size, style, and alignment. In many cases, the properties also include the text displayed for text type indicators.

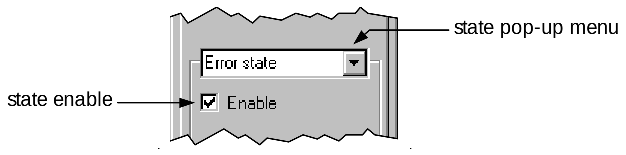
The state of an indicator cannot change its shape or dimensions. You cannot make an indicator that turns from a circle into a square or from a large circle to a small circle when changing states. To implement such functionality, place two indicators on top of each other.

- Enabled and disabled states:

An indicator need not have a set of properties defined for every state. If properties are defined for a certain state, the state is said to be enabled. If no properties are defined, the state is disabled, and the indicator will be invisible if in that state.

- Viewing and modifying states:

Each Indicator Properties page has a rectangle that encloses the properties that vary with the indicator state. At the top of the rectangle is the state pop-up menu, and the state enable check box:



The state pop-up menu and enable check box

All indicators of the type corresponding to a property page are displayed as if in the state selected in the state pop-up menu, and the properties displayed in the rectangle are the properties for that state. To view the properties for a state, or to view the indicator in a certain state, select that state in the pop-up menu.

To enable or disable a state, use the Enable check box. If the Enable check box is not checked, the state is disabled and the indicator will be invisible in that state.

If the Blink state is selected in the pop-up menu, indicators associated with alarms are displayed as flashing on the screen. Indicators associated with non-alarm registers are shown in the On state, and are never displayed as flashing. The properties shown in the rectangle still correspond to the Blink state, but the Blink state is never used for non-alarm indicators.

The Value of an Indicator

Digital indicators, analog indicators, and string indicators show the value of a register as text. The value of the indicator is constructed from the value of the register by prepending a prefix string and appending a suffix string to it. The prefix and suffix are fixed, and do not change with the state or value.

If the register associated with the indicator is in the error or masked state, it contains no value. The value of the indicator is then set to a string that you specify in the drawing properties. The strings for error and masked state are maintained separately and can be different. The prefix string and suffix strings are not affixed to the error and masked strings.

Dials, graphs, and X-Y position markers also display the value of one or more indicators. The values are not displayed as text, however.

Text Used in Indicators

The percentage sign (“%”) character has a special meaning in strings used in indicators. This includes text used in bistate and multistate indicators, and the prefix and suffix strings and error and masked strings of digital, analog, and string indicators, as well as the labels of X-Y position markers.

A percentage sign in indicator text will be replaced by the name of the register the indicator is associated with. If the register is called “Pol B HPA RF” then

```
% switched on
```

will be displayed as:

```
Pol B HPA RF switched on
```

To display a percentage sign (“%”), put two percentage signs (“%%”) in the text:

```
17%% aqueous solution of ethanol
```

will be displayed as:

```
17% aqueous solution of ethanol
```

Bistate Indicators

Bistate indicators change the way they look depending on the ON/OFF state of any type of register.

Bistate indicators can have one of the following six shapes:

Lines:	Straight lines
Splines:	Curved lines, using two end points and two “helper” points
Rectangles:	Rectangles, hollow or filled
Ellipses:	Ellipses, hollow or filled. This includes circles.
Text:	Text
Images:	Images from the image library. See *Images* on page 88 for details on image libraries.

The drawing properties of the shape (line width, image, etc.) as well as the text for the text shape vary according to the state of the register.

- Drawing bistate indicators:

To draw a bistate indicator, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Bistate Indicator Properties page in the Drawing Properties window and set the properties of the new indicator. Use the state pop-up menu to set the properties for the different states. The tab for the Bistate Indicator Properties page is marked with the same

icon as the "Bistate" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Bistate" Drawing Tool.

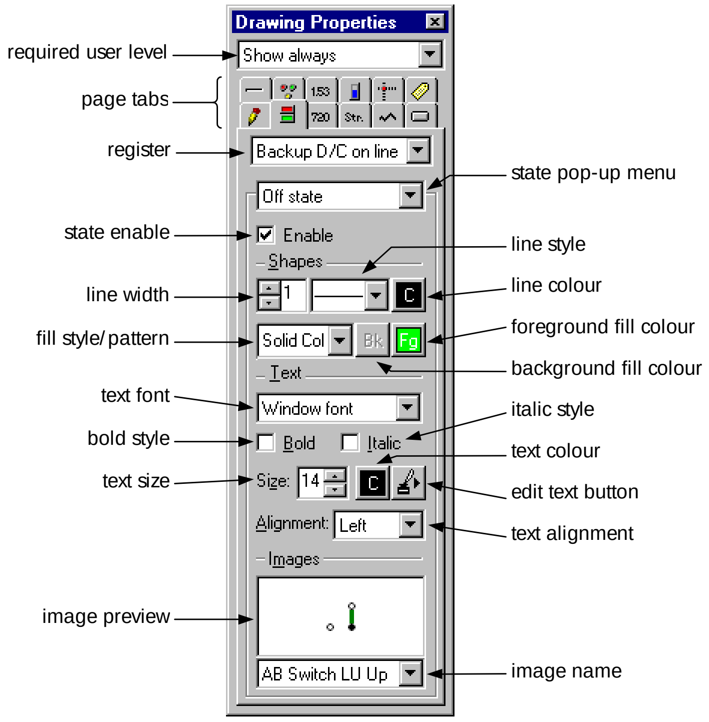
3. Choose "Bistate Indicators" from the "Draw" menu, or press the "Bistate" Drawing Tool button:

The "Bistate" Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one indicator, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.

4. Choose the secondary tool that corresponds to the shape (see above) that you want to draw.
5. Draw the indicator by clicking and dragging on the screen (Splines require two click-and-drag operations to be drawn).

The Bistate Indicator Properties Page



If no graphic object is currently selected, the Bistate Indicator Properties page shows the current drawing properties for bistate indicators. If any graphic objects are selected, the page shows all the properties applicable to the selected bistate indicators. Properties that are not applicable to any objects (such as line width for text shape indicators) appear grayed. If a particular property is different for two selected bistate indicators (e.g. two lines with different line widths are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose state you want to display. If you do not select a register, you will not be able to draw bistate indicators.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All bistate indicators will be displayed in the selected state. If you select “Blink state”, alarm indicators will flash.

- State enable:

Check this checkbox to enable the state. If you leave this box blank, the indicator will be invisible when in this state. A bistate indicator must have at least one state enabled.

- Line width:

Select the line width for lines and splines, and for the outline of rectangles and ellipses.

- Line style:

Select the line style (solid, dashed, dotted, etc.) for lines and splines, and for the outline of rectangles and ellipses.

- Line colour:

Select the colour for lines and splines, and for the outline of rectangles and ellipses.

- Fill style/pattern:

Select the fill style or fill pattern for rectangles and ellipses.

- Background fill colour:

If you selected a two-colour pattern as the fill style, select the background colour for the pattern here.

- Foreground fill colour:

If you selected “Solid colour” as the fill style, select the fill colour here. If you selected a two-colour pattern as the fill style, select the foreground colour for the pattern here.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here.

- Edit text button:

Press this button to edit the text of text shape indicators. The text is converted as described under Text Used in Indicators.

- Alignment:

Select the alignment (justification) for text here.

- Image name:

Select the image for image objects here. The image will appear in the image preview field.

- Image preview:

This field shows the image you have selected in the image name field.

Multistate Indicators

Multistate indicators reflect the state and value of a digital register. The indicator can have a different state for each value of the register.

Multistate indicators can have one of the following six shapes:

Lines:	Straight lines
Splines:	Curved lines, using two end points and two “helper” points
Rectangles:	Rectangles, hollow or filled
Ellipses:	Ellipses, hollow or filled. This includes circles.
Text:	Text
Images:	Images from the image library. See *Images* on page 88 for details on image libraries.

The drawing properties of the shape (line width, image, etc.) as well as the text for the text shape vary according to the state and value of the register.

- Drawing multistate indicators:

To draw a multistate indicator, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Multistate Indicator Properties page in the Drawing Properties window and set the properties of the new indicator. Use the state pop-up menu to set the properties for the different states and values. If a value you want to use is not shown in the menu, enter it into the field where the state is displayed, and press the Enter key. Do not prefix the value with “Value”, the Development System does this for you.

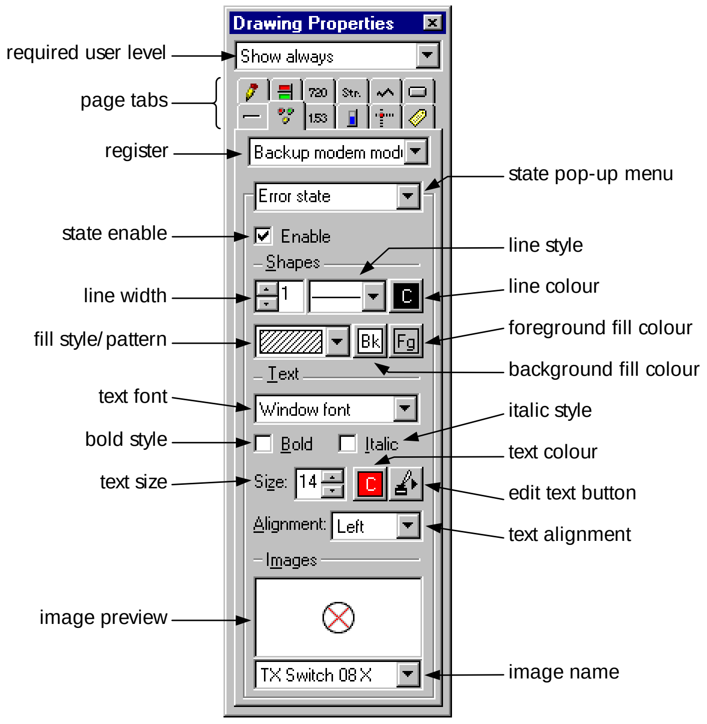
   The tab for the Multistate Indicator Properties page is marked with the same icon as the “Multistate” Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the “Multistate” Drawing Tool.
3. Choose “Multistate Indicators” from the “Draw” menu, or press the “Multistate” Drawing Tool button:

The “Multistate” Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one indicator, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.

4. Choose the secondary tool that corresponds to the shape (see above) that you want to draw.
5. Draw the indicator by clicking and dragging on the screen (Splines require two click-and-drag operations to be drawn).

The Multistate Indicator Properties Page



If no graphic object is currently selected, the Multistate Indicator Properties page shows the current drawing properties for multistate indicators. If any graphic objects are selected, the page shows all the properties applicable to the selected multistate indicators. Properties that are not applicable to any objects (such as line width for text shape indicators) appear grayed. If a particular property is different for two selected objects (e.g. two lines with different line widths are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose state and value you want to display. If you do not select a register, you will not be able to draw multistate indicators.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All multistate indicators will be displayed in the selected state. If you select "Blink state", alarm indicators will flash.

If a value you want to use is not shown in the menu, enter it into the field where the state is displayed, and press the Enter key. Do not prefix the value with "Value", the Development System does this for you.

- State enable:

Check this checkbox to enable the state or value. If you leave this box blank, the indicator will be invisible when in this state. A multistate indicator must have at least one state corresponding to a register value enabled.

- Line width:

Select the line width for lines and splines, and for the outline of rectangles and ellipses.

- Line style:

Select the line style (solid, dashed, dotted, etc.) for lines and splines, and for the outline of rectangles and ellipses.

- Line colour:

Select the colour for lines and splines, and for the outline of rectangles and ellipses.

- Fill style/pattern:

Select the fill style or fill pattern for rectangles and ellipses.

- Background fill colour:

If you selected a two-colour pattern as the fill style, select the background colour for the pattern here.

- Foreground fill colour:

If you selected "Solid colour" as the fill style, select the fill colour here. If you selected a two-colour pattern as the fill style, select the foreground colour for the pattern here.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here.

- Edit text button:

Press this button to edit the text of text shape indicators. The text is converted as described under Text Used in Indicators.

- Alignment:

Select the alignment (justification) for text here.

- Image name:

Select the image for image objects here. The image will appear in the image preview field.

- Image preview:

This field shows the image you have selected in the image name field.

Digital Indicators

Digital indicators display the value of a digital register. The text properties can vary according to the state of the register.

If the register is in the ON or OFF state, the digital indicator displays the value of the register. The value is displayed as a number, and optionally padded with zeros to a specified number of digits.

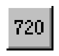
A prefix string is prepended to the value, and a suffix string is appended. Optionally, the prefix and suffix can be shown in a special margin. The prefix is always right-aligned in the margin, while the suffix is left-aligned.

If the register is in the error or masked state, a user-defined string is displayed. The strings for error and masked state are maintained separately and can be different. The prefix and suffix strings are not used with the error and masked strings, and any prefix and suffix margins are empty.

- Drawing digital indicators:

To draw a digital indicator, proceed as follows:

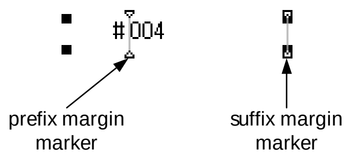
1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Digital Indicator Properties page in the Drawing Properties window and set the properties of the new indicator. Use the state pop-up menu to set the properties for the different states. The tab for the Digital Indicator Properties page is marked with the same icon as the "Digital" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Digital" Drawing Tool.
3. Choose "Digital Indicators" from the "Draw" menu, or press the "Digital" Drawing Tool button:



The "Digital" Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one indicator, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.

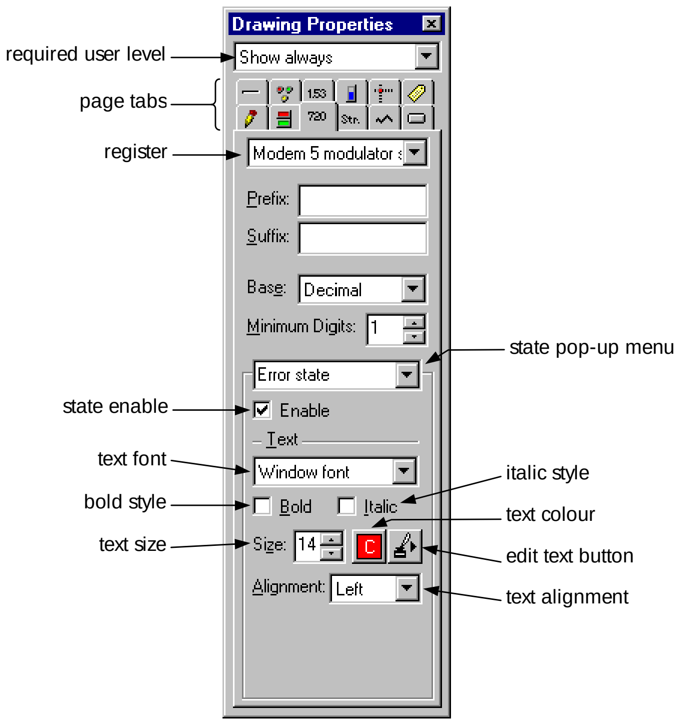
4. Draw the indicator by clicking and dragging on the screen.
5. If you want to display the prefix and/or suffix in a special margin, you must move the margin markers of the indicator. The margin markers are only visible when the "Reshape" tool is selected:



The margin markers

To attach the prefix or suffix to the value, move the margin marker all the way to the edge of the indicator, until the margin has disappeared.

The Digital Indicator Properties Page



If no graphic object is currently selected, the Digital Indicator Properties page shows the current drawing properties for digital indicators. If any graphic objects are selected, the page shows all the properties applicable to the selected digital indicators. Properties that are not applicable to any objects (such as text properties for disabled states) appear grayed. If a particular property is different for two selected digital indicators (e.g. two indicators with different suffixes are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

> Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose value you want to display. If you do not select a register, you will not be able to draw digital indicators.

- Prefix:

Enter the prefix here. The prefix is converted as described under Text Used in Indicators.

- Suffix:

Enter the suffix here. The suffix is converted as described under Text Used in Indicators.

- Base:

Select the numerical base that the number is to be displayed in.

- Minimum digits:

Select the minimum number of digits to display. If the value has less digits than is specified here, it is padded with zero to the required number of digits.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All digital indicators will be displayed in the selected state. If you select “Blink state”, alarm indicators will flash.

- State enable:

Check this checkbox to enable the state. If you leave this box blank, the indicator will be invisible when in this state. A digital indicator must have at least one state other than the Error or Masked state enabled.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here. For the On, Off, and Blink states, this button is wider and reads “Colour…” instead of just “C”.

- Edit text button:

This button is only visible for the Error and Masked states. Press this button to edit the text displayed for that state. The text is converted as described under Text Used in Indicators.

- Alignment:

Select the alignment (justification) for text here. The alignment does not affect any prefix or suffix that is shown in its special margin. The prefix is always right-aligned within the margin, the suffix left aligned.

Analog Indicators

Analog indicators display the value of an analog register. The text properties can vary according to the state of the register.

If the register is in the ON or OFF state, the analog indicator displays the value of the register. The register value is displayed as a number, either as is, or it can be modified using a factor and an offset. The number displayed is calculated from the value of the register as follows:

$$\text{display} = \text{value} \cdot \text{factor} + \text{offset}$$

To display the value unaltered, use a factor of 1 and an offset of 0.

You can specify the number of digits displayed after the decimal point. The number is rounded to the correct number of decimals. Optionally, a plus sign can be shown for positive numbers. You can also use exponential (scientific) notation for the number.

A prefix string is prepended to the value, and a suffix string is appended. Optionally, the prefix and suffix can be shown in a special margin. The prefix is always right-aligned in the margin, while the suffix is left-aligned.

Analog registers can be set to have a user configurable precision and units. The precision configured by the user will override the precision you specify when you create the analog indicator. The units configured by the user will be appended to the suffix specified in the indicator.

If the register is in the error or masked state, a user-defined string is displayed. The strings for error and masked state are maintained separately and can be different. The prefix and suffix strings are not used with the error and masked strings, and any prefix and suffix margins are empty.

- Drawing analog indicators:

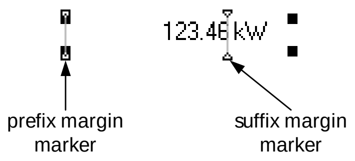
To draw an analog indicator, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Analog Indicator Properties page in the Drawing Properties window and set the properties of the new indicator. Use the state pop-up menu to set the properties for the different states. The tab for the Analog Indicator Properties page is marked with the same icon as the "Analog" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Analog" Drawing Tool.
3. Choose "Analog Indicators" from the "Draw" menu, or press the "Analog" Drawing Tool button:

The "Analog" Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one indicator, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.

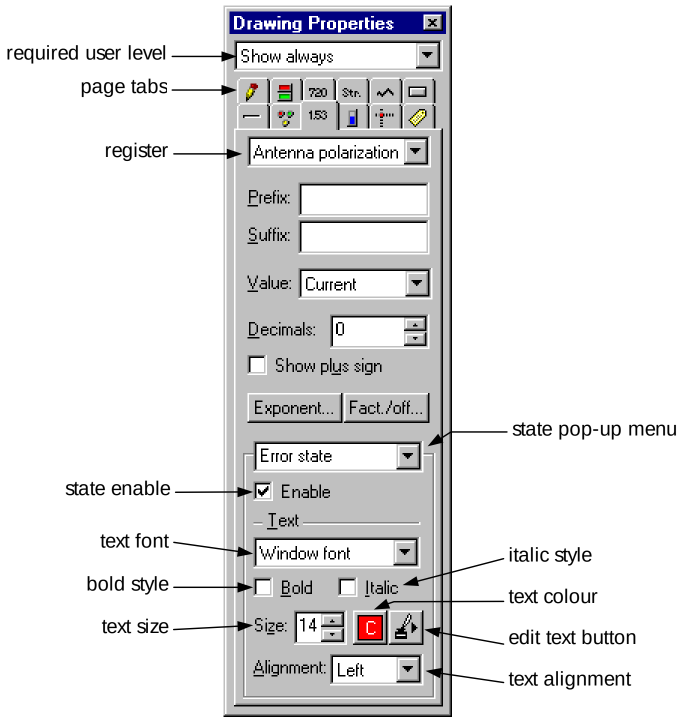
4. Draw the indicator by clicking and dragging on the screen.
5. If you want to display the prefix and/or suffix in a special margin, you must move the margin markers of the indicator. The margin markers are only visible when the "Reshape" tool is selected:



The margin markers

To attach the prefix or suffix to the value, move the margin marker all the way to the edge of the indicator, until the margin has disappeared.

The Analog Indicator Properties Page



If no graphic object is currently selected, the Analog Indicator Properties page shows the current drawing properties for analog indicators. If any graphic objects are selected, the page shows all the properties applicable to the selected analog indicators. Properties that are not applicable to any objects (such as text properties for disabled states) appear grayed. If a particular property is different for two selected analog indicators (e.g. two indicators with different suffixes are selected), that property will be shown in an indeterminate state.

▪ Required user level:
Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

▪ Page tabs:
Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

▪ Register:
Select the register whose value you want to display. If you do not select a register, you will not be able to draw analog indicators.

▪ Prefix:
Enter the prefix here. The prefix is converted as described under Text Used in Indicators.

▪ Suffix:
Enter the suffix here. The suffix is converted as described under Text Used in Indicators.

- Value:

Select which value to use.

Current:
Select this option for registers with a size of one value, or to use the value with the greatest index of a register with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the register's values.

Lowest:
Select this option to use the lowest of all the register's values.

- Decimals:

Enter the number of digits to appear after the decimal point. The number will be rounded to the specified number of decimals, or padded with zeros to the right, as necessary.

- Show plus sign:

Check this box if you want a plus sign to be prepended to positive numbers.

- The "Exponent…" button:

Press this button to switch exponential (scientific) notation on and off, and to configure the exponent. See *Analog Indicator Exponential Notation* on page 107 for a description of the Edit Exponential Notation dialog.

- The "Fact/off…" button:

Press this button to change the factor and offset of the indicator.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All analog indicators will be displayed in the selected state. If you select "Blink state", alarm indicators will flash.

- State enable:

Check this checkbox to enable the state. If you leave this box blank, the indicator will be invisible when in this state. An analog indicator must have at least one state other than the Error or Masked state enabled.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here. For the On, Off, and Blink states, this button is wider and reads "Colour…" instead of just "C".

- Edit text button:

This button is only visible for the Error and Masked states. Press this button to edit the text displayed for that state. The text is converted as described under Text Used in Indicators.

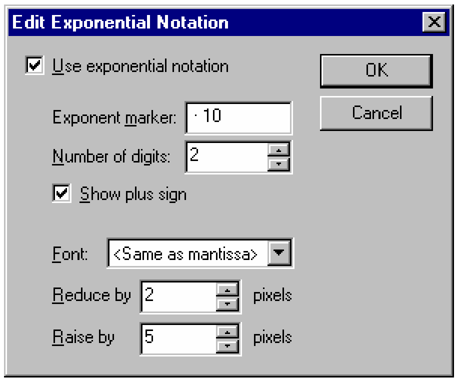
- Alignment:

Select the alignment (justification) for text here. The alignment does not affect any prefix or suffix that is shown in its special margin. The prefix is always right-aligned within the margin, the suffix left aligned.

Analog Indicator Exponential Notation

Analog indicators can display their register’s value using exponential (scientific) notation. To edit the exponential notation settings for analog indicators, press the “Exponent…” button in the Analog Indicator Properties page.

The Edit Exponential Notation Dialog



Edit Exponential Notation

☑ Use exponential notation

Exponent marker: · 10

Number of digits: 2

☑ Show plus sign

Font: <Same as mantissa>

Reduce by 2 pixels

Raise by 5 pixels

If no graphic object is currently selected, the Edit Exponential Notation dialog shows the current drawing properties for the exponent of analog indicators. If any graphic objects are selected, the dialog shows all the properties applicable to the selected analog indicators. Properties that are not applicable to any objects (such as the text properties for indicators that have no exponent) appear grayed. If a particular property is different for two selected analog indicators (e.g. two indicators with different exponent markers are selected), that property will be shown in an indeterminate state.

- Use exponential notation:

Check this box to use exponential (scientific) notation for the value.

- Exponent marker:

Enter the exponent marker here. The exponent marker is placed between the mantissa and the exponent. Usually, the exponent marker should be “ · 10”, “ E”, or “ e”.

- Number of digits:

Enter the number of digits you want to show in the exponent.

- Show plus sign:

Check this box if you want a plus sign to be prepended to positive exponents.

- Font:

Select the font for the exponent. Select <Same as mantissa> to use the same font for exponent and mantissa.

- Reduce by:

Select the amount (in points) that the exponent should be smaller than the mantissa. Please note than Windows cannot display all fonts in all sizes. The window font, for example, will usually not shrink below 14 point. Select “Arial”, “Times New Roman”, or “Courier New” if the font you selected refuses to shrink the specified amount.

This setting only affects the exponent itself, not the exponent marker.

- Raise by:

Select the number of points that the exponent should be raised above the mantissa and exponent marker. The baseline (bottom) of the exponent will be raised the specified amount above the baseline of the mantissa.

String Indicators

String indicators display the value of a string register. The text properties can vary according to the state of the register.

If the register is in the ON or OFF state, the string indicator displays the value of the register as an ASCII string.

A prefix string is prepended to the value, and a suffix string is appended. Optionally, the prefix and suffix can be shown in a special margin. The prefix is always right-aligned if it is displayed in the margin, while the suffix is left-aligned.

If the register is in the error or masked state, a user-defined string is displayed. The strings for error and masked state are maintained separately and can be different. The prefix and suffix strings are not used with the error and masked strings, and any prefix and suffix margins are empty.

- Drawing string indicators:

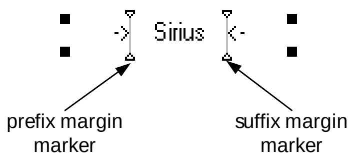
To draw a string indicator, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the String Indicator Properties page in the Drawing Properties window and set the properties of the new indicator. Use the state pop-up menu to set the properties for the different states. The tab for the String Indicator Properties page is marked with the same icon as the “String” Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the “String” Drawing Tool.
3. Choose “String Indicators” from the “Draw” menu, or press the “String” Drawing Tool button:

The “String” Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one indicator, choose “Lock Tool” from the “Draw” menu, or press the Drawing Tool button a second time.

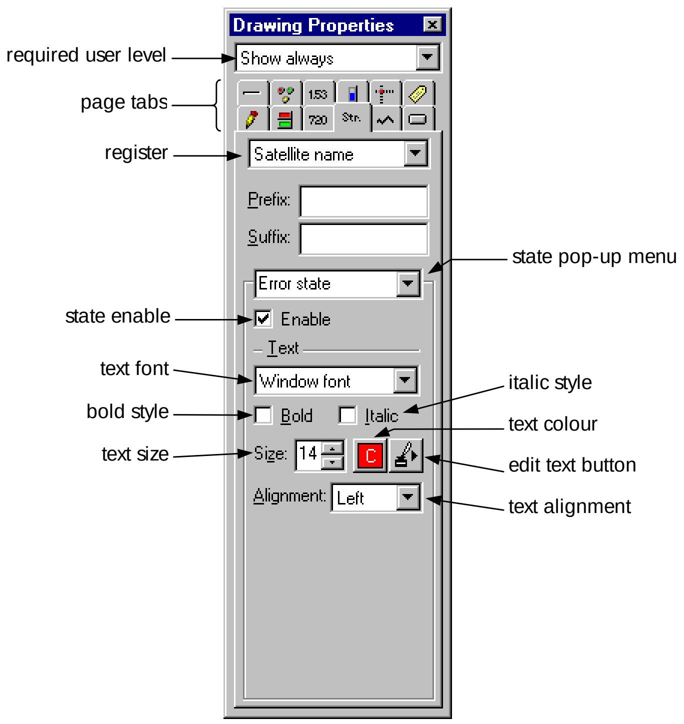
4. Draw the indicator by clicking and dragging on the screen.
5. If you want to display the prefix and/or suffix in a special margin, you must move the margin markers of the indicator. The margin markers are only visible when the “Reshape” tool is selected:



The margin markers

To attach the prefix or suffix to the value, move the margin marker all the way to the edge of the indicator, until the margin has disappeared.

The String Indicator Properties Page



If no graphic object is currently selected, the String Indicator Properties page shows the current drawing properties for string indicators. If any graphic objects are selected, the page shows all the properties applicable to the selected string indicators. Properties that are not applicable to any objects (such as text properties for disabled states) appear grayed. If a particular property is different for two selected string indicators (e.g. two indicators with different suffixes are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose value you want to display. If you do not select a register, you will not be able to draw string indicators.

- Prefix:

Enter the prefix here. The prefix is converted as described under Text Used in Indicators.

- Suffix:

Enter the suffix here. The suffix is converted as described under Text Used in Indicators.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All string indicators will be displayed in the selected state. If you select "Blink state", alarm indicators will flash.

- State enable:

Check this checkbox to enable the state. If you leave this box blank, the indicator will be invisible when in this state. A string indicator must have at least one state other than the Error or Masked state enabled.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Select the colour for text here. For the On, Off, and Blink states, this button is wider and reads "Colour…" instead of just "C".

- Edit text button:

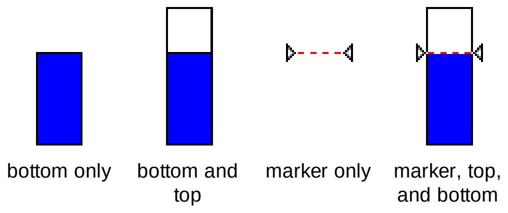
This button is only visible for the Error and Masked states. Press this button to edit the text displayed for that state. The text is converted as described under Text Used in Indicators.

- Alignment:

Select the alignment (justification) for text here. The alignment does not affect any prefix or suffix that is shown in its special margin. The prefix is always right-aligned within the margin, the suffix left aligned.

Dials

Dials display the value of an analog register in form of a slider or a coloured bar. A dial consist of three parts; a bottom, a top, and a marker:



Dials can also slide from left to right rather than from top to bottom. The marker can be an image or a line.

A dial has a bottom and a top value. The dial will be all the way at the bottom (or left) if the register has the bottom value or less, and all the way at the top (or right) if it has the top value. For values between the top and the bottom value, the dial will be between the bottom and the top.

- Drawing dials:

To draw a dial, proceed as follows:

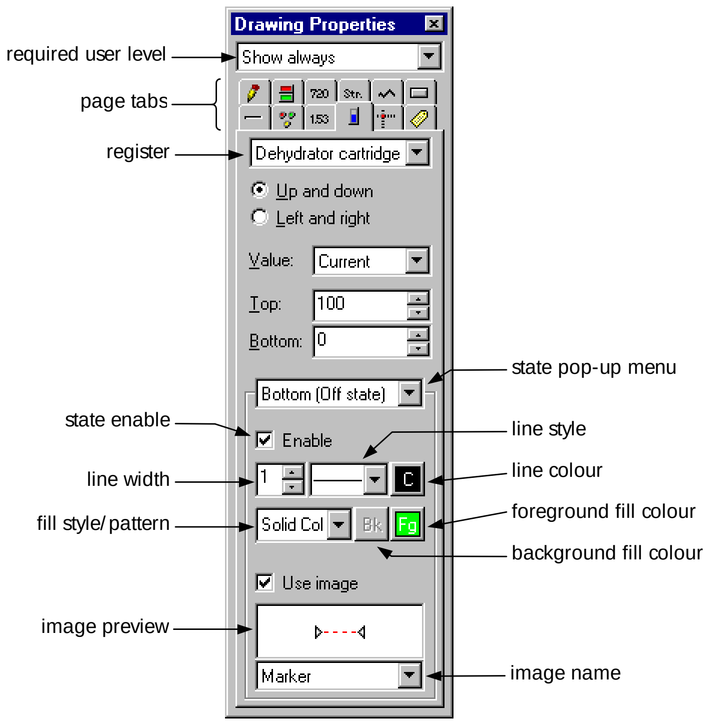
1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.

2. Go to the Dial Properties page in the Drawing Properties window and set the properties of the new dial. Use the state pop-up menu to set the properties for the different states. The tab for the Dial Properties page is marked with the same icon as the "Dials" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Dials" Drawing Tool.
3. Choose "Dials" from the "Draw" menu, or press the "Dials" Drawing Tool button:

The "Dials" Drawing Tool button

   You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one dial, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.
4. Draw the dial by clicking and dragging on the screen.

The Dial Properties Page



Show always
Dehydrator cartridge
Up and down
Left and right
Value: Current
Top: 100
Bottom: 0
Bottom (Off state)
Enable
1
Solid Col
Use image
Marker

If no graphic object is currently selected, the Dial Properties page shows the current drawing properties for dials. If any graphic objects are selected, the page shows all the properties applicable to the selected dials. Properties that are not applicable to any objects (such as line properties for disabled states) appear grayed. If a particular property is different for two selected dials (e.g. two dials with different top values are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose value you want to display. If you do not select a register, you will not be able to draw dials.

- Up and down / Left and right:

Choose the orientation of the dial. Up and down dials move vertically, left and right dials move horizontally.

- Value:

Select which value to use.

Current:
Select this option for registers with a size of one value, or to use the value with the greatest index of a register with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the register's values.

Lowest:
Select this option to use the lowest of all the register's values.

- Top:

Enter the register value for which the dial should appear all the way at the top.

If you specify a top value that is less than the bottom value, the drawing properties for the top and bottom will be reversed (i.e. the properties for the top will be used for the bottom and vice versa).

- Bottom:

Enter the register value for which the dial should appear all the way at the bottom.

If you specify a bottom value that is greater than the top value, the drawing properties for the top and bottom will be reversed (i.e. the properties for the top will be used for the bottom and vice versa).

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states and parts of the dial. All dials will be displayed in the selected state. If you select "Blink state", dials that show the value of alarm registers will flash.

