Table of Contents

List of Tables

9. Graphics

This section covers the creation, modification and maintenance of the live Graphics Screens within the ***MacroView*** Package.

9.1 Introduction

A ***MacroView*** graphic:

- Is integrated into an easy-to-use framework of schematics, overviews etc.
- Can be called up by the ***MacroView*** Navigator.
- Is created from an initial CAD diagram and has dynamic objects embedded within it.
- Consists of static `background' graphics and live animated objects that reflect the real-world process information.

This document describes how these graphics are created. It also describes the tools used to create the graphics and how these tools are used in the graphics creation process.

Documentation

The table below summarises the documentation you will need to create effective graphics.

Table 1: Summary of Recommended Documentation

Document	Section	Description
AutoSketch User Manual	-	The CAD package used to create ***MacroView*** Graphics.
Engineering Manual	All Sections	Describes how the basic ***MacroView*** system is structured and how to configure the system. In particular you will need to understand the entity structures.
This chapter		A description of the tools and steps required to create effective graphics.
Meta script Chapter		A complete description of the meta script language. Meta scripts are used extensively in graphics.
Meta script Tracer User Manual (Not yet Completed)		The Tracer Tool allows you to efficiently debug meta scripts.

The next table summarizes the main sections in this chapter

Table 2: Summary of Sections in This Chapter

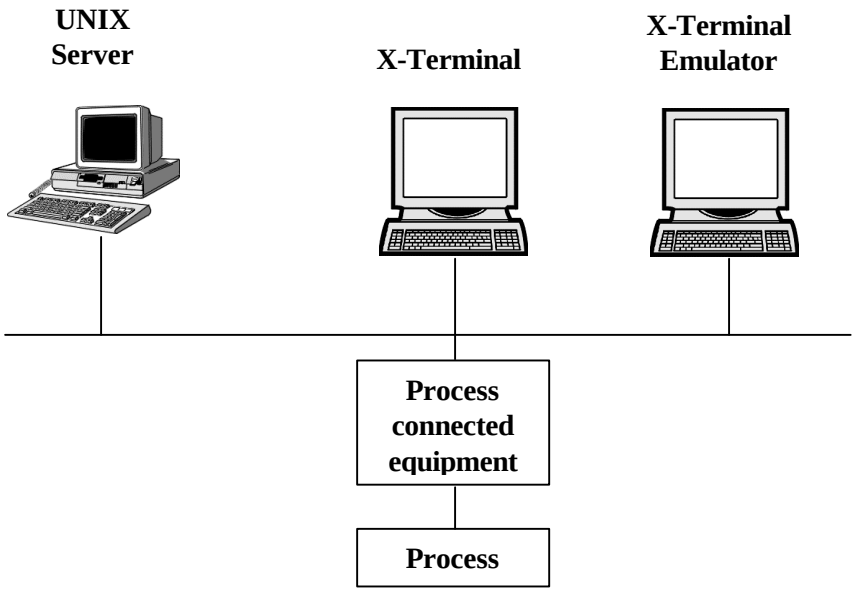
Title	Description
Introduction	Overview of this document and related documents.
Steps required to create a graphic	Overview of the steps required and how they relate to each other.
CAD development	Using the CAD package to define the background and the spatial layout of the objects.

Title	Description
Configuration	Allocating the graphics to page numbers and integrating them into the Navigator structure.
Convert 3	Converting the .dxf files to meta files.
ops3	Viewing the graphic, and initiating the object editor.
Using the Object Editor	Using the object editor to configure the graphics.
Static Reference	A section devoted to the static object features in the graphics.
Graphic Object Reference	A section that describes the individual objects and how they are configured.
Engineering Considerations	The section that deals with the engineering aspects such as the required files, programs, trouble shooting etc

Concepts of Displaying Graphics

Before we look at the actual mechanics of creating graphical images for ***MacroView***, a brief overview on the differences between UNIX and Windows NT systems, as far as display, generation and storage of graphic files is required.

Displaying *MacroView* Graphics on a UNIX System



The UNIX Server contains all the programs and displays to run the ***MacroView*** operations, configurator and graphics conversion utility. In addition to this the server can also have the AutoSketch CAD program installed, which allows the user to produce the drawing files used in the graphic displays. Programs such as ***MacroView*** operations, the configurator and the graphics conversion utility are always run on the UNIX server and, depending on your system layout, the displays for these programs are shown locally, at the server, or remotely at an X-Terminal or Microsoft Windows machine. The Microsoft Windows terminal will be running X-Terminal emulation software, such as PC-Xware and X-vision etc. These terminals utilise the X protocol, which

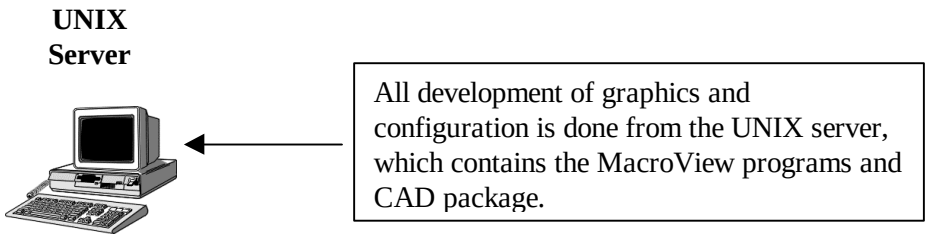
allows remote terminals to display and control programs running on the server and therefore contain no ***MacroView*** databases or programs.

When a Microsoft windows machine is being used as an engineering terminal, it is generally set up with the following tools:

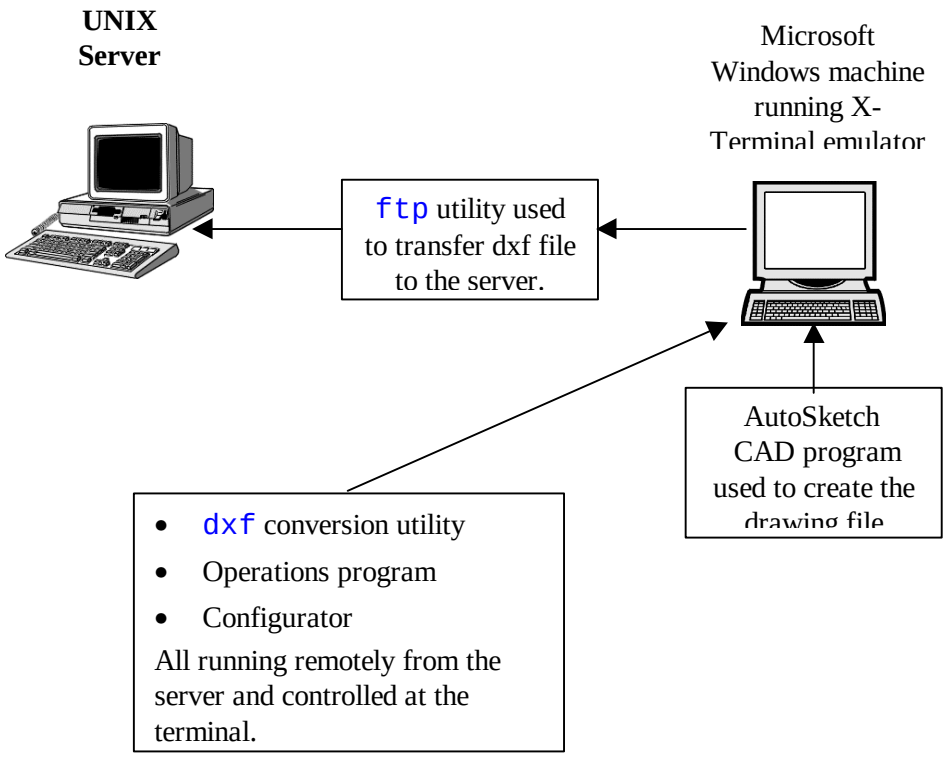
- X-Terminal emulation software. (PC-Xware, X-vision etc.), which provides a means to run remote operations, configuration, `dxf` conversions and a remote UNIX shell.
- An `ftp` (File Transfer Protocol) session to transfer the `.dxf` file to the UNIX server for conversion.
- AutoSketch for Windows

Generating *MacroView* Graphics on a UNIX System

In the UNIX system, the generation of graphics and configuration of the system can be carried out at the UNIX server or remotely, as described above.



When using remote terminals for system engineering the following diagram shows a typical scenario.

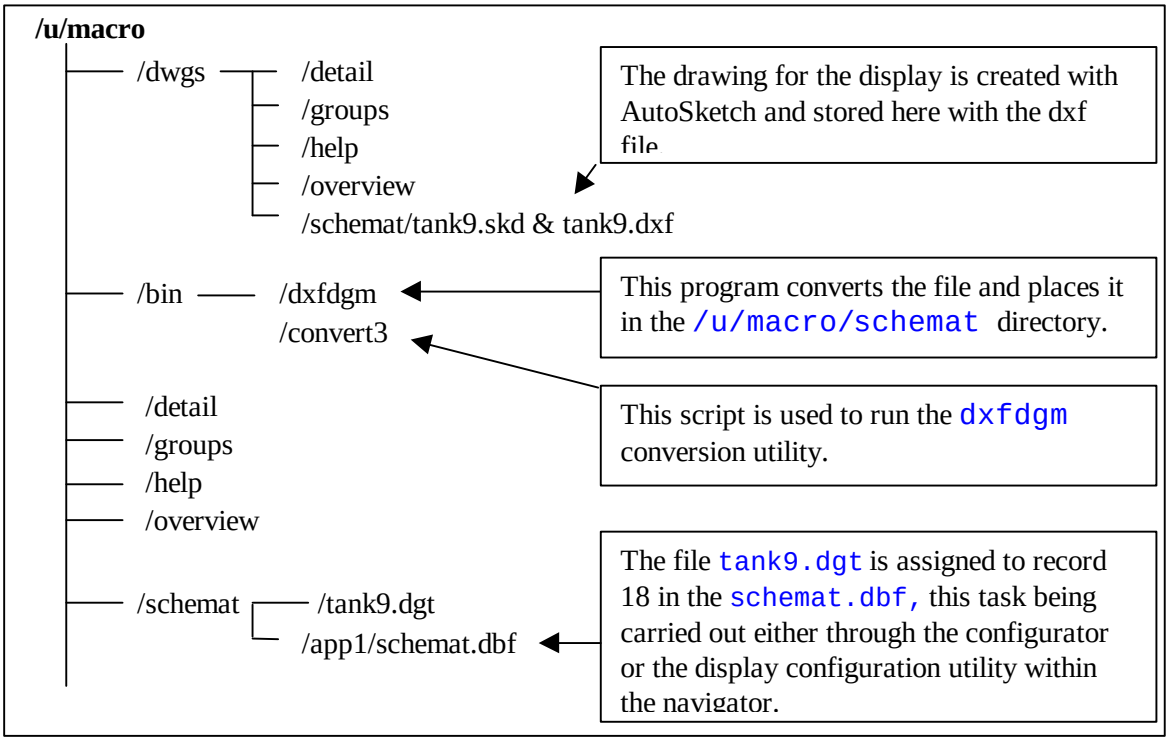


Steps for creating a display using X-Terminal emulator.

- The drawing is created in AutoSketch and then saved (`.skd`).
- Make or export a `dxf` file (`dxf`).
- Use ftp to transfer the `.skd` and `.dxf` files to the UNIX server. (The `dxf` file is the only one required, however, it is good house keeping to keep the `.skd` and `.dxf` files together)
- Run a remote `dxf` conversion session and convert the `dxf` to a metafile (`.dgt`)
- Run a remote configuration session and edit the display database if required this can also be done from the navigator.
- Run a remote operations session and call up your display for de-bugging.

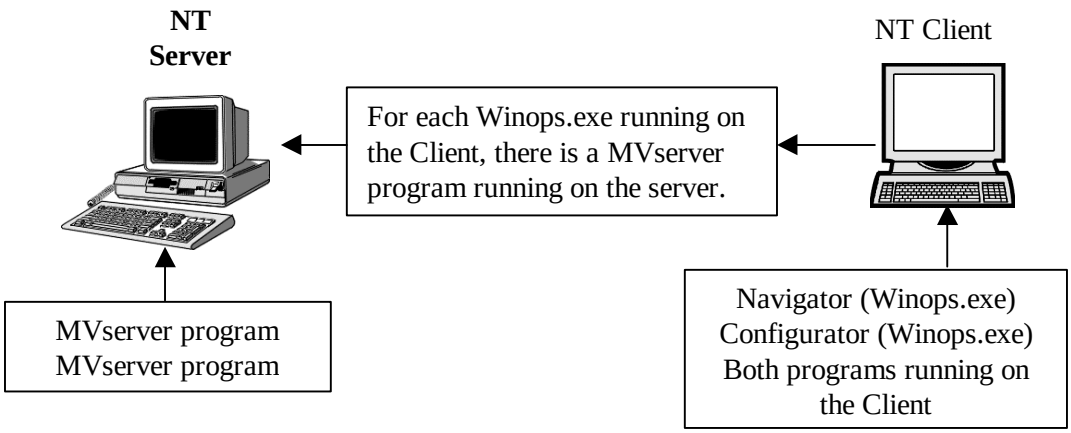
Directory Structure for UNIX System Graphics

The following diagram shows an example of a display, called tank9, which is to be displayed as page 18 of the Schematics application.



Displaying *MacroView* Graphics on an NT System

The Windows NT system uses the concept of Client/Server in a different way to the UNIX system, in that the NT Server runs a part of the ***MacroView*** programs and the Client runs the other part using its own copy of the displays. This ***MacroView*** Client, when installed, has its own directory and file structure containing the executable files for displaying ***MacroView*** graphic displays and converting drawing files to metafiles used in these displays. The ***MacroView*** Client can be installed on the NT server itself or on a remote Windows machine. The diagram below shows this concept.



In the NT system, when you start a ***MacroView*** graphics program on the Client, the Client runs a `winops.exe` and the server runs the `Mvserver` program. The Client is sent all the displays and metascript programs it requires for the Navigator and these files are stored in **the** `C:\Program Files\MacroView\Client\Bin\Cache\<hostname>` directory. When an application display is called up for the first time the `MVserver` program will send a copy of this file to the Client for storage in the Cache. This will continue to happen until such time as every display has been called and the Cache will have a copy of all these files.

Once the Client has a copy of a display file and this display is called up again, a call is made to the `MVserver` program to compare the date and time stamp of the file on the Client and Server. If the file on the Server is deemed to be a more recent file than the corresponding file on the Client, then the `MVserver` program sends this file to the Client again.

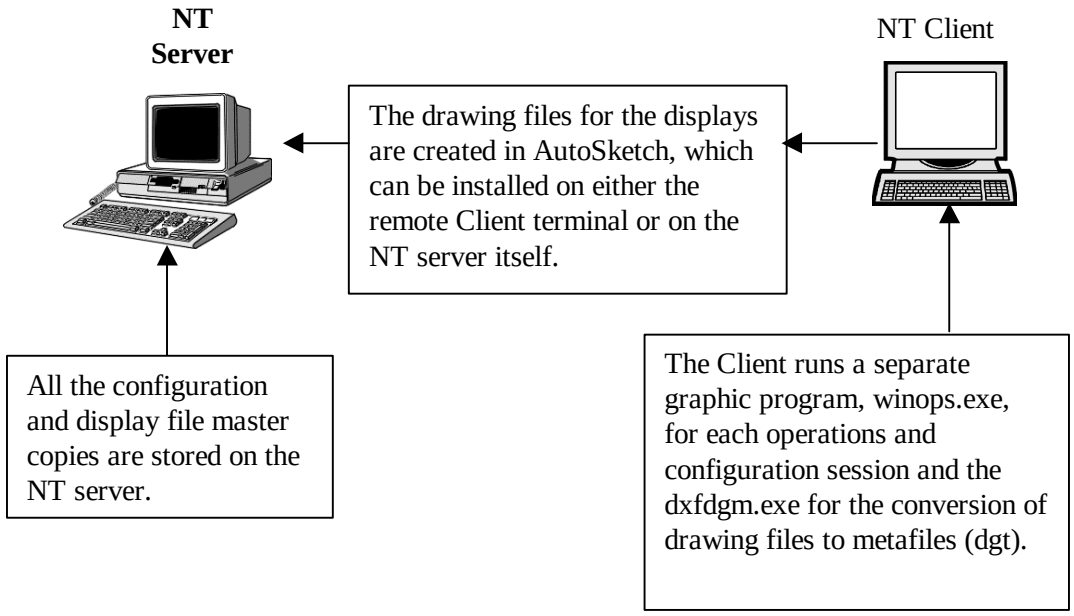
When a display is called up, a list of the entities and attributes required for this display is built and sent to the `MVserver` program. The `MVserver` program will then send the entity / attribute values to the Client and will then only update these values when it detects a change has occurred. When another display is called, the entity / attribute list is destroyed and a new list created for the display now being viewed.

Generating *MacroView* Graphics on an NT System

As shown in the above topic, the `MVserver` program runs on the Windows NT server and the ***MacroView*** Client can be running on the same machine or a remote machine. In either case the graphic display is first created in the drawing package i.e. AutoSketch, this drawing program being installed at the server or remote machine, depending on which machine is to be used as the engineering terminal.

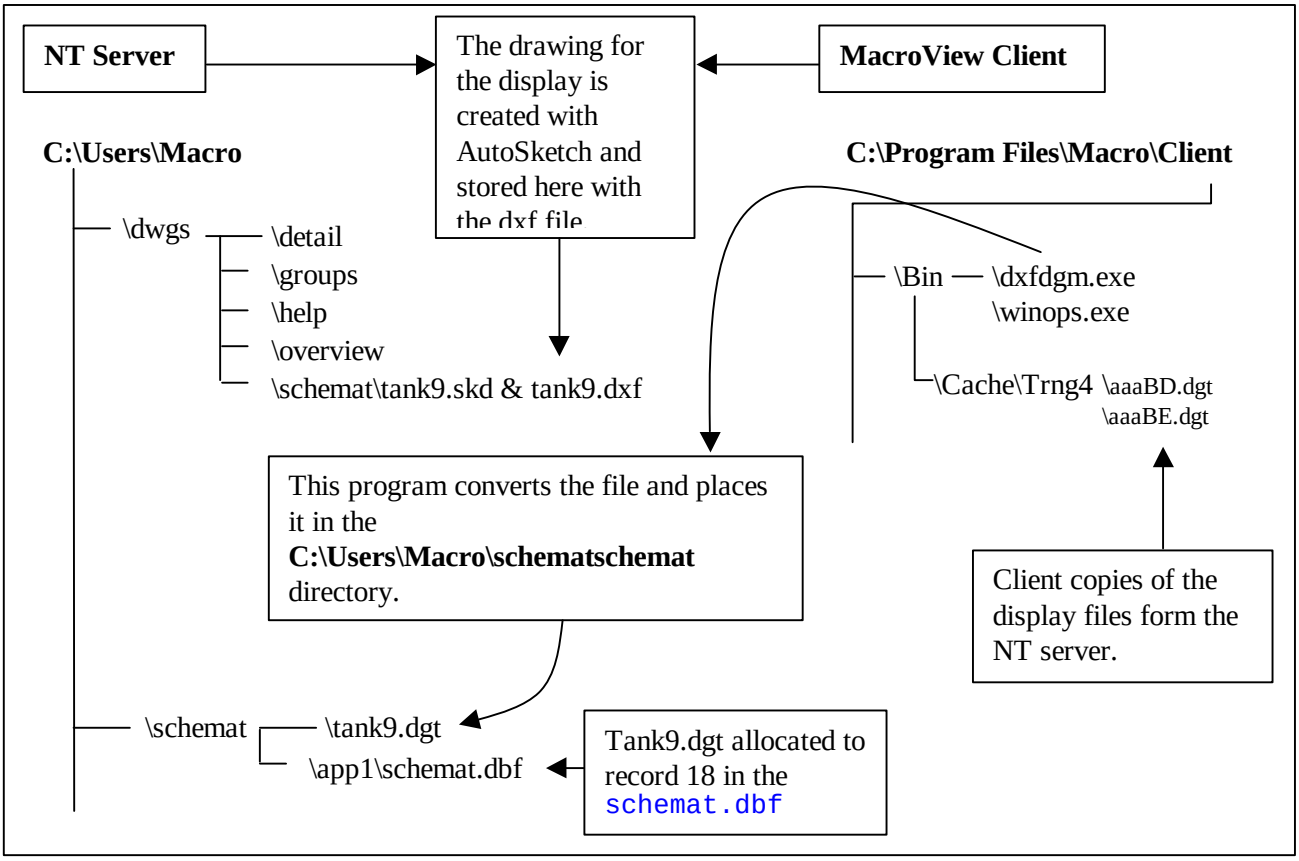
When you are using the remote terminal for engineering, it can be set up so that the drawing files reside on the NT server. The network pathing is set to that machine, or you can have the drawing files on the remote terminal and copy them to the server for conversion.

Once the drawing file is created and a dxf file produced, it is then converted to a .dgt file and stored in the `%HOME%\schemat` directory at the server for use within the graphic program you are using e.g. the ***MacroView*** navigator.



Directory Structure for NT System Graphics

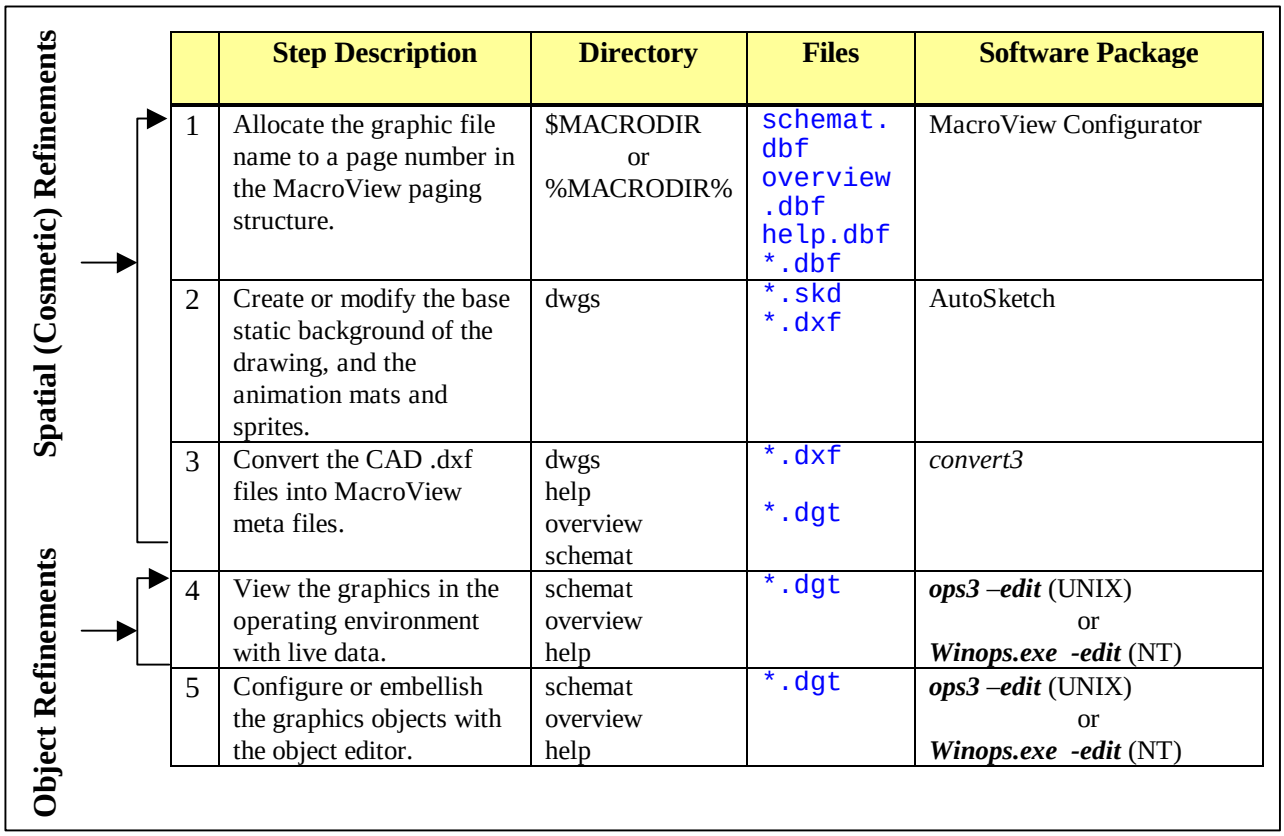
The following diagram shows an example of a display, called tank9, which is to be displayed as page 18 of the Schematics application.



General Steps Required to Create a Graphic

The diagram below shows the major steps required in creating a ***MacroView*** Graphic.

Table 3: Steps required to create a graphic



	Step Description	Directory	Files	Software Package
1	Allocate the graphic file name to a page number in the MacroView paging structure.	$MACRODIR or %MACRODIR%	`schemat.dbf` `overview.dbf` `help.dbf` `*.dbf`	MacroView Configurator
2	Create or modify the base static background of the drawing, and the animation mats and sprites.	dwgs	`*.skd` `*.dxf`	AutoSketch
3	Convert the CAD .dxf files into MacroView meta files.	dwgs help overview schemat	`*.dxf` `*.dgt`	*convert3*
4	View the graphics in the operating environment with live data.	schemat overview help	`*.dgt`	***ops3 –edit*** (UNIX) or ***Winops.exe -edit*** (NT)
5	Configure or embellish the graphics objects with the object editor.	schemat overview help	`*.dgt`	***ops3 –edit*** (UNIX) or ***Winops.exe -edit*** (NT)

Summary

Once you have allocated the graphic file name to a schematic, help or overview page, you should:

Refine the static (background) portion of the graphic and the spatial layout of the objects using the CAD package.

Refine the character of the objects using the graphic object editor.

