Developing Dialog Boxes for Personal Designer

Revision 1.2
July 20, 2011

Table of Contents

Prologue

The glossary and abbreviations appear at the beginning of this manual because a good understanding of these terms is **required** in order to understand what follows.

In order to obtain sufficient understanding of dialog boxes and how to develop them, it is recommended that the reader scrutinize the first sections of this manual:

Glossary and Abbreviations
Files that define Dialog Boxes
Properties
The Command File
Statements that define Dbox Characteristics
Statements that define Field Characteristics
Event Routines

The section on Functions and the section on Procedures can be lightly scanned on the first reading. This will provide a general understanding of the functionality that is available. Portions of these sections can be reread at the time specific information is needed on any function or procedure.

Glossary and Abbreviations

dbox Dialog Box, an on screen menu which interacts with the user by displaying information, accepting input and initiating event routines. A dialog box is graphically defined by a rectangular string entity carrying a property with the name DLG_BOX.

ER Event Routine, a set of ICL statements associated with a dbox or field that is executed when the indicated event occurs.

Example: fld_dig . . . end
Any ICL statements between the "fld_dig" statement and the "end" statement comprise the field digitize event routine and will be executed when a digitize is entered while the cursor is over the field in question.

The ER prefix for a dbox is "dlg_", the ER prefix for a field is "fld_".
The event type suffixes are: begin, dig (digitize), dsp (display), end, in_to, hlp (help), out_of, and over.

field An area within a dbox (usually only a portion of the whole dbox) in which graphical or textual information may be displayed and from which an event routine (ER) may be initiated. ER's are usually initiated by digitizing while the cursor is over the field, however, other events such as passing the cursor into or out of the field may also be used to initiate an ER. A field is

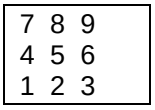
graphically defined by a rectangular string entity carrying a property with the name DLG_FLD.

Synonyms: button, recess

ICL Interface Command Language, the programming language used to control the behavior of dboxes. ICL is similar to UPL. ICL code is compiled by the PD command Build Dialog (not an external compiler).

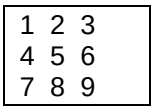
ID An ICL identifier statement identifies the ICL code that follows (until the next identifier statement of the same type) as pertaining to a particular dbox or field. The keywords for the two types of ID statements are DLG for a dbox and FLD for a field. The keyword is always followed by a unique number. Other ICL statements that define the characteristics of the dbox or field usually appear on the same line with the ID statement (or immediately following the ID statement).

K-position The position within a rectangular area. There are nine positions related to the numbering of a computer keyboard numeric key-pad.
Also K-pos.



MP Modifier Processor.

P-position The position within a rectangular area. There are nine positions related to the numbering of a telephone key pad.
Also P-pos.



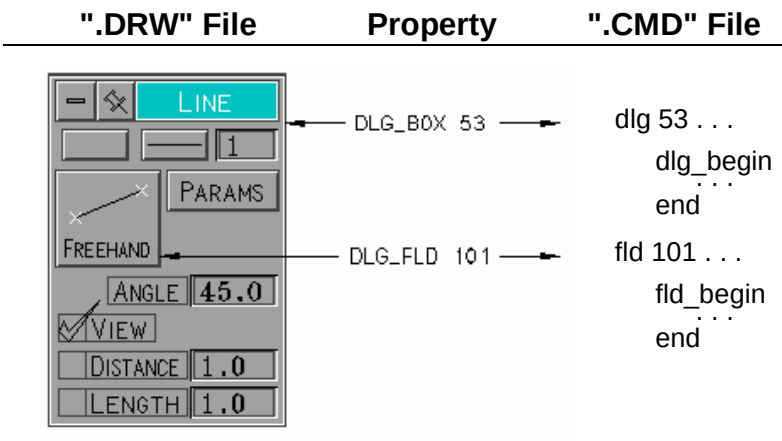
Statement An ICL keyword and all related parameters and/or arguments (if any). Because ICL is free format, a statement does not need to occupy a line by itself.

Files that define Dialog Boxes

Dboxes are comprised of two basic parts:

1. A graphical image defined in a Personal Designer ".DRW" file.
2. ICL instructions defined in a text file with the extension ".CMD".

Graphical elements in the .DRW file are tied to ICL instructions in the .CMD file by a keyword and an index number. In the .DRW file the keyword and index are stored in a property subrecord on a string entity. The ICL file is free format, however, the keyword is usually found at the beginning of a line followed by the index number.



Any fields or graphics that fall within a dbox boundary and are on the same layer as the dbox boundary will become part of the image and definition for that dbox.

Properties

There are three different properties used to define different dbox elements in the drawing file:

	Property Name	Property Contents
1.	DLG_BOX	<index> <bkg color>
	Identifies a dbox boundary.	
2.	DLG_FLD	<index> <flag>
	Identifies a field boundary within a dbox.	
3.	DLG_ICN	<index> <bkg color> <pos> <xoff> <yoff>
	Identifies an icon boundary. An icon is a graphical image that may be displayed in a field or as a cursor.	

<index> A unique numerical value. For DLG_BOX and DLG_ICN <index> must be unique within a menu file. For DLG_FLD <index> need only be unique within a dbox. For DLG_ICN <index> values 1 through 256 are reserved for cursor icons. Icon five (5) is the standard menu cursor.

<bkg color> Dbox or icon background color. If the value is negative, the background color is not used.

<flag> This is used only for fields. A value of 1 (one) indicates that the string boundary coordinates should be rounded to the nearest character cell so that textual or numerical information will fit properly within the field boundary. A value of 0 (zero) will cause the string boundaries to be rounded to the nearest pixel.

<pos> Note: <pos>, <xoff>, and <yoff> are used only with cursor icons.

<pos> is a P-position within the icon rectangle indicating the cursor "Hot Spot".

If <pos> is 0 (zero) then <xoff> and <yoff> are the offset of the Hot Spot in pixels from the lower left corner of the icon, otherwise <xoff> and <yoff> are ignored.

The Command File

The command file is a free format ASCII text file containing Interface Command Language (ICL) instructions that control the behavior of dboxes.

ICL is a programming language that has some similarity to UPL. ICL supports common programming constructs such as conditionals, looping, and many intrinsic functions and procedures.

A typical ICL file has a simple structure; one or more "Identifier" (ID) statements each of which is followed by one or more "Event Routines" (ER).

The first ID statement is always for a dbox which identifies all of the statements that follow (until the next dbox ID) as pertaining to a particular dbox. The dbox ID is followed by one or more dbox ER's and/or one or more field ID's and their related ER's.

ICL supports include statements which are used to insert ICL code from one file into another file. ICL also supports define statements which are used to substitute one text string for another. Defined words and phrases can make ICL code both more understandable and more terse.

An abbreviated example of an ICL file is shown below:

```
1  include define.cmd                        --standard definitions

3  define angle_select$ "fldiv(line_box,101) = 7" --local definitions
4  define angle_field$  "10"
5  define angle_on$     "set_fld_active(line_box,angle_field,1,1)"
6  define angle_off$    "set_fld_active(line_box,angle_field,1,0)"

7  dlg line_box position -5 mp_style_active cur_style dlg_arrow

8  dlg_out_of                    --dbox event routine
9   if vnphlp = 2001 then        --for cursor moving out of dbox
10   leave_action
11   if in_mp then
12    input(':')
13   endif
14   exec_fld_cmd(-1, 0, -4)
15   set_inactive_cur
16  endif
17 end

18 dlg_dig                   --dbox event routine for digitize in dbox,
19  activate                --but not over an active field
20  set_active_cur
21  if angle_select then
22   angle_on
23  else
24   angle_off
25  endif
26  if not_in_mp or mphlp <> line_box then
27   if vnphlp = 2001 then
28    input(':')
29   else
30    input("#13#INS LIN "@mp_char)
31   endif
32  endif
33 end

34 fld 101 type 32            --field identifier line (creation method)

35 fld_value 1 7             --field parameter initialization
36  1 2 3 4 5 6 7
37  331 332 333 334 335 336 337
38 end

39 fld_begin                 --field event routine for dbox startup
40  set_fldiv(-1, -1, mpsxt(1))
41 end

42 fld_end                   --field event routine for dbox termination
43  if angle_select then
44   set_mpsxt(1)
46   set_mps(10,10, 1)
47  else
48   set_mpsxt(fldiv(-1,-1))
49   set_mps(10,10, 0)
50  endif
51 end
```

The principal elements shown in the ICL example file are:

1. Standard Definitions (line 1)
   An "Include" statement is used to insert ICL code from another file.

```
include define.cmd                          --standard definitions
```

2. Local Definitions (lines 3 through 6)
   The "define" statement is used to replace one text string with another. This makes ICL code both more understandable and more terse.

```
define angle_select$ "fldiv(line_box,101) = 7" --local definitions
define angle_field$  "10"
define angle_on$     "set_fld_active(line_box,angle_field,1,1)"
define angle_off$    "set_fld_active(line_box,angle_field,1,0)"
```

3. Dbox Identifier (line 7)
   The "dlg" statement followed by "line_box" (which is defined as "53" in an include file) identifies all of the statements that follow (until the next dlg statement) as pertaining to dbox 53. The other statements on the dbox ID line set up the default characteristics of the dbox. These will be discussed in greater detail later.

```
dlg line_box position -5 mp_style_active cur_style dlg_arrow
```

4. Dbox Event Routines (lines 8 through 33)
   The first dbox ER is dlg_out_of which is executed when the cursor passes out of the dbox. The contents of this ER are enclosed in a conditional that limits its execution to the times when the INSert LINe command is active.

```
if vnphlp = 2001 then ... endif
```

   This ER accomplishes three tasks:

   a. Transfer the settings in the dbox fields to the MP by executing all of the fld_end ER's (field end).

```
exec_fld_cmd(-1, 0, -4)
```

   b. Change the dbox cursor type to inactive.

```
set_inactive_cur
```

   c. Switch the PD command from the MP to getdata.

```
if in_mp then
 input(':')
endif
```

   The second dbox ER is dlg_dig which is executed when a digitize is entered and the cursor is over the dbox, but not over an "active" field. This ER also accomplishes three tasks:

a. Change the dbox cursor type back to active.

```
set_active_cur
```

b. Reactivate all of the fields in the dbox. The conditional limits the activation of the angle value field to the case when the angle creation method is selected.

```
activate
if angle_select then
 angle_on
else
 angle_off
endif
```

Notice the improved readability due to the use of defined phrases. Without the use of these phrases the preceding code would read as follows:

```
set_fld_active(-1,0,1,1)
if fldiv(53,101) = 7 then
 set_fld_active(53,10,1,1)
else
 set_fld_active(53,10,1,0)
endif
```

c. If the current PD command is INSert LINe then pass from getdata to the MP. Otherwise, activate the INSert LINe command.

```
if not_in_mp or mphlp <> line_box then
 if vnphlp = 2001 then
  input(':')
 else
  input("#13#INS LIN "@mp_char)
 endif
endif
```

5. A Field Identifier (line 34)
The "fld" statement followed by the number 101 identifies all of the statements that follow (until the next fld statement) as pertaining to field 101. The "type" statement and the "fld_value" statement set up the default characteristics of the field. These will be discussed in greater detail later.

```
fld 101 type 32       --field identifier line (creation method)
fld_value 1 7         --field parameter initialization
 1 2 3 4 5 6 7
 331 332 333 334 335 336 337
end
```

6. Field Event Routines (lines 35 through 51)
The first field 101 ER is executed when the dbox is activated (put on the screen). This ER initializes the field value based on the mutually exclusive line creation modifiers (e.g. freehand, horizontal, vertical, etc.).

```
fld_begin            --field event routine for dbox startup
 set_fldiv(-1, -1, mpsxt(1))
end
```

The second field 101 ER is executed when the statement

```
exec_fld_cmd(-1, 0, -4)
```

in the dlg_out_of ER is executed. That is, this field ER is invoked by a statement in a completely different ER. Normally the fld_end ER is executed when the dbox is removed. However, in this case it is desirable to have all the fld_end ER's executed when the cursor passes out of the dbox (from the MP to getdata) so that the field settings can be transferred to their corresponding modifier values.

```
fld_end                  --field event routine for dbox termination
 if angle_select then
  set_mpsxt(1)           --mutually exclusive line creation modifiers
  set_mps(10,10, 1)      --angle method modifier on
 else
  set_mpsxt(fldiv(-1,-1))--mutually exclusive line creation modifiers
  set_mps(10,10, 0)      --angle method modifier off
 endif
end
```

Statements that define Dbox Characteristics

Kommentar [WC1]: Page: 10

There are three statements that define the default characteristics of a dbox. These statements must appear immediately after the dlg statement. They are:

active

The active statement initializes the dbox attribute bit table. The keyword is followed by a binary number that sets the bits. For example, active 0b1100001000010000 sets bits 1, 2, 7, and 12 on, and the other bits off. The following is the meaning for the set (ON) condition of these bits:

1 dlg_active_bit
The dbox is active. When a dbox is active the following ER's are enabled: dlg_dig, dlg_in_to, dlg_out_of, dlg_over and dlg_hlp.

2 Automatically remove this dbox when the command processor passes the end of the dispatch routine, that is just before the ">>" prompt appears again.

3 Do not remove this dbox when its parent is removed. The dbox parent is another dbox from which the command to activate the child dbox was initiated.

4 Do not make this dbox inactive when a child dbox is activated.

5 Ignore the "extra active" attribute bit (field bit 2) of all fields in this dbox. This field attribute bit is described below in the section on Statements that define Field Characteristics.

6 Automatically remove this dbox when the cursor passes out of the dbox.

7 Automatically "gray" fields in this dbox when the field active bit (bit 1) is cleared (set to zero). Automatically "ungray" fields in this dbox when the field active bit is set to one.

8 Automatically deactivate all fields in this dbox (set the field active bits to zero) whenever it is at least partially covered by another dbox. Any fields for which the field "extra active" bit (field bit 2) is set are not deactivated, unless bit 5 in the dbox attribute table is set.

9 dlg_auto_into_activate
Automatically activate the dbox (set bit 1 on) when the cursor passes into the dbox.

10 dlg_auto_outof_deactivate
Automatically deactivate the dbox (set dlg_active off) when the cursor passes out of the dbox.

11 Automatically activate all fields in this dbox (set the field bit 1 on) when the cursor passes into the dbox.

12 dlg_auto_outof_deactivate_flds
Automatically deactivate all fields in this dbox (set the field bit 1 off) when the cursor passes out of the dbox.

13 reserved

14 Clip the graphics window to the largest area not including this dbox.

15 Used internally, DO NOT MODIFY! Set if dbox is covered.

16 Used internally, DO NOT MODIFY! Set if cursor is in dbox.

cur_style

The cur_style statement initializes the cursor style. The key word is followed by the number of a dbox cursor icon (1 through 256).

position

The position statement defines the position of the dbox and the cursor at the time the dbox is activated (put on the screen). The number following the keyword is a value between -9 and 9.

