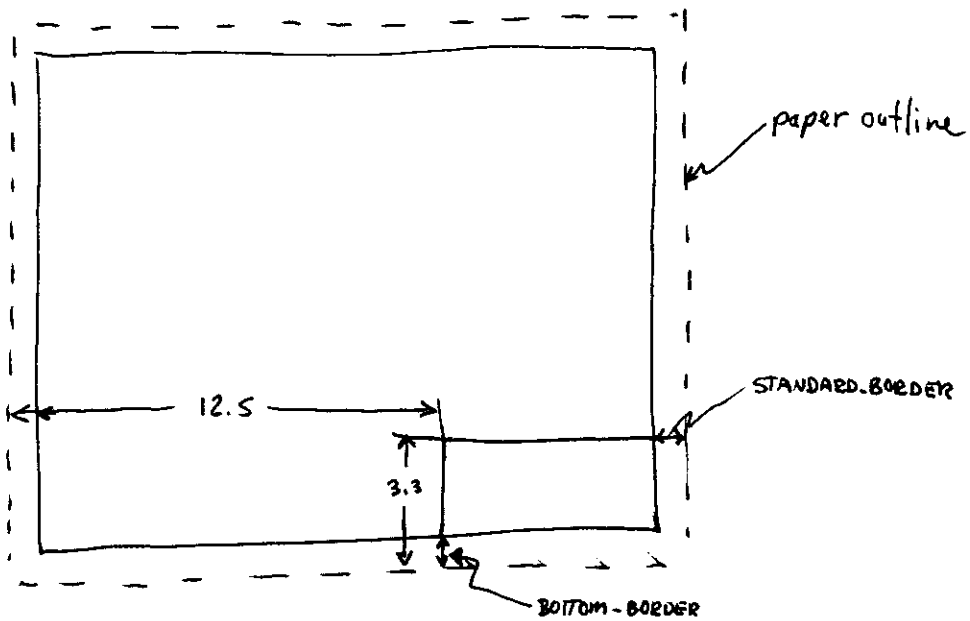
# DETAILED DESIGN DOCUMENT

## Interactive Powerflow

Vol. II

KeyPress Event

label box




Option BX 12.5 3.3 $\underbrace{\text{xx.x yy.y}}_{\text{option bottom right corner}}$

GraphClass VertexLabelCorner
" " " LabelText

also see: pscreateLabelBox (graphpscor.c)

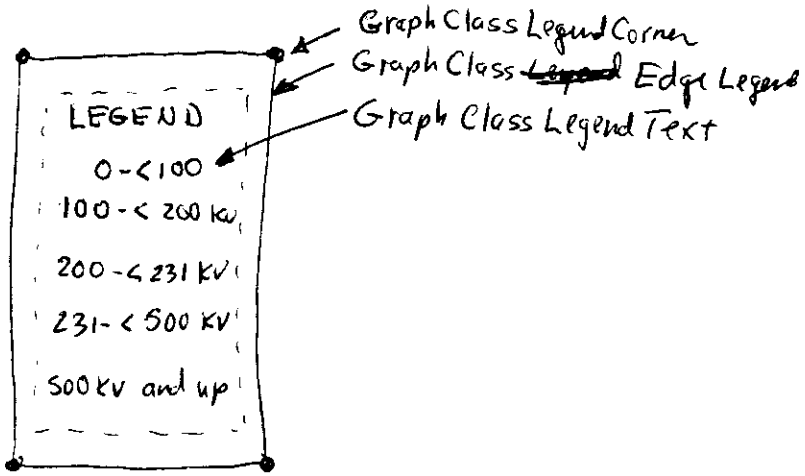
launch

term used to describe GUI when it starts up the powerFlow process.

see: ipf launch

legend

see: pscreateLegend (graphpscor.c)




in coordinate file

LG = XX.XX, YY.YY (in cm)

Letters/
Lettering

see: string vertexes
draw letters
draw character
character
edgeg.c - [illegible] lettering

library -

Not Covered By This Document!

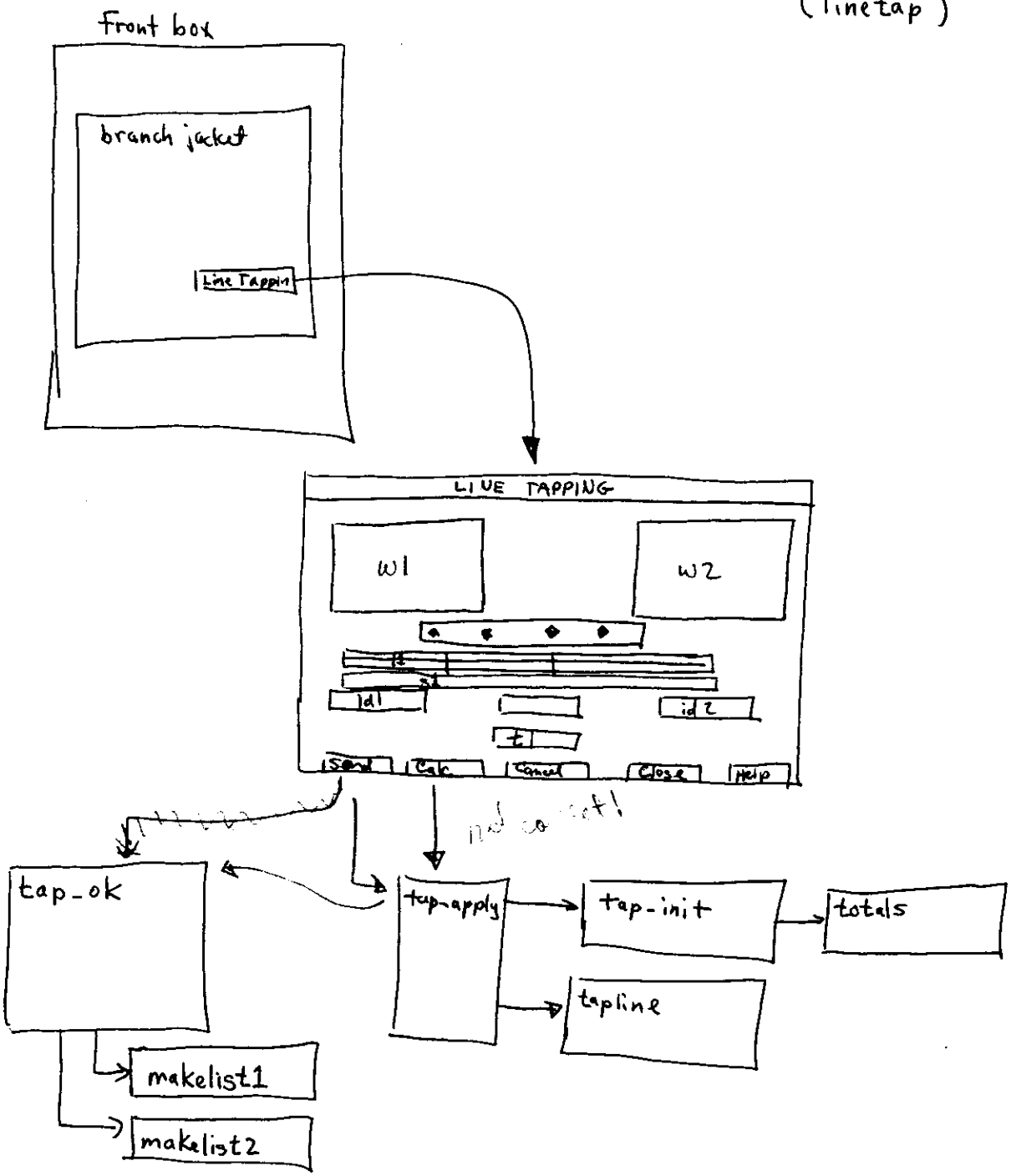
See "GUI Notes" Manual

line ?

line segment

also see: draw line segments

line tap
(linetap)



tap_apply

( calls tap_init
gets tunits (scale)
checks mid point bus name
calls tap line

tap_init

defines widget ids
checks that current bus name is available
puts curbus names in text widgets
clears w1, w2
asks pf for all lines (inbuf)
( adds lines to w1
builds tick marks.
calls totals
draws tick marks
sets utype

totals

gets tmiles, mi_array[], treact
zeros remainder of mi_array, react_array

(

tap_ok

calls tap_apply

makes list of pf lines to delete

send changes to pf

get tap name

create tap bus

create lines before tap

create lines after tap

f

LINE TAPPING

Bus 1 Line Data

L G4 HIGH 230 TAP

w1

Bus 2 Line Data

L TAP 230 INVT AC

w2

Percentage Miles Kilometers Section

n1 s1

50.0

G4 HIGH 230

Bus 1 Name

Reverse Scale

INVT AC 230

Bus 2 Name

Tapped Bus Name

t1 TAP 230

OK | Apply | Cancel | Close | Help

tap-ok

tap-apply

Ø

Ø

line tap

linetap

Command Entry

```
*[EOM]
gui->pf cnt: 7
/GET_DATA, TYPE=INPUT
L      G4 HIGH  230 INVT AC  23010 100  0027
gui->pf cnt: 8
/GET_DATA, TYPE=INPUT
TAP       230
```

(2)

wrong format - should be

```
/GET_DATA, TYPE=INPUT
B     TAP      230

/GET_DATA, TYPE=INPUT
 *** WARNING  Bus    230     0.0 on record
 *** WARNING   Adjacent bus names > G1
 *** WARNING   Adjacent bus names > G1X
/p_gtdata.f return status:  1 IPF state:  5
*[EOM]
```

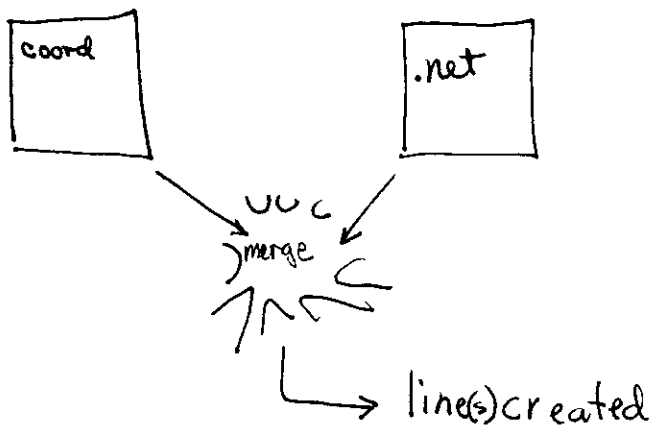
Close

lines (or tramission line branches) have their roots to the coordinate file. Lines can be defined with an "L" card, or implied by the powerflow.

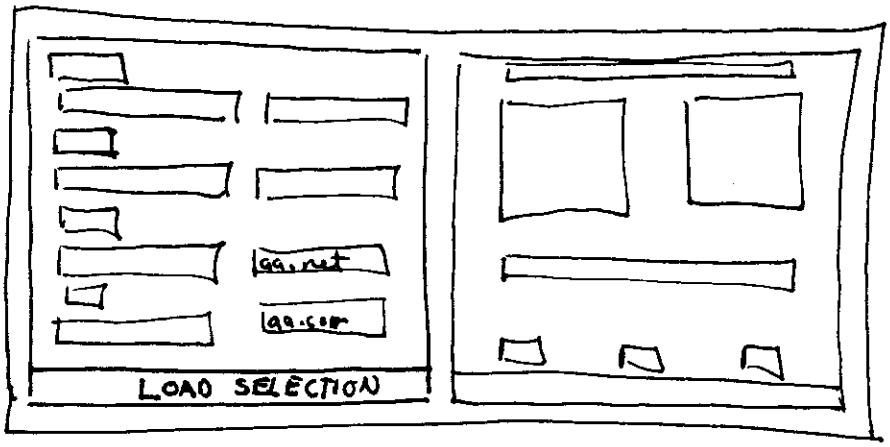
If the line is an "L" card in a coordinate file, then, the coord card shows the bus names at each end of the line and perhaps some "bend points!"

If the line is "implied", then the powerflow specifies that a STRAIGHT line is to be created between two buses. Both buses MUST be specified in the coordinate file. If one (or both) buses are not in the coordinate file, then the line CANNOT be drawn.

Creation of a line begins when the user names a coordinate file. More lines will be created when a network or base file is created



Step Two begins when the "Load Selection"




button is pushed. This button activates a MOTIF procedure (apply_file_selection) which gives a go-ahead for the C-language routine "apply_Files" to begin executing. "apply_Files" begin checking/~~and~~ loading the 5 possible input Files:

1) coordinate File
2) base File
3) network File
4) change File
5) command File

To make this description nice and easy, let's say that two Files are loaded - which is usually the case in 80% of the loads. For a more detailed description, see "coordinate file read" section of this manual. To make the long story short, the GRAPH_TABLE is created here, (creatigraphtbl & psbuildGraphCoord).

Now that the all-important ~~GRAPH DB~~ graph-db has been created and loaded, we continue in to details how lines will be expressed on the terminal screen on final hard copy output.

Rendering the final picture onto screen, now turns into a process of portraying everything as:

1) Vertexes
2) Edges

A vertex is a single plot coordinate where a pixel, symbol, name or other items are located. An edge will be thought of as a gadget with two of it's diagonally opposite corners denoted by vertexes.

A transmission line or transformer is represented as an edge in ~~[illegible]~~ pscor.db. There are <u>LINKS</u> which relate each line to buses and buses to lines. Also if the line has <u>bends</u>, then the line will have <u>sub-edges</u>. The <u>complex edge</u> is a straight line between two buses (and may not be drawn). The sub-edges are the smaller segments which make up the full transmission line.

Now that the vertexes and edges are in the pscor_db. They are not yet displayable on CRT screen. ~~These items must be t~~

The MOTIF gadgets must be created from this data. (Yes still another complicated step to perform.)

NOTE: If addition lines from powerflow are required, the mergeCoordBase routine is called to do this job. This process will not be discussed here.

graphCorOn (in graphdata.c) sets the "On" status to all records that came from the ~~coord~~ pscor_db vertexes or edges that are not "On" will not be shown on CRT screen.

createVertexGadgets (in vertex.c) searchs graph_db for vertexes that should be displayed.

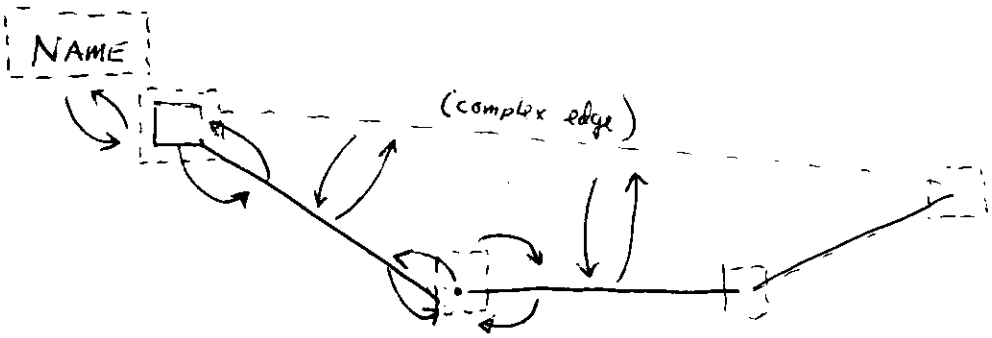
manageVertex (in vertex.c) is called next for each vertex

createVertexGadgetXY (in vertex.c)

# links

Graphical items are often <u>linked</u> to one another in various ways. For example bus symbols and bus name gadgets are linked together in such a fasion that moving a bus symbol, requires that the bus name label (or gadget) move with it.

Another such link is the bend-point, segment linkage - such that moving a bend-point will require that the two segments attached to this bend point be looked up.




Linking these items is done by calling the routine <u>addGraphLink</u>.