- State enable:

Check this checkbox to enable the selected part of the dial for the selected state. If you leave this box blank, the part of the dial will be invisible when in this state.

- Line width:

For the top and bottom parts, select the line width for the outline of the rectangle.

For the marker, select the width of the marker line. This field is not available if you are using an image as marker instead of a line.

- Line style:

For the top and bottom parts, select the line style (solid, dashed, dotted, etc.) for the outline of the rectangle.

For the marker, select the style of the marker line. This field is not available if you are using an image as marker instead of a line.

- Line colour:

For the top and bottom parts, select the colour for the outline of the rectangle.

For the marker, select the colour of the marker line. This field is not available if you are using an image as marker instead of a line.

- Fill style/pattern:

This field is only present for the top and bottom parts. Select the fill style or fill pattern for the rectangle.

- Background fill colour:

This field is only present for the top and bottom parts. If you selected a two-colour pattern as the fill style, select the background colour for the pattern here.

- Foreground fill colour:

This field is only present for the top and bottom parts. If you selected "Solid colour" as the fill style, select the fill colour here. If you selected a two-colour pattern as the fill style, select the foreground colour for the pattern here.

- Use image:

This field is only present for the marker. Check this box to use an image for the marker rather than a line.

- Image name:

This field is only present for the marker. Select the image for the marker. The image will appear in the image preview field. This field is not available if you are using a line as marker rather than an image.

- Image preview:

This field is only present for the marker. This field shows the image you have selected in the image name field.

Graphs

Graphs display all the values of an analog register as a bar or line graph. The first value is always shown on the left, and the last value on the right.

Contrary to other indicators, graphs do not have different states. Graphs always look the same regardless of whether the register is in the ON or OFF state, or whether it is an alarm or not. If the register is in the masked or error state, graphs are always invisible.

- Drawing graphs:

To draw a graph, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Graph Properties page in the Drawing Properties window and set the properties of the new graph. Use the state pop-up menu to set the properties for the different states. The tab for the Graph Properties page is marked with the same icon as the "Graphs" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Graphs" Drawing Tool.

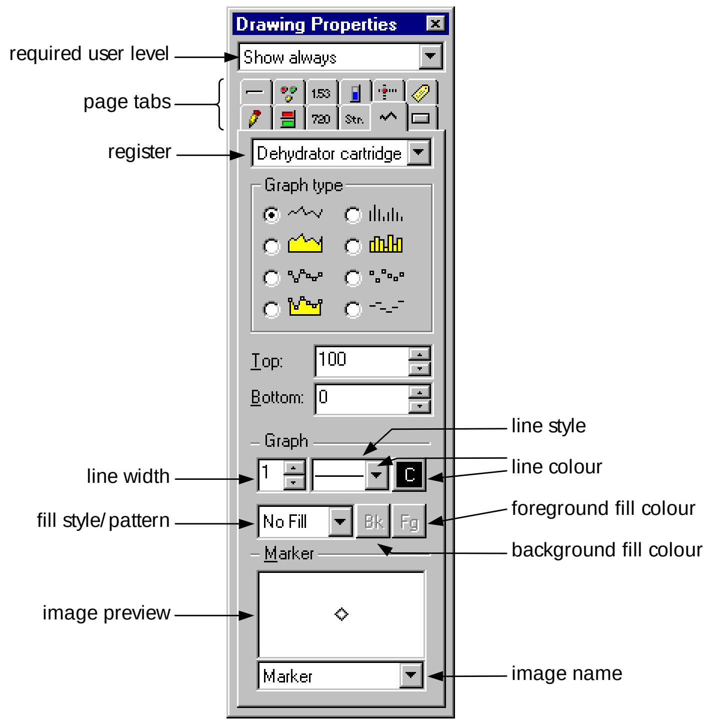
3. Choose “Graphs” from the “Draw” menu, or press the “Graphs” Drawing Tool button:

The “Graphs” Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one graph, choose “Lock Tool” from the “Draw” menu, or press the Drawing Tool button a second time.

4. Draw the graph by clicking and dragging on the screen.

The Graph Properties Page



Show always

Dehydrator cartridge

Top: 100

Bottom: 0

Graph: 1

No Fill Bk Fg

Marker: Marker

If no graphic object is currently selected, the Graph Properties page shows the current drawing properties for graphs. If any graphic objects are selected, the page shows all the properties applicable to the selected graphs. Properties that are not applicable to any objects (such as line properties for disabled states) appear grayed. If a particular property is different for two selected graphs (e.g. two graphs with different top values are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

▪ Page tabs:
Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

▪ Register:
Select the register whose values you want to display. If you do not select a register, you will not be able to draw graphs.

▪ Graph type:
Select the type of graph you want to use.

▪ Top:
Enter the register value for which the graph should touch the top of the rectangle.

▪ Bottom:
Enter the register value for which the graph should touch the bottom of the rectangle.

▪ Line width:
Select the line width for line graphs, and for the outline of the bars of bar graphs.

▪ Line style:
Select the line style (solid, dashed, dotted, etc.) for line graphs, and for the outline of the bars of bar graphs.

▪ Line colour:
Select the colour for line graphs, and for the outline of the bars of bar graphs.

▪ Fill style/pattern:
Select the fill style or fill pattern for filled line graphs and bar graphs.

▪ Background fill colour:
If you selected a two-colour pattern as the fill style, select the background colour for the pattern here.

▪ Foreground fill colour:
If you selected "Solid colour" as the fill style, select the fill colour here. If you selected a two-colour pattern as the fill style, select the foreground colour for the pattern here.

▪ Image name:
Select the image to be used as markers for graphs that use markers for the points. The image will appear in the image preview field.

▪ Image preview:
This field shows the image you have selected in the image name field.

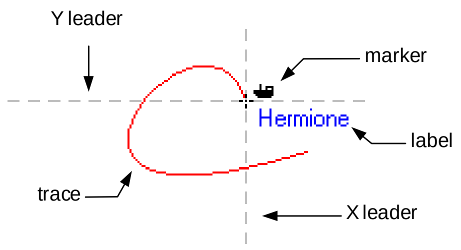
X-Y Position Markers

X-Y position markers are images that change their position within a bounding rectangle by using the value of two analog registers, one for the X coordinate, and one for the Y coordinate.

X-Y position markers can either display a static image, or a different image depending on the ON/OFF state of a register, similar to an image bistate indicator. You can also display a text label near the marker. The text label can contain the value of a string register, if desired.

X-Y position markers support leaders. The leaders are two lines, one horizontal, and one vetical, that go through the marker and extend to the edge of the bounding box to give a cross-hairs effect. The vertical line is called the X leader, because it has the same X coordinate as the marker. The horizontal line is called the Y leader, because it has the same Y coordinate. You can enable and disable the X and Y leader separately.

X-Y position markers also support a "trace". If the X and Y coordinate registers have a size of more than one value, you can display a curve that goes through all the points described by all the values.



▪ Drawing X-Y position markers:

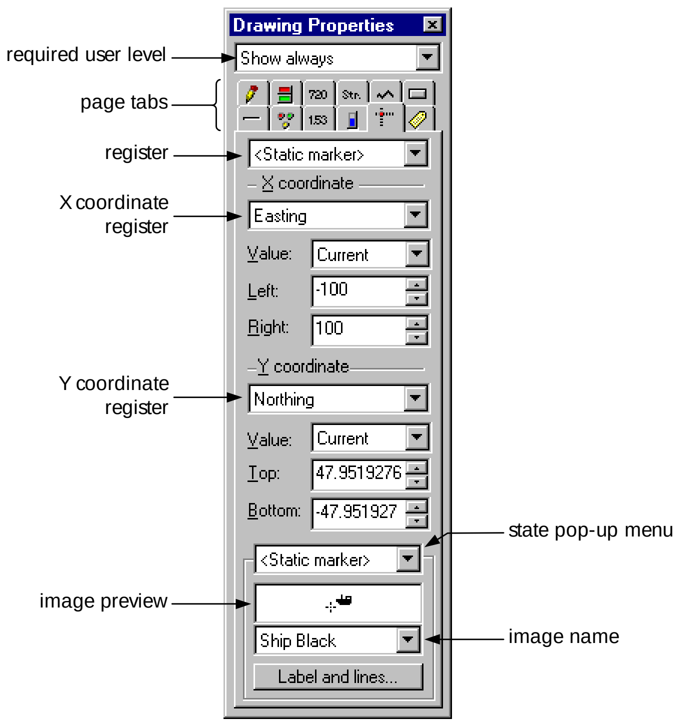
To draw an X-Y position marker, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the X-Y Position Marker Properties page in the Drawing Properties window and set the properties of the new marker. Use the state pop-up menu to set the properties for the different states. The tab for the X-Y position marker Properties page is marked with the same icon as the "X-Y Position Markers" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Markers" Drawing Tool.
3. Choose "X-Y Position Markers" from the "Draw" menu, or press the "Markers" Drawing Tool button:

   The "Markers" Drawing Tool button

   You will not be able to select this tool if you failed to select a register for the X or Y coordinate in step 2. If you want to draw more than one marker, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.
4. Draw the marker by clicking and dragging on the screen.
5. Move the label, if any to the desired position by clicking on it and dragging it, or resize it by dragging the label's handles. The label's handles are only visible, and you can only move the label, when the "Reshape" tool is selected.

The X-Y Position Marker Properties Page



If no graphic object is currently selected, the X-Y Position Marker Properties page shows the current drawing properties for X-Y position markers. If any graphic objects are selected, the page shows all the properties applicable to the selected X-Y position markers. Properties that are not applicable to any objects (such as ON state markers properties for static markers) appear grayed. If a particular property is different for two selected X-Y position markers (e.g. two X-Y position markers with different top values are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register on whose state the marker image depends. Select <Static marker> to show a static image that does not depend on the state of any register.

- X coordinate register:

Select the register whose value should determine the X (left-right) position of the marker. If you do not select a register, you will not be able to draw X-Y position markers.

- Value:

Select which value of the X coordinate register to use.

Current:
Select this option for registers with a size of one value, or to use the value with the greatest index of a register with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the register's values.

Lowest:
Select this option to use the lowest of all the register's values.

- Left:

Enter the register value for which the marker should appear at the left edge of the bounding box.

- Right:

Enter the register value for which the marker should appear at the right edge of the bounding box.

- Y coordinate register:

Select the register whose value should determine the Y (up-down) position of the marker. If you do not select a register, you will not be able to draw X-Y position markers.

- Value:

Select which value of the Y coordinate register to use.

Current:
Select this option for registers with a size of one value, or to use the value with the greatest index of a register with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the register's values.

Lowest:
Select this option to use the lowest of all the register's values.

- Top:

Enter the register value for which the marker should appear at the top edge of the bounding box.

- Bottom:

Enter the register value for which the marker should appear at the bottom edge of the bounding box.

- State pop-up menu:

Use this pop-up menu to view and change the properties for the different indicator states. All markers (except static markers) will be displayed in the selected state. If you select "Blink state", markers that show the state of alarm registers will flash.

Static markers only have one state, called "<Static marker>". Selecting <Static marker> does not affect the way non-static markers are shown.

- Image name:

Select the image to display here. The image will appear in the image preview field.

- Image preview:

This field shows the image you have selected in the image name field.

- The "Label and lines…" button:

Press this button to edit the properties for the label, the X and Y leaders, and the trace. See X-Y Position Marker Labels and Lines below for a description of the Edit Label and Lines dialog.

X-Y Position Marker Labels and Lines

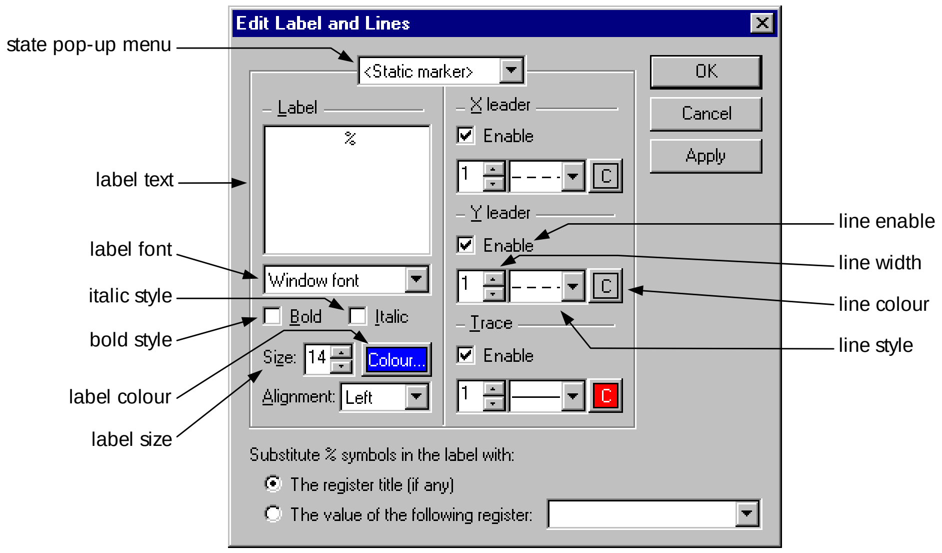
X-Y position markers can display a text label near the marker. The text label can contain the value of a string register, if desired.

X-Y position markers support leaders. The leaders are two lines, one horizontal, and one vertical, that go through the marker and extend to the edge of the bounding box to give a cross-hairs effect. The vertical line is called the X leader, because it has the same X coordinate as the marker. The horizontal line is called the Y leader, because it has the same Y coordinate. You can enable and disable the X and Y leader separately.

X-Y position markers also support a “trace”. If the X and Y coordinate registers have a size of more than one value, you can display a curve that goes through all the points described by all the values.

To edit the settings for the label, the leaders, and the trace, press the “Label and lines…” button in the X-Y Position Marker Properties page.

The Edit Label and Lines Dialog



Edit Label and Lines

<Static marker>

Label: %

Window font

[ ] Bold [ ] Italic

Size: 14 Colour...

Alignment: Left

X leader: [✓] Enable 1 - - - - C

Y leader: [✓] Enable 1 - - - - C

Trace: [✓] Enable 1 ——— C

Substitute % symbols in the label with:

[● The register title (if any)]

○ The value of the following register:

- State pop-up menu:
Use this pop-up menu to view and change the properties for the different indicator states. All markers (except static markers) will be displayed in the selected state. If you select “Blink state”, markers that show the state of alarm registers will flash.

Static markers only have one state, called “<Static marker>“. Selecting <Static marker> does not affect the way non-static markers are shown.

- Label text:
Enter the taxt that should be shown in the label here. The text is converted as described under Text Used in Indicators using the register whose state the marker reflects. You can choose to replace % symbols in the label text with the value of a string register rather than with the register name.

If you do not want a label, leave this field blank.

- Label font:
Select the font for the label. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

Bold style/italic style:

Check these boxes for bold and/or italic label text.

- Label size:

Select the text size (pitch) of the label here.

- Label colour:

Select the colour for the label text here.

- Alignment:

Select the alignment (justification) for the label text here.

- X Leader:

Shows the settings for the X (vertical) leader.

- Y Leader:

Shows the settings for the Y (horizontal) leader.

- Trace:

Shows the settings for the trace.

- Line enable:

Check this checkbox to enable the leader or trace for the selected state.

- Line width:

Select the line width for the leader or trace.

- Line style:

Select the line style (solid, dashed, dotted, etc.) for the leader or trace.

- Line colour:

Select the colour for the leader or trace.

- The "Apply" button:

Press this button to apply the settings to the currently selected X-Y position markers.

Controls

Controls are buttons the user can press to initiate certain actions. There are three types of controls:

- SCL program controls:

SCL program controls execute an SCL program when pressed. The button stays pressed until the program has finished executing, so that the user cannot execute the same program twice using the same button. Use this type of control to allow the user to control equipment.

SCL program controls are invisible to users who do not have Control privileges, unless the program they execute allows execution without signing on. See SCL Programs for details.

- Screen controls:

Screen controls take the user to a screen. If there is already a window showing the screen, that window is brought to the foreground, otherwise a new window containing the screen is opened. If the screen is configured as a dialog, a dialog containing the screen is shown. Use this type of control to allow the user to navigate the station.

- Network screen controls:

Network screen controls show a screen on a remote station. If U.P.M.A.C.S. is not connected to the station, an error message appears. Use this type of control to implement navigation buttons on network maps.

A control can have an optional caption. The caption is centered on the button, and is displayed using the default font and size specified for buttons in the Display control panel. The button caption is never obscured by other graphic objects, even if they are in front of the button. This allows you to place an object or an indicator between the button and its caption.

Buttons are usually the system button colour, specified in the Display control panel. For very large buttons, this can be undesirable, so U.P.M.A.C.S. allows buttons that are the same colour as the background colour of the screen. Use this feature sparingly, as buttons that have the same colour as the screen are difficult to see.

- Drawing controls:

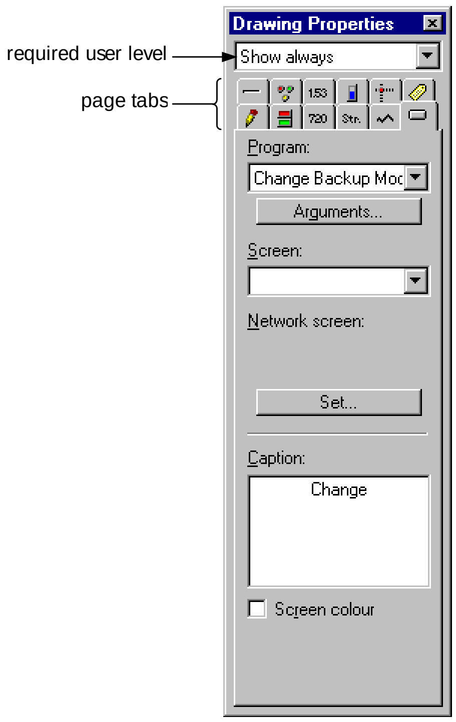
To draw a control, proceed as follows:

1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.
2. Go to the Control Properties page in the Drawing Properties window and set the properties of the new control. The tab for the Control Properties page is marked with the same icon as the “Control” Drawing Tool button (see picture below).
3. If you specify a program, the new control will be an SCL program control. If you specify a screen, the new control will be a Screen control. If you specify a network screen, the new control will be Network screen control. Make sure to specify one of the three, or you will not be able to select the “Controls” Drawing Tool.
4. Choose “Controls” from the “Draw” menu, or press the “Controls” Drawing Tool button:

   The “Controls” Drawing Tool button

   You will not be able to select this tool if you failed to specify a program, screen, or network screen in step 2. If you want to draw more than one control, choose “Lock Tool” from the “Draw” menu, or press the Drawing Tool button a second time.
5. Draw the control by clicking and dragging on the screen.

The Control Properties Page



If no graphic object is currently selected, the Control Properties page shows the current drawing properties for controls. If any graphic objects are selected, the page shows all the properties applicable to the selected controls. If no controls are among the selected objects, all properties appear grayed. If a particular property is different for two selected controls (e.g. two controls with different captions are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

SCL program controls are always invisible to users who do not have Control privileges, unless the program they execute allow execution without signing on. There is no need to specify "Control privileges" for an SCL program control to prevent unauthorized users from executing SCL programs.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Program:

For SCL program controls, select the program here. For Screen and Network screen controls, this field is blank.

If you select a program here, the “Screen” and “Network screen” fields are cleared. To turn a different type of control into an SCL program control, select it and then select a program here. Press the “Arguments…” buttons to specify arguments for the program. See Specifying Arguments for SCL Programs for a description of the Edit Program Arguments dialog.

- Screen:

For Screen controls, select the screen here. For SCL program and Network screen controls, this field is blank.

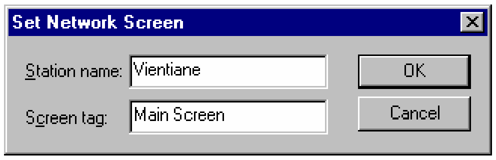
If you select a screen here, the “Program” and “Network screen” fields are cleared. To turn a different type of control into a Screen control, select it and then select a screen here.

- Network screen:

For Network screen controls, this field shows the station name and screen tag. The field has two lines: the station name is shown on the first line, the screen tag below it. For SCL program and Screen controls, this field is blank.

Press The “Set…” button set the station name and screen tag. If you press this button, the “Program” and “Screen” fields are cleared. To turn a different type of control into a Network screen control, select it and then press this button.

The Set Network Screen dialog has two text entry fields:



Station name: Vientiane
Screen tag: Main Screen

Station name:
Enter the name that the station will be connected as. This name must be the same as the name entered in the “Connect as:” field of the Connect To dialog of the U.P.M.A.C.S. Operate system.

Screen tag:
Enter the tag of the screen this button should activate.

- Caption:

Enter the caption of the button. The caption can have multiple lines.

- Screen colour:

Check this box if you want the button to be the same colour as the screen. This will make the button harder to see, but can avoid a cluttered appearance if you use very large buttons.

Labels

Labels are text objects. They never change appearance, but the text can contain placeholders for the name of a register. You can use labels to label indicators for user-definable registers. If the user changes the name of the register, the label will change accordingly.

The text of the label is converted in the same manner as text used in indicators, that is, any percentage signs appearing in the text will be replaced by the register name.

- Drawing labels:

To draw a label, proceed as follows:

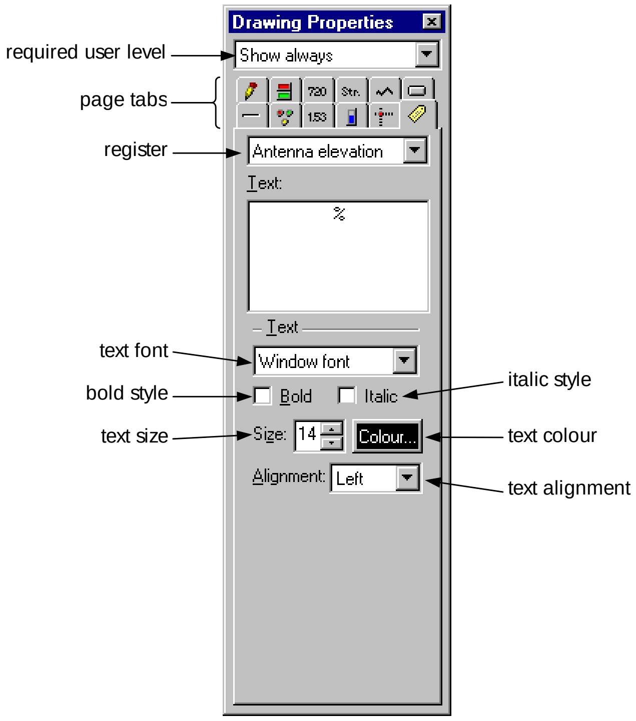
1. Make sure that no graphic objects are selected by clicking on some empty space in the editor window.

2. Go to the Label Properties page in the Drawing Properties window and set the properties of the new label. The tab for the Label Properties page is marked with the same icon as the "Labels" Drawing Tool button (see picture below). Make sure to select a register, or you will not be able to select the "Labels" Drawing Tool.
3. Choose "Labels" from the "Draw" menu, or press the "Labels" Drawing Tool button:

The "Labels" Drawing Tool button

You will not be able to select this tool if you failed to select a register in step 2. If you want to draw more than one label, choose "Lock Tool" from the "Draw" menu, or press the Drawing Tool button a second time.
4. Draw the label by clicking and dragging on the screen.

The Label Properties Page



Show always
Antenna elevation
Text: %
Text
Window font
Bold Italic
Size: 14
Alignment: Left

If no graphic object is currently selected, the Label Properties page shows the current drawing properties for labels. If any graphic objects are selected, the page shows all the properties applicable to the selected labels. If no labels are among the selected objects, all properties appear grayed. If a particular property is different for two selected labels (e.g. two labels with different fonts are selected), that property will be shown in an indeterminate state.

- Required user level:

Select the minimum user clearance or the user privilege for the object. If you select a clearance or privilege other than Show always, the object will only be visible if a user with the specified clearance or privilege has signed on.

The user clearance of each registered user is defined by the administrator of the Operate System. The privileges are assigned to different user clearances, also by the administrator.

Note: This field is common to all the property pages. A change will affect all types of objects.

- Page tabs:

Select the properties page here. If any graphic objects are currently selected, the tabs for pages that do not apply to any of the selected objects are grayed. You can click on any of the tabs to change to the appropriate page, even those that are grayed.

- Register:

Select the register whose name you wish to use in the text. If you do not select a register, you will not be able to draw labels.

- Text:

Enter the text of the label here. The text can be multiple lines. The text is converted as described under Text Used in Indicators.

- Text font:

Select the font for text. U.P.M.A.C.S. only supports a number of predefined fonts. You cannot add fonts to this menu.

- Bold style/italic style:

Check these boxes for bold and/or italic text.

- Text size:

Select the text size (pitch) here.

- Text colour:

Press this button to select the text colour.

- Text alignment:

Select the alignment (justification) for text here.

SABus Requests

You can add objects to you station file that enable third-party systems to poll U.P.M.A.C.S. via one or more of the serial ports of the computer, called uplink ports. The baud rate, character format, flow control, COM ports, etc. used for those ports are configured from within the Operate System. All you have to do to enable other systems to poll U.P.M.A.C.S. is provide a set of SABus requests, (queries and commands) in the station file.

The protocol used to poll U.P.M.A.C.S. is based on Scientific Atlanta's SABus protocol. You do not specify the complete protocol in the station file, but only the opcodes and parameters, and the data that will be returned for each request.

The exact serial protocol is described in *Appendix D: Uplink Port Protocol* on page 182, together with the two built-in commands used to sign on and off, and to acknowledge alarms.

Each SABus request object is identified by a single byte opcode (short for operation code) and a set of fixed parameters. When a packet of data is received on an uplink port, the opcode together with the fixed parameters is used to determine which SABus request object the packet refers to. The fixed parameters must follow the opcode immediately and without separators in the request packet.

The fixed parameters are used mostly for queries. A query is a request for which U.P.M.A.C.S. constructs the response for you from the values of registers you specify. Queries cannot decode any parameters that the request contains: you must create a separate query object for each possible set of parameters. Each of these query objects will have different fixed parameters, and use different registers to construct the response.

Example:

Your system talks to four up converters, and you want to provide a way to ask for the frequency of an up converter via an uplink port. You want your request to use the opcode "F" (you can use any opcode other than the two reserved opcodes "U" and "A"), followed by the converter number.

You must create four separate SABus queries, one for each up converter. Each request will have the same opcode "F", but a different fixed parameter, either "1", "2", "3", or "4". Each query will use the register that contains the frequency of the correct converter to construct the response.

When a request with opcode "F" is received on an uplink port, U.P.M.A.C.S. will check the first parameter to determine which of the four queries to execute, and then construct the response accordingly.

You can also specify fixed parameters for commands, although commands use an SCL program to evaluate the data and can decode parameters on the fly. You must use fixed parameters for a command if the command has the same opcode as a query, so U.P.M.A.C.S. can distinguish between them.

You must be careful to choose your fixed parameters so that there can never be any doubt about which SABus request object should be used for a particular packet of data. Also keep in mind that since the parameters are not separated by special separators, U.P.M.A.C.S. might not be able to tell where one parameter ends and the next begins. It is, for example, not possible to tell the difference between the parameter "A" followed by the parameter "BC", and the parameter "AB" followed by the parameter "C". Both simply look like "ABC" in the request packet. Similarly, if the opcode is followed by the string "10", it is not possible for U.P.M.A.C.S. to tell whether this is a single parameter "10", or a parameter "1" followed by a parameter "0". You can make sure that parameters are properly parsed by using the same width for corresponding parameters in all queries with the same opcode (e.g. "05" and "12"), or by using separation characters (e.g. ",3" and ",7"). The second method will not work on the last parameter, as there is not way to specify that a parameter should extend to the end of the data. If the opcode and fixed parameters of more than one request fit an incoming data packet, U.P.M.A.C.S. will pick one at random, and unexpected behaviour will result.

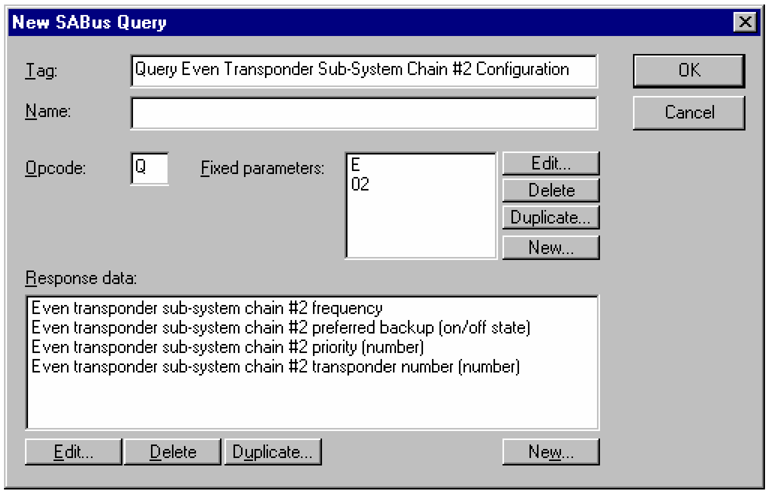
If a data packet is received on an SABus port, and no request with the corresponding opcode has the right set of parameters, a "Bad Parameter" error packet will be returned to the remote system. The packet will include the number of the parameter, i.e. whether the 1st, 2nd, or 3rd parameter could not be matched. You should therefore never lump several parameters together into a single fixed parameter, but specify them as separate fixed parameters.

SABus Queries

Queries are SABus requests for which U.P.M.A.C.S. constructs a response for you from a number of registers you specify. Queries cannot have any parameters beyond their fixed parameters. If you need different data to be returned for two different sets of queries, you must create two separate queries.

The response of a query is constructed from response data objects you specify.

The New SABus Query Dialog



Tag: Query Even Transponder Sub-System Chain #2 Configuration

Name:

Opcode: Q

Fixed parameters:
E
02

Response data:
Even transponder sub-system chain #2 frequency
Even transponder sub-system chain #2 preferred backup (on/off state)
Even transponder sub-system chain #2 priority (number)
Even transponder sub-system chain #2 transponder number (number)

- Tag:
Enter the tag by which the query is identified. Each SABus request, whether query or command, must have a unique tag.
- Name:
Enter the name of the query. Leave this field blank if you want to use the tag as name.
- Opcode:
Enter the opcode of the query. The opcode must be a single character.

The opcode of SABus requests does not have to be unique. However, you must make sure that the opcode together with the fixed parameters uniquely identify the request.
- Fixed parameters:
Lists all the fixed parameters. Use the buttons to create, delete, or edit the parameters. Use the "Duplicate…" button to duplicate the selected parameter.

The list shows the parameters in the order they must appear in the request packet. You can change the order of the parameters by grabbing them with the mouse and dragging them to a new position.

The fixed parameters are fixed strings, not regular expressions. They must appear in the packet exactly as shown in this list, without any separation characters. You must make sure that the fixed parameters together with the opcode uniquely identify the request.
- Response data:
Lists a brief description of all the response data objects. Use the buttons to create, delete, or edit the data. Use the "Duplicate…" button to duplicate the selected data object.

Each response data object will take the state or value of a register and format it according to your specifications. The resulting data strings will be concatenated to give the full response data string, which is sent as the data in the response packet. The response data strings will be assembled in the order shown in the list. You can change the order of the data objects by grabbing them with the mouse and dragging them to a new position.

See *Response Data Objects* on page 128 for details.

Response Data Objects

U.P.M.A.C.S. uses response data objects to assemble the response to SABus queries. Each data object takes information from one or more registers and encodes it according to your specifications. The resulting data strings are then concatenated to form the full response data. The full data is wrapped in a response packet together with the query's opcode, and sent back to the equipment. See *Appendix D: Uplink Port Protocol* on page 182 for a description of the response packet format.

You create response data objects from within the New SABus Query dialog. When you create a new data object, you will be asked to select the type of the new data object.

There are six types of data objects that encode the value or state of one or more registers:

- Register state response data
- Digital register value response data (number)
- Digital register value response data (strings)
- Analog register value response data
- String register value response data
- Register status bits response data

In addition, there are two types of data objects that do not take their values directly from registers:

- Processor response data
- Fixed data string response data

The different types of response data are described in the following chapters.

Register State Response Data

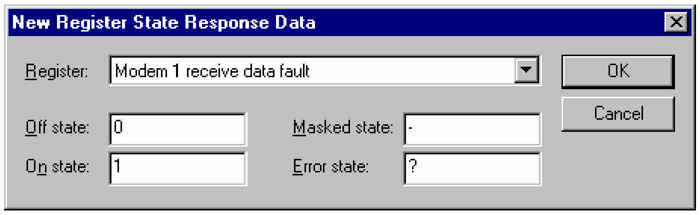
Register state response data objects use one of a number of fixed data strings, depending on the ON/OFF, error, and masked state of a register. The register can be any type.

Use register state response data objects to provide information about settings and alarms that are stored in a register's ON/OFF state, and that you want to represent using one or more bytes within the response. If you want to use a single bit to represent the state, use a register status bits response data object instead.

The data strings that are sent can be any sequence of printable characters (ASCII $20-$7E). Typically, you would use strings like "1" and "0", "on" and "off", "aut" and "man", "FLT" and "OK", etc. It is usually easiest for the remote system to decode the data if you use strings of equal length, but you can use strings of different lengths if you want. You should also specify strings to be sent when the register is in its error or masked state.

You can also use this type of response to send the error and masked state of a register. For example, you can send information about the masked state of a register by specifying the string "Msk" (or something similar) for the masked state, and "Unm" for the ON, OFF, and error states. Similarly, you could specify the string "E" for the error state, and "O" for the ON, OFF, and masked states to let a remote system query the error state of a register.