Value	Position Effect
-9 thru -1	The dbox is activated at the position it occupied when it was last on the screen and the cursor is moved to the absolute value P-position within the dbox.
0	The dbox is activated at the position it occupied when it was last on the screen and the cursor is not moved.
1 thru 9	The cursor is not moved and the dbox is activated such that its P-position is justified on the cursor.

Statements that define Field Characteristics

Kommentar [WC2]: Page: 12

There are six statements that define the default characteristics of a field. These statements must appear immediately after the fld statement. They are:

active

The active statement initializes the field attribute bit table. The keyword is followed by a binary number that sets the bits. The following is the meaning for the set (ON) condition of these bits:

1 fld_active
The field is active. When a field is active the following ER's are enabled: fld_dig, fld_in_to, fld_out_of, fld_over and fld_hlp.

2 Do not allow the fld_active bit to be changed to off (zero).

3 Override the function of the dbox auto-gray bit (dbox bit 7). That is, DO NOT automatically make this field gray because the field active bit (bit 1) has been cleared (set to 0).

4 This is a "dummy" field. That is there is no connection to a string entity in the drawing file. Dummy fields are used to store data, and dummy field ER's are used like subroutines.

5 Enable character wrap for this field.

6 Enable word wrap for this field.

7 Enable vertical scrolling for this field.

8 Enable horizontal scrolling for this field.

9 Display a "text cursor" at the end of the text displayed in this field.

10 reserved

11 Enable "Button Push". This is usually used with buttons rather than recess fields and will cause the button to move slightly as a visual acknowledgment of having been picked.

12 Used internally, DO NOT MODIFY! Set if field gray status needs to be changed when the field is uncovered.

13 Used internally, DO NOT MODIFY! Set if field display (contents) needs to be changed when the field is uncovered.

14 Used internally, DO NOT MODIFY! Field active bit (bit 1) value prior to automatic change.

15 Used internally, DO NOT MODIFY! Set if field is gray.

16 Used internally, DO NOT MODIFY! Set if cursor is in field.

bkg_color

The bkg_color statement initializes the background color for a field that is to display textual or numerical information. The keyword is followed by the color number.

cur_style

The cur_style statement initializes the field cursor style that is displayed when the cursor is over the field in question. The keyword is followed by the icon number of the desired cursor.

fgr_color

The fgr_color statement initializes the foreground color for a field that is to display textual or numerical information (i.e. the text color). The keyword is followed by the color number.

hlight_color

The hlight_color statement initializes the field highlight color. The keyword is followed by the color number. Usually color 15 is used if highlighting is desired and color zero (0) is used if highlighting is not desired.

type

The type statement initializes the field type. The keyword is followed by the type number which determines many field characteristics and behaviors. Some field types can store data. Some field types have a default ER for display (fld_dsp) and digitize (fld_dig).

It is important to note that when a toggle field value is set (with the set_fldiv function), the value entered is the toggle element number, NOT the actual value. However, when the field value is accessed (with the fldiv function) the value returned is the actual field value.

The field types and the default ER's are:

1 String field.
Data stored: Character data.
fld_dsp ER: Display stored character data.
fld_dig ER: None.

2 2-byte integer field.
Data stored: 2-byte integer data.
fld_dsp ER: Display stored integer data.
fld_dig ER: Activate and take input from the integer numeric key pad dbox (503, if not found then dbox 505 is used).

3 4-byte integer field.
Data stored: 4-byte integer data.
fld_dsp ER: Display stored integer data.
fld_dig ER: Activate and take input from the integer numeric key pad dbox (503, if not found then dbox 505 is used).

4 Real field.
Data stored: Real numeric data.
fld_dsp ER: Display stored real numeric data.
fld_dig ER: Activate and take input from the real number key pad dbox (505).

11 String toggle field.
Data stored: Character data.
fld_dsp ER: Display stored character data.
fld_dig ER: Toggle to the next string value. Several string values are stored and the field value is automatically switched from one to the next. This field type should be accompanied by a fld_value statement to initialize the field data.

12 2-byte integer toggle field.
Data stored: 2-byte integer data.
fld_dsp ER: Display stored integer data.

fld_dig ER: Toggle to the next value. Several integer values are stored and the field value is automatically switched from one to the next. This field type should be accompanied by a fld_value statement to initialize the field data.

13 4-byte integer toggle field.
Data stored: 4-byte integer data.
fld_dsp ER: Display stored integer data.
fld_dig ER: Toggle to the next value. Several integer values are stored and the field value is automatically switched from one to the next. This field type should be accompanied by a fld_value statement to initialize the field data.

14 Real toggle field.
Data stored: Real numeric data.
fld_dsp ER: Display stored real numeric data.
fld_dig ER: Toggle to the next value. Several real values are stored and the field value is automatically switched from one to the next. This field type should be accompanied by a fld_value statement to initialize the field data.

21 String/String toggle field.
Data stored: Character data.
fld_dsp ER: Character data which may be different from the stored character value.
fld_dig ER: Toggle to the next string value. Several string values are stored along with corresponding display strings. The field value is automatically switched from one value to the next and its corresponding display string is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

22 2-byte Integer/String toggle field.
Data stored: 2-byte integer data.
fld_dsp ER: Character data.
fld_dig ER: Toggle to the next integer value. Several integer values are stored along with corresponding display strings. The field value is automatically switched from one value to the next and its corresponding display string is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

23 4-byte Integer/String toggle field.
Data stored: 4-byte integer data.
fld_dsp ER: Character data.
fld_dig ER: Toggle to the next integer value. Several integer values are stored along with corresponding display strings. The field value is automatically switched from one value to the next and its corresponding display string is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

24 Real/String toggle field.
Data stored: Real numeric data.
fld_dsp ER: Character data.
fld_dig ER: Toggle to the next real value. Several real values are stored along with corresponding display strings. The field value is automatically switched from one value to the next and its corresponding display string is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

31 String/Icon toggle field.
Data stored: Character data.
fld_dsp ER: Icon.
fld_dig ER: Toggle to the next string value. Several string values are stored along with corresponding display icons. The field value is automatically switched from one value to the next and its corresponding display icon is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

32 2-byte Integer/Icon toggle field.
Data stored: 2-byte integer data.
fld_dsp ER: Icon.
fld_dig ER: Toggle to the next integer value. Several integer values are stored along with corresponding display icons. The field value is automatically switched from one value to the next and its corresponding display icon is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

33 4-byte Integer/Icon toggle field.
Data stored: 4-byte integer data.
fld_dsp ER: Icon.
fld_dig ER: Toggle to the next integer value. Several integer values are stored along with corresponding display icons. The field value is automatically switched from one value to the next and its corresponding display icon is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

34 Real/Icon toggle field.
Data stored: Real numeric data.
fld_dsp ER: Icon.
fld_dig ER: Toggle to the next real value. Several real values are stored along with corresponding display icons. The field value is automatically switched from one value to the next and its corresponding display icon is displayed. This field type should be accompanied by a fld_value statement to initialize the field data.

100 Null field. (All ER's for this type must be defined in ICL code.)
Data stored: None.
fld_dsp ER: None.
fld_dig ER: None.

101 Output field.
Data stored: Character data.
fld_dsp ER: None.
fld_dig ER: Issues the stored character string into the PD command stream.

103 Filelist field.
Data stored: None.
fld_dsp ER: None. File list is displayed either via make_file_list or dsp_fld.
fld_dig ER: Issues the file name and <CR> into the input stream. See filelist.cmd for example usage.

110 Assist text field.
Data stored: None.
fld_dsp ER: Displays current 'Assist Text' if any.
fld_dig ER: None.

Event Routines

Kommentar [WC3]: Page: 19

There is a default event action for some events if a specific ER for that event is not specified in the .cmd file. Note that the default action can be invoked from an ER using the exec_fld_cmd.

An ER is executed or its default event action is done when its specific event occurs, these are:

dlg_begin
When a dbox is activated, before the dbox image is put up on the screen and before any fld_begin ER's are executed. No default action.

dlg_dig
When a digitize is entered and the cursor is over the dbox, but not over an active field in the dbox. No default action.

dlg_dsp
When a dbox is activated, the very last ER type to be executed when putting up a dbox on the screen. No default action.

dlg_end
When a dbox is removed from the screen. No default action.

dlg_hlp
When the cursor crosses the dbox boundary while moving into the dbox, this ER is executed after dlg_in_to. No default action.

dlg_in_to
When the cursor crosses the dbox boundary while moving into the dbox. No default action.

dlg_out_of
When the cursor crosses the dbox boundary while moving out of the dbox. No default action.

dlg_over
When the cursor is moving anywhere within the dbox boundary. No default action.

dlg_start_up
When the menu is first activated or restored. This is where menu initialization happens. This ER is not associated with any dbox and should be in only one .cmd file. No default action.

fld_begin
When a dbox is activated, before the dbox image is put up on the screen, but

after the dlg_begin ER. If the dbox number is in the range 1 through 499 (which is reserved for dboxes that are intended to interact with the MP) and the field index number is in the range 1 to the number of MP words for the command in question, the field value will be initialized from the MP value for the MP word with the field's index number.

fld_dig
When a digitize is entered and the cursor is over the field. No default action.

fld_dsp
When a dbox is activated, after the dbox image is put up on the screen. The default action is to display the appropriate data from using the field's current value. This is either an icon or some text (which could represent a numerical value).

fld_end
When a dbox is removed from the screen. If the dbox number is in the range 1 through 499 (which is reserved for dboxes that are intended to interact with the MP) and the field index number is in the range 1 to the number of MP words for the command in question, the MP value for the MP word with the field's index number will be set from the field's value.

fld_hlp
When the cursor crosses the field boundary while moving into the field. This ER is executed after fld_in_to. No default action.

fld_in_to
When the cursor crosses the field boundary while moving into the field. No default action.

fld_out_of
When the cursor crosses the field boundary while moving out of the field. No default action.

fld_over
When the cursor is moving anywhere within the field boundary. No default action.

The order of events during dbox activation is as follows:

1. The dlg_begin ER.

2. All fld_begin ER's. The order of execution of the individual field ER's is dependent on the order that the field definition string entities appear in the PD part database, not the order that the field definitions appear in the ICL file.

3. The dbox base image is put on the screen.

4. All fld_dsp ER's. These have the same order dependence.

5. The dlg_dsp ER.

It is important that ICL code that causes any display activity is NOT put in either the dlg_begin or fld_begin ER's because this display activity will happen behind the dbox rather than on the dbox base image.

The order of events when the cursor is moving is described by the following pseudo code:

```
if only_dlg is on, then if the cursor is not in the only_dlg dbox then no action is taken.

if the cursor was previously in a field area of any dbox and the dbox dlg_active bit is set
and the fld_active bit is set then
        execute that field's fld_out_of ER

if the cursor was previously in any dbox and the dlg_active bit for that dbox is set then
        if the dlg_auto_outof_deactive bit is set then
                set the dlg_active bit off
        if the dlg_auto_outof_deactive_fld bit is set then
                set the fld_active bit off for all fields in the dbox,
                unless their fld_extra_active bit is set
        the dlg_out_of ER is executed
        if the dlg_auto_outof_takedown bit is set then
                delete the dbox

if the cursor is currently in a dbox then
        if the dlg_active bit or dlg_auto_into_activate bit is set then
                if the cursor was not previously in the dbox then
                        if the dlg_auto_into_activate bit is set then
                                set the dlg_active bit on
                        if the dlg_auto_into_activate_fld bit is set then
                                set the fld_active bit on for all fields in the dbox
                        execute the dlg_into ER
                        execute the dlg_help ER

                if the cursor is over a field in the dbox then
                        if the fld_active bit is set for that field then
                                if the cursor was not previously in this field then
                                        execute the fld_into ER
                                        execute the fld_help ER

                                execute the fld_over ER

                if the cursor is not over a field in the dbox then
                        execute the dlg_over ER
```

fld_value

fld_value is not truly an ER, but its syntax is similar. This statement is used to initialize field data during the process of ICL compilation (the PD build dialog command). The following are examples:

An integer field is initialized to the value 123.

```
fld 100 type 2
fld_value 123 end
```

An integer toggle field is initialized to have five possible values, 100, 125, 150, 175, or 200, and the initial value is the third element 150.

```
fld 100 type 12
fld_value 3 5
 100 125 150 175 200
end
```

An integer/string toggle field is initialized to have two possible values, 1 or 0, and the initial value is the second element 0. Each integer value is equated to a corresponding character string (1 = "Arrows In", 0 = "Arrows Out") which is displayed in the field.

```
fld 318 type 22
fld_value 2 2
 1           0
 "Arrows In" "Arrows Out"
end
```

An integer/icon toggle field is initialized to have three possible values, 22, 23, or 0, and the initial value is the third element 0. Each integer value is equated to a corresponding icon (22 = 306, 23 = 307, 0 = 338) which is displayed in the field.

```
fld 102 type 32
fld_value 3 3
  22  23   0
 306 307 338
end
```

Functions, Procedures & Keywords

A function is an intrinsic ICL feature that returns an integer, real, or string value. Only constants (no expressions) may be used as function arguments. A procedure is an intrinsic ICL feature that performs some service. Both constants and expressions may be used as procedure arguments. An expression is a function or several functions and constants combined by mathematical operations or string concatenation.

Some functions and procedures may be called from UPL. The syntax is the same in UPL and as it is in ICL. The symbol "*****UPL*****" will appear at the end of the title line for each function or procedure that has a UPL equivalent.

Example:
d_box_add(dlg) *****UPL*****

Special Abbreviations

In order to minimize redundant documentation some special abbreviations for argument names are defined here. These abbreviations may be the same as key words used in other places in this document and only have the meaning defined here when they are used as function and procedure arguments. Argument names that begin with an upper case letter must be constants, whereas argument names that begin with a lower case letter may be expressions. If the first letter is an "i" (upper or lower case) the implied type is integer. Similarly, "r" implies real and "s" implies string.

dlg
Integer dbox index number. Minus one (-1) may be used to indicate the "current" dbox number (last dbox index defined).

fld
Integer field index number. Minus one (-1) may be used to indicate the "current" field number (last field index defined). Zero (0) may be used to indicate that the function or procedure is relevant to all fields in the current dbox.

IVAR
This variable is used to work around the intrinsic function limitation that no expressions are allowed as input. The value of IVAR is the i_var function's variable index in which a variable value is passed to the function.

