Bonneville

POWER ADMINISTRATION

DETAILED DESIGN DOCUMENT

Interactive Powerflow

THIS BOOK CONTAINS PERSONAL TIBBITS AND REFFERENCES TO THE GUI (GRAPHICAL USER'S INTERFACE).

THIS BOOK IS INTENDED TO BE A DEBUGGING TOOL FOR NEW PROGRAMMERS, SUCH THAT IT WILL DIRECT NEW PROGRAMMERS TO THE CORRECT MODULE TO INVESTIGATE.

IN CASE OF CONFUSION WHILE TRYING TO DEBUG OR UNDERSTAND THE GUI PROGRAM, THIS BOOK IS A GOOD STARTING POINT. ANYTIME I HAVE ENCOUNTERED CONFUSION DURING DEBUGGING OR ENHANCING THE PROGRAM, I HAVE TRIED TO ADD (A) PAGE(S) TO THIS BINDER, HOPEFULLY TO HASTEN WORK AT A LATER TIME.

BEST SUGGESTION IS TO REVIEW THE "OVERVIEW" PAGE (IN THIS BINDER) AS A STARTING PLACE. ALSO SEE THE "VOCABULARY" PAGE.

(more)

APOLOGY

IN THE PAST, I HAVE NEVER FELT COMFORTABLE WITH OTHER PROGRAMMER'S NOTES AND/OR DOCUMENTAION. EACH PROGRAMMER SEEMS TO DEVELOP HIS/HER OWN PERSONAL VOCABULARY THAT IS UNCLEAR TO OUTSIDE PROGRAMMERS. AND I AM NO EXCEPTION. NORMALLY I DISCARD THEIR NOTES AND RE-WRITE MY OWN PERSONAL NOTES. NO DOCUMENTATION IS PERFECT AND WITH MY 25 YEARS OF PROGRAMMING I AM NO STRANGER TO THIS CONTROVERSY. I AM WELL AWARE THAT "NOBODY READS DOCUMENTATION" AND HAVE ANALYZED THE VARIOUS DOCUMENTING METHODS.
I BELIEVE DOCUMENTING PROGRAMS IS A VERY UNPRODUCTIVE PART OF PROGRAMMING AND TO GET AROUND THIS, I HAVE ELECTED TO WRITE MOSTLY HAND-WRITTEN PERSONAL MEMOS TO MYSELF.

I SINCERELY APOLYIZE FOR MY WRITING AND DOCUMENTATION STYLE TO ANYONE WHO SHOULD FIND THESE NOTES DIFFICULT TO FOLLOW.

WILLIAM E. ROGERS

abbreviated name
(bus)

add

see : create

create bus comp

addGraphLink (graphLink)

Allocate
Colorcell

R O U T I N E S

XAllocColor	(dpy, cmap, colorcell_def)
XAllocColorCells	(dpy, cmap, contig, plane_masks, nplanes,pexels, ncolors)
XAllocColorPlanes	(dpy, cmap, contig, pixels, ncolors, nreds, ngreens, nblues, rmask, gmark, bmask)
XAllocNamedColor	(dpy, cmap, colorname, colorcell_def, rgb_db_def)
XAllocStandColormap	()
XCopyColormapAndFree	(dpy, cmap)
XCreateColorMap	(dpy, w, visual, alloc)
XFreeColormap	(dpy, cmap)
XFreeColors	(dpy, cmap, pixels, npixels, planes)
XGetRGBColorMaps	(dpy, w, std_colormap, count, property)
XGetStandardColormap	(dpy, w, cmap_info, property)
XInstallColormap	(dpy, cmap)
XLookupColor	(dpy, cmap, colorname, rgb_db_def, hardware_def)
XParseColor	(dpy, cmap, spec, rgb_db_def)
XQueryColor	(dpy, cmap, colorcell_def)
XSetRGBColormaps	(dpy, w, std_colormap, count, property)
XSetStandardColormap	(dpy, w, cmap_info, property)
XSetWMColormapWindows	(dpy, w, colormap_windows)
XSetWindowColormap	(dpy, w, cmap)
XStoreColor	(dpy, cmap, colorcell_def)
XStoreColors	(dpy, cmap, colorcell_defs, ncolors)
XStoreNamedColor	(dpy, cmap, colorname, pexel, flags)
XUninstallColormap	(dpy, cmap)