The New Register State Response Data Dialog



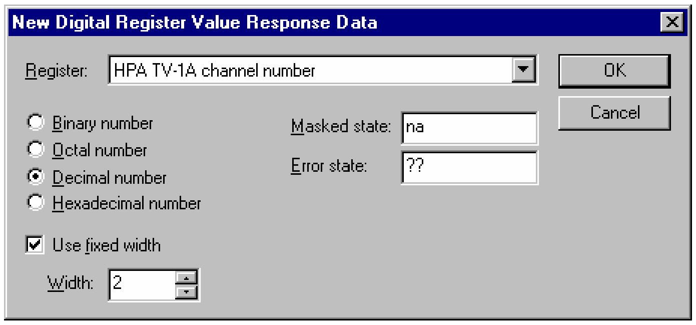
Register: Modem 1 receive data fault
Off state: 0
On state: 1
Masked state: -
Error state: ?

- Register:
Select the register whose state you want to use
- Off state:
Enter the data string to use if the register is in the OFF state.
- On state:
Enter the data string to use if the register is in the ON state.
- Masked state:
Enter the data string to use if the register is masked. If you do not specify a string for the masked state, an empty string will be used if the register is masked.
- Error state:
Enter the data string to use if the register is in its error state. If you do not specify a string for the error state, an empty string will be used if the register is in the error state.

Digital Register Value Response Data (Number)

This type of response data object simply formats the value of a digital register as a binary, octal, decimal, or hexadecimal number, written out in ASCII characters. Use this type of object with digital registers whose value represents a number, like a channel or satellite number.

The New Digital Register Value Response Data Dialog



Register: HPA TV-1A channel number
Binary number
Octal number
[✓ Decimal number]
Hexadecimal number
Masked state: na
Error state: ??
[✓] Use fixed width
Width: 2

- Register:
Select the register whose value you want to use
- Binary/Octal/Decimal/Hexadecimal number:
Select the base for the representation of the number
- Use fixed width:
Check this box to use a fixed width. The number will be padded to the left with 0's to fill the required width. If the number does not fit into the width specified, all 9's will be used (e.g. any number 100 and above encoded with a fixed width of 2 will give "99").

Width:
Enter the number of bytes to use as the fixed width.

- Masked state:
Enter the data string to use if the register is masked. If you do not specify a string for the masked state, an empty string will be used if the register is masked.

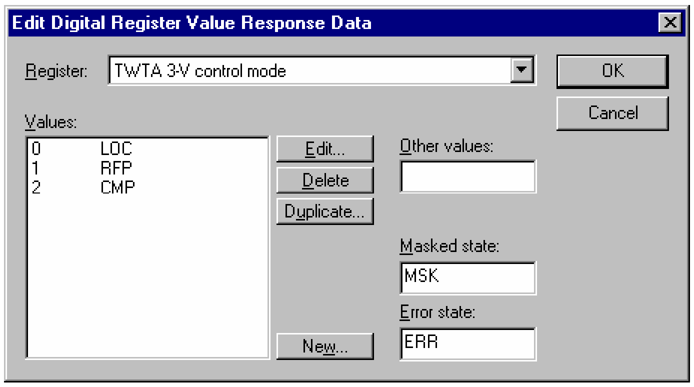
- Error state:
Enter the data string to use if the register is in its error state. If you do not specify a string for the error state, an empty string will be used if the register is in the error state.

Digital Register Value Response Data (Strings)

This type of response data objects uses a different data string for each register value you specify. Use this type of object for digital registers whose value represents different settings or conditions rather than numerical values. You can then specify a different string for each of the relevant values.

It is usually easiest for the remote system to decode the data if all the strings you specify have the same length, but you can use strings of different lengths if you want.

The New Digital Register Value Response Data Dialog


Edit Digital Register Value Response Data

Register: TWTA 3-V control mode

Values:

0	LOC
1	RFP
2	CMP

Other values:

Masked state: MSK

Error state: ERR

- Register:
Select the register whose value you want to use

- Values:
Shows a list of values and the strings that will be used for them. Use the buttons to create, delete, duplicate, or edit strings.

- Other values:
Enter the data string to use for all values that are not in the "Values" list. If you do not specify a string, an empty string will be used if the value is not in the list.

- Masked state:
Enter the data string to use if the register is masked. If you do not specify a string for the masked state, an empty string will be used if the register is masked.

- Error state:
Enter the data string to use if the register is in its error state. If you do not specify a string for the error state, an empty string will be used if the register is in the error state.

Analog Register Value Response Data

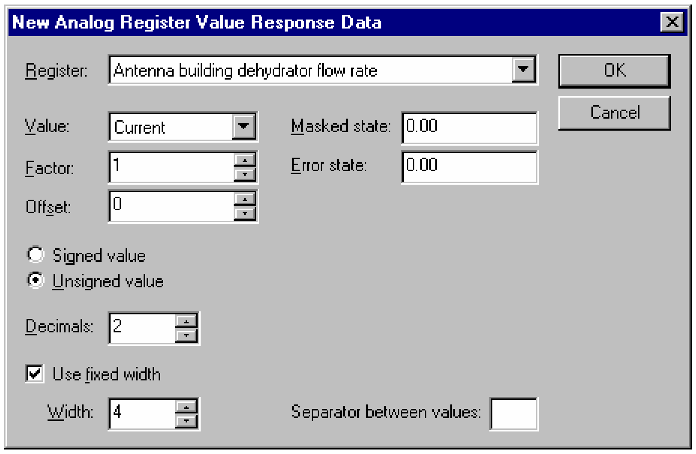
Analog resister value response data objects format the value of an analog register as number, written out in ASCII characters. You can provide a factor and an offset to be applied to the value before formatting it. The number written to the data string is calculated from the value of the register as follows:

number = value · factor + offset

To use the value unaltered, specify a factor of 1 and an offset of 0.

If the register has a size of more than one value, you can either format a single value, or all of the values one after the other, separated by a fixed separator string.

The New Analog Register Value Response Data Dialog


New Analog Register Value Response Data

Register: Antenna building dehydrator flow rate

Value: Current

Masked state: 0.00

Factor: 1

Error state: 0.00

Offset: 0

( ) Signed value
(●) Unsigned value

Decimals: 2

☑ Use fixed width

Width: 4

Separator between values:

- Register:
Select the register whose value you want to use
- Value:
Select which value to use.

Current:
Select this option for registers with a size of one value, or to use the value with the greatest index of a register with a size of more than one value. This corresponds to the last value added by an SCL program.

Highest:
Select this option to use the highest of all the register’s values.

Lowest:
Select this option to use the lowest of all the register’s values.

All values:
Select this option to write out all of the register’s values, one after the other. The values are separated by the separator, described below.

- Factor:
Enter the factor with which the value of the register is to be multiplied before writing it to the data string. The factor is applied before the offset.

- Offset:

Enter the offset that is to be added to the value of the register before writing it to the data string. The offset is applied after the factor.

- Signed/Unsigned value:

Choose whether you want to include a plus or minus sign before the number. If you choose "Unsigned value", all numbers less than 0 will be written as 0. If you choose "Signed value", the sign will be included for positive as well as negative numbers. The value 0 will also be prepended with a plus sign.

- Decimals:

Enter the number of digits to be written after the decimal point. The number will be rounded to the specified number of decimals, or padded with zeros to the right, as necessary. To use an implied decimal point, enter 0 for the decimals and specify a factor that will scale the number to remove the correct number of decimals. E.g., to imply 3 decimal digits, use a factor of 1000.

- Use fixed width:

Check this box to use a fixed width. The number will be padded to the left with 0's to fill the required width. If the number does not fit into the width specified, all 9's will be used (e.g. any number 100 and above encoded with a fixed width of 4 and one decimal will give "99.9").

Width:
Enter the number of bytes to use as the fixed width. The width does not include the plus or minus sign if you selected the "Signed number" option. It does, however, include the decimal point if you specified a number of decimals other than 0. Consequently, "-05.56" is considered to have length 5: two digits, a decimal point, and two decimal digits. The decimal point is counted, but the minus sign is not.

Masked state:
Enter the data string to use if the register is masked. If you do not specify a string for the masked state, an empty string will be used if the register is masked.

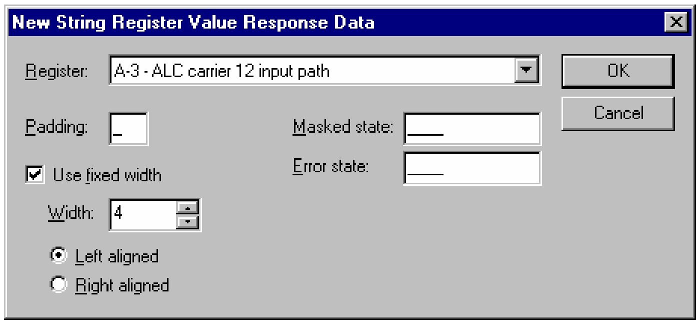
Error state:
Enter the data string to use if the register is in its error state. If you do not specify a string for the error state, an empty string will be used if the register is in the error state.

Separator between values:
Enter the separator that will be placed between the different values of a register that has a size of more than one value. This field is only available if you selected "All values" in the "Value" field.

String Register Value Response Data

String register value response data objects use the value of a string register as the data. Since the SABus protocol does not allow non-printable characters (ASCII \$00-\$1F and \$7F-\$FF), any non-printable characters in the register value will be replaced by a padding character you specify.

The New String Register Value Response Data Dialog



Register: A-3 - ALC carrier 12 input path
Padding: _
Masked state: ___
Error state: ___
[✓] Use fixed width
Width: 4
(●) Left aligned
( ) Right aligned

- Padding:

Enter the padding character here. The padding character is used to replace any non-printable characters (ASCII $00-$1F and $7F-$FF) and to pad the value to the correct width if you specified a fixed width.

- Use fixed width:

Check this box to use a fixed width. The string will be padded to the left or right with the padding character or truncated to fit the width exactly.

Width:
Enter the number of bytes to use as the fixed width.

Left aligned:
Choose this option to left-align the value of the register within the specified width. Any padding will be placed to the right of the value, or the rightmost characters will be clipped to make the value fit the width.

Right aligned:
Choose this option to right-align the value of the register within the specified width. Any padding will be placed to the left of the value, or the leftmost characters will be clipped to make the value fit the width.

- Masked state:

Enter the data string to use if the register is masked. If you do not specify a string for the masked state, an empty string will be used if the register is masked.

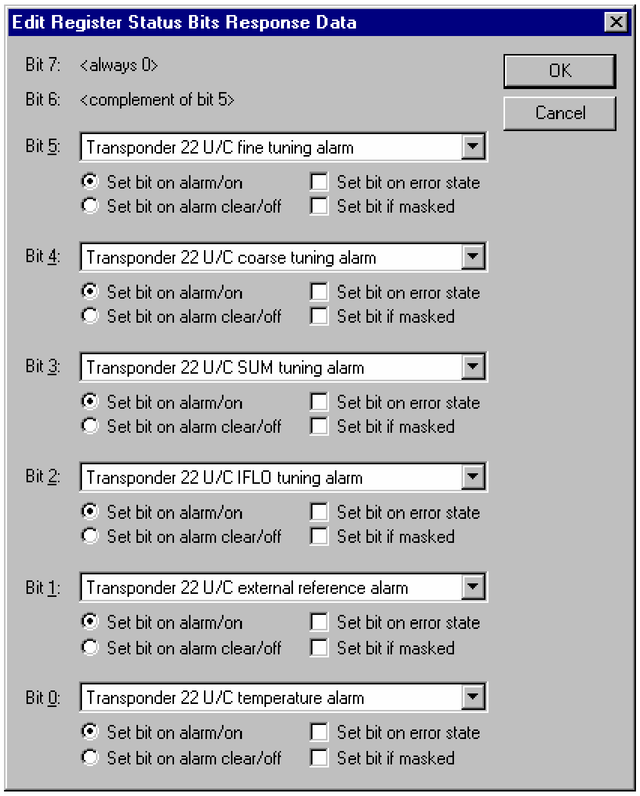
- Error state:

Enter the data string to use if the register is in its error state. If you do not specify a string for the error state, an empty string will be used if the register is in the error state.

Register Status Bits Response Data

Register status bit response data let you represent the ON/OFF state of up to six registers in a single byte. Because the SABus protocol only allows printable characters, only the six least significant bits (bits 0-5) of a byte can be used as status bits. Bit 7 is always 0, and bit 6 must be the complement of bit 5, that is, if bit five is 0, bit 6 must be 1, and vice versa.

The New Register Status Bits Response Data Dialog



Edit Register Status Bits Response Data

Bit 7: <always 0>

Bit 6: <complement of bit 5>

Bit 5: Transponder 22 U/C fine tuning alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

Bit 4: Transponder 22 U/C coarse tuning alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

Bit 3: Transponder 22 U/C SUM tuning alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

Bit 2: Transponder 22 U/C IFLO tuning alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

Bit 1: Transponder 22 U/C external reference alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

Bit 0: Transponder 22 U/C temperature alarm
(●) Set bit on alarm/on ( ) Set bit on alarm clear/off ☐ Set bit on error state ☐ Set bit if masked

- Bit 7/6:

You cannot change the behaviour of bits 6 and 7.

- Bit 5/4/3/2/1/0:

Select the register whose state you want this bit to represent. If you do not want to use a register for this bit, select <always 0> or <always 1>. The bit will then always be set to 0 or 1, respectively.

If you selected a register, you can set the following options:

Set bit on alarm/on:
Choose this option if you want the bit to be set (1) if the register is in the ON state, and cleared (0) if it is in the OFF state.

Set bit on alarm clear/off:
Choose this option if you want the bit to be cleared (0) if the register is in the ON state, and set (1) if it is in the OFF state.

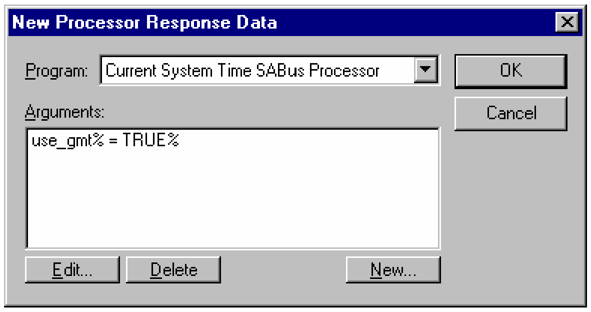
Set bit on error state:
Check this box if you want the bit to be set (1) if the register is in its error state. If you leave this box blank, the bit will be cleared (0) if the register is in the error state.

Set bit if masked:
Check this box if you want the bit to be set (1) if the register is masked. If you leave this box blank, the bit will be cleared (0) if the register is masked.

Processor Response Data

If none of the other types of response data fit your needs, you can use a processor response data object. The response data section of a processor response data object is constructed within an SCL program. The program can collect data from various registers and parameters and perform any necessary calculation or formatting, or other processing. See *Programs for Sources and SABus Response Data* in the *SCL Language Reference* for details.

The New Processor Response Data Dialog



Program: Current System Time SABus Processor

Arguments:

use_gmt% = TRUE%

- Program:

Select the program that will provide the data.

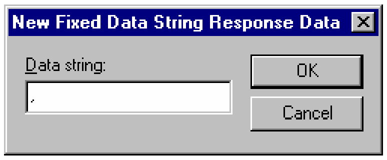
- Arguments:

Specify the program arguments. See *Specifying Arguments for SCL Programs* on page 67 for more information.

Fixed Data String Response Data

Use fixed data string response data objects to place fixed data, like separators or special markers, into your response.

The New Fixed Data String Response Data Dialog



Data string: ,

- Data string:

Enter the data.

SABus Commands

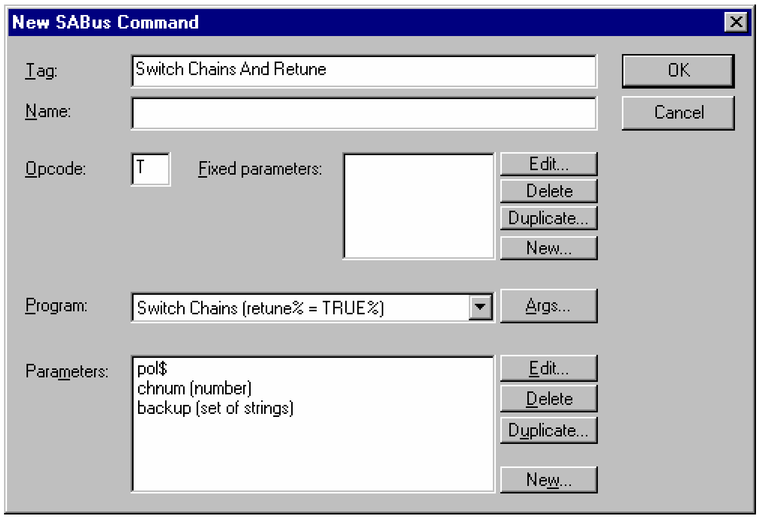
Commands are SABus Queries that rely completely on an SCL program to do all processing, and to construct a response. Commands can have variable parameters beyond the fixed parameters: When a command is received, U.P.M.A.C.S. decodes the additional parameters according to your specifications and places their values in SCL variables when it launches the command program. See *Programs for SABus Commands* in the *SCL Programming Reference* for details.

SABus commands will normally fail with the "USR" error message if the remote system on the SABus is not signed on. If you want a command to be available to systems that are not signed on, you must configure the program it executes to allow execution without signing on. See *SCL Programs* on page 58 for details.

SABus commands are primarily designed to implement controls that can be executed via an SABus request. To provide such a control, simply write an SCL program that performs the control action and returns an acknowledge or error response, as appropriate. Contrary to the programs used for control buttons, programs that are used for SABus commands should *not* require a confirmation of the action. Any confirmation should be the responsibility of the system that sent the command.

Another way of using SABus commands is to provide information to the remote system. This is usually done using SABus queries, but in certain situations, a query may not be suitable because it can only take fixed parameters. If you need to have a request with variable parameters that returns information, you can create an SABus command whose SCL program will not perform any actions, but simply construct an appropriate response and send it. If you intend to use a command in this way, make sure to configure the program you are going to use to allow execution without signing on, or the command will be rejected unless the client that sent it is currently signed on. See *SCL Programs* on page 58 for details.

The New SABus Command Dialog



Tag: Switch Chains And Retune
Name:
Opcode: T
Program: Switch Chains (retune% = TRUE%)
Parameters:
pol$
chnum (number)
backup (set of strings)

- Tag:
Enter the tag by which the command is identified. Each SABus request, whether command or query, must have a unique tag.

- Name:
Enter the name of the command. Leave this field blank if you want to use the tag as name.

- Opcode:
Enter the opcode of the command. The opcode must be a single character.

The opcode of SABus requests does not have to be unique. However, you must make sure that the opcode together with the fixed parameters uniquely identify the request.

- Fixed parameters:
Lists all the fixed parameters. Use the buttons to create, delete, or edit the parameters. Use the "Duplicate…" button to duplicate the selected parameter.

The list shows the parameters in the order they must appear in the request packet. You can change the order of the parameters by grabbing them with the mouse and dragging them to a new position.

The fixed parameters are fixed strings, not regular expressions. They must appear in the packet exactly as shown in this list, without any separation characters. You must make sure that the fixed parameters together with the opcode uniquely identify the request.

- Program:

Select the SCL programs you want to be executed when the command is received. The program arguments are shown in parentheses after the name of the program, but you only select the program from the lists, not the arguments. To change the arguments, use the "Args…" button. See *Specifying Arguments for SCL Programs* on page 67 for a description of the Edit Program Arguments dialog.

- Parameters:

Lists all the variable parameters by their variable names. Use the buttons to create, delete, or edit the parameters. Use the "Duplicate…" button to duplicate the selected parameter.

The variable parameters always appear after the fixed parameters in the request packet. The list shows the parameters in the order they must appear. You can change the order of the parameters by grabbing them with the mouse and dragging them to a new position.

The variable parameters are *not* used to distinguish between SABus requests. You must make sure that the *fixed* parameters alone, together with the opcode, uniquely identify the request.

See *Variable Command Parameters* below for details.

Variable Command Parameters

U.P.M.A.C.S. uses parameter decoding objects to decode the data that follows the fixed parameters in a command packet. Each variable parameter consists of an SCL variable name and a method for extracting the variable's value from the request packet. If a parameter cannot be decoded or is missing, or if there is extra data in the request packet once all parameters have been decoded, the request is rejected with an appropriate error message that includes the number of the parameter. The parameters are numbered starting with the fixed parameters, followed by the variable parameters.

Note: String parameters that do not use a variable are not counted for the purposes of parameter numbering. See *String Parameters* below for details.

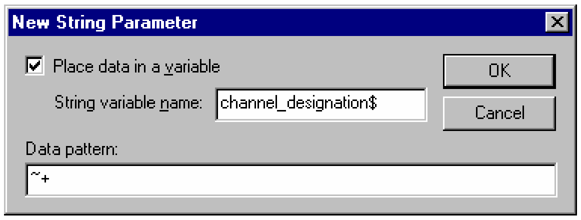
U.P.M.A.C.S. does not check that you do not use the same variable for two parameters. Make sure that you use each variable only once, and that you do not use any variables used for program arguments for parameters. If you use the same variable for more than one parameter, the variable will be set to *last* parameter in the list. If you use a variable that you also used for a program argument, the value you assigned in the program argument will be overwritten with the parameter value.

You cannot use array elements as parameter variables.

String Parameters

String parameters use a regular expression to parse a parameter from the request data. You can specify a string variable that will be set to the string that matched the expression. If you do not specify a variable, the parameter will not be skipped when assigning parameter numbers for error messages. This means that a string parameter that does not use a variable will have the same parameter number as the parameter that follows it. This is useful for specifying separator characters and parameter prefixes.

The New String Parameter Dialog



Place data in a variable: ✓
String variable name: channel_designation$
Data pattern: ~+

- Place data in a variable

Check this box to place the data that matches the pattern into a string variable so you can access it from within the program.

If you leave this check box blank, the parameter will have the same parameter number as the one that follows it when returning error messages. Leave this box blank for separators and parameter prefixes.

String variable name:
Enter the name of the SCL variable that will receive the data.

- Data pattern:

Enter a regular expression that the parameter must fit.

The regular expressions used for string parameters have a special feature. If you use an asterisk ("*"), plus sign, or range modifier on the *last* part of the expression, that part will be matched as often as possible. Thus, the expression "A#*" will match the letter "A" and all decimal digits that follow. The parts other than the last part behave as in any other regular expression, that is, if you specify a modifier they will be matched as few times as possible. ".*," will match all characters up to the first comma (the smallest possible match), not all characters up to the last comma.

See *Appendix A: Regular Expressions* on page 171 for details.

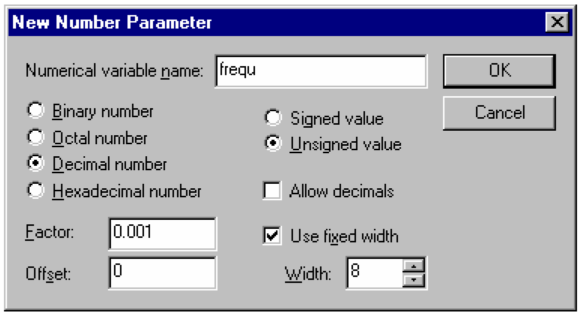
Number Parameters

Number parameters read a binary, octal, decimal, or hexadecimal number from the data and place the result in a numerical variable. You can provide a factor and an offset to be applied to the number after decoding it. The value written to the variable is calculated from the number read from the parameter list as follows:

value = number · factor + offset

To use the number unaltered, specify a factor of 1 and an offset of 0.

The New Number Parameter Dialog


New Number Parameter

Numerical variable name: frequ

( ) Binary number
( ) Octal number
[✓ Decimal number]
( ) Hexadecimal number

( ) Signed value
[✓ Unsigned value]

[ ] Allow decimals

[✓ Use fixed width]

Factor: 0.001

Offset: 0

Width: 8

- Numerical variable name:

Enter the name of the SCL variable that will receive the result.

- Binary/Octal/Decimal/Hexadecimal number:

Select the base in which the number appears in the request packet

- Factor:

Enter the factor with which the number in the request packet is to be multiplied before writing it to the variable. The factor is applied before the offset.

- Offset:

Enter the offset that is to be added to the number in the request packet before writing it to the variable. The offset is applied after the factor.

- Signed value:

Choose this option to allow a plus or a minus sign before the number.

- Unsigned value:

Choose this option to disallow plus and minus signs. If you choose this option, the number will always be positive or zero (unless you specify a negative factor).

- Allow decimals

Check this box to allow a decimal point. To use an implied decimal point, leave this box blank and specify a factor that will scale the number to have the correct number of decimals. E.g., to imply 3 decimal digits, use a factor of 0.001.

- Use fixed width:

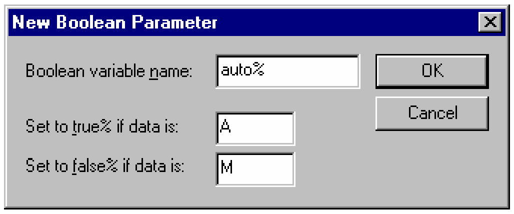
If you check this box, the number (including sign and decimal point) must be exactly the specified number of characters. Any digits outside this width will be treated as belonging to the next parameter, and an error will be returned if the number is shorter than the specified width.

Width:
Specify the width to use here.

Boolean Parameters

Boolean parameters set a Boolean variable to true% or false%, depending on which one of two strings is found in the request data. An error is returned if neither of the two strings is found.

The New Boolean Parameter Dialog



Boolean variable name: auto%

Set to true% if data is: A

Set to false% if data is: M

- Boolean variable name:

Enter the name of the SCL variable that will be set to true% or false%.

- Set to true% if data is:

Enter the string that will cause the variable to be set to true%. The string you enter here must be matched exactly; it is not a regular expression.

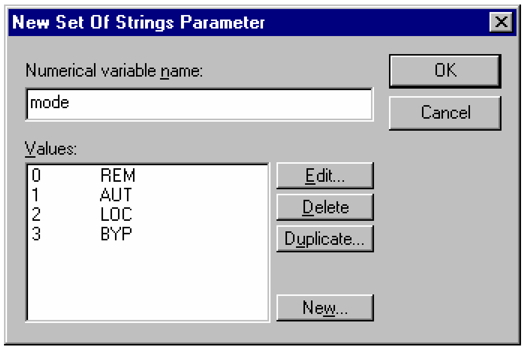
- Set to false% if data is:

Enter the string that will cause the variable to be set to false%. The string you enter here must be matched exactly; it is not a regular expression.

Set of Strings Parameters

Set of strings parameters set a numerical variable to different values depending on which of a number of strings is found in the request data. An error is returned if none of the strings is found.

The New Set Of Strings Parameter Dialog



Numerical variable name: mode

Values:

0	REM
1	AUT
2	LOC
3	BYP

- Numerical variable name:

Enter the name of the SCL variable that will receive the result.

- Values:

Shows a list of values and the strings that correspond to them. If the string is found, the variable will be set to the corresponding value. Use the buttons to create, delete, duplicate, or edit strings.

Make sure the strings uniquely identify the value. If you use the same string twice, or if there is a doubt about which string is present in the request packet, U.P.M.A.C.S. will use the first value that matches. U.P.M.A.C.S. cannot distinguish strings simply by length. If you have two strings "R" and "REM", U.P.M.A.C.S. might interpret the characters "REM" in the response as the value "R" followed by other parameters.

The strings you enter here must be matched exactly; they are not regular expressions.

Bits Parameters

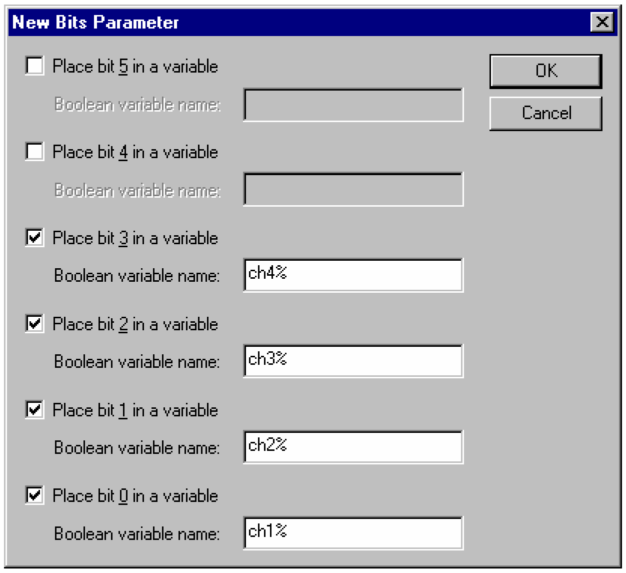
Bits parameters set up to six Boolean variables to true% or false%, depending on the values of the six least significant bits (bits 0-5) of a single data byte. Because the SABus protocol only allows printable characters, bits 6 and 7 cannot be used to hold data.

If the parameters represent the desired state of six separate settings, only use this type of parameter if all the states are needed at the same time. If the six bits represent six different settings that can be changed individually, you should provide a separate command to configure each of them. Otherwise, if the remote system wants to change only a single one of the settings, it must determine the current state of the remaining five bits and set them correctly in addition to specifying the changed setting.

You can also use this type of parameter to allow the remote system to specify what equipment to apply a particular setting to. You can, for example, provide a command that allows the system to switch one or more transfer switches at the same time. You could then use a bits parameter to specify which switches to switch. The remaining switches should then remain unaffected.

Even though bits 6 and 7 are not used, they cannot be freely set because the SABus protocol only allows printable characters (ASCII $20-$7E). U.P.M.A.C.S. will not accept requests that include non-printable characters in the parameter section of their data packet. The easiest way to ensure that all resulting characters are printable when setting bits in a byte is to set bit 7 to 0, and to set bit 6 to the complement of bit 5 (set bit 6 to 1 if bit 5 is 0 and vice versa). You can use another method if you like; U.P.M.A.C.S. does not enforce this exact behaviour, as long as the resulting byte represents a printable character.

The New Bits Parameter Dialog


New Bits Parameter

- [ ] Place bit 5 in a variable
  Boolean variable name:
- [ ] Place bit 4 in a variable
  Boolean variable name:
- [x] Place bit 3 in a variable
  Boolean variable name: ch4%
- [x] Place bit 2 in a variable
  Boolean variable name: ch3%
- [x] Place bit 1 in a variable
  Boolean variable name: ch2%
- [x] Place bit 0 in a variable
  Boolean variable name: ch1%

- Place bit 5/4/3/2/1/0 in a variable:

Check this box to assign a variable to the corresponding bit. If you leave the box blank, the corresponding bit will be ignored. If you want to enforce a bit being 0 or 1, you must place it in a variable and check it in the SCL program.

Boolean variable name:

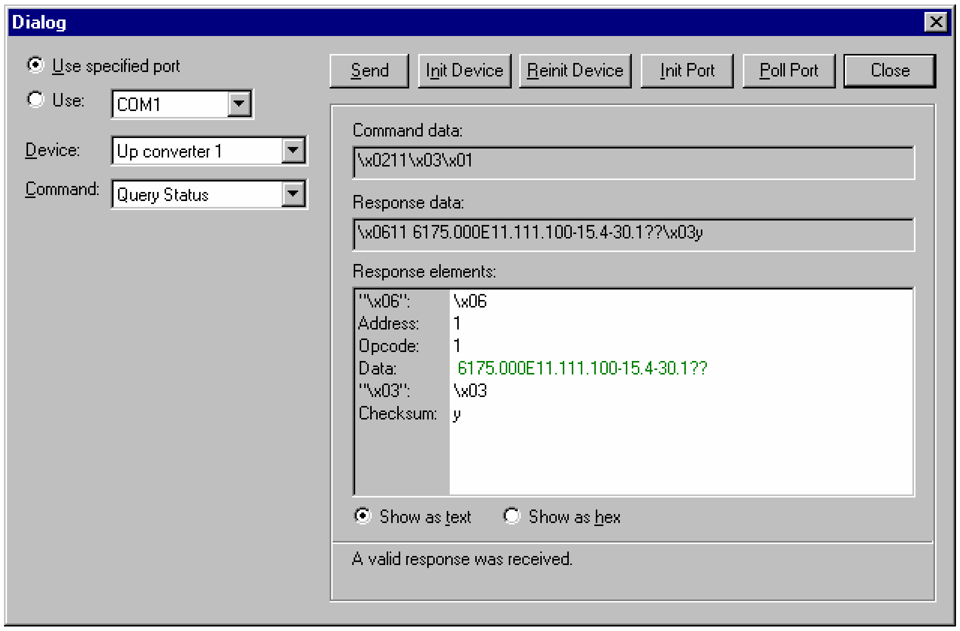
Enter the name of the SCL variable to use for the bit. The variable will be set to true% if the bit is set (1), and to false% if it is clear (0). Make sure to use a different variable for each bit.

TOOLS

Testing Serial Ports

The New Serial Port dialog provides functionality for testing the port settings and the devices attached to the port. You can send individual commands to equipment, or you can test the initialization and polling sequences. To pop up the Test Serial Port dialog, press the "Test…" button.

The Test Serial Port Dialog



[✓ Use specified port]
Use: COM1
Device: Up converter 1
Command: Query Status

Command data:

```
\x0211\x03\x01
```

Response data:

```
\x0611 6175.000E11.111.100-15.4-30.1??\x03y
```

Response elements:

```
"\x06":   \x06
Address:  1
Opcode:   1
Data:     6175.000E11.111.100-15.4-30.1??
"\x03":   \x03
Checksum: y
```

[✓ Show as text] Show as hex

A valid response was received.

- Use specified port:
Select this button if the hardware serial port to which the equipment is connected is the same port you specified in the New Port dialog.
- Use:
Select this radio button if the hardware serial port to which the equipment is connected is different than the port you specified in the New Port dialog. Select the hardware port you wish to use from the list. This is useful if you are developing a station to be used on a different computer, and the computer you are developing on does not have the required serial port.
- Device:
If you want to test a specific device, select that device here.
- Command:
If you want to test a specific command, select that command here. Specify values for all the command parameters immediately below. (In the sample dialog, the command has no parameters.)

