best

# **N**on-Procedural
# **O**bject
# **M**odule
# **A**pplication
# **D**evelopment
# **S**ystem

## Study Manual

Version 4.11
September 2001

# PREPARATION

1. Create a new directory **“\NOMTRAIN”**

2. Create a new shortcut to run **PVXWIN32.EXE** in this directory

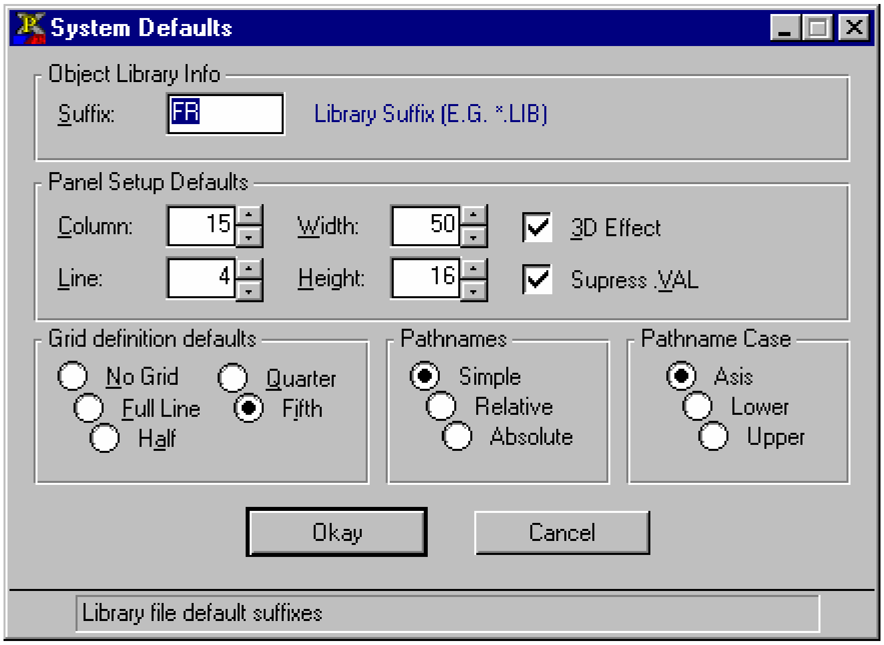
3. **RUN "*NOMADS"**

Menu options: LIBRARY / DICTIONARY / SECURITY / OPTIONS / HELP / QUIT

- Library
  - New
  - Open
  - Bulk edit
  - Compare
  - Merge
- Dictionary
  - Maintenance
  - Classes
- Security
  - Classifications
  - Users
- Options
  - System Defaults
  - Message Manager
  - Group Assignment
  - Change Directory

# System Defaults

System Defaults are default parameters/settings that are to be used for each new library created.These settings can later be overridden for each library through the Library Defaults screen. The System Defaults are stored in the ProvideX keyed file **‘PROVIDEX.NMD’**.



Object Library Info

Suffix: FR   Library Suffix (E.G. *.LIB)

Panel Setup Defaults

Column: 15   Width: 50   ☑ 3D Effect

Line: 4   Height: 16   ☑ Supress .VAL

Grid definition defaults

○ No Grid   ○ Quarter
○ Full Line   ● Fifth
○ Half

Pathnames

● Simple
○ Relative
○ Absolute

Pathname Case

● Asis
○ Lower
○ Upper

Library file default suffixes

# System Defaults (cont'd)

**Object Library Info:**

Suffix: The default suffix for any new libraries created (e.g. Suffix = EN, all libraries created will be: *library_name*.EN)

**Panel Setup Defaults:**

Column/Line: The top left position of any new panels created

Width/Height: The width and height of all new panels created

3D Effect: If the 3D effect indicator is on, all controls for all objects will be presented using 3D style controls, unless overridden in the library defaults.

Suppress .Val: The value of each control on a screen is automatically placed into the variable CTL_NAME.VAL$ (or CTL_NAME$ if the .VAL is suppressed) For example, if a screen has a list box defined as LBOX_1 then the value within this control will be found in the variable LBOX_1.VAL$ (or LBOX_1$ if the .VAL is suppressed)

**Grid Definition Defaults:** When a control is being sized, drawn or moved, the grid definition defines the minimum increments allowed for the height of a control when using the mouse or the resize option within NOMADS.

If the grid definition is set to 'Full Line', then the height of the control will automatically adjust or 'snap-to' one full line high when drawn. The control can not be less than one full line high and must be resized in increments of 1 full line (ie. 2, 3, 4, etc.). If the Grid definition is set to 'Half', then the height of the control will 'snap-to' a half line high. The control can not be less than a half line high and must be resized in increments of .5 (ie. 1,1.5, 2 etc.). However, the sizing can be overridden from the Edit menu bar option in the NOMADS designer screen or through the properties screen on each control.

**Pathnames:** This selection will determine whether the pathname will be written to the file as Simple: ABC, Relative: .\ABC, or Absolute: C:\ABC

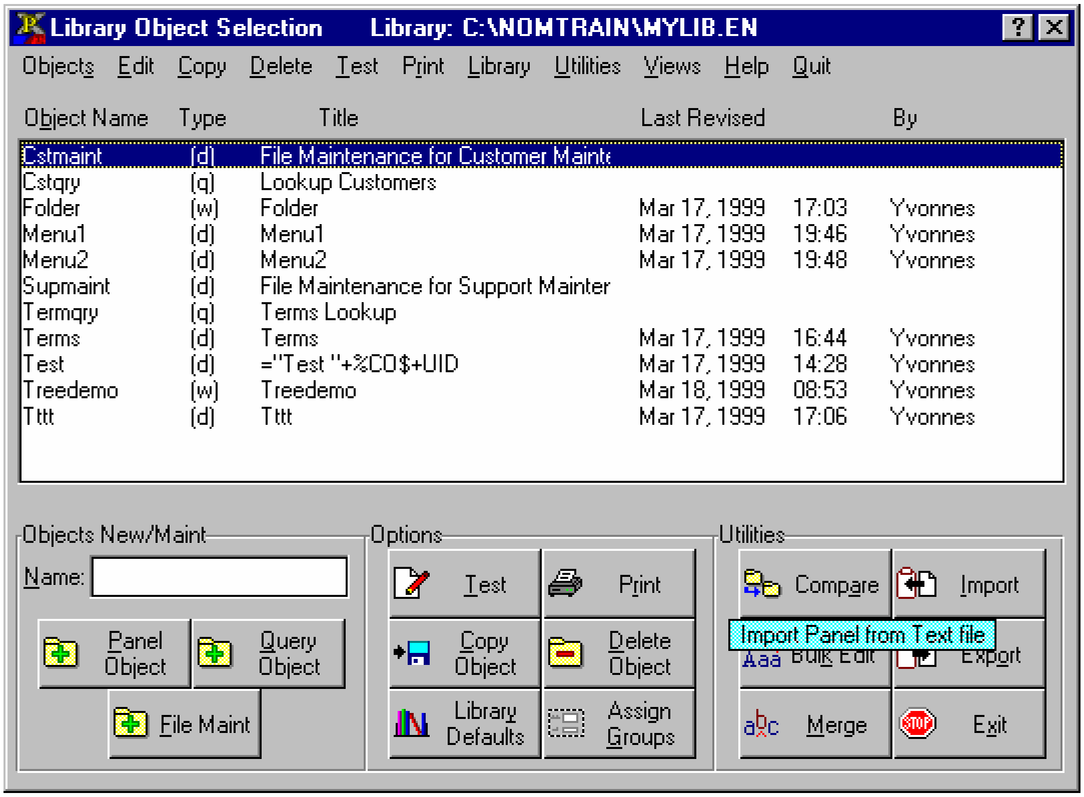
**Pathname Case:** This selection will determine if the pathname will be stored ASIS, Uppercase or lowercase.

# Creating A New Library And Setting Library Defaults

1. Select the **L**ibrary**/N**ew options from the NOMADS **II** Menu Bar

2. Create a new library called**: MYLIB.EN**

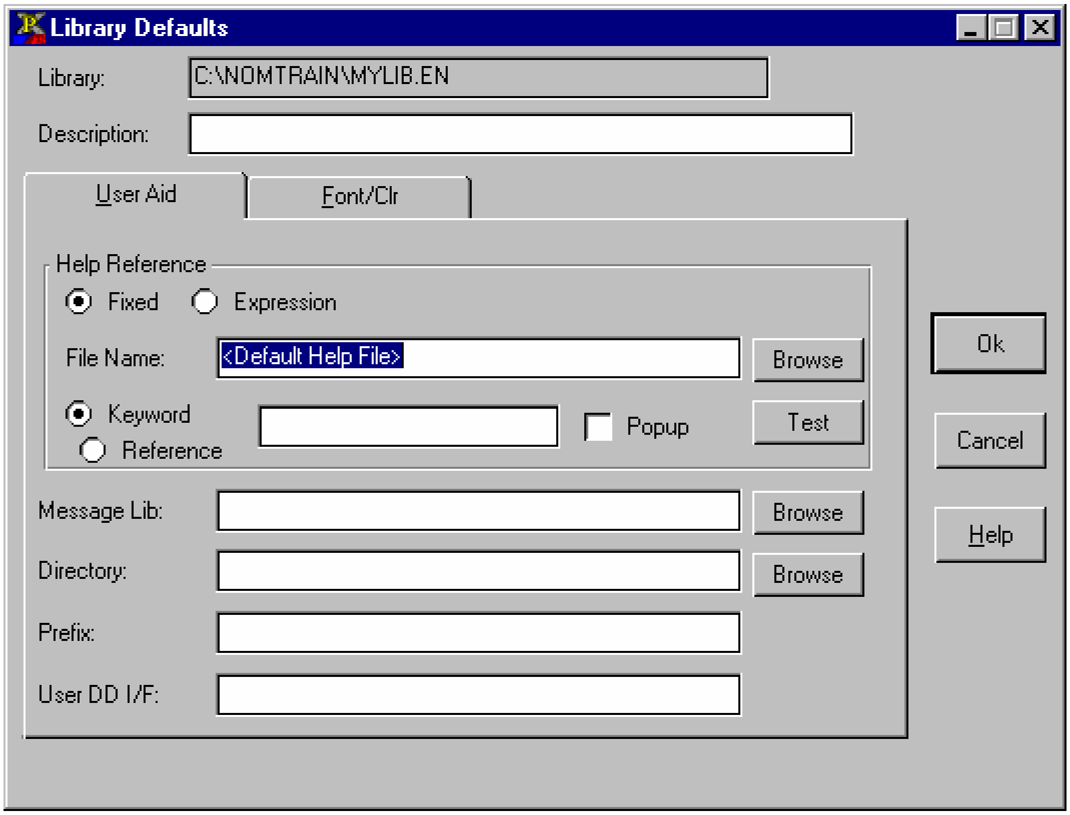
3. Select the ‘Library Defaults’ button



| Object Name | Type | Title | Last Revised | | By |
|---|---|---|---|---|---|
| Cstmaint | (d) | File Maintenance for Customer Mainte | | | |
| Cstqry | (q) | Lookup Customers | | | |
| Folder | (w) | Folder | Mar 17, 1999 | 17:03 | Yvonnes |
| Menu1 | (d) | Menu1 | Mar 17, 1999 | 19:46 | Yvonnes |
| Menu2 | (d) | Menu2 | Mar 17, 1999 | 19:48 | Yvonnes |
| Supmaint | (d) | File Maintenance for Support Mainter | | | |
| Termqry | (q) | Terms Lookup | | | |
| Terms | (d) | Terms | Mar 17, 1999 | 16:44 | Yvonnes |
| Test | (d) | ="Test "+%CO$+UID | Mar 17, 1999 | 14:28 | Yvonnes |
| Treedemo | (w) | Treedemo | Mar 18, 1999 | 08:53 | Yvonnes |
| Tttt | (d) | Tttt | Mar 17, 1999 | 17:06 | Yvonnes |

# Library Defaults Properties

Library Defaults are default parameters/settings that are to be used for each new panel created.These settings can later be overridden for each panel through the **Panel Header** Definition screen.



Library Defaults

Library: C:\NOMTRAIN\MYLIB.EN

Description:

Help Reference

● Fixed ○ Expression

File Name: <Default Help File>

● Keyword ○ Reference □ Popup

Message Lib:

Directory:

Prefix:

User DD I/F:

# Library Defaults Properties (cont'd)

## USER AID OPTIONS

**Library:** This is a locked field and will reflect the path to the current library file.

**Description:** A user defined description may be entered in order to identify the library

**Help Reference:**

File Name: Fixed: This field would contain the name of the windows help file

Example: NOMADS.HLP

Expression: The expression option indicates to NOMADS
that the help file to be displayed is stored in a string variable or simple expression

Example 1: "LREF.HLP;INTRODUCTION"
This example will display the language reference help file in an independent window and will be positions at the Inroduction section.

Example 2: "LREF.HLP;#123"
This example will display the language reference help file in an independent window (indicated by '#' ) and will be positioned at the context string value 123 which is the SET_FOCUS directive

Example 3: "LREF.HLP;$123"
This example will display the language reference help file in a popup (indicated by '$' ) and will be positioned at the context string value 123 which is the SET_FOCUS directive

Keyword: This is the index entry for the topic in the indicated help file.

Example: "LREF.HLP;INTRODUCTION"
INTRODUCTION is the keyword (or index entry) in the language reference help file.

Reference: This is the reference number for a specific topic in the context sensitive help.

Example: "LREF.HLP;#123"
123 is the context string (reference number) for the SET_FOCUS topic in the language reference help file.

Popup: When this option is selected the help topic will be display
as a popup rather than an independent window.

**Message Lib:** This field will contain the name of the default Message Library to be used when processing Panels.

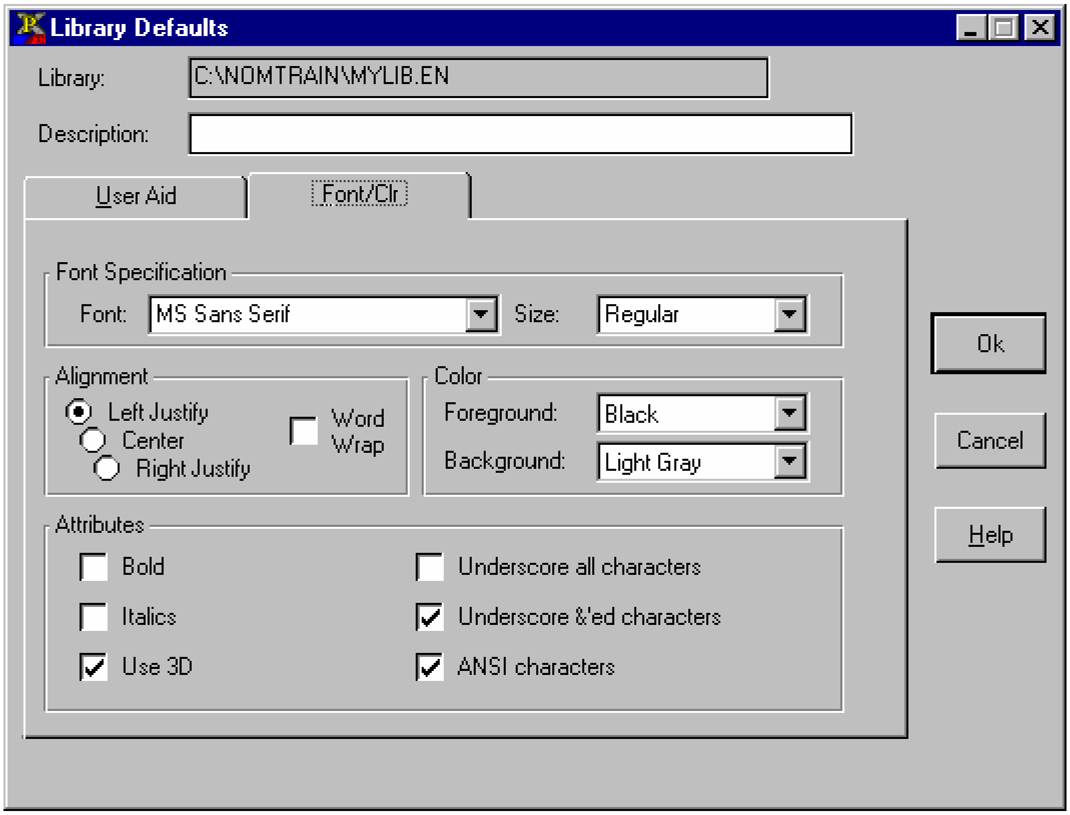
**Directory:** Before processing a panel NOMADS will change directory to the one defined in this field. When the panel is closed, NOMADS will switch back to the original directory

**Prefix:** Before processing a panel NOMADS will change the ProvideX Prefix to the one defined in this field. When the panel is closed, NOMADS will change back to the original Prefix.

# Library Defaults Properties (cont'd)

1. Set Defaults (Colour, Font, Default Help, 3D)

2. Record Changes (Return to Library Object Selection screen)



Library Defaults

Library: C:\NOMTRAIN\MYLIB.EN

Description:

Font Specification

Font: MS Sans Serif

Size: Regular

Alignment

● Left Justify

○ Center

○ Right Justify

□ Word Wrap

Color

Foreground: Black

Background: Light Gray

Attributes

□ Bold

□ Italics

☑ Use 3D

□ Underscore all characters

☑ Underscore &'ed characters

☑ ANSI characters

# Library Defaults Properties (cont’d)

## FONT/CLR OPTIONS

**Font Specification:** The user can select the default font and font size to be used on all panels within the library. This can be overridden in the panel header or on the individual controls.

**Alignment:** This is the alignment for the font to be used on all panels within the library. This can be overridden in the panel header or on the individual controls.

**Colour:**

Foreground: This is the default colour for all text on all panels within The library. This can be overridden in the panel header or on the individual controls.

Background: This is the default background colour of all panels within the library. This can be overridden in the panel header.

**Attributes:**

Bold: All text on all panels will be **Bolded**. This can be overridden in the panel header or on the individual controls

Italics: All text on all panels will be *Italicized* . This can be overridden in the panel header or on the individual controls

Use 3D: If the 3D effect indicator is on, all controls for all objects will be presented using 3D style controls, unless overridden in the library defaults.

Underscore all characters: All text on all panels will be Underlined. This can be overridden in the panel header or on the individual controls

Underscore all &'ed characters: When defining a control with text or Fonted Text if the user places an '&' within the text, NOMADS will underscore the letter following the '&'. This is used when defining hotkeys.

E.g. &Test results in Test

ANSI Characters: When this option is unselected or turned off, NOMADS allows you to access other character sets such as Wingdings

E.g. Test results in ❄☜💧❄

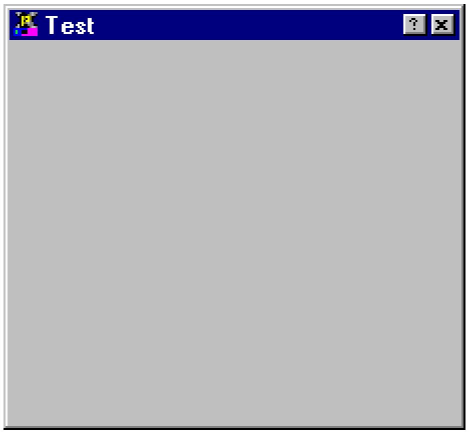
# Panels And Controls

A panel/form object is basically a screen layout, consisting of a series of controls.

1) Create a ‘Panel Object’ called **TEST**

2) Enter **TEST** in the ‘Name’ input field and press the button marked ‘Panel Object’

**Sample Results:**



Each panel consists of a variety of controls that can be used to display information, PERFORM / EXECUTE a piece of code, CALL another program or invoke another panel.

NOMADS currently supplies the following controls:

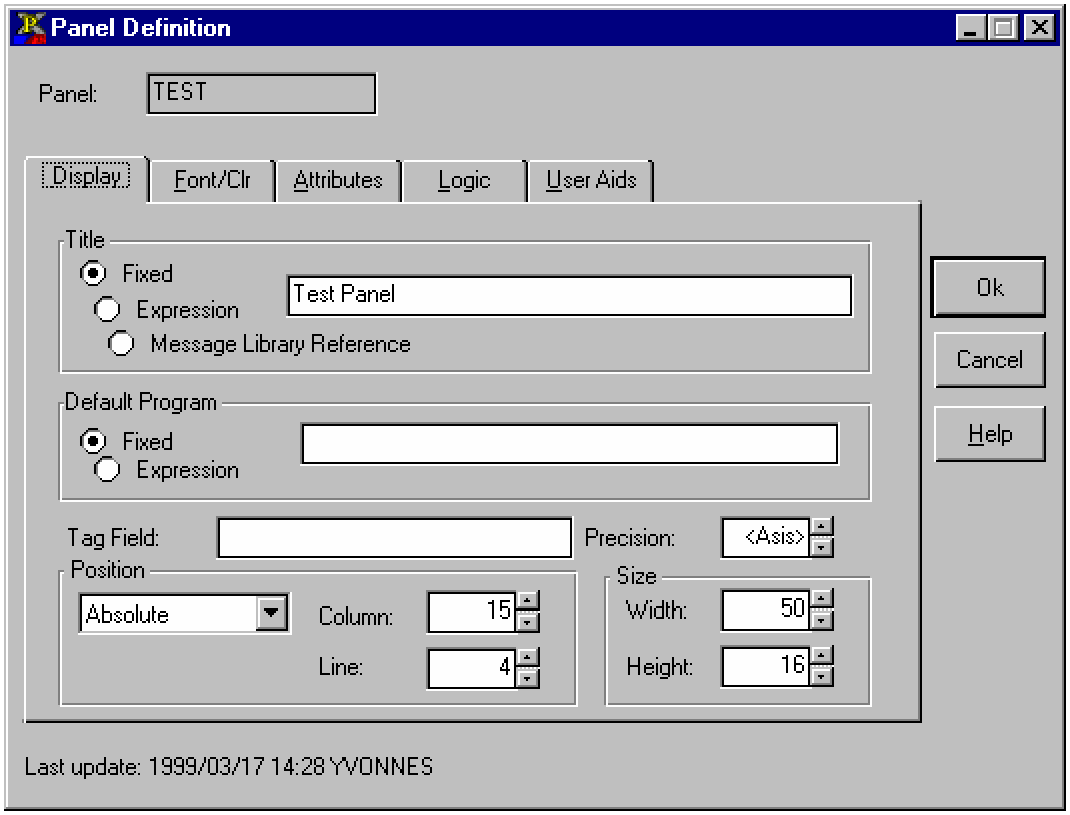
Button
Radio Button
Check Box/Tristate (states: 0,1,2)
List Box
Drop Box
Multi Line
Standard Text
Fonted Text
Image (‘PICTURE’ mnemonic…c:\bmps\picture.bmp)
Boxes
Vertical Scrollbar
Horizontal Scrollbar
Tab Selector (Folder)
External Tools (VBX’s)

# Panel Header Properties

The Panel Header contains the defaults for each individual panel. These settings can only be overriden by the settings on each individual control.