Functions

active_str(dlg, fld)
Returns a string that is the active bit table for the given dbox (fld = 0) or field.

biti(I1, I2)
Returns the bit status of the I2-th bit relative to the system variable IBUF(I1). Zero (0) = not set, one (1) = set.

biti_var
Rarely used function to be documented when needed.

char(I1)
Returns a string which is the ASCII equivalent of I1.

date
Returns the date as a string in the format mm/dd/yy.

dirchr
Returns a string which is the file name path separator character ('\' for DOS and '/' for UNIX)

dlg_active(DLG, I2)
Returns the bit status of the I2-th bit in the dbox attribute bit table.

dlg_curclr(DLG)
Returns the cursor color number for dbox DLG. (only valid for cursor styles 1 to 14, i.e. non-icon type cursors)

dlg_cursty(DLG)
Returns the cursor style number for dbox DLG.

dlg_dparent(DLG)
Returns the parent dbox number (i.e. the number of the dbox that activated dbox DLG). If the value returned is less than one, the dbox was not activated by another dbox but was activated as follows:

-2 from Modifier Processor (MP)
-4 from text being automatically routed from an alpha window to a dbox field
-5 from initial list of dialog boxes put up when the menu was restored
-6 from the dlg_start_up ER
-7 from the dlg_finish ER
-8 from internally in PD
-9 form the \D standard menu command
-10 from a UPL program
-101 to -3000 from menu registration event points in PD

Functions continued

dlg_fparent(DLG)
Returns the parent field number (i.e. the number of the field that activated dbox DLG). If the value returned is less than one, the dbox was not activated by a field.

dlg_lx(DLG)
Returns the pixel coordinate of the left edge of the dbox.

dlg_ly(DLG)
Returns the pixel coordinate of the bottom edge of the dbox.

dlg_offx(DLG)
Returns the x pixel coordinate of the offset of the dbox.

dlg_offy(DLG)
Return the y pixel coordinate of the offset of the dbox.

dlg_ux(DLG)
Returns the pixel coordinate of the right edge of the dbox.

dlg_uy(DLG)
Returns the pixel coordinate of the top edge of the dbox.

d_box_level(DLG) ***UPL***
Returns:
0 if the dbox is not on the screen.
1 if the dbox is on the screen, but at least partially covered by another dbox.
2 if the dbox is on the screen, not physically covered by another dbox, but not on the top of the internal dbox stack.
3 if the dbox is on the screen, not physically covered by another dbox, and is on the top of the internal dbox stack.

existf(SFILE_NAME)
Returns a one (1) if the file SFILE_NAME exists, otherwise returns a zero (0).

feivar
Rarely used function to be documented when needed.

fldiv(DLG, FLD) ***UPL***
Returns the integer value of a field.

fldrv(DLG, FLD) ***UPL***
Returns the real number value of a field.

fldsv(DLG, FLD) ***UPL***
Returns the string value of a field.

fld_active(DLG, FLD, I2)
Returns the bit status of the I2-th bit in the field attribute bit table.

fld_bkg(DLG, FLD)
Returns the background color assigned to the field.

fld_col(DLG, FLD)
Returns the column the text cursor is currently in. This is the location where text will be displayed when printed to the field. This is not the position of the cursor which is controlled by the mouse.

fld_curclr(DLG, FLD)
Returns the color of the cursor assigned to the field.

fld_cursty(DLG, FLD)
Returns the cursor style assigned to the field.

fld_dlgno(DLG, FLD)
Not implemented.

fld_fgr(DLG, FLD)
Returns the foreground color assigned to the field.

fld_hlt(DLG, FLD)
Returns the field highlight color assigned to the field.

fld_lx(DLG, FLD)
Returns the pixel coordinate of the left edge of the field.

fld_ly(DLG, FLD)
Returns the pixel coordinate of the bottom edge of the field.

fld_ncol(DLG, FLD)
Returns the number of character columns in the field.

fld_no(DLG, FLD)
Returns field ID number. This is only worthwhile if FLD = -1.

fld_nrow(DLG, FLD)
Returns the number of character rows in the field.

fld_row(DLG, FLD)
Returns the row the text cursor is currently in. This is the location where text will be displayed when printed to the field. This is not the position of the cursor which is controlled by the mouse.

fld_typ(DLG, FLD)
Returns the field type.

fld_ux(DLG, FLD)
Returns the pixel coordinate of the right edge of the field.

fld_uy(DLG, FLD)
Returns the pixel coordinate of the top edge of the field.

get_assist_str(I1, I2, I3)
Returns the assist message for the indexes I1, I2 and I3.

get_cdi
Returns number indicating the current default drive (DOS only). 0 = A:, 1 = B:, 2 = C:, etc.

get_cds
Returns a 2-character string which is the current default drive (DOS only). "A:", "B:", "C:", etc.

get_char(I1)
Wait until a character with the ASCII value I1 is entered. If I1 is zero (0), wait until any character is entered.

get_cwd
Returns current default directory for the indicated drive. 0 = default drive, 1 = A:, 2 = B:, 3 = C:, etc.

get_ima_list_str
Returns a list as a text string of images in the PD part.

get_kbd_status(IBIT)
Returns a 1 (key down) or 0 (key up) for the IBIT-th element of the status bit table for the keyboard. Valid values of IBIT:

1 insert toggle on
2 caps lock toggle
3 num lock toggle
4 scrl lock toggle
5 alt
6 ctrl
7 lshift
8 rshift
9 sysreg key
10 caps lock key
11 num lock key
12 scrl key
13 ralt
14 rctrl
15 lalt
16 lctrl

get_last_assist(IDX_PART)
Returns IDX_PART part of the last assist text index. Each assist text index consists of 3 index numbers. Valid values of IDX_PART are:

1	Returns first assist index that was saved with the save_last_assist procedure.
2	Returns second assist index that was saved with the save_last_assist procedure.
3	Returns third assist index that was saved with the save_last_assist procedure.
-1	Returns first assist index that is for the assist text currently being displayed.
-2	Returns second assist index that is for the assist text currently being displayed.
-3	Returns second assist index that is for the assist text currently being displayed.
other values	Returns 0.

get_lay_echo_str
Returns a list as a text string of the visible layers in the PD part.

get_lay_used_str
Returns a list as a text string of the occupied layers in the PD part.

get_lmask_str
Returns a list as a text string of the layers that are currently masked on for entity selection.

get_mouse_status
Returns the status of the buttons on the mouse. Each button on the mouse is represented by a bit in the integer returned, bit 0 is button 1, bit 1 is button 2, etc. A bit value of 1 indicates the button is currently down and 0 the button is currently up. For example on a 2 button mouse:

0	all buttons up
1	left button down, right button up
2	right button down, left button up
3	both buttons down

get_mouse_str(BUTTON_NO, NCLICK, KEY1, KEY2, KEY3)

Returns the string defined by the arguments from the mouse macro definition file (pd.mse).

BUTTON_NO mouse button number, usually 1, 2 or 3

NCLICK for single (1) or double click (2)

KEY1..3 which keyboard keys must be held down when the mouse click is made. Valid values are:

0 doesn't matter which keyboard keys are pressed
1 insert toggle on
2 caps lock toggle
3 num lock toggle
4 scrl lock toggle
5 alt
6 ctrl
7 left shift
8 right shift
9 sysreg key
10 caps lock key
11 num lock key
12 scrl key
13 right alt
14 right ctrl
15 left alt
16 left ctrl

get_view_list_str

Returns a list as a text string of the defined views and CPL's in the PD part.

info_arch

Returns the type of CPU architecture PD is currently running on.

1 SPARC
2 Intel 80x86
3 Motorola 68xxx
4 DEC Ultrix

info_db_ver

Returns database version number. 500 = version 5.00

info_mach

Returns 1 if Personal Machinist is installed, otherwise 0.

info_os

Returns the OS that PD is currently running under.

2 DOS Extended (PharLap)
3 Unix

Functions continued

info_pd_id
Returns 2 if running MicroDraft (2D) and 3 if running PD (3D).

info_pd_ver
Returns version number of PD. 600 = 6.00

info_surf
Returns 1 if surfacing option is installed, otherwise 0.

info_usgage
Returns the mode PD is being run in.
1 PD demonstration mode
2 PD production mode
3 PM demonstration mode
4 PM production mode

in_input
Returns 1 if get_fld_input is currently being executed by another ER, otherwise 0 is returned.

i_data(SCOMMON, OFFSET)
Returns an integer value from the common block SCOMMON at offset OFFSET.

i_var(I1)
This function returns various integer PD system parameters, dbox menu parameters, and general purpose dbox variables. The valid values are:

1 currently selected color
2 currently selected font
3 currently selected layer
4 current view number
5 number of entities in database (if greater than 32,767 then -1 is returned)
6 menu flag, 0 = menus off, 1 = menus on
7 echo command flag, 0 = echo PD commands, 1 = don't echo PD commands to prompt window
8 journal file flag, 0 = off, 1 = commands only, 2 = all output
9 current old style menu pick number
10 total number of old style menu pick boxes up
11 current x pixel location of mouse cursor (see 15 for y location)
12 current CPL number selected
13 network in use flag
14 current drawing in read only mode (1 = read only, 0 = read/write)
15 current y pixel location of mouse cursor (see 11 for x location)
16 current VNP help index (used for assist text)
17 current MP help index (used for assist text)
18 current Get Data help index (used for assist text)
67 error flag from last use of load_plot_device procedure

68 2D/3D mode flag, 2 = 2D (MicroDraft), 3 = 3D (PD)
69 file number which is at the top of the list in the current file list dbox
70 number of files in the file list in the current file list dbox
71 lower x pixel value of the current graphics window
72 lower y pixel value of the current graphics window
73 upper x pixel value of the current graphics window
74 upper y pixel value of the current graphics window
75 lower x pixel value of the screen
76 lower y pixel value of the screen
77 upper x pixel value of the screen
78 upper y pixel value of the screen
79 dbox ID number which mouse cursor is currently over (0 for none)
80 field ID number which mouse cursor is currently over (0 for none)
81 character entered by user when the last get_pick procedure was used
82 x pixel location of mouse cursor when the last get_pick procedure was used
83 y pixel location of mouse cursor when the last get_pick procedure was used
84 ID number of dbox mouse was over when the last get_pick procedure was used (0 if not over a dbox)
85 ID number of field mouse was over when the last get_pick procedure was used (0 if not over a field)
91 integer value of user input from the last call to the get_fld_input procedure
92 ASCII value of last character entered in the get_fld_input procedure
93 Type of file name entered in the get_fld_input procedure with options set to 6. Negative values indicate error in file name.
- -1 file name too long
- -2 file name has no characters
- -3 file name has invalid characters in it
- -4 file name has invalid drive character
- -5 too many '.'
- -6 too many '\'
- -7 invalid path
- -8 * or ? found in path
- -9 drive letter is not valid
- 0 normal file name without path or drive
- 1 normal file name with path or drive
- 2 file name with wild characters, without path or drive
- 3 file name with wild characters, with path or drive
- 4 directory name without drive
- 5 directory name with drive
- 6 drive letter only
- 11 normal file name with path or drive from root
- 13 file name with wild characters, with path or drive from root
- 14 directory name from root without drive
- 15 directory name from root with drive

101 to 2038 same as UPL sysvari intrinsic procedure
3001 to 3300 ICL programmer defined integer variables
4001 to 4010 integer data from last registration event (see Appendix A for details)
4901 current number of registered PD events

4902	maximum possible number of registered events (currently is 400)
4903	internal handle number of currently executing registered event (0 if none)
4904	ID number of currently executing registered event (0 if none)
4905	registration number of currently executing registered event (0 if none)
4906	type of action taken for currently executing registered event (0 if none) 1 = reg_d_box_add, 2 = reg_d_box_del, 3 = reg_exec_dlg, 4 = reg_exec_fld.
5001 to 5600	Data returned from last device procedure call

kbd(I1)
Returns the ASCII value of a keyboard entry. I1 is the wait time.
For the following values of I1:
= -1 Don't wait, just see if a character was entered.
= 0 Wait indefinitely.
> 0 Wait I1 seconds.

layer_color(IVAR)
The value returned depends on the "color by layer" status. If the PD mode is color by layer, the value returned is the color associated with the input layer value. If the PD mode is NOT color by layer, the value returned is the input value.

mps(IMP_WORD)
Returns the select status of an MP word. Zero (0) = not selected, one (1) = selected.

mpsv(IMP_WORD)
Returns the string associated with the value of an MP word for string value or file name type modifier words.

mpsxt(IMP_WORD)
Returns the word number of the selected MP word in a group of mutually exclusive MP words. Word_number is the number of any one of the mutually exclusive words.

mpv(IMP_WORD)
Returns the floating point value associated with an MP word.

mv_cpl(IVAR)
IVAR is the MV number. The integer value returned is the MV CPL number.

mv_name(IVAR)
IVAR is the MV number. The string value returned is the MV name (if any).

mv_scl(IVAR)
IVAR is the MV number. The real number value returned is the MV scale.

read_msg(IMSG)
Returns the string associated with message number IMSG from the pd.msg file.

r_data(SCOMMON, IOFFSET)
Returns a real number value from the common block SCOMMON at offset IOFFSET.

r_var(I1)
This function returns various real (floating point) PD system parameters, dbox menu parameters, and general purpose dbox variables. The valid values are:

1	current display scale
3	current default Z depth
91	real value of user input from the last call to the get_fld_input procedure
1001 to 2011	same as UPL sysvarr intrinsic procedure
3001 to 3100	ICL programmer defined real variables
4001 to 4006	real data from last registration event (see Appendix A for details).
5001 to 5600	Data returned from last device procedure call. Note: addressed on word boundaries.

s_data(SCOMMON, IOFFSET)
Returns a string value from the common block SCOMMON at offset IOFFSET.

s_var(I)
This function is similar to the i_var function except that it is for string values.

1 to 25	File names from PD config file lines 1 to 25
50	current drawing (part) file name
51	user name
52	network name
90	path of menu file name
91	input string from last get_fld_input
92	terminating character from last get_fld_input
3001 to 3020	ICL programmer defined string variables (128 characters maximum each)
4001 to 4020	string data from last registration event (see Appendix A for details)
5001 to 5600	Data returned from last device procedure call. Note: gets 2 characters starting at given word location.
6001 to 6600	Data returned from last device procedure call. Note: gets number of characters specified from last set_i_var(5000) starting at given word location.

time
Returns the time as a string in the format hh:mm.

xh_in_mv
Returns the MV number the mouse cursor is currently in, 0 if it is not in any MV.

