87214

Motif-based Display Editor/Manager (MEDM)

Operators Guide

Revised 11/18/94 by F. Vong

RECEIVED

FEB 14 1996

OSTI

- COMMAND LINE
- COMPOSITE PRODUCT
- MOTIF CONFORMANCE
- FONTS
 - DEFAULT, FIXED FONTS
 - DEFAULT, SCALABLE FONTS
 - USER SPECIFIED, SCALABLE FONTS
- ACCELERATORS and TRANSLATIONS
 - Main Display Popup Menu
 - Valuator/Scale
 - Text Entry/Text Field
 - Related Displays
 - Display Editing
- NEW FEATURES
 - IMAGE
 - SHELL COMMAND
 - CARTESIAN PLOT
 - MACRO SUBSTITUTION
 - POLYLINE AND POLYGON SUPPORT
 - DRAG-AND-DROP:
 - MEDM-SMART-STARTUP:
 - ASPECT RATIO-PRESERVING RESIZE:

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, 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. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED 85

Some operational hints about the use of the Motif-based display editor/manager (medm) are described here for the user. These are in some cases different from the conventions used in the Xlib-based dm/edd.

N.B. - Motif and other internalized applications must have proper locale and I18N environments setup. This includes, for X11R5, /usr/lib/X11/nls/ or the XNLSPATH environment variable set to the appropriate directory (usually .../lib/X11/nls under the X11R5 installed tree). Failure to do this will result in crashes of such applications!

COMMAND LINE:

MEDM is invoked as "medm" on the command line, with optional arguments for defining which mode to start the program in: edit (-e) or execute (-x), whether to allow remote display requests (-local if you don't want this executable to participate in remote display protocol: this feature is called "MEDM Smart Startup"), and an optional list of initial display files. (See COMPOSITE PRODUCT). Hence, the structure of the command line is:

```
medm [-x | -e] [-local | -cleanup] [-cmap] [-displayFont <font>]
      [-macro "<name>=<value>,<name>=<value>..." ]
      [-dg [<width>x<height>][+<x>+<y>]]
      [...X based options...] [display file names]

-x      startup medm in execute-only ("dm") mode
-e      startup medm in edit/execute ("edd/dm") mode (the default)
-local  don't participate in remote display protocol (default is
```

MASTER

```

        to participate and dispatch display requests to a remote
        MEDM if possible)
-cleanup    (seldom needed) support remote display protocol, but
            ignore existing MEDM and "take possession" of remote
            display responsibilities
-cmap       use private colormap to circumvent unallocable colors in
            default colormap (this may cause colormap flashing)
-displayFont <font> select aliased or scalable fonts,
            (see "Fonts" section below)
-macro "name=<value>,..." apply macro substitution as specified in
            the name=value... string to all command-line specified
            display list files (this option requires -x also)
-dg [width x height][+<x>+<y>]]
            specify the location and size of the display

*.adl       a list of ascii display list files

```

Example:

```

medm -x abc.adl def.adl      # starts up executing two displays
medm -e xyz.adl              # starts up editing one display
medm                         # starts up in default (edit) mode
medm -local                  # starts up in default (edit) mode
                             # and starts up new local executable
                             # which doesn't participate in remote
                             # display protocols
medm -local -x abc.adl       # starts up in execute mode, and does
                             # not take advantage of or support
                             # remote display protocol
medm -displayFont scalable   # starts up using default scalable
                             # fonts (medm default is aliased)
medm -x -macro "a=b,c=d" t1.adl # starts up in execute mode, performing
                             # macro substitution on all occurrences
                             # of $(a) and $(c) in display file
                             # t1.adl
medm -x -dg 100x100+100+100 t1.adl
                             # starts up in execute mode, resize the
                             # the display t1.adl to 100 by 100 pixels
                             # and move the display to location x = 100
                             # and y = 100

```

N.B. The usual X/Xt oriented command line arguments are supported also, by default

COMPOSITE PRODUCT:

MEDM performs the functions of both dm and edd

Once medm is running, displays are opened via selecting File -> Open... from the main menu and selecting the desired ADL (*.adl) file from the dialog box. The file is opened in either EDIT or EXECUTE mode, based on the selected mode in the main window. These toggle buttons allow the user to alternately go between EDIT and EXECUTE for all opened displays. When in EXECUTE mode, most EDIT functions are not available for selection, and normal EXECUTE semantics are provided.