If you do not specify a value for a particular command parameter (i.e., leave the field blank), a default value will be used. The default value for bistate parameters is OFF, the default value for digital and analog parameters is 0, and the default value for string parameters is an empty string.

Show as decimal / Show as hex (not shown):
If the command has any parameters of type digital that don't have value names, you can select the way you want to enter the parameter values here. Select "Show as decimal" to enter the values in decimal, select "Show as hex" to enter the values in hexadecimal.

Show as text / Show as hex (not shown):
If the command has any parameters of type string, you can select the way you want to enter the parameter values here. See *Appendix B: Entering Binary Data* on page 179 for details on entering binary data.

- Command data:

Shows the command data that the command sent. The command data is shown either as text or as hex values, whichever is selected below the "Response elements" box.

See *Appendix B: Entering Binary Data* on page 179 for details.

- Response data:

Shows the response data received from the device, if any. The response data is shown either as text or as hex values, whichever is selected below the "Response elements" box.

If a valid or error response was received, only the response will be shown in this field. Any extra data will be removed. If a timeout occurs, all data in the data buffer will be shown. Any data that does not fit into the data buffer size specified in the command will be thrown out.

See *Appendix B: Entering Binary Data* on page 179 for details.

- Response elements:

Shows the response data broken up into the different response elements in the command's template. Response data is shown in green, error codes are shown in red, and all other data is shown in black.

The response data is shown either as text or as hex values, whichever is selected below the "Response elements" box. See *Appendix B: Entering Binary Data* on page 179 for details.

- Show as text, Show as hex:

Select the way you want the command and response data to be displayed.

See *Appendix B: Entering Binary Data* on page 179 for details.

- The message field:

At the bottom of the dialog is a space for messages. A message saying whether a valid response was received or not will appear here when a command has been sent.

- The "Send" button:

Press this button to send the selected command to the device.

- The "Init Device" button:

Press this button to test the first-time initialization sequence of the selected device. The initialization sequence will be sent exactly as it would be by the Operate System when the port is first opened. This button is only avaliable if the device has a first-time or common initialization sequence.

- The "Reinit Device" button:

Press this button to test the reinitialization sequence of the selected device. The initialization sequence will be sent exactly as it would be by the Operate System when the device timed out. See This button is only avaliable if the device has a reinitialization or common initialization sequence.

- The "Init Port" button:

Press this button to test the first-time initialization sequence of all devices attached to the port. The initialization sequences will be sent exactly as they would be by the Operate System when the port is first opened. This button is only avaliable if at least one device device has a first-time or common initialization sequence.

- The "Poll Port" button:

Press this button to test the polling sequence of the port. The sequence will only be sent once, and not repeated.

- The "Close" button:

Press this button to close the dialog.

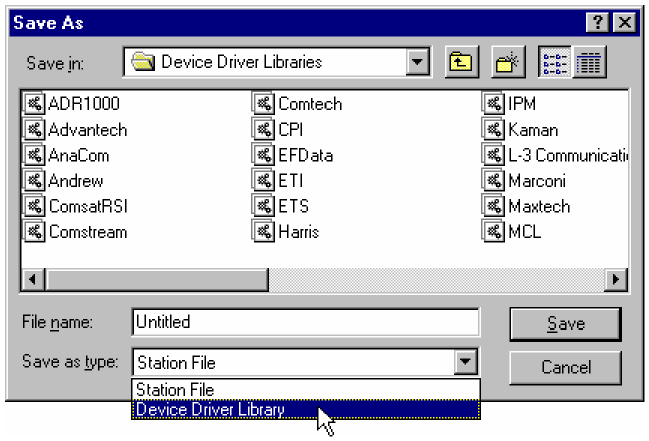
Device Driver Libraries

The U.P.M.A.C.S. Development System supports libraries of predefined device drivers. You can use driver libraries provided by UPMACS Communications, inc., or you can create your own libraries.

A device driver library is very similar to a station file, except that it contains only device drivers. You can import the device drivers into a station.

Creating and Editing Device Driver Libraries

To create a device driver library, choose "New…" from the file menu. This will create a new station. Add only device drivers to the station. When you save the new file, select "Device Driver Library" in the "Save as type" field:



The Save As dialog

You can save any station as a device driver library, as long as it contains only device drivers.

To edit a device driver library, load it by selecting "Open…" from the "File" menu, and choosing "Device Driver Libraries" in the "Files of type" field.

Exporting Device Drivers

Another way to create a device driver library is by exporting the device drivers of a station. This will create a library containing all device drivers in the station. To export device drivers, choose "Export Device Drivers…" from the "Special" menu. A dialog will pop up, asking you for the file name of the driver library.

Importing Device Drivers

To import device drivers into a station, choose "Import Device Drivers…" from the "Special" menu. A dialog will appear, asking you to select a driver library.

Once you have selected a library, the Import Device Drivers dialog appears. Select all the device drivers you want to import. To select more than one device driver, click while holding down the Shift or Ctrl keys.

Testing a Station File

To test a station file, you must save it and then load it in the U.P.M.A.C.S. Operate System. You can do both those things in one step by selecting "Save And Test Station" from the "Special" menu. This will save the station, activate or start the Operate System, and load the station into the Operate System.

Batch Processing of Registers

The U.P.M.A.C.S. Development System supports modifying, duplicating and deleting multiple registers in one operation. To manipulate batches of registers, select "Modify/Duplicate Registers…" from the "Special" menu.

The Modify/Duplicate Registers dialog will appear. There are four things you can do with this dialog:

- Finding tags:

You can find and select all registers whose tag contains a certain search text. Enter the text in the "Find" field and press the "Find All Tags" button.

- Modifying registers:

You can modify certain properties of all selected registers. Specify the changes you wish to make using the fields of the dialog, and press the "Replace All" button.

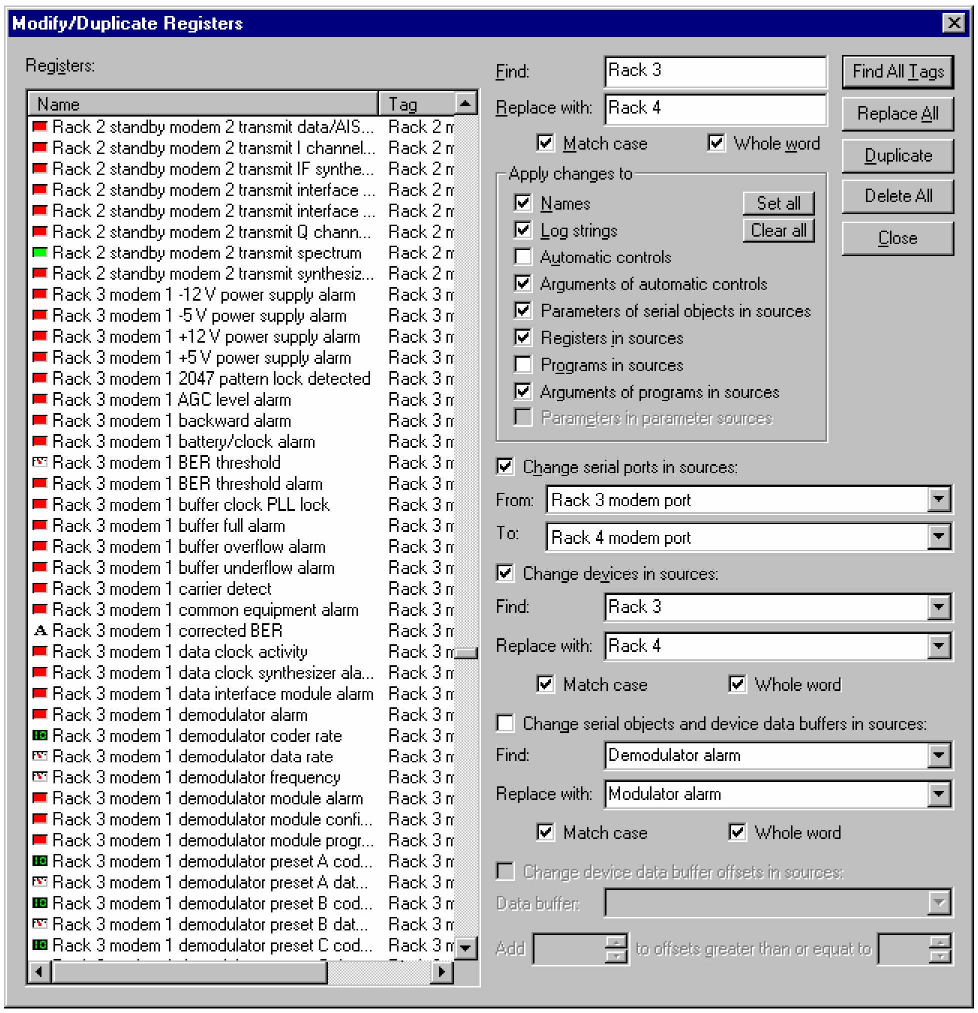
- Duplicating registers:

You can make modified duplicates of all selected registers. Specify the changes you wish to make using the fields of the dialog, and press the "Duplicate" button.

- Deleting registers:

You can delete all selected registers by pressing the "Delete All" button.

The Modify/Duplicate Registers Dialog



Modify/Duplicate Registers

Registers:

Name	Tag
Rack 2 standby modem 2 transmit data/AIS...	Rack 2 m
Rack 2 standby modem 2 transmit I channel...	Rack 2 m
Rack 2 standby modem 2 transmit IF synthe...	Rack 2 m
Rack 2 standby modem 2 transmit interface ...	Rack 2 m
Rack 2 standby modem 2 transmit interface ...	Rack 2 m
Rack 2 standby modem 2 transmit Q chann...	Rack 2 m
Rack 2 standby modem 2 transmit spectrum	Rack 2 m
Rack 2 standby modem 2 transmit synthesiz...	Rack 2 m
Rack 3 modem 1 -12 V power supply alarm	Rack 3 m
Rack 3 modem 1 -5 V power supply alarm	Rack 3 m
Rack 3 modem 1 +12 V power supply alarm	Rack 3 m
Rack 3 modem 1 +5 V power supply alarm	Rack 3 m
Rack 3 modem 1 2047 pattern lock detected	Rack 3 m
Rack 3 modem 1 AGC level alarm	Rack 3 m
Rack 3 modem 1 backward alarm	Rack 3 m
Rack 3 modem 1 battery/clock alarm	Rack 3 m
Rack 3 modem 1 BER threshold	Rack 3 m
Rack 3 modem 1 BER threshold alarm	Rack 3 m
Rack 3 modem 1 buffer clock PLL lock	Rack 3 m
Rack 3 modem 1 buffer full alarm	Rack 3 m
Rack 3 modem 1 buffer overflow alarm	Rack 3 m
Rack 3 modem 1 buffer underflow alarm	Rack 3 m
Rack 3 modem 1 carrier detect	Rack 3 m
Rack 3 modem 1 common equipment alarm	Rack 3 m
Rack 3 modem 1 corrected BER	Rack 3 m
Rack 3 modem 1 data clock activity	Rack 3 m
Rack 3 modem 1 data clock synthesizer ala...	Rack 3 m
Rack 3 modem 1 data interface module alarm	Rack 3 m
Rack 3 modem 1 demodulator alarm	Rack 3 m
Rack 3 modem 1 demodulator coder rate	Rack 3 m
Rack 3 modem 1 demodulator data rate	Rack 3 m
Rack 3 modem 1 demodulator frequency	Rack 3 m
Rack 3 modem 1 demodulator module alarm	Rack 3 m
Rack 3 modem 1 demodulator module confi...	Rack 3 m
Rack 3 modem 1 demodulator module progr...	Rack 3 m
Rack 3 modem 1 demodulator preset A cod...	Rack 3 m
Rack 3 modem 1 demodulator preset A dat...	Rack 3 m
Rack 3 modem 1 demodulator preset B cod...	Rack 3 m
Rack 3 modem 1 demodulator preset B dat...	Rack 3 m
Rack 3 modem 1 demodulator preset C cod...	Rack 3 m

Find: Rack 3

Replace with: Rack 4

[✓] Match case [✓] Whole word

Apply changes to

- [✓] Names
- [✓] Log strings
- [ ] Automatic controls
- [✓] Arguments of automatic controls
- [✓] Parameters of serial objects in sources
- [✓] Registers in sources
- [ ] Programs in sources
- [✓] Arguments of programs in sources
- [ ] Parameters in parameter sources

[✓] Change serial ports in sources:

From: Rack 3 modem port

To: Rack 4 modem port

[✓] Change devices in sources:

Find: Rack 3

Replace with: Rack 4

[✓] Match case [✓] Whole word

[ ] Change serial objects and device data buffers in sources:

Find: Demodulator alarm

Replace with: Modulator alarm

[✓] Match case [✓] Whole word

[ ] Change device data buffer offsets in sources:

Data buffer:

Add to offsets greater than or equal to

- Registers:
Select the registers that you want to process. To select multiple registers, click while holding down the Shift or Ctrl key.

The list shows both name and tag of the register. You can resize the two columns by dragging the edges of the column headers with the mouse. Click on the “Name” header to sort the registers by name, click on the “Tag” header to sort them by tag.

- Find:
Enter the search text here. The search text is used for finding tags, and for modifying and duplicating registers. When modifying or duplicating registers, the search text is replaced with the replace text for everything specified in the “Apply changes to” rectangle. In addition, if you are duplicating registers, the search text in the tag of the original register is replaced with the replace text to create the tag of the new register. This means that the tags of all registers you wish to duplicate must contain the search text.

- Replace with:
Enter the text with which you want to replace the search text here. The replace text is used for modifying and duplicating registers. When modifying or duplicating registers, the search text is replaced with the replace text for everything specified in the “Apply changes to” rectangle. In addi-

tion, if you are duplicating registers, the search text in the tag of the original register is replaced with the replace text to create the tag of the new register.

- Match case:

Check this box to find only text that matches the capitalization of the search text. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

- Whole word:

Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- The "Apply changes to" rectangle:

When modifying or duplicating registers, the search text is replaced with the replace text for everything that you specify in this box. You can select any number of the following options:

Names:
Change the name of the register. If you are duplicating registers, and you do not check this box, the new registers will have the same name as the originals, and you will have a hard time telling them apart. This box should usually be checked when duplicating registers.

Log strings:
Change all custom log strings of the register.

Automatic controls:
Change all automatic controls defined for the register. If the tag of the SCL program used for the automatic control contains the search text, the search text is replaced with the replace text to create a new tag. The control is modified to use the SCL program that has the new tag instead.

Arguments of automatic controls:
Change all argument values of automatic controls defined for the register. All text will be replaced in string arguments, numbers will be replaced in numerical arguments, and the values "true%" and "false%" will be replaced in Boolean arguments.

Parameters of serial objects in sources:
Change all parameters of serial data objects used by any serial data object sources. All text will be replaced in string parameters, numbers will be replaced in digital and analog parameters, and the values "ON" and "OFF" will be replaced in bistate parameters.

Registers in sources:
Change the registers used by any summary sources. If the tag of any register used by a source contains the search text, the search text is replaced with the replace text to create a new tag. The source is modified to use the register that has the new tag instead.

Programs in sources:
Change the programs used by any summary sources. If the tag of any program used by a source contains the search text, the search text is replaced with the replace text to create a new tag. The source is modified to use the program that has the new tag instead.

Arguments of programs in sources:
Change all argument values of programs used by any summary. All text will be replaced in string arguments, numbers will be replaced in numerical arguments, and the values "true%" and "false%" will be replaced in Boolean arguments.

Parameters in Parameter Sources:
This field is used for legacy parameter sources only. See *Batch Processing of Registers* in *Appendix E: Legacy Objects* on page 218 for details.

The "Set All" button:
Press this button to set all the check boxes.

The "Clear All" button:
Press this button to clear all the check boxes.

- Change serial ports in sources:

Check this box to modify any serial ports used by register sources. Select the ports you want to replace in the "From" and "To" fields below.

From:
Select the port you want to replace here.

To:
Select the port with which you want to replace the port.

- Change devices in sources:

Check this box to modify any devices used by register sources. Specify a search and replace string for the device's tag in the "Find" and "Replace with" fields below.

Find:
Enter the search text here. The search text in the tag of the original device is replaced with the replace text to create a new tag. The source is modified to use the device that has the new tag instead.

Replace with:
Enter the replace text here. The search text in the tag of the original device is replaced with the replace text to create a new tag. The source is modified to use the device that has the new tag instead.

Match case:
Check this box to find only text that matches the capitalization of the search string. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

Whole word:
Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- Change serial objects and device data buffers in sources:

Check this box to change any serial data objects used by serial data object sources. Specify a search and replace string for the data object's tag in the "Find" and "Replace with" fields below.

Find:
Enter the search text here. The search text in the tag of the original data object is replaced with the replace text to create a new tag. The source is modified to use the data object that has the new tag instead.

Replace with:
Enter the replace text here. The search text in the tag of the original data object is replaced with the replace text to create a new tag. The source is modified to use the data object that has the new tag instead.

Match case:
Check this box to find only text that matches the capitalization of the search string. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

Whole word:
Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- Change device data buffer offsets in sources:

These fields are used for legacy device sources only. See *Batch Processing of Registers* in *Appendix E: Legacy Objects* on page 218 for details.

- The "Find All Tags" button:

Press this button to select all registers whose tags contain the search text.

- The "Replace All" button:

Press this button to apply the changes you specified to all selected registers.

- The "Duplicate" button:

Press this button to create modified duplicates of all selected registers. The tags of the new registers will be created by replacing the search string in the original register's tag with the replace string. For that reason, the tag of all registers you are duplicating must contain the search text.

All changes you specified in the dialog will be applied to the new registers. If you did not check the "Names" box in the "Apply changes to" rectangle, the new registers will have the same name as the old ones, and it will be difficult to tell them apart. You should usually check the "Names" box.

- The "Delete All" button:

Press this button to delete all selected registers. This will also remove references to those registers from all summary sources that use them.

- The "Close" button:

Press this button to close the Modify/Duplicate Registers dialog.

Batch Processing of SCL Programs

The U.P.M.A.C.S. Development System supports modifying, duplicating and deleting multiple SCL programs in one operation. To manipulate batches of SCL programs, select "Modify/Duplicate SCL programs…" from the "Special" menu.

The Modify/Duplicate Programs dialog will appear. There are four things you can do with this dialog:

- Finding tags:

You can find and select all SCL programs whose tag contains a certain search text. Enter the text in the "Find" field and press the "Find All Tags" button.

- Modifying SCL programs:

You can modify certain properties of all selected SCL programs. Specify the changes you wish to make using the fields of the dialog, and press the "Replace All" button.

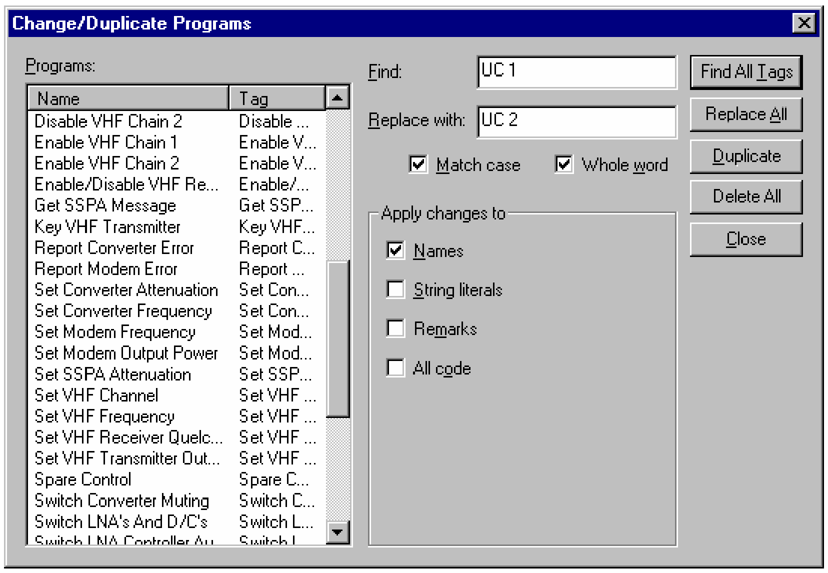
- Duplicating SCL programs:

You can make modified duplicates of all selected SCL programs. Specify the changes you wish to make using the fields of the dialog, and press the "Duplicate" button.

- Deleting SCL programs:

You can delete all selected SCL programs by pressing the "Delete All" button.

The Modify/Duplicate Programs Dialog


Change/Duplicate Programs

Programs:

Name	Tag
Disable VHF Chain 2	Disable ...
Enable VHF Chain 1	Enable V...
Enable VHF Chain 2	Enable V...
Enable/Disable VHF Re...	Enable/...
Get SSPA Message	Get SSP...
Key VHF Transmitter	Key VHF...
Report Converter Error	Report C...
Report Modem Error	Report ...
Set Converter Attenuation	Set Con...
Set Converter Frequency	Set Con...
Set Modem Frequency	Set Mod...
Set Modem Output Power	Set Mod...
Set SSPA Attenuation	Set SSP...
Set VHF Channel	Set VHF ...
Set VHF Frequency	Set VHF ...
Set VHF Receiver Quelc...	Set VHF ...
Set VHF Transmitter Out...	Set VHF ...
Spare Control	Spare C...
Switch Converter Muting	Switch C...
Switch LNA's And D/C's	Switch L...
Switch LNA Controller Au...	Switch L...

Find: UC 1

Replace with: UC 2

[✓] Match case [✓] Whole word

Apply changes to

[✓] Names

[ ] String literals

[ ] Remarks

[ ] All code

- Programs:

Select the SCL programs that you want to process. To select multiple programs, click while holding down the Shift or Ctrl key.

The list shows both name and tag of the program. You can resize the two columns by dragging the edges of the column headers with the mouse. Click on the “Name” header to sort the programs by name, click on the “Tag” header to sort them by tag.

- Find:

Enter the search text here. The search text is used for finding tags, and for modifying and duplicating programs. When modifying or duplicating programs, the search text is replaced with the replace text for everything specified in the “Apply changes to” rectangle. In addition, if you are duplicating programs, the search text in the tag of the original program is replaced with the replace text to create the tag of the new program. This means that the tags of all programs you wish to duplicate must contain the search text.

- Replace with:

Enter the text with which you want to replace the search text here. The replace text is used for modifying and duplicating programs. When modifying or duplicating programs, the search text is replaced with the replace text for everything specified in the “Apply changes to” rectangle. In addition, if you are duplicating programs, the search text in the tag of the original program is replaced with the replace text to create the tag of the new program.

- Match case:

Check this box to find only text that matches the capitalization of the search text. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

- Whole word:

Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- The “Apply changes to” rectangle:

When modifying or duplicating SCL programs, the search text is replaced with the replace text for everything that you specify in this box. You can select any number of the following options:

Names:
Change the name of the SCL program. If you are duplicating programs, and you do not check this box, the new programs will have the same name as the originals, and you will have a hard time telling them apart. This box should usually be checked when duplicating programs.

String literals:
Change all string literals in the program's code. String literals are enclosed in double quotes (`""`).

Remarks:
Change all remarks (`REM` statements) in the program's code. Remarks are not used by U.P.M.A.C.S. and are intended for explanatory notes in the program code.

All code:
Change all program code, even outside string literals and remarks. If you check this box, the "String literals" and "Remarks" options are superfluous.

- The "Find All Tags" button:

Press this button to select all SCL programs whose tags contain the search text.

- The "Replace All" button:

Press this button to apply the changes you specified to all selected programs.

- The "Duplicate" button:

Press this button to create modified duplicates of all selected programs. The tags of the new SCL programs will be created by replacing the search string in the original program's tag with the replace string. For that reason, the tag of all SCL programs you are duplicating must contain the search text.

All changes you specified in the dialog will be applied to the new SCL programs. If you did not check the "Names" box in the "Apply changes to" rectangle, the new programs will have the same name as the old ones, and it will be difficult to tell them apart. You should usually check the "Names" box.

- The "Delete All" button:

Press this button to delete all selected programs.

- The "Close" button:

Press this button to close the Modify/Duplicate Programs dialog.

Batch Processing of SABus Requests

The U.P.M.A.C.S. Development System supports modifying, duplicating and deleting multiple SABus requests in one operation. To manipulate batches of requests, select "Modify/Duplicate SABus Requests…" from the "Special" menu.

The Modify/Duplicate SABus Requests dialog will appear. There are four things you can do with this dialog:

- Finding tags:

You can find and select all requests whose tag contains a certain search text. Enter the text in the "Find" field and press the "Find All Tags" button.

- Modifying requests:

You can modify certain properties of all selected requests. Specify the changes you wish to make using the fields of the dialog, and press the "Replace All" button.

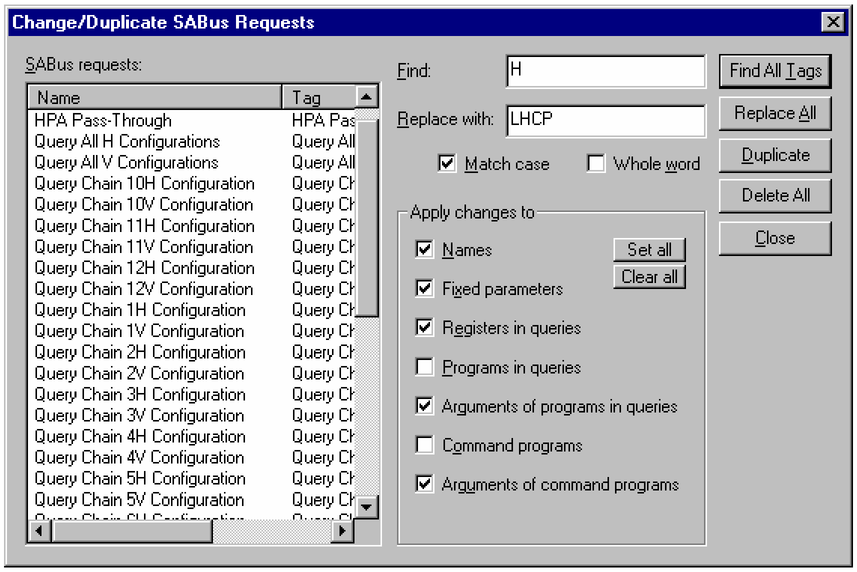
- Duplicating requests:

You can make modified duplicates of all selected requests. Specify the changes you wish to make using the fields of the dialog, and press the "Duplicate" button.

- Deleting requests:

You can delete all selected requests by pressing the "Delete All" button.

The Modify/Duplicate SABus Requests Dialog



Change/Duplicate SABus Requests

SABus requests:

Name	Tag
HPA Pass-Through	HPA Pas
Query All H Configurations	Query All
Query All V Configurations	Query All
Query Chain 10H Configuration	Query Ch
Query Chain 10V Configuration	Query Ch
Query Chain 11H Configuration	Query Ch
Query Chain 11V Configuration	Query Ch
Query Chain 12H Configuration	Query Ch
Query Chain 12V Configuration	Query Ch
Query Chain 1H Configuration	Query Ch
Query Chain 1V Configuration	Query Ch
Query Chain 2H Configuration	Query Ch
Query Chain 2V Configuration	Query Ch
Query Chain 3H Configuration	Query Ch
Query Chain 3V Configuration	Query Ch
Query Chain 4H Configuration	Query Ch
Query Chain 4V Configuration	Query Ch
Query Chain 5H Configuration	Query Ch
Query Chain 5V Configuration	Query Ch

Find: H

Replace with: LHCP

[✓] Match case [ ] Whole word

Apply changes to

[✓] Names
[✓] Fixed parameters
[✓] Registers in queries
[ ] Programs in queries
[✓] Arguments of programs in queries
[ ] Command programs
[✓] Arguments of command programs

- SABus requests:

Select the requests that you want to process. To select multiple requests, click while holding down the Shift or Ctrl key.

The list shows both name and tag of the requests. You can resize the two columns by dragging the edges of the column headers with the mouse. Click on the "Name" header to sort the requests by name, click on the "Tag" header to sort them by tag.

- Find:

Enter the search text here. The search text is used for finding tags, and for modifying and duplicating requests. When modifying or duplicating requests, the search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle. In addition, if you are duplicating requests, the search text in the tag of the original request is replaced with the replace text to create the tag of the new request. This means that the tags of all requests you wish to duplicate must contain the search text.

- Replace with:

Enter the text with which you want to replace the search text here. The replace text is used for modifying and duplicating requests. When modifying or duplicating requests, the search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle. In addition, if you are duplicating requests, the search text in the tag of the original request is replaced with the replace text to create the tag of the new request.

- Match case:

Check this box to find only text that matches the capitalization of the search text. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

- Whole word:

Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- The "Apply changes to" rectangle:

When modifying or duplicating parameters, the search text is replaced with the replace text for everything that you specify in this box. You can select any number of the following options:

Names:
Change the name of the parameter. If you are duplicating parameters, and you do not check this box, the new parameters will have the same name as the originals, and you will have a hard time telling them apart. This box should usually be checked when duplicating parameters.

Fixed parameters:
Change the fixed parameters of all requests.

Registers in queries:
Change the registers used by any response data objects in the queries. If the tag of any register used by a query contains the search text, the search text is replaced with the replace text to create a new tag. The query is modified to use the register that has the new tag instead.

Programs in queries:
Change the programs used by any processor response data objects in queries. If the tag of any program used by a query contains the search text, the search text is replaced with the replace text to create a new tag. The query is modified to use the program that has the new tag instead.

Arguments of programs in queries:
Change all argument values of programs used by any processor response data objects in queries. All text will be replaced in string arguments, numbers will be replaced in numerical arguments, and the values "true%" and "false%" will be replaced in Boolean arguments.

Command programs:
Change the programs used by commands. If the tag of any program used by a command contains the search text, the search text is replaced with the replace text to create a new tag. The command is modified to use the program that has the new tag instead.

Arguments of command programs:
Change all argument values of programs used by commands. All text will be replaced in string arguments, numbers will be replaced in numerical arguments, and the values "true%" and "false%" will be replaced in Boolean arguments.

The "Set All" button:
Press this button to set all the check boxes.

The "Clear All" button:
Press this button to clear all the check boxes.

- The "Find All Tags" button:

Press this button to select all requests whose tags contain the search text.

- The "Replace All" button:

Press this button to apply the changes you specified to all selected requests.

- The "Duplicate" button:

Press this button to create modified duplicates of all selected requests. The tags of the new requests will be created by replacing the search string in the original request's tag with the replace string. For that reason, the tag of all requests you are duplicating must contain the search text.

All changes you specified in the dialog will be applied to the new requests. If you did not check the "Names" box in the "Apply changes to" rectangle, the new requests will have the same name as the old ones, and it will be difficult to tell them apart. You should usually check the "Names" box.

- The "Delete All" button:

Press this button to delete all selected requests.

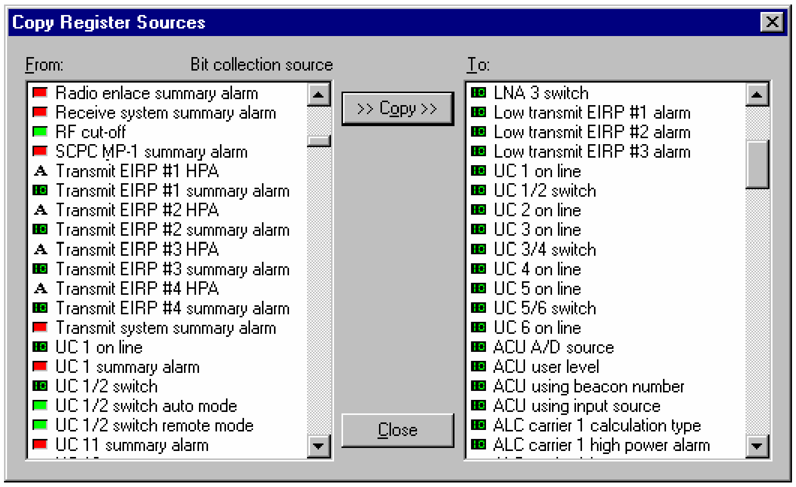
- The "Close" button:

Press this button to close the Modify/Duplicate Requests dialog.

Transferring Register Sources Between Registers

You can copy register sources from one register to one or more others. To do this, select "Copy Register Sources…" from the "Special" menu. The Copy Register Sources dialog will appear.