There is a fairly extensive system of maintaining these links. For futher information see:

replaceGraphLink
delete GraphLink
add GraphLink
} in graphdata.c

findEdgeVertex

replaceGraphLink

list
GraphElementT(s)

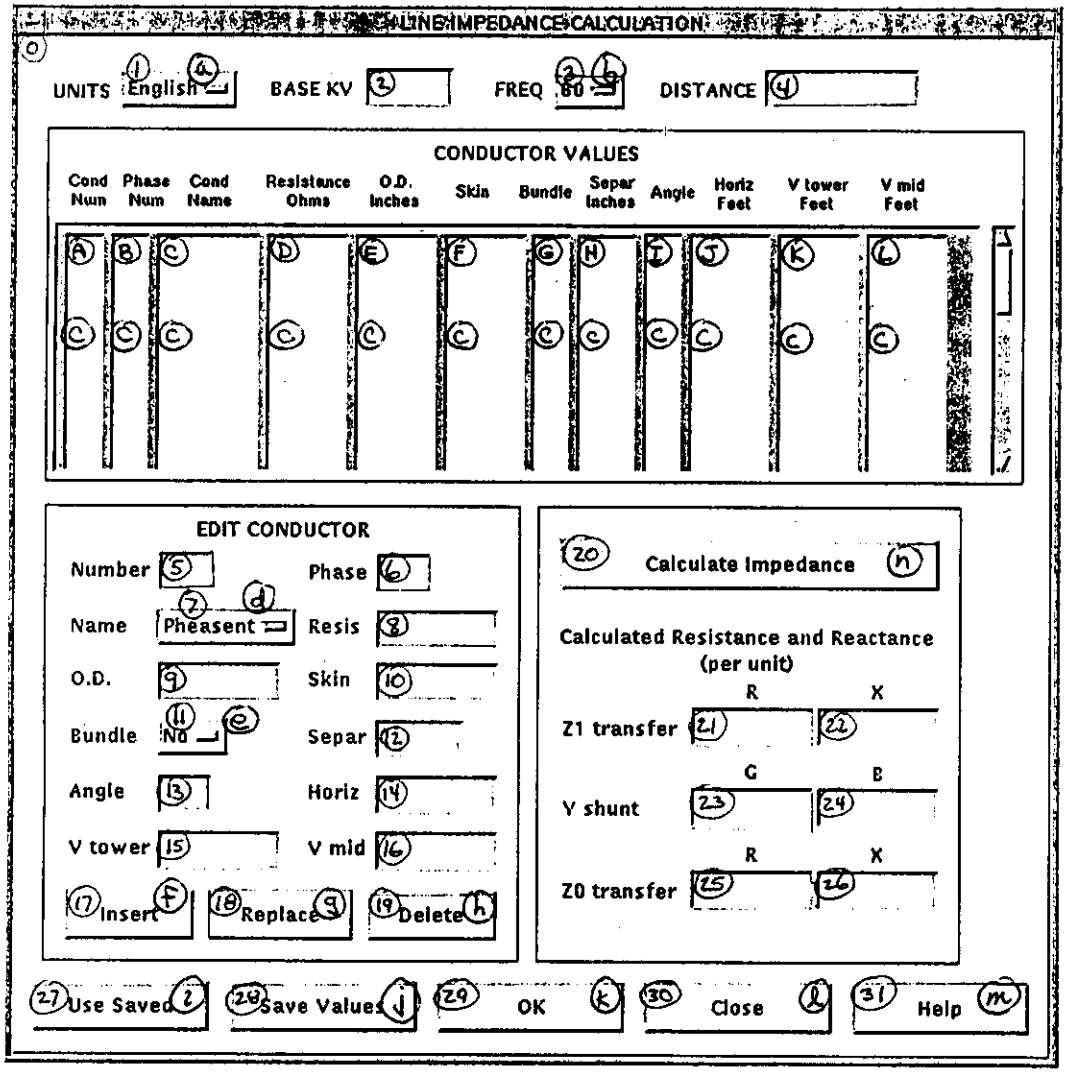
use: sprintfGraphElement (graphdata.c)
-or- printfGraphElement

true printGraphData (graphdata.c)
-or- printGraphData

desk copy

CALLBACKS: (pulldowns & pushbuttons)

a) line_z_units_change
b) line_z_freq_change
c) line_z_list_number_cb
d) set_std_cond_values
e) line_z_edit_bundle
f) line_z_edit_insert
g) line_z_edit_replace
h) line_z_edit_delete
i) manage_line_z_filesel
j) manage_line_z_save_dialog
k) line_z_ok_callback
   unmanage_line_z_dia
l) unmanage_line_z_dia
m) - none -
n) line_z_calc




WIDGETS:

0) line_z_dia_data_form

1) line_z_units_optmenu
2) line_z_basekv_text
3) line_z_freq_optmenu
4) line_z_distance_text

5) line_z_edit_number_text
6) line_z_edit_phase_text
7) line_z_wire_type_optmenu
8) line_z_edit_resist_text
9) line_z_edit_od_text
10) line_z_edit_skin_text
11) line_z_edit_bundle_optmenu
12) line_z_edit_separ_text
13) line_z_angle_text
14) line_z_edit_horiz_text
15) line_z_edit_vtower_text
16) line_z_edit_vmid_text
17) line_z_edit_insert_text
18) line_z_edit_replace_text
19) line_z_edit_delete_text

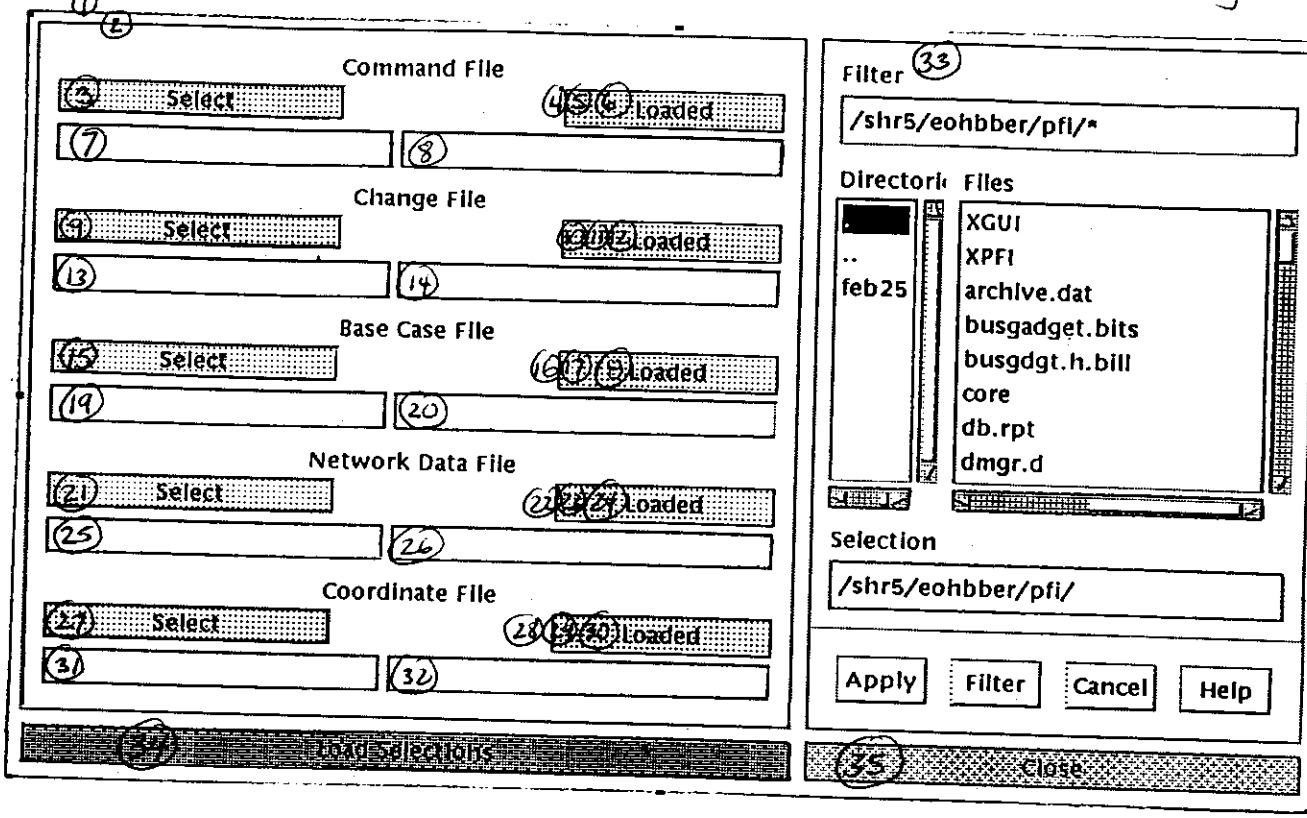
20) line_z_calc_z_pb
21) line_z_z1_R_text
22) line_z_z1_X_text
23) line_z_Y_G_text
24) line_z_Y_B_text
25) line_z_z0_R_text
26) line_z_z0_X_text

27) line_z_use_saved_pb
28) line_z_save_vals_pb
29) line_z_dia_ok_pb
30) line_z_dia_close_pb
31) line_z_dia_help_pb

A) line_z_number_list
B) line_z_phase_list
C) line_z_type_list
D) line_z_dia_resis_list
E) line_z_dia_diam_list
F) line_z_skin_list
G) line_z_bundle_list
H) line_z_separ_list
I) line_z_angle_list
J) line_z_horiz_list
K) line_z_vtower_list
L) line_z_vmid_list

~~pop~~ dialog
linezcalc

load dialog




Button 3, 9, 15, 21, 27 all call file_default_set

1 open_file_dialog
2 open_dia_base_form

3 open_dia_command_pb
4 open_dia_cmd_loaded_label
5 open_dia_cmd_ready_label
6 open_dia_cmd_noload_label
7 open_dia_command_dir_text
8 file_select_dia_command_text

9 open_dia_change_pb
10 open_dia_change_loaded_label
11 open_dia_change_ready_label
12 open_dia_change_noload_label
13 open_dia_change_dir_text
14 file_select_dia_change_text

15 open_dia_base_pb
16 open_dia_base_loaded_label
17 open_dia_base_ready_label
18 open_dia_base_noload_label
19 open_dia_base_dir_text
20 file_select_dia_base_text

21 open_dia_network_pb
22 open_dia_net_loaded_label
23 open_dia_net_ready_label
24 open_dia_net_noload_label
25 open_dia_network_dir_text
26 file_select_dia_network_text

27 open_dia_coord_pb
28 open_dia_coord_loaded_label
29 open_dia_coord_ready_label
30 open_dia_coord_noload_label
31 open_dia_coord_dir_text
32 file_select_dia_coord_text

33 file_selection_box_open

34 file_select_done_button
35 open_files_dia_cancel_pb

calls apply_Files

location

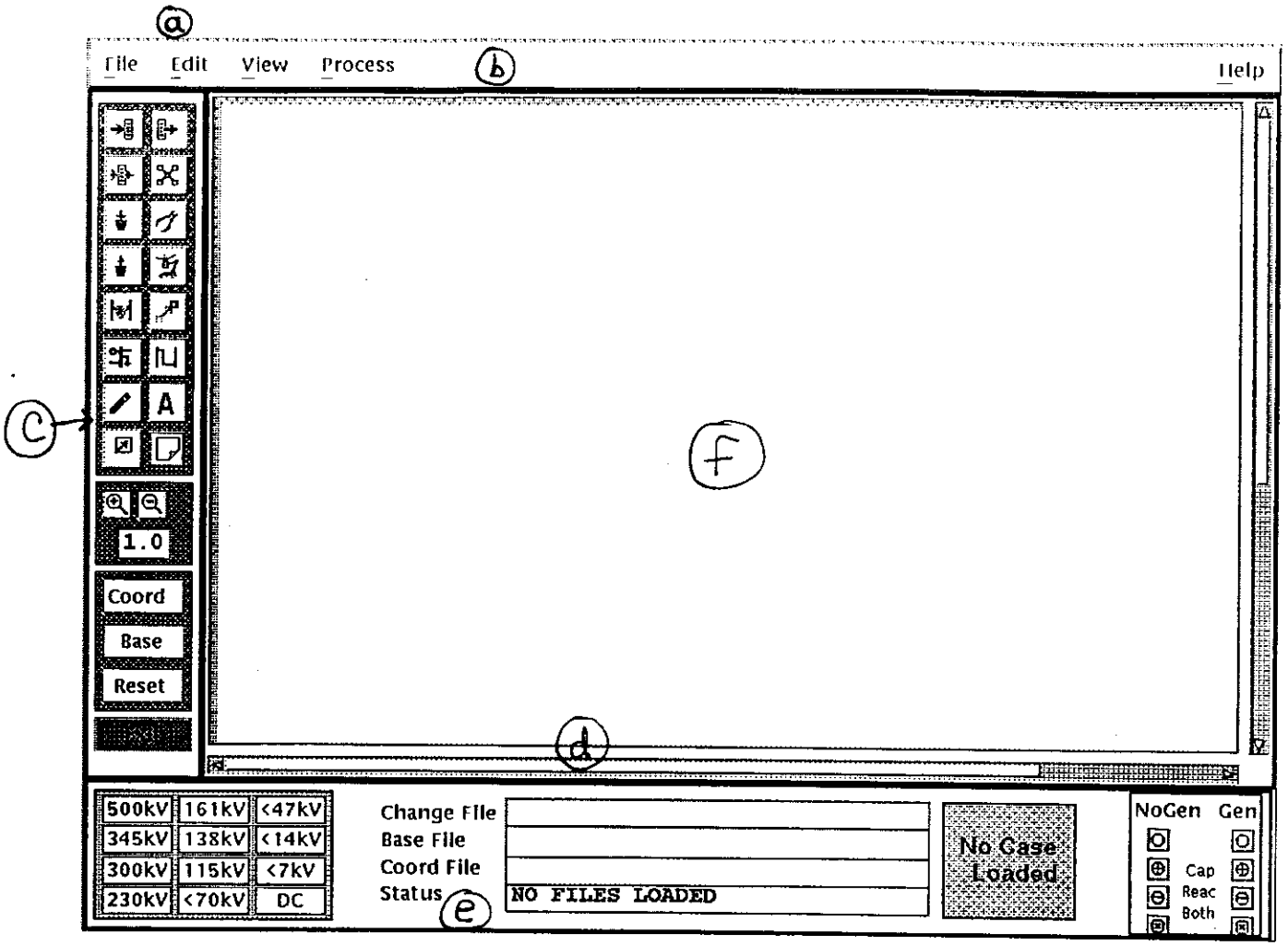
see: x - y location

loop

see: MainLoop

main

(gui_main)




All vuit data in main.u

a) gui_main
   help: manage_help_dialog

b) menuBar

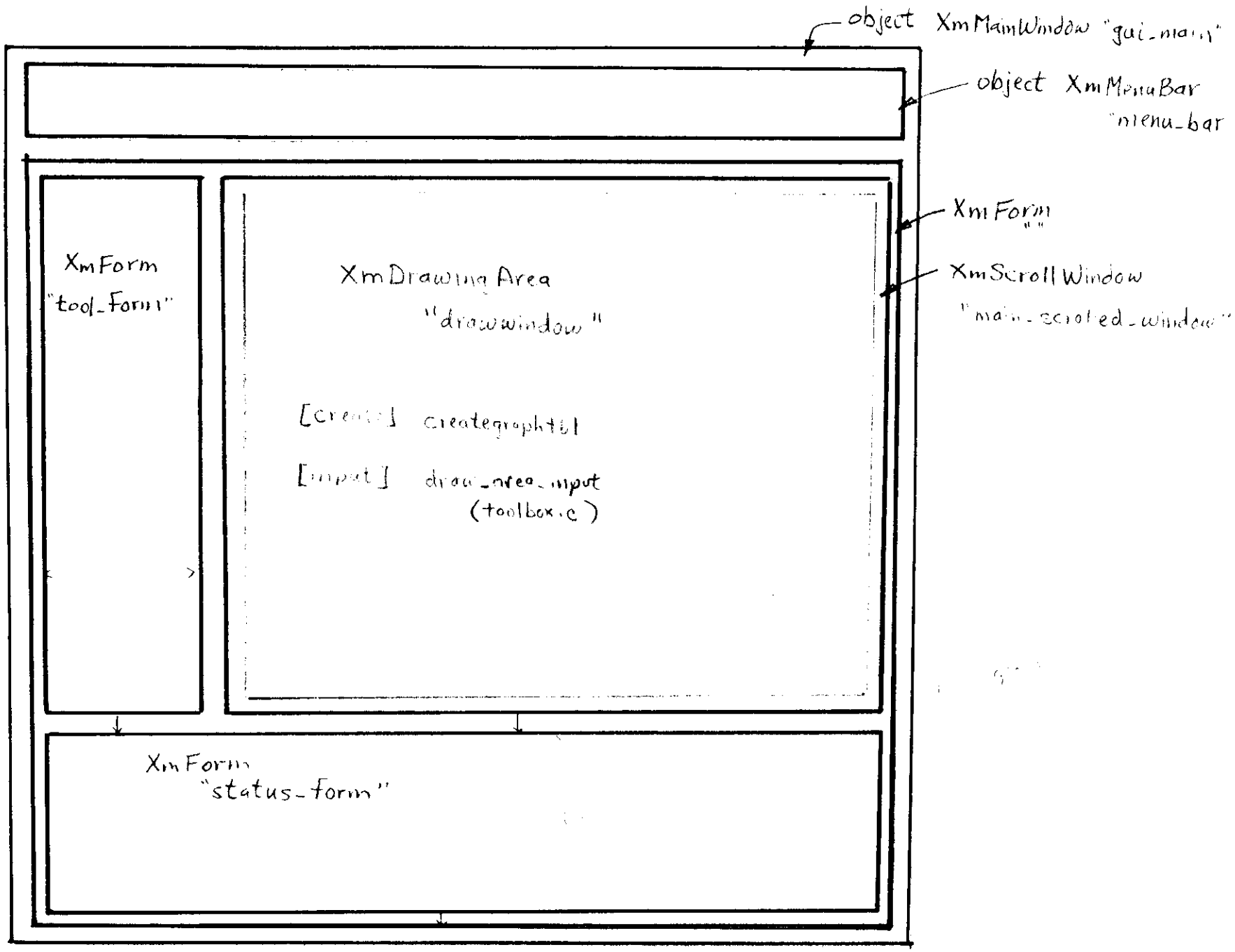
c) tool_form

d) main_scrolled_window

e) status_form

f) drawwindow
   Create: creategraphtbl
   input: draw_area_input

this list: /shr5/eohbber/rpt/main.lis


object XmMainWindow "gui_main"
object XmMenuBar "menu_bar"
XmForm
XmScrollWindow
"main_scrolled_window"
XmForm "tool_form"
XmDrawingArea
"drawwindow"
[create] creategraphtbl
[input] draw_area_input
(toolbox.c)
XmForm
"status_form"

MAIN.u

# MainLoop
# (XtAppMainLoop)

This does simply:

```
XEvent event,
while (1) {
        XtAppNextEvent ( app-context, &event )
        XtDispatchEvent ( event )
}
```

this is an infinite loop, gets all the events and takes action - used in gui.c

where app_context is:

```
XtAppContext  app-context;                          (in gui.h)
app-context = XtCreateApplicationContext();         (in gui.c)
```

see: XtAppMainLoop

# manageEdge

Source: edge.c

Purpose: manages the edge gadget.