1. Select the ‘Header’ option from Menu Bar (**P**anel/**H**eader) or Press the ‘Head’ button from the controls toolbar

2. Title: **Test Panel**



Panel Definition

Panel: TEST

Title
● Fixed Test Panel
○ Expression
○ Message Library Reference

Default Program
● Fixed
○ Expression

Tag Field:
Precision: <Asis>

Position
Absolute
Column: 15
Line: 4

Size
Width: 50
Height: 16

Last update: 1999/03/17 14:28 YVONNES

# Panel Header Properties (cont'd)

## OPTIONS

**Panel:** This is a locked field containing the name of the panel.

**Title:** The fixed title option will display the title as entered. The expression option indicates to NOMADS that the title to be displayed is stored in a string variable or simple expression. The expression will be evaluated when the panel is created.

**Default Program:** By entering the program name that contains all the statement labels that the Logic folder on the panel and controls refers to, all that is required in the logic fields is a ';' and the statement label

Example:
Default Program: PGM01
Pre-Display logic on panel header: PERFORM ";INIT_STUFF"
Logic that will be executed: PERFORM "PGM01;INIT_STUFF"

If the user wishes to execute a statement label in a different program, simply include the name of the program with the statement label

Example: PERFORM "PGM123;INIT_STUFF"

If a program name does not exist the default will be assumed.

**Tag Field:** Currently used by the File Maintenance System. This field will contain the name of the physical data file assigned to the File Maintenance panel.

**Precision:** ProvideX defaults to a precision of 2 decimal places. This can be overridden by setting the Precision in the Panel Header.

**Position:** The position has 3 options:

Absolute: This indicates that the panel is to be positioned on the screen at exactly the line and column defined.

Example:

| | |
|---|---|
| Position of Main Window: | Line 5, Column 5 |
| Panel defined with a position of: | Line 2, Column 2 |
| Panel will be displayed at: | Line 2, Column 2 |

Relative: This indicates that the panel is to be positioned on the line and column defined but relative to the main window.

Example:

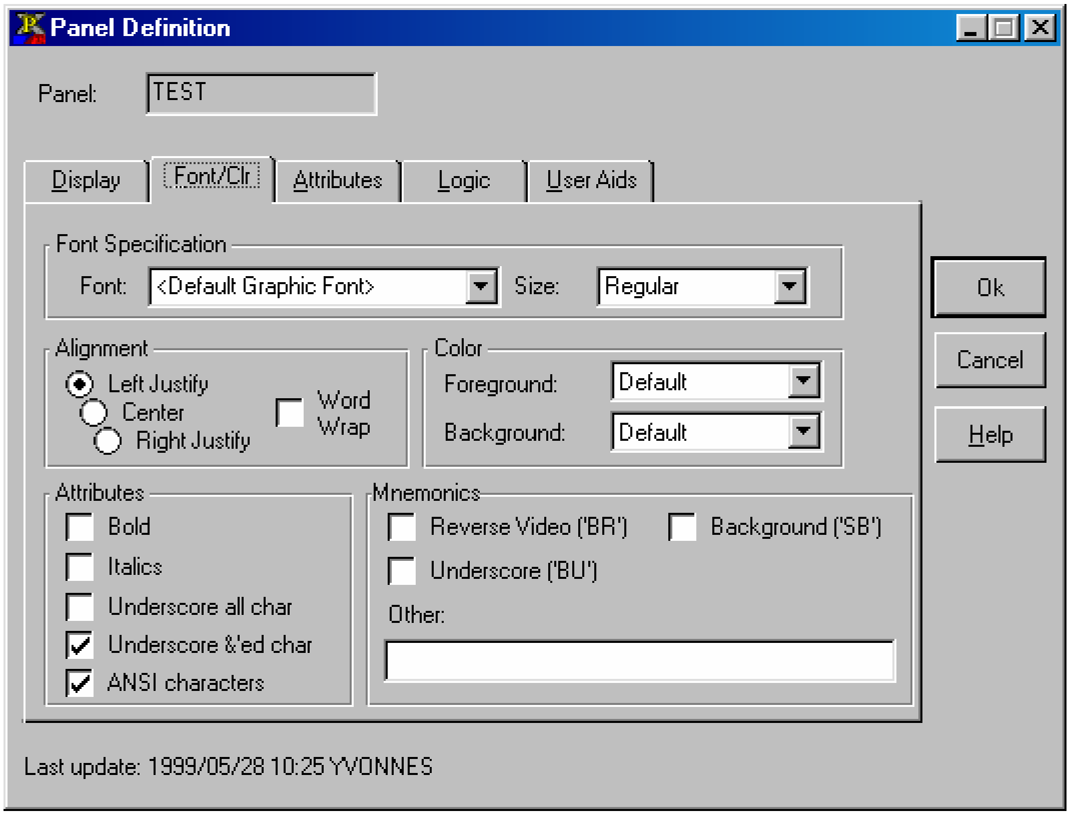
| | |
|---|---|
| Position of Main Window: | Line 5, Column 5 |
| Panel defined with a position of: | Line 2, Column 2 |
| Panel will be displayed at: | Line 7, Column 7 |

Center: This indicates that the panel is to be centered relative to the main window.

**Column/Line:** The top left position of the panel

**Width/Height:** The width and height of the panels

# Panel Header Properties (cont'd)



Panel: TEST

Font Specification

Font: <Default Graphic Font> Size: Regular

Alignment

● Left Justify
○ Center
○ Right Justify
□ Word Wrap

Color

Foreground: Default

Background: Default

Attributes

□ Bold
□ Italics
□ Underscore all char
☑ Underscore &'ed char
☑ ANSI characters

Mnemonics

□ Reverse Video ('BR') □ Background ('SB')
□ Underscore ('BU')

Other:

Last update: 1999/05/28 10:25 YVONNES

# Panel Header Properties (cont’d)

## FONT/CLR OPTIONS

**Font Specification:** The user can select the default font and font size to be used on all panels within the library. This can be overridden in the panel header or on the individual controls.

**Alignment:** This is the alignment for the font to be used on all panels within the library. This can be overridden in the panel header or on the individual controls.

**Colour:**

Foreground: This is the default colour for all text on all panels within the library. This can be overridden in the panel header or on the individual controls.

Background: This is the default background colour of all panels within the library. This can be overridden in the panel header.

**Attributes:**

Bold: All text on all panels will be **Bolded**. This can be overridden in the panel header or on the individual controls

Italics: All text on all panels will be *Italicized* . This can be overridden in the panel header or on the individual controls

Underscore all characters: All text on all panels will be Underlined. This can be overridden in the panel header or on the individual controls

Underscore all &'ed characters: When defining a control with text or Fonted Text if the user places an '&' within the text, NOMADS will underscore the letter following the '&'. This is used when defining hotkeys.
E.g. &Test results in Test

ANSI Characters: When this option is unselected or turned off, NOMADS allows you to access other character sets such as Wingdings
E.g. Test results in ❄☜💧❄

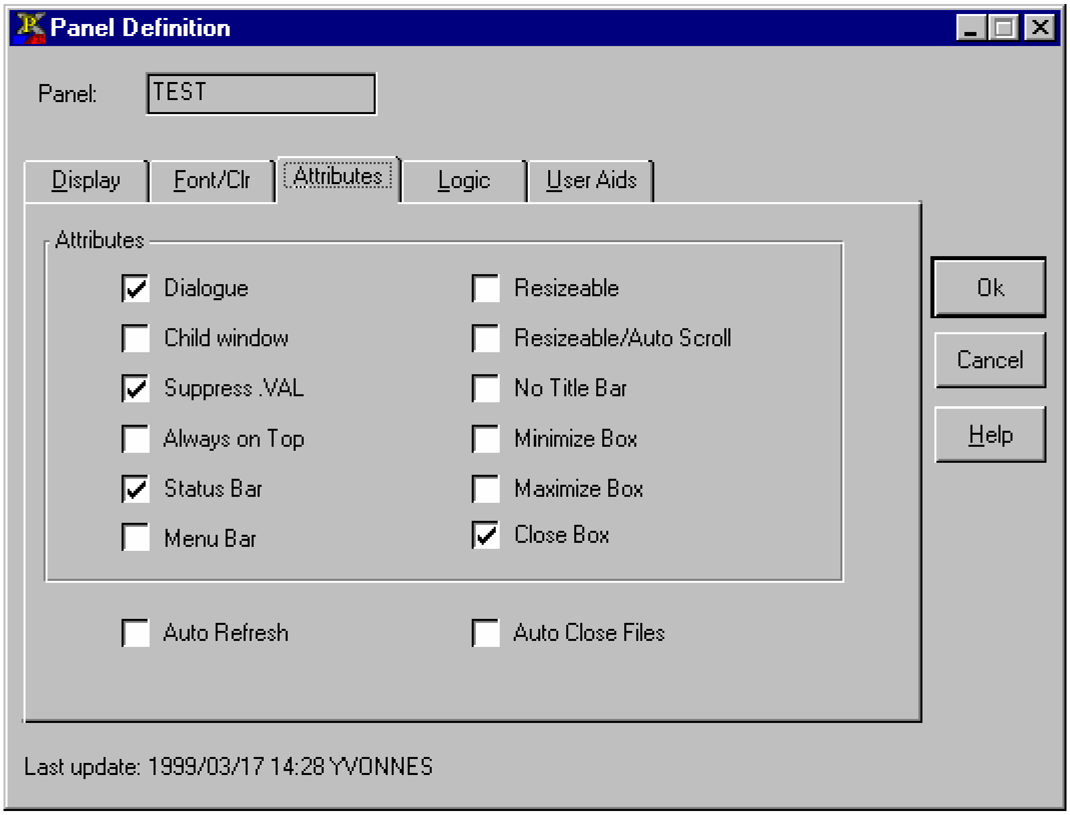
**Mnemonics:** There is also a choice of several mnemonics that can be set. These mnemonics will be only be applied to non-fonted text

Other: This field may contain any desired ProvideX attribute string that is to be output prior the display of the text. This should take the form of 'XX' or 'XX'+'YY'…

Example. 'BR'+'BU'

# Panel Header Properties (cont'd)

3. Activate the **Dialogue** and **Status Bar** options in the Attributes tab.



Panel: TEST

Attributes

☑ Dialogue
□ Child window
☑ Suppress .VAL
□ Always on Top
☑ Status Bar
□ Menu Bar

□ Resizeable
□ Resizeable/Auto Scroll
□ No Title Bar
□ Minimize Box
□ Maximize Box
☑ Close Box

□ Auto Refresh
□ Auto Close Files

Last update: 1999/03/17 14:28 YVONNES

# Panel Header Properties (cont'd)

## ATTRIBUTES OPTIONS

**Dialogue:** If dialogue  is selected the panel will be created as a free standing dialog box and not constrained by the main ProvideX window.  In addition, all controls including Menu bars will be disabled in all other windows on the screen.

**Child Window:** By selecting Child, the panel will be a child of the main window and cannot be moved outside of it.

**Suppress.Val:** The value of each control on a screen is automatically placed into the variable CTL_NAME.VAL$ (or CTL_NAME$ if the .VAL is suppressed)  For example, if a screen has a list box defined as LBOX_1 then the value within this control will be found in the variable LBOX_1.VAL$ (or LBOX_1$ if the .VAL is suppressed)

**Always on Top:** If this option is selected, this panel will always be displayed on top of any other open windows.

**Status Bar:** This option will display the "Message Bar" text in the status bar at the bottom of the panel.

**Menu Bar:** This option will display a previously defined menu bar at the top of the panel.  For a Menu bar to display, the panel must be a dialogue.

**Resizable:** This option will allow the panel to be resized by dragging the edges.

**Resizable/Auto Scroll:** This option will allow the panel to be resized, and when resized horizontal and/or vertical scrollbars will automatically appear.

**No Title Bar:** When this option is turned on, the panel is drawn without a title bar.

**Minimize Box:** When  selected, this option will display a Window Minimize icon in the top left corner of the panel.

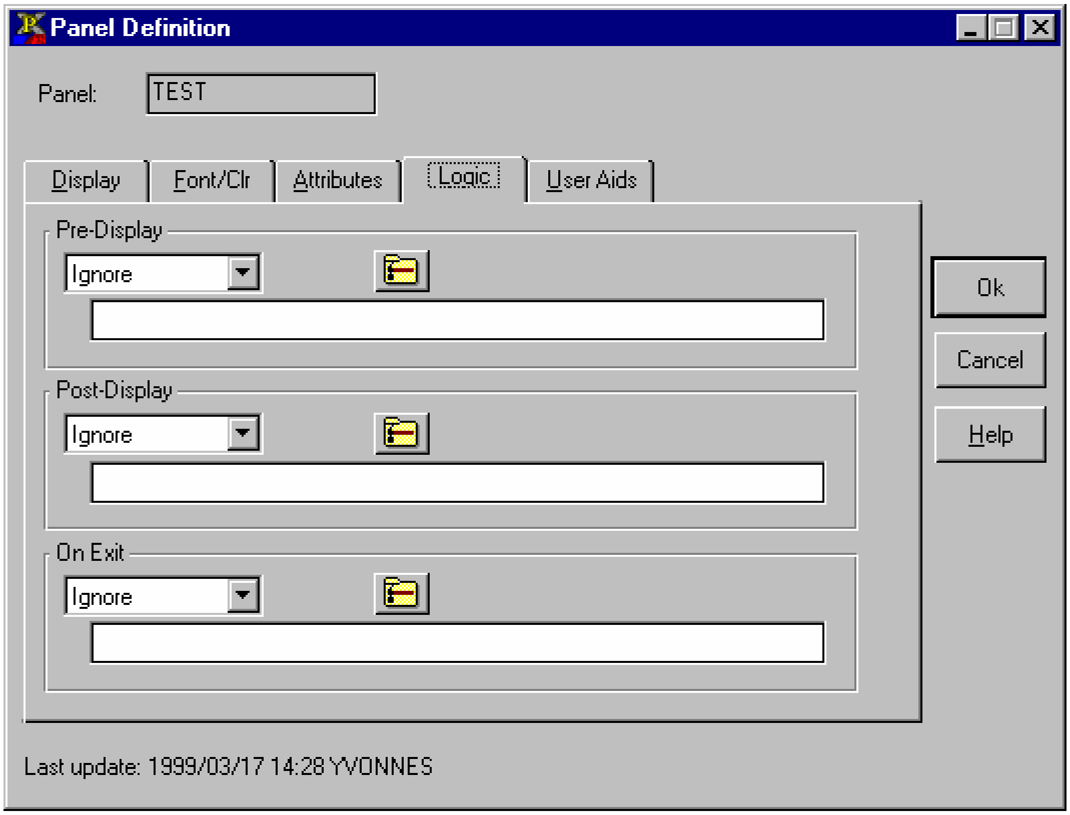
**Maximize Box:** When  selected, this option will display a Window Maximize icon in the top left corner of the panel.

**Close Box:** When  selected, this option will display a Window Close icon 'X' in the top right corner of the panel.

**Auto Refresh:** If this attribute is enabled, then the screen display is refreshed automatically when any control values are changed by the application.   This avoids the need to set REFRESH_FLG to 1 after changing the value of any of the screen variables.

**Auto Close Files:** If this attribute is enabled, then all NON-GLOBAL files that have been opened by the application are closed automatically when the panel terminates.

## Panel Header Properties (cont’d)



Panel Definition

Panel: TEST

Pre-Display: Ignore

Post-Display: Ignore

On Exit: Ignore

Last update: 1999/03/17 14:28 YVONNES

### LOGIC OPTIONS

**Pre-Display:** This logic will be processed before the panel is displayed.

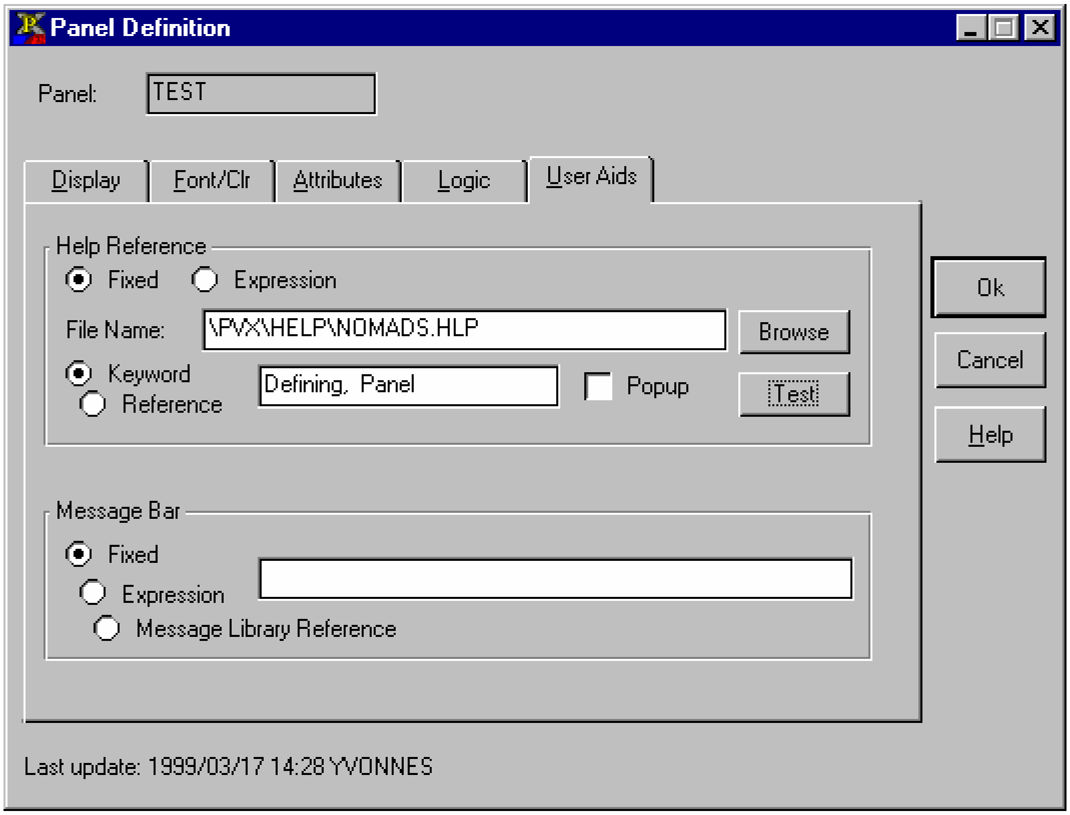
**Post-Display:** This logic will be processed after the panel has been drawn.

**On Exit:** This logic will be processed when the panel is closed. This logic will also be processed if the user selects the Windows Close icon 'X'

# Panel Header Properties (cont’d)

4. Enter **\PVX\HELP\NOMADS.HLP** in the Help Reference ‘File Name’ field

5. Enter **Defining,  Panel** for the ‘Keyword’



Panel Definition

Panel: TEST

Help Reference

● Fixed ○ Expression

File Name: \PVX\HELP\NOMADS.HLP

● Keyword ○ Reference: Defining, Panel

□ Popup

Message Bar

● Fixed

○ Expression

○ Message Library Reference

Last update: 1999/03/17 14:28 YVONNES

# Panel Header Properties (cont'd)

## USER AID OPTIONS

**Help Reference:**

File Name: Fixed: This field would contain the name of the Windows help file.

Example: NOMADS.HLP

Expression: The expression option indicates to NOMADS that the help file to be displayed is stored in a string variable or simple expression

Example 1: "LREF.HLP;INTRODUCTION"

This example will display the language reference help file in an independent window and will be positioned at the Inroduction section.

Example 2: "LREF.HLP;#123"

This example will display the language reference help file in an independent window (indicated by '#' ) and will be positioned at the context string value 123 which is the SET_FOCUS directive

Example 3: "LREF.HLP;$123"

This example will display the language reference help file in a popup (indicated by '$' ) and will be positioned at the context string value 123 which is the SET_FOCUS directive

Keyword: This is the index entry for the topic in the indicated help file.

Example: "LREF.HLP;INTRODUCTION"

INTRODUCTION is the keyword (or index entry) in the language reference help file.

Reference: This is the reference number for a specific topic in the context sensitive help.

Example: "LREF.HLP;#123"

123 is the context string (reference number) for the SET_FOCUS topic in the language reference help file.

Popup: When this option is selected the help topic will be displayed as a popup rather than an independent window.

