SSD DRIVES

DSI8000

Software Manual
Issue 4

HA471055U001

Although every effort has been taken to ensure the accuracy of this document it may be necessary, without notice, to make amendments or correct omissions. SSD Drives cannot accept responsibility for damage, injury, or expenses resulting there from.

Companies, names and data used as examples herein are fictitious unless otherwise stated. All trademarks and copyrights are acknowledged as the property of their respective owners.

Table of Contents

Getting Started

Welcome to DSI8000 - the latest operator interface configuration package from SSD Drives. DSI is designed to provide quick and easy access to the features of the TS8000 series of operator panels, while still allowing the advanced user to take advantage of high-end features, such as DSI's unique programming support.

System Requirements

DSI8000 is designed to run on PCs with the following specifications:

- A Pentium class processor as required by the chosen operating system.
- RAM and free disk space as required by the chosen operating system.
- An additional 50MB of disk space for software installation.
- A display of at least 800 by 600 pixels, with 256 or more colors is required, although 1024 by 768 pixels is highly recommended.
- An RS-232 or USB port for downloading to a TS8000 panel.

DSI8000 is designed to operate with all versions of Microsoft Windows from Windows 98 upwards. If you want to take advantage of the USB port provided by the TS8000 operator panels, you will need to use, as a minimum, Windows 98se. If you intend to use the USB port to remotely access the TS8000's CompactFlash card, we recommend that you use Windows 2000 or Windows XP. While Windows 98 is capable of accessing the card, the later versions of the operating system provide more robust operation, and are much better about when they choose to lock the card, thereby preventing the DSI runtime engine from writing data.

Installing The Software

If you downloaded the DSI8000 software from the SSD Drives website, simply execute the download file, and follow the instructions. If you received a copy of DSI on CD, place the CD in your system's CDROM drive, and follow the instructions that will appear. If no instructions appear, you may have auto-run disabled. In that case, select the Run option from the Start menu, and enter **x:\setup**, where **x** is the drive letter of your CDROM drive. Again, follow the resulting instructions, and the software will be installed.

Checking For Updates

If you have an internet connection, the Check for Update selection in the Help menu will go to the SSD Drives web site to check for updates to DSI8000. If a later version than the one you are using is found, DSI will ask if it should download the upgrade and update your software automatically. When the upgrade package executes, be sure to select the *Repair* option to update your installation properly.

Installing The USB Drivers

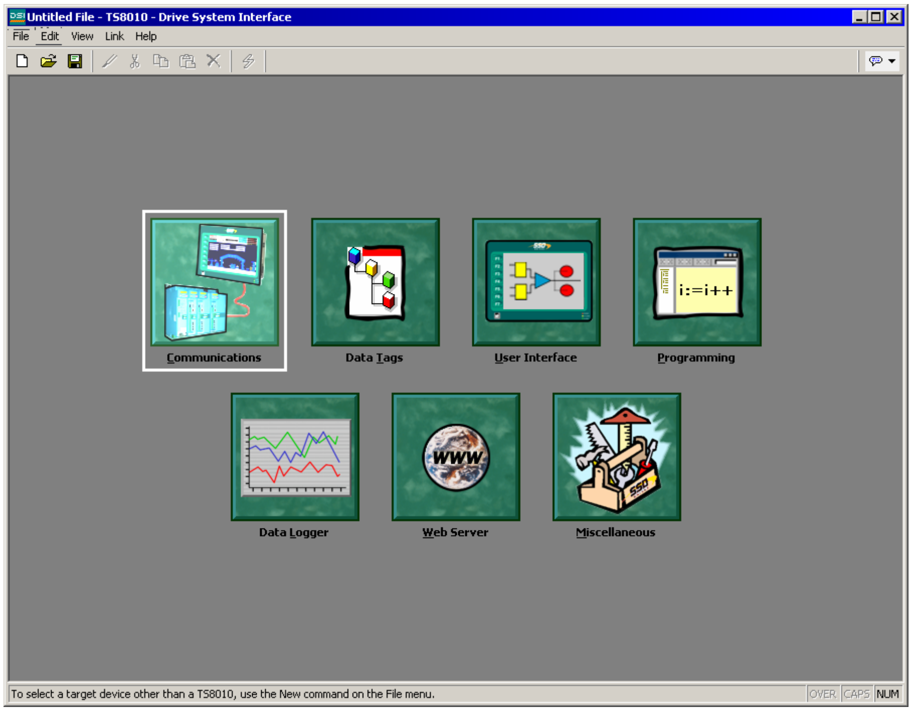
When you first connect a TS8000 panel to your PC using a USB cable, Windows will prompt you for the location of the drivers for the device. The default location for these drivers is C:\Program Files\SSD Drives\DSI8000\Device. When the Hardware Setup Wizard appears, choose the Browse option, and either point the Wizard at that location or whatever other location you specified during installation of the software. It is important that you perform this step correctly, or you may have to manually remove the drivers using the Device Manager, and repeat the installation once more. Windows XP users should note that DSI's USB drivers have not been digitally signed by Microsoft, and you will therefore see a dialog offering you the chance to stop the installation. You should be sure to select the *Continue* option to indicate that you do indeed wish to install this driver.

Please note that the driver installation will need to be performed twice: once for the HMI Loader, and a second time for the HMI itself.

Also note that for each new TS8000 unit that you connect to your development PC, you will need to reinstall the USB drivers. This is completely normal behavior. Each TS8000 USB driver chipset contains a unique Unit Serial ID that Windows recognizes as a new USB device. This feature allows you to have multiple TS8000's connected at the same time while preserving their individual identity.

DSI8000 Basics

To run DSI, select the DSI8000 icon from the SSD Drives folder on the Programs section of your Start Menu. The main DSI8000 screen will appear, showing the icons that are used to configure the various aspects of the operator panel's behavior.



To select a target device other than a TS8010, use the New command on the File menu.

The software is designed such that the first three icons are the only ones required for the majority of simple applications. The remainder of the icons provide access to the terminal's more advanced features, such as programming, data logging and the unit's web server.

Main Screen Icons

The sections below provide an overview of each icon in turn.

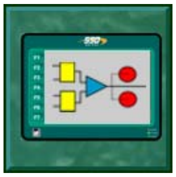
Communications

This icon is used to specify which protocols are to be used on the TS8000's serial ports and on the Ethernet port. Where master protocols are used (protocols by which the TS8000 initiates data transfer to and from a remote device) you can also use this icon to specify one or more devices to be accessed. Where slave protocols are used (protocols by which the TS8000 receives and responds to requests from remote devices or computer systems) you can specify which data items are to be exposed for read or write access. You can also use this icon to move data between one remote device and another via DSI's protocol converter.

Data Tags

This icon is used to define the data items to be accessed within the remote devices, or to define internal data items to store information within the terminal itself. Each tag has a variety of properties associated with it. The most basic property is formatting data, which is used to specify how the data held within a tag is to be shown on the terminal's display, and on such things as web pages. By specifying this information within the tag, DSI removes the need for you to re-enter formatting data each time a tag is displayed. More advanced tag properties include alarms that may activate when various conditions relating to the tag occur, or triggers, which perform programmable actions on similar conditions.

User Interface

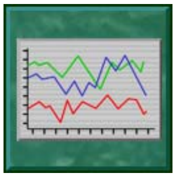


This icon is used to create and edit display pages, and to specify what actions should be taken when the operator panel's keys are pressed, released or held down. The page editor allows you to display various graphical items known as objects. These vary from simple items, such as rectangles and lines, to more complex items that can be tied to the value of a particular tag or expression. By default, such objects use the formatting information defined when the tag was created, but this information can be overridden if required.

Programming

This icon is used to create and edit programs using DSI8000's unique C-like programming language. These programs can perform complex decision making or data manipulation operations based upon any data items within the system. They serve to extend the functionality of DSI beyond that of the standard functions included in the software, thereby ensuring that even the most complex applications can be tackled with ease.

Data Logger



This icon is used to create and manage data logs, each of which can record any number of variables to the TS8000's CompactFlash card. Data may be recorded as quickly as once per second. The recorded values will be stored in CSV (comma separated variable) files that can easily be imported into applications such as Microsoft Excel. The files can be accessed by swapping-out the CompactFlash card, by mounting the card as a drive on a PC connected on the TS8000's USB port, or by accessing them via DSI's web server via the Ethernet port.

Web Server

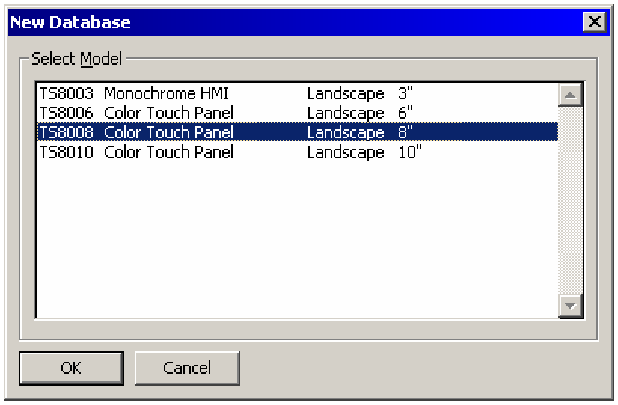
This icon is used to configure DSI's web server and to create and edit web pages. The web server is capable of providing remote access to the TS8000 via a number of mechanisms. First, you can use DSI to create automatic web pages which contain lists of tags, each formatted according to the tag's properties. Second, you can create a custom site using a third party HTML editor such as Microsoft FrontPage, and then include special text to instruct DSI to insert live tag values. Finally, you can enable DSI's unique remote access and control feature, which allows a web browser to view the TS8000's display and control its keyboard. The web server can also be used to access CSV files from the Data Logger.

Security Manager

This icon is used to create and manage the various users of the panel, as well as the access rights granted to them. Real names may also be given, which allows the security logger to record not only what data was changed and when, but also by whom the data was changed. The rights required to modify a particular tag, or to access a page, are set via the security properties of the individual item.

Selecting A Terminal

When DSI8000 first starts, it will assume that you are continuing to work with the same size TS8000 as was used previously. If DSI8000 has not been loaded before, it will assume that you are working with a TS8003. If you want to select a new model, select the New command from the File menu. The dialog shown will appear.



New Database

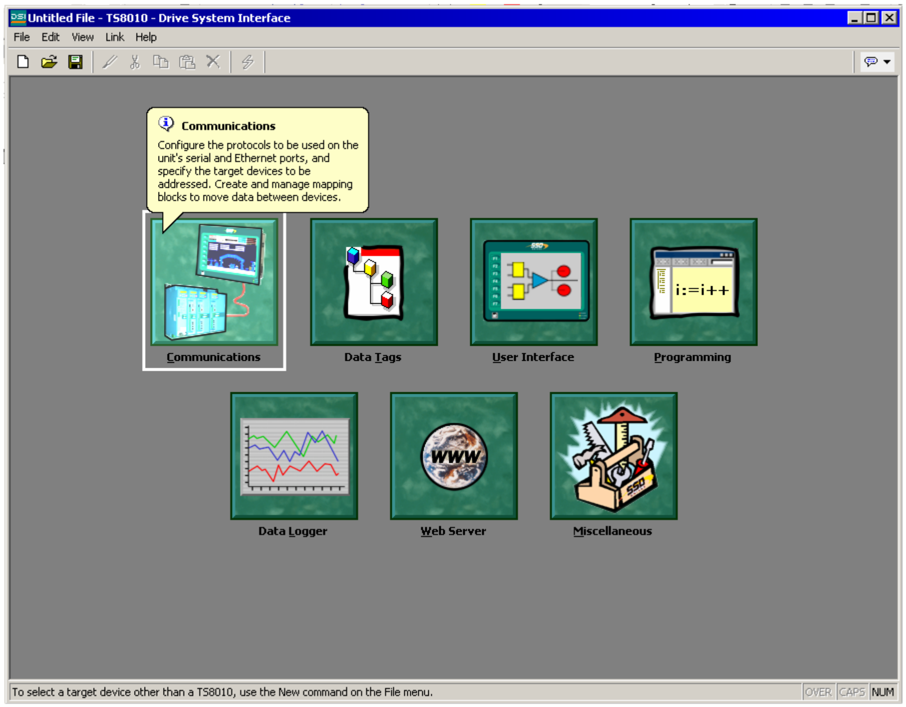
Select Model

TS8003	Monochrome HMI	Landscape	3"
TS8006	Color Touch Panel	Landscape	6"
TS8008	Color Touch Panel	Landscape	8"
TS8010	Color Touch Panel	Landscape	10"

The dialog lists the models supported by the current version of the software, providing a description of each terminal and the dimensions of its display. Selecting a terminal will create a blank database and reconfigure DSI to work with that specific model.

Using Balloon Help

DSI8000 provides a useful feature called Balloon Help.



Communications

Configure the protocols to be used on the unit's serial and Ethernet ports, and specify the target devices to be addressed. Create and manage mapping blocks to move data between devices.

To select a target device other than a TS8010, use the New command on the File menu.

This feature allows you to see help and context information for each icon in the main menu, or for each field in a dialog box or window. It is controlled via the icon at the right-hand edge of the toolbar, and can be configured to three modes, namely Do Not Display, in which case balloon help is disabled; When Mouse Over, in which case help is displayed when the mouse pointer is held over a particular field for a certain period of time; or When Selected, in which case help is always displayed for the currently selected field.

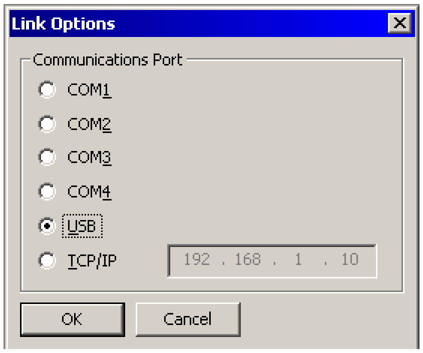
Working With Databases

DSI8000 stores all the information about a particular panel's configuration in what is called a database file. These files have the extension of **DSI**, although Windows Explorer will hide this extension if it is left in its default configuration. DSI database files differ from those used by previous SSD Drives products, in that they are text files which are thus far easier to recover in the case of accidental corruption. Databases are manipulated via the commands found on the File menu. These commands are standard for all Windows applications, and need no further explanation. The exception is Save Image, which will be covered later.

Downloading To A Terminal

DSI database files are downloaded to the TS8000 panel by means of the Link menu. The download process typically takes only a few seconds, but can take somewhat longer on the first download if DSI has to update the firmware in the operator panel, or if the panel does not contain an older version of the current database. After this first download, however, DSI uses a process known as incremental download to ensure that only changes to the database are transferred. This means that changes can be made in seconds, thereby reducing your development cycle time and simplifying the debugging process.

Configuring The Link



○ COM1
○ COM2
○ COM3
○ COM4
● USB
○ TCP/IP 192 . 168 . 1 . 10

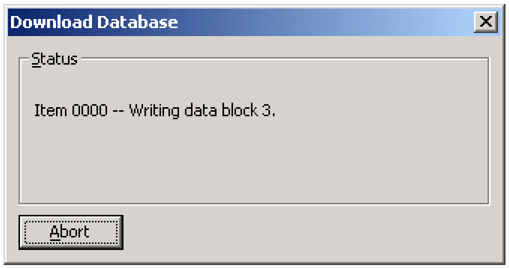
The programming link between the PC and the TS8000 is made using either an RS-232 serial port, a USB port, or a TCP/IP connection. Before downloading, you should use the Link-Options command to ensure that you have the desired method selected.

If you are using USB, you might also want to ensure that the TS8000's USB drivers have been correctly installed. To do this, connect the TS8000 panel, and, if the drivers have not previously been installed, follow the instructions at the start of this manual. Then, open the Device Manager for your operating system, and expand the USB icon to show the icon for the TS8000 Panel device. Ensure that this icon does not display a warning symbol. If it does, remove the device, unplug and reconnect the TS8000 panel, and verify that you have correctly followed the driver installation procedure.

Setting The IP Address

If you are using a TCP/IP connection, you should enter the IP address of the target device in the appropriate field in the dialog box. If you leave the panel's IP address as 0.0.0.0, DSI8000 will examine the currently loaded database to see if the panel's address can be determined from the configuration information. This feature removes the need to change the IP address when switching between databases intended for different terminals.

Sending The Database



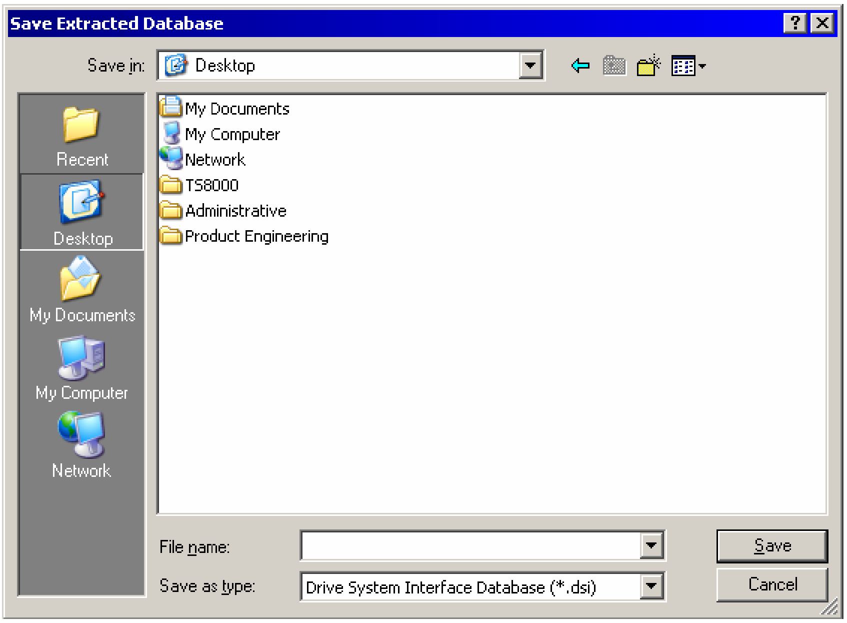
Item 0000 -- Writing data block 3.

Once the link is configured, the database can be downloaded using either the Link-Send or Link-Update commands. The former will send the entire database, whether or not individual objects within the file have changed. The latter will only send changes, and will typically take a much shorter period of time to complete. The Update command is typically the only one that you will need, as DSI will automatically fall-back to a complete send if the incremental download fails for any reason. As a shortcut, note that you can access Link-Update via the lightning-bolt symbol on the toolbar, or via the **F9** key on the PC.

Note that downloading via TCP/IP requires a CompactFlash card to be installed in the TS8000 if the device's firmware is to be upgraded. Since you may want to perform such upgrades at some point in time, it is highly recommended that you install a CompactFlash card in any device to which TCP/IP downloads are likely to be performed.

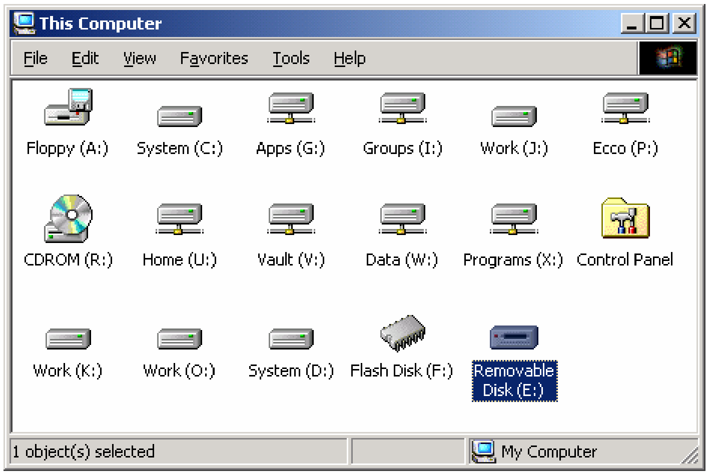
Extracting Databases

The Link-Support Upload command can be used to instruct DSI8000 whether or not it should include the information necessary to support database upload when sending a database to a TS8000 panel. Supporting upload will slow the download process somewhat, but will ensure that should you lose your database file, you will be able to extract an editable image from the terminal. If you lose your database file and you do not have upload support enabled, you will not be able to reconstruct your file without starting from scratch. To extract a database from a panel, use the Link-Extract command. This command will upload the database, and then prompt you for a name under which to save the file. The file will then be opened for editing.



Mounting The CompactFlash

If you are connected to a TS8000 panel via the USB port, you can instruct DSI to mount the unit's CompactFlash card as a drive within Windows Explorer. You can use this functionality to save files to the card or to read information from the Data Logger. The drive is mounted and dismounted by sending commands using the Mount Flash and Dismount Flash options on the Link menu. Once a command has been sent, the TS8000 panel will be reset, and Windows will refresh the appropriate Explorer windows to show or hide the CompactFlash drive.



Note that some caution is required when mounting the CompactFlash card.

- When the card is mounted, the TS8000 will periodically inform the PC if data on the card has been modified. This means that both the PC and the TS8000 will suffer performance hits if the card is mounted during data logging operations for longer than necessary.

- If you write to the CompactFlash card from your PC, the TS8000 will not be able to access the card until Windows releases its "lock" on the card's contents. This may take up to a minute, and will restrict data logging operations during that time, and prevent access to custom web pages. DSI will use the TS8000's RAM to ensure that no data is lost, but if too many writes are performed such that the card is kept locked for four minutes or more, data may be discarded. Note that Windows 98 is particularly bad at keeping the card locked when there is no need for it. Windows 2000 or Windows XP is thus the operating system of choice when using this feature.

- You should never attempt to format a CompactFlash card that you have mounted via the TS8000, whether it is via Windows Explorer or from the command prompt. Windows does not correctly lock the card during format operations, and the format may thus be unreliable and lead to subsequent data loss. See below for more information on how to properly format a CompactFlash card.

Formatting The CompactFlash

- The preferred method for formatting a card is via the Format Flash command on the Link menu. Selecting this command will explain that the formatting process will destroy all of the data stored on the CompactFlash card and offer you a chance to cancel the operation. If you elect to continue, the operator panel will be instructed to format the card. Note that this process may take several minutes for a large card. Slow formats on panels that are performing data logging may therefore result in gaps in the recorded data.

- A less attractive method of formatting a card is via a dedicated CompactFlash card reader connected to your PC. If you use this method, be sure to instruct Windows to format the card using the FAT16 file system. For very small or very large cards, Windows will most likely choose the wrong format by default. Worse still, some versions of Windows Explorer will not allow you to override the default format, forcing you to use the command line version FORMAT instead.

Sending The Time

The Link-Send Time command can be used to set the TS8000's clock to match that of the PC on which DSI is executing. Obviously, make sure your clock is right before you do this!

Using The Emulator

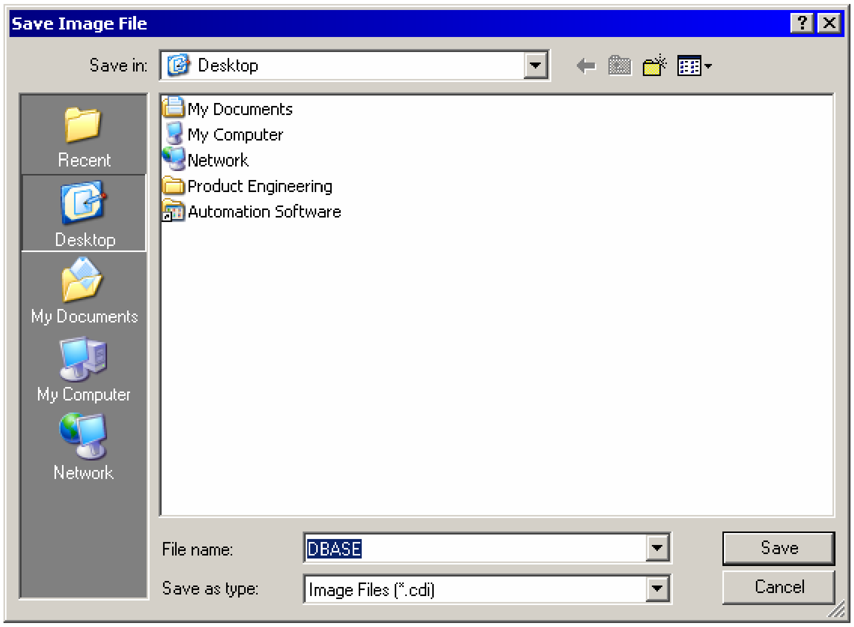
DSI8000 features an Emulator capable of reproducing some devices locally on a PC. This feature will only work on a PC with Windows XP or 2000. The only supported devices are the TS8006, TS8008, and TS8010.

The Emulator can be used to test not only the user interface portion of a database, but also the operation of the Data Logger and the Web Server. Note that the Data Logger data is saved in the computer RAM memory and therefore is not available on the hard drive. This means the memory will be dumped each time the Emulator is stopped. The RAM will not behave like the TS8000's Compact Flash card and file functions will not work properly.

Downloading to the Emulator will open a new window representing the TS8000 as shown. To download to the Emulator, the Link has to be configured by selecting Send To The Emulator on the Link Menu. The database can be downloaded in the Emulator by using either the Link-Ipdate or Link-Send functions.

Updating Via CompactFlash

If you need to update the database within a unit which is already installed at a customer's site, DSI allows you to save a copy of the database to a CompactFlash card, ship that card to your customer, and have the TS8000 load the database from that card. The process is performed via the Save Image command on the File menu.



Save Image File

Save in: Desktop

File name: DBASE

Save as type: Image Files (*.cdi)

The Save Image command will create a DSI8000 database image file with a CDI extension. It will also save three other files, with the extensions of BIN, LDR, and ROM. The image file must be given the name DBASE.CDI, and it, along with the BIN, LDR, and ROM files must be placed in the root directory of a CompactFlash card. To update a TS8000 panel, power down the unit, insert the CompactFlash card bearing the two files, and reapply power to the unit. The unit's boot loader will first check whether it needs to upgrade the unit's firmware, and once this process has been completed, the DSI runtime application will load the database stored on the card. The CompactFlash card can then be removed or left in place as required.

Guru Meditation Codes

If a problem with the DSI runtime application within the TS8000 operator panel results in the panel being reset, the condition that caused the fault will be logged. When the panel restarts, this information will be displayed in the form of a Guru Meditation Code. A typical code will have the format:

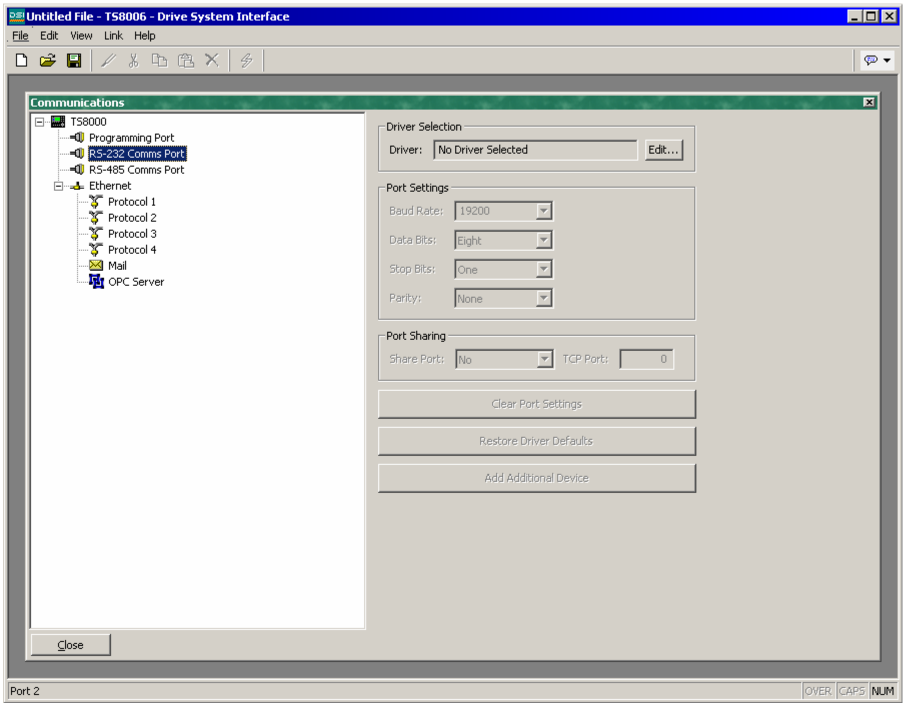
03-2004-1BE4-205

The message can be accepted by pressing the F1 key, at which point the terminal will resume normal operation. Note that communications, data logging and the web server are still active when the GMC is displayed - only the user interface is interrupted. This means that system disruption is minimized, and functions such as protocol conversion continue to operate.

Before accepting the message, you may wish to write down the code. You may then email it to SSD Drives technical support, so that one of our support engineers can track-down the cause of the problem. You may also want to send a copy of the terminal's database, and describe what you were doing when the terminal crashed.

Configuring Communications

The first stage of creating a DSI database is to configure the communications ports of the TS8000 panel to indicate which protocols you want to use, and which remote devices you want to access. These operations are performed from the Communications window, which is opened by selecting the first icon of the DSI8000 main screen.



TS8000
Programming Port
RS-232 Comms Port
RS-485 Comms Port
Ethernet
Protocol 1
Protocol 2
Protocol 3
Protocol 4
Mail
OPC Server

Driver Selection
Driver: No Driver Selected

Port Settings
Baud Rate: 19200
Data Bits: Eight
Stop Bits: One
Parity: None

Port Sharing
Share Port: No
TCP Port: 0

Port 2

As can be seen, the Communications window lists the unit's available ports in the form of a tree structure. TS8000 panels have three primary serial ports. They also provide a single Ethernet port which is capable of running four communications protocols simultaneously.

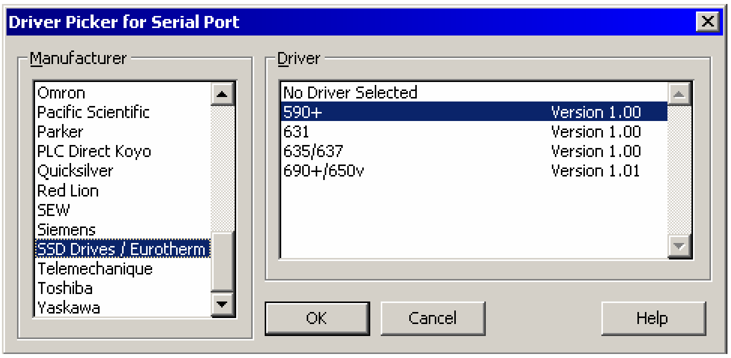
Serial Port Usage

When deciding which of the TS8000's serial ports to use for communications, note that:

- The TS8003 multiplexes a single serial communications controller between its RS-232 and RS-485 communication ports. This means that if either port is used for a slave protocol, the other port is unavailable. It also means that if a token-passing protocol such as Allen-Bradley DH-485 is employed, the other port is similarly disabled. Other TS8000 panels impose no such restrictions.

- The unit's programming port may be used as an additional communications port, but it will obviously not be available for download if it is so employed. This is not an issue if the USB port is used for such purposes, and it is highly recommended that you use this method of download if you want to connect serial devices via the programming port.

Selecting A Protocol



Manufacturer:
- Omron
- Pacific Scientific
- Parker
- PLC Direct Koyo
- Quicksilver
- Red Lion
- SEW
- Siemens
- SSD Drives / Eurotherm
- Telemechanique
- Toshiba
- Yaskawa

Driver:

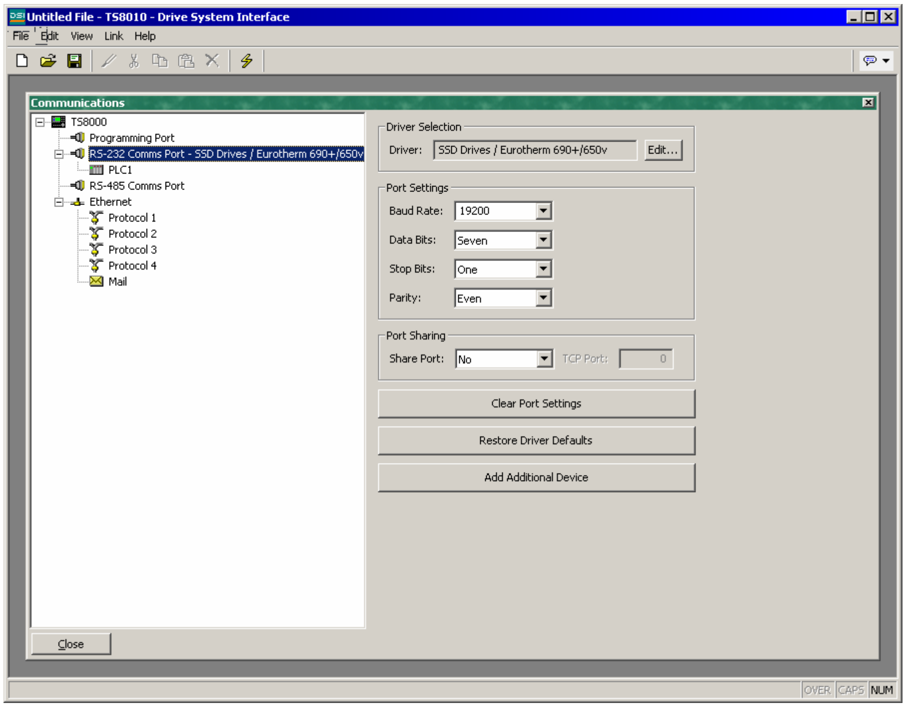
No Driver Selected	
590+	Version 1.00
631	Version 1.00
635/637	Version 1.00
690+/650v	Version 1.01

To select a protocol for a particular port, click on that port's icon in the left-hand pane of the Communications window, and press the Edit button next to the Driver field in the right-hand pane. The shown dialog box will appear.

Select the appropriate manufacturer and driver, and press the OK button to close the dialog box. The port will then be configured to use the appropriate protocol, and a single device icon will be created in the left-hand pane. If you are configuring a serial port, the various Port Settings fields (Baud Rate, Data Bits, Stop Bits and Parity) will be set to values appropriate to the protocol in question. You should obviously check these settings to make sure that they correspond to the settings for the device to be addressed.

Protocol Options

Some protocols require additional configuration of parameters specific to that protocol. These appear in the right-hand pane of the Communications window when the corresponding port icon is selected. The example below shows the additional parameters for the Allen-Bradley DH-485 driver, which appear under the Driver Settings section of the window.



- TS8000
  - Programming Port
  - RS-232 Comms Port - SSD Drives / Eurotherm 690+/650v
    - PLC1
  - RS-485 Comms Port
  - Ethernet
    - Protocol 1
    - Protocol 2
    - Protocol 3
    - Protocol 4
    - Mail

Driver: SSD Drives / Eurotherm 690+/650v

Baud Rate: 19200

Data Bits: Seven

Stop Bits: One

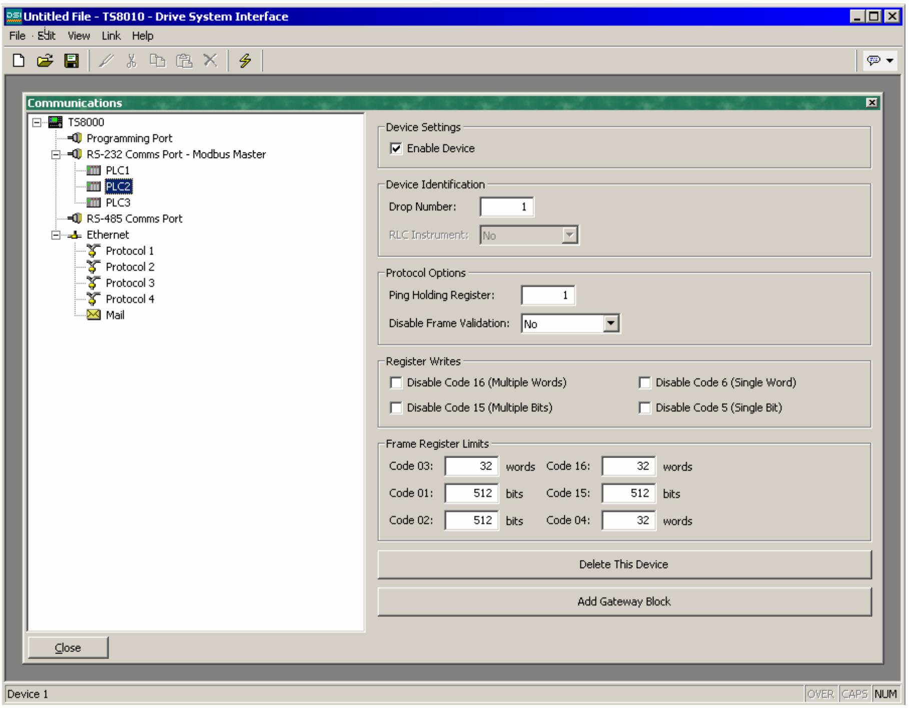
Parity: Even

Share Port: No

TCP Port: 0

Working With Devices

As mentioned above, when a communications protocol is selected, a single device is created under the corresponding port icon. In the case of a master protocol, this represents the initial remote device to be addressed via the protocol. If the protocol supports access to more than one device, you can use the Add Additional Device button included with the port icon's properties to add further target devices. Each device is represented via an icon in the left-hand pane of the Communications window, and, depending on the protocol in question, may have a number of properties to be configured.



Device Settings

☑ Enable Device

Device Identification

Drop Number: 1

RLC Instrument: No

Protocol Options

Ping Holding Register: 1

Disable Frame Validation: No

Register Writes

☐ Disable Code 16 (Multiple Words) ☐ Disable Code 6 (Single Word)

☐ Disable Code 15 (Multiple Bits) ☐ Disable Code 5 (Single Bit)

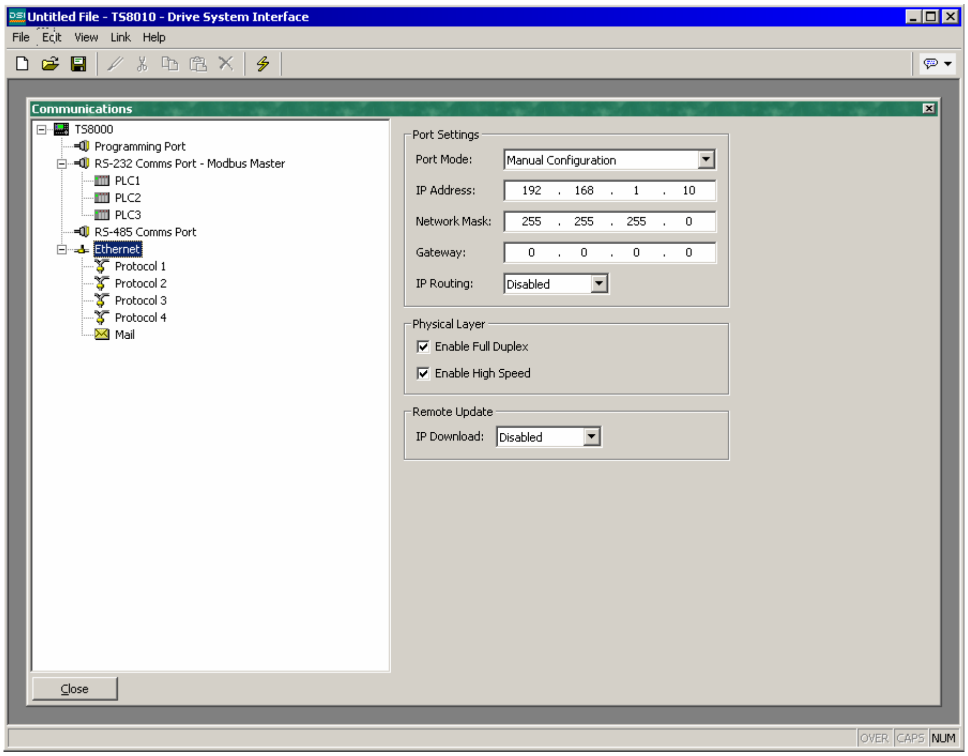
Frame Register Limits

Code 03:	32	words	Code 16:	32	words
Code 01:	512	bits	Code 15:	512	bits
Code 02:	512	bits	Code 04:	32	words

In the example above, the Modbus Universal Master protocol has been selected, and two additional devices have been created, indicating that a total of three remote devices are to be accessed. The right-hand pane of the window shows the properties of a single device. The Enable Device property is present for devices for all protocols, while the balance of the fields are specific to the protocol that has been selected. Note that the devices are given default names by DSI when they are created. These names may be changed by selecting the appropriate icon in the left-hand pane, and simply typing the new device name.

Ethernet Configuration

The TS8000's Ethernet port is configured via the Ethernet icon in the left-hand pane of the Communications window. When this icon is selected, the following settings are displayed.



Port Settings

Port Mode: Manual Configuration

IP Address: 192 . 168 . 1 . 10

Network Mask: 255 . 255 . 255 . 0

Gateway: 0 . 0 . 0 . 0

IP Routing: Disabled

Physical Layer

☑ Enable Full Duplex

☑ Enable High Speed

Remote Update

IP Download: Disabled

IP Parameters

The Port Mode field controls whether or not the port is enabled, and the method by which the port is to obtain its IP configuration. If DHCP mode is selected, the TS8000 will attempt to obtain an IP address and associated parameters from a DHCP server on the local network. If the unit is configured to use slave protocols or to serve web pages, this option will only make sense if the DHCP server is configured to allocate a well-known IP address to the MAC address associated with the unit, as otherwise, users will not be sure how to address the panel.

If the more common Manual Configuration mode is selected, the IP Address, Network Mask and Gateway fields must be filled out with the appropriate information. The default values provided for these fields will almost never be suitable for your application. Be sure to consult your network administrator when selecting appropriate values, and be sure to enter and download these values before connecting the TS8000 to your network. If you do not follow this advice, it is possible that you will cause problems on your network.

IP Routing

The IP Routing option can be used to enable or disable the routing of IP packets between the Ethernet port and any PPP connections made to or by the panel. You should not enable this feature unless you understand the implications of allowing such routing. Please refer to the Advanced Communications chapter of this document for more information.

Physical Layer

The Physical Layer options control the type of connection that the TS8000 will attempt to negotiate with the hub to which it is connected. Generally, these options can be left in their default states, but if you have trouble establishing a reliable connection, especially when connecting directly to a PC without an intervening hub or switch, consider turning off both Full Duplex and High Speed operation to see if this solves the problem.

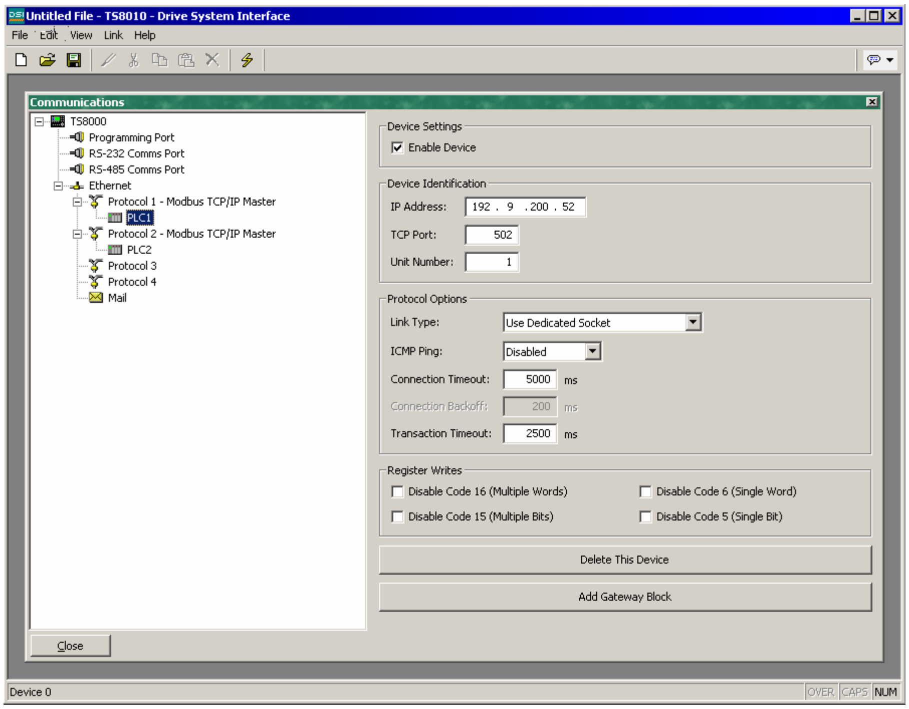
Remote Update

The Remote Update option is used to enable or disable firmware and configuration downloads to the panel via TCP/IP. As noted previously, remote firmware updates over TCP/IP require that the unit has been fitted with a CompactFlash card. Since downloads will more thank likely involve a firmware update at some point, such a card is highly recommended when using this feature.

Protocol Selection

Once the Ethernet port has been configured, you can select the protocols that you wish to use for communications. Up to four protocols may be used at once, and many of these protocols will support multiple remote devices. This means that you have several options when deciding how to mix protocols and devices to achieve the results you want.

For example, suppose you want to connect to two remote slave devices using Modbus over TCP/IP. Your first option is to use two of the Ethernet port's protocols, and configure both as Modbus TCP/IP Masters, with a single device attached to each protocol.



- TS8000
  - Programming Port
  - RS-232 Comms Port
  - RS-485 Comms Port
  - Ethernet
    - Protocol 1 - Modbus TCP/IP Master
      - PLC1
    - Protocol 2 - Modbus TCP/IP Master
      - PLC2
    - Protocol 3
    - Protocol 4
    - Mail

Device Settings

☑ Enable Device

Device Identification

IP Address: 192 . 9 . 200 . 52

TCP Port: 502

Unit Number: 1

Protocol Options

Link Type: Use Dedicated Socket

ICMP Ping: Disabled

Connection Timeout: 5000 ms

Connection Backoff: 200 ms

Transaction Timeout: 2500 ms

Register Writes

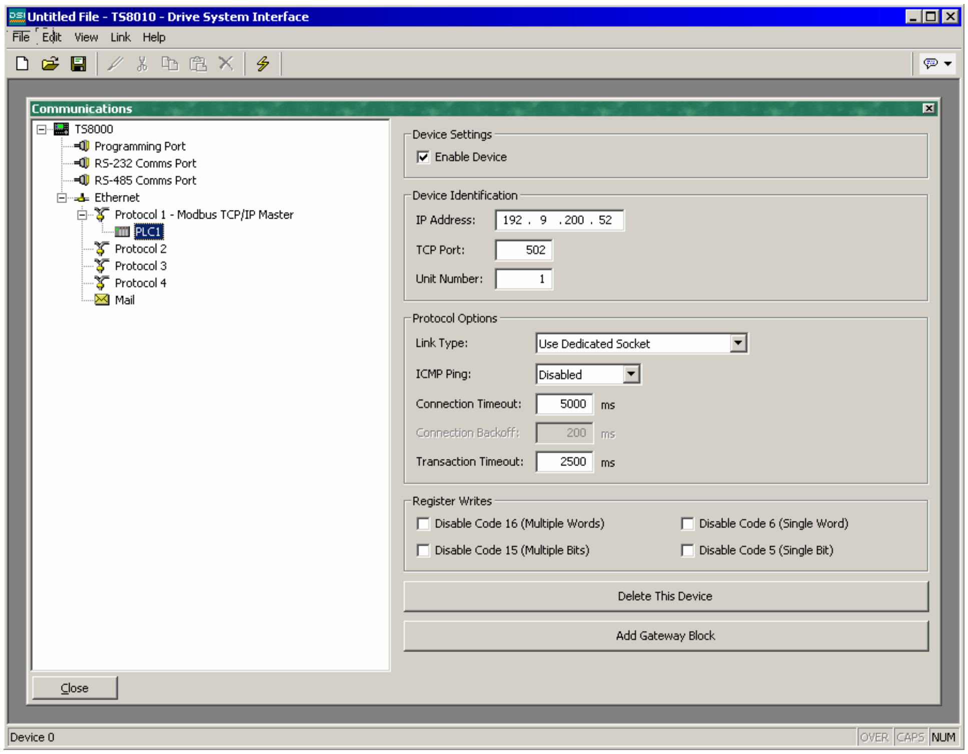
☐ Disable Code 16 (Multiple Words) ☐ Disable Code 6 (Single Word)

☐ Disable Code 15 (Multiple Bits) ☐ Disable Code 5 (Single Bit)

Device 0

For most protocols, this will produce higher performance, as it will allow simultaneous communications with the two devices. It will, however, consume two of the four available protocols, limiting your ability to connect via additional protocols in complex applications.

Your second option is to use a single protocol configured as a Modbus TCP/IP Master, but to add a further device so that both slaves are accessed via the same driver.



Device 0

TS8000
- Programming Port
- RS-232 Comms Port
- RS-485 Comms Port
- Ethernet
  - Protocol 1 - Modbus TCP/IP Master
    - PLC1
  - Protocol 2
  - Protocol 3
  - Protocol 4
  - Mail

Device Settings

☑ Enable Device

Device Identification

IP Address: 192 . 9 .200 . 52
TCP Port: 502
Unit Number: 1

Protocol Options

Link Type: Use Dedicated Socket
ICMP Ping: Disabled
Connection Timeout: 5000 ms
Connection Backoff: 200 ms
Transaction Timeout: 2500 ms

Register Writes

☐ Disable Code 16 (Multiple Words) ☐ Disable Code 6 (Single Word)
☐ Disable Code 15 (Multiple Bits) ☐ Disable Code 5 (Single Bit)

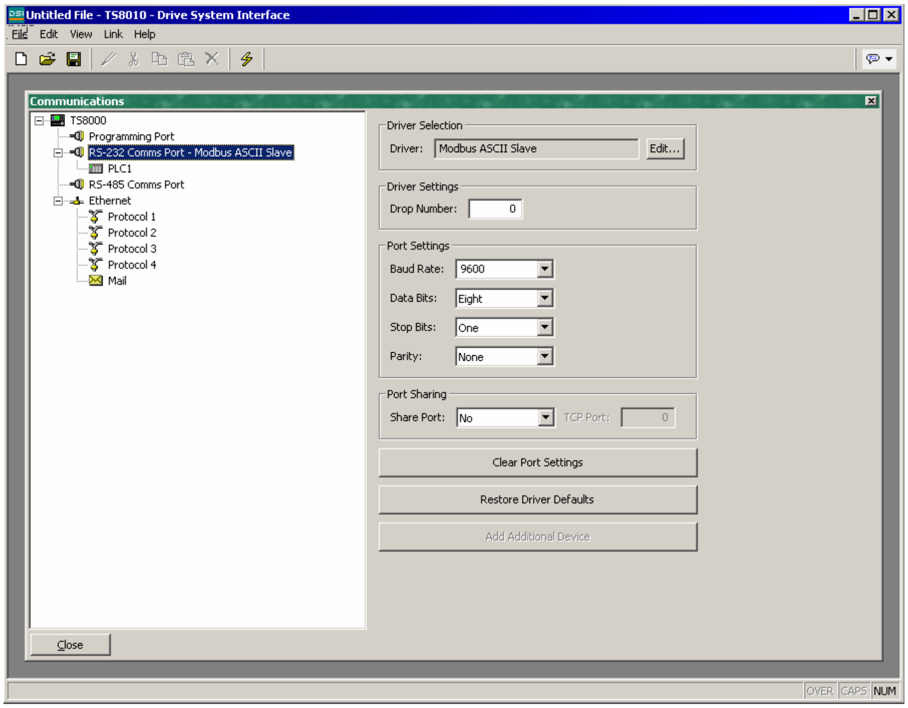
This will typically produce slightly reduced performance, as DSI will poll each device in turn, rather than talking to both devices at the same time. It will, however, conserve Ethernet protocols, allowing more complex applications without running out of resources.

Slave Protocols

For master protocols (those where the TS8000 initiates communication) there is no further configuration required under the Communications icon. For slave protocols (those where the TS8000 receives and responds to remote requests), however, the process is slightly more complex, as you must also indicate what data you wish to expose for remote access.

Selecting The Protocol

As with master protocols, the first stage is to select the protocol for the communications port that you wish to use. The example below shows the TS8000's RS-232 port configured for operation with the Modbus ASCII Slave protocol.



Driver Selection

Driver: Modbus ASCII Slave

Driver Settings

Drop Number: 0

Port Settings

Baud Rate: 9600

Data Bits: Eight

Stop Bits: One

Parity: None

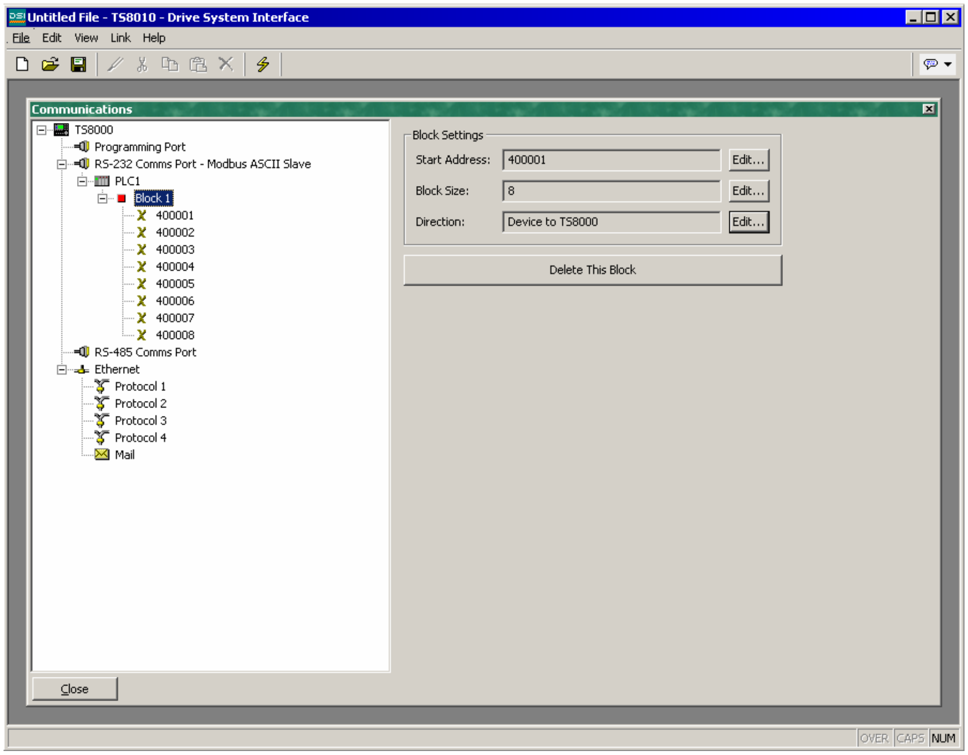
Port Sharing

Share Port: No TCP Port: 0

Note that a single device has been automatically created for the protocol. In the case of master protocols, this represents the remote device that the TS8000 will access. In this case, though, the device represents the Modbus slave that the TS8000 will itself embody. This means that only a single device is required, and that things such as the station number to which the TS8000 will respond are normally configured via the port settings rather than those of the device.

Adding Gateway Blocks

Having configured the protocol, you must now decide what range of addresses you want the slave protocol to expose. In this example, we want to use Modbus registers 40001 through 40008 to allow read and write access to certain data items in our database. We begin by selecting the device icon in the left-hand pane of the Communications window, and clicking the Add Gateway Block button in the right-hand pane. An icon to represent Block 1 will appear, and selecting it will show the following settings:



Block Settings

Start Address: 400001

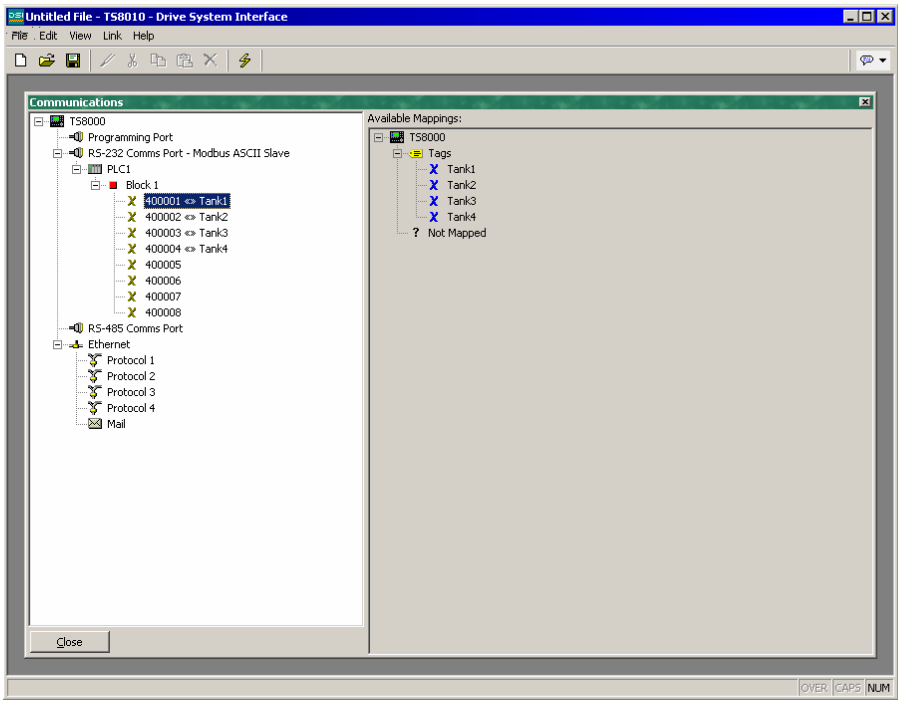
Block Size: 8

Direction: Device to TS8000

In the example above, we have configured the Start Address to 40001 to indicate that this is where we want the block to begin. We have also configured the Block Size to eight so as to allocate one Modbus register for each tag we want to expose. Finally, we have configured the Direction as Device to TS8000, to indicate that we want remote devices to be able to read and write data items exposed via this block.

Adding Items To A Block

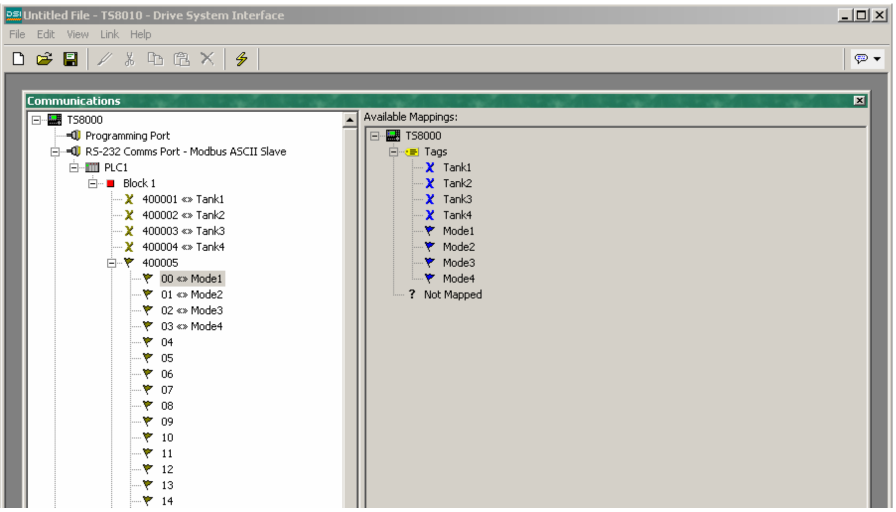
Once the block has been created and its size defined, entries appear in the left-hand pane of the window to represent each of the registers that the block exposes to remote access. When one of these entries is selected, the right-hand pane shows a list of available data items, comprising both tags from within your database, and data registers from any master communications devices that you have configured.



To indicate that you want a particular register within your gateway block to correspond to a certain data item, simply drag that item from the right-hand pane to the left-hand pane, dropping it on the appropriate gateway block entry. The example above shows how the registers 40001 through 40004 in the block above have been mapped to tags called Tank1 through Tank4.

Accessing Individual Bits

If your application requires it, you can expand individual elements within a Gateway Block to their constituent bits, and map a different data item to each bit. To do this, right-click on the individual element, and select Expand from the pop-up menu. The right-hand pane will be updated to show the individual bits that make up the register. These can be mapped using the drag-and-drop process described above.

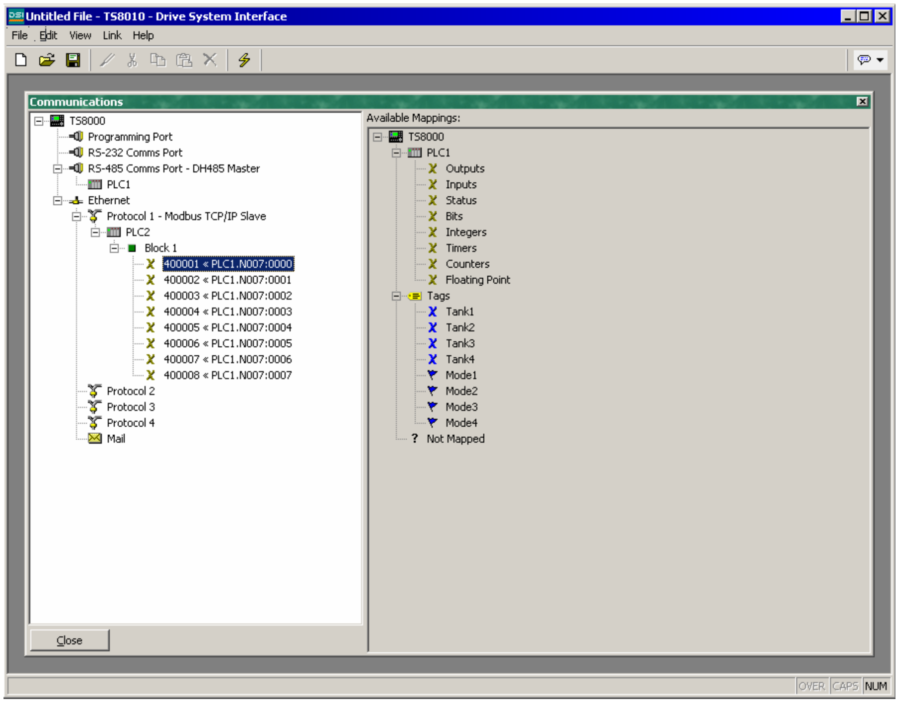


Protocol Conversion

In addition to exposing internal data tags via slave protocols, Gateway Blocks can also be used to expose data that is obtained from other remote devices, or to move data between two such master devices. This unique protocol conversion feature allows much tighter integration between elements of your control system, even when using simple, low-cost devices.

Master And Slave

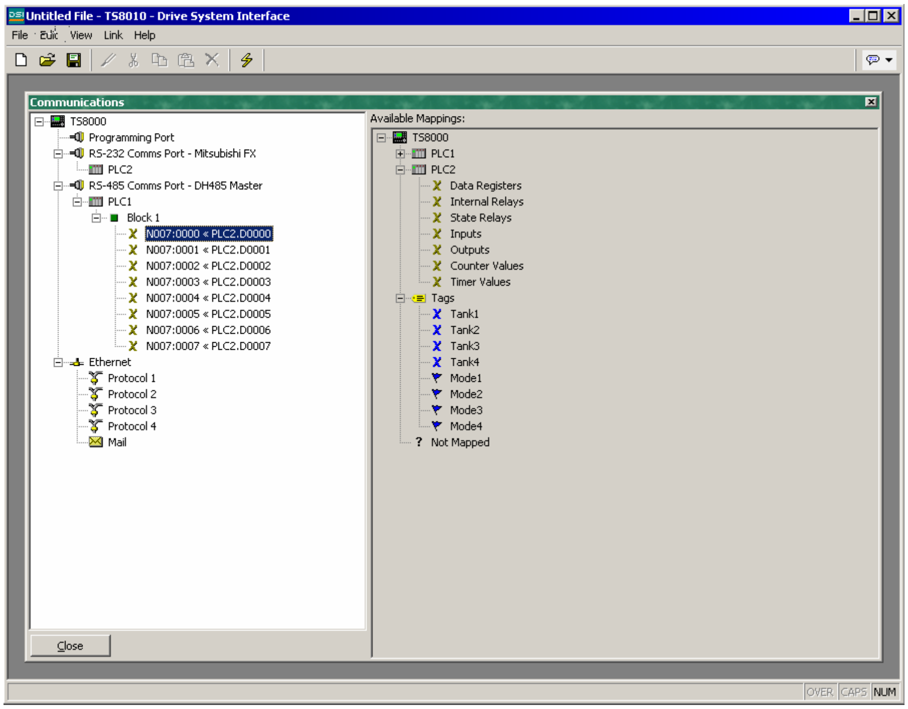
Exposing data from other devices over a slave protocol is simply an extension of the mapping process described above, except this time, instead of dragging a tag from the right-hand pane, you should expand the appropriate master device, and drag across the icon that represents the registers that you want to expose. You will then be asked for a start address in the master device, and the number of registers to map, and the mappings will be created as shown.



In this example, registers N7:0 through N7:7 in an Allen-Bradley controller have been exposed for access via Modbus TCP/IP as registers 40001 through 40008. DSI will automatically ensure that these data items are read from the Allen-Bradley PLC so as to fulfill Modbus requests, and will automatically convert writes to the Modbus registers into writes to the PLC. This mechanism allows even simple PLCs to be connected on an Ethernet network.

Master And Master

To move data between two master devices, simply select one of the devices, and create a Gateway Block for that device. You can then add references to the other device's registers just as you would when exposing data on a slave protocol. Again, DSI will automatically read or write the data as required, transparently moving data between the devices. The example below shows how to move data from a Mitsubishi FX into an Allen-Bradley PLC.



Gateway Block Ownership

One question that may occur to you is whether you should create the Gateway Block within the Allen-Bradley device, as in this example, or within the Mitsubishi device. The first thing to note is that there is no need to create more than a single block to perform transfers in a single direction. If you create a block in AB to read from MITFX, and a block in MITFX to write to AB, you'll simply perform the transfer twice and slow everything down. The second observation is that the decision as to which device should "own" the Gateway Block is essentially arbitrary. In general, you should create your blocks so as to minimize the number of blocks in the database. This means that if the registers in the Allen-Bradley lay within a single range, but the registers in the Mitsubishi are scattered all over the PLC, the Gateway Block should be created within the Allen-Bradley device so as to remove the need to create multiple blocks to access the different ranges of the Mitsubishi device.

Data Transformation

You may also use Gateway Blocks to perform math operations that your PLC might not otherwise be able to handle. For example, you may want to read a register from the PLC, scale it, take the square root, and write it back to another PLC register. To accomplish this, refer to the section on Data Tags, and create a mapped variable to represent the input value that will be read from the device. Then, create a formula to represent the output value, setting the expression so as to perform the required math. You can then create a Gateway Block targeted at the required output register, and drag the formula across to instruct DSI to write the derived value back to the PLC.

Advanced Communications

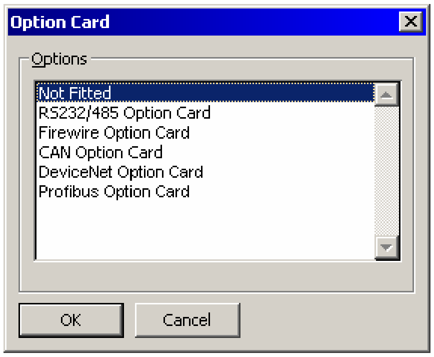
This chapter explains how to use some of the more advanced communications features that are supported by DSI8000. Simple applications may not require these features, and you may thus choose to skip this chapter and return to it later.

Using Option Cards

Each TS8000 panel is capable of hosting an individual option card to provide additional communications facilities. Option cards are currently available for:

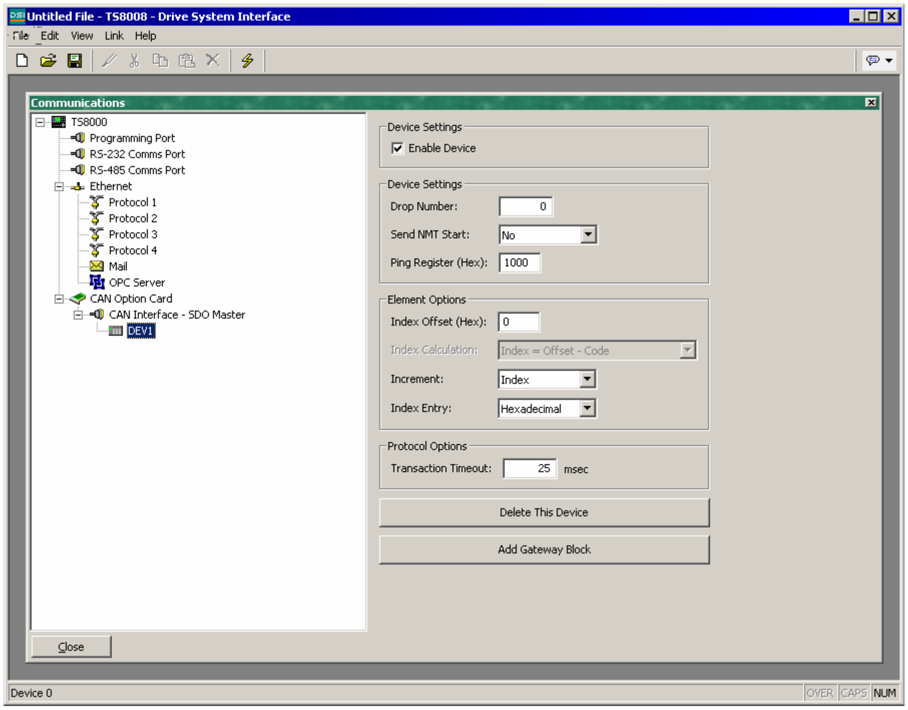
- Additional RS-232 and RS-485 Serial ports.
- IEEE 1394 Firewire networks.
- CANopen fieldbus networks.
- DeviceNet fieldbus networks.
- Profibus fieldbus networks.

Hardware installation instructions are provided with each card, so please refer to the supplied materials for information on fitting the card to the TS8000 panel. Once the card is installed, configuration is performed by selecting the TS8000 icon in the left hand pane of the of the Communications window, and clicking on the Edit button next to the option card property.



Not Fitted
RS232/485 Option Card
Firewire Option Card
CAN Option Card
DeviceNet Option Card
Profibus Option Card

Selecting the appropriate card will add an icon to the tree shown in the left hand pane of the window. This icon will in turn contain icons for the additional port(s) that are made available by the card. The example below shows a TS8000 with the CANopen option card installed.