Functions continued

xh_x(DLG, FLD, P-POS)
Returns the X pixel position of the mouse cursor. If DLG is 0, then the position value returned is relative to the lower left corner of the screen, otherwise it is relative to the given dbox and field number and P-POS. If FLD is 0, then position is relative to the DLG dbox. If P-POS is 0, then position is relative to the lower left corner of the field of dbox.

xh_xc(DLG, FLD, P-POS)
Returns the X character cell position times 10 of the mouse cursor. The value would need to be divided by 10 to get the actual character cell position. If DLG is 0, then the position value returned is relative to the lower left corner of the screen, otherwise it is relative to the given dbox and field number and P-POS. If FLD is 0, then position is relative to the DLG dbox. If P-POS is 0, then position is relative to the lower left corner of the field of dbox.

xh_y(DLG, FLD, P-POS)
Returns the Y pixel position of the mouse cursor. If DLG is 0, then the position value returned is relative to the lower left corner of the screen, otherwise it is relative to the given dbox and field number and P-POS. If FLD is 0, then position is relative to the DLG dbox. If P-POS is 0, then position is relative to the lower left corner of the field of dbox.

xh_yc(DLG, FLD, P-POS)
Returns the Y character cell position times 20 of the mouse cursor. The value would need to be divided by 10 to get the actual character cell position. If DLG is 0, then the position value returned is relative to the lower left corner of the screen, otherwise it is relative to the given dbox and field number and P-POS. If FLD, is 0 then position is relative to the DLG dbox. If P-POS is 0, then position is relative to the lower left corner of the field of dbox.

Procedures

action(icode)
Sets the terminating action code. (see define.cmd)

add_cr(dlg, fld, width, wrap)
Adds <CR><LF> sequences to the string data in the field "dlg, fld". If width is less than or equal to 0 then the number of columns in the specified field is used for the width value. If wrap is 1 then character wrap is done. If wrap is 2 then word wrap is done.

add_fldiv(dlg, fld, i1)
Add the value i1 to the value stored in the field.

append_dlg_cmd(dlg, cmd_type, cmd_str)
Compiles (build dialog cmd) cmd_str and appends it to the commands in the specified dbox ER. Valid values of cmd_type are:

1 = dlg_in_to	5 = dlg_out_of
2 = dlg_dig	6 = dlg_over
3 = dlg_begin	7 = dlg_hlp
4 = dlg_end	8 = dlg_dsp

append_fld_cmd(dlg, fld, cmd_type, cmd_str)
Compiles (build dialog cmd) cmd_str and appends it to the commands in the specified field ER. Valid values cmd_type are:

1 = fld_in_to	5 = fld_out_of
2 = fld_dig	6 = fld_over
3 = fld_begin	7 = fld_hlp
4 = fld_end	8 = fld_dsp

clear_input
Clears any input buffered by the input procedure.

clr_aw(iwin)
Clears the specified alpha window to its background color.

clr_fld(dlg, fld)
Clear (or blank) the "text" field.

copy_fldiv(from_dlg, from_fld, to_dlg, to_fld)
Copies the integer field value from one field to another.

copy_fldrv(from_dlg, from_fld, to_dlg, to_fld)
Copies the real (floating point) field value from one field to another.

copy_fldsv(from_dlg, from_fld, to_dlg, to_fld)
Copies the string field value from one field to another.

def_aw(i1, i2, i3, i4, i5, i6, i7, i8, i9, i10)
Define/modify an alpha window.
- i1 Window number.
- i2 Left edge in character cell coordinates.
- i3 Bottom edge in character cell coordinates.
- i4 Right edge in character cell coordinates.
- i5 Top edge in character cell coordinates.
- i6 Background color.
- i7 Page.
- i8 Scroll flag. Set to 1.
- i9 Priority.
- i10 1 = Save to menu, 0 = do not save

delay(i1)
Pause for i1 seconds.

device(driver_type, function_no)
Calls the specified device driver function. driver_type = 1 graphics driver, 2 input driver and 3 plotter driver. Data can be passed to the driver function with set_i/r/s_var(idx,data). idx is from 5001 to 5600 which gives the word position in the data passed to the driver. i/r/s_var(idx) can be used to get data returned from the driver. s_var will only get 2 character strings.

done_chr(ichar)
ASCII value of character to return when finished with the ER. Use 0 to return no character. (see define.cmd)

do_lay_echo(rpnt_flag)
Used to display the layer changes made with set_lay_echo procedure. If rpnt_flag is 0, then do not repaint entities on changed layers, otherwise repaint them. The procedure reset_lay_echo is automatically called after do_lay_echo is complete.

dsp_fld(dlg, fld)
Execute the field display ER.

d_box_add(dlg) ***UPL***
Activate or put the dbox on the screen. If the dbox is already on the screen it is moved to the foreground. The position of the dbox when it is put on the screen can be effected by the set_position command.

d_box_del(dlg) ***UPL***
Remove a dbox from the screen.

echo_cpl(flag)
If flag = 1, display CPL axes, otherwise remove any CPL axes that are displayed.

exec_dlg_cmd(dlg, iER) ***UPL***

Execute the dbox ER of type iER. iER has the following meanings:

1 = dlg_in_to	5 = dlg_out_of
2 = dlg_dig	6 = dlg_over
3 = dlg_begin	7 = dlg_hlp
4 = dlg_end	8 = dlg_dsp

exec_fld_cmd(dlg, fld, iER) ***UPL***

Execute the field ER of type iER. iER has the following meanings:

1 = fld_in_to	5 = fld_out_of
2 = fld_dig	6 = fld_over
3 = fld_begin *	7 = fld_hlp
4 = fld_end *	8 = fld_dsp *

* If iER is negative for these values, then the default event action is done instead of the defined ER. The default event action for fld_dsp can be accomplished by using the dsp_fld procedure.

file_list_display(sort_type, dsp_size, dsp_date, dsp_time)

Controls display of the file list field (field type 103). The file size, date and time can optionally be displayed and are always displayed in that order, but any combination of the three may be displayed.

sort_type = 0 display files in order found, = 1 display in alphabetical order

dsp_size = 0 don't display file size, > 0 display in specified column, < 0 display - dsp_size columns after the longest file name in the list.

dsp_date = 0 don't display file date, > 0 display in specified column, < 0 display - dsp_date columns after the longest file name in the list.

dsp_time = 0 don't display file time, > 0 display in specified column, < 0 display - dsp_time columns after the longest file name in the list.

flush_input

Any pending input from menu commands or execute files is flushed (removed) from the user input buffer.

format_r(rval, sformat)

Returns a string which is accessed via s_var(3001) given the floating point number rval and the C language style format sformat.

get_fld_input(dlg, fld, options, ins_cursor, ovs_cursor)
This procedure gets user input and stores the input into the given field's value.
options determines what type of input to get. If options is negative then the current value in the field is not cleared before getting input, otherwise it is.
= 2 - get numerical expression
= 6 - get file name
= 99 - get a text string
If 100 is added to the option value then input is restricted from the keyboard only.
ins_cursor id number of icon to use for the text cursor when in insert mode.
ovs_cursor id number of icon to use for the text cursor when in overstrike mode.

get_path(file_name, dlg, fld)
Stores the path portion of file_name into the field's value specified by dlg and fld.

get_pick(cur_style, cur_color)
Gets a pick from the user that is not interpreted by the menu processor to execute any dig ER's or to change the mouse cursor when moving into the various areas on the screen. cur_style and cur_color set the cursor style to use while get_pick is waiting for a pick. This returns the ASCII value of the character the user (1 for mouse digitize) in i_var(81), the x and y mouse pixel location in i_var(82) and i_var(83), the dbox id number in i_var(84) (0 if not in any dbox) and the field id number in i_var(85) (0 if not in any field).

get_root_fn(file_name, dlg, fld)
Stores the base portion (i.e. part of file name without path) of file_name into the field's value specified by dlg and fld.

grey_fld(dlg, fld, flag)
Greys (flag = 1) or ungrays (flag = 0) out the field specified by dlg and fld. If the field is covered or partially covered, it is grayed/ungrayed only after it is uncovered.

icon(i1)
Display the icon i1. The set_position command will determine the location that the icon appears. If set_positon is not used the lower left corner of the icon will justify on the lower left corner of the field (or dbox) from which the icon command is issued.

input(string)
Issue the text string into the PD command stream. Several input commands can be issued in one ER in order to concatenate several strings. It is important to note that this command does not take effect until after the ER is completed, no matter where in the ER it appears.

load_plot_device(file_name, err_flag, msg_flag)
Loads the plotter device driver specified by file_name and reads the associated pen file if necessary. If err_flag is 0 then it will be considered an error if the plotter is not ready (not on-line or not connected), otherwise it will not be considered an error (i.e. if you plan on plotting to a file instead of to the plotter device). If msg_flag is 0, then all messages are displayed ('loading driver..', and error messages). If msg_flag is 1, then the 'loading driver' message is suppressed. If msg_flag is 2 then all messages are suppressed, in which case the ICL code is expected to handle error reporting. The error flag returned from load_plot_device is returned in i_var(67).

make_file_list(file_name_seed)
Sets the file name seed from which a new file name list is made.

mnu_cmd(mnu_cmd)
Executes the old style '\' menu command string given by mnu_cmd. Do not include the '\' in mnu_cmd. This is the same as the PD 'MENu COMMand'.

mnu_off
Disables recognition of picks in dialog boxes.

mnu_on
Enables recognition of picks in dialog boxes.

move(dlg, icolor, flag)
Move dboxs on the screen. If the new location is determined from the user an outline of the dbox is used for the mouse cursor and is drawn in color given by icolor. The way the dbox is moved is determined by the value of flag as follows:

-2 move dbox to front
-1 move to new location specified by a mouse digitize.
0 move to new location specified by a mouse digitize and move to front.
1 move to new location specified by a mouse digitize, move to front and save as new default location of the dbox.
2 move to new location specified by the last set_position procedure executed.
3 move to new location specified by the last set_position procedure executed and move to front.
4 move to new location specified by the last set_position procedure executed, move to front and save as new default location of the dbox.

only_pick(dlg, fld)
Limits picks to a specific dbox and field. If fld is 0 then any field in the dbox may be picked, if dlg is 0 then anything is pickable.

open_assist_text(file_name, mem_flag)
Closes current assist text file and opens new assist text file. mem_flag controls how much of the file is in memory and how much is accessed from the disk file.

-2 keep index portion of file in memory, access the rest from disk
-1 keep all in memory (recommended value)
0 access all of file from disk

> 0 keep given number of bytes in memory

open_mouse_mac(file_name, mem_flag)
Closes current mouse macro file and opens new mouse macro file. mem_flag controls how much of the file is in memory and how much is accessed from the disk file.
-2 keep index portion of file in memory, access the rest from disk
-1 keep all in memory (recommended value)
0 access all of file from disk
> 0 keep given number of bytes in memory

play_icon(delay_time, start_icn, end_icn, dlg, fld, loc1, offx, offy, loc2)
This procedure is used for animation to play (display) a sequence of icons. The sequence of icons is all played in the same location. The location is relative to a dbox or field.
delay_time time between displaying icons from one to the next in 100th's of seconds
start_icn first icon number in sequence
end_icn last icon number in sequence
dlg dbox to display icons in
fld field to display icons in (0 to display in dbox)
loc1 p-location in dbox or field to display icon in
offx x offset in units of 10ths of character cells to display icons
offy y offset in units of 20ths of character cells to display icons
loc2 p-location in icon to justify icon location

prt(text)
Print string text in field associated with current ER, or in the command window if no field is associated with current ER.

prt_aw(iwin, icolor, string)
Print the string in the alpha window "iwin" and the color "icolor". The argument string may be an integer or real expression as well.

prt_fld(dlg, fld, icolor, text) ***UPL***
Print "text" in the designated field and the color "icolor". The argument string may be an integer or real expression as well. (see also clr_fld)

prt_gw(icolor, x_pix, y_pix, text)
Print "text" in color "icolor" at pixel location x_pix, y_pix on the screen.

push_button(dlg, fld, inout_flag, color1, color2, color3)
This procedure makes the field specified with dlg and fld appear to be pushed in (inout_flag = 1) or pushed out (inout_flag = 0). Color1/2/3 control the colors used around the field for the shading effect. This procedure is used in fld_dig ER's.

put_xh(dlg, fld, ipos)
Move the cursor to the p-positon in the field or dbox.

redo
Execute the PD redo function. This command will not effect the PD command stream.

reg_d_box_add(reg_id, reg_no, dlg)
This procedure registers dlg to be added when registration event reg_no occurs in PD. See Appendix A for a complete list and description of PD dialog box registration events. reg_id is a number given by the ICL programmer that can later be used in the unreg procedure to remove registered events.

reg_d_box_del(reg_id, reg_no, dlg)
This procedure registers dlg to be deleted when registration event reg_no occurs in PD. See Appendix A for a complete list and description of PD dialog box registration events. reg_id is a number given by the ICL programmer that can later be used in the unreg procedure to remove registered events.

reg_exec_dlg_cmd(reg_id, reg_no, dlg, er_type)
This procedure registers ER type er_type in dbox dlg to be executed when registration event reg_no occurs in PD. See Appendix A for a complete list and description of PD dialog box registration events. reg_id is a number given by the ICL programmer that can later be used in the unreg procedure to remove registered events. Valid values of er_type are:

1 = dlg_in_to	5 = dlg_out_of
2 = dlg_dig	6 = dlg_over
3 = dlg_begin	7 = dlg_hlp
4 = dlg_end	8 = dlg_dsp

reg_exec_fld_cmd(reg_id, reg_no, dlg, fld, er_type)
This procedure registers ER type er_type in field given by dlg and fld to be executed when registration event reg_no occurs in PD. See Appendix A for a complete list and description of PD dialog box registration events. reg_id is a number given by the ICL programmer that can later be used in the unreg procedure to remove registered events. Valid values of er_type are:

1 = fld_in_to	5 = fld_out_of
2 = fld_dig	6 = fld_over
3 = fld_begin	7 = fld_hlp
4 = fld_end	8 = fld_dsp

remove_cr(dlg, fld)
Removes <CR><LF> (ASCII 13 ASCII 10) and replaces them with a single space (ASCII 32). It also converts '$' to <CR><LF> sequence. This is normally used in conjunction with the add_cr procedure to remove/add <CR>'s before a string field is edited.

repaint(r1)
This command will execute PD repaint and zoom functions without any effect on the PD command stream. For values of r1 the following PD functions are executed:

r1 < 0.0	Zoom All
r1 = 0.0	Repaint
0.0 < r1 < 1.0	Zoom Down (screen scale is multiplied by r1)
r1 = 1.0	No effect
r1 > 1.0	Zoom Up (screen scale is multiplied by r1)

reset_lay_echo
Resets the layers changed by the set_lay_echo procedure to no layers changed. (see set_lay_echo and do_lay_echo)

restore_image
Invokes the restore image command. The user will be prompted to enter the image number unless an ER has been registered to registration point 1041 (see Appendix A).

restore_pen
Invokes the restore pen command. The user will be prompted to enter the pen file name unless an ER has been registered to registration point 1012 (see Appendix A).