MOTIF CONFORMANCE

MEDM conforms 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 (On Context...).

FONTS:

MEDM uses several methods for font specification. The flag

`-displayFont`

with one of {alias, scalable, } allows users to select the default (aliased names), default scalable, or user-specified scalable fonts. The default is `-displayFont alias`.

DEFAULT, FIXED FONTS:

For default, fixed fonts, MEDM uses "logical font names" internally, to deal with server font variations. Consequently, font aliases are necessary for the various servers on which medm will run. For the SUNs, the file fonts.alias.sun should be copied into a common (or user specific) font area as fonts.alias, and the font path for the server set to include that directory. The default (no `-displayFont` specified) is the same as: `% medm -displayFont alias` For example, a user may create an Medm directory, copy medm and fonts.alias.sun into that area, and then move (cd) to that directory. The user may then `% mv fonts.alias.sun fonts.alias % xset +fp $cwd/` To verify that the fonts are included, do a `% xset -q` to see that the font path is setup appropriately, and do a `% xlsfonts | grep widgetDM` to verify that the logical font names are resolved.

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:7000
% medm -displayFont scalable
```

USER-SPECIFIED, SCALABLE FONTS:

The user can invoke MEDM with the `-displayFont` option. MEDM then uses the specified font supplied by the X11R5 font server. 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
```

ACCELERATORS and TRANSLATIONS:

There are several built-in accelerators or translations for mouse and keyboard events in the Motif interface built with the Motif-based display manager.

Prominent among these are:

Main Display Popup Menu

The main display popup menu (with Print and Close options for the current display) is popped-up by depressing MB3 in nearly any location on the display window. One of these options is selected by releasing MB3 when the desired item is under the cursor and "rasied".

Valuator/Scale

The valuator/scale object has several modes of operation, which implement fine/coarse sensitivity, as well as direct value selection.

Dragging the valuator/scale with MB1 depressed moves and transmits values proportional to the range of the valuator scaled into the width of the valuator (usually $\geq 1\%$ for this application, depending upon the selected width of the valuator). Clicking MB1 on either side of the valuator selector moves and transmits a value increment or decrement of specified precision (for this application) of the valuator object. Clicking Control-MB1 on either side of the valuator selector moves and transmits a value increment or decrement of 10x specified precision of the valuator object.

In addition to the normal mouse-activated mode of usage for the valuator/scale, the arrow keys and shift key take on functions as well. Fine-grained Increment and Decrement are accomplished with the up/down or left/right arrow keys, depending upon the orientation of the valuator/scale. The slider will move in the direction "pointed to" by the arrow key being pressed (as expected), when input focus is set to that widget. For instance, will move the slider by +1 precision unit when the orientation is HORIZONTAL and processingDirection is MAX_ON_RIGHT.

In conjunction with increment/decrement, the key can be used to specify a coarse-grained increment/decrement. Hence, , for instance, will move the slider by +10 precision units when the orientation is VERTICAL and processingDirection is MAX_ON_TOP.

The specified precision for motion of the valuator is specified in a dialog box which is popped up by depressing MB3 in the valuator/scale. A series of toggle buttons with values of $\log_{10}(\text{precision})$ are selectable, with the current precision indicated by the depressed toggle.

Also, for the highest precision modification of associated process variables, the keyboard entry dialog box can be used to specify a direct-entry value to be written to the valuator and channel.

Text Entry/Text Field

Several edit modes are supported for the text entry or text field widget. In addition to point-click positioning, the left and right arrow keys allow the cursor to be positioned anywhere in the field. Similarly, backspace and delete allow characters to be removed from the string/field. Carriage Return () sends the value; additionally, leaving the widget also transmits the current value back to the application.

Related Displays

As in the Xlib-based display manager, the environment variable EPICS_DISPLAY_PATH should contain the directory where related displays (for a given display) can be located.

Display Editing

Based on the selected button in the object palette, the editor can be in CREATE or SELECT mode. These modes determine the semantics of button presses in an active display.