Device Settings

☑ Enable Device

Device Settings

Drop Number: 0

Send NMT Start: No

Ping Register (Hex): 1000

Element Options

Index Offset (Hex): 0

Index Calculation: Index = Offset - Code

Increment: Index

Index Entry: Hexadecimal

Protocol Options

Transaction Timeout: 25 msec

Note that the drivers available for a port will depend upon the connection type it supports. The CANopen option card, for example, shows a port that will only support drivers designed fir the CANopen communication standard.

Sharing Serial Ports

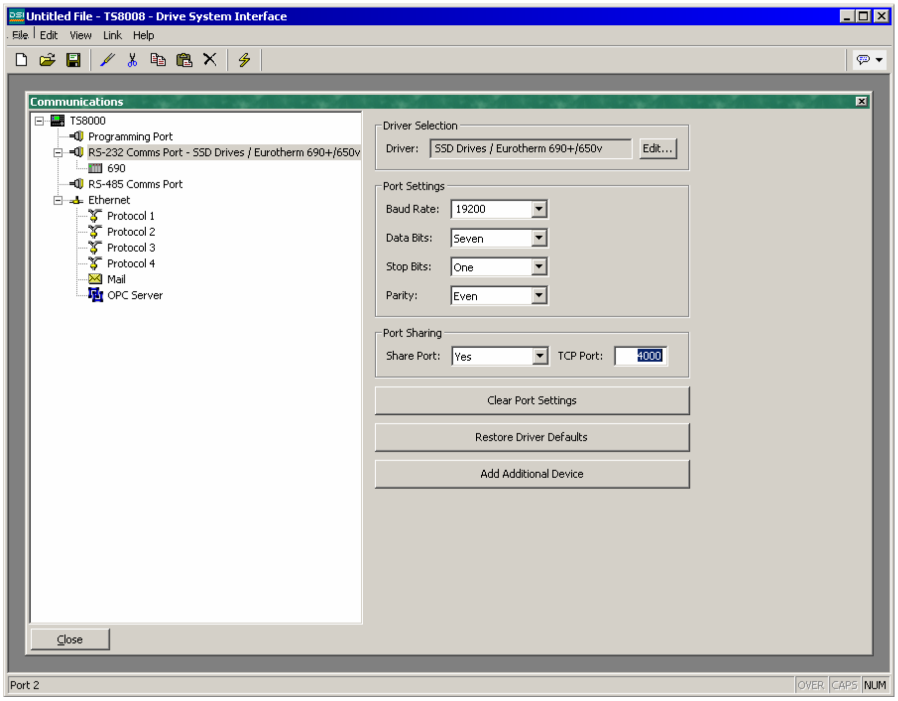
All TS8000 operator panels provide a so-called “port sharing” mechanism that allows either physical or virtual serial connections to be made to any device connected to the HMI. For example, you may be using the HMI with a 650v AC Drive, but since the drive has only one P3 programming port, you find that you are constantly swapping cables when modifying the drive’s program using CELite. By sharing the HMI’s communications port, you can send data directly to the drive, either from another serial port on the HMI, or by means of a virtual serial connection from your PC, made over an Ethernet link.

Enabling TCP/IP

The first configuration step when using port sharing is to enable the panel’s Ethernet port as described above. While you may not choose to use the virtual serial port facility, even the sharing of local ports is based upon TCP/IP protocol, which will not be available unless Ethernet is enabled. To enable Ethernet, select the Ethernet icon in the Communications window, and select the required configuration mode. For installations where Ethernet is not actually being used, you can select Manual Configuration and leave the rest of the options at their defaults.

Sharing The Required Port

The next step is to share the required port, which is done by selecting Yes in the Share Port property and by optionally entering a suitable TCP/IP port number. This number represents the virtual port that will be used to expose the serial port for access via TCP/IP.



Driver Selection

Driver: SSD Drives / Eurotherm 690+/650v

Port Settings

Baud Rate: 19200

Data Bits: Seven

Stop Bits: One

Parity: Even

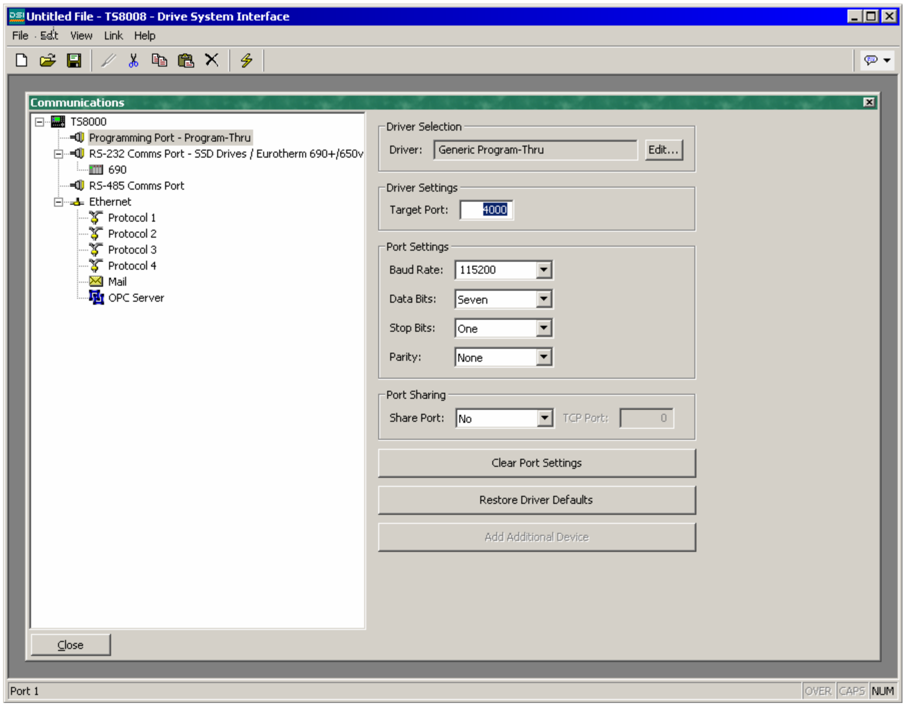
Port Sharing

Share Port: Yes TCP Port: 4000

If you leave this port setting at zero, a number of 4000 plus the logical index of the port will be used (to obtain the logical index of the port, count the port's position in the list, noting that the Programming Port is always logical port 1). You may use any number that is not already used by another TCP/IP protocol. If you are stuck for ideas, we recommend numbers between 4000 and 4099.

Connecting Via Another Port

If you want to use another port on the HMI to route data to the shared port, you must select the Generic Program Thru driver for that port, and configure this driver with the TCP/IP port number of the serial port that you have shared. In the example below, we are routing data from the Programming Port to a drive that is connected via the RS-232 Comms port.



Driver Selection

Driver: Generic Program-Thru

Driver Settings

Target Port: 4000

Port Settings

Baud Rate: 115200

Data Bits: Seven

Stop Bits: One

Parity: None

Port Sharing

Share Port: No TCP Port: 0

Note that the Baud Rate and other port settings do not have to be the same as those for the port which we are sharing. In the configuration shown above, data to and from the programming software is sent at a higher Baud Rate than the data to and from the drive, with the TS8000 doing the appropriate buffering and conversion.

In this example, to make use of the shared port, you would connect a spare serial port on your PC to the Programming Port on the TS8000, and configure the CELite programming software to talk to this COM port. As soon as the PC begins to talk to the drive, communications between the TS8000 and the drive will be suspended, and the TS8000's two ports will be "connected" in software, such that the PC will appear to be talking directly to the drive. If no data is transferred for more than a minute, communications between the TS8000 and the drive will be resumed.

Connecting Via Ethernet

Rather than using an additional serial port on your PC and on the HMI, it is possible to use a third-party utility to create what are known as Virtual Serial Ports on your computer. These appear to applications to be physical COM ports, but in fact, they send and receive data to a remote device over TCP/IP. By installing one of these utilities and configuring it to address the TS8000 HMI, you can have serial access to any devices connected to the HMI without any additional cabling. Indeed, there is no need to have any physical serial ports available on the PC at all.

Several third-party virtual serial port utilities are available as shareware on the Internet, as well as many commercial products that are available for a nominal license fee.

Pure Virtual Ports

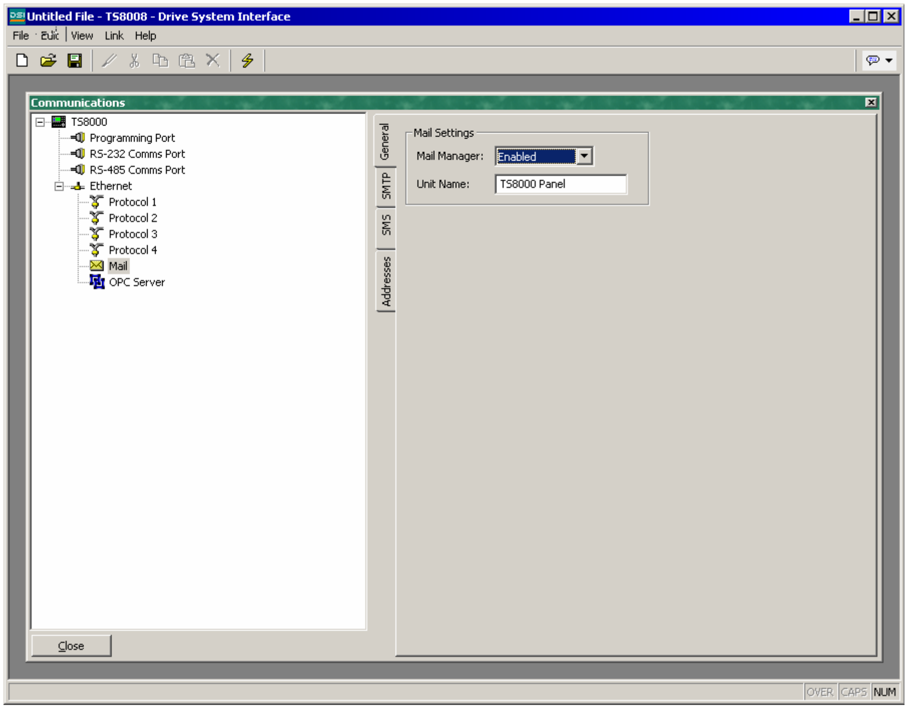
In some circumstances, you may want to use a spare serial port on a TS8000 to provide access to a remote device that is not otherwise connected to the HMI. Or you might want to use such a port to connect to a dedicated programming port on a device, even though the TS8000 is using another port to perform communications with that device. For example, if you have a 690+ AC Drive connected to a TS8000, you may typically communicate using Ethernet, or via the drive's P3 serial port. If you wish to use port sharing to remotely configure the drive, you may wish to connect the drive's P3 port to a spare port on the TS8000 so that you may then share this port via TCP/IP. To do this, configure the port in the usual manner, selecting the Virtual Serial Port driver for that port. Then, share the port as described above. This virtual Serial Port driver performs no communications activity of its own, but still allows the device to be shared for remote access.

Limitations

Note that some programming packages may not work with virtually or physically shared ports. Issues to watch out for are tight timeouts that do not allow the TS8000 time to relay the data to the device; a reliance on sending break signals or on the manipulation of hardware handshaking lines; or DOS-style port access such that the package cannot "see" the virtual serial ports. Luckily, these issues are rare, and most packages will happily communicate as if they were directly connected to the device in question.

Using Electronic Mail

DSI8000 can be configured to send email messages when alarm conditions are present, or when notification needs to be provided of other events within the system. The methods used to deliver email are configured via the Mail icon in the Communications window.



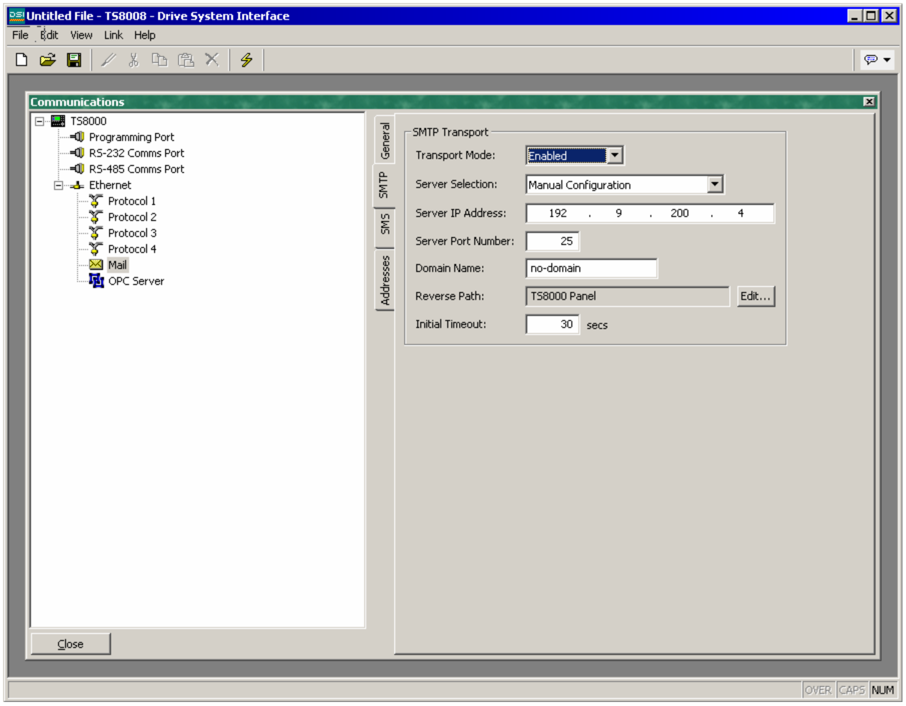
Mail Manager: Enabled

Unit Name: TS8000 Panel

The properties section of the General tab are used to enable or disable the mail manager, and to provide a name for the operator panel. This name will be used within email messages to identify the originator of the message. Applications will typically use the name of the machine to which the TS8000 is attached, or the name of the site that it is monitoring.

Configuring SMTP

The SMTP tab is used to configure the Simple Mail Transport Protocol. This is the standard protocol used to send email over the Internet or over other TCP/IP networks. SMTP addresses follow the familiar name@domain.extension convention. The configuration options for SMTP transport are shown below:



SMTP Transport

Transport Mode: Enabled

Server Selection: Manual Configuration

Server IP Address: 192 . 9 . 200 . 4

Server Port Number: 25

Domain Name: no-domain

Reverse Path: TS8000 Panel

Initial Timeout: 30 secs

- The *Transport Mode* property is used to enable or disable the transport. Note that the mail manager must be enabled via the General tab before the SMTP transport can be enabled. Note also that either SMTP or SMS must be enabled if the mail manager is able to deliver messages.

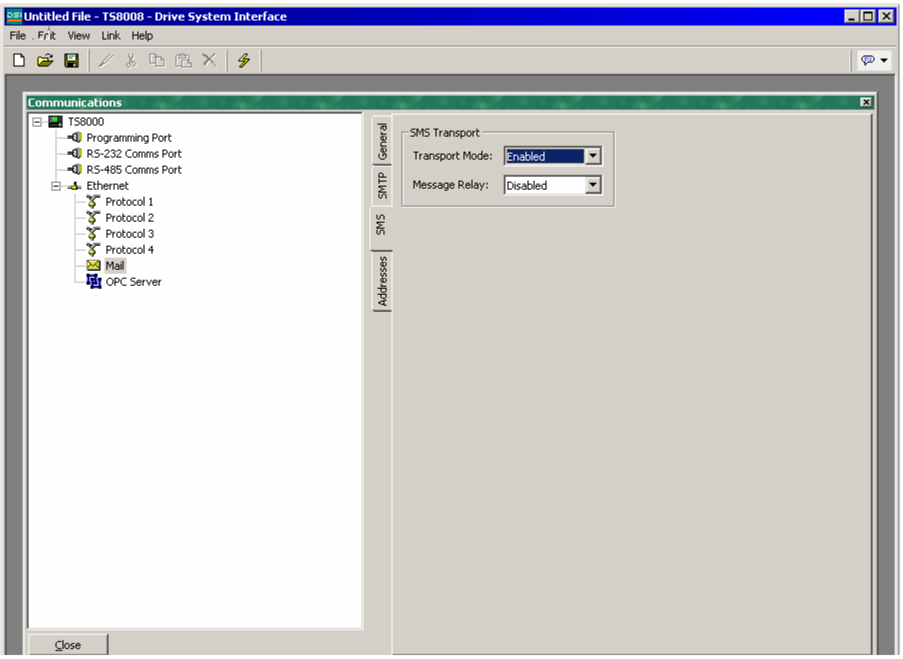
- The *Server Selection* property is used to define how the transport will locate an SMTP server. If Manual Selection is used, the *Server IP Address* property should be used to designate a server. If *Configured via DHCP* is selected, the unit's Ethernet port must be configured to use DHCP, and the network's DHCP server must be configured to designate an SMTP server via option 69.

- The *Server IP Address* property is used to designate an SMTP server when manual server selection is enabled. The server must be configured to accept mail from the panel, and to relay messages if required by the application.

- The *Server Port Number* property is used to define the TCP port number that will be used for SMTP sessions. The default value is 25. This value will be suitable for most applications, and will only need to be adjusted if the SMTP server has been reconfigured to use another port.
- The *Domain Name* property is used to specify the domain name that will be passed to the SMTP server in the HELO command. The vast majority of SMTP servers ignore this string. In the unlikely event that your SMTP server attempts to do a DNS lookup to confirm the identity of its client, you may need to enter something appropriate to your DNS configuration.
- The *Reverse Path* property is used to specify the email address that will be supplied as the originator of the messages sent by the operator panel. The property comprises a display name, and an email address. Since the panel is not capable of receiving messages, the email address will often be set to something that will return an "undeliverable" message if a reply is sent.
- The *Initial Timeout* property is used to specify how long the mail client will wait for the SMTP server to send its welcome banner. Some Microsoft servers attempt to negotiate Microsoft specific authentication with mail clients, thereby delaying the point at which the banner appears. You may want to extend this time period to 2 minutes or more when working with such servers.

Configuring SMS

The SMS tab is used to configure the Short Message Service. This transport is used to send text messages to cell phones via a GSM Modem. Email addresses for SMS comprise and international format telephone number, minus the introductory plus-sign. An example address in the United States would be 17045883246, while an example in the United Kingdom would be 441903737000. In each case, the address comprises the country code, the area code, and number. The configuration options for SMS are shown below:



SMS Transport

Transport Mode: Enabled

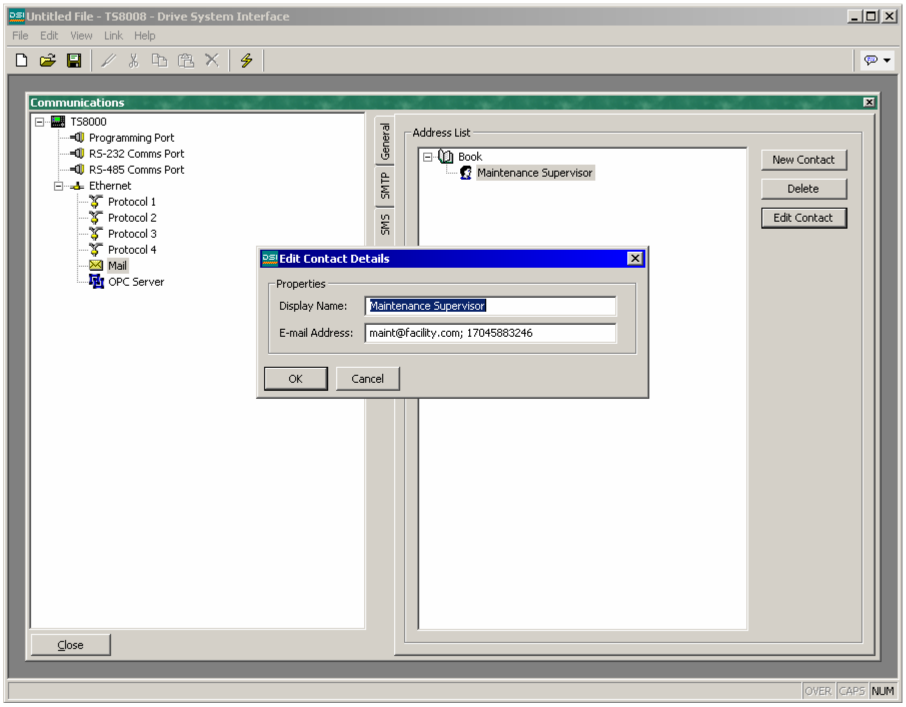
Message Relay: Disabled

- The *Transport Mode* property is used to enable or disable the transport. Note that the mail manager must be enabled via the General tab before the SMS transport can be enabled. Note also that either SMTP or SMS must be enabled if the mail manager is able to deliver messages.
- The *Message Relay* property is used to enable or disable the panel's SMS relay feature. If this feature is enabled, a user who receives an SMS message that has been sent to several recipients can reply to that message, and have the operator panel relay the message to the other recipients. This provides a simple conferencing facility between message recipients.

Note that for the SMS transport to operate, a GSM modem must have been installed in one of the unit's serial ports. Refer to later sections of this chapter for details on how to configure such a modem, and on the interaction of multiple modems.

The Address Book

The Addresses tab is used to define email recipients. An unlimited number of address book entries can be added, edited or deleted using the buttons in the right-hand pane. Each entry can refer to one or more email recipients from any of the transports enabled by the database. Recipients for multiple transports can be included in the same entry. The dialog used to define the properties of each recipient is shown below.



Display Name: Maintenance Supervisor

E-mail Address: maint@facility.com; 17045883246

- The *Display Name* property is used to define the human-readable name of the address book entry. This is the name that will be used for the display name of the SMTP recipients, and choosing an address book entry within DSI8000.

- The *Email Address* property is used to define one or more recipients for this address book entry. Multiple recipients should be separated by semicolons. The format of each recipient will depend upon the transport that is expected to deliver the message. In the example above, the address book entry refers to one SMTP recipient and one SMS recipient.

Working With Modems

This section explains how to configure your TS8000 panel to work either with modems or with direct serial connections to computers running the Windows operating system. Note that DSI8000's modem support is based upon the Point-To-Point Protocol, otherwise known as PPP. While protocols such as those by Modbus allow a single conversation to occur between any two devices, PPP is more akin to an Ethernet connection in that it allows an unlimited number of logical connections to exist on a single physical link. A single PPP connection can thus allow simultaneous access to the panel's TCP/IP download facility, its web server, its shared serial ports, and to any TCP/IP protocols that have been selected via the Communications window.

Some Typical Applications

The sections below list some typical applications using modem technology.

- A TS8000 in a remote location can send an email to a service engineer to inform him of a fault condition. By configuring an on-demand connection to an Internet Service Provider, the panel is instructed to automatically connect when an email is to be sent, and then to hang up when the message has been transferred.

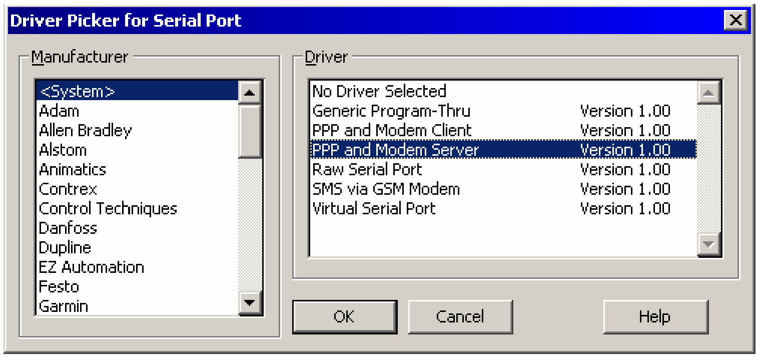
- A TS8000 in a remote location can send messages directly to the cell phones of a group of service engineers to inform them of a fault condition. By configuring a GSM modem with SMS support, the panel is instructed to notify the engineers of the fault by means of short text messages. Further, when a given engineer replies to the message to indicate that he will service the call, the TS8000 can optionally forward the reply to the rest of the group, letting them know that the issue is being attended to.

- A TS8000 in a remote location is configured to accept incoming connections from a PC based at a central office. Once the connection is made, the panel's database can be remotely upgraded by instructing the DSI8000 configuration to download via the TCP/IP link. If so configured, the panel's web server can be accessed so as to provide remote control facilities. By installing virtual serial port software on the PC and by enabling port sharing on the TS8000, a PLC programming package can be used to download to the controller connected to the operator panel - the software acts like it is talking over a standard COM port.

- A TS8000 in a remote location is configured to accept incoming connections from a SCADA system located in a central office. The SCADA package can use Modbus TCP/IP to access gateway blocks within the panel, thereby reading and writing data collected from devices connected to the TS8000's serial ports. The SCADA package can also make direct contact with devices connected to the panel by means of the TS8000's IP routing capability.

There are obviously many other examples that one can derive beyond those listed above.

Adding A Dial-In Connection

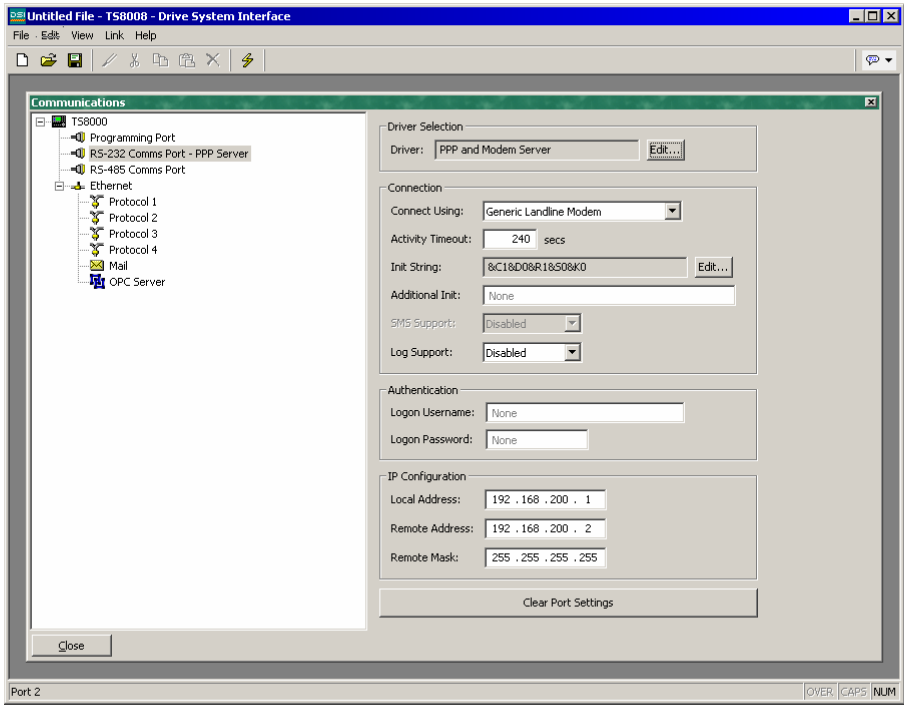


Manufacturer:
- <System>
- Adam
- Allen Bradley
- Alstom
- Animatics
- Contrex
- Control Techniques
- Danfoss
- Dupline
- EZ Automation
- Festo
- Garmin

Driver:

No Driver Selected	
Generic Program-Thru	Version 1.00
PPP and Modem Client	Version 1.00
PPP and Modem Server	Version 1.00
Raw Serial Port	Version 1.00
SMS via GSM Modem	Version 1.00
Virtual Serial Port	Version 1.00

To add a dial-in connection to your database, open the Communications window and select the serial port to which the connection will be made. Click on the Edit button of the Driver field in the right-hand pane, and select the *PPP and Modem Server* driver from the System selection of the dialog box. The right-hand pane will now show the modem configuration.



Driver Selection

Driver: PPP and Modem Server

Connection

Connect Using: Generic Landline Modem

Activity Timeout: 240 secs

Init String: &C1&D0&R1&S0&K0

Additional Init: None

SMS Support: Disabled

Log Support: Disabled

Authentication

Logon Username: None

Logon Password: None

IP Configuration

Local Address: 192 . 168 . 200 . 1

Remote Address: 192 . 168 . 200 . 2

Remote Mask: 255 . 255 . 255 . 255

The modem has the following configuration options.

- The *Connect Using* property is used to select the physical device to be used to make the connection. The devices supported at this time are direct serial connections to computers running the Microsoft Windows operating system, generic landline modems which implement the Hayes command set, and those for GSM mode cellular use.

- The *Activity Timeout* property is used to define how long a period must pass without the TS8000 sending a packet over the PPP link in order for the connection to be terminated. For dial-in connections, it is assumed that the connecting device is friendly, so no effort will be made to filter out optional packets that might result in the link staying active for long periods. Note that even if you want a permanent connection, you must enter a suitable timeout so as to allow the detection of dead links. This implies that so-called permanent connections may still drop on occasions, but since the client will immediately reestablish the link, this is not an issue.

- The *Additional Init* string is used with non-direct links, and provides a series of AT commands to be used to initialize the modem. The initial AT prefix is not required. Several commands may be combined by simply placing one after the other. The exact string that will be required for your modem is dependent upon its internal software, so if you contact Technical Support for assistance, be sure to have exact make and model information available.

- The *SMS Support* property is used to enable Short Message Service messaging when using a GSM modem. In order for SMS messaging to operate properly, you will also have to enable the SMS Transport using the Mail icon in the Communications window as described above.

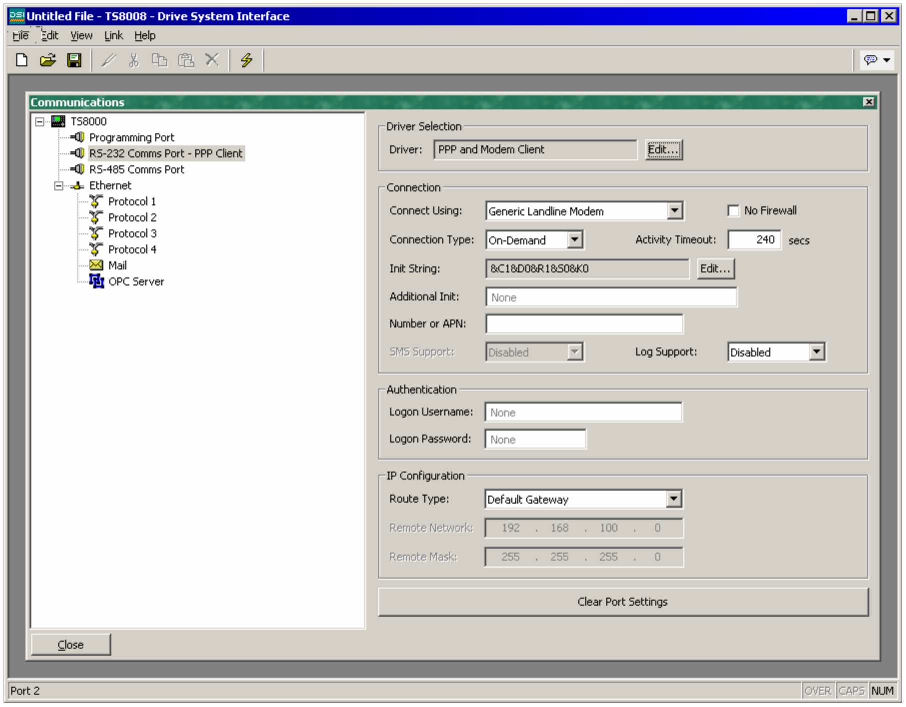
- The *Logon Username and Logon Password* properties are used to define the credentials that the remote client must provide in order to be allowed to connect to this device. The username is not case sensitive, while the password is. DSI8000's PPP implementation will ask its peer to use CHAP authentication to avoid transmitting or receiving plaintext password, but will fallback to using PAP if the remote client does not support CHAP.

- The *Local Address* property is used to define the IP address to be allocated to the local end of the connection. This will thus be the IP address of the TS8000 for this link. Please note that this must not be the same as the IP address of the TS8000's Ethernet port, as every physical IP interface must have a distinct IP address. The default value will work in most situations, unless your network design demands that you use a different setting.

- The *Remote Address* property is used to define the IP address to be allocated to the remote end of the connection. It is used together with the Remote Mask property to determine what packets will be routed to this connection. For most applications, a mask of 255.255.255.255 will be used, thereby instructing DSI to send via this interface only those packets directly bound for the remote client. A mask of 0.0.0.0, by contrast, will allow all packets that do not specifically match another interface to be forwarded to the remote client, presumably for further forwarding to the intended host. Intermediate masks may be used to control exactly which packets are sent.

Adding A Dial-Out Connection

Dial-out connections are added exactly as above, expect that the PPP and Modem Client driver should be selected for the required port. The configuration options for this modem are shown below.



Driver Selection
Driver: PPP and Modem Client

Connection
Connect Using: Generic Landline Modem
□ No Firewall
Connection Type: On-Demand
Activity Timeout: 240 secs
Init String: &C1&D0&R1&S0&K0
Additional Init: None
Number or APN:
SMS Support: Disabled
Log Support: Disabled

Authentication
Logon Username: None
Logon Password: None

IP Configuration
Route Type: Default Gateway
Remote Network: 192 . 168 . 100 . 0
Remote Mask: 255 . 255 . 255 . 0

The modem has the following properties that differ from those of dial-in connections:

- The *Connect Using* property is as for dial-in connections, with the addition of support for GPRS connections via a GSM modem. These connections differ from CSD connections in that they achieve much higher speeds, and are typically charged on the basis of how much data is transferred rather than how long the connection is maintained. GPRS connections may thus be configured for permanent connection, unless there is a need to provide downtime to allow SMS messages to be transferred.

- The *No Firewall* property is used to turn off the firewall protection that is otherwise provided for dial-out connections. This protection prevents incoming connections from being made to this interface, and prevents the TS8000 from sending certain diagnostic packets that might either provide a hacker with information about the system, or might be used by an attacker to keep a connection active in the absence of actual data transfer. If you are connecting directly to the Internet by means of this connection, you should not normally turn off the firewall. The firewall should be disabled only for connections to corporate networks or to other controlled environments.

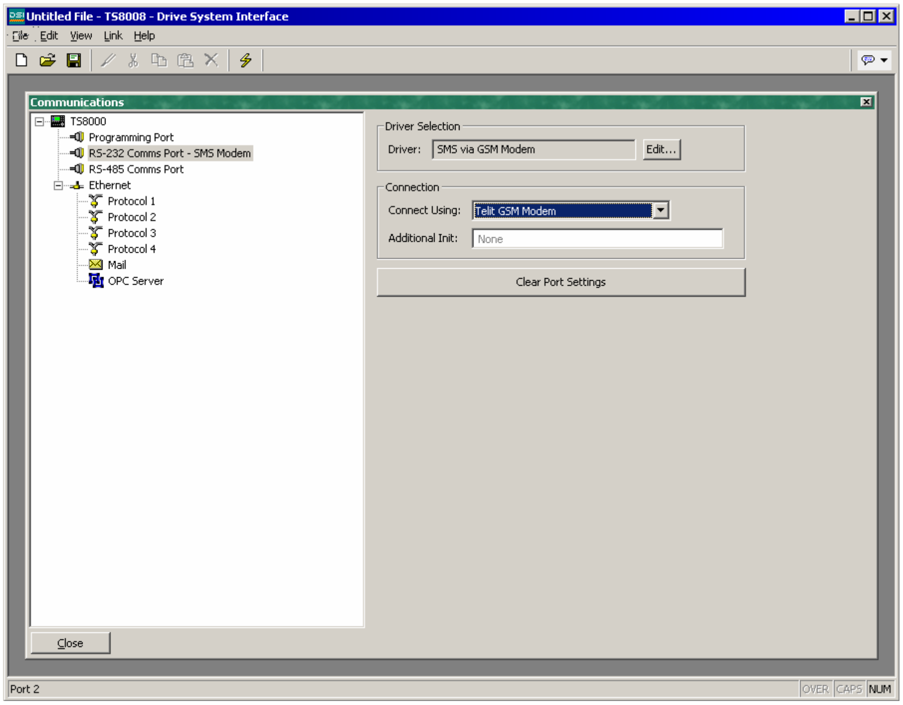
- The *Connection Type* property is used to indicate whether you want this connection to be permanently maintained, or whether you want it to be established automatically when an attempt is made to transfer data to hosts that are reachable via this interface. If you select an on demand connection, you must specify the timeout after which the link will be terminated if no packets have been transmitted by the TS8000.

- The *Logon Username and Logon Password* properties are used to define the credentials that will be passed to the remote server when attempting to initialize this connection. The username is not case sensitive, while the password is. DSI8000's PPP implementation will ask its peer to use CHAP authentication to avoid transmitting or receiving plaintext password, but will fallback to using PAP if the remote server does not support CHAP.

- The *Route Type* property is used to define the data that will be transferred via this interface. For on demand connections, this effectively defines when the connection will be activated. If *Default Gateway* is selected, any packets that do not match the address and netmask of the Ethernet connection will be sent to this interface. Note that in this mode, the Ethernet port must have a gateway setting of 0.0.0.0, or it will take all the packets and leave none to activate the modem. If *Specific Network* is selected, you must provide the address and netmask that defines the network to which packets will be routed.

Adding An SMS Connection

SMS connections are used when text messaging functionality is required, but where neither dial-in nor dial-out PPP connections will be established. They are configured as described above, except that the *SMS via GSM Modem* driver should be selected for the required port. The configuration options for this modem are shown below:



Driver: SMS via GSM Modem

Connect Using: Telit GSM Modem

Additional Init: None

The device properties are a subset of those provided for dial-in connections. SMS support is always enabled with this driver, but once again, note that in order for SMS messaging to operate properly, you will also have to enable the SMS transport using the Mail icon in the Communications window.

SMS Message Processing

When SMS messaging is enabled, the TS8000 will instruct the GSM modem to check for new incoming or outgoing messages every five seconds. Incoming messages are forwarded to the mail manager, which will optionally forward them to other users according to its configuration. Note that it is not possible to check for messages while the modem is connected to a CSD or GPRS session, so you will want to avoid using permanent connections when working with SMS. Note also that if more than one GSM modem is configured, all will be able to receive messages, but only the second modem will be used for sending.

Using Multiple Interfaces

Each TS8000 panel can support up to two modem independent connections. When combined with the Ethernet port, this gives a total of up to three distinct IP interfaces, all of which will operate according to the configuration parameters defined for each connection. This section describes how these multiple interfaces will interact, and how the TS8000 will decide where to send each packet of data.

Interface Selection

Each interface has an IP address and a network mask, which are used to decide whether to forward packets to that interface. For example, if the Ethernet interface is configured with an IP address of 192.168.1.0 and a network mask of 255.255.255.0, any packets for IP addresses starting with 192.168.1 will be sent to this interface. Likewise, if an on-demand modem connection has a remote IP address of 192.168.2.2 and a netmask of 255.255.255.255, sending a packet to address 192.168.2.2 will result in the connection being established.

Default Route

In addition, one single interface may also define a default route, which will be used to handle packets that do not specifically match any other interface. The method used to configure the route varies according to the interface type, as shown in the table below.

Interface	To Define Default Route
Ethernet	Enter a non-zero value for the *Gateway* property.
Dial-In	Enter 0.0.0.0 for the *Remote Mask*.
Dial-Out	Select Default Gateway for the *Route Type* property

Note again that only a single interface may define a default route. For example, a TS8000 panel may be connected to a number of Ethernet devices using an IP address of 192.168.1.0 and a netmask of 255.255.255.0, with no gateway defined. An on-demand modem connection may be configured to access an Internet Service Provider so as to send alarm emails. Its *Route Type* is set to Default Gateway, making it the route for any packets for IP addresses that do not match the network defined for the Ethernet port. The SMTP server is configured as 24.104.0.39, resulting in a dial-out connection when an attempt is made to send a message.

IP Routing

The Ethernet icon in the Communications window contains a property called IP Routing. If this facility is enabled, incoming packets from non-firewalled modem interfaces will be compared against the IP address and netmask for the Ethernet interface, and will be forwarded to that interface should a match occur. This facility is most often used with dial-in connections, and allows IP access to all devices connected to the Ethernet port, provided a suitable route is defined by the client.

Checking The Modem Status

In order to help debug modem connections, DSI8000 provides the GetInterfaceStatus function. This function takes a single argument, which is the numeric index of the required interface. Interface zero is always the panel's loopback interface. Next comes the Ethernet interface, if it is enabled, such that the first PPP interface is numbered 1 when Ethernet is disabled and 2 when it is enabled.

The function returns a string, which can be interpreted according to the following table:

Status	Meaning
Closed	The interface has not yet been initialized. This state will only occur for a short time during system start-up.
Init	The modem is being initialized. If the connection remains in this state, there are probably errors in the init strings being sent to the modem.
Idle	The link is idle. GSM modems will return a number at the end of the string to indicate signal strength. The next table explains how to interpret these values.
SMS	The modem is sending SMS messages, or polling the modem to see if new SMS message are available. If SMS messaging is enabled for a modem, you will see this state appear for a short period every five seconds.
Connecting	The modem is establishing a connection. This state typically appears only for client connections, and indicates that a call is being placed.
Listening	The modem is waiting for a call. This state appears only for server connections. GSM modems will also return an Idle state while waiting for a call in order to show signal strength.

Status	Meaning
Answer	The modem is answering a call and trying to negotiate the Baud rate for the connection. This state appears only for server connections. If the connection is established, the modem will enter the Connected state.
Connected	The modem has established a connection. This state will persist for only a short time, as the LCP negotiation process will begin after a small delay.
Neg LCP	The connection is negotiating LCP options. This process decides on a set of link protocol settings that are acceptable to both the client and the server.
Auth	The connection is performing the authentication process to ensure that the appropriate user credentials are used.
Neg IPCP	The connection is negotiating IPCP options. This process decides on a set of network protocol settings that are acceptable to both the client and the server.
Up	The connection is active and IP data can be exchanged.
Hanging Up	The modem is disconnecting. This state will exist for only a short time before the modem returns to Idle.

The signal strength values returned by GSM modems have the following meaning:

Value	Signal Strength
0	–113dBm or less.
1	–111dBm.
2-30	–109dBm to –52dBm in 2dBm steps.
31	–51dBm or greater.
99	Signal strength cannot be determined.

Cell phones typically interpret these values as follows when displaying signal strength:

Value	Strength	Number of Bars
5 or less.	–103dBm or less.	One
6 thru 9.	–101dBm thru –95dBm	Two
10 thru 14.	–93dBm thru –85dBm	Three
15 or greater.	–83dBm or greater.	Four

Configuring Data Tags

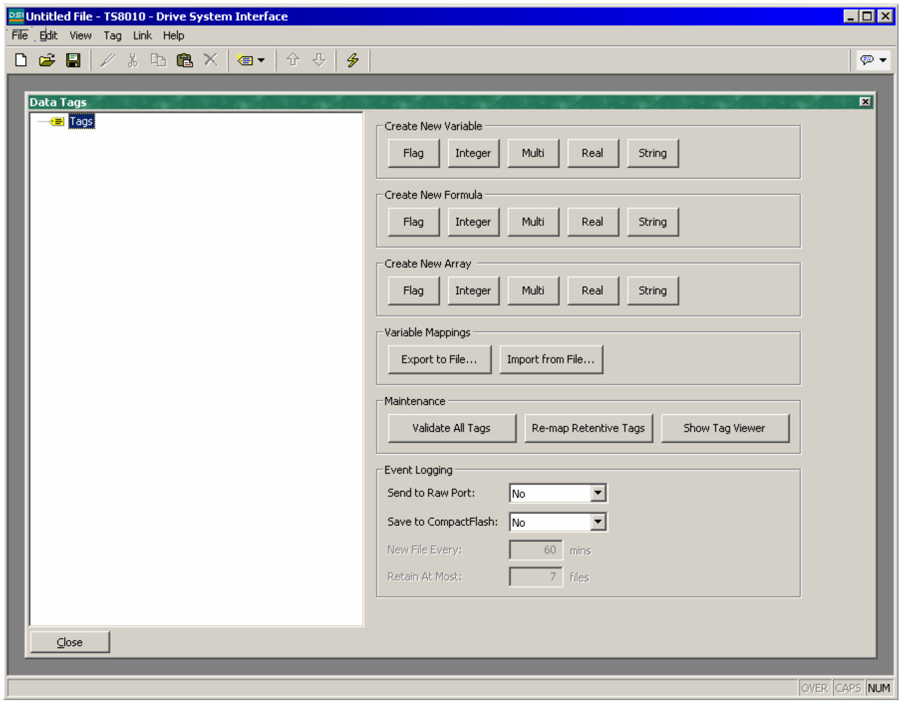
Once you have configured the communications options for your database, the next step is to define the data items that you want to display or otherwise manipulate. This is done by selecting the Data Tags icon from the main screen.

About Tags

Data Tags are named entities which represent data items within the operator terminal. Tags may be "mapped" to registers in remote devices, in which case DSI will automatically read the corresponding register when the tag is referenced or displayed. Similarly, if you change a mapped tag, DSI will automatically write the new value to the remote device.

Types Of Tags

When you first open the Data Tags window, you will see that the right-hand pane contains an apparently bewildering number of buttons that can be used to create different kinds of data tags. While all these buttons may seem a little intimidating at first, the fifteen different kinds of tag can be broken down into three families, each containing five members.


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Create New Variable
Flag Integer Multi Real String
Create New Formula
Flag Integer Multi Real String
Create New Array
Flag Integer Multi Real String
Variable Mappings
Export to File... Import from File...
Maintenance
Validate All Tags Re-map Retentive Tags Show Tag Viewer
Event Logging
Close
OVER CAPS NUM

Send to Raw Port: No
Save to CompactFlash: No
New File Every: 60 mins
Retain At Most: 7 files

Tag Families

The three families of tags are listed below:

- *Variables* represent a single data item within the terminal. Variables may be mapped to PLC registers, and may be configured as retentive, in which case their values will be kept in memory even when the operator panel is powered off. The defining characteristics of a variable are that they contain a single item, and that it is *in theory* possible to write to this item, even if in practice the variable is configured as read-only.

- *Formulas* represent derived values. They are a combination of other data items, typically combined using one or more math operations. For example, a formula might represent the sum of two tank levels. While a formula can be set to be equal to the contents of a PLC register, it is not truly mapped to that register, in that it can *never* be written to and thus cannot be considered to be equivalent to that register. The need for this restriction is obvious if you consider a formula such as Tank1+Tank2. What would it mean to write to this expression?

- *Arrays* represent a collection of data items within the terminal. They cannot be mapped to PLC registers, but instead they represent a list of values within the panel's own memory. These are typically used to store recipe data, or to build up collections of data for statistical analysis. They are not used in the majority of simple applications, but provide a powerful tool for more complex projects.

Tag Types

Each family contains five tag types, each of which holds a different kind of data.

- *Flag* tags represent a single true or false condition. When they are mapped to a register in a remote device, they will typically correspond to an internal coil or to a single digital I/O point. Flag formulas typically represent combinations of such items, or comparisons of numeric values.

- *Integer* tags represent 32-bit signed numbers. These tags can store values between –2,147,483,648 and +2,147,483,647. Even if a tag is mapped to a PLC register which contains only 16 bits of data, DSI performs its internal operations at the higher level of precision to ensure large intermediate values can be handled with ease.

- *Multi* tags represent numeric values that correspond to a number of distinct states. Thus, while an integer might represent a tank level, a multi tag will represent, say, one of three states of a machine, such as Stopped, Running or Paused. The distinction is obvious when you consider that a multi tag is displayed as one of a set of strings, while an integer is displayed as a number.

- *Real* tags represent 32-bit single precision floating-point numbers. These tags can store values between $\pm 10^{-38}$ and $\pm 10^{+38}$ with a precision of about 7 significant figures. While it is seldom necessary to use real tags to represent physical quantities (which typically have more tightly defined ranges) they are useful for performing statistical operations or other math functions.

- *String* tags represent an item of text made up of a number of characters. They are used to store such things as recipe names, or to process data received using Raw Port device drivers.

Tag Usage

Given all these various options, you may wonder why you would want to use tags in the first place? After all, DSI allows you to directly place a PLC register on a display page, so you can in fact configure a simple database without ever opening the Data Tags icon. The basic answers are as follows:

- Tags allow you to name data items, so you know which data item within the PLC you are referring to. Further, if the data in the PLC moves or if you decide to switch to an entirely different family of PLC, you can simply re-map the tags, and avoid having to make any other changes to your database.
- Tags allow you to avoid re-entering the same information again and again. When you create a tag, you specify how the tag is to be displayed. In the case of an integer tag, this means you tell DSI how many decimal places are to be used, and what units, if any, are to be appended to the value. When you place a tag on a display page, DSI knows how to format it without you having to do anything further. Similarly, if you decide to change the formatting, and perhaps switch from one set of units to another, you can do this in one place, without having to edit each display page in turn.
- On terminals with color displays, tags are used as the basis for color animation. The various colors that are defined for a tag can be used to specify the way in which other animation objects will be displayed. Without tags, you will have no way of changing the color of anything other than text-based data fields.
- Tags are the key to implementing slave protocols. DSI treats these protocols as mechanisms for exposing data items within the terminal. This allows the same data to be accessed via multiple ports, so that, for example, a machine setting could be changed by both a local SCADA package, and a similar package working over Ethernet from a remote site. Without tags, there would be nothing to expose, and this mechanism could not be implemented.
- Tags are used within DSI to implement many advanced features. If you want to use functionality such as alarms, triggers, data logging or the web server, you will have to use tags, period. The formatting data from the tag definition is typically required by all these features, so tags are mandatory for their operation.

In other words, tags will automate many tasks during programming, saving you time. Even if you decide not to use tags, many of the subsequent chapters of this manual refer to concepts discussed in this chapter. You should thus read it thoroughly before proceeding.

Creating Tags

To create a tag, either click on one of the buttons displayed when the Tags icon is selected in the left-hand pane of the Data Tags window, or use the new tag buttons on the toolbar. Either way, a new tag will be added to the tag list. To edit the tag's name, select the tag in the left-hand pane, and type in the new name.

Tag names must conform to the following rules:

- Tag names may not contain spaces or punctuation.
- Tag names must start with a letter of the alphabet or an underscore.
- Subsequent characters must be digits, letters or underscores.
- Names must not exceed 24 characters in length.

Editing Tags

When a tag is selected in the left-hand pane, the right-hand pane will change to display a number of tabs, each of which shows certain properties of the tag. Depending on the family and type of the tag, different tabs may be present, and each tab may contain different fields.

No matter what kind of tag is selected, the first tab in the right-hand pane is always the Data tab. This tab contains fields which indicate what data the tag is to represent, and how that data is to be stored - and perhaps transferred to or from a remote device. The exact contents of the tab will vary according to the family and type of the tag in question.

The second tab is always the Format tab. This indicates how the data in the tag is to be formatted when shown on the operator panel's display, or when presented to a user via any other mechanism, such as a web page. The Format tab will take the same form for all tags of the same type, such that all integer tags, for example, share the same set of properties.

The balance of the tabs define alarms and triggers for the tag. These are not included for string tags or for arrays. Alarms are used to detect a condition that needs to be brought to the operator's attention, or simply to log the fact that the condition has occurred. Triggers operate in a similar way, but instead of recording the condition, they are used to execute an action.

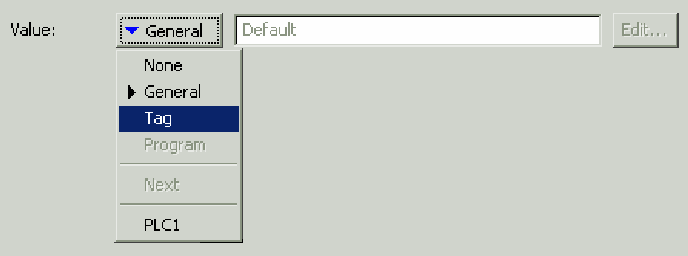
Editing Properties

Most properties are edited in ways that are self-evident to anyone who has used a Windows operating system. For example, you may be required to enter a numeric value, or to select an item from a drop-down list. Certain types of property, though, provide more complex editing options, and these are described below.

Expression Properties

Expression properties are capable of being set to:

- A constant value.
- The contents of a data tag.
- The contents of a register in a remote communications device.
- A combination of such items linked together using various math operators.

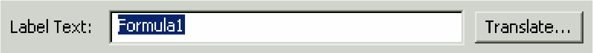


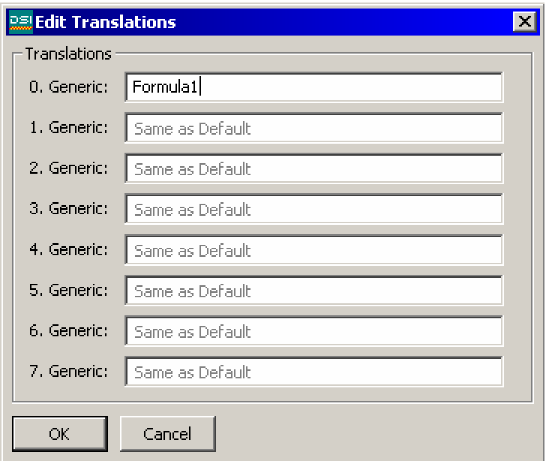
In its default state, the arrowed button immediately after the label of the property shows that the field is in General mode, and the edit box to the right of the button shows a grayed-out string which indicates the default behavior of the property.

If you are familiar with DSI's expression syntax (a complete description of which can be found in the Writing Expressions section) you can edit the property by typing an expression directly into the edit box. Most users, though, will choose to press the arrowed button and select from the menu of options that is presented.

- Selecting *Tag* will display a dialog box containing a list of data tags. You can select the tag that you want to be used to control this property. In some cases, you will also be given the chance to create a new tag and define its basic properties. This is not available when editing properties that belong directly to other data tags, as it is otherwise too easy to forget which tag you're editing.
- Selecting a *device name* will display a dialog box allowing you to choose a register within that remote communications device. The various communications devices are listed at the end of the menu in the order in which they were created.
- Selecting *Next* will set the property to be equal to the register which follows the last selected register within the last selected device. For example, if you have used the *device name* option to set a previous property to N7:10 of PLC1, selecting *Next* will set the current property to N7:11 of the same device.

Translatable Strings


Label Text: Formula1 Translate...


Edit Translations

Translations

0. Generic: Formula1
1. Generic: Same as Default
2. Generic: Same as Default
3. Generic: Same as Default
4. Generic: Same as Default
5. Generic: Same as Default
6. Generic: Same as Default
7. Generic: Same as Default

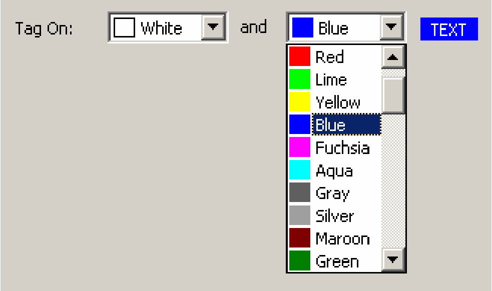
OK Cancel

DSI databases are designed to support multi-lingual operation, whereby any string which will be presented to the user of the operator panel is capable of being displayed in one of many different languages. To allow you to define these translations, properties which contain such strings have a button labeled Translate to their right-hand side.

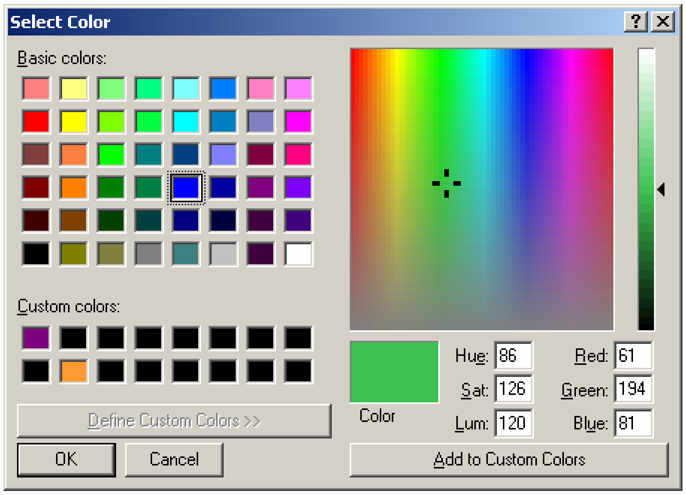
To enter the translations, click the button and the shown dialog box will appear.

If you do not enter text for a particular language, and that language is subsequently selected by the operator, DSI will use the American version by default. For information on how to configure a key or a menu to select a different language, refer to the User Interface section.

Color Properties


Tag On: White and Blue TEXT

Color properties within the data tag represent a foreground and a background color that will be used to display the tag's state in textual form. Either of these colors can then be used to define the color of other animation objects. The example shows a color pair being edited. The drop-down list contains the sixteen standard VGA colors, the user-defined custom colors, and fourteen shades of gray.


Select Color

Hue: 86 Red: 61
Sat: 126 Green: 194
Lum: 120 Blue: 81

The More option at the bottom of the list can be used to invoke the color selection dialog.

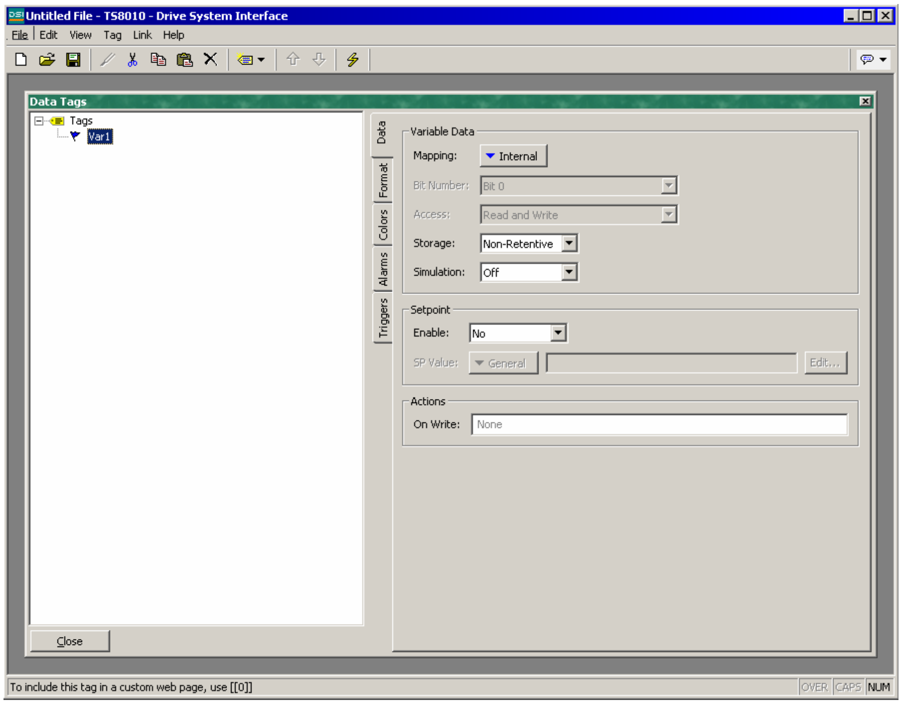
This dialog offers several ways of defining a color. You can pick from the palette, pick from the "rainbow" window, or enter the explicit HSL or RGB parameters. The dialog also allows custom colors to be added to the palette. These will appear whenever the dialog is invoked, and will also appear in the drop-down list described above. Note that not every color that is displayed in the "rainbow" will be capable of being rendered on the panel's 256-color display. Crimson will choose the nearest color within the abilities of the device.

Editing Flag Tags

You will recall that flag tags represent a true or false value. The following sections describe the various tabs that are displayed on the right-hand side of the Data Tags window when editing one of the various kinds of flag tags.

The Data Tab (Variables)

The Data tab of a flag variable contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Var1
Data
Format
Colors
Alarms
Triggers
Close
OVER CAPS NUM

Variable Data

Mapping: Internal

Bit Number: Bit 0

Access: Read and Write

Storage: Non-Retentive

Simulation: Off

Setpoint

Enable: No

SP Value: General Edit...

Actions

On Write: None

To include this tag in a custom web page, use [[0]]

- The *Mapping* property is used to specify if the variable is to be mapped to a register in a remote device, or if it exists only within the terminal. If you press the arrow button and select a device name from the resulting menu, you will be presented with a dialog box which will allow a PLC register to be selected.

- The *Bit Number* property is used when a flag variable is mapped to a PLC register which contains more than a single bit of information. The property is then used to indicate which bit within the register is to be accessed by the tag.

- The *Access* property is used to specify what sort of data transfers should be performed for a mapped variable. You may indicate that data is to be both read and written, or just read or written as appropriate. Write-only tags can be used to avoid unnecessary read operations on data which can only be changed by the terminal. They will typically be set to retentive as their value cannot be obtained from the PLC, and must therefore be stored by the terminal.

- The *Storage* property is used to indicate whether DSI should allocate FLASH memory within the panel in order to retain the value of the tag during power-down. Mapped tags which are not write-only cannot be set to retentive, as their values will in any case be read from the PLC, and it does not therefore make sense to waste local storage to retain data which will be overwritten.

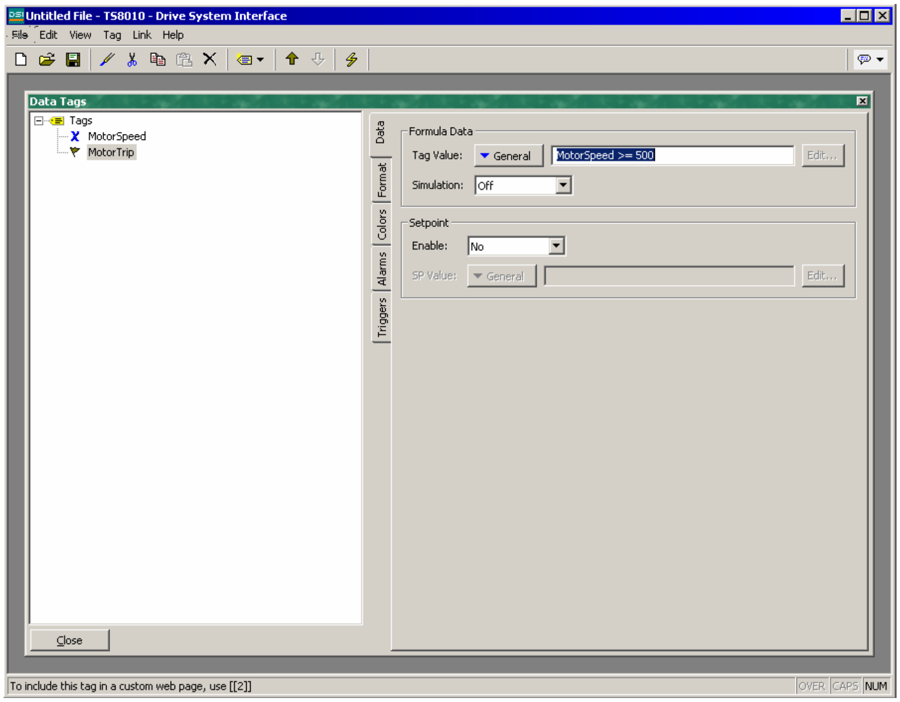
- The *Simulation* property is used to select the value that DSI will assign to this tag when displaying it within the display page editor. This facility can be useful for documenting databases, in that it allows a display page to be configured to represent a particular machine state, such that a screen capture can then be pasted into an operator manual or other documentation.

- The *Setpoint* properties are used to indicate whether a setpoint will be specified for this tag, and what that setpoint will be. Setpoints are used by certain alarm modes, and allow the actual state of a tag to be compared to its intended state. For example, a tag which represents the state of an input from a speed switch for a motor might have the motor's control output specified as a setpoint. This allows an alarm to be programmed to activate if the motor fails to start.

- The *On Write* property is used to define an action that will be executed when a change is made to the tag. This action may be used to update dependent values, or to perform other actions specific to the database. Care should be taken not to perform actions that are too complex, or system performance may be reduced.

The Data Tab (Formulas)

The Data tab of a flag formula contains the following properties:



Tag Value: General MotorSpeed >= 500

Simulation: Off

Enable: No

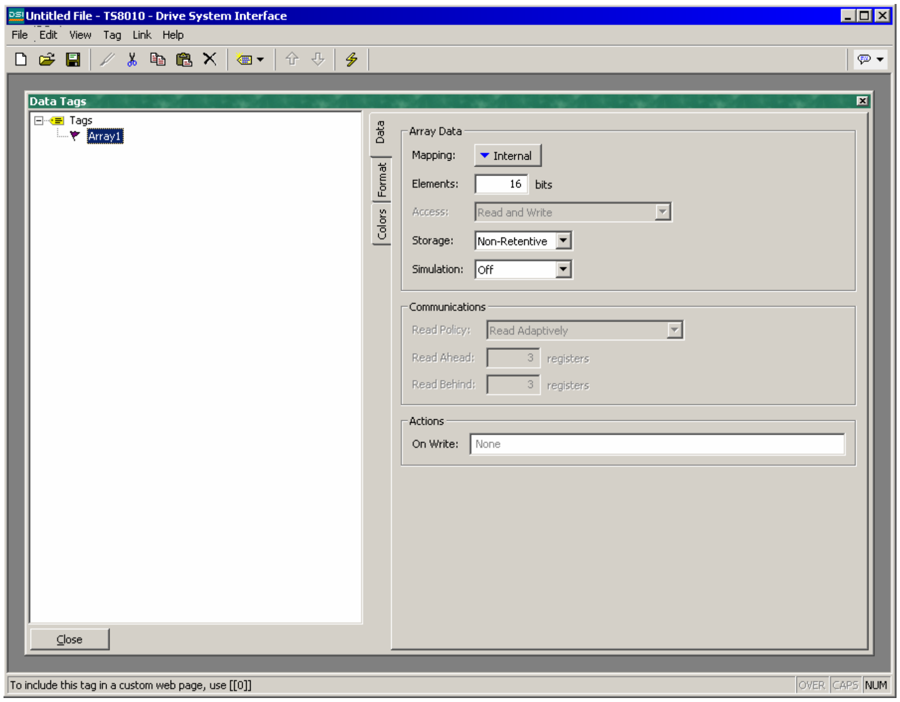
SP Value: General

To include this tag in a custom web page, use [[2]]

- The *Tag Value* property is used to specify the value that is to be represented by this tag. It is typically set to a logical combination of other tags or PLC registers, or to a comparison between numeric values. In the example shown above, the tag is configured to be true when a motor speed exceeds a certain value.
- The *Setpoint* and *Simulation* properties are as described for flag variables.

The Data Tab (Arrays)

The Data tab of a flag array contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Array1
Data
Format
Colors
Close
OVER CAPS NUM

Array Data

Mapping: Internal

Elements: 16 bits

Access: Read and Write

Storage: Non-Retentive

Simulation: Off

Communications

Read Policy: Read Adaptively

Read Ahead: 3 registers

Read Behind: 3 registers

Actions

On Write: None

To include this tag in a custom web page, use [[0]]

- The *Elements* property is used to indicate how many data items the array should hold. Array elements are referred to using square brackets, such that Array[0] is the first element, and Array[n-1] is the last element, where n is equal to the value entered for this property.
- The *Access* and *Storage* properties is as described for flag variables.
- The *Simulation* property is as described for flag variables. Note that the value to be simulated applies to all elements of the array. If you need to simulate on a per element basis, us a number of formulas to alias the array elements.
- The *Read Policy* property is used to define how DSI will read the data for arrays that are mapped to remote data items. The table below lists the various policies that can be configured, and describes their operation:

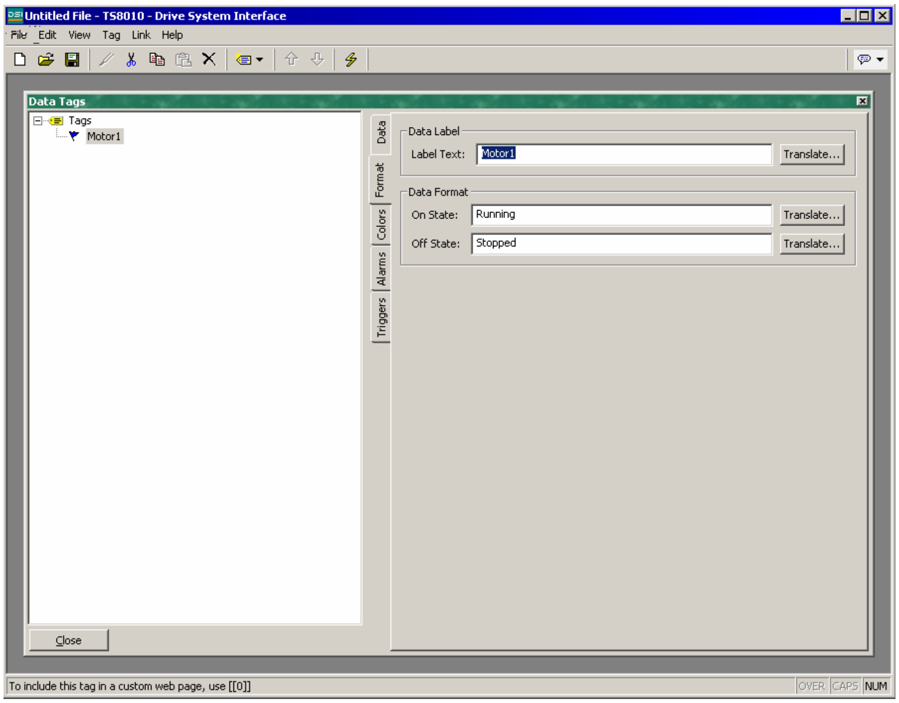
Mode	Description
Read Adaptively	Any referenced array elements will be added to the communications scan. Data either side of a referenced element, as defined by the *Read Ahead* and *Read Behind* properties, will be read as well. Old data may be displayed momentarily when an element from an adaptive array is first displayed on the panel.
Read Manually	The array will be read if and only if the ReadData function is called. This mode is useful for items that are read only rarely, or which are known not to change in the remote device.

Mode	Description
Read Whole Array	The entire array will be added to the communications scan if any element in the array is referenced. This mode ensures that all data items are available before they are referenced, but can lower system performance.

- The *Read Ahead* and *Read Behind* properties modify the behavior of the adaptive read policy by controlling how many adjoining registers will be read when a specific array element is referenced. The adjoining reads are used to maximize the chance of data being available when indirection is used to scroll up or down an array. The default values should be suitable for most applications.
- The *On Write* property is as described for flag variables.