K E Y W O R D S

Color

- XAllocColor
- XAllocNamedColor
- XFreeColors (dpy, cmap, pixels, npixels, planes)
- XLookupColor
- XParseColor
- XQueryColor
- XStoreColor
- XStoreColors
- XStoreNamedColor

ColorCell

- XAllocColorCell

XmGetColors

Colormap

- XAllocStandColormap
- XFreeColormap (dpy, cmap)
- XSetWMColormapWindows
- XInstallColormap
- XSetRGBColormaps
- XSetStandardColormap
- XSetWindowColormap
- XUninstallColormap

Get

- XGetRGBColorMaps
- XGetStandardColormap

Install

- XInstallColormap
- XUninstallColormap

Named

- XAllocNamedColor
- XStoreNamedColor

Planes

```
        XAllocColorPlanes
        XAllocate( , , , plane_mask, nplane, , , )
        XFreeColors( , , , , planes )
```

RGB

```
        XSetRGBColormaps
```

Set

```
        XAllocColor
        XAllocColorCell
        XAllocNamedColor
        XAllocStandColormap
        XInstallColormap
        XSetRGBColormaps
        XSetStandardColormap
        XSetWMColormapWindows
        XSetWindowColormap
```

Standard

```
        XAllocStandColormap
        XSetStandardColormap
        XUninstallColormap
```

Store

```
        XAllocColor
        XStoreColor             ( dpy, cmap, colorcell_def )
        XStoreColors            ( dpy, cmap, colorcell_defs, ncolors )
        XStoreNamedColor        ( dpy, cmap, colorname, pexel, flags )
```

S T R U C T U E S

XtStandardColormap

```
  struct{
        Colormap colormap;
        unsgn long red_max;
        unsgn long red_mult;
        unsgn long green_max;
        unsgn long green_mult;
        unsgn long blue_max;
        unsgn long blue_mult;
        unsgn long base_pixel;
        VisualID visualid;
        XID killid;
        } XtStandardColormap;
```

XGCValues.

XtGCMask

6

function
pad
pad
pad
pad
plane_mask
Foreground
background
line_width
line_style
cap_style
join_style
Fill_style
Fill_rule
arc_mode
pad xy
title
Stipple
ts_x_origin

XColor

```
  struct{
        unsgn long pixel;
        unsgn short red;
        unsgn short green;
        unsgn short blue;
        char pad;
        } XColor
```

see: checks

alpha_check

alpha_sp_check

alphanum_check

alphanum_sp_check

alpha_ul_sp_check

alpha to Float

see: atoF

area cards

[illegible]

[illegible] MOTIF [illegible]

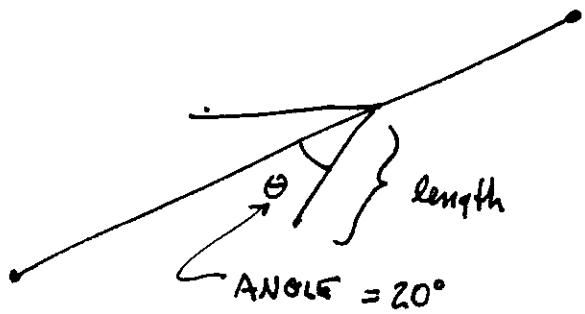
[illegible] (all-data.c)

[illegible]

[illegible] MOTIF [illegible]

(Handles 'A' and 'I' [illegible])

the "line flow" symbol which tells which direction power is flowing in line.



arrowOn must be set TRUE or arrow will not draw

edgeg → edge.arrowDirection has direction

see: ~~drawArrowXYA (arrowXya.c)~~
drawArrow (edgeg.c)
arrowOn
linedraw

also see: drawTransformer
Flow segment
XinArrowOn

edgeg -> edge. arrowOn

arrowOn set FALSE in setKvColor (1)
setOverloadColor (2)

