MEDM Reference Manual

Kenneth Evans, Jr.

September 2006

Advanced Photon Source
Argonne National Laboratory
9700 South Cass Avenue, Argonne, Illinois 60439

Contents

Copyright
Overview
New Features
History
Command Line
ADL Files
Site-Specific Customization
Connection Problems and Access Rights
Fonts
Initial Locations for Main Windows
Palettes
MEDM Objects
    Object Index
    Attribute Index
Composite Objects
CALC Expressions
Grid
Macro Substitution
Drag-And-Drop
Dialog Boxes
    Display List
    Message Window
    Statistics Window
    Print Setup
    PV Info
    PV Limits and Precision
Help
Smart Startup
Display Colors
Resizing Displays
Editing
Execute-Mode Popup Menu
Printing
Execute Menu
Color Conventions
Environment Variables
Building MEDM
CDEV Support

Glossary
Technical Support

Overview

MEDM stands for Motif Editor and Display Manager. It is a graphical user interface (GUI) for designing and implementing control screens, called displays, that consist of a collection of graphical objects that display and/or change the values of EPICS process variables. The supported objects include buttons, meters, sliders, text displays/entries, and graphs. It has two modes of operation, EDIT and EXECUTE. Displays are created and edited in EDIT mode, and they are run in EXECUTE mode. MEDM is an EPICS Extension. Much information, including this manual in HTML, Postscript, and PDF, can be found in the Advanced Photon Source EPICS pages for MEDM:

http://www.aps.anl.gov/epics/extensions/medm/index.php.

New Features

The following items indicate significant new features in each release. Sucessive releases also have bug fixes and other enhancements.

Version 3.1

3.1.0

- You can specify macros for a Composite defined by a file.
- There is a new option, &X, for the Execute Menu and Shell Command.
- The Shell Command now has a label, similar to the Related Display.
- A new MEDM object, the WheelSwitch, has been added. The WheelSwitch is a Controller that displays a number value. It has arrow buttons above and below the digits that are used to change individual digits via the mouse or keyboard. Values may also be entered via the keyboard.
- The maximum number of items in the Related Display and Shell Command has been changed from eight to sixteen. If, however, you use more than eight items, the ADL file may not work correctly in previous versions of MEDM. Other values may be specified in siteSpecific.h.
- Count for the Cartesian Plot may come from a process variable.
- There is a Retry Connections item on the Execute-Mode Popup Menu. In cases where MEDM does not see beacons, such as often happens with a PV Gateway, it may not reconnect when a server comes back up. This button can force it to try to reconnect.

Version 3.0

3.0.2

- The Slider now goes left and down as well as right and up.

3.0.1

- Sexagesimal Text Formats as used by astronomers have been added to the Text Monitor and Text Entry.

Version 2.3

2.3.6

- Drag and Drop has been enhanced so the PV Names are also selected into the CLIPBOARD and PRIMARY X selection. This means you can paste them using the usual X cut and paste mechanisms; for example, by clicking Btn 2 in an Xterm.
- The initial positioning of displays and related displays has been fixed in a way that should be consistent and independent of the Window Manager and platform. Depending on the Window manager you use, this may

mean the positions will be different from what they were previously. See the description of the Display and Related Display for more information on how displays are initially positioned.

- The logic for when there is no read access has been fixed. Objects are drawn as black rectangles when one of the associated process variables does not have read access. This is an uncommon situation.
- The Display List may be invoked from the MEDM main window as well as from the Execute-Mode Menu. It now includes the X Window ID of the MEDM main window. This helps in managing displays when there is more than one MEDM running.
- The Byte has been extended to 32 bits and converts its input to an unsigned long integer using the lower 32 bits. This makes it more useful in displaying hexadecimal values.
- MEDM may be built to use the JPT plotting package, developed at the Thomas Jefferson National Accelerator Facility, for the Cartesian Plot. This package has not been extensively tested with MEDM, however. Further, it is known that it does not implement all options of the MEDM Cartesian Plot correctly. See the Cartesian Plot for more details.
- The Cut-Copy-Paste keys (Shift-Del, Ctrl-Ins, and Shift-Ins, respectively) should work, and the position of pasted objects can be adjusted with the arrow keys as well as the mouse. See Menu Editing Operations for more details.
- MEDM no longer tries to connect to process variables for graphics objects and composite objects that have an otherwise unneeded PV associated with them that is there for the sole purpose of making them appear in the correct stacking order. That is, when both the Visibility and Color Mode, if relevant, are static. A side effect is that these objects will not appear white if the PV is invalid. See MEDM Objects for more information.
- There is no longer a requirement that a hidden Related Display be the same size as the associated graphic, or that it be under all graphics, or even that there be a graphic. This makes it easier to use hidden Related Displays with and in Composites.
- The axes in the Strip Chart can now be changed by the user, and defaults may be specified by the screen designer via the Strip Chart Data Dialog. In EXECUTE mode, the dialog box is brought up by the PV Limits item in the Execute Mode Menu.
- MEDM now pops up an existing display if it can, rather than creating a new one. The existing display must have the same name and macros. This applies to displays coming from the Related Display and displays specified with –attach on the command line. The old behavior can be restored in siteSpecific.h.
- Printing and Print Setup have been greatly enhanced and printing is now implemented on WIN32.
- There is an item on the Execute-Mode Popup Menu to mark hidden buttons (see Related Display) by surrounding them with a flashing marquee. This is for use when the user cannot find the buttons. There is also, an item to popup the MEDM main widow in case you cannot find it.
- The Bar Monitor now has options for down and left as well as for up and right. (See the Direction attribute.)
- The Image has been enhanced to read GIF89a files (the latest GIF specification). It now handles multiple-frame GIFs. Multiple-frame GIFs can be set to animate or to display a specific frame based on a CALC expression that can include up to four process variables. This opens a wide range of creative possibilities, including implementing Color Rules. There is also a new EPICS extension called Simple GIF Animator (SGA) that allows you to create and change animated GIFs.
- The Dynamic Attribute used by graphics objects has been extended to have its visibility be based on a CALC expression that can include up to four process variables and more.
- The Composite now has a Dynamic Attribute. This means whole sections of the display can be made to appear, disappear, or change based on the values of process variables. Also, by grouping any single object, it can be made to be Composite and therefore have a Dynamic Attribute. The objects for a Composite may now be read from an ADL file in addition to being defined through grouping.
- There is a siteSpecific.h file, where certain defaults may be defined when MEDM is built.

- The location of the HTML help file used for menu and context-sensitive help in MEDM may be specified in siteSpecific.h and may be overridden by the environment variable MEDM_HELP_URL.
- Colormaps for a display may be specified in the ADL file, a separate file from the ADL file, or by using the default. The separate-file capability was present before, but was broken. The default display colormap may be specified in siteSpecific.h. Different displays may, as before, have different colormaps.
- Input and output for Text Entry and Text Monitor, when they are formatted for octal or hexadecimal, have been made more consistent and less ambiguous. On update, the numbers are displayed with a leading 0 (octal) or 0x (hexadecimal) to indicate their format, but numbers may be entered with or without the leading 0 or 0x and still be interpreted according to the format. As an example, entering 10 will result in 8 decimal if the Text Entry has format octal, in 16 decimal if it has format hexadecimal, and in 10 decimal otherwise. See the Text Entry and the Text Format attribute for more details.
- Unix type builds are deprecated, which means they may still work but should be avoided.

2.3.5

- The limits and the precision for objects such as Meters, Sliders, and Text Updates can be specified at design time and changed at run time, instead of being set by fields such as HOPR, LOPR, and PREC in the associated record.
- The Menu and Choice Button have been fixed so their size is what is specified at design time and what is stored in the ADL file. They are now the same size in both EDIT and EXECUTE mode and have the same look as other MEDM objects. The objects are now smaller than they used to be. The impact is more noticeable for small versions of these objects, and they may need to be resized on existing screens to look attractive.
- There is an alternative, SciPlot, to the commercial package, XRT/Graph, used for the Cartesian Plot. SciPlot does not have all of the features of XRT/Graph, but it does have enough features to use for most plotting needs. It has the advantage of being free and available for platforms where XRT/Graph does not exist. The WIN32 version of MEDM now supports the Cartesian Plot.
- The DESC and RTYP fields of a process variable have been added to the PV Info Dialog Box.
- There is a "Save All" option on the File menu.
- There is a command-line option `(-noMsg)` to prevent the Message Window from popping up.
- The shadows on all items with shadows are now consistent. Highlights have been removed from all objects. Sizes are consistent in EDIT and EXECUTE mode and are what is specified in the ADL file except for minimum size requirements.
- MEDM is supported for CDEV.
- MEDM can be built with either Host or Unix type builds.

2.3.4

- The default startup option is `-local`. There is a new option, `-attach`, to either attach to an existing MEDM if there is an eligible one, or to make the current one eligible for attachment if there is not. You need to use `-attach` to achieve the former default startup behavior. This behavior is primarily applicable for a control room, where it is typically set in scripts. Having the default option be `-attach` was confusing and not appropriate for individual users.
- The overlapping rectangles on the Related Display label do not appear if the label text starts with "-".
- The Meter, Bar Monitor, and Scale Monitor have been fixed to be the size specified. Formerly, they were two pixels wider, this space being used for the border. There is now no border, and the top and bottom shadows are the same as for Motif widgets, rather than being drawn with another, simplified algorithm. The Bar Monitor appears raised. It is not expected these changes for consistency will significantly affect most display screens.
- The Bar Monitor has a no decorations option. This allows making bar graphs.
- MEDM is available for the WIN32 platform.

2.3.3

- Help is now functional. Help from the menus and context sensitive help uses Netscape. If Netscape is not up, it takes a little while for it to come up after the first help request; however, once it is up, new help topics appear quickly inside the existing Netscape. Bubble help has been added to the Object Palette to identify the objects.
- There is a greatly expanded MEDM reference manual in both hardcopy and HTML versions. You are probably looking at one or the other of them now.
- There is now an Undo.
- New editing operations through both the mouse and the keyboard have been added. You can now select objects that are under other objects and can resize with Ctrl-Arrow keys.
- An optional grid has been added for editing. It can be toggled on and off for each display, and the spacing is adjustable. Snap to grid has been implemented.
- "Undo", "Align", "Space Evenly", "Center", "Orient", "Size", "Grid", "Find Outliers", "Refresh", and "Edit Summary" have been added to the Edit menu.
- You can now change the foreground and background colors simultaneously for all grouped elements.
- There is a new PV Info feature.
- There is a new Execute Menu item on the Execute-Mode Popup Menu.
- A number of modifications have been made to the Related Display including executing it without showing the menu if there is only one item, the ability to replace the parent display, the ability to have labels rather than the overlapping squares on the menu button, and the ability to have a hidden button.
- The Cartesian Plot has a time format for the x axis, and there are several new user interactions, such as zoom, in EXECUTE mode.
- Shell commands, as well as commands on the Execute Menu, now accept special characters to include a process variable, include the name of the ADL file, or to prompt for user input.
- The MEDM main window, object palette, resource palette, and color palette now have fixed, default locations. You can change these locations with resources. See Initial Window Locations.
- The way in which MEDM starts up and connects to an existing MEDM has been made more robust. There should be minimal need to use -cleanup.
- You no longer have to respond to the "Do you really want to Exit?" dialog box. MEDM will still prompt you to save any unsaved files, however.
- Much of the coding has been rewritten to be clearer and better organized, and many bugs have been fixed.

History

MEDM was started by Mark Anderson at Argonne in 1990. It was originally partially based on the two programs, DM (Display Manager) and EDD (Edit Display), which had been developed at Los Alamos. These programs used the X library directly. MEDM handles both the construction and running of displays and has a Motif interface. Los Alamos has continued to develop DM and EDD, and today they have similar features to MEDM, but a different look and feel. The basic design and most of the development MEDM was done by Mark. Mark was a strong supporter of Motif in the days before it was as much of a standard as it is today.

Fred Vong took over MEDM in the fall of 1994, starting with Version 1.4.6. He made the program more responsive under heavy load and added a new ADL file format that used less space. Among other things, he added Channel Access security, rewrote the Strip Chart, added features to the Cartesian Plot, and added to the functionality of the Related Display. Fred was rewriting a large part of the coding to make it cleaner and improve the logic when he left Argonne in late 1996.

Ken Evans took over when Fred left starting with Version 2.3.0. He merged Fred's development version with the distributed version, and finished debugging what Fred had started. He wrote this reference manual and is responsible for the new features described above, as well as for substantial reorganization and cleanup of the code.

In addition to these primary developers, parts of MEDM have come from many members of the EPICS community and from many people in the computing community who provide public-domain versions of their routines. Mark Rivers of the University of Chicago did the original port to the WIN32 platform and has provided much help in getting MEDM running on several other platforms. Jie Chen of the Thomas Jefferson National Accelerator Facility provided the initial implementation of SciPlot and all of the CDEV support.

Requirements

MEDM requires X Windows, Motif and EPICS. Motif Version 1.2 or above is supported. MEDM attempts to conform to the Motif Style Guide. Hence standard mnemonics and accelerators are available for interface navigation. Also help is available in standard forms, including context-sensitive help. Netscape is required for most of the MEDM help.

MEDM is developed and tested in an environment using the Solaris operating system and the TED desktop environment as distributed by TriTeal Corporation. TED is a desktop environment conforming to the Common Desktop Environment (CED) standards and uses Motif Version 3. At this time MEDM has no special enhancements to support CDE features apart from Motif or features that are special to Motif Version 3. The main parts of the program build with no compiler warnings even with CMPILR=STRICT on this system. The program should run with versions of Motif 1.2 or higher, but it may need modifications to run on other operating systems. We will try to incorporate changes necessary to support other systems if you let us know.

Command Line

MEDM can be executed from the command line in the general form:

```
medm [options] [display-files] [&]
```

Options are:

```
[X options]
[-help | -h | -?]
[-version]
[-x | -e]
[-local | -cleanup | -attach]
[-cmap]
[-bigMousePointer]
[-noMsg]
[-displayFont font-spec]
[-macro "xxx=aaa,yyy=bbb, ..."]
[-dg [xpos[xypos]][+xoffset[+yoffset]]
[&]
```

Parameters in square brackets [] are optional. Italicized parameters are to be replaced by user-specified terms. The other parameters should be typed literally. The command-line options are described below. The *`display-files`* are a list of one or more ADL files and should be after all the options, except that the `–dg` option can appear within the *`display-files`*. The "&" causes the process to be run in the background on UNIX, usually a good idea for a graphical program.

Command-Line Options

X options	The usual X command-line arguments, such as –display and –geometry.
-help or -?	Print usage.
-version	Print the version.
-x	Startup MEDM in EXECUTE mode.
-e	Startup MEDM in EDIT mode. (This is the default.)
-local	Do not participate in remote display protocol: Do not transfer operation to an already-running MEDM, and do not allow transfer of operation from any other MEDM. (This is the default.)
-attach	Do participate in remote display protocol: Transfer operation to an existing MEDM if there is one which is accepting transfers, or have this one assume responsibility for accepting transfers if there is not one.) This can lead to more efficient operation and is appropriate if many MEDM screens are running on one display, such as in a control room. You must give the first MEDM time to establish itself before attempting to attach to it. See Smart Startup.
-cleanup	Participate in remote display protocol, but ignore any existing MEDM and take over responsibility for accepting remote transfers.
-cmap	Use a private X colormap to circumvent the problem of not having enough colors available in the colormap. (This option may cause colormap flashing.)
-bigMousePointer	Use big mouse cursor.
-noMsg	Set the Message Window to not pop up when messages are received. (It can be brought up via the View menu on the MEDM main window if you need to see it.)
-displayFont *font-spec*	Select aliased or scalable fonts. (See Fonts below.) The font-spec parameter can be “alias”, “scalable”, or an X font specification. When attaching, this argument should be the same for all participating MEDMs.
-macro *name=value*	Apply macro substitution to replace occurrences of $(*name*) with *value* in all command-line specified display list files.
-dg *geometry*	Specify the geometry (location and size) of the display window. The geometry syntax is similar to the X Windows -geometry option. (Note that -geometry specifies the location of the MEDM main window, and -dg specifies the location of the display windows.) The -dg option affects the ADL filenames following it.