The Format Tab

The Format tab of a flag tag contains the following properties:



Label Text: Motor1

On State: Running

Off State: Stopped

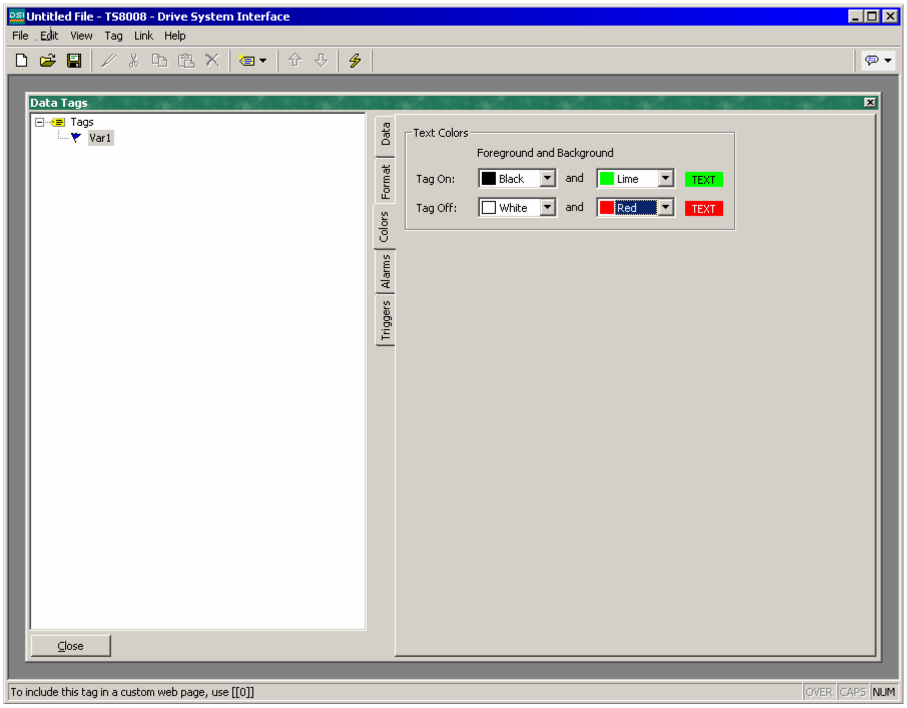
To include this tag in a custom web page, use [[0]]

- The *Label Text* property is used to specify the label that can be shown next to this tag when including the tag on a display page. The label differs from the tag name, in that the former can be translated for international applications, while the latter remains unchanged and is never shown to the user of the panel.
- The *On State* and *Off State* properties are used to specify the text to be displayed when the tag contains a non-zero and zero value, respectively. When you enter the text for the on state, DSI will attempt to generate

corresponding text for the off state by referring both to previously-created flag tags, and to its internal list of common antonymic pairs.

The Colors Tab

The Colors tab of a flag contains the following properties:



Text Colors

Foreground and Background

Tag On: Black and Lime TEXT

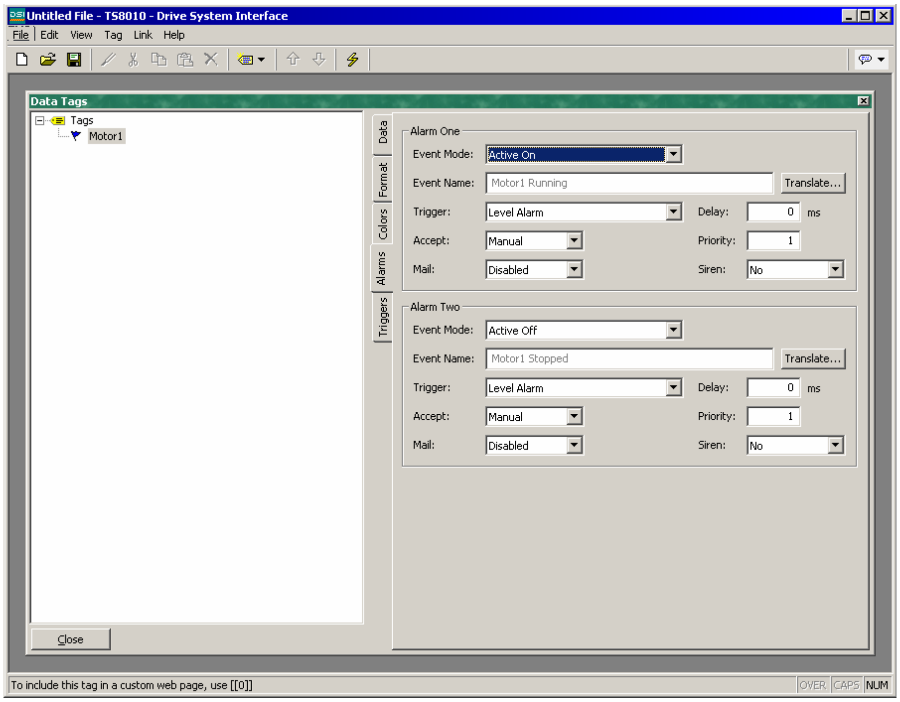
Tag Off: White and Red TEXT

To include this tag in a custom web page, use [[0]]

- The *Tag On* property is used to define the color pair to be used to display the tag when it is in the on state.
- The *Tag Off* property is used to define the color pair to be used to display the tag when it is in the off state.

The Alarms Tab

The Alarms tab of a flag variable or formula contains the following properties:



Alarm One

Event Mode: Active On

Event Name: Motor1 Running

Trigger: Level Alarm

Delay: 0 ms

Accept: Manual

Priority: 1

Mail: Disabled

Siren: No

Alarm Two

Event Mode: Active Off

Event Name: Motor1 Stopped

Trigger: Level Alarm

Delay: 0 ms

Accept: Manual

Priority: 1

Mail: Disabled

Siren: No

To include this tag in a custom web page, use [[0]]

- The *Event Mode* property is used to indicate the logic that will be used to decide whether the alarm should activate. The tables below list the available modes.

Mode	Alarm will activate when:
Active On	The tag is true.
Active Off	The tag is false.

- The following modes are only available when a setpoint is defined:

Mode	Alarm will activate when:
Not Equal to SP	The tag does not equal its setpoint.
Off When SP On	The tag does not respond to an ON setpoint.
On When SP Off	The tag does not respond to an OFF setpoint.
Equal to SP	The tag equals its setpoint.

- The *Event Name* property is used to define the name that will be displayed in the alarm viewer or in the event log as appropriate. DSI will suggest a default name based upon the tag's label, and the event mode that has been selected.

- The *Trigger* property is used to indicate whether the alarm should be edge or level triggered. In the former case, the alarm will trigger when the condition specified by the event mode first becomes true. In the latter case, the alarm will continue in the active state while the condition persists. This property can also be used to indicate that this alarm should be used as an event only. In this case, the alarm will be edge triggered, but will not result in an alarm condition. Rather, an event will be logged to the TS8000's internal memory.

- The *Delay* property is used to indicate how long the alarm condition must exist before the alarm will become active. In the case of an edge triggered alarm or event, this property also specifies the amount of time for which the alarm condition must no longer exist before subsequent reactivations will result in a further alarm being signaled. As an example, if an alarm is set to activate when a speed switch indicates that a motor is not running even when the motor has been requested to start, this property can be used to provide the motor with time to run-up before the alarm is activated.

- The *Accept* property is used to indicate whether the user will be required to explicitly accept an alarm before it will no longer be displayed. Edge triggered alarms must always be manually accepted.

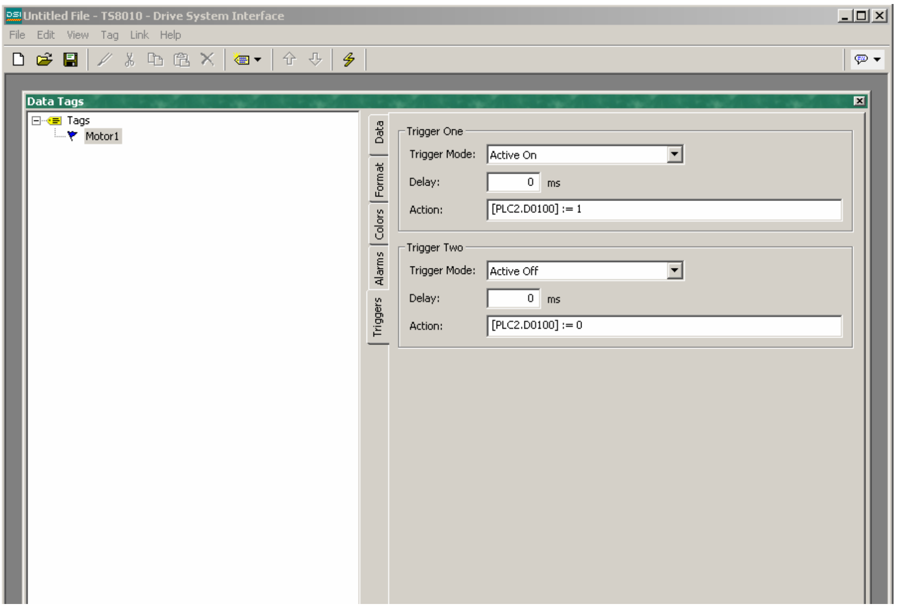
- The *Priority* property is used to control the order in which alarms are displayed by DSI's alarm viewer. The lower the numerical value of the priority field, the nearer to the top the alarm will be displayed.

- The *Email* property is used to specify the email address book entry to which a message should be sent when this alarm is activated. Refer to the Advanced Communications chapter for information on configuring email.

- The *Siren* property is used to indicate whether or not the activation of this alarm should also activate the TS8000 panel's internal speaker. While the speaker is active, the panel's display will also flash to better draw attention to the alarm condition.

The Triggers Tab

The Triggers tab of a flag variable or formula contains the following properties:



Trigger One

Trigger Mode: Active On

Delay: 0 ms

Action: [PLC2.D0100] := 1

Trigger Two

Trigger Mode: Active Off

Delay: 0 ms

Action: [PLC2.D0100] := 0

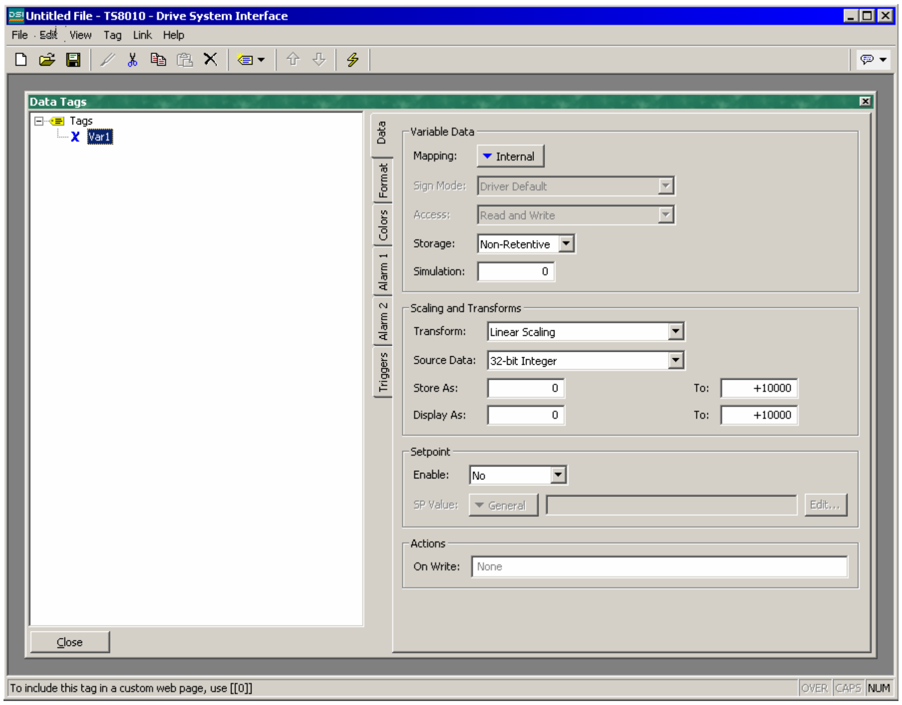
- The *Trigger Mode* property is as described for the Alarms tab.
- The *Delay* property is as described for the Alarms tab.
- The *Action* property is used to indicate what action should be performed when the trigger is activated. Refer to the Writing Actions section for a description of the syntax used to define the various actions that DSI supports.

Editing Integer Tags

You will recall that integer tags represent a 32-bit signed value. The following sections describe the various tabs that are displayed on the right-hand side of the Data Tags window when editing one of the various kinds of integer tags.

The Data Tab (Variables)

The Data tab of an integer variable contains the following properties:



Variable Data

Mapping: Internal

Sign Mode: Driver Default

Access: Read and Write

Storage: Non-Retentive

Simulation: 0

Scaling and Transforms

Transform: Linear Scaling

Source Data: 32-bit Integer

Store As: 0 To: +10000

Display As: 0 To: +10000

Setpoint

Enable: No

SP Value: General Edit...

Actions

On Write: None

To include this tag in a custom web page, use [[0]]

- The *Mapping* property is used to specify if the variable is to be mapped to a register in a remote device, or if it exists only within the terminal. If you press the arrow button and select a device name from the resulting menu, you will be presented with a dialog box which will allow a PLC register to be selected.
- The *Sign Mode* property is used to override the default behavior of the communications driver when reading 16-bit values from a remote device. The driver will normally make a decision about whether to treat these values as signed or unsigned, based upon how the data is normally used within the device. If you want to override this decision, set this property as required.

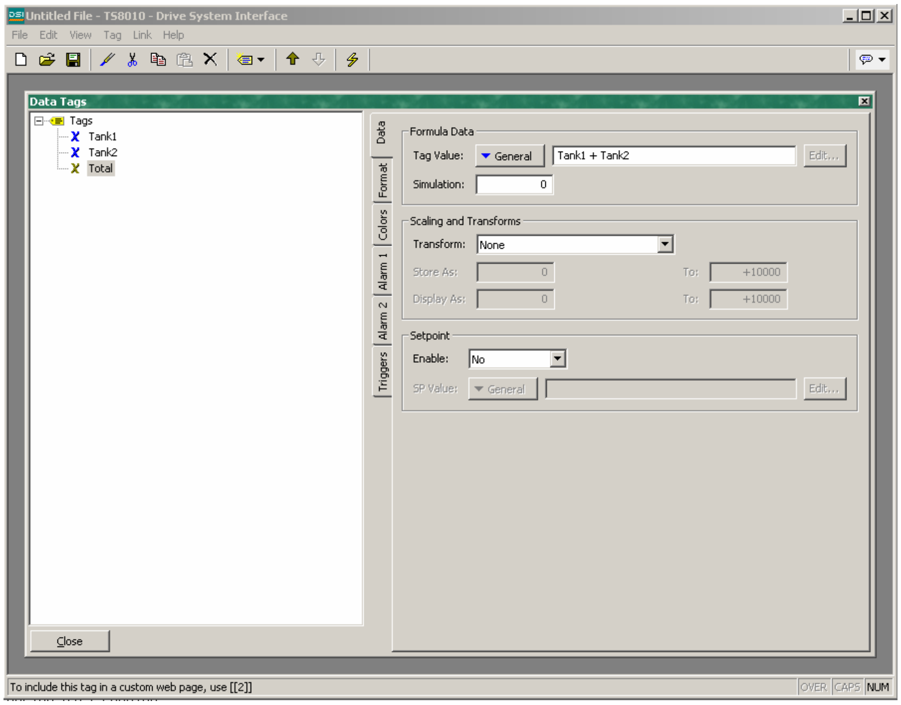
- The *Access* property is as described for flag variables.
- The *Storage* property is as described for flag variables.
- The *Simulation* property is as described for flag variables.
- The *Scaling and Transforms* properties are used to modify the data value as it is read and written from the remote device. When the linear scaling mode is selected, the *Store As* range indicates the upper and lower bounds of the variable within the PLC, while the *Display As* range indicates the corresponding values as they will be presented to the operator. The other modes are as follows:

Mode	Description
BCD to Binary	The BCD value is converted to binary.
Binary to BCD	The binary value is converted to BCD.
Swap Bytes in Word	The lower two bytes of the value are swapped.
Swap Bytes in Long	All four bytes of the value are swapped.
Swap Words	The upper and lower words of the value are swapped.
Reverse Bits in Byte	Bits 0 through 7 of the value are reversed.
Reverse Bits in Word	Bits 0 through 15 of the value are reversed.
Reverse Bits in Long	Bits 0 through 31 of the value are reversed.
Invert Bits in Byte	Bits 0 through 7 of the value are inverted.
Invert Bits in Word	Bits 0 through 15 of the value are inverted.
Invert Bits in Long	Bits 0 through 31 of the value are inverted.

- The *Setpoint* properties are used to indicate whether a setpoint will be specified for this tag, and what that setpoint will be. Setpoints are used by certain alarm modes, and allow the state of a tag to be compared to its intended state. For example, a tag which represents the temperature of a vessel might have a setpoint that indicates the required temperature. This will allow an alarm to activate if the vessel strays beyond a certain distance from its target.
- The *On Write* property is as described for flag variables.

The Data Tab (Formulas)

The Data tab of an integer formula contains the following properties:



Formula Data

Tag Value: General Tank1 + Tank2

Simulation: 0

Scaling and Transforms

Transform: None

Store As: 0 To: +10000

Display As: 0 To: +10000

Setpoint

Enable: No

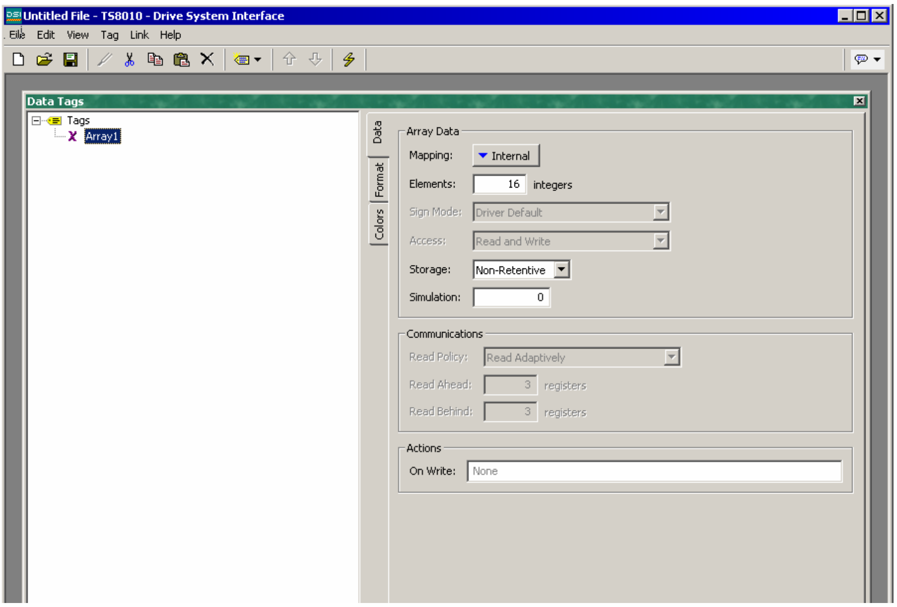
SP Value: General

To include this tag in a custom web page, use [[2]]

- The *Tag Value* property is used to specify the value represented by this tag. It is typically set to a combination of other tags, linked together using math operators. In the example above, the tag is set to be equal to the sum of two tank levels, therefore indicating the total amount of feedstock available.
- The *Scaling and Transforms* properties are as described for integer variables.
- The *Setpoint* properties are as described for integer variables.

The Data Tab (Arrays)

The Data tab of an integer array contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Array1
Data
Format
Colors

Array Data

Mapping: Internal

Elements: 16 integers

Sign Mode: Driver Default

Access: Read and Write

Storage: Non-Retentive

Simulation: 0

Communications

Read Policy: Read Adaptively

Read Ahead: 3 registers

Read Behind: 3 registers

Actions

On Write: None

- The *Mapping* property is used to specify if the variable is to be mapped to a register in a remote device, or if it exists only within the terminal. If you press the arrow button and select a device name from the resulting menu, you will be presented with a dialog box which will allow a PLC register to be selected.

- The *Elements* property is used to indicate how many data items the array should hold. Array elements are referred to using square brackets, such that Array[0] is the first element, and Array[n-1] is the last element, where n is equal to the value entered for this property.

- The *Sign Mode* property is used to override the default behavior of the communications driver when reading 16-bit values from a remote device. The driver will normally make a decision about whether to treat these values as signed or unsigned, based upon how the data is normally used within the device. If you want to override this decision, set this property as required.

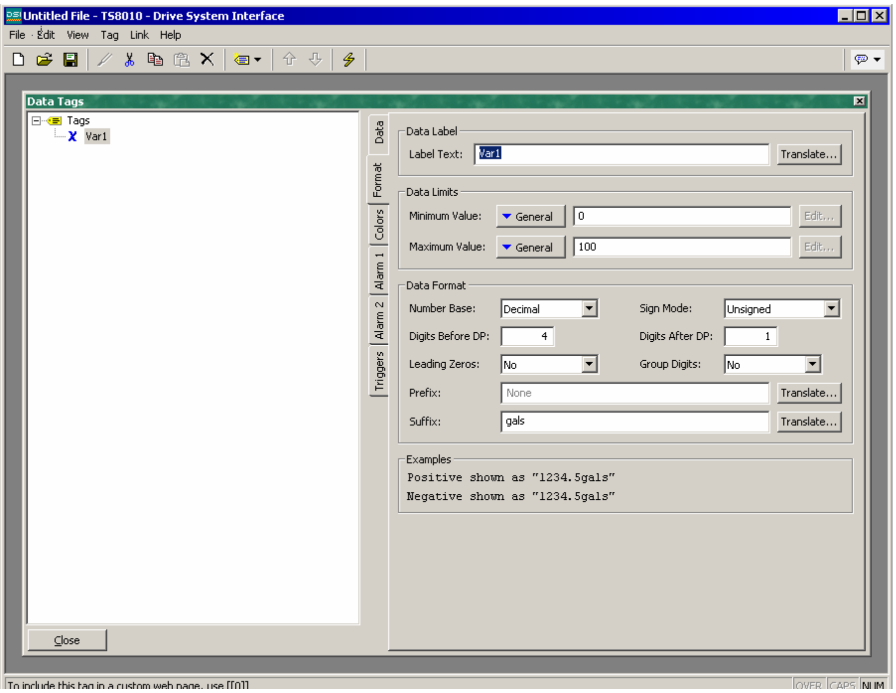
- The *Access* property is as described for flag variables.

- The *Storage* property is as described for flag variables.

- The *Communications Read Policy* is used to indicate which bits in a register are to be read and entered into the array. *Read Whole Array* reads the entire register and enters the information into the array. *Read Manually* reads only the specified bits from the register and enters them into the array. *Read Adaptively* reads the specified bit plus the number of surrounding bits indicated into the array.

The Format Tab

The Format tab of an integer tag contains the following properties:



Data Label

Label Text: Var1

Data Limits

Minimum Value: General 0

Maximum Value: General 100

Data Format

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 4

Digits After DP: 1

Leading Zeros: No

Group Digits: No

Prefix: None

Suffix: gals

Examples

```
Positive shown as "1234.5gals"
Negative shown as "1234.5gals"
```

To include this tag in a custom web page, use [[0]]

- The *Label Text* property is used to specify the label that can be shown next to this tag when including the tag on a display page. The label differs from the tag name, in that the former can be translated for international applications, while the latter remains unchanged and is never shown to the user of the panel.

- The *Minimum Value* and *Maximum Value* properties are used to define the limits used for data entry, and to provide similar limits for the various graphical objects which need to know the bounds within which the tag may vary, such as when scaling a tag's value for display as a bar-graph.

- The *Number Base* property is used to indicate whether the tag should be displayed in decimal, hexadecimal, binary, octal, or passcode. Decimal values may be signed or unsigned, while all other number bases imply unsigned operation. Passcode will display asterisks for values being entered and is intended for security purposes.

- The *Sign Mode* property is used to indicate whether or not a sign should be prefixed to the tag's value. If a hard sign is selected, either a positive or a negative sign will be prefixed as appropriate. If a soft sign is selected, a positive sign will not be shown, but a space will be prefixed instead.

- The *Digits Before DP* property is used to indicate how many digits should be shown before the decimal place, or, if no digits are to be shown after the decimal place, to indicate how many digits should be shown in total.

- The *Digits After DP* property is used to indicate how many digits should be shown after the decimal place. Decimal places are not supported if a number base other than decimal has been selected.

- The *Leading Zeroes* property is used to indicate whether zeros at the beginning of a number should be shown, or replaced with spaces.

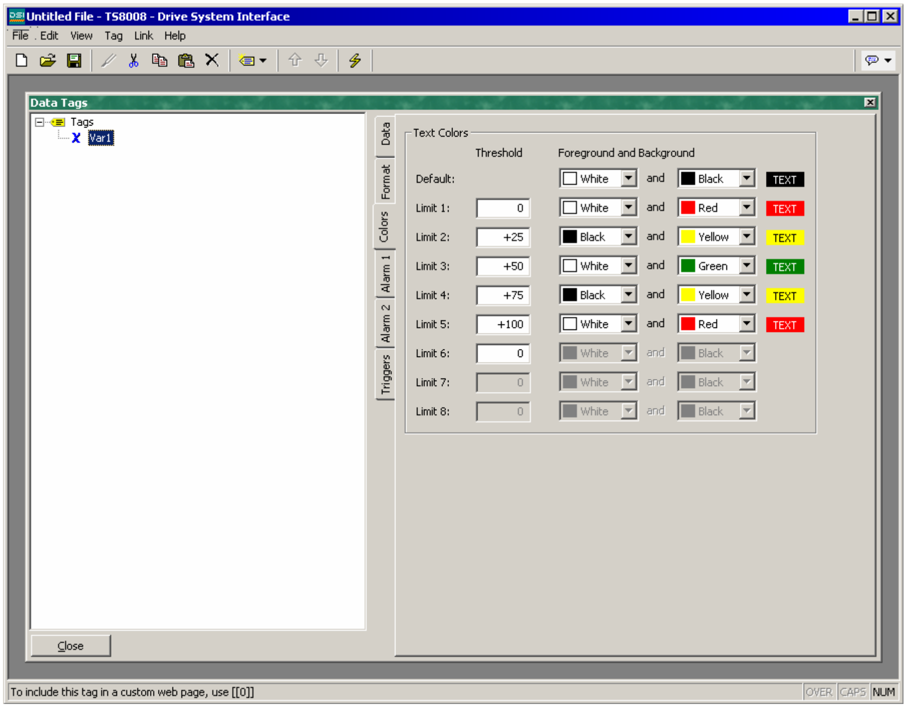
- The *Group Digits* property is used to indicate whether decimal values should have the digits before the decimal place grouped in threes, and separated with commas. Similar separation is performed on other number bases, using groupings and separators appropriate to the selected radix.

- The *Prefix* property is used to specify a translatable string that will be displayed in front of the numeric value. This is typically used to indicate units of measure.

- The *Suffix* property is used to specify a translatable string that will be displayed after the numeric value. This is also typically used to indicate units of measure.

The Colors Tab

The Colors tab of an integer contains the following properties:


Untitled File - TS8008 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Var1
Data
Format
Colors
Alarm 1
Alarm 2
Triggers
Close
OVER CAPS NUM

Text Colors

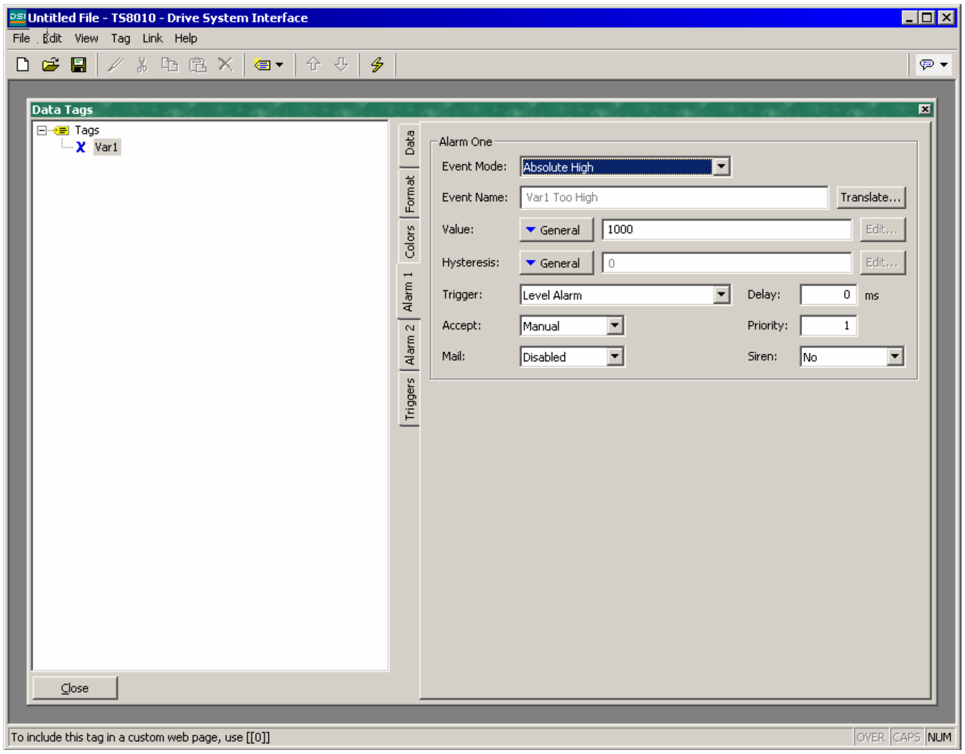
	Threshold	Foreground and Background			
Default:		White	and	Black	TEXT
Limit 1:	0	White	and	Red	TEXT
Limit 2:	+25	Black	and	Yellow	TEXT
Limit 3:	+50	White	and	Green	TEXT
Limit 4:	+75	Black	and	Yellow	TEXT
Limit 5:	+100	White	and	Red	TEXT
Limit 6:	0	White	and	Black	
Limit 7:	0	White	and	Black	
Limit 8:	0	White	and	Black	

To include this tag in a custom web page, use [[0]]

- The *Default* property is used to define the color pair that will be used to display the tag when its value is less than the *Limit 1* property.
- The remaining properties define limits, and color pairs, that will be used to display the tag when its value is greater than the corresponding limit, and less than the next limit. If the next limit is zero, the color pair will be used whenever the tag's value exceeds the specified limit.

The Alarm Tabs

Each Alarm tab of an integer variable or formula contains the following properties:



Event Mode: Absolute High
Event Name: Var1 Too High
Value: General 1000
Hysteresis: General 0
Trigger: Level Alarm
Delay: 0 ms
Accept: Manual
Priority: 1
Mail: Disabled
Siren: No

To include this tag in a custom web page, use [[0]]

- The *Event Mode* property is used to indicate the logic that will be used to decide whether the alarm should activate. The tables below list the available modes.

Mode	Alarm will activate when:
Data Match	The value of the tag is equal to the alarm's *Value*.
Data Mismatch	The value of tag is not equal to the alarm's *Value*.
Absolute High	The value of the tag exceeds the alarm's *Value*.
Absolute Low	The value of the tag falls below the alarm's *Value*.

The following modes are only available when a setpoint is defined.

Mode	Alarm will activate when:
Deviation High	The value of the tag exceeds the tag's *Setpoint* by an amount equal to or greater than the alarm's *Value*.
Deviation Low	The value of the tag falls below the tag's *Setpoint* by an amount equal to or greater than the alarm's *Value*.
Out of Band	The tag moves outside a band equal in width to twice the alarm's *Value* and centered on the tag's *Setpoint*.
In Band	The tag moves inside a band equal in width to twice the alarm's *Value* and centered on the tag's *Setpoint*.

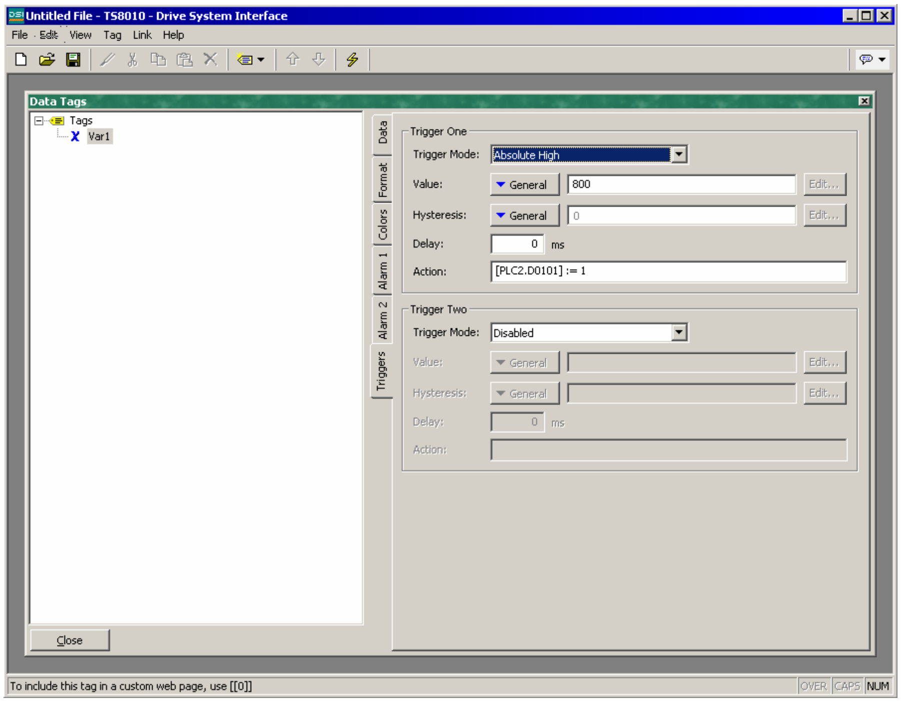
- The *Value* property is used to define either the absolute value at which the alarm will be activated, or the deviation from the setpoint value. The exact interpretation depends on the event mode as described above.

- The *Hysteresis* property is used to prevent an alarm from oscillating between the on and off states when the process is near the alarm condition. For example, for an absolute high alarm, the alarm will become active when the tag exceeds the alarm's value, but will only deactivate when the tag falls below the value by an amount greater than or equal to the alarm's hysteresis. Remember that the property always acts to maintain an alarm once the alarm is activated, and not to modify the point at which the activation occurs.

- The remainder of the properties are as described for the Alarms tab of flag tags.

The Triggers Tab

The Triggers tab of an integer variable or formula contains the following properties:



Untitled File - TS8010 - Drive System Interface

Data Tags

Tags
Var1

Trigger One
Trigger Mode: Absolute High
Value: General 800
Hysteresis: General 0
Delay: 0 ms
Action: [PLC2.D0101] := 1

Trigger Two
Trigger Mode: Disabled
Value: General
Hysteresis: General
Delay: 0 ms
Action:

To include this tag in a custom web page, use [[0]]

- The *Trigger Mode* property is as described for the Alarm tabs.

- The *Delay* property is as described for a flag tag's Alarms tab.

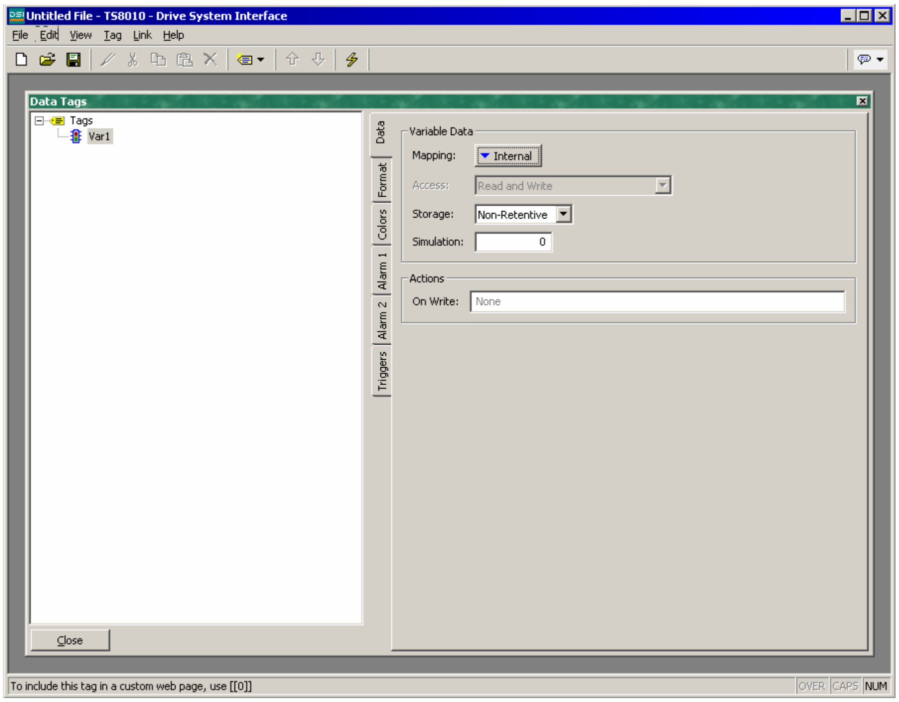
- The *Action* property is used to indicate what action should be performed when the trigger is activated. Refer to the Writing Actions section for a description of the syntax used to define the various actions that DSI supports.

Editing Multi Tags

You will recall that multi tags represent a 32-bit signed value, but are used to select between one of a number of text strings. The following sections describe the various tabs that are displayed on the right-hand side of the Data Tags window when editing a multi tag.

The Data Tab (Variables)

The Data tab of a multi variable contains the following properties:



Variable Data

Mapping: Internal

Access: Read and Write

Storage: Non-Retentive

Simulation: 0

Actions

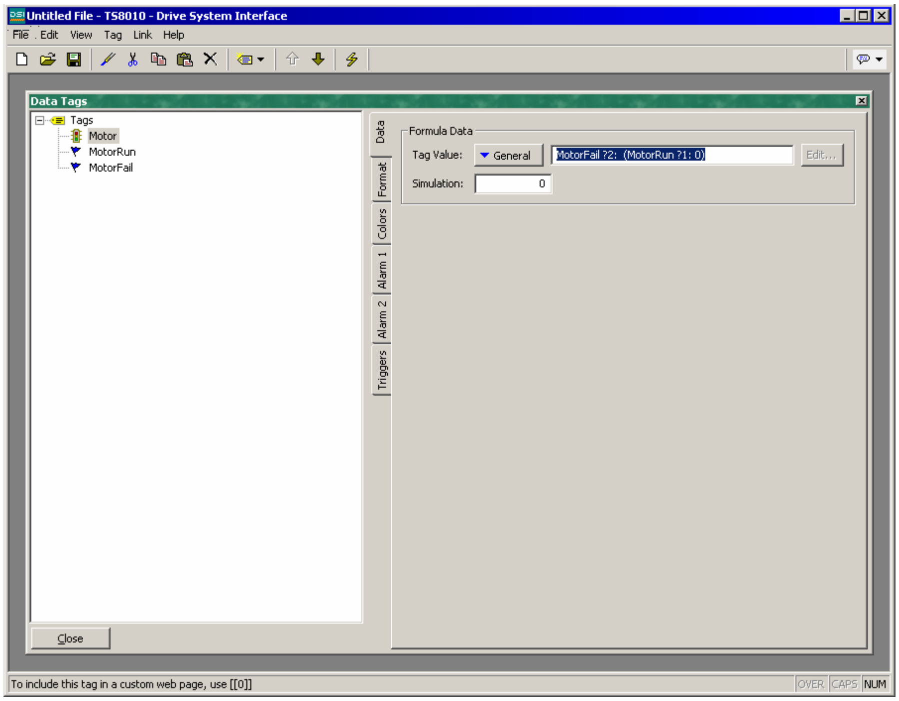
On Write: None

To include this tag in a custom web page, use [[0]]

- The *Mapping* property is used to specify if the variable is to be mapped to a register in a remote device, or if it exists only within the terminal. If you press the arrow button and select a device name from the resulting menu, you will be presented with a dialog box which will allow a PLC register to be selected.
- The remainder of the properties are as described for flag variables.

The Data Tab (Formulas)

The Data tab of a multi formula contains the following properties:



Formula Data

Tag Value: General MotorFail ?2: (MotorRun ?1: 0)

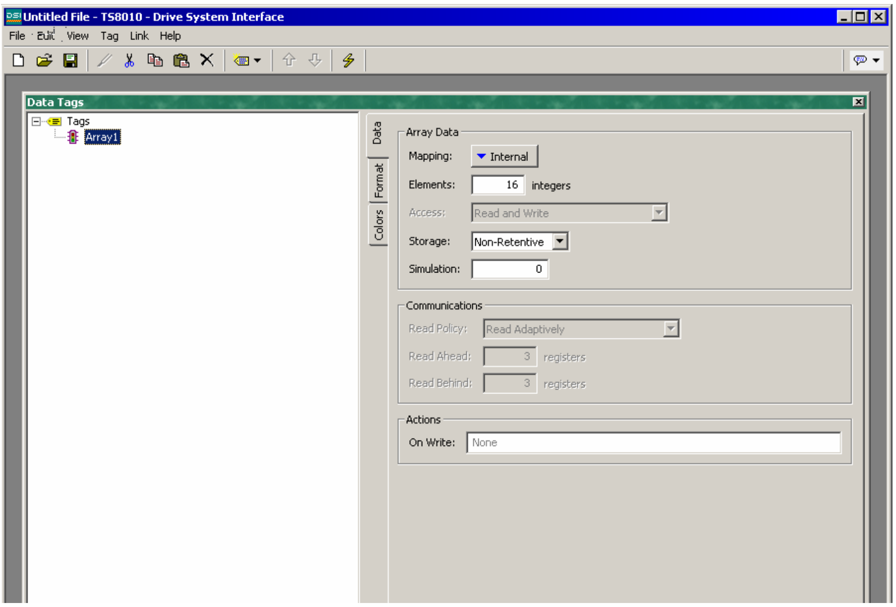
Simulation: 0

To include this tag in a custom web page, use [[0]]

- The *Tag Value* property is used to specify the value represented by this tag. It is typically set to a combination of other tags, linked together using math operators. In the example above, the tag is set equal to a value of one, two or three, depending on the state of three different flags. For more information on the **?:** operator used in this example, refer to the Writing Expressions section.
- The *Simulation* property is as described for flag tags.

The Data Tab (Arrays)

The Data tab of a multi array contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Array1
Data
Format
Colors

Array Data

Mapping: Internal

Elements: 16 integers

Access: Read and Write

Storage: Non-Retentive

Simulation: 0

Communications

Read Policy: Read Adaptively

Read Ahead: 3 registers

Read Behind: 3 registers

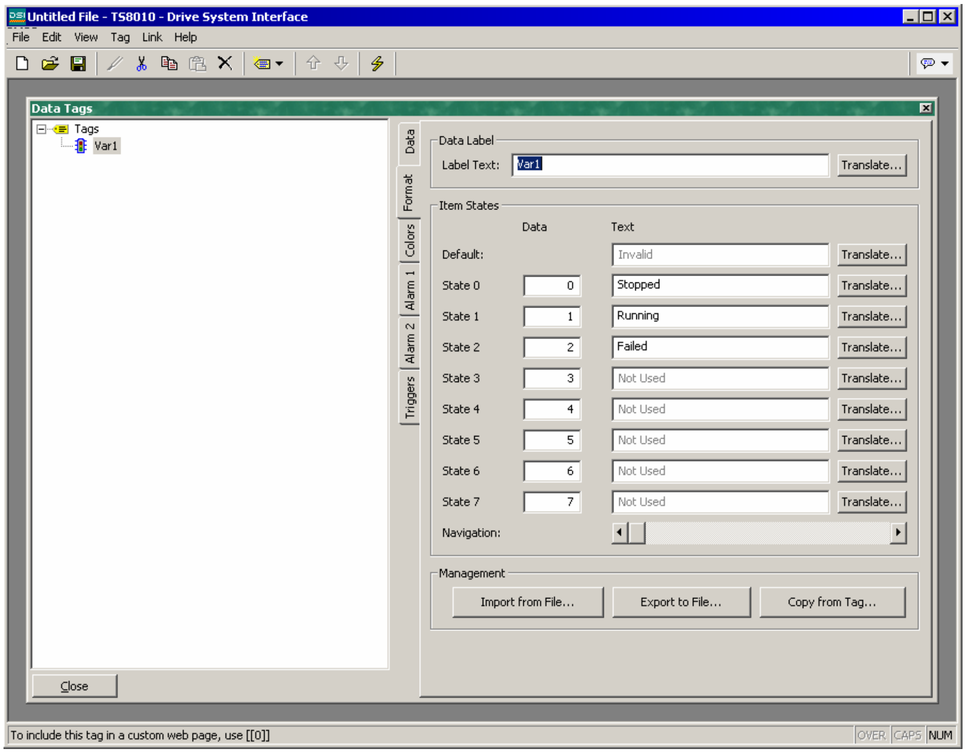
Actions

On Write: None

All of these properties are as described for flag arrays.

The Format Tab

The Format tab of a multi tag contains the following properties:



Untitled File - TS8010 - Drive System Interface

Data Tags

Data Label

Label Text: Var1

Item States

	Data	Text
Default:		Invalid
State 0	0	Stopped
State 1	1	Running
State 2	2	Failed
State 3	3	Not Used
State 4	4	Not Used
State 5	5	Not Used
State 6	6	Not Used
State 7	7	Not Used

To include this tag in a custom web page, use [[0]]

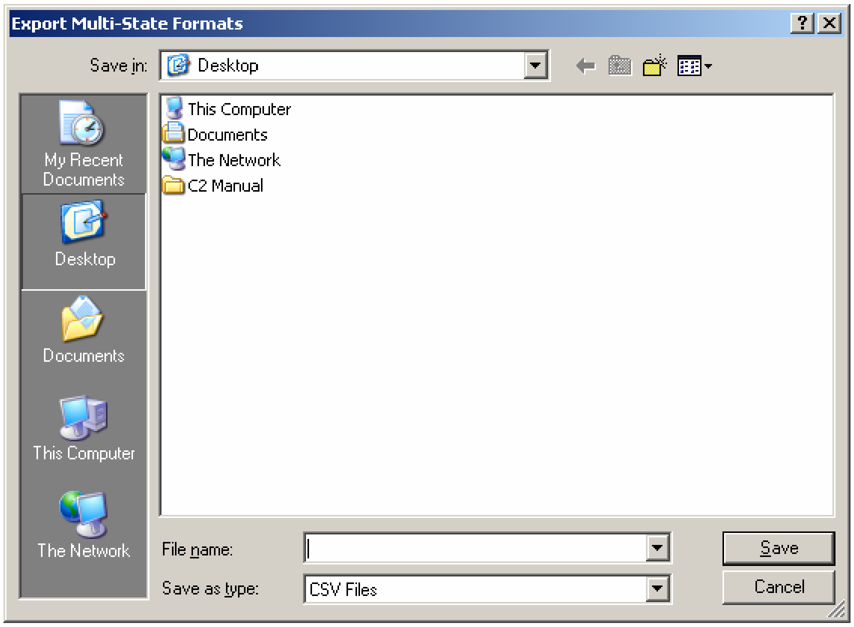
- The *Label Text* property is used to specify the label that can be shown next to this tag when including the tag on a display page. The label differs from the tag name, in that the former can be translated for international applications, while the latter remains unchanged and is never shown to the user of the panel.

- The *Item States* properties are used to define up to eight values which represent different states of the tag. Each state has an integer value associated with it, and a text string to indicate what should be displayed when the tag holds that value. At least two states must be defined, but the balance may be left in their default condition if they are not needed.

- The *Default* property is used to define the text to be displayed if the tag holds a value other than one of those listed in the item states.

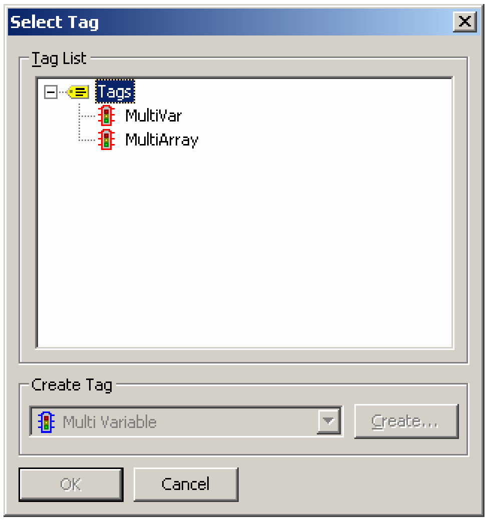
- The *Navigation* slider is used to step through the 512 states that can be defined for a particular tag. Moving the slider left and right will update the right-hand pane to show the selected states.

The *Export To File* button can be used to export state names and values to a CSV (Comma Separated Variable) file.



Save in: Desktop

Save as type: CSV Files



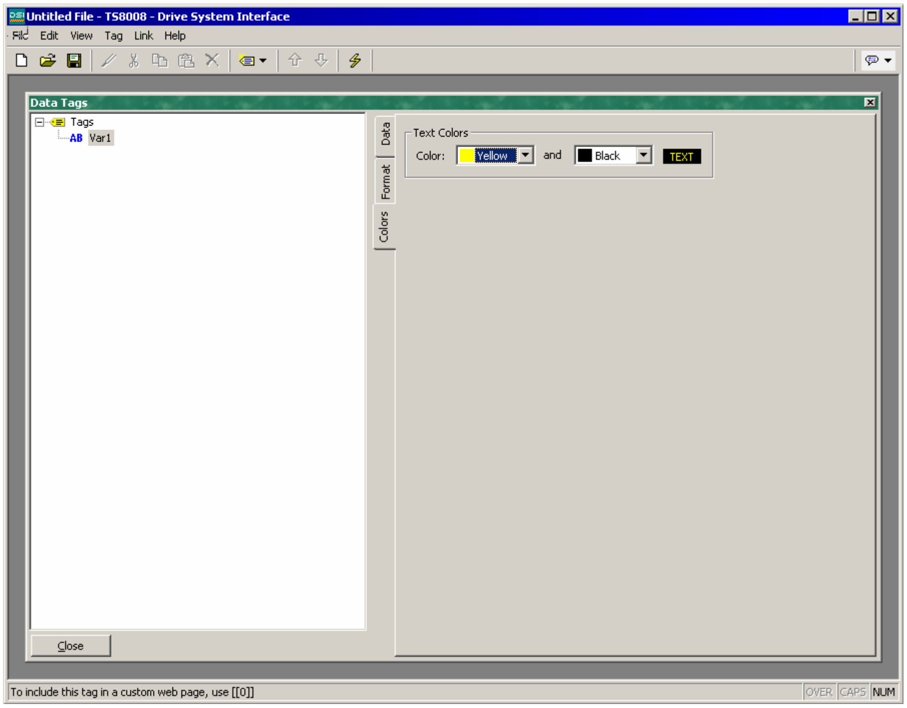
Create Tag: Multi Variable

The CSV file will contain a line for each defined state, stating the state label, the state value, and the text assigned to that state. If multiple languages are in use, an additional column will be provided for each language. The file type drop-down can be used to select a Unicode format file if you are using languages that cannot be represented in standard ASCII. The file can be subsequently re imported using the Import from File button.

The *Copy from Tag* button will display the shown dialog box. This dialog can be used to select another multi tag from which the format information is to be copied. This facility will save a lot of typing if the same format is to be used on several tags.

The Colors Tab

The Colors tab of a multi tag contains the following properties:


Untitled File - TS8008 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
Var1
Data
Format
Colors
Close
OVER CAPS NUM

Text Colors

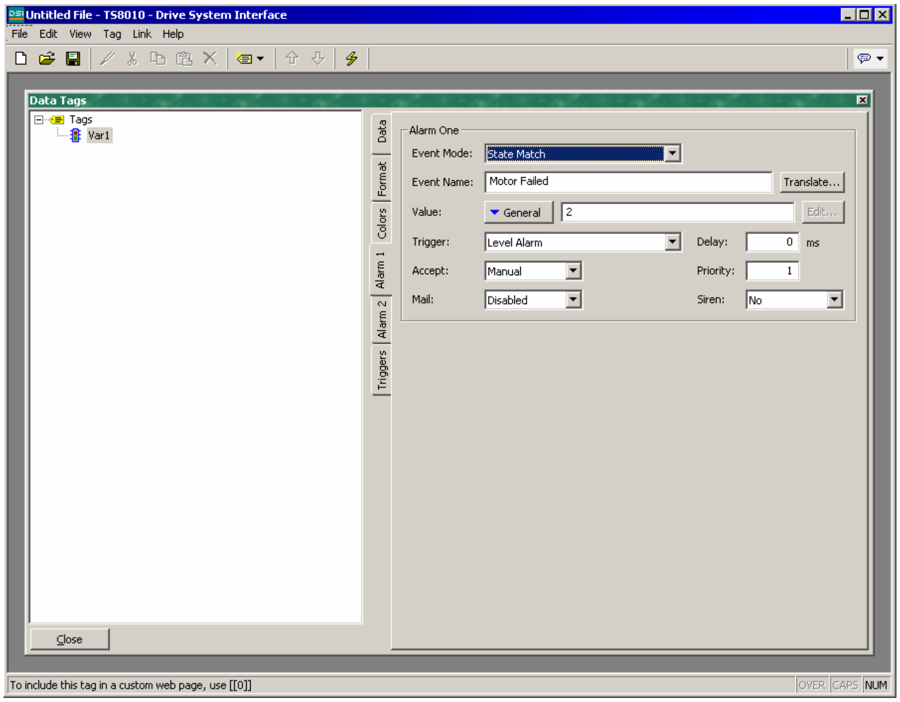
Color: Yellow and Black TEXT

To include this tag in a custom web page, use [[0]]

- The various color pairs are used to specify how the tag should be displayed when it is each of the states specified on the Format tab. As with the Format tab, the *Navigation* slider can be used to up and down the list of color pairs when more than eight states have been defined.

The Alarm Tab

Each Alarm tab of a multi variable or formula contains the following properties:



Alarm One

Event Mode: State Match

Event Name: Motor Failed

Value: General 2

Trigger: Level Alarm

Delay: 0 ms

Accept: Manual

Priority: 1

Mail: Disabled

Siren: No

To include this tag in a custom web page, use [[0]]

- The *Event Mode* property is used to indicate the logic that will be used to decide whether the alarm should activate. The table below lists the available modes.

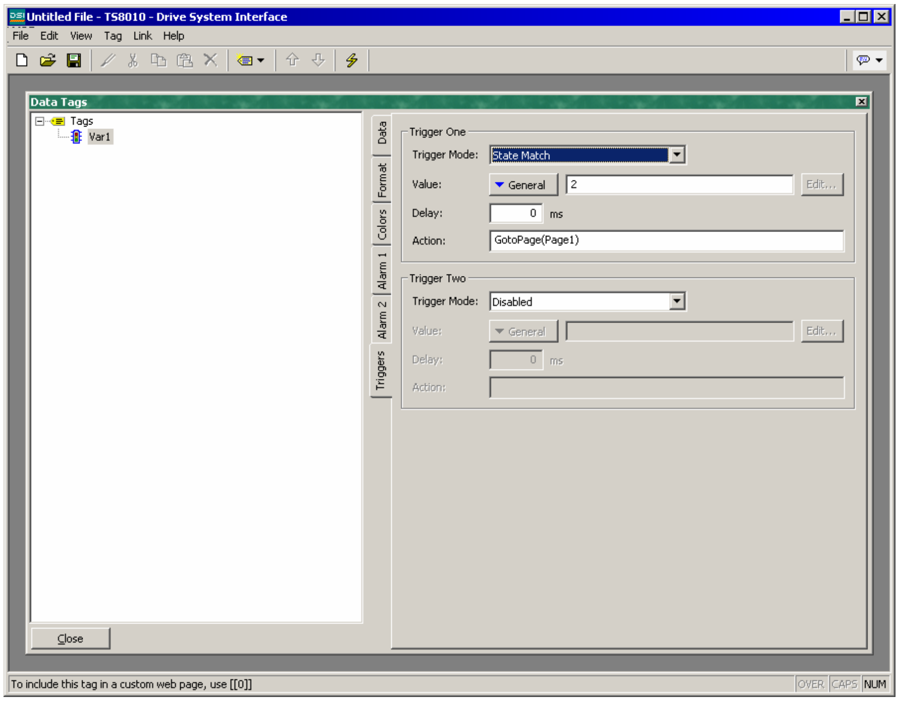
Mode	Alarm will activate when:
State Match	The value of the tag is equal to the alarm's *Value*.
State Mismatch	The value of tag is not equal to the alarm's *Value*.

- The *Value* property is used to define the comparison data for the alarm.

- The remainder of the properties are as described for the Alarms tab of flag tags.

The Triggers Tab

The Triggers tab of a multi variable or formula contains the following properties:



Untitled File - TS8010 - Drive System Interface

Data Tags

Trigger One

Trigger Mode: State Match

Value: General 2

Delay: 0 ms

Action: GotoPage(Page1)

Trigger Two

Trigger Mode: Disabled

Value: General

Delay: 0 ms

Action:

To include this tag in a custom web page, use [[0]]

- The *Trigger Mode* property is as described for the Alarm tabs.

- The *Delay* property is as described for a flag tag's Alarms tab.

- The *Action* property is used to indicate what action should be performed when the trigger is activated. Refer to the Writing Actions section for a description of the syntax used to define the various actions that DSI supports.

Editing Real Tags

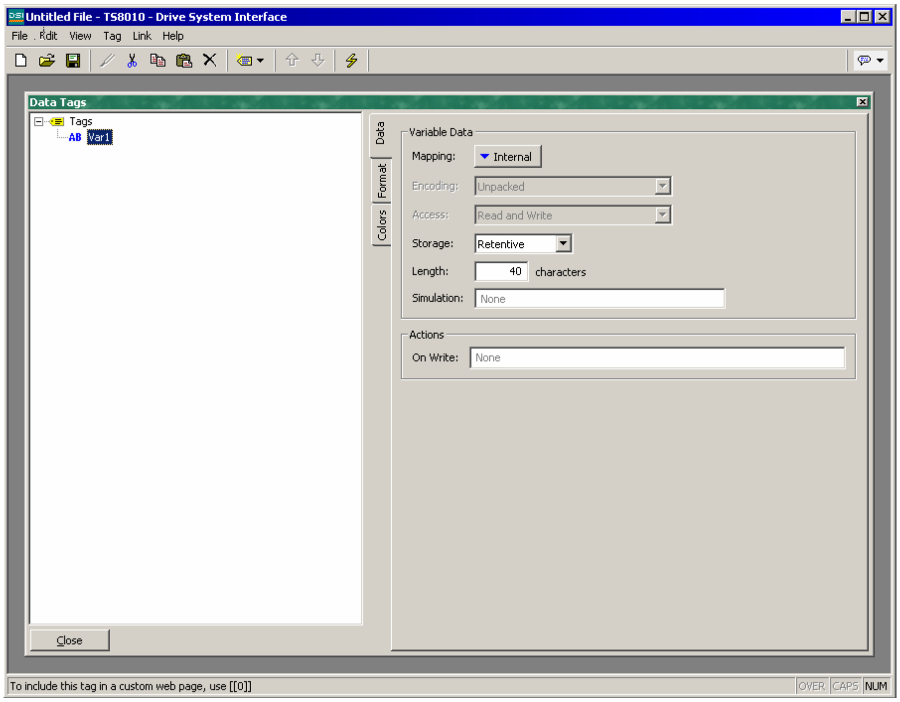
You will recall that real tags represent a single-precision floating-point value. All the tabs displayed for real tags are exactly the same as those displayed for integer tags, with the exception that data entered for items such as the value and hysteresis properties of alarms and triggers may contain decimals. You are thus referred to the sections on integer tags. You will notice some selections for integer tags that are not applicable to real tags.

Editing String Tags

You will recall that string tags represent an item of text, this being made up of a number of individual characters. The following sections describe the various tabs that are displayed on the right-hand side of the Data Tags window when editing one of the various string tags.

The Data Tab (Variables)

The Data tab of a string variable contains the following properties...



Variable Data

Mapping: Internal

Encoding: Unpacked

Access: Read and Write

Storage: Retentive

Length: 40 characters

Simulation: None

Actions

On Write: None

To include this tag in a custom web page, use [[0]]

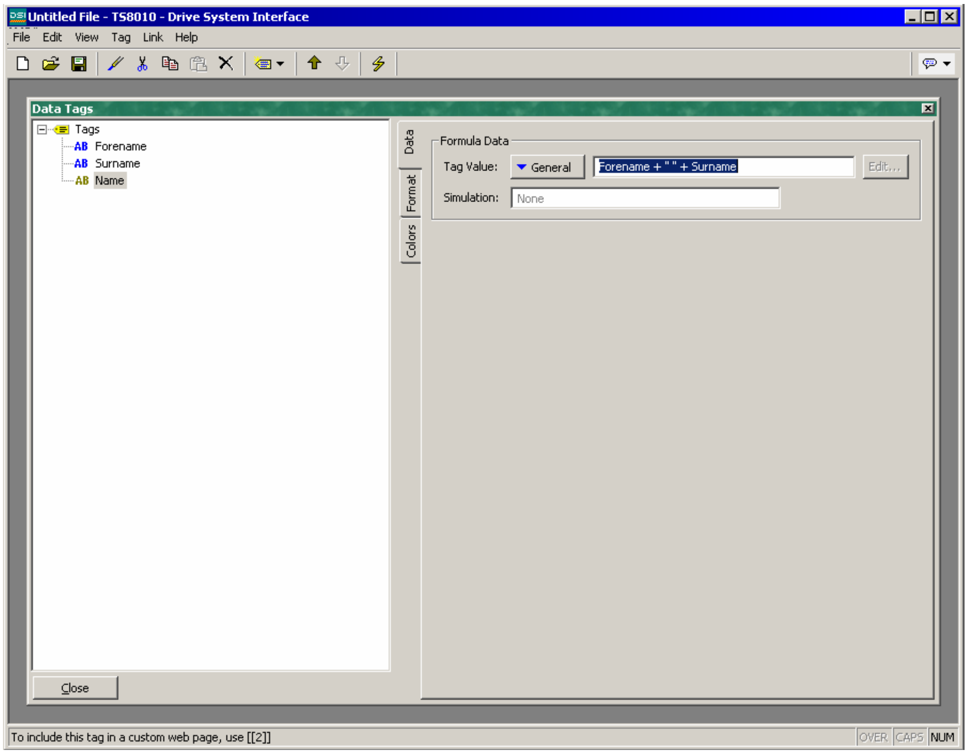
- The *Mapping* property is used to specify if the variable is to be mapped to a register in a remote device, or if it exists only within the terminal. If you press the arrow button and select a device name from the resulting menu, you will be presented with a dialog box that will allow a PLC register to be selected.
- The *Encoding* property is used to specify how text will be packed into mapped registers that contain more than 8 bits of data. Selecting unpacked will store one character per register no matter how large the register, leaving the high order bits empty. Selecting low-to-high packed mode will store one character

in each 8 bits of the target register, storing the first character in the lowest order bits. Selecting high-to-low packed mode will store one character in each 8 bits of the target register, storing the first character in the highest order bits

- The *Length* property is used to indicate how many characters of storage should be allocated for this string. A value need only be entered if you have configured the variable for retentive storage. Strings that are kept in the terminal's RAM and not committed to FLASH have no practical limit on their length.
- The remainder of the properties are as described for flag variables.

The Data Tab (Formulas)

The Data tab of a string formula contains the following properties:



Tag Value: General Forename + " " + Surname

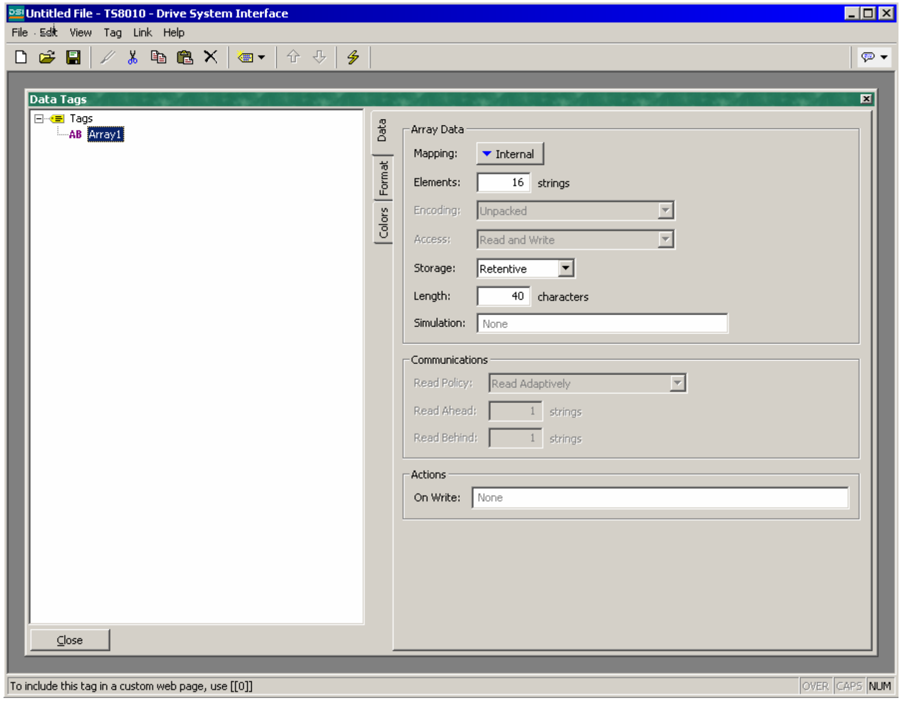
Simulation: None

To include this tag in a custom web page, use [[2]]

- The *Tag Value* property is used to specify the value represented by this tag. It is typically set to a combination of other tags, linked together using math operators or functions. In the example above, the tag is set equal to the combination of two strings variables, separated by a space. For more information on the operators and functions that can be used with strings, refer to the Writing Expressions section and the Function Reference at the end of this document.
- The *Simulation* property is as described for flag variables.

The Data Tab (Arrays)

The Data tab of a string array contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
AB Array1
Data
Format
Colors
Close
OVER CAPS NUM

Array Data

Mapping: Internal

Elements: 16 strings

Encoding: Unpacked

Access: Read and Write

Storage: Retentive

Length: 40 characters

Simulation: None

Communications

Read Policy: Read Adaptively

Read Ahead: 1 strings

Read Behind: 1 strings

Actions

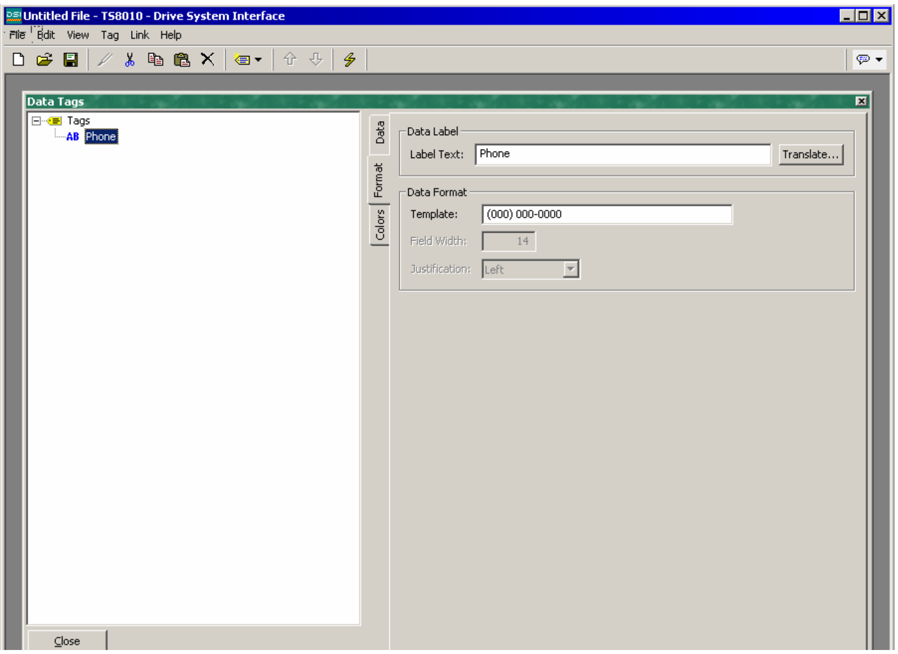
On Write: None

To include this tag in a custom web page, use [[0]]

- The *Length* and *Encoding* properties are as described for string variables.
- The remainder of the properties are as described for flag arrays.

The Format Tab

The Format tab of a string tag contains the following properties:


Untitled File - TS8010 - Drive System Interface
File Edit View Tag Link Help
Data Tags
Tags
AB Phone
Data
Format
Colors
Translate...
Close

Data Label

Label Text: Phone

Data Format

Template: (000) 000-0000

Field Width: 14

Justification: Left

- The *Label Text* property is used to specify the label that can be shown next to this tag when including the tag on a display page. The label differs from the tag name, in that the former can be translated for international applications, while the latter remains unchanged and is never shown to the user of the panel.

- The *Template* property is used to provide a "picture" of the string, thereby indicating what kind of characters can occur in each position. If a template is specified, data entry will be limited such that only the correct kind of character can be selected for each character in the string. The table below shows the meaning of the various special characters that can be included in a template:

Character	Permitted Characters				
In Template	A-Z	a-z	0-9	Space	Misc
A	Yes	-	-	-	-
a	Yes	Yes	-	-	-
S	Yes	-	-	Yes	-
s	Yes	Yes	-	Yes	-
N	Yes	-	Yes	-	-
n	Yes	Yes	Yes	-	-
M	Yes	-	Yes	Yes	-
m	Yes	Yes	Yes	Yes	-
0	-	-	Yes	-	-
X	Yes	Yes	Yes	Yes	Yes

The additional characters referred to by the "Misc" column are:

```
,.:;+-=!?%/$
```

Characters not included in the table are copied verbatim to the display.