**Message Bar:** The message bar is the message that will be displayed at the bottom of the panel. The Fixed option will display the text as entered. The Expression option indicates to NOMADS that the text to be displayed is stored in a string variable or simple expression. The expression will be evaluated when the object is created.

# Creating / Editing Controls (Red/Green Rectangle)

A control is defined using the keyboard or mouse. The size and location of a control is determined by the rectangle.

1. Create a button using the keyboard

   Use arrow keys to move / size the rectangle (message bar on bottom of panel will
   display current mode…default is *moving* mode)
   Use the spacebar to toggle between *moving* and *sizing* modes
   Hit the return key to force creation of button with ***Text*** = HIT ME (will display properties panel)

2. Edit the button using the keyboard

   Use arrow keys to position rectangle on button (top left corner).
   *Size:*
   a) Press ALT Edit / Size (blue outline will appear around button
   b) Use arrow keys to resize
   c) Press *Enter* (will display properties panel)

   *Move:*
   a) Press ALT Edit / Move (blue outline will appear around button
   b) Use arrow keys to move
   c) Press *Enter* to edit button properties

3. Delete the button and re-create it using the mouse

   a) Position rectangle on a blank entry on the panel using left button on the mouse.
   b) Hold left button down / drag mouse down and to the right to resize the rectangle.
   c) Release left button on the mouse to create a new control.
   d) Create button with ***Text*** = 'HIT ME'.

4. Edit the existing button using the mouse

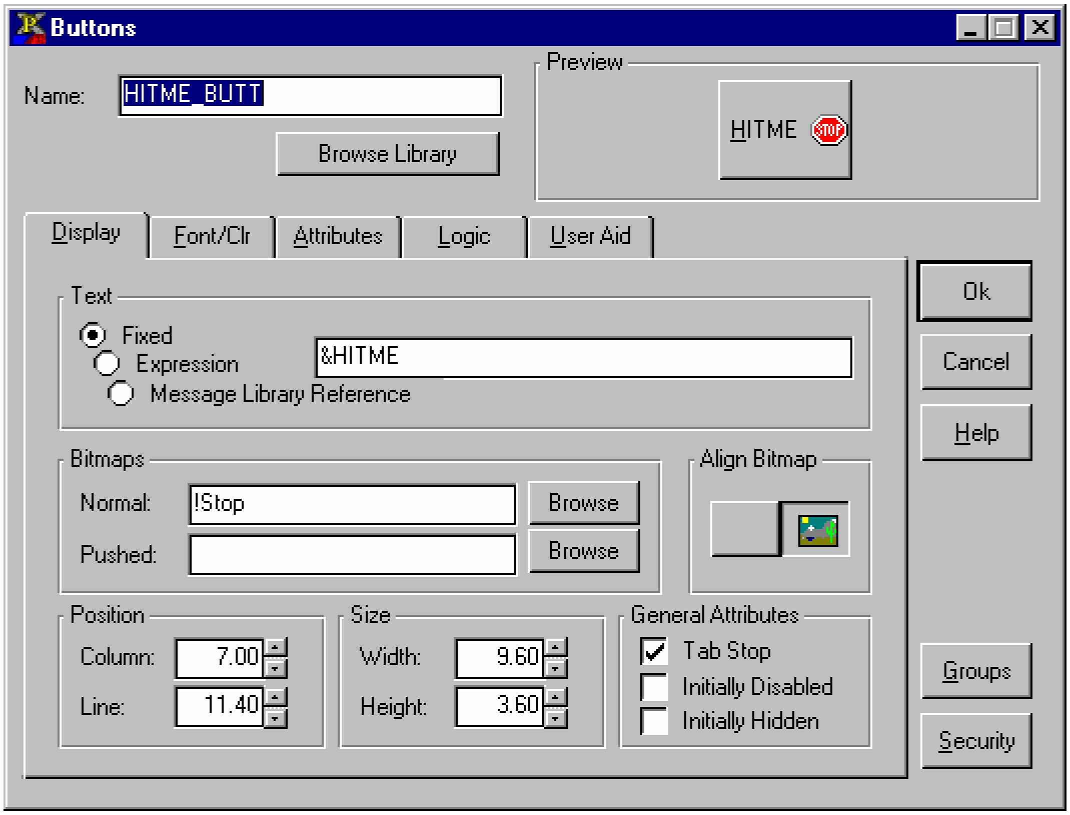
   Position mouse cursor on top left corner of control and press the left mouse button to highlight the object.

   *Size:* Double click on left button and hold down, drag mouse across screen to resize.
   *Move:* Single click on left button and drag to move.
   Click on right mouse button to edit properties.

# Button Properties

1. Create a button called **HITME_BUTT**

2. Text: **&Hitme**



Name: HITME_BUTT

Text
● Fixed
○ Expression
○ Message Library Reference
&HITME

Bitmaps
Normal: !Stop
Pushed:

Position
Column: 7.00
Line: 11.40

Size
Width: 9.60
Height: 3.60

General Attributes
☑ Tab Stop
□ Initially Disabled
□ Initially Hidden

# Button Properties (cont'd)

## DISPLAY OPTIONS

**Name:** This is the name of the control, there is also a browse library button available in order to browse the names of all the other buttons in the library. The name of the control also corresponds to a variable with the same name in ProvideX.

**Text:** The fixed text option will display the text as entered. The expression option indicates to NOMADS that the text to be displayed is stored in a string variable or simple expression. The expression will be evaluated when the object is created.

**Bitmaps:** ProvideX allows the user to place a bitmap on a control. By pressing the browse button the user can select an embedded bitmap from the options provided or enter the path and name of an external bitmap. The PUSHED selection is an optional second bitmap that will appear when the button is pushed.

**Align Bitmap:** For left or right justification of pictures on the buttons.

**Column/Line:** The top left position of the control.

**Width/Height:** The width and height of the control.

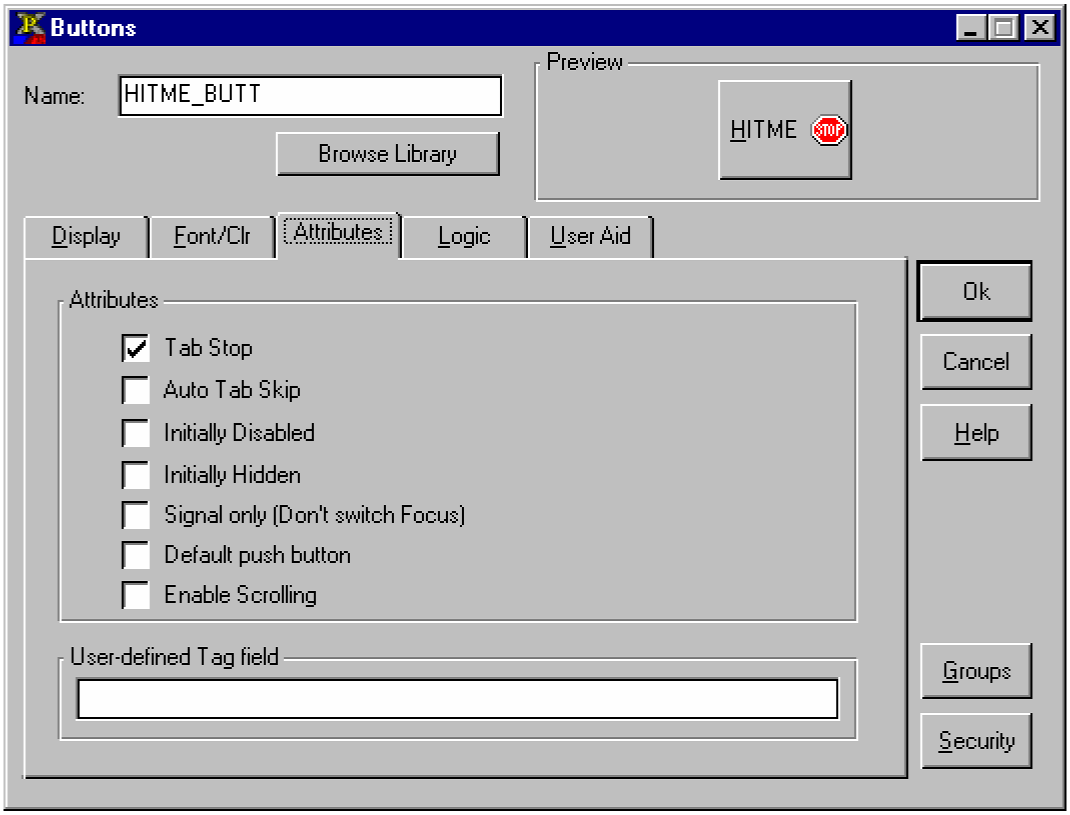
**General Attributes:**

- **Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).
- **Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.
- **Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Groups:** Quick access to Group Assignments.

**Security:** Quick access to the NOMADS Security options.

# Button Properties (cont'd)



Name: HITME_BUTT

Attributes

☑ Tab Stop
□ Auto Tab Skip
□ Initially Disabled
□ Initially Hidden
□ Signal only (Don't switch Focus)
□ Default push button
□ Enable Scrolling

User-defined Tag field

# Button Properties (cont'd)

## ATTRIBUTES OPTIONS

**Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).

**Auto Tab Skip:** When this option is selected the field will be skipped when tabbing forward, but can be accessed by tabbing back (shift+tab) or selecting it with the mouse.

**Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.

**Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Signal Only:** If focus is currently on another control such as a multi-line and the user presses the button, the 'When button pressed' logic will be executed but focus will return to the multiline.

**Default push button:** If this option is selected, whenever the user presses the ENTER key, the "When Button Pressed" Logic of this button will be executed. Note: There can only be one default push button per panel.

**Enable scrolling:** This option will enable the control to scroll with the panel if the 'Resizable/Auto Scroll' option is selected in the panel header.

**User-defined tag field:** This is a user defined data field, the contents of this field will be placed in a variable called *control_name*.TAG$. The tag field can be used to pass information for such things as Formatting, Error messages, Validation rules. The Tag field is passed to the input validator through field 3 – TAG_FIELD$.

# Button Properties (cont'd)

Assign a simple PVX command (**PRINT "OUCH"**) to the newly created button

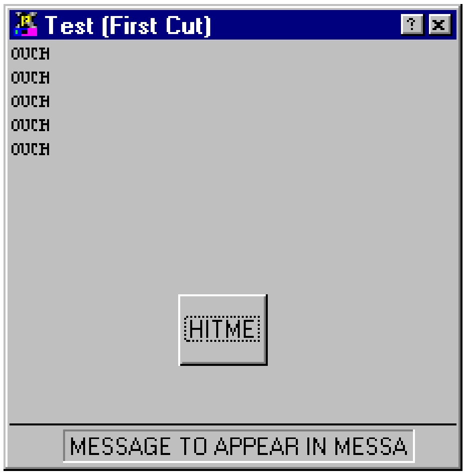
1. Select the 'Execute' Function from the '*When Button Pressed'* drop box.

2. In the Multi-line to the right of the function enter: **PRINT "OUCH"**.

3. Record Changes.

4. Test.

**Sample Results:**



```
OUCH
OUCH
OUCH
OUCH
OUCH
```

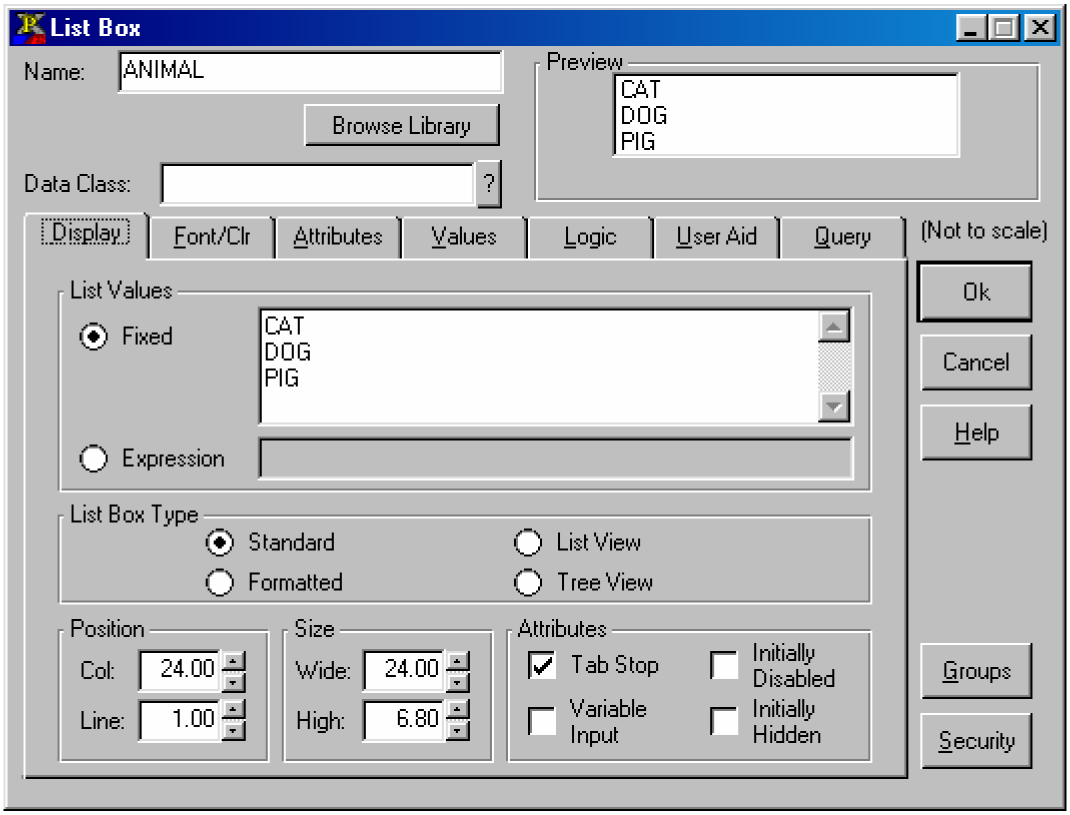
MESSAGE TO APPEAR IN MESSA

# List/Drop Box Properties

Drop and List boxes allow the user to select a value from a pre-defined list of selections. A list box shows the possible selections on the screen, while a drop box will 'Drop down' the selection list.

Drop/List boxes can optionally allow the user to enter a selection not contained in the list. These are known as "Variable" drop/list boxes.

1. Create a list box called **'ANIMAL'**
2. Enter the Fixed Values for the list box: **CAT**
   **DOG**
   **PIG**



Name: ANIMAL

Preview: CAT<br>DOG<br>PIG

List Values
● Fixed: CAT, DOG, PIG
○ Expression

List Box Type
● Standard ○ List View
○ Formatted ○ Tree View

Position: Col: 24.00 Line: 1.00

Size: Wide: 24.00 High: 6.80

Attributes: ☑ Tab Stop □ Initially Disabled □ Variable Input □ Initially Hidden

# List/Drop Box Properties (cont'd)

## DISPLAY OPTIONS

**Name:** This is the name of the control, there is also a browse library button available in order to browse the names of all the other Drop Boxes or List Boxes in the library. The name of the control also corresponds to a variable with the same name in ProvideX.

**List Values:** This is an initial list of values to be displayed in the drop/list box. The Fixed option will display the values as entered. The expression option indicates to NOMADS that the values to be displayed are stored in a string variable or simple expression. The expression will be evaluated when the control is drawn.

**Column/Line:** The top left position of the control.

**Width/Height:** The width and height of the control.

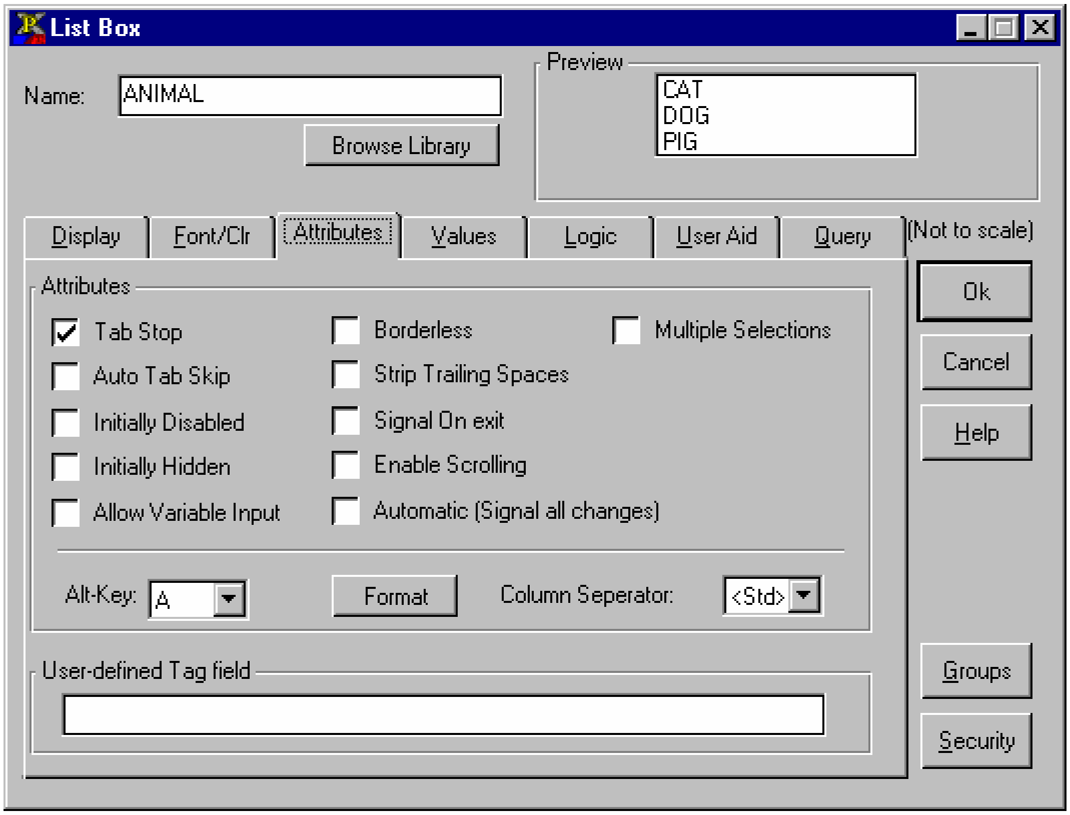
**Common Attributes:**

- **Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).
- **Variable Input:** If this option is selected. The user is able to type a value into the drop/list box without the value already being presented within it.
- **Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.
- **Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Groups:** Quick access to Group Assignments.

**Security:** Quick access to the NOMADS Security options.

# List/Drop Box Properties (cont'd)



List Box

Name: ANIMAL

Preview

CAT
DOG
PIG

(Not to scale)

Attributes

- ☑ Tab Stop
- ☐ Auto Tab Skip
- ☐ Initially Disabled
- ☐ Initially Hidden
- ☐ Allow Variable Input
- ☐ Borderless
- ☐ Strip Trailing Spaces
- ☐ Signal On exit
- ☐ Enable Scrolling
- ☐ Automatic (Signal all changes)
- ☐ Multiple Selections

Alt-Key: A

Column Seperator: <Std>

User-defined Tag field

# List/Drop Box Properties (cont'd)

## ATTRIBUTES OPTIONS

**Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).

**Auto Tab Skip:** When this option is selected the field will be skipped when tabbing forward, but can be accessed by tabbing back (shift+tab) or selecting it with the mouse.

**Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.

**Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Allow Variable Input:** If this option is selected. The user is able to type a value into the drop/list box without the value already being presented within it.

**Automatic:** If this option is selected, each time the value changes, the 'On Select' logic will be executed.

**Borderless:** When this option is selected the control will be drawn without a border surrounding it.

**Strip Trailing Spaces:** If this option is selected, NOMADS will strip off all trailing spaces on the return value.

**Signal on Exit:** Every time focus leaves the control, the "On Select" logic for the control will be executed.

**Enable scrolling:** This option will enable the control to scroll with the panel if the 'Resizable/Auto Scroll' option is selected in the panel header.

**Multiple Selections:** This new attribute allows the selection of multiple items from a list box. A range of values can be selected by holding the Shift key. Many individual entries can be selected by holding the Ctrl key. These selections are returned in the control name assigned to the list box separated by a delimiter.

**Alt-Key:** Select a 'HOT-KEY' that will be used to switch focus to this control.

Example: ALT-KEY = L - When the user presses ALT+L, focus will move to this control.

**Column Separator:** The data for each entry in the list box would consist of the contents of columns delimited by a separator. This option is applicable to a formatted list box.