TRUE in setSolnDataArrow (3)

s

assemble program

see: gui.m

atof

atof_cnt

atof_cnt_zero

```
# include <stdlib.h>
double atof(str)
  char *str;
```

routines in stdlib_ext.c

atof_cnt ensures null field doesn't bomb call

atof_cnt_zero same, also makes blanks zero

see: convert

gui
r -server -socketid 1040 -autostart

this param tells program NOT to autostart (in autostart.c)

in resource file:

```
XGUI* file_select_dia_network_text.value
XGUI*file_select_dia_coord_text.value
XGUI*file_select_dia_base_text.value
```

these files are automatically loaded

autostart.c has startup & autostart module

autostart is called by gui.c

also see : breakEdge

createBendArray (graph_data.c)

bus vertexes are "pixmaps", created by setting:

XmNlabelType, XmPIXMAP
XmNlabelPixmap, toolpix (see: ~~vert~~ createVertexGadgetXY)

pixmaps are stored at busgdgt.h

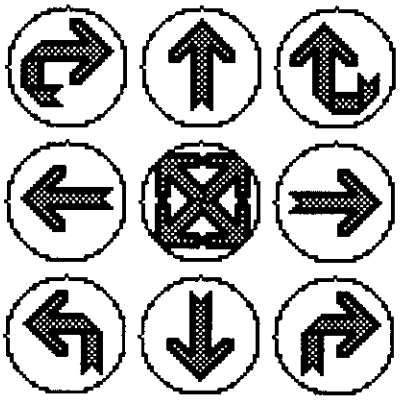
use /usr/bin/X11/bitmap busgdgt.h 11x11
to edit these bitmaps

also see: bitmap GUI NOTES (notebook)

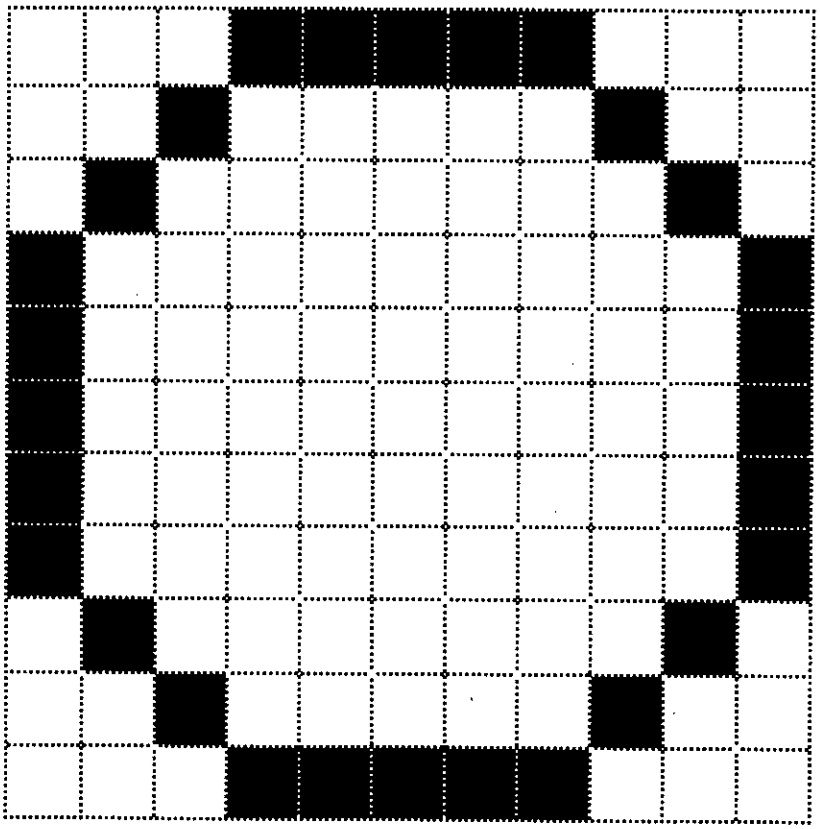
Bitmap editor

File | Edit | Filename: busgdgt.h Basename: busgdgt Size: 1

Clear
Set
Invert
Mark
Unmark
Copy
Move



Point
Curve
Line
Rectangle
Filled Rectangle
Circle
Filled Circle
Flood Fill
Set Hot Spot
Clear Hot Spot
Undo



to u : /usr/[illegible]/X11/bitmap ipfarson.h

BLAST_EM

PURPOSE: list of instructions to move UNIX code to VAX

(1) dlogin bpa50
    EOHBBER
    (password)