For example, to allow entry of a US telephone number, use a template of:

(000) 000-0000

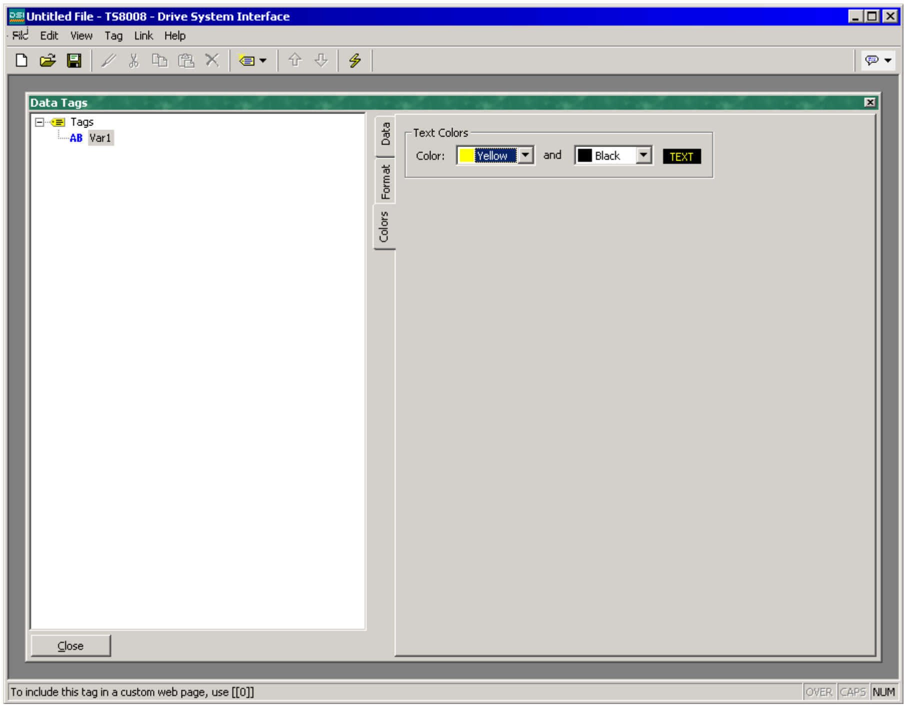
The parentheses, the space and the dash will all be included when the field is displayed, but only the 10 digits indicated by the '0' characters will be stored in the string. Similarly, if data entry is enabled for a field using this template, the cursor will skip the various non-numeric positions when moving left or right, and will only allow numeric characters to be entered for those positions which can be selected.

- The *Length* property is used in lieu of the template to indicate how many characters should be reserved on a page when displaying this string. If a string variable is marked as retentive, it makes sense for this property to be equal to the length entered on the *Data* tab, but this is not obligatory, as you may want to allocate more or less space on the display for layout purposes.

- The *Justification* property is used when a template is not specified, and indicates how strings shorter than the Length property should be positioned within the storage allocated for the string. It is distinct from the Justification property of the display format, in that it impacts the data that is actually stored.

The Colors Tab

The Colors tab of a string tag contains the following properties:



Text Colors

Color: Yellow and Black TEXT

To include this tag in a custom web page, use [[0]]

The tab is used to specify the default colors to be used to display this tag.

More Than Two Alarms

If your application requires more than two alarms (or indeed triggers) for a tag, define a formula to be equal in value to the primary tag, and set the extra alarms on the alias. For example, if you have a variable called Level which is mapped to N7:100 in a PLC, and you need to create a third alarm for that tag, create a variable called, say, LevelAlias and set its value property to Level. You can then set additional alarms on this alias tag.

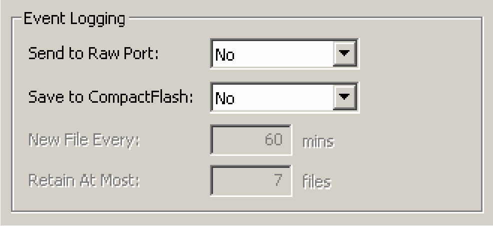
Validating Tags

Selecting the Tags icon in the left-hand pane of the Tags window will allow access to the Validate All Tags button. Pressing this button will recompile all expressions in your database, fixing any broken communications references and updating tag reference counts. You should not need to push this button unless you have removed and then replaced tags, and wish to repair the expression that will have been broken when the tags were deleted.

Exporting Tag Mappings

Selecting the Tags icon in the left-hand pane of the Tags window will also allow access to the tag import and export facilities. The Export to File button can be used to export the tag names and mappings to a CSV file for subsequent editing in Microsoft Excel or some other suitable tool. The Import from File button can then be used to re-import the file, changing the tag mappings in line with the changes made to the file. These facilities are useful when porting an application from one PLC to another, as it allows all the mappings to be changed in a single operation.

Logging Event Messages



Event Logging

Send to Raw Port: No

Save to CompactFlash: No

New File Every: 60 mins

Retain At Most: 7 files

When the Tags icon is selected in the left-hand pane, the right-hand pane of the Tags window contains options to control the logging of the messages generated by the alarms and events attached to each tag.

- The *Send to Raw Port* property is used to indicate which communications port events should be printed to. The port in question must have a raw port driver bound to it as described in the Using Raw Ports chapter. Note that a serial driver or a TCP/IP driver may be used as required by the application.

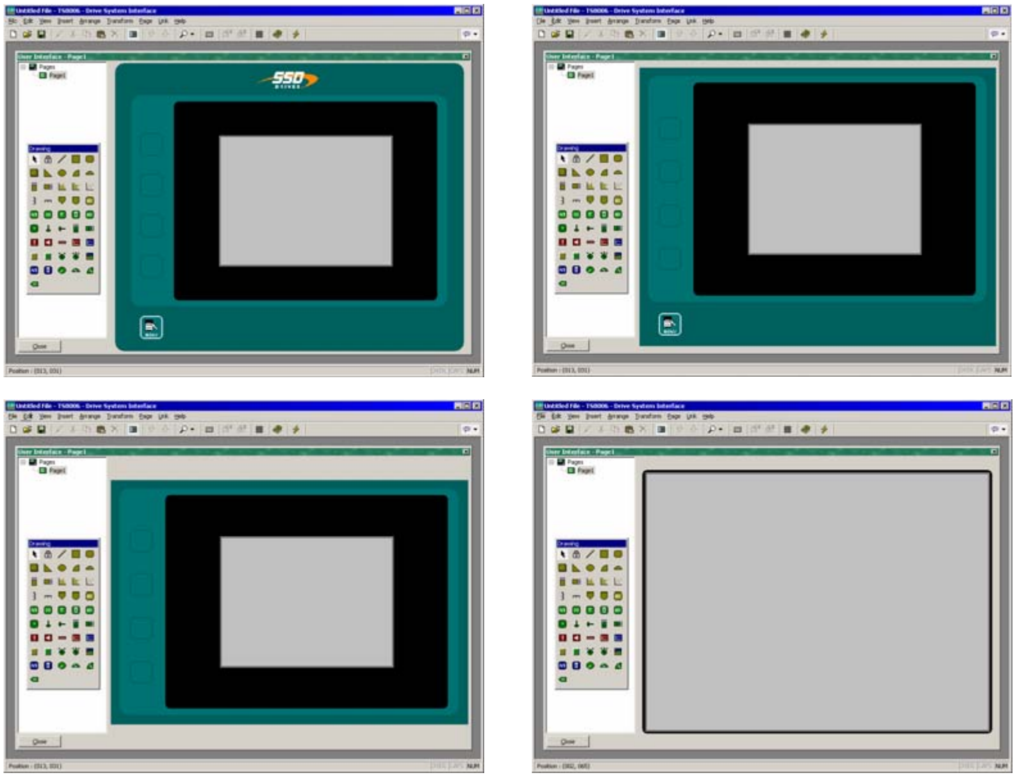
- The *Save to CompactFlash* property is used to enable the writing of events to CSV files on the card fitted to the panel. Events are stored using techniques similar to those for data logging. The *New File Every* and *Retain at Most* properties control how files are allocated. Refer to the Configuring Data Logging chapter for information on how the data is written and how files are named.

Configuring The User Interface (Touchscreens)

Now that you have configured your communications, and created data tags for the items that you wish to display, you can create display pages to allow the user to view or edit these data items. These pages are manipulated by selecting the User Interface icon from the main screen. Please note that this chapter refers specifically to the color touchscreen operator panels (TS8006, TS8008 and TS8010). If you are using a TS8003 operator panel with a monochrome display, please refer to the next chapter for configuration details.

Controlling The View

By default, the User Interface window shows the entire front panel of the TS8000, including the display and all the available keys. If you want to allocate more of your PC's screen to show the TS8000's display, you can use the four different zoom levels as shown below:



As you can see, at each level, fewer keys are shown, and more of the window is allocated to the display itself. The zoom level can be controlled from the View menu, by using the magnifying glass icon, or by pressing the **Alt** key together with the digits **1** through **4**.

Other View Options

As well as controlling the zoom, the View menu contains the following options:

- The *Page List* command can be used to show or hide the left-hand pane of the User Interface window. If the page list is disabled, even more space is made available for editing the display. The **F4** key toggles the page list on and off.
- The *Hold Aspect* command can be used to control whether or not DSI attempts to maintain the aspect ratio of the display. If aspect holding is enabled, a figure that would appear as, say, a circle on the TS8000 will appear as a perfect circle on your PC. If this mode is not selected, DSI can expand the display page to use more of the PC's screen, but at the expense of some distortion.

Other options are available during page editing, and are described below.

Using The Page List

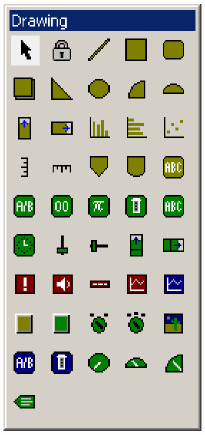
To create, rename or delete display pages, click on the left-hand pane of the User Interface window. The various commands on the Page menu can then be used to make the desired changes. Alternately, right-click on the required display page, and select from the menu.

To select a page, either click on the page in the page list, or use the up and down arrows in the toolbar. Alternatively, you can use the **Alt+Left** and **Alt+Right** key combinations to move up and down the list as required. These keys will work no matter which pane is selected.

Working With The Grid

The Show Grid command on the View menu can be used to show or hide an eight-pixel grid that is useful for aligning objects. Every eighth column of the grid is shown in a brighter color, as is every sixth row. Various drawing operations may be configured so as to "snap" to the grid points whether or not the grid is shown. The three separate actions of creating objects, moving objects and sizing objects may be controlled individually, or the Snap for All or Snap for None commands may be used to control all three actions at once.

The Drawing Toolbox



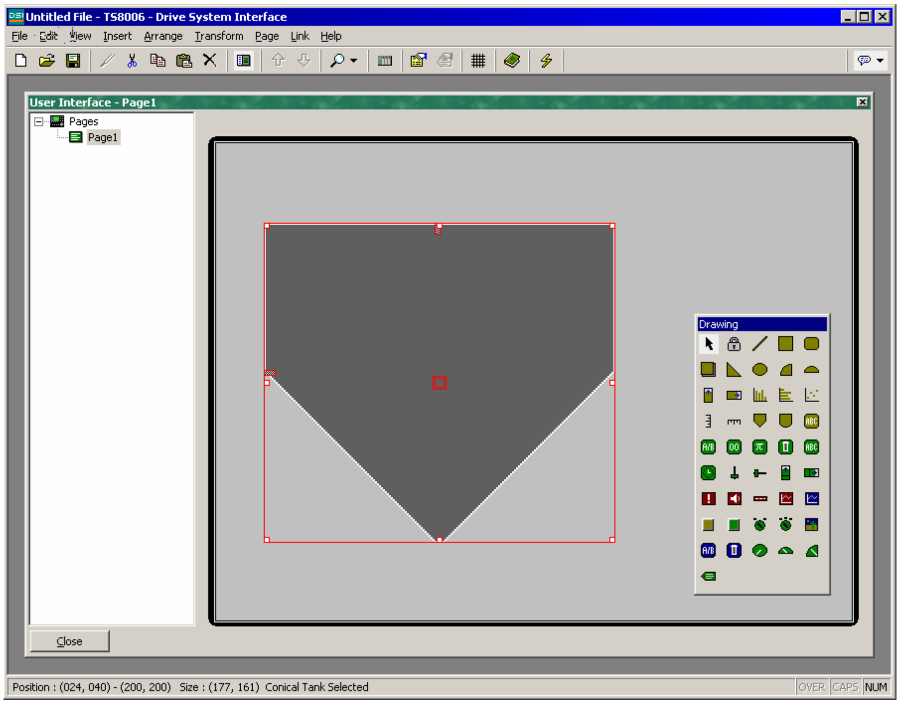
To edit the contents of a display page, first select the page as described above. Then, click on the green rectangle that represents the TS8000's display. A white rectangle will appear around the display to indicate that it has been selected, and a number of toolboxes will appear.

The drawing toolbox is used to add various elements, known as objects, to the display page. The first two icons control the insertion mode, while the balance of the icons represent individual objects. The objects shown in yellow are basic geometric and animation items, while the ones shown in green are rich objects that use formatting and other information from a data tag to control their operation. The objects shown in red are system items, such as the active alarm viewer.

Objects displayed in blue are enhanced version of other objects that have been recently added to the software. All of the commands contained in the toolbox can also be accessed via the Insert menu.

Adding Display Objects

To add a display object to a page, click on the required icon in the drawing toolbox, or select the required option from the Insert menu. The mouse cursor will change to an arrow with a crosshair at its base, and you will then be able to drag-out the required position of the object within the display window.

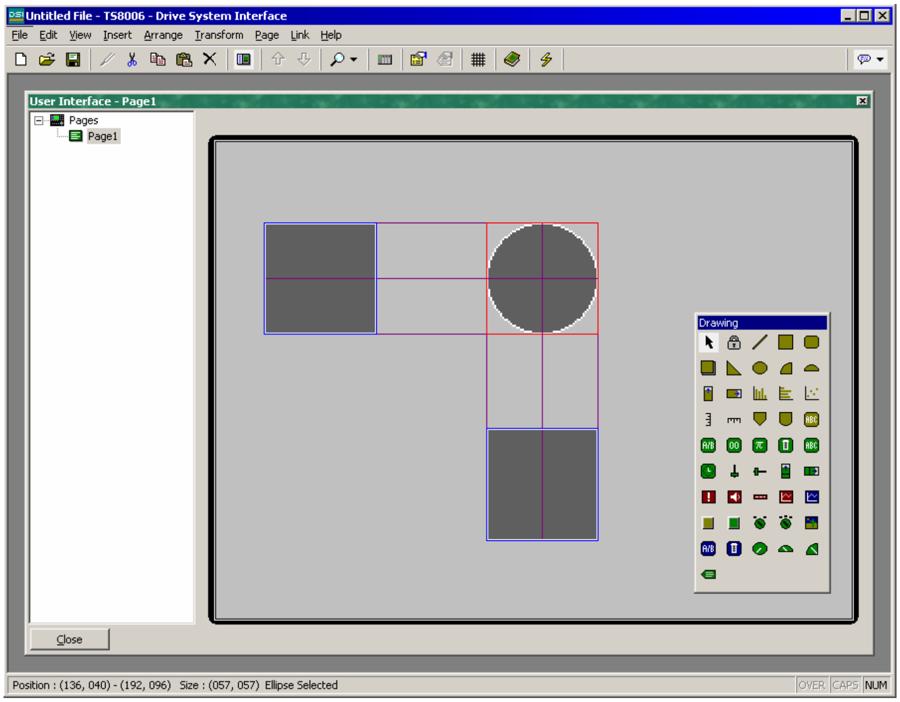


Position : (024, 040) - (200, 200) Size : (177, 161) Conical Tank Selected

Smart Alignment

If you have the Smart Align features of the View menu enabled, DSI will provide you with guidelines to help align a new object with existing objects, or with the center of the display. In the example shown above, the horizontal dotted line indicates that the center of the tank object is vertically aligned with the center of the display. With a little practice, this feature can make it very easy to align objects as they are created, without the need to go back and "tweak" your display pages to get the various figures into alignment.

In the Smart Align example shown below, a newly-created ellipse is being aligned with two existing rectangles. Guidelines are present at both the edges of the figures, and at the center, showing that both the edges and the centers are aligned. The red rectangle is highlighting the newly-created object, while the blue rectangles are highlighting the objects to which the guidelines have been drawn. Smart Align is also enabled when objects are moved or re-sized.


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
Close
OVER CAPS NUM

Position : (136, 040) - (192, 096) Size : (057, 057) Ellipse Selected

Keyboard Options

While creating a display object, the following keyboard options are available:

- Holding down the **Shift** key while dragging-out the object will cause the object to be drawn such that it is centered on the initial mouse position, with one of its corners defined by the current mouse position. This is useful for drawing symmetrical figures centered on an initial point.
- Holding down the **Ctrl** key while dragging-out the object will keep its horizontal and vertical sizes the same. This is useful when you want to be sure that you draw an exact circle or square using the ellipse or rectangle objects.

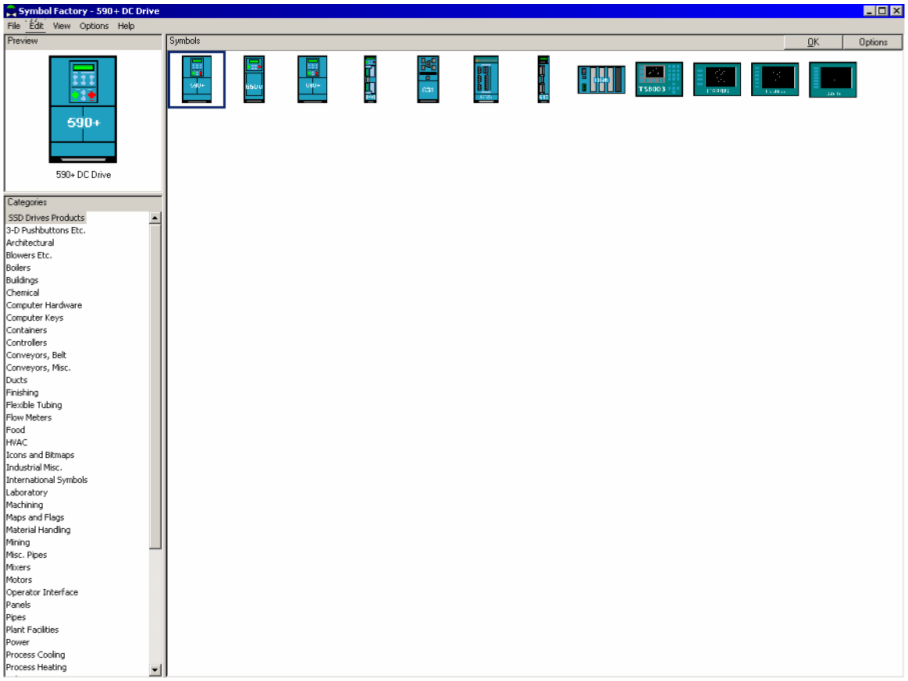
These options are also active when objects are re-sized.

Lock Insert Mode

The padlock icon on the drawing toolbox can be used to add a number of objects of the same basic type without having to click the toolbox icon for each item in turn. To cancel lock mode, click the padlock icon again, or press the **Escape** key. The same operation can be performed by using the Lock Mode command on the Insert menu.

Using The Symbol Library

To add an image from DSI8000's extensive symbol library, click on the Book icon in the toolbar, or select the Picture / Symbol command from the Insert menu. The symbol library will open at the last-accessed page, allowing an image to be selected.



Double-click on an image to select, and then drag-out the required size of the image as you would when inserting any other kind of object. The software will automatically create a Picture object containing the selected image. You should refer to the later sections of this manual for details on how this object might be further manipulated.

Selecting Objects

To select a display object, simply move your mouse pointer over the object in question, and perform a left-click. You will notice that while your pointer is hovering over an object, a bounding rectangle is drawn in blue to help show what will be selected. When the actual selection is performed, the rectangle will change to red, and handles will appear, so as to allow you to re-size the object as required. If you find that the object you want to select is hidden below another object, press the **Alt** key to allow the selection to be made.

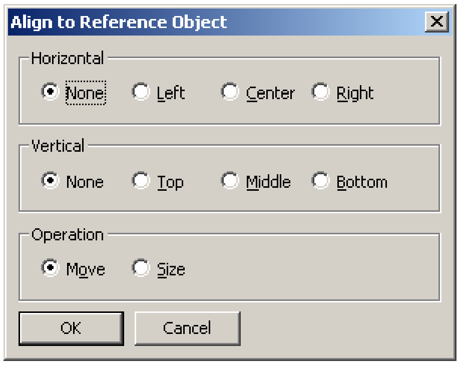
To select several objects, either drag-out a selection rectangle around the objects you want to select, or select each object in turn, holding down the **Shift** key to indicate that you want each object to be added to the selection. If multiple objects are selected, the red rectangle will surround all of the objects, and the handles can then be used to resize the objects as a group. The relative size and position of the objects will be maintained, as long as DSI can do so without violating minimum size requirements.

Moving And Resizing

Objects can be moved by first selecting them, and then by dragging them to the required position on the display page. If Smart Align is turned on, guidelines will appear to help you align the objects with other items on the page. Holding down **Ctrl** while moving an object will leave a copy of the object in its original position, thereby allowing duplicates to be created. You can also use the cursor keys to "nudge" the current selection a single pixel in the required direction. Holding down **Ctrl** while nudging will increase the movement of the objects by a factor of eight.

Objects can be resized by selecting them, and then by dragging the appropriate handle to the required position. Once again, if Smart Align is turned on, guidelines will appear to help you align the objects with other items on the page. The **Shift** and **Ctrl** keys can be used to modify the resize behavior as described in the Adding Display Objects section. Note that DSI will always constrain resizing operations so as to ensure that objects stay on the screen, and to make sure that items do not exceed their maximum permitted size, or shrink below the minimum size appropriate to their format.

Aligning Objects



Align to Reference Object

Horizontal: ● None ○ Left ○ Center ○ Right

Vertical: ● None ○ Top ○ Middle ○ Bottom

Operation: ● Move ○ Size

While the Smart Alignment options discussed above allow many alignment operations to be performed by hand, there are times that you will want the software to perform the alignment for you. This can be done by selecting a number of objects, starting with the object that you wish to use as the reference point for the alignment operation. Note that the reference object is always shown with a double-square at its center. Once you have made your selection, use the Align command on the Arrange menu to display the dialog box shown.

The Horizontal and Vertical settings can be used to indicate what type of alignment is to be performed, while the Operation setting indicates whether the objects should be resized or moved to achieve the desired result.

As an example, in Move mode, selecting Left for Horizontal will align the left-hand edges of all the objects with the left-hand edge of the reference object. Similarly, selecting Middle for vertical will align the objects so that the horizontal line through the center of each are aligned with the same line through the center of the reference object.

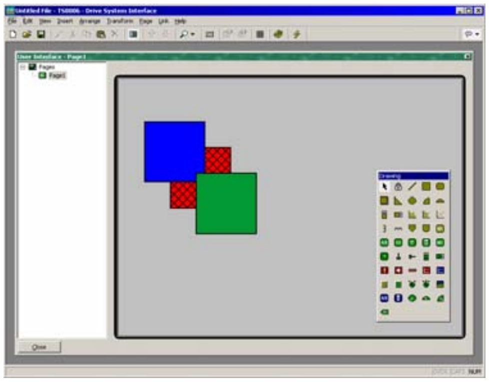
In Size mode, the edge-alignment operations work by growing the non-reference objects in order to achieve the desired results, while the center-alignment operations work by changing the height or width of the objects to make them match the reference object. You may want to experiment with Size mode to get a better idea of its operation.

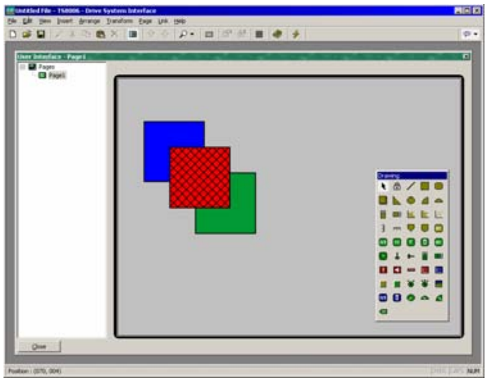
Spacing Objects

If you have a number of objects that you wish to space equally on the page, you may use the Space Equally Vertical or Space Equally Horizontal commands on the Arrange menu. The commands work on the currently selected objects, and attempt to reallocate the free space between the items to achieve equal spacing. The two outer objects will be left in their current positions. Note that the command may fail if an inappropriate set of objects are selected, and may not achieve perfect spacing if the available space is too limited.

Reordering Objects

Objects on a display page are stored in what is known as a "z-order." This defines the sequence in which the objects are drawn, and therefore whether or not a given object appears to be in front of or behind another object. In the first example below, the hatched square is shown behind the solid squares i.e. at the bottom of the z-order. In the second example, it has been moved to the front of the order, and appears in front of the other figures.


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
Close


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
Close

To move items in the z-order, select the items, and then use the various commands on the Arrange menu. The Move Forward and Move Backward commands move the selection one step in the indicated direction, while the Move To Front and Move To Back commands move the selection to the indicated end of the z-order. Alternatively, if you have a mouse that is equipped with a wheel, the wheel can be used to move the selection. Scrolling up moves the selection to the back of the z-order; scrolling down moves the selection to the front.

Grouping Objects

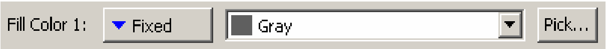
If you have several objects that you wish to treat as a single object, you may select them as described above and then use the Group command on the Arrange menu. You can perform the same operation by pressing the Ctrl+G key combination. Once a group has been created, it can be moved, sized and copied just like a single object. A group can be broken into its component objects by selecting it and using the Ungroup command, or the Ctrl+U key combination. Note that groups can comprise both objects and other groups, and that groups can be nested indefinitely. You should typically avoid excessive levels of grouping, however, as it can make it difficult to edit the most deeply nested objects.

Editing Objects

In addition to the above, objects can be edited in various ways:

- The various clipboard commands on the Edit menu (Cut, Copy and Paste), or the corresponding toolbar icons, can be used to duplicate items or move them around on a page or between pages. The Duplicate command can be used to perform a Copy operation, immediately followed by a Paste operation. Note that when a Paste is performed, DSI will offset the newly-pasted item if it will exactly overlay an item of the same type.
- The more detailed properties of an object can be edited by double-clicking the object, or by using the Properties command on the Edit menu. A dialog box will be displayed, allowing all of the objects to be accessed. The properties associated with each object will be described below.

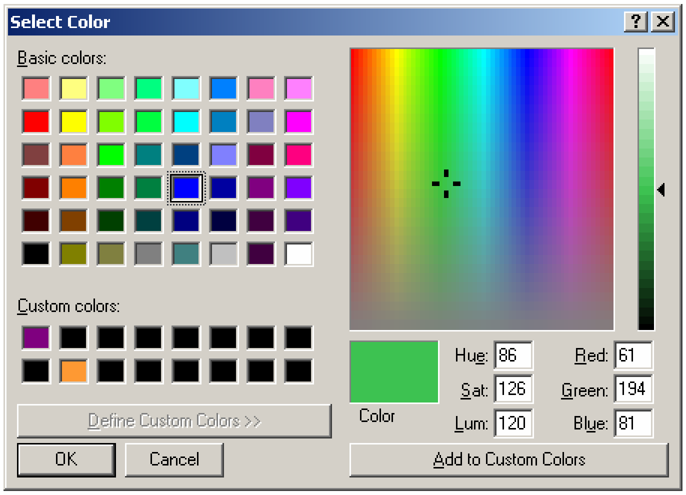
Defining Colors


Fill Color 1: Fixed Gray

Many properties of objects relate to the colors in which the object is to be drawn. The example shows one of the fill colors from a Rectangle object.

You will note that the color property is presented by means of a drop-down menu button, a drop-down list and the Pick button. The drop-down menu is used to select the color mode, which can be any one of the following:

- In *Fixed* mode, the color does not change, and is selected from the drop-down list, or by invoking the color selection dialog by pressing the Pick button.
- In *Tag Text* mode, the color is animated to match the foreground color defined by a particular tag. The specific tag can be selected by pressing the Pick button.
- In *Tag Back* mode, the color is animated to match the background color defined by a particular tag. The specific tag can be selected by pressing the Pick button.

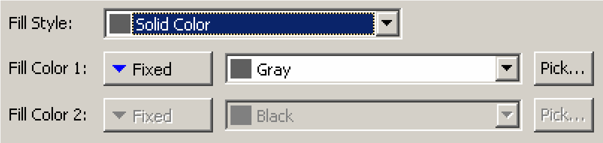


Hue: 86 Sat: 126 Lum: 120
Red: 61 Green: 194 Blue: 81

The drop-down list contains the fixed following colors: the sixteen standard VGA colors, the sixteen user-defined custom colors, and fourteen shades of gray ranging from white to block. The color selection dialog referenced above is shown here.

This dialog offers several ways of defining a color. You can pick from the palette, pick from the "rainbow" window, or enter the explicit HSL or RGB parameters. The dialog also allows custom colors to be added to the palette. These will appear whenever the dialog is invoked, and will also appear in the drop-down list described above. Note that not every color that is displayed in the "rainbow" will be capable of being rendered on the panel's 256-color display. DSI8000 will choose the nearest color within the abilities of the device.

Defining Fill Patterns

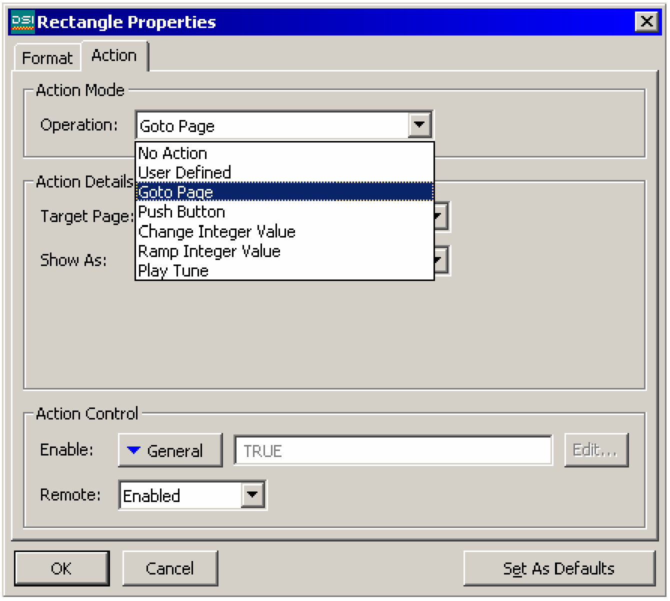

Fill Style: Solid Color

Fill Color 1: Fixed Gray

Fill Color 2: Fixed Black

A fill pattern is defined as shown. The *Fill Style* property is used to select the hatch or dotted pattern to be used, while the two color properties are used to define the colors to be used to form the pattern. Each color is defined as explained above. The second color is not required when a solid fill is selected.

Defining Actions


Rectangle Properties

Action Mode

Operation: Goto Page

Action Details

Target Page:

Show As:

Action Control

Enable: General TRUE

Remote: Enabled

Many objects can be made touch-sensitive such that certain actions will occur when they are pressed, held-down or released. To define the actions to be performed by an object, display the properties of that object and select the Action tab.

The drop-down list is used to select the action mode, each of which is described below.

Enabling Actions

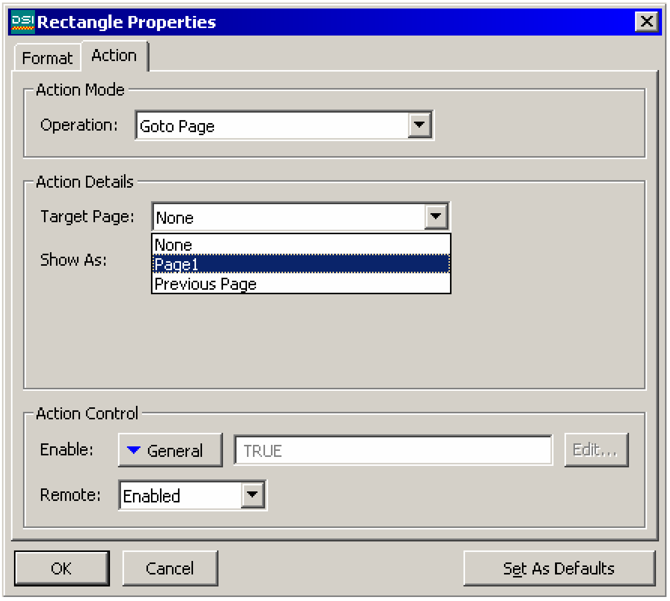
If you want to make a particular action dependent on some condition being true, enter an expression for that condition in the Enable field for the action in question. This expression may reference a flag tag directly, or may use any of the comparison or logical operators defined in the Writing Expressions section. If you need more complex logic, such that one of several actions is performed based on more complex decision-making, configure the object in user-defined mode and use it to invoke a program that implements the required logic. In addition, the Remote property can be used to enable or disable this action via the web server's virtual panel facility: in order for remote access to be allowed, the Enable expression must evaluate to a non-zero value, and the Remote property must be set to Enabled.

Action Descriptions

The sections below describe each available type of action. When each type is selected, the Action Details portion of the action dialog box will change to show the available options.

The Goto Page Action

This action is used to instruct the TS8000 to show a new page. The options are shown below:



Operation: Goto Page

Target Page: None

Show As:

Enable: General TRUE

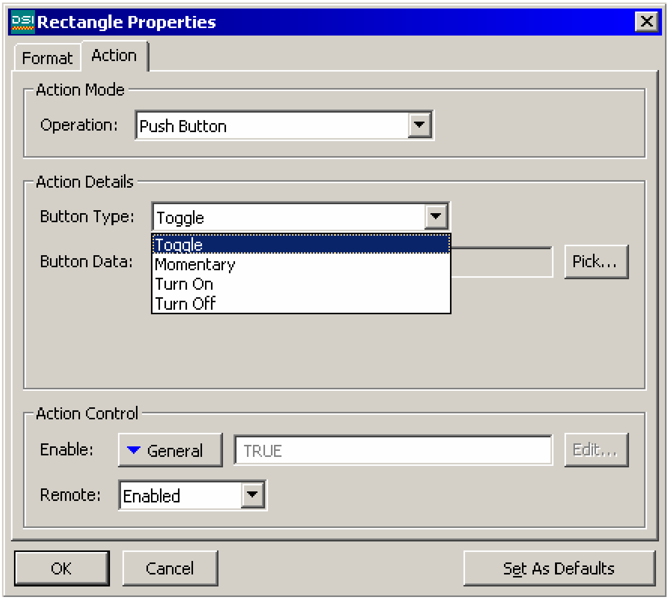
Remote: Enabled

- The *Target Page* property is used to indicate which page should be displayed. You can either choose a specific one to be displayed, or choose Previous Page to return to what was displayed before the current page was called.

- The *Show As* property is used to define how the page should be displayed. Aside from displaying it as a normal page, it can be shown as a popup page or as a popup menu. Both types of popup are shown on top of the existing page, and while they are displayed, the panel keys and touch-screen will assume the functions for the new page. Popup menus are displayed aligned to the left of the display so as to match up with the soft keys, while popup pages are displayed in the position indicated by the page properties. Note that an object or key on the new page must be assigned the **HidePopup( )** action to remove the popup.

The Push Button Action

This action is used to emulate a pushbutton. The options are shown below:



Operation: Push Button

Button Type: Toggle

Button Data:

Enable: General TRUE

Remote: Enabled

- The *Button Type* property is used to define the object's behavior.

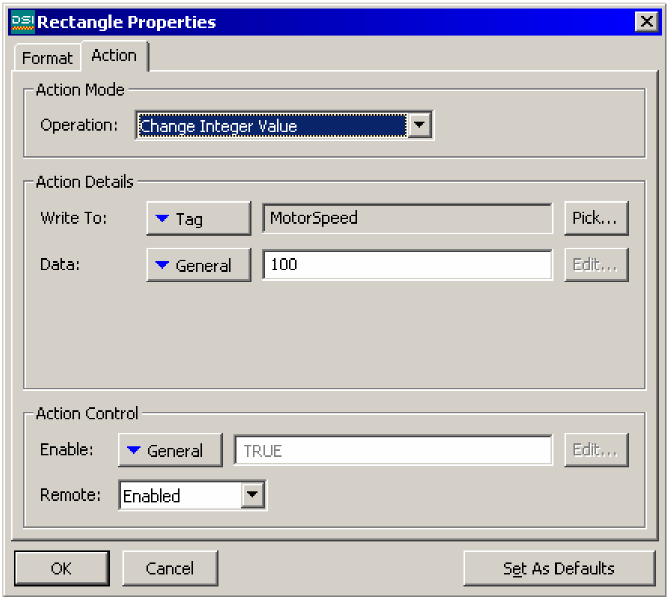
Button Type	Resultant Action
Toggle	Change the data state when the object is pressed.
Momentary	Set the data to 1 when the object is pressed. Set the data to 0 when the object is released.
Turn On	Set the data to 1 when the object is pressed.
Turn Off	Set the data to 0 when the object is pressed.

- The *Button Data* property is used to define the data to be changed.

In the example above, the object will toggle the value of the assigned tag.

The Change Integer Value Action

This action is used to write an integer value to a data item. The options are shown below.



Operation: Change Integer Value

Write To: Tag MotorSpeed

Data: General 100

Enable: General TRUE

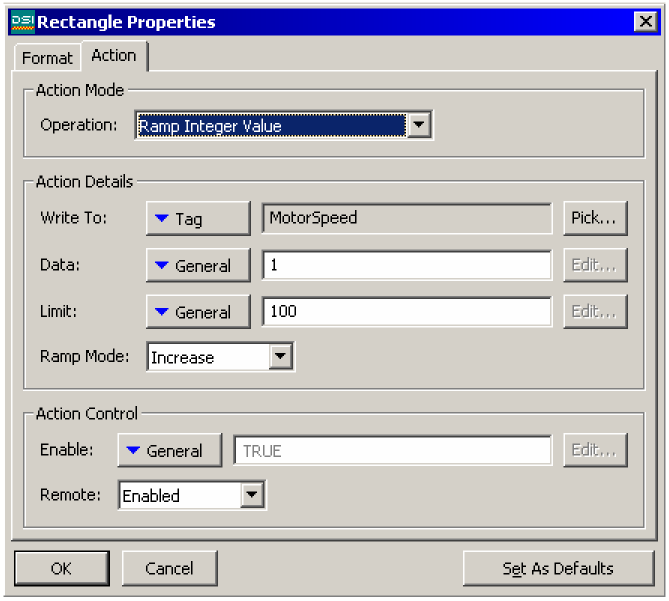
Remote: Enabled

- The *Write To* property is used to define the data item being changed.
- The *Data* property is used to define the data to be written.

In the example above, the object will set the MotorSpeed tag to 100.

The Ramp Integer Value Action

This action is used to increase or decrease the value of a data item. The options are shown below:



Operation: Ramp Integer Value

Write To: Tag MotorSpeed

Data: General 1

Limit: General 100

Ramp Mode: Increase

Enable: General TRUE

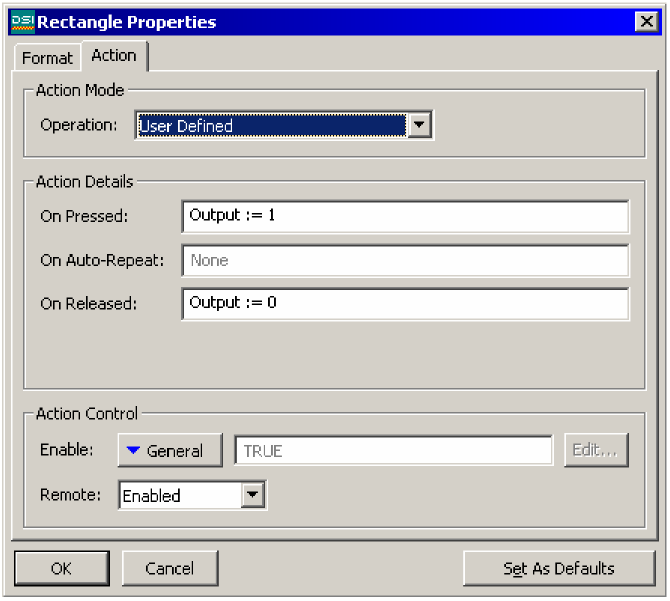
Remote: Enabled

- The *Write To* property is used to define the data item to be changed.
- The *Data* property is used to define the step by which to raise or lower the item.
- The *Limit* property is used to define the minimum or maximum data value.
- The *Ramp Mode* property is used to define whether to raise or lower the item.

In the example, holding the object will raise MotorSpeed by 1 until it reaches 100.

The User Defined Action

This action is used to perform complex operations. The options are shown below:



Operation: User Defined

On Pressed: Output := 1

On Auto-Repeat: None

On Released: Output := 0

Enable: General TRUE

Remote: Enabled

- The *On Pressed* property is used to define the action to be performed when the object is pressed.
- The *On Auto-Repeat* property is used to define the action to be performed when the object is pressed and then held down. The action occurs both on the initial depression and on subsequent auto-repeats, so there is no need to define both this property and On Pressed.
- The *On Released* property is used to define the action to be performed when the object is released.

The above properties for this action may invoke any of the functions from the Function Reference or the data modification operators described in the Writing Actions section, or it may run a program. In the example above, a user-defined action is used to implement a momentary pushbutton.

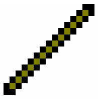
Using Default Settings

The dialog box used to edit the properties of each type of object has a button in its bottom right-hand corner labeled Set As Defaults. This button can be used to save the current settings of certain properties of the object as the default settings to be used when creating a new object. The same function can be performed by selecting the Save As Defaults command from the Edit menu, or by pressing the Ctrl+E key combination. Note that not all properties are included in the default settings - only those that refer to formatting, as opposed to the underlying data presented by the object, are saved. The default settings can be applied to the currently selected object or objects by using the Apply Defaults command from the Edit menu, or by pressing the Ctrl+L key combination.

Object Descriptions

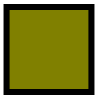
The sections below describe each object found in the drawing toolbox.

The Line Object

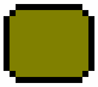


The *Line* object is a line drawn between two points. Its only property is the style of line to be used. In addition to the solid colors shown on the line toolbox, a number of dotted styles can also be accessed via the properties dialog box.

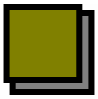
The Simple Geometric Objects



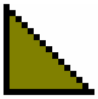
The *Rectangle* object is a rectangle with a defined outline and fill pattern. The fill pattern may be set to No Fill to draw the outline alone, or the outline may be set to None to draw a figure without a border.



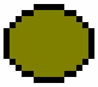
The *Round Rectangle* object is similar to the rectangle, but has rounded corners. When the object is selected, an additional handle appears, allowing the radius of the corners to be edited by dragging the handle from side to side.



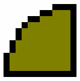
The *Shadow* object is similar to the rectangle, but with either a drop-shadow or with a shaded 3D effect. The object is often drawn so as to allow it to act as a frame around text objects or other groups of elements.



The *Wedge* object is a right-angled triangle located within one quadrant of a bounding rectangle. In addition to the outline and fill properties, the wedge has a property to indicate which quadrant it should occupy.



The *Ellipse* object is an ellipse with a defined outline and fill pattern. The fill pattern may be set to No Fill to draw the outline alone, or the outline may be set to None to draw a figure without a border.



The *Ellipse Quadrant* object is one quadrant of an ellipse. In addition to the outline and fill properties, the ellipse quadrant has a property to indicate which quadrant it should occupy.

The *Ellipse Half* object is one half of an ellipse. In addition to the outline and fill properties, the ellipse half has a property to indicate which of the four possible halves will be drawn.

The properties for these objects need little further explanation, other than to point out that the quadrant or half rendered by the Wedge, Ellipse Quadrant or Ellipse Half objects can also be edited via the command found on the Transform menu.

The Tank Objects

The *Conical Tank* object is a conical tank with a defined outline and fill pattern. When the object is selected, additional handles appear, allowing the exact shape of the tank to be modified by dragging the handles as required.

The *Round Bottomed Tank* object is a tank with a defined outline and fill pattern. When the object is selected, an additional handle appears, allowing the exact shape of the tank to be modified by dragging the handle as required.

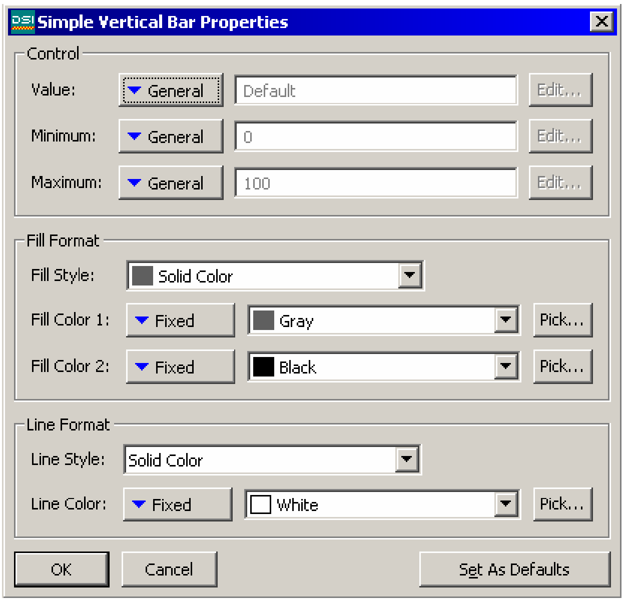
The properties for these objects need little further explanation.

The Simple Bar Objects

The *Simple Vertical Bar* object allows an expression to be drawn as a vertical bar-graph between specified minimum and maximum values. Additional properties allow the object's fill color and border style to be defined.

The *Simple Horizontal Bar* object allows an expression to be drawn as a horizontal bar-graph between specified minimum and maximum values. Additional properties allow the object's fill color and border style to be defined.

The properties are accessed by double-clicking the object:



Simple Vertical Bar Properties

Control

Value: General | Default | Edit...

Minimum: General | 0 | Edit...

Maximum: General | 100 | Edit...

Fill Format

Fill Style: Solid Color

Fill Color 1: Fixed | Gray | Pick...

Fill Color 2: Fixed | Black | Pick...

Line Format

Line Style: Solid Color

Line Color: Fixed | White | Pick...

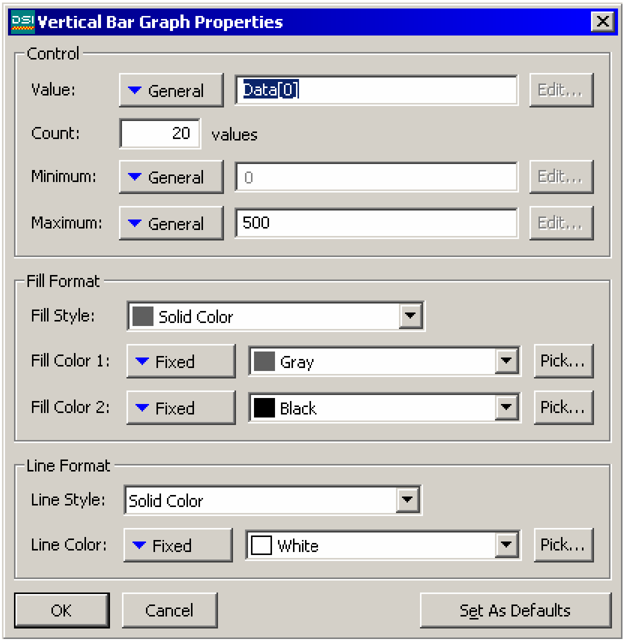
- The *Value* property is used to specify the value to be displayed. In the example given above, the object is configured to display the level of a tank.
- The *Minimum* and *Maximum* properties are used to specify the range of values to be shown. In the example above, a range of 0 to 1000 is specified.
- The *Fill Format* properties are used to define the fill color for the object. The filled area of the bar is drawn in the pattern and colors indicated, while the unfilled area is drawn with solid *Fill Color 2*.
- The *Line Format* properties are used to define the border for the object.

The Bar Graph Objects

The *Vertical Bar Graph* object displays a set of values from an array as a number of vertical bars. Each value is scaled according to the same minimum and maximum values. Between 2 and 400 values can be shown.

The *Horizontal Bar Graph* object displays a set of values from an array as a number of horizontal bars. Each value is scaled according to the same minimum and maximum values. Between 2 and 400 values can be shown.

The properties are accessed by double-clicking the object:



Vertical Bar Graph Properties

Control

Value: General Data[0]

Count: 20 values

Minimum: General 0

Maximum: General 500

Fill Format

Fill Style: Solid Color

Fill Color 1: Fixed Gray

Fill Color 2: Fixed Black

Line Format

Line Style: Solid Color

Line Color: Fixed White

- The *Value* property is used to specify the first array element to be shown.
- The *Count* property is used to specify the number of values to be shown.
- The *Minimum* and *Maximum* properties are used to specify the scaling.

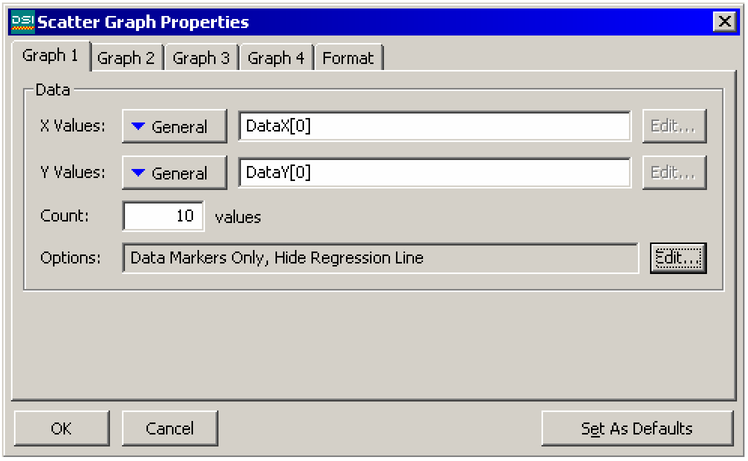
- The *Fill Format* properties are used to define the fill color for the object. The filled areas of the bars are drawn in the pattern and colors indicated, while the unfilled areas are drawn with solid *Fill Color 2*.

- The *Line Format* properties are used to define the border for the object.

The Scatter Graph Object

The *Scatter Graph* object displays one set of data values plotted against the other, with optional data point markers and regression line.

The properties are split over multiple tabbed pages, and are accessed by double-clicking the object. The Graph1 tab properties are shown below:



X Values: General DataX[0]

Y Values: General DataY[0]

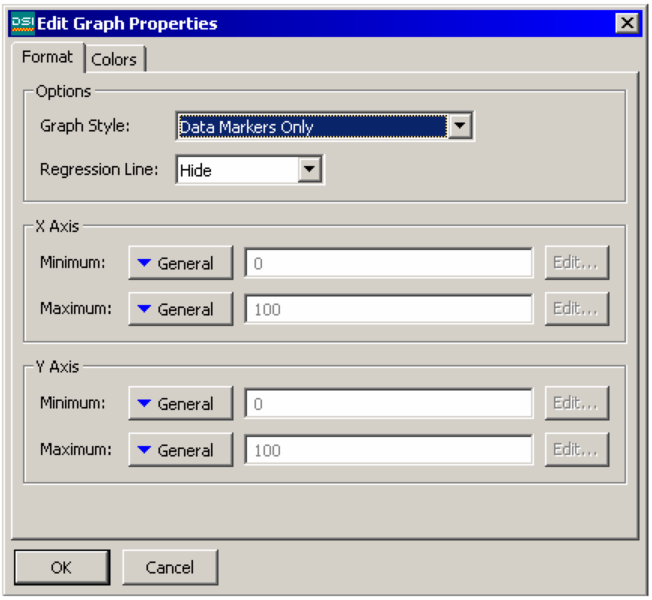
Count: 10 values

Options: Data Markers Only, Hide Regression Line

- The *X Values* property is used to define the array element that contains the first *x* coordinate to be plotted, while the *Y Values* property is used to define the array element that contains associated *y* coordinate.

- The *Count* property is used to specify the number of values to be shown.

The *Options* field can be expanded by selecting the Edit button to reveal the following properties as shown below:



Options

Graph Style: Data Markers Only

Regression Line: Hide

X Axis

Minimum: General 0 Edit...

Maximum: General 100 Edit...

Y Axis

Minimum: General 0 Edit...

Maximum: General 100 Edit...

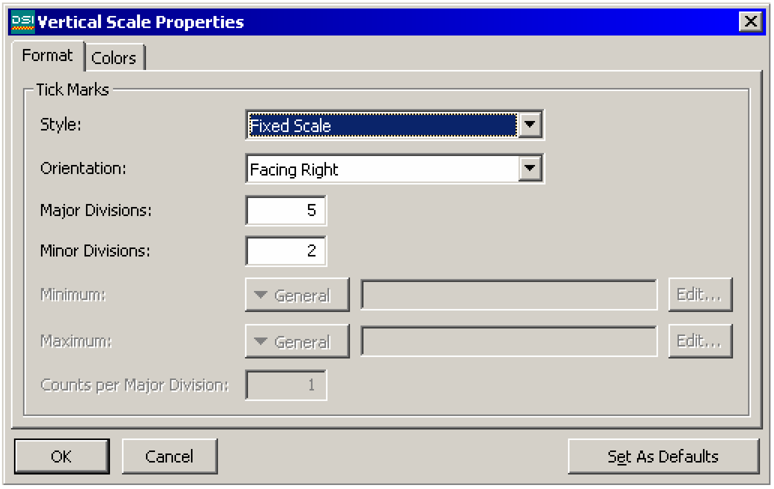
- The *Graph Style* property is used to select between a line graph, a line graph with data markers, or a set of data markers without a line. The various other formatting options on the Color page will be enabled or disabled as required.
- The *Regression Line* property is used to show or hide the regression line for the data sets. If the line is enabled, the software will calculate a best-fit line based upon the least-squares method, and draw it on top of the data sets.
- The *X Axis Minimum* and *X Axis Maximum* properties are used to specify the scaling for the horizontal axis, while the *Y Axis Minimum* and *Y Axis Maximum* properties are used to specify the scaling for the vertical axis.
- The remaining properties are used to define the background color of the object, and whether or not an outline should be drawn around its outer edge.

The Scale Objects

The *Horizontal Scale* object displays a scale with a specified number of minor and major divisions. It is often used to label other objects, such as bar graphs.

The *Vertical Scale* object displays a scale with a specified number of minor and major divisions. It is often used to label other objects, such as bar graphs.

The properties are accessed by double-clicking on the object:



Style: Fixed Scale

Orientation: Facing Right

Major Divisions: 5

Minor Divisions: 2

Minimum: General

Maximum: General

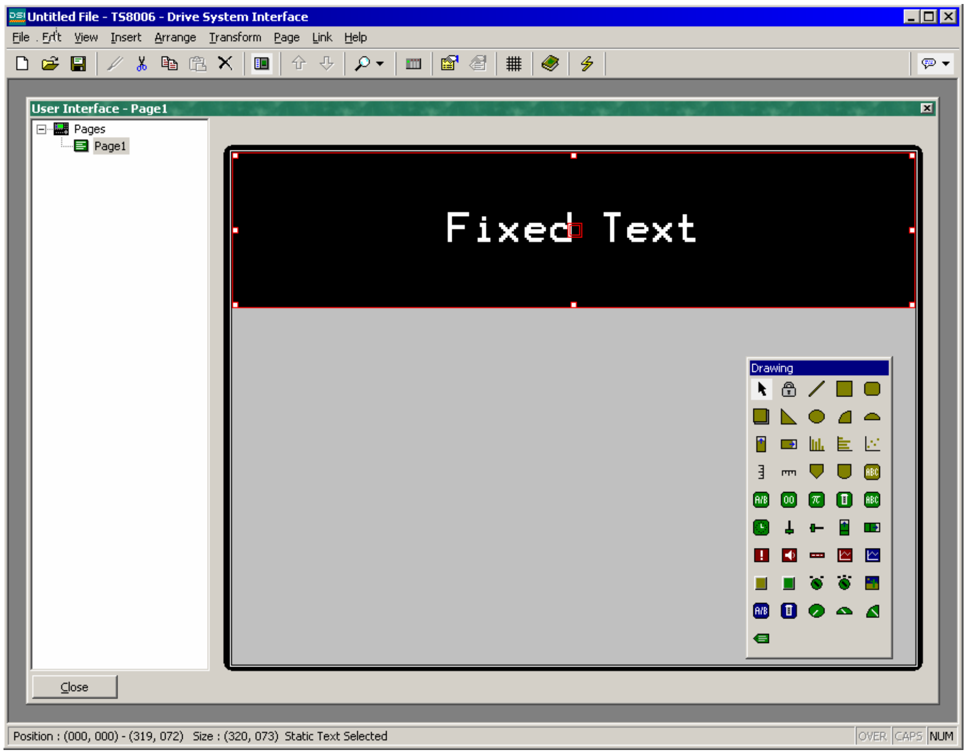
Counts per Major Division: 1

- The *Style* property selects between Fixed Scale and Adaptive Scale tick marks.
- The *Orientation* property is used to indicate the direction in which the tick marks should point. Vertical scales support selections of left and right, while horizontal scales support selections of up or down.
- The *Major Divisions* property is used to indicate into how many major divisions the scale should be divided. Large tick marks are drawn at each division. The lowest number of major divisions is one, in which case large tick-mark will be drawn at the ends of the scale, but not along its length.
- The *Minor Divisions* property is used to indicate into how many minor divisions each major division should be divided. Smaller tick-marks are drawn at each division. Selecting a value of one for this property will disable minor divisions.
- The remaining properties define the fill pattern and line style of the scale.

The Fixed Text Object

The *Fixed Text* object is used to add unchanging text to a page. The text is displayed in a specified font and color, and with a specified justification. The text can also be translated for international applications.

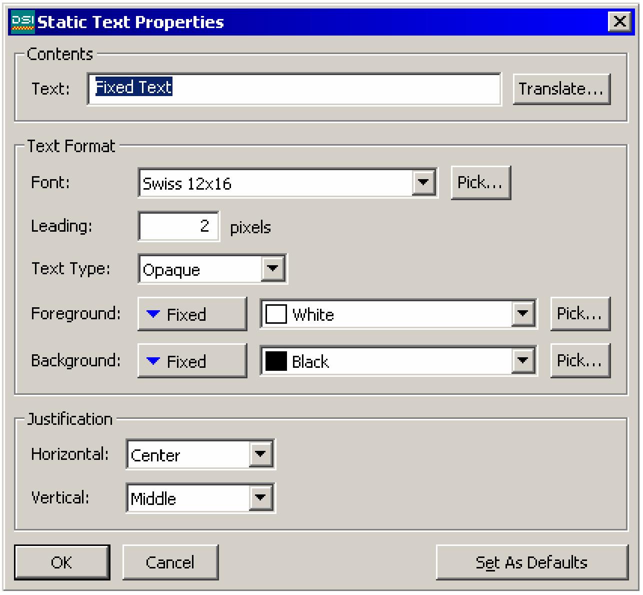
When the text is created, a cursor will appear, allowing the text to be entered:


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Fixed Text
Drawing
Close
OVER CAPS NUM

Position : (000, 000) - (319, 072) Size : (320, 073) Static Text Selected

The text editor supports cutting, pasting and all the other options normally found within a Windows editor. The editor will also configure the keyboard to use the appropriate Input Method Editor for the currently selected default language

Note that only the default language text can be edited directly. Other versions of the text must be edited via the properties dialog box, which is accessed by selecting the object and pressing Alt+Enter, or by selecting the Properties command from the Edit menu.



Contents

Text: Fixed Text

Text Format

Font: Swiss 12x16

Leading: 2 pixels

Text Type: Opaque

Foreground: Fixed □ White

Background: Fixed ■ Black

Justification

Horizontal: Center

Vertical: Middle

- The *Text* property is used to specify the text to be displayed. As mentioned above, the US English version of the text can also be edited directly on the display page when the object is created, or by clicking an existing object.

- The *Font* property is used to specify the font to be used. The font list comprises the eight resident fonts found in all terminals, plus any custom fonts already created in this database. The Pick button can be used to invoke the font selection dialog, allowing any font that is installed on your system to be rendered in a form that can be used by the target device.

- The *Text Type* property is used to indicate whether the text should be drawn with a solid or transparent background. Transparent text can be used to overlay multiple objects while still allowing those objects to be seen.

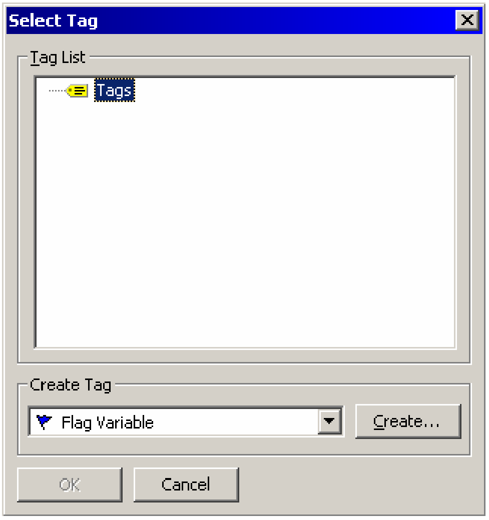
- The *Foreground* and *Background* properties are used to specify the colors to be used to draw the text. Obviously, having the same color for both settings will render the text unreadable.

- The *Horizontal* and *Vertical* justification properties are used to indicate where the text should be placed within the bounding rectangle of the object.

The Auto Tag Object

The *Auto Tag* object allows you to select a tag, and then automatically place the appropriate text object on the display. For example, selecting an integer tag will allow insertion of an appropriately-configured integer text object.

This is the icon you will use most often for adding tags to a page. It first displays the dialog box shown below to allow tag selection, and then creates one of the five tag text objects described in the next section. The new object will be configured so as to display the tag in question using its label and its formatting properties, as defined when the tag was created.



The Tag Text Objects

The tag text objects are used to display or edit an expression in textual form. Primarily, they are used to display tags, in which case the default format is taken from the Format tab associated with that tag in the Data Tags window. If a non-tag expression is entered, or if you want the formatting to differ from the default values for a tag, the format data can be overridden as required. There is one type of tag text for each tag family.

The *Flag Text* object is used to display a true or false condition.

The *Integer Text* object is used to display an integer expression.

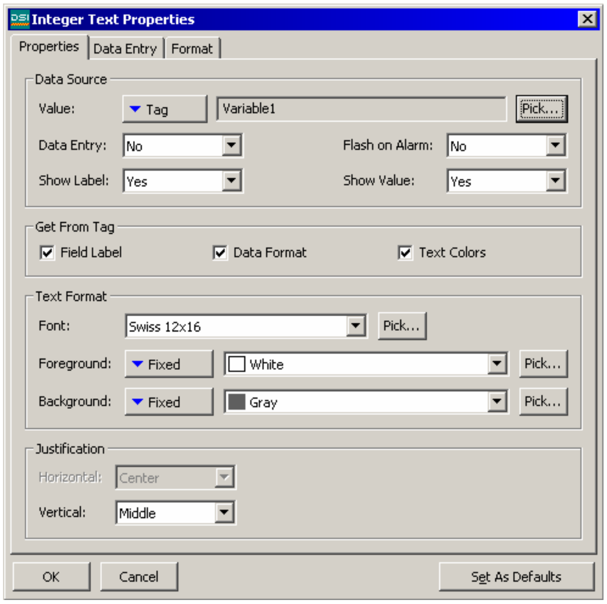
The *Real Text* object is used to display a floating-point expression.

The *Multi Text* object is used to display a multi-state condition.

The *String Text* object is used to display a string expression.

The properties of a tag text object are displayed using threeo tabbed pages.

The first page is more-or-less the same for all five object types:



Integer Text Properties

Properties | Data Entry | Format

Data Source
Value: Tag Variable1 Pick...
Data Entry: No
Flash on Alarm: No
Show Label: Yes
Show Value: Yes

Get From Tag
☑ Field Label ☑ Data Format ☑ Text Colors

Text Format
Font: Swiss 12x16 Pick...
Foreground: Fixed White Pick...
Background: Fixed Gray Pick...

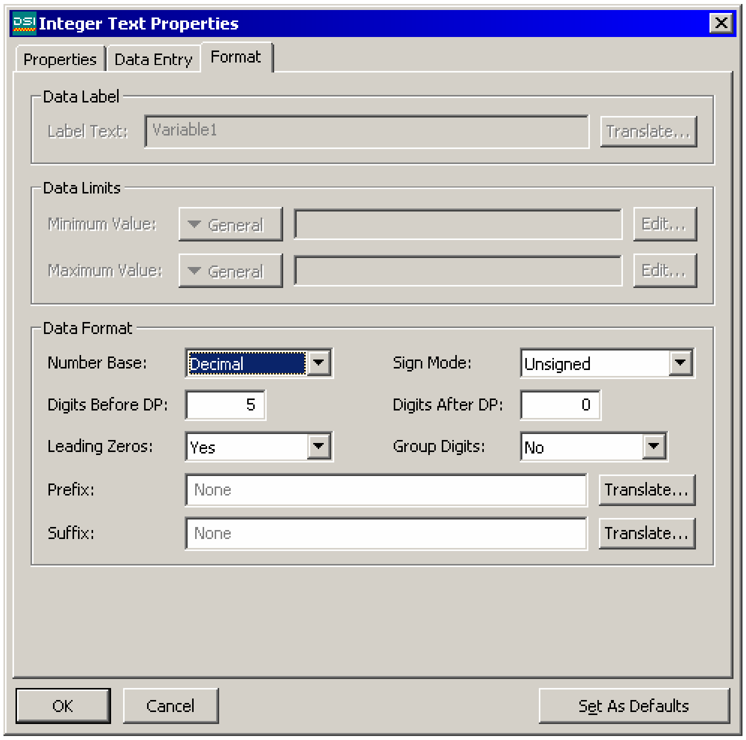
Justification
Horizontal: Center
Vertical: Middle

OK Cancel Set As Defaults

- The *Value* property is used to indicate from where the data for this object should be obtained. You may select a tag, a register in a communications device, or an expression which combines a number of such items. The data type of the item must be appropriate to the object in question (the Value property for an integer text object cannot be set equal to a string expression).

- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. For data entry to be enabled, the expression entered for the value property must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.
- The *Show Label* property is used to indicate whether or not you want the object to include a label to identify the data being displayed. If this property is set to yes, the label will be left-justified within the object's bounding rectangle, while the data itself will be right-justified. If this property is set to no, the Horizontal Justification property will be used to locate the data within the field.
- The *Get From Tag* properties are used to indicate from where the label text, the field format and the text colors should be obtained. The options presented depend on what was entered for the Value property. In each case, you may manually enter the data in the appropriate properties, or, assuming a suitable expression has been defined, you may instruct the object to get the required information from the underlying data tag.
- The *Flash on Alarm* property is used to indicate whether or not you want the text on the TS8000's display to flash if the tag entered in the value property is currently in an alarm state. This property is not available for string text objects, or for those objects which have a non-tag value defined for the value property.
- The balance of the properties control the font, colors and justification to be used when drawing the object. These properties require no further explanation.

The second page is used only for fields that are selected for data entry:



Data Label

Label Text: Variable1

Data Limits

Minimum Value: General

Maximum Value: General

Data Format

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 5

Digits After DP: 0

Leading Zeros: Yes

Group Digits: No

Prefix: None

Suffix: None

- The *Enable* property is used to define an expression that must be true in order for data entry to be permitted. This property may thus be used to implement a security system, or to restrict entry to certain machine states.

- The *Validate* property is used to define an expression that will be used to validate any entered values, e.g. **Amount %25 == 0** will only allow multiples of 25 to be entered into the variable Amount. The special system variable *Data* will hold the newly-entered value, but only during the execution of this expression. The code should evaluate to non-zero to allow entry, or zero to block it.

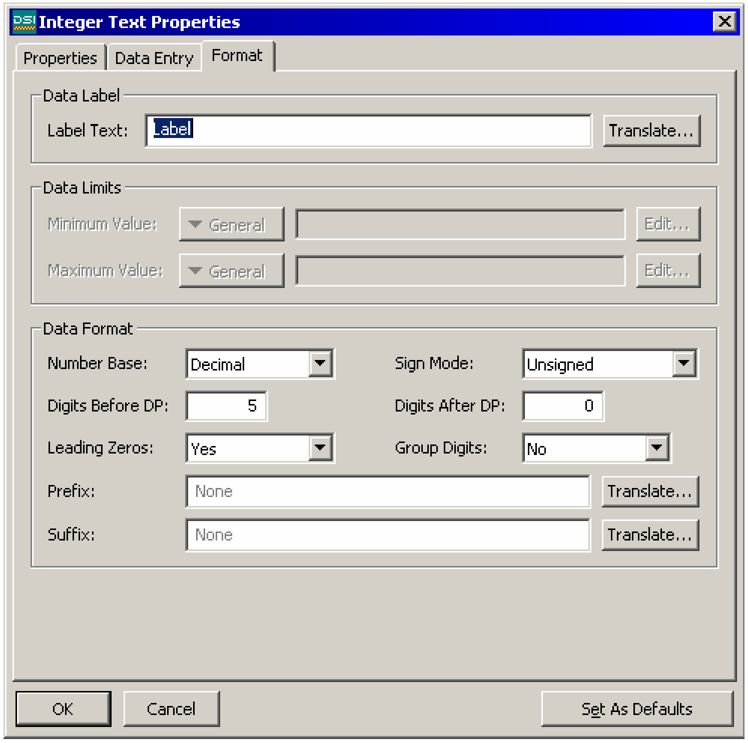
- The *On Selected* and *On Deselected* properties are used respectively to define actions to be executed when the user selects the field for entry, or when the user deselected the field by selecting another. The actions may invoke any of the various functions from the Function Reference or the data modification operators described in the Writing Actions section, or may run a program.

- The *On Entry Complete* and *On Entry Error* properties are used respectively to define actions to be executed when data entry is completed successfully, or when an invalid value is entered. The actions may invoke any of the functions from the Function Reference or the data modification operators described in the Writing Actions section, or may run a program.

- The *Keypad Type* property is used to select the type of keypad to be displayed when the value is edited. By default, a full keypad with raise and lower keys will be shown. The options available will vary according to the object type.

- The *Keypad Style* property is used to override the default color scheme associated with the popup keypad, and to substitute a high-contrast version in its place. This is used in low-visibility applications such as direct sunlight.

The third page varies according to the object in question, and displays the same information as the Format tab of the associated tag type. Different sections of the page will be enabled according to the settings provided for the Get Label and Get Format properties. The example below shows the Format tab for an integer text object.



Integer Text Properties

Data Label

Label Text: Label

Data Limits

Minimum Value: General

Maximum Value: General

Data Format

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 5

Digits After DP: 0

Leading Zeros: Yes

Group Digits: No

Prefix: None

Suffix: None

As can be seen, the properties shown are indeed identical to those shown on the Format tab of an integer tag. As mentioned above, the properties for the other types of object are similarly identical to those of the corresponding tag. You are thus referred to the earlier section of the manual regarding Data Tags for more information on each property.

Editing The Underlying Tag

If you want to edit a tag text object's properties, either double-click on the object, or right-click and select the Properties command from the resulting menu. If, however, you want to edit the properties of the tag that is being used to control the object, right-click and select the Tag Details command instead. The resulting dialog box will show the Data and Format tab from the Data Tags window, and allow you to change the various properties. Note that a change made via this mechanism will change all the objects controlled by that tag if those objects are configured to obtain their configuration from that source.