The Copy Register Sources Dialog



From list:
- Radio enlace summary alarm
- Receive system summary alarm
- RF cut-off
- SCPC MP-1 summary alarm
- Transmit EIRP #1 HPA
- Transmit EIRP #1 summary alarm
- Transmit EIRP #2 HPA
- Transmit EIRP #2 summary alarm
- Transmit EIRP #3 HPA
- Transmit EIRP #3 summary alarm
- Transmit EIRP #4 HPA
- Transmit EIRP #4 summary alarm
- Transmit system summary alarm
- UC 1 on line
- UC 1 summary alarm
- UC 1/2 switch
- UC 1/2 switch auto mode
- UC 1/2 switch remote mode
- UC 11 summary alarm

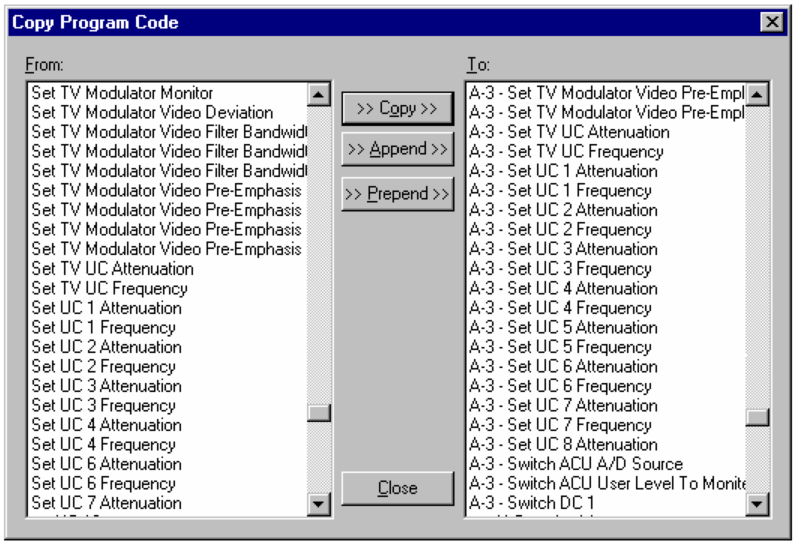
To list:
- LNA 3 switch
- Low transmit EIRP #1 alarm
- Low transmit EIRP #2 alarm
- Low transmit EIRP #3 alarm
- UC 1 on line
- UC 1/2 switch
- UC 2 on line
- UC 3 on line
- UC 3/4 switch
- UC 4 on line
- UC 5 on line
- UC 5/6 switch
- UC 6 on line
- ACU A/D source
- ACU user level
- ACU using beacon number
- ACU using input source
- ALC carrier 1 calculation type
- ALC carrier 1 high power alarm

- From:
Select the register whose source you want to copy here. The type of the source will appear above the right side of the list of registers.
- To:
Select all the register or registers to which you want to copy the source. To select multiple registers, click while holding down the Shift or Ctrl key. This list only shows registers of the same kind as the register you selected under "From". If no register is selected in the "From" list, this list is empty.
- The "Copy" button:
Press this button to copy the source of the register you selected in the "From" list to the registers you selected in the "To" list.
- The "Close" button:
Press this button to close the Copy Register Sources dialog.

Transferring Code Between Programs

You can copy code from one SCL program to one or more others. To do this, select "Copy Program Code…" from the "Special" menu. The Copy Program Code dialog will appear.

The Copy Program Code Dialog



From:

Set TV Modulator Monitor
Set TV Modulator Video Deviation
Set TV Modulator Video Filter Bandwidt
Set TV Modulator Video Filter Bandwidt
Set TV Modulator Video Filter Bandwidt
Set TV Modulator Video Pre-Emphasis
Set TV Modulator Video Pre-Emphasis
Set TV Modulator Video Pre-Emphasis
Set TV Modulator Video Pre-Emphasis
Set TV UC Attenuation
Set TV UC Frequency
Set UC 1 Attenuation
Set UC 1 Frequency
Set UC 2 Attenuation
Set UC 2 Frequency
Set UC 3 Attenuation
Set UC 3 Frequency
Set UC 4 Attenuation
Set UC 4 Frequency
Set UC 6 Attenuation
Set UC 6 Frequency
Set UC 7 Attenuation

To:

A-3 - Set TV Modulator Video Pre-Empl
A-3 - Set TV Modulator Video Pre-Empl
A-3 - Set TV UC Attenuation
A-3 - Set TV UC Frequency
A-3 - Set UC 1 Attenuation
A-3 - Set UC 1 Frequency
A-3 - Set UC 2 Attenuation
A-3 - Set UC 2 Frequency
A-3 - Set UC 3 Attenuation
A-3 - Set UC 3 Frequency
A-3 - Set UC 4 Attenuation
A-3 - Set UC 4 Frequency
A-3 - Set UC 5 Attenuation
A-3 - Set UC 5 Frequency
A-3 - Set UC 6 Attenuation
A-3 - Set UC 6 Frequency
A-3 - Set UC 7 Attenuation
A-3 - Set UC 7 Frequency
A-3 - Set UC 8 Attenuation
A-3 - Switch ACU A/D Source
A-3 - Switch ACU User Level To Monit
A-3 - Switch DC 1

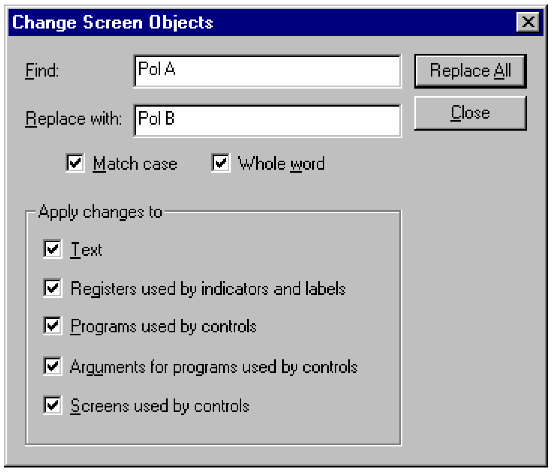
- From:
Select the program whose code you want to copy here.
- To:
Select the program or programs to which you want to copy the code. To select multiple programs, click while holding down the Shift or Ctrl key.
- The "Copy" button:
Press this button to copy the code of the program you selected in the "From" list to the programs you selected in the "To" list. The previous code of the programs will be lost.
- The "Append…" button:
Press this button to append the code of the program you selected in the "From" list to the end of the code of the programs you selected in the "To" list.
- The "Prepend…" button:
Press this button to prepend the code of the program you selected in the "From" list to the beginning of the code of the programs you selected in the "To" list.
- The "Close" button:
Press this button to close the Copy Program Code dialog.

Batch Processing of Graphic Objects

The Screen Editor supports modifying multiple graphic objects of different types in one operation. To manipulate batches of objects, select "Modify Screen Objects…" from the "Special" menu.

The Change Screen Objects will appear. It allows you to perform sophisticated search-and-replace operations on all currently selected graphic objects.

The Change Screen Objects Dialog



Find: Pol A

Replace with: Pol B

[✓] Match case [✓] Whole word

Apply changes to

[✓] Text

[✓] Registers used by indicators and labels

[✓] Programs used by controls

[✓] Arguments for programs used by controls

[✓] Screens used by controls

- Find:

Enter the search text here. The search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle.

- Replace with:

Enter the text with which you want to replace the search text here. The search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle.

- Match case:

Check this box to find only text that matches the capitalization of the search text. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

- Whole word:

Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- The "Apply changes to" rectangle:

When modifying or duplicating graphic objects, the search text is replaced with the replace text for everything that you specify in this box. You can select any number of the following options:

Text:
Change all text in all graphic objects, including captions of controls, and text, prefixes, and suffixes of indicators, and text of labels.

Registers used by indicators and labels:
Change the registers used by any indicator or label. If the tag of any screen used by an indicator or label contains the search text, the search text is replaced with the replace text to create a new tag. The object is modified to use the register that has the new tag instead.

Programs used by controls:
Change the programs used by any program controls. If the tag of any program used by a control contains the search text, the search text is replaced with the replace text to create a new tag. The control is modified to trigger the program that has the new tag instead.

Arguments for programs used by controls:
Change all argument values of programs used by any program controls. All text will be replaced in string arguments, numbers will be replaced in numerical arguments, and the values "true%" and "false%" will be replaced in Boolean arguments.

Screens used by controls:
Change the screens used by any screen and network screen controls.

Screen controls are modified as follows: If the tag of any program used by a screen control contains the search text, the search text is replaced with the replace text to create a new tag. The control is modified to show the screen that has the new tag instead.

For network screen controls, the search text is simply replaced in the screen name.

- The "Replace All" button:

Press this button to apply the changes you specified to all selected graphic objects.

- The "Close" button:

Press this button to close the Change Screen Objects dialog.

Deleting Off-Screen Objects

U.P.M.A.C.S. supports placing graphic objects partially or entirely outside the visible area of a screen. Since graphic objects that lie entirely outside the screen area are not visible in the Operate System, it is not usually desirable to have objects placed outside the screen area. If you want to remove all objects that lie entirely outside the screen area, select "Delete Off-Screen Objects" from the "Special" menu.

MENUS

The File Menu

- Close Window:

Closes the current SCL Program Editor or Screen Editor window.

- New:

Creates a new file. The file may be used as a station or as a device driver library.

- Open:

Opens an existing station file or device driver library for editing. This closes the file you are currently editing. If you haven’t saved the current file, you will be asked if you want to save.

- Load Images:

Loads images from an image library.

- Reload Images:

Reloads the last image library loaded.

- Save:

Saves the current file.

- Save As:

Saves the current file under a new name. May also be used to save driver libraries as station files and vice versa, subject to the limitations of device driver libraries.

- Print:

Prints the current SCL Program Editor window.

- Print Preview:

Lets you see what a printout of the current SCL Program Editor window will look like.

- Print Setup:

Lets you specify the printer and print options for printing SCL Program Editor windows.

- List of recently opened files:

Lets you open a file you recently used.

- Exit:

Exits the U.P.M.A.C.S. Development System. If you haven’t saved the current file, you will be asked if you want to save.

The Edit Menu

- Undo:

Undoes the last thing you did in the SCL Program Editor.

- Redo:

Redoes the last thing you undid in the SCL Program Editor.

- Cut:

Removes the current selection and places it on the clipboard.

- Copy:

Places a copy of the current selection on the clipboard.

- Paste:

Replaces the current selection with the content of the clipboard.

- Paste Special:

Allows you to insert tags of database objects into an SCL Program Editor window.

- Delete:

Deletes the current selection.

- Duplicate:

Makes a copy of all graphic objects currently selected.

- Select All:

Selects everything.

- Go To Line:

Lets you place the cursor on a specific line in an SCL Program Editor window.

- Find:

Allows you to search for text in an SCL Program Editor window.

- Find Again:

Finds the next occurrence of the last find.

- Replace:

Allows you to search for text in an SCL Program Editor window and replace it.

- Toggle Bookmark:

Adds or removes a bookmark in the current line of an SCL Program Editor window.

- Previous Bookmark:

Goes to the previous bookmark in an SCL Program Editor window.

- Next Bookmark:

Goes to the next bookmark in an SCL Program Editor window.

- Clear All Bookmarks:

Clears all bookmarks in an SCL Program Editor window.

- Properties:

Allows you to edit the properties of a screen or an SCL program when viewing its editor.

- SCL Syntax Colours:

Allows you to specify the syntax colours used by the SCL Program Editor.

The View Menu

- Device Drivers:

Shows the list of all device drivers.

- Ports:

Shows the list of all serial ports.

- Registers:

Shows the list of all registers.

- Screens:

Shows the list of all screens.

- Programs:

Shows the list of all SCL programs.

- SABus Request:

Shows the list of all SABus requests.

- Indicator Values:

Allows you to specify the values that the Screen Editor displays for digital indicators, analog indicators, and string indicators.

- Tool Bars:

Allows you to select which tool bars are visible.

- Drawing Properties:

Shows or hides the Drawing Properties window.

- Status Bar:

Shows or hides the status bar. The status is located at the bottom of the U.P.M.A.C.S. Development System window, and displays information about the Caps Lock and Num Lock keys, and about the overwrite mode and current line of the SCL Program Editor.

- Grid:

Shows or hides the graphics grid. This does not affect the grid snap.

- Grid Size:

Lets you change the spacing between grid points of the graphics grid.

- Align Grid With Screen:

Moves the graphics grid so that one grid point is aligned with the upper left-hand corner of the screen.

- Align Grid With Selection:

Moves the graphics grid so that one grid point is aligned with the current selection.

The New Menu

- Device Driver:

Creates a new device driver.

- Port:

Creates a new serial port.

- Register:

Creates a new register. You will be asked to select the type of register to create.

- Screen:

Creates a new screen.

- Program:

Creates a new SCL program.

- SABus Request:

Creates a new SABus request. You will be asked whether to create a query or a command.

The Draw Menu

- Lines:

Selects the line shape as the shape for static objects, bistate indicators, and multistate indicators.

- Splines:

Selects the Bezier spline (curved line) shape as the shape for static objects, bistate indicators, and multistate indicators.

- Rectangles:

Selects the rectangle shape as the shape for static objects, bistate indicators, and multistate indicators.

- Ellipses:

Selects the ellipse shape as the shape for static objects, bistate indicators, and multistate indicators.

- Text:

Selects the text shape as the shape for static objects, bistate indicators, and multistate indicators.

- Images:

Selects the image shape as the shape for static objects, bistate indicators, and multistate indicators.

- Select:

Allows you to select graphic objects.

- Reshape:

Allows you to select graphic objects, and adjust the shape of Bezier splines (curved lines), the margins of digital indicators, analog indicators, and string indicators, and the label rectangles of X-Y position markers.

- Objects:

Selects the Object drawing tool.

- 3D Objects:

Selects the 3D Object drawing tool.

- Bistate Indicators:

Selects the Bistate Indicator drawing tool.

- Multistate Indicators:

Selects the Multistate Indicator drawing tool.

- Digital Indicators:

Selects the Digital Indicator drawing tool.

- Analog Indicators:

Selects the Analog Indicator drawing tool.

- String Indicators:

Selects the String Indicator drawing tool.

- Dials:

Selects the Dial drawing tool.

- Graphs:

Selects the Graph drawing tool.

- X-Y Position Markers:

Selects the X-Y position marker drawing tool.

- Controls:

Selects the Control drawing tool.

- Labels:

Selects the Label drawing tool.

- Lock Tool:

Locks and unlocks the current drawing tool.

- Snap To Grid:

Switches grid snap on or off. While grid snap is on, all objects are created, moved, and resized to the nearest grid point.

- Get Properties From Selection:

Changes the current drawing properties (used to draw new objects) to match the properties of the currently selected objects.

The Arrange Menu

- Raise:

Moves the selected graphic objects one object further forward.

- Lower:

Moves the selected graphic objects one object further backward.

- Bring To Front:

Moves the selected graphic objects all the way to the front.

- Send To Back:

Moves the selected graphic objects all the way to the back.

- Flip Horizontally:

Flips (mirrors) the selected graphic objects left-to-right. All selected objects are flipped individually about their own centers. To flip a group of objects together about their common center, group them first. Some objects, like text and images, cannot be flipped.

- Flip Vertically:

Flips (mirrors) the selected graphic objects top-to-bottom. All selected objects are flipped individually about their own centers. To flip a group of objects together about their common center, group them first. Some objects, like text and images, cannot be flipped.

- Rotate 90° Left:

Rotates the selected graphic objects 90° to the left (counter-clockwise). All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

- Rotate 90° Right:

Rotates the selected graphic objects 90° to the right (clockwise). All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

- Rotate 180°:

Rotates the selected graphic objects by 180°. All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

- Align With Grid:

Aligns the currently selected graphic objects to the nearest grid point. All selected objects are aligned together as a group; their position relative to each other is not changed.

- Align Objects:

Allows you to align all selected graphic object with each other by popping up the Align Objects dialog.

- Group:

Groups all selected graphic objects so they behave as one single object.

- Ungroup:

Breaks up all selected graphic object groups into their component objects.

The Special Menu

- Save Device Drivers:

Saves all device drivers in the current station as a device driver library.

- Import Device Drivers:

Allows you to import device drivers from a device driver library.

- Save And Test Station:

Allows you to save the current station and test it by loading it in the U.P.M.A.C.S. Operate System all in one step.

- Modify/Duplicate Registers:

Allows you to modify, duplicate, or delete multiple registers at the same time.

- Modify/Duplicate Programs:

Allows you to modify, duplicate, or delete multiple SCL programs at the same time.

- Modify/Duplicate SABus Requests:

Allows you to modify, duplicate, or delete multiple SABus requests at the same time.

- Copy Register Sources:

Allows you to transfer register sources between registers.

- Copy Program Code:

Allows you to transfer program code between SCL programs.

- Modify Screen Objects:

Allows you to modify multiple graphic objects of different types at the same time.

- Delete Off-Screen Objects:

Deletes all graphic objects in the current Screen Editor which lie entirely outside the visible area of the screen.

The Window Menu

- Auto Size:

Adjusts the size of the current Screen Editor window to fit the size of the screen.

- Cascade:

Arranges all Screen Editor and SCL Program Editor windows so that they overlap.

- Tile Horizontally:

Arranges all Screen Editor and SCL Program Editor windows as horizontal non-overlapping tiles.

- Tile Vertically:

Arranges all Screen Editor and SCL Program Editor windows as vertical non-overlapping tiles.

- Arrange Icons:

Arranges all minimized Screen Editor and SCL Program Editor windows at the bottom of the U.P.M.A.C.S. Development System window.

- Close All:

Closes all Screen Editor and SCL Program Editor windows.

- List of open windows:

Allows you to bring a specific window to the front.

The Help Menu

- Development System:

Shows the help file for the Development System file.

- Device Drivers:

Shows the Device Drivers help file that describes how to develop device drivers..

- SCL Language:

Shows the SCL Language help file.

- Find Keyword:

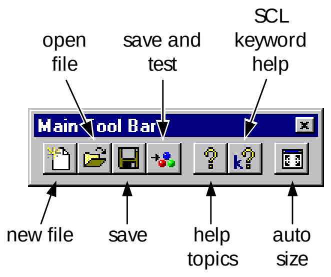
In an SCL Program Editor, displays help for the SCL keyword that the cursor is on (or that is currently selected) from the SCL Language help file.

- About U.P.M.A.C.S.:

Shows the version and copyright information for the version of the U.P.M.A.C.S. Development System you are currently running.

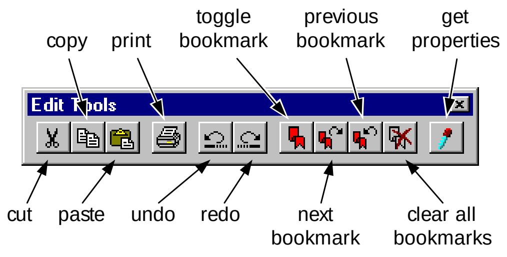
TOOLS BARS

The Main Tool Bar



- New file:
Creates a new file. The file may be used as a station or as a device driver library.
- Open file:
Opens an existing station file or device driver library for editing. This closes the file you are currently editing. If you haven't saved the current file, you will be asked if you want to save.
- Save:
Saves the current file.
- Save and test:
Allows you to save the current station and test it by loading it in the U.P.M.A.C.S. Operate System all in one step.
- Help topics:
Shows the Development System help file.
- SCL keyword help:
In an SCL Program Editor, displays help for the SCL keyword that the cursor is on (or that is currently selected) from the SCL Language help file.
- Auto size:
Adjusts the size of the current Screen Editor window to fit the size of the screen.

The Edit Tools



- Cut:
Removes the current selection and places it on the clipboard.
- Copy:
Places a copy of the current selection on the clipboard.

- Paste:

Replaces the current selection with the content of the clipboard.

- Print:

Prints the current SCL Program Editor window.

- Undo:

Undoes the last thing you did in the SCL Program Editor.

- Redo:

Redoes the last thing you undid in the SCL Program Editor.

- Toggle bookmark:

Adds or removes a bookmark in the current line of an SCL Program Editor window.

- Previous bookmark:

Goes to the previous bookmark in an SCL Program Editor window.

- Next bookmark:

Goes to the next bookmark in an SCL Program Editor window.

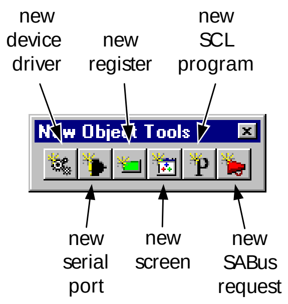
- Clear all bookmarks:

Clears all bookmarks in an SCL Program Editor window.

- Get properties:

In the Screen Editor, changes the current drawing properties (used to draw new objects) to match the properties of the currently selected objects.

The New Object Tools



- New device driver:

Creates a new device driver.

- New serial port:

Creates a new serial port.

- New register:

Creates a new register. You will be asked to select the type of register to create.

- New screen:

Creates a new screen.

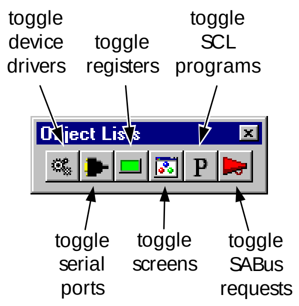
- New SCL program:

Creates a new SCL program.

- New SABus request:

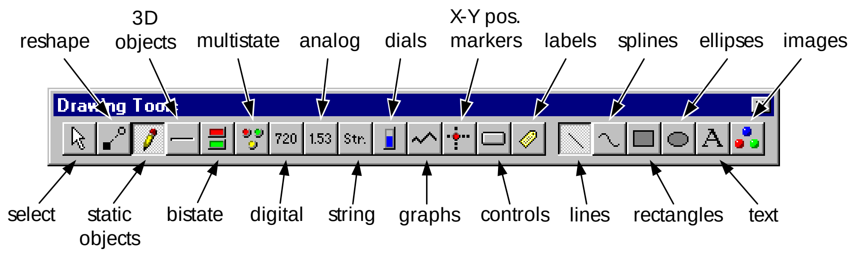
Creates a new SABus request. You will be asked whether to create a query or a command.

The Object Lists Tool Bar



- Toggle device drivers:
Shows or hides the list of all device drivers.
- Toggle serial ports:
Shows or hides the list of all serial ports.
- Toggle registers:
Shows or hides the list of all registers.
- Toggle screens:
Shows or hides the list of all screens.
- Toggle SCL programs:
Shows or hides the list of all SCL programs.
- Toggle SABus requests:
Shows or hides the list of all SABus requests.

The Drawing Tools



The Drawing Tools

- Select:
Allows you to select graphic objects.
- Reshape:
Allows you to select graphic objects, and adjust the shape of Bezier splines (curved lines), the margins of digital indicators, analog indicators, and string indicators, and the label rectangles of X-Y position markers.
- Static objects:
Selects the Object drawing tool. Click the button repeatedly to lock and unlock the tool.
- 3D objects:
Selects the 3D Object drawing tool. Click the button repeatedly to lock and unlock the tool.

- Bistate:

Selects the Bistate Indicator drawing tool. Click the button repeatedly to lock and unlock the tool.

- Multistate:

Selects the Multistate Indicator drawing tool. Click the button repeatedly to lock and unlock the tool.

- Digital:

Selects the Digital Indicator drawing tool. Click the button repeatedly to lock and unlock the tool.

- Analog:

Selects the Analog Indicator drawing tool. Click the button repeatedly to lock and unlock the tool.

- String:

Selects the String Indicator drawing tool. Click the button repeatedly to lock and unlock the tool.

- Dials:

Selects the Dial drawing tool. Click the button repeatedly to lock and unlock the tool.

- Graphs:

Selects the Graph drawing tool. Click the button repeatedly to lock and unlock the tool.

- X-Y pos. markers:

Selects the X-Y position marker drawing tool. Click the button repeatedly to lock and unlock the tool.

- Controls:

Selects the Control drawing tool. Click the button repeatedly to lock and unlock the tool.

- Labels:

Selects the Label drawing tool. Click the button repeatedly to lock and unlock the tool.

- Lines:

Selects the line shape as the shape for static objects, bistate indicators, and multistate indicators.

- Splines:

Selects the Bezier spline (curved line) shape as the shape for static objects, bistate indicators, and multistate indicators.

- Rectangles:

Selects the rectangle shape as the shape for static objects, bistate indicators, and multistate indicators.

- Ellipses:

Selects the ellipse shape as the shape for static objects, bistate indicators, and multistate indicators.

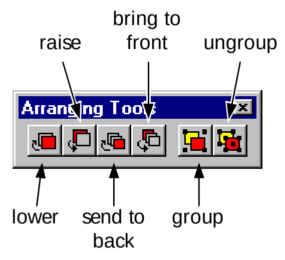
- Text:

Selects the text shape as the shape for static objects, bistate indicators, and multistate indicators.

- Images:

Selects the image shape as the shape for static objects, bistate indicators, and multistate indicators.

The Arranging Tools



- Lower:

Moves the selected graphic objects one object further backward.

- Raise:

Moves the selected graphic objects one object further forward.

- Send to back:

Moves the selected graphic objects all the way to the back.

- Bring to front:

Moves the selected graphic objects all the way to the front.

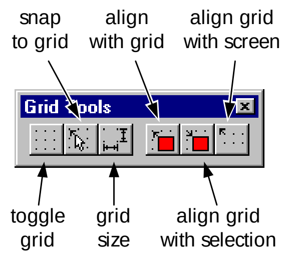
- Group:

Groups all selected graphic objects so they behave as one single object.

- Ungroup:

Breaks up all selected graphic object groups into their component objects.

The Grid Tools



- Toggle grid:

Shows or hides the graphics grid. This does not affect the grid snap.

- Snap to grid:

Switches grid snap on or off. While grid snap is on, all objects are created, moved, and resized to the nearest grid point.

- Grid size:

Lets you change the spacing between grid points of the graphics grid.

- Align with grid:

Aligns the currently selected graphic objects to the nearest grid point. All selected objects are aligned together as a group; their position relative to each other is not changed.

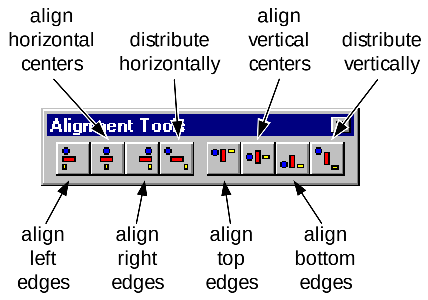
- Align grid with selection:

Moves the graphics grid so that one grid point is aligned with the current selection.

- Align grid with screen:

Moves the graphics grid so that one grid point is aligned with the upper left-hand corner of the screen.

The Alignment Tools



- Align left edges:

Aligns the left edge of all objects with the leftmost object.

- Align horizontal centers:

Aligns the horizontal centers of all objects at the center of the selection.

- Align right edges:

Aligns the right edge of all objects with the rightmost object.

- Distribute horizontally:

Spaces the objects evenly along the horizontal axis.

- Align top edges:

Aligns the top edge of all objects with the topmost object.

- Align vertical centers:

Aligns the vertical centers of all objects at the center of the selection.

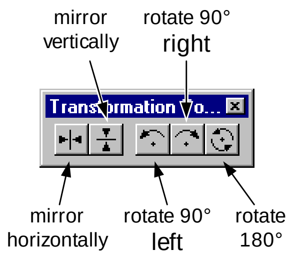
- Align bottom edges:

Aligns the bottom edge of all objects with the bottommost object.

- Distribute vertically:

Spaces the objects evenly along the vertical axis.

The Transformation Tools



- Mirror vertically:

Mirrors (flips) the selected graphic objects top-to-bottom. All selected objects are flipped individually about their own centers. To flip a group of objects together about their common center, group them first. Some objects, like text and images, cannot be flipped.

- Mirror horizontally:

Mirrors (flips) the selected graphic objects left-to-right. All selected objects are flipped individually about their own centers. To flip a group of objects together about their common center, group them first. Some objects, like text and images, cannot be flipped.

- Rotate 90° left:

Rotates the selected graphic objects 90° to the left (counter-clockwise). All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

- Rotate 90° right:

Rotates the selected graphic objects 90° to the right (clockwise). All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

- Rotate 180°:

Rotates the selected graphic objects by 180°. All selected objects are rotated individually about their own centers. To rotate a group of objects together about their common center, group them first. Some objects, like text and images, cannot be rotated.

APPENDICES

Appendix A: Regular Expressions

Regular expressions are used as mask patterns, prefixes, and suffixes, as error patterns, as alarm triggers for string registers, and in filter sources and strings sources. A regular expression is basically a search string with added capabilities. The simplest regular expression simply matches a string:

```
Deconstructionism
```

matches "Deconstructionism".

A regular expression consists of one or more sections, separated by vertical bars ("|"). The regular expression matches a string, if any of the sections match:

```
Zebra|Aardvark
```

matches both "Zebra" and "Aardvark".

Each section consists of a series of concatenated parts, optionally followed by modifiers.

Parts

A part can be one of the following:

- Characters
- Sets of characters
- Special characters
- An entire regular expression, enclosed in parentheses ("( )").

Each part matches one or more characters.

Character Parts

Any character that does not have a special meaning (i.e. any character except "|", "(", ")", "{", "[", "?", "*", "+", ".", "~", "&", "@", "#", "$", "%", and "\") simply matches itself:

```
Q
```

matches "Q".

```
=
```

matches "=".

To match a character with special meaning, prepend it with a backslash:

▪ Sequence	▪ Character matched
\(	left parenthesis ("(")

\)	right parenthesis (")")
\{	left brace ("{")
\[	left square bracket ("[")
\?	question mark ("?")
*	asterisk ("*")
\+	plus sign ("+")
\.	period (".")
\~	tilde ("~")
\&	ampersand ("&")
\@	commercial at ("@")
\#	octothorpe ("#")
\$	dollar symbol ("$")
\%	percent sign ("%")
\\	backslash ("\")

You can match a special set of non-printable characters by using the following character sequences:

▪ Sequence	▪ Character matched	▪ Code (hexadecimal)
\0	null character	$00
\b	backspace	$08
\t	tab	$09
\n	linefeed	$0A
\v	vertical tab	$0B
\f	form feed	$0C
\r	carriage return	$0D

To match any other non-printable character, use \x followed by two hexadecimal digits specifying the character code. Here are some examples:

▪ Sequence	▪ Character matched	▪ Code (hexadecimal)
\x02	start transmission	$02
\x03	end of transmission	$03
\xFF	delete	$FF
\xB7	ASCII $B7	$B7
\x69	capital letter "E"	$69

All other characters can just be entered plainly. If you feel so inclined, however, you can use a backslash followed by that character.

Here are some examples:

▪ Sequence	▪ Character matched	▪ Code (hexadecimal)
\a	letter "a"	$97
\6	digit six	$56

\/	slash	$47
\R	capital letter “R”	$82

Character Set Parts

A set of characters is enclosed in square brackets (“[ ]”). The set matches any character in it, or, if the open square bracket is followed by a caret (“^”), any character not in it (this is called an inverted set):

```
[AY#]
```

matches “A”, “Y”, and “#”.

```
[^AY#]
```

matches any character but “A”, “Y”, and “#”.

To include a caret in the set, place it anywhere but the beginning:

```
[AY^#]
```

matches “A”, “Y”, “^”, and “#”.

To include a closing square bracket (“]”) in the set, place it as the first character in the set (after the caret, for an inverted set):

```
[]AY]
```

matches “]”, “A”, and “Y”.

```
[^]AY]
```

matches any character but “]”, “A”, and “Y”.

To enter a whole range of characters, separate the two endpoints by a dash (“-”):

```
[A-Z]
```

matches any capital letter.

```
[0-9A-F]
```

matches any capital hex digit.

The endpoint of one range can be the starting point of the next:

```
[A-H-Z]
```

matches any capital letter (the same as [A-Z]).

To include a dash, make it the first or last character in the range, after the caret for inverted sets:

```
[AZ-]
```

matches "A", "Z", and "-".

```
[-01]
```

matches "-", "0", and "1".

```
[^-+]
```

matches any character but "-" and "+".

To include a dash and a closing square bracket, place the bracket at the beginning, and the dash at the end:

```
[]AB-]
```

matches "]", "A", "B", and "-".

```
[^]XY-]
```

matches any character but "]", "X", "Y", and "-".

Non-printable characters can be entered as described in 1.:

```
[\n\r]
```

matches a carriage return or a line feed.

```
[\\\x01]
```

matches "\" or an ASCII $01(SOH)

Closing square brackets, dashes, and carets can be placed anywhere in the set, if they are prepended by a backslash:

```
[\^\-\]ABC]
```

matches "^", "-", "]", "A", "B", and "C".

Special Character Parts

The following special characters match all characters of a specific type:

▪ Character	▪ Description	▪ Characters matched
.	period	any character
~	tilde	any printable character (ASCII $20-$7E)
&	ampersand	any alphanumeric character ("0"-"9", "A"-"Z", "a"-"z")
@	commercial at	any letter ("A"-"Z", "a"-"z")
#	octothorpe	any digit ("0"-"9")
$	dollar sign	any hexadecimal digit ("0"-"9", "A"-"F", "a"-"f")
%	percent sign	any binary digit ("0" and "1")

Modifiers

Each part may be followed by one of four modifiers:

- A question mark (“?”)
- An asterisk (“*”)
- A plus sign (“+”)
- A range, enclosed in braces (“{ }”)

The modifiers control how many times a part is matched. A part without a modifier is matched exactly once.

The Question Mark Modifier

A part that is followed by a question mark (“?”) is matched zero times or once. This means that it may but need not appear in the string to be matched:

```
w?
```

matches the empty string and “w”.

```
.?
```

matches the empty string or a single character.

The Asterisk Modifier

A part that is followed by an asterisk (“*”) is matched zero or more times:

```
$*
```

matches the empty string, as well as any sequence of hex digits, e.g. “4”, “34F2”, and “deadbeef”.

```
[\t ]*
```

matches the empty string and any combination of tabs and spaces.

The Plus Sign Modifier

A part that is followed by a plus sign (“+”) is matched one or more times:

```
A+
```

matches one or more “A”s, e.g. “A”, “AA”, “AAAAAAAAAAAA”, etc.

```
(so)+
```

matches one or more “so”s, e.g. “so”, “soso”, “sososo”, etc.

Range Modifiers

A part that is followed by a number enclosed in braces is matched that many times:

```
#{12}
```

matches any twelve-digit number.

A part followed by two numbers enclosed in braces (separated by a comma) is matched at least as many times as the first number, and at most as many times as the second.

```
[A-Z]{5,10}
```

matches between five and ten capital letters.

If the second number is missing, then the part is matched as many times as the first number, or more:

```
${4,}
```

matches four or more hex digits.