Examples

Start up in default EDIT mode:

```
medm &
```

Start up executing two displays:

```
medm -x abc.adl def.adl &
```

Start up in EDIT mode with a new, local executable:

```
medm –local &
```

Start up using default scalable fonts:

```
medm -displayFont scalable &
```

Start up in execute mode, performing macro substitution of all occurrences of $(a) and $(c) in the display file, xyz.adl:

```
medm -x -macro "a=b,c=d" xyz.adl &
```

Start up in execute mode; resize the display window corresponding to abc.adl to 100 by 100 pixels and move it to the location, x = 100 and y = 100; and move the display window corresponding to def.adl to x = 400 and y = 150:

```
medm -x –dg 100x100+100+100 abc.adl –dg +400+150def.adl &
```

ADL Files

The files that contain the information for each display are called ASCII Display List (ADL) files and should have an extension of .adl. These files are created or modified when you save a display that you have edited. The files are readable and can also be edited by hand if you understand the file format. The file format is not described here, but it should not be difficult to make many types of changes without knowing it precisely. MEDM should be able to read files that were written in earlier formats, and files can be saved with the file format in use for Versions 2.1.x as well as the default format, which was initiated with Version 2.2.0.

The following logic is used to open ADL files from the command line and from Related-Display execution:

1. Try to open the file with the name as given. If the name is not an absolute pathname, then this means to try to open the file relative to the current working directory (the directory where MEDM was started).
2. If the file is being opened as a result of a related-display button, then try to open the file with the name prepended with the pathname of the display that contained the button.
3. Try to open the file with the name prepended by each pathname in the EPICS_DISPLAY_PATH.

Site-Specific Customization

Certain defaults, such as fonts and colors, may be specified in the siteSpecific.h file when MEDM is built. This provides an alternative to using the defaults in use at the Advanced Photon Source. Some of these defaults were decided a long time ago and cannot be changed because it would break too many existing screens. Often there are better alternatives today. In particular, the use of scalable fonts is highly recommended. See the siteSpecific.h file in the MEDM distribution for more information.

Connection Problems and Access Rights

If an MEDM object cannot connect to its associated process variables, that object is represented by a white rectangle, or in the cases of the Rectangle, Oval, Arc, Text, Polygon, and Polyline drawing objects, by coloring the object white. The objects are white whether the connection was never made or whether it was made and subsequently lost. They should revert to their normal state whenever the connection is valid. In some cases PV Info will give useful information, even if the connection is lost, and the objects are white.

Similarly, if there is are connections to the process variables associated with an object but one of them does not have read access, the object is represented by a black rectangle. The drawing objects are also represented by black rectangles, unlike the case for no connection where the object changes color. The reason is that drawing objects are too likely to be black normally. If the read access changes, the black rectangles appear or disappear as appropriate. Note that black rectangles imply there is a valid connection, just not read access. In other words, white overrides black. It should be kept in mind that the Image will be also black in EXECUTE mode if the Image Calc expression is invalid. It should not be hard to determine which is the cause of a black rectangle for the Image, however. Also note that non-updating graphics objects, even though they have a process variable associated with them, do not connect and hence do not turn black if the process variable does not have read access. Denial of read access is uncommon, so you may never see these black rectangles.

If the process variables associated with an object do not have write access, the cursor turns into a circle with a diagonal line through it when it is over a controller. You will not be able to change the value in this case.

PV Info is useful in troubleshooting any of these cases.

Fonts

MEDM uses several methods for font specification, invoked on the command line by the option:

`-displayFont` *`font-spec`*

where *font-spec* is one of {"alias", "scalable", or an X font specification, such as -bitstream-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1}. The default is -displayFont alias.

Aliased Fixed Fonts

For default, fixed fonts, MEDM uses font aliases internally. This makes it possible to deal with server font variations. The aliased fonts are not in the default X database. To make these fonts available you need to do the following:

Aliased Fonts in X Windows

Copy the following lines into your miscellaneous fonts.alias file. The standard place is /usr/lib/X11/fonts/misc/fonts.alias. You can get your own particular font path from "xset q". Note that these specifications are the standard MEDM font specifications as used at the Advanced Photon Source, but you are free to define another set of font specifications that gives better results for your machine or your own preferences. Keep in mind that displays may not look good if the fonts used are significantly different from those used by the designer of the display.

```
widgetDM_4   -misc-fixed-medium-r-normal--8-60-100-100-c-50-iso8859-1
widgetDM_6   -misc-fixed-medium-r-normal--8-60-100-100-c-50-iso8859-1
widgetDM_8   -misc-fixed-medium-r-normal--9-80-100-100-c-60-iso8859-1
widgetDM_10  -misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
widgetDM_12  -misc-fixed-medium-r-normal--13-100-100-100-c-70-iso8859-1
widgetDM_14  -misc-fixed-medium-r-normal--14-110-100-100-c-70-iso8859-1
widgetDM_16  -misc-fixed-medium-r-normal--15-120-100-100-c-90-iso8859-1
widgetDM_18  -sony-fixed-medium-r-normal--16-120-100-100-c-80-iso8859-1
widgetDM_20  -misc-fixed-medium-r-normal--20-140-100-100-c-100-iso8859-1
widgetDM_22  -sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1
widgetDM_24  -sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1
widgetDM_30  -adobe-times-medium-r-normal--25-180-100-100-p-125-iso8859-1
widgetDM_36  -adobe-helvetica-medium-r-normal--34-240-100-100-p-176-iso8859-1
widgetDM_40  -adobe-helvetica-bold-r-normal--34-240-100-100-p-182-iso8859-1
widgetDM_48  -adobe-helvetica-bold-r-normal--34-240-100-100-p-182-iso8859-1
widgetDM_60  -adobe-helvetica-bold-r-normal--34-240-100-100-p-182-iso8859-1
```

Then run "xset fp rehash" to install them. On Linux you may need to run "`/etc/init.d/xfs restart`" to restart the X font server. You can see if the fonts are there with "xlsfonts" or "xlsfonts | grep widgetDM". You can get more information from any of the general-purpose books on X Windows.

Aliased Fonts in Exceed (An X Server for Microsoft Windows)

There are two main ways to install the MEDM fonts in Exceed. For both ways you go to the Xconfig utility for Exceed and then go to Font Management. How you preceed from there may depend on your version of Exceed. The following directions are for Exceed 8 and should work at least through Exceed 10. In the Font Management dialog choose Edit.

1. Adding a font server is the easiest approach, but only works if you have a font server available. Choose the "Add Font Server…" button, and configure the server by filling in the dialog box. See your system manager for the parameters, which may be something like tcp/helium:7100, which is the suggested server and port for the Advanced Photon Source. The font server must be available through your network, and the MEDM font aliases must have been installed on it, probably as above for X Windows.

2. You can install the font aliases locally in Exceed. Choose the "Import Aliases…" button. You will need a text file, perhaps called medm.ali, with the above aliases in it. Type in the location of this file for the "From:" entry, and then be sure to select "(All Directories)" for the "To:" entry. (It is not the default.)

There are more complete directions including pictures on the EPICS page for MEDM:

http://www.aps.anl.gov/asd/controls/epics/EpicsDocumentation/ExtensionsManuals/MEDM/MEDMFonts.htm,

and you can download `medm.ali` from there if you want to use the Advanced Photon Source font choices.

Aliased Fonts for other Platforms and X Servers

You will have to determine how to install font aliases for your particular system. You can use the directions above as a guideline.

Default Scalable Fonts

The user can invoke MEDM with the -displayFont scalable option. MEDM then uses the default Speedo outline font (bitstream) supplied by the X11R5 font server. Users should add a font server to their font path via:

```
% xset +fp tcp/:
```

for example:

```
% xset +fp tcp/phebos:7100
% medm -displayFont scalable
```

You can get the name of the font server by running "fsinfo" on the machine that has the server.

User-Specified Scalable Fonts

The user can invoke MEDM with the -displayFont option. MEDM then uses the specified font supplied by the X11R5 fontserver. This font should be an XLFD name (all 14 hyphens, other fields can be wildcarded though) and scalable (i.e., point and pixel size = 0). Users should add a font server to their font path as above.

For example:

```
% medm –displayFont -bitstream-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
```

Initial MEDM Window Locations

The MEDM main window, the object palette, the resource palette, and the color palette are located at the right edge of the screen by default. The spacing between them depends on the window manager title bar and may not be correct for all window managers. They can be changed using the standard format for X Windows geometry by the following resources (shown with typical values):

```
Medm.mainMW.geometry: -5+5
Medm.objectS.geometry: -5+137
Medm.resourceS.geometry: -5+304
Medm.colorS.geometry: -5-5
```

You could put these lines in your .Xdefaults file, for example.

Palettes

Object Palette

Resource Palette

Color Palette

MEDM Objects

The objects that can be used in MEDM fall into four categories, Graphics, Monitors, Controllers, and Special. Graphics are items such as text, lines, and images. Monitors are objects that monitor the state or values of

process variables. Controllers are objects that change the values of process variables. There is only one miscellaneous object, the Select Arrow, which is not really an object but an editing mode.

Some objects are widgets. This means they have their own X window, which is a child of the drawing area of the display. The Graphics objects, such as the Rectangle, Oval, Image, *etc.* are drawn on the drawing area. Every object with a widget lies above the drawing area and hence above all drawn objects. Further, all graphics objects that are updating because their color or visibility depends on a process variable or because they are an animated Image will appear above non-updating graphics objects.

If you want a non-updating graphic object to be properly stacked among updating objects, enter something (for example, "Dummy") for Channel A but leave its Visibility Mode and Color Mode (if the object has one) as Static. It does not matter what is entered for Channel A. MEDM will not search for the process variable, but will treat the object as updating so that it will be drawn in the proper stacking order when other updating objects change.

In EDIT mode, all of these objects are represented on the Object Palette, and this palette can be used to create new instances of the objects on the display.

Object Index

Graphics	Monitors	Controllers	Special
Arc	Bar Monitor	Choice Button	Composite
Image	Byte Monitor	Menu	Display
Line	Cartesian Plot	Message Button	
Oval	Meter	Related Display	
Polygon	Scale Monitor	Shell Command	
Polyline	Strip Chart	Slider	
Rectangle	Text Monitor	Text Entry	
Text		Wheel Switch	

Attribute Index

Attributes are sets of properties. These sets of properties may be common to several, or even most, objects. Simple attributes have one property that can take on one of several values. Compound attributes have more than one property. To prevent repetition, the sets of attributes are described here, rather than with the objects that possesses them. The compound attributes are listed first, then the simple objects in alphabetical order.

Compound	Simple		
Object	Cartesian Plot Axis Style	Edge Style	Related Display Mode
Basic	Cartesian Plot Range Style	Erase Mode	Related Display Visual
Dynamic	Cartesian Plot Style	Fill Mode	Stacking
Monitor	Cartesian Plot Time Format	Fill Style	Text Align
Control	Color Mode	Image Type	Text Format
Limits	Color Rule	Image Calc	Time Units
Plot	Direction	Label	Visibility Mode
Plot Axis		Plot Mode	Visibility Calc

Objects

Arc

The Arc has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground, Style, Fill, Line Width), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. In addition, there is a Begin Angle and a Path Angle. The Arc is drawn on the display drawing area and is not a widget. The Begin Angle is the start of the arc in degrees. The Path Angle is the length of the arc in degrees. (Internally the angles are specified in integer 1/64-degree units.)

Bar Monitor

The Bar Monitor has Object (X Position, Y Position, Width, Height), Monitor (Readback Channel, Foreground, Background), Limits (Low Limit, High Limit, Precision), Label, Direction, Fill Mode, and Color Mode attributes.

The Bar Monitor is designed to display values as a bar that expands or contracts as the value changes. The Bar Monitor can have several decorations as specified by the Label, can go up, down, left, or right, as specified by the Direction attribute, and can start from the edge or from the center, as specified by the Fill Mode attribute. The Bar Monitor with the Label set to "no decorations" can be used to make bar graphs.

On some platforms the Bar Monitor may flash as the value changes. There is an option in siteSpecific.h that will fix this.

Byte Monitor

The Byte Monitor has Object (X Position, Y Position, Width, Height), Monitor (Readback Channel, Foreground, Background), Direction, and Color Mode attributes. It also has a Start Bit and End Bit.

The Byte Monitor is designed to monitor mbbiDirect and mbboDirect records. It was developed at the Thomas Jefferson National Accelerator Facility (formerly CEBAF) by David M. Wetherholt. It can also be used to display values in binary. It masks off the lower 32 bits and treats them as an unsigned long integer.

The Start Bit and End Bit are integers in the range 0-31, specifying the starting and ending bits to be displayed. The End Bit can be smaller than the Start Bit. The foreground color indicates the bit is set, and the background color indicates it is not set. Only two directions are supported, right (horizontal with Start Bit at the left and End Bit at the right) and down (vertical with Start Bit at the top and End Bit at the bottom). You can obtain the other two directions by reversing the Start Bit and End Bit. The Byte looks better if the width is a multiple of the number of bits displayed plus one.

Cartesian Plot

The Cartesian Plot has Object (X Position, Y Position, Width, Height), Plot (Title, X Label, Y Label, Foreground, Background), Plot Style, Plot Mode, and Erase Mode attributes. In addition, there is a Count, X/Y/Trace Data, Axis Data, a Trigger Channel, and an Erase Channel, which are described in this section.

The logic of the Cartesian Plot is separate from the plotting package that implements it. At present MEDM supports three implementations.

XRT/Graph

The original and primary implementation is by the commercial plot package, XRT/graph by the KL Group. MEDM supports both Versions 2.4 and 3.0 of XRT/Graph. The implementation of these two versions is significantly different, but the plots look the same. The XRT/graph package must be purchased. To build MEDM with XRT/Graph, the makefile variable XRTGRAPH must be set to the path for XRT/Graph. After the usual compile and link, the executable must be patched to enable the Cartesian Plot. To do this, the proper, licensed XRT patch files must be available. If these files exist on the computer on which MEDM is built, the MEDM Makefile will do the patch as part of the build. The patch can also be done later on a computer with the proper files. XRT/Graph is a very flexible and full-featured plotting package. One can assume it has been tuned for performance. It arguably produces the best implementation of the Cartesian Plot.