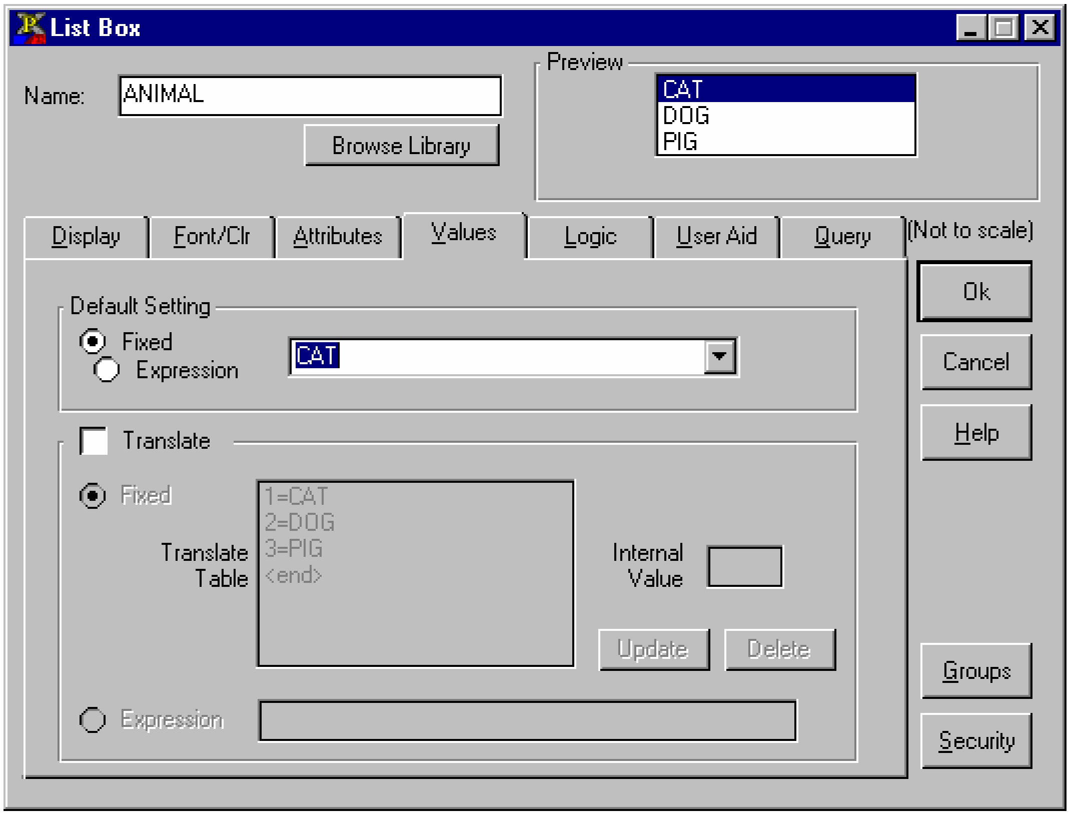
**User-defined tag field:** This is a user defined data field, the contents of this field will be placed in a variable called *control_name*.TAG$. The tag field can be used to pass information for such things as Formatting, Error messages, Validation rules. The Tag field is passed to the input validator through field 3 – TAG_FIELD$.

**Format:** Press this button to setup the formatting definition for a formatted list box.

# List/Drop Box Properties (cont'd)

3. Select **CAT** from the 'Default Setting' drop box

The default setting is the initial value (selection) to be highlighted in the list box.This is accomplished internally with the **LIST_BOX WRITE** directive.



Name: ANIMAL

Preview
CAT
DOG
PIG

(Not to scale)

Default Setting
● Fixed
○ Expression
CAT

□ Translate

● Fixed

Translate Table
1=CAT
2=DOG
3=PIG
<end>

Internal Value

○ Expression

## VALUES OPTIONS

**Default Setting:** This will set the value to be highlighted when the control is drawn. The fixed option will display the value as entered. The expression option indicates to NOMADS that the value to be initially displayed is stored in a string variable or simple expression. The expression will be evaluated when the object is created.

**Translate:** A single character translation table. When the translate option is selected a default translation value is assigned. This can be changed by selecting the value, changing the internal value and pressing update.

# List/Drop Box Properties (cont'd)

Add select logic to list box **ANIMAL** to print a message box

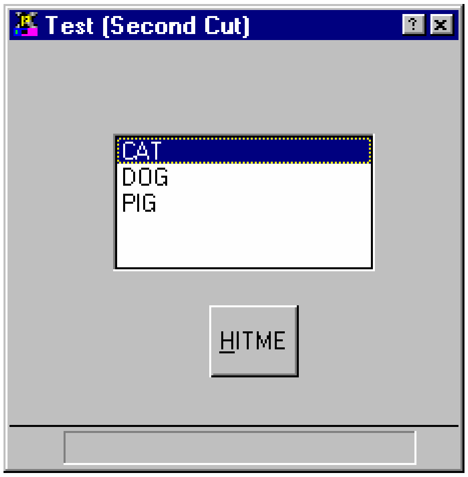
1. Select the 'Execute' function from the '*When Entry Is Selected From List Box'* drop box

2. In the multi-line to the right of the function enter the PVX command: **MSGBOX "VALUE="+ANIMAL$**

3. Record Changes

4. Test

**Sample Results:**



# Invoking A Panel/Passing Arguments To A Panel

1. Start a new PVX session.

2. To invoke an object from a standard ProvideX program use either the PROCESS directive or a CALL "*WINPROC".

   Enter the PVX command: **CWDIR "\NOMTRAIN"**
   Enter the PVX command: **PROCESS "TEST", "MYLIB.EN"**
   (Press *F4* on panel to exit back to session)

3. ALT- tab back to the first PVX session and perform the following:

   a) Change **HITME_BUTT** button to use ARG_1$ and CMD_STR$:
      i) Select the 'Execute' function from the '*When Button Pressed'* drop box
      ii) In the multi-line to the right of the function
      enter the PVX command: **ARG_1$=ANIMAL$, CMD_STR$="END"**
      Note: (END must be uppercase)

   b) Remove the '*When Entry Is Selected From List Box'* logic on the **ANIMAL** listbox by choosing the 'Ignore' function

   c) Record changes

4. ALT Tab to the second PVX session
   Enter the following PVX command: **PROCESS "TEST", "MYLIB.EN", X$**
   The following action occurs:

   Pressing the **HITME_BUTT** button will pass the list box selection into ARG_1$, CMD_STR$ = "END" will close panel.

   **PRINT X$** (Will contain list box selection value of ARG_1$)

5. Change the 'Text' on the button to display **&Ok**

   Remove from the '*When Button Pressed'* logic: ARG_1$=ANIMAL$,CMD_STR$="END"

6. Create a new button

   Text = **&Cancel**
   '*When Button Pressed'* logic = END (works same as CMD_STR$="END")

7. Add Logic in the Header Definition of the panel called **TEST**.

   Pre-display: **Execute ANIMAL$=ARG_1$**
   On Exit: **Execute ARG_1$=ANIMAL$**

# Invoking A Panel/Passing Arguments To A Panel (cont'd)

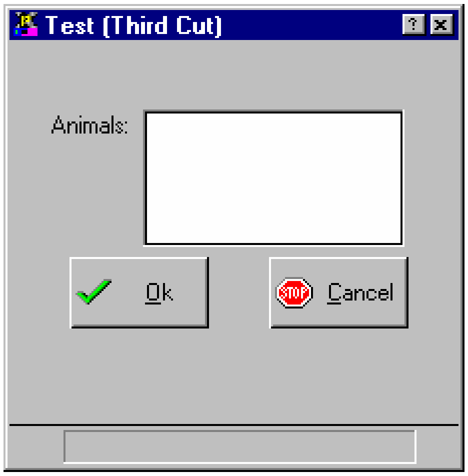
8. Switch to other PVX session and perform the following steps:

   a) X$="DOG"; PROCESS "TEST","MYLIB.EN",X$
      (X$ will highlight "DOG" in list box)
   b) Change list box selection
   c) Press cancel button (close panel)
   d) PRINT X$ (will have new list box selection)

9. Edit **TEST** panel to apply fonted text, tips, messages (etc.)

---

**Sample Results:**



# Loading A List Box From A Program

1. ALT- tab to the other PVX Session

2. In the current directory create a program to load the list box on the panel.

   Enter the following:

```
0010  FL=UNT;OPEN(FL)”.”
0020  READ (FL, END=1000)R$
0030  IF LEN(R$)=3 GOTO 20
0040  IF LCS(MID(R$,LEN(R$)-3))=”.bmp” LIST_BOX LOAD ANIMAL.CTL,0,R$(1,LEN(R$)-4)
0050  GOTO 20
1000  CLOSE (FL)
1010  END
```

   SAVE as “PROG01”

3. Switch to other PVX session and edit the list box **ANIMAL** on the panel **TEST** to use the program in the ‘Prior List Box Display’ logic

   a) Select the ‘Perform’ function from the ‘Prior List Box Display’ drop box.

   b) In the multi-line to the right of the function enter the following: **“PROG01”**

   c) Clear **Fixed List** & **Default Setting** entries.

   d) Record and test.

# User Defined Controls

User defined CTLs are used to associate a program or an event with a specific function key. User CTLS can be local to a panel or global to all libraries.  Local CTLs will be executed before global controls.

### Defining local CTLs

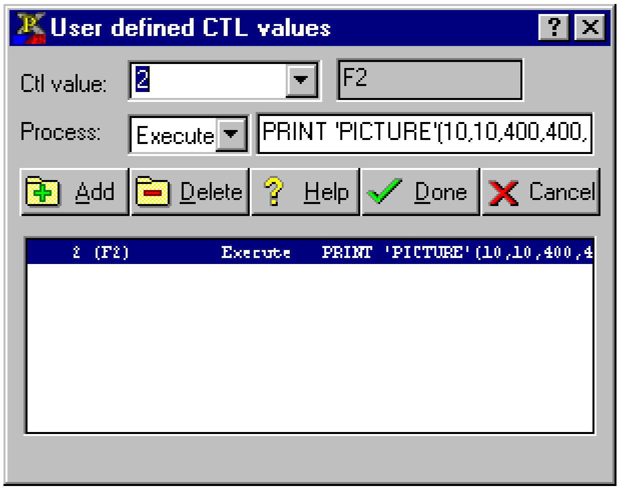
Assigning the F2 key to the **TEST** panel

1. Select the 'User Ctls' option from the Menu Bar (**P**anel/**U**ser Ctls) or Press the 'Ctls' button from the controls toolbar

2. Select **F2** from the 'Ctl Value' drop box

3. Select 'Execute' from the 'Process' drop box and enter the following PVX command:

   **PRINT 'PICTURE'(10,10,400,400,ANIMAL$+".bmp")**



Ctl value: 2 F2

Process: Execute PRINT 'PICTURE'(10,10,400,400,

```
2 (F2)          Execute   PRINT 'PICTURE'(10,10,400,4
```

### Defining global CTLS

Using a generic program to handle function keys for all panels

1. Alt-tab to other PVX session

2. Enter the following lines of code:
   0010 IF CTL=3 THEN MSGBOX "We hit the F3 key", "FYI"
   0020 EXIT

3. Save as "FKEYS"

4. Load the variable **%NOMADS_FKEY_HANDLER$** with "FKEYS"

5. Invoke the panel using the PROCESS directive: PROCESS "TEST","MYLIB.EN"

# Event Driven Programming

This exercise has been designed to show you how a file maintenance screen is created, from drawing the objects, to writing and connecting them with the code that will establish their behaviour.

This exercise consists of creating a Terms code file maintenance program. The data file will be a **DIRECT** file with a 2 character key.

The format will be:

| | | | |
|---|---|---|---|
| KEY | 2Ch | Terms code | |
| FLD1 | 2Ch | Terms code | Name: TRM_CD$ |
| FLD2 | 30Ch | Description | Name: TRM_DESC$ |
| FLD3 | 10Dig | Date | Name: TRM_DATE$ |

1. ALT-Tab to other PVX Session

2. Create file: **DIRECT "TERMCODE",2**

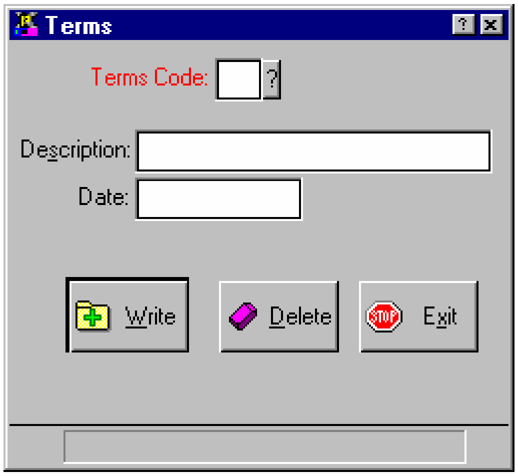
3. Return to the library screen and create a new panel object called **TERMS**.

4. Activate the **DIALOGUE** and **STATUS BAR** options in the ‘Attributes’ tab of the Panel Header Definition.

5. Create 3 buttons:

   Text = **&Write**
   **&Delete**
   **E&xit** (select ‘END’ function as the ‘When Button Pressed logic’)

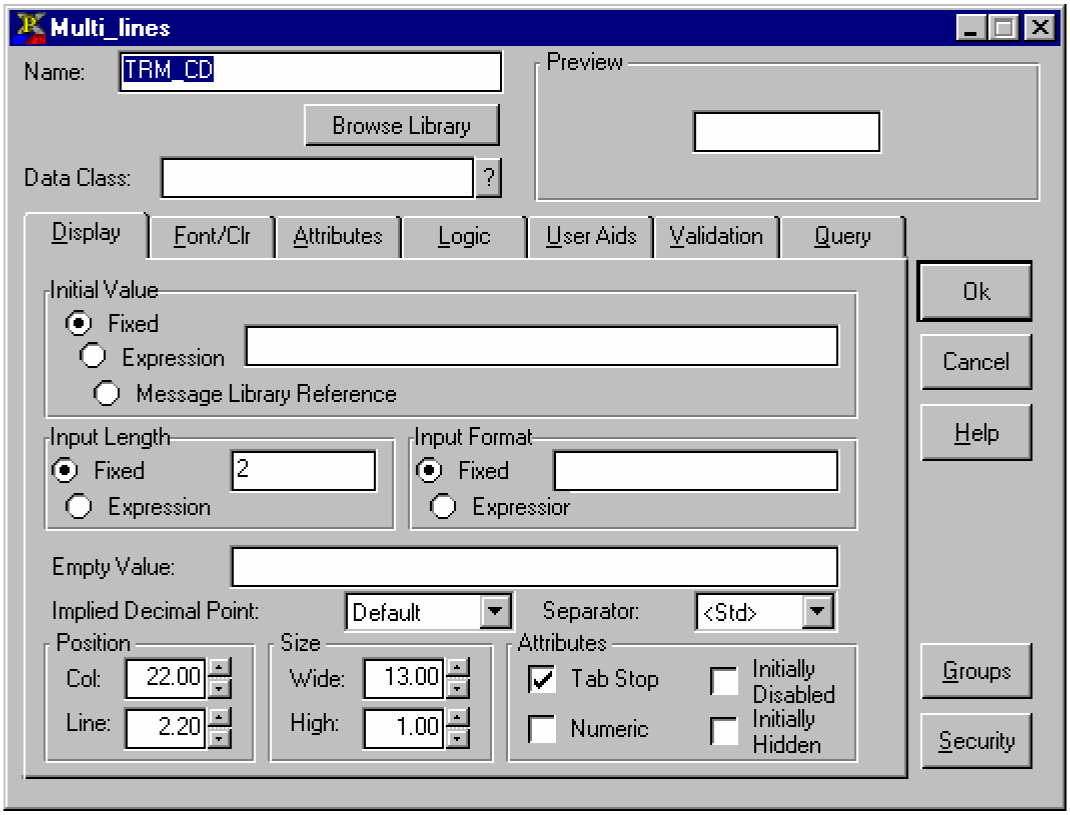
**Sample Results:**



# Multi Line Properties

1. Create a Multi-line called **TRM_CD**

2. Set the 'Input Length' to **2**



Multi_lines

Name: TRM_CD

Data Class:

Initial Value: ● Fixed ○ Expression ○ Message Library Reference

Input Length: ● Fixed 2 ○ Expression

Input Format: ● Fixed ○ Expression

Empty Value:

Implied Decimal Point: Default

Separator: <Std>

Position: Col: 22.00 Line: 2.20

Size: Wide: 13.00 High: 1.00

Attributes: ☑ Tab Stop □ Numeric □ Initially Disabled □ Initially Hidden

# Multi Line Properties (cont'd)

## DISPLAY OPTIONS

**Name:** This is the name of the control, there is also a browse library button available in order to browse the names of all the other Check Boxes in the library. The name of the control also corresponds to a variable with the same name in ProvideX.

**Data Class:** Select the pre-defined data class to apply to the control.

**Initial Value:** The fixed value option will give the control an initial value of exactly what is entered. The expression option indicates to NOMADS that the value is stored in a string variable or simple expression. The expression will be evaluated when the control is drawn.

**Input Length:** The fixed length option will give the control a fixed length (maximum length) of exactly what is entered. The expression option indicates to NOMADS that input length is stored in a string variable or simple expression. The expression will be evaluated when the control is drawn.

**Input Format:** The fixed input format (format mask) option will give the control a format mask of exactly what is entered. The expression option indicates to NOMADS that the format mask is stored in a string variable or simple expression. The expression will be evaluated when the control is drawn.

**Empty Value:** This is the value to be displayed when the field is empty. Example: "Today's Date"

**Implied Decimal Point:** When using a numeric format mask and the implied decimal point is "Always" the value entered would be displayed starting at the far right of the decimal point.

Example: 100 would display as 1.00

The "Never" option would display the value starting at the left of the decimal.

Example: 100 would display as 100.00

**Separator:** In the case of multiple lines of text, each line will be delimited with the selected separator.

**Column/Line:** The top left position of the control.

**Width/Height:** The width and height of the control.

**Common Attributes:**

**Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).

**Numeric:** When this option is selected, the value entered will result in a numeric value/variable. Default is string.

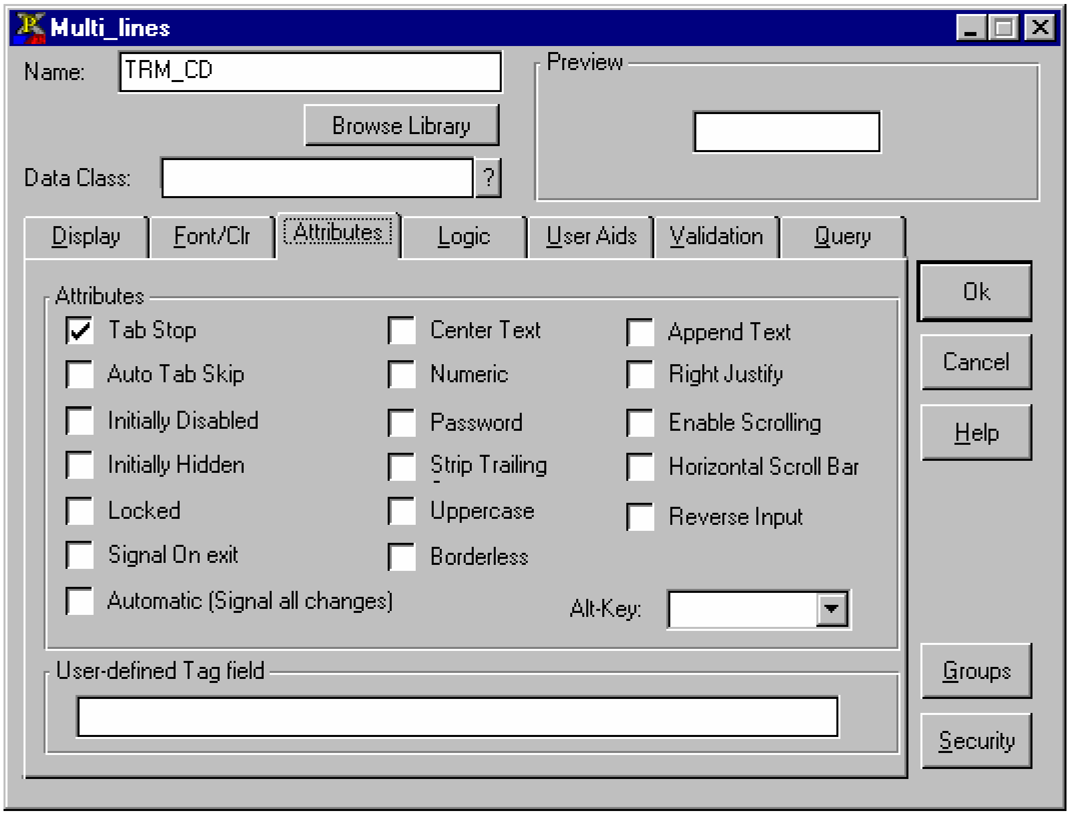
**Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.

**Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Groups:** Quick access to Group Assignments.

**Security:** Quick access to the NOMADS Security options.

# Multi Line Properties (cont'd)



Name: TRM_CD

Data Class:

Attributes

☑ Tab Stop
□ Auto Tab Skip
□ Initially Disabled
□ Initially Hidden
□ Locked
□ Signal On exit
□ Automatic (Signal all changes)

□ Center Text
□ Numeric
□ Password
□ Strip Trailing
□ Uppercase
□ Borderless

□ Append Text
□ Right Justify
□ Enable Scrolling
□ Horizontal Scroll Bar
□ Reverse Input

Alt-Key:

User-defined Tag field

# Multi Line Properties (cont'd)

## ATTRIBUTES OPTIONS

**Tab Stop:** If the Tab stop option is selected the control can be 'Tabbed' into and will appear in the Tab list. If the tab stop is not selected the control can only be accessed via the mouse or by an ALT-key sequence or Hot Key (if defined).

**Auto Tab Skip:** When this option is selected the field will be skipped when tabbing forward, but can be accessed by tabbing back (shift+tab) or selecting it with the mouse.

**Initially Disabled:** A control that is initially disabled will appear on the panel when it is drawn but the user will be unable to select it. To enable the control, use the ENABLE CONTROL directive or the group enable subprogram *wingrp.

**Initially Hidden:** A control that is initially hidden will not appear on the panel when it is drawn. To have a hidden control become visible use the SHOW directive or the group show subprogram *wingrp.

**Locked:** If the Multi-line is locked the user will be unable to make changes to the data contained in the multi-line.

**Signal on Exit:** Every time focus leave this control, the "On Select" logic for the control will be executed.

**Automatic:** If this option is selected, each time the value changes, the 'On Select' logic will be executed.

**Center Text:** Center the text within the multi-line.

**Numeric:** When this option is selected, the value entered will result in a numeric value/variable. Default is string.

.

**Password:** If the password check box is marked the multi-line input field is considered confidential. This means the data in the field will be replaced with a $.