Sections

A section matches a string whose characters match each of its parts in sequence:

```
Elk
```

matches "E", followed by "l", followed by "k", i.e. "Elk".

```
r#d#
```

matches "r", followed by a digit followed by "d", followed by a digit, amongst others "r8d9" and "r2d2".

```
C[16][46]
```

matches a "C", followed by "1" or "6", followed by "4" or "6", i.e. "C14", "C16", "C64", and "C66".

```
Dum-?(didl-?)+dum
```

matches "D", followed by "u", "m", perhaps a "-", one or more "didl"s or "didl-"s, followed by "d", "u", and "m", e.g.:

```
Dum-didl-dum
Dum-didldidl-dum
Dum-didl-didldidl-dum
Dumdidl-didldidl-didldidl-dum
```
etc.

Note: No section of a regular expression may be constructed such that is could match zero characters. The following expressions are not valid sections:

```
.*              🡐 error!
#*\t*           🡐 error!
a?b?c?d?e?      🡐 error!
```

Each of these is an invalid section, as they would match an empty string.

Examples of regular expressions

```
\x02.*\x03
```

matches an STX (ASCII $02) / ETX (ASCII $03) character pair with any number of characters in between.

```
[\x06\x21].*\x03
```

matches an ACK (ASCII $06) or NAK (ASCII $21) / ETX (ASCII $03) character pair with any number of characters in between.

```
No (blue|green|red) dogs? (allowed|wanted)!|Hand me that b[aeiu]ll(, please)?\.
```

matches all of the following:

```
No blue dog allowed!
No green dog allowed!
No red dog allowed!
No blue dogs allowed!
No green dogs allowed!
No red dogs allowed!
No blue dog wanted!
No green dog wanted!
No red dog wanted!
No blue dogs wanted!
No green dogs wanted!
No red dogs wanted!
Hand me that ball.
Hand me that bell.
Hand me that bill.
Hand me that bull.
Hand me that ball, please.
Hand me that bell, please.
Hand me that bill, please.
Hand me that bull, please.
```

Here is another example:

```
t-000#s|We have liftoff!
```

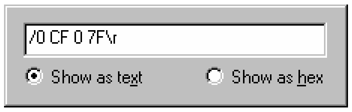
matches all of the following:

```
t-0009s
t-0008s
t-0007s
t-0006s
t-0005s
t-0004s
t-0003s
t-0002s
t-0001s
t-0000s
We have liftoff!
```

Appendix B: Entering Binary Data

Some data strings can contain non-printable characters. The entry fields for such data strings allow you to select two different ways of entering data: hexadecimal format, and text format. You can switch between entry modes by clicking the "Show as hex" and "Show as text" radio buttons.

Entering Data in Text Format



If the "Show as text" button is pressed, you can enter data normally as text. Special sequences of characters are used to enter non-printable characters that you cannot type on the keyboard. All these sequences begin with a backslash ("\"):

Sequence	Character	Code (hexadecimal)
\0	null character	$00
\b	backspace	$08
\t	tab	$09
\n	linefeed	$0A
\v	vertical tab	$0B
\f	form feed	$0C
\r	carriage return	$0D

to specify any other non-printable character, use \x followed by two hexadecimal digits specifying the character code. Here are some examples:

Sequence	Character	Code (hexadecimal)
\x02	start transmission	$02
\x03	end of transmission	$03
\xFF	delete	$FF
\xB7	ASCII $B7	$B7
\x69	capital letter "E"	$69

To specify a backslash, use two backslashes in a row:

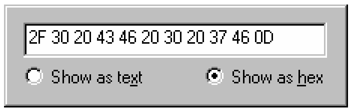
Sequence	Character	Code (hexadecimal)
\\	backslash	$92

Printable characters, with the exception of the backslash, are just entered plainly. If you feel so inclined, however, you can use a backslash followed by that character.

Here are some examples:

Sequence	Character matched	Code (hexadecimal)
\a	letter "a"	$97
\6	digit six	$56
\/	slash	$47
\R	capital letter "R"	$82

Entering Data in Hexadecimal Format



2F 30 20 43 46 20 30 20 37 46 0D

Show as text [✓ Show as hex]

If the "Show as hex" button is pressed, you can enter data as hexadecimal numbers separated by spaces. Each number corresponds to one character.

Appendix C: Entering Special Characters

The U.P.M.A.C.S. Development System allows you to enter special characters, like the degree symbol (“°”) or the micro symbol (“µ”) easily using special key combinations:

Symbol:	Key combination:
±	Ctrl + Shift + “=“ *or* Ctrl + “+” (numerical keyboard)
×	Ctrl + Shift + “8” *or* Ctrl + “*” (numerical keyboard)
÷	Ctrl + “/” *or* Ctrl + “/” (numerical keyboard)
µ	Ctrl + “M”
°	Ctrl + “0” (zero)
^{1}	Ctrl + “1”
^{2}	Ctrl + “2”
^{3}	Ctrl + “3”
½	Ctrl + Shift + “2”
¾	Ctrl + Shift + “3”
¼	Ctrl + Shift + “4”

If you press Ctrl + Space at any time, a menu with all available special characters, including letters with diacritical marks (accents, Umlaut) will appear.

Appendix D: Uplink Port Protocol

Packet Format

The uplink port protocol is based on Scientific Atlanta’s SABus protocol. Each request has the following format:

Byte Number:	Value:
1	STX (Hex $02)
2	SABus address (configurable from the Operate System)
3	request opcode
4 to n−2	request parameters (if required)
n−1	ETX (Hex $03)
n	LRC checksum of bytes 1 to n−1

The LRC is the logical XOR of all other bytes in the packet, including the STX and ETX.

If the request was processed successfully, the controller will respond with a response packet:

Byte Number:	Value:
1	ACK (Hex $06)
2	SABus address (same as request packet)
3	request opcode (same as request packet)
4 to n−2 n+2	response parameters (if required)
n−1	ETX (Hex $03)
n	LRC checksum of bytes 1 to n−1

If an error occurred processing the packet, the control system will respond with an error packet:

Byte Number:	Value:
1	NAK (Hex $15)
2	SABus address (same as request packet)
3	request opcode (same as request packet)
4 to n−2	error message
n−1	ETX (Hex $03)
n	LRC checksum of bytes 1 to n−1

Requests with parity errors, a bad start or stop character, bad address, or bad checksum are ignored.

Error Messages

The table below shows all the built-in error messages that may be returned. Variable parameters are shown enclosed in angle brackets (“<>“), but the brackets do not actually appear in the error message. They are shown merely for clarity.

Error Message:	Meaning:
OPC	unrecognized opcode.
PRM<nn>	bad parameter number <nn>. <nn> is always two digits
PRM##	too many parameters (The "##" appears as is in the error response. It is *not* shown as a placeholder for a number here.)
USR	insufficient user privilege level
PRG<nnnn><msg>	SCL program error on line <nnnn>. <nnnn> is always four digits, <msg> is a variable length error description.

You can define any number of other error messages you may require for your protocol.

If no station file is loaded, U.P.M.A.C.S. returns the OPC error code to all requests except the user request (see below).

Built-In Requests

The two requests described below are built into the U.P.M.A.C.S. Operate System. You cannot redefine the opcodes for those two requests.

There are no spaces between the parameters; the spaces included in the descriptions are there merely for the sake of better readability. Variable parameters are shown enclosed in angle brackets ("<>"), and literal characters that appear as is in the request or response packet are shown in single quotation marks. Neither the angle brackets or quotation marks are part of the packet; they are shown merely for clarity.

- User Request

This request returns information about the current user, or it signs on or off as a user. In order to execute SCL programs that require signing on, you must use this command to sign on to the system. Signing on from an uplink port is *not* the same as signing on locally.

Opcode: 'U' (character code $55)

This request may be used in one of three ways:

Query user:

- Request:

STX <address> 'U' ETX LRC

- Response:

If no user is signed on, the response will be:

ACK <address> 'U' '-' ETX LRC

If the user <user name> is signed on from the uplink port on which the request was received, the response will be:

ACK <address> 'U' 'R' <user name> ETX LRC

If the user <user name> is signed on from locally, the response will be:

ACK <address> 'U' 'L' <user name> ETX LRC

If the user <user name> is signed on from a remote computer, the response will be:

ACK <address> 'U' 'N' <user name> ETX LRC

If the user <user name> is signed from the uplink port <port name> (other than the one on which the request was received) the response will be:

ACK <address> 'U' 'U' <port name> ':' <user name> ETX LRC

Sign on

- Request:

STX <address> 'U' 'R' <user name> ETX LRC

Attempt to sign on as the user <user name>. <user name> is purely for informational purposes and can be freely chosen as any combination of printable characters (character codes $20 to $7E). U.P.M.A.C.S. does not do any user name or password checking on uplink port sign on requests.

- Response:

ACK <address> 'U' ETX LRC

The command will fail with the error message `USR` if someone with equal or higher privilege level is already signed on.

Sign off

- Request:

STX <address> 'U' '-' ETX LRC

Sign off.

- Response:

ACK <address> 'U' ETX LRC

The command will fail with the error message `USR` if no one is signed on from the uplink port on which the request was received.

- Acknowledge Alarms Request

This request allows a remote system to acknowledge all alarms on all screens.

Opcode: 'A' (character code $41)

- Request:

```
STX <address> 'A' ETX LRC
```

Acknowledges all alarms on all screens.

- Response:

```
ACK <address> 'A' ETX LRC
```

The command will fail with the error message `USR` if no one is signed on from the uplink port on which the request was received, or if that uplink port does not have sufficient clearance to acknowledge alarms.

Appendix E: Legacy Objects

Versions of U.P.M.A.C.S. prior to v6.0 used some database objects that are obsolete in v6.0. These objects are called legacy objects. There are three types of legacy objects:

Serial Communications:

- Legacy data masks
- Legacy device drivers

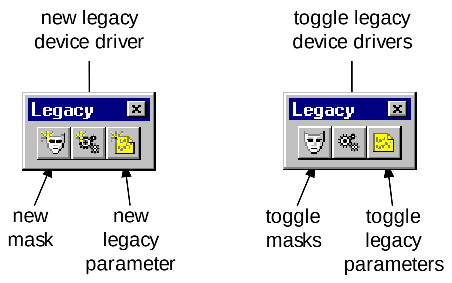
Data Storage:

- Legacy parameters

Device drivers contain commands, responses, messages, and replies

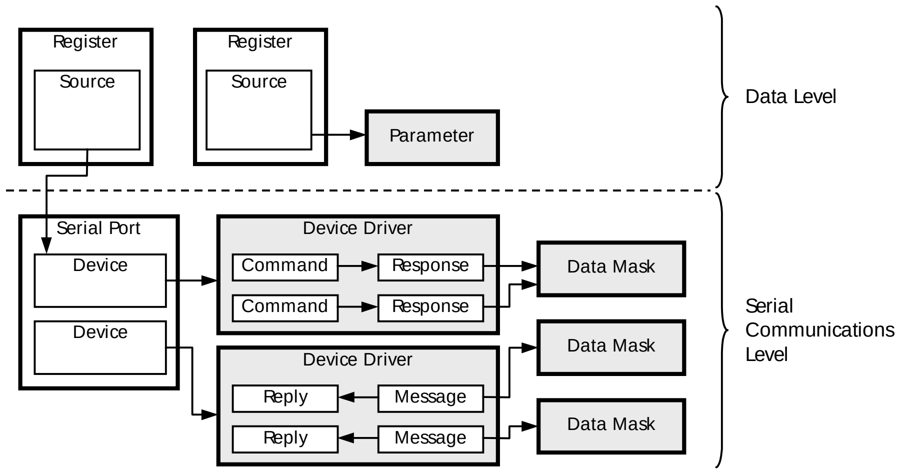
There are also some capabilities of serial ports that relate to legacy devices, as well as a number of sources for use with legacy devices and parameters.

To view legacy object lists, choose "Masks…", "Device Drivers…", or "Parameters…" from the "Legacy Objects" section of the "View" menu. To create legacy objects, select "Mask…", "Device Driver…", or "Parameter…" from the "Legacy Object" section of the "New" menu.



Legacy Object Tool Bars

A station that contains legacy objects usually has a structure that looks something like this:



Legacy Device Serial Communication

Serial Communication Objects

A brief description of the legacy objects used for serial communications is given below. The objects are described in more detail in their respective sections.

- Legacy devices

A legacy device is a device that uses a legacy device driver, rather than a regular device driver.

- Legacy device drivers

Each legacy device in a port uses a legacy device driver to specify the behaviour of the equipment. There is one legacy device driver for each type of equipment, but devices with the same characteristics (make, model, serial address) share a device driver. Legacy device drivers do not support parameters, so you must define a separate device driver for each device address. No two legacy devices on the same port may share the same driver.

- Legacy commands

Each legacy device driver may contain a number of commands. A command contains a string of characters or binary data that the equipment will understand, as well as information about the response that the equipment sends. Legacy commands do not support parameters, so you must define a separate command for each set of command parameters. Legacy commands do not contain the information about the equipment's response, they use responses (see below) for that purpose.

- Legacy responses

Legacy device drivers may also contain a number of responses. Each response represents the data sent by the equipment in response to a command. If the equipment sends the same data in response to two separate commands, these commands share a response. Each legacy device in a serial port that uses a specific driver contains one data buffer for each response in that driver. Separate devices that use the same driver have separate data buffers, to hold the different response data sent by their respective pieces of equipment.

- Legacy messages

Legacy device drivers may also contain a number of messages. Each message represents data that the equipment may send out of its own accord, without having been polled. Each device in a serial port that uses a specific driver contains one data buffer for each message in that driver. Separate devices that use the same driver have separate data buffers, to hold the different message data sent by their respective pieces of equipment.

- Legacy replies

Legacy device drivers may also contain a number of replies. A reply contains a string of characters or binary data that is sent in response to a message received by the equipment.

- Legacy data masks

Legacy data masks are used to specify what the data in a legacy response or message looks like. The data for many different responses or messages in many different drivers could have the same format, and many responses and messages from many different drivers may hence share a data mask. Many pieces of equipment use Scientific Atlanta's SABus protocol, for instance. In the SABus protocol, all response data follows a fixed format. All responses in all device drivers for equipment that supports SABus protocol can therefore share one single data mask. Data masks are global objects and not tied to serial ports, devices, or device drivers.

The Polling Process

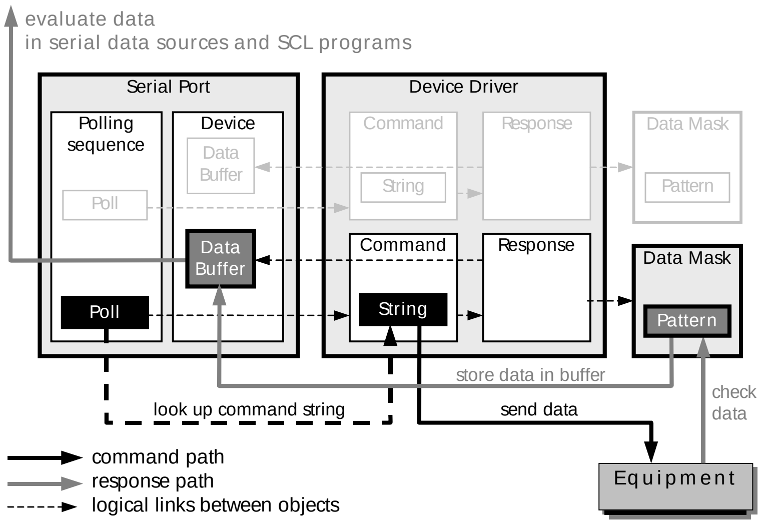
Most equipment never sends data out of its own accord, but only in response to a command. U.P.M.A.C.S. will send commands to such equipment continuously, and evaluate the data returned. Which commands are sent to what piece of equipment on the port in what order, is determined by the polling sequence.

The polling sequence consists of a series of polls. Each poll in the polling sequence contains a reference to a command in a device. The command, in turn, contains a reference to a response. The response itself contains a reference to a data mask that describes the format of the response data.

Each poll is executed as follows:

- Look up the command string in the command
- Send the string to the equipment
- Look up the format of the response data from the data mask of the command's response
- Check the data received from the equipment against that format
- If the data is OK, place it in the device's data buffer that corresponds to the response
- Evaluate the response data

Some commands may not have a response. For polls with such commands, only the first two steps apply.



Receiving Unsolicited Data

In some rare instances, a piece of equipment is designed to send data out of its own accord, rather than as a response to a command. This type of data is called "unsolicited data." You can configure a port to wait for unsolicited data rather than poll the equipment.

Each possible chunk of data that the equipment can send is described by a message in the equipment's device driver. The message contains a reference to a data mask that describes the data of the message, and a reference to a reply that will be sent in response to the message.

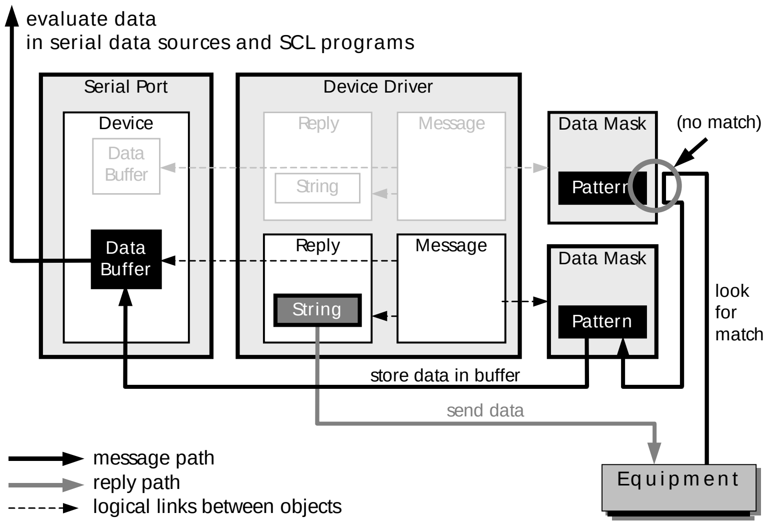
Unsolicited data is processed as follows:

- Wait for data to come in on the port
- Find a message whose data mask the data fits
- Place the data in the device's data buffer that corresponds to the message
- Evaluate the message data
- Look up the reply string in the message's reply

- Send the reply

Most messages will not have a reply. For such messages, only the first four steps apply.

As with response data, U.P.M.A.C.S. will automatically update all registers that take their value from a message data buffer when a message is received. In addition, however, you can specify an SCL program to do additional processing of the data, to send custom replies, or to perform any actions that may be necessary as a result of message.



Device Initialization

Legacy devices only have one type of initialization sequence. The initialization sequence is always used for the first-time initialization. It can optionally also be used for reinitialization. The initialization sequences for first-time initialization and reinitialization are always the same for legacy devices

Port Access Synchronization

Although these synchronization mechanisms are available on any port, they are only meant to be used on ports that use a polling sequence. Only commands and responses are synchronized, messages and replies are not. On ports that wait for unsolicited data, the equipment, not U.P.M.A.C.S., is the bus master (the device that decides who is allowed to send data), and you should never send any data to such a port unless the equipment attached to it specifically requested you to.

Designing Legacy Device Drivers

Step 1: Data Masks

You need to create a data mask for each type of response your equipment can have. Responses for different commands, even for different equipment, can often share the same mask, as long as they look similar and require the same timeout interval and buffer size.

You also need to create a data mask for every message your equipment might send. In theory, different messages in the same device driver could have the same data mask, but in practice this makes it impossible for U.P.M.A.C.S. to distinguish between the two messages, and it will pick one at random when data matching that mask is received. Messages in different device drivers can also have the same data mask, but if you use both devices on the same port, the same problem will occur. Make sure that all data masks used for messages on a single port are different enough for U.P.M.A.C.S. to distinguish what data constitutes which message.

U.P.M.A.C.S. does not distinguish between data masks for responses and data masks for messages. You could theoretically use the same data mask for messages and responses, but since the requirements for the two types of objects are very different, this is unlikely to be useful in real life.

The mask contains a pattern that describes the character sequence of the response or message. The pattern is a regular expression, a very versatile extended search string. The exact format of U.P.M.A.C.S. regular expressions is discussed in Appendix A. For data masks used in responses, you can write an expression that matches the response data very closely, or you can opt for a more flexible pattern. If you write a pattern that matches the response more closely, you are likely to need a greater number of different data masks, but the responses will be checked more thoroughly for correctness. Take some time to decide which responses you want to group together to use a single mask, and which responses you want to separate. Make sure you have acquainted yourself with regular expressions before doing so.

For data masks used in messages, you will usually want to write an expression that matches a message as closely as possible, as message data masks cannot be reused in any case.

If possible, it is a good idea to design masks that recognize the beginning of a response or message as well as the end. This will enable U.P.M.A.C.S. to eliminate garbage data coming in before the response data as well as after it.

Step 2: Device Drivers

You need to create a device driver for each type of equipment with which you want U.P.M.A.C.S. to communicate. If you have equipment that supports daisy chaining on a single port, you should write a separate driver for each device address. You can do this by writing the driver for the first address, and duplicating it as needed. You can then modify the commands, responses, messages, and replies to reflect the different addresses.

Devices contain commands, responses, messages and replies. Commands and responses are used for polling equipment; messages and replies are used for equipment that sends unsolicited data. It is possible for a single device to contain all four types of objects, but in practice this is unlikely to be useful, as equipment that sends unsolicited data cannot be polled. You will normally find that drivers you develop will contain either commands and responses, or messages and replies, but not both sets.

Since the device commands require the use of responses, you should create responses before you create commands.

- Responses

You will need one response object for each response the equipment sends. If the equipment sends the exact same data in response to two commands, you need only one response for it. Some equipment, for example, responds with a value (e.g. the current frequency) both to the associated set and query commands. In such a case, you need only one response.

If two responses look alike, but contain different information, you will need separate responses (they will share a data mask, however). You will need two separate responses for transmit and receive data rates of a modem, for example, even if the data is formatted exactly alike. This is necessary to tell the two pieces of information apart.

Many pieces of equipment send a generalized acknowledge response for all set commands. This might be a simple OK packet that is always identical, or it might be a response that contains the command opcode and possibly the data you sent. In this case, you will only need one response to cover all commands, since the actual data is of no consequence.

Remember that you need a response every time the equipment sends data, even if you are not planning to use the data for anything. U.P.M.A.C.S. uses the response to check the data that the equipment returns, regardless of whether anything further is done with it. If you fail to provide a response for a command that the equipment responds to, the data the equipment sends will remain in the computer's hardware receive buffer and may cause erroneous behavior.

- Commands

You can now create the commands that U.P.M.A.C.S. will send to your equipment. The commands defined here can only use fixed, pre-defined strings. You cannot dynamically specify any parameters for the command.

If an equipment command has a parameter that can take a fixed number of values, define a separate command for each of the values. A smart switch, e.g., might have a "set backup mode" command, with a number as a parameter that represents "manual," "revertive auto," and "non-revertive auto" modes. In that case, you cannot make one "set backup mode" command. Make three separate commands, one for "set manual mode," one for "set revertive auto mode," and one for "set non-revertive auto mode."

If an equipment command has a parameter that can assume a continuous range of values, you should not make a command for it. You should not make a "set frequency" command for a converter, for example. Simply provide a response for the command, and construct the command on the fly in the control that sets the frequency.

Since messages may require the use of replies, you should create replies before you create messages.

- Replies

Create a reply for each string that may need to be sent in response to a message from a device.

You can only create replies with a fixed string. If a message needs to send a reply that contains a parameter that can assume a variable range of values, like a frequency or a time value, you should not create a reply object for it. Such replies must be created on the fly in an SCL program.

If the reply to a message can take one of a small number of forms, define a reply for each variant you want to use. An SCL program that processes a message can then send these predefined replies, without having to construct them within the SCL code.

- Messages

You can now create messages for all the data that the equipment may send. You should use a different data mask for each message, because the mask is used to determine which message is received. If you use the same mask for more than one message, U.P.M.A.C.S. will not be able to tell which message it is dealing with when it receives data that matches that mask, and it will simply pick one message at random.

A message can be assigned with a reply. The reply will be sent whenever the message is received. You can only specify one reply for each message, and the reply can only contain a fixed string. If the message can have one of several different replies under different circumstances, do not assign a reply to the message. Instead, specify an SCL program to process messages when you add the device to its serial a port, and let that program determine which reply should be sent and send it. The program can either send any of the predefined replies you created, or it can create one on the fly.

Step 3: Serial Ports

Define one serial port for every physical port to which equipment is connected. Add a device for each piece of equipment on the port.

If the equipment attached to the port needs to be polled, you must configure the port to poll the equipment. You can then specify the polling sequence. A port need not contain a polling sequence; if you do not create any polls, the port will simply remain idle until an SCL program sends a command.

If the equipment on the port sends data out of its own accord, you must configure the port to wait for unsolicited data. Since such ports rarely have more than one piece of equipment connected to them, you will usually create only one device. If it is necessary or convenient, however, you can create more than one device: just make sure that all messages in all devices use different masks, or U.P.M.A.C.S. will not be able to distinguish between them.

For ports that wait for unsolicited data, you can also specify an SCL program to do additional processing, or to perform necessary actions, when a message is received. You can either specify single program to be used for all devices, or a separate program for each device.

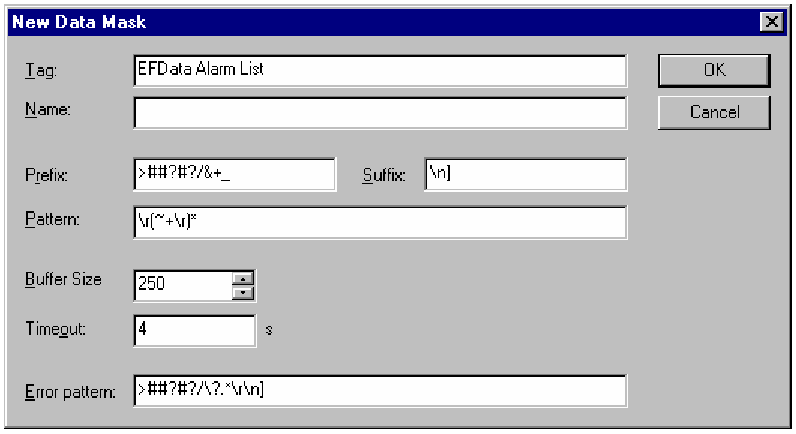
Legacy Data Masks

A data mask provides information about the format of data that equipment sends in response to a command or out of its own accord. Data masks are global objects that can be shared by multiple responses and messages in multiple device drivers.

The data mask provides patterns to recognize valid data as well as error messages sent by the equipment. Valid data is divided into three sections: A prefix, the data, and a suffix. Basically, the data is all information that you are interested in, and the prefix and suffix are any synchronization and housekeeping information that you want to discard.

If you are planning to use device drivers from a device driver library (see Device Driver Libraries for details), you do not need to create data masks for them. All necessary data masks are imported with the device drivers.

The New Data Mask Dialog



Tag:	EFData Alarm List
Name:	
Prefix:	>##?#?/&+_
Suffix:	\n]
Pattern:	\r(~+\r)*
Buffer Size	250
Timeout:	4 s
Error pattern:	>##?#?/\?.*\r\n]

- Tag:

Enter the tag by which the mask is identified. Each mask must have a unique tag.

- Name:

Enter the name of the mask. Leave this field blank if you want to use the tag as name.

- Prefix:

Enter a regular expression for the response or message header here. Headers usually consist of a synch byte, the device addresses, and sometimes a byte count and the command opcode. U.P.M.A.C.S. will strip the prefix from the data before processing it. Leave this field blank if your response or message has no header, or if you want to use a device driver-specific prefix. If you specify a prefix in the device driver and in the data mask, the prefix of the data mask will be used.

See *Appendix A: Regular Expressions* on page 171 for details.

- Suffix:

Enter a regular expression for the response or message trailer here. Trailers usually consist of a checksum and often a terminating character. U.P.M.A.C.S. will strip the suffix from the data before processing it. Leave this field blank if your response or message has no trailer, or if you want to use a device driver-specific suffix. If you specify a suffix in the device driver and in the data mask, the suffix of the data mask will be used.

See *Appendix A: Regular Expressions* on page 171 for details.

- Pattern:
Enter a regular expression for the response or message data here. You cannot leave this field blank, as all responses and messages must have data. If a response or a message has no data, but consist only of a header and a trailer, you have to treat part of the header or trailer as data.

See *Appendix A: Regular Expressions* on page 171 for details.

- Buffer size:
Enter the size of the data buffer here. The data buffer must be large enough to hold not only the data, but also the prefix and the suffix. If the response or message has a fixed length, enter that length here. If it has a variable length, enter a buffer size large enough to hold the largest possible response or message. If a response or a message is too long to fit in the buffer, U.P.M.A.C.S. will not recognize it as valid, even if it is.

- Timeout:
This field is used differently for responses and messages.

For Responses, enter the maximum time the equipment is allowed to respond. If no valid response has been received after the number of seconds specified, a timeout is reported.

For messages, enter the inter-character timeout. This is the maximum time the equipment is allowed to wait between two characters in the same message. If no new characters are received within this number of seconds, U.P.M.A.C.S. will assume that the equipment is done sending data.

You can enter fractions of a second for the timeout.

- Error pattern:
Enter a regular expression by which an error response or error message sent by the equipment can be recognized. If the equipment does not send error responses or messages, or if you want to use a device driver-specific error format, leave this field blank. If you specify an error pattern in the device driver and in the data mask, the error pattern of the data mask will be used.

The error pattern is not used in conjunction with the prefix or suffix. Include the error response header and trailer in the error pattern.

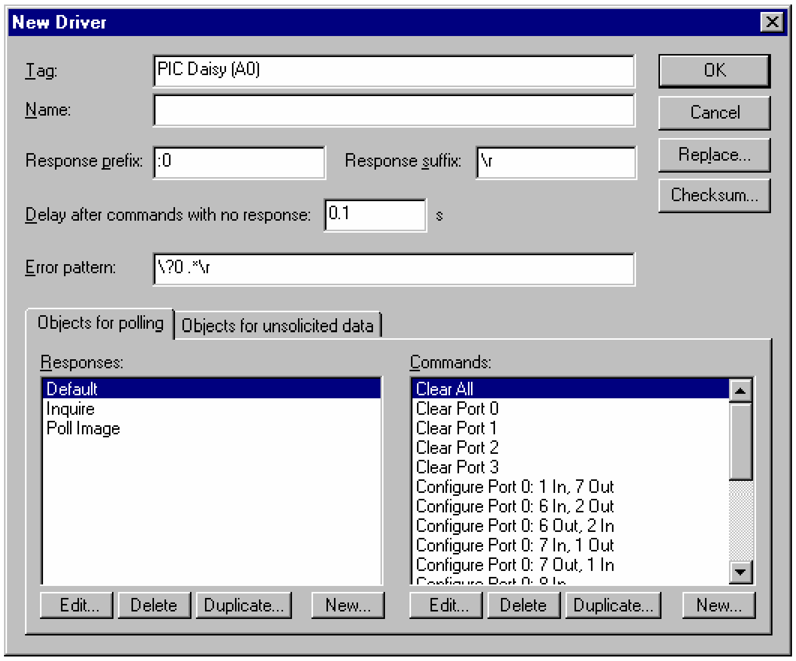
Make sure that the error responses are distinguishable from valid responses. If an error response fits the prefix, pattern, and suffix of the mask, it will be treated as data.

Legacy Device Drivers

A device driver provides information about the command set of a particular piece of equipment. Identical pieces of equipment share a common device driver.

You can create device drivers from within the Development System, or you can import drivers from a library. See Device Driver Libraries for details on creating and using device driver libraries.

The New Device Driver Dialog



Tag: PIC Daisy (A0)
Name:
Response prefix: :0
Response suffix: \r
Delay after commands with no response: 0.1 s
Error pattern: \?0 .*\r

Responses:
Default
Inquire
Poll Image

Commands:
Clear All
Clear Port 0
Clear Port 1
Clear Port 2
Clear Port 3
Configure Port 0: 1 In, 7 Out
Configure Port 0: 6 In, 2 Out
Configure Port 0: 6 Out, 2 In
Configure Port 0: 7 In, 1 Out
Configure Port 0: 7 Out, 1 In

- Tag:

Enter the tag by which the driver is identified. Each driver must have a unique tag.

- Name:

Enter the name of the device driver. Leave this field blank if you want to use the tag as name.

- Prefix:

Enter a regular expression for the response or message header here. Headers usually consist of a synch byte, the device addresses, and sometimes a byte count and the command opcode. U.P.M.A.C.S. will strip the prefix from the data before processing it. Leave this field blank if your responses or messages have no header, or if you want to use a data mask-specific prefix. If you specify a prefix in the device driver and in the data mask, the prefix of the data mask will be used.

See *Appendix A: Regular Expressions* on page 171 for details.

- Suffix:

Enter a regular expression for the response or message trailer here. Trailers usually consist of a checksum and often a terminating character. U.P.M.A.C.S. will strip the suffix from the data before processing it. Leave this field blank if your responses or messages have no trailer, or if you want to use a data mask-specific suffix. If you specify a suffix in the device driver and in the data mask, the suffix of the data mask will be used.

See *Appendix A: Regular Expressions* on page 171 for details.

- Delay after commands with no response:

If a command has no response, U.P.M.A.C.S. will send the next command immediately afterwards, without delay. Some equipment needs a small interval between commands to function properly. If your equipment needs a delay after a command that it does not send a response for, enter it here, in seconds. You can enter fractions of a second.

- Error pattern:

Enter a regular expression by which an error response or error message sent by the equipment can be recognized. If the equipment does not send error responses or messages, or if you want to use a data mask-specific error format, leave this field blank. If you specify an error pattern in the device driver and in the data mask, the error pattern of the data mask will be used.

The error pattern is not used in conjunction with the prefix or suffix. Include the error response header and trailer in the error pattern.

Make sure that the error responses are distinguishable from valid responses. If an error response fits the prefix, pattern, and suffix of the mask, it will be treated as data.

See *Appendix A: Regular Expressions* on page 171 for details.

- Objects for polling/Objects for unsolicited data:
Use these tabs to view responses and commands, or messages and replies. This tab only affects which objects are displayed, not which type of objects are contained in the device driver. You can always have all four types of objects in the same device driver.