(2) (in UNIX window)
    su sccs
    "library"
    cd /shrunis/ipf/to_vms/dan_gui
    gui_blastem_to_vms
    (answer YES to 1st & 3rd question)
        1) Copy all files? YES
                (ignore dcp error on [ ]bldguieu.m.univel)
        2) Copy individual files (y|n) NO
        3) Create a new gui.c w/ vuit? YES
        4) Buit another executable? NO

(3) (in VMS window - same as (1) above)
    SET DEF [IPF.SRC.GUI.REFUNIX]
    MMS/DESCRIPTION=BLDGUILV.M/IGNORE=WARNINGS
                (ignore long stream of warnings)
    MMS/DESCRIPTION=BLDGUIEV.M/IGNORE=WARNINGS
                (ignore long stream of warnings)
                (can be 3 letters, i.e. MMS/DES=BLDGUIEV.M/IGN=WAR)

    (now to test)
    GUI -DISPLAY DS5005:0

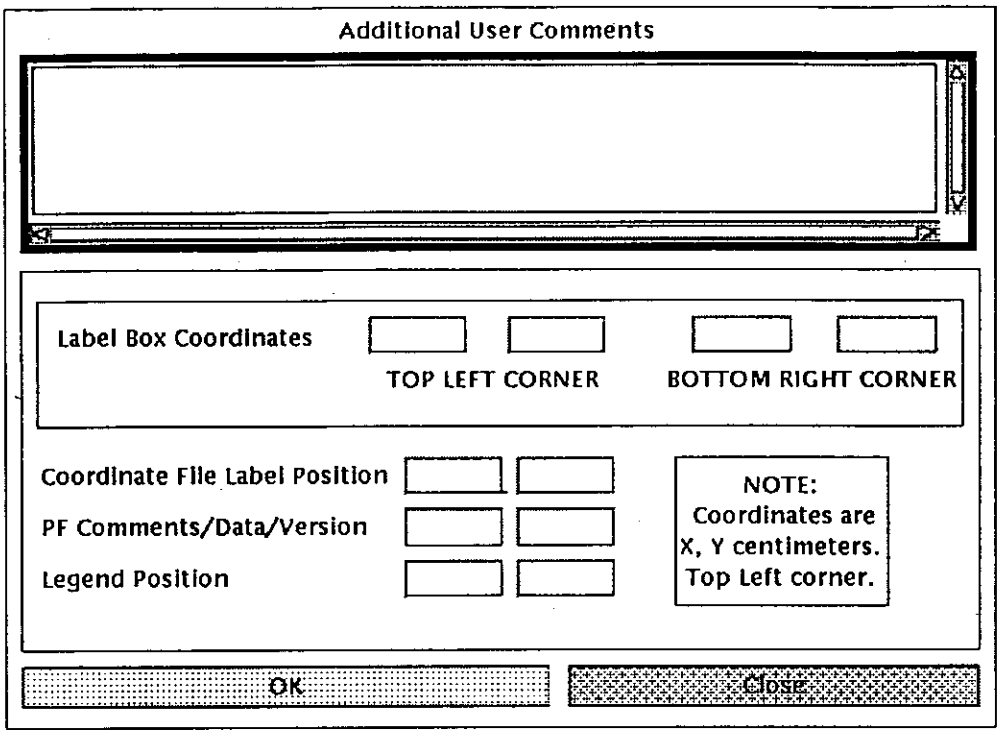
    @DAN_PUSH_IPF.COM gui.exe
    @DAN_PUSH_IPF.COM gui.uid
    PUSH_FILE

also see: GUI (VMS)

source: ds5005::/shr5/eohbber/pfi/doc/ blastem.doc

boilerplate dialog

(old)



a) user_comments_scroll_window
b) plot_comments_text
c) plot_user_comments_sw
d) Boilier_plate

e) Label_coord_form
f) label_box_x
g) label_box_y
h) label_box_se_x
i) label_box_se_y
j) coord_name_x
k) coord_name_y
l) pf_data_x
m) pf_data_y
n) legend_x_text
o) legend_y_text