**Strip Trailing Spaces:** If this option is selected, NOMADS will strip off all trailing spaces on the return value.

**Uppercase:** Convert to uppercase.

**Borderless:** If the Multi-line is borderles only the text portion of the Multi-line will be displayed. A borderlines multi-line is a simple means of presenting variable data to the user. The main advantage of using borderlines Multi-lines to present data is that the form can be redesigned without any program changes.

**Append Text:** When tabbing to an input field (multi-line) cursor automatically moves to the end of the input.

**Right Justify:** Right justify the text within the multi-line.

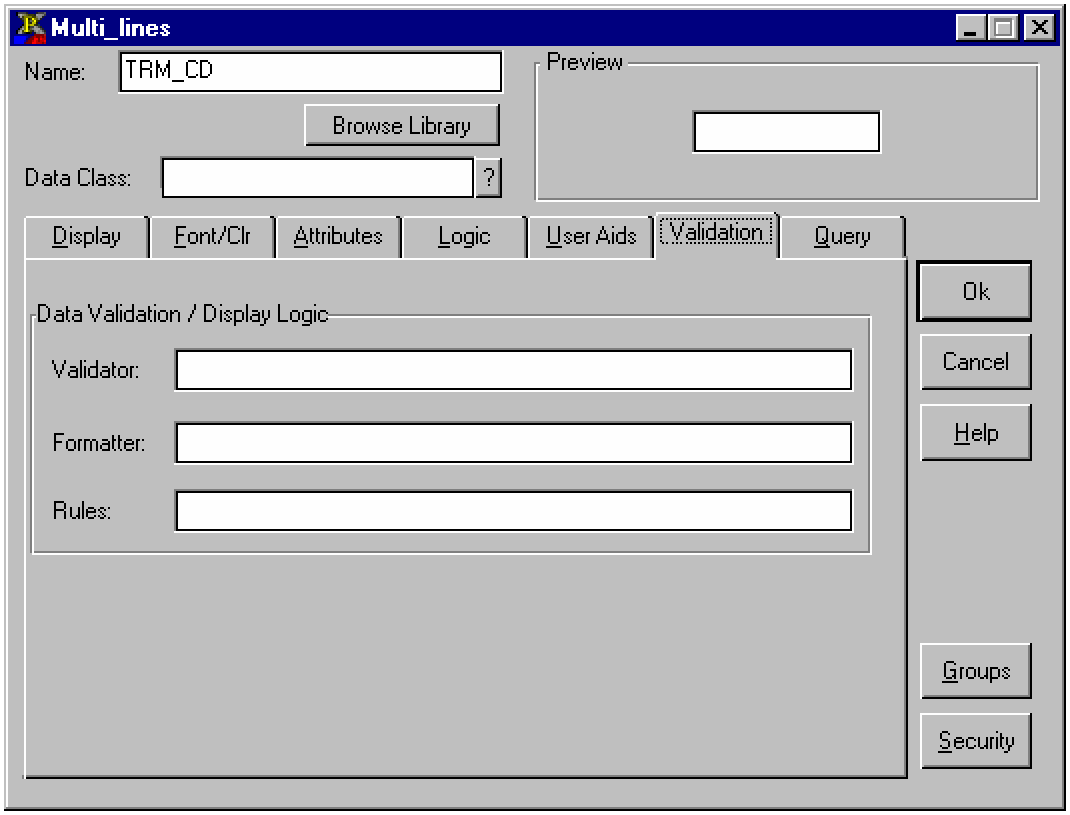
**Enable scrolling:** This option will enable the control to scroll with the panel if the 'Resizable/Auto Scroll' option is selected in the panel header.

**Alt-Key:** Select a 'HOT-KEY' that will be used to switch focus to this control.

Example: ALT-KEY = L - When the user presses ALT+L, focus will move to this control.

**User-defined tag field:** This is a user defined data field, the contents of this field will be placed in a variable called *control_name*.TAG$. The tag field can be used to pass information for such things as Formatting, Error messages, Validation rules. The Tag field is passed to the input validator through field 3 – TAG_FIELD$.

# Multi Line Properties (cont'd)



Name: TRM_CD

Data Class:

Data Validation / Display Logic

Validator:

Formatter:

Rules:

# Multi Line Properties (cont'd)

## VALIDATION OPTIONS

**Validator:** The name of program that will be called whenever this field is input to validate the input and convert it to internal format. There are five arguments passed on the call:

Example:

```
CALL "<input_validator>" ,  INPUT_FIELDS$, ERR_MSG$, TAG_FIELD$,
OLD_VALUE$, INPUT_EOM$
```

Where:

INPUT_FIELD$ will contain the input the user entered and the program should place the internal format value

ERR_MSG$ if set to non-null should contain the error message to display

TAG_FIELD$ is the TAG field for the field

OLD_VALUE$ will contain the old value for the field

INPUT_EOM$ will have the terminator for the input

**Formatter:** The name of a program that will be called to convert the data from the internal format to the format used in the display.

Example:

```
CALL "<output_prog>",  DATA_VALUE$, TAG_FIELD$
```

Where:

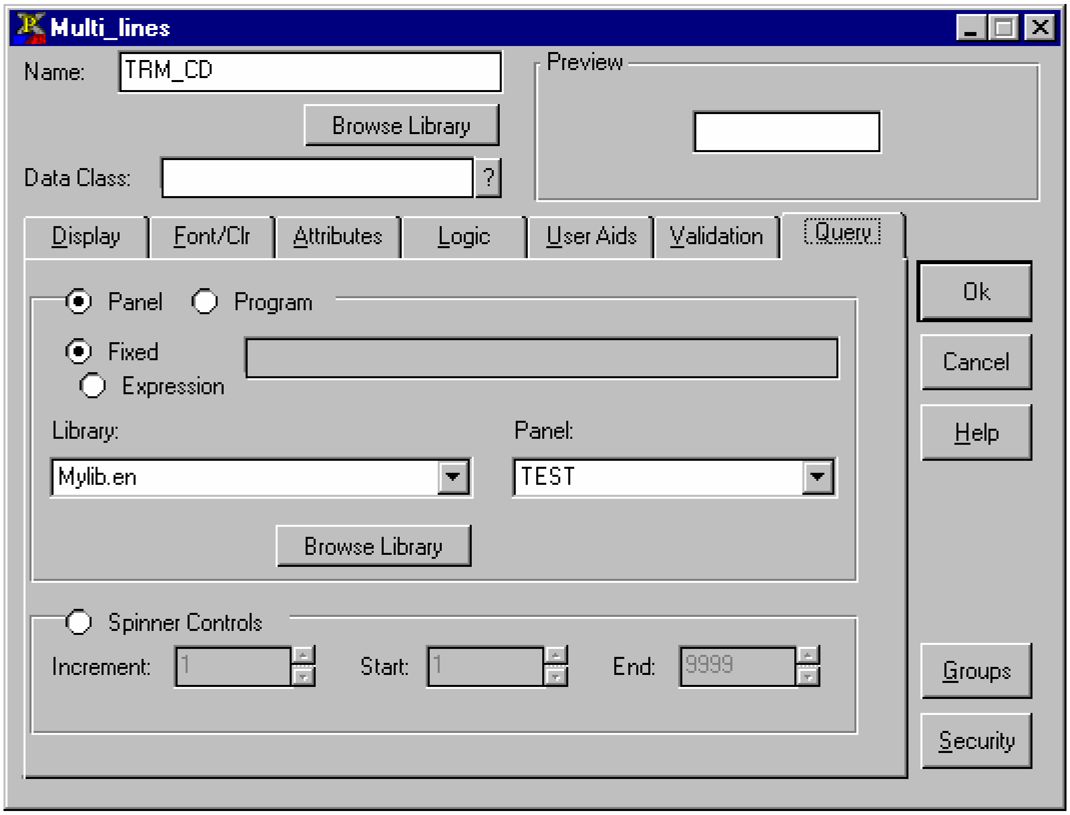
DATA_VALUE$ will be passed the internal value of the field and the program should convert it for display.

TAG_FIELD$ is the TAG field for the field

**Rules:** This list can contain a series of values/ranges of valid data values. For Example: "A,C, I-K"

The program *WIN/DATE can be used as an example of an Input validation/formatting subprogram.

# Multi Line Properties (cont'd)



Name: TRM_CD

Data Class:

● Panel ○ Program

● Fixed ○ Expression

Library: Mylib.en

Panel: TEST

○ Spinner Controls

Increment: 1 Start: 1 End: 9999

## QUERY OPTIONS

**Panel:** The fixed panel option will process the panel and library that has been selected. The expression option indicates to NOMADS that the panel or panel and library name is stored in a string variable or simple expression. The expression will be evaluated when the control is created.

**Program:** The fixed program option will execute the program that has been entered. The expression option indicates to NOMADS that the program to be executed is stored in a string variable or simple expression. The expression will be evaluated when the control is created.

**Spinner:** Example: Increment = 1, Start = 1, End = 9999.

The above example would place a vertical scrollbar on the control and give the control an initial value of 1, by moving the scrollbar up or down, the value would increase or decrease by 1 with the highest value being 9999.

**NOTE: If a Query is specified for a multi-line control, a button will be drawn beside the control.**

# Attaching Code To Controls

1. Create a new program called: **“PROG02”**

2. Initialization/wrapup logic.

```
0010 INIT:
0020 MSGBOX "We Are At The Init Logic","Init Box"
0030 LET FL=UNT; OPEN (FL)"TERMCODE"
0040 EXIT
0050 WRAPUP:
0060 CLOSE (FL)
0070 MSGBOX "We Are At The Wrapup Logic","See Ya"
0080 RETURN
```

3. In the Logic tab of the Header Definition Panel add the following:

Pre-display: PERFORM “PROG02;INIT”
On Exit: PERFORM “PROG02;WRAPUP”

4. ‘When Button Pressed’ Logic of the **&Write** button

```
1010 WRITE_IT:
1020 IF NUL(TRM_DESC$) THEN MSGBOX "Missing Desc","Big Err"; LET NEXT_ID=TRM_DE
1020:SC.CTL; RETURN
1030 WRITE (FL,KEY=TRM_CD$)IOL=8000
1040 MSGBOX "Record Updated","So!!"
1050 LET NEXT_ID=TRM_CD.CTL; RETURN
8000 IOLIST TRM_CD$,TRM_DESC$,TRM_DATE$
```

5. Attach the following code to the ‘On Select’ Logic of the **TRM_CD** control:

```
2010 READ_IT:
2020 LET REFRESH_FLG=1
2030 READ (FL,KEY=TRM_CD$,DOM=NEWONE)IOL=8000
2040 RETURN
2050 NEWONE:
2060 LET TRM_DESC$="",TRM_DATE$=""
2070 RETURN
```

6. Attach the following code to the ‘When Button Pressed’ Logic of the **&Delete** button:

```
3010 DELETE_IT:
3020 REMOVE (FL,KEY=TRM_CD$,DOM=BADTRM)
3030 MSGBOX "Record Deleted!!","Way To Go Man!!"
3040 READ DATA FROM "" TO IOL=8000
3050 LET REFRESH_FLG=1
3060 LET NEXT_ID=TRM_CD.CTL; RETURN
3070 BADTRM:
3080 MSGBOX "No Such Record","Ooops!!"
3090 RETURN
```

# Attaching Code To Controls (cont’d)

7. Change **PROG01** to retrieve keys from the file **TERMCODE**

LOAD “PROG01”

Change lines 10 and 30 to read:

```
10 FL=UNT;OPEN (FL)”TERMCODE”
30 LIST_BOX LOAD ANIMAL.CTL,0,R$
```

8. Apply a Query to the **TRM_CD** field on the **TERMS** panel

Edit the **TRM_CD** multi-line:

a) Resize the multi-line to allow for the query button which will be drawn next to the input field
b) Select the ‘Query’ tab and enter the panel **TEST** as the ‘Query Panel’

The following action occurs:

The selection from the list box is passed to the multi-line which in turn executes the ‘On Select*’* logic for the **TRM_CD**, thus loading all screen inputs (REFRESH_FLG=1).

# *WINGRP and Grouping Controls

Controls may be grouped to simplify Showing, Hiding, Disabling, and Enabling multiple controls. A group is simply a name to which the control is associated. Any control or Graphic object can belong to one or more groups.

Once a control has been assigned to a group, the group can be manipulated by using the ***WINGRP** program.

The following six label entry points can be referenced in ***WINGRP**:

**ENABLE:**
**DISABLE:**
**SHOW:**
**HIDE:**
**LOCK:** (for multi-line controls only)
**UNLOCK:** (for multi-line controls only)

Example:

To enable a group of controls:

```
CALL "*WINGRP;Enable", xxxxxx.grp$
```
where xxxxxx is the group name

Note: The ENABLE CONTROL directive is used by *WINGRP to enable all controls belonging to group xxxxxx

### Using *WINGRP to enable/disable selected buttons on the TERMS panel

1. Assign a 'Group Name' called **MOOCOW** to the **&Write** and **&Delete** buttons

2. Set the 'Initially Disabled' attribute for the **&Write** and **&Delete** buttons

3. Load "PROG02"

4. Insert the following line of code:

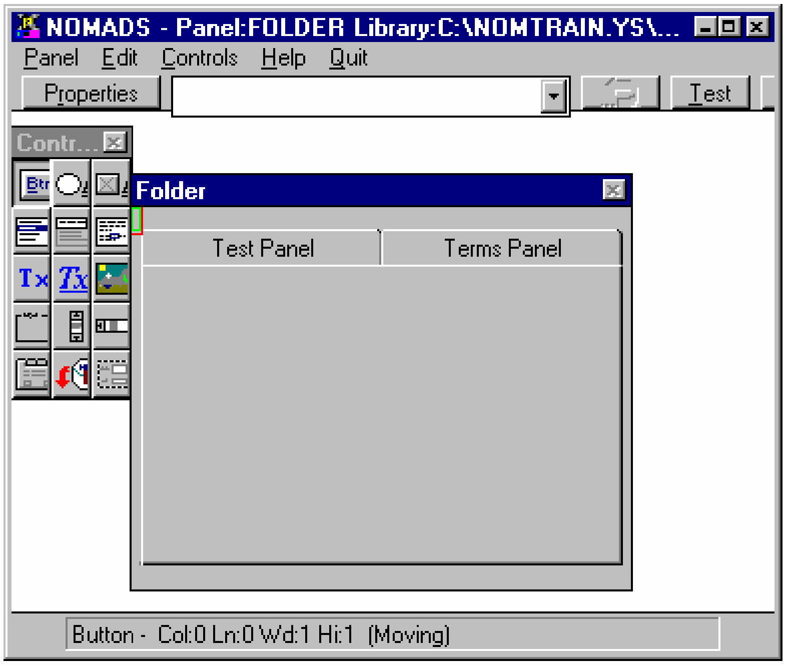
```
2025 IF NUL(TRM_CD$) THEN CALL "*WINGRP;DISABLE",MOOCOW.GRP$ ELSE CALL
"*WINGRP;ENABLE",MOOCOW.GRP$
```

5. Test **TERMS** panel

# Defining Folders

A folder control allows a Panel to include multiple sub-panels from which the user may select. A folder set appears like a series of File Folders on the screen from which the user may select either by clicking the mouse on the 'Tab' area of a folder or by using the arrow keys while focus is on the folder tabs.

### Create a new panel object called FOLDER



Button - Col:0 Ln:0 Wd:1 Hi:1 (Moving)

1. Make panel size larger than default. The main panel must be large enough to hold the sub-panels.

2. Activate the Dialogue and status bar attributes.

3. Create a Folder control within the panel – almost the full panel size.

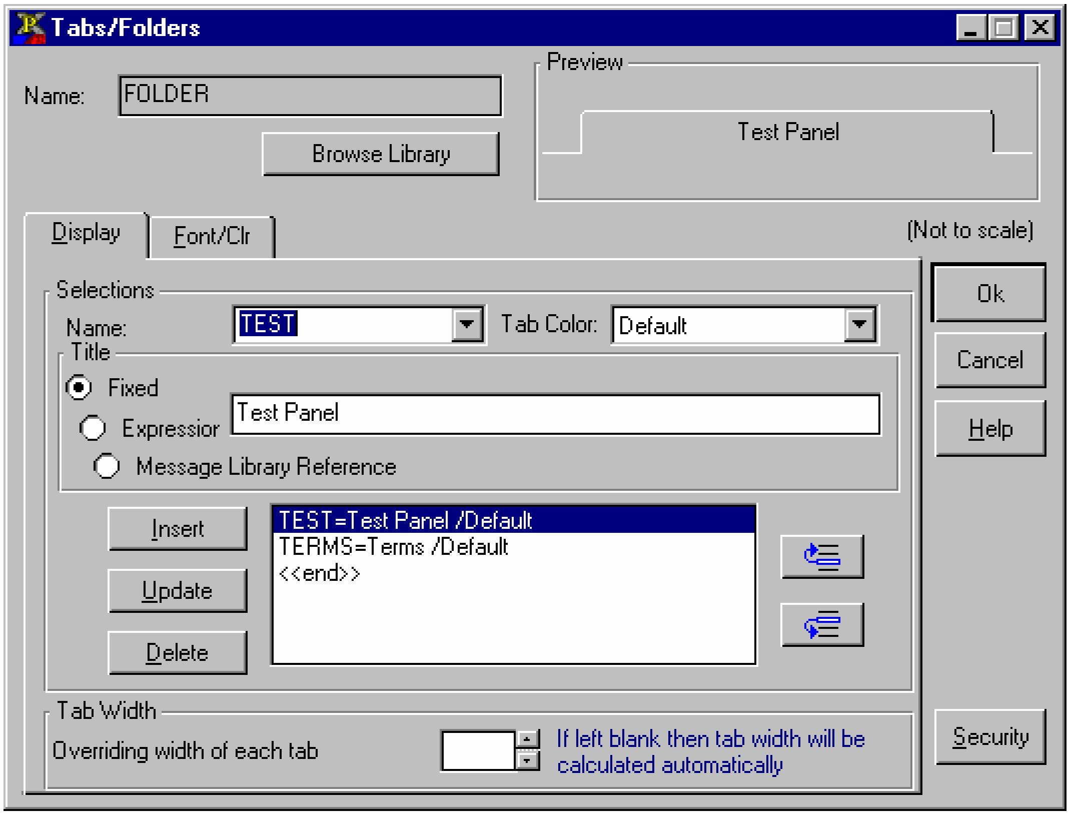
4. Add **TEST** and **TERMS** panels to the folder.

# Defining Folders (cont'd)

1. **Create a folder with 2 tabs:**

Name: **TEST** Title: **Test Panel**

Name: **TERMS** Title: **Terms Panel**



Name: FOLDER

Selections

Name: TEST

Tab Color: Default

Title

● Fixed

○ Expression

○ Message Library Reference

Test Panel

TEST=Test Panel /Default
TERMS=Terms /Default
<<end>>

Tab Width

Overriding width of each tab

If left blank then tab width will be calculated automatically

# Defining Folders (cont’d)

## DISPLAY OPTIONS

**Name:** This is the name of the control, there is also a browse library button available in order to browse the names of all the other folders in the library.  The name of the control also corresponds to a variable with the same name in ProvideX.

**Selections:**

Name: The Name of the panel to be displayed within the folder. The drop box will supply you with the names of all panels within the currently library.

Title: The Title to be displayed on the folder tab. The title can be defined as a fixed value, simple expression or can point to a reference in a user defined message library.

Colour: The colour of the whole tab.

**Tab Width:** The width of each tab.  By default the width is determined by the width of the folder and the number of tabs.

**Security:** Quick access to the NOMADS Security options

# Data Classes

Data classes can be used to simplify the entry and definition of common data elements. Data Classes are stored in the ProvideX keyed file ‘PROVIDEX.DCL’. Each class has its own unique key, consisting of 30 characters.

**Build a data class that will later be assigned to the data element: CUST_DATE in the Data Dictionary**

| | |
|---|---|
| Class Name: | **DATE** |
| Description: | **Date Class** |
| Internal Data Type: | **String** |
| Size: | **10** |
| Control Type: | **Input Field** |
| Input Length: | **10** |

In the Values/Validation tab enter:

| | |
|---|---|
| Validator: | ***win/date;validate** |
| Formatter: | ***win/date;display** |

**Class Name:** The key to the data classification file. It can be up to 12 characters name, consisting of the characters A-Z, 0-9, underscore or dot (blanks are not allowed. The class name must be unique.

**Description:** The generic description of the data class. It will be copied to the Data Dictionary description whenever a data class is used in the data dictionary elements definition.
Example: Date, GL Account Number

**Internal Data Type:** The format of the data when manipulated by the program. The two types of internal representations are String and Numeric.

**Size:** The maximum size that the data element will achieve while in memory. For string variables, this is the maximum string length. For numeric variables, the value is entered as nnn.dd where nnn is the maximum number of digits in front of the decimal point and dd is the number of digits after the decimal point (ie. The precision of the number)

**Default Control Type & Property:** The selected folder indicates the default type of control that will be used to represent data that belongs to this class. This is used primarily for the File Maintenance system.

**Data Class List:** A list of all data class definitions. Select a class from the list and the definition is displayed on the maintenance screen. It may then be updated or erased.