rgrid
Repaints the grid. Use set_i_var(1511,<1 or 0>) to turn the grid on or off.
The following example will toggle and repaint the grid:

```
set_i_var(1511, 1 - i_var(1511))
rgrid
```

run_upl(sfile_name)
Run a UPL program.

save_image
Invokes the save image command. The user will be prompted to enter the image number unless an ER has been registered to registration point 1040 (see Appendix A).

save_initial_dlg(0)
Saves the set of dboxs currently on the screen as the set of dboxs to be restored when the menu is initially brought up. Must have the one argument as 0 (zero).

save_last_assist
Saves the current assist indices. These can later be used with the get_assist_str and get_last_assist procedures.

save_position(dlg)
Saves the current position of the dbox dlg as the default position when the dbox is brought up.

scrl_fld(dlg, fld, nlines)
Vertically scrolls the specified field by nlines. nlines can be positive or negative to control direction of scroll.

sel_coord_dsp(on_off, units, view, grid, prec, xyz, space)
Controls the mouse cursor coordinate display.

send_input
Causes the characters in the current input buffer (set up by the input procedure) to be immediately issued into the command stream before executing the remaining portion of the ER.
Warning! Use of this function is risky, only use when absolutely necessary and thoroughly test.

set_active_str(dlg, fld, to_dlg, to_fld)
Reads the active bit table from the specified dbox or field and converts it to a user readable string format and stores it in the to_dlg, to_fld field.

set_assist_fld(dlg, fld)
Sets the field which the assist text will be displayed in.

set_assist_str(index1, index2, index3, text)
Writes string text into the current assist text file for the given index values.

set_aw_to_fld(awin, dlg, fld, cursor_style)
Redirects output that would normally be displayed in the alpha window "awin" to be displayed in the field specified by dlg and fld. cursor_style is the icon cursor number to use for the text cursor.

set_bit_i(i1, i2, i3)
Sets bit i2 in the PD system variable IBUF(i1) to the value i3.

set_button_push(flag, style, delay, size, color_dir)
Sets the button push parameters.

flag	0 - turn button push feature off, 1 - turn button push feature on
style	valid values are 1 or 2
delay	delay time between the pushed in and out appearance of the button in 100th's of a second
size	how far to move for the button push for style 1 or size of border for style 2
color_dir	for style = 1 this is the p-position direction the button is to move. Five (5) is not a valid value. for style = 2 a border is drawn around the button "size" pixels wide in color "color_dir".

set_cmask(icolor)
Sets the entity color selection mask for getdata. Valid values for icolor:

< 0	allow all entities to be picked except those with specified color
= 0	allow all entities to be picked
> 0	only allow entities with specified color to be picked

set_control_color(icolor)
Sets the display color for characters with an ASCII value less than 32.

set_cur_text(cursor_style, loc, off_x, off_y, txt_color, txt_str)
Sets up text to be displayed dynamically with the mouse cursor as it is moved on the screen. When the currently displayed cursor is equal to cursor_style, then the text given in txt_str will be displayed in color txt_color at the p-position loc relative to the current mouse cursor position, and offset by off_x, off_y pixels. To turn off cursor text display off, call set_cur_text with cursor_style = 0.

set_cwd(path)
Specifies a new current working (default) directory path. On DOS systems a drive letter may also be included which will set a new default drive.

set_dbl_click(time, size)
Sets the mouse double click parameters. If the second click does not occur within "time" (expressed in 100th's of a second), or if the mouse is moved more than "size" mouse units, then it will be considered 2 single clicks.

set_dlg_active(dlg, i2, i3)
Set bit i2 in the dbox attribute bit table to i3.

set_dlg_cmd(dlg, cmd_typ, cmd_string) ***UPL***
Compiles (build dialog cmd) the ICL source code in cmd_str and replaces the ER specified by dlg and cmd_typ with the compiled code. Valid values of cmd_type are:

1 = dlg_in_to	5 = dlg_out_of
2 = dlg_dig	6 = dlg_over
3 = dlg_begin	7 = dlg_hlp
4 = dlg_end	8 = dlg_dsp

set_dlg_curclr(dlg, ival)
Sets the dlg dbox's cursor color to ival.

set_dlg_cursty(dlg, ival)
Sets the dlg dbox's cursor style to ival.

set_dlg_dparent(dlg, ival)
Sets the dlg dbox's parent dbox to ival.

set_dlg_fparent(dlg, ival)
Sets the dlg dbox's parent field to ival.

set_dlg_offx(dlg, ival)
Sets the dlg dbox's default x position (pixel location) to ival.

set_dlg_offy(dlg, ival)
Sets the dlg dbox's default y position (pixel location) to ival.

set_dmask(entity_type)
Sets entity type selection mask for getdata.
< 0 allow all entities to be picked except those of the specified type
= 0 allow all entities to be picked
> 0 only allow entities of specified type only to be picked

set_emask(entity_type)
Sets the valid entity types for getdata ident. This is initially called with entity_type = 0 to make all entity types selectable. Additional calls to set_emask will limit which entity types are selectable. The second call will limit the selectable entities to type "entity_type" (see the UPL manual or the Database spec for entity type numbers). Additional calls will add to the list of selectable entity types.

set_feivar(i,i)
Rarely used specialized procedure to be documented when needed.

set_fldiv(dlg, fld, i) *****UPL*****
Set the value of a field to integer value i. Data type conversion is done if necessary.

set_fldrv(dlg, fld, r) *****UPL*****
Set the value of a field to real value r. Data type conversion is done if necessary.

set_fldsv(dlg, fld, s) *****UPL*****
Set the value of a field to the string s. Data type conversion is done if necessary.

set_fld_active(dlg, fld, i3, i4)
Sets bit i3 in the field attribute bit table to i4.

set_fld_bkg(dlg, fld, i)
Sets the background color of the specified field to i.

set_fld_cmd(dlg, fld, cmd_typ, cmd_str) ***UPL***
Compiles (build dialog cmd) the ICL source code in cmd_str and replaces the field ER specified by dlg, fld and cmd_typ with the compiled code. Valid values of cmd_type are:

1 = fld_in_to	5 = fld_out_of
2 = fld_dig	6 = fld_over
3 = fld_begin	7 = fld_hlp
4 = fld_end	8 = fld_dsp

set_fld_col(dlg, fld, i)
Sets the text cursor column position of the specified field to i.

set_fld_curclr(dlg, fld, i)
Sets the field's cursor color for the specified field to i.

set_fld_cursty(dlg, fld, i)
Sets the field's cursor style to i.

set_fld_fgr(dlg, fld, i)
Sets the foreground color of the specified field to i.

set_fld_hlt(dlg, fld, i)
Sets the highlight color of the specified field to i. Usually 0 for no highlighting and 15 to highlight field when cursor is over the field.

set_fld_row(dlg, fld, i)
Sets the field's text cursor row to i.

set_fmask(font)
Sets entity line font selection mask for which line fonts may be idented in getdata. (see also set_xfmask)

< = 0	allow all fonts to be picked
> 0	allow only the specified font number to be idented

set_i_data(scommon, ioffset, i3)
Puts the integer value i3 in the common block scommon at offset ioffset.

set_i_var(i1, i2)
Sets the i_var(i1) integer variable to the value i2. The valid values of i1 are:

1	sets currently selected color
2	sets currently selected font
3	sets currently selected layer
4	sets current view number
7	sets echo command flag, 0 = echo PD commands, 1 = don't echo PD commands to prompt window

12	sets current CPL number selected
16	sets current VNP help index (used for assist text)
17	sets current MP help index (used for assist text)
18	sets current Get Data help index (used for assist text)
69	sets file number which is at the top of the list in the current file list dbox
101 to 2038	same as UPL sysvari intrinsic procedure
3001 to 3300	sets ICL programmer defined integer variables
4001 to 4010	sets integer data from last registration event (see Appendix A for details)
4906	set to 1 to stop executing in more registered events for the current PD event
5000	sets number of characters to get in s_var(6xxx)
5001 to 5600	sets data for next device procedure call

set_lay_echo(from_lay, to_lay, flag)
Sets the layers from layer from_lay to layer to_lay to be visible (flag = 1), invisible (flag = 0) or to toggle the layer status (flag = -1).

set_lmask(from_lay, to_lay)
Sets entity layer selection mask for which layers may be idented in getdata. Sets layers from from_lay to to_lay (both must be positive or negative).

< 0	allow all layers to be picked except those in the specified range
= 0	allow all layers to be picked
> 0	only allow layers in specified range to be picked

set_mouse_str(index1, index2, index3, index4, index5, text)
Writes string text into the current mouse macro file for the given index values.

set_mps(imp_word_begin, imp_word_end, iselect)
Sets the MP word selection status of the range imp_word_begin to imp_word_end. One (1) is selected, zero (0) is not selected.

set_mpsf(imp_word)
Sets the MP word selection status to false (not selected).

set_mpst(imp_word)
Sets the MP word selection status to true (selected).

set_mpsxt(imp_word)
Sets the MP word selection status to true (selected). Any mutually exclusive words associated with imp_word will be set to false.

set_mpv(imp_word, r2)
Sets the value associated with the MP word to r2 (integer or real).

set_mp_char(ichr)
Sets the character (given the ASCII value by ichr) which can be entered by the user in the MP to invoke its associated dbox, if any. Normally this character is '@' (64). Set to 0 to disable this feature.

set_number_format(type, width, n_dec)
Sets the parameters on how real numbers are converted internally to strings for display.

type	1 convert to integer 2 floating point (default) 3 same as 2 but remove any trailing 0's 4 same as 3 but also remove trailing '.'
width	number of characters to format number in. If less than 0, then left justify; greater than 0, right justify. If 0 (default) then just use the minimum number of characters necessary.
n_dec	number of decimal places to display. If < 0 then use E format, if 0 (default) then use as many decimal places as is necessary.

set_position(i1, i2, i3, i4, i5)
Defines the position for the next d_box_add, icon, or move procedure. The i1 p-position in the item that is to be located will be justified on the i2 p-position of the target item. i3 is the positioning method as follows:

0	???
1	The target item is the entire screen. i4 and i5 are not used.
2	The target item is the graphics window. i4 and i5 are not used.
3	The target item is the dbox i4. i5 is not used.
4	The target item is the field i5 in dbox i4.
5	The target item is the menu i4. i5 is not used.
6	The target item is the MV i4. i5 is not used.
7	The target item is the absolute pixlel coordinates i4,i5. i2 is not used.

set_r_data(scommon, ioffset, r3)
Puts the real value r3 in the common block scommon at offset ioffset.

set_r_var(i1, r2)
Sets the r_var(i1) real number variable to the value r2.

1	sets current display scale
3	sets current default Z depth
1001 to 2011	same as UPL sysvarr intrinsic procedure
3001 to 3100	sets ICL programmer defined real variables
5001 to 5600	sets data for next device procedure call. Note: addressed on word boundaries.

set_status_fmt(item, sformat)
Sets status window item to "C" style format sformat. (see update_status_info)

set_s_data(scommon, ioffset, s3)
Puts the string value s3 in the common block scommon at offset ioffset.

set_s_var(i1, s2)
Sets the s_var(i1) string variable to the value s2.

1 to 25	sets file names from PD config file lines 1 to 25
51	sets user name
52	sets network name
3001 to 3020	sets ICL programmer defined string variables (128 characters maximum each)
4001 to 4020	sets string data from last registration event (see Appendix A for details)
5001 to 5600	sets string date for next device procedure call.

set_xfmask(font)
Sets entity line font selection mask for which line fonts may be idented in getdata. (see also set_fmask)
< = 0 allow all line fonts to be picked
> 0 allow all fonts to be selected except the specified font number

toggle(dlf, fld, itog) ***UPL***
Toggles the specified toggle type field by changing the toggle position by itog amount. itog can be positive or negative.

undo
Executes the PD undo function. This command will not effect the PD command stream.

unreg(reg_id, reg_no)
Unregisters events registered with reg_dbox_add, reg_dbox_del, reg_exec_dlg and reg_exec_fld. All events registered with the given reg_id and reg_no are unregistered. If reg_id is 0 then all events registered with reg_no will be unregistered. If reg_no is 0 then all events registered with reg_id will be unregistered. If both reg_id and reg_no are 0 then all registered events will be unregistered.

update_status_info(iflag)
Updates display of status window. Valid values of iflag are:
-1 turn status window off and set the default condition to be status window off
0 turn status window off
1 clear status window and redisplay all data
2 update changed data in status window

Keywords

Conditionals

if

then

elseif

else

endif

and

or

Looping

goto

active

bkg_color

cur_color

cur_style

Procedures continued

define

end

fgr_color

fld

hlight_color

include

none

position

type

Special Characters

@ String concatenation.

**$** Word or phrase preceding "$" is to be interpreted directly. It is not translated because of a previous define statement.

PIXL and BEVL Properties

These properties may be attached to String entities to produce pixel based images.
With the exception of BEVL properties types 3 and 4 the string entities must be 5-vertex closed rectangles.

BEVL Properties

The primary purpose of the BEVL property is to make a string look like a raised button or a recess field. The data format is as follows:

Property Name	Data
BEVL	<type> <data>...<data>...

If <type> is an odd number the original string is drawn.
If <type> is an even number the original string is NOT drawn.

The data following the type value will vary depending on the type.

Purpose	Type	Data
Button	1-2	<npix> <color1> <color2>
Polyfill	3-4	<color1>
Button w/ Backgrnd	5-6	<npix> <color1> <color2> <color3>
Tall Recess Field	7-8	<npix> <color1> <color2> <npix2>
Recess Field	9-10	<npix> <color1> <color2> <npix2>

The most commonly used forms of the BEVL property are:

BEVL (all data omitted)	Makes a raised button, no background color.
BEVL 5 1 8 15 <bkgcol>	Makes a raised button, <bkgcol> is the background color.
BEVL 10 -1 15 0	Makes a recess field.
BEVL 3 <fillcol>	Fills a string polygon with the color <fillcol>. If <fillcol> is omitted the entity color is used.

Type	Data
1-2	<npix> <color1> <color2>

<npix> = Number of pixels for box border.
+ is inside of string.
- is outside of string.
<color1> = Color of lower and right edges.
<color2> = Color of top and left edges.

BEVL Properties continued

Type Data
3-4 <color1>

<color1> = Color of polyfill.
if <color1> is omitted <color1> = entity color.

Type Data
5-6 <npix> <color1> <color2> <color3>

<npix> = Number of pixels for box border.
+ is inside of string.
- is outside of string.
<color1> = Color of lower and right edges.
<color2> = Color of top and left edges.
<color3> = Color of background.

Types 9 and 10 automatically widen the field so that there is a small gap between the recess border and text displayed in the field.