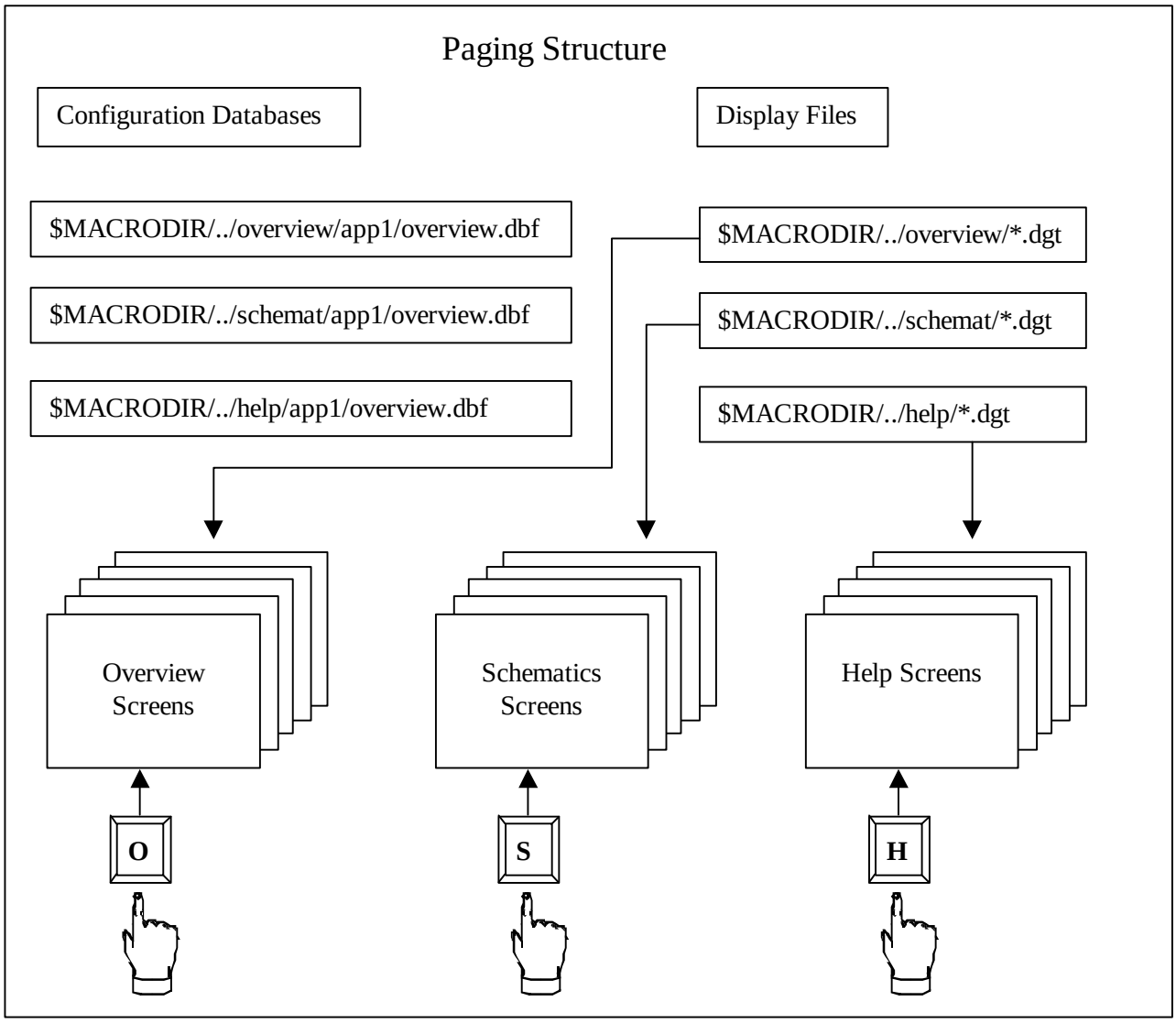
9.2 Allocating Graphics to Page Numbers

Overview

To help the user locate the graphics quickly, the graphics are assigned page numbers so that they can be integrated into the Navigator structure. Once the page numbers have been assigned:

i. The graphics will appear in the menu system.

ii. The Navigator will locate the graphics by page number.

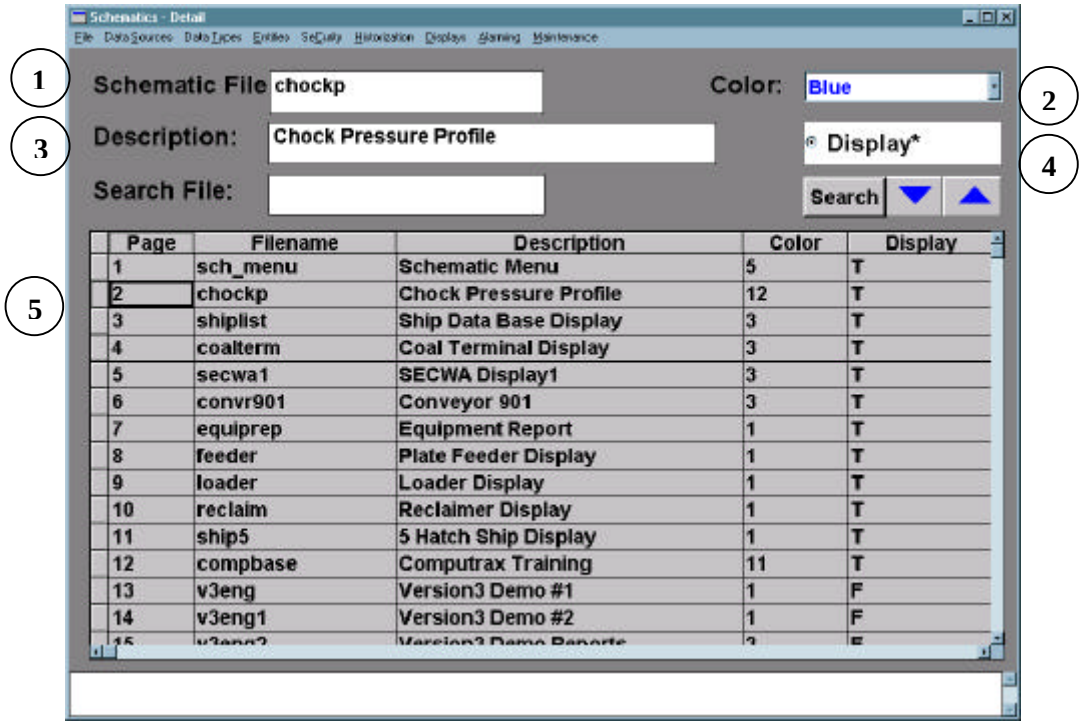


The free-form graphics are held in three separate directories. These graphics are called up by either clicking on the Navigator icons or pressing the Navigator quick keys.

9.3 Configuring the Graphic Page Numbers

The diagram below shows how the graphic files are allocated page numbers using the configurator. Just select ***Displays*** then ***Schematics: Detail***, ***Overview: Detail*** or ***Help:Detail***.

Graphic Page Number Allocation in the Configurator



Page	Filename	Description	Color	Display
1	sch_menu	Schematic Menu	5	T
2	chockp	Chock Pressure Profile	12	T
3	shiplist	Ship Data Base Display	3	T
4	coalterm	Coal Terminal Display	3	T
5	secwa1	SECWA Display1	3	T
6	convr901	Conveyor 901	3	T
7	equiprep	Equipment Report	1	T
8	feeder	Plate Feeder Display	1	T
9	loader	Loader Display	1	T
10	reclaim	Reclaimer Display	1	T
11	ship5	5 Hatch Ship Display	1	T
12	compbase	Computrax Training	11	T
13	v3eng	Version3 Demo #1	1	F
14	v3eng1	Version3 Demo #2	1	F

Items:

1. **File Name:** The name of the .dgt file (meta file). If it is in the schemat, overview or help directories, just enter the filename.
2. **Color:** This is the color in which the description will appear.
3. **Description:** This is the description that will appear in the relevant menu.
4. **Display:** Activate this button if you want the description to be put in the title bar.
5. **Page Numbers:** The record numbers shown in the Browse widget act as the page numbers to be used by the operators. For example, if the schematic *elevator* is on record 3 in the schemat database, then the operator would hit the schematic icon, then Page, then 3 to get to it.

Configuring the Page Numbers:

To allocate a graphic a page number:

1. From the menu, select ***Displays*** then ***Schematics:Detail***, ***Overview:Detail*** or ***Help:Detail***.
2. Click on the record in the Browse widget to edit a record or

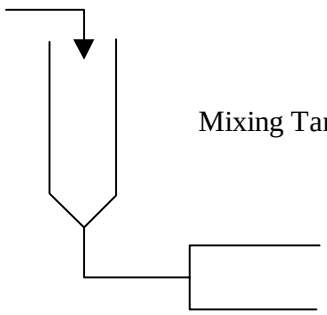
   Use the ***Add Blank*** or ***Add Like*** At option in the menu.
3. Modify the Description, Color and Filename of the record.
4. You may also search for a particular graphic using the Search facility. Simply type in the name of the graphic or part of the name of a graphic and click on the Search Up or Down keys.

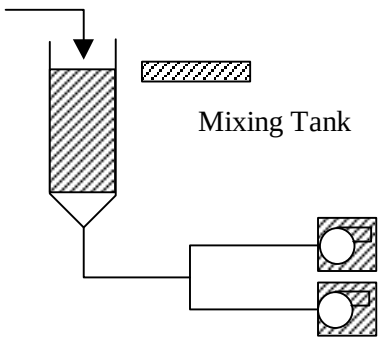
9.4 Creating and Modifying the CAD Diagram

Various elements of the ***MacroView*** graphic are created using a CAD package such as AutoSketch (see note below). These elements include:

i. The static (background) portions of the graphic (E.g. tank and pipe outlines as well as labels).

ii. The dynamic object mats and metasprite. Mats are rectangular areas on the graphic that show the outline (size and location) of the objects. Metasprites are CAD objects that change shape, colour and position with the process. Later, the various characteristics of these mats and sprites

Static (Background)
1. Background or Static information (e.g. Flow lines and vessel lines) is simply drawn onto the diagram using the mouse in the color you want. 2. Alternatively, you can import an existing .dxf file into AutoSketch. 3. Simply, draw the shapes with the mouse.
Creation of the static background 

Static (Background)
1. The outline of the dynamic objects (e.g. live flow values or dynamic color changers) is added to the graphic. These are the areas on the screen that move or change color to reflect the real world. The mats and sprites are added and animated. 2. Later, the characteristics of the object are embellished in the graphic object editor.
Addition of the Animation mats 
3. The ability to turn "dead" static graphics into live functioning screens depends on your ability to master the available graphic objects.

are developed in the graphics object editor.

Note: If you use a CAD package other than AutoSketch, you need to save your graphic as a .dxf file and then import it into AutoSketch for the addition of the dynamic graphic objects. Later, the list of fully supported CAD packages will be increased.

Before You Start

Before you start creating the graphic, we recommend the following:

i. You should be thoroughly familiar with the drawing package you are going to use. In particular, you should practice the following tasks:

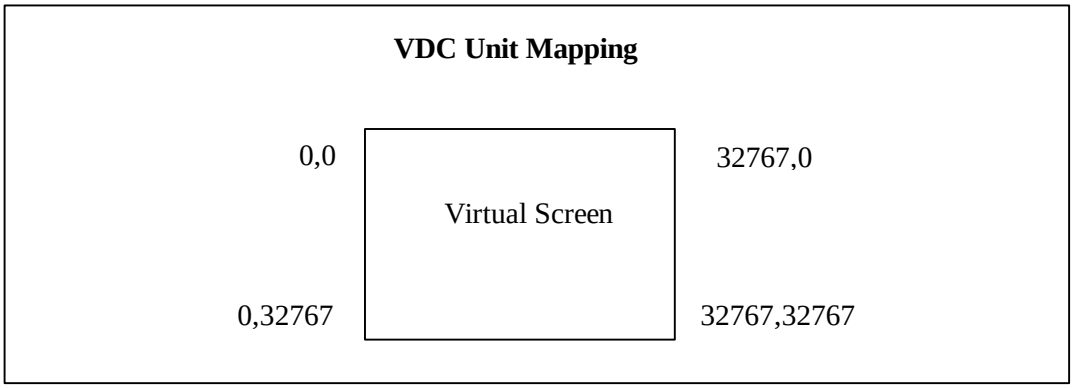
- Using snap, ortho and attach.
- Setting colours.
- Adding and editing text.
- Drawing lines, polylines and filled regions.
- Changing layers.
- Importing parts.
- Grouping and ungrouping objects.
- Storing the file in `.dxf` format.

ii. You should create the drawing in the `dwgs` directory under the ***MacroView*** directory - this is typically `/u/macro/dwgs`. The convert program will look for your drawing there.

iii. We recommend you set up the drawing size, text size, etc. to the default values shown on the next page.

VDC Units

In various parts of this manual, there are references to VDC units. VDC (Virtual Display Co-ordinates) map any screen to a set of x and y co-ordinates with 0,0 as the top left hand corner and 32767, 32767 as the bottom right hand corner. I.e.:



By using VDC units, the definition of a point on the screen is independent of the size and resolution of the screen.

Default Settings

The following settings are recommended as the basis for starting a new AutoSketch drawing. AutoSketch Version 2 for Windows is the preferred CAD package for drawing development.

Table 4: Default Setting for AutoSketch Drawings

Setting	Value	Extra Information
Attach	Node Point	ON
Colour	1 to 8	
Snap	0.1	ON
Grid	1	ON
Drawing Limits	20, 17 (For full screen, or others for different sized screens such as pop-ups.) Prepare a drawing sheet whose area covers the region that needs to be converted.	
Layer	Layer 1	All visible except layer 10
Line Type	Solid	Scale factor 1
Pattern	Solid	
Polyline	Width 0	Solid fill
Property	Colour only	
Selection Area	1 %	
Text (Screen)	Standard, Height 0.35, Width factor 0.75, Left Justified, Angle 0, Oblique Angle 0. (These can be bigger or smaller as desired.)	
Text (Identifiers)	Standard, Height 0.01, Width factor 1, Left Justified, Angle 0, Oblique Angle 0, color magenta, Layer 2.	
Units	Decimal, 1 decimal Absolute/relative showing	
Ortho	ON	
CAPS LOCK	ON	
Printer Settings	XPOINT	9.45

Note: We **strongly recommend** that you use at least a 486 -25 computer, or that you set the text display to show as rectangles.

Working with AutoSketch

The AutoSketch product from Auto Desk has been successfully used to produce graphic displays for ***MacroView*** for many years. Its success is based on the fact that AutoSketch is a cost effective and easy to learn package, which provides all the necessary tools to create graphic displays for use with ***MacroView***. Whether you use AutoSketch for DOS or Windows will depend on the platform you are using. For instance, when using SCO UNIX, and using the UNIX server as your engineering terminal, it is likely you will utilise the SCO Merge utility and run AutoSketch for DOS. If you are using Windows NT or any Microsoft Windows machine as an engineering terminal, you are likely to use AutoSketch for Windows.

Loading AutoSketch for Windows [1]

If you intend working with AutoSketch, you need Version 3.1, release 2.0 or greater. If using 2.0, you also need to load the FIX patch from Auto desk by:

i. First loading AutoSketch and register the copy for their FIX patch. This includes the language extensions.
   - The fix patch is sent following registration of your copy of AutoSketch.
   - This patch is required to use the toolbox extensions.

ii. Installing the patch as described with the disk.

Note: The patch is not required for version 2.1 of AutoSketch.

Loading the *MacroView* Toolbox

The toolbox option is used to provide a means of adding MAT's to your AutoSketch drawing, these MAT's are then defined as any of a range of dynamic graphic objects through the EZ-Eng utility.

To load the toolbox:

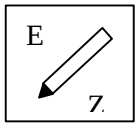
i. Place the ***MacroView ez-eng*** disk in the drive and type `d:setup` (use the appropriate drive letter). Follow the instructions that appear.

ii. Once the ***ez-eng*** icons appear, you ***must*** double click the README icon.

iii. The readme icon contains additional information for setting up the EZ-Eng system. Essentially, you must copy parameters to the start-up command line.

Starting AutoSketch

Run AutoSketch by double clicking on the EZ-Eng icon.



This will initialise the AutoSketch variables required to use the EZ-Eng toolbox.

Note: Do not use the EZ-Eng toolbox in AutoSketch unless it has been started via the EZ icon.

[1] Note: If you have to use an earlier AutoSketch version, it will be necessary for you to manually create the ID's for each widget. These ID's must be unique on a given graphic and must be grouped as described in the next section.

The EZ-Eng toolbox will not be loaded automatically on the first run of AutoSketch, it must be opened and will reside in the directory specified on installation of the EZ-Eng toolbox, typically:

```
C:\WSKETCH2\TOOLBOX
```

Types of Dynamic Objects

Once you have created the static parts of the graphic you can start adding the dynamic objects. Essentially there are three kinds of dynamic objects:

i. Dynamic Graphic Objects associated with mats (e.g. Browse widgets, buttons, etc.) Mats are rectangular shapes that show the size and position of the objects.

ii. Graphic Objects associated with CAD objects (metasprites).

iii. Objects having no graphic representation. (E.g. Cyclic and Trigger programs).

Adding Mats to the Graphic

A mat is a rectangular outline of a graphic object that defines the position and size of that graphic object like a Browse widget, slider, scrollbar, etc. Once a mat has been created in a display we can use the ***MacroView*** EZ-Eng utility to define the mat as on of a number of dynamic graphic objects. This EZ-Eng utility allows you to directly edit the display file (dgt) whilst operating in the ***MacroView*** graphic environment and, therefore, does not require continually returning to the CAD program to define the behaviour of graphic objects. The EZ-Eng functionality will be described in detail later in this chapter.

To add a mat and animate it so that it can be used by the graphics object editor:

i. Select the mat icon from the EZ-Eng toolbox.

ii. Draw the mat in the position and size of the desired object: or,

iii. Draw a box with the desired size and position of the object you wish to create. Manually add the modifier text defining the box as a mat. See the example below:

#M #I ACIDFLO

The box is drawn in the size of the dynamic graphic object, e.g. the size of a button you wish to create.
The `#M` modifier identifies the box as a mat and **the** `#I ACIDFLO` is a unique label, which is given to each mat.
HINT: Creating your mats in this way will allow you to give your own unique labels to each mat, rather than having the labels assigned by the EZ-Eng toolbox utility. It is occasionally useful to assign a meaningful label to a mat for debugging purposes.
The box and the text is grouped together using the AutoSketch grouping function and, when converted, the MacroView `xops3` program will see the object as a mat.

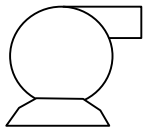
Adding Meta Sprites

Meta sprites are graphics objects that are associated with CAD objects and the ***MacroView*** EZ-Eng utility is used to define the behaviour of these objects in the display. For example, a pump icon that changes colour and shape is a metasprite. In essence, the creation of a metasprite includes:

i. Drawing the CAD parts of the object and grouping them in a single group.

ii. Adding a polyline to show the extents of angular action or a single line to show linear motion.

iii. Adding an identifier using the Ident icon, or manually typing in your own unique label and group the initial group, optional polyline or line, and the ID.



#METASPRITE
#I FDPUMP

The object is drawn in AutoSketch and grouped with the #METASPRITE text. The `#METASPRITE` modifier identifies the shape as the object you wish to apply the behavior to and **the** `#I FDPUMP` is a unique label, which is given to each mat or metasprite

NOTE: Creating your metasprite in this way will allow you to give your own unique labels to each mat, rather than having the labels assigned by the EZ-Eng toolbox utility. It is occasionally useful to assign a meaningful label to a metasprite for debugging purposes.

Note: For a more detailed description of metasprites, please see the detailed description in the graphics objects reference section.

Adding a Non-graphic Object

- Non graphic objects are objects that exist in the graphic but display no associated visual graphics. Examples of this include cyclic programs, colour tables etc.
- You need not add any elements in the CAD diagram.
- The objects are added later using the graphic object editor. You can modify the defaults using the associated graphic object editors.

Modifying a Mat or Meta Sprite

You may freely change the shape and position of the mat or metasprite.

A mat metasprite can not be copied as it contains an ID that would be duplicated

Saving the CAD Diagram as a `.dxf` File

i. Once you have created the CAD diagram and added the objects (mats, sprites and non-graphic objects), you should export the file in .dxf format using the AutoSketch export facility.

ii. The file should be stored in the `$MACRODIR/../dwgs` directory.

iii. Alternatively, you may create ***schemat***, ***overview*** and ***help*** subdirectories off the `dwgs` directory and save the `.dxf` (and `.skd`) files in these directories.

Deleting Mats and Sprites

To delete an object, you must delete the object in the CAD diagram.

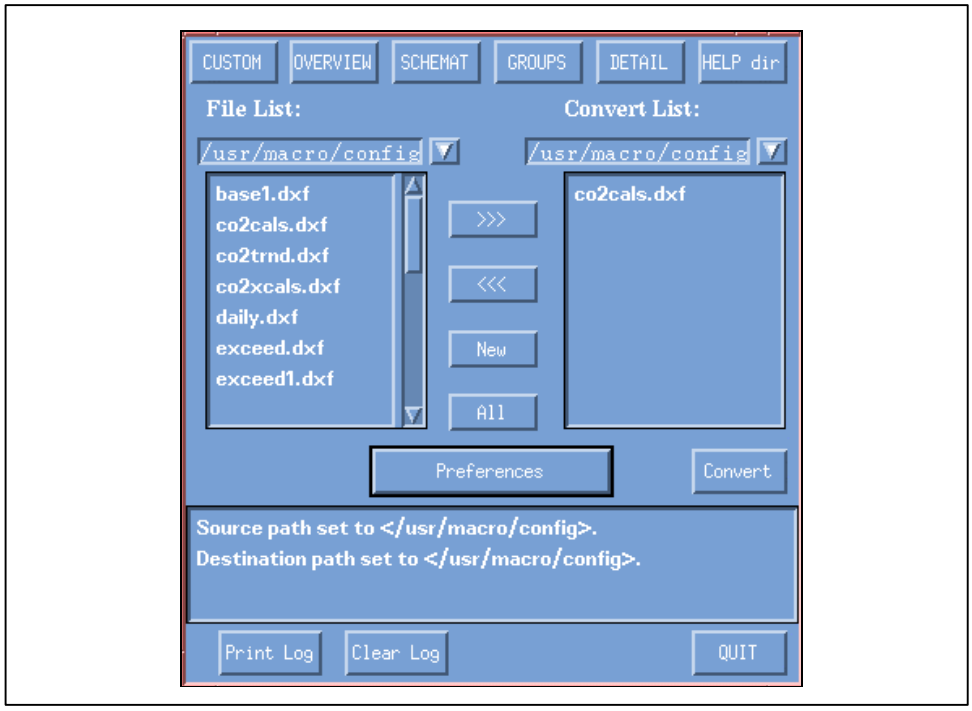
9.5 Converting .dxf Files to Meta Files

Once you have created your drawing in AUTOSKETCH and saved it in the `.dxf` format in the `dwgs` directory, you must now convert it to the metafile format.

The metafile format is compact, and can be displayed much faster than the `dxf` format, hence, it is ideal for the process control industry.

To convert from `dxf` to metafile, you need to do the following:

i. Start up the `convert3` program. This is done by clicking on the `convert3` icon (your system administrator will be able to set this up for you) or just type `convert3` at the UNIX prompt. The window displayed below will appear.



ii. Choose the files you want converted.

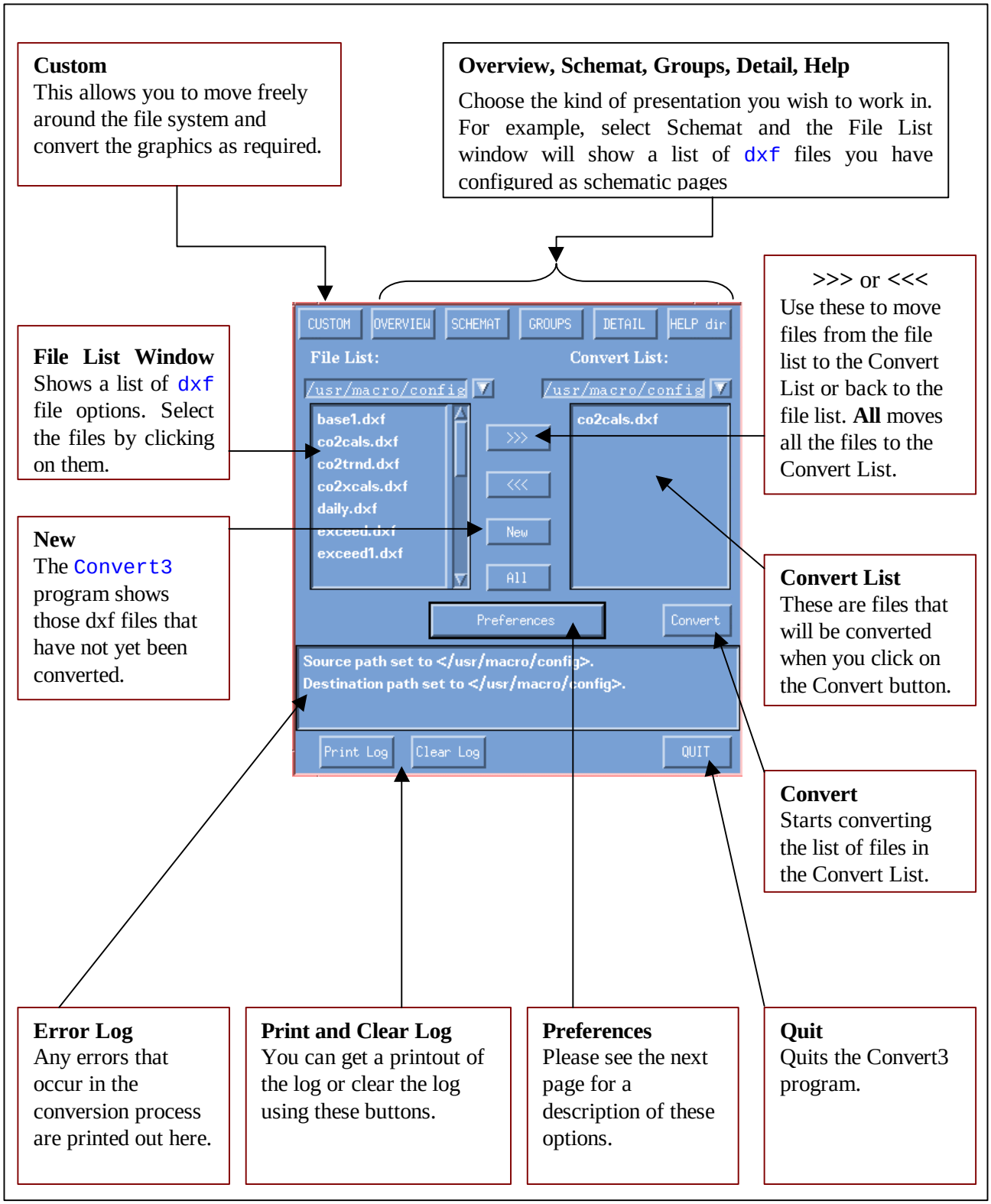
iii. Select the "`Convert`" option. (The next section - "Using the `Convert3` Screen" explains how you do this.)

iv. In a different window, check out your new graphic from the operations program. If necessary modify the cosmetics of the drawing, save it as a `dxf` file and repeat step (ii) through (iv) until you are satisfied.

v. Exit the `convert3` program.

Using the Convert3 Window

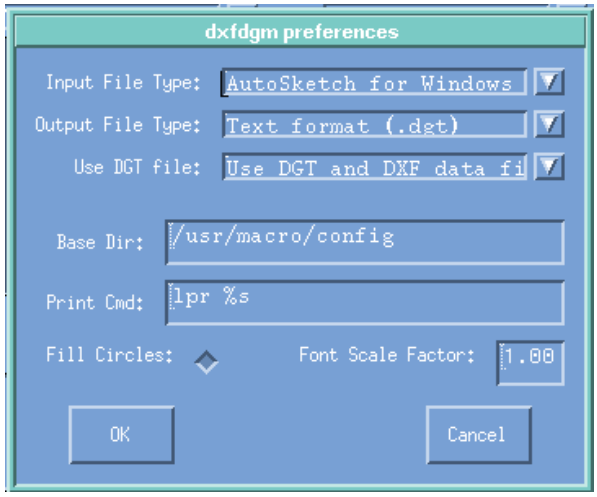
The diagram below shows the main features of the `Convert3` window.



Convert 3 Preferences

The diagram below shows the preferences screen.

Convert 3 Preferences Screen



1. **Input File Type:** Choose the type of CAD package you used to create the graphic. Each CAD package has slight differences that have an effect on the graphic generation.
2. **Output File Type:** Currently, only text format (.dgt) files are created. In future releases, binary format `.dgb` files will be supported.
3. **Use DGT File:** Normally, the Convert3 program combines the spatial information of the CAD package with the dynamic information in the existing `.dgt` file from the forms editor. You can however, specify that the dynamic information in the existing .dgt file is disregarded. (This is typically for older graphics where all the spatial and dynamic data is held in the CAD diagram.
4. **Base Directory:** This is typically the `$MACRODIR` directory. Since the `Convert3` program is "file-aware", you need to specify this directory.
5. **Print Cmd:** This is the command that would be used when you need to print out the `Convert3` log.
6. **Fill Circles:** You can specify whether you want the circles filled in the conversion process by clicking on this button.
7. **Font Scaling:** The font-scaling factor can be used as a fine tuning tool particularly in the conversion of older graphics. To increase the font size, select a value larger than 1. To reduce the font size, select a value smaller than 1.
8. **OK and Cancel:** Select OK if you are happy with the preferences or Cancel if you don't want these preferences

Convert3 Conversion Steps

The main steps to be carried out when using the convert3 program are shown in the table below.