When the select button is depressed in the object palette, the editor is in SELECT mode and the following semantics apply:

MB1:	select an object (or objects) on the screen and highlight. the resource palette is updated to reflect the selected item's internal data. a drag operation under MB1 selects a group of
------	---

objects on the screen (including those objects wholly bounded by the selection rectangle).

Shift-MB1: multiple-select. a set of objects are selected for operations (such as grouping).

MB2: selected object(s) are moved while MB2 is depressed and deposited on button release. if no objects are currently selected, the object under the cursor when MB2 is depressed is made the current object for moving. to cancel the effect of the current move, the cursor may be dragged off the current display window and the button released. this cancels the effect of the move.

Ctrl-MB2: selected object(s) are resized while MB2 is depressed. if no objects are currently selected, the object under the cursor when MB2 is depressed is made the current object for resizing. to cancel the effect of the current resize, the cursor may be dragged off the current display window and the button released. this cancels the effect of the resize.
N.B. this mechanism performs ABSOLUTE resizing, in which all objects are extended in width and height by the magnitude of the x and y mouse motion. for PROPORTIONAL resizing in which selected objects resize consistently, the object must first be grouped.

MB3: popup applicable menus.

When an object (e.g. rectangle) button is depressed in the object palette, the editor is in CREATE mode and the following semantics apply:

MB1: an object of current type is created, starting at the origin of button press, of size determined by button release (a bounding rectangle is rubberbanded). the resource palette is updated to reflect the object's internal data.

MB2: (as in SELECT mode above)
selected object(s) are moved while MB2 is depressed and deposited on button release. if no objects are currently selected, the object under the cursor when MB2 is depressed is made the current object for moving. to cancel the effect of the current move, the cursor may be dragged off the current display window and the button released. this cancels the effect of the move.

Ctrl-MB2: (as in SELECT mode above)
selected object(s) are resized while MB2 is depressed. if no objects are currently selected, the object under the cursor when MB2 is depressed is made the current object for resizing. to cancel the effect of the current resize, the cursor may be dragged off the current display window and the button released. this cancels the

effect of the resize.

N.B. this mechanism performs ABSOLUTE resizing, in which all objects are extended in width and height by the magnitude of the x and y mouse motion. for PROPORTIONAL resizing in which selected objects resize consistently, the object must first be grouped.

MB3: (as in SELECT mode above)
popup applicable menus.

For fine-tuning placement of objects when editing, use the arrow keys (up/down/left/right) to move selected objects in the display one pixel at a time in the direction of the arrow. Note that many widgets trap input events, therefore the cursor should not be positioned over a widget when this motion is being requested. Input focus must be on the display (or its top level shell) for this mechanism to work.

(Implementation Note – if an event handler was used instead of the drawingAreaCallback this could possibly be circumvented).

NEW FEATURES

:

IMAGE

– images can be imported for inclusion in a display. The image icon (camera) can be selected, and the location and size of the desired image then selected by depressing and dragging MB1 in the display. A file selection box then prompts the user for the display file to be incorporated. At the present time, GIF is the only supported format, and files must have the ".gif" suffix.

SHELL COMMAND

– shell commands can be incorporated into a display by selection of the "!" icon, followed by MB1 depression and dragging in the display. The labels and commands (and optional arguments) are then specifiable in the "Label/Cmd/Args" dialog box. Note: for prompted input, a question mark "?" in the command or args field instructs the program to popup a dialog to allow the user to complete the command string before execution. Also note: the shell command blocks by default upon execution. To support asynchronous behavior, the command or argument should be followed by the ampersand "&" to instruct the system to run the command in the background. All stdout/stderr output, at this time, is still tied to the controller terminal window. Also note that shell command and argument strings can be macro-substituted via the \$(name) construct (see related display description in this document).

CARTESIAN PLOT

– cartesian plot data will utilize up to 2 Y axes for display. Trace 0 will use the left Y axis, traces 1–7 (if applicable) will use the right Y axis. Hence, to plot power vs thermocouple values, trace 0 could be power, and traces 1–7 could be the thermocouples (since these will probably all have the same scale, sharing the Y axis is preferable).