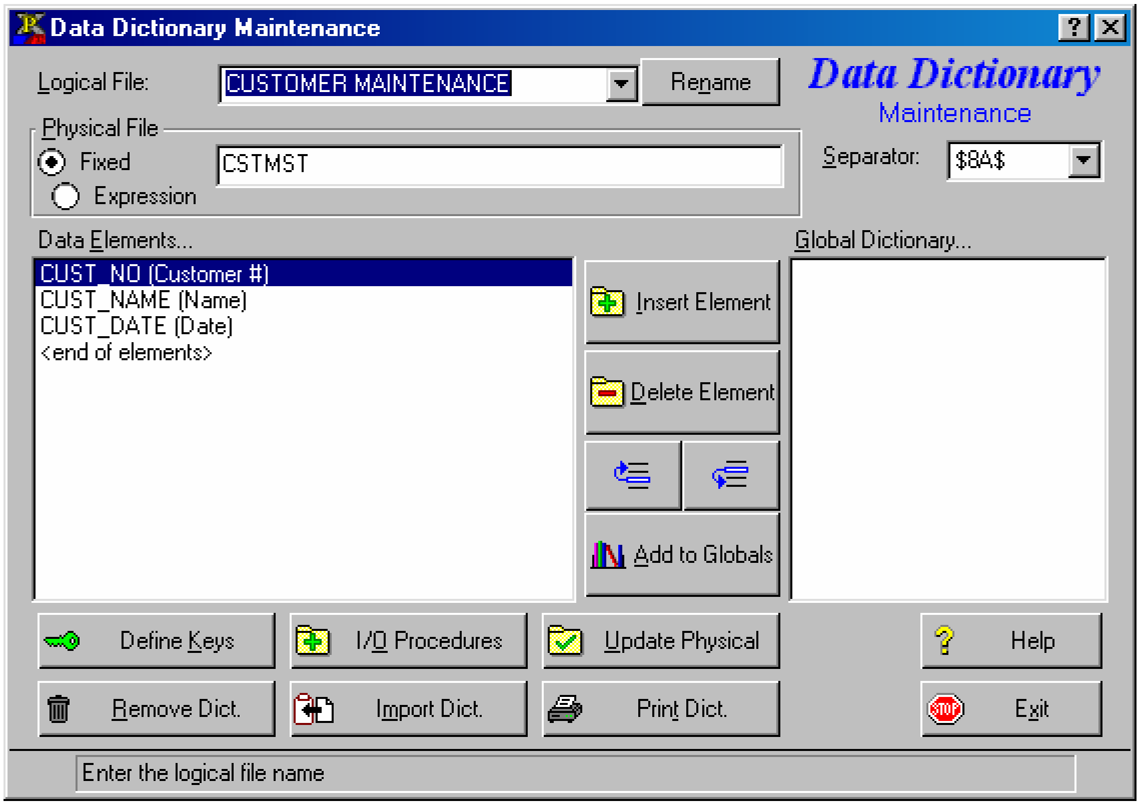
# Data Dictionary

The Data Dictionary is a utility that is used to define and maintain your file definitions. This file definition is stored in the ProvideX keyed files 'PROVIDEX.DDF' and 'PROVIDEX.DDE' as well as imbedded in the header record of the physical file, thus eliminating any IOLIST references in your programs.

1. Select Dictionary / Maintence from the NOMADS II menu bar.

   a) Logical File: **Customer Maintenance**

   b) Physical File: **CSTMST**



Data Dictionary Maintenance

Logical File: CUSTOMER MAINTENANCE

Physical File

● Fixed CSTMST

○ Expression

Separator: $8A$

Data Elements...

CUST_NO (Customer #)
CUST_NAME (Name)
CUST_DATE (Date)
<end of elements>

Enter the logical file name

# Data Dictionary (cont'd)

## HEADER INFORMATION

**Logical File:** The user defined descriptive name given to each file dictionary (max. length is 30 characters)

**Rename:** Press this button to change the logical file name.

**Physical File:** The path and name of the keyed file. (max. length is 23 character). The path and file name may also be provided in the form of an expression in the format of "=expression" which will be evaluated at run-time

Example: CST_MST, =%XFILE$

**Separator:** Select the field separator to be used in the file. The standard ProvideX separator is $8A$. The most common separators are provided in the list, although any hex value may be entered. Select NONE for binary file.

**Data Elements:** This is a list of the names of the data fields which make up the file.

**Global Dictionary:** The Global Dictionary is a collection of data elements which are common to several file. Defining the elements once and adding it to the Global Dictionary allows the user the ability to easily insert this element into any dictionary.

**Insert Element:** To add an element or field to a data dictionary, highlight the position in the Data Elements list where the element is to be inserted then press this button. The element description panel will then be displayed.

**Delete Element:** Highlight the element in the Data Elements list to be removed, and press this button. To delete an item from the Global dictionary, select Global Dictionary from the Logical File drop box, highlight the element and press this button.

**Move Keys:** Highlight an item in the Data Elements list, then press the up or down button to shift the element into the next or previous position.

**Add to Globals:** Highlight the field in the Data Elements list, and press this button to add the field to the Global Dictionary.

**Define Keys:** Select this button to define the file key(s)

**I/O Procedures:** This option is not currently available

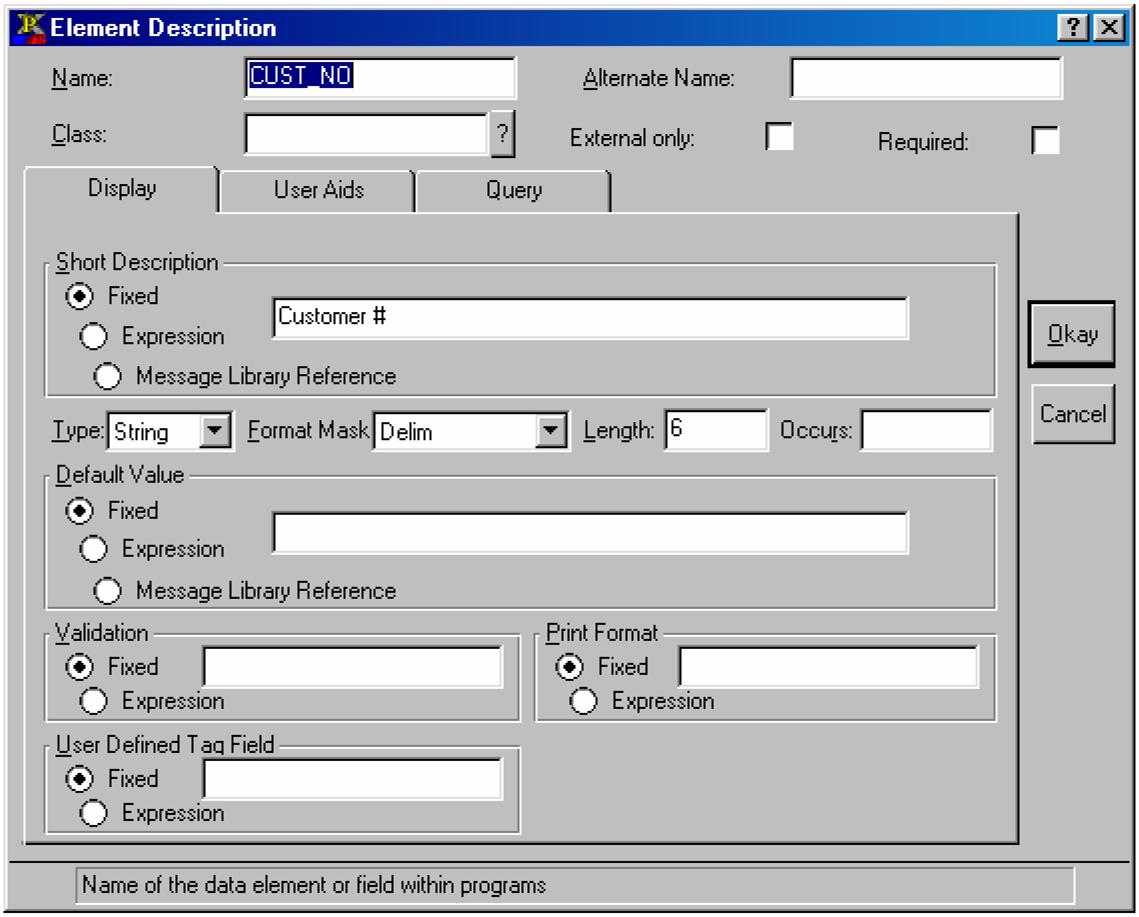
**Update Physical:** Press this button to create / update the data file.

**Remove Dict.:** Remove all entries from the PROVIDEX.DDE (data elements) and PROVIDEX.DDF (data files) files. This will not remove the data dictionary from the physical file.

**Import Dict:** Copy the elements from another data dictionary into the current one.

**Print Dict:** Print the detailed or summary data dictionary definition for the specified file. Definitions include file and field details.

# Data Dictionary (cont'd)



Name: CUST_NO
Alternate Name:
Class: ?
External only: □
Required: □

Short Description
● Fixed
○ Expression
○ Message Library Reference
Customer #

Type: String
Format Mask: Delim
Length: 6
Occurs:

Default Value
● Fixed
○ Expression
○ Message Library Reference

Validation
● Fixed
○ Expression

Print Format
● Fixed
○ Expression

User Defined Tag Field
● Fixed
○ Expression

Name of the data element or field within programs

# Data Dictionary (cont'd)

## DEFINING ELEMENTS

**Name:** The name of the element or field ( max. 30 characters)

**Alternate Name:** As alternate name for the field which will be used in the iolist (String fields must end with '$')
Example: Name: CST_ID
Alternate Name: A1$

**Class:** Name of the data class to be associated with this element. When a class is selected, default values from the class definition are inserted into the element definition.

**External Only:** This is a flag to indicate that the element forms part of an external key and is not embedded in the iolist. If the element forms part of an external key and is embedded in the iolist, do not check this box. An external key may be made up of segments which can to a mixture of external only and embedded elements.

**Required:** This is a flag to indicate that the element must contain data before the file can be written. (This field is used with File Maintenance)

**Short Description:** A brief description of the element (max. 20 characters)

**Type:** The type of field, either numeric or string.

**Length:** Maximum length of the data field.

**Occurs:** If the elements represents an array, enter the dimension.
Example: 1:3
(the table must still be dimmed within the program)

**Format Mask:** This defines the format for writing the field to the file
Delimited: Includes a field separator at the end of the element
Padded: Element is padded with spaces based on the defined length
Fixed: Pads the element to the specified length when writing to the file, but strips training spaces when being read from the file.
Substring: Similar to fixed, but will take up to but not including the field separator.
Last-substing: Similar to fixed, but will take up to and including the separator and then discard the separator.

**Print Format:** Format mask for output.

**Default Value:** The value to be used when the field is first initialized.

**Validation:** Comma separated validation rules
Example: 1-3,B,D

**Help-id:** This is the help file associated with the element.
Example: NOMADS.HLP

**Query-id:** This is the Query associated with the element

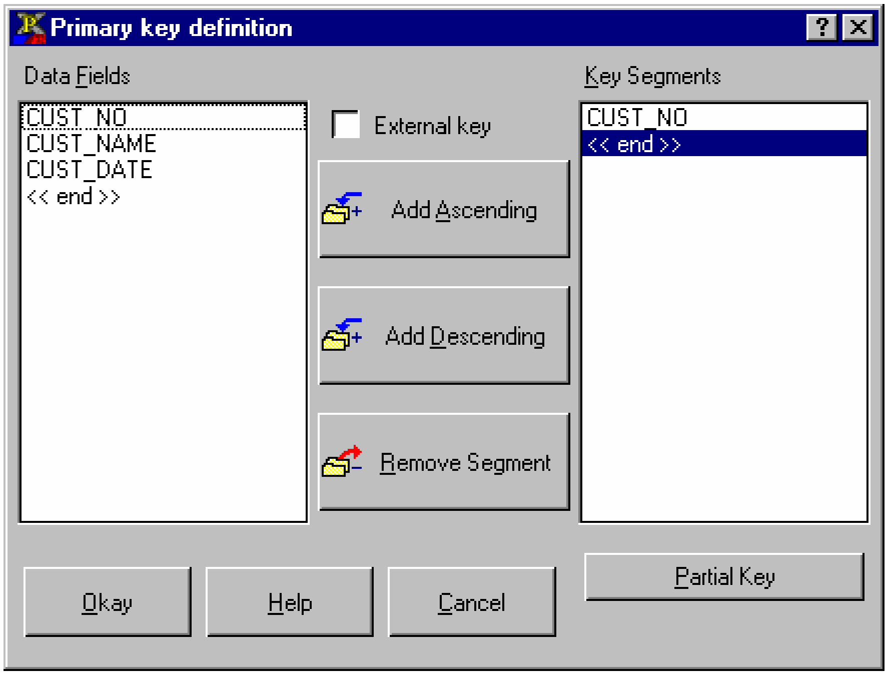
**Longwinded description:** Detailed description of the element (max. 240 characters)

# Data Dictionary (cont'd)

**Define the following three data elements:**

| **Name:** | **CUST_NO** | **CUST_NAME** | **CUST_DATE** |
|---|---|---|---|
| **Class:** | | | DATE |
| **Required:** | | Yes | Yes |
| **Short Description:** | Customer # | Name | Date |
| **Type:** | String | String | String |
| | | | |
| **Length:** | 6 | 30 | 10 |
| **Format Mask:** | Delim | Delim | Delim |
| **Query Tab:**<br>**Library:**<br>**Panel:** | MYLIB.EN<br>CSTQRY | | |
| **User Aids Tab:**<br>**Longwinded Description:** | | Please enter a name | Valid date format is YYYY/MM/DD |

# Data Dictionary (cont'd)



## DEFINING KEYS

**Data Fields:** This contains a list of the file elements or data fields which may be used to construct the key.

**External Key:** Check this box if the primary key for the file is external.

**Key Segments:** This contains a list all the file elements or data fields which currently make up the key.

**Add Ascending:** Highlight an item in the 'Data Fields' list and press this button to insert this element into the 'Key Segments' list. This portion of the key will be in ascending order.

**Add Descending:** Highlight an item in the 'Data Fields' list and press this button to insert this element into the 'Key Segments' list. This portion of the key will be in descending order.

**Remove Segment:** Highlight an item in the key segments list and press this button to remove a segment from the key definition.

**Partial Key:** This is a key segment composed of a portion of a data field (i.e. a substring). Highlight the item in the Key Segments list and press this key to indicate the field offset where the substring begins (the first position is offset 1) and the number of characters in the substring. If a length of zero is entered, the length will default to the total number of characters starting from the offset value.

**Define the following keys:**

1. Primary key is **CUST_NO**

2. Alternate key 1 is **CUST_NAME+CUST_NO**

## Embedding the Data Dictionary (pressing the 'Update Physical' button)

Once the data dictionary has been defined, press the ‘Update Physical’ button.

If the physical file does not exist, it is created and the dictionary embedded.

If the file exists but is empty, then it is erased, recreated, and the dictionary embedded.

If the physical file exists and its definition (key structure, separator and embedded data dictionary) corresponds with the new definition, the new data dictionary is embedded. Any existing data is not touched.

If the physical file definition does not correspond with the new definition, a warning message appears to advise you of the fact, and offers three options:

Convert existing data: Creates a new file, embeds the data dictionary, converts and copies the existing records to the new file, erases the original file and renames the new one.

Clear existing data: Erases the original file, creates a new one and embeds the data dictionary.

Rewrite data dictionary only: Embeds the new data dictionary in the existing file.

## Accessing the Embedded Data Dictionaries

1. Alt-tab to the other PVX session

2. Access the primary embedded Data Dictionary as follows:

```
OPEN(1,IOL=*)”CSTMST”;PRINT LST(IOL(1));CLOSE (1)
```

3. Access the alternate embedded Data Dictionary as follows:

```
OPEN(2,IOL=^)”CSTMST”;PRINT LST(IOL(2));CLOSE (2)
```

# File Maintenance

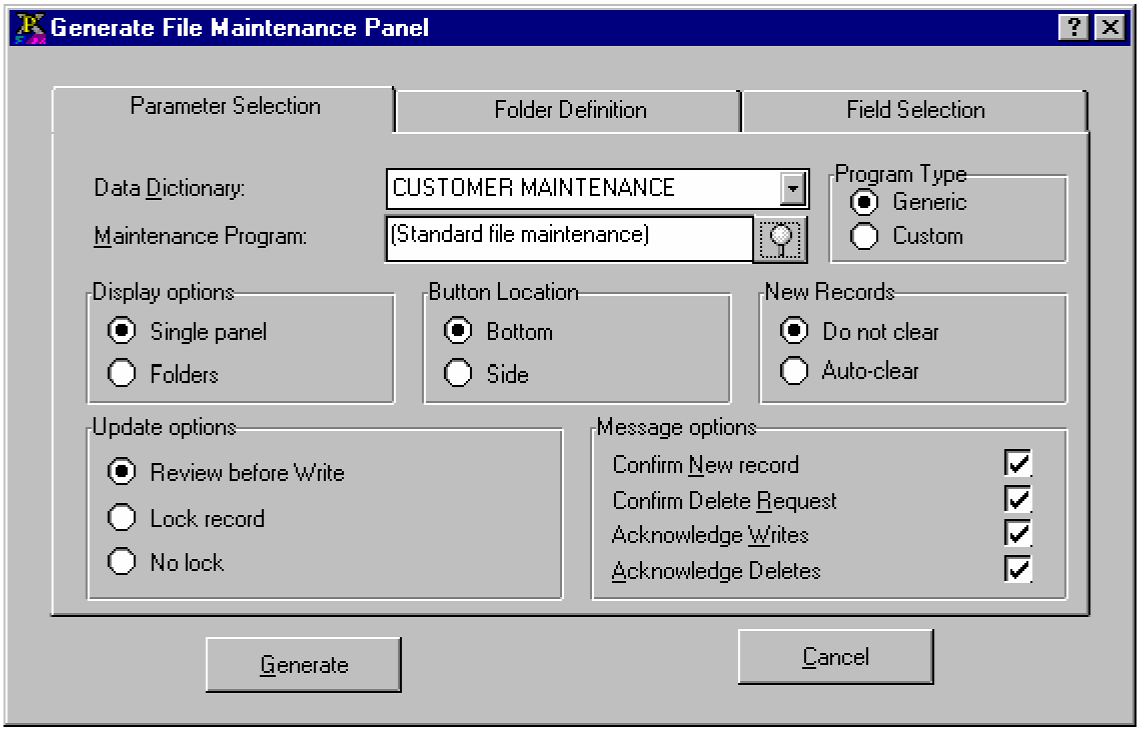
The NOMADS File Maintenance system automatically creates file maintenance panels for any file defined in the Data Dictionary. These panels can invoke either a generic or custom file maintenance program which allows the user to write, update and delete records, as well as browse the file.

### Defining a Standard File Maintenance

1. To invoke the File Maintenance system enter **CSTMAINT** in the 'Name' input field and press the 'File Maint' button

2. In the Paramter Selection tab, select **Customer Maintenance** from the Data Dictionary drop box.

3. Press the 'Generate' button



Data Dictionary: CUSTOMER MAINTENANCE

Maintenance Program: (Standard file maintenance)

Program Type: ● Generic ○ Custom

Display options: ● Single panel ○ Folders

Button Location: ● Bottom ○ Side

New Records: ● Do not clear ○ Auto-clear

Update options: ● Review before Write ○ Lock record ○ No lock

Message options:
Confirm New record ☑
Confirm Delete Request ☑
Acknowledge Writes ☑
Acknowledge Deletes ☑

# File Maintenance

## PARAMETER SELECTION

**Data Dictionary:** Select the file from the Data Dictionary List for which a File Maintenance Object is to be generated.

**Maintenance Program:** If desired, enter the name of a customized file maintenance that will be called by the File Maintenance object. The standard file maintenance program is *WIN/FLMAINT.

**Program Type:** Indicate whether the file maintenance object will use a pre-existing generic file maintenance program (such as the standard *WIN/FLMAINT program) or generate a unique custom file maintenance program specific to the related file.

**Display Options:** Indicate whether to generate a single panel or panel with multiple folders.

**Button Location:** Indicate whether the Write / Delete / Browse / Clear / Exit buttons are to be located at the bottom of the panel or along the side of the panel.

**New Records:** Indicate whether previous data is to be left or cleared from the data fields when a new record is being processed.

**Update Options:** Indicate whether a records contents are to be reviewed to determine if another user may have changed any fields before the record is written, or if the record is to be locked or not locked.

**Message Options:** Indicate whether messages should be displayed to confirm new records and / or deletions before they happen, and whether to acknowledge updates and deletions after they occur.

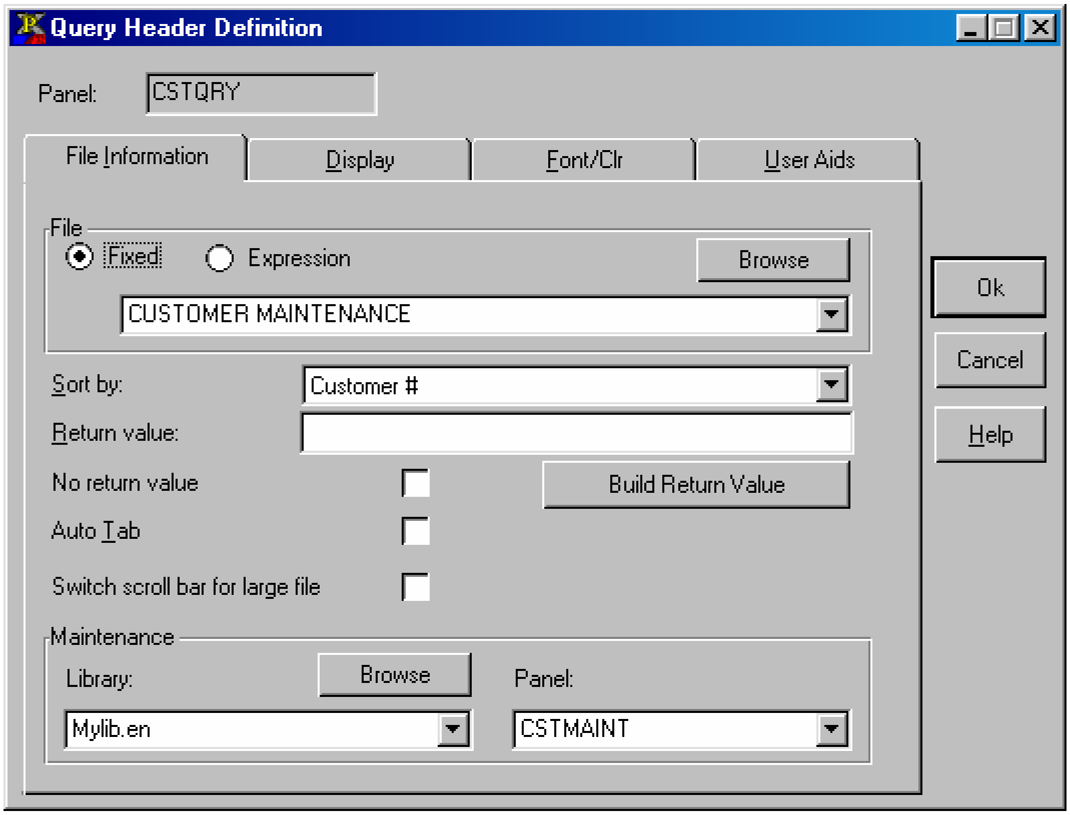
# Queries

A query consists of a Panel which displays records from a data file and returns a value associated with a record selected by the user (default is the value in the first column). A query may be defined for an existing Data Dictionary Definition or a file that supports multiple record types (non-normalized files).