Step	Objective	Procedure	Result
1	Start the `convert3` program.	Click on the `convert3` icon or type convert3 at the UNIX prompt.	The main `convert3` window will appear.
2	Get a list of `dxf` files available for conversion.	Click on any one of the following buttons: Schemat, Overview, Help, Groups, Detail or Custom. If you click on Custom you can change directories and move around the file system.	A list of relevant `dxf` files will appear in the File List Window.
3	Select the files that you want converted.	You can select files as follows: • Individually pressing on the file name and pressing the "**>>**" button. • All the files by clicking the "**ALL**" button. • All the files that have not been converted by pressing the "**NEW**" button.	The file names to be converted will be shown in the Convert List.
4	Convert the selected `dxf` files to metafile format.	Click on the "**CONVERT**" button.	The conversion process will begin. If necessary, a list of error messages will appear in the log window.
5	Quit the convert3 program.	Press the "**QUIT**" button.	The window will disappear.

9.6 Examining the Graphic Using *MacroView* Operations Program

Once you have converted the graphics to `.dgt` (metafile format) the ***MacroView*** operations program, called `xops3` for UNIX, or `winops.exe` for Windows NT, can be run on this display file (`.dgt`). These programs are generally started from an icon, however, they could be started by the appropriate command line instruction. In UNIX systems, for example, there is usually a script in the `$HOME/bin` directory which runs the xops3 program on the Navigator display and effectively starts the Navigator. This script is usually called `ops3`, however, you should check with the ***MacroView*** system administrator if this file is not present. In the majority of cases the displays you are generating will be used within the ***MacroView*** Navigator environment and it is

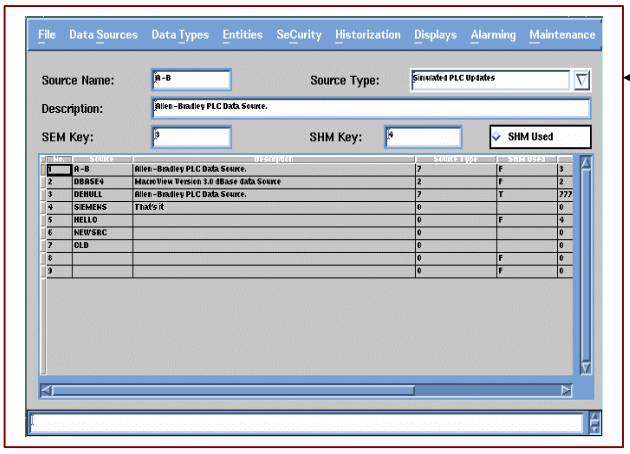
When the graphic comes up, mats that have not been fully configured will appear as shaded rectangles. Metasprites will appear as the grouped top layer of the graphic objects.

Setting the System into Simulate Mode

To test out the graphic, we recommend you set the sources of type 1, 5 and 6 are changed into simulation mode using the configurator. This will show randomly changing values in the dynamic objects.

To change the source, use the configurator.

Putting the System into Simulation Mode



Changing to Simulation Mode:
Change the DCS drivers to Simulated Update and the PLC drivers to Simulated PLC sources.

Objective

Put the driver into simulation mode to exercise the dynamic objects.

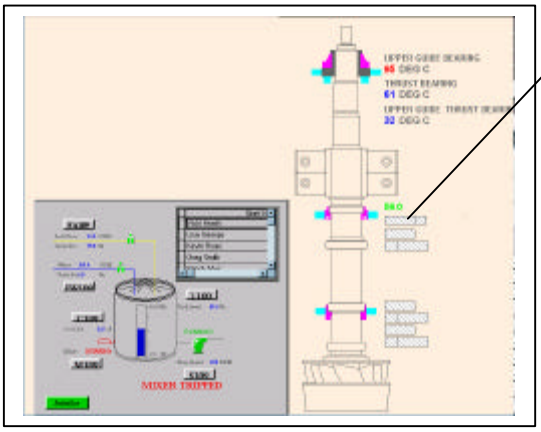
Procedure

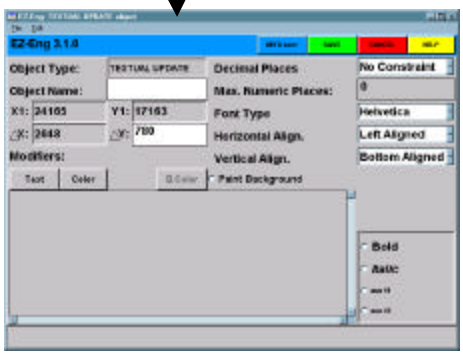
1. Start up the configurator and from the menu, select ***Data Sources:Sources:Detail***.
2. Change each DCS (sorted image) driver to a Simulated Update (type 1) driver
3. Change each PLC (shared memory) driver to a Simulated PLC (type 7) driver.
4. Quit the Configurator and restart ops3

9.7 Using the Graphic Object Editor

The graphic Object Editor, or EZ-Eng utility, can be used to configure and embellish the raw object mats and metasprites, which have been created in the CAD environment. The operations program is started in engineering mode with the `ops3 -edit` command or from an icon, which has been set up to do this. When you are in this mode, the right mouse button is used to click on a mat or metasprite to call up various graphic-editing forms. The diagram below shows how the Graphic Object Editor is invoked.

Graphic Object Editor Overview





Procedure

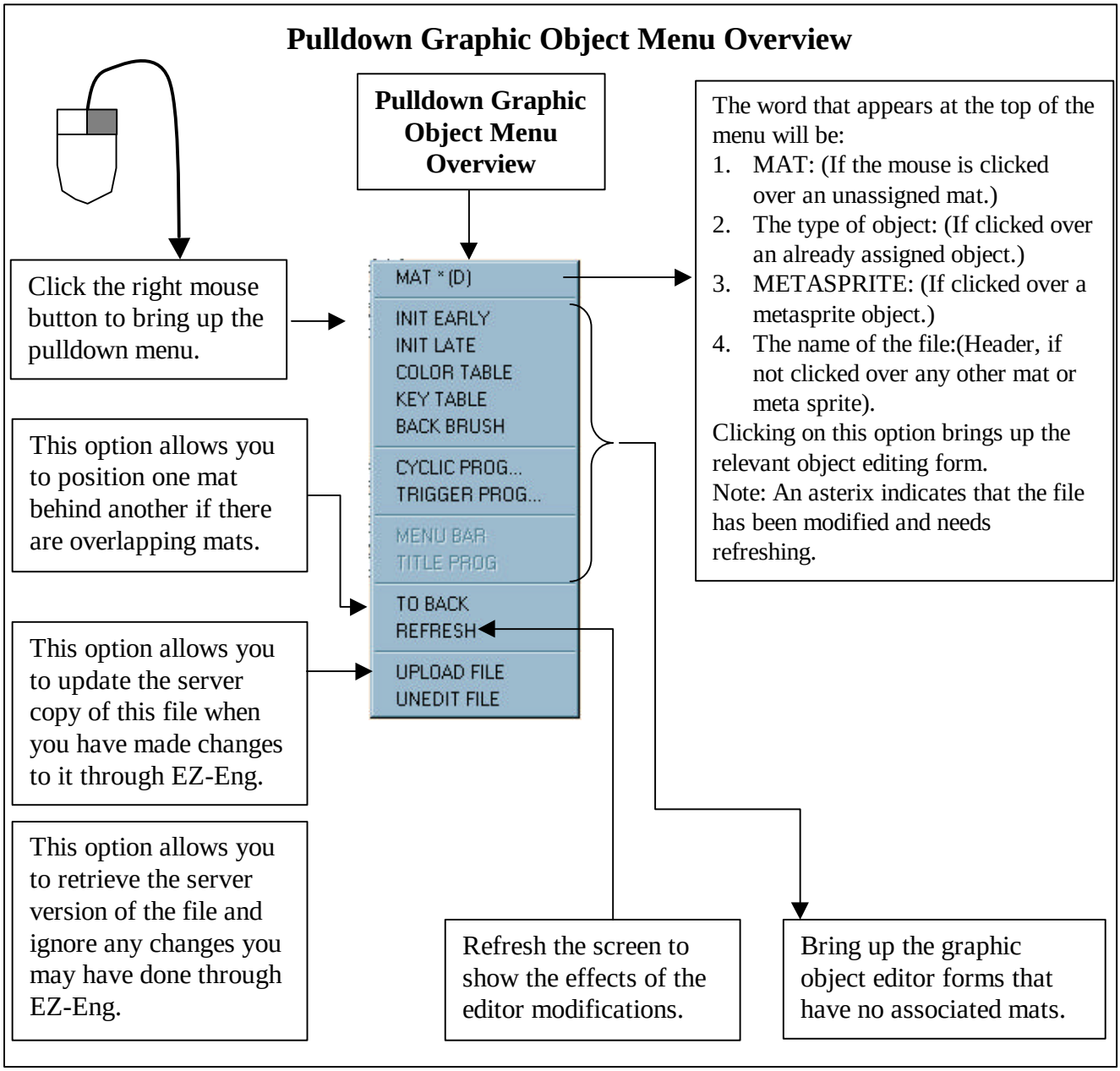
1. Start up **ops3** in engineering mode, using the method described above. Select the graphic you wish to work on.
2. Move the cursor over the mat or sprite you wish to work on. Press the right mouse button. The pulldown menu will appear providing a list of options. The diagram on the next page shows the various options that are available depending on which object you have selected.
3. If the word MAT appears, you have not yet assigned an object to the mat. Click on the word MAT and the MAT editor form will appear.
4. If you have already assigned an object to the mat, the name of the assigned object will appear in the top position in the menu. Click, with the right mouse button, on this and you will go directly to the relevant form.
5. If you have not yet assigned an object type to the mat, just select the object type from the array of keys. The relevant graphic object editor form will appear showing the current or default item settings.
6. You may now proceed to modify the object settings. Once complete, press the Save button. The form will disappear.
7. To edit a non-graphic object, e.g. cyclic, select the item to be edited from the pulldown menu. The relevant object editor will appear for editing.
8. To view the changes, press the right mouse button to bring up the pulldown menu and select refresh. The graphic will redraw.

Notes:

1. To change an already assigned object type to a new object type, you must first turn it back to a mat using the menu in the object editor form.
2. An asterix will appear in the pulldown menu next to the object type if a refresh is required.

9.8 The Pulldown Graphic Object Menu

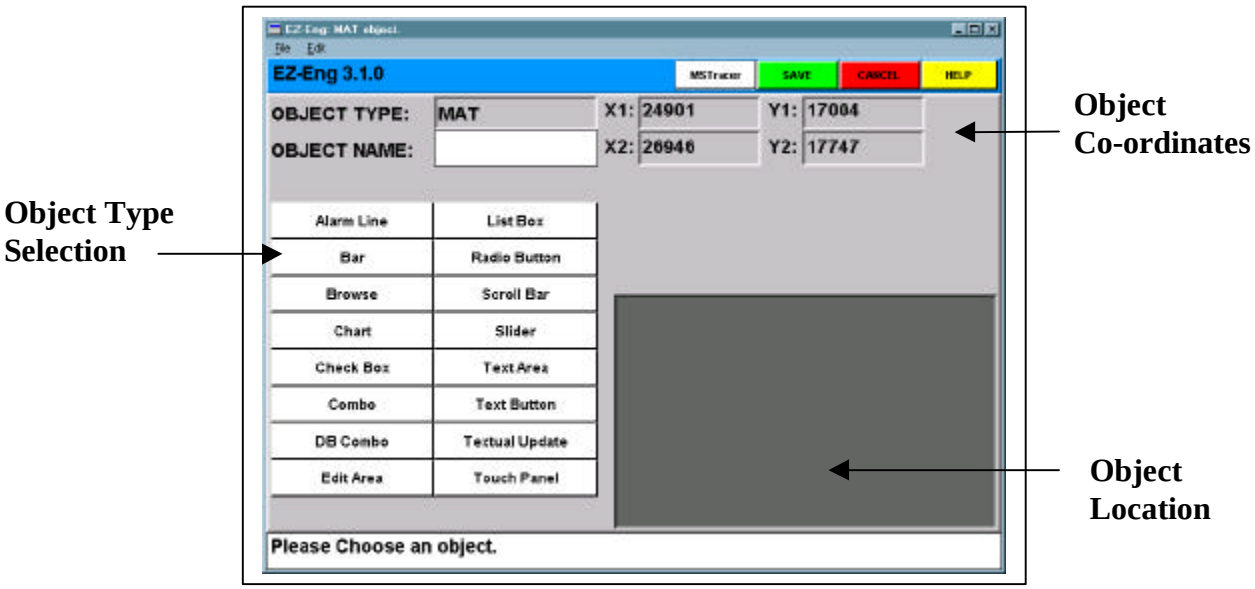
The diagram below shows the details of the pulldown menu and the various functions associated with it.



The MAT Object

The Mat Object is used to represent an area of the screen, which will be developed into a dynamic graphic object to display data in ***MacroView***. The definition of functionality for each mat is selected from a pop up form and the diagram below shows the main features of this mat object editor form.

Mat Object Editor Form



1. **Purpose:** The Mat Object Editor is used to:
   - Identify the selected object area spatially and by name.
   - Assign an object structure to the raw mat.
   - Act as a branching point to the relevant graphic object editor.
2. **Identify the selected mat:** The mat outline appears in the object locator and the object name (if already assigned) appears in the object window.
3. **Assigning a structure to the graphic object:** When one of keys in the object type selector is clicked, the system creates the associated structure in the .dgt file and (if not yet defined) adds the default variables and programs to populate the object.
4. **Object Editor branching point:** The array acts as a branching point to the relevant graphic object editor form.

Procedures:

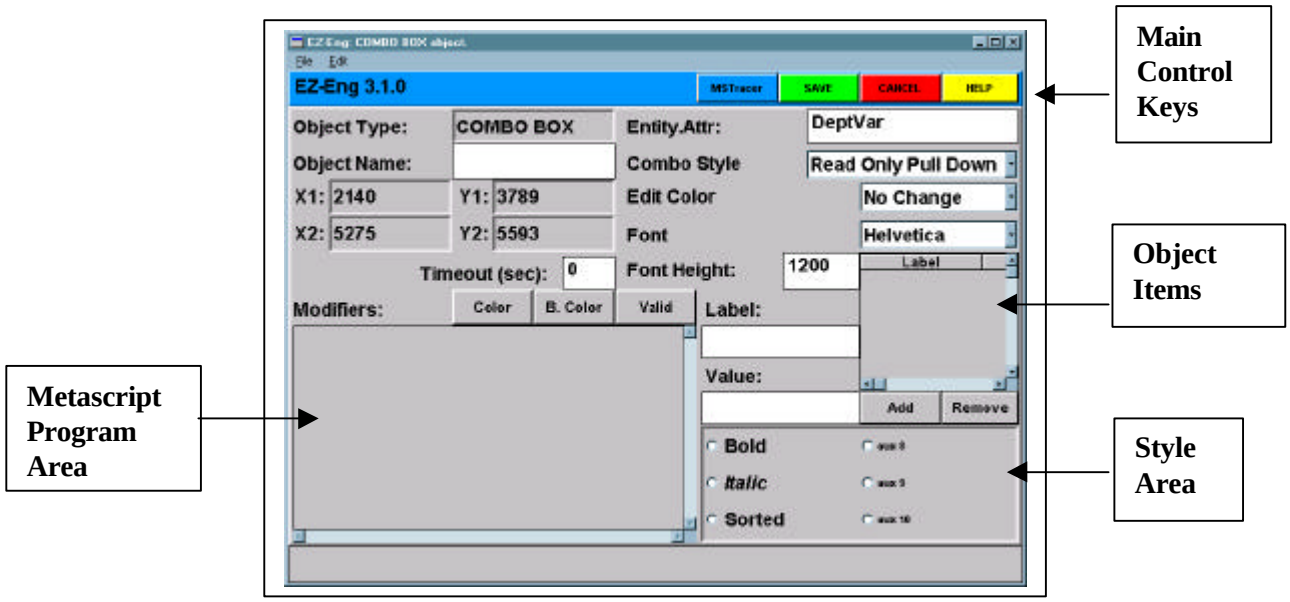
1. Move the cursor over the mat to be edited and right mouse click to bring up the pulldown menu.
2. Select MAT to bring up the Mat object.
3. Confirm you have selected the correct mat using the object locator.
4. Click on the required object type key in the array to bring up the relevant graphic object editor form.

Graphic Object Editor Forms

The diagram below shows a typical Graphic Object Editor Form. The form has various functions that are common to all editor forms. These are discussed in the diagram.

Common Object Editor Functions

(Combo Box Example)



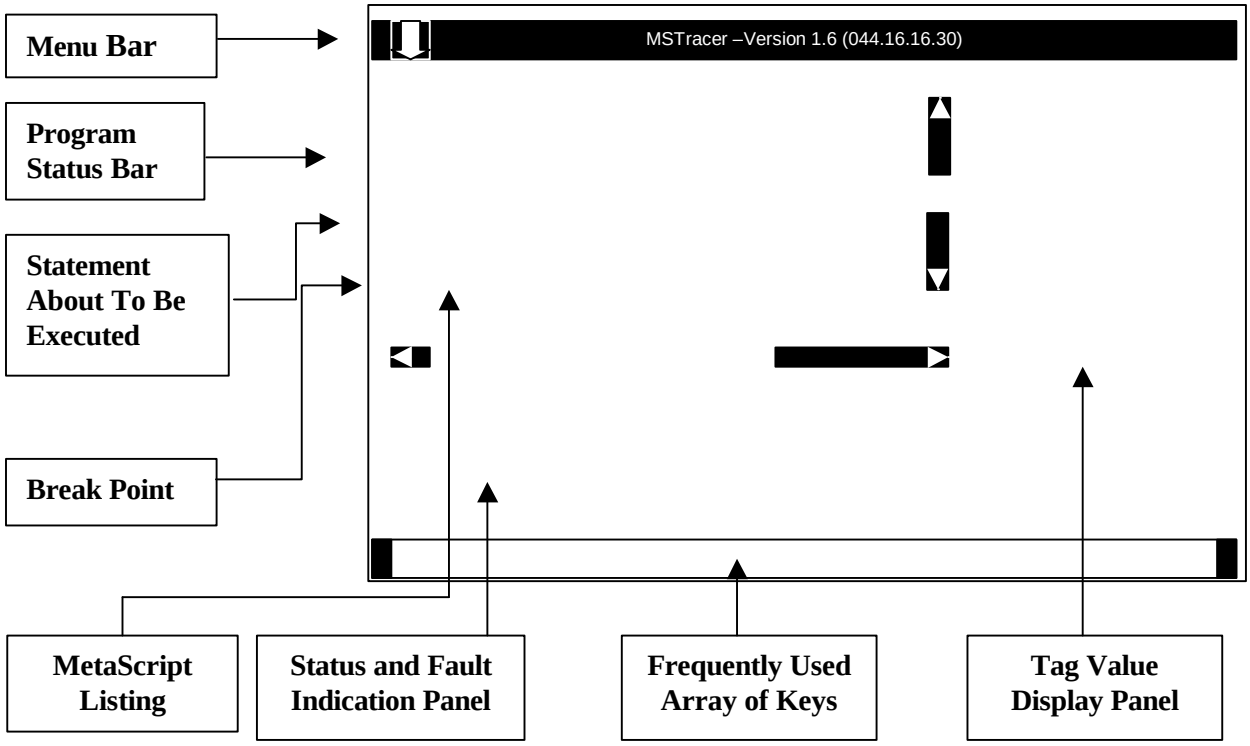
Common functions

1. **General:** Each graphic object editor form has certain common function keys as discussed below. Since these functions are common to all forms, they are not repeated in the object editor reference section.
2. **Help:** This key brings up the on-line help function. The help function will be related to the form being edited. I.e. context-sensitive.
3. **MS Tracer:** The MS Tracer program will be invoked to develop and test the program in the meta script program area so that meta scripts may be effectively traced and debugged.
4. **Save:** The graphic object is stored back into the dgt file.
5. **Close:** The form editor is closed.
6. **Items:** These are the individual elements that make up the graphic objects.
7. **Style Area:** Where text is involved in an object, you may choose the various elements of style such as Bold and Italic.
8. **Meta Script Program Area:** This area shows the program associated with the highlighted program key. The Accept key will save the changes, the Cancel key will bring back the previous meta script.
9. **Single Line Text Edit or Prompt:** This prompt shows a message indicating the current program being modified.

9.9 Meta Script Tracer

Complex or difficult metascript programs may be efficiently tested and debugged using the metascript Tracer tool. The diagram below shows the main components of the metascript tracer.

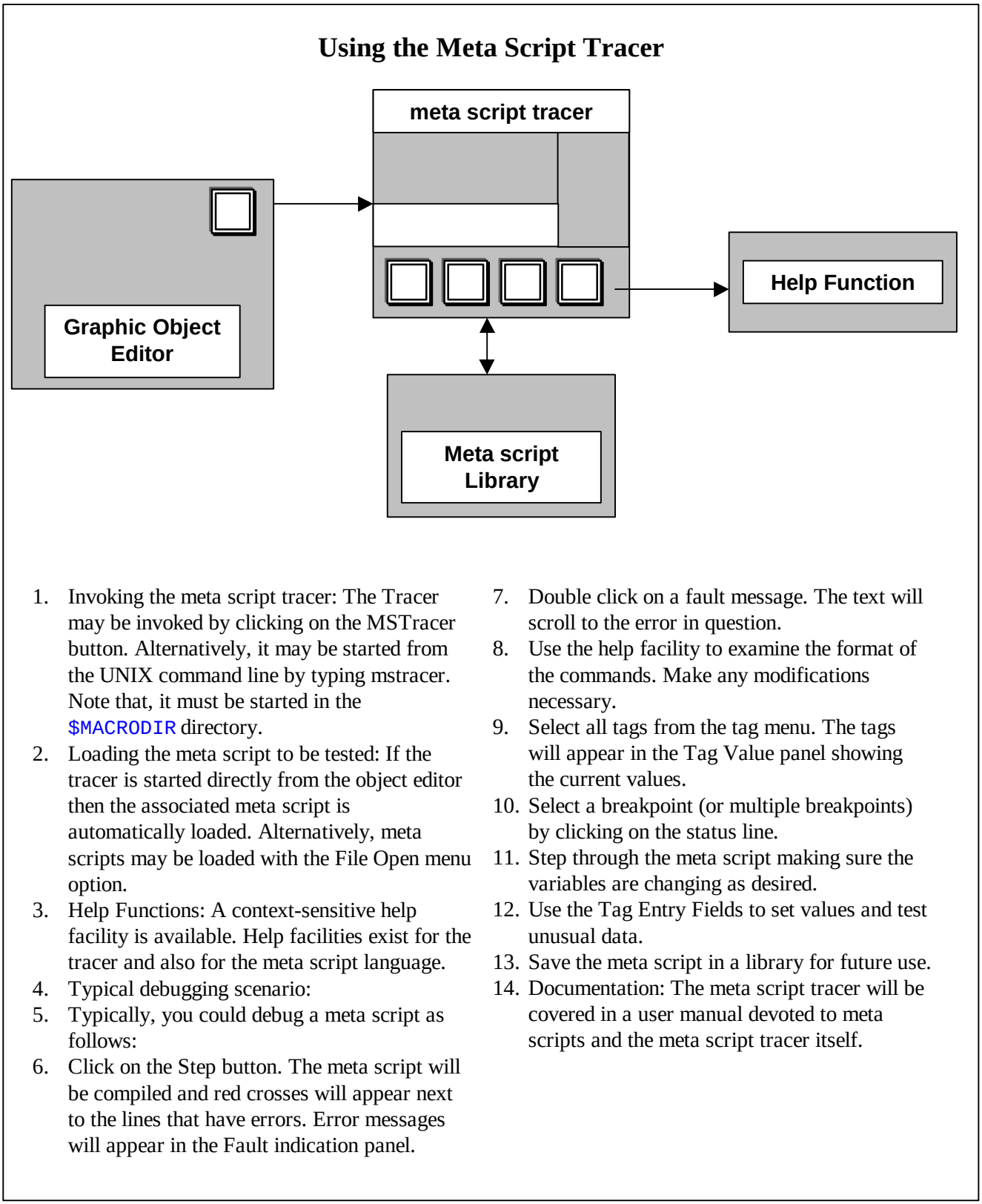
Meta Script Tracer Main Components



1. **Structure:** The meta script Tracer consists of a group of functional panels designed to provide visibility to the key attributes of a working meta script program.
2. **Meta Script Listing:** A listing of the meta script program under test is shown in a scrollable window. The standard cut, copy and paste keys function in this and other areas.
3. **Program Status Bar:** This provides a simple visible indication of the status of the program lines. Errors are shown as an X, breakpoints as a large dot and the statement about to be executed is shown as an arrow.
4. **Status and Fault Indication Panel:** Messages and errors are written to this panel during the tests. System messages are also written to this panel.
5. **Frequently Used Array Of Keys:** These keys include program control functions such as Run, Step, Cont (for continue) and Halt. They also include file and system functions such as Save, Quit and Help. Other functions are available from the menu bar.
6. **Tag Entry Fields:** These fields may be used to specify tag (variable) names and their values.
7. **Tag Value Display Panel:** Selected variables and their current values are shown in the tag value window. These indications are updated whenever the program stops.
8. **Menu Bar:** The menu bar has additional functions that are not covered by the frequently-used keys

Using the Meta Script Tracer

The meta script Tracer may be invoked directly from the Object editor. The diagram below shows how the meta script tracer may be used.

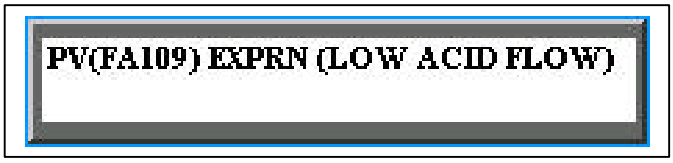


1. Invoking the meta script tracer: The Tracer may be invoked by clicking on the MSTracer button. Alternatively, it may be started from the UNIX command line by typing mstracer. Note that, it must be started in the `$MACRODIR` directory.
2. Loading the meta script to be tested: If the tracer is started directly from the object editor then the associated meta script is automatically loaded. Alternatively, meta scripts may be loaded with the File Open menu option.
3. Help Functions: A context-sensitive help facility is available. Help facilities exist for the tracer and also for the meta script language.
4. Typical debugging scenario:
5. Typically, you could debug a meta script as follows:
6. Click on the Step button. The meta script will be compiled and red crosses will appear next to the lines that have errors. Error messages will appear in the Fault indication panel.
7. Double click on a fault message. The text will scroll to the error in question.
8. Use the help facility to examine the format of the commands. Make any modifications necessary.
9. Select all tags from the tag menu. The tags will appear in the Tag Value panel showing the current values.
10. Select a breakpoint (or multiple breakpoints) by clicking on the status line.
11. Step through the meta script making sure the variables are changing as desired.
12. Use the Tag Entry Fields to set values and test unusual data.
13. Save the meta script in a library for future use.
14. Documentation: The meta script tracer will be covered in a user manual devoted to meta scripts and the meta script tracer itself.

9.10 Alarm Line

The Alarm Line is used to display the most recent message sent to the system alarms or alarms message files.