Format: manageEdge (GraphElement *pedge)

# manageVertex

Given a graphElement, create a MCTIF gadget.

Bus → createBusGadget w/ [illegible]

Name → createStringGadget w/ formatted
adjust size

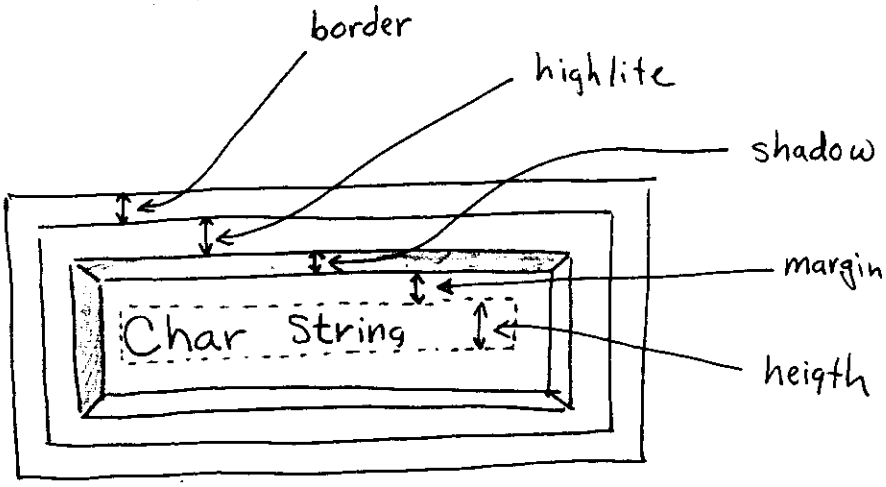
Group → createStringGadget w/ [illegible]

Bend
Draw
BorderCorner
PaperCorner
LegendCorner
LabelCorner
→ createPtGadget

Comment
SubComment
OrgComment
Font
LegendText
LabelText
→ createStringGadget w/ "Temp-str"

set graphvertex → wid
graphvertex → display = On
update

margin width




```
XtVaGetValues ( wid, XmNheight, & text_ht,
                     XmNborderWidth, & border_wd,
                     XmNshadowThickness, & shadow_wd,
                     XmNhighlightThickness, & hilite_wd,
                     NULL );
```

see: debug mask

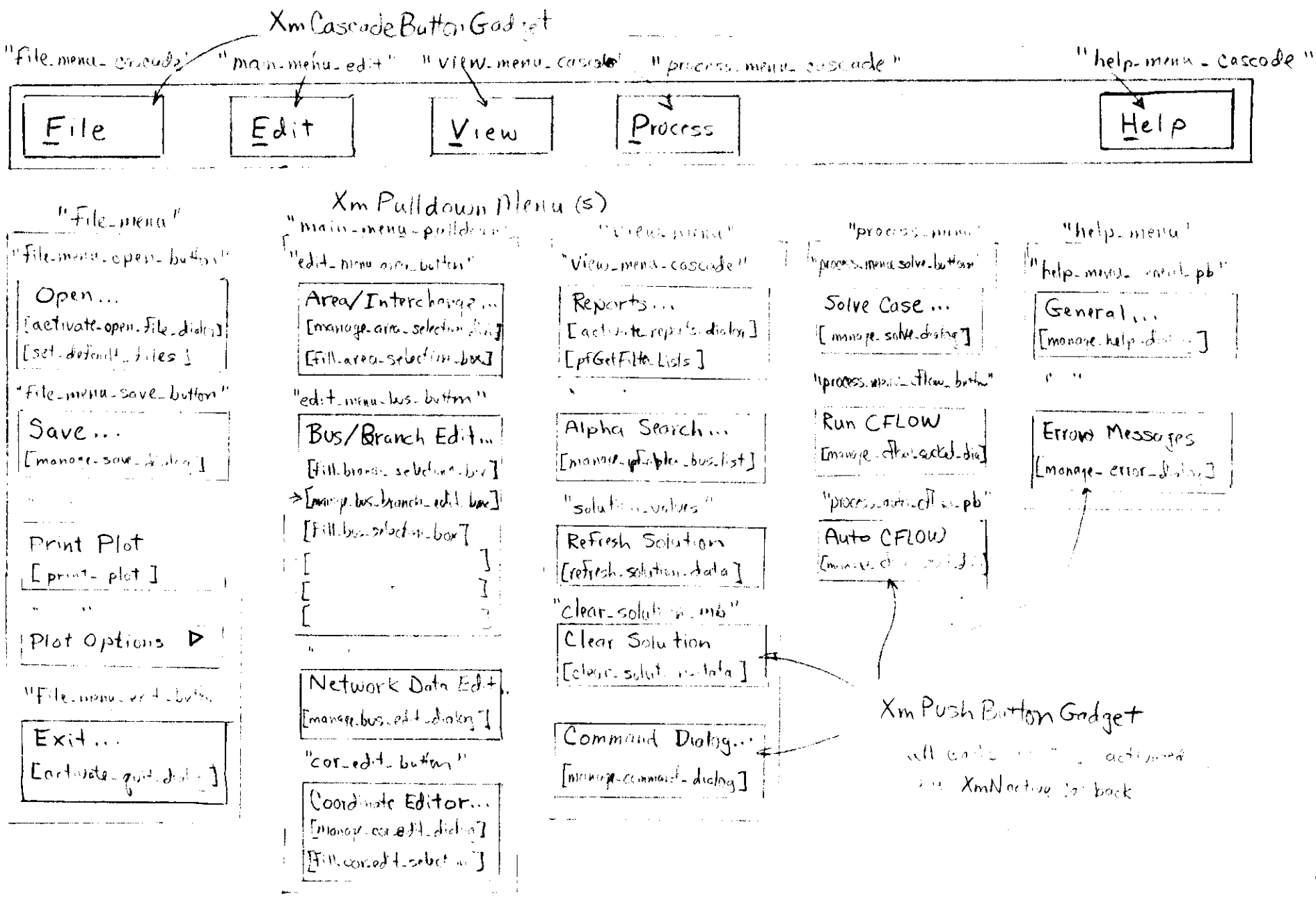
Max

See: psFindGraphMax (graphpscor.c)

scans coord file records - so that plot on screen can be shifted early.

```
memset ( &rec->cor.key[0], '\0', sizeof( ) )
```

Fills zeros in field.


XmCascadeButtonGadget
"file_menu_cascade"
"main_menu_edit"
"view_menu_cascade"
"process_menu_cascade"
"help_menu_cascade"
File
Edit
View
Process
Help
XmPulldownMenu(s)
"File_menu"
"file_menu_open_button"
Open...
[activate_open_file_dialog]
[set_default_files]
"file_menu_save_button"
Save...
[manage_save_dialog]
Print Plot
[print_plot]
Plot Options ▷
"File_menu_exit_button"
Exit...
[activate_quit_dialog]
"main_menu_pulldown"
"edit_menu_area_button"
Area/Interchange...
[manage_area_selection_dialog]
[Fill_area_selection_box]
"edit_menu_bus_button"
Bus/Branch Edit...
[Fill_branch_selection_box]
[manage_bus_branch_edit_box]
[Fill_bus_selection_box]
Network Data Edit...
[manage_bus_edit_dialog]
"cor_edit_button"
Coordinate Editor...
[manage_coredit_dialog]
[Fill_coredit_selection]
"view_menu"
"view_menu_cascade"
Reports...
[activate_reports_dialog]
[pfGetFilterLists]
Alpha Search...
[manage_pfTable_bus_list]
"solution_values"
Refresh Solution
[refresh_solution_data]
"clear_solution_mb"
Clear Solution
[clear_solution_data]
Command Dialog...
[manage_command_dialog]
"process_menu"
"process_menu_solve_button"
Solve Case...
[manage_solve_dialog]
"process_menu_cflow_button"
Run CFLOW
[manage_cflow_socket_dia]
"process_auto_cflow_pb"
Auto CFLOW
"help_menu"
"help_menu_general_pb"
General...
[manage_help_dialog]
Errors Messages
[manage_error_dialog]
XmPushButtonGadget
XmNactivateCallback
MENUBAR

merge

(base/coord file merge)

the process of matching coordinate bus names with powerflow bus names and putting them on the screen.

Done by mergeCoordBase (in graphdata.c)
which in turn calls merge_wp

Modules are disk files that have one or more C-language routines. All modules have names ending with ".c" Module names cannot be longer than 12 characters or problem with the merge process will occur!

ai_data.c
autostart.c
base_data.c
Branch.c
busfilter.c
bussect.c
chgdata.c
chgdatatst.c
chkentry.c
cmdprsgcb.c
cmdprsgcu.c
colorx.c
convert.c
~~coord_data.c~~ obsolete
crtLnLst.c
curbus.c
darrowxya.c disable_h.c
dmgr.c
dmgrdoc.c
edge.c
edges.c
~~edge_research.c~~
em.c
errdlg.c
execsrv.c
filedlgrtn.c

~~filetest.c~~
fixvuit.c
fm.c
fmtpg.c
getrpts.c
goipf.c
~~graphbase.c~~ gotime.c
~~graphcor.c~~ obsolete
graphdata.c
graphpscor.c
gui.c
help_cb.c
initwin.c
ipc_cb.c
ipcclient.c
ipcclntshl.c
ipcsrvstbs.c
ipf_rsrc.c
is_float.c
isvalidp.c
itoa.c
line2rect.c launch_srv.c
linetap.c
linetap2.c
linezcalc.c
[illegible]

openfiles.c
pf_cb.c pf_descript.c
pqcurve.c
printopts.c
pscordat.c
random.c
reformat.c
selection.c
shift_str.c
slctbus.c
slctdmo.c
stdlib_ext.c
str_util.c
stringpart.c
substation.c [illegible].c
toolbox.c
uscanc.c [illegible]
utils.c
vertex.c

Shown modules as of ~~Dec 1993~~ MAR [illegible]

see: get values
passing params

header: #include <Xm.h>

MOTIF

Following steps to build simple stand-alone MOTIF application from scratch:

1) vuit

2) select XmMainWindow from Windows parts box, place in work area.

3) name - say... "new_program"

4) select XmMenuBar from Menus parts box, put in above window.

select XmCascadeButtonGadget from Menus parts box, put in menu bar.

select XmPulldownMenu from Menus box, put in CascadeButtonGadget

select XmPushButton from Menus parts box, put in PulldownMenu.

add XmNactiveCallback to the pushbutton.

Click on Callback Editor to bring up another dialog.

name an exit procedure, say.... "exit"

Set this procedure to Exit Application

5) File - Save

6) Activities - Generate Application Files - in C

(Generate Build Procedure button should be "down" on 1st run.)

OK ( file named "makefile" will be generated )

7) in vi, make following changes:

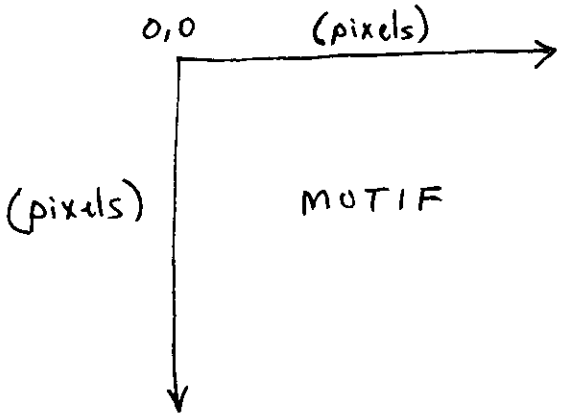
| OLD | NEW |
|---|---|
| TOP=/usr/lib/DXM | TOP=/usr |
| UIL=$(TOP)/clients/uil/uil | UIL=uil |
| -I$(TOP)/lib \ | -I$(TOP)/include \ |
| -I$(TOP)/lib/Xt \ | -I$(TOP)/include/Xt \ |
| | -I$(TOP)/include/Xm \ |

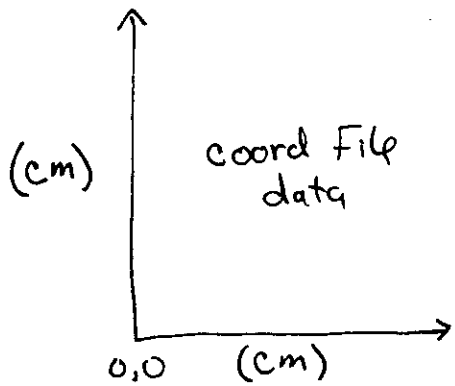
8) make -f makefile

"new_program.uid" will be generated - THIS FILE SHOULD BE REMOVED AFTER EACH VUIT CHANGE AND BEFORE THE MAKE.

9) (notice the new file "new_program")

type "new_program"




Conversion between MOTIF coordinates (for screen only) to cm coordinates (for coord & ps) should be handle with great care. ALWAYS use the following routines for this conversion:

pscoord_to_MOTIF_y
pscoord_to_MOTIF_x
MOTIF_to_ps_y
MOTIF_to_ps_x

# MOTIF STRINGS

**See: Compound Strings**

mouse actions

most buttons and "bus gadgets" have pushbutton callbacks i.e. XmNactivateCallback

bus gadgets on [illegible] by ButtonPress
ButtonRelease

see: [illegible]
draw [illegible] CB (+ [illegible])
cursor location

See: update

move_object_xy

Source: graphdata.c

purpose: updates db & moves objects

format: move_object_xy ( GraphElement ,
int x,
int y )

also see: move_object

Source: graphdata.c

purpose: higher level move of graph element

format: move_object (GraphElement)

"multiply defined" error

Occurs during MAKE

* checked that suspect module (if procedure)

  User Procedures should have

  ☐ Generate in Application

  ↑
  UP

* something MAKE will look for missing modules in older versions of library

  Make sure module is not missing from ~~latest~~ most recent library version

name vertexes

See: bus vertexes

source: graphdata.c

purpose: makes clean code

format: nearend ( GraplElement *vertex,
GraphElement * edge )

also see: bus_edit_dialog
(in buseditn.u)
(in busfilter.c)

this is not same as <u>bus_front_box</u>

A network file is same as a base powerflow file only it is binary and faster to load.