- Responses:
This box shows all the responses you have defined for the device driver. Use the buttons at the bottom of the box to add, remove, and edit responses. See Responses for a description of the New Response dialog. If the dialog shows messages instead of responses, click on "Objects for polling" to view responses and commands.

- Commands:
This box shows all the commands you have defined for the device driver. Use the buttons at the bottom of the box to add, remove, and edit commands. See Commands for a description of the New Command dialog. If the dialog shows replies instead of commands, click on "Objects for polling" to view responses and commands.

- Messages (not shown):
This box shows all the messages you have defined for the device driver. Use the buttons at the bottom of the box to add, remove, and edit messages. See Messages for a description of the New Message dialog. If the dialog shows responses instead of messages, click on "Objects for unsolicited data" to view messages and replies.

- Replies (not shown):
This box shows all the replies you have defined for the device driver. Use the buttons at the bottom of the box to add, remove, and edit replies. See Replies for a description of the New reply dialog. If the dialog shows commands instead of replies, click on "Objects for unsolicited data" to view messages and replies.

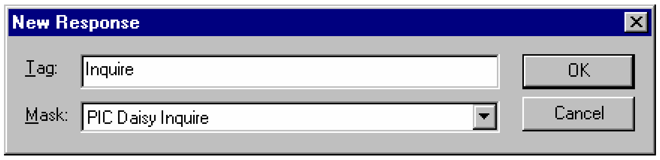
- The "Replace…" button:
The device driver dialog incorporates search and replace functionality. To bring up the Search And Replace dialog, press the "Replace…" button. See *Search and Replace in Legacy Device Drivers* on page 214 for a description of the Search And Replace dialog.

- The "Checksum…" button:
Press this button to add a checksum to all command and reply strings, or to recalculate the checksum for all command and reply strings. See *Calculating Checksums* on page 217 for a description of the Add/Change Checksum dialog.

Legacy Responses

Responses provide room for data received from the equipment in response to a command. Each device creates a data buffer for each response in its driver. Device responses that look alike but contain different data need separate responses.

The New Response Dialog



Tag: Inquire
Mask: PIC Daisy Inquire

- Tag:

Enter the tag by which the response is identified. Each response and message in a driver must have a unique tag. A response cannot have the same tag as another response or a message.

- Mask:

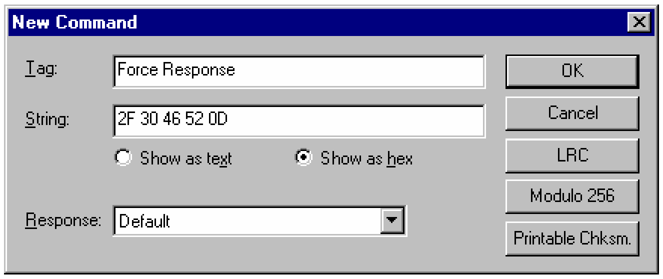
Select the data mask to use for this response.

Legacy Commands

Commands provide predefined command strings used for polling and controls. You need a command object for each separate string you want to send during the polling and initialization sequences. You should also provide additional commands for use in controls and for testing.

Commands always have a fixed data string that is sent. U.P.M.A.C.S. v5.5 does not support commands with variable parameters. If you have a command that needs parameters, you must either define a separate command object for each possible parameter value, or you must construct the command on the fly in an SCL program. The latter option is only possible within controls: The polling and initialization sequences only support pre-defined commands.

The New Command Dialog


New Command

Tag: Force Response

String: 2F 30 46 52 0D

Show as text [✓ Show as hex]

Response: Default

- Tag:

Enter the tag by which the command is identified. Each command and reply in a driver must have a unique tag. A command cannot have the same tag as another command or a reply.

- String:

Enter the data to be sent here. See *Appendix B: Entering Binary Data* on page 179 for details.

- Show as text, Show as hex:

Select the way you want to enter the command string. See *Appendix B: Entering Binary Data* on page 179 for details.

- Response:

Select the response to use for this command. Select <none> if the equipment does not send data in response to this command. Do not select <none> if the equipment sends data, even if you intend to discard the data.

- The "LRC" button:

Press this button to add an LRC checksum to the command string. LRC stands for Longitudinal Redundancy Check, and is calculated by XORing all the character codes in the string together.

- The "Modulo 256" button:

Press this button to add a modulo 256 checksum to the command string. This checksum is calculated by adding all the character codes in the string, and taking the sum modulo 256.

- The "Printable Chksm." button:

This is a special checksum designed to produce a checksum that is always a printable character. It is calculated as follows:

- Subtract 32 from each character code,
- add the results together,
- take the sum modulo 95,
- add 32 to the result.

This checksum is used, for example, by Miteq equipment.

Example:

To calculate the checksum for the string "Prometheus", proceed as follows:

Subtract 32 from each character code:

Char:	P	r	o	m	e	t	h	e	u	s
Code:	80	114	111	109	101	116	104	101	117	115
	-32	-32	-32	-32	-32	-32	-32	-32	-32	-32
result:	48	82	79	77	69	84	72	69	85	83

Add the results together:

48 + 82 + 79 + 77 + 69 + 84 + 72 + 69 + 85 + 83 = 748

Take the sum modulo 95:

748 mod 95 = 83

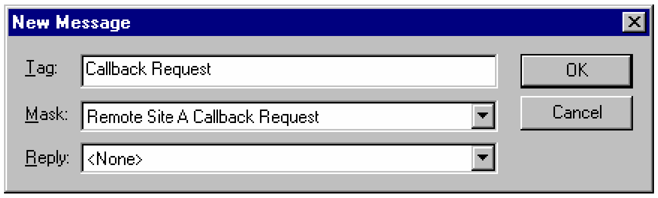
And add 32 to the result:

83 + 32 = 115 (" s ")

Legacy Messages

Messages describe messages that the equipment might send to U.P.M.A.C.S. out of its own accord and provide room to store the data received. Each device creates a data buffer for each message in its driver. Device messages that contain different data need separate message objects. Each message should use a different data mask to make it possible to determine which message an incoming data block is.

The New Message Dialog


New Message

Tag: Callback Request

Mask: Remote Site A Callback Request

Reply: <None>

- Tag:

Enter the tag by which the message is identified. Each message and response in a driver must have a unique tag. A message cannot have the same tag as another message or a response.

- Mask:

Select the data mask to use for this message.

- Reply:

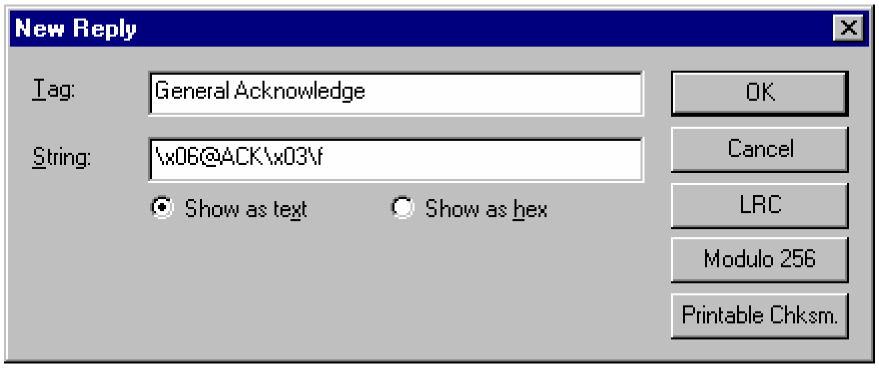
Select the response to send in response to this message. Select <none> if the equipment does not expect a reply to this message, or if you want to generate a dynamic reply using an SCL program.

Legacy Replies

Replies provide predefined reply strings used to reply to messages.

Replies always have a fixed data string that is sent. U.P.M.A.C.S. v5.5 does not support replies with variable parameters. If you have a reply that needs parameters, you must either define a separate reply object for each possible parameter value, or you must construct the reply on the fly in an SCL program. (You can specify an SCL program used to process incoming messages on a port or device, and that program can send custom data in reply to a message. See Serial Ports for details.)

The New Reply Dialog



Tag: General Acknowledge

String: \x06@ACK\x03\f

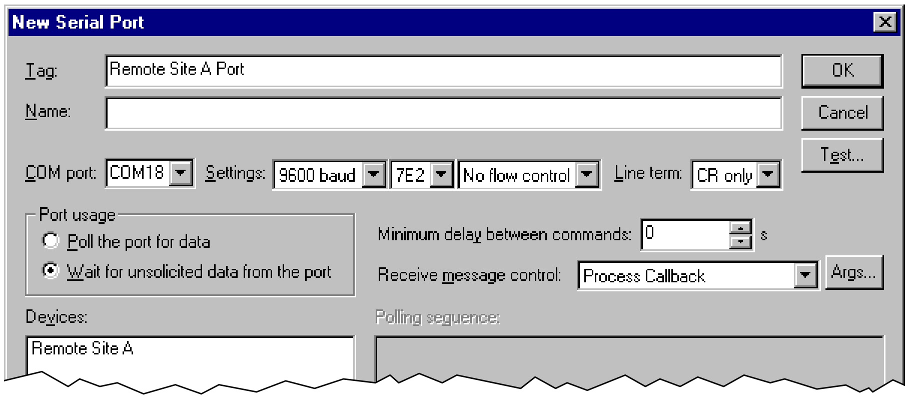
[✓ Show as text] Show as hex

- Tag:
Enter the tag by which the reply is identified. Each reply and command in a driver must have a unique tag. A reply cannot have the same tag as another reply or a command.
- String:
Enter the data to be sent here. See *Appendix B: Entering Binary Data* on page 179 for details.
- Show as text, Show as hex:
Select the way you want to enter the reply string. See *Appendix B: Entering Binary Data* on page 179 for details.
- The "LRC" button:
Press this button to add an LRC checksum to the reply string. LRC stands for Longitudinal Redundancy Check, and is calculated by XORing all the character codes in the string together.
- The "Modulo 256" button:
Press this button to add a modulo 256 checksum to the reply string. This checksum is calculated by adding all the character codes in the string, and taking the sum modulo 256.
- The "Printable Chksm." button:
This is a special checksum designed to produce a checksum that is always a printable character. See *The "Printable Chksm." button* on page 196 for details on printable checksums.

Serial Ports

Serial ports provide capabilities for triggering an SCL Program when a message is received.

The New Serial Port Dialog



Tag: Remote Site A Port
Name:
COM port: COM18
Settings: 9600 baud 7E2 No flow control
Line term: CR only
Port usage
( ) Poll the port for data
(•) Wait for unsolicited data from the port
Minimum delay between commands: 0 s
Receive message control: Process Callback
Devices:
Remote Site A
Polling sequence:

The list below describes only those fields pertaining to the message control. For the remaining fields, see *Serial Ports* on page 13.

- Receive message control:

This field is only enabled if the port is configured to wait for unsolicited data. You can select a program here that will be executed every time a valid message is received on the port. The program arguments are shown in parentheses after the name of the program, but you only select the program from the lists, not the arguments. To change the arguments, use the "Args…" button. See Specifying Arguments for SCL Programs for a description of the Edit Program Arguments dialog.

You can use the control to decode the message and perform control actions depending on its content, or to send custom replies to messages that require non-fixed data in the reply.

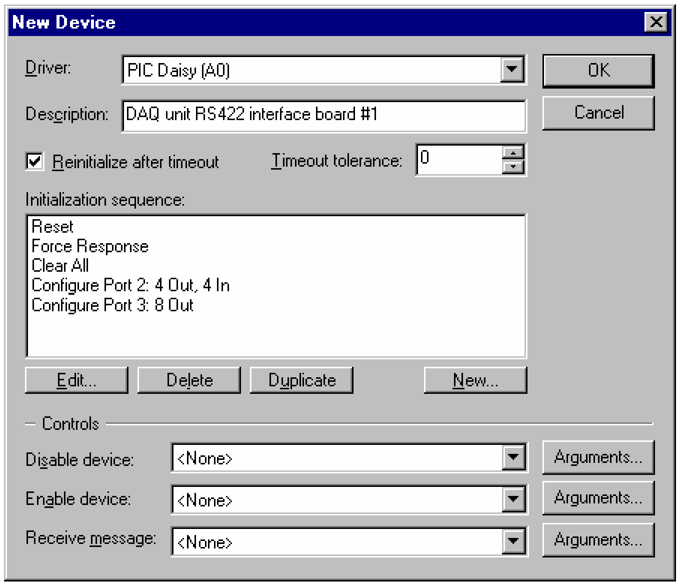
You can also specify a separate program for each device in the device's configuration dialog. If you specify a control in the port and a device, the device's control will be used.

Legacy Devices

Devices provide information about a piece of equipment connected to a serial port. The device specifies which driver to use, and how to initialize the equipment.

You can also specify automatic controls to be executed when the device is enabled or disabled. Disabling a device from the Devices dialog in the Operate System or from an SCL program automatically masks all registers associated with the device, but it is sometimes necessary to perform additional maintenance or to mask additional registers. The automatic controls allow you to do this.

The New Device Dialog


New Device

Driver: PIC Daisy (A0)

Description: DAQ unit RS422 interface board #1

☑ Reinitialize after timeout

Timeout tolerance: 0

Initialization sequence:

Reset
Force Response
Clear All
Configure Port 2: 4 Out, 4 In
Configure Port 3: 8 Out

Controls

Disable device: <None>

Enable device: <None>

Receive message: <None>

- Driver:

Select the device driver that the device uses. Drivers that are already used by devices in the port do not appear in the list. If you change the driver, all polls of the initialization sequence that use commands with names that don't exist in the new driver will be removed. You can only change the driver when creating a new device, not when you are configuring an existing one.

- Description:

Enter a description by which the user can recognize the device. This will usually be something like "Modem 1" or "UPS". Make sure all devices on all ports have different descriptions.

- Reinitialize after timeout:

Check this box if you want U.P.M.A.C.S. to re-send the initialization sequence every time the device times out. If the device does not have an initialization sequence, this check box is ignored.

The initialization sequence is not sent immediately after a timeout. U.P.M.A.C.S. will send the initialization sequence before the next time the polling sequence is started.

- Timeout tolerance:

Enter the number of consecutive timeouts to ignore here. If your equipment times out sporadically, but you do not want all of these timeouts to be reported, enter the number of timeouts you allow here. If you enter 3, e.g., the device has to time out four times in a row for a timeout to be reported.

- Initialization sequence:

This field is only enabled if the serial port is configured to poll equipment, rather than wait for unsolicited data. It shows the sequence of commands used to initialize the equipment. Use the buttons in the box to add, remove, and edit polls. The polls will be sent in the order shown. You can change the order of the polls by grabbing them with the mouse and dragging them to a new position.

See The New Initialization Command Dialog for a description of the New Initialization Command dialog.

- Controls:

Select the automatic controls for enabling and disabling the device here, as well as the receive message control for ports that wait for unsolicited data. The program arguments are shown in parentheses after the names of the programs, but you only select the programs from the lists, not the arguments. To change the arguments, use the “Args…” buttons. See Specifying Arguments for SCL Programs for a description of the Edit Program Arguments dialog.

Disable device:
Select the SCL program to be executed when the device is disabled. This control is also executed on startup if the device is initially disabled.

Enable device:
Select the SCL program to be executed when the device is enabled. This control is not executed on startup if the device is initially enabled.

Receive message:
This field is only enabled if the serial port is configured to wait for unsolicited data, rather than for polling the equipment. You can select a program here that will be executed every time a valid message is received from the device. You can use the control to decode the message and perform control actions depending on its content, or to send custom replies to messages that require non-fixed data in the reply.

You can also specify a global program to be used for all devices in the New Serial Port dialog. If you specify a control in the device and the serial port, the device’s control will be used.

Legacy Sources

Types of Legacy Sources

There are two types of legacy source that all registers share:

- Legacy device processor sources
- Legacy parameter sources

Bistate registers can have the following additional source types:

- Legacy device bit mask sources
- Legacy device search string sources

Digital registers can have the following additional source types:

- Legacy device direct sources
- Legacy device thresholds sources
- Legacy device strings sources
- Legacy device bit collection sources

Analog registers can have the following additional source type:

- Legacy device level sources

String registers can have the following additional source types:

- Legacy device literal sources

- Legacy device filter sources

All of the sources except for legacy parameter sources use data from a response or message buffer of a device on a serial port. You can specify which part of the data is to be used, and how.

Specifying the Relevant Part of the Serial Data

There are two ways in which to specify the part of the data that you wish to use.

- Using an offset from the beginning of the data:

You can specify a byte offset from the beginning of the data buffer. The section of the data used by the source will begin a fixed number of bytes from the beginning of the data. Use this method if you know how many bytes from the beginning of the data you are interested in is located.

- Using a search key:

You can use a search key to specify the beginning of the relevant part of the data. The section of the data used by the source will begin a fixed number of bytes before or after the end of the first occurrence of the search key in the data. Use this method if the data in the response or message is variable length and separated by separators, like spaces or commas. You can also use this method if the data is preceded by some character or characters: a piece of equipment might mark the position of a frequency with the letter F, for example.

If you know how many bytes the data you are interested in occupies, you can also specify a length. If you do not specify a length, the section of data used extends to the end of the available data. In certain cases, you will have to specify a fixed length.

Numerical Data Types

If the source has to translate the data into a number, you can choose one of the following translation methods:

Byte (unsigned):
This type requires a fixed length of 1 byte. The byte value is interpreted as an unsigned number ranging from 0 to 255.

Byte (signed):
This type requires a fixed length of 1 byte. The byte value is interpreted as an signed number ranging from -128 to 127.

Multibyte (lo-hi, unsigned):
This type requires a fixed length of 2, 3, or 4. The bytes are interpreted as a 16, 24, or 32 bit unsigned number using the first byte as the least significant byte (low byte), and the last byte as the most significant byte (high byte). The resulting number ranges are as follows:

- 2 bytes: 0 to 65,535
- 3 bytes: 0 to 16,777,215
- 4 bytes: 0 to 4,294,967,295

Multibyte (lo-hi, signed):
This type requires a fixed length of 2, 3, or 4. The bytes are interpreted as a 16, 24, or 32 bit signed number using the first byte as the least significant byte (low byte), and the last byte as the most significant byte (high byte). The resulting number ranges are as follows:

- 2 bytes: -32,768 to 32,767
- 3 bytes: -8,388,608 to 8,388,607

- 4 bytes: -2,147,483,648 to 2,147,483,648

Multibyte (hi-lo, unsigned):
This type requires a fixed length of 2, 3, or 4. The bytes are interpreted as a 16, 24, or 32 bit unsigned number using the first byte as the most significant byte (high byte), and the last byte as the least significant byte (low byte). The resulting number ranges are as follows:

- 2 bytes: 0 to 65,535
- 3 bytes: 0 to 16,777,215
- 4 bytes: 0 to 4,294,967,295

Multibyte (hi-lo, signed):
This type requires a fixed length of 2, 3, or 4. The bytes are interpreted as a 16, 24, or 32 bit signed number using the first byte as the most significant byte (high byte), and the last byte as the least significant byte (low byte). The resulting number ranges are as follows:

- 2 bytes: -32,768 to 32,767
- 3 bytes: -8,388,608 to 8,388,607
- 4 bytes: -2,147,483,648 to 2,147,483,648

BCD:
This type requires a fixed length. The data is interpreted as a Binary Coded Decimal number. In binary coded decimal, each nibble (hex digit) in a byte represents one decimal digit. Four bytes of data containing the following four byte values:

$20 $34 $00 $50

represents the number 20,340,050.

Decimal:
This type interprets the data as a number written out in base 10 as a series of the ASCII characters "0" to "9". Leading spaces will be ignored, as will any characters that appear after the number.

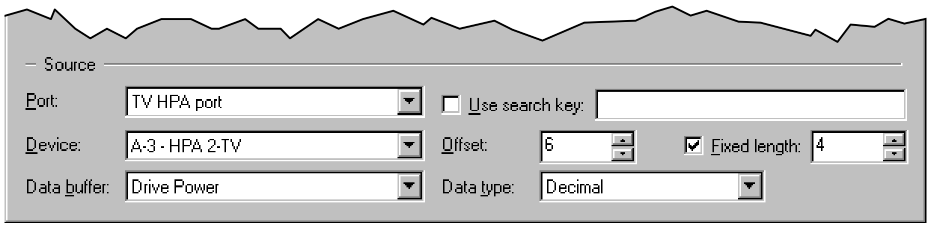
Hexadecimal:
This type interprets the data as a number written out in base 16 as a series of the ASCII characters "0" to "9" and "A" to "F" or "a" to "f". Leading spaces will be ignored, as will any characters that appear after the number.

Binary:
This type interprets the data as a number written out in base 2 as a series of the ASCII characters "0" and "1". Leading spaces will be ignored, as will any characters that appear after the number.

Octal:
This type interprets the data as a number written out in base 8 as a series of the ASCII characters "0" to "7". Leading spaces will be ignored, as will any characters that appear after the number.

The Legacy Device Serial Data Source Dialogs

The Source dialogs for sources that use serial data share the following fields:



Source
Port: TV HPA port
Device: A-3 - HPA 2-TV
Data buffer: Drive Power
☐ Use search key:
Offset: 6
☑ Fixed length: 4
Data type: Decimal

The list below describes only those fields that all the dialogs have. For the remaining fields, see the section on the appropriate source.

- Port:

Select the serial port to which the device is attached.

- Device:

Select the device.

- Data buffer:

Select the response or message whose data you wish to use.

- Use search key:

Check the check box if you want to use a search key. If you check this box, the offset will be calculated from the end of the first match for the search key. Enter a regular expression for the key into the entry field.

See *Appendix A: Regular Expressions* on page 171 for details.

- Offset:

Enter the zero-based offset of the data within the buffer. If you did not specify a search key, the offset is calculated from the beginning of the buffer. If you specified a search key, the offset is calculated from the end of the first match for the search key.

If the data you are interested in is right at the beginning of the buffer, or right after the search key, enter 0.

If there are n bytes between the beginning of the buffer or end of the search key and the data, enter n.

If the data includes the last n bytes of the search key, enter -n.

Note: If you do not specify a search key, the offset is calculated from the beginning of the response or message data, not the beginning of the prefix. The prefix and suffix are stripped from a response or message before any processing is done.

- Fixed length:

Check this box if you know the length of the data, in bytes. Enter the number of bytes in the entry field.

If you selected either of the Byte data types you must specify a length of 1. If you selected one of the Multibyte data types, you must specify a length of 2, 3, or 4. If you selected BCD as data type, you must specify a length, but the length can be anything.

If you selected any other data type, or if the source does not use data types, and you do not specify a length, the data will be assumed to extend to the end of the response.

Since the Decimal, Hexadecimal, Octal, and Binary data types ignore any characters that appear after the number, it is not necessary to specify a length if the number is followed by a terminating character. If, however, the number is followed immediately by something that may be interpreted as an ASCII digit, you must specify a length.

- Data type:

Not all of the Serial Data Source dialogs have this field. The Processor Source dialog, the Search String Source dialog, the Strings Source dialog, the Literal Source dialog, and the Filter Source dialog do not have this field.

Select the data type here. See *Numerical Data Types* on page 202 for details.

Legacy Device Processor Sources

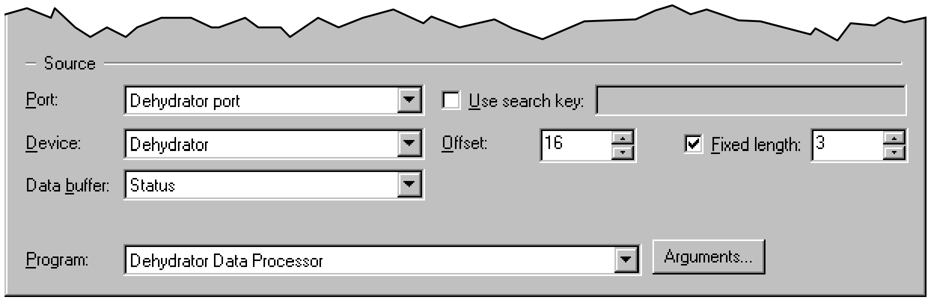
Legacy device processor sources take information from a response data buffer in a serial device. You have to provide an SCL program to decode the data. See Programs for Sources, Checksums, and SABus Response Data in the SCL Programming Language Help for details.

If the response buffer does not contain data because of a timeout, the register will go into its error state.

If the device that the information comes from is disabled, or if all commands that use the response specified are disabled, the register will be auto masked.

Use legacy device processor sources for registers whose value comes from data in a legacy response, but none of the other legacy device sources types can be used to interpret the data.

The Processor Source Dialog


Source
Port: Dehydrator port
☐ Use search key:
Device: Dehydrator
Offset: 16
☑ Fixed length: 3
Data buffer: Status
Program: Dehydrator Data Processor

The list below describes only those fields that are specific to the Processor Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Program:

Select the SCL program that is to do the evaluation of the data. The program arguments are shown in parentheses after the name of the program, but you only select the program from the lists, not the arguments. To change the arguments, use the "Arguments…" button. See Specifying Arguments for SCL Programs for a description of the Edit Program Arguments dialog.

Legacy Parameter Sources

Legacy parameter sources take information from a legacy parameter. The value of the register will be determined as follows:

- Bistate registers:

The register will be in the ON state if the parameter's value is "ON". The register will be in the OFF state if the value is "OFF". Any other value will cause the register to go into the error state. The comparison is not case sensitive, i.e. "On", "on", and "oN" will also cause the register to go into its ON state.

- Digital registers:

If the value of the register contains a number within the range of digital registers, the register's value will be set to that number. Otherwise, the register will go into its error state.

The number can either be a decimal number, a hexadecimal number prefixed with "$", an octal number prefixed with "&", or a binary number prefixed with "%".

- Analog registers:

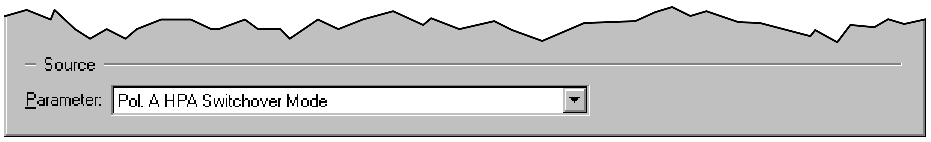
If the value of the register contains a number written out in decimal format, the register's value will be set to that number. Otherwise, the register will go into its error state.

- String registers:

The register will contain the value of the parameter.

Use legacy parameter sources for registers that represent a value maintained by U.P.M.A.C.S. internally, and that must remain unchanged if U.P.M.A.C.S. is restarted.

The Parameter Source Dialog



Source

Parameter: Pol. A HPA Switchover Mode

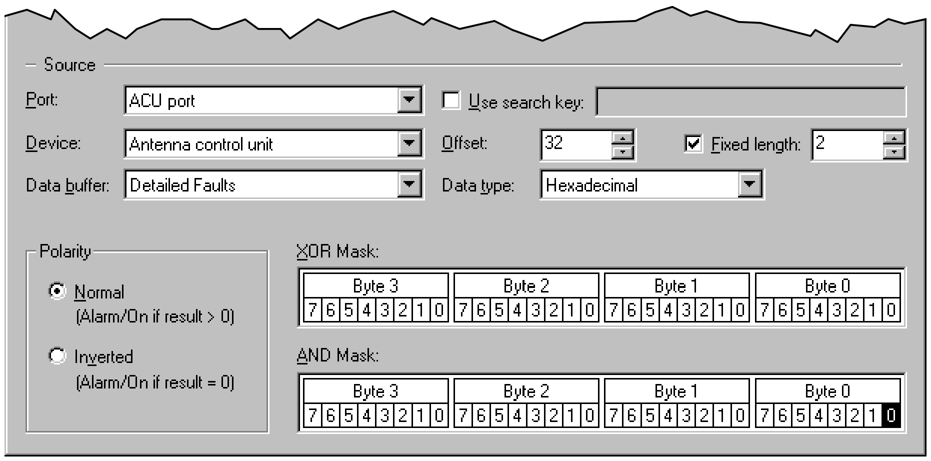
- Parameter:

Select the parameter to use.

Legacy Device Bit Mask Sources

Legacy device bit mask sources are just like regular bit mask sources, except that they use data from a legacy response or message.

The Bit Mask Source Dialog



Source

Port: ACU port

☐ Use search key:

Device: Antenna control unit

Offset: 32

☑ Fixed length: 2

Data buffer: Detailed Faults

Data type: Hexadecimal

Polarity

(●) Normal (Alarm/On if result > 0)

( ) Inverted (Alarm/On if result = 0)

The list below describes only those fields that are specific to the Processor Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

- AND mask:

Select the bits you want to use. The least significant bit and byte are shown on the right. Black bits will be used; white bits will be masked out to 0s. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

- Polarity:

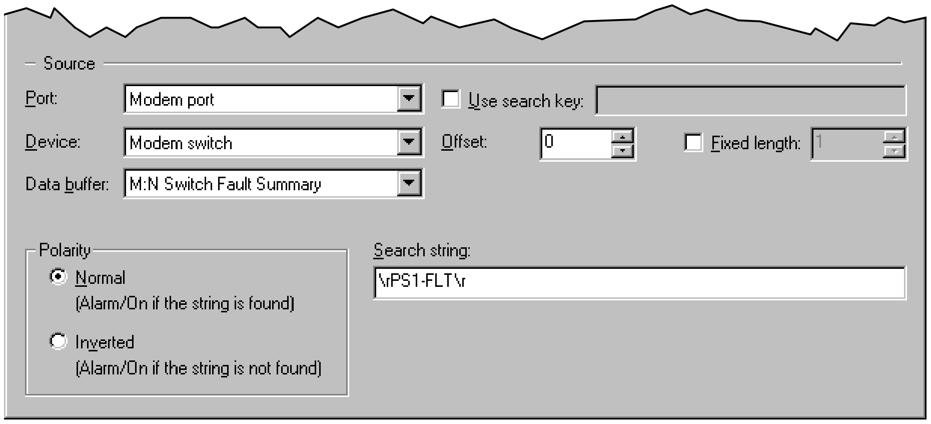
Select the polarity of the register. An alarm can either be triggered by any result greater than 0 (normal polarity), or by the result 0 only (inverted polarity).

Legacy Device Search String Sources

Registers with legacy device search string sources are set to the ON or OFF state depending on whether a regular expression was found in the data section or not.

Use legacy device search string sources to implement alarms that are triggered or cleared when data sent by a piece of equipment contains certain keywords. This is useful, for example, for equipment that returns a string listing all currently active fault conditions.

The Search String Source Dialog


Source
Port: Modem port
☐ Use search key:
Device: Modem switch
Offset: 0
☐ Fixed length: 1
Data buffer: M:N Switch Fault Summary
Polarity
[✓ Normal] (Alarm/On if the string is found)
Inverted (Alarm/On if the string is not found)
Search string: \rPS1-FLT\r

The list below describes only those fields that are specific to the Search String Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Search for:

Enter the regular expression to search for.

See *Appendix A: Regular Expressions* on page 171 for details.

- Polarity:

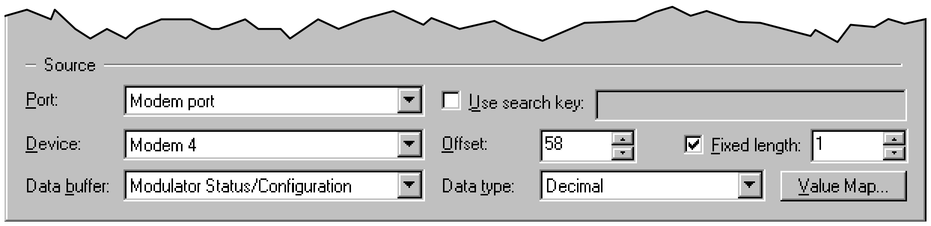
Select the polarity of the register. An alarm can either be triggered if the regular expression is found (normal polarity), or if it is not found (inverted polarity).

Legacy Device Direct Sources

Legacy device direct sources use a number extracted from a response or message as the value of a digital register. Use legacy device direct sources for digital registers whose value appears as is in a response or message.

You can also specify a value map for the source. Sometimes, you want the register to have different values from the number extracted from the buffer. Other times, different values in the buffer correspond to the same register value. A value map will allow you to substitute other values for values calculated by the source. Normally, the result of the calculation described above is used directly as the value of the register. If the result is 2, the value of the register will also be 2. If, however, you would prefer the value of the register to be 4 if the result is 2, then you could specify a value map that maps a source value of 2 to a register value of 4. You can specify register values to be substituted for any number of source values.

The Direct Source Dialog


Source

Port: Modem port

Use search key: [ ]

Device: Modem 4

Offset: 58

Fixed length: [✓] 1

Data buffer: Modulator Status/Configuration

Data type: Decimal

The Direct Source dialog contains only the fields common to all serial data Source dialogs. See The Legacy Device Serial Data Source Dialogs for details on the fields of this dialog.

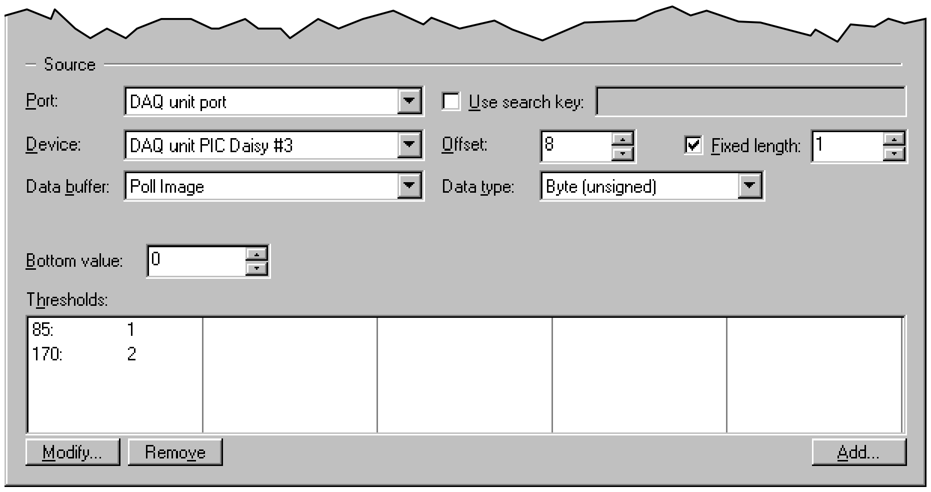
- The "Value Map" button:

Click this button to edit the value map. The value map allows you to substitute different values for the values extracted from the response.