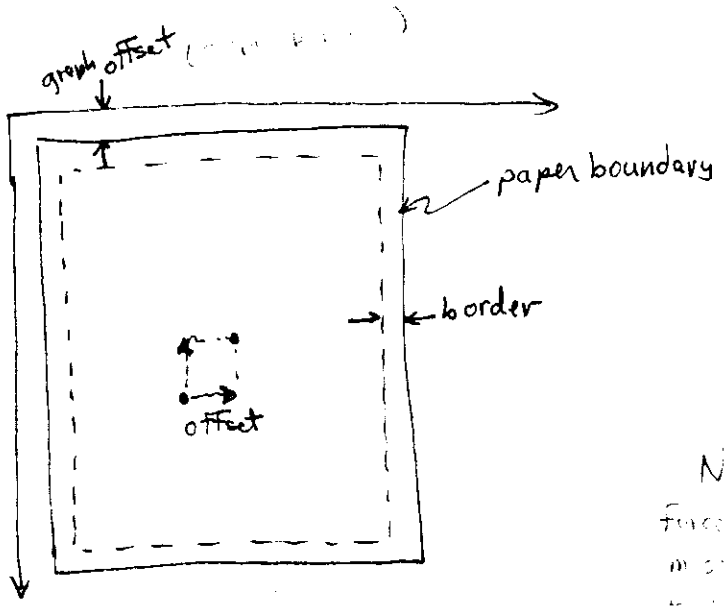
p) Boilier_plate_ok_pb
q) printopt_close

border

is the line (box) drawn around a plot.

also - the area where no plots should be plotted. (A blank zone).

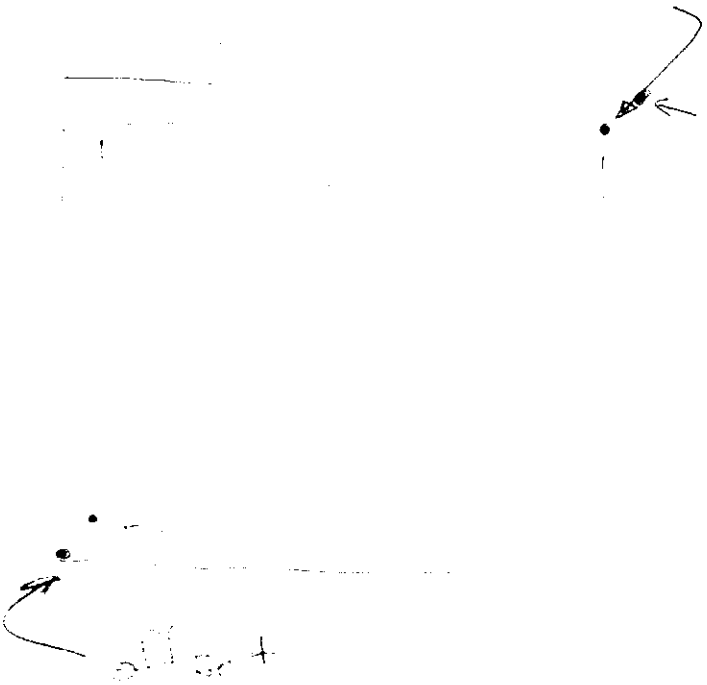
border is used only when a plot appears too close to the paper edge.



see: refference frame

BORDER CARD

BOrder = xx.xx , yy.yy



<u>border</u> is based on the <u>paper size</u> (upper right)
border is <u>not</u> a box!

also see:

coordinate options

see: label box

psCreateLabelBox (graphpscor.c)

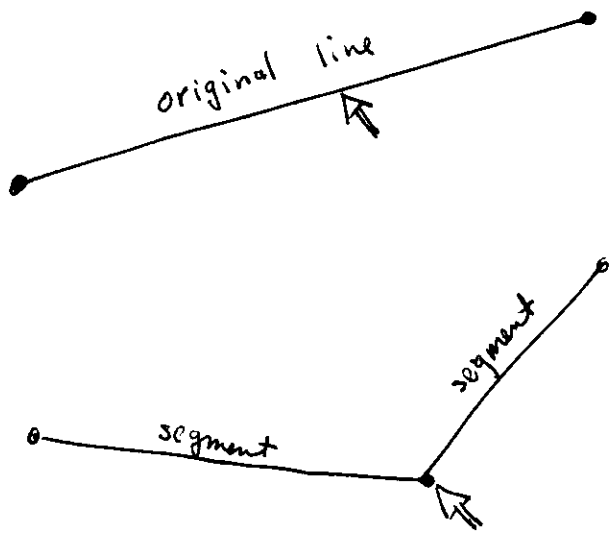
psDrawBox (graphpscor.c)

psCreatePaperEdges

see: coord branches
(for format of record)

breakEdge

the words actually means to "break" a single straight edge (line) in to 2 segments which are joined together somewhere between the original line's endpts.