NEW
Coord File

new coord files can be created by selecting "NEW" under "File" pull down. This calls <u>create_from_scratch</u> (filedlgrtn.c)

null char

```
if ( a[0] == '\0' )
```

obsolete

obsolete modules — new

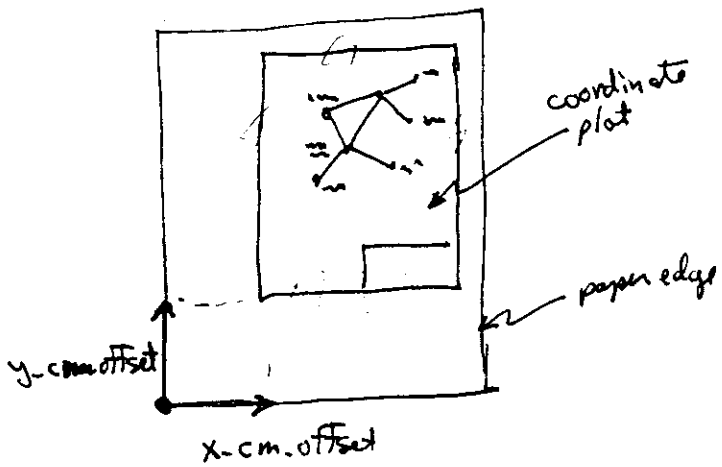
graph_base.c →

coord_data.c → pscoordat.c

graph_cor.c → graphpscor.c

[illegible] is obsolete [illegible]

means of adjusting location of plot data from a .cor file.




offsets <u>ARE</u> scaled!

graphOffset ~~is~~ <u>NOT</u> scaled!

see: graphOffset
border
scale
refference frame
[illegible]
[illegible] OFFset
[illegible]
[illegible]

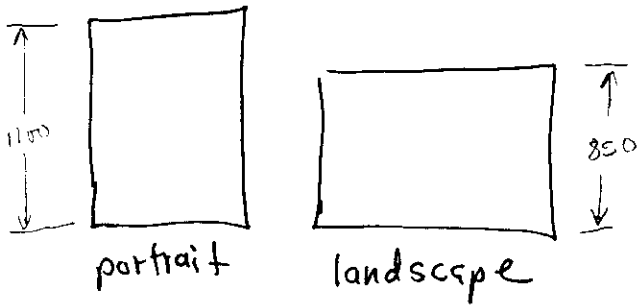
See: page options dialog
plot options dialog
★ <u>coordinate file</u> options

read by: read_print_options
decoded by: process_optrecord
} printopts.c

write_out_option_cards (~~pscordat.c~~) (printopts.c)

also see: command line options
coordinate options

# orientation




~~see: portrait_mode in printopts.c~~

data comes from reading the radio button position of "print_portrait_rb" in get_paper_size

also see: paper_height
Resource Frame

# outage

wire Clippers denote bus or line is to be removed. Changes are sent to powerflow.

toolbox.c has "tbOutage" which processes this outage.

lines ----> pf_get_lines

buses -----> sendOutageCard

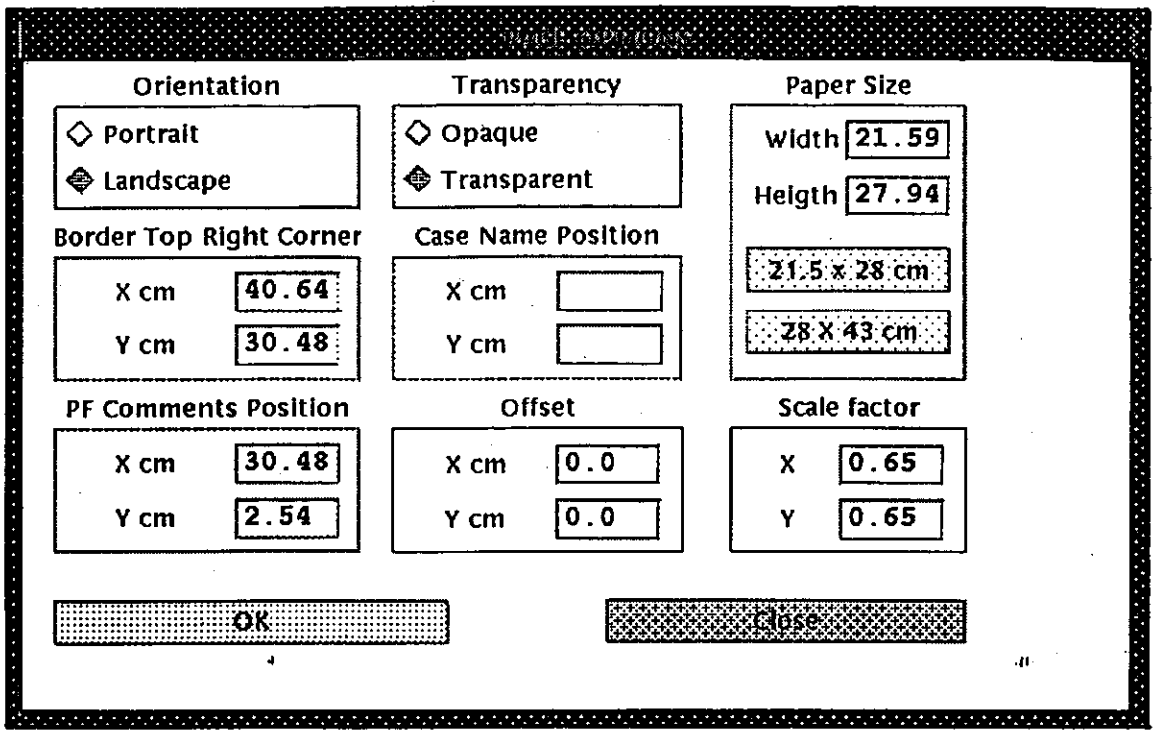
# Overview

When the program starts up — the **gui** executable begins to operate. The operation can best be understood by examining the **gui.c** file. It's the first program which leads to all other routines.

The gui program is a collection of files required as a group to execute.

| | |
|---|---|
| gui | executable |
| XGUI | resource file |
| *.c | C-Language source modules |
| *.u | (MOTIF modules) |
| gui.uil | |
| gui.uid | |
| gui.m | make file (make -f gui.m) |
| templates | |

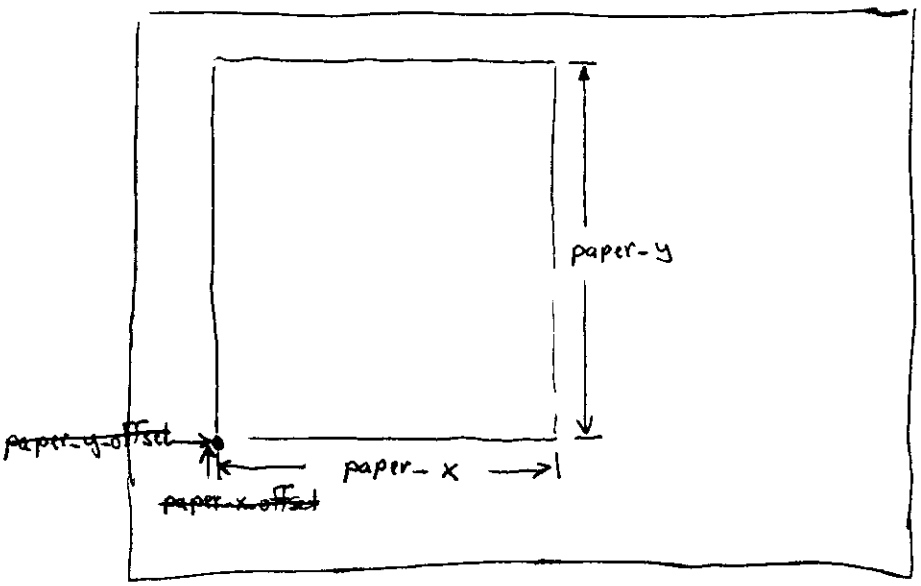
page options dialog




```
 A) print_opt_page_dialog

 1) print_portrait_rb            process_prtopt_rb
 2) print_landscape_rb
 3) print_border_xpos_text
 4) print_border_ypos_text
 5) print_cmnt_xpos_text
 6) print_cmnt_ypos_text

 7) print_opaque_rb
 8) print_transparent_rb
 9) print_case_xpos_text
10) print_case_ypos_text
11) print_x_offset_text
12) print_y_offset_text

13) print_size_x_text
14) print_size_y_text
15) print_standard_text         set_standard_paper_size
16)     (none)                  set_double_paper_size
17) print_x_scale_text
18) print_y_scale_text

19) print_page_OK_pb
20) print_page_close_pb
```

pgopt.lst

# Graph Class Paper-Vertex (9)




paper-x-offset
paper-y-offset provides a means of automatically shifting paper AND graph data together.

papr edge

Papr edges

paper height

important to know the paper height so that coordinate conversion will work correctly

[illegible] paper height [illegible] postscript [illegible]

see: orientation

get_paper_size (g [illegible])

set_standard_paper [illegible] (print piece)

" device "

params

(on VAX)

run gui (can't use params!)

gui -fg white

use set sym == $DISK1:[PSAP]GUI]GUI.EXE

see: command line arguments

passing params
(MOTIF - to - C)

```
char subtype
void routine_name( Widget w, XtPointer tag, XmAnyCallbackStruct *cbs)
{
    char *pctag;
    pctag = (char *) tag;
    sub_type = pctag [0];          /* gets a character */
}
```

also see: command line arguements
pointers (param problems)

params

passing **array** values

See example

in createBendArray (graphdata.c)

see: getPerimeterPts (GraphElement, GraphElement, x, y x2, y2)
(edge.c)

squarePerimeter
circlePerimeter

platform

see: ALPAA VMS
VAX VMS } in "gui notes" book

See: print plot
post script

plot options dialog

**Bus Detail**

**Bus Name**
- ◆ Abbreviation
- ◇ Full Name & KV

**Bus Voltage**
- ◆ kV
- ◇ Per Unit

**Generation**
- ◆ Draw Generation as Needed
- ◇ Always Draw Generation
- ◇ Draw No Generation

**Shunt**
- ◆ Draw Shunt as Needed
- ◇ Always Draw Shunt
- ◇ Draw No Shunt

- ▤ Angle
- ☐ Load
- ☐ Cut Branches
- ▤ Outages

**Branch Detail**

**Parallels**
- ◆ Combined
- ◇ Separate

- ☐ Trans_Tap
- ☐ Compensation
- ▤ Outages

**Values**
- ◆ Normal
- ◇ Difference

**Diagram Type**
- ◆ PQ Flow
- ◇ MVA & I
- ◇ Loss
- ◇ Interchange
- ◇ Coordinates

**Flow Detail**
- ▤ P Sending End
- ▤ Q Sending End
- ☐ P Recieving End
- ☐ Q Receiving End

OK  Close

```
 1) print_busname_abbrv_rb
 2) print_busname_full_rb
 3) print_volts_kv_rb
 4) print_volts_pu_rb
 5) print_gen_rb
 6) print_al_gen_rb
 7) print_no_gen_rb
 8) print_shunt_rb
 9) print_al_shunt_rb
10) print_no_shunt_rb
11) print_angle_tb
12) print_load_tb
13) print_cut_tb
14) print_branch_outage_tb

15) print_lines_comb_rb
16) print_lines_sep_rb
17) print_x_taps_tb
18) print_comp_tb
19) print_branch_outage_tb

20) print_values_norm_rb
21) print_values_diff_rb

22) print_pq_flow_rb
23) print_mvai_rb
24) print_loss_rb
25) print_diff_rb
26) print_inter_rb
27) print_coord_rb

28) print_p_send_tb
29) print_q_send_tb
30) print_p_receive_tb
31) print_q_receive_tb

32) print_options_ok_pb
33) print_options_close_pb
```

source: plot_detail.rpt

pointers
(param problems)

```
main
{ GraphElement *ptr
    routine1 ( & ptr )

}


routine1 ( GraphElement **pptr )
{
  GraphElement *ptr

    routine2 ( & pptr );
  *pptr = ptr;
}


routine2 ( GraphElement **pptr )

{ GraphElement *ptr
    dbcreate ( & ptr );
} *pptr = ptr;
```

powerFlow details are not covered in this manual. GUI interfaces to the IPF via the inter-process-communication (ipc)

see: send command to pF

see: ipf - launch

also see:

PQ curves



# 3.29 REACTIVE CAPABILITY CURVES (QP, QX, QN)

Three records are required to define a curve: QP, QX, and QN. They may appear anywhere in the input stream although they normally are put immediately after the bus record to which the curve applies. Each curve applies only to the bus named, and there is a limit of 300 curves.

## Description

The generator capability curve model is a piece-wise linear representation of a synchronous machine capability curve. As shown in the fiugre, the generator capability curve model consists of a series of points on the P-Q diagram. Each point is defined by specifying a value for P followed by values for Qmax and Qmin.

If the minimum absolute value for P is less than the first entered value (P1), then the model will set the values for Qmax and Qmin equal to Qmax1 and Qmin1. For any point ABS (Pgen) between P1 and Pmax, the model will linearly interpolate between the Q values for Pj just greater than and Pj-1 just less than ABS(Pgen). Pgen greater than Pmax generates a fatal data error.

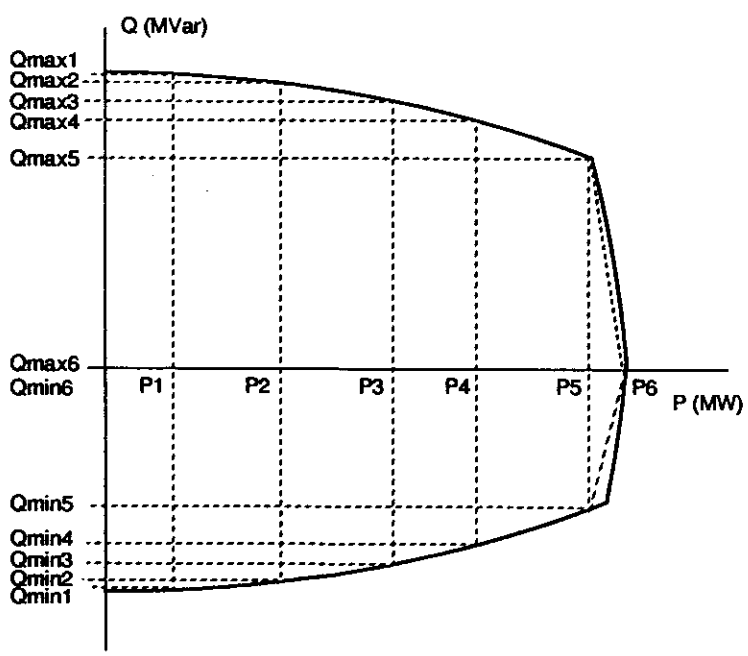



**Figure 3-44. Generator Capability Curve Model**

## Processing

Before solution of the case, each BE, BG, BQ, BX, and BS bus is checked to see if a curve is to be used to set its Q limits. If not, the Qmin and Qmax already stored is used, that is, those read from the bus record or calculated from a prior solution. If a curve is active, the values calculated using it replace those formerly stored. Original input values from the bus record are not saved.

PQ curves



If the bus is an area slack machine with an active curve, its Q linits are recalculated at any point during the solution when its Pgen changes mroe than 1 MW. If Pgen exceeds the curve limits, a fatal error will result.

**Table 3-30. Column Description for Reactive Capability CurvesM**

| Column | ID Field | Format | Description |
|---|---|---|---|
| 1-2 | yes | A2 | Record Code —<br>QP for Pgen values (positive values only)<br>QX for Qmax values (positive values)<br>QN for Qmin values (negative values) |
| 3 | no | A1 | Change code — For QP record only:<br>D = Delete curve for this bus.<br>F = Fix Q limits (retain curve, but do not use it).<br>C = Use curves that was previously deactivated. |
| 4-5 | no | A2 | Per unit code:<br>Blank = values are in MW/MVAR<br>PU = values are in per unit on bus Pmax |
| 6 | no | A1 | Activity flag; ' ' = active; '*' = inactive |
| 7-14 | yes | A8 | Bus name |
| 15-18 | yes | F4.0 | Base kV |
| 19-78 | no | 10F6.0 | Up to ten values for P, Qmax, or Qmin depending on the card type. The values for P can be in any order, but the related Q values must correspond. Entries must be in consectutive fields with no blank entries between. |

PQ Curves



PU activity

| C | SUBTYPE | CHGCODE | ACTIVITY | NAME | BASE KV | PGEN VALUES 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 6 7 8 9 0 1 2 3 4 5 6 7 8 | 9 0 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 | 6 7 8 9 0 | 1 2 3 4 5 | 6 7 8 9 0 |
| Q | P | | | | | | | | | | | | | | | |
| Q | P | | | | | | | | | | | | | | | |
| Q | P | | | | | | | | | | | | | | | |

= Required in most cases; blanks or zeros are unacceptable; exceptions are noted.