### Defining a Query using a Data Dictionary Definition

1) To invoke the Query system enter **CSTQRY** in the 'Name' input field and press the 'Query Object' button

2) In the File Information tab, select **Customer Maintenance** from the 'File' drop box



Panel: CSTQRY

File

● Fixed ○ Expression

CUSTOMER MAINTENANCE

Sort by: Customer #

Return value:

No return value □

Auto Tab □

Switch scroll bar for large file □

Maintenance

Library: Mylib.en

Panel: CSTMAINT

# Queries (cont'd)

## FILE INFORMATION

**File:** The name of the primary query file to be displayed. By default the query system will provide a drop down list of all known files in the ProvideX Data Dictionary. The user may select one of the files presented or enter the name of the physical name of the file. If the file is not maintained in the Data Dictionary, the user will be required to manually define the data fields. Manually defined files may have embedded or external keys, or may be sort files. The 'Fixed' file option will use the filename as entered. The 'Expression' option indicates to NOMADS that the name of the file to be used is stored in a string variable or simple expression. The expression will be evaluated at run-time.

**Sort By:** Select the key by which the file is to be sorted in the query. (At run-time, you can override this setting by loading the %NOMADS_QUERY_KNO variable with an alternate key number.

**Return Value:** Define the value to be returned when a record is selected. The value is defined in terms of a ProvideX expression. You can return a value, which consists of a single field, or multiple fields concatenated in a format that you can then parse after the value is returned.

Examples:
```
PROD_ID$
CST_SMN$+","+CST_NAME$(1,1)
PAD(X1$,6)+PAD(X2$,12)+STR(LNK.AMT:"00000.00")
SALE.FLD#1*100
```

If left blank, the default return value is the value in the first column of the query. External keys can be accessed using the variable PRIME_KEY$. Return values can be defined by entering an expression in the 'Return value' input field, or by pressing the 'Build Return Value' button for assistance.

**Build Return Value:** This button will access a panel designed to assist in building the return value. (See Build Return Value panel description).

**No Return Value:** When checked, the argument containing the return value is not changed when exiting the query. When a query is invoked, a Start At value may be passed as an argument. When the query exits, the same argument is used for the return value. If this option is checked, rather than loading this argument with the value of the currently selected record, the argument is left unchanged. In this way, the value in a control with an attached query is not changed when the query ends.

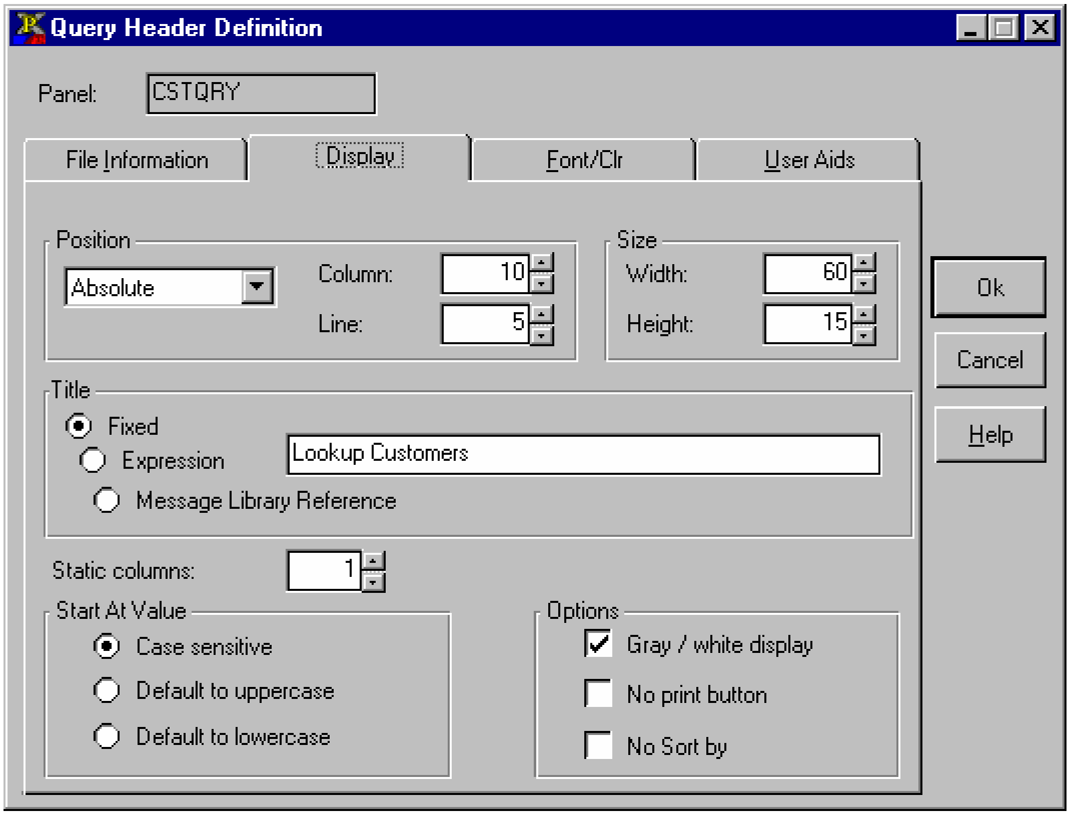
**Auto Tab:** If selected, a tab character is PREINPUT when the query is exited, so that focus is automatically sent to the next object.

**Read as Record** This option is available only to manually-defined queries, and is designed to force the query to read a file using READ RECORD. This enables the query to read and display data from files that have fixed-length records, which may contain binary data.

**Maintenance:** The name of the maintenance program for the file. If included, the user can invoke this program from the query to update the file by pressing the Maintenance button in the top right corner of the query panel when the query is run. The key of the currently selected record is passed to the maintenance program as ARG_1$, and the value returned in this argument is used to reposition the file. If no records exist to be displayed in the query, the Maintenance program is automatically invoked. If left blank, the Maintenance button is suppressed when the query is run.

# Queries (cont'd)

3) In the Display tab, enter **Lookup Customer** for the 'Title'



Panel: CSTQRY

Position: Absolute

Column: 10

Line: 5

Size

Width: 60

Height: 15

Title

● Fixed

○ Expression

○ Message Library Reference

Lookup Customers

Static columns: 1

Start At Value

● Case sensitive

○ Default to uppercase

○ Default to lowercase

Options

☑ Gray / white display

□ No print button

□ No Sort by

# Queries (cont'd)

## DISPLAY INFORMATION

**Position:** The position has 3 options:

Absolute: The panel will be positioned on the screen at exactly the line and column defined.

| | | |
|---|---|---|
| Example: | Position of Main Window: | Line 5, Column 5 |
| | Panel defined with a position of: | Line 2, Column 2 |
| | Panel will be displayed at: | Line 2, Column 2 |

Relative: The panel will be positioned on the line and column defined, but relative to the main window.

| | | |
|---|---|---|
| Example: | Position of Main Window: | Line 5, Column 5 |
| | Panel defined with a position of: | Line 2, Column 2 |
| | Panel will be displayed at: | Line 7, Column 7 |

Center: The panel will be centered on the screen.

**Column/Line:** The top left position of the panel.

**Width / Height** The width and height of the panels. Minimum width is 42 and minimum height is 10.

**Title:** The ‘Fixed’ title option will display the title as entered. The ‘Expression’ option indicates to NOMADS that the title to be displayed is stored in a string variable or simple expression. The expression will be evaluated when the panel is created.

**Static Columns:** The number of columns in a query which remain fixed when the display is shifted horizontally. (It is always the leftmost columns which are fixed).

**Start At Value:** Set the case sensitivity of the Start At Value. Case sensitivity will apply to *all* sort keys.

**Options:**

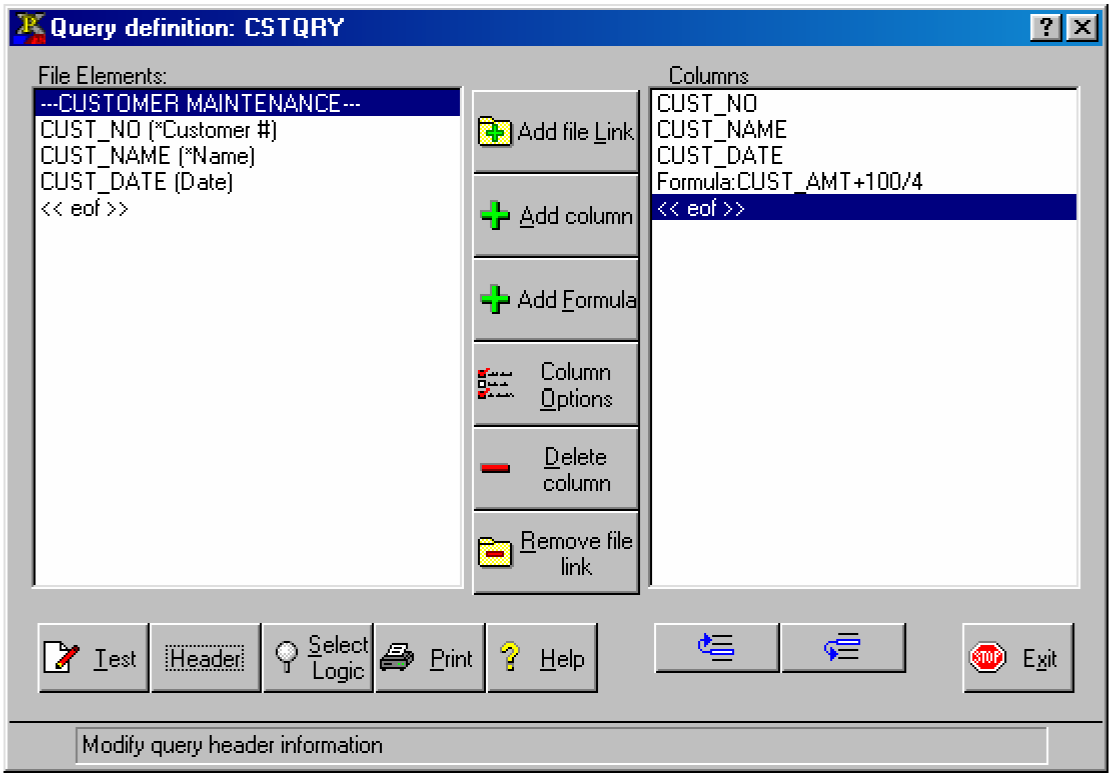
**Gray/White Display:** Display option – The records are displayed with alternating gray / white backgrounds. If this option is not selected, the background is white.

**No Print Button:** When checked, the query will not have a Print button.

**No Sort By:** When checked, the Sort By drop box is not displayed. The file will be sorted by the sort key selected in the File Information panel, and cannot be changed by the user at run-time.

# Queries (cont'd)

4) Select from the ‘File Elements’ list box the columns to display in the query



Query definition: CSTQRY

File Elements:

---CUSTOMER MAINTENANCE---
CUST_NO (*Customer #)
CUST_NAME (*Name)
CUST_DATE (Date)
<< eof >>

Columns

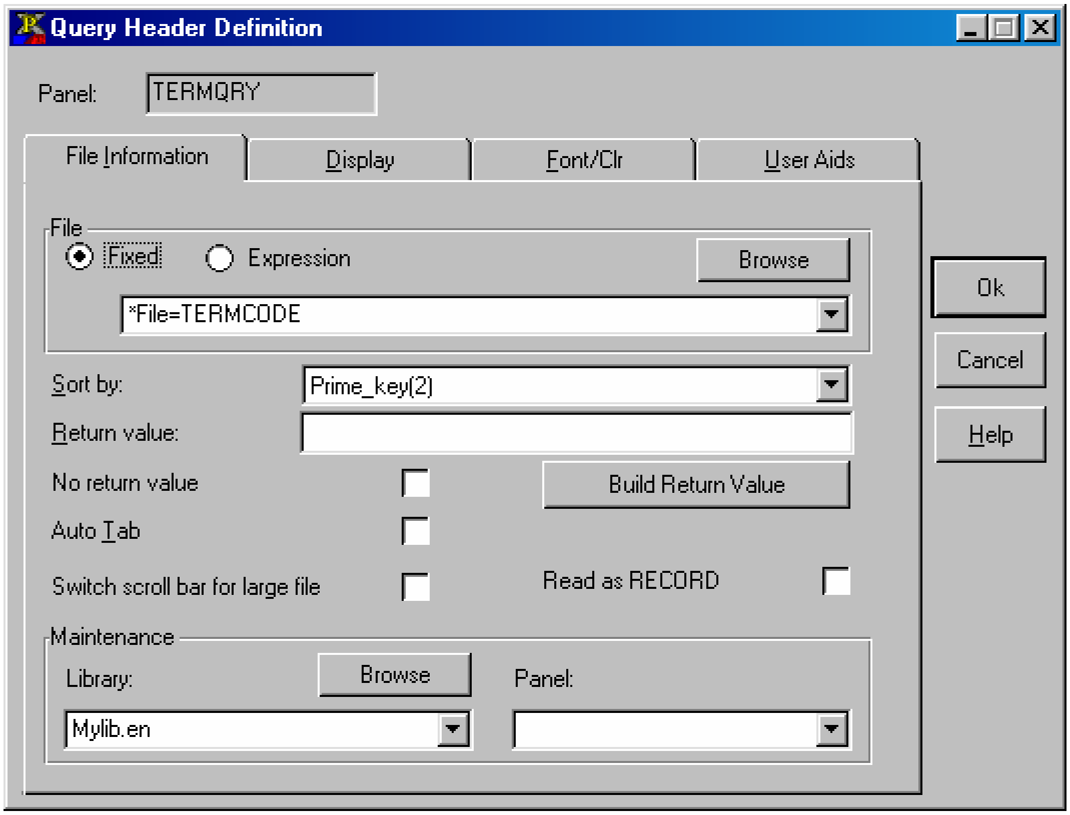
CUST_NO
CUST_NAME
CUST_DATE
Formula:CUST_AMT+100/4
<< eof >>

Modify query header information

# Defining A Manual Query For Non-Normalized Files

1. To invoke the Query system enter **TERMQRY** in the ‘Name’ input field and press the ‘Query Object’ button

2. In the File Information tab, enter **TERMCODE** as the ‘File’



Query Header Definition

Panel: TERMQRY

File

● Fixed ○ Expression

*File=TERMCODE

Sort by: Prime_key(2)

Return value:

No return value □

Auto Tab □

Switch scroll bar for large file □

Read as RECORD □

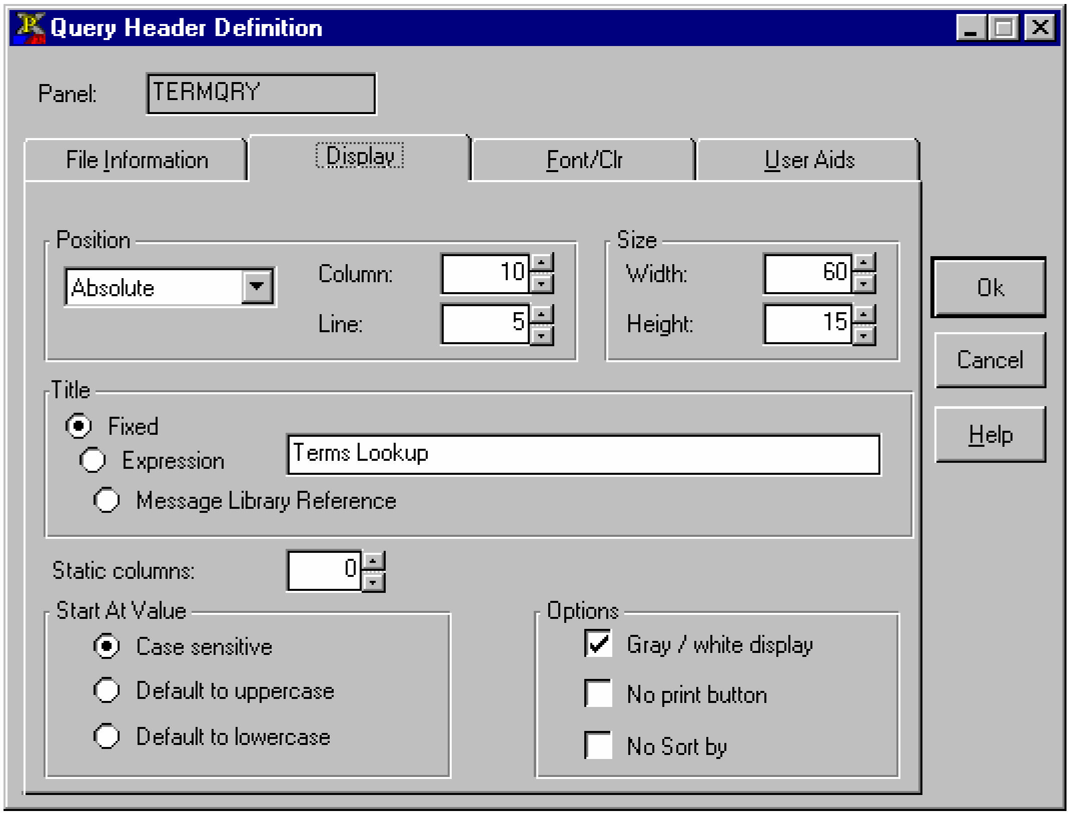
Maintenance

Library: Mylib.en

Panel:

# Defining A Manual Query For Non-Normalized Files (cont'd)

3. In the Display tab, enter **Terms Lookup** for the 'Title'



Query Header Definition

Panel: TERMQRY

Position: Absolute

Column: 10

Line: 5

Size

Width: 60

Height: 15

Title

● Fixed

○ Expression

○ Message Library Reference

Terms Lookup

Static columns: 0

Start At Value

● Case sensitive

○ Default to uppercase

○ Default to lowercase

Options

☑ Gray / white display

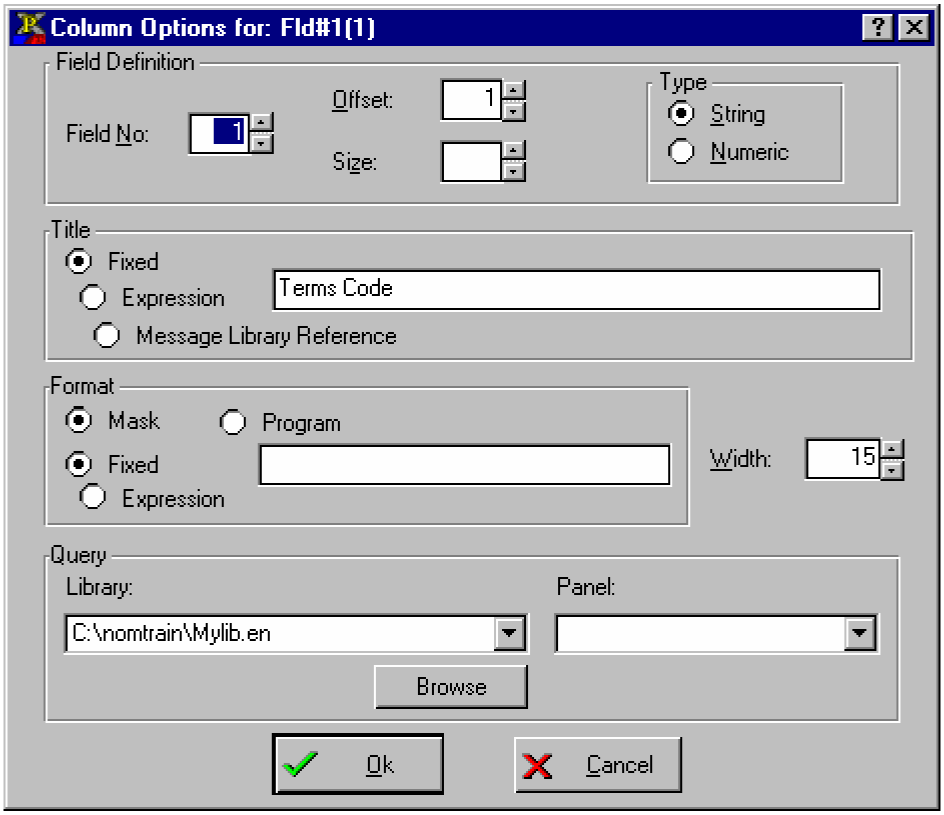
□ No print button

□ No Sort by

# Defining A Manual Query For Non-Normalized Files (cont'd)

**Building Columns for your Query:**

| | |
|---|---|
| Field No: | **1** |
| Offset: | **1** |
| Type: | **String** |
| Title: | **Terms Code** |
| Width: | **15** |
| Field No: | **2** |
| Offset: | **1** |
| Title: | **Description** |
| Type: | **String** |
| Width: | **30** |
| Field No: | **3** |
| Offset: | **1** |
| Type: | **String** |
| Title: | **Date** |
| Width: | **12** |
| Format: | **0000/00/00** |



Column Options for: Fld#1(1)

Field Definition

Field No: 1

Offset: 1

Size:

Type
● String
○ Numeric

Title
● Fixed
○ Expression
○ Message Library Reference

Terms Code

Format
● Mask ○ Program
● Fixed
○ Expression

Width: 15

Query

Library: C:\nomtrain\Mylib.en

Panel:

# User Defined Message Libraries

The Message Library Maintenance system allows the user to set up message libraries which can be accessed by user programs. In this way, you can set up standard messages that can be accessed by many programs. The message library system also facilitates the creation of a multi-lingual system.