Alarm Line Main Components



Main Features

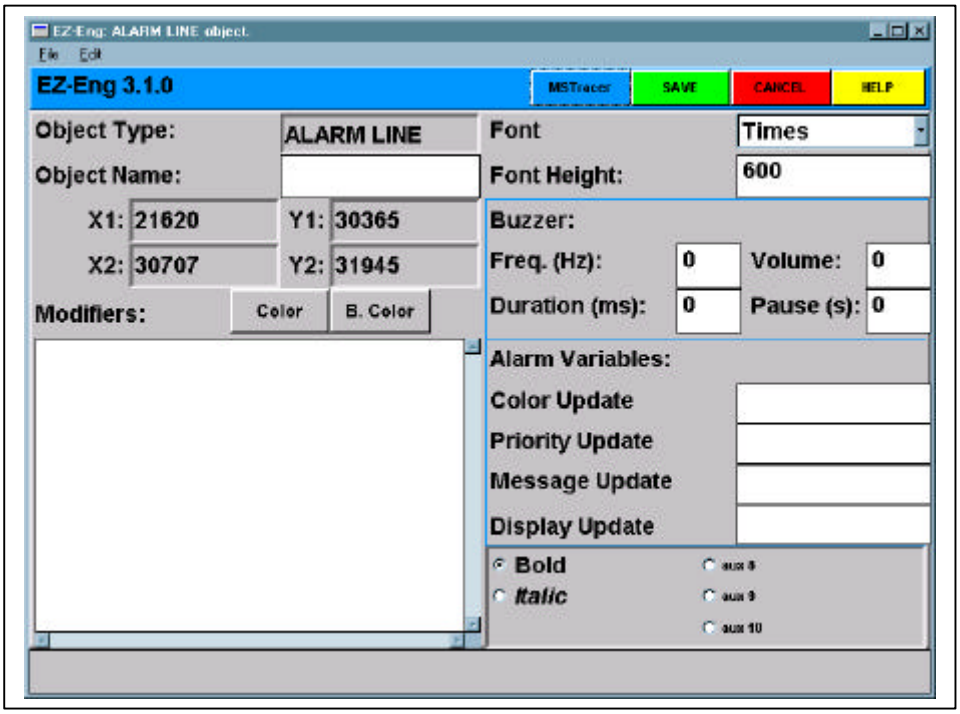
1. **Structure:** The Alarm Line is a single line of text that shows the most recent alarm or message to be processed by the alarm manager.
2. **Purpose:** The Alarm Line provides a visual indication of the last alarm or message to occur. The object also manages the internal beeper within the computer.
3. **Buzzer Characteristics:** You may specify various buzzer characteristics such as the tone of the buzzer, the on-time and off-time of the buzzer etc.
4. **Alarm Variables:** These are variables that hold certain parameters associated with the alarm line. They are updated by the alarm line and include such parameters as the colour, priority, message and display associated with the latest alarm.
5. **Warning:** In its current form, the Alarm Line must be used with caution since it does not prioritize the alarms. It always displays the most recent message irrespective of the priority. It is better to devise an operating strategy around the Navigator Alarms page or Alarm pop-up.

Creation Procedure:

1. Create and Animate the mat in the CAD diagram. Make the mat the size of the Alarm Line.
2. Using the Alarm Line form, enter the items so as to meet the design requirements.

The diagram below provides an overview of the main configurable items of the Alarm Line Object.

Alarm Line Configuration Items



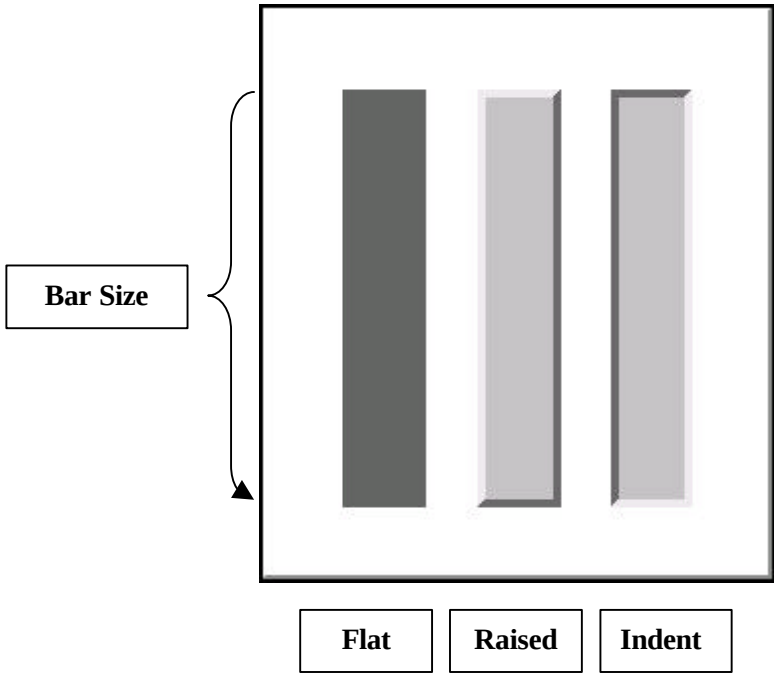
Configuration Form

1. **Object Type:** Alarm Line: The Alarm Line is used to display the most recent alarm or message received by the system.
2. **Object Name:** The name or ID of the Alarm Line object.
3. **Font Height:** This is the size of the message information in VDC units.
4. **Buzzer Frequency:** The frequency of the Tone in Hertz. (If you set this to zero, the buzzer will not sound.)
5. **Buzzer Volume:** The percentage volume of the buzzer.
6. **Alarm Variables:** Configure the names of variables that the alarm line object writes to each time a new message is sent to the alarm line. When the alarm line receives a new alarm or message, it will update.
   - **The Color Update** variable with the color associated with the latest message.
   - **The Priority Update** variable with the priority associated with the latest message.
   - **The Message Update** variable with the latest message.
   - **The Display Update** variable with the (superfind) display associated with the latest message.
7. **Style*:** These are the various items of style that you can use with the alarm line text. E.g. Bold Italic etc. (* Focus, Strikeout, Underline are not supported yet.)
8. **Buzzer Duration:** The on time or mark time of the buzzer in milliseconds.
9. **Buzzer Pause:** The off time or space-time of the buzzer in seconds.

9.11 Bar Object

The bar is used to display process data or variables as a dynamically changing bar, which can indicate values vertically or horizontally.

Bar Main Components



Main Features

1. **Structure:** The bar is a rectangular shape that appears flat, raised or indented and has a size that varies with the results of a meta script.
2. **Purpose:** The bar can be used to provide a visual representation of an analog value.
3. **Bar Size:** The size of the bar varies with the results of a meta script. Typically, this could be a real-time entity.attribute.
4. **Style:** You may choose from flat, indented or raised. The thickness of the bar may also be specified. Note that the style may be dynamic.
5. **Orientation:** The direction that the bar will increase in size - either up, down, left or right.
6. **Colors and Patterns:** You may specify the color (foreground color), pattern style and background color of the bar. You may also choose to have the highlighted and shaded edge colors calculated automatically or simply be white and black.

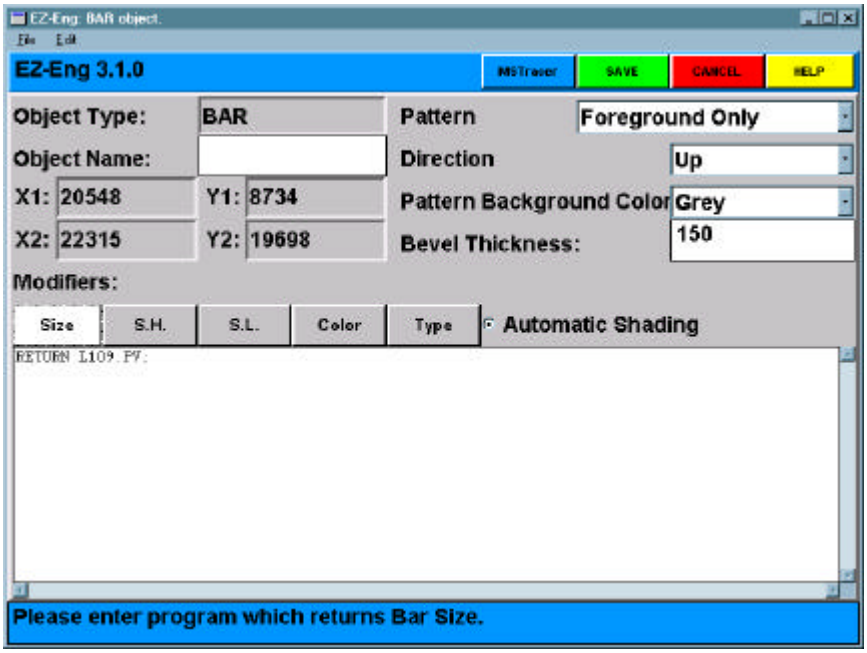
   Note: The Patterns are only valid if the bars are flat.
7. Scale High and Scale Low: Meta scripts specify the maximum and minimum scales of the values.

Creation Procedure:

1. Create and Animate the mat in the CAD diagram. Make the mat the size of the fully extended bar.
2. Using the bar form, enter the items so as to meet the design requirements.

The diagram below provides an overview of the main configurable items of the bar Object.

Bar Configuration Items



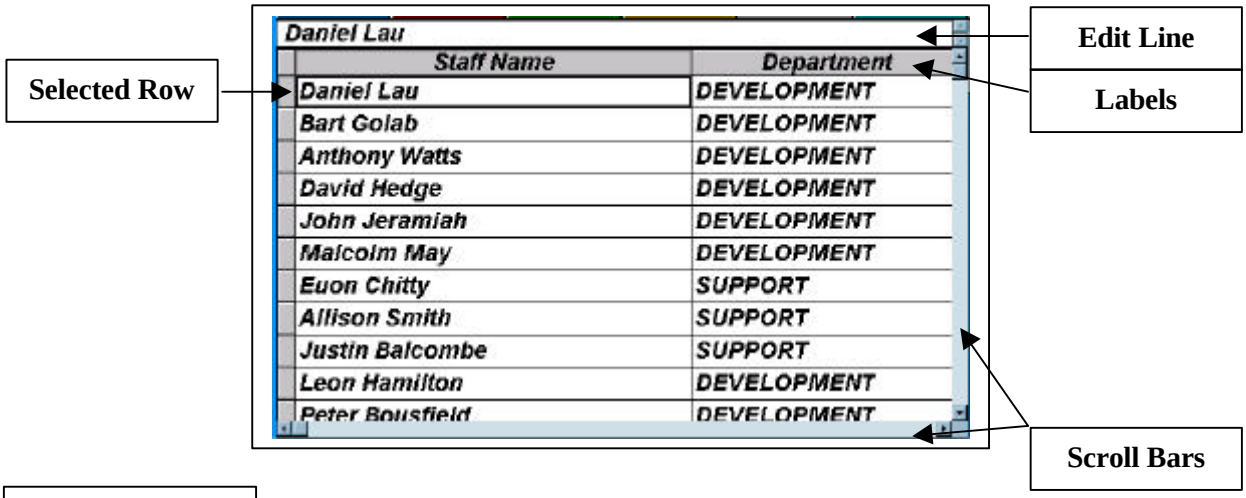
Configuration Forms

1. **Object Type: Bar:** The bar is used to display process data or variables. The size of the bar varies in direct relation to the analog value.
2. **Object Name:** The name of the bar object.
3. Bar Size: This meta script returns a value that will determine the size of the bar. For a vertical bar, the height will vary in relation to this meta script. In a horizontal bar, the width will vary with this meta script between the calculated SH and SL programs.
4. **Orientation:** The direction the bar will increase as the value returned by the bar size meta script increases.
5. **Style:** The style of the bar. This is either flat, indented or raised.
6. **Bevel Thickness:** The thickness of the indented or raised bar.
7. **Automatic Shading:** The color of the highlighted and shaded edges will be automatically calculated. (If automatic shading is not selected, the edges will appear white and black).
8. **Pattern:** The pattern on the bar surface if the bar is flat.
9. **Color:** A meta script that returns the foreground (primary color) of the bar.
10. **Background Color:** If a pattern has been chosen, the background color is the second color in the pattern.
11. **S.H:** The high scale to be used in calculating the size of the bar.
12. **S.L:** The low scale to be used in calculating the size of the bar.

9.12 Browse Box

The Browse Widget provides a tabular, scrollable representation of databases and historical data. The diagram below shows the main features of the Browse widget.

Browse Widget Main Elements



Main Features

1. **Purpose:** The Browse widget is a graphical tabular representation of a database, a view of a database or a historical view.
2. **Scroll bars:** Vertical and horizontal scrollbars allow the user to navigate through the database.
3. **Views:** You may use the CREATE VIEW command to select certain fields, filter and order a database.
4. **Colors and Background Colors:** These may be defined by a meta script for each row of the view.
5. **Labels:** You may define the labels for each field.
6. **Editing:** You may specify an edit line above or below the Browse widget. (Or not at all).
7. **Validate:** You may specify a validate meta script that will ensure only valid data is written to the database.
8. **Cell Text Styles:** The font, size and style may be defined.
9. **Actions:** You may specify an action meta script that is executed when a row is double clicked.
10. **Memo Field Features:** You may specify the default number of lines for a memo field, a default column width and a width factor. Memo fields that do not fit in the column will be indicated by a series of periods at the base of the cell. I.e. . . . . .
11. **Cell Resizing:** You may drag the borders of the cell to resize them.
12. **Message Control:** Under program control, you may send messages to the Browse widget. (Please see the messages summary to be added to this document.)
13. **Record Number Synchronization:** You may link browse widgets with other visible Browse widgets and charts using the message passing mechanisms.
14. **Backing Store:** In general, only the cells that have changed will be updated. You can force all the cells to be updated whenever there is a change by setting BACKING environment variable to 1.

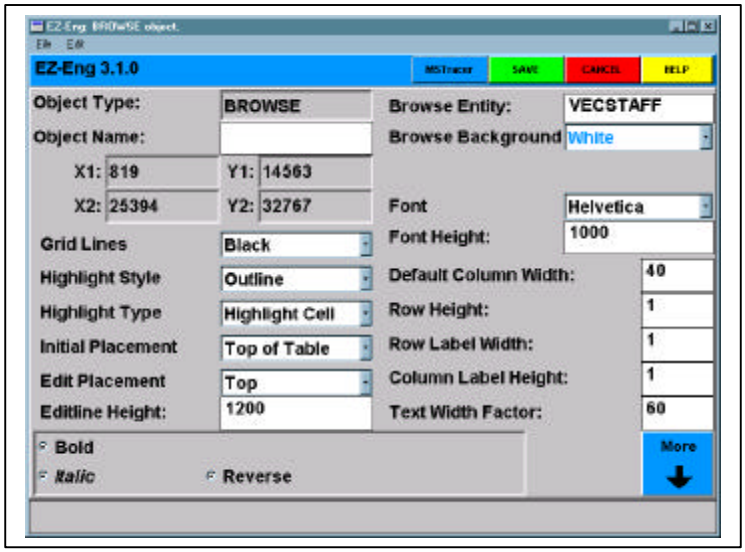
Procedure:

1. Create and animate the mat in the CAD diagram using the normal procedures.
2. Use both pages of the form editor to modify the various items in the widget. **Note:** To be effective, the browse widget should not be too small. A minimum 1/4 page is recommended.

Browse Configuration Forms

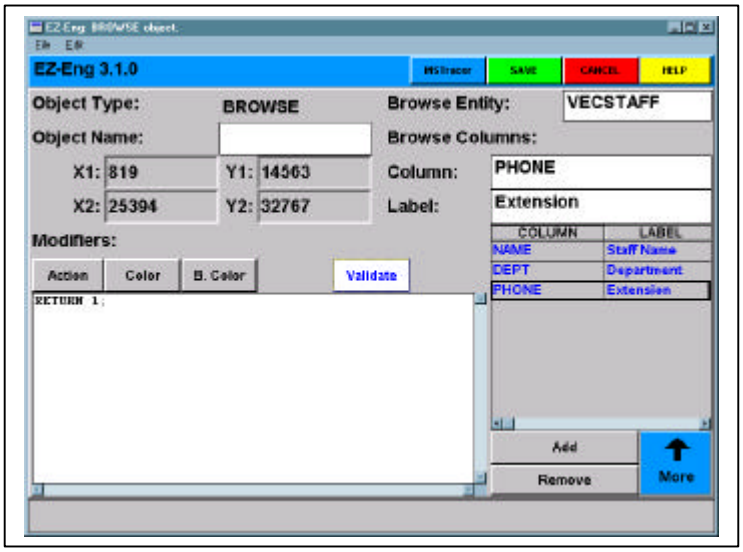
The diagrams below and on the next page provide an overview of the main configurable items of the Browse Widget.

Browse (Page 1) Configuration Items



1. **Object Type:** Browse: The type of object being edited
2. **Object Name:** The name or ID of the object.
3. **Browse Entity:** The entity name or the VIEW of the database.
4. **Background Style:** Defines the style of the browse background whether grids, panels* or no separation for each cell.
5. **Highlight Style:** Defines how the cells will be highlighted when selected.
6. **Initial Placement:** Defines which record will be selected when the browse widget is first brought up.
7. **Edit Placement:** The position (relative to the Browse widget) of the Edit Line.
8. **EditLine Height:** The height in VDC units of the optional edit line. Note: see the section on VDC units. The full screen is 32767 units high.
   Note: If no edit line is required, this should be 0.
9. **Label Background (color):** The background color of the label text situated at the top of each column.
10. **Default Column Width:** The width of the columns in number of characters for those fields that do not have any inherent character width E.g. memo fields.
11. **Row Height:** The height of the rows in a number of lines for those fields that have typically more than a single line of information. (E.g. memo fields)
12. **Text Width Factor:** The actual width of the column is calculated as the width of the largest character times the number of characters times this width factor. (Used in Memo fields.
13. **Font Type:** The font to be used for the contents of the cells, edit line and the labels.
14. **Font Height:** The font height in VDC units. **Note:** see the description of VDC units.
15. **Style Box:** The various style attributes that are applied to the text in the browse cells, the labels and the edit-line.
16. **Focus*:** The browse object asks for the focus when the window is first brought up. (Not Yet Supported)
17. **More:** Goes to the next screen.

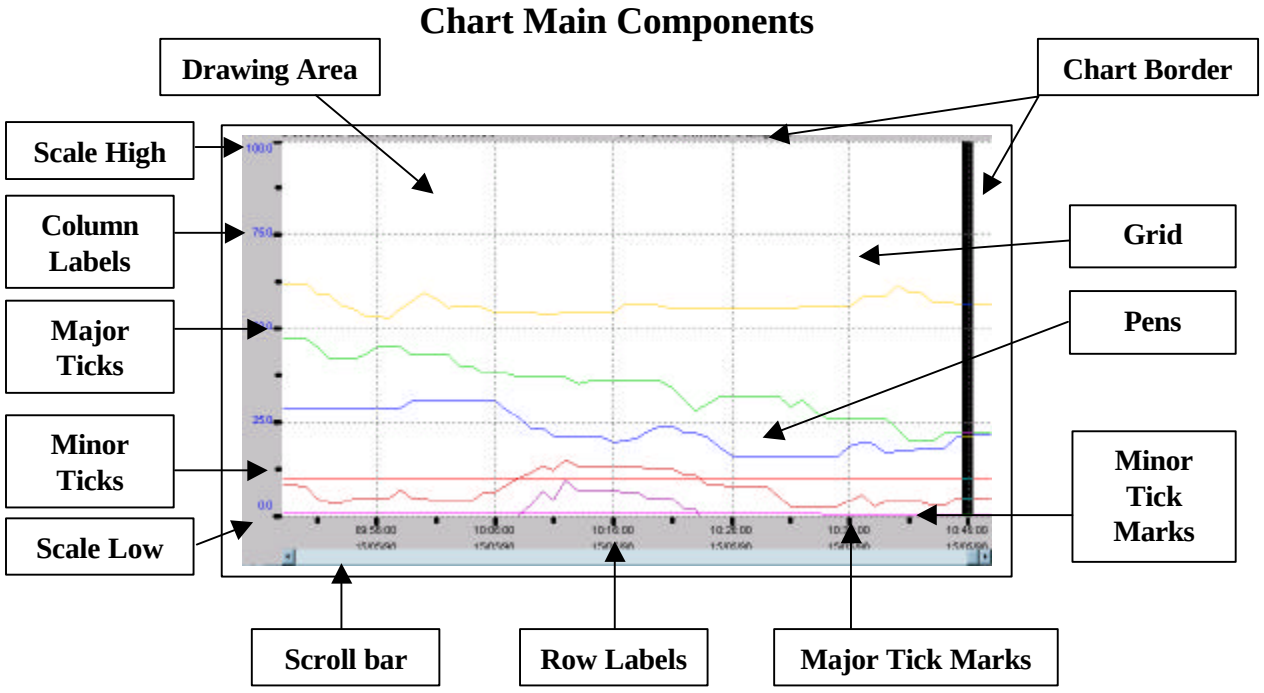
Browse (Page 2) Configuration Items



1. **Color:** The Color meta script determines the color of the text in the rows. The color is determined independently for each row of information that is displayed. The meta script is calculated and the returned color name is used for the text color.
2. **B. Color:** The Background Color meta script determines the color of the text background in the rows. The background color is determined independently for each row of information that is displayed. The meta script is calculated and the returned color name is used for the text color.
3. **Action:** The action meta script is executed whenever a row in the browse widget is double clicked.
   Note: The cell that has been clicked can be determined by the arguments provided in the action program.
4. **Column:** The name of a field or column in a database entity or view that you wish to have displayed in the browse widget.
5. **Label:** This is the label that will appear at the top of the column in the browse widget.
   For each column required in the browse, one Label Name, one column (field) Name and possibly one validate program will be required.
   If no columns are added, the entire view will be displayed using the field names as labels.
6. **Validate:** The VALIDATE meta script must return a 1 if it determines that the input is valid. It returns a 0 if it calculates the input as invalid. The edited value is held in a variable called SetRequest until the validate program has been completed. It is only required if an edit line is selected.
7. **Add:** Use the Add button to add additional columns to the browse widget.
8. **Remove:** Remove a column that has been selected.
9. **More:** Returns to the previous screen.

9.13 Chart Object

The chart object provides a multi-pen scrollable graphical view of a set of columns in a database, view or a historic view. The chart widget offers a rich set of facilities likely to satisfy most requirements in the industrial environment.



1. **Types:** Line charts, area charts and bar charts are supported.
2. **Scroll bars:** Horizontal, vertical and external scroll bars are supported.
3. **Axis Information:** You may define the axis labels, major and minor tick marks, fonts, colors and precision of the row and column labels.
4. **Borders:** The size, pattern and color of the chart borders may be specified.
5. **Canvas:** The color and pattern of the chart canvas as well as the canvas borders may be specified. Horizontal and Vertical grids may also be used.
6. **Pens:** The number of pens is only limited by visual and machine constraints. You may write meta scripts that define the color of the pen or area, the type of marker, the color of marker and the scale high and scale low values. You may also choose different connecting line types and methods of connection.
7. **Messages:** The chart accepts and responds to various messages issued by other widgets such as buttons. The table in this section defines the various messages.
8. **Operations:** The chart may be scrolled using the X and Y axis scroll bars or an external scroll bar. Alternatively, messages may be used to locate the chart in a particular position. The mouse may be used to set the record number.
9. **Scaling:** The axes may be dragged to provide different scaling. The mouse may be used to outline a zoom area (rubber banding) and messages may be sent to the chart to affect a new scale.

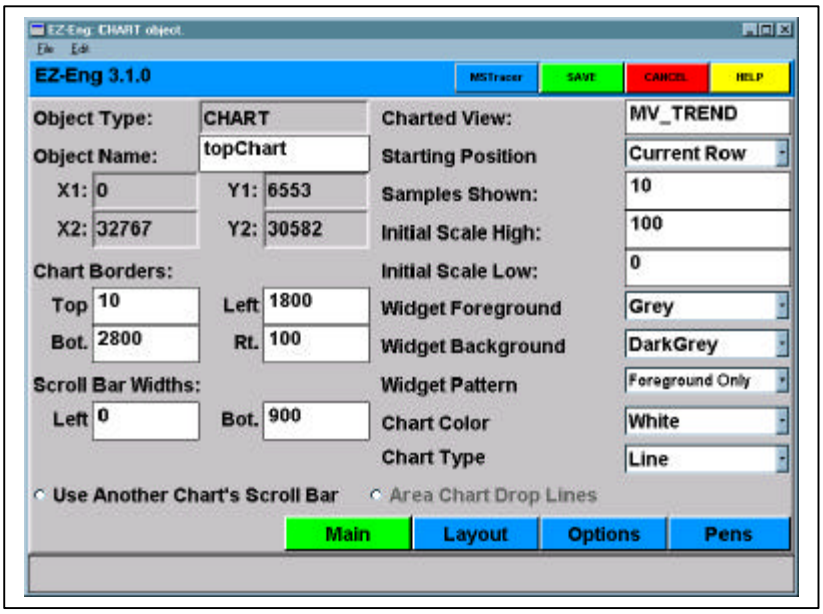
Configuration:

1. Draw a mat in the CAD diagram.
2. Use the chart object editor forms to tailor the chart to your requirements.

   Note: There are 4 object editor forms - Main, Layout, Options, and Pens. Each Pen must be separately configured.

The diagrams below show an overview of the configurable items of the Chart Object.

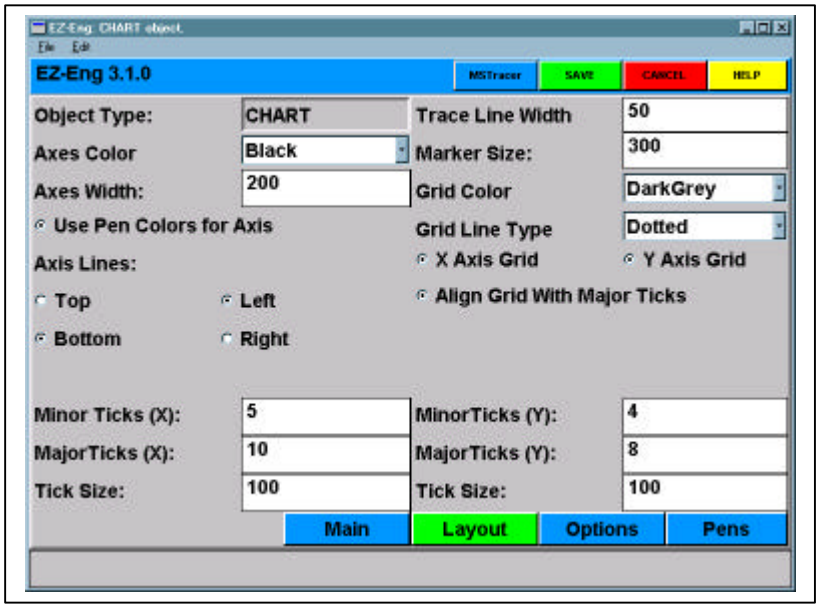
Chart Configuration Items (Main Page)



Configuration Form

1. **Object Type: Chart:** The chart object provides a multi-pen scrollable graphical view of a set of columns in a database, view or a historic view. The chart widget offers a rich set of facilities likely to satisfy most requirements in industrial applications.
2. **Object Name:** The name or ID of the chart object. Other objects such as Buttons use this name to send messages such as Change Scale, Re-display etc.
3. **Chart Border:** The width of the borders in VDC units. This is the distance from the edge of the pen drawing area to the edge of the chart. The scroll bars and scales go inside this area.
4. **Scroll Bar Width:** The width of the Horizontal and Vertical Scroll Bars in VDC units.
5. **Use Another Chart's Scroll Bar:** If this is set, the chart will attempt to find an external scroll bar on the same graphic that is referencing the same view. Use this to co-ordinate multiple trends.
6. **Chart Type:** The type of chart whether Line (connected data points), Bar (flat, raised or indented) or Area (filled area below the data to the X axis or to the next area.)
7. **Area Chart Drop Lines:** Activates lines that drop down from the vertices of the area chart to the X axis to differentiate the samples.
8. **Charted View:** The name of the view being charted. This must be a dBase entity, Table Variable or Historic View created prior to the
9. **Starting Position:** When the chart is first brought up, the chart will display this position. Options include, Bottom and Top of View or the current record number.
10. **Samples Shown:** The desired number of samples or columns to be shown when the chart is first called up. Messages and Rubber banding can alter the number of samples shown.
11. **Initial Scale High and Scale Low:** Scale high and Low (in % between 0 and 100%). The individual pen values are first scaled to their own pen scales and then scaled to the axis scale high and low values. The user may modify this scale once the chart is brought up.
12. **Widget Foreground:** The foreground(primary) color of the entire background of the widget.
13. **Widget background:** The background (secondary) color of the entire background of the widget.
14. **Widget Pattern:** The brush style of the entire background of the widget.
15. **Chart Foreground:** The foreground (primary) color of the chart drawing area (canvas).
16. **Chart Background:** The background (secondary) color of the chart drawing area (canvas).
17. **Chart Pattern:** The pattern style used for the chart drawing area (canvas)
18. **Chart Type:** The way data is to be displayed e.g. Line chart, Area chart and Bar chart etc.