= Optional; blanks or zeros are acceptable.

= Not Applicable.

**Figure 3-45. Reactive Capability Curve QP Record**

PQ Curves

```
( POWERFLOW, CASEID = TESTQP, PROJECT = TEST-BENCHMARKS )
/ P_INPUTLIST,FULL \
/ P_OUTPUTLIST,FULL \
/ P_ANALYSIS_RPT,LEVEL=4 \
/ COMMENTS \
C TESTQP - BASIC NINE-BUS CASE with QP curve data
/ NEW_BASE,FILE = TESTPQ.BSE \
B      GEN1     16.5 2                    240306.2150.0 -1001040
B      GEN1 HI   230 2
BQ     GEN2     18.0 1                    180163.0120.0-80.01025
.      NNNNNNNNVVVVPPPPP.PPPPP.PPPPP.PPPPP.PPPPP.
QP     GEN2     18.0 165.0 180.0 150.0
QN     GEN2     18.0 -80.0  -5.0 -90.0
QX     GEN2     18.0 120.0   5.0 130.0
B      GEN2 HI   230 1230.0
BQ     GEN3     13.8 2                    13085.0080.00-60.01025
.      NNNNNNNNVVVVPPPPP.PPPPP.PPPPP.PPPPP.PPPPP.
.    MISSING P VALUES - (DEACTIVATED)
.QP     GEN3     13.8
.QN     GEN3     13.8 -80.0   0.0 -90.0
.QX     GEN3     13.8 120.0   0.0 130.0
B      GEN3 HI   230 2
B      STA A     230 1125.050.00
B      STA B     230 290.0030.00
B      STA C     230 2100.035.00
T      GEN1 HI   230 GEN1    16.5                05760            23000 1650
L      GEN1 HI   2302STA A    230        01000 08500       08800
L      GEN1 HI   230 STA B    230        01700 09200       07900
T      GEN2 HI   230 GEN2    18.0                06250            23000 1800
L      GEN2 HI   230 STA A    230        03200 16100       15300
T      GEN3 HI   230 GEN3    13.8                05860            23000 1380
L      GEN3 HI   230 STA B    230        03900 17000       17900
L      GEN3 HI   230 STA C    230        01190 10080       10450
(      NNNNNNNNVVVVPPPPP.PPPPP.PPPPP.PPPPP.PPPPP.
.   NON_EXSITENT BUS (DEACTIVATED)
.QP     GEN4     18.0 165.0 180.0 150.0
.QN     GEN4     18.0 -80.0   0.0 -90.0
.QX     GEN4     18.0 120.0   0.0 130.0
( END )
```

sets of values, always stay together, pgm sets

P-Q test data

pq curves

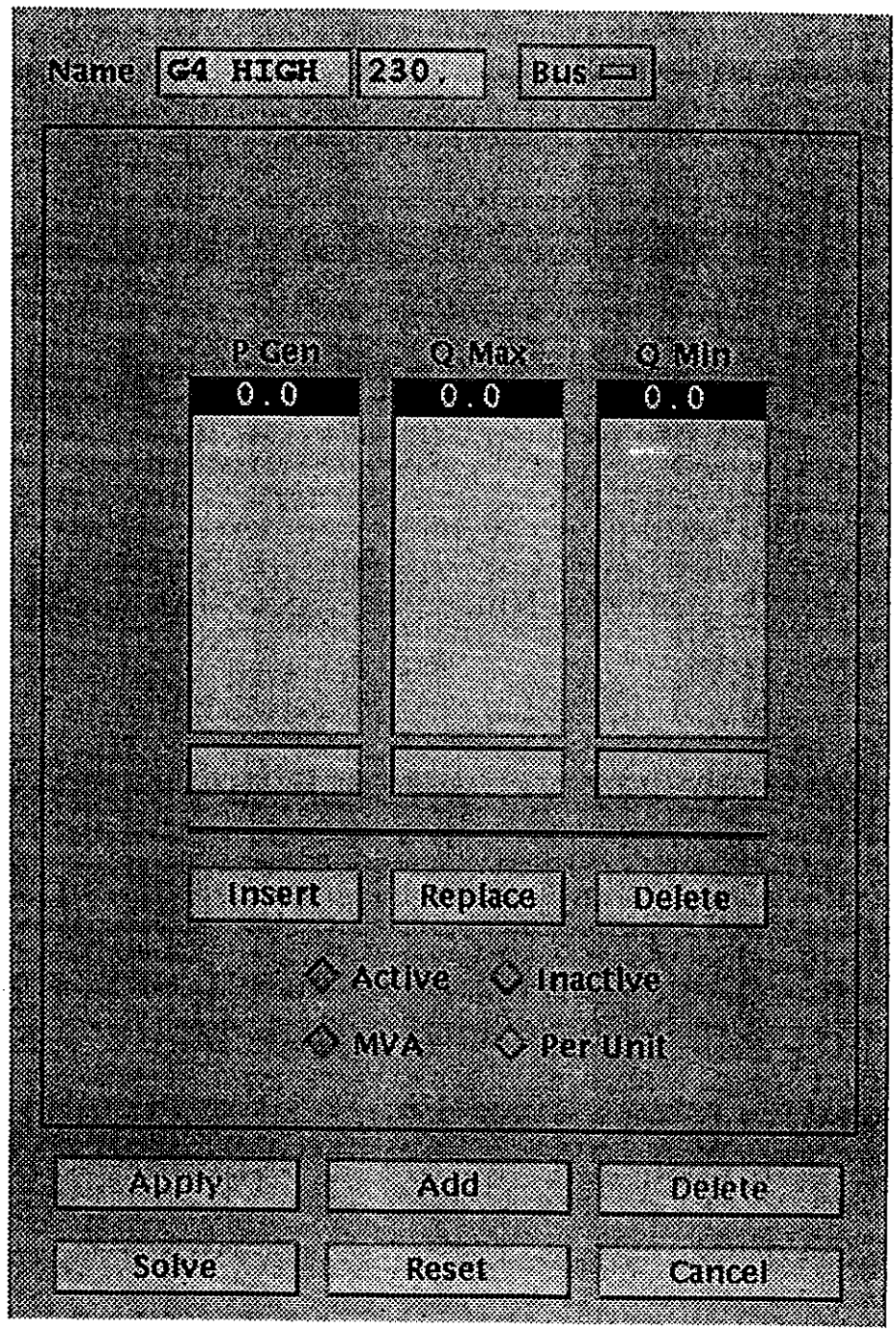
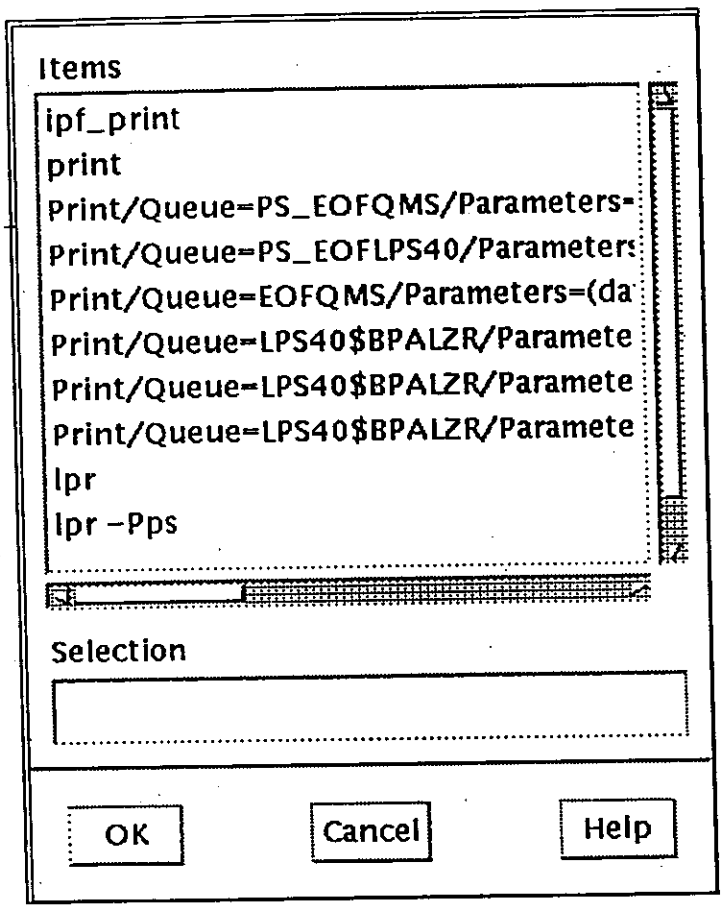



**Figure 4-15. P-Q Generation Dialog Box**

also see: print PS Bus Record
print Graph Element
print Graph Data
print PS Branch Record
~~[illegible]~~
true print Graph Data (all Graph Elements)

printer dialog



printopts.c

also see: page options dialog
write_out_options_cards. (pscordat.c)
printopts.u

The pull down "Print Plot" or comment dialog "Plot now" calls routine "print_plot" (in pf_cb.c).

2 Files are created:

aa.tmp (by psWriteCoordFile)
aa.ps

A printer port is looked up.

2 commands are then sent to the powerflow

```
/PLOT
 aa.tmp
 aa.ps
 Comments        (From comments text)
 *EOM

/SYSCAL
 ~~PRINT~~ LPR
   aa.ps
 *EOM
```

(that's it!)

printGraphData

source: graphdata.c

purpose: quick debug listing

format: printGraphElement ( GraphElement* ptr )

printAllGraphData ( 0, 0, 0 ) (graphdata.c)

# printopts routine

This routine called by filedlgrtn.c finds all the <u>Options</u> ('O') records, ignoring buses and branches.

Key routine is **read_print_options**, where db is accessed and widgets are set. Only records beginning with "O" are process. Bus, Branch, Coord cards are ignored.

Source: graphdata.c

purpose: prints to File in PS Format

a routine in graphdata.c
purpose: prints bus to file in PS format
format:

```
printPSBusRecord ( FILE *pcorFile,
                   GraphElement *pv )
```

where: pcorfile is file to write to
GraphElement is a record (line, bus, bend, etc)
that exists in graph_db

procedure

a terminology of ~~WOIT~~ MOTIF

all users' screen actions such as pushbuttons

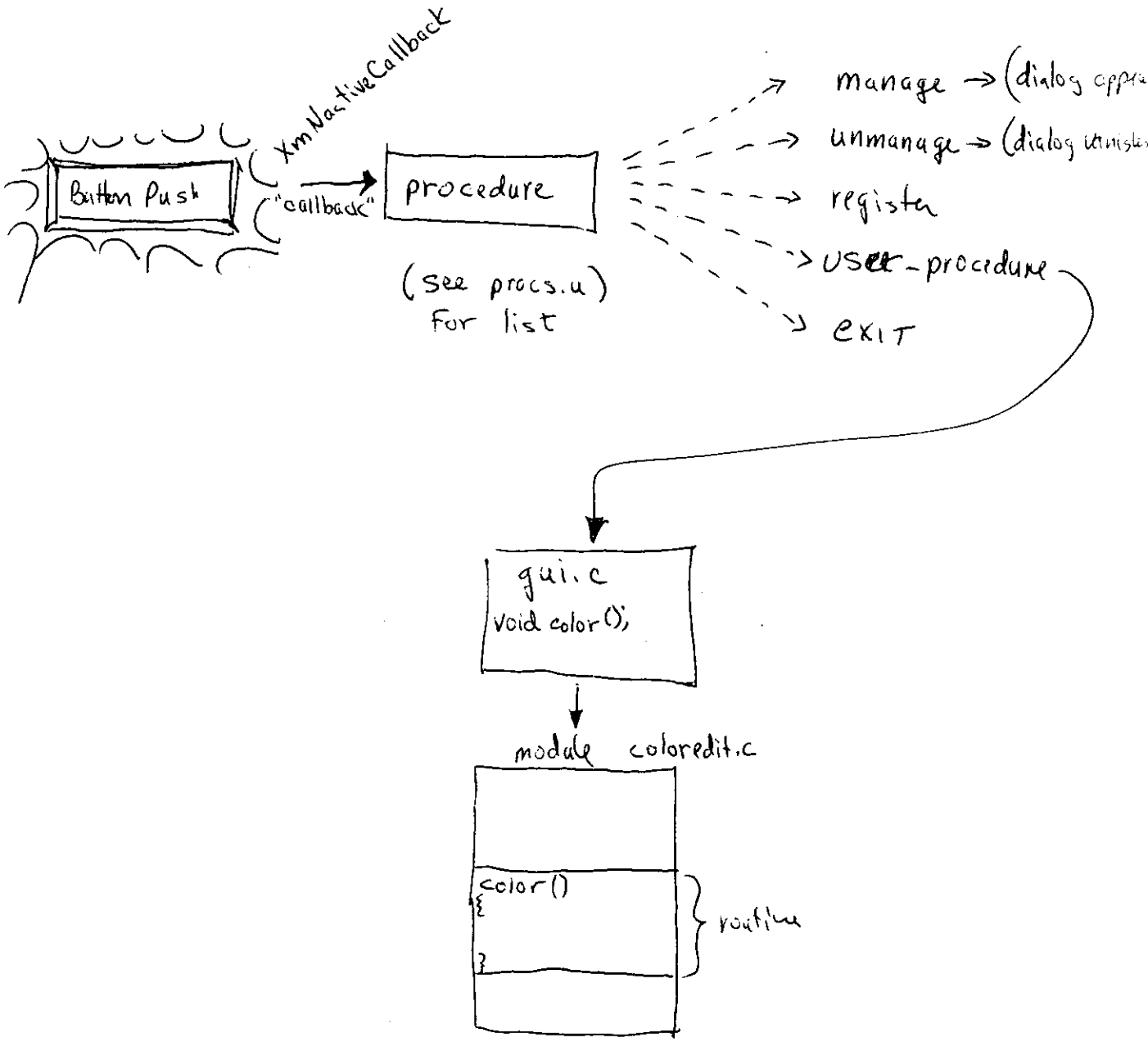
calls a "procedure" which in turn, ~~calls~~ will:

1. manage another dialog
2. Unmanage a dialog
3. Call a C routine
4. Register
5. Exit Application

most used.