The Multi-Line Text Object

The *Multi-Line Status Text* object is used to display an on-off value, but split over several lines. This allows larger amounts of text to be shown, perhaps to provide prompts or help information to the operator.

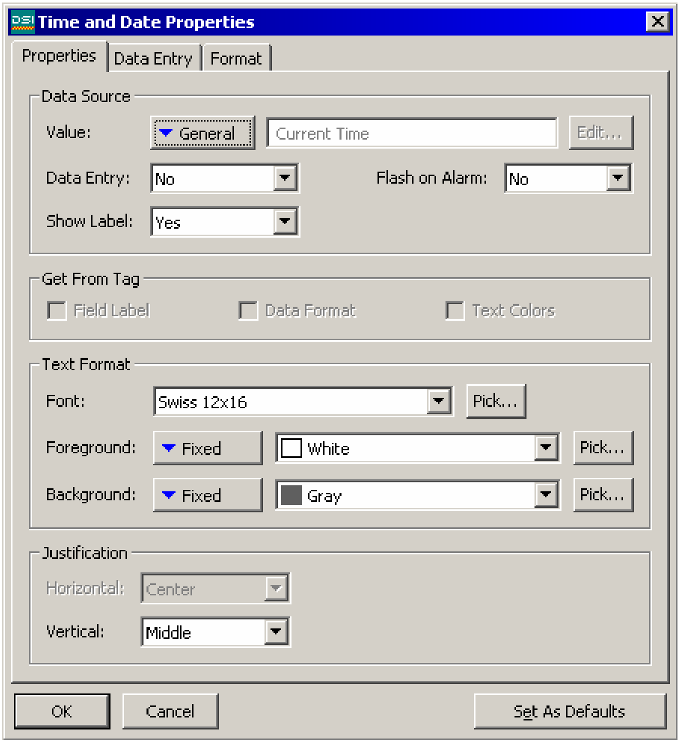
The *Multi-Line Multi Text* object is used to display one of a series of text values, but split over several lines. This allows larger amounts of text to be shown, perhaps to provide prompts or help information to the operator.

Each of these objects is as the associated single-line object, except that they do not support data entry. The text string to be displayed is broken into lines by the inclusion of vertical bar ("|") characters wherever a line-break is required.

The Time And Date Object

The *Time and Date* object is used to display the current time and date, or to display the contents of a time and date expression. It can also be used to edit such an expression, or to set the operator panel's real time clock.

The properties of a time and date object are displayed using three tabbed pages. The first page is shown below:



Data Source

Value: General | Current Time

Data Entry: No

Flash on Alarm: No

Show Label: Yes

Get From Tag

□ Field Label □ Data Format □ Text Colors

Text Format

Font: Swiss 12x16

Foreground: Fixed | White

Background: Fixed | Gray

Justification

Horizontal: Center

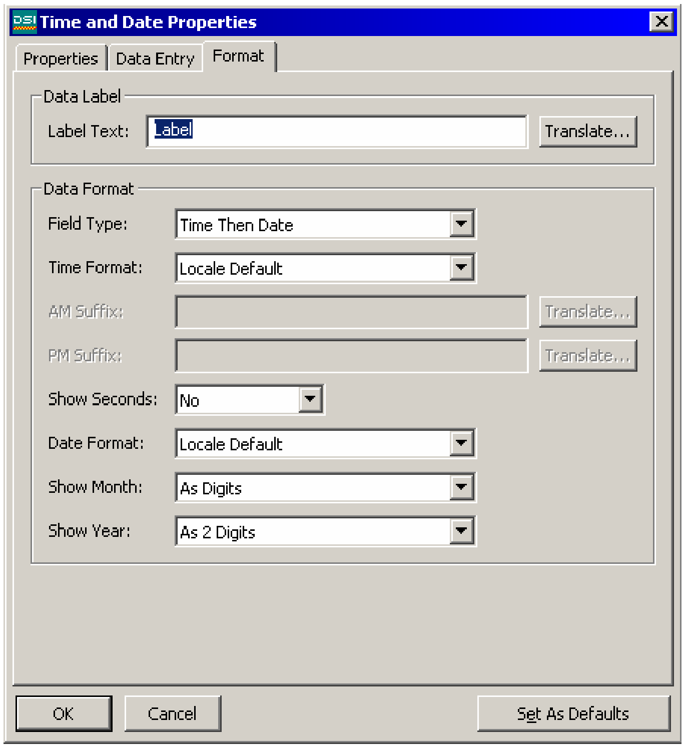
Vertical: Middle

- The *Value* property is used to indicate the time and date value to be displayed. If no value is entered, the current time and date is shown. If an expression is entered, it is taken to represent the number of seconds that have elapsed since

January 1, 1997. Such values are typically obtained using the various time and date functions described in the Function Reference.

- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. If no value property has been defined, this amounts to changing the current time or date. If a value property has been entered, the expression entered must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.
- The balance of the properties are as described for tag text objects. While it may look odd to have Get Label and Flash On Alarm properties, remember that the value property may be a tag, and so DSI does have access to the tag label and to the tag's alarm state, should you decide to use them.

The second page contains the data entry properties. These are as described for text tag objects. The tab is shown below:



Data Label

Label Text: Label

Data Format

Field Type: Time Then Date

Time Format: Locale Default

AM Suffix:

PM Suffix:

Show Seconds: No

Date Format: Locale Default

Show Month: As Digits

Show Year: As 2 Digits

- The *Label Text* property is used to define an optional label for the object.
- The *Field Type* property is used to indicate whether the field should display the time, the date or both. In the last case, this property also indicates in which order the two elements should be shown.
- The *Time Format* property is used to indicate whether 12-hour (civil) or 24-hour (military) time format should be used. As with other properties,

leaving this set to Locale Default will allow DSI to pick a suitable format according to the language selected within the operator panel.

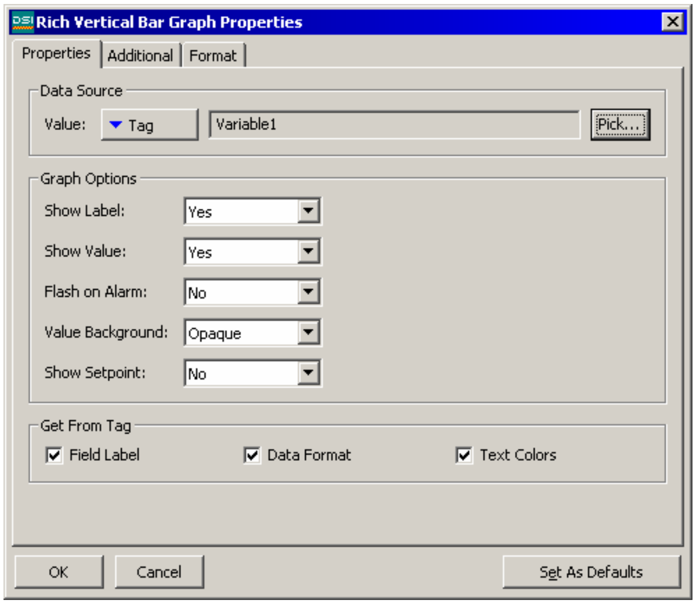
- The *AM Suffix* and *PM Suffix* properties are used with 12-hour mode to indicate the text to be appended to the time field in the morning and afternoon as appropriate. If you leave the property undefined, DSI will use a default.
- The *Show Seconds* property is used to indicate whether the time field should include the seconds, or whether it should just comprise hours and minutes.
- The *Date Format* property is used to indicate the order in which the various date elements (i.e. date, month and year) should be displayed.
- The *Show Month* property is used to indicate whether the month should be displayed as digits (i01 through 12) or as its short name (Jan though Dec).
- The *Show Year* property is used to indicate whether the date field should include the year, and if so, how many digits should be shown for that element.

The Rich Bar Objects

The *Rich Vertical Bar* object allows you to display a more complex bar-graph which includes a label, a numeric version of the data being displayed, and tick markers to indicate any associated setpoint.

The *Rich Horizontal Bar* object allows you to display a more complex bar-graph which includes a label, a numeric version of the data being displayed, and tick markers to indicate any associated setpoint.

The operation of these rich objects is analogous to that of the various tag text objects, in that they are capable of deriving much of the required formatting information from the tag used as their controlling value. Just as with tag text objects, two tabbed pages are used to edit the objects' properties. The first of these pages is shown below:



Data Source

Value: Tag Variable1

Graph Options

Show Label: Yes

Show Value: Yes

Flash on Alarm: No

Value Background: Opaque

Show Setpoint: No

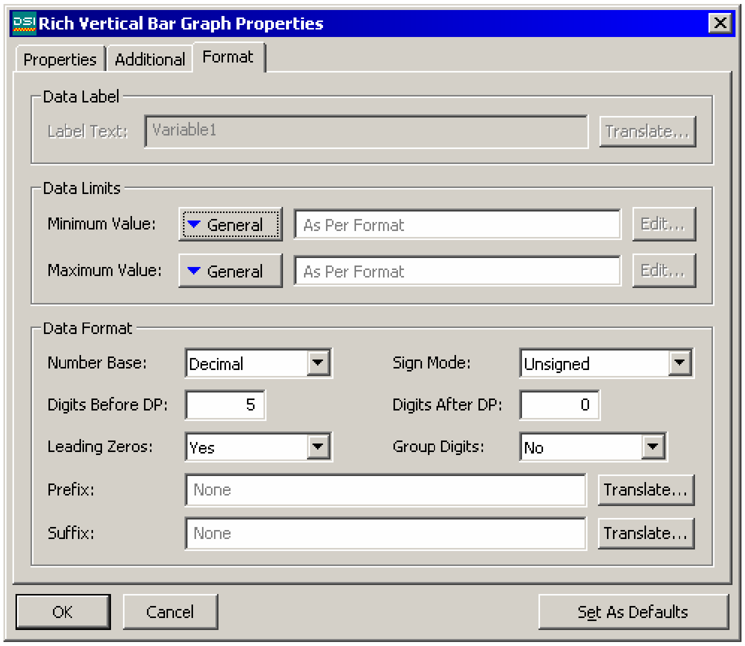
Get From Tag

☑ Field Label ☑ Data Format ☑ Text Colors

- The *Value* property is used to define the value to be displayed.
- The *Show Label* property is used to indicate whether a label should be included with the bar-graph. For vertical graphs, the label is included at the bottom; for horizontal graphs, it is included at the left-hand side. If a tag is used for the value property, the label may be obtained from that tag. Otherwise, it must be entered on the Format tab of the dialog box.
- The *Show Value* property is used to indicate whether the value of the data should be displayed within the graph itself. If a tag of the appropriate data type is used for the value property, the format may be obtained from the tag. Otherwise, as with the label, it must be entered on the Format tab.
- The *Show Setpoint* property is used to indicate whether tick marks should be added either side of the bar to indicate the setpoint for the controlling value. This option is only available if a tag has been entered for the value field.
- The *Flash on Alarm* property is used to indicate whether or not you want the text on the TS8000's display to flash if the tag entered in the value property is currently in an alarm state. This property is not available for those objects that have a non-tag value defined for the value property.
- The *Value Background* property is used to indicate whether the value should be drawn with a solid or transparent background. The choice of format will typically depend upon the visibility of the value against the bar itself.
- The *Get From Tag* properties are used to indicate from where the label text, the field format and the text colors should be obtained. The options presented depend on what was entered for the Value property. In each case, you may manually enter the data in the appropriate properties, or, assuming a suitable expression has been defined, you may instruct the object to get the required information from the underlying data tag.

The second page contains additional formatting information for the field.

The third page contains the label and formatting information for the field:



Rich Vertical Bar Graph Properties

Data Label

Label Text: Variable1 Translate...

Data Limits

Minimum Value: General As Per Format Edit...

Maximum Value: General As Per Format Edit...

Data Format

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 5

Digits After DP: 0

Leading Zeros: Yes

Group Digits: No

Prefix: None Translate...

Suffix: None Translate...

The properties shown are as described for an integer tag, and you are thus referred to the earlier section of the manual that refers to Data Tags for more information. Note that the existence of this object explains why one must enter minimum and maximum values for formulas, when such tags can never be the subject of data entry.

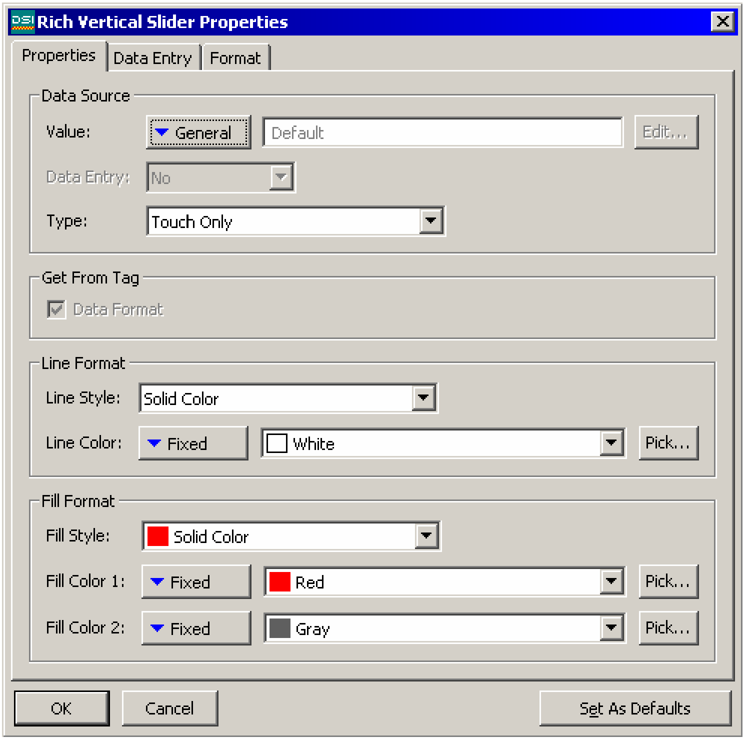
The Rich Slider Objects

The *Vertical Slider* object displays a value, typically from an integer tag, as a slider that can either display a value, or allow it to be manipulated by touching a specific location on the object, or by pressing buttons at either end.

The *Horizontal Slider* object displays a value, typically from an integer tag, as a slider that can either display a value, or allow it to be manipulated by touching a specific location on the object, or by pressing buttons at either end.

Just as with other rich objects, the slider objects are capable of deriving much of the required formatting information from the tag used as their controlling value. Just as with tag text objects, multiple tabbed pages are used to edit the object’s properties.

The first of these pages is shown below:



Data Source

Value: General Default

Data Entry: No

Type: Touch Only

Get From Tag

☑ Data Format

Line Format

Line Style: Solid Color

Line Color: Fixed □ White

Fill Format

Fill Style: 🟥 Solid Color

Fill Color 1: Fixed 🟥 Red

Fill Color 2: Fixed ⬛ Gray

- The *Value* property is used to indicate from where the data for this object should be obtained. You may select a tag, a register in a communications device, or an expression that combines a number of such items.

- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. Selecting Local will enable data entry, but prevent access via the virtual panel facility of the web server. For data entry to be enabled, the expression entered for the value property must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.

- The *Type* property is used to indicate the type of slider to be displayed. The three types of sliders allow data entry via buttons, via direct manipulation, or via both methods. You should note that direct manipulation can be somewhat risky, in that accidental touches may result in large changes to process values.

- The *Get From Tag* property is used to indicate whether the data format should be obtained from the controlling value, or from the Format page. Note that most of the format values are unused, save the minimum and maximum values.

- The *Line Format* properties are used to define the style of the object's outline.

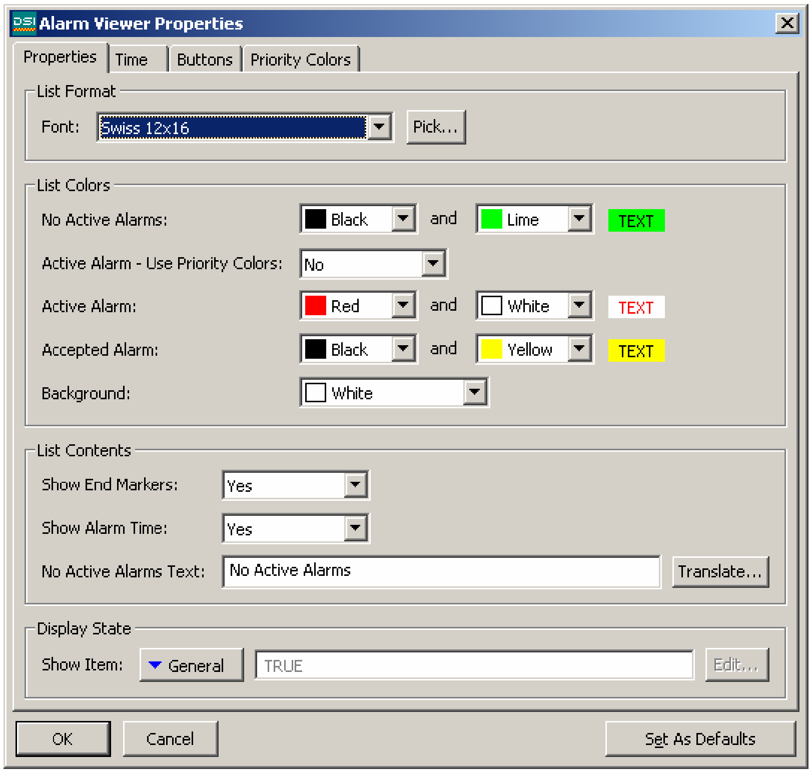
- The *Fill Format* properties are used to define the color and style of the object's background and of the slider itself. The background is drawn in Fill Color 2, while the slider is drawn in either Fill Color 1, or the combination of the two colors that is specified by the Fill Style.

The second page is used to control data entry, and functions are as for tag text objects. You are thus referred to the earlier section for more information. The third page is used to define the optional label, and the minimum and maximum values, possibly by means of a complete data format. This page functions as was previously discussed for integer data tags, and you are referred to that section for further details.

The Alarm Viewer Object

The *Alarm Viewer* object is used to provide the operator with a method to view and accept active alarms. Differing color pairs are used to show the various alarm states. Additional data about the alarms may be displayed if required.

If you use manual accept alarms in your system, you should provide a page that contains an alarm viewer to make sure the operator can accept these alarms. You may wish to consider creating a popup page and using it to display the alarm viewer, although the size restrictions on popups may cause you to reject this idea. The properties of the alarm viewer are displayed on four tabbed pages, the first of which is shown below.



Alarm Viewer Properties

List Format

Font: Swiss 12x16

List Colors

No Active Alarms: Black and Lime TEXT

Active Alarm - Use Priority Colors: No

Active Alarm: Red and White TEXT

Accepted Alarm: Black and Yellow TEXT

Background: White

List Contents

Show End Markers: Yes

Show Alarm Time: Yes

No Active Alarms Text: No Active Alarms

Display State

Show Item: General TRUE

- The *Font* property is used to select the font to be used to draw the object. A fixed pitch font should ideally be used to ensure that the various data fields remain in the correct alignment.
- The *List Colors* properties are used to define the foreground and background colors used to display each alarm state. The default values should be acceptable for most applications. The selection “Active Alarm – Use Priority Colors” can be

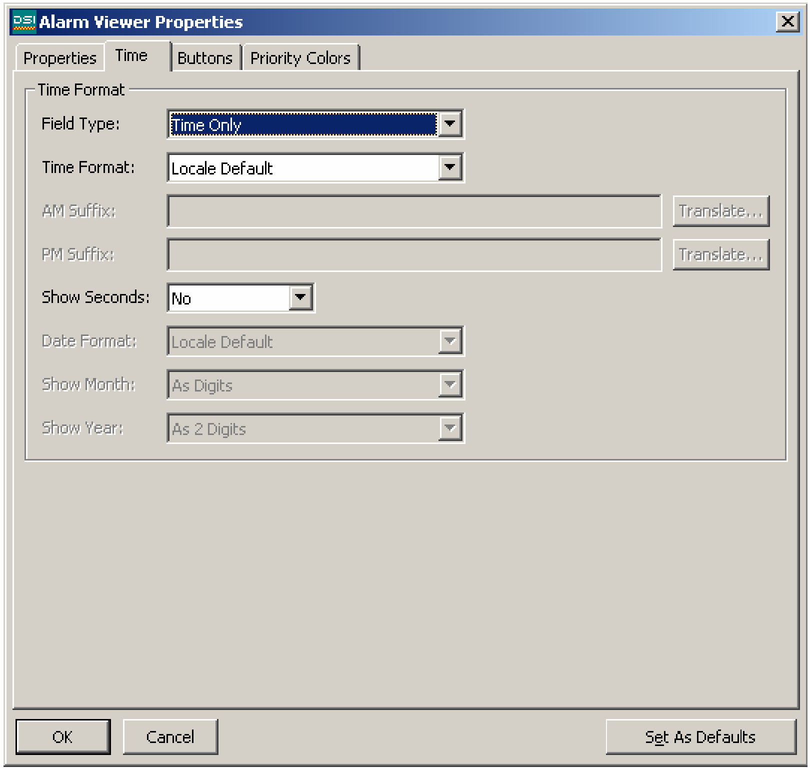
set to Yes, in which case the "Active Alarm" color selection below it will be disabled, and color selection based on the tag priority will be enabled on the Priority Colors page.

- The *Show End Markers* property is used to indicate whether to display a column that contains markers showing the beginning and end of the alarm list. If this column is omitted, the object will take less space, but it will be harder for the operator to determine the limits of the list.

- The *Show Alarm Time* property is used to indicate whether or not the time at which the alarm occurred should be included in the object. If the time is displayed, the second tab is used to define the format to be used.

- The *No Active Alarms Text* property is used to override the default text that is displayed when no alarms are present, or to enter localized versions of this text on systems that support multiple languages.

The second page of properties is used to define the format of the alarm time. The properties are as defined for the time and date object.



Alarm Viewer Properties

Time Format

Field Type: Time Only

Time Format: Locale Default

AM Suffix:

PM Suffix:

Show Seconds: No

Date Format: Locale Default

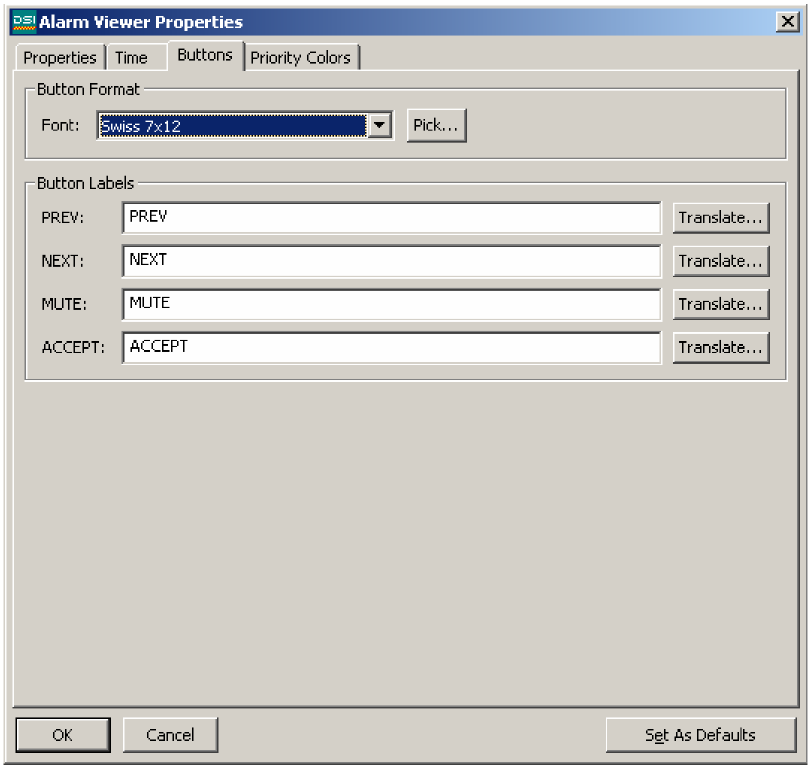
Show Month: As Digits

Show Year: As 2 Digits

The third page of the properties is used to control how the object's buttons are labeled.

- The *Font* property is used to select the required font.

- The *Button Labels* properties are used to override the default label that is shown on each of the four buttons, or to enter localized versions of this text on systems that support multiple languages. Setting all four of the text items to a single * will disable the buttons, and blank that area on the display.



Alarm Viewer Properties

Properties | Time | Buttons | Priority Colors

Button Format

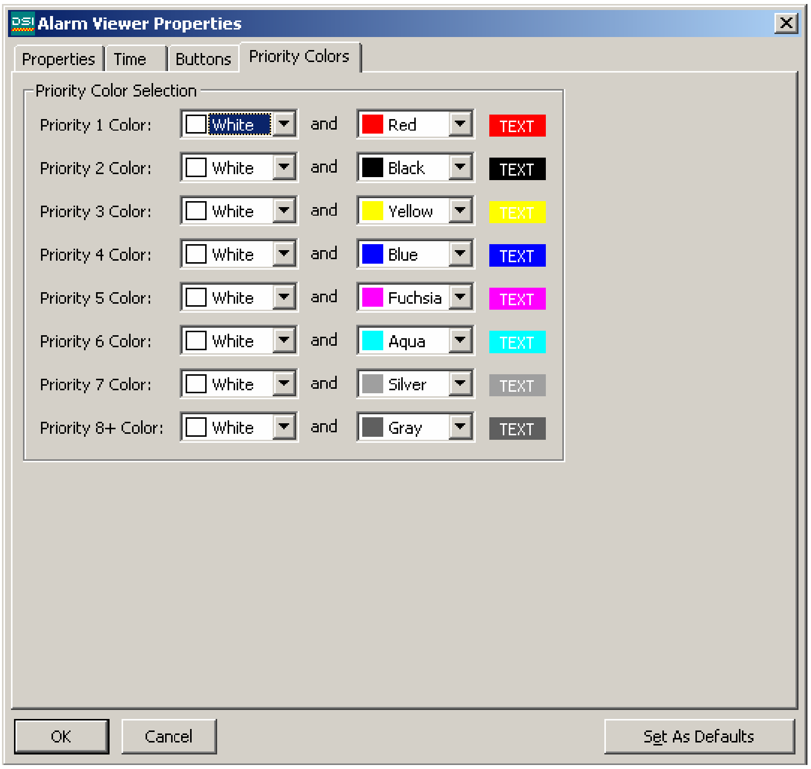
Font: Swiss 7x12 Pick...

Button Labels

PREV:	PREV	Translate...
NEXT:	NEXT	Translate...
MUTE:	MUTE	Translate...
ACCEPT:	ACCEPT	Translate...

OK Cancel Set As Defaults

The final page of the properties is used to select the colors of the alarm text when "Active Alarm – Use Priority Colors" is set to Yes on the Properties page.



Alarm Viewer Properties

Properties | Time | Buttons | Priority Colors

Priority Color Selection

Priority 1 Color:	White	and	Red	TEXT
Priority 2 Color:	White	and	Black	TEXT
Priority 3 Color:	White	and	Yellow	TEXT
Priority 4 Color:	White	and	Blue	TEXT
Priority 5 Color:	White	and	Fuchsia	TEXT
Priority 6 Color:	White	and	Aqua	TEXT
Priority 7 Color:	White	and	Silver	TEXT
Priority 8+ Color:	White	and	Gray	TEXT

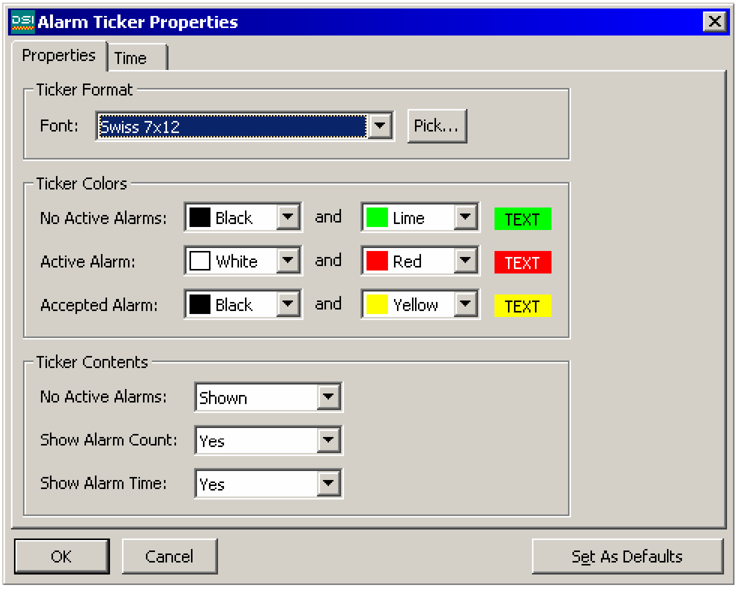
OK Cancel Set As Defaults

Up to eight pairs of colors may be assigned to tags with priorities 1 through 8. A priority value greater than 8 will use the setting for priority 8.

The Alarm Ticker Object

The *Alarm Ticker* object scrolls through the active alarms in the system. It takes up a single line, and the whole of the display width. It does not allow the operator to accept the alarms.

The properties of the alarm ticker are displayed on two tabbed pages. The first of these is shown below:



Ticker Format

Font: Swiss 7x12

Ticker Colors

No Active Alarms:	■ Black	and	🟩 Lime	TEXT
Active Alarm:	□ White	and	🟥 Red	TEXT
Accepted Alarm:	■ Black	and	🟨 Yellow	TEXT

Ticker Contents

No Active Alarms: Shown

Show Alarm Count: Yes

Show Alarm Time: Yes

- The *Font* property is used to select the font to be used to draw the object. A fixed pitch font should ideally be used to ensure that the various data fields remain in the correct alignment.

- The *Ticker Colors* properties are used to define the foreground and background colors used to display each alarm state. The default values should be acceptable for most applications.

- The *No Active Alarms* property is used to indicate whether a message should be displayed when no alarms are present, or whether the bar should be left blank in these circumstances.

- The *Show Alarm Count* property is used to indicate whether the number of currently active alarms should be displayed in the object. Unless display space is restricted, showing this field typically improves operator readability.

- The *Show Alarm Time* property is used to indicate whether or not the time at which the alarm occurred should be included in the object. If the time is displayed, the second tab is used to define the format to be used.

The second tab is the time format, and is as described for the time and date object.

The Event Viewer Object

The *Event Viewer* object is used to provide the operator with a method to view the events recorded in the system's event log. It will always take up the whole of the display width, but can be restricted to less than the full height if required.

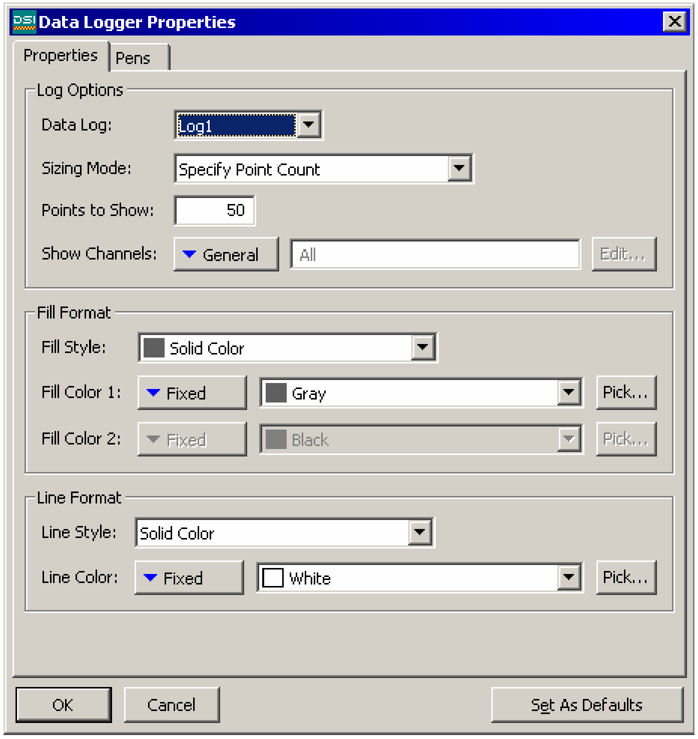
The properties of this object are essentially the same as those for the alarm viewer. You are thus referred to the earlier section for more details. The only additional property is *Show Event Type*, which is used to indicate whether or not each row should be labeled with the kind of event that resulted in the log entry. The possible event types are alarm activations, alarm acceptances, alarm deactivations and event activations.

The Trending Objects

The *Data Logger* object provides a fixed view of the data contained within a data logger. The number of data points to be displayed may be defined, and channels may be shown or hidden using a bit-mask.

The *Trend Viewer* object provides a more advanced interactive view of a data logger, allowing the operator to zoom in, zoom out, and to scroll backwards and forwards through historical data that is saved in the logger's history buffer.

The data logger is configure via a single property page, shown below:



Data Logger Properties

Properties | Pens

Log Options

Data Log: Log1

Sizing Mode: Specify Point Count

Points to Show: 50

Show Channels: General | All

Fill Format

Fill Style: Solid Color

Fill Color 1: Fixed | Gray

Fill Color 2: Fixed | Black

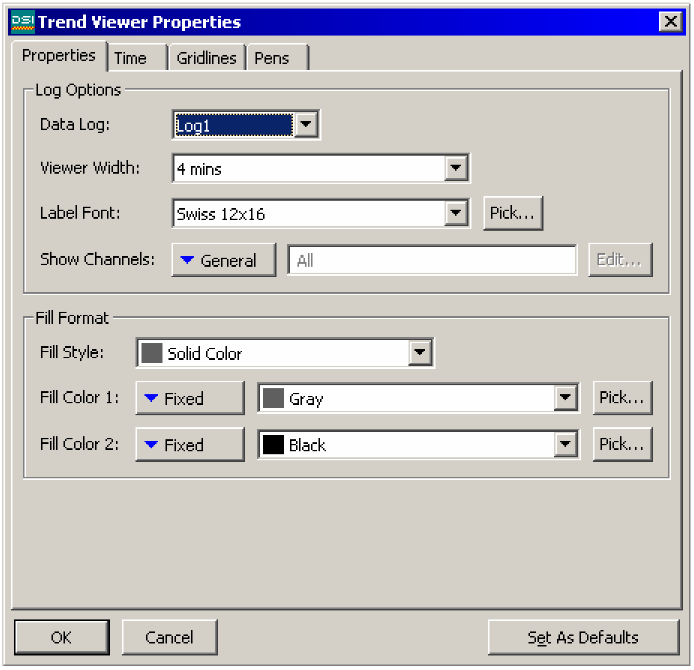
Line Format

Line Style: Solid Color

Line Color: Fixed | White

- The *Data Log* property is used to select the data log to be displayed.
- The *Sizing Mode* property is used to indicate whether you wish to specify the number of data points to be displayed, or whether you want the software to display one data point for each horizontal pixel of the object. The *Points to Show* property is used to specify the number of points to be displayed when the Sizing Mode is so configured.
- The *Show Channels* property is used to specify an optional integer value that controls which channels are displayed. If an expression is entered, channel *n* will be shown if and only if the n^{th} bit of the value is set. The n^{th} bit is defined as the bit having the weight of 2^n, such that the lowest-order bit is bit 0.
- The *Fill Format* properties are used to define the background of the object.
- The *Line Format* properties are used to specify the format of the optional border around the object. Note that these properties do no change the pens used to draw the actual channel data: the colors of these lines are defined by the system.

The trend viewer object is similar, but has a second page that is used to specify the time and date format that will be used to indicate the extremities of the displayed data. The first page of the properties is shown below:



Log Options

Data Log: Log1

Viewer Width: 4 mins

Label Font: Swiss 12x16 Pick...

Show Channels: General All Edit...

Fill Format

Fill Style: Solid Color

Fill Color 1: Fixed Gray Pick...

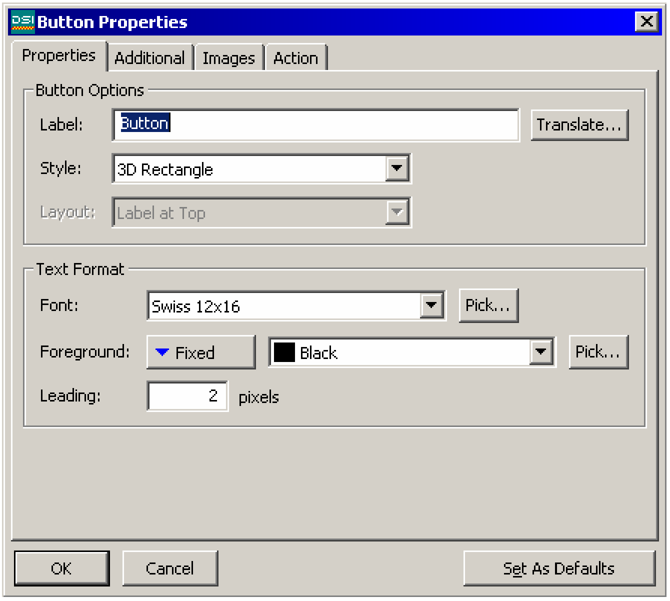
Fill Color 2: Fixed Black Pick...

- The *Data Log* property is used to select the data log to be displayed. If you want the operator to be able to scroll backwards through historical data, be sure to enable the log's history buffer. Refer to the Data Logging chapter for details.
- The *Viewer Width* property is used to define the default amount of data to be shown when the object is first displayed. Note that the operator can zoom in or out as required, and may thus choose to show more or less data.
- The *Label Font* property is used to define the font used to draw the various labels that adorn the object. The default font will typically be too large for applications where the object does not take up the entire screen.
- The *Show Channels* property is as defined for the data logger object.
- The *Data Gridlines* properties define the major and minor divisions to be used to label the vertical axis of the viewer. Note that each tag that is displayed is scaled according to its own format properties, and that different tags may thus have different scaling. You ideally should define gridlines that make sense for all tags that are to be shown, and ensure that you label the display page to let the operator know what scaling you have selected.
- The *Show Divisions* property is used to indicate whether gridlines should be drawn for the time axis. The major and minor divisions to be used are chosen by the system according to the current zoom level.

The General Button Object

The *General Button* object displays an animated button that can respond to user input. Several different button styles are provided, including one that uses custom images from the software's image library.

The properties of the general button are defined using four tabs. The first of these tabs is shown below.



Button Options

Label: Button

Style: 3D Rectangle

Layout: Label at Top

Text Format

Font: Swiss 12x16

Foreground: Fixed Black

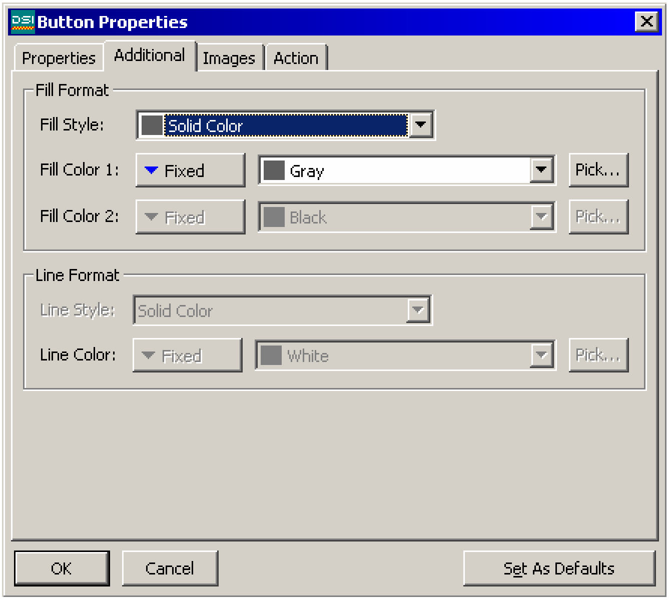
Leading: 2 pixels

- The *Label* property is used to define the text to be shown on the button.
- The *Style* property is used to define the style of button to be displayed.

Style	Description
Round	A round button comprising two concentric circles.
Flat Rectangle	A rectangular button comprising two nested rectangles.
3D Rectangle	A rectangular button drawn using 3D coloring effects.
3D Rectangle w. Bevel	A rectangular button with more pronounced 3D effects.
Custom Images	A button based upon two custom images.

- The *Layout* property is used to indicate where, if anywhere, the label should be placed when using custom images to define the button's appearance.
- The *Text Format* properties are used to define the label font and coloring.

The second page is used to define additional formatting information:



Fill Format

Fill Style: Solid Color

Fill Color 1: Fixed Gray Pick...

Fill Color 2: Fixed Black Pick...

Line Format

Line Style: Solid Color

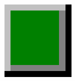
Line Color: Fixed White Pick...

- The *Fill Format* properties are used to define the color and pattern to be used to fill the interior of the button. For 3D buttons, this color is used to draw the button face, while the system uses various shades of grey to draw the border so as to get the 3D effect. For buttons based on custom images, the fill format defines the background that is to be drawn underneath the images.

- The *Line Format* properties are used to define the style and color of the lines that make up the outlines of the Round and Flat Rectangle styles. The properties are not used when other styles are selected.

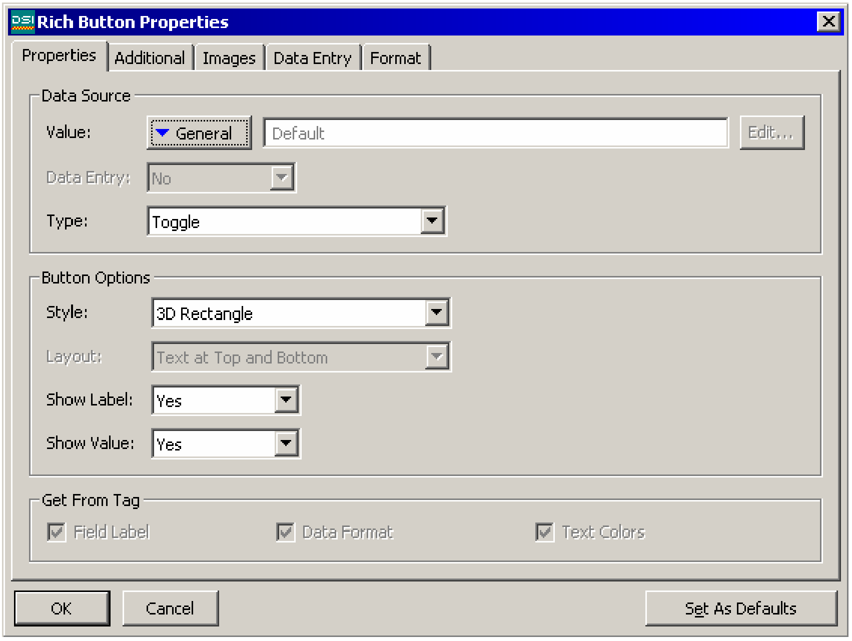
The third tab is used for buttons of the Custom Image style to define the images to be shown when the button is in the release and pressed states. Image selection is described in detail under the picture object, and you are thus referred to those sections. The fourth tab is used to define the action to be performed by the button. You are referred to the earlier section of this chapter for more details.

The Rich Button Object



The *Rich Button* object displays an animated button that is used to control the state of a flag tag. While the same functionality can be achieved using a general button, the rich version automatically obtains data from the underlying tag.

The properties of the rich button are defined using five tabs. The first of these tabs is shown below:



Data Source

Value: General Default

Data Entry: No

Type: Toggle

Button Options

Style: 3D Rectangle

Layout: Text at Top and Bottom

Show Label: Yes

Show Value: Yes

Get From Tag

☑ Field Label ☑ Data Format ☑ Text Colors

- The *Value* property is used to indicate from where the data for this object should be obtained. You may select a tag, a register in a communications device, or an expression that combines a number of such items. The data type of the item must be an integer or a flag.

- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. Selecting Local will enable data entry, but prevent access via the virtual panel facility of the web server. For data entry to be enabled, the expression entered for the value property must be capable of being changed. For example, if a formula is entered, data entry will not be permitted. Most buttons will have data entry enabled.

- The *Type* defines the action to be taken when the button is pressed and released.

Button Type	The Button Will…
Toggle	Change the data state when the object is pressed.
Momentary	Set the data to 1 when the object is pressed. Set the data to 0 when the object is released.
Turn On	Set the data to 1 when the object is pressed.
Turn Off	Set the data to 0 when the object is pressed.

- The *Style* property is as defined for general buttons.

- The *Layout* property is used to define where the optional label and data values are to be placed relative to the button itself when custom images are used. The text fields are always placed within the button for the other button styles.

- The *Show Label* property is used to indicate whether or not you want the object to include a label to identify the data being displayed and controlled.
- The *Show Data* property is used to indicate whether or not the object should include the associated data value. For buttons configured with a type of toggle or momentary, the displayed data value is the value of the underlying tag. For other buttons, the displayed value is the value to which the tag will be set.
- The *Get From Tag* properties are used to indicate from where the label text, the field format and the text colors should be obtained. The options presented depend on what was entered for the Value property. In each case, you may manually enter the data in the appropriate properties, or, assuming a suitable expression has been defined, you may instruct the object to get the required information from the underlying data tag.

The second page defines additional formatting information, and is as described for the general button object. The third page is used to define the custom images to be used to reflect the button states, and is as described for the general button, except that different images can be specified depending on whether the underlying tag is on or off. You should again refer to the picture object for information on selecting images. The fourth page defines a number of properties specific to data entry. These are as defined for the flag tag text object, and you should refer to that section for details. The fifth and final page defines the label and format to be used for the object, and is as defined for flag tags.

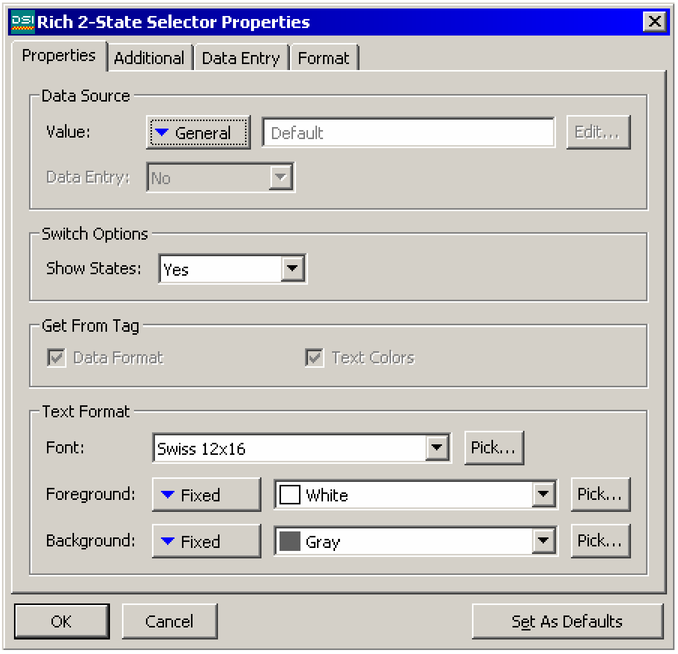
The Selector Objects

The *Rich 2-State Selector* object displays a rotary-style switch that can be used to turn-on and turn-off a flag tag. As with all rich objects, most of the configuration data can be obtained from the underlying tag.

The *Rich Multi-State Selector* object displays a rotary-style switch that can be used to turn on and turn off a multi tag. As with all rich objects, most of the configuration data can be obtained from the underlying tag.

Each of these objects displays a circular selector switch within the area used to define the object. If the object is tall enough that the circular switch has sufficient space above it, labels can be added to the object to allow the various states to be identified.

Both objects are configured using four tabbed pages, the first of which is shown below:



Rich 2-State Selector Properties

Properties | Additional | Data Entry | Format

Data Source

Value: General Default Edit...

Data Entry: No

Switch Options

Show States: Yes

Get From Tag

☑ Data Format ☑ Text Colors

Text Format

Font: Swiss 12x16 Pick...

Foreground: Fixed □ White Pick...

Background: Fixed ■ Gray Pick...

OK Cancel Set As Defaults

- The *Value* property is used to indicate from where the data for this object should be obtained. You may select a tag, a register in a communications device, or an expression that combines a number of such items. The data type of the item must be appropriate to the object in question eg. the Value property for a multi-state selector object cannot be set equal to a string expression.

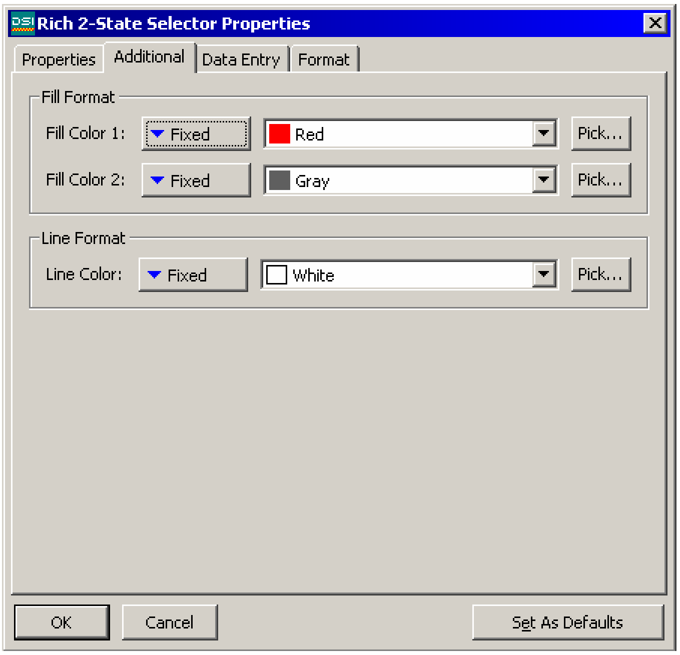
- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. Selecting Local will enable data entry, but prevent access via the virtual panel facility of the web server. For data entry to be enabled, the expression entered for the value property must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.

- The *Show States* property is used to indicate whether or not you want the object to attempt to label each of the possible states of the tag. The states will only be shown if sufficient space exists at the top of the object. It is also important to select a small enough font to avoid overlapping text.

- The *Get From Tag* properties are used to indicate from where the data format and the associated text colors should be obtained. The options presented depend on what was entered for the Value property. In each case, you may manually enter the data in the appropriate properties, or, assuming a suitable expression has been defined, you may instruct the object to get the required information from the underlying data tag.

- The *Text Format* properties are used to select the font and text colors for the state labels. If the object is configured to obtain its text colors from the underlying tag, the color fields will be disabled.

The second tab contains additional formatting information.



Rich 2-State Selector Properties

Fill Format

Fill Color 1: Fixed — 🟥 Red

Fill Color 2: Fixed — ⬛ Gray

Line Format

Line Color: Fixed — □ White

- The *Fill Color 1* property is used to define the color of the rectangular portion of the selector that moves in order to indicate the tag state. The *Fill Color 2* property is used to define the color of the rest of the object.

- The *Line Format* property is used to define the color of the various lines that are used to draw the object. These include the circle around the selector, and the four lines that define the rectangle within the object.

The fourth page defines a number of properties specific to data entry. These are as defined for the tag text objects, and you should refer to that section for details. The fourth page defines the label and format to be used for the object, and is as defined for flag tags or multi-state tags, depending on which type of selector is being configured. You are once again referred to the chapter of Tags for information on the various formatting options.

The Picture Object

The *Picture* object is used to display one of a number of images, based upon an optional data value. The images may be manipulated in various ways, and may be moved within the object according to internal or external data values.

The object provides exhaustive facilities for displaying BMPs, JPGs, or WMF image files from DSI8000's extensive image library or from third-party clipart providers. Five separate tabbed pages control the various options.

The first tab is used to select the images to be displayed.

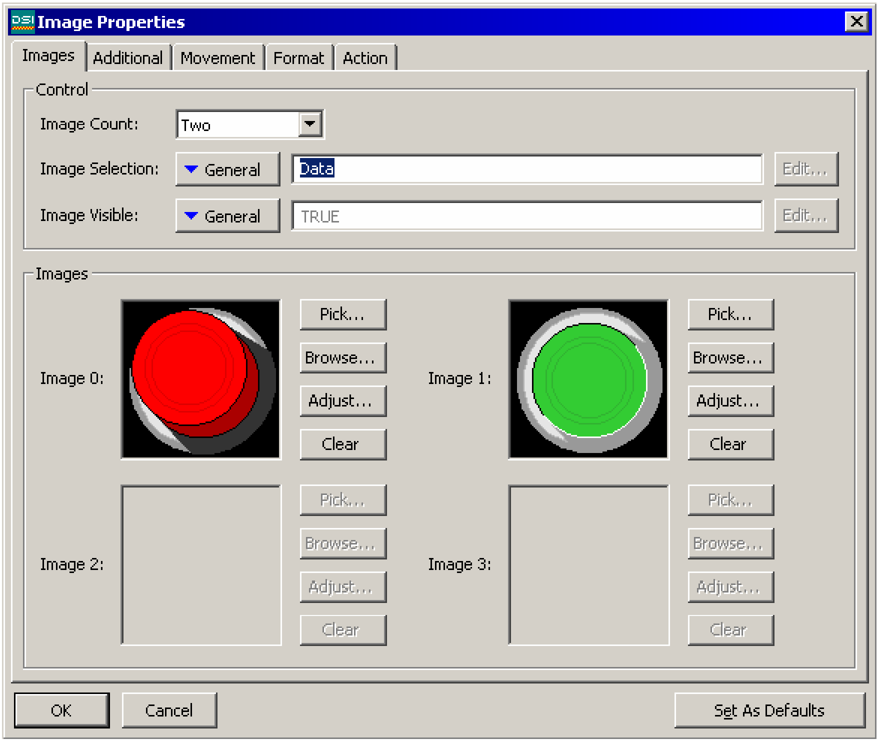


Image Properties

Control

Image Count: Two

Image Selection: General Data

Image Visible: General TRUE

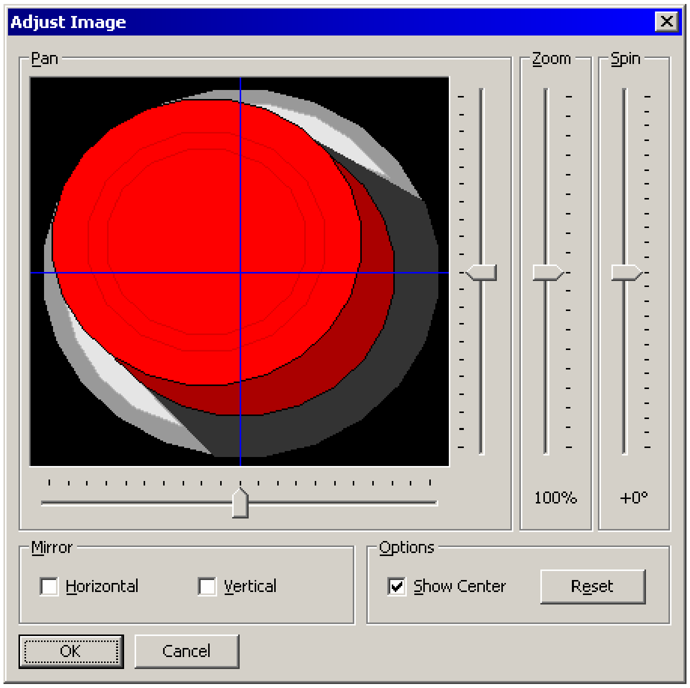
- The *Image Count* property is used to indicate how many different images should be displayed by this object. Up to ten different images can be shown.

- The *Image Selection* property is a numeric value used to select between the various images if an Image Count of greater than one has been configured. A value of zero will display Image 0 and so on.

- The *Image Visible* property is a true-or-false value used to hide or show the selected image. If you want the image to be hidden, you must not select No Fill for the Background when defining the background format.

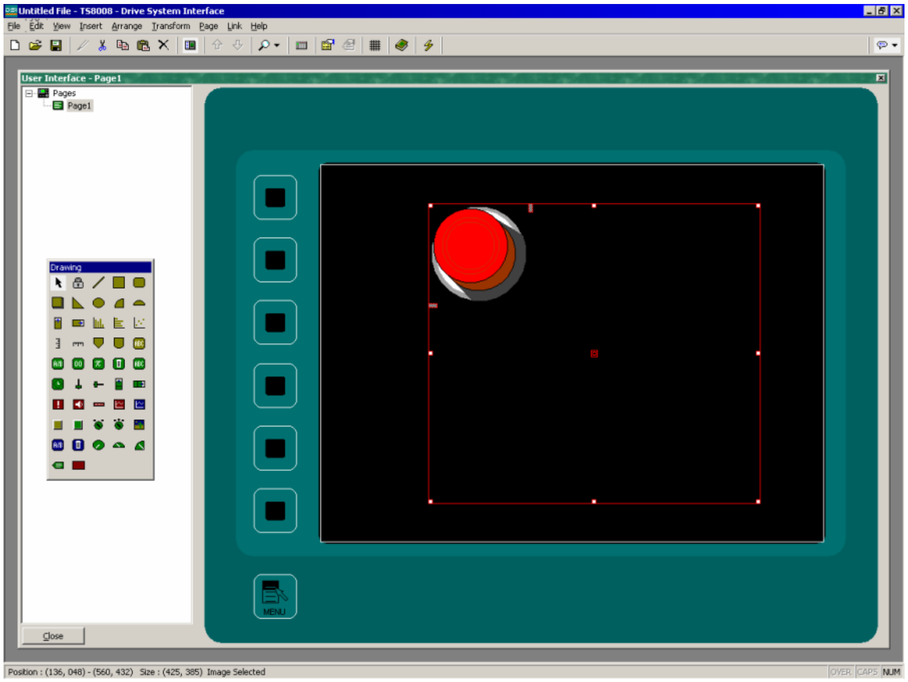
- The *Image* properties define each particular image. The Pick button next to each image can be used to launch the image library to allow an image to be selected; the Browse button can be used to open a file containing a BMP, a JPG or a WMF image file; the Clear button can be used to remove the image; and the Adjust button can be used to edit the image as discussed below.

If you use one of the Adjust buttons to manipulate an image, you will first be warned about the problems you will encounter if you then try to download a database containing manipulated images using earlier versions of the Windows operating systems. Assuming you are happy to rule-out the use of such earlier releases, the following dialog box will appear:



The various sliders can be used to pan, zoom and spin the image, while the checkboxes can be used to mirror it horizontally or vertically. The Show Center checkbox shows or hides the blue lines that mark the center of the image, while the Reset button can be used to restore the image to its original state. The various manipulation options are typically used to modify an image in order to create various different states for use in animation.

The second tab of the Picture object's properties contains any additional images that could not be displayed on the first page. It is only required when the Image Count is set to a value greater than four. The third tab controls movement of the image within the object. To enable this facility, drag either or both of the shaded rectangular handles in the top-left corner of the object so as to define a sub-region in which the image should reside.


Untitled File - TS8008 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
MENU
Close
OVER CAPS NUM

Position : (136, 048) - (560, 432) Size : (425, 385) Image Selected

The now-reduced image can be moved within the whole object by defining values to control its horizontal and vertical position. These values are defined together with minimum and maximum limits that specify the values corresponding to the extremes of the image's movement within the object's bounding rectangle.

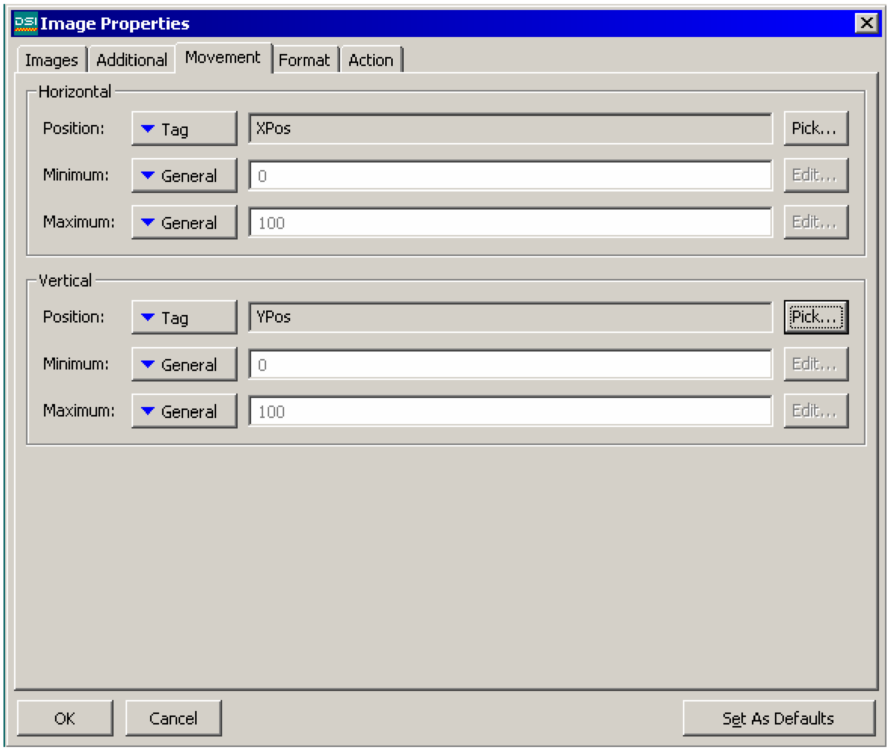


Image Properties
Images Additional Movement Format Action
OK Cancel Set As Defaults

Horizontal

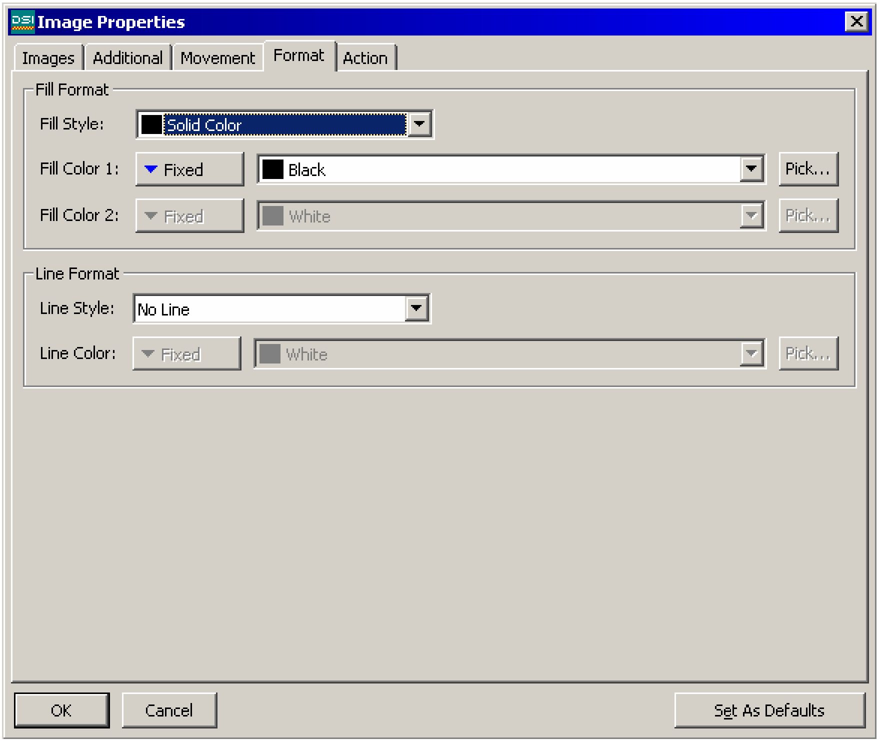
Position:	Tag	XPos	Pick...
Minimum:	General	0	Edit...
Maximum:	General	100	Edit...

Vertical

Position:	Tag	YPos	Pick...
Minimum:	General	0	Edit...
Maximum:	General	100	Edit...

In this example, setting **XPos** to 0 will place the image at the left of the object, while setting it to 100 will place it at the far right. Similar behavior in respect of the up and down limits of the object can be obtained via the value stored in **YPos**.

The fourth tab of the object's properties are used to control basic formatting.



Fill Format
Fill Style: Solid Color
Fill Color 1: Fixed Black
Fill Color 2: Fixed White

Line Format
Line Style: No Line
Line Color: Fixed White

- The *Fill Format* properties are used to define the background pattern for the object. Note that if you want to animate the object in any way, you should specify some sort of background color so that the system can erase old images.
- The *Line Format* properties are used to define the object's outline style.

The final tab of the object's properties is used to define an optional action that can be triggered when the operator touches the image. You are referred to the earlier section on assigning actions to objects for more details of how to configure this functionality.

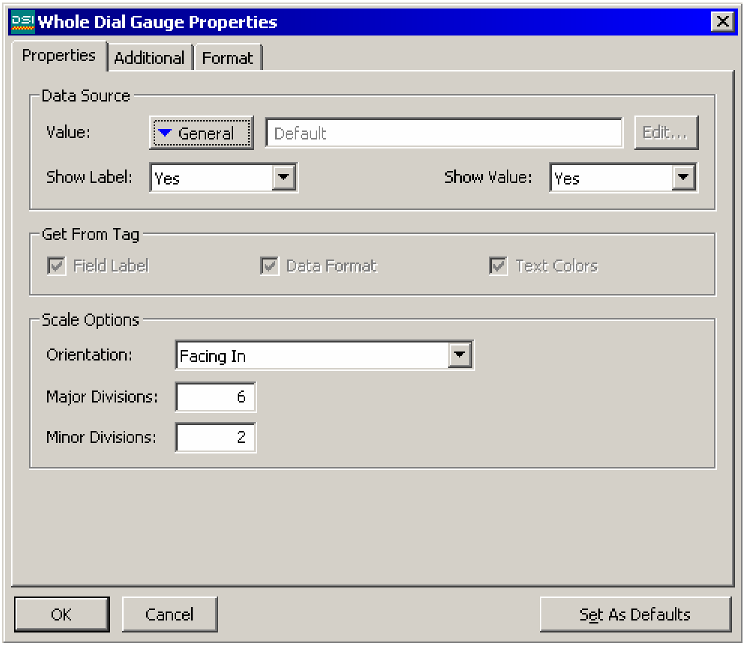
The Dial Gauge Objects

The *Full Dial Gauge* object displays an integer value as a pointer with a 270° swing within a full circle. The object can optionally display the data value, and the associated data label. The number of scale divisions can also be defined.

The *Half Dial Gauge* object functions like the full dial gauge, but displays a pointer with a 180° swing within a half-circle. Other options remain the same.

The *Quarter Dial Gauge* object functions like the full dial gauge, but displays a pointer with a 90° swing within a quarter-circle. The other formatting and display options remain the same.

Just as with other rich objects, the dial gauge objects are capable of deriving much of the required formatting information from the tag used as their controlling value. Just as with tag text objects, multiple tabbed pages are used to edit the objects' properties. The first of these pages is shown below.


Whole Dial Gauge Properties

Data Source

Value: General | Default | Edit...

Show Label: Yes

Show Value: Yes

Get From Tag

☑ Field Label ☑ Data Format ☑ Text Colors

Scale Options

Orientation: Facing In

Major Divisions: 6

Minor Divisions: 2

- The *Value* property is used to define the value to be displayed.

- The *Show Label* property is used to indicate whether a label should be included with the gauge. The label is displayed in the center of the object, above the optional value. If a tag is used for the value property, the label may be obtained from that tag. Otherwise, it must be entered on the Format tab of the dialog box.

- The *Show Value* property is used to indicate whether the value of the data should be displayed within the gauge. If a tag of the appropriate data type is used for the value property, the format may be obtained from the tag. Otherwise, as with the label, it must be entered on the Format tab.

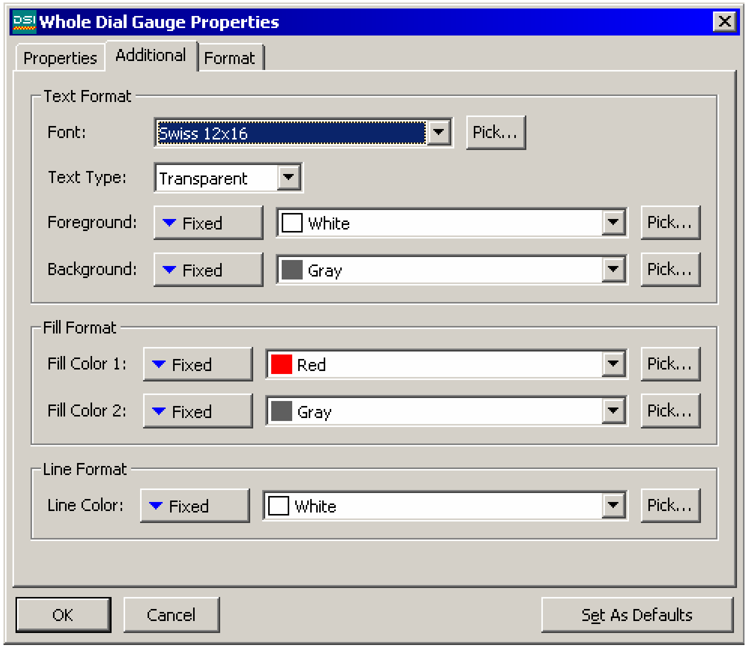
- The *Get From Tag* properties are used to indicate from where the label text, the field format and the text colors should be obtained. The options presented depend on what was entered for the Value property. In each case, you may manually enter the data in the appropriate properties, or, assuming a suitable expression has been defined, you may instruct the object to get the required information from the underlying data tag.

- The *Orientation* property is used to indicate the direction in which the scale's minor tick marks should point. The major tick-marks always point inwards.

- The *Major Divisions* property is used to indicate into how many major divisions the scale should be divided. Large tick marks are drawn at each division. The lowest number of major divisions is one, in which case large tick-marks will be drawn at the extremes of the scale, but not along its arc.

- The *Minor Divisions* property is used to indicate into how many minor divisions each major division should be divided. Smaller tick-marks are drawn at each division. Selecting a value of one for this property will disable minor divisions.