The following user interactions are available in EXECUTE mode for the Cartesian Plot when built with XRT/graph 3.0 and when XRT_EXTENSIONS is defined in the Makefile:

Ctrl-Btn1	Graphics zoom. Drag to select the area of the graph and release to zoom. This action blows up the graph to only show the area in the rubberband rectangle.
Shift-Btn1	Axis zoom. Drag to select the range of the axes and release to zoom. This action changes the ranges of the axes and repositions the axes so they are visible.
Ctrl-Btn2	Scaling. Move up or down to increase or decrease the size.
Shift-Btn2	Translation. Move to shift the graph.
r	Press r to reset the graph from changes made with Btn1 and Btn2.
Btn3	Bring up the Cartesian Plot Axis Data dialog box to change axis parameters.
Ctrl-Btn3	Bring up the XRT Property Editor dialog box. The property editor allows you to change most of the graph properties. The editor is complex, but the operation is intuitive. Select the upper tab that corresponds to the element of the graph you want to edit. If there is a tree control on the left, select the specific object to edit, then find and edit the property on the lower tabs. Keep in mind that it is possible to confuse MEDM by changing the properties this way, since the changes are internal and not necessarily known to MEDM. Consequently, the use of this feature is not supported. It can be useful if used sensibly, however. The graph can be restored by closing and reopening the display. For more information about the property editor and the various graph properties, see *the XRT/graph Programmer's Guide and Reference Manual, Version 3.0*.

SciPlot

MEDM also supports SciPlot, a freely distributable package originally written by Rob McMullen at the University of Texas. Its web page no longer exists. The version used in MEDM has been significantly modified to add features and fix bugs, but SciPlot is essentially the work of Rob McMullen. There is a license, which is included in the files. At present, SciPlot does not support a Y axis, fill, or time labels, as well as some less-important features. However, it has enough features to be very useful, and it does some things better than XRT/Graph. It has been used extensively and is tested. It can be used where XRT/Graph is not available, for instance on the WIN32 platform. To build MEDM with SciPlot, define the makefile variable SCIPLOT, preferably setting it to YES.

JPT

JPT, the Jefferson Plotting Toolkit, developed at the Thomas Jefferson National Accelerator Facility by Ge Lei, may also be used with the MEDM Cartesian Plot. This package has a second Y axis, fill, and time labels, unlike SciPlot. However, at this time it has not been extensively tested with MEDM, and it is known that it does not work correctly in several cases, in particular when the X variable is a vector. There are problems with the second Y axis and the time labels, as well. Be sure that it works for what you want to do before using it. To build MEDM with JPT, define the makefile variable JPT, preferably setting it to YES.

If none of the makefile variables XRTGRAPH, SCIPLOT, or JPT is defined, MEDM can still be built but will ignore any Cartesian Plots. Any Cartesian Plots in an existing ADL file will be lost if the file is modified and saved.

The Cartesian Plot is the most complicated MEDM object. It has many options and can be confusing. The descriptions presented here are intended as a reference, not as a tutorial. A good approach is to change the various options and see what happens. It is usually a good design choice to keep the plot as simple as possible.

An MEDM Cartesian Plot consists of an X and one or two Y axes on which data can be plotted. The sets of data are called traces and consist of a set of (x, y) pairs and a color. The traces correspond to curves on the plot. Currently there can be up to eight traces on a plot. The first trace is plotted on the Y1 axis, and the remaining traces (if any) are all plotted on the Y2 axis. The X and Y1 axes may have a label, but the Y2 axis does not. Each trace can (but does not have to) have a process variable from which to get the x values and another from which to get the

y values. These process variables can be array process variables, such as Waveforms, or they can be scalar process variables with only a single value.

There are eight possible kinds of traces as seen in the following table. The traces for a given plot do not all have to be of the same kind – they can be mixed. (In the table Nx is the number of elements in the process variable specified for x, and Ny is the number of elements in the process variable specified for y. The letter n denotes a number greater than one, and a blank indicates no process variable is specified. The axis limits LOPR and HOPR denote the limits obtained from Channel Access for the process variable. Typically, these are the fields LOPR and HOPR in the associated record. Count is the specified Count for the Cartesian Plot, which is described in more detail below.)

Kinds of XY Traces								
Nx	**Ny**	**Type**	**Points**	**Xmin**	**Xmax**	**Ymin**	**Ymax**	**NPoints**
n	n	X,Y Vector	x(i), y(i)	LOPR	HOPR	LOPR	HOPR	Min(Nx, Ny)
n	1	X Vector, Y Scalar	x(i), y	LOPR	HOPR	LOPR	HOPR	Nx
1	n	Y Vector, X Scalar	x, y(i)	LOPR	HOPR	LOPR	HOPR	Ny
n		X Vector	x(i), i	LOPR	HOPR	0	Count - 1	Nx
	n	Y Vector	i, y(i)	0	Count - 1	LOPR	HOPR	Ny
1	1	X,Y Scalar	x(i), y(i)	LOPR	HOPR	LOPR	HOPR	Count
1		X Scalar	x(i), i	LOPR	HOPR	0	Count - 1	Count
	1	Y Scalar	i, y(i)	0	Count - 1	LOPR	HOPR	Count

The specified Count for the Cartesian Plot may be an integer or the name of a process variable. If the value starts with a non-digit, then it is considered to be a process variable name, and the value of Count will come from the process variable. If the process variable is not found or its value is less than 1, Count will be assumed to be zero.

If one of the process variables is an array with more than one element and if Count is a number, then the specified Count is ignored and the value shown in the last column of the table is used. Under the same circumstances, if Count is a name, then it is used only if it is greater than 0 and less than what would be used if it were a number. That is, Count from a process variable can only restrict NPoints to a lower number than it would be otherwise. The points are plotted from i = 0 to NPoints - 1 and update as the values change. In the cases where one of the process variables is not specified, the history is plotted on that axis against values from 0 to NPoints – 1 on the other axis.

In the remaining cases, where neither pocess variable is an array with more than one element, Count corresponds to a history of the process variable. (These are the cases where NPoints is shown as Count in the table.) Each time the process variable changes (or when either one changes, in the case of X, Y Scalar) a new point is plotted until there are Count points. The points are plotted from i = 0 to the lesser of Count -1 and the number of updates. When the Plot Mode is “plot n pts & stop,” no more points are plotted. When the Plot Mode is “plot last n pts,” the earliest point is discarded, the others are moved down, and the latest is plotted at the end. In the cases where one of the process variables is not specified, the history is plotted on that axis against values from 0 to Count – 1 on the other axis.

The X/Y/Trace Data are entered via the Cartesian Plot Data dialog box with entries for the process variables associated with x and y and an entry for the color. The color is picked by clicking on the color entry and using the color palette. The process variable names can both be left blank on a given row, in which case the row is ignored. The first trace (the one that uses the Y1 axis) is the first valid one, and does not have to be in the first row. The data entry is different from that for entry boxes in the resource palette. Typing is not lost, for example, when

the entry box loses focus, and it is not necessary (or useful) to type a carriage return to retain the data. The data are retained only when the apply button is pressed.

The Axis Data are entered via the Cartesian Plot Axis Data dialog box. This same dialog box is available in EXECUTE mode by clicking Btn 3 on the Cartesian Plot. The dialog box consists of an area for each of the three axes (X, Y1, and Y2). For each axis there is a Plot Axis (Axis Style, Axis Range, Minimum Value, Maximum Value, Time Format) attribute. The maximum and minimum values are only relevant if the Axis Range is "user-specified", and the time format is only relevant for the X axis when the Axis Style is "time". The values for the maximum and minimum values are only retained if a carriage return is typed in the box, although typing is not lost when the entry box loses focus.

In order to use the time format meaningfully, the process variable data must be in suitable time units. The basic time unit is a second. The EPICS time base, corresponding to zero seconds, is 00:00:00 January 1, 1990, Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT). This is the time base used in EPICS timestamps, and it is *assumed* to be the time base for any process variable data. Note that it is not the same as local time. The time base in MEDM is set to 00:00:00 January 1, 1990, local time. By setting the time base as local time in MEDM, the effect is to make the EPICS UTC time numbers appear as local time, rather than the less intuitive UTC. For times less than one day and for time formats that do not include the day or year, the data could just as well be seconds since midnight UTC, since the plotted points on this type of axis would be indistinguishable from those plotted with the actual base.

It should be noted that there is an error introduced in converting the number of seconds into a floating point number for plotting. This error corresponds to about 2 minutes per year, amounting to 20 min. by the year 2000.

The Trigger Channel is a process variable that causes the entire plot to be updated. If there is a trigger channel, the plot is updated whenever the value of that process variable changes. Otherwise, each individual trace is updated whenever any of the data for that trace changes.

The Erase Channel is a process variable that causes erasing of the plot. If there is an Erase Channel, the plot erases when the process variable turns either zero or non-zero, depending on the Erase Mode. The Erase Mode is only relevant if there is an erase channel.

Choice Button

The Choice Button has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), Color Mode, and Stacking attributes. The Choice Button is used for ENUM process variables and is a means for picking the ENUM choices via buttons.

No more than 16 ENUM strings can be obtained from Channel Access. It is possible for an ENUM process variable to have more than 16 states, however. The .STAT field that exists for most process variables is an example. MEDM will print an error message if a Choice Button receives a value for an ENUM process variable that is beyond the range known by MEDM, even though that value may be valid otherwise. This is a limit of Channel Access, not MEDM.

Display

The Display is a special object. It has Object (X Position, Y Position, Width, Height) and Grid attributes. In addition there are Foreground and Background colors, and a Colormap, which is the name of a display colormap file. See Grid for a description of the parameters in the Grid attribute, and see Display Colors for a description of display colormaps and colormap files. The Background color is the color of the display background, and the Foreground color is the color of the grid.

The X and Y positions are saved in the ADL file. The display should appear at the coordinates specified in the ADL file when the display is first brought up. These coordinates are the coordinates of the area inside the title bar and borders. The title bar and borders are managed by the Window manager and may vary in size from system to system. The coordinates are updated in the Display object only when the Display is selected in the resource palette by clicking on the display background or using the Select Display item in the Edit Menu in EDIT mode. They are not updated when you drag the display to new positions. If you want to have the display come up at a specified position, be sure the X and Y positions are what you want when the file is saved. The screen should come up in the position it was in when saved if you click on the display object just before saving.

Image

The Image has Object (X Position, Y Position, Width, Height) and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. Color Mode in the Dynamic Attribute is not relevant and is not used. There are also Image Type, Image Name, and Image Calc attributes. The Image is drawn on the display drawing area and is not a widget.

MEDM only supports Graphics Interchange Format (GIF) image files, both the older GIF87a and the newer GIF89a formats. The GIF image files may include multiple frames, transparency, and most other features supported by the GIF format except local color maps. Transparency only substitutes the display background color for the transparent color. The image will be transparent when on the background color, but objects stacked under the image do not show through. The Image Name attribute is the name of the GIF file. When an Image is inserted in EDIT mode, a file selection box prompts the user for the image file to be incorporated.

MEDM will look for this file with the specified name, then with the path of the display prepended, then with each of the paths in the EPICS_DISPLAY_PATH prepended.

If the GIF image file has multiple frames, the designer can choose to animate the images or to display a specified frame. The default is to animate if there is more than one frame. To display a specific frame, enter an expression for the Image Calc that returns a frame number. Frame numbers start with 0. The syntax for the Image Calc expression is the same as that for the Dynamic Attribute, and it uses channels A-D in the Dynamic Attribute. The only difference is that the CALC expression should return a frame number, not True or False. The value will be rounded to the nearest integer. Frame numbers that are too high will use the last frame, and frame numbers that are too small will use the first frame. The Image will be black in EXECUTE mode if the Image Calc expression is invalid.

The visibility of the Image is determined in the usual way by the Dynamic Attribute if there is a channel specified. The visibility is independent of whether the image is animating or not.

A single-frame Image with no process variables specified for the Dynamic Attribute is treated the same as other Graphics Objects. It is drawn on the display background and will be under any objects that are updating. A multiple-frame Image even with no process variables specified for the Dynamic Attribute does update and so will appear among the updating objects in the stacking order and hence over non-updating objects. This is a concern only for overlapping objects. See the introduction to MEDM Objects for more information on stacking order.

A GIF image may contain up to 256 colors. On an 8-bit X display there are only 256 available colors in the colormap, some of which are almost certainly being used. Consequently, MEDM has an algorithm to reduce the number of colors in the image if the X colormap is full. This algorithm makes the image appear less bright. Images may also appear to have different brightness at different times, depending on the state of the X colormap when the MEDM display is loaded. There should be no problem for a 24-bit X display. The problem will be reduced on an 8-bit display if MEDM has its own colormap. See the command-line options for how to specify a private colormap for MEDM.

Color Rules may be implemented using the Image. To do this, place your object over an Image that is large enough to provide a border around the object. You can use a GIF image file that has one frame for each color you want to display. These frames only need to be one pixel wide and high in the GIF file, saving storage space and loading time. MEDM will resize them as it does any image. Use the Image Calc expression and the process variables in the Dynamic Attribute to specify when to display each color.

There is an EPICS extension called Simple GIF Animator (SGA) that allows you to create and modify animated GIFs for use in MEDM. There are also many other GIF animators available, especially for WIN32. It does not make any difference which platform you use to create or manipulate the images. You cannot use SGA or most of the animators to create the images themselves, only to add, delete, and rearrange them plus modify some of the GIF parameters. There are many drawing and image editing programs that allow you to create and edit images. For UNIX, the GNU Image Manipulation Program (GIMP) is a good choice and is free.

Line

A Line is a Polyline with just two points. See Polyline.

Meter

The Meter has Object (X Position, Y Position, Width, Height), Monitor (Readback Channel, Foreground, Background), Limits (Low Limit, High Limit, Precision), Label, and Color Mode attributes. It displays the value of the process variable on a meter with a dial.

Menu

The Menu has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), and Color Mode attributes. The Menu is used for ENUM process variables and is a means for picking the ENUM choices via a menu. Note that if the Color Mode is alarm, the foreground color, not the background color, is set to the alarm colors. The background color should be choosen to contrast with all the alarm colors.

No more than 16 ENUM strings can be obtained from Channel Access. It is possible for an ENUM process variable to have more than 16 states, however. The .STAT field that exists for most process variables is an example. MEDM will print an error message if a Menu receives a value for an ENUM process variable that is beyond the range known by MEDM, even though that value may be valid otherwise. This is a limit of Channel Access, not MEDM.

Message Button

The Message Button has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), and Color Mode, attributes. In addition, there are a Message Label, a Press Message, and a Release Message. The Message Label is the label on the Message Button. The Press Message and the Release Message are the values to which to set the process variable when the button is pressed or released, respectively. These values should be commensurate with the type of the process variable. (It is not wise to send a non-numeric string to a DOUBLE process variable, for example.) The label and message strings can be up to 255 characters.

Oval

The Oval has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground, Style, Fill, Line Width), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. The Oval is drawn on the display drawing area and is not a widget.

Polygon

The Polygon has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground, Style, Fill, Line Width), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. The Polygon is drawn on the display drawing area and is not a widget.

Polygons are drawn by pressing Btn1 to generate vertices or Shift-Btn1 to generate vertices which are multiples of 45 degrees from the previous point. This allows easy horizontal or vertical line generation, for instance. Button releases are ignored. The Polygon is finished when there is a second click at the same location. The Polygon is always displayed as closed while it is being drawn.

Polygons can be edited by selecting them, then pressing Btn1 on a vertex and dragging the vertex.

The Object Attributes for a Polygon represent the bounding box of the polygon. Accordingly, editing the vertices or changing the Line Width for the Polygon may change the object size and location.

Note that Polygons in outline fill mode are similar to Polylines, except that Polygons close the figure and Polylines do not.

Polyline

The Polyline has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground, Style, Line Width), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. The Polyline is drawn on the display drawing area and is not a widget.

Polylines are drawn by pressing Btn1 to generate vertices or Shift-Btn1 to generate vertices which are multiples of 45 degrees from the previous point. This allows easy horizontal or vertical line generation, for instance. Button releases are ignored. The Polyline is finished when there is a second click at the same location.