Cartesian Plots can be made of scalars, scalar vs. scalar, vector, vector vs. scalar, and vector vs. vector. For "incomplete" data such as vector (vs. nothing) the other independent variable will be filled in with an index (element position number). Vector vs. scalar allows users to simulate display of error bars by displaying a set (vector) of data at a point (similarly, scalar vs. scalar with one scalar value fixed and the other varying, with a count greater than 1 and erase oldest ON can show "history" of values at a point). Vector vs. vector supports "ordered pair plots" where users can supply an X vector of data and a Y vector

of data, with the resultant plot being (x[0],y[0]), (x[1],y[1]), etc.

MACRO SUBSTITUTION

– MEDM now supports macro substitution (including nested, or parent-to-child substitution) for related displays.

Related displays can be called with an arguments string of the form "name=value[,name=value,...]". The related display then substitutes in its space all occurrences of "\$ (name)" with "value".

A parent related display can pass to its child related display a value in its space by passing an argument string of the form "name=\$(name)XYZ", etc.

Shell command and argument strings are also substituted in related displays, and referenced similarly via the \$(name) construct.

POLYLINE AND POLYGON SUPPORT

– polylines and polygons are available from the object palette, in both "constrained" and "unconstrained" form. Unconstrained drawing allows the user to click (MB1) and generate vertices for a polyline or polygon arbitrarily. SHIFT-MB1 allows the user to generate vertices which are multiples of 45 degrees from the previous point (this allows easy horizontal or vertical line generation, for instance).

Note that polygons in fill mode "outline" are similar to polylines, except that polygons will close the figure and polylines do not.

Also note that Vertex editing is now available for Line, Polyline and Polygon objects. Selection of vertices of selected objects via MB1-press and subsequent drag until MB1-release allows vertices to be moved.

DRAG-AND-DROP:

MEDM, in execute mode, allows via the drag-and-drop mechanism (MB2), the deposition of controller object process variable names onto other control objects or programs. For example, depressing MB2 on a controller (e.g., valuator) will retrieve the underlying channel's name, and allow it to be dropped onto other objects (e.g., text fields in the same or other Motif applications) or other programs (e.g., KM – the knob manager program, which for instance, can then assign that channel to the specified knob and initiate physical control).

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", a remote MEDM request protocol has been established.

MEDM by default now looks for other MEDM's running (on ANY machine) which have the same display device as the requested display. If one is found, the display request is forwarded to the remote MEDM for processing. This saves startup time of several MEDM processes which are all going to the same display.

To override this behavior, use the "-local" parameter on the command line. MEDM invoked with -local (in either edit or execute mode) will not participate in the "smart-startup" protocol and act independently of other MEDMs. (The default is not -local, i.e., DO participate in smart-startup protocol).

N.B. – this request protocol does not work when the two machines running MEDM do not share a file namespace. Also note that this obviates the MEDM startup library and application messaging scheme

which has been discussed.

MEDM should always be stopped via the File->Exit menu selection. If MEDM is killed via a SIGKILL or SIGSTOP signal, some X window property cleanup may be required. If MEDM is invoked, makes a remote display request and returns (very quickly) but no display is ever created, then do the following:

```
% xprop -root | grep MEDM          /* see if an MEDM property is stored */
If there is one, then do the following:
% xprop -root -remove MEDM_EDIT_REQUEST
or
% xprop -root -remove MEDM_EXEC_REQUEST
depending on which mode MEDM you are requesting (medm or medm -x).
```

Alternatively, medm can also be started up in "cleanup mode", via
% medm -cleanup ...

This starts up MEDM in a mode which ignores remote MEDMs, and does the property X property cleanup on exit.

ASPECT RATIO-PRESERVING RESIZE:

MEDM in execute mode allows run-time resizing of displays. Since the user is totally free to pick sizes to his liking, the aspect ratio of the original display can be severely distorted. Users wishing to preserve the aspect ratio of a display at run-time can do a SHIFT-modified resize. To do this, the user invokes the resize of the display (window) as appropriate for the window manager in use, with the Shift key depressed at the time of completion of the resize operation. Note that the resizing is not constrained at the time of resize, but upon completion the display will be resized to ("snap to") the appropriate sizes of the width and height dimensions. Both reducing and enlarging of the display is supported.