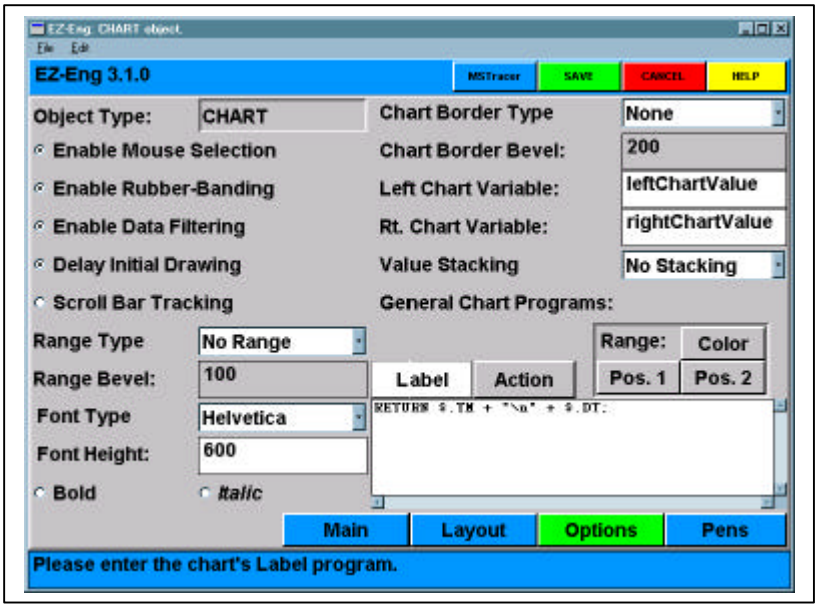
Chart Configuration Items (Layout Page)



Configuration Form

1. **Axes Color:** The color of the X and Y axes. (See note on using Pen Colors for the Y axis below.)
2. **Axes Width:** The width of the X and Y axes in VDC units.
3. **Use Pen Colors for Axis:** The Y axis will label will be drawn in the same color as the selected Pen. This color will override the Axes Color as defined above as soon as a pen is selected.
4. **Axis Lines:** Decide whether you want Top, Left, Bottom or Right Axis Lines. The axis lines are subtle lines drawn along each axis, just outside the chart drawing area.
5. **Minor Ticks(X):** The number of samples per minor tick marks displayed across the X range. The number of graduations will be one more than this.
6. **Major Ticks (X):** The number of samples per major tick marks displayed across the X range. Scale labels will also be displayed at these locations.
7. **Tick Size:** The size of the minor Tick marks in VDC units. Major tick marks are double this size.
8. **Trace Line Width:** The width in VDC units of the pens used for LINE type trends.
9. **Marker Size:** The size in VDC units of the markers used to show data positions in the chart.
10. **Minor Ticks(Y):** The number of minor tick marks displayed up the Y range. The number of graduations will be one more than this.
11. **Major Ticks (Y):** The number of major tick marks displayed across the Y range. Scale labels will also be displayed at these locations.
12. **X Axis Grid:** An X axis (horizontal) grid is drawn.
13. **Y Axis Grid:** A Y axis (vertical) grid is drawn.
14. **Align with Major Ticks:** If set, the X axis and Y axis grids are aligned with the major ticks. If not, they are aligned with the minor ticks.
15. **Grid Color:** the color of the grid lines.
16. **Grid Line Type:** The grid line type - whether dotted, solid, dashed etc

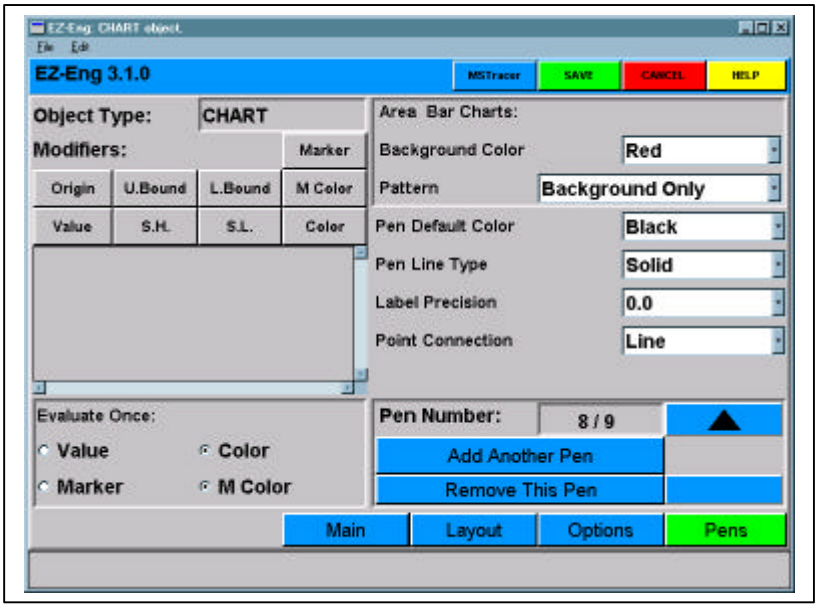
Chart Configuration Items (Options Page)



Configuration Form

1. **Enable Mouse Selection:** If this is selected, you may use the mouse to select a record.
2. **Enable Rubber Banding:** If this is selected, the user may zoom into an area of the chart by "drawing" an outline on the chart with the mouse.
3. **Enable Data Filtering:** MacroView currently uses the codes -666, -777, -888 and less than -999 to show quality problems with historical data. If you set this flag, data points with values close to these values will not be displayed.
4. **Delay Initial Drawing:** This allows the chart to be configured by sending it several messages, the last of which tells it to draw. This is necessary, because some configuration can only be done by way of messages, and this prevents unnecessary redraws.
5. **Range Type:** This allows you to define whether you want a certain range of the chart to be highlighted and the type of highlighting. E.g. It can be used to highlight a good range of values within the chart.
6. **Range Bevel:** The thickness of the bevel of the range area.
7. **Value Stacking:** The kind of stacking of the values that is to be performed.
8. **Font Type:** The font used for all axis labels.
9. **Font Height:** The desired font height in VDC units. Note that the display manager will choose a font closest to this height so variations will occur.
10. **Font Style Attributes:** Decide whether to use Bold or Italic fonts for the labels.
11. **Chart Border Type:** The chart drawing area may be a line, have no line and may be indented or raised.
12. **Chart Border Bevel:** If the chart drawing area is raised or indented, this is the thickness of the bevel in VDC units.
13. **Left and Right Chart Variables**: These variables are constantly updated with the record numbers of the left most and right-most record numbers currently being displayed.
14. **Label Program:** The program that will determine the field (column) that will return the labels to be displayed on the X-axis. For historical views, you can use RETURN $.DT + "\n" + $.TM; (“\n” is a carriage return)
15. **Action:** Double clicking on the chart will activate this action program.
16. **Range Color, Pos1 and Pos2:** These are the meta scripts that determine the Color of the Range band, and the upper and lower bounds of the Range band.

Chart Configuration Items (Options Page)



Configuration forms

1. **Overview:** This configuration page covers the individual features of each pen.
2. **Value:** This is a meta script that determines the value to be displayed. This is evaluated for every data value for that pen that is being displayed.
3. **S.H. and S.L:** The meta script that returns the scale high and scale low values of the pen.
4. **Origin:** A meta script that returns the origin of the pen.
5. **Upper and Lower Bounds:** Meta scripts that return the upper and lower percentage of the chart that the trace is to be scaled to. E.g., if you wanted the trace to only occupy the area between 25 and 50% then the meta script should return these values.
6. **Color:** A meta script that returns the color of the point being charted.
7. **Marker:** A meta script that returns the marker number. Marker numbers are as follows:
   0 = No Marker. 1 = Dot. 2 = Diamond.
   3 = Square. 4 = Plus. 5 = Asterisk.
   6 = Circle. 7 = Cross. 8 = Triangle.
8. **Marker Color:** A meta script that returns the color of the marker at the data point for that pen.
9. **Evaluate Once Only buttons:** If these options are selected (depressed) the Value, Color, Marker Type and Marker Color are evaluated only once before the pen is displayed. If they are not selected, they are evaluated for every data value in the chart. (This provides much more flexibility but is slower to draw.)
10. **Area and Bar Charts:** Specify the background color and pattern. This is the secondary color and pattern type for the Area and Bar charts.
    Note: Beveled Bar Charts don't use this option.
11. **Pen Default Color:** Used to draw the color of the axes if the pen is selected.
12. **Pen Line Type:** The line type used to draw the line charts - E.g. dotted, solid, dashed etc.
13. **Label Precision:** The number of decimal places in the labels.
14. **Point Connection:** The method of connecting the data points - E.g. No connection (data points only), Standard line connection, step-wise connection. (Step-wise is good for digital points.)
15. **Pen Number:** The pen number being viewed and edited. Use the arrow keys to edit the configuration of another pen.
16. **Add Another Pen, Remove this Pen:** Use these keys to add or remove pens to the chart.
    Note: The first pen may not be removed, only edited.

Chart Messages

The table below summarises the various messages that can be sent to the chart object.

The operator interaction section described the operator interface functionality that is available within a chart window. Additional operator interface functionality can be added by utilising other dynamic graphic objects and the `SEND` meta script command. The `SEND` command allows a meta script to send a string message to an object such as a chart object. I.e. `SEND "message" TO "ObjectName";` the chart object performs specific actions if it receives specific forms of messages. (Note that the messages required for the initial setup are usually sent from the INITLATE program. The messages that the chart object understands are listed in the table below:

Table 5: CHART Message Processing Table

Message format	Action Performed
DisplayFirst	Change the chart display so that the first row in the entity is displayed on the chart. This does not change the current row of the entity.
DisplayLast	Change the chart display so that the last row in the entity is displayed on the chart. This does not change the current row of the entity.
ResetYScale	Resets the Y axis labels so that a full scale is displayed. This is useful for returning an operator to full scale once a detailed zoom has been performed.
SetNumDisplayRows**(nRows)**	Sets the number of data rows to be displayed at one time in the chart drawing area. The chart drawing area is automatically redisplayed. Note: that the number of rows actually displayed will depend on both the value specified as an argument and the pixel width available for display.
Display	Redisplays the chart. This message is handy when other objects change information that the chart object does not continuously check for. An example is the high and low scale programs associated with each value specified for display in the chart. These scale programs are only executed when the entire chart is redisplayed. Hence, if the display scale limits are changed as a result of operator action (outside of the chart object), a "Display" message can be sent to the chart to redisplay the drawing area which will also re-run the scale programs.
Display(Line) Display(Area) Display(Bar) Display(RaisedBar) Display(IndentedBar)	All of these messages will result in a redisplay of the chart with the chart type being specified as an argument in the message. The typical usage of this message is to allow the operator to change the chart type at run time.
SetStackingOff	Turns value stacking off at run time. This message will result in a chart redisplay if stacking was not already off.
SetStackingOn	Turns value stacking on. This message will result in a chart redisplay if stacking was not already on.
SetStacking100%	Turns 100% value stacking on. This message will result in a chart

Message format	Action Performed
	redisplay if stacking was not already set to 100% stacking.
SetDisplayOff(**column**)	Turns the display of a specified chart column value off. By default all column values specified for the chart are displayed. The display of a particular column can be disabled by sending a SetDisplayOff message with the column number being deactivated as an argument. Column numbering starts from 1. **Note:** Turning the display of a column off will not result in a redisplay of the chart area. A subsequent "Display" message is needed to do this. This allows multiple SetDisplayOff messages to be sent without repeated chart drawing.
SetDisplayOn(**column**)	Turns the display of a specified chart column value on. If the column value is already being displayed then the message is ignored. Otherwise, this message works much like the SetDisplayOff message.
DisplayOnNow(**column**)	Turns the display of a specified chart column value on and redisplays the particular chart value information immediately. The redisplay action differs depending upon the current chart type. For bar and area charts, the entire chart area is redrawn due to the nature of the graphic display information. For line charts, the redisplay simply involves the redrawing of trace associated with the column specified as an argument.
ToggleDisplay(**column**)	Toggles the display of the specified chart column value between on and off. If the column value display is turned off then the chart area is redrawn. If the column value display is being turned on then the redisplay procedure works in exactly the same way as the DisplayOnNow message.
UseScalesFrom(**column**)	By default the Y axis labelling is based on the scales for the first column value being displayed in the chart. This message specifies to the chart object that Y-axis labelling is now to be based upon the scale information from the specified column number. **Note:** This message does not result in a redisplay of the Y scale labels. Only internal flags are affected.
SetScaleColor(**colorNumber**)	Sets the color for the Y-axis labels. The color has to be specified as a numeric argument. **Note:** This message does not result in a redisplay of the Y scale labels. Only internal flags are affected.
DisplayScales	Redisplays the Y-axis label and graduations. This message is typically used in conjunction with the UseScalesFrom and SetScaleColor messages

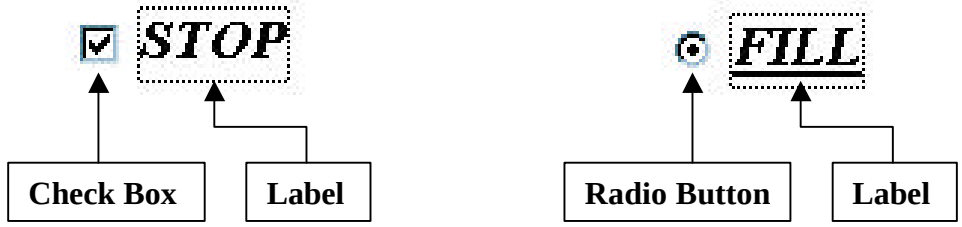
Table 6: CHART Message Processing Table

Message format	Action Performed
DisplayCurrentRow DisplayCurrentRow(Left) DisplayCurrentRow(Right) DisplayCurrentRow(Middle)	When a DisplayCurrentRow message is sent to the chart object, the object will ensure that the current row is being displayed on the chart in the position on the screen specified as an argument. If no argument is supplied then a default of the middle of the chart drawing area is assumed. Note that if the current row is already being displayed in the specified screen position, no action is taken.
ConnectLines(column) ConnectLines(None, column) ConnectLines(Stepwise, column)	This message specifies to the chart object how the lines are to be connected when drawing a line chart for a particular chart column number. The ConnectLines(column) message indicates that straight lines are to be used to join chart points. The ConnectLines (None, column) message indicates that no lines are to be used. In this case the visual display relies on the display of marker graphics instead of lines. The ConnectLines (Stepwise, column) message indicates that the lines are to be drawn in a step fashion where the change from one value to another is represented by an immediate level change halfway between sample point, similar to a logic diagram. Note that this message does not result in display changes. A subsequent "Display" message is required to display the changes.

9.14 Check Box

Check Boxes and Radio Buttons provide an indication of the digital status of a variable. They also provide a means of toggling the variable. The only difference between a Check Box and a radio button is their shape. Check Boxes are rectangles and radio buttons are diamond shaped. Radio buttons are traditionally in a group of buttons where only one button is active at any one time.

Check Box Main Elements



Main Features

1. **Structure:** The Check Boxes and Radio Buttons consist of an indicator (a rectangle in the case of a checkbox and a diamond in the case of the radio button) and a label. The state of the button is taken from the CHECK BOX meta script. When the button is checked (depressed) by a mouse click, the CHECK meta script is executed. When the button is unchecked (raised) by a mouse click, then the UNCHECK meta script is executed.
2. **Purpose:** The Check Box or Radio Button can be used for two main purposes: (i) To display the digital status and (ii) To run a program that could change the variable.
3. **Label:** The text string that is displayed next to the indicator.
4. **Label Features:** You may specify the color, font and size of the text. You may also specify attributes such as BOLD and ITALIC.
5. **Label Color and Background Color:** You may specify these with meta scripts.
6. **Update:** The time following a user change before updating from the field starts again.

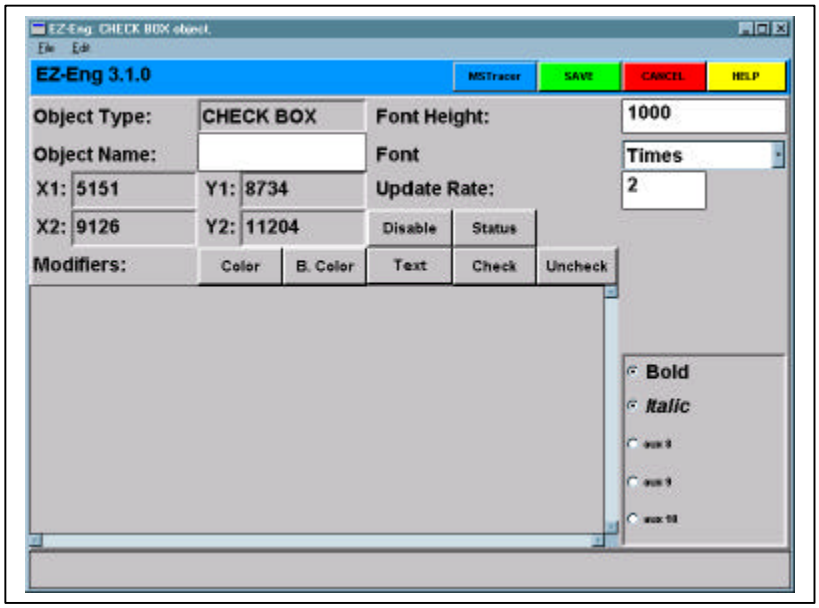
Creation Procedure:

1. Create and animate the mat in the CAD diagram.
2. Use the CHECK BOX or RADIO BUTTON form to satisfy the design requirements.

Check Box and Radio Button Configuration Form

The diagram below provides an overview of the main configurable items of the Check Box or Radio Button Object.

Check Box and Radio Button Configuration Items



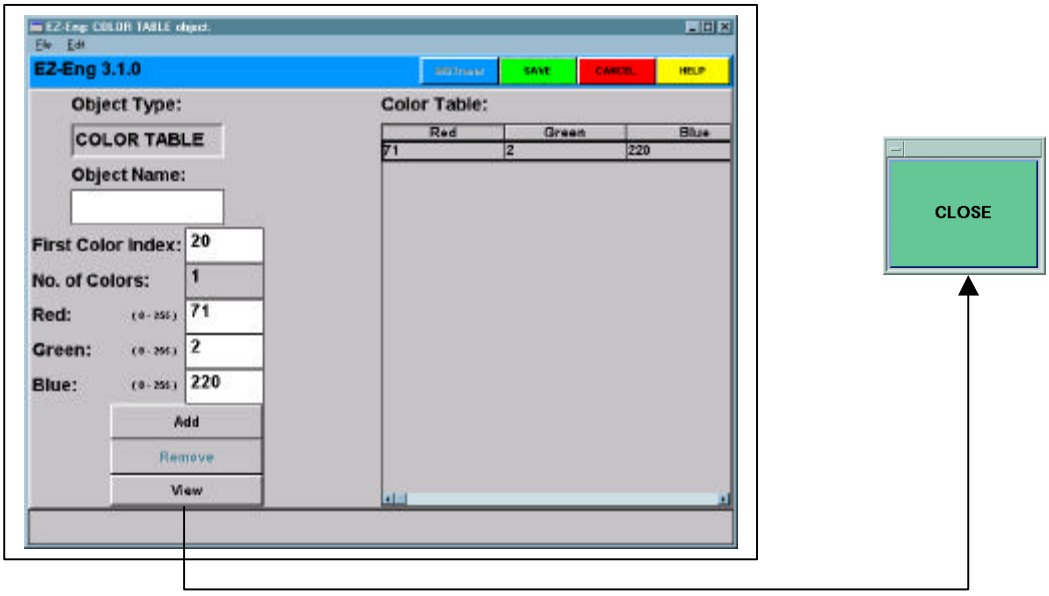
Configuration forms

1. **OBJECT TYPE:** Check Box or Radio Button: The Check Boxes and Radio Buttons consist of an indicator (a rectangle in the case of a Check Box and a diamond in the case of the radio button) and a label. The state of the button is taken from the Check Box meta script. When the button is checked (depressed) by a mouse click, the check meta script is executed. When the button is unchecked (raised) by a mouse click, then the UNCHECK meta script is executed.
2. **Object Name:** The name or ID of the object.
3. **Checkbox:** This is a meta script that determines the state of the Check Box or radio button. The meta script must return a non zero for a true, (checked) (indented) box and it must return a zero for a false (unchecked) raised box.
4. **Check:** This meta script is executed when the user attempts to check (indent) a box or button that is in the raised condition.
5. **Uncheck:** This meta script is executed when the user attempts to uncheck (raise) a box or button that is in the indented condition.
6. **Update:** The time following a user-adjustment before the system again starts to update the Check Box or Radio Button from the field. This gives the system time to write the value to the field and read the value back before the value set by the user is changed to the new current value.
   Note: This should be set to 0 if the variables are being used for indication only.
7. **Color:** A meta script that returns the color of the label text.
8. **B.Color:** A meta script that returns the background color of the label.
9. **Text:** The label string, that is to appear next to the Check Box or Radio Button.
10. **Text Size:** The text size in VDC units.
11. **Text Type:** The font used with the label I.e. Courier, Times or Helvetica.
12. **Style:** The various style attributes to be applied to the label. E.g. BOLD, ITALIC etc.
13. **Focus*:** The check box or radio button asks for the focus when the window is first brought up. (*=Not yet supported).

9.15 Colour Table

This is used to create custom colours for a graphic.

Colour Table Configuration Items



Main Features

1. **Structure:** The custom colors are defined for each graphic. The object consists of a list of colors. Each member of the list contains an index number and the 3 principal colors; Red, Green and Blue.
2. **Purpose:** The Color Table Object defines the custom colors (in addition to the standard 16 colors) for the graphic.
3. **Object Name:** The object name of the Color Table. This is used for your reference only.
4. Index: This is the number associated with the color. In a meta script, you would return this number if you wanted to select the first color. Add 1 for the second color etc.
5. **R, G and B:** These are the Red, Green and Blue color components for the color. Each component is ranged between 0 and 255.
6. **Add:** Adds a color to the color table.
7. **Remove:** Removes the selected color from the color table.
8. **View:** Pops up a window displaying selected color.

Creation Procedure:

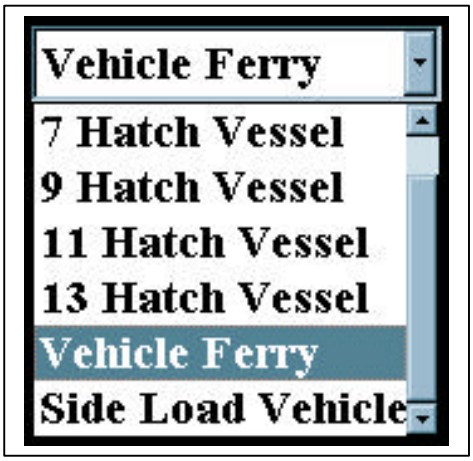
1. There is no mat associated with the Color Table. In Engineering mode, just use the right button in a blank section of the display and select the ***Color Table*** option to modify the color table form.
2. Add a new index using the Add button.
3. Modify the values for that index.
4. View the color using the View button.
5. If the first color in the index is 20, then any object in this metafile which has a color program which returns 20 will be shown in this color.

   Note: The first colour index can be defined by the user and it is important to remember that each additional colour added to the colour table is given the next sequential number. E.g. If the first colour is 20 then the next is 21 and so on.

9.16 Combo Box

The combo box consists of a scrollable pull down list that enables a user to select one of the options, which is then displayed as the selected option.

Combo Box Main Components



Main Features

1. **Structure:** The combo box consists of three main components: (i) A scrollable list of options, (ii) An edit/selection area and (iii) a pull down menu icon. These components behave slightly differently with the various styles of combo boxes.
2. **Purpose:** The combo box allows a user to select an option, (optionally edit the selection) and assign an associated value to an entity.attribute or variable.
3. Labels and Values: Generally, there is a configured list of labels with associated values.
4. **Editing:** If the combo box style allows editing you may allow the user to modify the entry. A validate script may also be used.
5. **Colors:** You may specify the color and background colors of the labels using meta scripts and also the color of the Edit Line text during editing.
6. **Combo Styles:** Three options are available: pulldown editable, pulldown non-editable and Down (always showing and editable.)
7. **Text Style:** You may choose the fonts, styles, text height etc. of the list.

Operations:

To use a combo box, a user will:

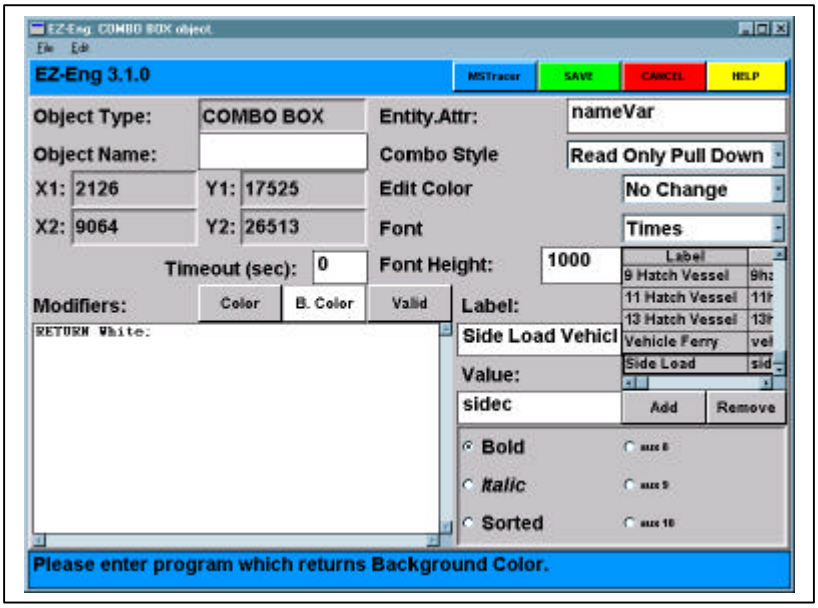
1. Click on the down arrow to extend the list of options. (If it is not a list style combo box.)
2. Use the down & up cursor keys or mouse to find the part of the list that is relevant,
3. Click on the required label. At this stage the label will appear in the top window of the combo box after validation,
4. Edit the label (if an editable combo has been specified.) Click elsewhere in the window or on the downward pointing arrow to activate the validate program and initiate the write request.

Creation Procedure:

1. In the CAD diagram, use the mat icon to draw a mat in the position of the target combo.
2. Using the combo object editor form, add the features such as labels and values to the object.

The diagram below provides an overview of the main configurable items of the Combo Object.