Type Data
9-10 <npix> <color1> <color2> <npix2>

<npix> = Number of pixels for box border.
+ is inside of string.
- is outside of string.
<color1> = Color of lower and right edges.
<color2> = Color of top and left edges.

PIXL Properties

This property is used to create simple pixel based drawings and automatically sized text inside a rectangular string boundary.

Property Name	Data
PIX?????	<type> <data>...<data>...

Any property name beginning with PIX may be used. If the name PIXL is used, a TX subrecord (if present) will be appended to the data.

Purpose	Type	Data
Gtext	1	<boxjust> <color> <text>...
PD Text	2	<boxjust> <color> <angle> <textjust> <font> <slant> <hgt> <wdt> <lnsp> <text>...
Connected Lines	3	<boxjust> <scale> <maxX> <maxY> <color> <p1.x> <p1.y> <p2.x> <p2.y> ...repeat...
Disconnected Lines	4	<boxjust> <scale> <maxX> <maxY> <color> <p1.x> <p1.y> <p2.x> <p2.y> ...repeat...
Disconnected (mul-col)	5	<boxjust> <scale> <maxX> <maxY> <color> <p1.x> <p1.y> <p2.x> <p2.y> ...repeat...
Polyfill	6	<boxjust> <scale> <maxX> <maxY> <color> <pn.x> <pn.y> ...repeat...
PD Text w/ Shadow	200+	same as type 2

PIXL Properties continued

The following parameters are the same for all pixel object types:

<boxjust>	K-Position Pixel object justification in rectangle.
<color>	Pixel object color.

The following parameters are specific to each pixel object type:

<angle>	PD text angle (0 or 90).
<textjust>	PD text justification (1=left, 2=right, 3=center).
	if <textjust> <> (1 or 2 or 3) then
	if <boxjust> = (1 or 4 or 7) then
	<textjust> = 1 (left)
	else if <boxjust> = (2 or 5 or 8) then
	<textjust> = 3 (center)
	else if <boxjust> = (3 or 6 or 9) then
	<textjust> = 2 (right)
	endif
	endif
<font>	PD text font.
<slant>	PD text slant.
<hgt>	PD text height.
	If <hgt> = 0.0 then <hgt> = 50% of box height.
	if <hgt> < 0.0 then
	abs(<hgt>) is % of box height above and below text.
	<hgt> = box height * (1.0 + <hgt> * 0.02)
	endif
<wdt>	PD text width.
	if <wdt> = 0.0 then
	Total text width is constrained by box width.
	endif
	if <wdt> < 0.0 then
	Total text width is constrained by abs(<wdt>).
	endif
<lnsp>	PD text line spacing.
<text>	Text string.
<scale>	Pixel object scale factor.
	if <scale> <= 0 then
	<scale> is determined by <maxX>, <maxY>,
	and box size.
	endif
<maxX>	Pixel object X extent.
<maxY>	Pixel object Y extent.
<pn.x>	Vertex n X-coordinate.
<pn.y>	Vertex n Y-coordinate.

Shadow Color for types 200+:

```
if <type> >=200 and <type> <= 215 then
    shadow color = <type> - 200
else if <type> = 216 then
    shadow color = <color>
else if <type> = 217 then
    if BEVL <color3> is between (0 and 6) then
        if <color> = 15 then
            shadow color = 0
        else
            shadow color = 15
        endif
    else
        if <color> = 0 then
            shadow color = 15
        else
            shadow color = 0
        endif
    endif
else if <type> = 218 then
    shadow color = abs(<color>-15)
else if <type> = 219 then
    if <color> is between (0 and 6) then
        shadow color = 15
    else
        shadow color = 0
    endif
endif
```

The PD Dialog Box Menu

If you wish to modify or add to the PD dbox menu it is best to adhere to its conventions. The PD dbox menu definition is contained in many files which are described below.

DEFINE.CMD

This file contains definition statements that are globally useful. Some of these definitions are described here. The phrases appear in approximately the order that they appear in the file DEFINE.CMD.

mp_style_active

To be placed on a dlg statement line. Use for commands that interact with the modifier processor. Sets active bits 1, 2, 7, 12.

button_push

To be placed on a fld statement line. Will cause the field to have the push button acknowledgment.

The following are cursor types. These may be placed on the dlg or fld statement lines.

fld_arrow	White Arrow
inactive_cur	"Pick to Activate" cursor
move_cur	Four-direction Arrow
dlg_arrow	Black Arrow
hand_cur	Pointing Hand

The following are the defaults that are used for the dlg and fld statements if the corresponding statements are not specified on the dlg or fld statement line.

cur_style	fld_arrow	(white arrow)
hlight_color	00	(don't highlight)
bkg_color	7	(field background color 7)
fgr_color	0	(field text color 0)
position	5	(dbox activated centered over cursor)
active	bnormal_active	(bit 1 set)

rad_to_deg

Multiply by this to convert radians to degrees, divide to convert degrees to radians.

no_hilight

Put this phrase on a fld statement line to indicate that the field should NOT be highlighted when the cursor moves over the field. *(Since this is the default this does not need to be done unless the default is changed.)*

hilight

Put this phrase on a fld statement line to indicate that the field should be highlighted when the cursor moves over the field.

The following phrases are intended to be used in conditional statements to determines the current command status. For example:
if in_vnp then (if the command is in the Verb/Noun Processor then)
if in_dig_mode (if the command is in Getdata Digitize Mode then)

in_vnp
not_in_vnp
in_mp
not_in_mp
in_getdata
not_in_getdata
in_dig_mode
not_in_dig_mode
in_end_mode
not_in_end_mode
in_ent_mode
not_in_ent_mode

vnphlp
This is a global variable that contains the current VNP index.

mphlp
This is a global variable that contains the current Modifier Processor index.

dighlp
This is a global variable that contains the current Getdata help index.

do_fld_dig
This is used when a fld_dig ER is specified and the default fld_dig ER action is to be combined with other statements. For example:

```
fld 401 type 2                --active layer
fld_dig
   do_fld_dig
   set_i_var(3, fldiv(-1,-1)) --set current
   update_status_info(2)      --force call to status to update
changed items
end
```

push_in
This is used with the procedure push_button to make a button field look pushed in.
The syntax is: push_button(dlg,fld,push_in)

pop_out
This is used with the procedure push_button to make a button field look popped out (normal).
The syntax is: push_button(dlg,fld,pop_out)

microDRAFT
This phrase is used in conditional statements to determine whether microDRAFT (the 2-D version) or PD (the 3-D version) is running. For example:
if microDRAFT then (if this is the 2-D version then)

graphics_x_min
This is a global variable containing the left edge of the graphics window in pixel coordinates.

graphics_y_min
This is a global variable containing the bottom edge of the graphics window in pixel coordinates.

graphics_x_max
This is a global variable containing the right edge of the graphics window in pixel coordinates.

graphics_y_max
This is a global variable containing the top edge of the graphics window in pixel coordinates.

screen_x_min
This is a global variable containing the left edge of the screen in pixel coordinates.

screen_y_min
This is a global variable containing the bottom edge of the screen in pixel coordinates.

screen_x_max
This is a global variable containing the right edge of the screen in pixel coordinates.

screen_y_max
This is a global variable containing the top edge of the screen in pixel coordinates.

over_box
This is a global variable containing the dbox number that the cursor is currently over.

over_fld
This is a global variable containing the field number that the cursor is currently over.

from_dbox
A function that returns the parent dbox of the dbox in which this phrase appears.

from_dfld
A function that returns the parent field of the dbox in which this phrase appears.

activate
A procedure that makes all fields active (bit 1 set to 1) in the dbox in which this phrase appears.

deactivate
A procedure that makes all fields inactive (bit 1 set to 0) in the dbox in which this phrase appears. This will not effect fields for which bit 2 is set to 1.

set_active_cur
Sets the dbox cursor to the black arrow, usually to replace the "Pick to Activate" cursor.

set_inactive_cur
Sets the dbox cursor to the "Pick to Activate" cursor.

check_mark
Sets up the type and fld_value statements to produce a check mark field. Position one (1) is "checked" (field value 1), position 2 is "unchecked" (field value 2). The default condition is "checked". It is best to include a fld_begin ER that initializes the "checked/unchecked" condition. Remember that when setting this type of field it is the position, not the value, that is entered.

check_mark_off
Same as check_mark except that the default condition is "unchecked".

pinned
Used with conditionals to determine whether the dbox in which the phrase appears is "pinned down" (field 219, position 2, value 1).

not_pinned
Used with conditionals to determine whether the dbox in which the phrase appears is NOT "pinned down" (field 219, position 1, value 0).

mp_char
A string field value containing the current special character that activates the dbox associated with the current Modifier Processor index.

move_button
This phrase completely defines a "Move" field (the field that is used for relocating a dbox) for the dbox in which it appears.

move_dlg_to_top
This phrase defines the dlg_dig ER to have the function that will cause the dbox to be moved to the foreground (if it is not already). It is important to note that if this phrase is followed by another dlg_dig ER, the ER defined by this phrase will be overwritten.

dummy_100
This is a shorthand definition of a type 100 dummy field. It is equivalent to the following statements (which would follow a fld statement):

```
type 100 active 0b0001
```

dummy_2
This is a shorthand definition of a type 2 dummy field. It is equivalent to the following statements (which would follow a fld statement):

```
type 2 active 0b0001
```

take_down_button
This phrase defines a "Take Down" field for the dbox in which it appears (the field that is used for removing a dbox from the screen, typically a square button with a dash in the upper left corner of the dbox). It should be accompanied by a fld_dig ER containing the action desired at the time the dbox is removed, for example:

```
take_down_button
fld_dig
   take_down_done_action
end
```

pin_down_button
This phrase completely defines a "Pin Down" field for the dbox in which it appears. This is the field that is used to hold a dbox on the screen even after the command has been terminated.

m_pin_down_button m_pin_end
These two phrases are similar to the single pin_down_button phrase, except that both phrases must be used, and other statements may be placed between the m_pin_down_button phrase and the m_pin_end phrase.

DLGNUM.CMD
This file contains definitions for most of the dbox numbers. A related file is NURBNUM.CMD.

DIAL.CMD
This file contains all of the other .CMD files as include files (possibly nested include files).

STARTUP.CMD
This file contains the dlg_start_up ER.

DLG*.DRW and USER*.DRW

Any PD part begining with "DLG" or "USER" is assumed to be part of the menu.

***.CMD**

All ICL files should use the extention ".CMD".

The Build Dialog Command

The PD command Build Dialog (BU DIAL for short) is used to compile dialog box drawings and ICL files into a menu.

The three elements that may be compiled are:

1. The dbox base images.
2. The icon images.
3. The ICL files.

These elements may be compiled separately or all together in one command. The simplest way to compile dboxes is to activate a drawing with dbox base images and icons and then enter the command BU DIAL<CR>. This will cause all of the dboxes and icons defined in the active drawing to be added to (or to modify) the menu file that is active in PD at the time the command is given. For this method the ICL file will also be compiled, and its name is assumed to be the same as the drawing name except that it has the extension ".CMD". The ICL file must contain all of the ICL code for all of the dboxes in the drawing, or have "include" references to other ICL files with the needed code. Any time a dbox base image is compiled, its ICL file must also be compiled. Compiling the base image alone will cause any existing ICL code for the dbox being compiled to be erased. However, it is possible to compile ICL code without compiling the base image or icons.

The build dialog command has the following modifiers:

ICons	Compile icons.
NOICons	Do not compile icons.
DBoxs	Compile dbox base images.
NODBoxs	Do not compile dbox base images.
DOCMD	Compile the ICL file.
NOCMD	Do not compile the ICL file.
CMDfile	ICL file name to compile. If not specified, assumed to be the same as the current drawing name with the ".CMD" extension.
LOGfile	Log file name. If not specified, assumed to be the same as the current drawing name with the ".LOG" extension. This file will contain a record of the ICL compilation including any error messages.
BLAYer	Beginning layer.
ELAYer	Ending layer. These modifiers may be used to restrict the compilation of base images and icons to those on the specified layers.
BDIALog	Beginning dbox number.

EDIALog Ending dbox number.
These modifiers may be used to restrict the compilation of base images to the specified range of dbox ID numbers.

The following examples illustrate some other ways that you might want to use the build dialog command:

To compile an individual command file (common during the development process):

>>BU DIAL NODB NOIC CMD [icl file name]<CR>

To compile one or more dboxes that are isolated on a layer in a drawing that has several dbox definitions:

>>BU DIAL BLAY [layer] ELAY [layer] CMD [icl file name]<CR>

For example, the Insert Line dboxes are in the drawing DLGGEO.DRW on layer six. The ICL file for insert line is INSLIN.CMD.

>>BU DIAL BLAY 6 ELAY 6 CMD INSLIN<CR>

A Step-by-Step Example

1. From DOS change the directory to the directory that contains the dbox definition files (probably \PD\NEWMENU).

2. Activate a new drawing. The naming convention for dbox definition drawings is DLG?????.DRW or USER????.DRW. A good name for this drawing would be DLGMAKE.DRW.

3. >>SELect PIXel ALL. You will be placing special properties on string entities in your drawing that will have various pixel based visual effects such as making them look like raised buttons and recessed fields. This command will allow you to see what your dbox looks like before it is compiled.

4. >>SELext GRid G .125 ON. Because the dbox is comprised of very precise images, it is important to be sure that the geometry that defines the dbox image (especially the dbox boundary and field boundaries) falls on a 0.0625 grid snap. The UPL program RECTIFY may be used to adjust the coordinates of lines and strings to fall exactly on the nearest 0.0625 grid snap.

5. >>SELect COLor DEFault 8. ALL dbox boundaries and field boundaries should be dark gray. This will make them easier to identify when they are overlapping one another.

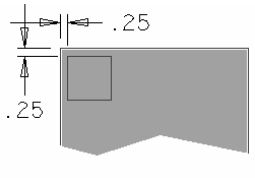
6. >>RESTore TFont TF000444.FNT. Text font 444 was specifically developed for dboxes. It was created to look good with the fewest possible vectors. Also, the lower case letters look like small upper case letters, which are easier to read.

7. >>INSert RECtangle STring COLor 8 DX 6.5 DY18.375:x1y1;. This will create the string that will be used to define your dbox boundary. Note that it is positioned entirely in the first Cartesian quadrant.

8. >>INSert PROPerty NAME DLG_BOX VALue '1565 7':[pick the dbox string]. This identifies this string as the boundary for dbox 1565 with background color 7 (light gray). >>REPAint to see the result.

9. >>INSert PROPerty NAME BEVL VALue '1 1 15 15':[pick the dbox string]. This will put a thin white boarder around the dbox. >>REPAint to see the result.