Polylines can be edited by selecting them, then pressing Btn1 on a vertex and dragging the vertex.

The Object Attributes for a Polyline represent the bounding box of the lines. Accordingly, editing the vertices or changing the Line Width for the Polyline may change the object size and location.

Rectangle

The Rectangle has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground, Style, Fill, Line Width), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. The Rectangle is drawn on the display drawing area and is not a widget.

Related Display

The Related Display has Object (X Position, Y Position, Width, Height) and Visual attributes. In addition, there are Foreground, Background, Label, and Label/Name/Args attributes. The Foreground and Background colors are changed by clicking to bring up the Color Palette. The Label is the optional label that appears on the button. Normally, whether there is a label or not, there are two overlapping squares on the button to denote a Related Display control. However, if the label text begins with "-" (for example "-My Label"), the overlapping squares will not appear. The Display Label, Display Files, Arguments, and Policy are specified for each related display in the Label/Name/Args dialog box. There currently can be up to sixteen related displays for each button, and this can be changed in siteSpecific.h.

The Related Display provides a means of bringing up more displays. It is a menu button that usually has a graphic consisting of two overlapping squares and a label. See the paragraph above for other options for the graphic and label. The menu items denote the related displays that can be displayed. If there is only one related display specified in the menu, then the two overlapping squares and/or label appear in the middle of the button, and the related display is activated immediately without bringing up a menu. It is not activated until the button is released, so you can depress the button to check if there is more than one menu item or not, then abort by releasing the button away from the Related Display. If there is more than one item on the menu, the squares and/or label are at the left of the button. If Ctrl-Btn1 is used in place of Btn1 to select the new display, then the parent display goes away and is replaced by the new display. The new display can also be configured in EDIT mode to always replace the parent.

In the EDIT-mode dialog box for specifying the menu items there are four columns. The first column, the Display Label, is the label on the menu item in the menu that is brought up when the Related Display button is pressed. The second column, the Display File, is the name of the ADL file that will be displayed when the menu item with that label is selected. The third column, the Arguments, is macro substitutions that will be used as if the file were specified on the command line with a -macro argument. For example, the string:

```
SECTOR=S10, BPM=P2
```

will cause all occurrences of $(SECTOR) in the file to be replaced with S10 and all occurrences of $(BPM) with P2. Finally, the fourth column, the Policy, specifies whether to remove the parent display or not when the new one is activated. (The parent display is the one that contains the Related Display button.) The parent display will be replaced independently of the Policy if Ctrl-Btn1 is used instead of Btn1. Rows in the dialog box with blanks are ignored, even if they are before or between non-blank rows.

Neither labels or display labels can contain double quotes (").

When the parent display is replaced by the new display, its upper-left corner should be at the former position of the parent, unless the new display was already in existence. In that case the existing display is popped up, and its position is not changed.

Hidden buttons are implemented by setting the Visual attribute of the Related Display to invisible and placing a graphic over the related display. In this case there should only be one item in the list of displays. The graphic should make the operation of the related display clear. When the user clicks on the graphic, the Related Display will be executed. The Related Display is drawn as a stippled pattern to identify it. If the graphic is smaller, the stipple will show. If the graphic is larger, it will be possible to click on parts of the graphic without executing

the Related Display. Note that when two objects of the same size are on top of each other, then the outlines shown when doing dragging, pasting, *etc.* in EDIT mode will cancel, making the objects appear to not be included. Also in EDIT mode clicking on the two objects in the proper relative positions will select the graphic. To select the hidden Related Display, you can use Ctrl-Btn1. (See Mouse Editing Operations.) Since users are used to seeing the overlapping rectangles represent a related display, some thought should be used before implementing graphics to replace them.

The logic that is used to locate the specified files in the file system is described under ADL Files.

Scale Monitor

Also known as an Indicator. The scale monitor has Object (X Position, Y Position, Width, Height), Monitor (Readback Channel, Foreground, Background), Limits (Low Limit, High Limit, Precision), Label, Direction, and Color Mode attributes. The Scale Monitor displays the value of the process variable on a scale. The limits of the scale are the HOPR and LOPR values for the record associated with the process variable by default but may be set via the PV Limits Dialog Box.

Shell Command

The Shell Command has Object (X Position, Y Position, Width, Height) attributes. In addition, there are Foreground, Background, Label, and Label/Cmd/Args attributes. The Foreground and Background colors are changed by clicking to bring up the Color Palette. The Label is the optional label that appears on the button. Normally, whether there is a label or not, there is a “!” symbol on the button to denote a Shell Command control. However, if the label text begins with "-" (for example "-My Label"), the “!” symbol will not appear. The Command Label, Command, and Arguments are specified for each command in the Label/Cmd/Args dialog box. There currently can be up to sixteen commands for each button, and this can be changed in siteSpecific.h.

The Shell Command provides a means of running shell commands externally to MEDM. It is a menu button that usually has a graphic consisting of an “!” and a label. See the paragraph above for other options for the graphic and label. The menu items denote the commands that can be executed. If there is only one command specified in the menu, then the “!” and/or label appear in the middle of the button, and the command is activated immediately without bringing up a menu. It is not activated until the button is released, so you can depress the button to check if there is more than one menu item or not, then abort by releasing the button away from the Shell Command. If there is more than one item on the menu, the “!” and/or label are at the left of the button.

The Command Label is the label on the menu item in the menu that is brought up when the Shell Command button is pressed. The string in Command will be concatenated with the string in Arguments with a space in between, and the resulting string will be executed as a system command when the menu item with that label is selected. (The Arguments are actually superfluous, since any command arguments can just as well be included in Command.) MEDM will block until the command is executed, so it is almost always wise to include an “&” at the end of the Command (or the Arguments if they re used) so that it will execute in the background.

If the command contains a “?”, then the rest of the command will be ignored, and a dialog box will prompt the user to complete (or otherwise edit) the command.

The command can also contain any of the special characters that are available for the Execute Menu, however, it is suggested that “?” rather than “&?” be used. The Shell Command works just like the command part of the Execute Menu specification, and there are examples there.

Slider

Also known as Valuator. The Slider has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), Limits (Low Limit, High Limit, Precision), Label, Direction, and Color Mode attributes. There is also a Precision attribute. The Precision attribute is different from that in the Limits Attribute and specifies by how many units the slider moves when using Btn1 or the arrow keys as described below. It does not determine the precision for the value or limit numbers when these are shown on the slider. These are determined by the Limits Attribute.

Dragging the Slider button with Btn1 transmits values. The sensitivity with which values can be selected this way depends on the range of the slider and not on the Precision. Clicking Btn1 in the space on either side of the

slider button increments or decrements the value by an amount equal to the Precision. Clicking Ctrl-Btn1 in the space on either side increments or decrements the value by an amount equal to 10 times the Precision

In addition, the arrow keys increment or decrement the value by an amount equal to the Precision, and the Ctrl-Arrow keys increment or decrement the value by an amount equal to 10 times the Precision. The left and right arrow keys are operational when the Direction is right, and the up and down arrow keys are operational when the Direction is up. The arrow keys only function when the focus is on the slider button or the space on either side of it.

In EXECUTE mode the specified precision for motion of the valuator can be changed via a dialog box which is popped up by depressing Btn3 on the Slider object. A row of toggle buttons at the top sets the Precision as powers of 10 of the numbers on the buttons. The Precision can also be set in the entry box below the buttons. In addition, the value can be set directly in the lower entry box, providing an alternative to using Btn1 and the arrow keys.

Strip Chart

The Strip Chart has Object (X Position, Y Position, Width, Height), Plot (Title, X Label, Y Label, Foreground, Background), and Units attributes. In addition, there are Period and Units attributes. The period is the time between updates. The Units attributes are set in a dialog box. For each pen in the Strip Chart there is a Channel (process variable name) and a Color which can be specified in the Strip Chart Data Dialog Box. There currently can be up to 8 pens.

In EXECUTE mode the user can change the default values via the same Strip Chart Data Dialog.

Text

The Text object has Object (X Position, Y Position, Width, Height), Basic Attribute (Foreground), and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D), and Alignment attributes. In addition, there is a Text attribute, which is the text to be displayed. This text can have up to 255 characters. The Text object is drawn on the display drawing area and is not a widget. The text may extend beyond the height and width specified for the Text object or may not use up all the specified height and width. The Text can be sized as well as possible to the actual size of the text by using the Size option on the Edit Menu. This helps when trying to align the Text object with other objects.

The Text object may be created in two ways. You can drag a rectangle to the desired size, the same as for most other objects, or you can click and release Btn1 on the display drawing area. If you click and release Btn1, then the cursor changes to an I-beam and you can type the text directly onto the display rather than entering it in the Resource Palette. The text is finished when one of the following happens: (1) A button is pressed; (2) Return is typed; or (3) The pointer leaves the drawing area (either leaves the display or goes on top of a widget in the drawing area). The size of the text depends on the size of the previous object (not necessarily a Text object) in the Resource Palette. If the height is larger than 40, then 40 is used. The color is the same as the color of the previous object. You can control the size and color by first selecting an object (most usefully a Text object) with the desired size and color, then creating the new Text object by clicking and releasing Btn1.

The font used depends on the height of the Text object using an algorithm that is internal to MEDM. This algorithm may not be the most sensible, but it has been kept to avoid changing the layout of old displays. You adjust the font by adjusting the height, usually by experimentation. The text typically does not fill all of the specified height for the Text object. The text cannot contain double quotes (").

Text Monitor

Also known as Text Update. The Text Update has Object (X Position, Y Position, Width, Height), Monitor (Readback Channel, Foreground, Background), Limits (Precision), Alignment, Format, and Color Mode attributes. The Text Update is drawn on the display drawing area and is not a widget. It is the only graphics object without a Dynamic Attribute. It displays the value of the process variable in the chosen format.

Text Entry

The Text Entry has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), Limits (Precision), Format, and Color Mode attributes. The Text Entry provides a means of

displaying the value of a process variable in an entry box, where it can be edited and changed. The value is not sent until Return is pressed with the focus in the entry box. If the focus leaves the entry box before Return is pressed, the value of the process variable is not changed and the current value is restored into the entry box. The resulting value of the process variable is shown when and only when the cursor leaves the entry box. The value after leaving the entry box can be used to verify that the input was accepted.

If the native format of the process variable associated with the Text Entry is DBR_STRING or DBR_CHAR, then the string entered in the Text Entry will be sent to the process variable. If the native type is DBR_ENUM, then it will first try to match the entered string to one of the state strings of the process variable. If that fails, it will try to interpret the string as a number representing one of the state strings and send that number to the process variable.

For the native types for which a number is meaningful, the behavior of the Text Entry will depend on the format of the Text Entry. If the Text Entry has a format of octal, then numbers may be input with or without a leading 0 and will be interpreted as octal. That is, entering either 010 or 10 results in 8 decimal. Entering 8 is an error. The numbers will be displayed with a leading 0 when they are updated. (Tactually, the value displayed is determined by a routine in EPICS base. Older versions of EPICS base do not add the leading 0.) Similarly, if the Text Entry is formatted to hexadecimal, then numbers may be input with or without a leading 0x or 0X and will be interpreted as hexadecimal. In this case, 0x10, 010 or 10 will result in 16 decimal. In all other formats, input values will be interpreted as decimal unless they begin with 0x or 0X, in which case they are interpreted as hexadecimal. Input numbers with a leading 0 in these formats will not be interpreted as octal because of the chance for error or confusion. If you want to input octal, you must format the Text Entry to octal.

The Text Entry in MEDM is a valid drop site only if its Format is a string. The reason is to prevent trashing the value when the process variable name for the entry is inadvertently dropped on the entry. If you want to use drag and drop, the Text Entry must be formatted as a string.

WheelSwitch

The WheelSwitch has Object (X Position, Y Position, Width, Height), Control (Control Channel, Foreground, Background), Limits (Low Limit, High Limit, Precision), and Color Mode attributes. In addition, there is a Format. The WheelSwitch is a Controller which, like the Text Monitor, displays a number value. It has arrow buttons above and below the digits that are used to change individual digits, and it also accepts keyboard input.

The up and down arrow buttons are the main feature of the WheelSwitch. You click them to increment the digit. They repeat when held down. Only the buttons that give allowable values, that is, those between the limits in effect and allowed by the format, are visible at any time. The arrow buttons can also be navigated and operated via the keyboard. The up and down arrow keys increment or decrement the selected digit. They repeat at the same rate as the arrow buttons unless you have AutoRepeat set. Then they repeat at the keyboard repeat rate. The left and right arrows keys move the selected digit. The selected digit should be highlighted when the pointer is inside the WheelSwitch. Clicking on an arrow button moves the selected digit to that digit, in addition to incrementing or decrementing the digit. For small sizes of the WheelSwitch, the triangular parts of the arrow buttons are small, but they actually encompass the surrounding rectangle for purposes of clicking.

The WheelSwitch will also accept input of a new value via the keyboard. When you start typing, the arrow buttons will disappear and what you type will be displayed. The input will flash to indicate it is not yet a valid value. If the number you are typing is invalid for the format or the limits, it will beep and not let you enter the character that makes the number invalid. A minus sign can be typed at any time and toggles having a minus sign at the front of the number. Typing a plus sign will delete a minus sign. Backspace or Delete should delete the last character. (This is the only way to correct a typing mistake.) The arrow keys cannot be used to move within the input. Escape aborts the input. If the pointer leaves the WheelSwitch while you are in input mode, that will also abort the input. Type Enter to finish the input.

The Format is a string of characters that specifies how the value will be formatted. Its syntax is based on the format specification used in the C language, but is more limited. Valid formats are of the form "xxx%[space|+]m.nfyyy". They consist of an arbitrary string of characters xxx (the prefix), followed by a % sign, followed by a flag, which can be a space or blank, followed by an integer m representing the total number of digits including the sign and the decimal point, followed by a decimal point, followed by an integer n representing the number of digits after the decimal point, followed by an f, followed by an arbitrary string of characters yyy (the

postfix). If the flag is +, then the sign is always shown when the value is displayed. If the flag is a space, then the sign is shown only when the number is negative. The format, apart from the prefix and postfix, is interpreted as it is in the C language, but all the options allowed by C are not available. The WheelSwitch will check your format and suggest an alternative if it does not like it. If the Format is not specified, then the WheelSwitch calculates it based on the low and high limits and the precision.

Sample Formats				
% 6.2f	% 8.3f	%+6.2f	% 6.2f sec	Time: % 6.2f sec
Output Examples for These Formats				
10.00	100.000	+10.00	10.00 sec	Time: 10.00 sec
1.00	1.000	+1.00	1.00 sec	Time: 1.00 sec
-0.01	-0.001	-0.01	-0.01 sec	Time: -0.01 sec
99.99	999.999	+99.99	99.99 sec	Time: 99.99 sec
-99.99	-999.999	-99.99	-99.99 sec	Time: -99.99 sec

In EXECUTE mode, when using the arrow buttons or arrow keys, the number cannot be set to be larger than the format or low and high limits will allow. When typing the value, there is no similar limit. If the number is larger than the format will hold or outside the low and high limits, the digits are replaced by astericks, but a minus sign remains if the value is negative. Astericks, when they appear, may not be exactly lined up with the arrow buttons when proportional fonts are used, since they may be a different width than a digit. The limits and precision can be changed via the PvLimits dialog.

Composite Objects

Composite objects are created by grouping other objects via commands on the Edit menu. For many operations they are treated as one object.