Combo Box Configuration Items



1. **Object Type:** Combo Box: As discussed on the previous page, the combo box consists of three main components: (i) A scrollable list of options. (ii) An edit/selection area and (iii) A pull down menu icon. These components behave slightly differently with the various styles of combo boxes.
2. **Object Name:** The name or ID of the object.
3. **Entity.attr:** The target for the selected variable. When the user selects an option, the associated value is written to this entity.attribute or variable.
4. **Combo Style:** You may choose from three styles of combo. These include Read only pulldown, Editable pulldown, and Editable Down.
5. **Edit Color:** The edit color is the color of the edit line text while an edit operation is being performed.
6. Time-out: If the user begins editing the combo box value, the dynamic updating of the value is suspended for this time-out period. Setting this to zero makes the combo non-editable.
7. **Label:** The labels are displayed in the scrollable list. You need one label for each option.
8. **Value:** These are the values that correspond to the labels. These will be written to the target entity.attribute. You need a value for each option. **Note:** The labels and values may be the same.
9. **Color:** The color meta script determines the color of.
10. **B.Color:** The background color meta script determines the color of the background of the list of labels.
11. **Text Style:** There are various style attributes that may be applied to the text in the Combo list. I.e: BOLD or ITALIC.
12. **Font Height:** The font height in VDC units. Please see the description "VDC Units" on page7.
13. **Font Type:** The font to be used for the contents of the cells, edit line and the labels. The size of the edit line portion of a combo box is based on this value.
14. **Validation:** This is used to determine if the entity.attribute (or variable) should be set to the value of the selection. To determine the selection, the variable SetRequest will contain the value just selected. This may be used to prevent the user from selecting certain items. E.g.

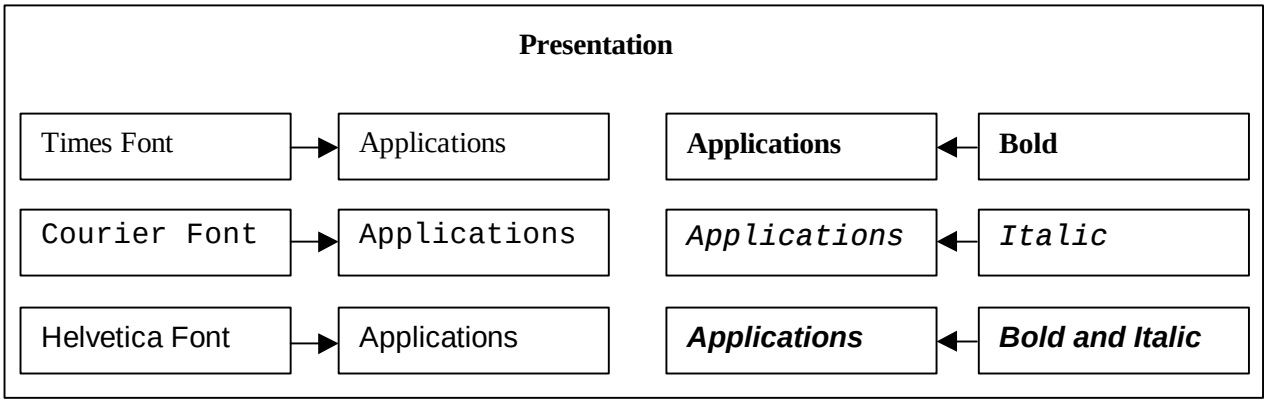
```
IF ( SetRequest = 5 .AND. UserName
<> "Engineer" ) RETURN 0;

ELSE RETURN 1;
```

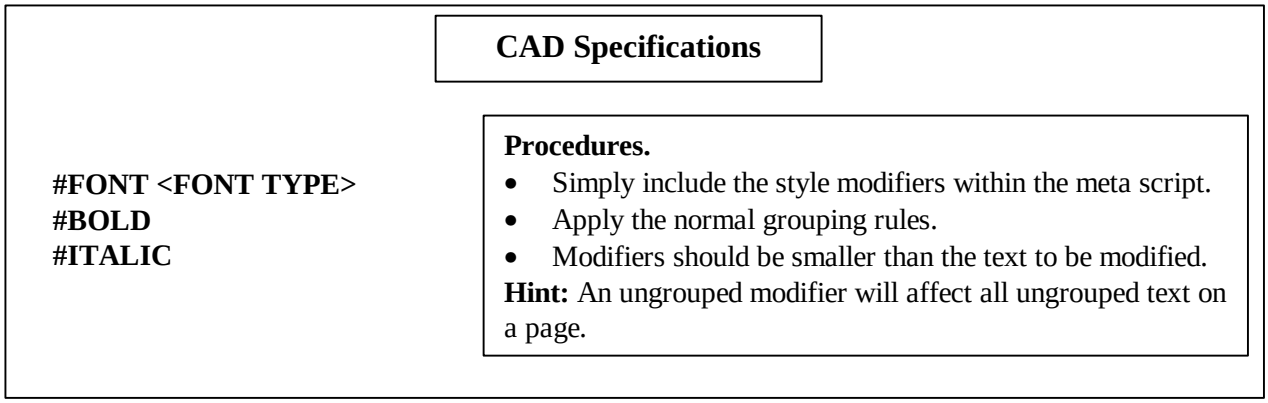
9.17 Common Objects

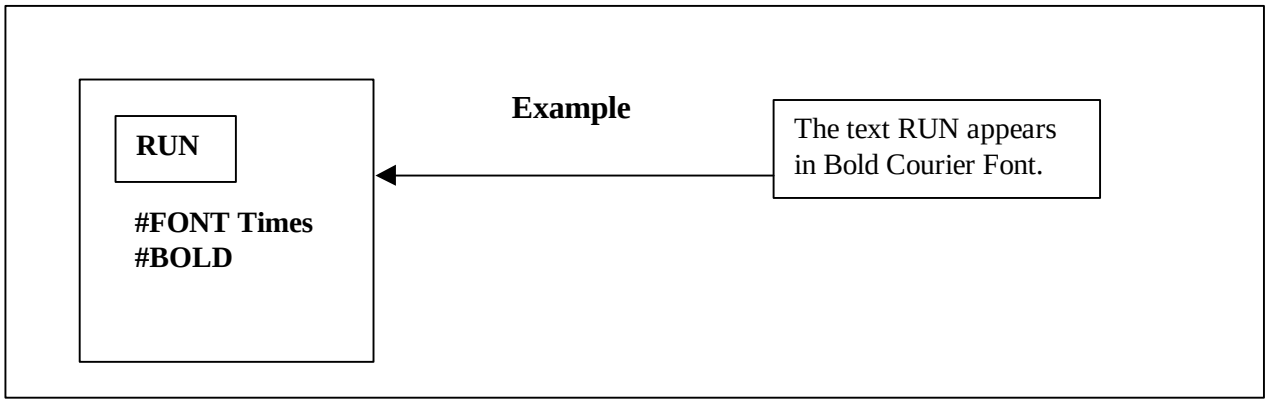
There are various objects that are common to all the objects. For example, the style of text may relate to static text etc. These are discussed in this section Style (#FONT, #BOLD, #ITALIC)

These common modifiers may be used for static text.

These modifiers are not appropriate to mats or meta sprites.







Note: These setting can be applied to text objects through the Object editor functions or to text objects, which you create in your CAD drawing. The above examples refer to the functionality being applied within the CAD drawing.

Table 7: Style Modifiers

Modifiers	Description
FONT default argument options example	Defines the font of a text object. Courier. Times, Courier, Helvetica. **#FONT Times**
BOLD default example	Makes the text in any object bold. If there is no **#BOLD** modifier, the text is shown as regular. **#BOLD**
ITALIC default example	Makes the text in any object italic. If there is no **#ITALIC** modifier, the text is shown regular. **#ITALIC**

9.18 Complex Move (#MX, #MY)

The Complex Move modifier moves a shape to x and y co-ordinates that are defined in separate x and y co-ordinate expressions. The complex move object has been included here purely for backward compatibility.

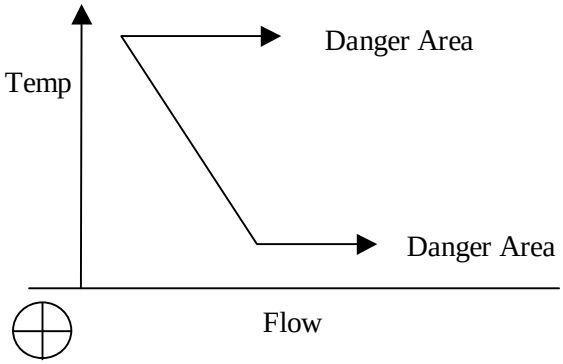
NOTE: *The preferred approach is to use the metasprite object. (See "Meta Sprites with Complex Movement" on page74.) This is preferable because of the flexibility of the meta scripts which determine the position of the object.*

General Form

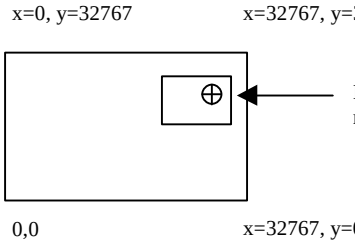
```
#MOVE
#MX <x expression>
#MY <y expression>
```

Where the x expression and y expressions are expressions that return x and y co-ordinates in VDC units for the position of the shape

Examples




```
#MOVE
#MX 16000 + PV(FIC100)40
#MY 16000 + (PV(TIC100) - 100) *40
```



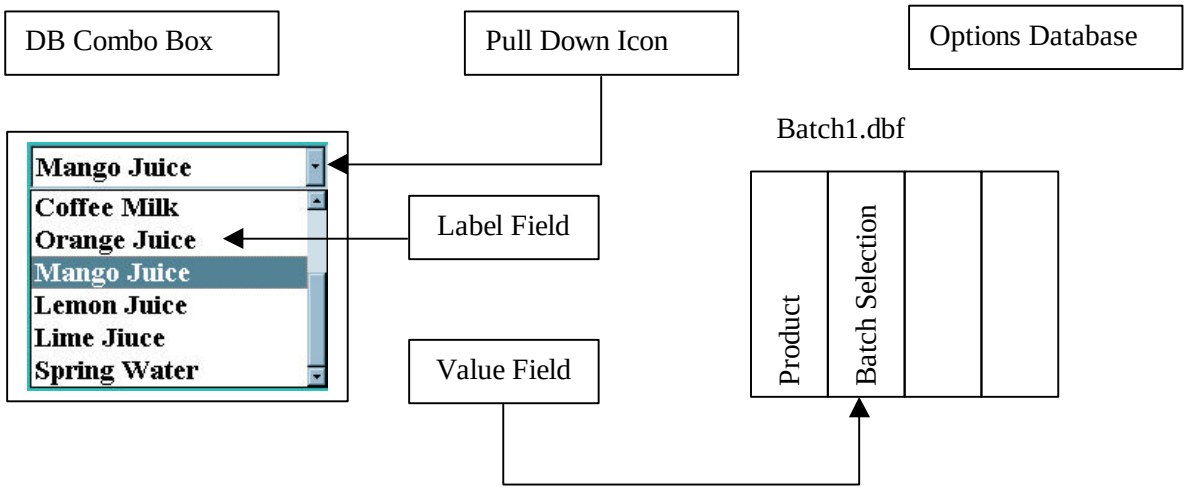
Procedure:

1. Draw the shape that is to move around the screen and group it on its own.
2. Enter the `#MOVE, #MX` and `#MY` modifiers.
3. Group the shape and the modifiers.

9.19 Database Combo

The Database Combo is similar to the normal Combo Box, however, instead of the options being hard coded, they are read from a database, a view of a database, or a table variable.

DBCombo Widget Main Elements



1. **Structure:** The elements and functions of the database combo box are constructed from an options database. The labels come from a column of the database and the values associated with those labels come from a second column.
2. **Purpose:** The DBCombo allows a user to select a label (from the list of options from a filtered database) and the associated value (from another field in the database) will be written to the target entity.attribute (or variable).
3. **Database Definition:** The DBCombo reflects the state of the database. This means that the DBCombo may be modified without changing the graphic display.
4. **DBCombo Styles:** You may choose from several styles of DBCombo. These include pulldown, list, editable etc.
5. **Filters:** A filter may be applied to the database.
6. List Style: You may choose the fonts, styles, text height etc. of the list.
7. **Editing:** You may allow the user to modify the values. The value can be configured to change color during editing. You may also specify a validate meta script.
8. **Writing to the target:** Various parameters, such as security and a time-out value, affect how the value is written to the target entity.attribute or variable.

Operations:

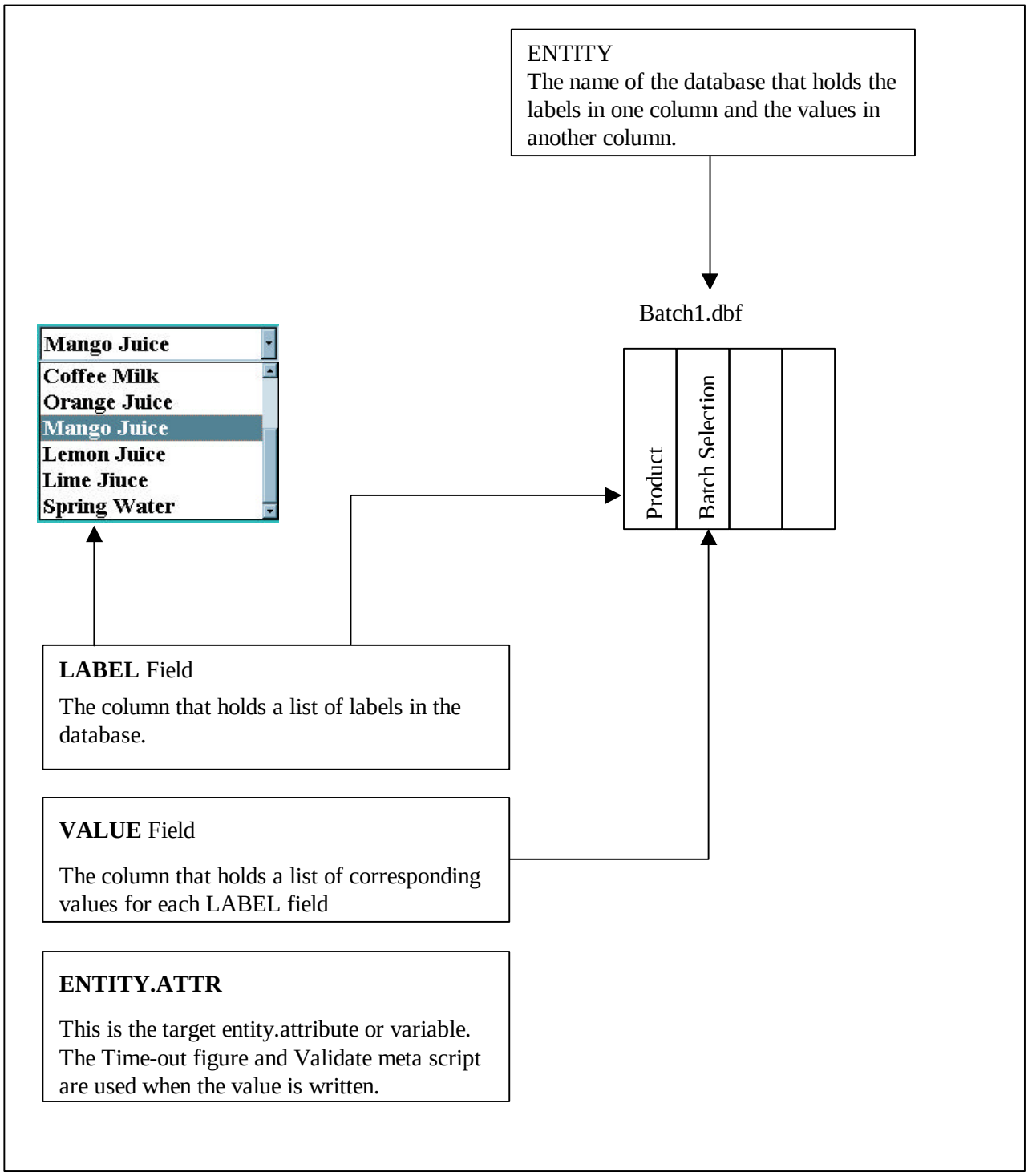
To use a combo box, a user will:

1. Click on the down arrow to extend the list of options. (If it is not a list style combo box).
2. Use the down & up cursor keys or mouse to find the part of the list that is relevant.
3. Click on the required label. At this stage the label will appear in the top window of the combo box and the selected value will be assigned to the variable SetRequest until the validation program is complete.
4. Edit the label (if an editable combo has been specified). Press enter to activate the validate program and initiate the write request.

Creation Procedure:

1. In the CAD diagram, use the mat icon to draw a mat in the position of the target combo.
2. Create a VIEW or TABLE VARIABLE from which the DBCombo will get its' labels and values.
3. Using the DB Combo object editor form, add the features such as labels and values to the object.

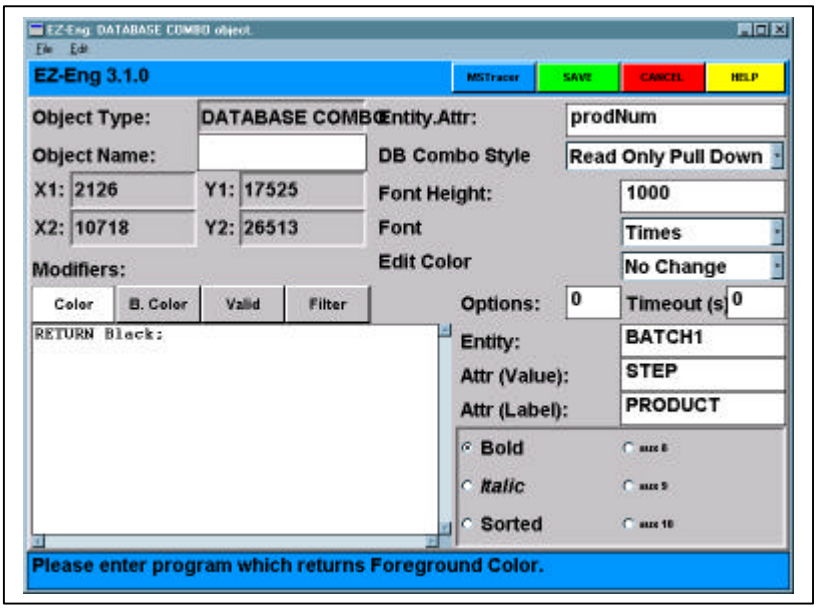
Explanation of Terms



DBCOMBO Configuration Form

The diagram below provides an overview of the main configurable items of the DBCombo Object.

DBCombo Box Configuration Items

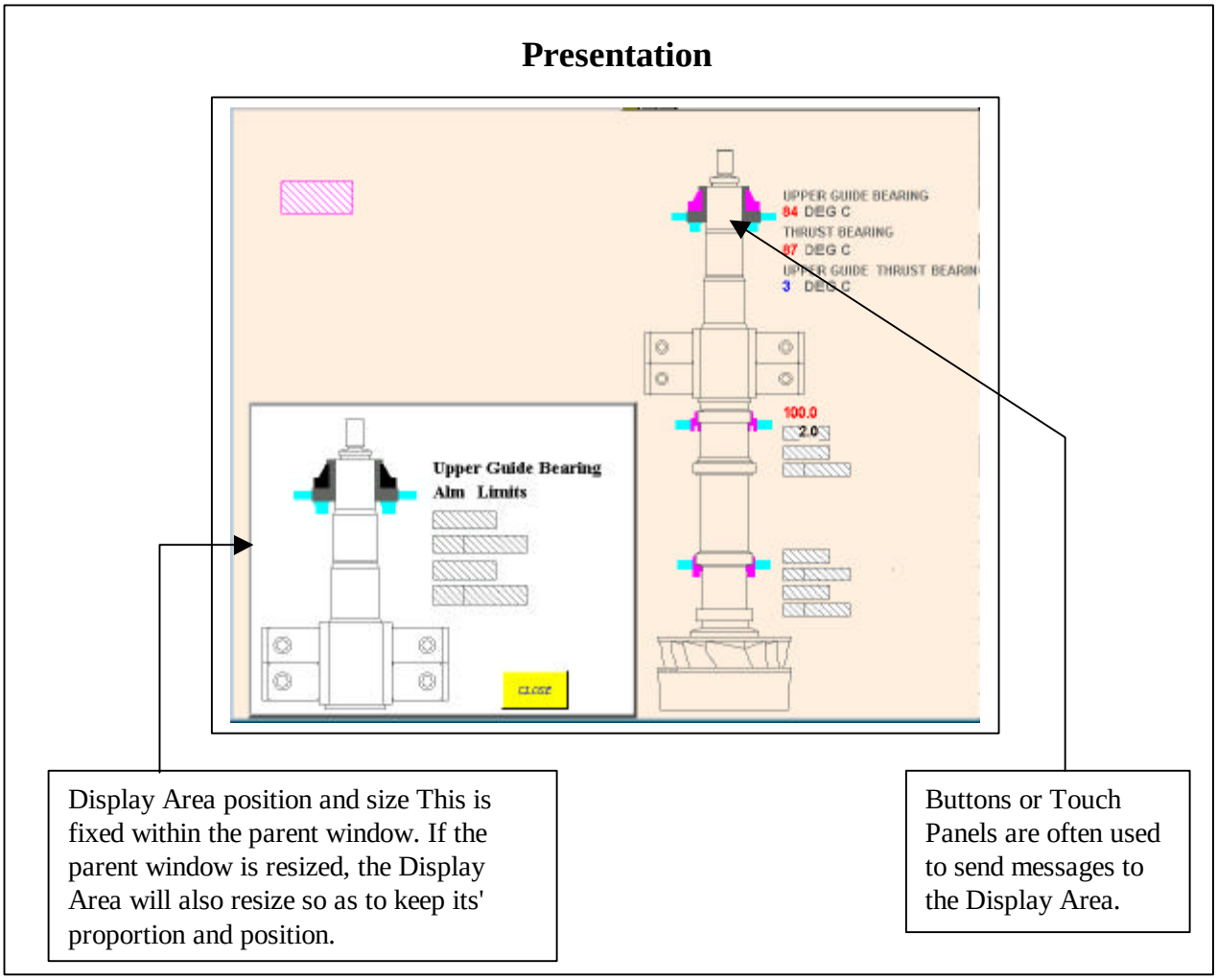


1. **Object Type:** Database Combo Box: The Database Combo Box is a combo box that gets it's labels and values from columns (fields) in a database table. The table may be filtered.
2. **Entity.Attr:** This is the target entity and attribute (or variable) of the Combo Box. When a label is selected, the associated value from the database is written to this entity.attribute or variable. The normal security structure applies.
3. **Entity:** The name of the VIEW that holds the labels in one column and the values in another column. (Or both in the same column.)
4. **Field(Label):** The name of the field which holds a list of labels in the ENTITY database. These labels will appear in the database combo box.
5. **Field(Value):** The name of the field which holds a list of associated values in the ENTITY database. When a label is clicked, the associated value will be written to the target entity.attribute or variable. Note: The label and value entries may be the same.
6. **Filter dBase filter language:*** Similar (but not identical) to meta script. The filter is used to select valid labels and values from a database.
7. **Valid:** The selected VALUE is stored in the variable SetRequest. The valid meta script is executed when the user selects an item off the combo box.
8. **Time-out:** If the user begins editing the combo box value, then the dynamically updating value is suspended for this TIMEOUT period. A zero (0) makes the combo non-editable.
9. **Edit Color:** The edit color is the color that the editline text is shown in when an edit operation is being performed.
10. **DB Combo Style:** This is the style of combo box. I.e a read-only or read/write and pull-down or always displayed.
11. **B. Color:** The Background Color meta script determines the background color of the elements in the list of options.
12. **Style:** There are various style attributes that may be applied to the text in the DBCombo list. I.e.:
13. **Font Height:** The font height in VDC units. Please see the description of VDC units. The size of the edit portion of the combo is based on this number.
14. **Font Type:** The font to be used for the contents of the cells, edit line and the labels.

9.20 Display Area (#DISPLAY)

A Display Area is an area physically located within the parent graphic that is under the control of a different metafile. You can send messages to the Display Area to change the presentation. (Typical messages include Panning and Zooming messages)

This Display Area is defined in the CAD drawing and at this time can not be created through the Object editor.



The Display Area is normally configured to display a default of some kind when the display is first called up. Any graphic object, which is cable of having an action associated with it, can be used to send a message to the Display Area. This would typically be another display, as in the example above, where the user can click on various parts of the turbine to see more information in the Display Area.

Table 8: Display Area Modifiers

Modifier	Description
DISPLAY	Creates a Display Area within the main area that is under the control of the specified metafile.
form	`#DISPLAY <metafile>`
argument	`<metafile>` This is the name of the metafile that is to be used for the Display Area.
example	`#DISPLAY tuntrd`
RANGE(optional)	You can use the RANGE modifier to directly specify the co-ordinates of the displayed metafile.
form	`#RANGE <XTopLeft>, <YTopLeft>, <XBottomRight>, <YBottomRight>`
argument	The arguments define the top left and bottom right co-ordinates directly in VDC units. The VDC units for the whole parent window are 0,0 for the top left corner of the parent window and 32767,32767 for the bottom corner of the window. This is used in the rare case that you only want a portion of the metafile to be displayed.
default	If no arguments are specified, the Display Area will cover the parent window completely.

example	`#DISPLAY grpfp`
	`#RANGE 0,0,4095, 32767`
	This example shows the first faceplate in the Group display area.

#ID	You should assign an ID name to the Display Area. Once defined, you can use this name in your `SEND <message> TO` statements. Please see the section on Object ID's for more information.

Messages

The table below summarises the various messages that can be sent to a Display area, a Paging Display[2] or a Group Display.

Typically, additional operator interface functionality can be added by utilising other dynamic graphic objects (such as buttons) and the SEND meta script command. The SEND command allows a meta script to send a string message to an object such as a Display Area object. I.e. `SEND "message" TO "ObjectName";` The Display Area object performs specific actions if it receives specific forms of messages. The messages that the Display Area, Paging Display or Group Display objects understand are listed in the table below:

[2] Group Display and Paging Display are advanced features not included in this manual.