The second page defines the formatting options:



Text Format
Font: Swiss 12x16
Text Type: Transparent
Foreground: Fixed White
Background: Fixed Gray

Fill Format
Fill Color 1: Fixed Red
Fill Color 2: Fixed Gray

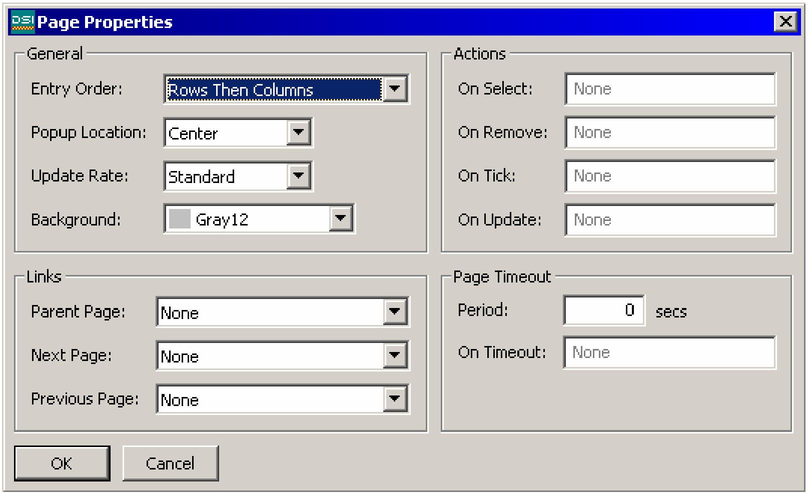
Line Format
Line Color: Fixed White

- The *Text Format* properties are used to define the font to be used to display the optional data value and data label, and the colors to be used for the text. In addition, a property is provided to define whether the font should be opaque, or whether the pointer should be visible through the text.
- The *Fill Format* properties are used to define the background color of the dial, and the color to be used to draw the interior of the pointer and the scale tick marks. Fill Color 1 defines the background; Fill Color 2 the other items.
- The *Line Format* property is used to define the color of the lines used to demark the pointer, the scale tick-marks and the outline of the dial gauge itself.

The third page is used to define the optional label, and the minimum and maximum values, and the data format for the optional data value. This page functions as was previously described for integer data tags, and you are referred to that section for further details.

Defining Page Properties

Each page has a number of properties that can be accessed via the Page menu:



Page Properties

General

Entry Order: Rows Then Columns

Popup Location: Center

Update Rate: Standard

Background: Gray12

Actions

On Select: None

On Remove: None

On Tick: None

On Update: None

Links

Parent Page: None

Next Page: None

Previous Page: None

Page Timeout

Period: 0 secs

On Timeout: None

- The *Entry Order* property is used to define how the cursor on the operator panel will move between data entry fields. The setting determines whether fields organized in a grid will be entered in row or column order.

- The *Popup Location* property is used to define the location of popup windows or the popup keypad when this display page is visible. You may wish to adjust this property to keep the popups away from important data items.

- The *Update Rate* property is used to define how frequently items on the display are updated. As update rates increase in frequency, overall communications performance of the operator interface panel may decrease. This selection should be left at the default setting when possible.

- The *Background* property is used to define the background color of the display page. Note that the background cannot be animated, as a change in the color would force the whole page to redraw, thereby impairing performance.

- The *On Select* and *On Remove* properties are used to define actions to be performed when the page is first selected for display, or when the page is removed from the display. Refer to the Writing Actions section and the Function Reference for a list of supported actions. Refer to the Data Availability section in this chapter for details of a timeout than can occur when using these properties.

- The *On Tick* property is used to define an action that will run every second during the period for which this page is displayed. Refer to the Writing Actions section and the Function Reference for a list of supported actions. If a lack of data availability results in this action being unable to execute, it will be skipped and retried one second later.

- The *On Update* property is use to define an action that will be run each time the page is redrawn. Refer to the Writing Actions section and the Function Reference for a list of supported actions. If a lack of data availability results in this action being unable to execute, it will be skipped and retried on the next update. You should note that you can severely reduce system performance by performing complex actions on every display update. You should also note that in many cases, actions that you may think need to be run on each update can be performed using triggers, or by using mapping blocks.
- The *Parent Page* property is used to indicate the page to be displayed when the panel's **Exit** key is pressed while this page is active. Selection of this page can be overridden using the techniques below.
- The *Next Page* property is used to indicate the page to be displayed when the panel's **Next** key is pressed while this page is active, and when the cursor is on the last data entry field on the page. This selection can also be overridden.
- The *Previous Page* property is used to indicate the page to be displayed when the panel's **Prev** key is pressed while this page is active, and when the cursor is on the first data entry field on the page. This selection can also be overridden.
- The *Period* is the time in seconds to wait before performing the action specified.
- *On Timeout* is the action to be performed when the period of time has expired.

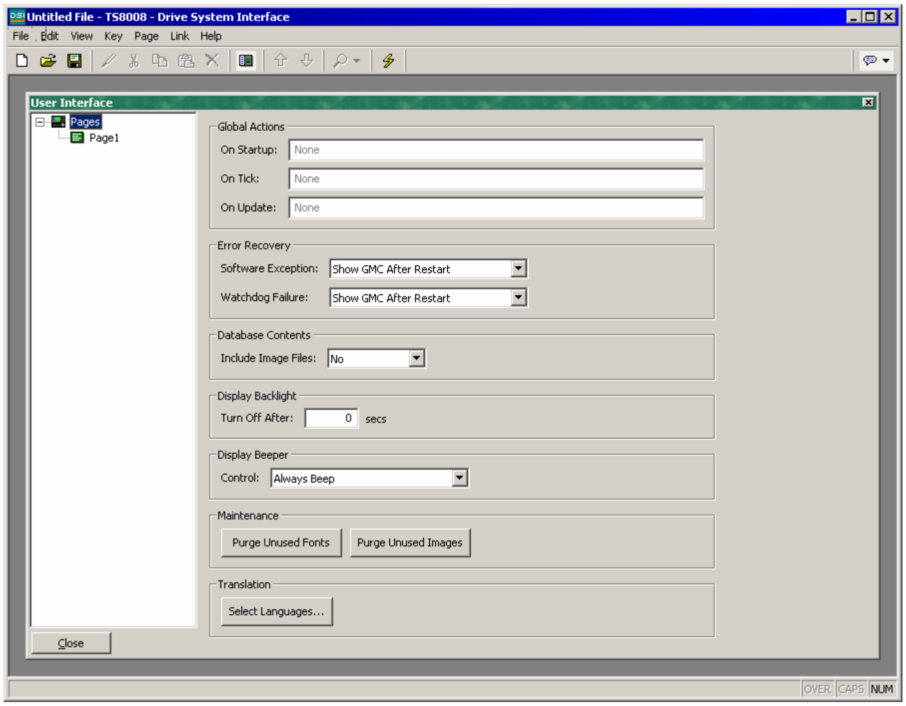
If you have too many data entry fields to fit on a single page, the Next Page and Previous Page properties can be used to link together a series of pages to allow the operator to edit the fields in sequence. DSI will automatically position the cursor appropriately, such that if the **Prev** key is pressed on the first field of a page, the previous page will be activated with the cursor on the last field of that page.

Defining System Actions

In addition to the various actions that can be defined via page properties, DSI gives you the ability to define an action to be run when the system first starts, and an action to be run once a second, no matter which page is displayed. These actions can be accessed by selecting the Pages icon in the left-hand pane of the User Interface window.

Additional System Properties

In addition to the system actions described above, the property page, accessed by selecting the Pages icon, gives you access to a number of other system-wide parameters.



Global Actions

On Startup: None

On Tick: None

On Update: None

Error Recovery

Software Exception: Show GMC After Restart

Watchdog Failure: Show GMC After Restart

Database Contents

Include Image Files: No

Display Backlight

Turn Off After: 0 secs

Display Beeper

Control: Always Beep

Maintenance

Translation

- The *Error Recovery* properties are used to define the behavior of the system when it encounters a software problem, or when the display fails to update for a long period of time as a result of a coding error. By default, the system will reset and display a so-called Guru Meditation Code to help the development engineers in tracking-down the problem. You may disable the display of this code to allow the system to recover more quickly and without user intervention.

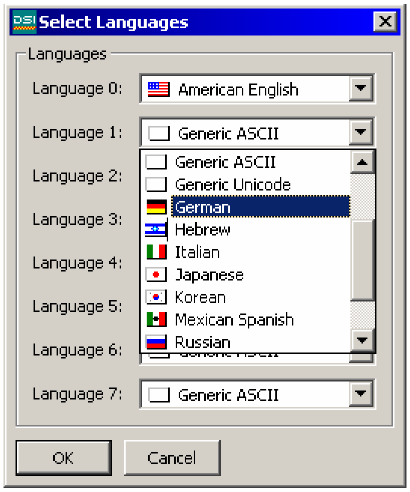
- The *Include Image Files* property is used indicate that DSI8000 should save within the database file a copy of any disk-based images that have been imported into the project. By default, the filename alone of the image is stored, thereby requiring you to have the images available on disk whenever you are working on the project. If you wish to create a single file that contains all the required data for the project, enable this option. Note that databases that contain image files will typically be very large, and may prevent upload support from working.

- The *Turn Backlight Off* property is used to configure the system so as to disable the display backlight after the specified number of seconds of inactivity. This facility can be used to extend the life of the backlight components. The operator can reactivate the backlight by pressing a key, or touching an active area of the touch screen. The key-press or touch is ignored until the backlight is lit.

- The *Maintenance* buttons can be used to remove unused fonts or images from the database, thereby reducing file size and lowering memory usage. You should typically use these options before releasing a database for use in the field, as they will remove most of the detritus that accumulates during development.

- The *Select Languages* button is described below.

Selecting Languages



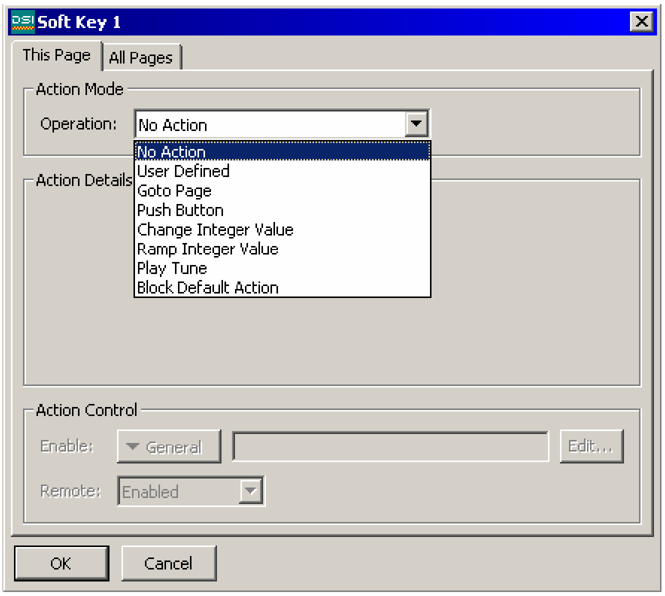
Language 0: American English
Language 1: Generic ASCII
Language 2:
Language 3:
Language 4:
Language 5:
Language 6:
Language 7: Generic ASCII

To select the various languages to be supported within your database, select the Pages icon within the User Interface window, and press the Select Languages button to display the shown dialog box. Up to eight different languages can be defined, each of which can be chosen from a drop down list. DSI8000 will re-configure Windows to use the appropriate Input Method Editor whenever a complex (ie. Unicode-based) language is being edited. In order to use this facility, you should ensure that you have the required language support installed by referring to the appropriate Windows documentation. If you wish to use a Unicode-based language that is not included in the drop down list, select Generic Unicode mode instead. You will then be able to enter any Unicode characters, although you will have to manually select the appropriate keyboard mode or Input Method Editor.

Changing The Language

To configure a key or an object to change the language displayed by the operator panel, select User Defined mode and enter SetLanguage(n) as the On Pressed property, where n is a number between 1 and 8, according to the language to be displayed. The display page will be redrawn in the selected language, with any text for which translations have been entered - including fixed text, tag labels and tag formatting information - adjusted as appropriate. Pages that are subsequently displayed will also be drawn in the selected language.

Defining Key Behavior



Operation: No Action
Enable: General
Remote: Enabled

In addition to defining actions that occur when objects are touched, you may define actions to be executed when keys are pressed. To do this, first change the zoom level so that the required key can be seen. Then, double-click the key to produce the following dialog box:

You will note that this dialog box has two tabbed pages. The first page is used to define what will happen when the key in question is pressed when the current page is selected. The second page is used to define what will happen if the key is pressed when any page is selected. The first type of action is called a *local* action, while the second type is called a *global* action. The color used to display the key will change according to which actions are defined:

F1 If the key is displayed in PURPLE, a local action is defined for this PAGE.

F1 If the key is displayed in GREEN, a GLOBAL action is defined.

F1 If the key is displayed in BLUE, local and global actions are BOTH defined.

Once you have defined an action, you can right-click on the key and use the resulting menu to select either Make Global or Make Local to change the action type. These options will not be available if both types of action have already been defined.

Blocking Default Actions

When defining keys actions, you may use the Block Default Action selection as a place-holder to prevent further processing. As an example, suppose you have configured F1 to perform a global action, but want to prevent this action from being invoked on a particular page. By configuring F1 on that page as Block Default Action, the global action will not occur.

Data Availability

DSI's communications infrastructure reads only those data items which are required for the current page. This means that when a page is first selected, certain data items may not be available. For a display object, this is no problem, as the object simply displays an undefined state (typically a number of dashes) until the data becomes available. For actions, though, things can get more complex.

For example, suppose a local action increases the speed of a motor by 50 rpm. If the motor speed is not referenced on the previously displayed page, then, when the page is first displayed, DSI will not know the current speed, and will thus be unable to write the new value. To handle this, if the operator attempts to perform an action for which the required data is not available, the TS8000 panel will display a "NOT READY" message until the key in question is released. The operator must then wait a short while, and try the operation again. In practice, communications updates normally take place quickly enough that even the most nimble-fingered operator will be hard pressed to get the message to appear, but since it may on occasions be seen, it is worth explaining.

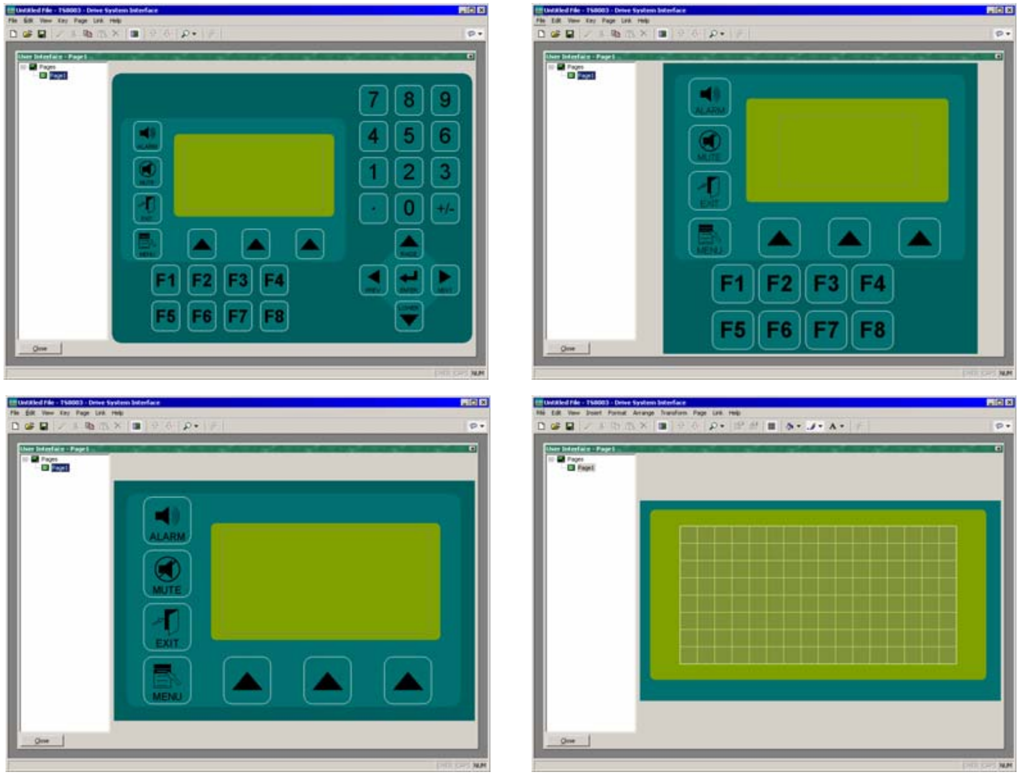
A slightly more complex issue comes about if the action defined by a page's On Select property is unable to proceed because it also finds that required data is not available. Here, DSI will wait up to thirty seconds for the data to arrive. If it does not, the action will not be performed, and a "TIMEOUT" message will be displayed for the operator. This timeout mechanism is required to avoid problems should a communications link become severed.

Configuring The User Interface (Pushbutton Displays)

Now that you have configured your communications, and created data tags for the items that you wish to display, you can create display pages to allow the user to view or edit these data items. These pages are manipulated by selecting the User Interface icon from the main screen. Please note that this chapter refers specifically to the monochrome pushbutton operator panel (TS8003). If you are using a TS8006, TS8008, or TS8010 operator panel with a color display, please refer to the previous chapter for configuration details.

Controlling The View

By default, the User Interface window shows the entire front panel of the TS8003, including the display and all the available keys. If you want to allocate more of your PC's screen to show the TS8003's display, you can use the four different zoom levels as shown below:



As you can see, at each level, fewer keys are shown, and more of the window is allocated to the display itself. The zoom level can be controlled from the View menu, by using the magnifying glass icon, or by pressing the Alt key together with the digits 1 through 4.

Other View Options

As well as controlling the zoom, the View menu contains the following options:

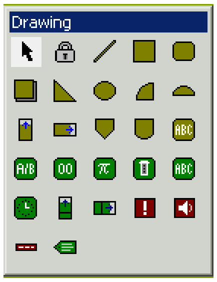
- The *Page List* command can be used to show or hide the left-hand pane of the User Interface window. If the page list is disabled, even more space is made available for editing the display. The **F4** key toggles the page list on and off.
- The *Hold Aspect* command can be used to control whether or not DSI attempts to maintain the aspect ratio of the display. If aspect holding is enabled, a figure that would appear as, say, a circle on the TS8003 will appear as a perfect circle on your PC. If this mode is not selected, DSI can expand the display page to use more of the PC's screen, but at the expense of some distortion.

Other options are available during page editing, and are described below.

Display Editor Toolboxes

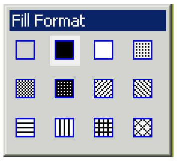
To edit the contents of a display page, first select the page as described above. Then, click on the green rectangle that represents the TS8003's display. A white rectangle will appear around the display to indicate that it has been selected, and a number of toolboxes will appear.

The Drawing Toolbox



The drawing toolbox is used to add various elements, known as objects, to the display page. The first two icons control the insertion mode, while the balance of the icons represent individual objects. The objects shown in yellow are basic geometric and animation items, while the ones shown in green are rich objects that use formatting and other information from a data tag to control their operation. The objects shown in red are system items, such as the active alarm viewer. Objects displayed in blue are enhanced version of other objects that have been recently added to the software. All of the commands contained in the toolbox can also be accessed via the Insert menu.

The Fill Format Toolbox

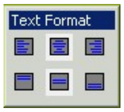


The fill format toolbox is used to control the pattern that will be used to fill a display object. If one or more objects are selected, clicking on a fill pattern will change all the selected items to use that pattern. If nothing is currently selected, clicking on a pattern will set the default pattern for newly-created objects. The various options can also be accessed via the Format menu, or via the paint-can icon on the toolbar.

The Line Format Toolbox

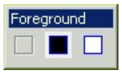
The line format toolbox is used to control the color that will be used to draw an outline around a display object. If one or more objects are selected, clicking on a line color will change all the selected items to use that color. If nothing is currently selected, clicking on a color will set the default outline color for newly-created objects. The various options can also be accessed via the Format menu, or via the paintbrush icon on the toolbar.

The Text Format Toolbox



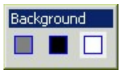
The text format toolbox is used to control the horizontal and vertical alignment of objects which contain text elements. If one or more such objects are selected, clicking on an icon will change all the selected items to use the selected justification. If nothing is currently selected, clicking on an icon will set the default format for newly-created objects. The various options can also be accessed via the Format menu.

The Foreground Toolbox



The foreground toolbox is used to control the foreground color for objects which contain text elements. If one or more such objects are selected, clicking on an icon will change all the selected items to use the selected color. If nothing is currently selected, clicking on an icon will set the default color for all newly-created objects. The various options can also be accessed via the Format menu.

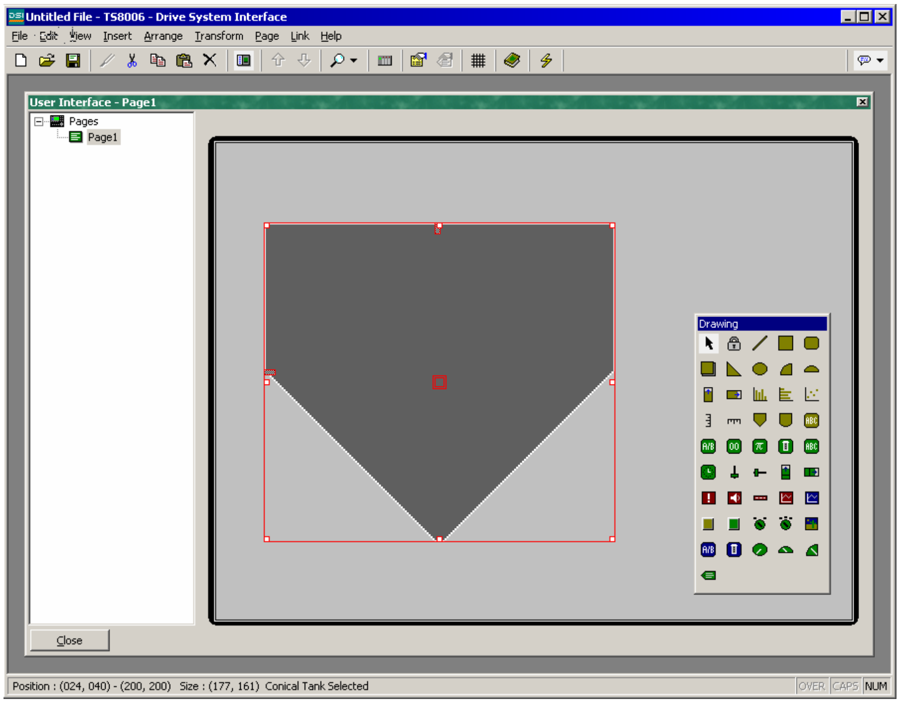
The Background Toolbox



The background toolbox is used to control the background color for objects which contain text elements. If one or more such objects are selected, clicking on an icon will change all the selected items to use the selected color. If nothing is currently selected, clicking on an icon will set the default color for all newly-created objects. The various options can also be accessed via the Format menu.

Adding Display Objects

To add a display object to a page, click on the required icon in the drawing toolbox, or select the required option from the Insert menu. The mouse cursor will change to an arrow with a crosshair at its base, and you will then be able to drag-out the required position of the object within the display window.


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
Close
OVER CAPS NUM

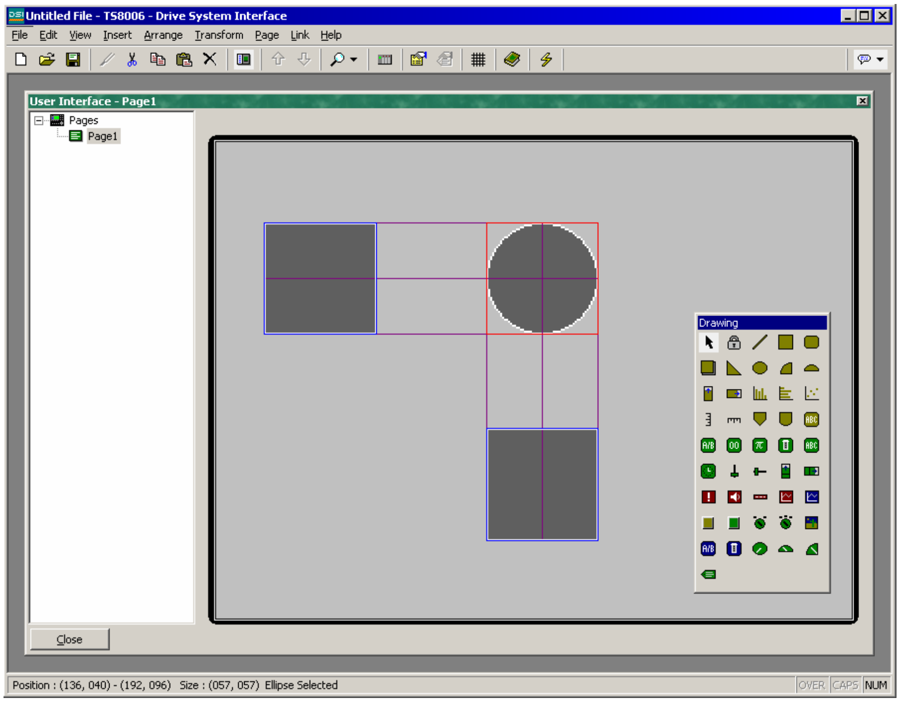
Position : (024, 040) - (200, 200) Size : (177, 161) Conical Tank Selected

Smart Alignment

If you have the Smart Align features of the View menu enabled, DSI will provide you with guidelines to help align a new object with existing objects, or with the center of the display. In the example shown above, the horizontal dotted line indicates that the center of the tank object is vertically aligned with the center of the display. With a little practice, this feature can make it very easy to align objects as they are created, without the need to go back and "tweak" your display pages to get the various figures into alignment.

In the Smart Align example shown below, a newly-created ellipse is being aligned with two existing rectangles. Guidelines are present at both the edges of the figures, and at the center, showing that both the edges and the centers are aligned. The red rectangle is highlighting the newly-created object, while the blue rectangles are highlighting the objects to which the guidelines have been drawn.

Smart Align is also enabled when objects are moved or re-sized.


Untitled File - TS8006 - Drive System Interface
File Edit View Insert Arrange Transform Page Link Help
User Interface - Page1
Pages
Page1
Drawing
Close
OVER CAPS NUM

Position : (136, 040) - (192, 096) Size : (057, 057) Ellipse Selected

Keyboard Options

While creating a display object, the following keyboard options are available:

- Holding down the Shift key while dragging-out the object will cause the object to be drawn such that it is centered on the initial mouse position, with one of its corners defined by the current mouse position. This is useful for drawing symmetrical figures centered on an initial point.
- Holding down the Ctrl key while dragging-out the object will keep its horizontal and vertical sizes the same. This is useful when you want to be sure that you draw an exact circle or square using the ellipse or rectangle objects.

These options are also active when objects are re-sized.

Lock Insert Mode

The padlock icon on the drawing toolbox can be used to add a number of objects of the same basic type without having to click the toolbox icon for each item in turn. To cancel lock mode, click the padlock icon again, or press the Escape key. The same operation can be performed by using the Lock Mode command on the Insert menu.

Selecting Objects

To select a display object, simply move your mouse pointer over the object in question, and perform a left-click. You will notice that while your pointer is hovering over an object, a bounding rectangle is drawn in blue to help show what will be selected. When the actual selection is performed, the rectangle will change to red, and handles will appear, so as to allow you to re-size the object as required. If you find that the object you want to select is hidden below another object, press the Alt key to allow the selection to be made.

To select several objects, either drag-out a selection rectangle around the objects you want to select, or select each object in turn, holding down the Shift key to indicate that you want each object to be added to the selection. If multiple objects are selected, the red rectangle will surround all of the objects, and the handles can then be used to resize the objects as a group. The relative size and position of the objects will be maintained, as long as DSI can do so without violating minimum size requirements.

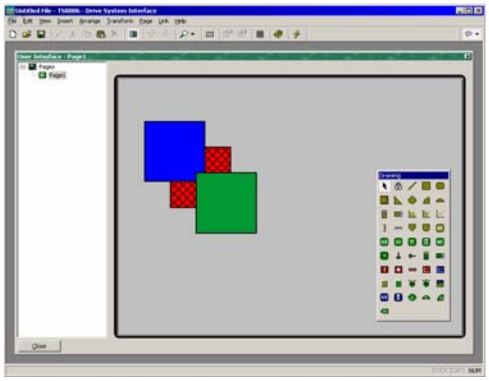
Moving And Resizing

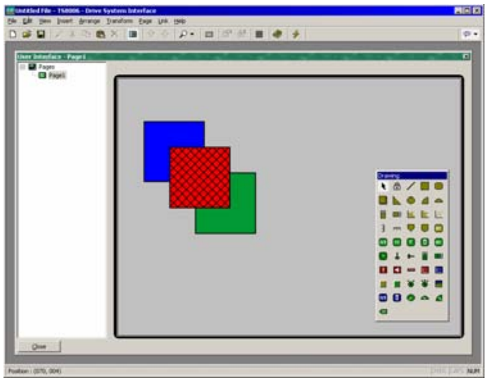
Objects can be moved by first selecting them, and then by dragging them to the required position on the display page. If Smart Align is turned on, guidelines will appear to help you align the objects with other items on the page. Holding down Ctrl while moving an object will leave a copy of the object in its original position, thereby allowing duplicates to be created. You can also use the cursor keys to "nudge" the current selection a single pixel in the required direction. Holding down Ctrl while nudging will increase the movement of the objects by a factor of eight.

Objects can be resized by selecting them, and then by dragging the appropriate handle to the required position. Once again, if Smart Align is turned on, guidelines will appear to help you align the objects with other items on the page. The Shift and Ctrl keys can be used to modify the resize behavior as described in the Adding Display Objects section. Note that DSI will always constrain resizing operations so as to ensure that objects stay on the screen, and to make sure that items do not exceed their maximum permitted size, or shrink below the minimum size appropriate to their format.

Reordering Objects

Objects on a display page are stored in what is known as a "z-order." This defines the sequence in which the objects are drawn, and therefore whether or not a given object appears to be in front of or behind another object. In the first example below, the hatched square is shown behind the solid squares i.e. at the bottom of the z-order. In the second example, it has been moved to the front of the order, and appears in front of the other figures.





To move items in the z-order, select the items, and then use the various commands on the Arrange menu. The Move Forward and Move Backward commands move the selection one step in the indicated direction, while the Move To Front and Move To Back commands move the selection to the indicated end of the z-order. Alternatively, if you have a mouse that is equipped with a wheel, the wheel can be used to move the selection. Scrolling up moves the selection to the back of the z-order; scrolling down moves the selection to the front.

Editing Objects

In addition to the above, objects can be edited in various ways:

- The various clipboard commands on the Edit menu (Cut, Copy and Paste), or the corresponding toolbar icons, can be used to duplicate items or move them around on a page or between pages. The Duplicate command can be used to perform a Copy operation, immediately followed by a Paste operation. Note that when a Paste is performed, DSI will offset the newly-pasted item if it will exactly overlay an item of the same type.
- The various formatting properties (fill pattern, outline color, text justification and so on) can be changed by selecting a object, and then either clicking the various buttons in the appropriate toolboxes or by using the associated commands on the Format menu. If multiple objects have been selected, DSI will apply the changes to all selected objects.
- The more detailed properties of an object can be edited by double-clicking the object, or by using the Properties command on the Edit menu. A dialog box will be displayed, allowing all of the objects to be accessed. The properties associated with each object will be described below.

Object Descriptions

The sections below describe each object found in the drawing toolbox.

The Line Object

The *Line* object is a line drawn between two points. Its only property is the style of line to be used. In addition to the solid colors shown on the line toolbox, a number of dotted styles can also be accessed via the properties dialog box.

The Simple Geometric Objects

The *Rectangle* object is a rectangle with a defined outline and fill pattern. The fill pattern may be set to No Fill to draw the outline alone, or the outline may be set to None to draw a figure without a border.

The *Round Rectangle* object is similar to the rectangle, but has rounded corners. When the object is selected, an additional handle appears, allowing the radius of the corners to be edited by dragging the handle from side to side.

The *Shadow* object is similar to the rectangle, but with a drop-shadow located to the bottom right of the figure. The object is often drawn with no fill pattern, so as to allow it to act as a frame around text objects.

The *Wedge* object is a right-angled triangle located within one quadrant of a bounding rectangle. In addition to the outline and fill properties, the wedge has a property to indicate which quadrant it should occupy.

The *Ellipse* object is an ellipse with a defined outline and fill pattern. The fill pattern may be set to No Fill to draw the outline alone, or the outline may be set to None to draw a figure without a border.

The *Ellipse Quadrant* object is one quadrant of an ellipse. In addition to the outline and fill properties, the ellipse quadrant has a property to indicate which quadrant it should occupy.

The *Ellipse Half* object is one half of an ellipse. In addition to the outline and fill properties, the ellipse half has a property to indicate which of the four possible halves will be drawn.

The properties for these objects need little further explanation, other than to point out that the quadrant or half rendered by the Wedge, Ellipse Quadrant or Ellipse Half objects can also be edited via the command found on the Transform menu.

The Tank Objects

The *Conical Tank* object is a conical tank with a defined outline and fill pattern. When the object is selected, additional handles appear, allowing the exact shape of the tank to be modified by dragging the handles as required.

The *Round Bottomed Tank* object is a tank with a defined outline and fill pattern. When the object is selected, an additional handle appears, allowing the exact shape of the tank to be modified by dragging the handle as required.

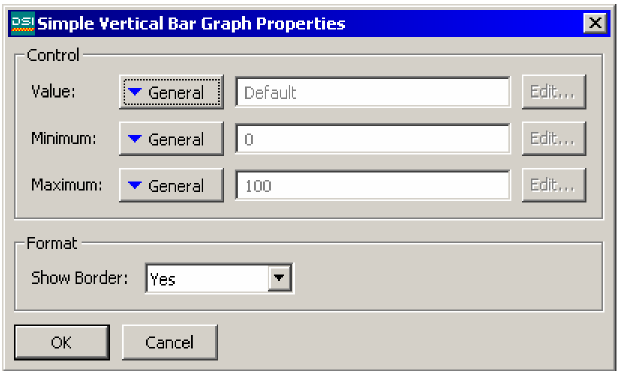
The properties for these objects need little further explanation.

The Simple Bar Objects

The *Simple Vertical Bar* object allows an expression to be drawn as a vertical bar-graph between specified minimum and maximum values. An additional property allows the object's border to be displayed or hidden.

The *Simple Horizontal Bar* object allows an expression to be drawn as a horizontal bar-graph between specified minimum and maximum values. An additional property allows the object's border to be displayed or hidden.

The properties are accessed by double-clicking the object:



Simple Vertical Bar Graph Properties

Control

Value: General Default

Minimum: General 0

Maximum: General 100

Format

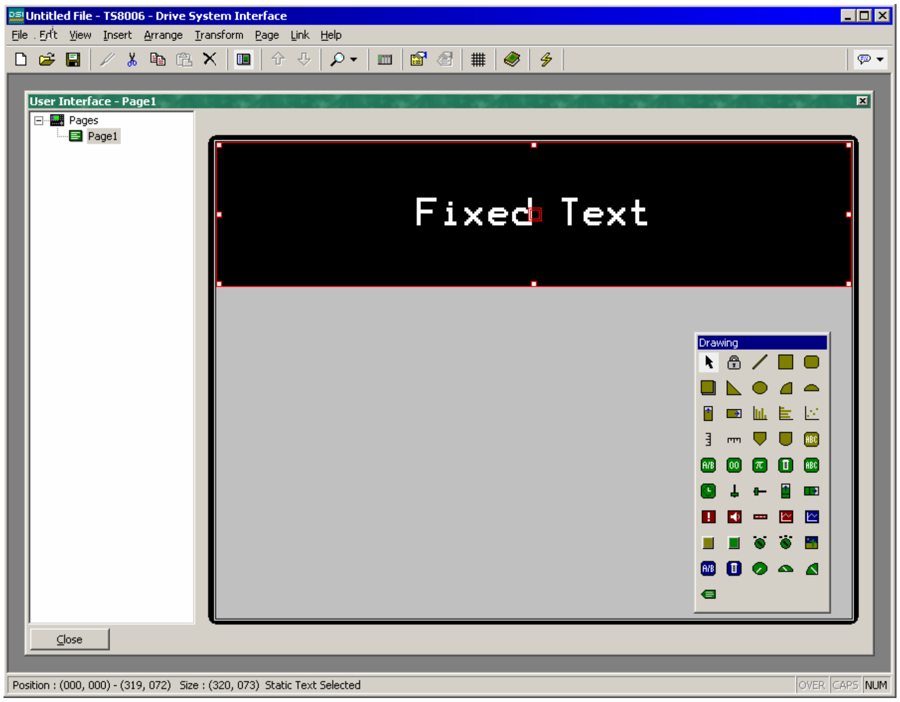
Show Border: Yes

- The *Value* property is used to specify the value to be displayed. In the example given above, the object is configured to display the level of a tank.
- The *Minimum* and *Maximum* properties are used to specify the range of values to be shown. In the example above, a range of 0 to 1000 is specified.
- The *Show Border* property is used to display or hide the object's border.

The Fixed Text Object

The *Fixed Text* object is used to add unchanging text to a page. The text is displayed in a specified font and color, and with a specified justification. The text can also be translated for international applications.

When the text is created, a cursor will appear, allowing the text to be entered:



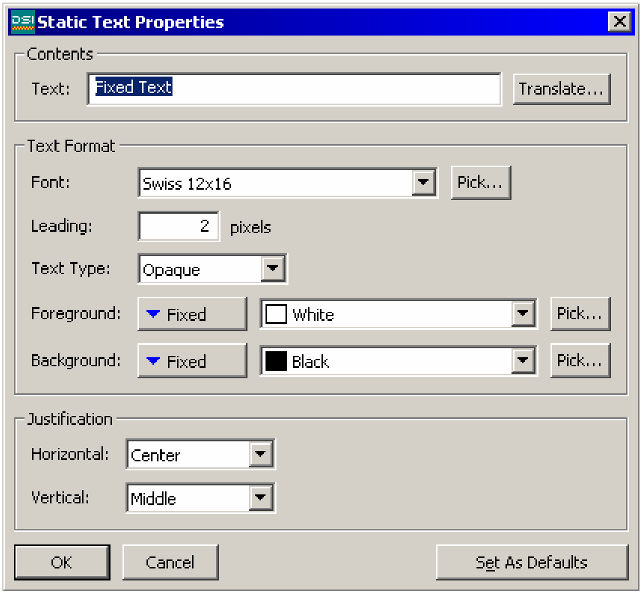
Untitled File - TS8006 - Drive System Interface

User Interface - Page1

Fixed Text

Position : (000, 000) - (319, 072) Size : (320, 073) Static Text Selected

Only the US English text can be edited directly. The international versions of the text must be edited via the properties dialog box, which is accessed by selecting the object and pressing **Alt+Enter**, or by selecting the Properties command from the Edit menu.



Static Text Properties

Contents

Text: Fixed Text

Text Format

Font: Swiss 12x16

Leading: 2 pixels

Text Type: Opaque

Foreground: Fixed White

Background: Fixed Black

Justification

Horizontal: Center

Vertical: Middle

- The *Text* property is used to specify the text to be displayed. As mentioned above, the US English version of the text can also be edited directly on the display page when the object is created, or by clicking an existing object.

- The *Font* property is used to specify the font to be used. This property can also be edited by using the font button on the toolbar, or by using the Format menu.

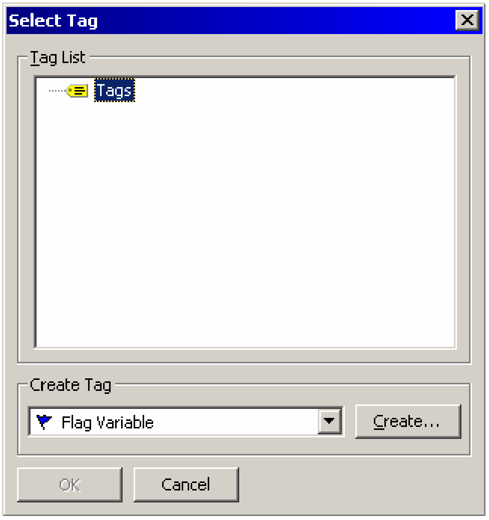
- The *Foreground* and *Background* properties are used to specify the colors to be used to draw the text. Obviously, having the same color for both settings will render the text unreadable. Selecting None for the background will create transparent text, allowing underlying objects to be seen through the letters.

- The *Horizontal* and *Vertical* justification properties are used to indicate where the text should be placed within the bounding rectangle of the object. These properties can also be edited via the associated toolbox, or via the Format menu.

The Auto Tag Object

The *Auto Tag* object allows you to select a tag, and then automatically place the appropriate text object on the display. For example, selecting an integer tag will allow insertion of an appropriately-configured integer text object.

This is the icon you will use most often for adding tags to a page. It first displays the dialog box shown below to allow tag selection, and then creates one of the five tag text objects described in the next section. The new object will be configured so as to display the tag in question using its label and its formatting properties, as defined when the tag was created.



The Tag Text Objects

The tag text objects are used to display or edit an expression in textual form. Primarily, they are used to display tags, in which case the default format is taken from the Format tab associated with that tag in the Data Tags window. If a non-tag expression is entered, or if you want the formatting to differ from the default values for a tag, the format data can be overridden as required. There is one type of tag text for each tag family.

The *Flag Text* object is used to display a true or false condition.

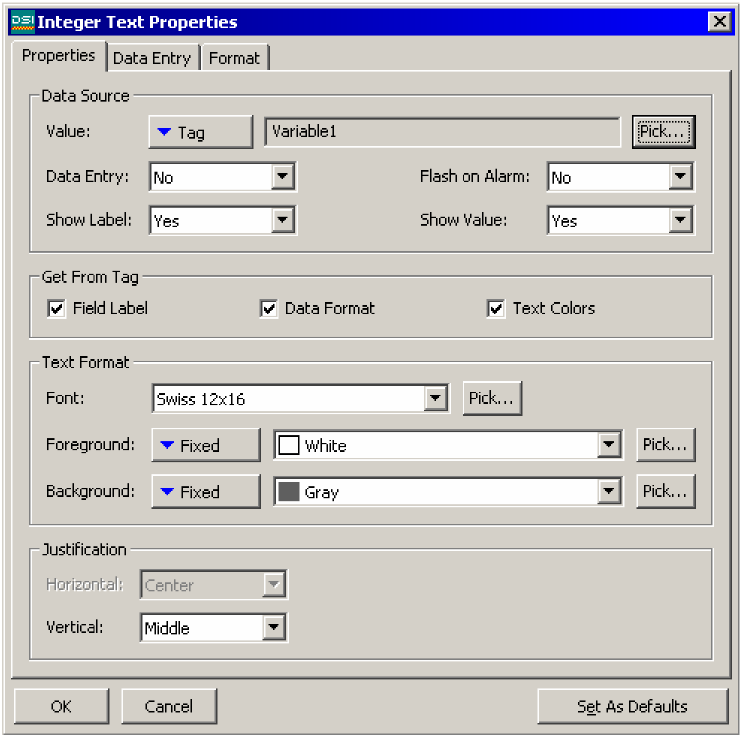
The *Integer Text* object is used to display an integer expression.

The *Real Text* object is used to display a floating-point expression.

The *Multi Text* object is used to display a multi-state condition.

The *String Text* object is used to display a string expression.

The properties of a tag text object are displayed using two tabbed pages. The first page is more-or-less the same for all five object types:



Data Source

Value: Tag Variable1

Data Entry: No

Flash on Alarm: No

Show Label: Yes

Show Value: Yes

Get From Tag

☑ Field Label ☑ Data Format ☑ Text Colors

Text Format

Font: Swiss 12x16

Foreground: Fixed □ White

Background: Fixed ■ Gray

Justification

Horizontal: Center

Vertical: Middle

- The *Value* property is used to indicate from where the data for this object should be obtained. You may select a tag, a register in a communications device, or an expression which combines a number of such items. The data type of the item must be appropriate to the object in question (the Value property for an integer text object cannot be set equal to a string expression).

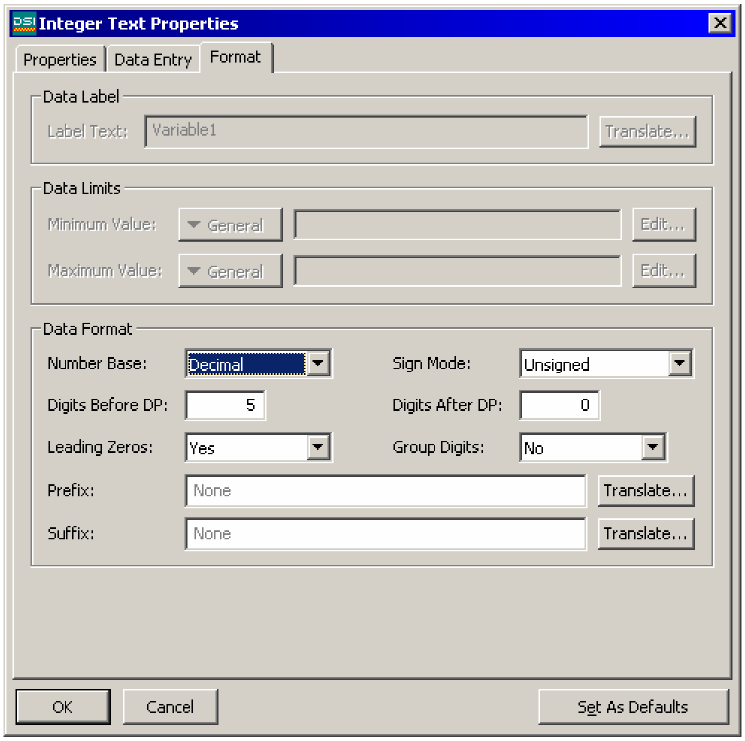
- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. For data entry to be enabled, the expression entered for the value property must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.

- The *Show Label* property is used to indicate whether or not you want the object to include a label to identify the data being displayed. If this property is set to yes, the label will be left-justified within the object's bounding rectangle, while the data itself will be right-justified. If this property is set to no, the Horizontal Justification property will be used to locate the data within the field. Note that this property can be edited via the Field Label commands on the Format menu. When no object is selected, these commands can also be used to set the default value for newly-created objects.

- The *Get Label* property is used to indicate from where the label text should be obtained. The options presented depend on what was entered for the value property. If a tag has been selected, you will be given the option of using the tag's default label, or entering a new label on the Format tab of the dialog box. If something else has been selected, you will only have the second option.

- The *Get Format* property is used to indicate from where the formatting information for this object should be obtained. The options presented depend on what was entered for the value property. If a tag of the correct data type has been selected, you will be given the option of using the tag's default formatting, or entering modified information on the Format tab of the dialog box. If something else has been selected, you will only have the second option.
- The *Flash on Alarm* property is used to indicate whether or not you want the text on the TS8003's display to flash if the tag entered in the value property is currently in an alarm state. This property is not available for string text objects, or for those objects which have a non-tag value defined for the value property.
- The balance of the properties control the font, colors and justification to be used when drawing the object. These properties require no further explanation.

The second page varies according to the object in question, and displays the same information as the Format tab of the associated tag type. Different sections of the page will be enabled according to the settings provided for the Get Label and Get Format properties. The example below shows the Format tab for an integer text object:



Label Text: Variable1

Minimum Value: General

Maximum Value: General

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 5

Digits After DP: 0

Leading Zeros: Yes

Group Digits: No

Prefix: None

Suffix: None

As can be seen, the properties shown are indeed identical to those shown on the Format tab of an integer tag. As mentioned above, the properties for the other types of object are similarly identical to those of the corresponding tag. You are thus referred to the earlier section of the manual regarding Data Tags for more information on each property.

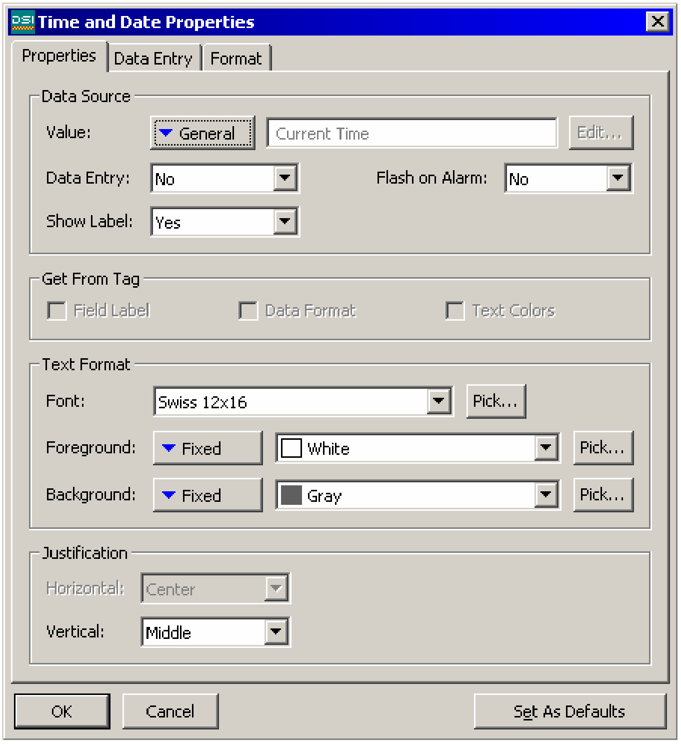
Editing The Underlying Tag

If you want to edit a tag text object's properties, either double-click on the object, or right-click and select the Properties command from the resulting menu. If, however, you want to edit the properties of the tag that is being used to control the object, right-click and select the Tag Details command instead. The resulting dialog box will show the Data and Format tab from the Data Tags window, and allow you to change the various properties. Note that a change made via this mechanism will change all the objects controlled by that tag if those objects have the Get Label or Get Format properties set to From Tag Properties.

The Time And Date Object

The *Time and Date* object is used to display the current time and date, or to display the contents of a time and date expression. It can also be used to edit such an expression, or to set the operator panel's real time clock.

The properties of a time and date object are displayed using two tabbed pages. The first page is shown below:



Time and Date Properties

Data Source

Value: General | Current Time | Edit...

Data Entry: No

Flash on Alarm: No

Show Label: Yes

Get From Tag

□ Field Label □ Data Format □ Text Colors

Text Format

Font: Swiss 12x16 | Pick...

Foreground: Fixed | □ White | Pick...

Background: Fixed | ■ Gray | Pick...

Justification

Horizontal: Center

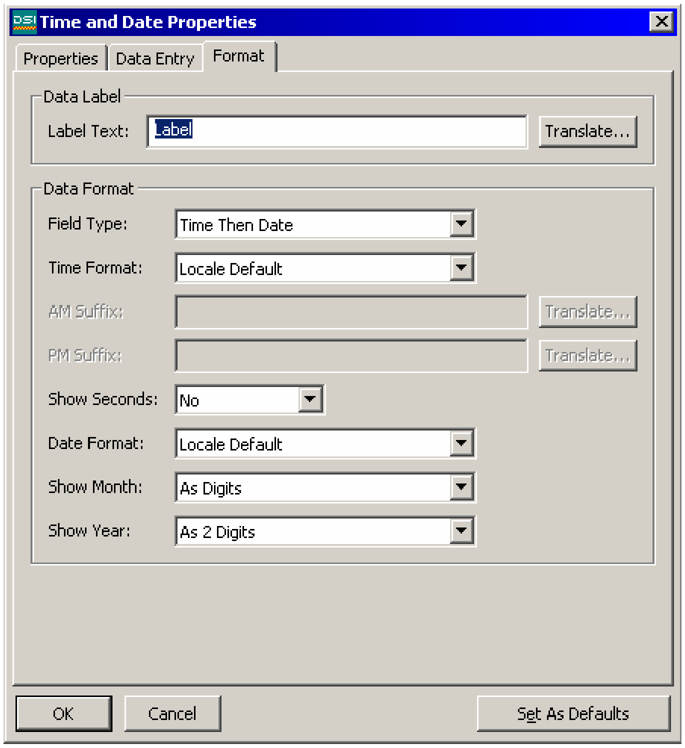
Vertical: Middle

- The *Value* property is used to indicate the time and date value to be displayed. If no value is entered, the current time and date is shown. If an expression is entered, it is taken to represent the number of seconds that have elapsed since

January 1, 1997. Such values are typically obtained using the various time and date functions described in the Function Reference.

- The *Data Entry* property is used to indicate whether or not you want the user of the operator interface panel to be able to change the underlying value via this object. If no value property has been defined, this amounts to changing the current time or date. If a value property has been entered, the expression entered must be capable of being changed. For example, if a formula is entered, data entry will not be permitted.
- The balance of the properties are as described for tag text objects. While it may look odd to have Get Label and Flash On Alarm properties, remember that the value property may be a tag, and so DSI does have access to the tag label and to the tag's alarm state, should you decide to use them.

The second tab is shown below:



Label Text: Label

Field Type: Time Then Date

Time Format: Locale Default

AM Suffix:

PM Suffix:

Show Seconds: No

Date Format: Locale Default

Show Month: As Digits

Show Year: As 2 Digits

- The *Label Text* property is used to define an optional label for the object.
- The *Field Type* property is used to indicate whether the field should display the time, the date or both. In the last case, this property also indicates in which order the two elements should be shown.
- The *Time Format* property is used to indicate whether 12-hour (civil) or 24-hour (military) time format should be used. As with other properties, leaving this set to Locale Default will allow DSI to pick a suitable format according to the language selected within the operator panel.

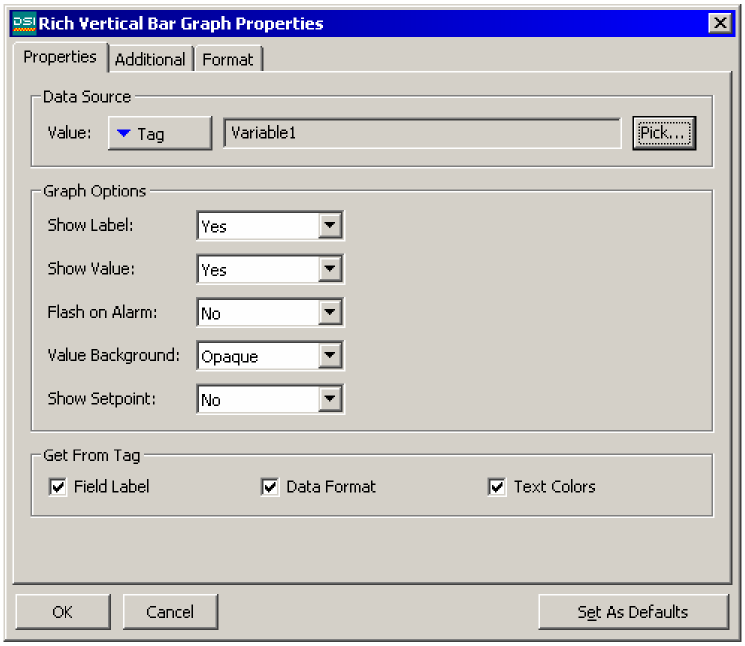
- The *AM Suffix* and *PM Suffix* properties are used with 12-hour mode to indicate the text to be appended to the time field in the morning and afternoon as appropriate. If you leave the property undefined, DSI will use a default.
- The *Show Seconds* property is used to indicate whether the time field should include the seconds, or whether it should just comprise hours and minutes.
- The *Date Format* property is used to indicate the order in which the various date elements (i.e. date, month and year) should be displayed.
- The *Show Month* property is used to indicate whether the month should be displayed as digits (i01 through 12) or as its short name (Jan though Dec).
- The *Show Year* property is used to indicate whether the date field should include the year, and if so, how many digits should be shown for that element.

The Rich Bar Objects

The *Rich Vertical Bar* object allows you to display a more complex bar-graph which includes a label, a numeric version of the data being displayed, and tick markers to indicate any associated setpoint.

The *Rich Horizontal Bar* object allows you to display a more complex bar-graph which includes a label, a numeric version of the data being displayed, and tick markers to indicate any associated setpoint.

The operation of these rich objects is analogous to that of the various tag text objects, in that they are capable of deriving much of the required formatting information from the tag used as their controlling value. Just as with tag text objects, two tabbed pages are used to edit the objects' properties. The first of these pages is shown below:



Rich Vertical Bar Graph Properties

Data Source

Value: Tag Variable1

Graph Options

Show Label: Yes

Show Value: Yes

Flash on Alarm: No

Value Background: Opaque

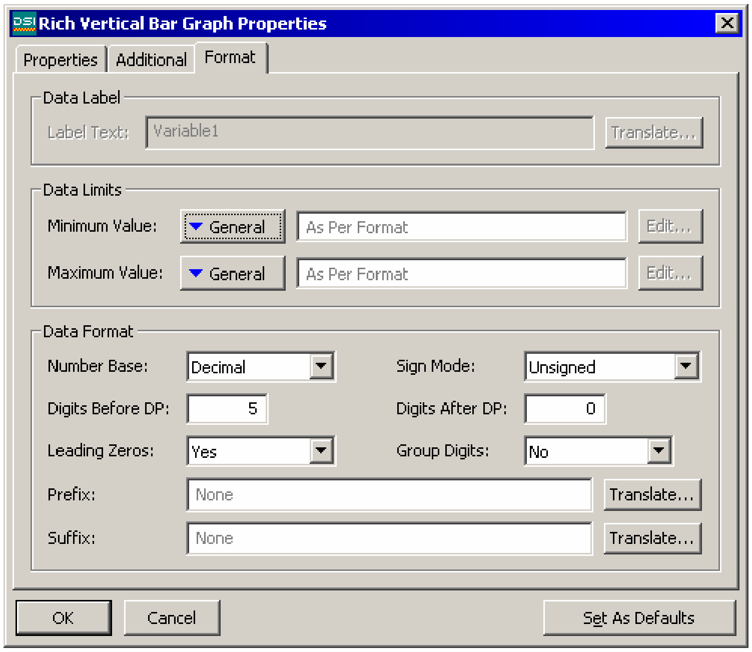
Show Setpoint: No

Get From Tag

☑ Field Label ☑ Data Format ☑ Text Colors

- The *Value* property is used to define the value to be displayed.
- The *Show Label* property is used to indicate whether a label should be included with the bar-graph. For vertical graphs, the label is included at the bottom; for horizontal graphs, it is included at the left-hand side. If a tag is used for the value property, the label may be obtained from that tag. Otherwise, it must be entered on the Format tab of the dialog box.
- The *Show Value* property is used to indicate whether the value of the data should be displayed within the graph itself. If a tag of the appropriate data type is used for the value property, the format may be obtained from the tag. Otherwise, as with the label, it must be entered on the Format tab.
- The *Show Setpoint* property is used to indicate whether tick marks should be added either side of the bar to indicate the setpoint for the controlling value. This option is only available if a tag has been entered for the value field.
- The *Get Label* and *Get Format* properties are as defined for the various tag text objects. The format is not required if the show value property is set to No.
- The *Fill* property is used to indicate the pattern to be used for the active portion of the bar. If you find that your bar-graph does not appear to work, make sure you have not left this property set to None.
- The *Font* property is used to indicate the font to be used to display the value embedded in the graph, if such a value is enabled via the Show Value property.

The second page contains the label and formatting information for the field:



Data Label

Label Text: Variable1

Data Limits

Minimum Value: General As Per Format

Maximum Value: General As Per Format

Data Format

Number Base: Decimal

Sign Mode: Unsigned

Digits Before DP: 5

Digits After DP: 0

Leading Zeros: Yes

Group Digits: No

Prefix: None

Suffix: None

The properties shown are as described for an integer tag, and you are thus referred to the earlier section of the manual that refers to Data Tags for more information. Note that the existence of this object explains why one must enter minimum and maximum values for formulas, when such tags can never be the subject of data entry.

The System Objects

The *Alarm Viewer* object is used to provide the operator with a method to view and accept active alarms. It will always take up the whole of the display width, but can be restricted to less than the full height if required.

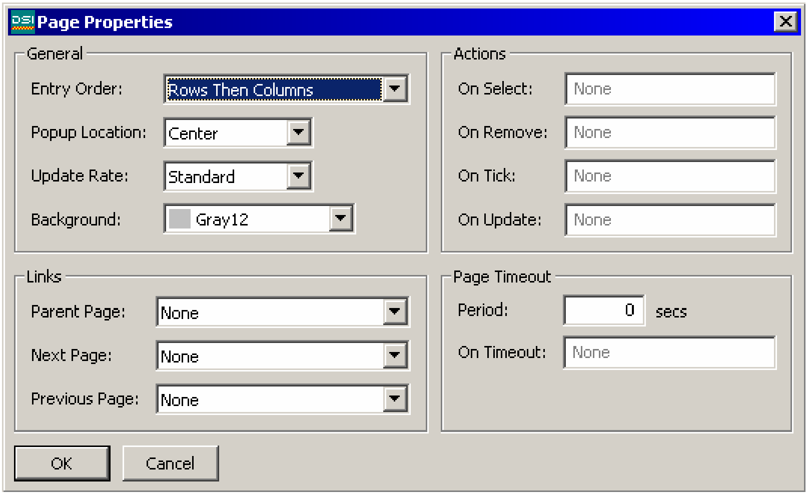
The *Alarm Ticker* object scrolls through the active alarms in the system. It takes up a single line, and the whole of the display width. It does not allow the operator to accept the alarms.

The *Event Viewer* object is used to provide the operator with a method to view the events recorded in the system's event log. It will always take up the whole of the display width, but can be restricted to less than the full height if required.

If you use manual-accept alarms in your system, you should provide a page which contains an alarm viewer to make sure the operator can accept these alarms. You may also wish to add the alarm ticker to other pages to make the operator aware of alarms while they are viewing other pages. Similarly, if you use events, you should provide a page which contains an event viewer to allow the operator to see what events have occurred.

Defining Page Properties

Each page has a number of properties that can be accessed via the Page menu:



Page Properties

General

Entry Order: Rows Then Columns

Popup Location: Center

Update Rate: Standard

Background: Gray12

Actions

On Select: None

On Remove: None

On Tick: None

On Update: None

Links

Parent Page: None

Next Page: None

Previous Page: None

Page Timeout

Period: 0 secs

On Timeout: None

- The *On Select* and *On Remove* properties are used to define actions to be performed when the page is first selected for display, or when the page is removed from the display. Refer to the Writing Actions section and the Function Reference for a list of supported actions. Refer to the Data

Availability section in this chapter for details of a timeout than can occur when using these properties.

- The *On Tick* property is used to define an action that will run every second during the period for which this page is displayed. Refer to the Writing Actions section and the Function Reference for a list of supported actions. If a lack of data availability results in this action being unable to execute, it will be skipped and retried one second later.
- The *Period* is the time in seconds to wait before performing the action specified.
- *On Timeout* is the action to be performed when the period of time has expired.
- The *Parent Page* property is used to indicate the page to be displayed when the panel's **Exit** key is pressed while this page is active. Selection of this page can be overridden using the techniques below.
- The *Next Page* property is used to indicate the page to be displayed when the panel's **Next** key is pressed while this page is active, and when the cursor is on the last data entry field on the page. This selection can also be overridden.
- The *Previous Page* property is used to indicate the page to be displayed when the panel's **Prev** key is pressed while this page is active, and when the cursor is on the first data entry field on the page. This selection can also be overridden.

If you have too many data entry fields to fit on a single page, the Next Page and Previous Page properties can be used to link together a series of pages to allow the operator to edit the fields in sequence. DSI will automatically position the cursor appropriately, such that if the **Prev** key is pressed on the first field of a page, the previous page will be activated with the cursor on the last field of that page.

- The *Entry Order* property is used to define how the cursor on the operator panel will move between data entry fields. The settings determine whether fields organized in a grid will be entered in row or column order.
- The *Update Rate* property is used to define how frequently items on the display are updated. As update rates increase in frequency, overall performance of the operator interface panel may decrease. This selection should be left at the default setting when possible.

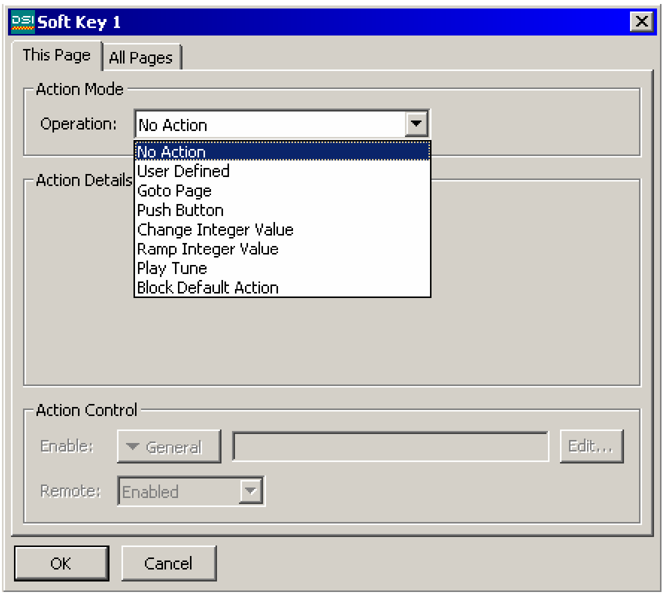
Defining System Actions

In addition to the various actions that can be defined via page properties, DSI gives you the ability to define an action to be run when the system first starts, and an action to be run once a second, no matter which page is displayed. These actions can be accessed by selecting the Pages icon in the left-hand pane of the User Interface window.

Defining Key Behavior

The previous sections have provided a detailed description of how to use the TS8003's display to get information to the operator. All that remains to complete the User Interface configuration is to define how the operator is to use the TS8003's keyboard to interact with the system.

To define the actions to be performed by a key, select a zoom level that allows you to see the key in question. For example, if you want to configure one of the function keys, select zoom level one or two as appropriate. Then, double-click the key to display the following:



Operation: No Action

Enable: General

Remote: Enabled

You will note that this dialog box has two tabbed pages. The first page is used to define what will happen when the key in question is pressed when the current page is selected. The second page is used to define what will happen if the key is pressed when any page is selected. The first type of action is called a *local* action, while the second type is called a *global* action. The color used to display the key will change according to which actions are defined:

If the key is displayed in PURPLE, a local action is defined for this PAGE.

If the key is displayed in GREEN, a GLOBAL action is defined.

F1 If the key is displayed in BLUE, local and global actions are BOTH defined.

Once you have defined an action, you can right-click on the key and use the resulting menu to select either Make Global or Make Local to change the action type. These options will not be available if both types of action have already been defined.

Enabling Actions

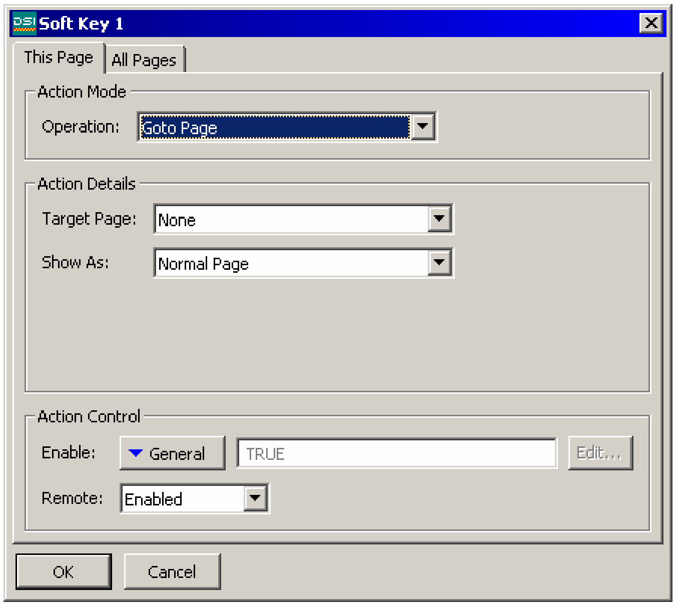
If you want to make a particular action dependent on some condition being true, enter an expression for that condition in the Enable field for the action in question. This expression may reference a flag tag directly, or may use any of the comparison or logical operators defined in the Writing Expressions section. If you need more complex logic such that one of several actions is performed based on more complex decision-making, configure the key in user defined mode and use it to invoke a program that implements the required logic.

Action Descriptions

The sections below describe each available type of action. When each type is selected, the Action Details portion of the action dialog box will change to show the available options.

The Goto Page Action

This action is used to instruct the TS8003 to show a new page. The options are shown below:



Operation: Goto Page

Target Page: None

Show As: Normal Page

Enable: General TRUE

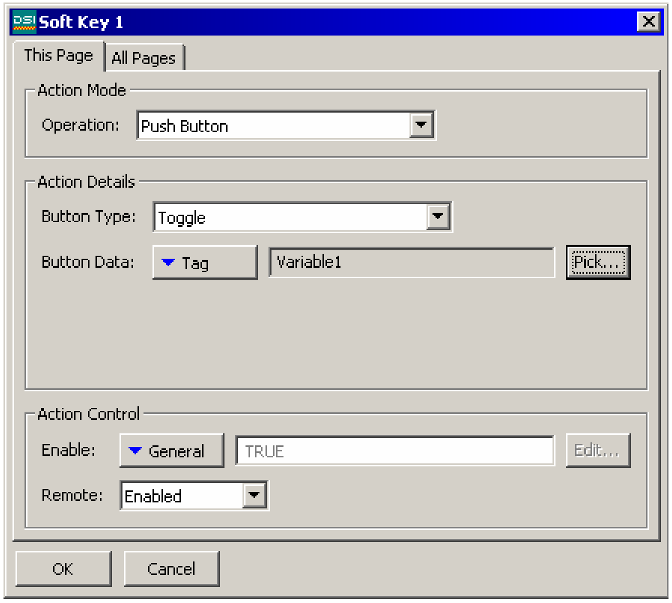
Remote: Enabled

- The *Target Page* property is used to indicate which page should be displayed. You can either choose a specific one to be displayed, or choose Previous Page to return to what was displayed before the current page was called.

- The *Show As Popup* selection causes the target page to be displayed as a popup on top of the current page. While the popup is displayed, the panel keys will assume the definitions established for that page, with the exception of the exit key. The exit key is used to remove the popup from the display.

The Push Button Action

This action is used to emulate a pushbutton. The options are shown below:



Operation: Push Button

Button Type: Toggle

Button Data: Tag Variable1

Enable: General TRUE

Remote: Enabled

- The *Button Type* property is used to define the key's behavior.

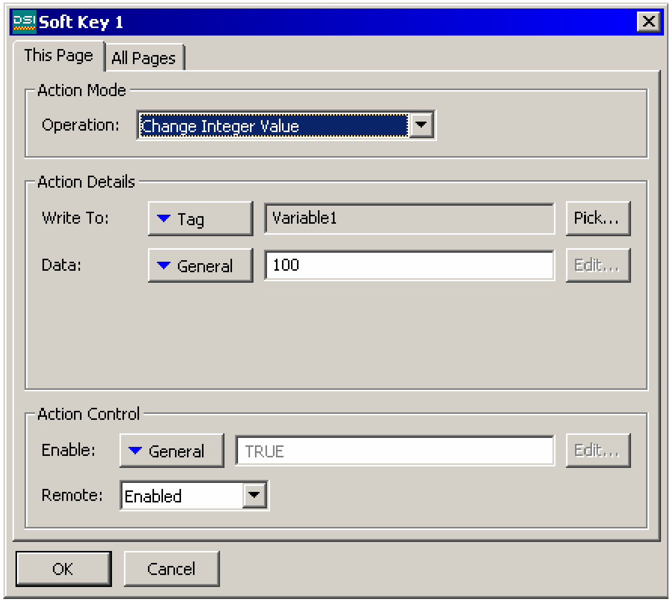
Button Type	The Button Will:
Toggle	Change the data state when the key is pressed.
Momentary	Set the data to 1 when the key is pressed. Set the data to 0 when the key is released.
Turn On	Set the data to 1 when the key is pressed.
Turn Off	Set the data to 0 when the key is pressed.