(User Procedure)

"Gimme in Application" is OUT


XmNactiveCallback
Button Push
"callback"
procedure
(see procs.u)
for list
manage → (dialog appe
unmanage → (dialog unmanag
register
user_procedure
exit
gui.c
void color();
module coloredit.c
color()
{
}
routine

procedures (list)
as of 1/25/96

```
UNKNOWN !***VUIT_Action Register ***

MANAGE area_selection_dialog
MANAGE bus_branch_select_dialog
MANAGE bus_edit_dialog
MANAGE bus_sect_dialog
MANAGE cflow_selection_dialog
MANAGE cflow_socket_request_dia
MANAGE error_message_dialog
MANAGE exit_warning_box
MANAGE help_annotate_dialog
MANAGE help_dialog
MANAGE ipc_command_board
MANAGE ipf_alpha_bus_list_dialog
MANAGE ipf_report_list_dialog
MANAGE line_Z_calc_dialog
MANAGE line_z_filesel
MANAGE line_z_save_dialog
MANAGE open_file_dialog
MANAGE pf_report_dialog
MANAGE plot_options_dialog
MANAGE print_dialog
MANAGE print_opt_page_dialog
MANAGE printer_select_dia
MANAGE printer_select_dialog
MANAGE save_file_dialog
MANAGE save_network_dialog
MANAGE select_reports_dialog
MANAGE solve_dialog
MANAGE unimplemented_feature_box


UNMANAGE area_selection_dialog
UNMANAGE bus_branch_edit_dialog
UNMANAGE bus_front_box
UNMANAGE bus_output_dialog
UNMANAGE bus_sect_dialog
UNMANAGE cont_type_warning_form
UNMANAGE cor_edit_dia
UNMANAGE help_annotate_dialog
UNMANAGE help_dialog
UNMANAGE ipc_command_board
UNMANAGE ipf_alpha_bus_list_dialog
UNMANAGE ipf_report_list_dialog
UNMANAGE line_Z_calc_dialog
UNMANAGE line_tap_dialog
UNMANAGE line_z_save_dialog
UNMANAGE open_file_dialog
UNMANAGE open_file_dialog
UNMANAGE pf_report_dialog
UNMANAGE printer_select_dialog
UNMANAGE save_file_dialog
UNMANAGE save_network_dialog
UNMANAGE select_reports_dialog
UNMANAGE solve_dialog
```

Compiled by program: procs_list.c

| C-ROUTINE TO CALL: | OFFICAL VUIT NAME: |
|---|---|
| alpha_check | *** |
| alphanum_check | *** |
| alphanum_sp_check | *** |
| apply_files | *** |
| apply_files | *** |
| cancel_bus_settings | *** |
| cflow_debug_cb | *** |
| cflow_kill_cb | *** |
| cflow_launch_cb | *** |
| change_cursor_to | *** |
| change_print_plot_opts | *** |
| clear_solution_data | *** |
| cor_selection_edit | *** |
| create_cont_record | *** |
| create_dc_2_term_rec | *** |
| create_dc_multi_term_rec | *** |
| create_equiv_rec | *** |
| create_equiv_rec | *** |
| create_from_scratch | *** |
| create_line_rec | *** |
| create_pq_record | *** |
| create_reac_rec | *** |
| create_regxfmr_rec | *** |
| create_xfmr_rec | *** |
| creategraphtbl | *** |
| data_check | *** |
| decimal_check | *** |
| digit_check | *** |
| draw_area_input | *** |
| edit_apply | *** |
| edit_bus | *** |
| edit_bus_close | *** |
| edit_init | *** |
| edit_reset | *** |
| edit_send_to_pf | *** |
| error_dia_help_cb | *** |
| exit_ipf | *** |
| exit_ipf | !***VUIT_Action |
| exit_ipf_quick | *** |
| file_check_and_save_cb | *** |
| file_default_set | *** |
| file_name_set | *** |
| file_save_cb | *** |
| fill_area_selection_box2 | *** |
| fill_branch_jacket_cb_sb | *** |
| fill_bus_dialog_cb | *** |
| get_bus_alpha_select | *** |
| get_bus_selection | *** |
| graphBendToggleLabel | *** |
| help_annotate_get | file_save_proc; |
| help_annotate_get | *** |
| help_annotate_remove | *** |
| help_annotate_save | *** |
| help_dialog_page_down | *** |
| help_dialog_page_up | *** |
| help_expose_callback | *** |
| help_file_name_set | *** |
| help_input_callback | *** |
| ipc_commandString_xtoc | *** |
| ipf_alpha_srch_value_chg | *** |
| ~~ipf_bus_list_select~~ | alpha_bus_list_select; |
| line_pq_edit_delete | *** |
| line_pq_edit_insert | *** |
| line_pq_edit_replace | *** |
| line_pq_list_cb | *** |

```
66          line_z_list_number_cb      ***
67                       loadArea2      ***
68        load_all_edit_widget_id      ***
69                      overstrike      ***
70                     pfAlphaList      ***
71                pfGetFilterLists      ***
72                     pfGetReport      ***
73          pfget_solution_params      manage_solve_dialog;
74                       pfinit_cb      ***
75                   pfsolution_cb      ***
76                  printGraphData      ***
77                      print_plot      ***
78       process_pq_radio_buttons      ***
79              process_prtopt_rb      ***
80             process_regxfmr_rb      ***
81          refresh_solution_data      ***
82             reports_file_ok_cb      ***
83                      reset_data      ***
84                        sect_bus      ***
85                       sect_init      manage_bus_sect_dialog;
86                       sect_init      ***
87                         sect_ok      ***
88                        sect_tie      ***
89     send_del_data_to_powerflow      ***
90     send_del_data_to_powerflow      ***
91     send_mod_data_to_powerflow      ***
92           setCurBusDefaultName      ***
93                    set_bus_type      ***
94                   set_cont_type      ***
95               set_default_files      ***
96             set_dia_flow_deflts      ***
97   set_graph_unit_and_origin_cb      ***
98          set_printer_selection      ***
99           set_regxfmr_jckts_cb      ***
100                     solve_reset      ***
101    special_selection_action_cb      ***
102                      tap_apply      ***
103                       tap_init      ***
104                         tap_ok      ***
105         tools_set_view_mode_cb      ***
106                   tools_zoom_cb      ***
```

procs.u

This file holds all the procedures

Some procedures are in their respective *.u Files

# PSCoordFileRecord

PSCoordFile Record

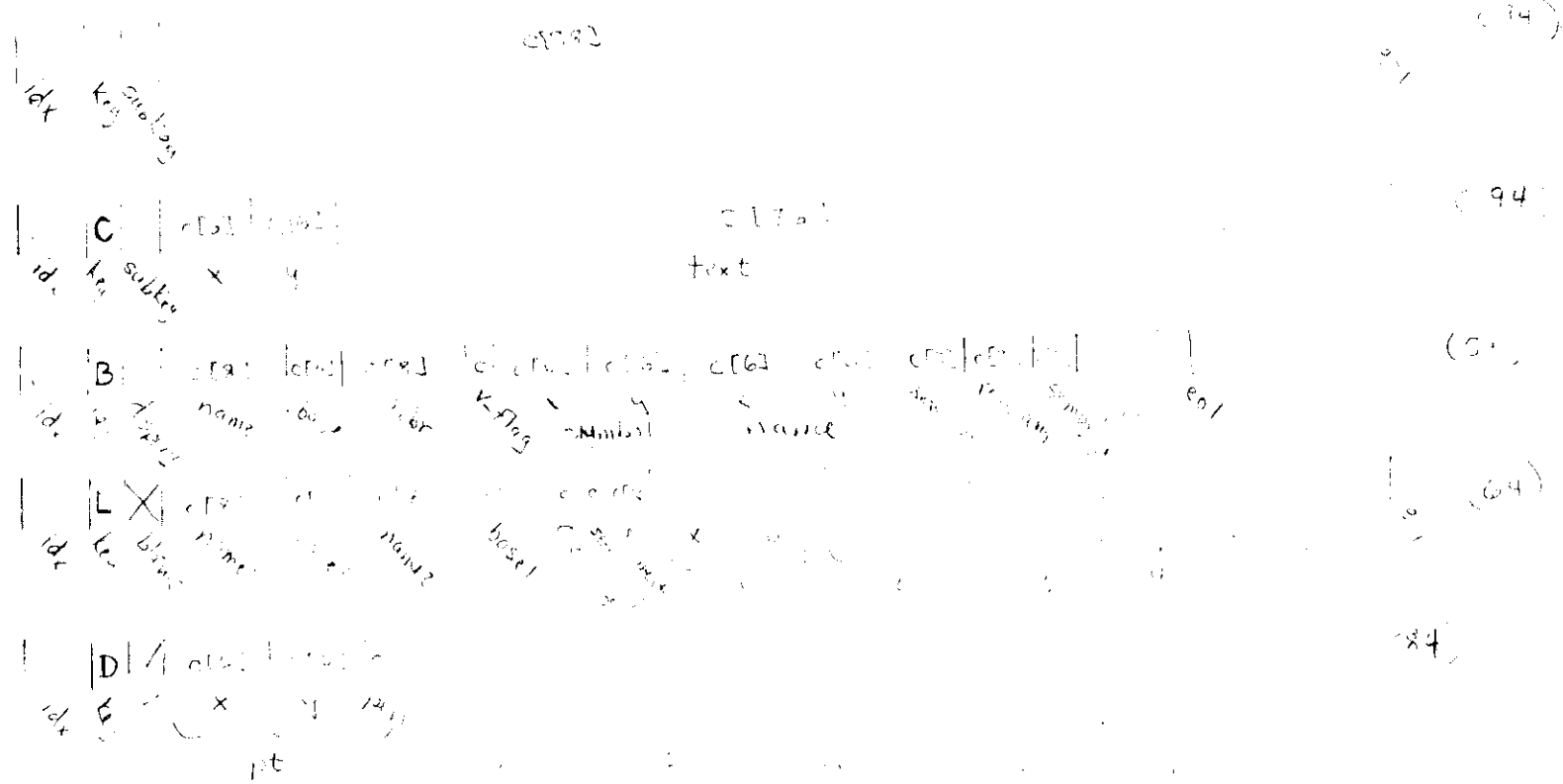
PSAnyRecord
cor

PS Comment Record
comment

PS Bus Record
bus

PS BranchRecord
branch

PS DrawRecord
draw



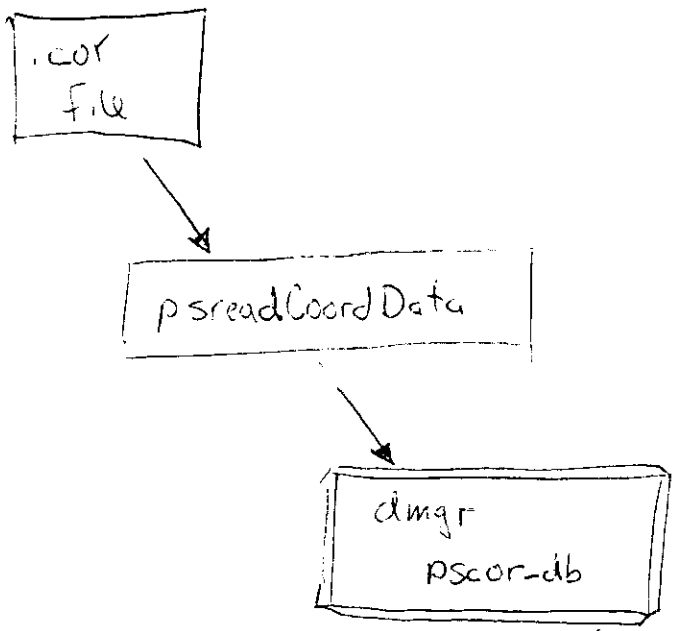

always: read coord file

pscor_db

defined as SCHEMA in pscordat.c

see coord_data.h for typedefs

read & loaded by psreadCoordData
(In pscordat.c)



standardized routine to make an edge graphelement

```
#include "graph_data.h"

GraphElement    *vertex_1;
GraphElement    *vertex_2;
GraphElement    *edge_created;
PSBranchRecord  *coordrec;


pscreateGraphCorEdge (  vertex_1,      } (From 7th param in
                        vertex_2,      }  createGraphVertex)
                        coordrec.key,
                        0,                    (level)
                        GraphClassEdgeSection,
                        & edge_created)
```

this routine includes the <u>linkage</u> logic.

PUT_DATA



## PUT_DATA, TYPE = COMMENTS

This command replaces case comments, along with case ID, project name, and headers..

There is a related command
`/GET_DATA, TYPE = COMMENTS`
which obtains the corresponding data including header 1 which is not modifiable. Header 1 is formatted to include case name, case description, program version, date, etc.. Up to 20 comments are allowed. The 2 header records must be present even if blank.

The sent values are encoded in the character array `out_buffer` in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables returned. Headers and comments are 132 characters including the ID field (H or C).

```
/PUT_DATA, TYPE = COMMENTS
CASE_ID = < case name >                    10 chars
CASE_DS = < case description >             20 chars
H < header 2 information >
H < header 3 information >
C < comment text >
...
C < comment text >
```

return status: status = 0 : success
1 : errors

Note: that only header 2 & 3 are returned to powerflow

header 1 is "read only"

## GET_DATA, TYPE = COMMENTS

This command obtains case comments, along with case ID, project name, and headers..

There is a related command
`/PUT_DATA, TYPE = COMMENTS`
which modifies the corresponding data except for header 1 which is not modifiable. Header 1 is formatted to include case name, case description, program version, date, etc.. Up to 20 comments are returned. The 3 header records are always returned.

The returned values are encoded in the character array `out_buffer` in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables returned. Headers and comments may be up to 132 characters.

```
/GET_DATA, TYPE = COMMENTS
CASE_ID = < case name >                    10 chars
CASE_DS = < case description >             20 chars
H < header 1 information >
H < header 2 information >
H < header 3 information >
C < comment text >
...
C < comment text >
```

return status: status = 0 : success
1 : errors

Note: header 1 is -- "read only"

radio button

[illegible] button state

range checks

see: checks

reactance

see: coord options

BUS_detail = Shunt

= AL_Shunt

= NO_Shunt

see: process_optrecord

option_spec_init

read

[illegible] card → [illegible] (psendot.c)
([illegible])

read_[illegible]options ([illegible])

psbalanceGraphCond ([illegible])

psDecomposes ([illegible])

[illegible]

[illegible] ([illegible])

[illegible] ([illegible])

[illegible]

[illegible]

read coord File

psreadCoordData (pscordat.c) reads the coord data using **fgets**, filling card_data [162]

card_data | B G4 HIGH 230G4HIGH 6.22 22.94 5.23 23 |

← 162 chars →

space is allocated using PSCoordFileRecord (see coord-data.h)

| | idx | key | subkey | | | | | | | | | | | | | eol | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ? | sub | | | | | | | | | | | | | | any |
| rec: | | B | | name | base | abbr | | | | | gen | rac | syn | red | | | bus |
| | | L | t | name | base | name 2 | base | id | d | sym | | | | | | | branch |
| int | 1 | 1 | | 8 | 4 | 8 | | | | | | | | | | | |

bends

ptr to this "workarea" rec = &coord_rec

area above "key" is totally zeroed

record is copied over with memcpy:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| &rec | | B | | G4 HIGH | 230 | G4 HIGH | | 6.22 | 22.94 | 5.23 | 23.67 | | (bus struct) |

index number is set in int space

rec → cor.idx = numCards

check is performed to ensure bus is not a duplicate

db_search(&pscor, &rec, &junk, COORD_KEY_NAME1_BASE1)

insert into dmgr

db_insert(&pscor-db, &rec, &junk)

continue as long as fgets does not return NULL

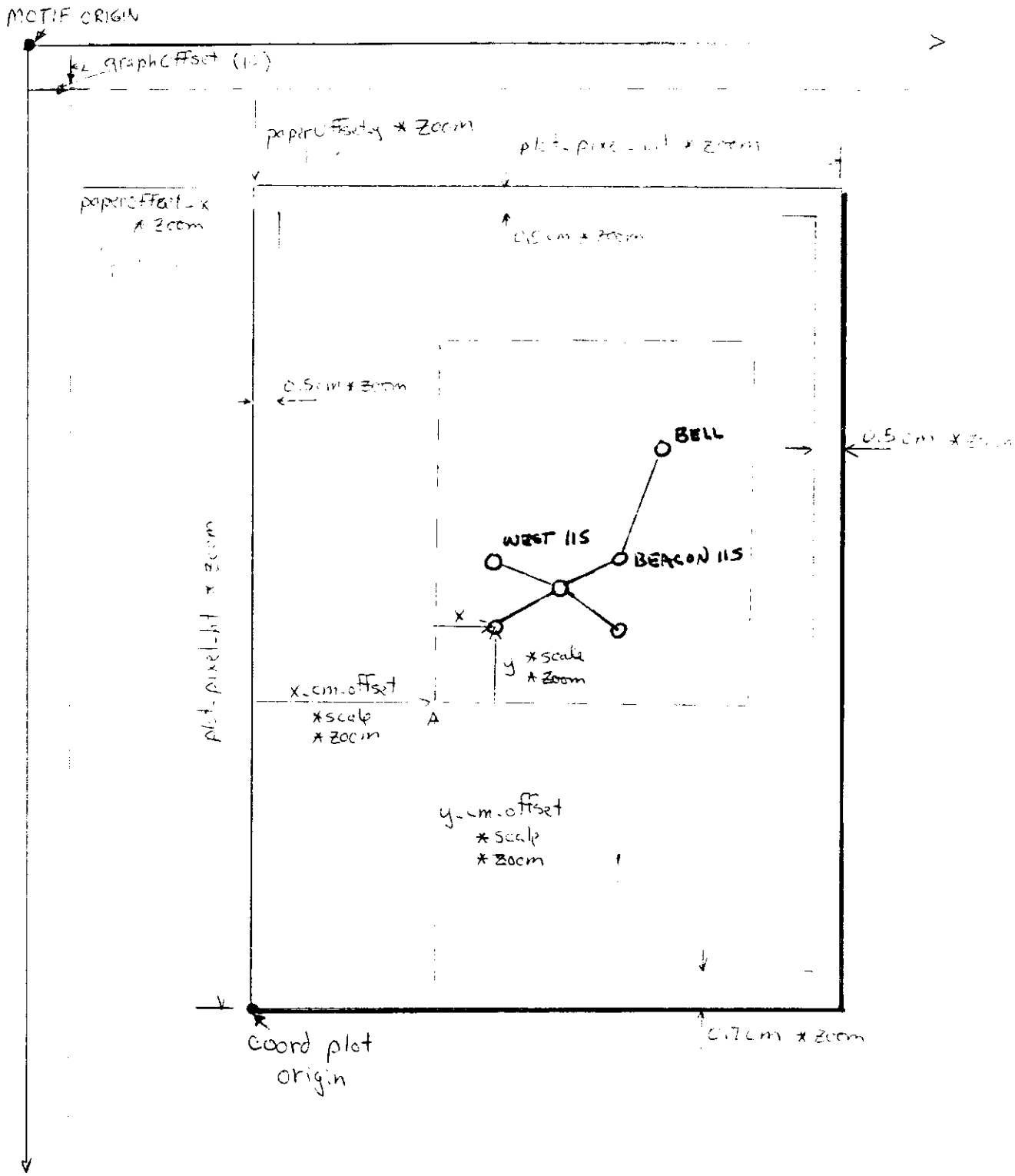
next step: init_print_opts
read_print_options
creategraphtbl (graphdata.c)
psbuildGraphCoord (graphp[illegible].c)
graphEorOn
createVertexGadgets, drawGraphEdges

redraw

see: refresh

redraw_graph_with_new_dply_opts
(printopts.c)

# Referrence Frame
(total system)




All numbers from coord-do are assumed to be in cm but are in units of pixels

Paper size is never scaled!

reference frame

(Coord → MOTIF conversion)

Steps to conversion:

1) Add x,y to their respective x,y offsets
2) Scale the above result
3) Convert Y-coord to MOTIF (origin at top)
   by subtracting y from plot height
4) Add graphOff to x,y

refresh

refreshGraph (pE_cb.c)

draw vertex data

Refresh 1/2

REFRESH
has to UPDATE all VOLTAGE data and LINE FLOW data.