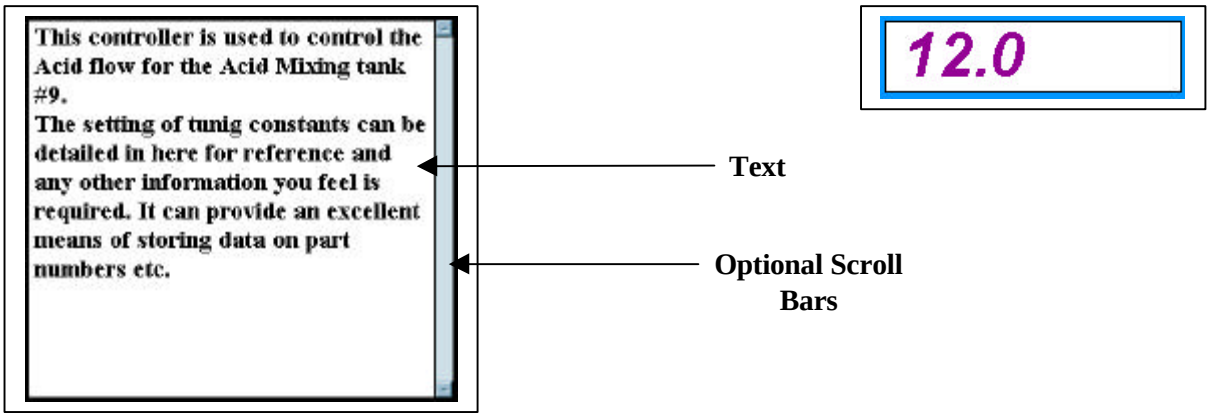
Table 9: Message Processing Summary

Message Format	Action Performed
Display(<meta filename>)	Used to change the contents of a display area. **Example:** **SEND "Display(../schemat/file3)" TO "QuickDisplay";**
SetDefaultEntity(<entity>)	Used to set the wild card values of a display that contains the $ character will have the <entity> substituted for all $s. Used particularly in detail displays. **Example:** **SEND "SetDefaultEntity(FIC100)" TO "QuickDisplay";**
Pan(horizontal, vertical)	If panning is enabled, then scroll horizontally by a fraction of a page given by horizontal and scroll vertically by a fraction of a page given by vertical. Do nothing if panning is not enabled or if the window is already scrolled fully in the requested direction(s). Negative values of horizontal scroll to the left, while positive values of horizontal scroll to the right. Likewise, negative values of vertical scroll toward the top of the image, while positive values of vertical scroll toward the bottom. **Examples:** Pan (10, -100) moves one tenth of a page to the right and one page up. Pan (-500, 0) moves five pages to the left.
PanEnable	Enable Panning on the currently visible image. Scrollbars are drawn to allow viewing the entire 0-32767 VDC coordinate space for each axis.
PanDisable	Disable Panning. If panning was enabled then the scrollbars are removed and the image is resized slightly to use the newly vacant space.
ZoomIn(magnification)	If zooming is enabled, scale the currently visible image by magnification. The magnification parameter is expected to be a floating-point number in the range 1.0 < magnification <= 10.0 The window is positioned such that the center of the image prior to zooming remains the center of the image after zooming. If zooming is not enabled, this message is ignored.
ZoomOut	If zooming is enabled and the image has been magnified, revert to the magnification that was in use prior to the last ZoomIn operation. If zooming is not enabled or if no prior ZoomIn operation has been performed, this message is ignored. Note that it is never possible to zoom out beyond the image rectangle (specified in the metafile and possibly over-ridden by a ViewRect message).
ZoomEnable	ZoomEnable
ZoomDisable	Disables zooming operations.
ViewRect(left, top, right, bottom)	Resets the image to the given rectangle, specified in VDC units. If panning is enabled, then scroll bars are drawn so that the entire 0-32767 VDC coordinate space can be viewed. Any previous zooming information is discarded. Note that this message can be used even if panning and zooming are disabled.
ViewReset()	Reset the image to the original size and VDC positioning. NB: The original settings are over ridden with ViewRect() message.

9.21 Edit Area

There are three Edit Area objects which provide facilities for the entry and modification of text data: (i) The Edit Line, (ii) The Edit Box and (iii) The Text Editor.

Edit Area Main Components



1. **Structure:** There are three Edit Area objects which providing facilities for the entry and modification of text data. (i) The Edit Line displays a single line of editable text. (ii) The Edit Box provides a scrollable window of editable text. (iii) The Text Editor (which is similar to an Edit Box but has simple editor features available with the right mouse key when not in the engineering mode).
2. **Purpose:** The main purpose of the Edit Areas is to display and edit text in a line or in a convenient scrollable area.
3. **Entity.Attribute:** This is the entity and attribute (or variable) of the text to be viewed and edited. Often, the attribute is the field name of a dBase memo field.
4. Scroll bars: You may specify horizontal and/or vertical scrollbars.
5. **Wordwrap:** You may specify that the text word wraps in the allocated area if not using horizontal scrollbars.
6. **Text and Background Attributes:** You may specify the color of the text and the background color of the text. You may also specify the font, font size and attributes such as BOLD and ITALIC.
7. **Editing Facilities:** You may specify the maximum number of characters the user can type in, the "editing in progress" color, a maximum editing time-out, and a validation meta script. You may also specify that the object is Read Only. Additional features such as Find and Replace are available with the Text Editor object. Note that some operating systems may not allow this and that it is disabled in the engineering mode.

Operations:

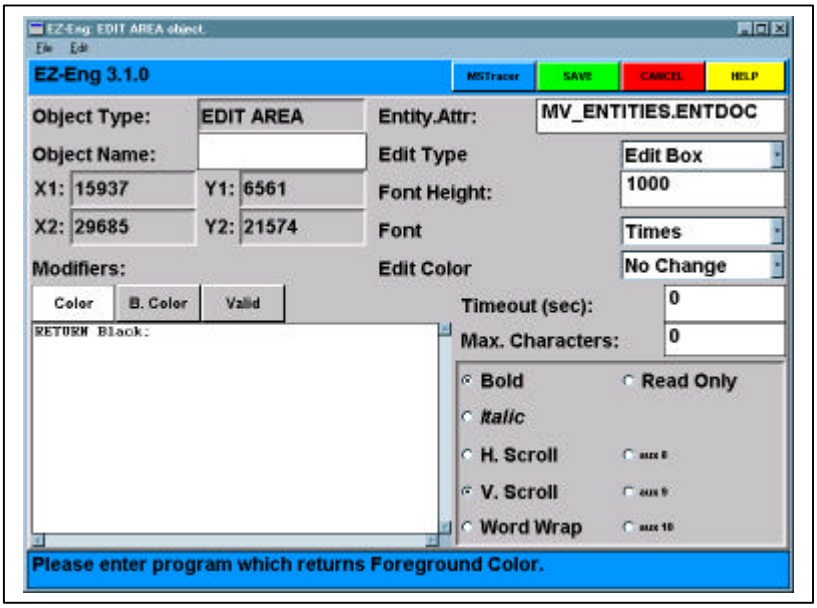
1. Uses the scrollbars, cursor keys or page-up, page-down keys or mouse to view the text information. You may use the arrow keys to scroll the Edit Line. The right mouse button brings up further options with the Text Editor object.
2. Edit the text using the usual edit keys.

Creation Procedure:

1. Create a mat in the CAD diagram.
2. Use the Edit Area form to achieve the design requirements.

The diagram below shows the main configurable items of the Edit Area.

Edit Area Configuration Items



1. **Object Type:** Edit Area: As described on the previous page. There are three Edit Area objects which providing facilities for the entry and modification of text data. (i) The Edit Line displays a single line of editable text. (ii) The Edit Box provides a scrollable window of editable text. (iii) The Text Editor (which is similar to an Edit Box but has simple editor features available with the right mouse key).
2. **Object Name:** The name or ID of the object.
3. **Entity.Attr:** The entity and attribute (or variable) of the text to be viewed and edited. For the Edit Area this often refers to a dBase memo field.
4. **Edit Type:** Either an Edit Line (for a simple line of editable text), an Edit Box (for a scrollable area of editable text) or a Text Editor which is similar to an Edit Box but offers additional Editing Facilities selected with the right mouse button.
5. **Color (meta script):** This is the meta script program that calculates the color of the text.
6. **B.Color:** This is the meta script program that calculates the background color of the text.
7. **Edit Color:** When a user initiates an edit operation, the text will turn this color to show it is under the control of the user and not being updated.
8. **Time-out:** If an edit operation is started but not completed in this time-out period, then the screen will revert to updating mode after the last user key press. The value will remain static after the data entry is complete for this period.
9. **Valid:** The valid meta script must return a 1 for the write process to be undertaken. The valid meta script may check the contents of the SetRequest variable for the text that is about to be written to the entity.attribute (or variable).
10. **Max Characters:** The maximum, number of characters the user can type. (0 means no limit).
11. **Word Wrap:** The text will Word Wrap in the edit box.
12. **Read Only:** No editing is allowed.
13. **V.Scroll:** If specified, a vertical scrollbar will be displayed as part of the object.
14. **H. Scroll:** If specified, a horizontal scrollbar will be specified as a part of the object. Note that, if word wrap is selected, no horizontal scrollbar will be shown even if H.Scroll is selected.
15. **Focus*:** The Edit Area asks for the focus when the screen is first brought up. (*=Not yet supported).
16. **Font:** This is the font of the labels. I.e. Courier, Times or Helvetica.
17. **Font Size:** This is the size of the label text in VDC units.
18. **Style:** These are the various items of style applied to the label text. e.g, Bold, Italic etc.

9.22 Focus Issues

In the design of an efficient input screen, it is most important that you provide assistance to the operator in moving the focus to the relevant objects.

In this context, when an object has the focus, all key presses are sent to the object[3]. If the object has no use for these key presses, they are sent on to the parent of the object. E.g. if the focus is on a button, and the user presses the G key, the button will have no use for this key press so it will pass it on to the parent who, if this is within the Navigator, would display the Groups application.

Changing focus with the Tab keys.

The operator can change focus from Object to Object using the TAB keys. The functions are as follows

TAB or (Control TAB) moves the focus to the next object in the metafile.

Shift TAB or (Control Shift TAB) moves the focus to the previous object in the metafile.

Note that the Control key option is preferable because TAB and Shift TAB keys are accepted within edit areas as data and not as "Change Focus" commands.

Using the SEND "Focus" command

To move the focus to a particular object, use the SEND "Focus" command. e.g.

```
SEND "Focus" TO "batchInput";
```

This will send the focus to the object called batchInput.

You may also send the focus using the metafile: object form of the command, which sends the focus to an object in another window, specified by the metafile label in the command. The following objects are some of the objects that accept this command:

- Edit Lines, Edit Boxes, Text Buttons, Picture Buttons and combo boxes.

[3] An item with focus will have a black outline. The “validate program” of an object is a good place to send "Focus" as it can be used to lead a user through a data entry page.

9.23 Graphic Regions

These are regions or elements that act as the static background or form part of an object (e.g. colour changers.)

Graphic Objects

The table below shows the various kinds of graphic objects supported by the operating program.

Note: In each case, the object must be grouped with the modifier.

Table 10: Graphic Region Summary

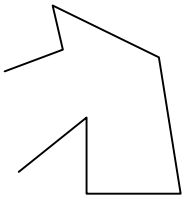
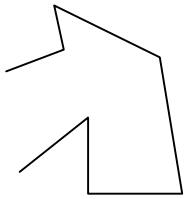
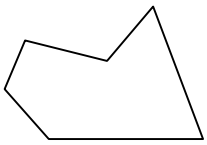
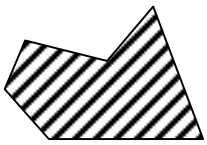
Object type	CAD Specification	Operations Program	Comments
Polyline			
Pattern Fill (Filled Polyline)	 #PATT 3		Please see the note on filled areas. **Note:** Polygons should not have four points otherwise they will be considered to be rectangles.
Rectangle			
Filled Rectangle			Please see the note on filled areas.

Table 11: Graphic Region Summary

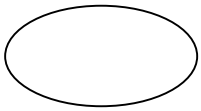
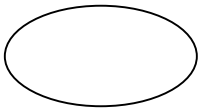
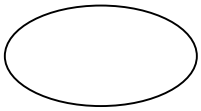
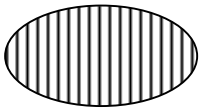
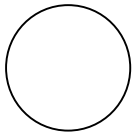
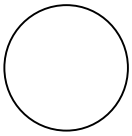
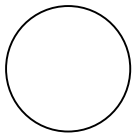
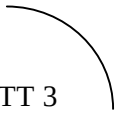
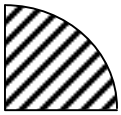
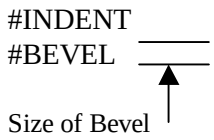
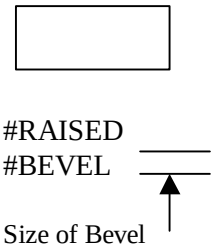
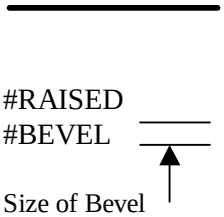
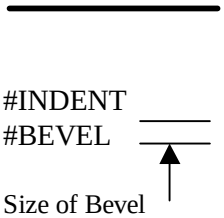
Object type	CAD Specification	Operations Program	Comments
Ellipse			
Filled Ellipse	 #PATT 3		Please see the note on filled areas.
Circle			
Filled Circle	 #PATT 3		Please see the note on filled areas.
Arc			
Filled Arc (Pie Slice)	 #PATT 3		Please see the note on filled areas.

Table 12: Graphic Region Summary

Object type	CAD Specification	Operations Program	Comments
Indented Panel			The thickness of the bevel is taken from the height of the #BEVEL modifier. The color of the indented panel is taken from the CAD rectangle or polyline.
Raised Panel			The thickness of the bevel is taken from the height of the #BEVEL modifier. The color of the raised panel is taken from the CAD rectangle.
Shaded Line (raised)			The thickness of the bevel is taken from the height of the #BEVEL modifier. The color of the shaded line is taken from the CAD line. **Note:** Shaded lines must be vertical or horizontal.
Shaded Line (indented)			The thickness of the bevel is taken from the height of the #BEVEL modifier. The color of the shaded line is taken from the CAD line. **Note:** Shaded lines must be vertical or horizontal.

Note on Filled Areas

The next diagram shows the preferred method of creating filled regions.

CAD Specification

Color of Object defines foreground color.

Resulting Presentation

Filled Areas
Use both the object color and #PATT modifiers.
The resulting background will have:

1. The foreground color set to the color of the CAD object,
2. The background color set to the color of the #PATT modifier,
3. The pattern or texture set to the pattern number defined in the #PATT modifier.

#PATT 9

Color of Modifier defines the pattern background colour.

Number defines pattern style. (Please see the pattern summary below.) **(1 to 11)**

Procedures

1. Draw the object in the desired foreground color,
2. For a textured background, type in the `#PATT <patternNum>` modifier in the desired background color.
3. Group the modifier and object in a single group. (patternNum = 1 to 11)

Pattern Summary

1	All Background	
2	All Foreground	
3	Mostly Foreground	
4	Mostly Background	
5	Even Mix of Foreground and Background	
6	Horizontal Lines	
7	Vertical Lines	
8	Diagonal Lines	
9	Diagonal Lines	
10	Grid	
11	CrossHatch	

9.24 Graphic Text

This is text that is entered as part of the background of a graphic or as part of an object within an updating graphic.

Presentation

The table below shows how the various attributes of the text information are passed across from the CAD diagram to the operational graphic.

Note: You can combine various attributes of style together. For example, to have text that is bold and in Times font, you would specify the **#BOLD** and **#FONT Times**.

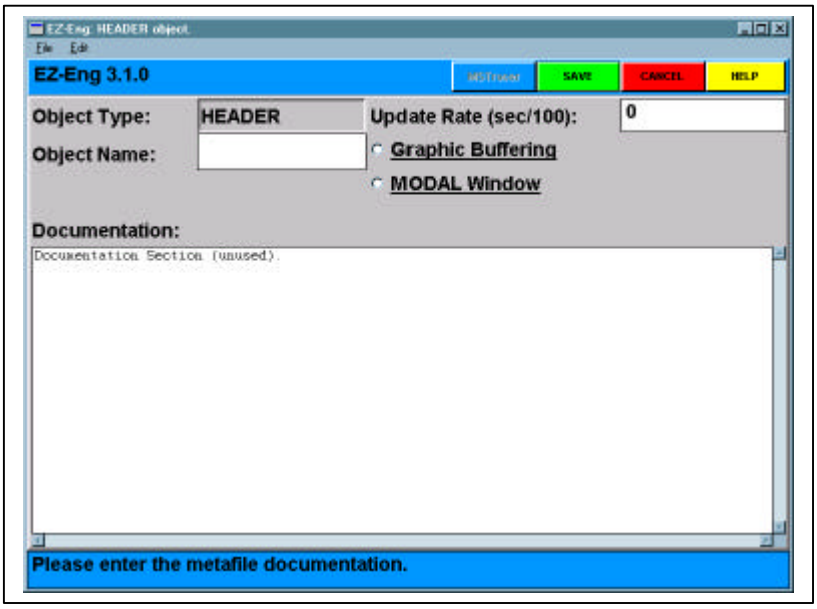
Table 13: Text Attributes

Attribute	Example	Method of Definition	Comment
Bold	Text Information	#BOLD	Please see the STYLE section in this chapter.
Italic	Text Information	#ITALIC	
Times Font	Text Information	#FONT Times	
Courier Font	Text Information	#FONT Courier	
Helvetica Font	Text Information	#FONT Helvetica	
Text Height	Text Information	CAD specification	The system will choose the font closest to the required height.
Text Alignment (horizontal and vertical.)	Text Information	CAD diagram	Based on the insertion point.

9.25 Header Information

The Header is used to define the basic metafile properties such as the scan rate etc. There is a default header for every metafile.

Header



Main Features

1. **Object Type:** The Header is used to define the basic metafile properties such as the Update rate. Enter the ID of the object in here.
2. **Purpose:** The Header object provides a means of defining the basic metafile attributes.
3. **Graphic Buffering:** If specified, this enables an automatic repaint of areas of the graphic that are revealed when a dynamic object moves.
4. **Modal Window:** The window will claim and hold the focus when it is brought up. It cannot be covered by other windows. Typically, you would specify a model window if you want an important dialog box to be acknowledged.
5. **Update Rate:** You may specify the rate at which the system updates the live values on the graphic. A slower update period will reduce the system loading significantly. Zero (0) is used as the default.
6. **Documentation:** You may freely enter text in the documentation area to record information such as the author of the graphic, its purpose, date of creation/modification etc.
7. **Note:** The documentation should not contain a colon (:).

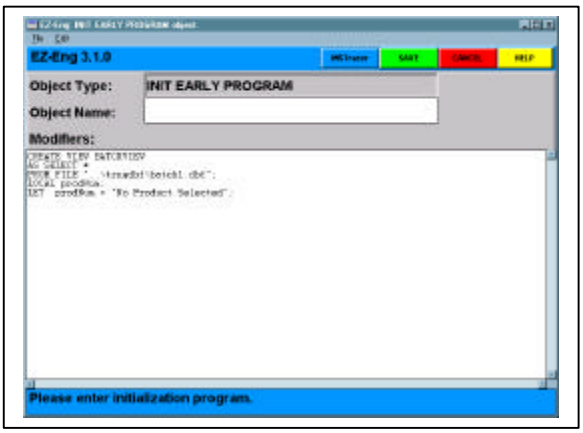
Creation Procedure:

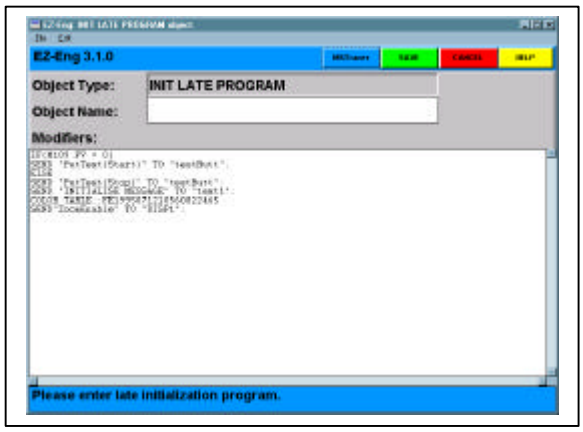
1. There is no mat associated with the Header. In Engineering mode, just use the right button in a blank section of the display and select the header option to modify the header form.
   Note: You must press the right mouse button in an area where there are no objects.

9.26 Init Early and Init Late

These meta scripts are executed when the graphic is first brought up (init early) and as the last meta script in the first pass of the `dgt` file (init late).

Init Early and Init Late Configuration





Main Features

1. **Structure:** The Init Early and Init Late objects are meta scripts that are embedded in the dgt file.
2. The Init Early meta script is the first meta script executed when the graphic is first brought up.
3. The Init Late meta script is run when all the dynamic elements of the graphic have been retrieved. It is the last meta script executed in the first pass of the dgt file.
4. **Purpose:** The Init Early and Init Late meta script are available to initialize variables, views etc. when the display is first brought up. These meta scripts are only run once.
5. **Object Name:** The object name of the meta script. This is used for your own reference only.
6. Modifiers: This is the actual meta script that will be executed.
7. **Modifiers:** This is the actual meta script that will be executed.

Creation Procedures

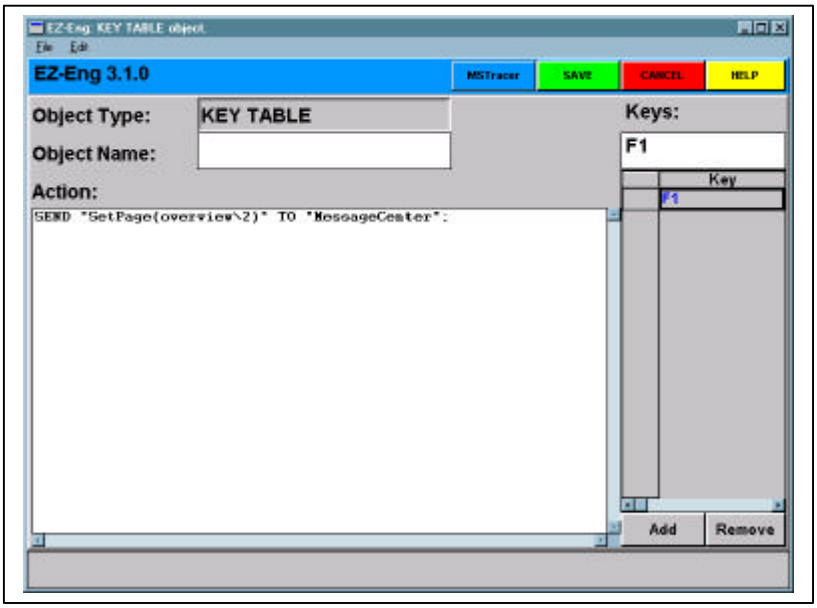
1. There is no mat associated with the Init Early or Init Late meta scripts. In Engineering mode, just use the right button in a blank section of the display and select the init or initiate options to modify the meta scripts.
2. You may use the meta script tracer to check out the programs.

Hint: If you want to send messages to objects in your metafile, you must put the SEND command in the Init Late meta script. Variables and views should be initialized in the Init Early meta script.

9.27 Key Table

This is used to define the actions associated with the given key press.

Key Table



Main Features

1. **Structure:** The key table defines the keys for the current graphic and all children of the graphic. The object consists of a list of keystrokes and associated meta scripts.
2. **Purpose:** The Key Table enables a user to define the actions for a given keystroke. When the keystroke is detected, the associated meta script is activated.
3. **Object Name:** The object name of the Key Table. This is used for your own reference only.
4. Key: This defines the keystroke. The following formats are valid: `<KeyName>` and `CTRL-<KeyName>.`
5. **Action:** This is the meta script that will be executed when the key combination is recognized.

Creation Procedures

6. There is no mat associated with the Key Table. In Engineering mode, just use the right button in a blank section of the display and select the key Table option to modify the key table form.
7. **Add a New Key using the Add button:**
   - Modify the key combination by typing it in above the list.
   - Add the associated meta script.
   - If necessary, check out the meta script using the meta script tracer.

9.28 Line Types

These are lines that are drawn on the graphic as part of the static background or as a part of an object.

Table 14: Line Type Conversion

DXF Line Type	Operational appearance	Operational Line Type
Dashed	___ ___ _____ ___	Dashed Line
Hidden	___ ___ _____ ___	Dashed Line
Center	___ . ___ . ___ . ___	Dash-Dot Line
Dot	. . . . .	Dotted Line
Dots	. . . . .	Dotted Line
Dash Dot	___ . ___ . ___ . ___	Dash Dot Line
Divide	___ . . ___ . . ___ . .	Dash Dot Dot Line

9.29 Listbox

The Listbox provides a scrollable list of options. When the user clicks on one of the options, the meta script program associated with that option is executed.

Listbox Main Elements

Main Features

1. **Structure:** This list box provides a scrollable list of options. If the list is too large for the assigned area, scrollbars will be displayed.
2. **Purpose:** The list box provides a list of available options. When an item is double clicked the associated meta script is executed. The list box can also be used to set the value of a variable or entity.attribute.
3. **Label Features:** You may specify the font and size of the text. You may also specify attributes such as BOLD and ITALIC. You may specify a color and background color meta script.
4. **Actions:** For each label, there is an associated meta script, which will be executed when the label is double clicked.
5. **Focus*:** You can specify that the keyboard focus is sent to the list box when the graphic is first brought up. (This is not yet supported).

Operations:

To use a list box, a user will:

1. Use the up and down cursor keys to find the part of the list that is relevant.
2. Click on the required option label. At this stage, the meta script action associated with the option label will be executed.

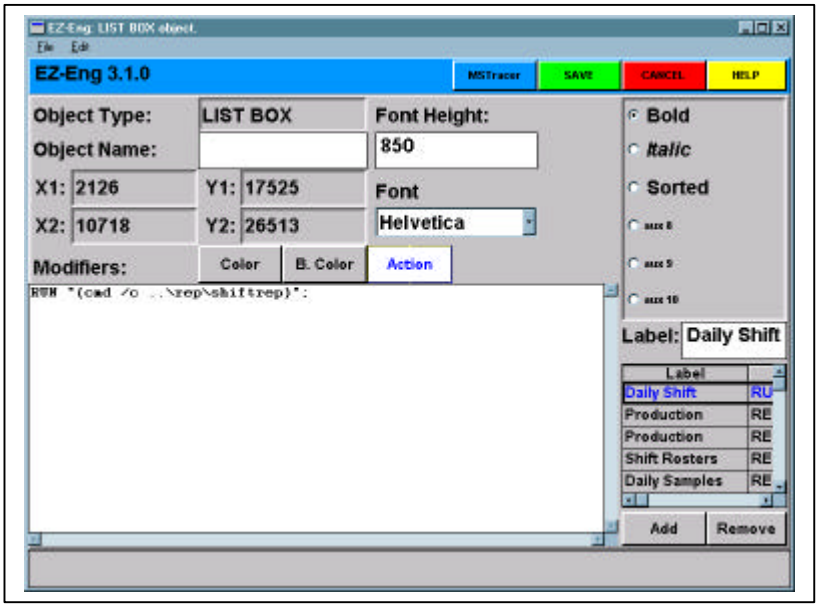
Creation Procedure:

1. Create a mat in the CAD diagram.
2. Use the LIST BOX form to satisfy the design requirements.

Listbox Configuration Form

The diagram below provides an overview of the main configurable items of the Listbox Object.

Listbox Configuration Items



Configuration form

1. **Object Type:** Listbox: Provides a scrollable list of options. When the user clicks on one of the options, the associated meta script is executed.
2. **Object Name:** The name of the object.
3. **Labels:** The actual text that is to appear as one of the selectable options in the list box. There is an action associated with each of these labels. Please see the Action item in this section.
4. **Actions:** The meta scripts that are associated with the labels in the list box. When the label is double clicked, the corresponding action meta script is executed.
   Note: The correct syntax for the metascript commands used for the action field is explained in the Metascript chapter.
5. **Text Type:** This is the font of the labels.
6. Text Size: This is the size of the label text in VDC units.
7. **Style:** These are the various items of style applied to the label text.
8. Color (meta script): This is the meta script program that calculates the color of the list box.
9. **B.Color:** This is the meta script program that calculates the background color of the list box.
10. **Focus*:** The listbox asks for the focus when the screen is first brought up.

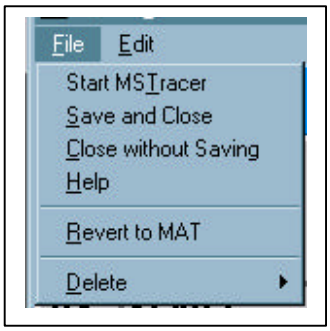
*** Not yet Supported.**

9.30 Menu Bar

The Menu Bar object is used to create a menu in a graphic.

Note that it is only possible to have one menu bar per operational program. (Also note that it is not possible to use the menu bar within the Navigator.) In essence this means you can create your own separate applications, to run independently from the Navigator, and have a Menu system in this application.