one of the toolbox icon sets up action so when mouse button is clicked - program will seek a line to break (or bend).

bus

First [illegible]

[illegible] FindFirstBusPC (pcordata.c)

bus exists

use

busFoundInPF(busname)

↑

12 char.

int busFoundInPF (busname)

↑

12 chars

returns 0 not found

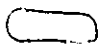
1 Found

source: bussect.c

Bus Icon Sets

bus_shapes_dialog

putting button calls: set-bus-icon (pf_cb.c)

descrip	Small	Med	Large
empty circle	SCN	MCN	LCN
Filled circle	SCB	MCB	LCB
Square	SSN	MSN	LSN
Filled square	SSB	MSB	LSB
bar	SBN	MSN	LSN
Filled bar	SBB	MBB	LBB
			LRR
			LRS

icon_bus_shape	CIRCLE	0	"C"
	SQUARE	1	"S"
	BAR	2	"B"
	RECT	3	"R"
icon_bus_style	HOLLOW	0	"N"
	FILLED	1	"B"

"R"

~~bus_bus~~ - key = ptr to gadget.

icon_bus_radius

icon_bus_width

icon_bus_height

bus name vertexes

bus vertexes
(create)

psbuildGraphCoord

- createGraphVertex (Symbol)
- createGraphVertex (name)
- pscreateGraphCorEdges (all bends)

} graphpscor.c

— later — (in Filedlgrtn)

createVertexGadgets (vertex.c)

manageVertex

See: Creating Vertexes

button sensitivity

use disable-pushbutton ("name")
-or-
enable-pushbutton ("name")
} utils.c

-or-

XtSetSensitive (wid, Boolean v1)

button state

get state:

```
(int)XmToggleButtonGetState (wid)
(int)XmToggleButtonGadgetGetState (wid)
```

don't mix widgets with Gadgets!!!
or it won't work!!!

must include:

```
#include <Xm/ToggleBG.h>
 -or-
#include <Xm/ToggleB.h>
```

set state:

```
set_button_state ( "name", state )
```

```
#define ON True
        OFF False
```

coord file symbol for BOX

(draws box from x, y to BOTTOM RIGHT corner)

see: box

Case id

see:

PUT_DATA, TYPE= COMMENTS

pf_descrip.c

In library - get magazine

"Technology Review"

Subject: Anticounterfeiting measures.

Callback

Defination: a term used in Xwindows, when an event occurs, this event performs a "callback" – telling a routine to execute.

events are: pushbutton, expose, etc

i.e. user uses mouse, clicks on a portion of screen containing a pushbutton, causing an "event" to occur, which then looks up that button's "callback", and calls the desired procedure. Procedure in turn (may) call a C routine.

change

see: update

set_values

ASCII Character Set

CHARACTERS ?