5 ways to start the refresh:
1) Push SOLVE button
2) Change SOLUTION DATA ON radio button
3) Set COLOR BY KV toggle button
4) Set COLOR BY OVERLOAD toggle button
5) Push APPLY button on display menu dialog

1,3,4 call routines which in turn call REFRESH_SOLUTION_DATA.

1) PFSOLUTION_CB sends instructions to powerflow, which then solves the case.

2)

3) Set FLAG False first.

4) Set FLAG True first.

5)

---

REFRESH_SOLUTION_DATA

Makes the decision to call 1) refreshGraph -or-
2) clear_solution_data

---

REFRESHGRAPH

1) fetch windows

2) clears_solution_data

3) read_overload_boxes

4) get_bus_name_solution_opt

5) get_bus_solution_opt

6) starts SOLVE_WP

7) refresh_comments

---

SOLVE_WP

LOOP through all vertexes:

1) builds the "get_data" string with each BUS
2) sends to powerflow
3) receives back from powerflow a string containing bus data and all connecting line data
3) calls PROCESS_UPDATE_DATA

---

PROCESS_UPDATE_DATA

digests the pfdataret

for each IPF line, calls UPDATEOUTPUT

---

FOREACHIPFLINE

Finds list of pointers to all the line breaks

Screens line for unwanted junk (like error msg)

and calls UPDATEOUTPUT

---

UPDATEOUTPUT

if bus: call UPDATE_VERTEX

edge: call UPDATE_EDGE

---

UPDATE_EDGE

LOOPS and finds all FAR VERTEX from graphelement search

calls CHECKSOLNEDGE

---

CHECKSOLNEDGE

decides whether to call a) SETOVERLOADCOLOR -or-
b) SETKVCOLOR

calls setSolnDataArrow

---

SETSOLNDATAARROW

This is the biggie!

Reviews all the option pushbuttons and writes the desired string accordenly

Extensive reference to psolnData structure.

Determinds: Arrow on/off
direction
Transformer symbol required
label on/off

XtVaSetValues call transfer all data to edge widget

---

(EVENT) ( Called upon expose only )

EDGEG (edgeg.c)

XDrawLine

drawArrow/drawTransformer

drawCharsCenteredAboveLine

drawText

register_name

this is required to ensure hash lookup can find the widgets.

All text widgets, push buttons, toggle buttons, anything managed or unmanaged during ipT executing should be "registered".

also see: vuit_main_template_c (source code)
hash lookup

rescale_graph
(graphpscor.c)

Re-draws display at a different scale

FULL_RESCALE – rescales the paper edge
PARTIAL_RESCALE – rescales only the buses & lines
(i.e. grid grows/shrinks within the paper)

Routine performs a simple loop to extract each graphelement vertex from dmgr & and change the x, y coords. Then adjust the vertex gadgets. Then update all edges.

# resource file

file in user's home directory that sets some of MOTIF's behavior.

some examples:

```
XGUI * error_message_dialog*x: 850
XGUI * error_message_dialog*y: 650

XGUI *x : 0
XGUI *y : 0

XGUI* tool_dialog.width: 115
XGUI* tool_dialog.heigth: 520
```

see: autostart
write_colors_to_xgui (coloredit.c)

```
XGUI * solve_dialog * XmToggleButton * set : False
```

any child

only this child (parent)

must match material in gui.c (vuit_main_template_c)

> Note: Resource File MUST be in HOME directory!

Save comments

pourite Cornetile (pseudobase)

in PART II

loops the graph elements find gaters

gets string from google [illegible] :

Save coord

1) Pull down **File** menu - let go on **Save...**

2) save dialog pops up - Push **Save Coordinate File**

3) which calls **save_coord_cb** (in Filedlgrtn.c)

if file already exists - then the **save_coord_file_error_box** appears
ask is overwrite is wanted

calls **psWriteCoordFile** (in pscordut.c)

4) loops thru original pscor_db and transfer options to new file lst. **write_out_option_cards** ( )

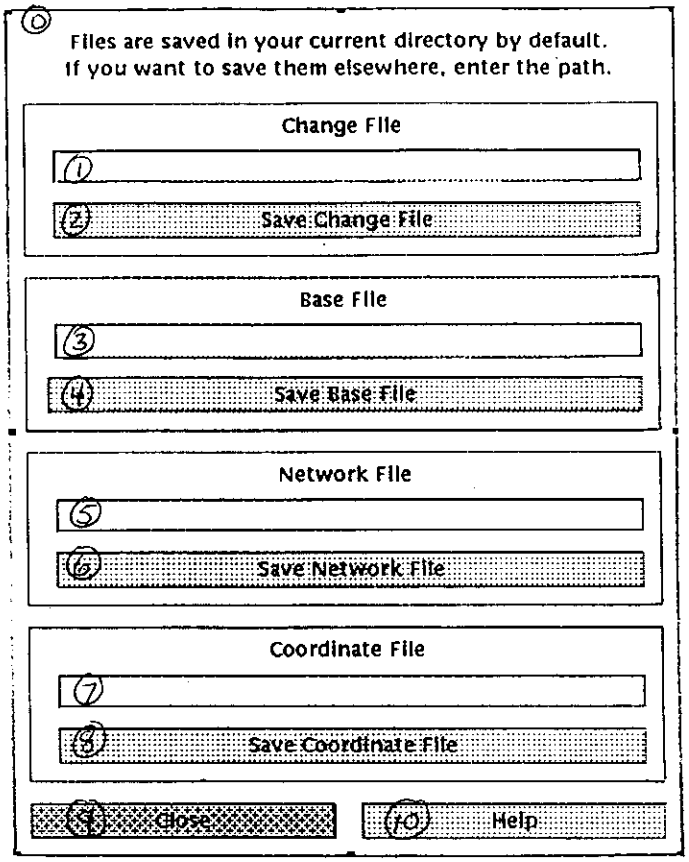
5) loops thur graph_db & writes buses
using **printPSBusRecord** (in graphdata.c)

6) loops thur graph_db & writes branches
using **printPSBranchRecord** (in graphdata.c)

NOT THE BEST WAY TO ORGANIZE CODE!
CODE IS SCATTERED IN THREE MODULES!
ALL CODE THAT DEALT WITH EACH DB
IS GROUPED TOGETHER. HOWEVER IF
PROGRAMMER WANTS TO WORK ON SAY,
SAVING COORD DATA, THEN HE NEEDS
TO CHECK OUT THREE MODULES!

save defines

Save dialog
Old




```
 0 save_file_dialog

 1 save_change_text
 2 save_change_pb     save_change_cb

 3 save_base_text
 4 save_base_pb       save_base_cb

 5 save_net_text
 6 save_net_pb        manage_save_net_dialog

 7 save_coord_text
 8 save_coord_pb      save_coord_cb
                      (filedlgprtn.c)

 9 save_close_pb      unmanage_save_file_dialog
10 save_help_pb
```

save options

Scan

[illegible].c

[illegible]

# Scale

this is a value determined to allow the plot to fit on a piece of paper - usually 8½ by 11.

There is NO general scale - actually is an X-scale and a Y-scale. Normally x-scale = y-scale.

Scale can be set using the PAGE OPTIONS dialog. Or by editing the SCale = x.xx, y.yy line in a .cor file.

See: offset

zoom-Factor

rescale

# SCHEMA

[illegible]

[illegible] SCHEMA [illegible][] =

```
{
    { 'n', 'c', [illegible] offset },
    { 'd', 'c', size, offset },
                  :
                  :
};
```

[illegible] =

| | |
|---|---|
| [illegible]-schema | ([illegible]) |
| [illegible] | (base_[illegible]) |
| char_schema | ([illegible] datum) |
| [illegible] | ([illegible]) |
| [illegible] | ([illegible]) |
| sub_schema | ([illegible]) ([illegible] |
| [illegible] | → ([illegible]) |

Segment

see: Flow segment

selection.c

Bill's major works.

Basically is a collection of routines that takes data from any of 3 sources: coord card, selection box or graph element and displays it in the proper pop-up boxes for editing.

There are:

11 bus data routines

3 continuation data routines

9 branch data routines

8 (minor) branch creation routines

3 area interchange routines

29 support routines

---

also see: Field_spec

self test

[illegible]

use gu +debug 128 . . . .

to get the self test push button under "Process"

this calls procedures system_tests & system_test_pcdr

see: system test

test_graphdata (gr_[illegible])

send
command_to_pf

Commands are often generated by GUI, and sent to IPF by the ipc_synch_rw routine.

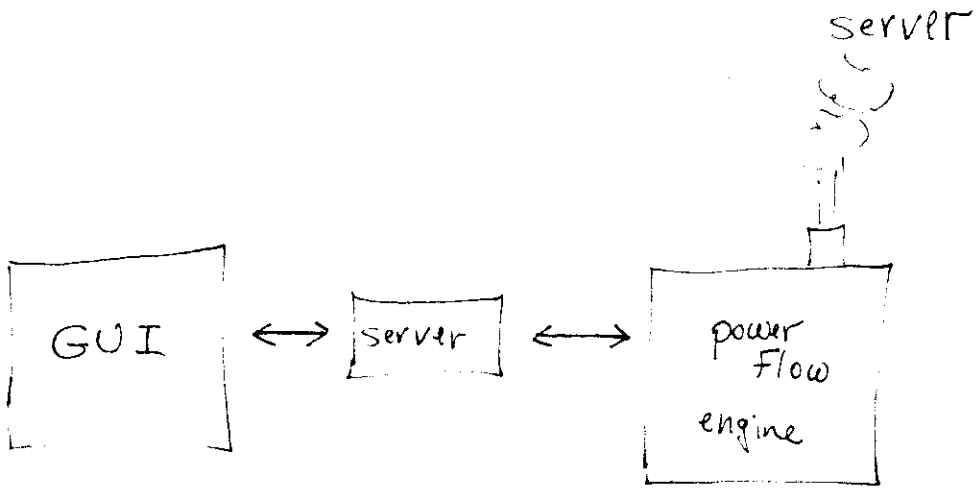
also see: inter_process_communication

sensitive

XtSetSensitive ( widget, False )

make button look gray & disables the button so no routines or procedures are called (i.e. it cannot be pushed)

also see: disable_pushbuttm (utils.c)
enable_pushbuttm




gui -server ... ... (don't connect to powerflow)

+server default

-servername (name of server)

/shrunis/ipfexe/ipfsrvnew

-otherwise-

defaults to:

"ipfsrv" in home directory

GUI generates a charactor string such as

"/get_data, type=COMMENTS\n *[EOM]"

cr

ipc_synch_rw( inbuf, outbuf );

From pf

above line send charactor string to powerflow which takes action and returns the required data.

set-jacket

(selection.c)

set-jacket

setobject
(also getobject)

routine used by several modules to set & get the object (window) which was set when a graph-request was clicked on. These routines are in graph_data.c.

set values

see: get values

XtVaSetValues (pw, [illegible]
XmNlabelString, str, NULL)

see XtVaGetValues (get values)

slider bar

see: color edit
slider bars

# socket

```
#include <sys/types.h>
#include <sys/socket.h>

socket_num = socket( int af, int type, int protocol)
```

socket_num:
- x socket #
- -1 error

af:
- always AF_INET (only one supported by unix)

socket procedure

(1) main() calls ipc_startup (ipc_cb.c)

(2) calls ipc_startup.1

which calls create_socket1 (ipc_net.c)

(in /shrunis/ipf/src/ipc/srcnf)

solution data

Solve dialog

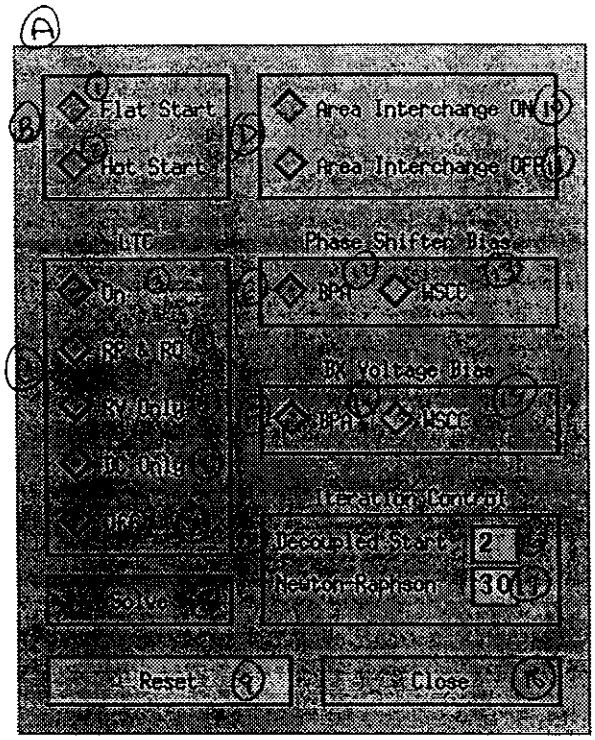



**Figure 4-33. Solve Dialog Box**

```
 A) solve_dialog
 B) solve_start_frame
 C) solve_LTC_frame
 D) solve_area_frame
 E) solve_phase_bias_frame
 F) solve_BX_bias_frame
 G) solve_iteration_form

 1) solve_flat_start_rb
 2) solve_hot_start_rb
 3) solve_LTC_on_rb
 4) solve_LTC_RPRQ_rb
 5) solve_LTC_RV_rb
 6) solve_LTC_DC_rb
 7) solve_LTC_off_rb

 8) solve_solve_pbpf     pfsolution_cb    (pf_cb.c)
 9) solve_reset_pb       solve_reset      (pf_cb.c)

10) solve_area_on_rb
11) solve_area_off_rb
12) solve_phase_BPA_rb
13) solve_phase_WSCC_rb
14) solve_BX_BPA_rb
15) solve_BX_WSCC_rb
16) solve_decoupled_text
17) solve_newton_text

18) solve_close_pb       unmanage_solve_dialog
```

GraphSourceCoord ≡ from coord file
GraphSourceBase ≡ from base File

example:

```
GraphElement *pelement;
pelement -> source = GraphSourceCoord;
```

see: explode

start GUI

See: overview
GUI
gui start

Start

see - gui start

startup

# Start up
(in autostart.c)

Bill's routine to act as major switch to debug options.

* Using -debug 2 allows two extra debug buttons to appear on the cascade menu.

* set_kv_colors - pre-define colors

---

## Autostart

* fetch_file_windows (helps slow computers bring up GUI faster)

* set_default_files (remove CANCEL button from FILE OPEN dialog - set READY TO LOAD label if default files exist. define some button colors.