10. Create a rectangular string 1.5 by 1.5 with the upper left corner 0.25 from the left edge and top edge of the dbox.

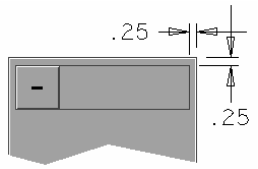


11. Add the following properties to the new string:

    a. NAME DLG_FLD VAL '200 0'. 200 is the standard field number for the Take Down field.
    b. NAME BEVL [no value]. A BEVL property with a blank value will create edges on the field to make it look like a raised button.
    c. NAME PIXL VAL '6 5 1 10 2 0 0 0 10 0 10 2 0 2 0 0'. This makes the dash symbol centered in the field.

    This will create the 'Take Down' button. >>REPAint to see the result.

12. Create a rectangular string from the lower right corner of the Take Down button to within 0.25 of the right edge and 0.25 of the top edge of the dbox.



13. Add the following properties to the new string:

    a. NAME DLG_FLD VAL '220 0'. 220 is the standard field number for the Move field.
    b. NAME BEVL VAL '5 1 15 15 5'. This will make a white boarder around the field and make the field background teal (color 5).
    c. NAME PIXL VAL '219 5 15 0 0 0 0 0.625 0 1.5 Dbox'. This will put text in the center of the field. The text will be .625 high (10 pixels) and the width will be constrained so that the text will never go outside the field boundary. The text will be white with a black drop-shadow.

    This will create the 'Move' button. >>REPAint to see the result.

14. Save the part, shell to DOS, edit the file DLGMAKE.CMD. If you chose a different name for the drawing, edit a file having that name with the '.CMD' extension. Create the following text in this file:

```
include define.cmd --all definitions

--Make Dboxes-------------------------------------------------

dlg dlgmake_box position -5 active 0b1000001 cur_style dlg_arrow

take_down_button
fld_dig
 take_down_done_action
end

move_button
```

15. Save the ICL file and return to PD.

16. >>BUild DIALog. This will compile your dbox with the two fields you defined.

17. Make a way to activate your dbox. Open the User dbox by moving over the tab that says "User".

18. Pick the button in the user box that says "Edit".

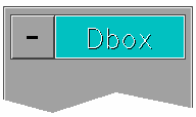
19. Pick a button to edit, for example the 'M' (for Make).

20. Enter the following text:

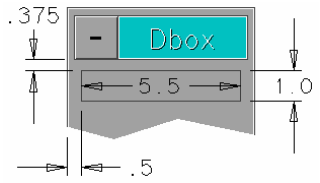
    \D1565\

    Pick the "Done" button on the "Edit Dialog Box" dbox. Then pick the "Take Down" button on the "Edit Dialog Box" dbox.

21. Pick the "M" button on the user dbox to activate your dbox. Test the Move button and the Take Down button.



22. Create a rectangular string 5.5 wide by 1.0 high with the upper left corner 0.375 below the Take Down button and 0.5 to the right of the left edge of the dbox.



23. Add the following properties to the new string:

    a. NAME DLG_FLD VAL '301 1'. 301 is an arbitrary value for the field ID. The one (1) indicates that the field boundary should be adjusted to fall on the nearest character cell boundary. For fields that are meant to display text, the height should be multiples of 1.0 (1 row) and the width should be multiples of 0.5 (1 column).
    b. NAME BEVL VAL '10 -1 15 0'. This will make the field look recessed.

    >>REPAint to see the result.

24. Move a copy of this field down exactly 1.375.

25. >>EDIt PROPerty NAME DLG_FLD :[pick the copied field]. Change the value to 302 1 so that the new field will be unique from the field you copied.

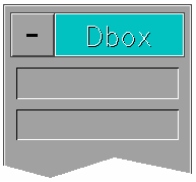
26. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

```
fld 301 type 22                          --pixel on/off
fld_value 1 3
           0          1           2
 "Pixel Off" "Pixel On" "Pixel All"
end
fld_begin
 set_fldiv(-1,-1,3)
end
fld_dig
 do_fld_dig
 set_i_var(1329,fldiv(-1,-1))
 repaint(0.0)
end

fld 302 type 22                          --dial scale on/off
fld_value 1 2
             0            1
 "DialScl Off" "DialScl On"
end
fld_dig
 do_fld_dig
 if fldiv(-1,-1) = 0 then
  input('#13#SEL DIAL SCLOFF#13#')
 else
  input('#13#SEL DIAL SCLON#13#')
 endif
end
```

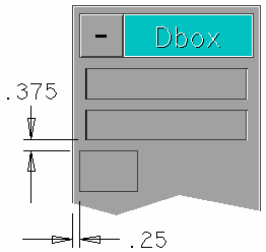
27. Save the ICL file and return to PD.

28. >>BUild DIALog. This will compile your dbox adding the new fields 301 and 302.

29. Activate your dbox again by picking the "M" button on the "User" dbox. Now try out the new fields. The first one (field 301) will cycle through the three pixel conditions. The second one (field 302) will toggle the dial scale on and off. If you set these fields to "Pixel On" and "DialScl On" you will be able to see exactly what your dbox base image will look like when it is compiled. It is best not to do construction, modification, or file your part while in the "DialScl On" mode. The "DialScl ON" mode can be simulated by setting the screen scale to 0.23 (>>ZOom SCale .23).



30. Create a rectangular string 2.0 wide by 1.375 high with the upper left corner 0.375 below the second recessed field (field 302) and 0.25 to the right of the left edge of the dbox.



31. Add the following properties to the new string:

    a. NAME DLG_FLD VAL '303 0'.
    b. NAME BEVL [no value].

    >>REPAint to see the result.

32. Move a copy of this field as follows:

    >>MOVe Copy:[pick the new string];x0y0,ix2n2,ix-4iy-1.375ix,ix2n2;

    This will make an array of buttons that is 3 across and 2 down.

33. Edit the DLG_FLD property (>>EDIt PROPerty NAME DLG_FLD :) on each of these strings so that the field ID numbers are as shown below:

34. Inside field 303 draw an image that will represent the INSert RECTangle command, and inside field 304 draw an image that will represent the SELect GRid command. For example:

35. Add the following property to fields 306, 307, and 308:

    NAME PIXL VAL '2 5 0 0 0 0 0 0.625 0 1.5 '. **It is very important to be sure that the last character in this property value is a space.**

36. >>EDIt TEXt:[pick the string that is field 306] ; [enter the text "Del"]. This illustrates the fact that the value for a PIXL property may be partially contained in a TX subrecord. This means that a string with this type of property can be treated like a text entity as far as the EDIt TEXt command is concerned.

37. Repeat step 36 adding the text string "Mov" to field 307 and "Str" to field 308. *Field 305 will remain unused for the time being.*

38. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

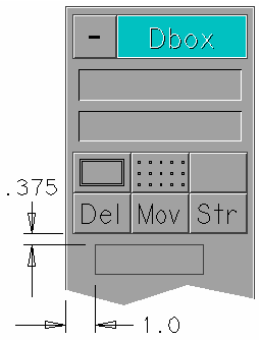
```
fld 303 type 100                          --INS RECT
fld_dig
 if vnphlp <> 2154 then
  input('#13#INS RECT STR COLOR 8 @')
 endif
end

fld 304 type 100                          --SEL GRID
fld_dig
 if vnphlp <> 2009 then
  input('#13#SEL GRID @')
 endif
end

fld 305 type 100 fld_dig end              --NOT USED

fld 306 type 100                          --DEL ENT:
fld_dig
 if vnphlp <> 2005 then
  input('#13#DEL ENT:')
 endif
end

fld 307 type 100                          --MOVE
fld_dig
 if vnphlp <> 1003 and fldiv(-1,309) = 0 then
  input('#13#MOVE:')
 elseif vnphlp <> 2008 and fldiv(-1,309) = 1 then
  input('#13#MOVE COPY:')
 endif
end

fld 308 type 100                          --STRETCH
fld_dig
 d_box_add(stretch_box)
end
```

39. Save the ICL file and return to PD.

40. >>BUild DIALog. This will compile your dbox adding the new fields 303 through 308.

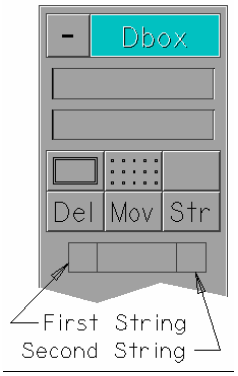
41. Activate your dbox again by picking the "M" button on the "User" dbox. Test the new fields.

42. Create a rectangular string 3.75 wide by 1.0 high with the upper left corner 0.375 below the "Del" button (field 306) and 1.0 to the right of the left edge of the dbox.



43. >>INSert PROP NAME BEVL VAL '2 0':[pick the new string]. This will make this string invisible when the dbox is compiled.

44. >>MOVe Copy:[pick the new string];x0y0,ix1;.



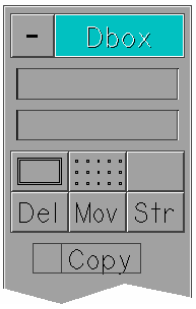
45. >>INSert PROP NAME DLG_FLD VAL '309 0':[pick the first of the two new strings (the one on the left). Be sure to pick it where only the first string can be identified]. This makes this string into field 309.

46. Add the following property to the second string:

    NAME PIXL VAL '2 4 0 0 0 0 0 0.625 0 1.5 '. **It is very important to be sure that the last character in this property value is a space.**

47. >>EDIt TEXt:[pick the second string, be sure to pick it where only the second string can be identified] ; [enter the text "Copy"]. This makes this string into a label.

48. >>STRetch WINdow:[make a window around the right end of the second string] x0y0,ix-1;. This shortens the label so that it is entirely within field 309. The left end of the label is shortened to make room for the "Check Mark" icon.



49. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

```
fld 309 check_mark                          --COPY
fld_dig
 do_fld_dig
 if vnphlp <> 1003 and fldiv(-1,309) = 0 then
  input('#13#MOVE:')
 elseif vnphlp <> 2008 and fldiv(-1,309) = 1 then
  input('#13#MOVE COPY:')
 endif
end
```

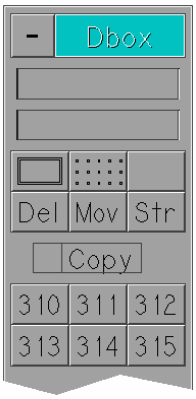
50. Save the ICL file and return to PD.

51. >>BUild DIALog. This will compile your dbox adding the new field 309.

52. Activate your dbox again by picking the "M" button on the "User" dbox. Test the new field. It allows the "Mov" button to also do MOVe Copy.

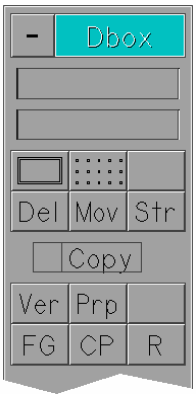
53. Move a copy of fields 303 through 308 down 4.5. Move only the field boundaries, not the graphics for INSert RECtangle or SELect GRid.

54. Edit the DLG_FLD property (>>EDIt PROPerty NAME DLG_FLD :) on each of the new strings so that the field ID numbers are as shown below:



55. Run the UPL program CP. When CP prompts for a "Property Type", enter "PIXL" followed by a space. Pick field 313, followed by field 310. This will copy the property and text from field 313 to field 310. Repeat this process, copying from field 310 to field 311 (leave field 312 blank).

56. Edit the text on these fields so that they appear as follows:



57. Edit the BEVL property on these fields so that they have the following value:

    5 1 8 15 4

58. Edit the PIXL property on these fields so that they have the following value:

    219 5 15 0 0 0 0 0.625 0 1.5

    Be sure that the last character is a space.

59. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

```
fld 310 type 100 fld_dig run_upl(s_var(90)@"VER") end
fld 311 type 100 fld_dig run_upl(s_var(90)@"VPROP") end
fld 312 type 100 fld_dig end
fld 313 type 100 fld_dig run_upl(s_var(90)@"FG") end
fld 314 type 100 fld_dig run_upl(s_var(90)@"CP") end
fld 315 type 100 fld_dig run_upl(s_var(90)@"RECTIFY") end
```

60. Save the ICL file and return to PD.

61. >>BUild DIALog. This will compile your dbox adding the new fields 310 through 315.

62. Activate your dbox again by picking the "M" button on the "User" dbox. These new buttons run UPL programs that are useful for developing dboxes.

VER	Similar to the PD verify command.
VPROP	Similar to VER except for properties.
FG	Move selected entities to the foreground (later in the database).
CP	Copy properties.
R	(Rectify) Make string and line coordinates fall on a 0.0625 grid snap.

    Your dbox is starting to become a useful tool for dbox development.

63. Move a copy of the strings that define fields 313, 314, and 315 down 1.625.

64. Edit the DLG_FLD properties so that they become 316, 317, and 318.

65. Edit the BEVL properties on fields 316, 317, and 318 to change the value to blank (the default for a raised button).

66. To completely remove the text from fields 316 and 317 delete the PIXL property, and then edit the text on the string and erase the text.

67. On field 318 edit the text to say "Txt". Edit the PIXL property so that it has the following value:

    2 5 0 0 0 0 0 0.625 0 1.5

    Be sure that the last character is a space.

68. Create a rectangular string centered in fields 316 and 317.

69. Run the UPL program CP (you could use your dbox to do this). At the prompt enter "BEVL" followed by a space. Then pick any string that is a raised button without a colored background, followed by the string in the middle of field 316.

70. Repeat step 69 except pick a recessed field and then the string in the middle of field 317. >>REPAint to see the result.

71. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

```
fld 316 type 100
fld_dig
 input('#13#INS PROP NAME BEVL :')
end

fld 317 type 100
fld_dig
 input('#13#INS PROP NAME BEVL VAL "10 -1 15 0":')
end

fld 318 type 100
fld_dig
 input('#13#INS PROP NAME PIXL VAL "2   5  0 0 0 0 0 0.625 0 1.5 ":')
end
```

72. Save the ICL file and return to PD.

73. >>BUild DIALog. This will compile your dbox adding the new fields 316 through 318.

74. Activate your dbox again by picking the "M" button on the "User" dbox.
    Field 316 adds the raised button property.
    Field 317 adds the recessed field property.
    Field 318 adds the text PIXL property.

75. As you may have noticed by this time, one of the easiest ways to create the graphics for a dbox is to copy the various elements from another dbox definition. The dbox you are building will facilitate this process. Next, move a copy of the FG button (field 313) down 3.25. This button was chosen because it is similar to the buttons you are about to make.

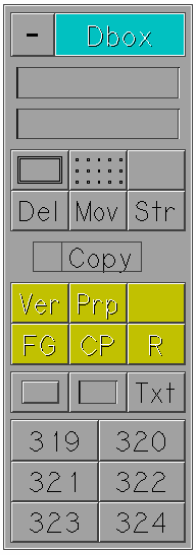
76. Stretch the right end of the button 1.0 to the right.