ASCII Decimal Number	Character	Meaning	ASCII Decimal Number	Character	Meaning
0	NUL	Null	32	SP	Space or blank
1	SOH	Start of heading	33	!	Exclamation mark
2	STX	Start of text	34	"	Quotation mark
3	ETX	End of text	35	#	Number sign
4	EOT	End of transmission	36	$	Dollar sign
5	ENQ	Enquiry	37	%	Percent sign
6	ACK	Acknowledgement	38	&	Ampersand
7	BEL	Bell	39	'	Apostrophe
8	BS	Backspace	40	(	Left parenthesis
9	HT	Horizontal tab	41	)	Right parenthesis
10 A	LF	Line feed	42	*	Asterisk
11 B	VT	Vertical tab	43	+	Plus sign
12 C	FF	Form feed	44	,	Comma
13 D	CR	Carriage return	45	-	Minus sign or hyphen
14 E	SO	Shift out	46	.	Period or decimal point
15 F	SI	Shift in	47	/	Slash
16	DLE	Data link escape	48	0	Zero
17	DC1	Device control 1	49	1	One
18	DC2	Device control 2	50	2	Two
19	DC3	Device control 3	51	3	Three
20	DC4	Device control 4	52	4	Four
21	NAK	Negative acknowledgement	53	5	Five
22	SYN	Synchronous idle	54	6	Six
23	ETB	End of transmission block	55	7	Seven
24	CAN	Cancel	56	8	Eight
25	EM	End of medium	57	9	Nine
26	SUB	Substitute	58	:	Colon
27	ESC	Escape	59	;	Semicolon
28	FS	File separator	60	<	Left angle bracket
29	GS	Group separator	61	=	Equal sign
30	RS	Record separator	62	>	Right angle bracket
31	US	Unit separator	63	?	Question mark
64	@	At sign	96	`	Grave accent
65	A	Upper case A	97	a	Lower case a
66	B	Upper case B	98	b	Lower case b
67	C	Upper case C	99	c	Lower case c
68	D	Upper case D	100	d	Lower case d
69	E	Upper case E	101	e	Lower case e
70	F	Upper case F	102	f	Lower case f
71	G	Upper case G	103	g	Lower case g
72	H	Upper case H	104	h	Lower case h
73	I	Upper case I	105	i	Lower case i
74	J	Upper case J	106	j	Lower case j
75	K	Upper case K	107	k	Lower case k
76	L	Upper case L	108	l	Lower case l
77	M	Upper case M	109	m	Lower case m
78	N	Upper case N	110	n	Lower case n
79	O	Upper case O	111	o	Lower case o
80	P	Upper case P	112	p	Lower case p
81	Q	Upper case Q	113	q	Lower case q
82	R	Upper case R	114	r	Lower case r
83	S	Upper case S	115	s	Lower case s
84	T	Upper case T	116	t	Lower case t
85	U	Upper case U	117	u	Lower case u
86	V	Upper case V	118	v	Lower case v
87	W	Upper case W	119	w	Lower case w
88	X	Upper case X	120	x	Lower case x
89	Y	Upper case Y	121	y	Lower case y
90	Z	Upper case Z	122	z	Lower case z
91	[	Left square bracket	123	{	Left brace
92	\	Back slash	124	|	Vertical line
93	]	Right square bracket	125	}	Right brace
94	^ or ↑	Circumflex or up arrow	126	~	Tilde
95	← or _	Back arrow or underscore	127	DEL	Delete

Characters.
(my design)