The Composite has Object (X Position, Y Position, Width, Height) and Dynamic Attribute (Color Mode, Visibility, Visibility Calc, Channels A-D) attributes. Color Mode in the Dynamic Attribute is not relevant and is not used. In addition there is a Composite File attribute.

MEDM is designed so that when you change the foreground or background colors of a Composite object, you change the foreground or background colors of all the objects in the Composite simultaneously. Therefore, in order to change the colors of many objects at the same time, it is only necessary to group them first, making them Composite, and then change the color via the Resource Palette.

After a Composite object has been created, say by grouping some object, it can be made to get its objects from an ADL file, rather than use the one or ones it currently contains. Just enter the name of a valid ADL file for the Composite File attribute. The Composite will ignore things like the display and colormap in the ADL file and just use the objects in it. The positions of the objects will be adjusted so the upper left corner of the bounding box of all of them is at the specified x and y values of the Composite. The specified height and width of the Composite are adjusted to those of the bounding box. This feature allows designing a complicated Composite independently of its role in the current display without the need to group and ungroup the objects in order to edit them. This is especially useful if there are overlapping composites.

If the file has macros, you can specify them by placing a semi-colon after the file name and entering them as on the command line. For example:

```
beamline.adl;sector=1A,corrector=H3
```

If you do not specify macros in this way, any existing macros in the ADL file which uses this Composite will still be passed to the file. If you use this feature, those macros will not be passed, and the ones you specify will be used instead.

Since the Composite has a Dynamic Attribute, its visibility may be made to depend on a CALC expression involving up to four process variables or the other visibility features of a Dynamic Attribute. When the Composite is not visible, all of its objects are hidden. The objects however continue to receive updates from Channel Access if they were ever visible. This feature can be used to make parts of a display only appear under certain conditions. More generally, by using overlapped Composites, whole sections of a display can be made to appear, disappear, or change from one thing to another depending on the conditions. When using this feature, keep in mind the following considerations. It uses more resources to update Composites in addition to the objects they contain, in part because the Composite and its objects are always overlapped. It uses even more resources when Composites are overlapped. This may be a problem when there are rapid updates, but should be fine if the updates are infrequent. If objects inside the Composite have visibility that depends on the same process variables as those on which the Composite depends, there will be flashing when the objects update. There may be flashing in other cases, as well. If so, a possible solution is to limit the complexity of what you are doing.

A Composite may consist of only a single object. By enclosing any object in a Composite, it can be made to have a Dynamic Attribute.

In MEDM for reasons of efficiency, only the parts of a display that need to be refreshed are actually refreshed. If the visibility of a Composite changes, the display may be updated only within the bounds of the Composite. Since the text in a Text object may extend beyond the bounds of the object, you should insure that such text does not also extend beyond the Composite boundaries, which are determined by the size of the Text object, not the size of its text. Otherwise, the visibility of the extended part of the text may not be handled correctly.

Attributes

Object

All objects have Object attributes.

X Position	x coordinate of the top left corner of the object relative to the display.
Y Position	y coordinate of the top left corner of the object relative to the display.
Width	Width of the object.
Height	Height of the object.

Basic

For the Arc, Oval, Polygon, Polyline, Rectangle, and Text.

Foreground	Foreground color. Click to bring up the Color Palette to choose the color.
Style	Edge style for the object. See Edge Style.
Fill	Fill style for the object. See Fill Style.
Line Width	Line width in pixels. A line width of 0 is allowed. By X Windows convention, it may be drawn with a faster algorithm, but the look is X server dependent, and it may not update properly. For the Polygon and Polyline, changing the Line Width changes the object size and location accordingly. For the other graphics objects, the object size remains constant.

Dynamic

For the Arc, Oval, Polygon, Polyline, Rectangle, Text, Image, and Composite.

Color Mode	Color Mode for the object. See the Color Mode attribute. Not used for Image and Composite.

Visibility	Visibility mode for the object. See the Visibility attribute.
Color Rule	The color rule number for the object. (Integer number.) See Color Rules.
Visibility Calc	A CALC expression that determines whether the object is displayed or not. The expression should return 0 for False and anything else for True. See CALC Expression.
Channel A	Name of the main process variable associated with the object. An A in the CALC expression is replaced by the value of this process variable, and the values corresponding to G through L are obtained from this process variable.
Channel B	Name of the second process variable associated with the object. A B in the CALC expression is replaced by the value of this process variable.
Channel C	Name of the third process variable associated with the object. A C in the CALC expression is replaced by the value of this process variable.
Channel D	Name of the fourth process variable associated with the object. A D in the CALC expression is replaced by the value of this process variable.

Monitor

For the Bar, Byte, Meter, Scale Monitor, and Text Monitor.

Readback Channel	Name of the process variable to be read by the object.
Foreground	Foreground color. Click to bring up the Color Palette to choose the color.
Background	Background color. Click to bring up the Color Palette to choose the color.

Control

For the Choice Button, Menu, Message Button, Slider, and Text Entry.

Control Channel	Name of the process variable to be controlled by the object.
Foreground	Foreground color. Click to bring up the Color Palette to choose the color.
Background	Background color. Click to bring up the Color Palette to choose the color.

Limits

For the Meter, Bar Monitor, Scale Monitor, Slider, Text Monitor, and Text Entry. See the PV Limits Dialog Box for how these are used and specified.

Low Limit Source	Source for the low limit value (Channel, Default, User Specified)
Low Limit Channel Value	Value when source is Channel. (Floating point number.)
Low Limit Default Value	Value when source is Default. (Floating point number.)
Low Limit User Value	Value when source is User Specified. (Floating point number.)
High Limit Source	Source for the high limit value (Channel, Default, User Specified)
High Limit Channel Value	Value when source is Channel. (Floating point number.)
High Limit Default Value	Value when source is Default. (Floating point number.)

High Limit User Value	Value when source is User Specified. (Floating point number.)
Precision Source	Source for the precision value (Channel, Default, User Specified)
Precision Channel Value	Value when source is Channel. (Floating point number.)
Precision Default Value	Value when source is Default. (Floating point number.)
Precision User Value	Value when source is User Specified. (Floating point number.)

Plot

For the Cartesian Plot and Strip Chart.

Title	Title for the plot object.
X Label	X-axis label for the plot object.
Y Label	Y-axis label for the plot object.
Foreground	Foreground plot color. Click to bring up the Color Palette to choose the color.
Background	Background plot color. Click to bring up the Color Palette to choose the color.
Package	(Not Implemented.)

Plot Axis

For the Cartesian Plot.

Axis Style	Axis style for the plot axis. See Cartesian Plot Axis Style.
Axis Range	Source of the range for the plot axis. See Cartesian Plot Range Style.
Minimum Value	Minimum value shown on the plot axis. (Floating point number.)
Maximum Value	Maximum value shown on the plot axis. (Floating point number.)
Time Format	Time format for the axis. See Time Format.

Cartesian Plot Axis Style

For the Cartesian Plot.

linear	Use a linear axis.
log10	Use a log axis.
time	Use a time axis.

Cartesian Plot Range Style

For the Cartesian Plot.

from channel	Get the axis range from the process variable. (Usually LOPR and HOPR fields.) The range returned by the record support may not be appropriate for all fields of the record, may not be appropriate for the current values of the process variable, or LOPR and HOPR may not have been set to the best values.

user-specified	Specifically specify the minimum and maximum values for the axis.
auto-scale	Let the graph routine decide on the axis range depending on the data. This usually gives an attractive graph, but may cause the graph to flash if the range changes frequently.

Cartesian Plot Style

For the Cartesian Plot.

point	Plot the data as points.
line	Plot the data as lines.
fill-under	Plot the data as lines which are filled under (or over) from the line to the axis.

Cartesian Plot Time Format

For the Cartesian Plot when there is a time axis.

hh:mm:ss	*E.g.* 22:05:10, 05:43:35
hh:mm	*E.g.* 22:05, 05:44
hh:00	*E.g.* 22:00, 06:00
MMM DD YYY	*E.g.* Jun 27, 1997
MMM DD	*E.g.* Jun 27
MMM DD hh:00	*E.g.* Jun 27 22:00
wd hh: 00	*E.g.* Tue 22:00

Color Mode

For the Bar, Byte, Choice Button, Menu, Message Button, Meter, Scale Monitor, Slider, Text Entry, Text Monitor, and all objects with a Dynamic attribute.

static	Show the object in its normal colors.
alarm	Show the object in alarm colors based on the severity of the associated process variable. The alarm colors are Green for NO_ALARM, Yellow for MINOR_ALARM, Red for MAJOR_ALARM, White for INVALID_ALARM, and Gray if the alarm has an unknown value (because of an error in the record, for example).
discrete	If color rules are implemented and the object has a Dynamic attribute, use the color rules. See Color Rules. Otherwise the same as static.

Color Rule

This is part of the Dynamic attribute that is used if color rules are implemented. See Color Rules.

set #1	Use color rules defined by set #1.
set #2	Use color rules defined by set #2.
set #3	Use color rules defined by set #3.

set #4	Use color rules defined by set #4.

Direction

For the Bar Monitor, Byte Monitor, Scale Monitor, and Slider.

up	Direction for the object is up or up/down.
right	Direction for the object is right or left/right.
down	Only used in the Bar Monitor. Otherwise, MEDM treats this the same as up.
left	Only used in the Bar Monitor. Otherwise, MEDM treats this the same as right.

Edge Style

This is part of the Basic attribute.

solid	Use solid lines.
dash	Use dashed lines.

Erase Mode

For the Cartesian Plot with an erase channel.

if not zero	Erase the plot if the erase-channel process variable is not zero.
if zero	Erase the plot if the erase-channel process variable is zero.

Fill Mode

For the Bar Monitor.

from edge	Fill from the lower end to the current value.
from center	Fill from zero to the current value.

Fill Style

This is part of the Basic attribute.

solid	Fill the whole shape with color.
outline	Only color the outline.

Image Type

For the Image.

no image	Do not display the image.
gif	The image is Graphics Interchange Format (GIF), the only allowable image type at this time.

Label

For the Meter, Bar Monitor, Scale Monitor, and Slider.

none	No extra features, except the limits are shown for the meter. You can eliminate the limits on the meter by resizing it vertically so they do not show.
no decorations	Same as none except for the Bar Monitor. For the Bar Monitor only the background and the bar show. This allows making bar graphs in MEDM.
outline	Show limits.
limits	Show limits and a box for the value (except there is no box for the Slider).
channel	Show limits, a box for the value, and the process variable name (except there is no box for the Slider).

Plot Mode

For the Cartesian Plot with a scalar process variable.

plot n pts & stop	Plot n points corresponding to the first n changes of the process variable, then do not plot any more points.
plot last n pts	Plot n points corresponding to the last n changes of the process variable, overwriting previous points.

Related Display Mode

For the Related Display.

create new display	Create a new display and leave the current one.
replace display	Create a new display that replaces the current one.

Related Display Visual

For the Related Display.

menu	Use a pull down menu for the choices.
a row of buttons	Use a row of buttons for the choices.
a column of buttons	Use a column of buttons for the choices.
invisible	Do not show anything for the choices. This mode is intended to be used with a graphic or other object on top of the related display. The graphic should make the operation of the Related Display clear. In EXECUTE mode, the Execute-Mode Popup Menu contains an item to toggle the makring of hidden buttons in case the user cannot find them.

Stacking

For the Choice Button.

column	The buttons are arranged in a row. (This appears to be a mistake, which will not be corrected because of existing screens.)
row	The buttons are arranged in a column. (This appears to be a mistake, which will not be corrected because of existing screens.)
row column	The buttons are automatically arranged in rows and columns.

Text Align

For the Text and Text Monitor.

horiz. left	Align the text at the top left of the object. (Capital letters will line up with the top of the object, and the text will start at the left of the object.)
horiz. centered	Align the text at the top center of the object. (Capital letters will line up with the top of the object, and the text will be horizontally centered in the object.)
horiz. right	Align the text at the top right of the object. (Capital letters will line up with the top of the object, and the text will end at the right of the object.)
vert. top	No longer used. In ADL files MEDM treats this the same as horiz. left (as it used to) for backward compatibility.
vert. centered	No longer used. In ADL files MEDM treats this the same as horiz. center (as it used to) for backward compatibility.
vert. bottom	No longer used. In ADL files MEDM treats this the same as horiz. right (as it used to) for backward compatibility.

Text Format

For the Text Entry and Text Monitor. For all of the formats, the result depends on the number itself and the precision as returned from channel access for the process variable. The precision is typically the PREC field for the associated record and may be changed by the PV Limits Dialog Box. The PREC field is only determined when the process variable is connected, so subsequent changes to the PREC do not affect the format until MEDM is restarted.

decimal	The text is a number with or without a decimal point, but not in exponential notation. *E.g.*10000.00 or 10000. Very large numbers, which do not fit in the allocated space for text, are converted to exponential regardless. It is what is returned by the EPICS routine cvtDoubleToString().
exponential	The text is in exponential notation. *E.g.* 1.00e+04.
eng. notation	The text is in engineering notation. *E.g.* 10.00e+03.
compact	The text is supposedly in the most compact form of decimal or exponential. (Similar to the %g format in C.) The form may not be the most compact, however. It is what is returned by the EPICS routine cvtDoubleToCompactString().
truncated	The text is truncated to the largest integer. *E.g.* 10000.
hexadecimal	The text is truncated to the nearest integer and shown in hexadecimal. *E.g.* 0x3e8. Input is interpreted as hexadecimal.
octal	The text is truncated to the nearest integer and shown in octal. *E.g.* 01750. Input is interpreted as octal.
string	This is the same as decimal except that for large numbers or precision, it can be in exponential format. It is what is returned by the EPICS routine cvtDoubleToString().
sexagesimal	The text is in degrees or hours, minutes, and seconds with colons between the three

	fields. *E.g.* 12:45:10.2. The degrees or hours field is the integer part of the number, and the fractional part is expressed as an integer number of minutes and seconds plus fractions of a second. The example, 12:45:10.2, is the number 12.752833 expressed as 12 degrees or hours, 45 minutes, and 10.2 seconds. The precision determines the number of digits after the first colon. For precisions of 0, 1, 2, 3, 4, 5, and 6 the number above would be expressed as 13, 12:5, 12:45, 12:45:1, 12:45:10, 12:45:10.2, and 12:45:10.20, respectively. On input colons may be replaced by spaces or omitted, and a decimal point may be used. The inputs 10.75, 10:45, 10.45.0, and 10 45 00 are all treated as a quarter to eleven. The inputs -1, :-1, and ::-1 are interpreted as -1:00:00, -:01:00, and -0:00:01, respectively.
sexagesimal-dms	This is the same as sexagesimal except that the number is assumed to be in radians and is expressed as degrees, minutes, and seconds with 2π radians representing 360 degrees. On input the number is multiplied by $180/\pi$ and then converted to sexagesimal as above. On output the number is converted as sexagesimal above and then multiplied by $\pi/180$ to get radians.
sexagesimal-hms	This is the same as sexagesimal except that the number is assumed to be in radians and is expressed as hours, minutes, and seconds with 2π radians representing 24 hours. On input the number is multiplied by $12/\pi$ and then converted to sexagesimal as above. On output the number is converted as sexagesimal above and then multiplied by $\pi/12$ to get radians.

Time Units

For the Strip Chart.

milli-second	The update period is in milli-seconds.
second	The update period is in seconds.
minute	The update period is in minutes.