- The *Button Data* property is used to define the data to be changed by the key.

In the example above, the key will toggle the value of the **Output** tag.

The Change Integer Value Action

This action is used to write an integer value to a data item. The options are shown below:



Operation: Change Integer Value

Write To: Tag Variable1

Data: General 100

Enable: General TRUE

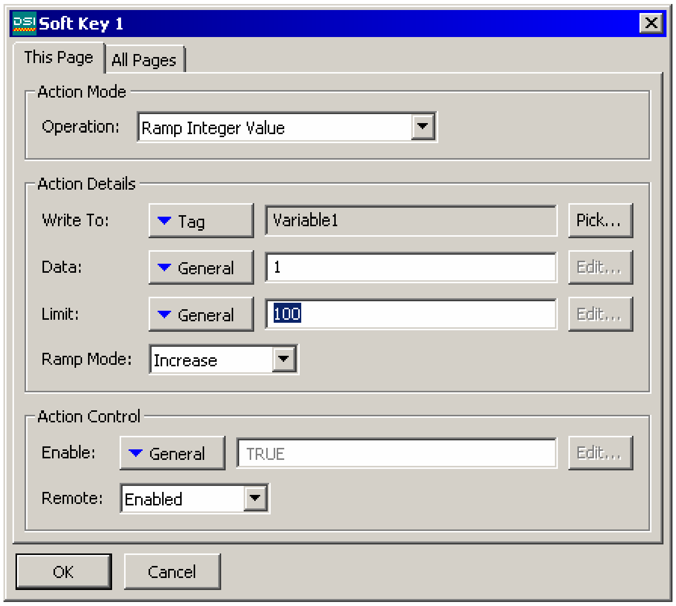
Remote: Enabled

- The *Write To* property is used to define the data item to be changed.
- The *Data* property is used to define the data to be written.

In the example above, the key will set the **Variable1** tag to 100.

The Ramp Integer Value Action

This action is used to increase or decrease a data item. The options are shown below:



Operation: Ramp Integer Value

Write To: Tag Variable1

Data: General 1

Limit: General 100

Ramp Mode: Increase

Enable: General TRUE

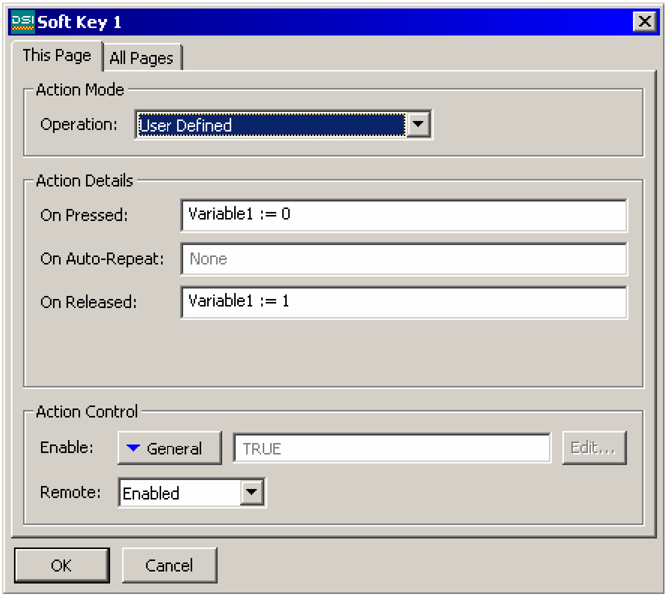
Remote: Enabled

- The *Write To* property is used to define the data item to be changed.
- The *Data* property is used to define the step by which to raise or lower the item.
- The *Limit* property is used to define the minimum or maximum data value.
- The *Ramp Mode* property is used to define whether to raise or lower the item.

In the example above, holding the key will raise **Variable1** by 1 until it reaches 100.

The User Defined Action

This action is used to do anything else you desire. The options are shown below:



Soft Key 1

This Page | All Pages

Action Mode

Operation: User Defined

Action Details

On Pressed: Variable1 := 0

On Auto-Repeat: None

On Released: Variable1 := 1

Action Control

Enable: General TRUE

Remote: Enabled

- The *On Pressed* property is used to define the action to be performed when the key is pressed. This action may invoke any of the functions from the Function Reference or the data modification operators described in the Writing Actions section, or it may run a program.
- The *On Auto-Repeat* property is used to define the action to be performed when the key is pressed and then held down. The action occurs both on the initial depression and on subsequent auto-repeats, so there is no need to define both this property and On Pressed. This action may invoke any of the functions from the Function Reference or the data modification operators described in the Writing Actions section, or it may run a program.
- The *On Released* property is used to define the action to be performed when the key is released. This action may invoke any of the functions from the Function Reference or the data modification operators described in the Writing Actions section, or it may run a program.

In the example above, a user defined action is used to implement a momentary pushbutton.

Block Default Action

This action does not actually do anything, but can be used as a place-holder to prevent further processing. As an example, suppose you have configured F1 to perform a global action, but want to prevent this action from being invoked on a particular page. By configuring F1 on that page as Block Default Action, the global action will not occur.

Changing The Language

To configure a key to change the language displayed by the operator panel, select User Defined mode and enter SetLanguage(n) as the On Pressed property, where n is a number between 1 and 8, according to the language to be displayed. The display page will be redrawn in the selected language, with any text for which translations have been entered, including fixed text, tag labels and tag formatting information, adjusted as appropriate. Pages that are subsequently displayed will also be drawn in the selected language.

Advanced Topics

The following sections deal with more advanced issues relating to keyboard actions.

Action Processing

When a key is pressed or released, DSI goes through a defined sequence when deciding what to do with the event. If any stage results in some action being performed, the sequence is stopped, and the later stages do not get a chance to process the key.

The sequence is as follows:

1. If a display object is selected for user interaction, it is given a chance to process the key. Active data entry fields will consume the Raise, Lower, Exit and Enter keys, plus whatever other keys are appropriate to the operation being performed. For example, integer entry fields will also consume the numeric keys.
2. If a display object is selected for user interaction and the Next or Prev keys are pressed, DSI will attempt to find the next or previous display object which also desires user interaction. If any such field exists, the key will be consumed, and that object will be activated.
3. If a local action is defined, the action is performed and the key consumed.
4. If a global action is defined, the action is performed and the key consumed.
5. If the key remains unconsumed, the default actions are implemented:

Event	Action
Next Key Pressed	Displays the page's Next Page, if one is defined.
Prev Key Pressed	Displays the page's Previous Page, if one is defined.
Exit Key Pressed	Displays the page's Parent Page, if one is defined.
Menu Key Pressed	Displays the fist page in the page list.
Mute Key Pressed	Silences the TS8003's internal alarm speaker.

As mentioned above, configuring a key for any global or local action (even one that does nothing, such as Block Default Action) prevents this sequence from proceeding. It should be obvious, then, why such an action is useful, even though at first sight it serves no purpose.

Data Availability

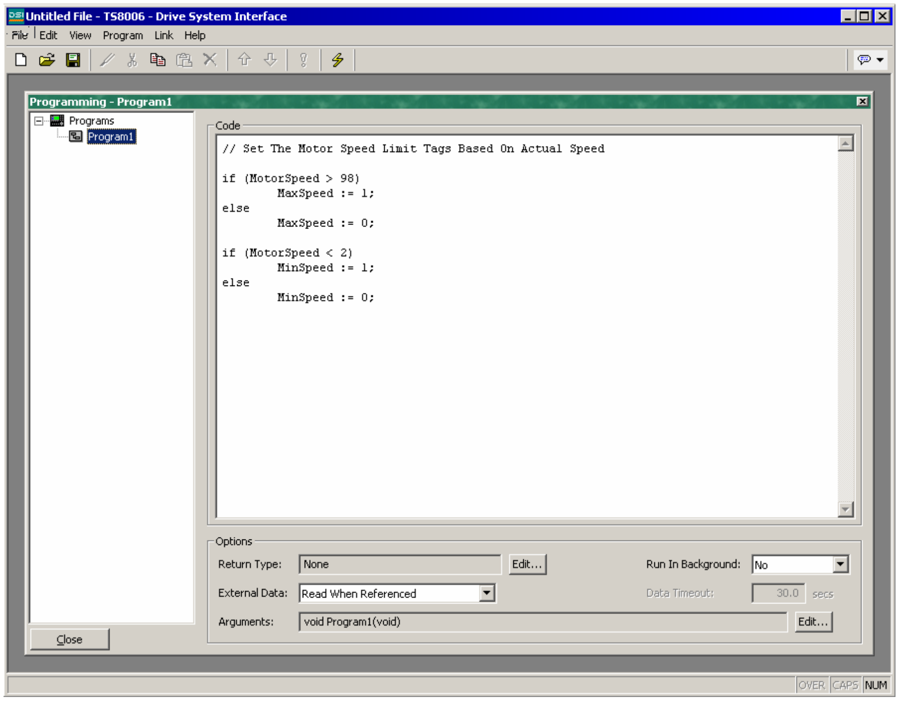
DSI's communications infrastructure reads only those data items which are required for the current page. This means that when a page is first selected, certain data items may not be available. For a display object, this is no problem, as the object simply displays an undefined state (typically a number of dashes) until the data becomes available. For actions, though, things can get more complex.

For example, suppose a local action increases the speed of a motor by 50 rpm. If the motor speed is not referenced on the previously displayed page, then, when the page is first displayed, DSI will not know the current speed, and will thus be unable to write the new value. To handle this, if the operator attempts to perform an action for which the required data is not available, the TS8003 panel will display a "NOT READY" message until the key in question is released. The operator must then wait a short while, and try the operation again. In practice, communications updates normally take place quickly enough that even the most nimble-fingered operator will be hard pressed to get the message to appear, but since it may on occasions be seen, it is worth explaining.

A slightly more complex issue comes about if the action defined by a page's On Select property is unable to proceed because it also finds that required data is not available. Here, DSI will wait up to thirty seconds for the data to arrive. If it does not, the action will not be performed, and a "TIMEOUT" message will be displayed for the operator. This timeout mechanism is required to avoid problems should a communications link become severed.

Configuring Programs

The previous sections of this manual describe how you can use actions to perform all manner of operations in response to key presses or changes in data tags. If you need to perform an action that is too complex to fit on a single line, or that demands more complex decision-making logic, you can use the Programming icon from the main screen to create and manipulate programs. You should note that many applications will not need programs. You may thus choose to skip this chapter if desired.



```
// Set The Motor Speed Limit Tags Based On Actual Speed

if (MotorSpeed > 98)
        MaxSpeed := 1;
else
        MaxSpeed := 0;

if (MotorSpeed < 2)
        MinSpeed := 1;
else
        MinSpeed := 0;
```

Return Type: None

External Data: Read When Referenced

Arguments: void Program1(void)

Run In Background: No

Data Timeout: 30.0 secs

Using The Program List

To create, rename or delete programs, click on the left-hand pane of the User Interface window. The various commands on the Program menu can then be used to make the desired changes. Alternatively, right-click on the required program, and select from the menu.

To select a program, either click on the name in the list, or use the up and down arrows in the toolbar. Alternatively, you can use the Alt+Left and Alt+Right key combinations to move up and down the list as required. These keys will work no matter which pane is selected.

Editing Programs

To edit a program, simply edit the program text using the large area in the right-hand pane of the Programming window. When you have finished, press the Ctrl+T key combination or select the Translate command from the Program menu. This will read the program and check it for errors. If an error is found, a dialog box will be displayed, and the cursor will be moved to the approximate position of the error. If no errors exist, a dialog box will be displayed to confirm this fact, and the program will be translated into DSI's internal format for subsequent execution by the operator panel.

Program Properties

The various fields at the bottom of the right-hand pane are used to edit program properties:

- The *Return Type* property is used to indicate whether this program should simply perform a series of actions, or whether it will perform a calculation and return the value of that calculation to the user. Programs that return values are described in more detail below.

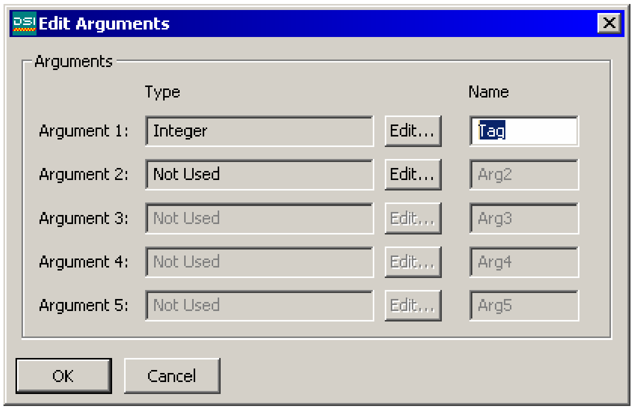
- The *Run In Background* property is used to indicate whether DSI should wait for the program to complete execution before continuing with processing whatever task invoked the program. For example, if this property is set to No, running a program in response to a key being pressed will result in a pause in display updates until the program completes (since most programs take very little time to execute, this may not even be noticeable). If this property is set to Yes, display updates will continue immediately, and the program will execute at a lower priority in the background. Only one background program will run at once, so subsequent requests are queued for later execution. Note also that programs which return values cannot be run in the background, as their return value would then not be available for the caller to use.

- The *External Data* and *Timeout* properties are used to control how the program interacts with DSI's communication infrastructure with respect to external data items to which the program makes reference. You will recall that DSI only reads data items when they are used. This property is used to control the exact interpretation of this rule with respect to programs:

Mode	Behavior
Read When Referenced	External data used by the program will be added to the comms scan whenever the program is referenced. If the program is referenced by a display page, the data will be read when that page is displayed; if the program is referenced by a global action or a trigger, the data will be read at all times. This is the default mode, and is acceptable for all programs, except those that use very large amounts of external data.
Read Always	External data used by the program will be read at all times, whether or not the program is referenced. This means that the program will always be ready to run, and that the operator will not see the "NOT READY" message that might otherwise occur when the program is first referenced. The downside of this mode is that comms performance may be reduced if large amounts of data are referenced by the program.

Mode	Behavior
Read When Executed	External data used within the program will be read only when the program is invoked. The program will wait for the period defined in the timeout property for such data to be available. If the data cannot be read (perhaps because a device is offline) the program will not execute. This mode is typically used with globally-referenced programs that consume large amounts of data that would otherwise slow down the communications scan.
Read But Run Anyway	External data will be treated as described for Read Always mode, but the program will execute whether or not the data has been read successfully. The operator will therefore never see the "NOT READY" message, but if a device is offline, there is no guarantee that the program's data items contain valid data.

- The *Arguments* property is used to specify up to five arguments that can be passed into the program. Each argument has a name and a data type, as specified by the dialog box that is displayed upon pressing the Edit button.



	Type	Name
Argument 1:	Integer	Tag
Argument 2:	Not Used	Arg2
Argument 3:	Not Used	Arg3
Argument 4:	Not Used	Arg4
Argument 5:	Not Used	Arg5

Passing arguments to programs is described below in more detail.

Adding Comments

You can add comments to your programs in two ways. Firstly, you can use the // sequence to introduce a comment which will continue for the rest of the current line. Secondly, you can use the /* sequence to introduce a single or multi-line comment. This comment will continue until the */ sequence appears. The sample below shows both commenting styles:

```
// This is a single-line comment

/* This is line 1 of the comment
   This is line 2 of the comment
   This is line 3 of the comment */
```

A single-line comment may also be placed at the end of a line which contains code.

Returning Values

As mentioned above, programs can return values. Such programs can be invoked by other programs or by expressions anywhere in the database. For example, if you want to perform a particularly complex decode on a number of conditions relating to a motor and return a value to indicate the current state, you could create a program that returns an integer like this:

```
If ( MotorRunning )
        return 1;
else {
        if ( MotorTooHot )
                return 2;
        if ( MotorTooCold )
                return 3;
        return 0;
        }
```

You could then configure a multi-state formula to invoke this program, and use that tag's format tab to define the names of the various states. The invocation would be performed by setting the tag's Value property to Name(), where Name is the name of the program in question. The parentheses are used to indicate a function call, and cannot be omitted.

A Word Of Caution

Note that you have to exercise a degree of caution when using programs to return values. In particular, you should avoid looping for long periods of time, or performing actions that make no sense in the context in which the function will be invoked. For example, if the code fragment above called the GotoPage function to change the page, the display would change every time the program was invoked. Therefore, keep programs that return values simple, and always consider the context in which they will be run. If in doubt, avoid doing anything other than simple math and if statements.

Passing Arguments

As also mentioned above, program can accept arguments. As an example, suppose you want to write a program called **FindMean** to take the average of two values. The program could be configured to accept two integer arguments, **a** and **b**, as shown in the example given when defining the purpose of the *Arguments* property. The program would also be configured so as to return a integer value. The code within the program would then be defined as:

```
return ( a + b ) / 2;
```

Once this program has been created and translated, you will be able to enter an expression such as **FindMean (Tag1, Tag2)** to invoke it with the appropriate arguments. In this case, the expression will be equal to the average of **Tag1** and **Tag2**.

Programming Tips

The sections below provide an overview of the programming constructs supported by DSI. The basic syntax used is that of the C programming language. Note that the aim is not to try and teach you to become a programmer, or to master the subtleties of the C language. Such topics are beyond the scope of this document. Rather, the aim is to provide a quick overview of the facilities available, so that the interested user might experiment further.

Multiple Actions

The simplest type of program comprises a list of actions, with each action taking up a single line, and being followed by a semicolon. All of the various actions defined in the Writing Actions section are available for use. Simple programs like this are typically used where combining the actions in a single action definition would otherwise prove unreadable.

The sample shown below sets several variables, and then changes the display page:

```
Motor1 := 0;
Motor2 := 1;
Motor3 := 0;

GotoPage(Page1);
```

The actions will be executed in order, and the program will then return to the caller.

If Statements

This type of statement is used within a program to make a decision. The construct consists of an **if** statement with a condition in parentheses, followed by an action (or actions) to be executed if the condition is true. If more than one action is specified, each should be placed on a separate line, and curly-brackets should be used to group the statements together. An optional **else** clause can be used to provide for code to be run if the condition is false.

The example below shows an if statement with a single action:

```
if ( TankFull )
          StartPump := 1;
```

The example below shows an if statement with two actions:

```
if ( TankEmpty )

          {
          StartPump := 0;
          OpenValue := 1;
          }
```

The example below shows an if statement with an Else clause:

```
if ( MotorHot )
          StartFan := 1;
else
          StartFan := 0;
```

Note that it is very important to remember to place the curly-brackets around groups of actions to be executed in the if or Else portion of the statement. If you omit the brackets, DSI8000 will most likely misunderstand exactly which actions you want to be dependent upon the if condition. Although line breaks are recommend between actions, they are not used to figure out what is and is not included within the conditional statement.

Switch Statements

A switch statement is used to compare an integer value against a number of possible constants, and to perform an action based upon which value is matched. The exact syntax supports a number of options beyond those shown in the example below, but for the vast majority of applications, this simple form will be acceptable.

This example below will start a motor selected by the value in the **MotorIndex** tag:

```
switch ( MotorIndex )
        {
        case 1:
                MotorA := 1;
                break;
        case 2:
        case 3:
                MotorB := 1;
                break;
        case 4:
                MotorC := 1;
                break;
        default:
                MotorD := 1;
                break;
        }
```

A value of 1 will start motor A, a value of 2 or 3 will start motor B, and a value of 4 will start motor C. Any value which is not explicitly listed will start motor D. Things to note about the syntax are the use of curly-brackets around the **case** statements, the use of **break** to end each conditional block, the use of two sequential **case** statements to match more than one value, and the use of the optional **default** statement to indicate an action to perform if none of the specified values is matched by the value in the controlling expression. If this syntax looks too intimidating, a series of **if** statements can be used instead to produce the same results, but with marginally lower performance, and somewhat less readability.

Local Variables

Some programs use variables to store intermediate results, or to control one of the various loop constructs described below. Rather than defining a tag to hold these values, you can declare what are known as local variables using the syntax shown below:

```
int     a ;         // Declare local integer 'a'
float   b ;         // Declare local real    'b'
cstring c ;         // Declare local string  'c'
```

Local variables may optionally be initialized when they are declared by following the variable name with **:=** and the value to be assigned. Variables which are not initialized in this manner are set to zero, or an empty string, as appropriate.

Note that local variables are truly local in both scope and lifetime. This means that they cannot be referenced outside the program, and they do not retain their values between function invocations. If a function is called recursively, each invocation has its own variables.

Loop Constructs

The three different loop constructs can be used to perform a given section of code while a certain condition is true. The while loop tests its condition before the code is executed, while the do loop tests the condition afterwards. The for loop is a quicker way of defining a while loop, allowing you to combine three common elements into one statement.

You should note that some care is required when using loops within your programs, as you may make a programming error which results in a loop that never terminates. Depending on the situation in which the program is invoked, this may seriously disrupt the terminal's user interface activity, or its communications. Loops which iterate too many times may also cause performance issues for the subsystem that invokes them.

The While Loop

This type of loop repeats the action that follows it while the condition in the while statement remains true. If the condition is never true, the action will never be executed, and the loop will perform no operation beyond evaluating the controlling condition. If you want more than one action to be included in the loop, be sure to surround the multiple statements in curly-brackets, as with the if statement. The example below initializes a pair of local variables, and then uses the first to loop through the contents of an array, totaling the first ten elements, and returning the total value to the caller:

```
int i := 0, t := 0;

while ( i < 10 )
        {
        t := t + Data[i];
        i := i + 1;
        }

return t;
```

The example below shows the same program, but rewritten in a compressed form. Since the loop statement now controls only a single action, the curly-brackets have been omitted:

```
int i := 0, t := 0;

while ( i < 10 )
        t += Data[i++];

return t;
```

The For Loop

You will notice that the while loop shown above has four elements:

1. The initialization of the loop control variable.

2. The evaluation of a test to see if the loop should continue.

3. The execution of the action to be performed by the loop.
4. The making of a change to the control variable.

The for loop allows elements 1, 2 and 4 to be combined within a single statement, such that the action following the statement need only implement element 3. This syntax results in something similar to the FOR-NEXT loop found in BASIC and other such languages. Using this statement, the example given above can be rewritten as:

```
int i, t;

for ( i := t := 0; i < 0; i++ )
          t += Data[i];

return t;
```

You will notice that the for statement contains three distinct elements, each separated by semicolons. The first element is the initialization step, which is performed once when the loop first begins; the next is the condition, which is tested at the start of each loop iteration to see if the loop should continue; the final element is the induction step, which is used to make a change to the control variable to move the loop on to its next iteration. Again, remember that if you want more than one action to be included in the loop, include them in curly-brackets.

The Do Loop

This type of loop is similar to the while loop, except that the condition is tested at the end of the loop. This means that the loop will always execute at least once. The example below shows the example from above, rewritten to use a do loop:

```
int i := 0, t := 0;

do {
          t += Data[i];
          } while ( ++i < 10 );

return t;
```

Loop Control

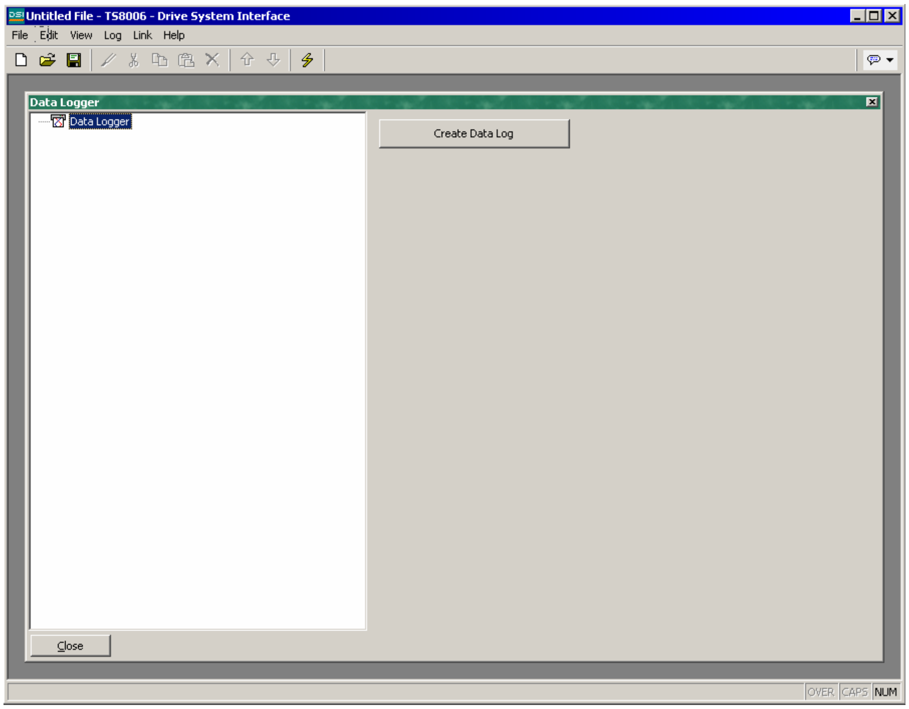
Two additional statements can be used within loops. The **break** statement can be used to terminate the loop early, while the **continue** statement can be used to skip the balance of the loop body and begin another iteration without executing any further code. To make any sense, these statements must be used with **if** statements to make their execution conditional. The example below shows a loop which terminates early if another program returns true:

```
for ( i := 0; i < 10; i++ )

        {
        if ( LoopAbort() )
                break;
        LoopBody();
        }
```

Configuring Data Logging

Now that you have configured the core of your application, you may decide to make use of DSI's data logger to record certain tag values to CompactFlash. Data recorded in this way is stored in industry-standard comma-separated variable (CSV) files, and can easily be imported into applications such as Excel using a variety of methods. To configure data logging, select the Data Logger icon from the main screen:


Untitled File - TS8006 - Drive System Interface
File Edit View Log Link Help
Data Logger
Data Logger
Create Data Log
Close
OVER CAPS NUM

Creating Data Logs

You may use the Create Data Log button to create as many data logs as you need. Since each log can record an unlimited number of data tags, most applications will only use a single log. However, since each log has a fixed set of properties in terms of its sample rate, you may decide to use multiple logs if you wish to sample different data at different rates.

Using The Log List

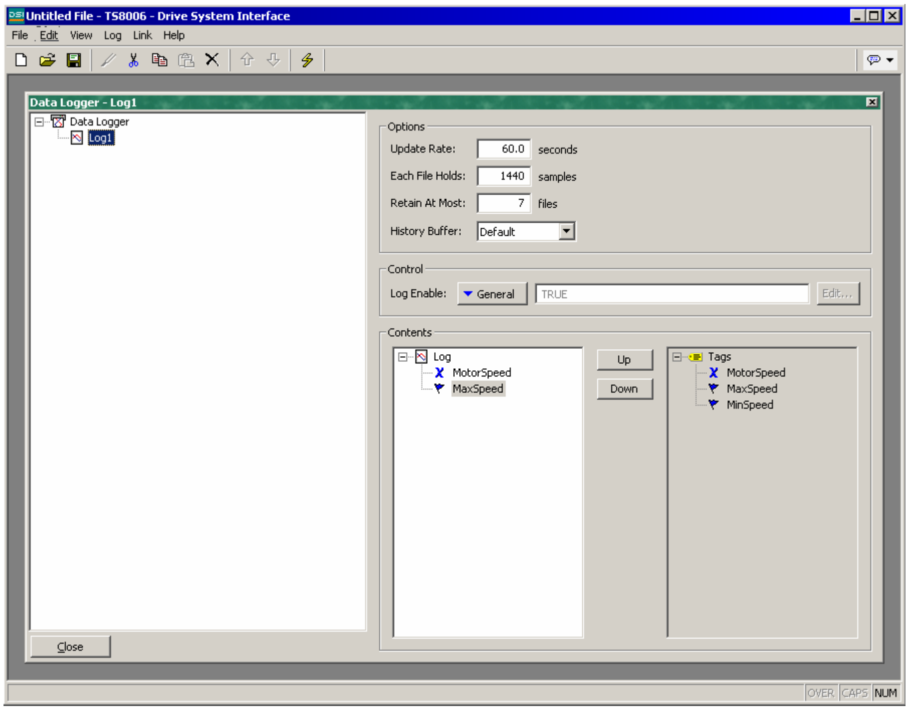
To rename or delete data logs, click on the left-hand pane of the Data Logger window. The commands on the Log menu can then be used to make the desired changes. Alternatively, you may right-click on the required data log, and select from the menu.

Note that the name of a data log must be eight characters or less in length. This is because the name will be used to define the directory under which the log files are stored, and the TS8000 panel is not able to handle names that do not conform to FAT-style 8.3 naming.

To select a data log, either click on the name in the list, or use the up and down arrows in the toolbar. Alternatively, you can use the Alt+Left and Alt+Right key combinations to move up and down the list as required. These keys will work no matter which pane is selected.

Data Log Properties

Each data log has the following properties:



Update Rate: 60.0 seconds

Each File Holds: 1440 samples

Retain At Most: 7 files

History Buffer: Default

Log Enable: General TRUE

- The *Update Rate* property is used to indicate how often DSI will take a sample of the data items to be logged. The fastest sample rate is one second, but note that using such a high rate will produce very large amounts of data. All of the tags in the log will be sampled at the same rate.

- The *Each File Holds* property is used to indicate how many samples will be included in each log file. When this many samples have been recorded, a new log file will be created using a different name. Typically, this value is set such that each log file contains a sensible amount of data. For example, the log shown above is configured to use a new log file each day.

- The *Retain At Most* property is used to indicate how many log files will be kept on CompactFlash before the oldest file is deleted. This property should be set so as to allow whatever is consuming the logged information to extract the data from the TS8000 panel before the information is deleted. The log shown above is configured to retain a week's worth of data.

- The *Log Enable* property is used to allow or inhibit logging. If the entered expression is true, logging will be enabled. If the expression is false, logging will be disabled. If no expression is entered, logging is enabled by default.

- The *History Buffer* property is used to indicate how much RAM should be allocated for the history buffer for this data logger. The history buffer is used to support the historical trending user interface object, and allows the user to scroll backwards to view older data than would otherwise be available. No more than a total of 256K should be allocated to all data logs.

- The *Contents* property is used to indicate which tag should be logged. The first list shows the selected tags, while the second shows those that are available within the database. Tags can be added to the log by double-clicking them in the right-hand list; they can be removed by double-clicking them in the left-hand list, or by pressing the Del key while the tag is selected. The Up and Down buttons can be used to move tags within the list.

Log File Storage

As mentioned above, a data log stores its data in a series of files on the operator panel's CompactFlash card. These files are placed in a subdirectory named after the data log, with this directory being stored under a root directory entry called LOGS. The files are named after the time and date at which the log is scheduled to begin. If each file contains an hour or more of information, the files will be named YYMMDDhh.CSV, where YY represents the year of the file, MM represents the month, DD represents the date, and hh represents the hour. If each file contains less than one hour of information, the files will instead be named MMDDhhmm.CSV, with the initial six characters as described above, and the trailing mm representing the minute at which the log began. These rules ensure that each log file has a unique name.

The Logging Process

DSI's data logger operates using two separate processes. The first samples each data point at the rate specified in its properties, and places the logged data into a buffer within the RAM of the TS8000 panel. The second process executes every two minutes, and writes the data from RAM to the CompactFlash card. This structure has several advantages:

- Writes to the CompactFlash card are guaranteed to begin only on a two-minute boundary - that is, at exactly 2, 4 or 6 minutes past the hour, and so on. This means that if your TS8000 panel supports hot-swapping of CF cards, you can wait for the next burst of writes to start, and, when the CompactFlash activity LED on the front of the panel ceases to flicker, you are guaranteed to have until the start of the next two-minute interval before further writes will be attempted. This means that you can remove the card without fear of data corruptions. As long as you insert a new card before four minutes have elapsed, no data will be lost.

- Writes to the CompactFlash achieve a much higher level of performance, by avoiding the need to continually update the card's file system data structures for every single sample. For logs configured to sample at very high data rates, the bandwidth of a typical CompactFlash card would not allow data to be written reliably in the absence of such a buffering process.

Note that because data is not committed to CompactFlash for up to two minutes, up to this amount of log data may be lost when the terminal is powered-down. Further, if the terminal is powered-down while a write is in progress, the CompactFlash card may be corrupted. To ensure that such corruption is not permanent, the TS8000 panel uses a journaling system that caches writes to additional non-volatile memory within the terminal. If the panel detects that a write was interrupted during power-down, the write will be repeated when power is reapplied, thereby reversing any corruption, and repairing the CompactFlash card.

This means that if you want to remove a CompactFlash card from a panel performing data logging, you must observe the procedure described above with respect to the activity LED, and only remove power when the activity has ceased. If you are not sure if the terminal was powered-down correctly, reapply power, allow a CompactFlash write sequence to complete, and power down according to the correct procedure. The card can then be removed safely.

Since the gyrations required to remove a CompactFlash card are somewhat complex, DSI provides two other mechanisms for accessing log files, thereby eliminating the need for such removals. These methods are described below.

Accessing Log Files

There are two additional methods of accessing log files:

- The less preferable method is to mount the card as a drive on a PC via the process described at the start of this manual, so that the logs can be copied using Windows Explorer. Note that Windows 2000 or above is recommended when using this method, as earlier versions of Windows may otherwise lock the CompactFlash card and disrupt data logging.
- The preferred method is to use the web server as described in the next chapter. With the web server enabled, log files can be accessed over the panel's Ethernet port, using either a web browser, such as Microsoft Internet Explorer, or by using the automated process implemented by the WebSync utility that is provided with the DSI8000 configuration software.

Using WebSync

The WebSync utility, which will be stored in the directory specified when the software was installed, can be executed to synchronize a directory on a PC with the contents of an operator panel's data logs. You may decide to configure an application, such as the Windows Scheduler (or perhaps a cron daemon), to run this utility on a regular basis, or you may use a command line switch to instruct WebSync to perform the polling automatically. You may also decide to host WebSync on a central server so that the log files can be made available to selected users on your corporate network.

WebSync Syntax

WebSync is invoked from the command line using the following syntax:

```
websync {switches} <hostname>
```

where <hostname> is replaced with the IP address of the panel to be polled.

Optional Switches

The switches field may contain one or more of the following options:

- –terse can be used to suppress progress information.
- –poll <n> can be used to poll the terminal every n minutes.
- –path <dir> can be used to specify dir as the directory to hold the log files.

Example Usage

As an example, the following command line:

```
websync –poll 10 –path C:\Logs 192.9.200.52
```

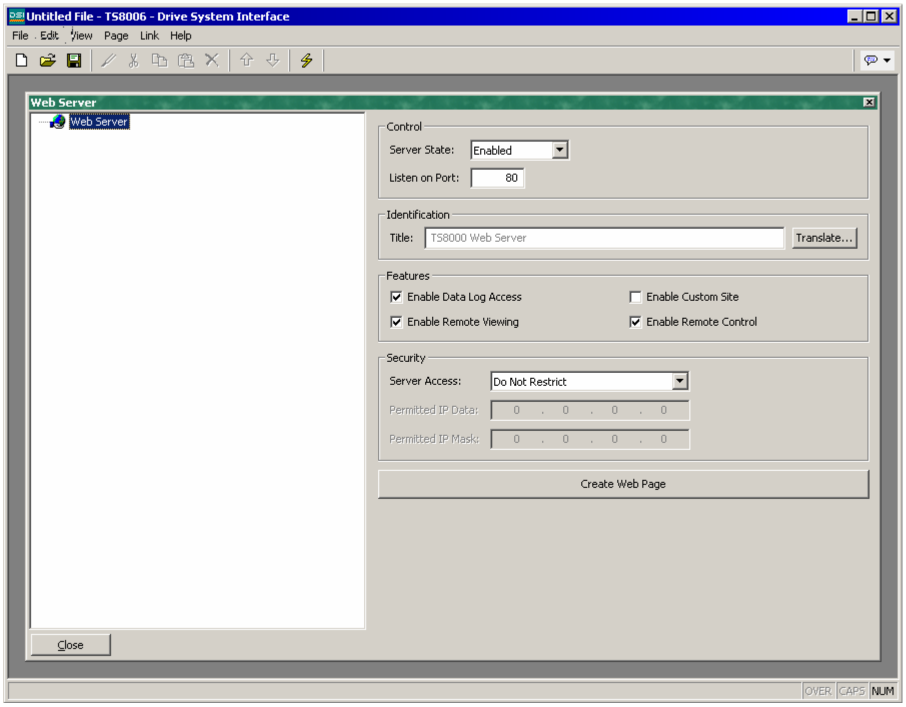
will read the log files for all data logs on the terminal with the IP address of 192.9.200.52, and will store these logs under subdirectories of the C:\Logs directory. WebSync will continue to execute, and will repeat the polling process every ten minutes. The polling interval must obviously be set such that it is much less than the sampling rate times the number of samples in a file times the number of log files to be retained. If this constraint is met, the directory on the PC will accumulate copies of all the log files from the terminal.

Configuring The Web Server

DSI's web server can be used to expose various data via the TS8000 panel's Ethernet port, allowing remote access to diagnostic information, or to the values recorded by the Data Logger. The web server is configured by selecting the Web Server icon from the main screen.

Web Server Properties

The web server has the following properties:



Server State: Enabled

Listen on Port: 80

Title: TS8000 Web Server

☑ Enable Data Log Access

□ Enable Custom Site

☑ Enable Remote Viewing

☑ Enable Remote Control

Server Access: Do Not Restrict

Permitted IP Data: 0 . 0 . 0 . 0

Permitted IP Mask: 0 . 0 . 0 . 0

- The *Enable Server* property is used to enable or disable the web server. If the server is enabled, the panel will monitor port 80 for incoming requests, and will fulfill the requests as required. If the server is disabled, connections to this port will be refused. Remember that in order for the server to operate, the panel's Ethernet port must have been enabled via the Communications window.
- The *Title* property is used to provide the title to be shown on the web server menu. This title can be used to differentiate between several terminals on a network, thereby ensuring that the correct terminal is being accessed.
- The *Enable Data Log Access* property is used to enable or disable web access to the files created by the Data Logger. Obviously, this facility must be enabled if the WebSync utility is to be used to copy the log files to a PC.

- The *Enable Remote Viewing* property is used to enable or disable a facility by which a web browser can be used to view the current contents of a TS8000's display. This facility is very useful when remotely diagnosing problems that an operator may be having with the operator panel or the machine it controls.

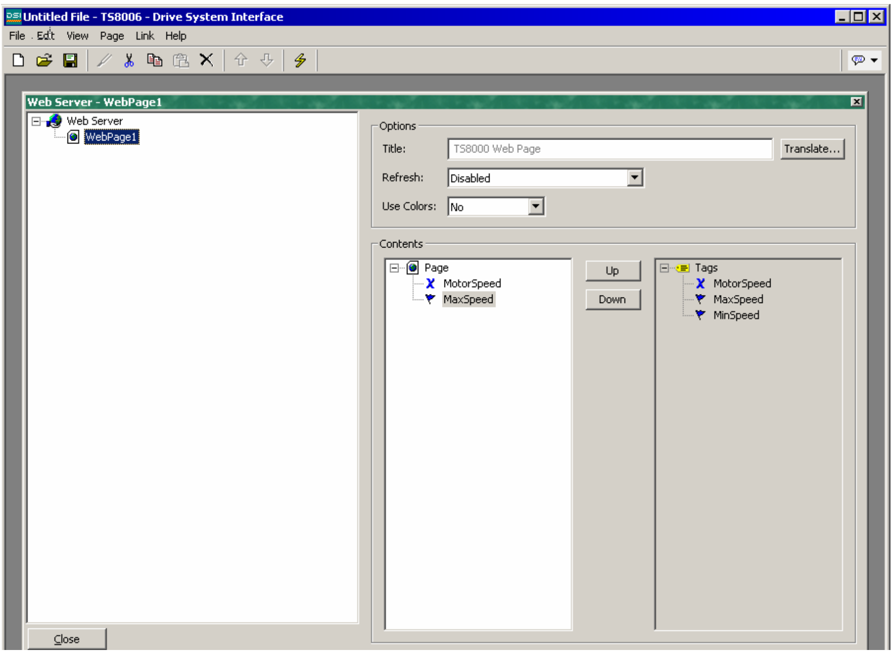
- The *Enable Remote Control* property is used to enable or disable an option by which the remote viewing facility is extended to allow a web browser to be used to simulate the pressing of keys on the operator panel, thereby allowing remote control of the panel or the machine it controls. While this feature is extremely useful, care must be taken to use the various security parameters to avoid unauthorized tampering with a machine. The use of an external firewall is also strongly recommended if the panel is reachable from the Internet.

- The *Enable Custom Site* property is used to enable or disable a facility by which files stored in the WEB directory of the CompactFlash card are exposed via the web server. This facility is described in more detail below.

- The *Security* properties are used to restrict web server access to hosts whose IP address matches the mask and data indicated. All access may be restricted, or the filter may be used to restrict only attempts to use the remote control facility. It is your responsibility to use an external firewall to prevent unauthorized access if the remote control facility is enabled, as the IP filter may be defeated by certain advanced hacking techniques, and is not warranted by SSD Drives.

Adding Web Pages

In addition to the facilities described above, the web server supports the display of generic web pages, each of which contains a predefined list of tag values. These pages are created by pressing the Create Web Page button below the web server properties, and are stored in a list similar to that used for display pages, data logs and so on.



Untitled File - TS8006 - Drive System Interface

Web Server - WebPage1

Web Server
- WebPage1

Options

Title: TS8000 Web Page

Refresh: Disabled

Use Colors: No

Contents

Page
- MotorSpeed
- MaxSpeed

Tags
- MotorSpeed
- MaxSpeed
- MinSpeed

Each web page has the following properties:

- The *Title* property is used to identify the web page in the menu presented to the user via their web browser. Although the title is translatable, current versions of DSI use only the US version of the text.
- The *Refresh* property is used to indicate whether or not the web browser should be instructed to refresh the page contents automatically. Update rates between 1 and 8 seconds are supported. Note that the amount of flicker exhibited by the web browser will vary according to the exact package used and the performance of the machine being employed. The update is not intended to be flicker-free.
- The *Contents* property is used to indicate which tags should be included on the page. The first list shows the selected tags, while the second shows those that are available within the database. Tags can be added to the page by double-clicking them in the right-hand list; they can be removed by double-clicking them in the left-hand list, or by pressing the Del key while the tag is selected. The Up and Down buttons can be used to move tags within the list.

Using A Custom Web Site

While the standard web pages provide quick-and-easy access to the data within the terminal, you may find that your inability to edit their precise formatting leaves your artistic capabilities somewhat frustrated. You may thus use the terminal's custom site facility to create a completely custom web site using your favorite third-party HTML editor, and (by inserting certain special sequences and storing the resulting files on the panel's CompactFlash card) expose this site using the panel's web server.

Creating The Site

The web site may use any HTML facilities supported by your browser, but must not use ASP, CGI or other server-side tricks. The filenames used for the HTML files and associated graphics must also comply with the old-style 8.3 naming convention. This means that file extensions will be, for example, HTM instead of HTML, and JPG instead of JPEG. This also means that the body of the filename must be eight characters or less, and that you must not rely on the difference between upper- and lower-case to differentiate between pages. You may use any directory structure, as long as you once again ensure that your directories observe the 8.3 naming convention and do not rely on case differences.

Embedding Data

To embed tag data within a web page, insert the sequence [[N]], replacing N with the index number of the tag in question. This index number is displayed on the status bar when a tag is selected within the Data Tag window, and more-or-less corresponds to the order in which the tags were created. When the web page containing this sequence is served, the sequence will be replaced by the current value of the tag, formatted according to the tag's properties.

Deploying The Site

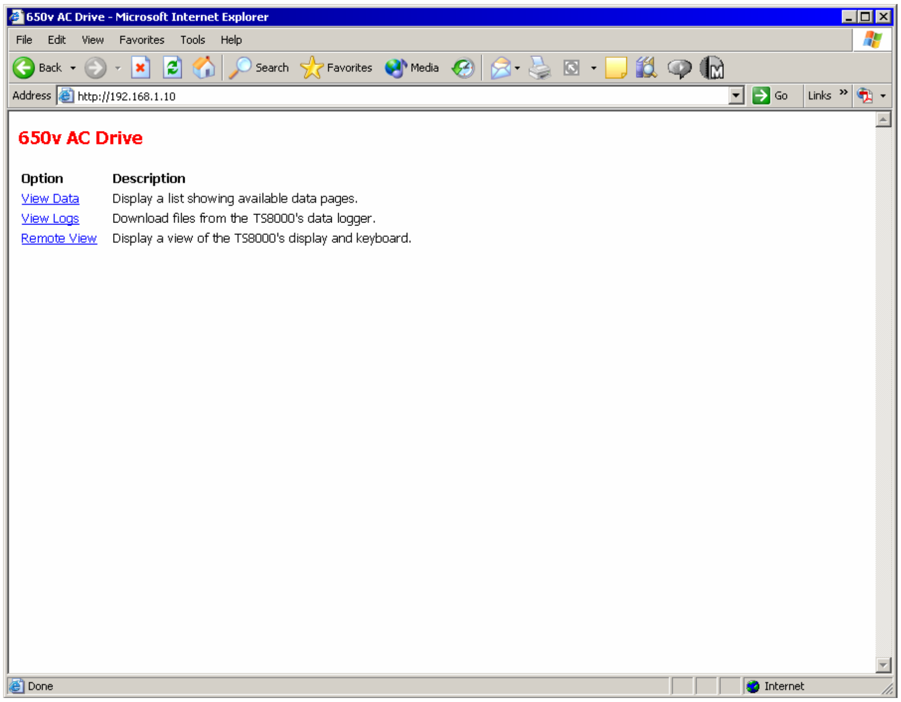
To deploy your custom web site, copy it into the \WEB directory on the CompactFlash card to be installed in the terminal. To copy the files, either mount the card as a drive on your PC as described at the start of this manual, or use a suitable card writer connected to your PC. Make sure that the *Enable Custom Site* property is set, and the custom site will appear on the web server menu. When the site is selected, a file called DEFAULT.HTM within the \WEB directory will be displayed. Beyond that point, navigation is according to the links within the site.

CompactFlash Access

Note that in order to serve custom web pages, or to provide access to the panel's data logger, the web server needs to be able to access the unit's CompactFlash card. If you have mounted the card as a drive on your PC and performed write operations, you may have to wait a minute or so for the PC to unlock the card and allow the terminal to get access. If you are using an operating system earlier than Windows 2000 to perform such an operation, you may find that your PC locks the card when the drive is first mounted, whether or not a write is performed. Again, this lock will be released within a minute or so.

Web Server Samples

The picture below shows the main menu displayed by the web server:

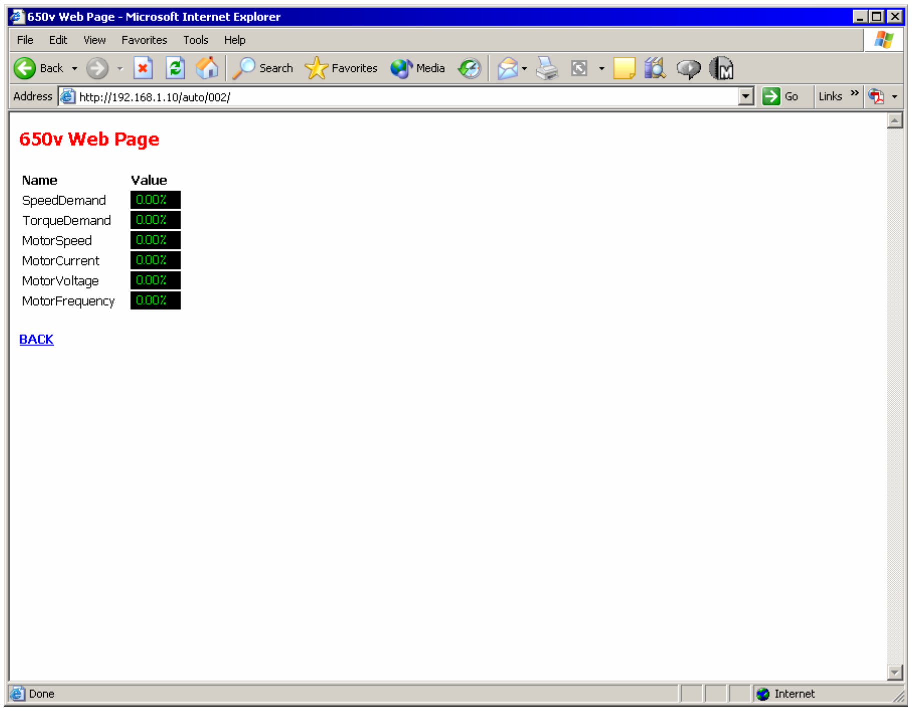


Address: http://192.168.1.10

650v AC Drive

Option	Description
View Data	Display a list showing available data pages.
View Logs	Download files from the TS8000's data logger.
Remote View	Display a view of the TS8000's display and keyboard.

The picture below shows a standard web page containing tags:


650v Web Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Done
Internet

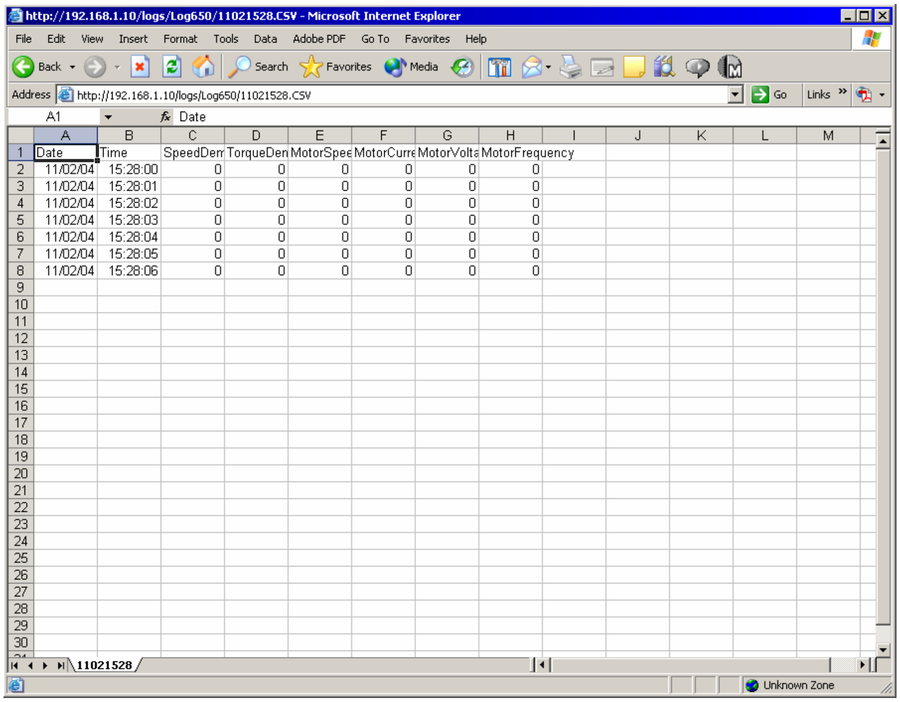
http://192.168.1.10/auto/002/

650v Web Page

Name	Value
SpeedDemand	0.00%
TorqueDemand	0.00%
MotorSpeed	0.00%
MotorCurrent	0.00%
MotorVoltage	0.00%
MotorFrequency	0.00%

BACK

The picture below shows the contents of a given log file:

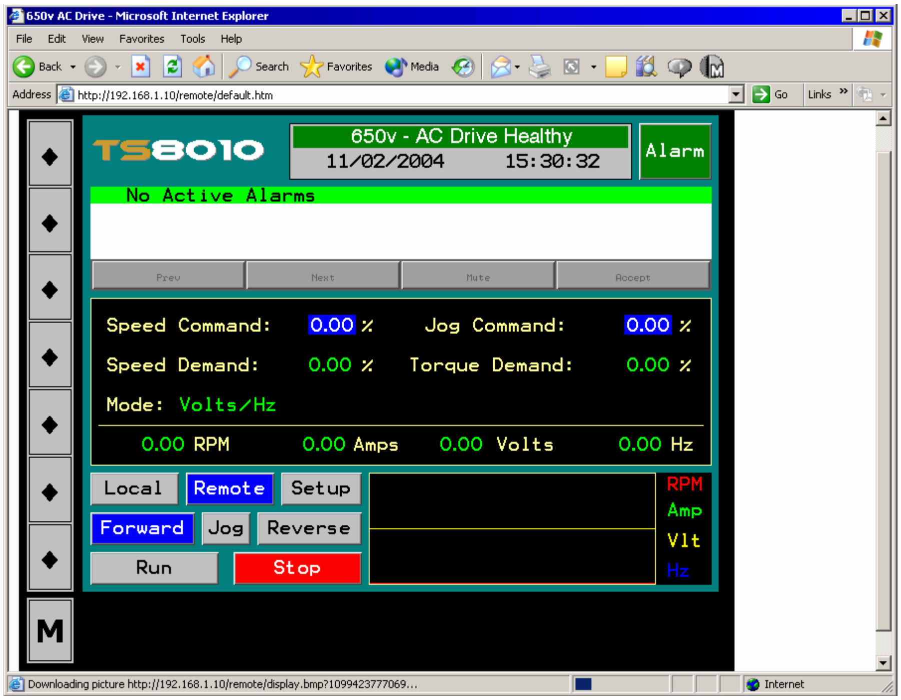

http://192.168.1.10/logs/Log650/11021528.CSV - Microsoft Internet Explorer
File Edit View Insert Format Tools Data Adobe PDF Go To Favorites Help
Back Search Favorites Media
Address
Go Links
A1
11021528
Unknown Zone

http://192.168.1.10/logs/Log650/11021528.CSV

fx Date

	A	B	C	D	E	F	G	H
1	Date	Time	SpeedDem	TorqueDen	MotorSpee	MotorCurre	MotorVolta	MotorFrequency
2	11/02/04	15:28:00	0	0	0	0	0	0
3	11/02/04	15:28:01	0	0	0	0	0	0
4	11/02/04	15:28:02	0	0	0	0	0	0
5	11/02/04	15:28:03	0	0	0	0	0	0
6	11/02/04	15:28:04	0	0	0	0	0	0
7	11/02/04	15:28:05	0	0	0	0	0	0
8	11/02/04	15:28:06	0	0	0	0	0	0

The picture below shows the remote viewing and/or control display:



Address: http://192.168.1.10/remote/default.htm

650v - AC Drive Healthy
11/02/2004 15:30:32

No Active Alarms

Speed Command: 0.00 % Jog Command: 0.00 %
Speed Demand: 0.00 % Torque Demand: 0.00 %
Mode: Volts/Hz

0.00 RPM 0.00 Amps 0.00 Volts 0.00 Hz

Downloading picture http://192.168.1.10/remote/display.bmp?1099423777069...

Using The Security Manager

DSI8000 contains powerful features to allow you to define which operators have access to which display pages, and limit those operators who are able to make changes to sensitive data. The software also contains a security logging facility that can be used to record changes to data values indicating when the change occurred, and by whom it was performed.

Security Basics

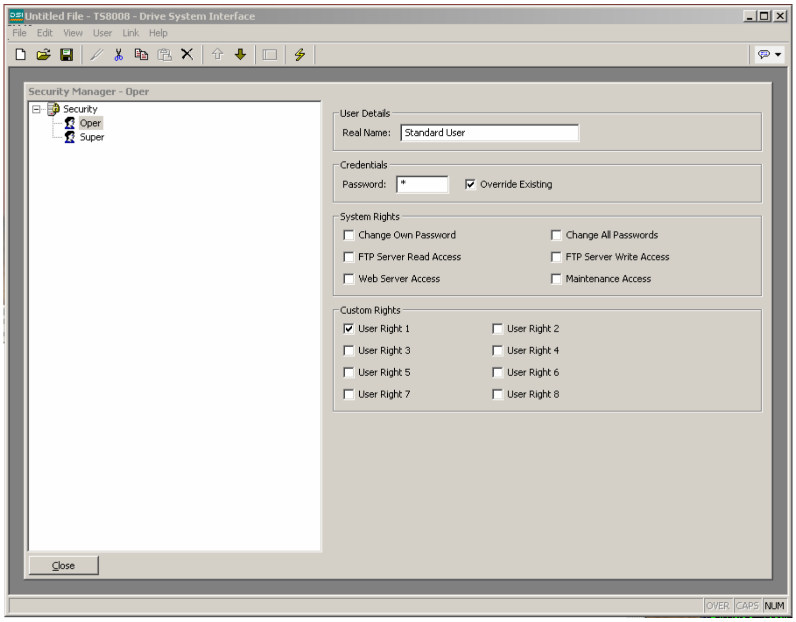
The follow sections details some of the basic concepts used by the security system.

Object Based Security

DSI's security system is object-based. This means that security characteristics are applied to a display page or to a tag, and not to the user interface element that accesses the page or makes a change to the tag. The alternative subject-based approach typically means that you have to be careful to apply security settings to every single user interface element that might change restricted data. DSI's approach avoids this duplication and ensures that once you have decided to protect a tag, it will remain protected throughout your database.

Named Users

DSI8000 supports the ability to create any number of users, each of whom will have a username, a real name and a password. The username is a case-insensitive string with no embedded spaces that is used to identify the user when logging on, while the real name is typically a longer string that is used within logon files to record the human-readable identity of the user making a change. Note that you are free to use these fields in other ways if it suits your application: You may, for example, create users that represent groups of individuals or perhaps roles, such as Operators, Supervisors and Managers. You may also decide to use the real name to hold an item such as a clock number to tie user identities into your MRP system.



User Details

Real Name: Standard User

Credentials

Password: * ☑ Override Existing

System Rights

☐ Change Own Password ☐ Change All Passwords
☐ FTP Server Read Access ☐ FTP Server Write Access
☐ Web Server Access ☐ Maintenance Access

Custom Rights

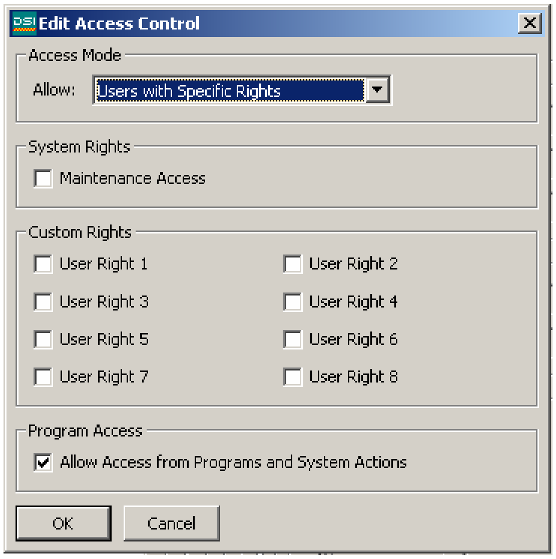
☑ User Right 1 ☐ User Right 2
☐ User Right 3 ☐ User Right 4
☐ User Right 5 ☐ User Right 6
☐ User Right 7 ☐ User Right 8

User Rights

Each user is granted zero or more access rights. A user with no rights can access those objects that merely require the identity of the user to be recorded, whereas users with more rights can access those objects that demand those rights to be present. Rights are divided into System Rights and User Rights, with the former controlling access to facilities within the DSI8000 software, and the latter being available for general use. For example, User Right 1 might be used within your database to control access to production targets; only users whom you want to be able to vary such things would then be assigned this right.

Access Control

Objects that are subject to security have an associated access control setting.



Edit Access Control

Access Mode

Allow: Users with Specific Rights

System Rights

□ Maintenance Access

Custom Rights

□ User Right 1 □ User Right 2
□ User Right 3 □ User Right 4
□ User Right 5 □ User Right 6
□ User Right 7 □ User Right 8

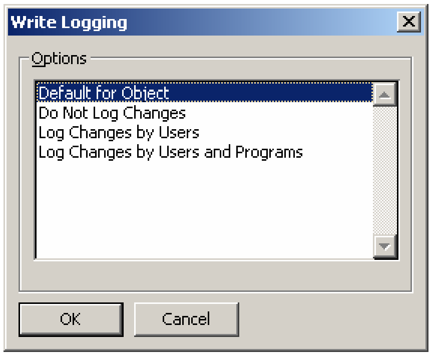
Program Access

☑ Allow Access from Programs and System Actions

This setting allows you to specify whether the item can be accessed by anyone, by any operator whose identity is known, or by users with specific user rights. The access control setting also allows you to specify whether a tag can be changed by a program running as a result of something other than user action. This facility allows you to guarantee that no background changes occur to sensitive data, even if a programming error attempts to make such a change.

Write Logging

Tags also have a write logging property.



Write Logging

Options

Default for Object
Do Not Log Changes
Log Changes by Users
Log Changes by Users and Programs

This indicates whether changes made to a tag by users or by programs should be logged. This facility allows you to create an audit trail of changes to your system, thereby simplifying fault finding and providing quality-control information as to process configuration. Note that care should be taken when logging changes made by programs, as certain database may log unmanageable amounts of data in such circumstances.

Default Access

To speed the configuration process, DSI8000 also provides the ability to specify default access and write logging parameters for mapped tags, internal tags and display pages. The differentiation between mapped and unmapped tags is important in systems where all changes to external data must be recorded, but where data internal to DSI can be manipulated without the need for such an audit trail.

On-Demand Logon

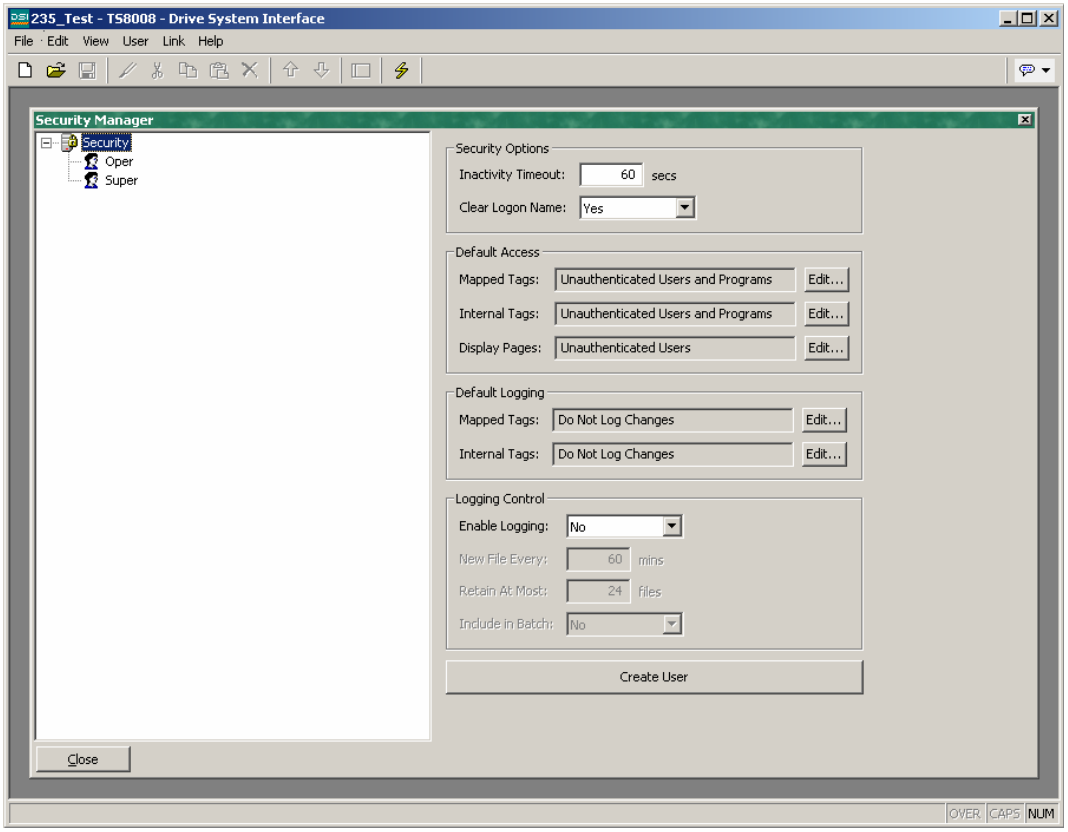
DSI's security system supports both conventional and on-demand logon. A conventional logon can occur when a user interface element such as a pushbutton is used to activate the Log On User action or to call the UserLogOn() function. On-demand logon occurs if the operator attempts an action without sufficient access rights, and if a failed logon attempt has not occurred within the same action. For example, a user may press a button that runs a program to reset a number of values. As soon as the program attempts to change a value that requires security access, the system will prompt for logon credentials. This method reduces operator interaction, and produces a more responsive system.

Maintenance Access

The system also provides a facility called Maintenance Mode to allow the user inactivity timeout to be overridden during system commissioning. This mode is activated if a display page is marked as being accessible with the Maintenance Access right, and if the current user has gained access to the page as a result of that right. Use of this mode avoids the need to logon repeatedly when testing the system.

Security Settings

The security system settings are accessed via the Security Manager icon.



Security Options

Inactivity Timeout: 60 secs

Clear Logon Name: Yes

Default Access

Mapped Tags: Unauthenticated Users and Programs

Internal Tags: Unauthenticated Users and Programs

Display Pages: Unauthenticated Users

Default Logging

Mapped Tags: Do Not Log Changes

Internal Tags: Do Not Log Changes

Logging Control

Enable Logging: No

New File Every: 60 mins

Retain At Most: 24 files

Include in Batch: No

The available properties are as follows:

- The *Inactivity Timeout* property is used to indicate how much time must pass without user input before the current user is automatically logged off. Too high a value for this setting will produce an insecure system, while too low a value will produce a system that is awkward for operators.

- The *Clear Logon Name* property is used to indicate whether or not the username should be cleared before asking the operator to logon. If this setting is disabled, the previous username will be displayed, and only the password will need to be re entered. Enabling this feature produces higher security, and may be required to comply with security standards in certain industries.

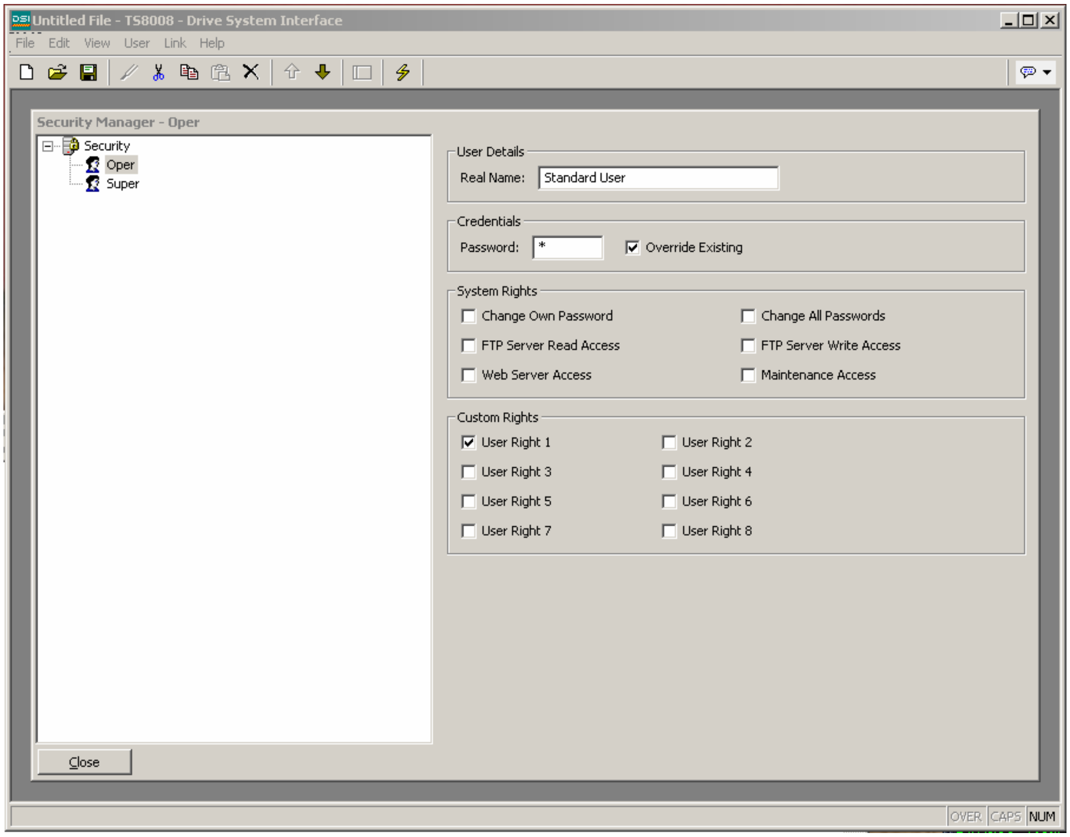
- The *Default Access* properties are used to indicate the access to be provided to various objects should no specific access be defined for that item. The settings are as described in the Access Control section above.

- The *Default Logging* properties are used to indicate whether changes to mapped and unmapped tags should be logged should no specific logging criteria be defined for a tag. It is not possible to log programmatic access by default, as such logging should be carefully considered to avoid excessive log activity.

- The *Logging Control* properties are used to define whether and how the security logs should be created. Refer to the Configuring Data Logging chapter for information on how the data is written and how files are named.

Creating Users

You may use the Create User button to create as many users as you need. The users may be renamed or deleted using the left-hand pane. To select a user, either click on the name in the list, or use the up and down arrows in the toolbar. Alternatively, you can use the Alt+Left and Alt+Right key combinations to move up and down the list as required. These keys will work no matter which pane is selected.


Untitled File - TS8008 - Drive System Interface
File Edit View User Link Help
Security Manager - Oper
Security
Oper
Super
Close
OVER CAPS NUM

User Details

Real Name: Standard User

Credentials

Password: * ☑ Override Existing

System Rights

☐ Change Own Password ☐ Change All Passwords
☐ FTP Server Read Access ☐ FTP Server Write Access
☐ Web Server Access ☐ Maintenance Access

Custom Rights

☑ User Right 1 ☐ User Right 2
☐ User Right 3 ☐ User Right 4
☐ User Right 5 ☐ User Right 6
☐ User Right 7 ☐ User Right 8

The available properties are as follows:

- The *Real Name* property is used to record the user's identity in security logs, and in the Security Manager object that is used to change passwords from the operator terminal. If maximum security is required, the user name should not be easily derived from the real name.
- The *Password* property is used to specify an initial password for this user. The password is case-sensitive and comprises alphanumeric characters. Note that if the *Override Existing* box is checked, any changes made to this password

from the operator panel itself will be overridden when this database is downloaded to the panel.