Legacy Device Thresholds Sources

Legacy device thresholds sources are just like regular thresholds sources, except that they use data from a legacy response or message.

The Thresholds Source Dialog


Source

Port: DAQ unit port

Use search key: [ ]

Device: DAQ unit PIC Daisy #3

Offset: 8

Fixed length: [✓] 1

Data buffer: Poll Image

Data type: Byte (unsigned)

Bottom value: 0

Thresholds:

85:	1
170:	2

The list below describes only those fields that are specific to the Thresholds Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Bottom value:

Enter the value that the register should take if the number lies below any of the thresholds.

- Thresholds:

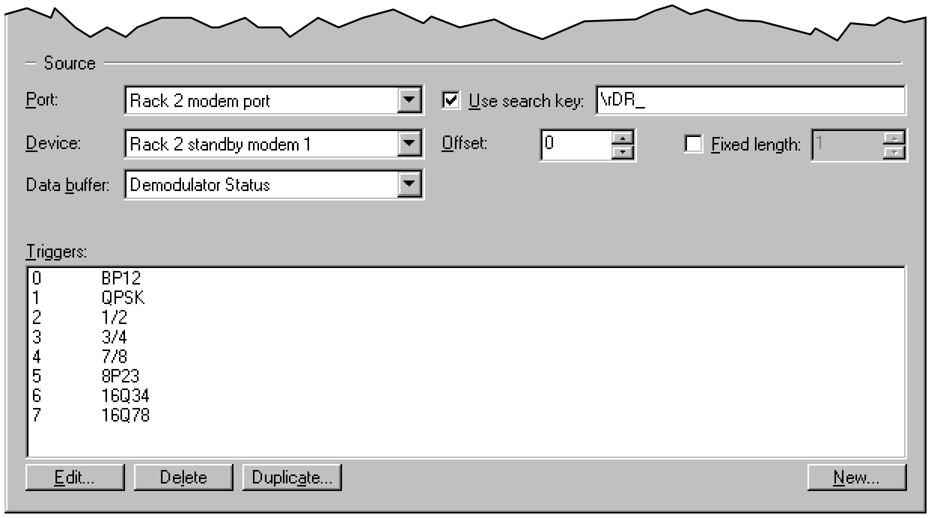
Lists all the thresholds and their corresponding values. Use the buttons to add, remove, or modify thresholds and their values.

Legacy Device Strings Sources

Legacy Device Strings sources translate the data section into a value for a digital register using a set of regular expressions, called “triggers.” If the data matches one of the triggers, the register’s value is set to the value with which it is associated. If none of the triggers match, the register goes into the error state.

Use legacy device strings sources for registers whose value reflects a set of states that are represented by different strings in a response or message.

The Strings Source Dialog



Port: Rack 2 modem port

[✓] Use search key: \rDR_

Device: Rack 2 standby modem 1

Offset: 0

[ ] Fixed length: 1

Data buffer: Demodulator Status

Triggers:

0	BP12
1	QPSK
2	1/2
3	3/4
4	7/8
5	8P23
6	16Q34
7	16Q78

The list below describes only those fields that are specific to the Strings Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Triggers:

Lists all the possible values and the regular expressions that trigger them. The register will assume the first value whose regular expression matches the data.

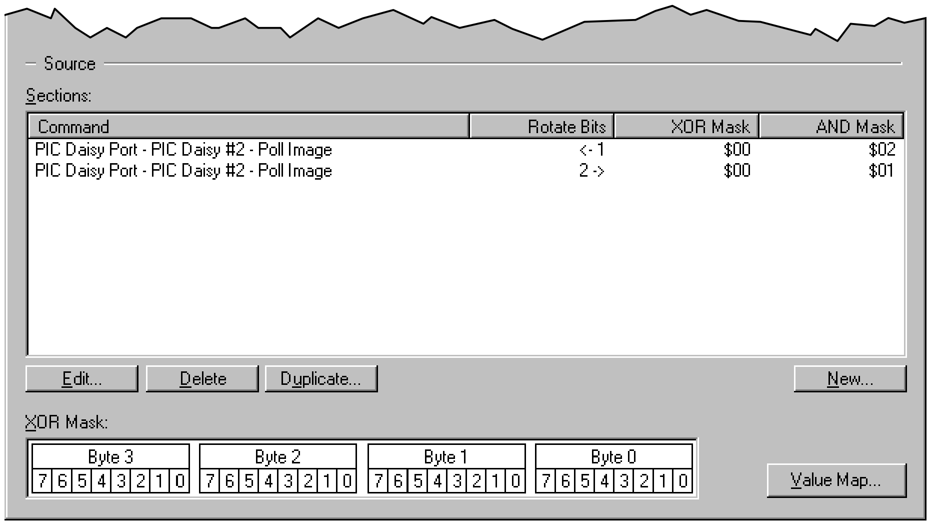
See *Appendix A: Regular Expressions* on page 171 for details.

Use the buttons to create, delete, and edit the expressions. Use the “Duplicate…” button to duplicate the selected trigger.

Legacy Device Bit Collection Sources

Legacy device bit collection sources are just like regular bit collection sources, except that their bit sections use legacy devices.

The Bit Collection Source Dialog



Command	Rotate Bits	XOR Mask	AND Mask
PIC Daisy Port - PIC Daisy #2 - Poll Image	<- 1	$00	$02
PIC Daisy Port - PIC Daisy #2 - Poll Image	2 ->	$00	$01

- Sections:

Lists the commands, rotate values, and XOR and AND masks of all the sections. Use the buttons to create, delete, or edit sections. Use the "Duplicate…" button to duplicate the selected section.

See *Legacy Device Bit Sections* below for a description of the New Bit Section dialog.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte.

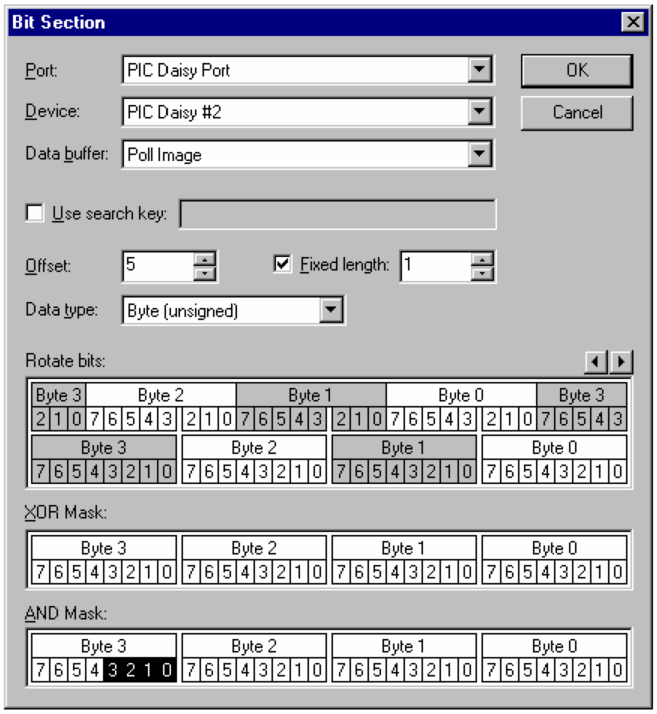
- The "Value Map" button:

Click this button to edit the value map. The value map allows you to substitute different values for the values calculated by the source.

Legacy Device Bit Sections

Legacy device bit sections are just like regular bit sections, except that they use data from a legacy response or message.

The Bit Section Dialog



Port: PIC Daisy Port
Device: PIC Daisy #2
Data buffer: Poll Image
☐ Use search key:
Offset: 5
☑ Fixed length: 1
Data type: Byte (unsigned)

Although legacy device bit sections are not sources in themselves, but elements of a legacy device bit collection source, the Bit Section dialog has all the fields that you would find in a Legacy Device Serial Data Source dialog. The list below describes only those fields that are specific to the Bit Section dialog. For a description of the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Rotate bits:

Specify the number of bits to rotate to the left or right. Press the ◀ ▶ arrows on the top right until the bits align the way you would like them.

- XOR mask:

Select the bits you want to invert. The least significant bit and byte are shown on the right. Black bits will be inverted; white bits will not be inverted. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte. The masks are applied after the bit rotation.

- AND mask:

Select the bits you want to use. The least significant bit and byte are shown on the right. Black bits will be used; white bits will be masked out to 0s. Click on a bit to toggle it, click on the byte number above the bits to set or clear all the bits in that byte. The masks are applied after the bit rotation.

Legacy Device Level Sources

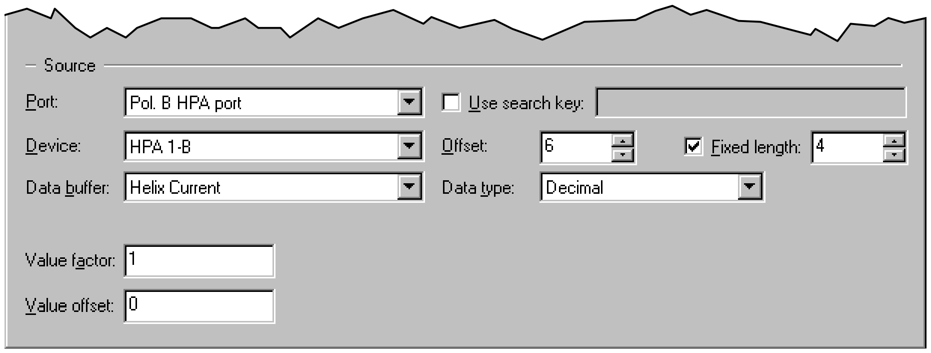
Legacy device level sources use a number extracted from a data buffer as the value of an analog register. The decimal data type for level sources supports decimal points and exponent notation. You can provide a factor and an offset for the value. The value of the register is calculated from the number extracted from the buffer as follows:

value = number · factor + offset

To use the number unaltered, specify a factor of 1 and an offset of 0.

Use legacy device level sources for analog registers whose value appears as is in a legacy response or message.

The Level Source Dialog



Port: Pol. B HPA port
☐ Use search key:
Device: HPA 1-B
Offset: 6
☑ Fixed length: 4
Data buffer: Helix Current
Data type: Decimal
Value factor: 1
Value offset: 0

The list below describes only those fields that are specific to the Level Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Value factor:

Enter the factor with which the number from the buffer is to be multiplied before writing it to the register. The value factor is applied before the value offset.

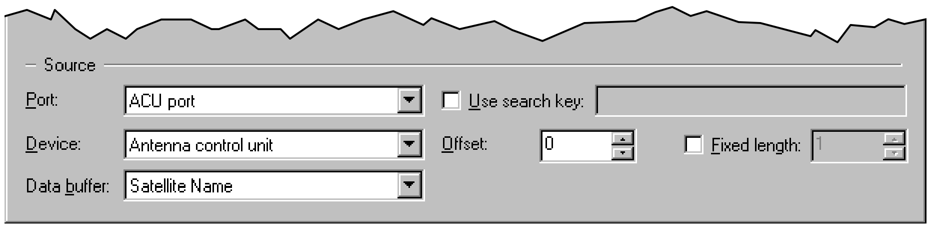
- Value offset:

Enter the offset that is to be added to the number from the buffer before writing it to the register. The value offset is applied after the value factor.

Legacy Device Literal Sources

Legact device literal sources use the data section from the data buffer as the value of the string register. Use legacy device literal sources for string registers whose value appears as is in a legacy response or message.

The Literal Source Dialog



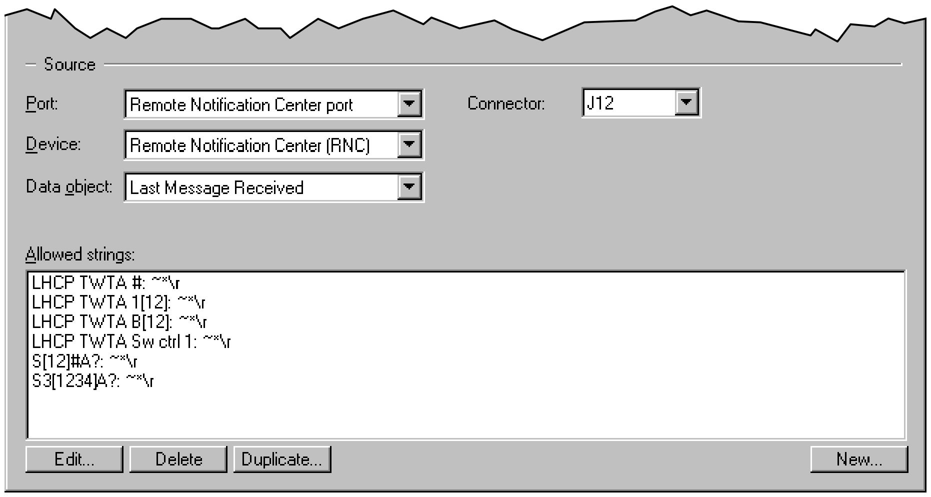
Port: ACU port
☐ Use search key:
Device: Antenna control unit
Offset: 0
☐ Fixed length: 1
Data buffer: Satellite Name

The Literal Source dialog contains only the fields common to all serial data Source dialogs. See The Legacy Device Serial Data Source Dialogs for details on the fields of this dialog.

Legacy Device Filter Sources

Legacy device filter sources are just like regular filter sources, except that they use data from a legacy response or message.

The Filter Source Dialog



Port: Remote Notification Center port

Connector: J12

Device: Remote Notification Center (RNC)

Data object: Last Message Received

Allowed strings:

```
LHCP TWTA #: ~*\r
LHCP TWTA 1[12]: ~*\r
LHCP TWTA B[12]: ~*\r
LHCP TWTA Sw ctrl 1: ~*\r
S[12]#A?: ~*\r
S3[1234]A?: ~*\r
```

The list below describes only those fields that are specific to the Filter Source dialog. For the remaining fields, see The Legacy Device Serial Data Source Dialogs.

- Allowed strings:

Lists regular expressions describing patterns that the data must match. The value of the register will only be changed if the data matches one of the allowed strings.

See *Appendix A: Regular Expressions* on page 171 for details.

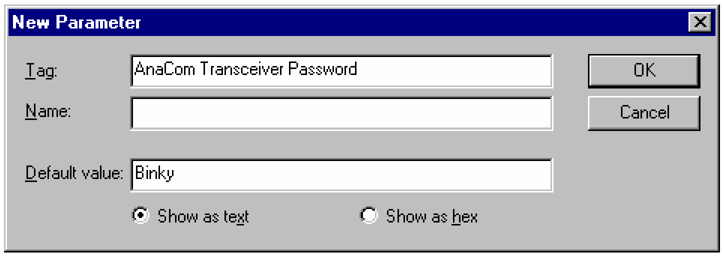
Use the buttons to create, delete, and edit the expressions.

Legacy Parameters

Legacy parameters contain persistent data, that is data that will be remembered from one launch of U.P.M.A.C.S. to the next, is stored in parameters. Parameters have very limited functionality in themselves, and are meant to be used by SCL programs and by registers with legacy parameter sources.

A parameter contains arbitrary data. The first time a station file is used on a particular computer, the parameter is initialized to a default value. After that, the value of the parameter will remain the same across launches of U.P.M.A.C.S.

The New Parameter Dialog



Tag: AnaCom Transceiver Password

Name:

Default value: Binky

[✓ Show as text] Show as hex

- Tag:

Enter the tag by which the parameter is identified. Each parameter must have a unique tag.

- Name:

Enter the name of the parameter. Leave this field blank if you want to use the tag as name.

- Default value:

Enter the value that the parameter should have when the station file is used for the first time.

- Show as text, Show as hex:

Select the way you want to enter the default value. See *Appendix B: Entering Binary Data* on page 179 for details.

Search and Replace in Legacy Device Drivers

The New Device Driver dialog incorporates search and replace capabilities. To pop up the Search And Replace dialog, press the "Search…" button. The dialog enables you to change the device address, perform a search and replace in all command and reply strings, replace one response or reply with another, or replace one data mask with another.

The Search And Replace Dialog

The Search And Replace dialog has five pages that allow you to do different kind of replacements. The pages are described in detail below. To show a particular page, click on the corresponding tab at the top of the dialog.

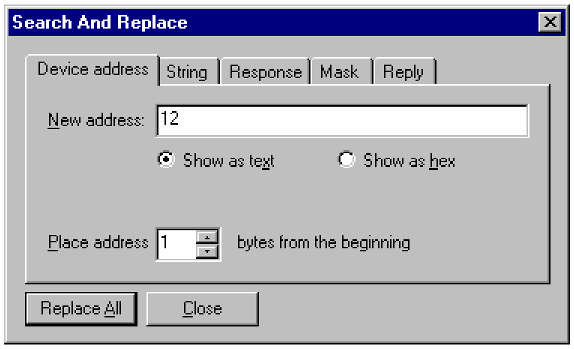
- The "Replace All" button:

Press this button to perform the replacement you specified in the current tab for all the commands, responses, messages, or replies.

- The "Close" button:

Press this button to close the Search And Replace dialog.

The Device Address Page



New address: 12

(● Show as text) ○ Show as hex

Place address 1 bytes from the beginning

This page allows you to replace the device address in the command and reply strings. The address will replace whatever bytes happen to be at the position you indicate. The address will replace as many bytes as it is long.

- New address:

Enter the new address here. See *Appendix B: Entering Binary Data* on page 179 for details.

- Show as text, Show as hex:

Select the way you want to enter the address. See *Appendix B: Entering Binary Data* on page 179 for details.

- Place Address … bytes from the beginning:

Enter the number of bytes that appear before the address. If there is one synch byte and one byte count byte, for example, enter 2. If the address is at the beginning of the string, enter 0.

Example:

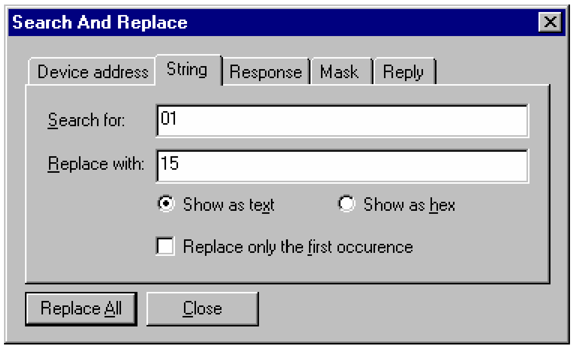
If you enter 1 in the Place "Place Address ... bytes from the beginning" field, and the address is "12" (2 characters long), then it will replace characters number 2 and 3 in every command:

{01 Status}

will become:

{12 Status}

The String Page



Search for: 01

Replace with: 15

[✓ Show as text] Show as hex

☐ Replace only the first occurence

This page allows you to replace a series of characters in the command and reply strings.

- Search for:

Enter the string to search for. See *Appendix B: Entering Binary Data* on page 179 for details.

- Replace with:

Enter the string with which to replace the search string. See *Appendix B: Entering Binary Data* on page 179 for details.

- Show as text, Show as hex:

Select the way you want to enter the strings. See *Appendix B: Entering Binary Data* on page 179 for details.

- Replace only the first occurrence:

Check this box if you want to replace the search string only once per command or reply. If you do not check this box, the search string will be replaced multiple times within the same string, if it occurs more than once.

Example:

If you search for "01" to be replaced with "15", then the string

{01 Read Channel 01}

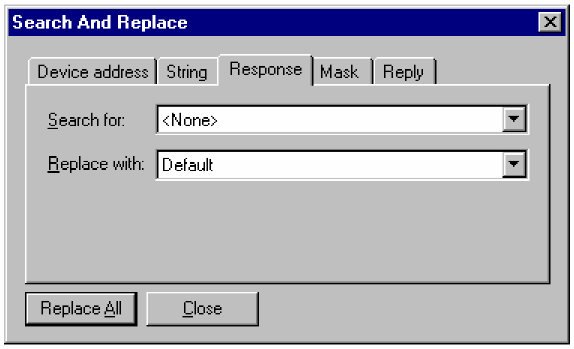
will become, if you leave the box blank:

{15 Read Channel 15}

if you check the box:

{15 Read Channel 01}

The Response Page



Search for: <None>

Replace with: Default

This page allows you to change all commands that use a specific response to use another response instead.

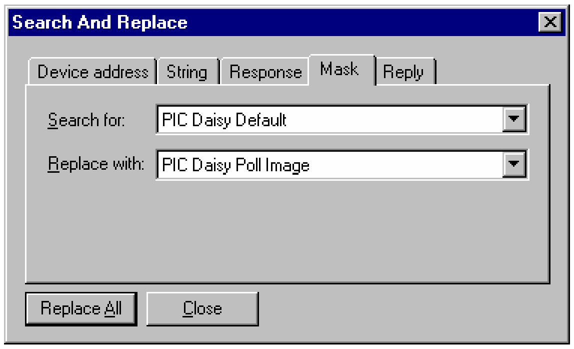
- Search for:

Enter the response to be replaced here. If you select <none>, the new response will be used by all commands that do not currently use a response.

- Replace with:

Enter the new response here. If you select <none>, the commands will be changed to use no response.

The Mask Page



Search for: PIC Daisy Default

Replace with: PIC Daisy Poll Image

This page allows you to change all responses and messages that use a specific data mask to use another mask instead.

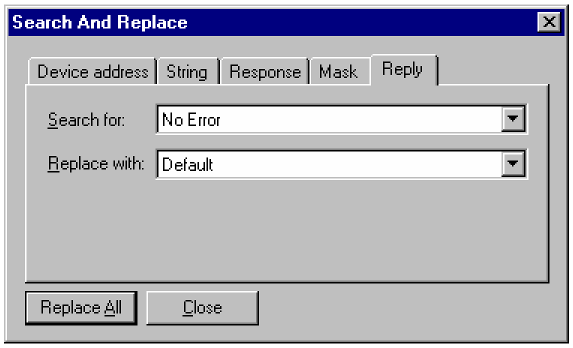
- Search for:

Enter the data mask to be replaced here.

- Replace with:

Enter the new data mask here.

The Reply Page



Search for: No Error

Replace with: Default

This page allows you to change all messages that use a specific reply to use another reply instead.

- Search for:

Enter the reply to be replaced here. If you select <none>, the new reply will be used by all messages that do not currently use a reply.

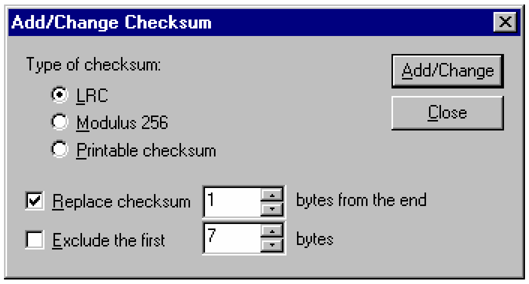
- Replace with:

Enter the new reply here. If you select <none>, the messages will be changed to use no reply.

Calculating Checksums

The New Device Driver dialog allows you to add or recalculate a checksum for all the commands and replies in the driver simultaneously. To pop up the Add/Change Checksum dialog, press the "Checksum…" button.

The Add/Change Checksum Dialog



Type of checksum: [✓ LRC] Modulus 256 Printable checksum

[✓] Replace checksum 1 bytes from the end

[ ] Exclude the first 7 bytes

- Type of Checksum:

Select the type of checksum here. See *The New Command Dialog* on page 196 for a description of the different types of checksums.

- Replace checksum … bytes from the end:

If you check this check box, an existing checksum will be recalculated. Enter the number of bytes that come after the checksum in the entry field. If the checksum you wish to recalculate is the last byte in the string, enter 0.

To add a checksum to the end, leave the check box blank.

- Exclude the first … bytes:

Check this checkbox if the checksum does not include all the characters in the command or reply string. Specify the number of bytes to exclude from the beginning. (All bytes that come after the checksum are excluded whether you check this box or not.)

- The “Add/Change” button:

Press this button to apply the changes you specified to all command and reply strings in the driver.

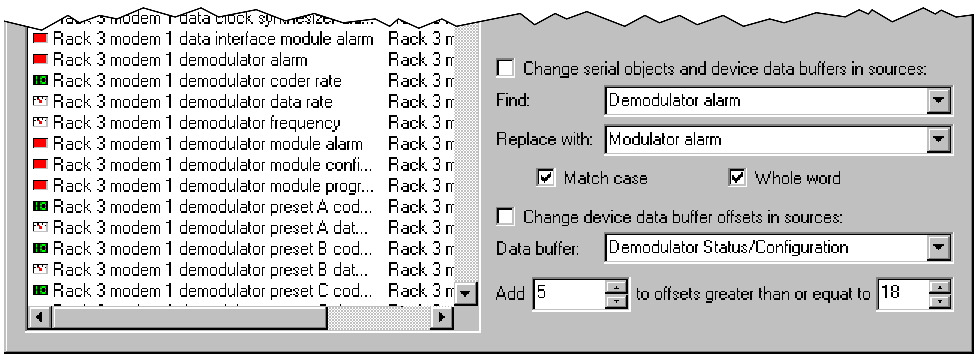
- The “Close” button:

Press this button to close the Add/Change Checksum dialog.

Batch Processing of Registers

The Modify/Duplicate Registers dialog provides capabilities for modifying legacy device data sources.

The Modify/Duplicate Registers Dialog



Rack 3 modem 1 data interface module alarm Rack 3 m
Rack 3 modem 1 demodulator alarm Rack 3 m
Rack 3 modem 1 demodulator coder rate Rack 3 m
Rack 3 modem 1 demodulator data rate Rack 3 m
Rack 3 modem 1 demodulator frequency Rack 3 m
Rack 3 modem 1 demodulator module alarm Rack 3 m
Rack 3 modem 1 demodulator module confi... Rack 3 m
Rack 3 modem 1 demodulator module progr... Rack 3 m
Rack 3 modem 1 demodulator preset A cod... Rack 3 m
Rack 3 modem 1 demodulator preset A dat... Rack 3 m
Rack 3 modem 1 demodulator preset B cod... Rack 3 m
Rack 3 modem 1 demodulator preset B dat... Rack 3 m
Rack 3 modem 1 demodulator preset C cod... Rack 3 m

☐ Change serial objects and device data buffers in sources:
Find: Demodulator alarm
Replace with: Modulator alarm
☑ Match case ☑ Whole word
☐ Change device data buffer offsets in sources:
Data buffer: Demodulator Status/Configuration
Add 5 to offsets greater than or equat to 18

The list below describes only those fields pertaining to legacy device data sources. For the remaining fields, see *Batch Processing of Registers* on page 145.

- The "Apply changes to" rectangle (not shown):

You can select the following option for legacy parameter sources:

Parameters in parameter sources (not shown):
Change the legacy parameters used by any legacy parameter sources. If the tag of any parameter used by a source contains the search text, the search text is replaced with the replace text to create a new tag. The source is modified to use the parameter that has the new tag instead.

- Change serial objects and device data buffers in sources:

Check this box to change any legacy device responses and messages used by legacy device serial data sources. Specify a search and replace string for the data buffer’s tag in the “Find” and “Replace with” fields below.

Find:
Enter the search text here. The search text in the tag of the original data buffer is replaced with the replace text to create a new tag. The source is modified to use the data buffer that has the new tag instead.

Replace with:
Enter the replace text here. The search text in the tag of the original data buffer is replaced with the replace text to create a new tag. The source is modified to use the data buffer that has the new tag instead.

Match case:
Check this box to find only text that matches the capitalization of the search string. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

Whole word:
Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

- Change device data buffer offsets in sources:

Check this box to modify the offset values in sources that use serial data from a device data buffer. See Specifying the Relevant Part of the Serial Data for details on the offsets.

Data buffer:
Enter the name of the data buffer for which offsets should be changed. The name must be an exact match.

Add … to offsets greater than or equal to:
Enter the number to add to the offset (negative to subtract), and the minimum offset value that should be changed. No offset smaller than the minimum will be changed. Enter a minimum of 0 to change all offsets.

Batch Processing of Parameters

The U.P.M.A.C.S. Development System supports modifying, duplicating and deleting multiple parameters in one operation. To manipulate batches of parameters, select "Modify/Duplicate Parameters…" from the "Special" menu.

The Modify/Duplicate Parameters dialog will appear. There are four things you can do with this dialog:

- Finding tags:

You can find and select all parameters whose tag contains a certain search text. Enter the text in the "Find" field and press the "Find All Tags" button.

- Modifying parameters:

You can modify certain properties of all selected parameters. Specify the changes you wish to make using the fields of the dialog, and press the "Replace All" button.

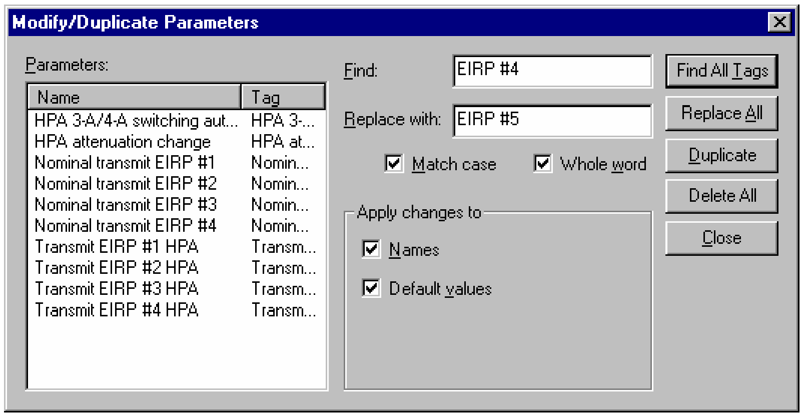
- Duplicating parameters:

You can make modified duplicates of all selected parameters. Specify the changes you wish to make using the fields of the dialog, and press the "Duplicate" button.

- Deleting parameters:

You can delete all selected parameters by pressing the "Delete All" button.

The Modify/Duplicate Parameters Dialog


Modify/Duplicate Parameters

Parameters:

Name	Tag
HPA 3-A/4-A switching aut...	HPA 3-...
HPA attenuation change	HPA at...
Nominal transmit EIRP #1	Nomin...
Nominal transmit EIRP #2	Nomin...
Nominal transmit EIRP #3	Nomin...
Nominal transmit EIRP #4	Nomin...
Transmit EIRP #1 HPA	Transm...
Transmit EIRP #2 HPA	Transm...
Transmit EIRP #3 HPA	Transm...
Transmit EIRP #4 HPA	Transm...

Find: EIRP #4

Replace with: EIRP #5

☑ Match case ☑ Whole word

Apply changes to

☑ Names

☑ Default values

▪ Parameters:
Select the parameters that you want to process. To select multiple parameters, click while holding down the Shift or Ctrl key.

The list shows both name and tag of the parameter. You can resize the two columns by dragging the edges of the column headers with the mouse. Click on the "Name" header to sort the parameters by name, click on the "Tag" header to sort them by tag.

▪ Find:
Enter the search text here. The search text is used for finding tags, and for modifying and duplicating parameters. When modifying or duplicating parameters, the search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle. In addition, if you are duplicating parameters, the search text in the tag of the original parameter is replaced with the replace text to create the tag of the new parameter. This means that the tags of all parameters you wish to duplicate must contain the search text.

▪ Replace:
Enter the text with which you want to replace the search text here. The replace text is used for modifying and duplicating parameters. When modifying or duplicating parameters, the search text is replaced with the replace text for everything specified in the "Apply changes to" rectangle. In addition, if you are duplicating parameters, the search text in the tag of the original parameter is replaced with the replace text to create the tag of the new parameter.

▪ Match case:
Check this box to find only text that matches the capitalization of the search text. If you leave this box blank, upper and lower case letters are treated the same when searching for the search text.

▪ Whole word:
Check this box to look only for complete words when searching for the search text. If you leave this box blank, the Development System will search for partial words as well as entire words.

▪ The "Apply changes to" rectangle:
When modifying or duplicating parameters, the search text is replaced with the replace text for everything that you specify in this box. You can select any number of the following options:

Names:
Change the name of the parameter. If you are duplicating parameters, and you do not check this box, the new parameters will have the same name as the originals, and you will have a hard time telling them apart. This box should usually be checked when duplicating parameters.

Default values:
Change the default value of the parameter.

- The “Find All Tags” button:

Press this button to select all parameters whose tags contain the search text.

- The “Replace All” button:

Press this button to apply the changes you specified to all selected parameters.

- The “Duplicate” button:

Press this button to create modified duplicates of all selected parameters. The tags of the new parameters will be created by replacing the search string in the original parameter’s tag with the replace string. For that reason, the tag of all parameters you are duplicating must contain the search text.

All changes you specified in the dialog will be applied to the new parameters. If you did not check the “Names” box in the “Apply changes to” rectangle, the new parameters will have the same name as the old ones, and it will be difficult to tell them apart. You should usually check the “Names” box.

- The “Delete All” button:

Press this button to delete all selected parameters.

- The “Close” button:

Press this button to close the Modify/Duplicate Parameters dialog.

CONTACT INFORMATION

U.P.M.A.C.S. Communications Inc.

714 36th Avenue, Suite 301
Lachine, Québec
Canada
H8T 3L8

Tel: 1-514-697-5500
Toll free: 1-877-697-5500 (Canada & US only)
E-Mail: support@upmacs.com

UPMACS is on the Web at http://www.upmacs.com