Visibility Mode

This is part of the Dynamic attribute.

static	The object is displayed always. Updating objects are always above non-updating objects. Specifying a process variable in the Dynamic Attribute makes the object update and insures that it will retain its stacking order among other updating objects even if it never changes its state and is always displayed. See MEDM Objects for an efficient way to do this. If retaining stacking is not necessary, it is more efficient to not specify a process variable if you always want it to display the object.
if not zero	The object is displayed if the process variable is not zero.
if zero	The object is displayed if the object is zero.

CALC Expressions

CALC expressions are used to determine visibility for objects with a Dynamic Attribute and to determine the frame number for the Image, which also has a Dynamic Attribute. For visibility the expression should return 0 for False and anything else for True. For the frame number for the Image it should return a number, which will be rounded to the nearest integer. Frame numbers start with 0.

The complete syntax of the CALC expression is described in detail in the Record Reference Manual found in the Epics Documentation under IOC software for any version of EPICS base, through at least Base 3.13. It is a case-independent C expression that can include functions such as ABS, SQR, MIN, MAX, CEIL, FLOOR, LOG, LOGE, EXP, SIN, SINH, ASIN, COS, COSH, ACOS, TAN, TANH, and ATAN, as well as the usual C operators, except that != is replaced by # and == is replaced by =. The letters A-L obtain their values from the Dynamic Attribute's Channels A-D and are replaced as follows.

A	The value of Channel A.
B	The value of Channel B.
C	The value of Channel C.
D	The value of Channel D.
E	Reserved.
F	Reserved.
G	The COUNT of Channel A.
H	The HOPR of Channel A.
I	The STATUS of Channel A.
J	The SEVERITY of Channel A.
K	The PRECISION of Channel A.
L	The LOPR of Channel A.

Examples are:

Show the object whenever the value of Channel A is zero (The same as a Visibility of “if zero”):

```
!A
```

Show the object whenever the value of Channel A is not zero (The same as a Visibility of “if not zero”):

```
A
```

Show the object whenever the value of Channel A is 12:

```
A=12
```

Show the object whenever the value of Channel A is not 12:

```
A#12
```

Show the object whenever the values of Channel A, Channel B, and Channel C are all negative:

```
A<0&&B<0&&C<0
```

Show the object whenever the main process variable is within 90% of its HOPR or LOPR limits:

```
A<.9*L||A>.9*H
```

Show the object whenever the SEVERITY of the main process variable is not 0:

```
!J
```

Grid

There is a grid in EDIT mode that can be toggled on and off. Each display has its own grid and grid properties. These are saved in the ADL file. The grid properties (the grid spacing, whether the grid shows, and whether snap to grid is in effect) for the current display can be set via the Grid options on the Edit menu or on the Resource Palette for the display. The grid is drawn in the foreground color of the display. This color can be

changed in the Resource Palette. The Space Evenly options on the Edit menu use the current grid spacing, which can be changed to allow different spacing when using these options.

When snap to grid is enabled, moving with Btn2 (but not the arrow keys), pasting, and object creation constrain the upper left corner of the (group of) selected objects to be at a grid point. For object creation all corners are constrained to be at grid points. These operations do not change the locations of selected objects relative to each other. Objects can also be aligned to the grid via the Edit menu. The Edit-menu align options align each selected object, and may change the locations of selected objects relative to each other.

Macro Substitution

Strings of the form $(name) in an ADL file can be replaced by some other string, both on the command line and when calling a related display. Specific directions for each of these cases are given in the correspoonding sections of the manual. In general, there is an argument string of the form “name1=value1 [,name2=value2]…”. All occurrences of “$(name1)” in the ADL file are replaced with “value1”, then all occurences of $(name2) are replaced by value2, *etc*. The substitition is recursive; that is, if value1 contains an occurrence of $(name2), then when name2=value2 is processed, that occurrence will be replaced by value2.

For example, if a Readback Channel for a Meter is entered in the Resource Palette as "S$(sector):$(corrector):CurrentAO", and the argument string contains "sector=1A, corrector=H3", then the Readback Channel actually used will be "S1A:H3:CurrentAO". The same ADL file can be used for sector 38B and corrector H4 by using the argument string "sector=38B, corrector=H4", with a resulting Readback Channel of "S38B:H4:CurrentAO".

Drag-And-Drop

MEDM has a drag-and-drop mechanism in EXECUTE mode, using Btn2. This allows dragging the process variable names associated with an MEDM object to another place. The place must be a valid Motif drop site. Such places include most Motif text fields. Programs like Probe, the knob manager (KM), and HistTool, which are EPICS extensions, have been designed to have valid Motif drop sites, and it is easy to drag a process variable name from an MEDM display to them, rather than having to enter the name manually.

Pressing the Escape key will abort the drag and drop. There may be a do-not-enter symbol (circle with a bar across it) when the drag is over an invalid drop site and an arrow when the drop site is valid. There is also always a symbol composed of two overlapping sheets of paper that indicate this is a copy operation (the only type available in MEDM). When dropping the process variable names, they will appear to go back to where they originated if the drop site is invalid and will appear to be absorbed by the drop site otherwise.

The valid drop places unfortunately do not include XTerm’s and most other X clients, and the Motif Btn2 drag-and-drop is not the same as the common Btn2 paste associated with the X selection or the clipboard. However, at the start of the Drag and Drop process the PV names are also put into the X selection and the clipboard. So, even though you cannot drag the PV names into an Xterm or other X client, you can select them by using Btn 2 as for Drag and Drop, release Btn2 to end the Drag and Drop, and then click Btn2 in an Xterm. If there is more than one PV name associated with the object, the pasted PV names are separated by a space.

In addition, Namecapture, is a program designed to help with transferring names obtained by Motif drag-and-drop to other programs. It automatically converts a dropped name to an X selection so the common Btn2 paste works. There are also ways in Namecapture to add the dropped names to a file, even to generate lines in a BURT request file. Both Namecapture and KM are described in the online EPICS documentation for EPICS extensions.

The color of the names depends on the alarm severity. Green is NO_ALARM; Yellow is MINOR_ALARM; Red is MAJOR_ALARM; White is INVALID_ALARM; and Gray means the alarm has an unknown value (owing to an error in the record, for example).

It is not necessary to drop the name in a valid place; it can be read and discarded. Consider using the PV Info feature to look at (as opposed to dragging and dropping) process variable names.

The Text Entry in MEDM is a valid drop site only if its Format is a string. The reason is to prevent trashing the value when the process variable name for the entry is inadvertently dropped on the entry.

For the Cartesian Plot, the names are in columns corresponding to the traces. The last row corresponds to the trigger and erase channels, whether they are defined or not.

Dialog Boxes

Display List Dialog Box

The Display List Dialog Box is invoked from the Execute-Mode Menu. It has a list of the current displays and the X Window ID of the main MEDM window. You can select one or more displays. There are buttons to Raise or Close the selected displays and to Close, Refresh, or get Help for the dialog box itself. Displays that are raised should appear on the current Workspace. Displays that are closed, whether from this dialog box or from the Exeute-Mode Popup Menu, must be reopened from the File menu. The dialog box should automatically refresh without using the Refresh button.

Message Window

The Message Window can be accessed from the View Menu in the MEDM main window. Most of the error and warning messages from MEDM are posted in this dialog box. They are also printed to the terminal or console from which MEDM was started. There are two categories of messages. By default the important messages cause the Message Window to pop up. The others are just written to the Message Window whether it is visible or not. By unchecking the "Raise Message Window When Important Message is Posted" checkbox, you can keep the window from popping up for important messages as well. You can always pop up the Message Window yourself at any time from the MEDM main window. If you encounter problems, it is a good idea to do this.

There are buttons to Close the Message Window, Clear the messages, Print the messages, Mail the messages, and get Help. Printing uses the default print command. See Printing and the Print Setup Dialog Box for more information. Mail brings up a dialog box for sending the information to a mail recipient. Mail is not implemented for WIN32. The Help button should bring you to this section of the manual in your browser.

The text in the Message Window is selectable, so you can copy it to other programs.

Statistics Window Dialog Box

The Statistics Window can be accessed from the View Menu in the MEDM main window. It shows statistics about various things that are happening in MEDM. These statistics are accumulated by a routine that is called by a timer set to 1 second. The actual elapsed time between calls may be slightly different, depending on the load in MEDM. There are Close, Reset, and Mode buttons at the bottom of the dialog box. The Close button closes the dialog box. The reset button resets the start time and the accumulated values to zero. The mode button toggles between the two modes of displaying statistics, either for the last interval or for the accumulated averages.

Last Interval Mode

Time Interval: The actual time for the last interval.

CA Channels: The number of channels (one per process variable) requested by all the objects in all the screens.

CA Channels Connected: The number of channels successfully connected.

CA Incoming Events: The number of events received from Channel Access in the last interval. An event is received for a channel when:

- A significant change in the connection occurs (*e.g.* connect, disconnect, reconnect).
- There is a significant change in the channel (*e.g.* value, alarm status, severity status).
- A requested value is returned.
- There is a change in access rights (read, write).

MEDM Objects Updating: The number of objects that could potentially be updated (redrawn on the screen). (An MEDM object can have more than one process variable associated with it.)

MEDM Objects Updated: The number of objects actually updated in the last interval. (Can be larger than the number updating if some were updated more than once).

Update Requests: Number of object updates requests that were queued for processing.

Update Requests Discarded: Number of objects that got a new update request before the old one was processed.

Accumulated Averages Mode

CA Incoming Events: Average of CA Incoming Events (see above) over the time elapsed since the last reset.

MEDM Objects Updated: Average of MEDM Objects Updating.

Update Requests: Average of Update Requests.

Update Requests Discarded: Average of Update Requests Discarded.

Total Time Elapsed: The time over which the averages are calculated.

Print Setup Dialog Box

The Print Setup dialog box is invoked from the Print Setup item on the File Menu in the MEDM main window. In this dialog box you can change the print command, the orientation, the paper size, the title, the width, the height, whether to print the date or the time, whether to print to a file, and the filename to which to print. By default MEDM prints a screen dump of the display using the print command in Portrait mode with the display sized to fit the paper. Above the display is printed the date, a title that is the name of the display without the path, and the time. You have the option of printing no title, the long or short name of the display, or your own title. When printing to a printer or file, keep in mind that the output is Postscript.

When you change the settings in the Print Setup dialog box, they are not recorded until you press the Apply button. If you close the dialog box, they will be lost. If you press Cancel, the settings will be returned to the ones that were previously stored. If you press Print, you will be promted to save the settings if they have not been saved. If there is more than one display, you will be prompted for the display to print with a special cursor. The Help button should bring you to this section of the manual in your browser.

More information on printing a display is given under Printing.

PV Info Dialog Box

The PV Info dialog box is invoked from the Execute-Mode Menu. You are prompted with a PV cursor to pick an object on the display. The dialog box then shows information about all the process variables associated with that object. For each process variable the information includes the name, the description, the record type (AO, AI, CALC, *etc.*), the native type (DBF_DOUBLE, DBF_STRING, DBF_ENUM, *etc.*), the count (length of the array or 1 if not an array), the access rights (RW), the IOC, the current value with associated timestamp, and its alarm status, as well as other information depending on the native type. Only the value of the first element of an array is shown. The record type is a newer feature of Channel Access and may not be available if the version running in the IOC is older. The information shown is that stored by MEDM, except for the current value and the associated time stamp, which are obtained when you pick the object. Items such as HOPR and LOPR are the startup values and are not updated automatically except when the connection changes. It should be possible to get some of this information if the process variable has lost its connection; that is, when the object is white but was previously not white.

When choosing an object with the PV cursor, the smallest object under the cursor will be chosen. If the object selected by the PV cursor is part of a Composite that has process variables associated with it, then the dialog box will show the process variables for the object if it has any. Otherwise it will show the process variables for the Composite. If an object that is part of a Composite is hidden because the Composite is currently not visible, then the process variables for the Composite will be shown. If the object is not part of a Composite and is not currently visible, the dialog box will still show its process variables. The name of the MEDM object is shown in the dialog box to help avoid confusion in ambiguous cases.

When running through some older versions of the Gateway, there may be a long delay (1 minute) before the dialog box comes up. This is owing to a bug in the portable Channel Access server, which has now been fixed.

PV Limits Dialog Box

You can invoke the PV Limits dialog box from the Execute-Mode Menu in EXECUTE mode and from the Channel Limits button in the Resource Palette in EDIT mode. When invoked from the Execute-Mode Menu, you are prompted with a PV cursor to pick an object on the display.

The limits of an object are the endpoints of the range of values that is displayed. For example, a Meter has a maximum and a minimum value, and values beyond these limits are not displayed correctly. The Meter becomes "pinned". The precision is the number of digits displayed after the decimal point. The limits and the precision of numbers displayed in an object in can be specified at design time and changed at run time for all objects for which this is relevant. This is accomplished through the PV Limits Dialog Box. This dialog box looks the same in EDIT as in EXECUTE mode, but the meaning and the available options may be different. In all cases, options that are not meaningful are not selectable and appear grayed out.

In EXECUTE mode the process variable name will appear at the top of the dialog box. IN EDIT mode it will just have EDIT Mode Limits at the top.

There are three items that can be set in the dialog box: the Low Limit, the High Limit, and the Precision. For each of these items, the value used for the item may come from three different sources: Channel, Default, and User Specified.

Channel

In either EXECUTE or EDIT mode specifying Channel means the values in EXECUTE mode come from the record associated with the process variable. This is the default and formerly was the only option available in MEDM. Typically, the High Limit comes from the HOPR field, the Low Limit comes from the LOPR field, and the precision comes from the PREC field. More accurately, they are whatever is returned from the record-support routine, get_graphic_double(), when Channel Access asks for these limits and precision. This depends on the record in question and on which field in the record. Fields other than the VAL field often just return the default value zero for all three items, even when HOPR, LOPR, and PREC are returned for the VAL field. Some records, especially homegrown ones, may not even have these fields and may not return anything sensible. Moreover, if the person who designed the database did not specifically specify values for the HOPR, LOPR, and PREC fields (a not uncommon occurrence), the limits and/or precision will default to zero. In most cases, however, the database designer has specified the values to be ones appropriate for use in applications like MEDM. That is why these fields are there in the first place. These values are fixed in the record, and they apply to all objects using that process variable in all MEDM screens in all running MEDMs. If both the High Limit and the Low Limit are zero, MEDM converts the High Limit to 1.0.

Default

In EXECUTE mode specifying Default means the values used are those specified by the designer of the MEDM screen or the MEDM defaults if none were specified. The MEDM defaults are 0.0 for the High Limit, 1.0 for the Low Limit, and 0 for the precision. The default value cannot be changed in EXECUTE mode. Use User Specified instead. In EDIT mode the screen designer specifies the default values via the text entry box and by picking this source option specifies that these values will be used unless the user specifies something different. The screen designer can thus override what the database designer specified (or failed to specify).

User Specified

This option is only available in EXECUTE mode. The user of the screen can set the values to whatever he or she wants. It may be appropriate, for example, to set a reduced range for an object when doing fine-tuning. The user can override both the screen designer and the database designer by using this option and entering new values. He can revert to either of the other sources by choosing those options.

Note that the screen designer can set the source to be Channel or Default and can set the default value. The user can set the source to be any of the three and can set the user-specified value. If the source is Default, he will get the default specified by the designer, and if it is Channel, he will get the value from LOPR or HOPR. The value for the channel cannot be set at any time. It comes from the process variable. The dialog box will not respond to attempts to set items that are not meaningful or allowed.