* apply_files
  tries to load possible default files

* set_button_state
* change_cursor_to

} initialize toolbox button

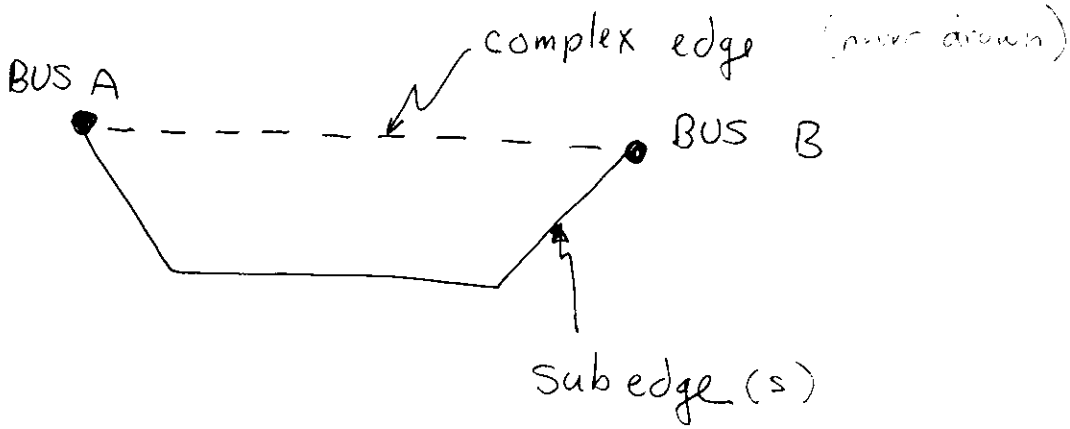
See: autostart

State

see: button state

~~button~~ s

# subedge




see: manageEdge

# switchEdge

Source: graphdata.c

purpose: transfer one endpt to a new vertex

Format: switchEdge (pedge, poldvertex, pnewvertex)

# Synchronous

Term used in inter-process-communiction. Implies that GUI sends a command to powerFlow and sits idle, waiting for a response.

Asynchronous - means to send & forget, until interrupted by a return command.

See: inter process communication

if -debug 2048 is used, some extra cascade buttons are available under "Process" which will perform some limited testing.

see: self test
debug

tap

see: line tap

templates

vuit_include_template_c (23 lines)

vuit_main_template_c
hash routines
Fetches

vuit_makeFile_template_c

vuit_stubs_template_c

test

# text boxes (gadgets)

text boxes are text gadgets where users can edit/enter data.

also see: lookup_and_get_Field (selection.c)
lookup_and_Fill_Field

checks

XmText

Working with text widgets requires some unique logics. Most important - it is desired to leave the right side null-filled (not blank-filled) in order to enable users to quickly add new data. I.E. users should not have to delete these spaces in order to add new data.

The left side of a text field should have all leading blanks removed for <u>numeric data</u> only. For alpha data the situlation is different as straqtically placed spaces are critical. e.g. tA zone is not the same as a At zone! NEVER remove leading spaces from an alpha text field! Routine "lookup_and_fill_field_w_dec" is programmed to handle this.

toggle button

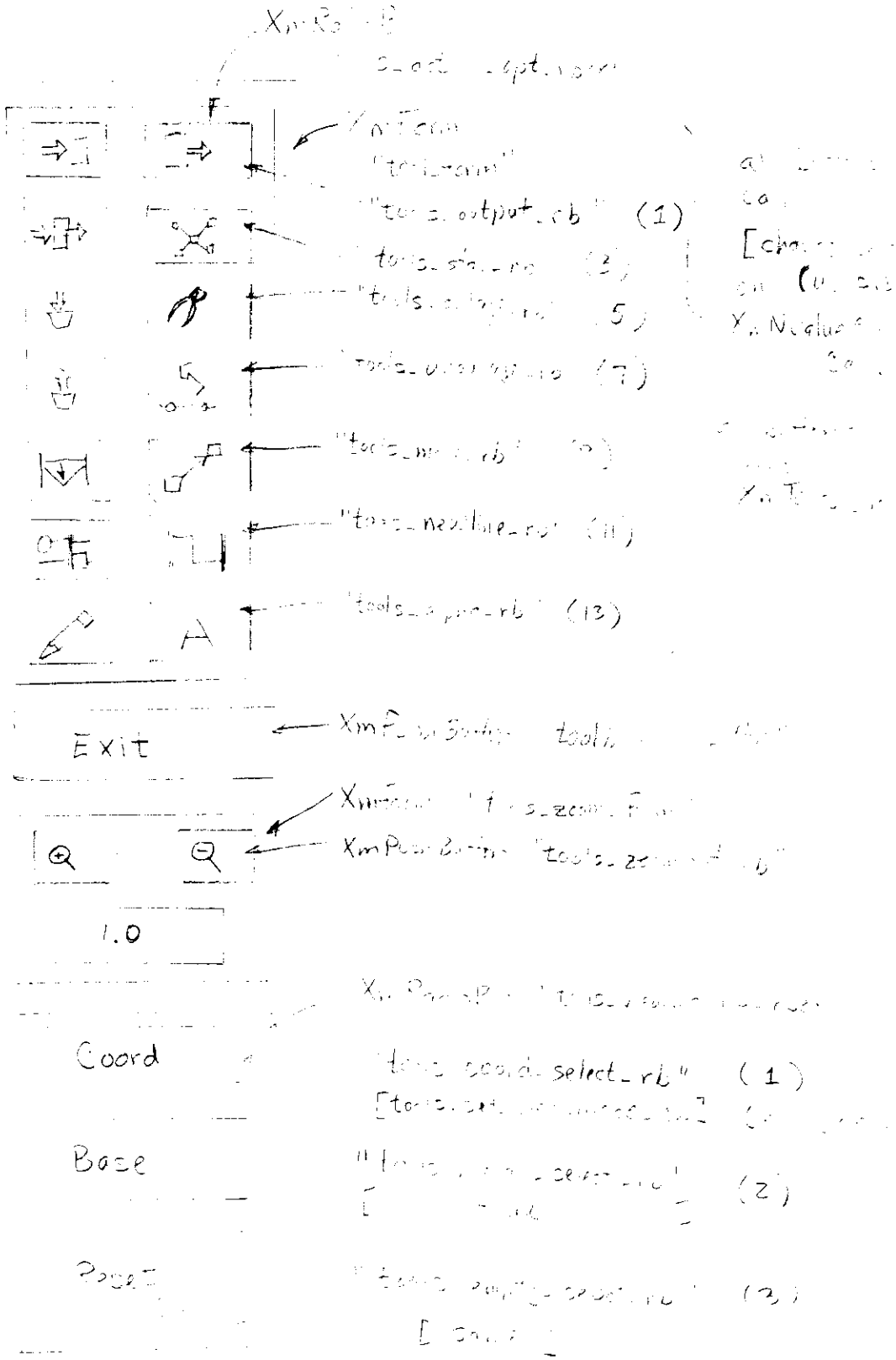
see button state

button sensitivity (gray out)

disable-pushbutton ("name") } (~~print~~ pts ~~c~~
enable-pushbutton ("name") } utils.c

thickness

see: margin width


"tools_output_rb" (1)
(5)
(7)
(11)
"tools_arrow_rb" (13)
Exit
1.0
Coord
"tools_coord_select_rb" (1)
Base
(2)
(3)

# transformerOn

clues whether to draw transformer symbol.
Generally if kv's at each end don't
match, then line is a transformer
(see edgeq.c)

see:

also see: arrowOn
drawline (edge.c)

trap door

r -debug 2048 enables test to work

see: dmgr debug

# truePrintGraphData

Source: graphdata.c

purpose: print ALL GraphElements

Format: truePrintGraphData (0, 0, 0)

# turnOnSubEdge

source: graphdata.c

purpose: manages, (makes visible) <u>all</u> subedges (line segments) of a line.

See: manageEdge

See: "gui.uil file"

"unresolved" error

occurs during MAKE

routine is not in any module!

update

see: changeGraphBusVertex (oldname, newname)
changeVertexName (ptr, pvetc)
switchEdge (edge, old vertex, new vertex)
edgeupdate
move_object_xy (GraphElement, x, y)

utils.c

collection of "workhorse" routines

widget_id ..... gets Widget id
Vuit_MU ..... manage/unmanage widgets
cstring_to_string
get_file_name_from
change_cursor
change_cursor_to
month_to_three_char
three_char_to_month
left_justify_text
disable_pushbutton .... gray out pushbutton
enable_pushbutton ....
get_widget_text
implied_dec_rw
answer
user_cancel_cb
userCancel

see: db-valid-db (dmgr.c)

coord_db_not_valid (graphpscor.c)

Version

in vwt_main_template_c (line [illegible])

see: createGraphVertex

vertex is our term used to denote a gadget (a point or item) on screen.

the Graphic display on screen is composed of the basic elements <u>VERTEXES</u> & <u>EDGES</u>.

vertex Gadget
(Create)

[illegible] routines are available to create [illegible]
manager decides which to use depending
on CLASS plus some minor requirements
[illegible]

Bus → createBusGadget

None → createStringGadget
Group
Comment
SubComment
OrgComment
Font
LegendTex
LabelText

Band → createPtGadget
Draw
Border
Paper
Legend
Label

[illegible] manager [illegible]
[illegible]

Vocabulary

BASE FILE - (NET file) - ascii file contain the POWERFLOW input data.

BRANCH - a line which represents a high voltage transmission line

BUS - a substation "node" to which BRANCHes (transmission) lines are connected to.

CALLBACK - instructions for a WIDGET such that if a user pushes the mouse cursor on it, will cause program to call a routine.

COORDINATE FILE - ascii disk file containing the original graphical coordinates of a graph.

COORD_DB - dmgr (data manager) file which hold all coordinate file RECORDS.

CORE - a computer "dump" file of a crashed execution.

DATA MANAGER - data base system to organize the data used by GUI. 4 such data bases exist - graph_db, coord_db, cmd_db and chg_db.

DIALOG - a framed display created by MOTIF.

DRAG - the act of moving the mouse to, then pushing a mouse button down and HOLDING it down while moving the mouse. Usually this will cause an icon to be moved across the terminal screen.

EDGE - a straight line in the MAIN WINDOW having two x,y coordinates & stored in the graph_db data base.

EXPOSE - action which occurs when a DIALOG is moved to the "top of the pile" (or an overlying DIALOG is removed), such that the underlying dialog is "exposed". MOTIF must then redraw (or restore) the graphics that was hidden underneath.

GRAPHELEMENT - one "record" of the GRAPH_DB file.

GRAPH_DB - dmgr (data manager) file which holds all the graph data (or GRAPHELEMENTS)

GRAPH LINK - system used by the dmgr (data manager) to link certain data records together. i.e. a BRANCH has a LINK to two bus VERTEXes.

GUI - the Graphical Users Interface. The very program itself! Pronouned "GOOEY".

INTER PROCESS COMMUNICATION - act of sending data between GUI and POWERFLOW. (GUI and POWERFLOW are separate programs)

LIBRARY - System Analysis methold of holding all the modules to GUI.

LINKS - see GRAPH LINK

MAIN WINDOW - the "graphics" window showing the buses, branches or "grid" in the GUI program.

MANAGE - (verb) to make a WIDGET appear on screen (usually by creating it)

MERGE - the process of joining the GUI data and the POWERFLOW data

to make a common case

MODULE - computer file contain one or more ROUTINEs

MOTIF - programmable computer program that generates displays for GUI.

PF - abbreivation for POWERFLOW.

POWERFLOW - Fortran computer program written by System Analysis which performs the voltage, current and power solutions.

PROCESS - an executing program within the computer. (several PROCESSes may be executing at once)

RECORD - one line of any acsii computer file

RESOURCE FILE - the XGUI file (or XGUI.DATA) which holds some default values for the GUI program.

ROUTINE - a subroutine of C code.

SEGMENTATION FAULT - causes computer bomb - usually by writing at a bad address such that the address is not the proper beginning of an allocated memory location.

TOOLBOX - The 16 buttons on the left side of the GUI DIALOG.

VERTEX - a graphical point having an x,y coordinate & stored in the graph_db data base.

VERTEX GADGET - MOTIF widget which appears on screen.

VUIT - program to edit the MOTIF graphics.

WIDGET - a graphical item visible on screen (usually rectangular) e.g. pushbuttons, scroll bars, text labels.

vuit

a "toolkit" used to edit widgets

use: vuit gui.uil

(runs on unisrv only - use "rsuni" to get on unisrv)

vuit_main_template_c

has: hash routines
fetches

also see: templates

see: widget-id

get widget data

workproc

see: XtAppAddWorkProc

also see: print

write normally implys output to a file.
print is normally to screen.

See. pswriteCoordFile

see: resource file

# XmString

see: compound string

## XmTextSetString (widget, string)

(Puts string into a text window

Required:
```
#include <Text.h>
Widget widget;
char string[120];
```
)

## str = XmTextGetString (widget)

(Gets pointer to string in text widget.

Required:
```
Widget widget;
char *str
```
)

details of lesser used commands on next page:

XmTextInsert ( w, pos, str );

XmTextGetLastPosition ( w )

XmTextGetTopCharacter ( w, pos )

XmTextPosToXY

---

XmTextPosition pos (a typedef)

XmTextVerifyCallbackStruct

# XtAppAddWorkProc

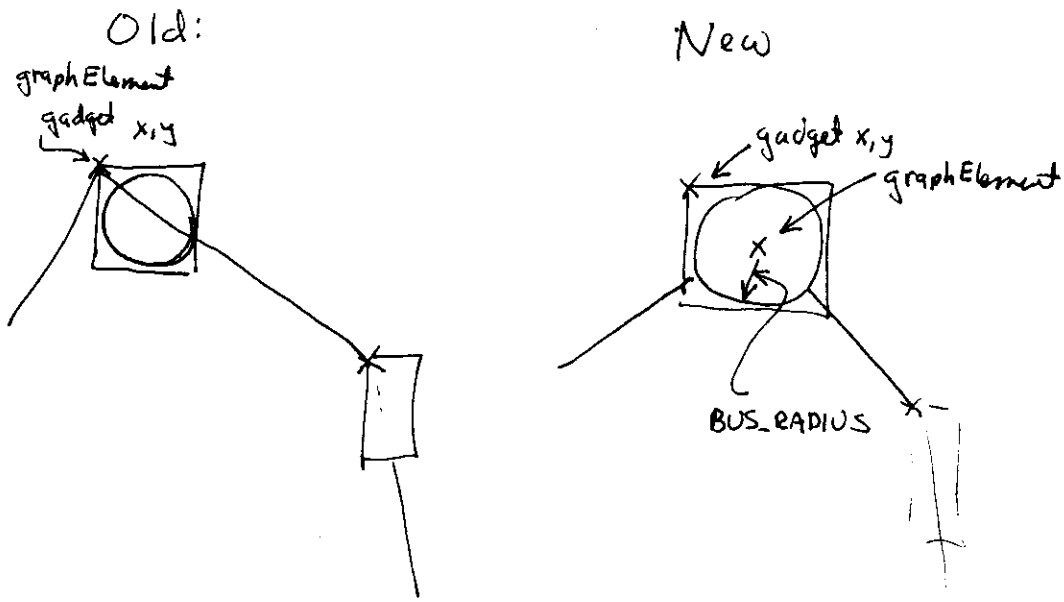
calls a routine & makes routine compute on lower priority.

i.e.

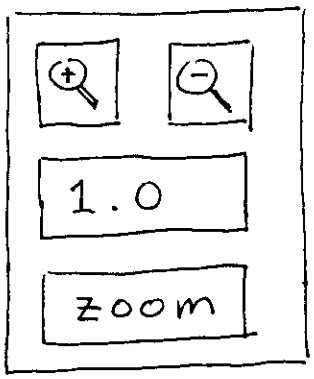
```
editor_wpid = XtAppAddWorkProc ( app-context, merge_wp, &status)
```

calls this routine.

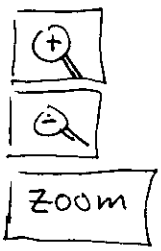
X-y location
(bus)



# zoom




3 buttons call tools_zoom_cb (toolbox.c)




tag = 0

= 1

tag = 9

<u>zoom_factor</u> is used by various routines

<u>rescale_graph</u> resizes the display
(graphpsetor.c) (uses a multiplier - not the zoomfactor)