### Message File Stucture

The messages are stored in a keyed file. The file consists of a 10 character key and it's corresponding text.

### Multi-lingual Capability

A system can be made multi-lingual by retrieving messages exclusively from message libraries, which may be translated using the Message Library Maintenance system. The same system can run in English, French, German, etc. simply by pointing to the corresponding message library.

The File Maintenance and Query systems available in NOMADS also have multi-lingual capabilities. Prompt and button messages, as well as all warning and error messages used in the user interfaces of these systems are retrieved from an English message library called *msglib.en. This library may be copied to another language file using the name *msglib.xxx (where xxx is a language suffix) and translated using Message Library Maintenance. Then set the language suffix in the NOMADS system defaults (Options/System Defaults) to the same suffix, and the Query and File Maintenance systems will retrieve their messages from this file. The language suffix is stored in the variable %NOMAD_DEF_SFX$.

# User Defined Message Libraries (cont'd)

### Creating a French Message Library

1. In the System Defaults panel set the library suffix to **.FR**

2. In the Message Library Maintenance Utility enter ***msglib.fr** as the 'Library File'.

3. Press the 'Reference File' button and enter ***msglib.en**

4. In the Copy/Merge Message Files tab enter ***msglib.en** in the 'Copy from' input field.

5. Press the **Copy** button to copy all message in the english library to the french library.

### Translating Messages

1. In the Message Maintenance tab change the 'Text' for the 'Message key' **&CLEAR**, to read **&Le Clear.**

2. Test the previously defined file maintenance panel: **CSTMAINT**

## Setting Up application specific message libraries

### Establishing your message library

In the Library Defaults Panel enter **mylib.msg** as the name of the **Message Library** you wish to reference. If the file does not exist, the system will create one.

### Accessing the message library

The **MSG( )** function is used to access your messages.
Identify the message by using its key as the function argument:

E.g. MSGBOX MSG("NO_DATA"),MSG("WARNING")

### Replacing all hard coded messages with the MSG function

1. Select Options/Message Manager in the NOMADS II menu bar to invoke the Message Maintenance Utility

2. Enter **mylib.msg** in the 'Library File' input field

3. Create 2 records:
   'Message Key' will be **&INIT**, 'Text' will be **We Are At The Init Logic**.
   'Message Key' will be **&INIT_TITL**, 'Text' will be **Init Box**.

4. Alt tab to other PVX session

5. Load "PROG02"

6. Change line 20 to read: MSGBOX MSG("&INIT"),MSG("&INIT_TITL")

7. Test the previously defined file maintenance panel: **TERMS**

# Advanced Data Dictionary and File Maintenance
# (Accessing cross-reference files)

1. **Build a Data Class that will be assigned to the data element: SUP_STS**

| | |
|---|---|
| Class Name: | **STATUS** |
| Description: | **Resolved** |
| Internal Data Type: | **String** |
| Size: | **1** |
| Control Type: | **Check Box** |
| Xlate Table: | **NY** |

2. **Create a new Data Dictionary.**

a) Logical File: **Support Maintenance**
b) Physical File: **SUP_MAST**

**Define Data Elements:**

| **Name:** | **SUP_ID** | **SUP_CUST** | **SUP_NAME** | **SUP_DESC** | **SUP_STS** |
|---|---|---|---|---|---|
| **Class:** | | | | | STATUS |
| **Short Description:** | Call Id | Customer Id | Support Person | Description | Status |
| **Type:** | String | String | String | String | String |
| | | | | | |
| **Length:** | 4 | 6 | 30 | 30 | 1 |
| **Format Mask:** | Delim | Delim | Delim | Delim | Delim |
| **Query Tab:**<br>**Library:**<br>**Panel:** | MYLIB.EN<br>SUPQRY | | | | |

3. **Define keys:**

a) Primary key is **SUP_ID**
b) Alternate key is **SUP_NAME+SUP_ID**

4. **Create the file with the embedded iolist:** Press 'Update Physical'

5. **Generate a file maintenance panel called SUPMAINT**

| | |
|---|---|
| Data Dictionary: | **Support Maintenance** |
| Program Type: | **Custom** |
| Maintenance Program: | **SUP_MAINT** |

# Advanced Data Dictionary and File Maintenance (cont’d)
# (Accessing cross-reference files)

## Using the Validator and Formatter Programs to access secondary files

The File Maintenance system generates a panel for a single Data Dictionary definition. Therefore to display data from a secondary file, you must customize the program generated from File Maintenance.

One way to do this is to assign a validator (input program) and a formatter (output program) to the field.

Here are the necessary steps to accomplish this task:

1. Edit the multi-line called **SUP_CUST** in the panel **SUPMAINT**
2. Select the ‘Validation’ tab
3. In the ‘Validator’ input field enter **SUP_MAINT;PROCESS_CUST**
4. In the ‘Formatter’ input field enter **SUP_MAINT;PROCESS_CUST**
5. Select the ‘Logic’ tab
6. Select the ‘Execute’ function from the ‘On Select’ drop box and enter **REFRESH_FLG=1** in the input field
7. Create a new multi-line called **%CUST_NAME** to display the customer name from the **CUST_NAME** field in the cross reference file ‘**CSTMST**’. Place the new multi-line to the right of the **SUP_CUST** input field.
8. Make the new multi-line **Borderless** and **Locked** with **no Tab Stop**
9. Alt tab to other PVX session
10. Load the custom File Maintenance program called ‘**SUP_MAINT**’
11. Enter the following lines of code:

```
20010 PROCESS_CUST:
20020 ENTER IN_DATA$,ERR_MSG$,ERR=*NEXT
20030 LET %CUST_NAME$=””;IF IN_DATA$=”” GOTO DONE
20040 X=FFN(“CSTMST”)
20050 IF X=-1 THEN LET XX=HFN; OPEN (XX,IOL=*)”CSTMST”;X=XX
20060 READ (X,KEY=IN_DATA$,DOM=NOT_THERE)
20070 %CUST_NAME$=CUST_NAME$
20080 GOTO DONE
20090 NOT_THERE:
20100 ERR_MSG$=”Customer does not exist”
20110 GOTO DONE

24010 DONE:
24020 IF XX CLOSE (XX)
24030 EXIT
```

# Queries – Creating Links To Cross-Reference Files

The Query file link allows you to retrieve additional data from associated cross-reference files. There is a maximum of 9 cross-reference files.

The Primary file will be **SUP_MAST**. The secondary file will be **CSTMST**.

Here are the necessary steps to link the 2 files:

1. Create a new query object called **SUPQRY**

2. Invoke the Query system. In the 'File Information' tab select **Support Maintenance** from the 'File' drop box

3. Elements to select from the primary file:
   **SUP_ID**
   **SUP_CUST** (this element will be the link to the secondary file)

4. Press the 'Add File Link' button to build the link

5. Select **Customer Maintenance** from the 'File' drop box

6. Enter **SUP_CUST$** in the 'Key Definition' field

7. Press the 'Write' button to create the link

8. The 'File Elements' list box in the 'Query Definition' panel will now have the data elements for the secondary file

9. Select **CST.CUST_NAME** from the elements list

## Using a customized menu bar to link to panels

1. Create a new panel object called: **MENU1** that will contain the menu bar.

2. In the Attributes tab of the Panel Header Definition screen activate the **Dialogue** and **Menu Bar** options.

3. Create a button:

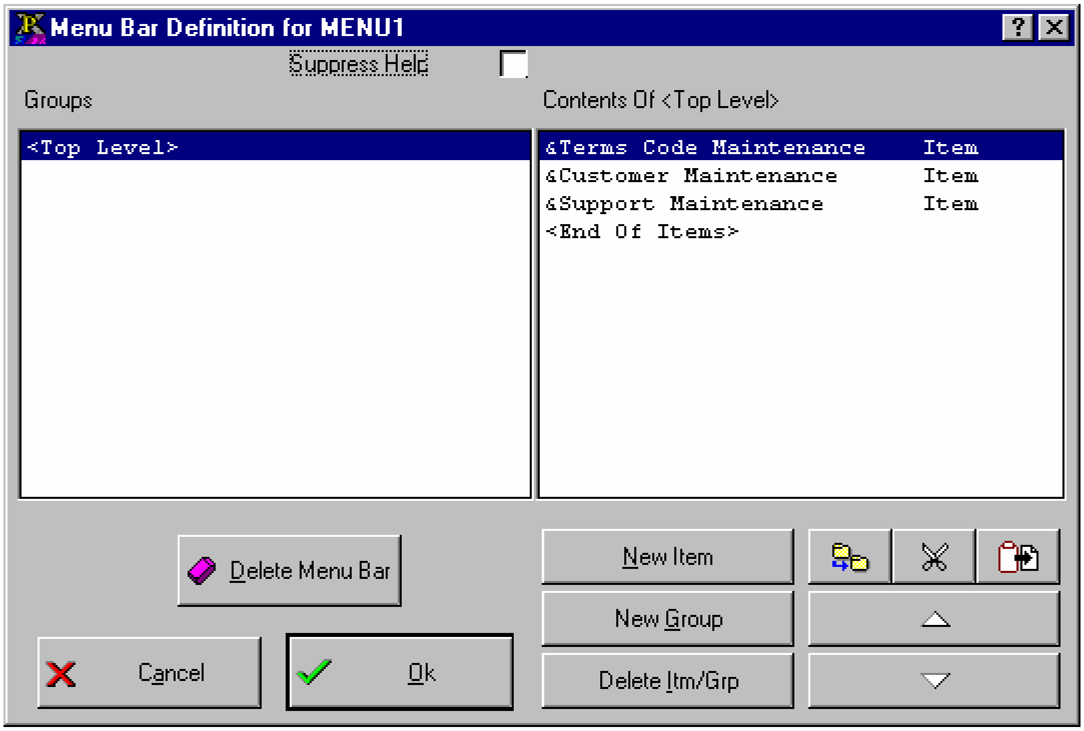
   Text: **&Exit**
   (In the **Logic** tab, select the 'END' function from the 'When Button Pressed' drop box)

4. Use the Image control to assign a bitmap as the background for the panel. Have the image fill the entire panel.
   Example: Picture Path = **C:\WINDOWS\CLOUDS.BMP**

5. To enter the Menu Bar Maintenance Utility select **P**anel/**M**enus

**Menu Bar Definition**

| | | | |
|---|---|---|---|
| Item: | **&Terms Code Maintenance** | Function: Link | **"TERMS"** |
| | **&Customer Maintenance** | | **"CSTMAINT"** |
| | **&Support Maintenance** | | **"SUPMAINT"** |



Menu Bar Definition for MENU1

Suppress Help □

Groups

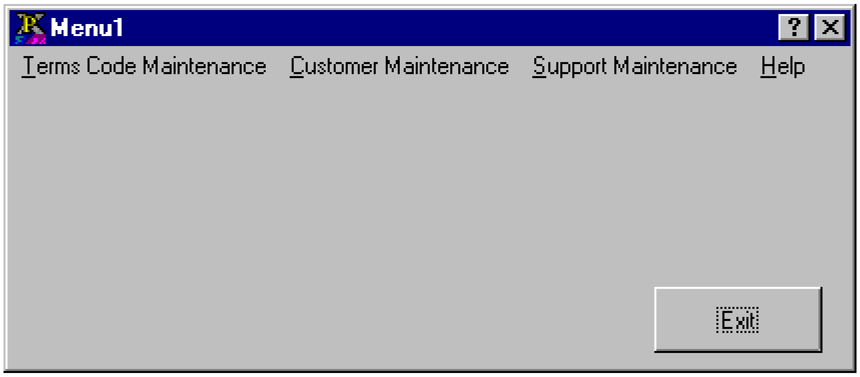
```
<Top Level>
```

Contents Of <Top Level>

```
&Terms Code Maintenance      Item
&Customer Maintenance        Item
&Support Maintenance         Item
<End Of Items>
```

# Putting A Front-End On Your Application (cont'd)

**Sample Results:**



## Using buttons to link to panels

1. Create a new panel object called **MENU2.**

2. In the Attributes tab of the Panel Header Definition screen activate the **Dialogue** option.

3. Create 4 buttons:

   Button 1:

   Text: **&Exit**
   (In the **Logic** tab, select the 'END' function from the 'When Button Pressed' drop box)

   Button 2:

   Text: **&Terms Code Maintenance**
   (In the **Logic** tab, select the 'Link' function from the 'When Button Pressed' drop box and enter "**TERMS"** in the input field)

   Button 3:

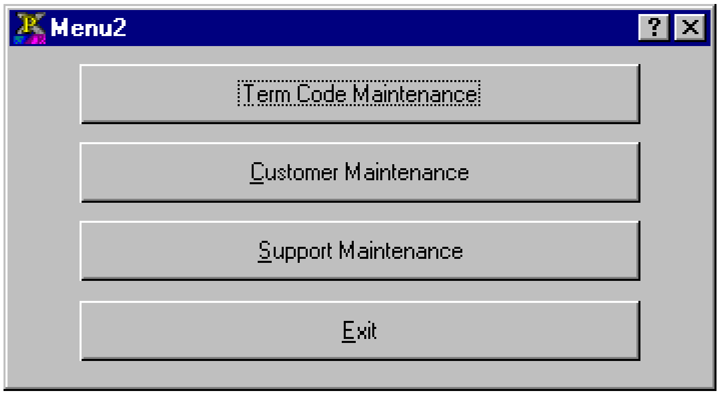
   Text: **&Customer Maintenance**
   (In the **Logic** tab, select the 'Link' function from the 'When Button Pressed' drop box and enter "**CSTMAINT"** in the input field)

   Button 4:

   Text: **&Support Maintenance**
   (In the **Logic** tab, select the 'Link' function from the 'When Button Pressed' drop box and enter "**SUPMAINT"** in the input field)

4. Use the Image control to assign a bitmap as the background for the panel. Have the image fill the entire panel.
   Example: Picture Path = **C:\WINDOWS\CLOUDS.BMP**

**Sample Results:**



## Running The Application From Your Desktop

One of the simplest means of launching an application is directly from the desktop. This can be done by creating shortcuts that run PVXWIN or PVXWIN32 and the required panel.

For example, to run the panel **MENU1** from the library **MYLIB.EN**, you would create a shortcut to run…

```
C:\PVX\PVXWIN32.EXE *winstar -arg MENU1 MYLIB.EN
```

For the ‘Start In’ directory enter: **C:\NOMTRAIN**

This shortcut can then be placed directly on the end-user’s desktop or in a folder or folder structure to make up a menuing system. One nice feature of this approach is that the user can have custom menus and can copy the frequently used short cuts directly to their task bar and/or desktop.

When running an application directly from a short cut, we suggest that you make the panels ’Dialogue’ and specify that the initial window be Minimized within the shortcut.

## PROGRAMMING INTERFACE

To interface NOMADS object to programs the designer specifies PERFORM or CALL as the Process Logic for a control.

**When using the PERFORM, the following variables are provided to the program:**

| **Control Variables** | **Description** |
|---|---|
| %NOMADS_AUTO_QRY | If non-zero, Nomads will Automatically initiate the Query Window if a value fails the Input Validation Routine |
| %NOMADS_FKEY_HANDLER$ | Name of program to handle function keys |
| %NOMADS_NOTEST | Disable use of Test screen |
| %NOMADS_ONEXIT$ | Program called on Exit of any panel |
| %NOMADS_PROCESS$ | Pre-processing for panel name & library |
| %NOMAD_CENTER_WDW | If set to non-zero, center next panel |
| %NOMAD_ENTER_TAB | If set to non-zero,  ENTER = TAB |
| %NOMAD_ESC_SEL | If set to non-zero, then process field when ESC is hit. |
| %NOMAD_MENU$ | Contains the Menu Group or item to be pasted to a Menu Bar definition in a different panel. |
| %NOMAD_MSGMNT$ | Contains the name of the message library currently being updated in the Message Library Maintenance. |
| %NOMAD_OPEN_LOAD | If set to non-zero the panel library will be opened with an “OPEN LOAD” directive otherwise a standard “OPEN” will be used.  The OPEN LOAD can be used to improve panel display performance. |
| %NOMAD_PRG_CACHE | Program cache count |
| %NOMAD_QRY_BTN$ | Bitmap shown on query button (Default is ?) |
| %NOMAD_QRY_TIP$ | Tip displayed when mouse is focused on query button |
| %NOMAD_QRY_WIDE | Width of query panel |
| %NOMAD_QUERY_KNO | If non-zero *winqry uses this value to override the key number used to sort the query. |
| %NOMAD_QUERY_RETKNO | *winqry returns the value of the last key number used to sort the query. |
| %NOMAD_RELATIVE_WDW | If set to non-zero, the next window will be relative to current window |
| %NOMAD_SCRIPT_FN | Script playback file # |
| %NOMAD_STK$ | Error handler stack |
| %NOMAD_TIMEOUT | Input timeout value |
| %NOMAD_WIN_VER | Windows Version |
| %NOMAD_XCHAR | Text width (single character value in pixels) |
| %NOMAD_XMAX | Screen width (single character value in pixels) |
| %NOMAD_YCHAR | Text height (single character value in pixels) |
| %NOMAD_YMAX | Screen height (single character value in pixels) |
| %SCR_3D | Use 3D flags |
| %SCR_DEF_ATTR$ | Default Attributes |
| %SCR_DEF_H_FL$ | Default library file |
| %SCR_DEF_H_ID$ | Default library key |
| %SCR_LIB | Screen library file # |
| %SCR_LIB$ | Screen library path |
| _EOM$ | Contains the hex code  of the key that triggered the last control. E.g. $02$ = double click |
| ARG_1$ … ARG_20$ | Any arguments passed on the PROCESS/CALL *WINPROC |

## PROGRAMMING INTERFACE

| | |
|---|---|
| CHANGE_FLG | This field will contain a non-zero value whenever a control changes on a panel. NOMADS will automatically increment this variable whenever the value changes thus allowing the application to determine if there has been any changes to the panel contents. The programmer is responsible for resetting this field to zero when required. |
| CMD_STR$ | The logic processing to be done next. To terminate the current panel set this variable to "END" |
| DEFAULT_PROG$ | Default program for screen |
| DISP_CMD$ | Command after display of panel |
| EXIT_CMD$ | Command prior to EXIT |
| FLDR_DEFAULT_PROG$ | Default program for folder |
| FOLDER_ID$ | Contains the current sub-panel object name (if any) |
| ID | Contains the current controls CTL identifier |
| ID$ | the name of the current control |
| IGNORE_EXIT | If set to a non-zero value the window close/F4 key is ignored and if set during panel Exit logic, the panel will not be closed |
| IGNORE_EXIT$ | If set to a “Y” value the window close/F4 key is ignored and if set during panel Exit logic, the panel will not be closed |
| INITIALIZE_FLG | If set to: 1 - resets screen 2 - resets all folders 3 – resets current folder |
| INIT_TEXT$ | This Variable can be assigned the initial list for loading a control (Note: the INIT_TEXT$ variable will only be looked at once, before the control is drawn) |
| INIT_VAL$ | This Variable can contain the default value of a control. |
| JMP_LVL | Jumpto level |
| MAIN_SCRN_K$ | Screen key |
| MNU_LN$ | Menu definition |
| NEXT_FOLDER | Set to FLDR.xxxx.CTL to change folders |
| NEXT_ID | If changed by the program this will represent the CTL identifier of the next control to receive focus. |
| NO_FLUSH | Don’t clear input when NEXT_ID is set |
| PRIOR_VAL | Prior Value |
| PRIOR_VAL$ | Prior Value |
| QRY_VAL$ | This variable will contain the value that was returned from the query. |
| REFRESH_FLG | If set to a non-zero value all fields will be updated on the screen |
| REPLACEMENT_FOLDER$ | Name of folder to replace current (available in pre-display logic) |
| REPLACEMENT_LIB$ | Name of library to replace current (available in pre-display logic) |
| REPLACEMENT_SCRN$ | Name of screen to replace current (available in pre-display logic) |
| SCRN_ID$ | Contains the panel object name |
| SCRN_K$ | Key prefix for screen |
| SCRN_LIB$ | Contains the object library pathname |
| TAB_TABLE$ | Tab table |
| xxxxxx.CTL | Control CTL values for each screen |
| xxxxxx {.VAL}$ | Control value (the .VAL can be suppressed) |
| xxxxxx.TAG$ | User defined tag value as per the control definition |

## PROGRAMMING INTERFACE

*ggggg*.GRP$ — Contains a string defining the controls and display elements that belong to the group *ggggg*. These variables are used in conjunction with the subprogram *WINGRP for handling groups.

FLDR.xxxxxx.CTL — Contains the CTL value used to reference the Tabs within a Folder.