Presentation



In the example shown here, the File pull down menu has been selected either by clicking on the Applications word or pressing the ALT or META key together with the A key.

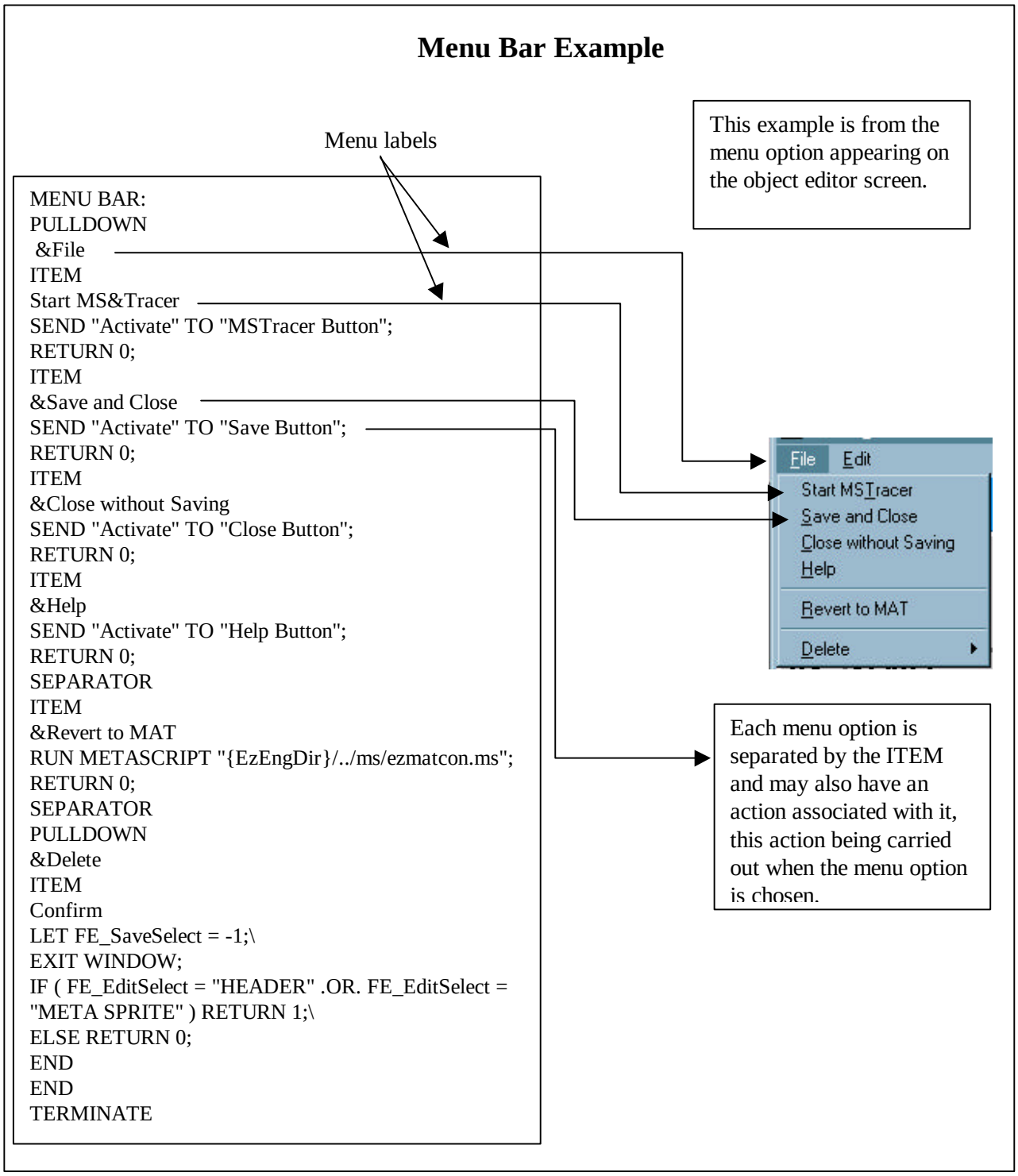
Main Features

1. **Structure:** The menu bar is a standard motif type menu object.
2. **Pulldown:** These are the individual menu structures. Whenever one of these is reached, a new menu structure is created. Note that you may "nest" menu structures within menu structures.
3. **Items:** These are the individual elements in the menu. They each have an optional action program and an optional disable program.
4. **Action Programs:** Each item in the pulldown menu may have an associated action meta script.
5. **Disable Program:** Each option in the pulldown menu may have an associated disable meta script. Use these to disable the action or menu.
6. **Accelerator Keys:** These be specified by preceding the accelerator key in the item label by an Ampersand (&).
7. **Separators:** These are lines that are drawn within the menu bar to logically separate various components of the menu.
   Note: The best way to understand the structure is to look at the example on the next page.

Creation Procedure:

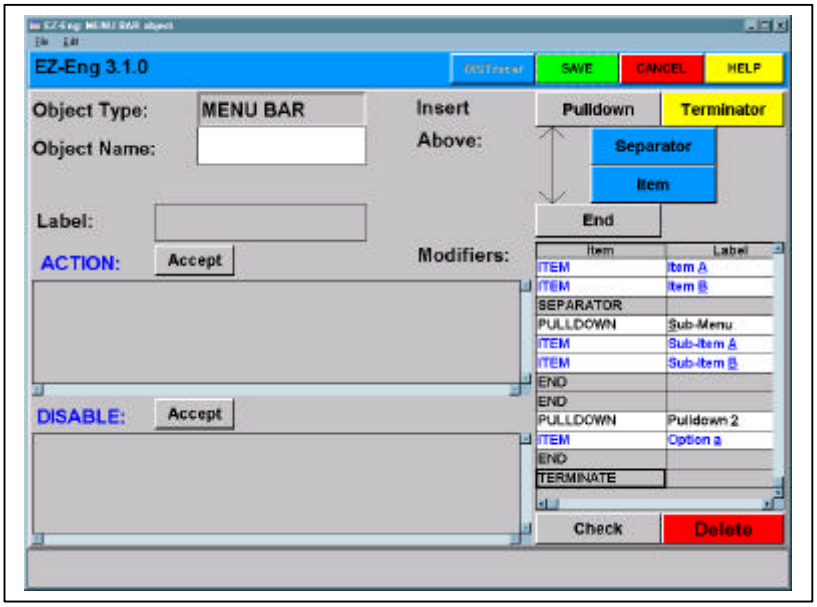
1. There is no mat associated with the Menu Bar Object. In Engineering mode, use the right button in a blank section of the display and select the menu bar option to call up the menu bar form.
2. **To Add a Pulldown:** Click on the insertion point in the Browse widget, Click on the pulldown button and add a label. After you have added all the items, sub-pulldowns and separators, you must end the pulldown group with the End item
3. **To Add an Item:** Click on the insertion point in the Browse widget, Click on the item button. Add a label, an optional action and an optional disable program.
4. **To Add a Separator:** Click on the insertion point in the Browse widget and then click on the separator button.
5. **To Terminate the Menu Bar Object:** Click on the last entry in the Browse widget and then click on the Terminator button.

Menu Bar Example



The diagram below provides an overview of the main configurable items of the Menu Bar Object.

Menu Bar Configuration Items



Configuration form

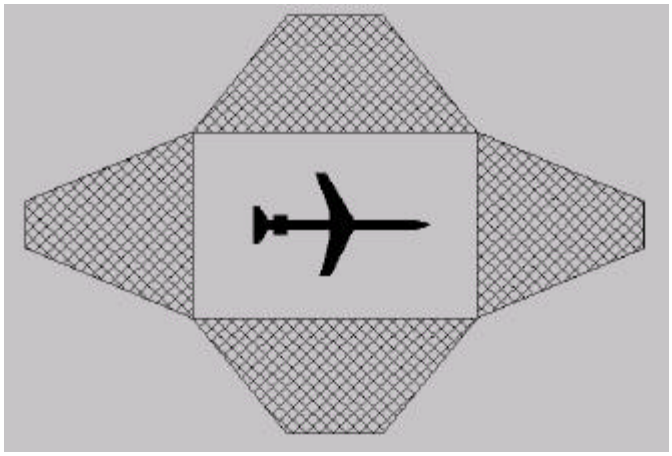
1. **Object Type:** Menu Bar: The menu bar is used to create a menu in an application. Note: that you may only have one menu bar per operations program (ops3) and you may not add a menu bar to applications running within the Navigator. (Except on pop-ups)
2. **Object Name:** The name of the menu bar object.
3. Item Browse Widget: This is a list of items including pulldowns, items, separators, ends and terminators.
4. **Label:** The name that will appear in the menu bar. Precede the accelerator key with an Ampersand (&).
5. **Action:** The meta script that will be executed if the item in the menu bar is selected.
6. **Disable:** If this meta script evaluates to true, (i.e. 1) then the item in the menu bar will be disabled.
7. **Pulldown:** Inserts a new menu sub-structure below the selected item in the Browse widget.
8. **Separator:** Inserts a new separator below the selected item in the Browse widget.
9. **Item:** Inserts a new item below the selected item in the Browse widget. The item should have a label and optionally, an action and disable meta script.
10. **End:** This marks the end of the pulldown submenu. The End item refers to the closest pulldown item above it in the browse widget.
11. **Terminate:** Inserts a terminate at the bottom of the Browse widget. This marks the end of the menu bar object. There should only be one Terminate item.

Note: After saving a menu bar for the first time, it may be necessary to resize the window after refreshing.

9.31 Meta Sprite

The Meta Sprite object uses meta scripts to control the colour, shape, scale and either position or angle of a graphic object.

Meta Sprite Main Components



Main Features

1. **Structure:** The meta sprite comprises a group of multiple graphics objects. The object's color, shape, scale and either position or angles are each controlled by an associated meta script.
2. **Purpose:** The meta sprite can be used to represent both simple and more complex dynamic animations such as Moves, Shape Changers, Dials etc.
3. **Color:** You may specify the color of the meta sprite with a meta script.
4. Shape: Choose from several CAD shapes using the shape meta script.
5. **Scale:** You can dynamically scale the object with the scale meta script.
6. **Value:** You can specify the position of the meta sprite for linear movement sprites or the angle with angular meta sprites.

Meta Sprite Options:

There are four kinds of meta sprite. Each is defined slightly differently. (Each has the ability to change color, shape and scale.)

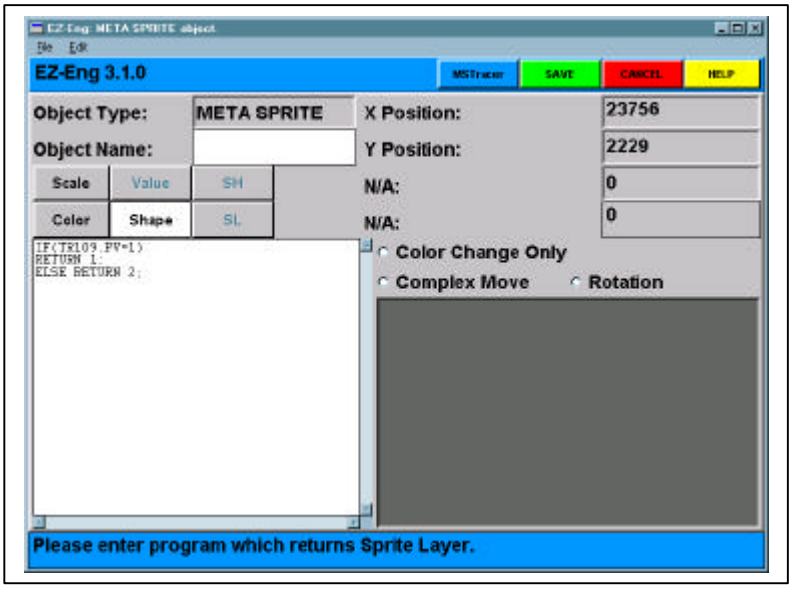
1. **Meta sprites that have no movement:** Please see page71.
2. **Meta sprites that have linear movement:** Please see page72.
3. **Meta sprites that rotate:** Please see page73 and
4. **Meta sprites that have a complex movement:** Please see page74.

Technical Note: Some meta sprites may not be visible during engineering due to the current scale, color or shape program. In order to allow easy identifications and selection of meta sprites, ops3 may be run with a "highlight" option. This option will draw a bounding box around each metasprite displayed.

If the meta sprites overlap, then selection with the right mouse button may select the incorrect metasprite.

The top meta sprite may be placed temporarily behind the meta sprites it covers by selecting it and selecting the "TO BACK" option on the mouse menu. This will redraw the screen with the selected metasprite at the bottom of the overlap stack. When the screen is next refreshed from the dgt file, the original order of display will be restored.

Meta Sprites with No Movement



Configuration Forms

1. **Object Type:** Meta Sprite with no movement: The meta sprite comprises a group of multiple graphics objects. The object's color, shape and scale are controlled by an associated meta script.
2. **Object Name:** The optional name or ID of the meta sprite object.
3. **Scale:** A meta script that returns a value to be used as a scaling factor for the CAD object. A value less than 1 will shrink the size, greater than 1 will enlarge it.
4. **Color:** A program that determines the color of the object. Returning -1 will keep the metasprite the same color it was drawn in.
5. **Shape:** A meta script that returns the CAD layer number on which the object is defined. In AutoSketch, 1 is the first layer, 2 is the second layer etc.
6. **SH:** Disabled for non-moving meta sprites.
7. **SL:** Disabled for non-moving meta sprites.
8. **Color Change Only:** This option is disabled if linear move is specified in the CAD drawing. It will mark the metasprite as having color change only, no scale or shape programs are executed.

Creation Procedure

1. Draw the objects in the sprite. Group the objects in a single group.
2. Add an identifier using the #I icon and group the #I with the first group.
3.

Color Change Only Example:

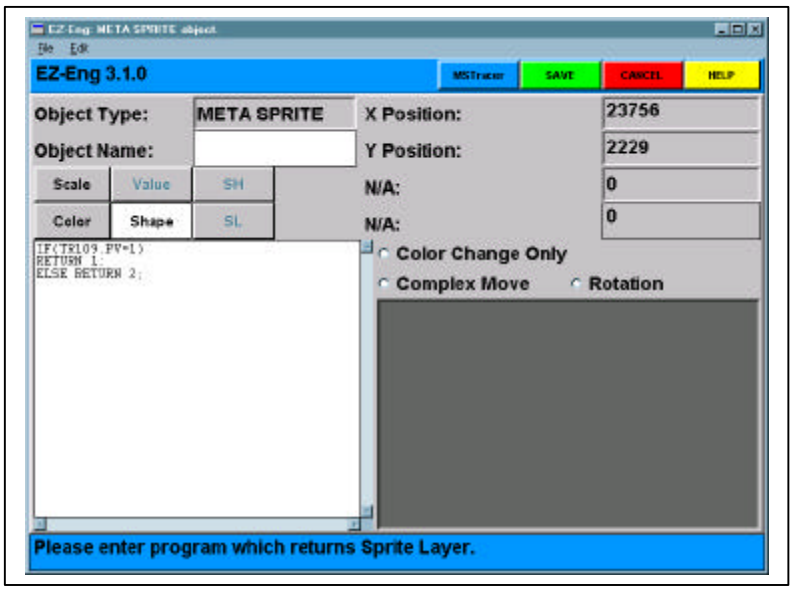
```
Color Chng only :T
Complex Move   :F
Rotation       :F
Color          :IF (TANKLVL.ALM = 1)
                  RETURN Red;
                  ELSE RETURN Green;
Others         :N/A
```

Scale and/or Shape Change Example:

```
Color Chng only :F
Complex Move  :F
Rotation      :F
Scale   :IF (V102.PV.ALM = 0)
           RETURN 1.0;
           ELSE RETURN 1.5;
Color   :IF (V102.PV.ALM = 0)
           RETURN Grey;
           ELSE RETURN Red;
Shape   :Return V102.PV + 1;
Others  :N/A
```

Note: A closed valve would be drawn on layer 1and an open valve on layer 2

Meta Sprites with Linear Movement



Configuration Forms

1. **Object Type:** Meta Sprite with a linear **movement:** The meta sprite comprises a group of multiple graphics objects which change position under the control of the associated meta scripts.
2. **Object Name:** The optional name or ID of the meta sprite object.
3. **Scale:** A meta script that returns a value to be used as a scaling factor for the CAD object. a value less than 1 will shrink the size, greater than 1 will enlarge it.
4. **Color:** A program that determines the color of the object. Returning -1 will keep the metasprite the same color it was drawn in.
5. **Shape:** A meta script that returns the CAD layer number on which the object is defined. In AutoSketch, 1 is the first layer, 2 is the second layer etc.
6. **Value:** A meta script that determines the value to be used in determining the position on the polyline.
7. **SH:** A meta script that returns the high scale value to be used in calculating the position.
8. **SL:** A meta script that returns the low scale value to be used in calculating the position.
9. **X,Y Positions:** The x and y reference position taken from the CAD diagram. Not editable.

Creation Procedure

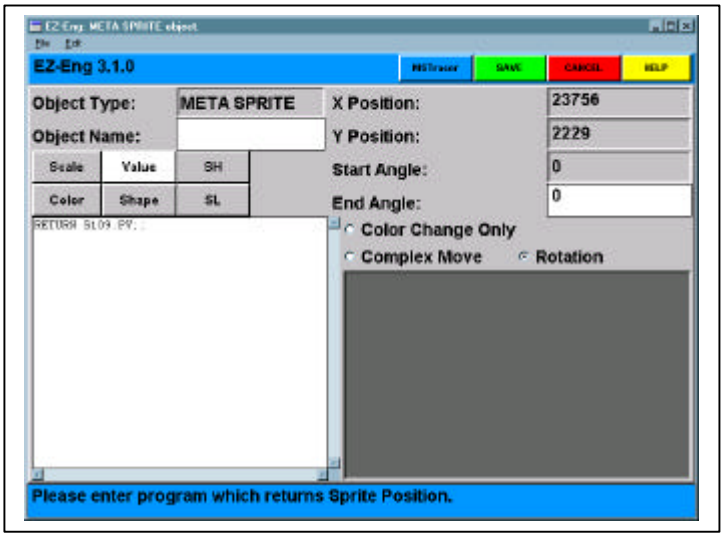
1. Draw the objects that are to move and group them.
2. Draw a line to show the direction and extent of the movement.
3. Group the initial group, ID and polyline.

Linear position Example:

```
Color Chng only   :F (disabled)
Complex Move      :F (disabled)
Rotation  :F (disabled)
Scale     :RETURN 1.0; // No scaling
Color     :RETURN -1; // Use color of object
Shape     :Return 1; // No change
Value     :RETURN TANKLVL.PV
SH        :RETURN TANKLVL.SH;
SL        :RETURN TANKLVL.SL
Others    :N/A
```

Note: A closed valve would be drawn on layer 1 and an open valve on layer 2.

Meta Sprites that Rotate



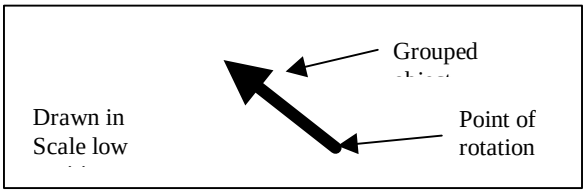
Configuration Forms

1. **Meta Sprites that Rotate:** The meta sprite comprises a group of graphics objects whose angle of rotation changes under the control of an associated meta script.
2. **Object Name:** The optional name or ID of the meta sprite object.
3. **Scale:** A meta script that returns a value to be used as a scaling factor for the CAD object. a value less than 1 will shrink the size, greater than 1 will enlarge it.
4. **Color:** A program that determines the color of the object. (-1 uses the color of the object in the CAD diagram.)
5. **Shape:** A meta script that returns the CAD layer number on which the object is defined. In AutoSketch, 1 is the first layer, 2 is the second layer etc.
6. **Value:** A meta script that determines the position of the meta sprite.
7. **SH:** A meta script that returns the high scale value to be used in calculating the angle.
8. **SL:** A meta script that returns the low scale value to be used in calculating the angle.
9. **X,Y Position:** The x and y reference position taken from the CAD diagram. Not editable.
10. **End Angle:** The angle measured from the scale low position to the scale high position. The angle should be given in tenths of a degree. For example, if the pointer were to end up directly opposite the scale low position, you would enter 1800 for 180 degrees.

Note: If you want the movement to be counter clockwise, enter a negative number here..

Creation Procedure

1. Draw the objects that are to rotate in their scale low position and group them.
2. Choose the meta sprite macro button #I and click on the object at the point of rotation.
3. Group the objects and the ID into a single group.

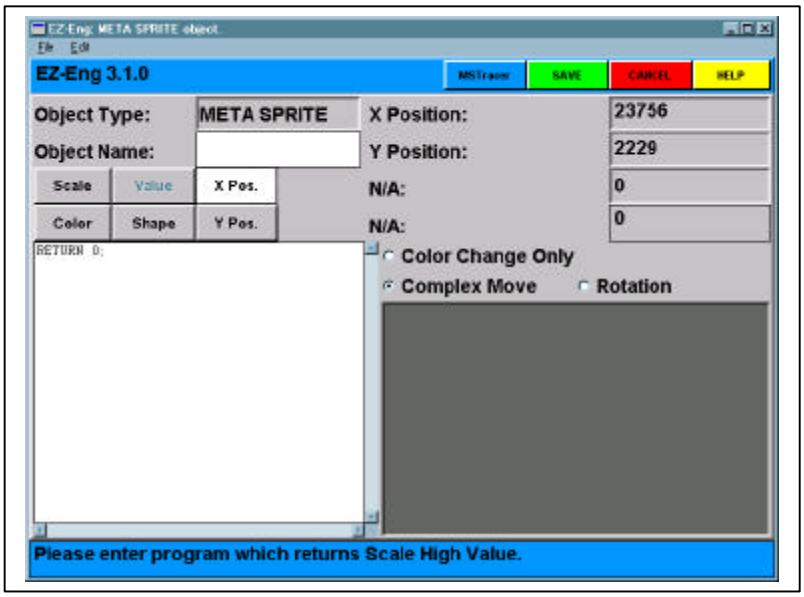


4. Complete the definition of the object in the object editor.

Rotating Meta Sprite Example:

Color Chng only :F
Complex Move :F
Rotation :T
Scale :RETURN 1.0; // No scaling
Color :RETURN 1.0; // Use color of object
Shape :RETURN 1; // No change
Value :RETURN VALVEPID.OUTPUT;
End Angle: 1800
SH :RETURN TANKLVL.SH;
SL :RETURN TANKLVL.SL
Others :N/A

Meta Sprite with Complex Movement



Configuration Forms

12. **Meta Sprite with complex movement:** The meta sprite comprises a group of multiple graphics objects that move over the display area under control of the x and y co-ordinate calculation meta scripts.
13. **Object Name:** The optional name or ID of the meta sprite object.
14. Scale: A meta script that returns a value to be used as a scaling factor for the CAD object. a value less than 1 will shrink the size, greater than 1 will enlarge it.
15. **Color:** A program that determines the color of the object. (Enter a -1 if you want to take the color of the CAD object.)
16. **Shape:** A meta script that returns the CAD layer number on which the object is defined. In AutoSketch, 1 is the first layer, 2 is the second layer etc.
17. **Value:** Not applicable with complex move meta sprites, disabled.
18. **X Position, Y Position:** The meta scripts calculate the X and Y positions in VDC units of the reference point of the meta sprite. The VDC units should fall in the range 0 and 32767. (Please see "VDC Units" on page7.)

Creation Procedure

19. Draw the objects in the sprite. Group the objects in a single group.
20. Add an identifier near the center of the object using the #I icon and group the #I with the first group.

Complex Move Meta Sprite Example:

```
Color Chng only    :F
Complex Move       :T
Rotation           :F
Scale      :RETURN 1.0; // No scaling
Color      :RETURN 1.0; // Use color of object
Shape      :RETURN 1; // No change
Value      :N/A
X Posn     :RETURN 327 * CAR1.XPOS;
Y Posn     :RETURN 327 * CAR1.YPOS;
Others     :N/A
```

Further meta sprite Discussions

The meta sprite is one of the most important and versatile objects available in the graphic generation process. The descriptions below further clarify the use of this object.

meta sprite Features Combinations

Not all combinations of features are valid at any one time. The table below indicates those features that may be used in a single meta sprite. To use the table, look at the type of meta sprite in the left column. If the cell in the associated row has the word YES in it, then you may use that feature. e.g. With a linear movement meta sprite, you may have a scale change, a colour change, and a position change but not a rotation.

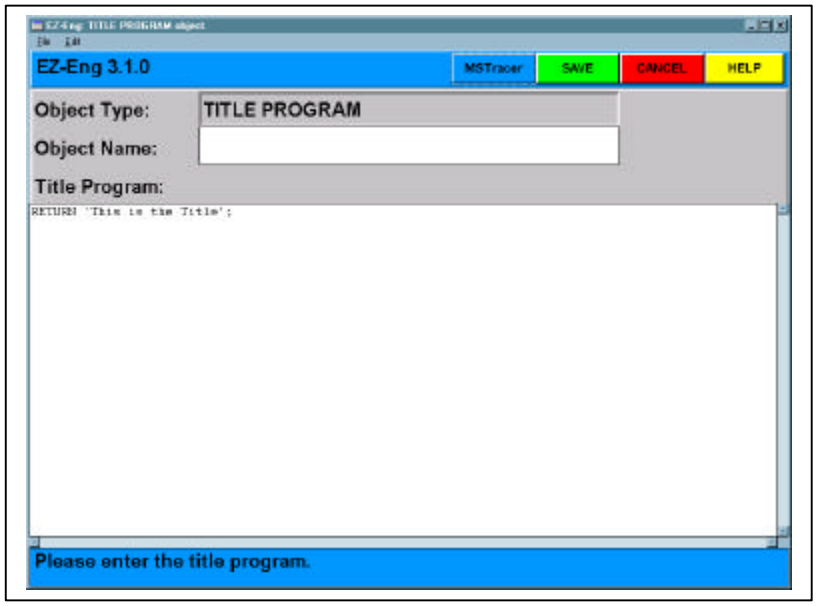
Table 15: meta sprite feature combinations

Type of meta sprite	Scale	Colour	Shape	Rotation	Position
No Movement	Yes	Yes	Yes	No	No
Linear Movement	Yes	Yes	Yes	No	Yes
Rotation	Yes	Yes	Yes	Yes	No
Complex Movement	Yes	Yes	Yes	No	Yes

9.32 Title

This object is used to create the title in the banner of a window.

Title



Configuration form

1. **Object Type:** Menu Bar: The menu bar is used to create a menu in an application. Note: that you may only have one menu bar per operations program (ops3) and you may not add a menu bar to applications running within the Navigator. (Except on pop-ups)
2. **Object Name:** The name of the menu bar object.
3. Item Browse Widget: This is a list of items including pulldowns, items, separators, ends and terminators.
4. **Label:** The name that will appear in the menu bar. Precede the accelerator key with an Ampersand (&).
5. **Action:** The meta script that will be executed if the item in the menu bar is selected.
6. **Disable:** If this meta script evaluates to true, (i.e. 1) then the item in the menu bar will be disabled.
7. **Pulldown:** Inserts a new menu sub-structure below the selected item in the Browse widget.
8. **Separator:** Inserts a new separator below the selected item in the Browse widget.
9. **Item:** Inserts a new item below the selected item in the Browse widget. The item should have a label and optionally, an action and disable meta script.
10. **End:** This marks the end of the pulldown submenu. The End item refers to the closest pulldown item above it in the browse widget.
11. **Terminate:** Inserts a terminate at the bottom of the Browse widget. This marks the end of the menu bar object. There should only be one Terminate item.

Note: After saving a menu bar for the first time, it may be necessary to resize the window after refreshing.