Both limits and precision are relevant and may be specified for the Meter, Bar Monitor, Scale Monitor, Slider, and WheelSwitch. Only the precision is relevant and may be set for the Text Monitor and Text Entry. None of these is relevant for other MEDM objects and the dialog box should not be available, nor should there be limits attributes stored in the ADL file. The Cartesian Plot has other ways to set its limits. The limits for the Strip Chart are set in the Strip Chart Data Dialog.

Strip Chart Data Dialog Box

You may invoke the Strip Chart data dialog box from the PV Limits item in the Execute-Mode Menu in EXECUTE mode and from the Channel/Color button for the Strip Chart in the Resource Palette in EDIT mode. When invoked from the Execute-Mode Menu, you are prompted with a PV cursor to pick an object on the display, the same as for the PV Limits Dialog Box. In EDIT mode you can specify the channel, the color of the line for that channel, the source for the Low Limit, the Low Limit, the source for the High Limit, and the High Limit. There is more information about limits under the PV Limits Dialog Box. The precision cannot be specified although it will affect the number of figures after the decimal point displayed for the Low and High Limits in the dialog box. It is the higher of the value for the PREC field or 2. In EXECUTE mode you can specify all except the channel, which is fixed. This includes the color.

The source may be Channel, Default, or User Specified and works the same way as for the PV Limits Dialog. Keep in mind that the screen designer can set the source to be Channel or Default and can set the default value. The user can set the source to be any of the three and can set the user-specified value. If the source is Default, he will get the default specified by the designer, and if it is Channel, he will get the value from LOPR or HOPR. The value for the channel cannot be set at any time. It comes from the process variable. The dialog box will not respond to attempts to set items that are not meaningful or allowed.

In EXECUTE mode you can also change the Period and the Units. In EDIT mode these are set on the Resource Palette and they are the values used if there is no user intervention.

The values entered are not applied until the Apply button is pressed. The Cancel button discards values since the last Apply and dismisses the dialog box. The Apply button does not dismiss the dialog box. If there has been no Apply, the original values are unchanged after Cancel. The Apply button may also be used to reset the Strip Chart. Text values will be reformatted when the Apply button is pressed and when other controls are used. If you press Return in a text box, the highlighted button, usually the Apply button will be activated instead.

Help

Help from the menus and context sensitive help uses Netscape. If Netscape is not up, it takes a little while for it to come up after the first help request; however, once it is up, new help topics appear quickly inside the existing Netscape. The Netscape window raises so as to be visible whenever help is called. There is a dialog box for Help on Help (in case Netscape is not working) describing the Netscape help and what is necessary to make it work. The name for the Netscape executable can be specified in the environment variable NETSCAPEPATH. The URL for the HTML file that Netscape reads can be specified in the environment variable MEDM_HELP_URL. Otherwise, MEDM uses the MEDM default, which is a file at Argonne National Laboratory, perhaps outside your firewall. The default can also be changed in siteSpecific.h when MEDM is built.

For WIN32 MEDM will call the default browser with the MEDM help URL. The browser does not have to be Netscape.

There is bubble help to identify the objects in the Object Palette, and there is a summary of EDIT operations on the Edit menu, which appears on the main window and as a popup menu when editing a display.

Each display has a provision for help on the display. This help is implemented via the Help option on the window menu button (the button with a short bar on it located at the left of the title bar for the display). This button does not exist unless the environment variable MEDM_HELP is defined. In that case when this option is selected from the window menu, MEDM makes a system call of the form “$MEDM_HELP full-adl-pathname &”. The value of MEDM_HELP should be the name of a program that uses the full pathname of the ADL file for the display to provide help, for example, by bring up Netscape with an appropriate page based on the filename. What the program does or how it is written is independent of MEDM. This feature is different from and should not be confused with the browser help.

Medm Smart Startup

In order to amortize the cost of Xt Intrinsics and Motif initialization over many displays, and to offer an efficient way to start up MEDM displays “after the fact”, an MEDM remote request protocol has been established.

If MEDM is started with the "-attach" option on the command line, it looks for another MEDM running (on any host machine) which has the same X Windows display as the requested display. (Note that the X Windows display refers to the environment variable DISPLAY or a display specified on the command line as *e.g.* “-display phoebus:0”, which is different from an MEDM display as specified by an ADL file.) If one is found, the MEDM display request is forwarded to the remote MEDM for processing.

What is common among all attached MEDMs is that they were started from the same X display. They may have been started on different machines on different networks by different users. Note that the user and access permissions will be the same as for the original, host MEDM. This may lead to confusing behavior if successive MEDMs are started from different machines, from different networks, or by different users.

The default behavior, equivalent to "-local" on the command-line, is to not participate in the smart-startup protocol and to act independently of other MEDMs. Also, you can use the "-cleanup" parameter on the command line to ignore any existing MEDMs and to use this MEDM as the remote MEDM for future remote requests.

MEDM implements the smart startup by setting a property on the root window of the display. This property stores the X window id of the MEDM that is available for remote requests. An MEDM using "-attach" looks for the property when it starts up. If it finds the property, it gets the X id from the property and begins sending all of the information from its command line to the host MEDM by generating ClientMessage events. These messages are passed on the display, not on the machines where the MEDMs may be running. The host MEDM processes these events along with its other events, such as ButtonPress, for example, and starts up the requested ADL displays. After all the information has been sent the second MEDM exits.

If MEDM crashes badly or is killed with a SIGKILL, which cannot be trapped, this property may still be left on the root window. This should not cause a problem as the next MEDM started with "-attach" will find out that the window id is not valid and start itself as if "-cleanup" had been specified.

You can look at and remove the property yourself if you wish:

```
% xprop -root | grep MEDM
```

If there is one, you can remove it by doing the following:

```
% xprop -root -remove <property-name>
```

where <property-name> is the name of the property found. (The property name will have the MEDM version and have _EDIT_ or _EXEC_ in the name, depending on how the remote MEDM was started.) Most times, MEDM will clean up properly when it exits, even if it was stopped abnormally. It is no longer necessary to use "-cleanup" to clean up the root window property, and in fact it should rarely be necessary to use it at all.

Display Colors

Each display in MEDM has its own colormap with up to 65 colors. The colors of all the objects and the display itself are taken from this colormap. These colors may be seen, for example, in the color palette. The colors for each object are specified as a number, which is an index into the colormap array. If the colormap for a display is changed, the colors of the object will be different if the color for that index is different. Different displays may simultaneously have different colormaps. Of course, if there are more total colors than the X colormap can hold, some of these will not display correctly. This is particularly a problem for 8-bit X displays. It can be overcome by specifying a private X colormap for MEDM using the command-line options. The display colormap may be specified in one of two ways.

The first way is by specifying the Colormap attribute of the Display object. The Colormap property is the name of a file containing a "color map" block. The specified file may have other blocks than a "color map" block and, in fact, any ADL file containing a "color map" block is a valid choice.

The second way is by specifying a "color map" block in the ADL file. If the Colormap attribute is blank, this is done for you automatically when you save the display in an ADL file. You may use such a block as a

template to define your own colors. The items in the "color map" block are the RGB values of each color index. The format should be obvious from the template.

A display created from New on the File Menu has the MEDM internal default colormap. This default is specified in the siteSpecific.h file in the MEDM distribution and may be changed if you build MEDM yourself.

When a display is saved, if there is a Colormap specified in the Display object, no "color map" block is written to the ADL file. Otherwise, the current display colormap is written as a "color map" block. Note that setting the Colormap property to blank will result in the current colormap, not the default colormap, being written to the ADL file. Specifying a colormap file saves room in the ADL file and allows using a consistent colormap for a group of displays at the expense of reading an additional file when executing the display. If the Colormap is not specified, and the colormap in the ADL file is not edited, then the colormap will be consistent for all displays created with the same MEDM. The latter has been the usual method and has a number of advantages.

There is no means in MEDM of editing the display colormap except by editing a colormap file. Keep in mind that once displays are created, changing the colormap globally may make them unaesthetic. As with the choice of fonts, the colormap strategy is one that should be made early and wisely.

Resizing Displays

Displays in MEDM do not have resize handles and may only be resized by invoking the resize method appropriate for the window manager in use. This is typically by right clicking the button on the left of the title bar.

In EXECUTE mode, the contents of the display will be scaled to fit the new size of the display. If you wish to preserve the aspect ratio of a display, you can terminate the resize option with Shift-Click instead of Click. In that case MEDM modifies the size you have chosen by decreasing the dimension that is too large. Text tends to scale more attractively if you keep the aspect ratio fixed.

In EDIT mode only selected objects are scaled to fit the new size of the display. The others are left where they were. This gives you some flexibility. There is no aspect-ratio preserving option.

Editing

When MEDM is in EDIT mode, editing operations are initiated through mouse or pointer input, keyboard input, or menu selection.

Mouse Editing Operations
Keyboard Editing Operations
Menu Editing Operations (Edit Menu)

Mouse Editing Operations

Several editing operations using the mouse are initiated by choosing one or more objects with the mouse. If the mouse is clicked, the object under the cursor is chosen. If the mouse is dragged, a rubberband box appears, and all of the objects included within the box when the button is released are chosen. Chosen objects should be distinguished from selected objects. Chosen objects are those chosen for the current operation. Selected objects are those that are highlighted. Chosen objects can be added to or removed from the selected objects or may replace them, depending on the operation.

	Select Mode (Pointing Hand Cursor)
Btn1	Vertex Edit. If a Line, Polyline, or Polygon was previously the only object selected, and if the button is pressed near a vertex, dragging moves the vertex. During dragging, when the Shift key is depressed, the dragging is constrained to be in a direction that is a multiple of 45 degrees. (The Shift key must be pressed after the button is pressed or the operation will be treated as a Shft-Btn1 operation.) Select objects. If the conditions for a vertex edit are not satisfied, then choose objects by dragging or clicking. These become the new selected objects. If only one object was previously selected, however, then additional clicking on that element toggles its selection. Clicking on the background toggles selection of the display. In the case of

	completely overlapping objects, only the top one can be selected in this way.
Shift-Btn1	Add or remove objects from the selected objects. Choose objects by dragging or clicking. The chosen objects are removed from or added to the selected objects, depending on whether they were included or not before.
Ctrl-Btn1	Select an object that may be under another object. Choose objects by clicking. Successive clicks will cycle the choice through any overlapped objects. The chosen object will be the new selected object. Only one object can be selected this way.
Btn2	Move objects. Choose an object by pressing the button on it, and then drag the object to a new position. If more than one object was previously selected, then dragging any one of the selected objects will move all of them. In the case of completely overlapping objects, only the top one can be moved in this way. An object cannot be moved off the visible part of the display in this way. (The comparable keystroke operation does allow moving off the visible part of the display.) Btn2 supports snap to grid. The Motif Cancel key (usually Esc or Escape) cancels the move as long as the button has not been released.
Shift-Btn2	Not used.
Ctrl-Btn2	Resize objects. Choose an object by pressing the button on it, then drag the object to resize it. If more than one object was previously selected, then clicking and dragging any one of the selected objects will resize all of them. The top left corner is held fixed during resizing. In the case of completely overlapping objects, only the top one can be resized in this way. The Motif Cancel key (usually Esc or Escape) cancels the resize as long as the button has not been released. (You can release the Crtl key before pressing Cancel, though.)
Btn3	Popup the editing menu.

	Create Mode (Crosshair Cursor)
Btn1	Create the object in the method appropriate for the object, usually by dragging to the desired size. (The Text, Polyline, and Polygon objects have special create methods.) The object must start on the visible part of the display and not on top of an object that is a widget, but it may extend past the visible part if the button is released outside the display or it may end on a widget. (It is not suggested to do this, however.) The mode will then change to Select Mode. Create supports snap to grid and aligns all corners to the grid, not just the upper left corner. The Motif Cancel key (usually Esc or Escape) cancels the create as long as the button has not been released.
Btn2	Not used.
Btn3	Popup the editing menu.

Keyboard Editing Operations

The keyboard editing operations duplicate actions that can be done by mouse operations. There is much finer control, however, with the keyboard operations

Arrow Key	Move selected objects. Objects can be moved off the visible part of the display.
Shift-Arrow Key	Same as for Arrow Key.
Ctrl-Arrow Key	Resize selected objects. The top left corner remains fixed during resizing.

Menu Editing Operations

In EDIT mode clicking Btn3 anywhere on the display brings up an editing menu. This menu has the following items. Except for the Object option, these are the same items as for the Edit menu on the main MEDM window.

Undo	Reverts to the last saved state of the display. The state is saved before most operations that would seriously change the display and also when Undo is chosen. It is not saved on keyboard editing operations. Repeated Undo's toggle between the current state and the last state, so you can pick the one you like best.
Object	This is another way to select an object to create besides using the Object Palette. There are three pull-right submenus that group the objects into Graphics, Monitors, and Controllers.
Cut	Deletes the selected objects and saves them in an internal clipboard. This operation can also be invoked by Shift-Del.
Copy	Copies the selected objects to the internal clipboard. This operation can also be invoked by Ctrl-Insert.
Paste	Pastes the objects in the internal clipboard into the display. They are selected and can be moved with the mouse or arrow keys until a mouse button is released or the Enter key is pressed. Afterwards they remain selected and can be moved using the means described above for moving selected objects. During the paste operation the objects are constrained to remain in the visible part of the display. The Motif Cancel key (usually Esc or Escape) cancels the paste. Paste supports snap to grid. If snap-to-grid is enabled, the arrow keys move the objects by one grid spacing. Otherwise, they move them by one pixel. Ctrl with the arrow keys moves the objects faster, and Shift with the arrow keys, faster yet. With the mouse they will snap to the nearest grid point. The paste operation can also be invoked by Shift-Insert.
Raise	Raises the selected objects in the display list. Note that graphics objects are drawn on the background of the display window and will always be below the other types of objects.
Lower	Lowers the selected objects in the display list. Note that graphics objects are drawn on the background of the display window and will always be below the other types of objects.
Group	Makes a composite object of the selected objects.
Ungroup	Decomposes the selected composite object into its components.
Align	Left: Align the left edges of all selected objects. Horizontal Center: Align the centers horizontally of all selected objects. Right: Align the right edges of all selected objects. Top: Align the top edges of all selected objects. Vertical Center: Align the centers vertically edges of all selected objects. Bottom: Align the bottom edges of all selected objects. Position to Grid: Align the upper left corner of all selected objects to the grid Edges to Grid: Align the edges of all selected objects to the grid. This may both move and resize the objects.
Space Evenly	Horizontal: Make the horizontal spacing between the selected objects equal to the grid spacing. The leftmost object will not move. Best applied to a row of objects. Does not affect object size.