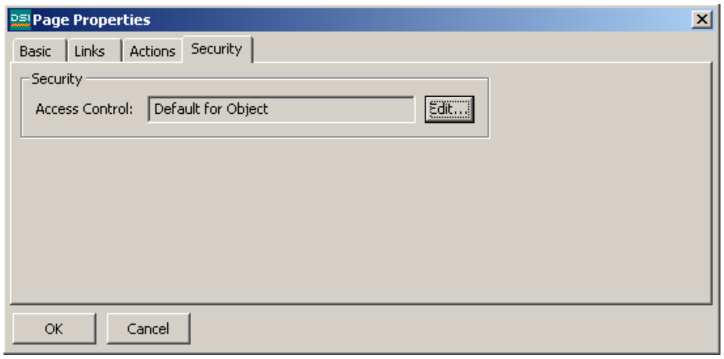
- The *System Rights* properties are used to grant a user the ability to perform certain system actions. The properties relating to password changes are self-explanatory, while the user of Maintenance Mode is described above.
- The *Custom Rights* properties are used to grant a user certain rights which may then be used within the database to allow access to groups of tags or display pages. The exact usage of these rights is up to the system designer.

Specifying Tag Security

Each writable tag has a tab called Security which is used to define the access control and write logging settings for that tag. If you do not define specific settings, the system will use the appropriate default settings, depending on whether it is mapped to external data.

Specifying Page Security

The access control settings for a display page are defined via the Properties dialog.



Access Control: Default for Object

Once again, if no setting is defined, default settings will be used.

The Security Manager Object

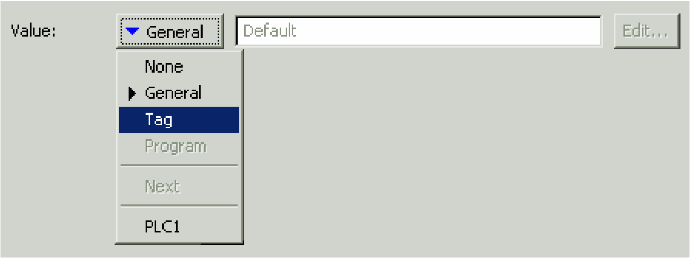
The *Security Manager* object is used to display the names of users present on the system. It can be used to change a user's password, depending on the rights allocated to the active user.

The only editable properties of this object define the fonts to be used, and whether or not the object should be displayed. Refer to other objects for descriptions of these settings.

Security Related Functions

Please refer to the Function Reference section of this manual for details on the UserLogOn(), UserLogOff() and TestAccess() functions. This third function is useful when changing many values from within a program, as it allows you to force an access check early in the code to avoid making changes only to have later operations fail due to insufficient user rights.

Writing Expressions



You will recall from the earlier sections of this manual that many fields within DSI are configured as what are called expression properties. You will further recall that these fields are configured by means of a user interface element similar to that shown here.

In many situations, you will be configuring these properties to be equal to the value of a tag, or to the contents of a register in a remote communications device, in which case your selection will be made simply by clicking the appropriate option on the drop-down menu, and then selecting the required item from the resulting dialog box.

There will be situations, though, when you want to make a property dependent on a more complex combination of data items, perhaps using some math to combine or compare their values. Such eventualities are handled via what are known as expressions, which can be entered in the property's edit box whenever General mode is selected via the drop-down.

Data Values

All expressions contain at least one data value. The simplest expressions are thus references to single constants, single tags or single PLC registers. If you enter either of the last two options, DSI will simplify the editing process by automatically changing the property mode as appropriate. For example, if you enter a tag name in General mode, DSI will switch to Tag mode, and show the tag name in the selection field.

Constants

Constants represent—not surprisingly—constant numbers or strings.

Integer Constants

Integer constants represent a single 32-bit signed number. They may be entered in decimal, binary, octal or hexadecimal as required. The examples below show the same number entered in the four different number bases:

Base	Example
Decimal	123
Binary	0b1111011
Octal	0173
Hexadecimal	0x7B

Character Constants

Character constants represent a single ASCII character, encoded in the lower 8 bits of a 32-bit signed number. A character constant comprises a single character enclosed in single quotation marks, such that 'A' can be used to represent a value of 65. Certain otherwise unprintable or unrepresentable characters can be encoded using what are called escape sequences, each of which is introduced with a single backslash:

Sequence	Value	ASCII
\a	Hex 0x07, Decimal 7	BEL
\t	Hex 0x09, Decimal 9	TAB
\n	Hex 0x0A, Decimal 10	LF
\f	Hex 0x0C, Decimal 12	FF
\r	Hex 0x0D, Decimal 13	CR
\e	Hex 0x1B, Decimal 27	ESC
\x*nnn*	The hex value represented by *nnn*.	-
\nnn	The octal value represented by *nnn*.	-
\\\\	A single backslash character.	-
\\'	A single quotation mark character.	-
\\"	A double quotation mark character.	-

Logical Constants

Logical constants represent a 1 or 0 value that is used to indicate the truth or otherwise of a yes-or-no expression. An example of something that can be assigned to be equal to a logical constant is a tag that represents a digital output in a PLC. Logical constants can either be entered simply as 1 or 0, or by use of the keywords true or false.

Floating-Point Constants

Floating-point constants represent a 32-bit single-precision floating-point value. They are represented as you might expect—by the integer portion, followed by a single decimal point, followed by the fractional portion. Exponential notation is not supported.

String Constants

String constants represent sequences of characters. They comprise the characters to be represented, enclosed in double quotation marks. For example, the string "ABCD" represents a four-character string, comprising the values 65, 66, 67 and 68. Actually, five bytes are used to store the string, with a null value being appended to indicate the end of the string. The various escape sequences discussed above may also be used within strings.

Tag Values

The value of a tag is represented in an expression by the tag name. Upper-case and lower-case characters are considered equivalent when finding the required tag. Also, once an expression has been entered, any changes to the name of the tag will modify all of the expressions that make reference to it, so there is no need to re-edit the expressions to “fix” the name.

Communications References

References to registers in master communications devices can be entered into an expression by means of a syntax comprising an opening square bracket, the register name, and a closing square bracket. An optional device name may be prefixed to the register name and separated by a period. The device name need not be specified for registers in the first (or only) device within the database. Examples of this syntax are shown below:

Example	Meaning
[D100]	Register D100 in first device.
[AB.N7:0]	Register N7:0 in device AB.
[FX.D100]	Register D100 in device FX.

Simple Math

As mentioned above, expressions often contain more than one data value, with their values being combined mathematically. The simplest of these expressions may add a pair of values, while a more complex expression might obtain the average of three values. These operations are performed using the familiar syntax you will have seen in applications such as Excel. The examples below show the basic operations that can be performed:

Operator	Priority	Example
Addition	Group 4	Tag1 + Tag2
Subtraction	Group 4	Tag1 - Tag2
Multiplication	Group 3	Tag1 * Tag2
Division	Group 3	Tag1 / Tag2
Remainder	Group 3	Tag1 % Tag2

Although the examples show spaces surrounding the operators, these are not required.

Operator Priority

You will have noticed the Priority column in the above table. As you no doubt recall from your algebra classes, when several operators are used together, they are evaluated in a defined order. For example, multiplication is always evaluated before addition. DSI implements this ordering by means of what are known as operator priorities, with each operator being put in a group, and with operators being applied in order from the lowest numbered group to the highest. Except where noted otherwise in the text, operators within a group are evaluated left-to-right. The default order of evaluation can be overridden by using parentheses.

Type Conversion

Normally, DSI will automatically decide when to switch from evaluating an expression in integer math to evaluating it using floating-point. For example, if you divide an integer value by a floating-point value, the integer will be converted to floating-point before the division is carried out. However, there will be some situations where you want to force a conversion to take place.

For example, suppose you are adding together three integers which represent the levels in three tanks, and then dividing the total by the tank count to obtain the average level. If you use an expression such as (Tank1+Tank2+Tank3)/3 then your result may not be as accurate as you demand, as the division will take place using integer math, and the average will not contain any decimal places. To force DSI to evaluate the result using floating-point math, the simplest technique is to change the 3 to 3.0, thereby forcing DSI to convert the sum to floating-point before the division is performed. A slightly more complex technique is to use syntax such as float(Tank1+Tank2+Tank3)/3. This invokes what is known as a “type cast” on the term in parentheses, manually converting it to floating-point.

Type casts may also be used to convert a floating-point value to an integer value, perhaps deliberately giving-up some precision from an intermediate value before storing it in a PLC register. For example, the expression int(cos(Theta)*100) will calculate the cosine of an angle, multiply this value by 100 using floating-point math, and then convert it to an integer, dropping any digits after the decimal place.

Comparing Values

You will quite often find that you wish to compare the value of one data with another, and make a decision based on the result. For example, you may wish to define a flag formula to show when a tank exceeds a particular value, or you may wish to use an If statement in a program to execute some code when a motor reaches its desired speed. The following comparison operators are provided:

Operator	Priority	Example
Equal To	Group 7	Data == 100
Not Equal To	Group 7	Data != 100
Greater Than	Group 6	Data > 100
Greater Than or Equal To	Group 6	Data >= 100
Less Than	Group 6	Data < 100
Less Than or Equal To	Group 6	Data <= 100

Each operator produces a value of 0 or 1, depending on the condition it tests. The operators can be used on integers, floating-point values, or text strings. If strings are compared, the comparison is case-insensitive ("abc" is considered equal to "ABC").

Testing Bits

DSI allows you to test the value of a bit within a data value by using the bit selection operator, which is represented by a single period. The left-hand side of the operator should be the value in which the bit is to be tested, and the right-hand side should be an expression indicating the bit number to test. This right-hand value should be between 0 and 31. The result of the operator is equal to 0 or 1 depending on the value of the bit in question.

Operator	Priority	Example
Bit Selection	Group 1	Input.2

The example shown tests bit 2 (the bit with a value of 4) within the indicated tag.

If you want to test for a bit being equal to zero, you can use the logical NOT operator:

Operator	Priority	Example
Logical NOT	Group 2	!Input.2

This example is equal to 1 if bit 2 of the indicated tag is equal to 0, and vice versa.

Multiple Conditions

If you want to define an expression that is true if a number of conditions are *all* true, you can use the logical AND operator. Similarly, if you want to define an expression that is true if *any* of a number of conditions are true, you can use the logical OR operator. The examples below show each operator in use:

Operator	Priority	Example
Logical AND	Group 11	A>10 && B>10
Logical OR	Group 12	A>10 || B>10

The logical AND operator produces a value of 1 if and only if the expressions on the left-hand and right-hand sides are true, while the logical OR operator produces a value of 1 if either expression is true. Note that, unlike the bitwise operators referred to elsewhere in this section, the logical operators stop evaluating once they know what the answer will be. This means that in the above example for logical AND, the right-hand side of the operator will only be evaluated if A is greater than 10, as, if this were not true, the result of the AND operator must already be zero. While this property makes little difference in the examples given above, if the left-hand or right-hand expressions call a program or make a change to a data value, this behavior must be taken into account.

Choosing Values

You may find situations where you want to select between two values (be they integers, floating-point values or strings) depending on the value of some condition. For example, you may wish to set a motor's speed equal to 500 rpm or 2000 rpm based on a flag tag. This operation can be performed using the **?:** operator, which is unique in that it takes three arguments, as shown in the example below:

Operator	Priority	Example
Selection	Group 13	Fast ? 2000 : 500

This example will evaluate to 2000 if **Fast** is true, and 500 otherwise. The operator can be thought to be equivalent to the **If** function found in applications such as Microsoft Excel.

Manipulating Bits

DSI also provides operators to perform operations that do not treat integers as numeric values, but instead as sequences of bits. These operators are known as bitwise operators.

Bitwise AND, OR, XOR

These three bitwise operators each produce a result in which each bit is defined to be equal to the corresponding bits in the values on the operator's left-hand and right-hand sides, combined using a specific truth-table:

Operator	Priority	Example
Bitwise AND	Group 8	Data & Mask
Bitwise OR	Group 9	Data | Mask
Bitwise XOR	Group 10	Data ^ Mask

The table below shows the associated truth tables:

A	B	A & B	A | B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Shift Operators

DSI also provides operators to shift an integer ***n*** bits to the left or right:

Operator	Priority	Example
Shift Left	Group 5	Data << 2
Shift Right	Group 5	Data >> 2

Each example shifts **Data** two bits in the specified direction.

Bitwise NOT

Finally, DSI provides a bitwise NOT operator to invert the sense of the bits in a value:

Operator	Priority	Example
Bitwise NOT	Group 2	~Mask

This example produces a value where every bit is equal to the opposite of its value in Mask.

Indexing Arrays

Elements within an array tag can be selected by following the array name with square brackets that contain an indexing expression. This expression must range from 0 to one less than the number of elements in the array. If you create a 10-element array, for example, the first element will be Name[0] and the last will be Name[9].

Indexing Strings

Square brackets can also be used to select characters within a string. For example, if you have a tag called Text that contains the string "ABCD", then the expression Text[0] will return a value of 65, this being equal to the ASCII value of the first character. Index values beyond the end of the string will always return zero.

Adding Strings

As well as adding numbers, the addition operator can be used to concatenate strings. Thus, the expression "AB"+"CD" evaluates to "ABCD". You may also use the addition operator to add an integer to a string, in which case a single character equal to the ASCII code represented by the integer is appended to the data in the string.

Calling Programs

Programs that return values may be invoked within expressions by following the program name with a pair of parentheses. For example, Program1()*10 will invoke the associated program, and multiply the return value by 10. Obviously, the return type for Program1 must be set to integer or floating-point for this to make sense.

Using Functions

DSI provides a number of predefined functions that can be used to access system information, or to perform common math operations. These functions are defined in detail in the Function Reference. They are invoked using syntax similar to that for programs, with any arguments to the function being enclosed within the parentheses. For example, cos(0) will invoke the cosine function with an argument of 0, returning a value of +1.0.

Priority Summary

The table below shows the priority of all the operators defined in this section:

Group	Operators
Group 1	.
Group 2	! ~
Group 3	* / %
Group 4	+ -
Group 5	<< >>
Group 6	< > <= >=
Group 7	== !=
Group 8	&
Group 9	|
Group 10	^
Group 11	&&
Group 12	||
Group 13	?:

Operators in the lower-numbered groups are applied first.

Writing Actions

While expressions are used to define values, actions are used to define what you want to happen when a trigger or other event occurs. Since the vast majority of the actions in a database will relate to key-presses, and since DSI provides a simple method of defining commonly-used actions via the dialog box discussed in the User Interface section, you will often be able to avoid writing actions "by hand". Actions are needed, though, if you want to use triggers, write programs, or use a key in User Defined mode.

Changing Pages

To create an action that changes the page shown on the panel's display, use the syntax **GotoPage(Name)**, where **Name** is the name of the display page in question. The current page will be removed, and the new page will be displayed in its place.

Changing Numeric Values

DSI provides several ways of changing data values.

Simple Assignment

To create an action that assigns a new value to a tag or to a register in a communications device, use the syntax **Data:=Value**, where **Data** is the data item to be changed, and **Value** is the value to be assigned. Note that **Value** need not just be a constant value, but can be any valid expression of the correct type. Refer to the previous section for details of how to write expressions. For example, code such as **[N7:0]:=Tank1+Tank2** can be used to add two tank levels and store the total quantity directly in a PLC register.

Compound Assignment

To create an action that sets a data value equal to its current value combined with another value by means of any of the operators defined in the previous section, use the syntax **Data***op***=Value**, where **Data** is the tag to be changed, **Value** is the value to be used by the operator, and *op* is any of the available operators. For example, the code **Tag+=10** will increase **Tag** by a value of 10, while **Tag*=10** will multiply the current value by 10.

Increment And Decrement

To create an action that increases a data value by one, use the syntax **Data++**. To create an action that decreases a tag by one, use the syntax **Data--**. Note that the ++ or -- operators may be placed before or after the data value in question. In the former case, the value of the expression represented by **++Data** is equal to the value of **Data** *after* it has been incremented. In the latter case, the expression is equal to the value *before* it has changed.

Changing Bit Values

To change a bit within a tag, use the syntax **Data.Bit:=1** or **Data.Bit:=0** to set or clear the bit as required, where **Data** is the tag in question and **Bit** is the zero-based bit number. Note again that the value on the right-hand side of the **:=** operator can be an expression if desired, such that an example such as Data.1:=(Level>10) can be used to set or clear a bit depending on whether or not a tank level exceeds a preset value.

Running Programs

Programs may be invoked within actions by following the program name with a pair of parentheses. For example, Program1() will invoke the associated program. The program will execute in the foreground or background as defined by the program's properties.

Using Functions

DSI provides a number of predefined functions that can be used to perform various operations. These functions are defined in detail in the Function Reference. They are invoked using syntax similar to that for programs, with any arguments to the function being enclosed within the parentheses. For example, SetLanguage(1) will set the terminal language to 1.

Operator Priority

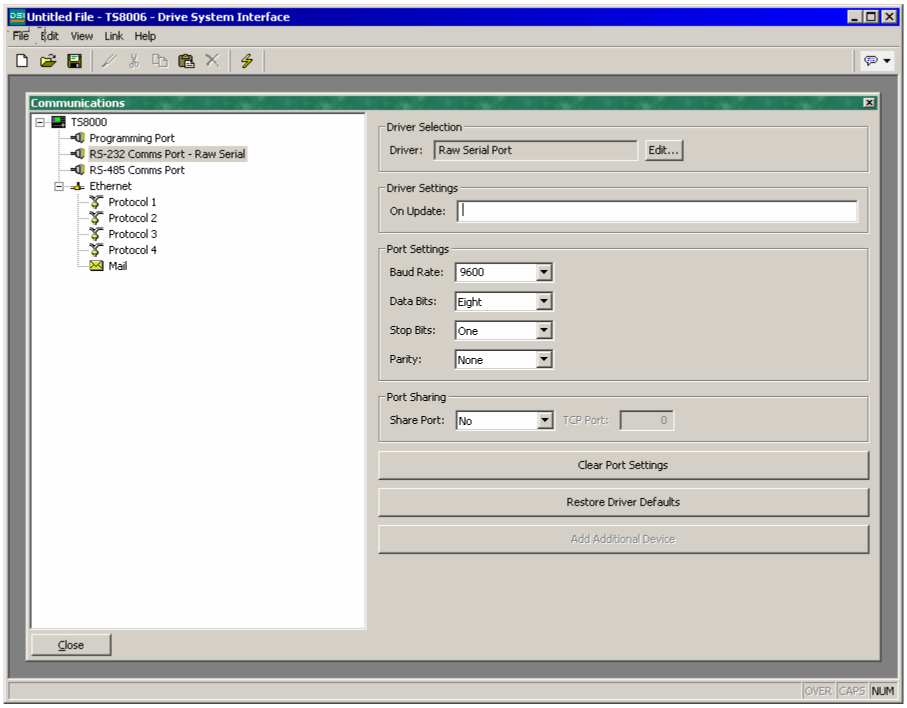
All assignment operators fall into Group 14. In other words, they will be evaluated after all other operators in an action. They are also unique in that they group right-to-left. This means that code such as Tag1:=Tag2:=Tag3:=0 can be used to clear all three tags at once.

Using Raw Ports

In order to allow customers to implement simple ASCII protocols without requiring custom drivers, DSI provides a new facility whereby the software's programming language can be used to directly control either serial ports or TCP/IP network sockets. This functionality is known as raw port access. It also replaces the General ASCII Frame protocol by providing a function to perform the parsing operations that the driver previously implemented. Note that if you are not using custom ASCII protocols, but are instead using the standard drivers provided with DSI, you can skip this section.

Configuring A Serial Port

To use a serial port in raw mode, select the Raw Serial Port driver as shown:



Driver: Raw Serial Port

On Update:

Baud Rate: 9600

Data Bits: Eight

Stop Bits: One

Parity: None

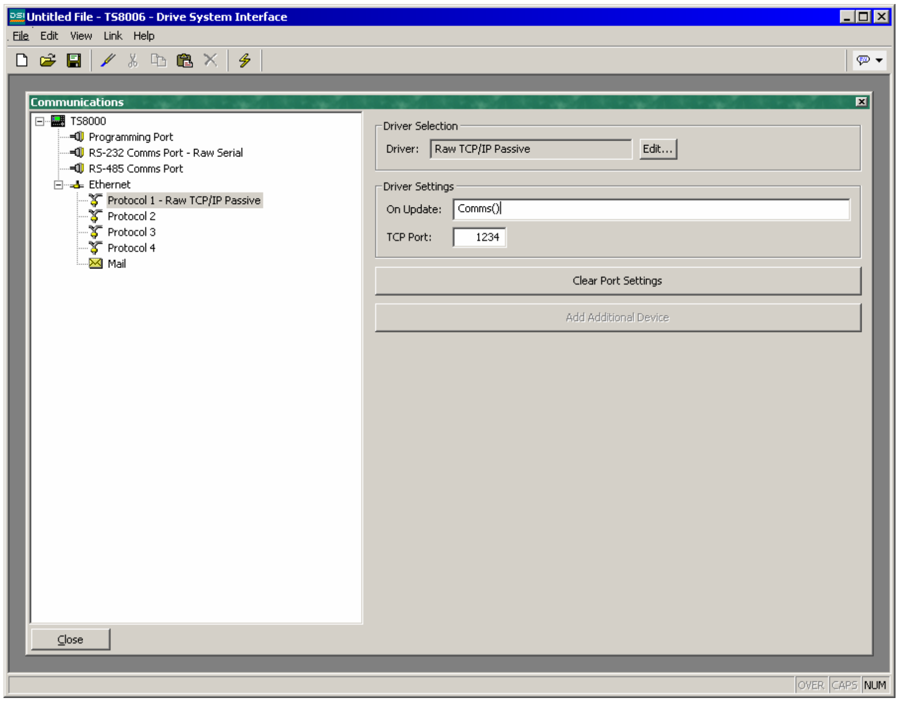
Share Port: No

TCP Port: 0

The port's Baud rate and other byte format parameters should be configured to indicate the required communications settings, and the On Update property should be set to specify the program that will be performing the communication. This program will be called continually by the port's communications task.

Configuring A TCP/IP Socket

To use a TCP/IP socket in raw mode, select the Raw TCP/IP Passive driver as shown:



Driver: Raw TCP/IP Passive

On Update: Comms()

TCP Port: 1234

The On Update property is configured as described above, while the Port property should be configured to indicate which TCP port you want the driver to monitor. The driver will accept connections on this port, and then call the On Update program to handle communications.

Reading Characters

To read data from a raw port a character at a time, use the **PortRead** function, as documented in the Function Reference section of this manual. As with all raw port functions, the *port* argument for this function is calculated by counting down the list of ports in the left-hand pane of the Communications window, with the programming port being port 1.

The example below shows to use **PortRead** to accept characters:

```
int Data;

for(;;) {

        if ((Data := PortRead(2, 100)) >= 0 ) {

                /* Add code to process data */
                }
        }
```

Note that by passing a non-zero value for the *period* argument, the need to call the **Sleep** function is removed. <u>If you use a zero value for this argument, you must make sure that you suspend the communications task at some point, or you will disrupt system operation.</u>

Reading Entire Frames

To read an entire frame from a raw port, use the **PortInput** function, as documented in the Function Reference section of this manual. This function allows you to specify frame delimiters, the required frame length and a frame timeout, thereby removing the need to write your own receive state machine. As sample program is shown below:

```
cstring input;
int     value;

for(;;) {

        input := PortInput(5, 42, 13, 0, 0);

        if ( value := TextToInt(input, 10) ) {

                Speed := value;

                PortPrint(5, "Value is ");
                PortPrint(5, IntToText(value,10,5));
                PortPrint(5, "\r\n");
                }
        }
```

The example above listens on a TCP/IP socket for a frame which starts with an asterisk and ends with a carriage return. It then converts the frame to a decimal value, stores this in an integer tag, and echoes the value back to the client.

Sending Data

To send data on a raw port, use the **PortWrite** or **PortPrint** functions, as documented in the Function Reference section of this manual. The first function sends a single byte, while the second function sends an entire string. To send numeric values, use the **IntToText** function to convert them into strings.

System Variable Reference

The following pages describe the various system variables that exist within DSI8000. These system variables can be invoked within actions or expressions as described in the previous two chapters.

How Are System Variables Used

System variables are used either to reflect the state of the system, or to modify the behavior of the system in some way. The former type of variable will be read-only, while the latter type can have a value assigned to it.

ActiveAlarms

Argument	Type	Description
value	int	Returns a count of the currently active alarms.

Description

Returns a count of the currently active alarms.

Variable Type

integer.

Access Type

Read only.

CommsError

Argument	Type	Description
value	int	Returns a status bit mask.

Description

Returns a bit mask indicating whether or not each communications device is offline. A value of 1 in a given bit position indicates that the corresponding device is experiencing comms errors. Bit 0 (ie. the bit with a value of 1) corresponds to the first communication device.

Variable Type

integer.

Access Type

Read only.

DispBrightness

Argument	Type	Description
value	int	Number indicating display brightness.

Description

Returns a number indicating the brightness of the display from 0 to 100 expressed as a percent, with zero being off.

Variable Type

integer.

Access Type

Read / write.

DispContrast

Argument	Type	Description
value	int	Number indicating display contrast as a percent.

Description

Returns a number indicating the amount of display contrast from 0 to 100 expressed as a percent, with zero being no contrast.

Variable Type

integer.

Access Type

Read / write.

DispCount

Argument	Type	Description
value	int	Number indicating display updates.

Description

Returns a number indicating the number of display updates since last reset.

Variable Type

integer.

Access Type

Read / write.

DispUpdates

Argument	Type	Description
value	int	Number indicating display update rate.

Description

Returns a number indicating how fast the display updates.

Variable Type

integer.

Access Type

Read only.

IsSirenOn

Argument	Type	Description
value	int	Status indicating siren status.

Description

Returns true if the panel's sounder is on or false otherwise.

Variable Type

integer.

Access Type

Read only.

PI

Argument	Type	Description
value	float	Number 3.14159274.

Description

Returns *pi* as a floating-point number.

Variable Type

Floating point.

Access Type

Read only.

Function Reference

The following pages describe the various functions that exist within DSI. These functions can be invoked within actions or expressions as described in the previous two chapters. Functions that are marked as *active* may not be used in expressions that are not allowed to change values (in the controlling expression of a display object). Functions that are marked as *passive* may be used in any context.

Abs(*value*)

Argument	Type	Description
value	int / float	The value to be processed.

Description

Returns the absolute value of the argument. In other words, if ***value*** is a positive value, that value will be returned; if ***value*** is a negative value, a value of the same magnitude but with the opposite sign will be returned.

Function Type

This function is passive.

Return Type

int or float, depending on the type of the ***value*** argument.

Example

```
Error := abs(PV – SP)
```

acos(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns the angle ***theta*** in radians such that **cos(*theta*)** is equal to ***value***.

Function Type

This function is passive.

Return Type

float.

Example

theta := acos(1.0)

asin(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns the angle *theta* in radians such that sin(*theta*) is equal to *value*.

Function Type

This function is passive.

Return Type

float.

Example

```
theta := asin(1.0)
```

atan(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns the angle ***theta*** in radians such that tan(*theta*) is equal to ***value***.

Function Type

This function is passive.

Return Type

float.

Example

theta := atan(1.0)

atan2(*a*,*b*)

Argument	Type	Description
a	float	The value of the side that is opposite the angle theta.
b	float	The value of the side that is adjacent to the angle theta

Description

This function is equivalent to atan(*a*/*b*), except that it also considers the sign of both *a* and *b*, and ensures that the return value is in the appropriate quadrant. It is also capable of handling a zero value for *b*, thereby avoiding the infinity that would result if the single-argument form of tan were used instead.

Function Type

This function is passive.

Return Type

float.

Example

theta := atan2(1,1)

Beep(*freq*, *period*)

Argument	Type	Description
freq	int	The required frequency in semitones.
period	int	The required period in milliseconds.

Description

Sounds the terminal's beeper for the indicated period at the indicated pitch. Passing a value of zero for *period* will turn off the beeper. Beep requests are not queued, so calling the function will immediately override any previous calls. For those of you with a musical bent, the *freq* argument is calibrated in semitones. On a more serious "note", the Beep function can be a useful debugging aid, as it provides an asynchronous method of signaling the handling of an event, or the execution of a program step.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Beep(60, 100)
```

ClearEvents()

Argument	Type	Description
None		

Description

Clears the list of events displayed in the event log.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
ClearEvents()
```

CloseFile(*file*)

Argument	Type	Description
file	int	File handle as returned by OpenFile.

Description

Closes a file previously opened in a call to FileOpen().

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
CloseFile(hFile)
```

CompactFlashEject()

Argument	Type	Description
None		

Description

Ceases all access of the CompactFlash card, allowing safe removal of the card.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
CompactFlashEject()
```

CompactFlashStatus()

Argument	Type	Description
None		

Description

Returns the current status of the CompactFlash slot as an integer.

Value	State	Description
0	Empty	Either no card is installed or the card has been ejected via a call to the **CompactFlashEject** function.
1	Invalid	The card is damaged, incorrectly formatted or not formatted at all. Remember only FAT16 is supported.
2	Checking	The TS8000 is checking the status of the card. This state occurs when a card is first inserted into the TS8000.
3	Formatting	The TS8000 is formatting the card. This state occurs when a format operation is requested by the programming PC.
4	Locked	The operator interface is either writing to the card, or the card is mounted and Windows is accessing the card.
5	Mounted	A valid card is installed, but it is not locked by either the operator interface or Windows.

Function Type

This function is passive.

Return Type

int.

Example

```
d := CompactFlashStatus()
```

ControlDevice(*device, enable*)

Argument	Type	Description
device	int	Device to be enabled or disabled.
enable	int	Determines if device is enabled or disabled.

Description

Allows the database to disable or enable a specified communications device. The number to be placed in the *device* argument to identify the device can be viewed in the status bar of the Communications category when the device name is highlighted.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
ControlDevice(1, true)
```

Copy(*dest*, *src*, *count*)

Argument	Type	Description
dest	int / float	The first array element to be copied to.
src	int / float	The first array element to be copied from.
count	int	The number of elements to be processed.

Description

Copies *count* array elements from *src* onwards to *dest* onwards.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Copy(Save[0], Work[0], 100)
```

cos(*theta*)

Argument	Type	Description
theta	float	The angle, in radians, to be processed.

Description

Returns the cosine of the angle *theta*.

Function Type

This function is passive.

Return Type

float.

Example

```
xp := radius*cos(theta)
```

CreateDirectory(*name*)

Argument	Type	Description
name	cstring	The directory to be created.

Description

Creates a new directory on the CompactFlash card. Note that the filing system used on the card does not support long filenames, and that if backslashes are included in the pathname to separate path elements, they must be doubled-up per the rules for string constants as described in the chapter on Writing Expressions. To avoid this complication, forward slashes can be used in place of backslashes with no need for such doubling. The function returns a value of one if it succeeds, and a value of zero if it fails.

Function Type

This function is active.

Return Type

int.

Example

```
Result := CreateDirectory(“/LOGS/LOG1”)
```

CreateFile(*name*)

Argument	Type	Description
name	cstring	The file to be created.

Description

Creates an empty file on the CompactFlash card. Note that the filing system used on the card does not support long filenames, and that if backslashes are included in the pathname to separate path elements, they must be doubled-up per the rules for string constants as described in the chapter on Writing Expressions. To avoid this complication, forward slashes can be used in place of backslashes with no need for such doubling. The function returns a value of one if it succeeds, and a value of zero if it fails. Note that the file is not opened after it is created. A subsequent call to **OpenFile()** must be made to read or write data.

Function Type

This function is active.

Return Type

int.

Example

```
Success := CreateFile("/LOGS/CUSTOM/MYFILE.TXT")
```

DataToText(*data, limit*)

Argument	Type	Description
data	int	The first element in an array.
limit	int	The number of elements to process.

Description

Forms a string from array, taking each array element to be a single ASCII character.

Function Type

This function is active.

Return Type

cstring.

Example

```
String := DataToText(Data[0], 8)
```

Date(*y, m, d*)

Argument	Type	Description
y	int	The year to be encoded, in four-digit form.
m	int	The month to be encoded, from 1 to 12.
d	int	The date to be encoded, from 1 upwards.

Description

Returns a value representing the indicated date as the number of seconds elapsed since the datum point of January 1, 1997. This value can then be used with other time/date functions.

Function Type

This function is passive.

Return Type

int.

Example

```
t := Date(2000,12,31)
```

DecToText(*data, signed, before, after, leading, group*)

Argument	Type	Description
data	int	Numeric data to be formatted.
signed	int	0 – unsigned, 1 – soft sign, 2 – hard sign.
before	int	Number of digits to the left of the decimal point.
after	int	Number of digits to the right of the decimal point.
leading	int	0 – leading zeros, 1 – no leading zeros.
group	int	0 – no grouping, 1– group digits in threes.

Description

Formats the value in *data* as a decimal value according to the rest of the parameters. The function is typically used to generate advanced formatting option via programs, or to prepare strings to be sent via a raw port driver.

Function Type

This function is passive.

Return Type

cstring.

Example

```
Text := DecToText(var1, 2, 5, 2, 1, 1)
```

Deg2Rad(*theta*)

Argument	Type	Description
theta	float	The angle to be processed.

Description

Returns *theta* converted from degrees to radians.

Function Type

This function is passive.

Return Type

float.

Example

Load := Weight * cos(Deg2Rad(Angle))

DeleteDirectory(*name*)

Argument	Type	Description
name	cstring	The directory to be deleted.

Description

Remove a directory, its subdirectories, and contents from the CompactFlash card. Note that the filing system used on the card does not support long filenames, and that if backslashes are included in the pathname to separate path elements, they must be doubled-up per the rules for string constants as described in the chapter on Writing Expressions. To avoid this complication, forward slashes can be used in place of backslashes with no need for such doubling. The function returns a value of one if it succeeds, and a value of zero if it fails.

Function Type

This function is active.

Return Type

int.

Example

```
Success := DeleteDirectory("/LOGS/CUSTOM")
```

DeleteFile(*name*)

Argument	Type	Description
name	cstring	The file to be deleted.

Description

Closes then deletes a file from the CompactFlash card. Note that the filing system used on the card does not support long filenames, and that if backslashes are included in the pathname to separate path elements, they must be doubled-up per the rules for string constants as described in the chapter on Writing Expressions. To avoid this complication, forward slashes can be used in place of backslashes with no need for such doubling. The function returns a value of one if it succeeds, and a value of zero if it fails.

Function Type

This function is active.

Return Type

int.

Example

Result := DeleteFile(hFile)

DevCtrl(*device, function, data*)

Argument	Type	Description
device	int	The index of the device to be controlled.
function	int	The required function to be executed.
data	cstring	Any parameter for the function.

Description

This function is to be used to perform a specific operation on a communications device. The number to be placed in the ***device*** argument to identify the device can be viewed in the status bar of the Communications section of DSI8000 when the device itself is highlighted in the tree on the left. The specific action to be performed is indicated by the ***function*** parameter, the values of which will depend upon the type of device being addressed. The ***data*** parameter may be used to pass additional information to the driver. Most drivers do not support this function. Where supported, the operations are driver specific, and are documented separately.

Function Type

This function is active.

Return Type

int.

Example

Refer to the comms driver application notes for specific examples.

DisableDevice(*device*)

Argument	Type	Description
device	int	The device to be disabled.

Description

Disables communications for the specified device. The number to be placed in the *device* argument to identify the device can be viewed in the status bar of the Communications category when the device name is highlighted.

Function Type

The function is passive.

Return Type

This function does not return a value.

Example

```
DisableDevice(1)
```

DispBrightness

Argument	Type	Description
value	int	Number indicating display brightness.

Description

Returns a number indicating the brightness of the display from 0 to 100 expressed as a percent, with zero being off.

Variable Type

integer.

Access Type

Read / write.

DispContrast

Argument	Type	Description
value	int	Number indicating display contrast as a percent.

Description

Returns a number indicating the amount of display contrast from 0 to 100 expressed as a percent, with zero being no contrast.

Variable Type

integer.

Access Type

Read / write.

DispOff()

Argument	Type	Description
None	float	Turns backlight to display off.

Description

Turns backlight to display off.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
DispOff()
```

DispOn()

Argument	Type	Description
None		Turns backlight to display on..

Description

Turns backlight to display on.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
DispOn()
```

DispUpdates

Argument	Type	Description
value	int	Number indicating display update rate.

Description

Returns a number indicating how fast the display updates.

Variable Type

integer.

Access Type

Read only.

DrvCtrl(*port, function, data*)

Argument	Type	Description
port	int	The index of the driver to be controlled.
function	int	The required function to be executed.
data	cstring	Any parameter for the function.

Description

This function is to be used to perform a specific operation on a communications driver. The number to be placed in the ***port*** argument to identify the driver is the port number to which the driver is bound. The specific action to be performed is indicated by the ***function*** parameter, the values of which will depend upon the driver itself. The ***data*** parameter may be used to pass additional information to the driver. Most drivers do not support this function. Where supported, the operations are driver specific, and are documented separately.

Function Type

This function is active.

Return Type

int.

Example

Refer to the comms driver application notes for specific examples.

EnableDevice(*device*)

Argument	Type	Description
device	int	The device to be enabled.

Description

Enables communications for the specified device. The number to be placed in the *device* argument to identify the device can be viewed in the status bar of the Communications category when the device name is highlighted.

Function Type

This function is passive.

Return Type

This function does not return a value.

Example

```
EnableDevice(1)
```

exp(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns e (2.7183) raised to the power of *value*.

Function Type

This function is passive.

Return Type

float.

Example

Variable2 := exp(1.609)

exp10(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns 10 raised to the power of ***value***.

Function Type

This function is passive.

Return Type

float.

Example

Variable4 := exp10(0.699)

Fill(*element, data, count*)

Argument	Type	Description
element	int / float	The first array element to be processed.
data	int / float	The data value to be written.
count	int	The number of elements to be processed.

Description

Sets ***count*** array elements from ***element*** onwards to be equal to ***data***.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Fill(List[0], 0, 100)
```

Find(*string,char,skip*)

Argument	Type	Description
string	cstring	The string to be processed.
char	int	The character to be found.
skip	int	The number of times the character is skipped.

Description

Returns the position of *char* in *string*, taking into account the number of *skip* occurrences specified. The first position counted is 0. Returns -1 if *char* is not found. In the example below, the position of ":", skipping the first occurrence is 7.

Function Type

This function is passive.

Return Type

int

Example

```
Position := Find("one:two:three",':',1)
```

FindFileFirst(*dir*)

Argument	Type	Description
dir	cstring	Directory to be used in search.

Description

Returns the filename of name of the first file or directory located in the ***dir*** directory on the CompactFlash card. Returns an empty string if no files exist or if no card is present. This function can be used with the **FindFileNext** function to scan all files in a given directory.

Function Type

This function is active.

Return Type

cstring.

Example

Name := FindFileFirst("/LOGS/LOG1")

FindFileNext()

Argument	Type	Description
None		

Description

Returns the filename of the next file or directory in the directory specified in a previous call to the **FindFileFirst** function. Returns and empty string if no more files exist. This function can be used with the **FindFileFirst** function to scan all files in a given directory.

Function Type

This function is active.

Return Type

cstring.

Example

```
Name := FindFileNext()
```

FormatCompactFlash()

Argument	Type	Description
None		

Description

Formats the CompactFlash card in the terminal, thereby deleting all data on the card. You should thus ensure that the user is given appropriate warnings before this function is invoked.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
FormatCompactFlash()
```

GetDate (*time*) and Family

Argument	Type	Description
time	int	The time value to be decoded.

Description

Each member of this family of functions returns some component of a time/date value, as previously created by **GetNow**, **Time** or **Date**. The available functions are as follows:

Function	Description
GetDate	Returns the day-of-month portion of ***time***.
GetDay	Returns the day-of-week portion of ***time***.
GetDays	Returns the number of days in ***time***.
GetHour	Returns the hours portion of ***time***.
GetMin	Returns the minutes portion of ***time***.
GetMonth	Returns the month portion of ***time***.
GetSec	Returns the seconds portion of ***time***.
GetWeek	Returns the week-of-year portion of ***time***.
GetWeeks	Returns the number of weeks in ***time***.
GetWeekYear	Returns the week year when using week numbers.
GetYear	Returns the year portion of ***time***.

Note that **GetDays** and **GetWeeks** are typically used with the difference between two time values to calculate how long has elapsed in terms of days or weeks. Note also that the year returned by ***GetWeekYear*** is not always the same as that returned by ***GetYear***, as the former may return a smaller value if the last week of a year extends beyond year-end.

Function Type

These functions are passive.

Return Type

int.

Example

```
d := GetDate(GetNow() – 12*60*60)
```

GetInterfaceStatus(*port*)

Argument	Type	Description
interface	int	The interface to be queried.

Description

Returns a string indicating the status of the specified TCP/IP interface. Refer to the earlier chapter on Advanced Communications for details of how to calculate the value to be placed in the ***interface*** parameter, and of how to interpret the returned value.

Function Type

This function is passive.

Return Type

cstring.

Example

```
EthernetStatus := GetInterfaceStatus(1)
```

GetMonthDays(*y, m*)

Argument	Type	Description
y	int	The year to be processed, in four-digit form.
m	int	The month to be processed, from 1 to 12.

Description

Returns the number of days in the indicated month, accounting for leap years etc.

Function Type

This function is passive.

Return Type

int.

Example

```
Days := GetMonthDays(2000, 3)
```

GetNetGate(*port*)

Argument	Type	Description
port	int	The index of the Ethernet port. Must be zero.

Description

Reports the IP address of the port's default gateway as a dotted-decimal text string.

Function Type

This function is passive.

Return Type

cstring.

Example

```
Gate := GetNetGate(0)
```

GetNetId(*port*)

Argument	Type	Description
port	int	The index of the Ethernet port. Must be zero.

Description

Reports an Ethernet port's MAC address as 17-character text string.

Function Type

This function is passive.

Return Type

cstring.

Example

```
MAC := GetNetId(0)
```

GetNetIP(*Port*)

Argument	Type	Description
port	int	The index of the Ethernet port. Must be zero.

Description

Reports an Ethernet port's IP address as dotted-decimal text string.

Function Type

This function is passive.

Return Type

cstring.

Example

```
IP := GetNetIP(0)
```

GetNetMask(*port*)

Argument	Type	Description
port	int	The index of the Ethernet port. Must be zero.

Description

Reports an Ethernet port's IP address mask as a dotted-decimal text string.

Function Type

This function is passive.

Return Type

cstring.

Example

```
Mask := GetNetMask(0)
```

GetNow()

Argument	Type	Description
None		

Description

Returns the current time and date as the number of seconds elapsed since the datum point of January 1, 1997. This value can then be used with other time/date functions.

Function Type

This function is passive.

Return Type

int.

Example

```
t := GetNow()
```

GetNowDate()

Argument	Type	Description
None		

Description

Returns the number of seconds in the days that have passed since January 1, 1997.

Function Type

This function is passive.

Return Type

int.

Example

d: = GetNowDate()

GetNowTime()

Argument	Type	Description
None		

Description

Returns the time of day in terms of seconds.

Function Type

This function is passive.

Return Type

int.

Example

```
t := GetNowTime()
```

GetUpDownData(*data, limit*)

Argument	Type	Description
data	int	A steadily increasing source value.
limit	int	The number of values to generate.

Description

This function takes a steadily increasing value and converts it to a value that oscillates between 0 and *limit* – 1. It is typically used within a demonstration database to generate realistic looking animation, often by passing DispCount as the *data* parameter so that the resulting value changes on each display update. If the GetUpDownStep function is called with the same arguments, it will return a value indicating the direction of change of the data returned by GetUpDownData.

Function Type

This function is passive.

Return Type

int.

Example

```
Data := GetUpDownData(DispCount, 100)
```

GetUpDownStep(*data, limit*)

Argument	Type	Description
data	int	A steadily increasing source value.
limit	int	The number of values to generate.

Description

See GetUpDownData for a description of this function.

Function Type

This function is passive.

Return Type

int.

Example

```
Delta := GetUpDownStep(DispCount, 100)
```

GotoPage(*name*)

Argument	Type	Description
name	Display Page	The page to be displayed.

Description

Selects page ***name*** to be shown on the terminal's display.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

GotoPage(Page1)

GotoPrevious()

Argument	Type	Description
None		

Description

Causes the panel to return to the previous page shown on the terminal’s display.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
GotoPrevious()
```

HidePopup()

Argument	Type	Description
None		

Description

Hides the popup that was previously shown using **ShowPopup**.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
HidePopup()
```

IntToText(*data*, *radix*, *count*)

Argument	Type	Description
data	int	The value to be processed.
radix	int	The number base to be used.
count	int	The number of digits to generate.

Description

Returns the string obtained by formatting ***data*** in base ***radix***, generating ***count*** digits. The value is assumed to be unsigned, so if a signed value is required, use **Sgn** to decide whether to prefix a negative sign, and then use **Abs** to pass the absolute value to **IntToText**.

Function Type

This function is passive.

Return Type

cstring.

Example

PortPrint(1, IntToText(Value, 10, 4))

IsDeviceOnline(*device*)

Argument	Type	Description
device	int	Reports if device is online.

Description

Reports if device is online or not. As device is marked as offline if a repeated sequence of communications error have occurred. When a device is in the offline state, it will be polled periodically to see if has returned online.

Function Type

This function is passive.

Return Type

int.

Example

```
Okay := IsDeviceOnline(1)
```

Left(*string*, *count*)

Argument	Type	Description
string	cstring	The string to be processed.
count	int	The number of characters to return.

Description

Returns the first *count* characters from *string*.

Function Type

This function is passive.

Return Type

cstring.

Example

AreaCode := Left(Phone, 3)

Len(*string*)

Argument	Type	Description
string	cstring	The string to be processed.

Description

Returns the number of characters in *string*.

Function Type

This function is passive.

Return Type

int.

Example

```
Size := Len(Input)
```

Log(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns the natural log of ***value***.

Function Type

This function is passive.

Return Type

float.

Example

Variable1 := log(5.0)

Log10(*value*)

Argument	Type	Description
value	float	The value to be processed.

Description

Returns the base-10 log of *value*.

Function Type

This function is passive.

Return Type

float.

Example

```
Variable3 := log10(5.0)
```

MakeFloat(*value*)

Argument	Type	Description
value	int	The value to be converted.

Description

Reinterprets the integer argument as a floating-point value. This function does not perform a type conversion, but instead takes the bit pattern stored in the argument, and assumes that rather than representing an integer, it actually represents a floating-point value. It can be used to manipulate data from a remote device that might actually have a different data type from that expected by the communications driver.

Function Type

This function is passive.

Return Type

float.

Example

```
fp := MakeInt(n);
```

MakeInt(*value*)

Argument	Type	Description
value	float	The value to be converted.

Description

Reinterprets the floating-point argument as an integer. This function does not perform a type conversion, but instead takes the bit pattern stored in the argument, and assumes that rather than representing a floating-point value, it actually represents an integer. It can be used to manipulate data from a remote device that might actually have a different data type from that expected by the communications driver.

Function Type

This function is passive.

Return Type

int.

Example

```
n := MakeInt(fp);
```

Max(*a*, *b*)

Argument	Type	Description
a	int / float	The first value to be compared.
b	int / float	The second value to be compared.

Description

Returns the larger of the two arguments.

Function Type

This function is passive.

Return Type

int or float, depending on the type of the arguments.

Example

```
Larger := Max(Tank1, Tank2)
```

Mean(*element*, *count*)

Argument	Type	Description
element	int / float	The first array element to be processed.
count	int	The number of elements to be processed.

Description

Returns the mean of the ***count*** array elements from ***element*** onwards.

Function Type

This function is passive.

Return Type

float.

Example

```
Average := Mean(Data[0], 10)
```

Mid(*string*, *pos*, *count*)

Argument	Type	Description
string	cstring	The string to be processed.
pos	int	The position at which to start.
count	int	The number of characters to return.

Description

Returns *count* characters from position *pos* within *string*, where 0 is the first position.

Function Type

This function is passive.

Return Type

cstring.

Example

Exchange := Mid(Phone, 3, 3)

Min(*a*, *b*)

Argument	Type	Description
a	int / float	The first value to be compared.
b	int / float	The second value to be compared.

Description

Returns the smaller of the two arguments.

Function Type

This function is passive.

Return Type

int or float, depending on the type of the arguments.

Example

```
Smaller := Min(Tank1, Tank2)
```

MulDiv(*a, b, c*)

Argument	Type	Description
a	int	First value.
b	int	Second value.
c	int	Third value.

Description

Returns a*b/c. The intermediate math is done with 64-bit integers to avoid overflows.

Function Type

This function is passive.

Return Type

int.

Example

```
d := MulDiv(a, b, c)
```

MuteSiren()

Argument	Type	Description
None		

Description

Turns off the operator panel's internal siren.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
MuteSiren()
```

Nop()

Argument	Type	Description
None		

Description

This function does nothing.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Nop()
```

OpenFile(*name, Mode*)

Argument	Type	Description
name	cstring	The file to be opened.
mode	int	The mode in which the file is to be opened.

Description

Returns a handle to the file ***name*** located on the CompactFlash card. This function is restricted to a maximum of four open files at any given time. The CompactFlash card cannot be unmounted while a file is open. The only acceptable ***mode*** parameter is currently zero, which indicates read-only access to an ASCII file.

Function Type

This function is active.

Return Type

int.

Example

hFile := OpenFile("/LOGS/LOG1/01010101.csv", 0)

PI

Argument	Type	Description
value	float	Number 3.14159274.

Description

Returns *pi* as a floating-point number.

Variable Type

Floating point.

Access Type

Read only.

PlayRTTTL (*tune*)

Argument	Type	Description
Tune	cstring	The tune to be played in RTTTL representation.

Description

Plays a tune using the terminal's internal beeper. The ***tune*** argument should contain the tune to be played in RTTTL format - the format used by a number of cell phones for custom ring tones. Sample tunes can be obtained from many sites on the World Wide Web.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
PlayRTTTL("TooSexy:d=4,o=5,b=40:16f,16g,16f,16g,16f.,16f,16g,16f,16g,16g#.,16g#,16g,16g#,16g,16f.,16f,16g,16f,16g,16f.,16f,16g,16f,16g,16f.,16f,16g,16f,16g,16g#.,16g#,16g,16g#,16g,16f.,16f,16g,16f,16g,32f.")
```

PopDev(*element*, *count*)

Argument	Type	Description
element	int / float	The first array element to be processed.
count	int	The number of elements to be processed.

Description

Returns the standard deviation of the ***count*** array elements from ***element*** onwards, assuming the data points to represent the whole of the population under study. If you need to find the standard deviation of a sample, use the **StdDev** function instead.

Function Type

This function is passive.

Return Type

float.

Example

Dev := PopDev(Data[0], 10)

PortClose(*port*)

Argument	Type	Description
port	int	Closes the specified port.

Description

This function is used in conjunction with the active or passive TCP raw port drivers to close the selected port by gracefully closing the connection that is attached to the associated socket.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
PortClose(6)
```

PortInput(*port*, *start*, *end*, *timeout*, *length*)

Argument	Type	Description
port	int	The raw port to be read.
start	int	The start character to match, if any.
end	int	The end character to match, if any.
timeout	int	The inter-character timeout in milliseconds, if any.
length	int	The maximum number of characters to read, if any.

Description

Reads a string of characters from the ***port*** indicated by port, using the various other parameters to control the input process. If ***start*** is non-zero, the process begins by waiting until the character indicated by this parameter is received. If ***start*** is zero, the receive process begins immediately. The process then continues until one of the following conditions has been met:

- ***end*** is non-zero and a character matching ***end*** is received.
- ***timeout*** is non-zero, and that period passes without a character being received.
- ***length*** is non-zero, and that many characters have been received.

The function then returns the characters received, not including the ***start*** or ***end*** byte. This function is used together with Raw Port drivers to implement custom protocols using DSI's programming language.

Function Type

This function is active.

Return Type

cstring.

Example

Frame := PortInput(1, '*', 13, 100, 200)

PortPrint(*port, string*)

Argument	Type	Description
port	int	The raw port to be written to.
string	cstring	The text string to be transmitted.

Description

Transmits the text contained in ***string*** to the port indicated by ***port***. The port must be configured to use a raw driver, such as the raw serial port driver, or either of the raw TCP/IP drivers. The data will be transmitted, and the function will return. The port driver will handle handshaking and control of transmitter enable lines as required.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
PortPrint(1, "ABCD")
```

PortRead(*port*, *period*)

Argument	Type	Description
port	int	The raw port to be read.
period	int	The time to wait in milliseconds.

Description

Attempts to read a character from the port indicated by *port*. The port must be configured to use a raw driver, such as the raw serial port driver, or either of the raw TCP/IP drivers. If no data is available within the indicated time period, a value of –1 will be returned. Setting *period* to zero will result in any queued data being returned, but will prevent DSI from waiting for data to arrive if none is available.

Function Type

This function is active.

Return Type

int.

Example

```
Data := PortRead(1, 100)
```

PortWrite(*port, data*)

Argument	Type	Description
port	int	The raw port to be written to.
data	int	The byte to be transmitted.

Description

Transmits the byte indicated by ***data*** on the port indicated by ***port***. The port must be configured to use a raw driver, such as the raw serial port driver, or either of the raw TCP/IP drivers. The character will be transmitted, and the function will return. The port driver will handle handshaking and control of transmitter enable lines as required.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
PortWrite(1, 'A')
```

PostKey(*code, transition*)

Argument	Type	Description
code	int	Key code.
transition	int	Transition code.

Description

Adds a physical key operation to the queue.

Function Type

This function is passive.

Return Type

Void.

Example

```
PostKey(0x80, 0)
```

Code	Key
0x80	Soft Key 1
0x81	Soft Key 2
0x82	Soft Key 3
0x83	Soft Key 4
0x84	Soft Key 5
0x85	Soft Key 6
0x86	Soft Key 7
0x90	Function Key 1
0x91	Function Key 2
0x92	Function Key 3
0x93	Function Key 4
0x94	Function Key 5

Code	Key
0x95	Function Key 6
0x96	Function Key 7
0x97	Function Key 8
0xA0	ALARMS
0xA1	MUTE
0x1B	EXIT
0xA2	MENU
0xA3	RAISE
0xA4	LOWER
0x09	NEXT
0x08	PREV
0x0D	ENTER

Transition	Operation
0	Post key down, then key up
1	Post key down only
2	Post key up only
3	Post key repeat only

Power(*value*, *power*)

Argument	Type	Description
value	int / float	The value to be processed.
power	int / float	The power to which ***value*** is to be raised.

Description

Returns ***value*** raised to the ***power***-th power.

Function Type

This function is passive.

Return Type

int or **float**, depending on the type of the ***value*** argument.

Example

```
Volume := Power(Length, 3)
```

Rad2Deg(*theta*)

Argument	Type	Description
theta	float	The angle to be processed.

Description

Returns *theta* converted from radians to degrees.

Function Type

This function is passive.

Return Type

float.

Example

```
Right := Rad2Deg(Pi()/2)
```

Random(*range*)

Argument	Type	Description
range	int	The range of random values to produce.

Description

Returns a pseudo-random value between 0 and *range*-1.

Function Type

This function is passive.

Return Type

int.

Example

Noise := Random(100)

ReadData(*array[element],count*)

Argument	Type	Description
None	int	Reads the indicated bit(s) out of the specified PLC.

Description

This function is used with mapped arrays which have their read policy set to *Manual*. It instructs DSI to read count elements from the indicated element array. The function will return immediately, and the read will be performed on the next comms scan.

Function Type

This function is active.

Return Type

int.

Example

ReadData(*data, count*)

Argument	Type	Description
data	any	First array element to be read.
count	int	Number of elements to be read.

Description

Requests that ***count*** elements from array element ***data*** onwards to read on the next comms scan. This function is used with arrays that have been mapped to external data, and which have their read policy set to *Read Manually*. The function returns immediately, and does not wait for the data to be read.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
ReadData(array1[8], 10)
```

ReadFile(*file, chars*)

Argument	Type	Description
file	int	File handle as returned by OpenFile.
chars	int	Number of characters to be read.

Description

Returns a string up to 512 characters in length from the specified file.

Function Type

This function is active.

Return Type

cstring.

Example

```
Text := ReadFileLine(hFile, 80)
```

ReadFileLine(*file*)

Argument	Type	Description
file	int	File handle as returned by OpenFile.

Description

Returns a single line of text from file.

Function Type

This function is active.

Return Type

cstring.

Example

```
Text := ReadFileLine(hFile)
```

RenameFile(*handle*, *name*)

Argument	Type	Description
handle	int	File handle.
name	cstring	New file name.

Description

Returns a non-zero value upon a successful file rename.

Function Type

This function is passive.

Return Type

int.

Example

```
Result := RenameFile(File, “NewName.txt”)
```

Right(*string*, *count*)

Argument	Type	Description
string	cstring	The string to be processed.
count	int	The number of characters to return.

Description

Returns the last *count* characters from *string*.

Function Type

This function is passive.

Return Type

cstring.

Example

```
Local := Right(Phone, 7)
```

Scale(*data, r1, r2, e1, e2*)

Argument	Type	Description
data	int	The value to be scaled.
r1	int	The minimum raw value stored in ***data***.
r2	int	The maximum raw value stored in ***data***.
e1	int	The engineering value corresponding to ***r1***.
e2	int	The engineering value corresponding to ***r2***.

Description

This function linearly scales the ***data*** argument, assuming it to contain values between ***r1*** and ***r2***, and producing a return value between ***e1*** and ***e2***. The internal math is implemented using 64-bit integers, thereby avoiding the overflows that might result if you attempted to scale very large values using DSI's own math operators.

Function Type

This function is passive.

Return Type

int.

Example

```
Data := Scale([D100], 0, 4095, 0, 99999)
```

SendMail(*rcpt, subject, body*)

Argument	Type	Description
rcpt	int	The recipient's index in the database's address book.
subject	cstring	The required subject line for the email.
body	cstring	The required boy text of the email.

Description

Sends an email from the operator interface. The function returns immediately, having first added the required email to the system's mail queue. The message will be sent to the mail server that was configured for the database.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
SendMail(1, "Test Subject Line", "Test Body Text")
```

Set(tag, value)

Argument	Type	Description
tag	int or real	The tag to be changed.
value	int or real	The value to be assigned.

Description

This function sets the specified tag to the specified value. It differs from the more normally used assignment operator in that it deletes any queued writes to this tag and replaces them with an immediate write of the specified value. It is thus used in situations where DSI8000's normal write behavior is not required.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Set(Tag1, 100)
```

SetLanguage(*code*)

Argument	Type	Description
code	int	The language to be selected.

Description

Set the terminal's current language to that indicated by *code*.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
SetLanguage(1)
```

SetNetConfig(port, addr, mask, gate)

Argument	Type	Description
port	int	The index of the Ethernet port. Must be zero..
addr	int	The required IP address for the port.
mask	int	The required netmask for the port.
gate	int	The required default gateway for the port.

Description

Overrieds the database settings for the Ethernet port. The various IP parameters are 32-bit integers that can optionally be fromed from strings using the TextToAddr() function. Note that setting all three of the IP values to zero will reset the port's settings to the database defaults. Note also that the unit must be power-cycled before the new values will take effect.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
SetNetConfig(0, 0, 0, 0)
```

SetNow(*time*)

Argument	Type	Description
time	int	The new time to be set.

Description

Sets the current time via an integer that represents the number of seconds that have elapsed since January 1, 1997. The integer is typically generated via the other time/date functions.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
SetNow(252288000)
```

Sgn(*value*)

Argument	Type	Description
value	int / float	The value to be processed.

Description

Returns –1 if ***value*** is less than zero, +1 if it is greater than zero, or 0 if it is equal to zero.

Function Type

This function is passive.

Return Type

int or **float**, depending on the type of the ***value*** argument.

Example

```
State := Sgn(Level)+1
```

ShowMenu(*name*)

Argument	Type	Description
name	Display Page	Display page to show as popup menu.

Description

Displays the page specified as a popup menu. This function is only available with on units fitted with touch-screens. Popup menus are shown on top of whatever is already on the screen, and are aligned with the left-hand side of the display.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
ShowMenu(Page2)
```

ShowPopup(*name*)

Argument	Type	Description
name	Display Page	The page to be displayed as a popup.

Description

Shows page ***name*** as a popup on the terminal's display. The popup will be centered on the display, and shown on top of the existing page. The popup can be removed by calling the **HidePopup()** function. It will also be removed if a new page is selected by using the **GotoPage** function or a suitably defined keyboard action.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
ShowPopup(Popup1)
```

sin(*theta*)

Argument	Type	Description
theta	float	The angle, in radians, to be processed.

Description

Returns the sine of the angle *theta*.

Function Type

This function is passive.

Return Type

float.

Example

```
yp := radius*sin(theta)
```

SirenOn()

Argument	Type	Description
None		

Description

Turns on the operator panel's internal siren.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

SirenOn()

Sleep(*period*)

Argument	Type	Description
period	int	The period for which to sleep, in milliseconds.

Description

Sleeps the current task for the indicated number of milliseconds. This function is normally used within programs that run in the background, or that implement custom communications using Raw Port drivers. Calling it in response to triggers or key presses is not recommended.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
Sleep(100)
```

Sqrt(*value*)

Argument	Type	Description
value	int / float	The value to be processed.

Description

Returns the square root of ***value***.

Function Type

This function is passive.

Return Type

int or **float**, depending on the type of the ***value*** argument.

Example

Flow := Const * Sqrt(Input)

StdDev(*element*, *count*)

Argument	Type	Description
element	int / float	The first array element to be processed.
count	int	The number of elements to be processed.

Description

Returns the standard deviation of the ***count*** array elements from ***element*** onwards, assuming the data points to represent a sample of the population under study. If you need to find the standard deviation of the whole population, use the **PopDev** function instead.

Function Type

This function is passive.

Return Type

float.

Example

```
Dev := StdDev(Data[0], 10)
```

StopSystem()

Argument	Type	Description
None		

Description

Stops the operator interface to allow a user to update the database. This function is typically used when serial programming is required with respect to a unit whose programming port has been allocated for communications. Calling this function shuts down all communications, and thereby allows the port to function as a programming port once more.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
StopSystem()
```

Strip(*text, target*)

Argument	Type	Description
text	cstring	The string to be processed.
target	int	The character to be removed.

Description

Removes all occurrences of a given character from a text string.

Function Type

This function is passive.

Return Type

cstring.

Example

Text1 := “Mississippi”

Text2 := Strip(Text1, ’s’)

Text2 now contains “Miiippi”.

Sum(*element*, *count*)

Argument	Type	Description
element	int / float	The first array element to be processed.
count	int	The number of elements to be processed.

Description

Returns the sum of the *count* array elements from *element* onwards.

Function Type

This function is passive.

Return Type

int or float, depending on the type of the *value* argument.

Example

```
Total := Sum(Data[0], 10)
```

tan(*theta*)

Argument	Type	Description
theta	float	The angle, in radians, to be processed.

Description

Returns the tangent of the angle *theta*.

Function Type

This function is passive.

Return Type

float.

Example

```
yp := xp * tan(theta)
```

TestAccess(*rights, prompt*)

Argument	Type	Description
rights	int	The required access rights.
prompt	cstring	The prompt to be used in the log-on popup.

Description

Returns a value of **true** or **false** depending on whether the current user has access rights defined by the ***rights*** parameter. This parameter comprises a bit mask representing the various user defined rights, with bit 0 (the bit with a value of 0x01) representing User Right 1, bit 1 (the bit with a value of 0x02) representing User Right 2 and so on. If no user is currently logged on, the system will display a popup to ask for user credentials, using the ***prompt*** argument to indicate why the popup is being displayed. The function is typically used in programs that perform a number of actions that might be subject to security, and that might otherwise be interrupted by a log-on popup. By executing this function before the actions are performed, you can provide a better indication to the user as to why a log-on is required, and you can avoid a security failure part way through a series of operations.

Function Type

This function is passive.

Return Type

Int.

Example

```
if (TestAccess(1, "Clear All Data?") )

        {

        Data1 := 0;

        Data2 := 0;

        Data3 := 0;

        }
```

TextToAddr(*addr*)

Argument	Type	Description
addr	cstring	The address in dotted-decimal form.

Description

Converts a dotted-decimal string into a 32-bit IP address.

Function Type

This function is passive.

Return Type

Int.

Example

```
IP := TextToAddr(“192.168.1.10”)
```

TextToFloat(*string*)

Argument	Type	Description
string	cstring	The string to be processed.

Description

Returns the value of ***string***, treating it as a floating-point number. This function is often used together with **Mid** to extract values from strings received from raw serial ports. It can also be used to convert other string values into floating-point numbers.

Function Type

This function is passive.

Return Type

float

Example

Data := TextToFloat("3.142")

TextToInt(*string*, *radix*)

Argument	Type	Description
string	cstring	The string to be processed.
radix	int	The number base to be used.

Description

Returns the value of ***string***, treating it as a number of base ***radix***. This function is often used together with **Mid** to extract values from strings received from raw serial ports. It can also be used to convert other string values into integers.

Function Type

This function is passive.

Return Type

int.

Example

```
Data := TextToInt("1234", 10)
```

Time(*h, m, s*)

Argument	Type	Description
h	int	The hour to be encoded, from 0 to 23.
m	int	The minute to be encoded, from 0 to 59.
s	int	The second to be encoded, from 0 to 59.

Description

Returns a value representing the indicated time as the number of seconds elapsed since midnight. This value can then be used with other time/date functions. It can also be added to the value produced by **Date** to produce a value that references a particular time and date.

Function Type

This function is passive.

Return Type

int.

Example

```
t := Date(2000,12,31) + Time(12,30,0)
```

UserLogOff()

Argument	Type	Description
none		

Description

Causes the current user to be logged-off the system. Any future actions that require security access rights will result in the display of the log-on popup to allow the entry of credentials.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
UserLogOff( )
```

UserLogOn()

Argument	Type	Description
None		

Description

Forces the display of the log-on popup to allow the entry of user credentials. You do not normally have to use this function, as DSI8000 will prompt for credentials when any action that requires security clearance is performed.

Function Type

This function is active.

Return Type

This function does not return a value.

Example

```
UserLogOn( )
```

WaitData(data, *count, time*)

Argument	Type	Description
data	any	First array element to be read.
count	int	Number of elements to be read.
time	int	The timeout period in milliseconds.

Description

Requests that ***count*** elements from array element ***data*** onwards to read on the next comms scan. This function is used with arrays that have been mapped to external data, and which have their read policy set to Read Manually. Unlike **ReadData()**, the function waits for up to the time specified by the ***time*** parameter in order to allow the data to be read. The return value is one if the read completed within that period, or zero otherwise.

Function Type

This function is active.

Return Type

int.

Example

```
Status := WaitData(array1[8], 10, 100)
```

WriteFile(*file, text*)

Argument	Type	Description
file	int	File handle as returned by OpenFile.
text	cstring	Text to be written to the File.

Description

Writes a string up to 512 characters in length to the specified file and returns the number of bytes successfully written.

Function Type

This function is active.

Return Type

int.

Example

```
Count := WriteFile(hFile, "Writing text to file.")
```

WriteFileLine(*file, text*)

Argument	Type	Description
file	int	File handle as returned by OpenFile.
text	cstring	Text to be written to the File.

Description

Writes a string to the specified file and returns the number of bytes successfully written, including the carriage return and line feed characters that will be appended to each line.

Function Type

This function is active.

Return Type

int.

Example

Count := WriteFileLine(hFile, “Writing text to file.”)

UK
SSD Drives Ltd
New Courtwick Lane
Littlehampton
West Sussex BN17 7RZ
Tel: +44 (0)1903 737000
Fax: +44 (0)1903 737100

CANADA
SSD Drives Inc
880 Laurentian Drive
Burlington
Ontario L7N 3V6
Tel: +1 (905) 333 7787
Fax: +1 (905) 632 0107

CHINA
SSD Drives Ltd
Room 1603, Hua Teng Edifice
302# Jin Song San Qu
Chaoyang District,
Beijing 100021
P.R. China

DENMARK
SSD Drives AB
Enghavevej 11
DK-7100
Vejle
Tel: +45 (0)70 201311
Fax: +45 (0)70 201312

FRANCE
SSD Drives SAS
15 Avenue de Norvège
Villebon sur Yvette
F-91953 Courtaboeuf Cedex
Paris
Tel: +33 - 1 69 18 51 51
Fax: +33 - 1 69 18 51 59

GERMANY
SSD Drives GmbH
Von-Humboldt-Strasse 10
64646 Heppenheim
Tel: +49 (6252) 798200
Fax: +49 (6252) 798205

ITALY
SSD Drives SPA
Via Gran Sasso 9
20030 Lentate Sul Seveso
Milano
Tel: +39 (0362) 557308
Fax: +39 (0362) 557312

SWEDEN
SSD Drives AB
Montörgaten 7, SE-302 60
Halmstad
Tel: +46 (0)35-17 73 00
Fax: +46 (0)35-10 84 07

U.S.A.
SSD Drives Inc
9225 Forsyth Park Drive
Charlotte
North Carolina 28273
Tel: +1 (704) 588 3246
Fax: +1 (704) 588 3249

08/31/05

Local availability and service support also in:

ARGENTINA • AUSTRALIA • AUSTRIA • BELGIUM • BRAZIL • CHILE • CHINA • COLOMBIA • CYPRUS
CZECH REPUBLIC • EGYPT • GREECE • HONG KONG • HUNGARY • ICELAND • INDONESIA • IRAN
IRELAND • ISRAEL • JAPAN • KENYA • KOREA • LITHUANIA • MALAYSIA • MOROCCO • NETHERLANDS
NEW ZEALAND • NORWAY • PHILIPPINES • POLAND • PORTUGAL • ROMANIA • SINGAPORE • SOUTH AFRICA
SPAIN • SWITZERLAND • TAIWAN • THAILAND • TURKEY • UNITED ARAB EMIRATES

Local Address

www.SSDdrives.com

DSI8000 Software Manual