77. Move a copy of this field as follows:

    >>MOVe Copy:[pick the new string];x0y0,ix3,ix-3iy-1.375ix,ix3,ix-3iy-1.375ix,ix3;

    This will make an array of buttons that is 2 across and 3 down.

78. Edit the DLG_FLD property (>>EDIt PROPerty NAME DLG_FLD :) on each of the new strings so that the field ID numbers are as shown below:



79. Edit the text on these fields so that they appear as follows:

80. Shell to DOS, edit the ICL file, and add the following text at the end of the file:

```
fld 319 type 100 fld_dig input('#13#EDIT TEXT:') end
fld 320 type 100 fld_dig input('#13#EDIT PROP NAME PIXL:') end
fld 321 type 100 fld_dig input('#13#EDIT PROP NAME BEVL:') end
fld 322 type 100 fld_dig input('#13#EDIT PROP NAME DLG_BOX:') end
fld 323 type 100 fld_dig input('#13#EDIT PROP NAME DLG_FLD:') end
fld 324 type 100 fld_dig input('#13#EDIT PROP NAME DLG_ICN:') end
```

81. Save the ICL file and return to PD.

82. >>BUild DIALog. This will compile your dbox adding the new fields 319 through 324.

83. You can add one more item to the command file as a convenience as well as a learning experience. Shell to DOS, edit the ICL file, and add the following text after the dlg ID statement at the beginning of the file:

```
dlg_begin
 input('#13#RESTORE TF TF000444.FNT#13#')
end
```

    This will cause the dbox text font to be automatically restored whenever you activate your new dbox.

84. Save the ICL file and return to PD.

85. >>BUild DIALog NODBox NOICon CMDfile DLGMAKE. This will compile only the ICL file for your dbox. This can be quite a bit faster than compiling everything (i.e., ICL code, graphics and icons).

Your dbox is now complete. Of course you could have built all of the graphics, edited the entire ICL file, and then compiled everything just one time. The purpose of the process used in this example was to illustrate the connection between each item in the dbox and its related properties and ICL code.

You may find that using your new dbox will assist you in developing other dboxes.

GOOD LUCK!

Appendix A
Dialog Box Registration

ICL commands can be automatically invoked by registering them to execute when a particular event occurs in PD.

Information directly related to the registered event is passed to and from the invoked ICL code via the i_var, r_var, s_var and set_i_var, set_r_var set_s_var ICL functions. The following information is common to all events:

i_var(4901) = current number of events registered
i_var(4902) = maximum number of events that can be registered
i_var(4903) = current registered event handle number (= 0 if not currently executing ICL code because of a registered event)

if i_var(4903) is not equal to 0 then
- i_var(4904) = current registered event ID number (as specified in reg_d_box_add, reg_d_box_del, reg_exec_dlg or reg_exec_fld command as reg_id)
- i_var(4905) = current registered event registration number (as specified in reg_d_box_add, reg_d_box_del, reg_exec_dlg or reg_exec_fld command as reg_no)
- i_var(4906) = current registration event type.
  1 - reg_d_box_add
  2 - reg_d_box_del
  3 - reg_exec_dlg
  4 - reg_exec_fld

Note that more than one function can be registered to an event. The most recent registered functions for a given event are executed first. During execution of a registered function, i_var(4907) can be set to 1. This will suppress execution of other functions with the same reg_no for the duration of the current event.

The following is a list of registration numbers (reg_no) available in PD (see reg_d_box_add, reg_d_box_del, reg_exec_dlg_cmd, reg_exec_fld_cmd):

1 Before the Verb Noun command processor is invoked
- Input: none
- Output: i_var(4001) = vnp_no. If vnp_no <> 0 then use vnp_no to determine which VNP command to execute next. If i_var(4001) = 0 then get VNP command as usual from user.

2 After a VNP command is done and before the 'auto take down' DLG bit causes the d_box_del to be executed
Input: none
Output: none

3 In the Modifier Processor, can override the default. This event only will occur if the dlg MP character (normally '@') is set to 0 (ASCII NUL).
Input: none
Output: if i_var(4001) = 0 then do default d_box_add with current modifier index number. if i_var(4001) = dbox_no then do d_box_add with dbox_no.

4 In the status window update routine
Input: i_var(4002) = status flag,
i_var(4003) = mode
1 = clear status window and display all data.
2 = update changed data in status window.
3 = remove status window display.
4 = clear status window and display all data, don't check dlgs.
i_var(4004) = status window number.
Output: if i_var(4001) = 1, do not continue with normal status window update. if i_var(4001) = 2 enable event numbers 101 to 166.
i_var(4001) = 3 is same as i_var(4001) = 2 except that all status items are reinitialized so they will be forced to be updated.

5 Not used.

6 Invoked before the XH coordinates are displayed in the XH coordinate display window.
Input: s_var(4001) = string to be displayed in XH coordinate window.
Output: if i_var(4001) = 1 then string is not displayed in XH window.
if i_var(4001) = 0 then the string is displayed in the window as it normally would be.

7 At the beginning of getdata.
Input: i_var(4003) = current number of digitizes.
i_var(4004) = maximum number of digitizes to get.
i_var(4005) = the current keyboard macro set in use.
Output: i_var(4001) = 1 return immediately from getdata.
i_var(4001) = 0 proceed with normal getdata processing.

8 At the end of getdata.
Input: i_var(4003) = current number of digitizes.
i_var(4004) = maximum number of digitizes to get.
i_var(4005) = the current keyboard macro set in use.
Output: none

9 When an invalid numeric expression is entered during get_fld_input ICL command.
Input: none
Output: none

101-166 When status item 1 through 66 is going to be updated.
Input: s_var(4001) = string that would have been used to update the status item.
Output: none

1001 get part file name
Input: none
Output:
if i_var(4001) = 1 then
s_var(4001) = file_name
i_var(4010) = term_char
term_char is the ASCII value that terminated file name input (usually 32 or 13).
if i_var(4009) = 1 then
there are wild characters (* or ?) in file name.
else
i_var(4009) = 0
if i_var(4001) = 0 then
get file name as usual from user.

1002 edit text
Input: none
Output: none

1003 TYPe command file name
Input: none
Output: See 1001

1004 not used

1005 RESTore TABlet file name
Input: none
Output: See 1001

1006 RESTore MENu file name
Input: none
Output: See 1001

1007 RESTore TFONT file name
Input: none
Output: See 1001

1008 GETDATA RUN (UPL) file name
Input: none
Output: See 1001

1009 SELect JOURnal file name
Input: none
Output: See 1001

1010 SAVe/RESTore bit map file name
Input: none
Output: See 1001

1011 VNP RUN (UPL) file name
Input: none
Output: See 1001

1012 RESTore PEN file name
Input: none
Output: See 1001

1013 RESTore HELP file name
Input: none
Output: See 1001

1014 RESTore MODifiers file name
Input: none
Output: See 1001

1015 SAVE MODifiers file name
Input: none
Output: See 1001

1016 DIR command file name seed
Input: none
Output: See 1001

1017 INSert DFILe file name
Input: none
Output: See 1001

1018 EXIT/(FILE?) file name to save part as
Input: none
Output: See 1001

1019 EXECute file name
Input: none
Output: See 1001

1020 UNLoad API program file name
Input: none
Output: See 1001

1021 LOAD API program file name
Input: none
Output: See 1001

1022 LOAD API program file name and command line
Input: none
Output: none

1023 LOAD API program command line
Input: none
Output: none

1024 BUIld MENu file name
Input: none
Output: See 1001

1025 SAVE CALibration file name
Input: none
Output: See 1001

1026 BUIld TFONT file name
Input: none
Output: See 1001

1027 CONSTruct PART file name
Input: none
Output: See 1001

1028 INS TFILE file name
Input: none
Output: See 1001

1029 INSert FIGure file name
Input: none
Output: See 1001

1030 DOS command string
Input: none
Output:
if i_var(4001) = 1 then
s_var(4001) = command_string
i_var(4010) = term_char
term_char is the ASCII value that terminated command string input (usually 32 or 13).
if i_var(4001) = 0 then
get command string as usual from user.

1031 UNIX shell command string
Input: none
Output: See 1030

1032 MENu COMMand string
Input: none
Output: See 1030

1033 UNIX dir command file seed
Input: none
Output: See 1001

1034 UNIX LS command string
Input: none
Output: See 1030

1035 UPL accept command for numbers
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = accept_num

1036 INSert PART file name
Input: none
Output: See 1001

1037 APPEnd file name
Input: none
Output: See 1001

1038 PUT SHADE file name
Input: none
Output: See 1001

1039 INSert TEXT text string
Input: i_var(4002) = insert text type
1 = INSert TEXt
3 = INSert TNOde
Output: if i_var(4001) = 1 then
i_var(4010) = term_char (normally 13 or 3).

1040 SAVe DISPlay display number
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = disp_no (1.0 to 32767.0)

1041 RESTore DISplay display number
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = disp_no (1.0 to 32767.0)

1042 DELete VIEW view number
Input: none
Output: if i_var(4001) = 1 then
i_var(4002) = view_no
i_var(4010) = term_char (normally 13 or 3).

1043 CHAnge LAYer/VVIs/COLor/FONt layer/vvis/color/font number
Input: i_var(4002) = change function
1 = change layer
2 = vvis
3 = line font
5 = color.
Output: if i_var(4001) = 1 then
i_var(4003) = new_att
i_var(4010) = term_char (normally 13 or 3).

1044 ZOOm UP/DOWn/SCL
Input: i_var(4002) = zoom function
1 = ZOOm UP
2 = ZOOm DOwn
3 = ZOOm SCL
Output: if i_var(4001) = 1 then
r_var(4001) = zom_num
if i_var(4002) = 1 then
new scale = current screen scale*zom_num
else if i_var(4002) = 2 then
new scale = current screen scale/zom_num
if i_var(4002) = 3 then
new scale = zom_num
i_var(4010) = term_char (normally 13 or 3).

1045 SELect CPL cpl number
Input: none
Output: if i_var(4001) = 1 then
i_var(4002) = cpl_number
i_var(4010) = term_char (normally 13 or 3).

1046 SET TRAP size
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = new_trap_size
(new_trap_size should be in range 0.0 to 1.0e10)
i_var(4010) = term_char (normally 13 or 3).

1047 SET CHT value
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = new_cht
(new_cht should be in range 0.0001 to 10.0)
i_var(4010) = term_char (normally 13 or 3).

1048 SELect LAYer number
Input: none
Output: Output: if i_var(4001) = 1 then
set_i_var(4004, lay_number)
(lay_number should be in range 1 to 256)
i_var(4010) = term_char (normally 13 or 3).

1049 SELect FONT number
Input: none
Output: Output: if i_var(4001) = 1 then
i_var(4004) = font_number
(font_number should be in range 1 to 16)
i_var(4010) = term_char (normally 13 or 3).

1050 SELect FONT SCL number
Input: none
Output: if set_i_var(4001, 1) then
r_var(4001) = new_soft_font_scl
(new_soft_font_scl should be in range 0.0001 to 10000.0)
i_var(4010) = term_char (normally 13 or 3).

1051 RESTore VIEW number
Input: none
Output: Output: if i_var(4001) = 1 then
i_var(4004) = view_number
(view_number should be in range 1 to 256)
i_var(4010) = term_char (normally 13 or 3).

1052 SELect ZDEPth value
Input: none
Output: if i_var(4001) = 1 then
r_var(4001) = new_zdepth
i_var(4010) = term_char (normally 13 or 3).

1053 RESTore AV number
Input: i_var(4003) = max AV number
i_var(4004) = current view number
Output: if i_var(4001) = 1 then
i_var(4005) = av_view_number
(av_view_number should be in range 1 to i_var(4004))
i_var(4010) = term_char (normally 13 or 3).

1054 Before exiting with no file name, i.e. before exit back to DOS.
Input: none
Output: none

1055 EDIT TABlet command
Input: s_var(4001) = old tablet menu command string
Output: if i_var(4001) = 1 then
s_var(4001) = new_def
(new_def is the new tablet menu command string)
i_var(4010) = term_char
3 = Abort command
other = Continue with command

1056 EDIT MEnu for existing menu command
Input: s_var(4001) = old menu command string
Output: if i_var(4001) = 1 then
s_var(4001) = new_def
(new_def is the new menu command string)
i_var(4010) = term_char
3 = Abort command
other = Continue with command

1057 EDIt MEnu for new menu command
Input: s_var(4001) = old menu command string
Output: if i_var(4001) = 1 then
s_var(4001) = new_def
(new_def is the new menu command string)
i_var(4010) = term_char
3 = Abort command
other = Continue with command

1058 EDIt STRing MACro command
Input: s_var(4001) = old string macro defintion (user already has entered
macro string word in the EDIt STRing MACro command)
Output: if i_var(4001) = 1 then
s_var(4001) = new_def
(new_def is the new string macro)
i_var(4010) = term_char
3 = Abort command
other = Continue with command

1059 When help system is invoked
Input: none
Output: none

1060 At replace existing view (Y/N)' question
Input: none
Output: if i_var(4001) = 1 then
i_var(4010) = ians
-1 = Abort command
0 = No
1 = Yes

1101 Invoked from EXIT command when the REPlace modifier is not selected and the file name given in the exit command exists.
Input: s_var(4001) = part file name
Output: i_var(4001) = 1 indicates ICL code handled error
i_var(4001) = 0 indicates handle error as if no ICL code was registered to this event.

1102 Invoked from the CONstruct PARt command when the file name given already exists.
Input: s_var(4001) file name of part that already exists
Output: i_var(4010) = -1 means abort construct part command.
i_var(4010) = 0 means DO NOT OVERWRITE this file.
i_var(4010) = 1 means DO OVERWRITE this file.

1103 Invoked at end of CHANge MV command.
Input: none
Output: none

1104 Invoked at end of DEFine MV command.
Input: none
Output: none

1105 Invoked at end of CHANge APPearance command.
Input: none
Output: none

1106 Invoked at end of PUT MV command.
Input: none
Output: none

1107 Invoked in SAVe DISplay command when the given display number <= 0.
Input: none
Output: if i_var(4001) = 1 then
the negative display number is allowed.
else if i_var(4001) = 0 then
display a warning message and reprompt user for a new display number.

1108 Invoked in SAVe DISPlay command when the given display number already exists in the part.

Input: none
Output: if i_var(4001) = 1 then
i_var(4010) = ians
-1 = Abort command.
0 = No, DO NOT replace.
1 = Yes, replace display.

1109 Invoked in DEFine VIEw command when the new view number is needed.
Input: none
Output: if i_var(4001) = 1 then
i_var(4010) = term_char (normally 13 or 3).
if term_char >= 13 then
i_var(4005) = view_no (for the newly defined view).

INDEX