	Vertical: Make the vertical spacing between the selected objects equal to the grid spacing. The topmost object will not move. Best applied to a column of objects. Does not affect object size. 2-D: Line up an array of objects. Best applied to objects already somewhat lined up in rows and columns. The objects will be aligned as well as spaced. The top left object will be at the upper left corner of the former bounding box of all the objects. The objects will be vertically spaced according to the grid spacing and the average height, and horizontally spaced according to the grid spacing. Does not affect object size.
Center	Horizontally in Display: Center the selected objects horizontally with respect to the display. If there is more than one object selected, the group is centered. Use Align in combination with Center to center each selected object. Vertically in Display: Center the selected objects vertically with respect to the display.
Orient	Flip Horizontally: Flip the selected objects horizontally about the midpoint. Flip Vertically: Flip the selected objects vertically about the midpoint. Rotate Clockwise: Rotate the selected objects clockwise about the midpoint. Rotate Counterclockwise: Rotate the selected objects counterclockwise about the midpoint. These operations in most cases change only the location of the objects. They do not change the orientation of what is in the objects. For example, text still reads from left to right and meters are never upside down. The internal orientation of Polylines, Polygons, and Arcs does change. Note that because of roundoff, the objects may creep if these operations are replied repeatedly. This tends to happen if the height or width is not an even number of pixels. In addition, objects are constrained to have coordinates greater than or equal to zero.
Size	Same Size: Make the height and width of all the selected objects the same and equal to the average height and width of the selected objects. Text to Contents: Make the sizes of selected text objects be just large enough to fit their contents. Only the horizontal position and the width of the enclosing box are changed. The appearance and position of the text itself should not change. Other types of selected objects are ignored, so this can be done for all text objects in the display by selecting everything first. Resizing the text objects should make alignment and spacing easier.
Grid	Toggle Show Grid: Toggle whether the grid is displayed or not. (This can also be done in the Resource Palette for the Display.) Toggle Snap to Grid: Toggle whether the Snap to Grid is in effect or not. (This can also be done in the Resource Palette for the Display.) When Snap to Grid is in effect, objects created or moved with the mouse will align their upper left corners to the grid. Grid Spacing: Brings up a dialog box to specify the grid spacing. The current spacing is shown. (This can also be done in the Resource Palette for the Display.)
Unselect	Unselects all objects.
Select All	Selects all the objects in the display.
Select Display	Selects the display. This can be used if it is not easy or possible to click on the display background. Note that when the display is selected, the x and y values of the display are updated and will be saved in the ADL file if the display is saved.
Find Outliers	Find objects that are totally or partially off the visible part of the display. (This can happen, for example, if the display is resized.) The number of such objects will be presented in a dialog box. If there are any objects, they will be selected, and there will

	be a button on the dialog box to list them on the MEDM Message Window. They can subsequently be moved or deleted if desired.
Refresh	Redraw the display. This will cause widgets that have been moved to be back in the correct stacking order.
Edit Summary…	Pops up a summary of pointer and key edit operations.

Execute-Mode Popup Menu

In EXECUTE mode there is a popup menu on each display that is invoked by pressing Btn3 anywhere on the background or on any object (*e.g.* the Cartesian Plot or Slider) that does not respond to Btn3 itself. This menu contains the following items:

Print	Prints the display. See Printing.
Close	Closes the display.
PV Info	Prompts you with a PV cursor to choose an object, then displays information about any process variables associated with that object. See PV Info under Dialog Boxes.
PV Limits	Prompts you with a PV cursor to choose an object, then displays a dialog box letting you change the limits for the process variable associated with that object. See PV Limits and Strip Chart Data Dialog under Dialog Boxes.
MEDM Main Window	Pops up the MEDM main window. This is useful if you cannot find it.
Display List	Brings up a dialog box with a list of the file names of all the displays currently open. You can Raise or Close the selected display. The name of the selected display will be in the clipboard, so it can be copied somewhere else. The See Display List Dialog Box under Dialog Boxes.
Toggle Hidden Button Markers	Marks any hidden buttons in the display by surrounding them with a flashing marquee border.
Refresh	Redraw the display.
Retry Connections	Causes MEDM to reissue search requests for unconnected PVs. This should not normally be necessary, as outstanding search requests will happen automatically whenever a server comes up. If MEDM is on a different subnet than the server, often the case when the server is a PV Gateway, then it may not know a server has come up, and this menu item may be useful to cause it to reconnect. It should be used with care as (1) it is a kludge and (2) it can cause unnecessary network traffic.
Execute	Brings up a sub menu with options that are determined by the MEDM_EXECUTE_LIST environment variable. See Execute Menu.

Printing

In either EXECUTE or EDIT mode a screen dump of the display can be printed by using the Print item on the main File Menu. If there is more than one display, you can choose the desired display with the special cursor that appears. In EXECUTE mode you can also use the popup menu invoked by Btn3 on a display. The print output is a picture of the display with optional date, time, and title above the picture. You can change several print options, including the print command and whether to print to a file, via the Printer Setup item on the File Menu.

MEDM includes defaults for the print options for most major platforms. If you prefer different defaults, they can be set in siteSpecific.h when MEDM is built. More information is available in the file. On UNIX systems

the supplied default print command uses the environment variable PSPRINTER. This allows you to use a different printer for MEDM than the one specified in the environment variable PRINTER. It must be a Postscript printer. The default print command gives the same behavior as older versions of MEDM, which used PSPRINTER with a fixed print command of "lp -c -d$PSPRINTER".

If the environment variable MEDM_PRINT_CMD exists, it will be used for the default print command. This allows you to customize the print command without rebuilding MEDM.

On the WIN32 platform, printing directly to the printer is not supported, but you can set the print command to use another program that can print or view Postscript, such as Ghostview. A good choice is the lpr command that comes with Exceed. This command is of the form:

```
lpr server-name postscript-printer-name user-name
e.g. lpr helios mcr2 jones
```

An alternative method of printing on WIN32 is to use Alt+PrintScreen to copy the window to the clipboard, and then paste the clipboard into an application, such as Paint, that can print it. You can also print to a file.

Execute Menu

The Execute Menu, accessed by pressing Btn3 in EXECUTE mode, is a configurable menu. The configuration is specified in the MEDM_EXEC_LIST environment variable. This variable is of the form:

```
name1; program1[[:name2; program2]…].
```

The items between colons represent each menu item. All of the characters up to the first semi-colon in each item are the text that appears on the menu. The remaining characters represent the system command that will be executed when the item is selected. Note that on UNIX, if the command is not run in the background (using & at the end), it will hold up MEDM until it finishes. On WIN32, if a colon is followed by a backslash (\), then it will be assumed that it is part of a path name and it will not be counted as a menu-item delimiter.

The system command can include the following special characters:

&P	The name(s) of the process variable(s). A PV cursor will appear to choose the object for which the process variable(s) are wanted. If there is more than one name, the names are space delimited.
&A	The full path name of the ADL file associated with the display.
&T	The short name of the file. (This should be the same as the title of the display.)
&X	The X window id of the ADL screen. This can be used with commands like Xwd, for example.
&?	Truncate the command at this point and prompt the user to finish or otherwise edit the command.

Examples are:

```
setenv MEDM_EXEC_LIST 'Probe;probe &P &:PV Name(s);echo &P'
setenv MEDM_EXEC_LIST 'Full Name;echo &A:Short Name;echo &T'
setenv MEDM_EXEC_LIST 'XTerm;xterm -fg black -bg white &'
setenv MEDM_EXEC_LIST 'XTerm;xterm &?'
setenv MEDM_EXEC_LIST ' Dump;xwd -id &X | xwdtopnm | pnmtops | lpr &'
```

A WIN32 example is:

```
set MEDM_EXEC_LIST=Probe;"c:\Program Files\EPICS WIN32
Extensions\probe.exe" &P:ADL File;echo &A:PV Names(s);echo &P
```

Color Rules

Color Rules are a means of changing the color of an object based on a calculation. They are potentially part of the Color Rule part of the Dynamic Attribute but are not currently supported. This capability can be accomplished by an animated image, however, as described in the section on the Image.

Color Conventions

The following color conventions should be used when creating displays:

Controls

Any control button, slider, menu, related display, text entry, etc. that makes something happen should be BLUE (second row blue in the MEDM default color palette). This is useful because operators can quickly observe "hot spots" on a display where they can make something happen (change a value, call up another display, etc). The context or icon of "the blue thing" should adequately convey WHAT will happen if they click on it.

Text

All text updates should be displayed on a background that allows all alarm colors to be visible (if the alarm dynamic is set). We usually use light gray or dark gray (almost black). Another "RULE OF THUMB" that we tell the operators is "If the text is WHITE, don't believe the reading. The computer cannot communicate to the process variable". Therefore, white static text is discouraged.

Likewise, since RED, GREEN, and YELLOW reflect alarm conditions, these colors should not be used for static text or text updates.

No colors are "disallowed" from being used for graphical images (magnets, waveguides, etc) because it is usually very obvious that these are pictures and NOT dynamic objects.

Colors in General

- DIN Standards 4844 and 5381 should be followed whenever possible.
- Red is a danger color. It means Halt, Stop, Prohibited, etc.
- Yellow is used as a warning color.
- Green means Safety, OK, On etc.
- Blue is used for giving directions, advice, signs, etc.

Environment Variables

MEDM uses the following environment variables:

EPICS_DISPLAY_PATH	A colon-separated (semi-colon-separated on WIN32) list of directories in which to look for display files that do not have absolute path names or names that begin with "./" or "../". Only looks in the current working directory if not specified. See ADL Files and Related Display.
MEDM_EXEC_LIST	A list of commands for the Execute-Mode Popup Menu. See the Execute Menu for the format.
MEDM_HELP	Name of the command used for the Help item in the Window Manager Menu for a display. This menu item does not appear if MEDM_HELP is not defined.
MEDM_HELP_URL	The URL where Netscape or another browser can find the HTML file used for all the menu and context-sensistive help in MEDM, except for that obtained from the Window Manager Menu. The default is specified in siteSpecific.h when MEDM is built.

MEDM_MAIL_CMD	Name of the command used to send mail from dialog boxes that support it. The default is “mail”.
MEDM_PRINT_CMD	Name of a command to be used to print in MEDM. The default is to use the MEDM default print command. See Printing.
NETSCAPEPATH	The name of the desired Netscape executable used for Help. It can be a full path name. The default is “netscape”.
PSPRINTER	Name of a Postscript printer that may be specified in the print command. See the Print Setup dialog box.

Building MEDM

MEDM is built with the EPICS build system. Explaining this system is beyond the scope of this manual. If you want to build MEDM, you should look at the EPICS documentation. The relevant manual is *IOC Applications: Building and Source/Release Control*, found under IOC software for the version of EPICS base that you are using. There are also comments in sections of this MEDM manual (Requirements, XRT/Graph, Sciplot, CDEV) on build parameters that are specific to MEDM.

MEDM has been successfully built on Solaris, Sun 4, HPUX, Linux, VMS, and WIN32.

CDEV Support

MEDM can be built to support CDEV rather than EPICS Channel Access. This capability has been provided by Jie Chen at the Thomas Jefferson National Accelerator Facility, and questions about the CDEV features of MEDM should be related to him at chen@jlab.org. CDEV provides an interface, not only to Channel Access, but also to any other network services that support CDEV at the expense of adding another layer to the control system. With CDEV support, you can specify messages in the form of "device verb attribute" where you would specify just a process variable name with Channel Access. In its simplest form, this message could be just the process variable name when interfacing to Channel Access. Apart from its supporting messages, the basic operation of MEDM is the same whether it is built for Channel Access or CDEV. ADL files that reference process variables by name should work with MEDM under CDEV. ADL files that use "device verb attribute" messages will not work with the non-CDEV MEDM.

In order to use MEDM with a CDEV service, the service has to support "get attribute" at least. If one wants to monitor a set of strings as choices, the service has to return a cdevData object with "displayHigh", "displayLow", and "value" tags in it. If there is not "displayHigh" and "displayLow", MEDM will not display data properly.

To build MEDM for CDEV, you need to define the makefile variable MEDM_CDEV. Setting it to YES is suggested.

Glossary

ADL Stands for ASCII Display List. The file format or the file that describes a display. The filename ends in .adl.

CDEV Stands for Control Device Interface. CDEV is a set of object-oriented software tools providing an interface to underlying systems, typically control system interfaces, such as Channel Access. CDEV has primarily been developed at the Thomas Jefferson National Accelerator Facility in collaboration with the EPICS community. The CDEV home page is http://www.jlab.org/cdev.

Channel Access A set of routines to provide network-independent access to process variables.

CDE Stands for the Common Desktop Environment. A technology specification developed jointly by Hewlett-Packard Company, IBM Corporation, Novell, Inc., and SunSoft, Inc. It defines a consistent set of application programming interfaces (API) that can be

implemented on operating environments that support X Windows desktop computers and OSF Motif. Its intent is to provide users with a consistent graphical user interface across workstations, X-Terminals, and PC's.

Display In MEDM a display is a window on the computer screen that contains objects that monitor or control process variables. This MEDM definition should be contrasted with the X Window's definition of a display, which is a set of one or more screens and a keyboard driven by a single X server, or the X Window DISPLAY environmental variable, which represents a particular screen on the X display.

EPICS Stands for Experimental Physics and Industrial Control System. EPICS is a set of software tools and applications originally developed by Argonne National Laboratory and Los Alamos National Laboratory for the purpose of building distributed control systems to operate devices such as Particle Accelerators, Large Experiments, and Telescopes. There is extensive information on EPICS available in the EPICS web pages.

EPICS Extension One of a number of programs that are not part of EPICS base but which have been designed to work intimately with EPICS. MEDM is an example. Most of the extensions are described in the online EPICS documentation for EPICS extensions.

HTML Stands for Hypertext Markup Language. An authoring language for Internet documents originally developed at CERN.

IOC Stands for Input/Output Controller. A VME/VXI based chassis containing a Motorola 68xxx processor, various I/O modules, and VME modules that provide access to other I/O busses. It contains a memory-resident database of control records.

Motif Motif is a complete set of widgets supplied by the Open Software Foundation (OSF) designed to implement the application look and feel specified in the *Motif Style Guide* and *Motif Application Environment Specification*. The routines are based on the Xt Intrinsics.

Process variable The name of a field of a record in a database, usually but not necessarily an IOC database. The name is of the form record.field. If the field is missing, it is assumed to be VAL. For example: S1A:H1:CurrentAO.SCAN.

URL Stands for Uniform Resource Locator. An identifier that specifies where a document is located on the Internet and the protocol used to exchange the document.

Object An MEDM object is one of several entities, such as a Meter, Related Display, Cartesian plot, etc., which can appear on an MEDM display.

WIN32 The Microsoft Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP operating systems.

Widget An interface component, such as a scroll bar, menu, or button on a user interface. Most MEDM objects are widgets and are children of the drawing area of the display. The others are drawn on the drawing area of the display.

Xt Intrinsics A basic set of routines for building and using widgets.

X Windows A network-based, graphics windowing system for workstations. It was developed by MIT and has become an industry standard.

Technical Support

If you have problems, comments, or questions about MEDM you can send them to evans@aps.anl.gov. If you wish to report a bug, it is essential that you send enough information for the bug to be reproduced. It would be helpful to include the MEDM version number, how MEDM was started, the circumstances in which the problem occurs, and an explicit list of steps to duplicate the problem. A copy of the ADL file or how it can be found is

usually helpful. The resources available for technical support are extremely limited. Please try hard to resolve problems yourself first and to prepare a complete and thought-out problem report otherwise.

Another source of help is the EPICS Tech-Talk forum described at http://www.aps.anl.gov/epics/tech-talk/index.php. This page includes an archive of articles and directions for joining the forum. You may present your problem and often get practical responses from the many users of EPICS and MEDM.

Copyright

Experimental Physics and Industrial Control System (EPICS)

Copyright, 1995-2006, The University of California, The University of Chicago

Portions of this material resulted from work developed under a U.S. Government contract and are subject to the following license: For a period of five years from March 30, 1993, the Government is granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, and perform publicly and display publicly. Upon request of Licensee, and with DOE and Licensors approval, this period may be renewed for two additional five-year periods. Following the expiration of this period or periods, the Government is granted for itself and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, distribute copies to the public, perform publicly and display publicly, and to permit others to do so.

NEITHER THE UNITED STATES NOR THE UNITED STATES DEPARTMENT OF ENERGY, NOR ANY OF THEIR EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.*

Document

Modified September 22, 2006