drawn by: drawChars.c ([illegible].c)
data in: EdgeGP.h
drawChars verter.c

Characters

See: EdieGph

char null

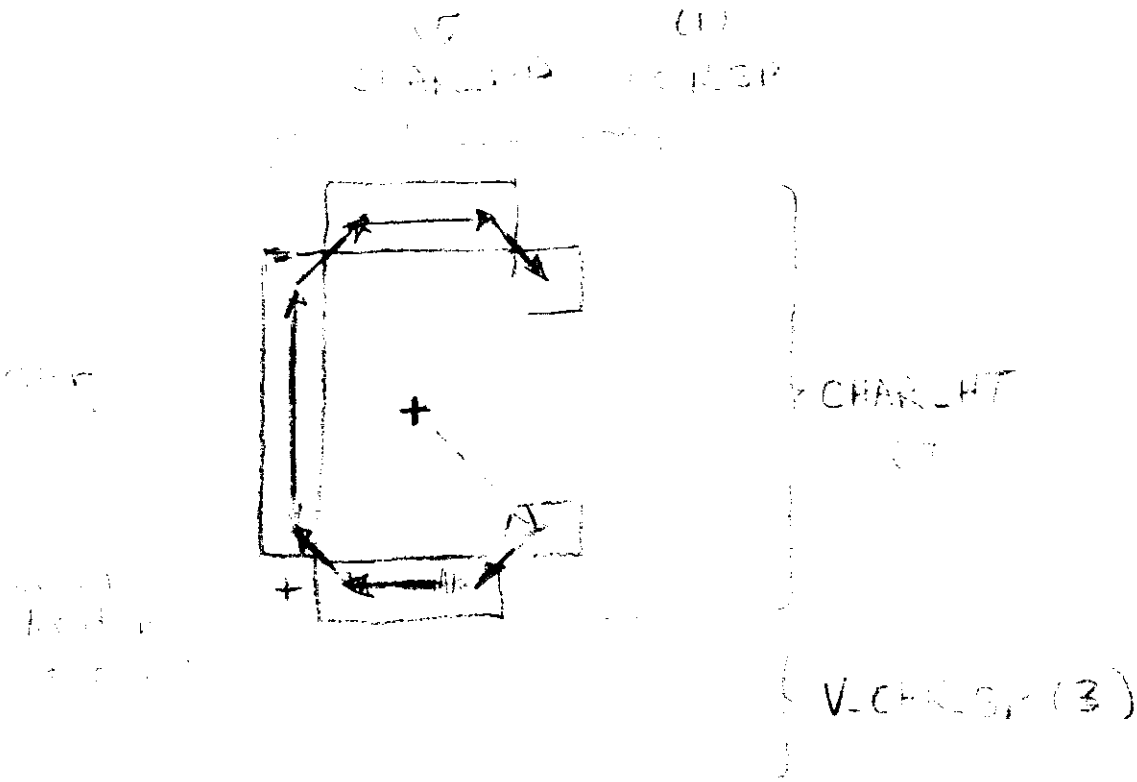
```
if (a[0] != '\0')
```

character size

size of EDGE lettering set by LTR_SIZE
in edgeg.c

size of VERTEX lettering set by LTR_SIZE
in vertex.c

Chardraw



"C" x: 10 2 -1 -2 -1 0 1 2 1 0
y: 2 1 0 -1 -4 -1 0 1 0

Note there are separate indices for vertex and edge

vertexic draws vertices
edgeic draws the edges

Checks

MEMO

From: John Rutis

To: Planning Methods Section

Date: 15 Sept 1992

Subject: Data checking functions

Recently it was decided that we should add data checking functions to the GUI text boxes. I have created a number of functions that check each character as it is entered and only allow specified sets of characters. If the character entered matches the allowed set, it is entered in the text box - if it does not match, it is rejected and the bell rings. The following is a list of the current functions and the character sets they allow. Each function is called by the XmNmodifyVerifyCallback of the text box.

callback procedure	character set
digit_check	only digits 0-9
int_check	digits 0-9 and '-' in column 1
decimal_check	digits 0-9 and '.' and '-' in colunm 1
alpha_check	only letters a-z,A-Z
alpha_sp_check	letters a-z,A-Z and space
alphanum_check	letters a-z,A-Z and digits 0-9
alphanum_sp_check	letters a-z,A-Z and digits 0-9 and space
alpha_ul_sp_check	letters a-z,A-Z and digits 0-9 and '_' and space

If there are any other sets needed, please let me know.

There is also a new function, data_check, that checks the completed entry in the text box. It is called by the XmNloosingFocusCallback. Currently, it only checks integers. Later it will be expanded to check floating point and regular expressions. It requires that you enter a template in the callback tag field specifying ranges and values that are valid for that text box. The values and ranges are enclosed in brackets - []. Ranges are delimited by '<' and values and range sets by ','. Following are some examples:

[300]	match only the integer 300
[-200<200]	any integer from -200 to 200
[100,200,300,400]	only 100 or 200 or 300 or 400
[100<200,300<400]	any integer between 100 & 200 or between 300 & 400
[100<200,300,400]	any integer between 100 & 200 or 300 or 400

The template may be up to 80 characters including the brackets. If the value is outside the range(s), an error box informs the user that the value is out of range but allows keeping the value.

These callbacks should be added to all text boxes that should limit what the user can input.

see: chkentry.c module

Checks

MEMO

m: John Rutis

To: Planning Methods Section

Date: 15 Sept 1992

Subject: Data checking functions

Recently it was decided that we should add data checking functions to the GUI text boxes. I have created a number of functions that check each character as it is entered and only allow specified sets of characters. If the character entered matches the allowed set, it is entered in the text box - if it does not match, it is rejected and the bell rings. The following is a list of the current functions and the character sets they allow. Each function is called by the **XmNmodifyVarifyCallback** of the text box.

callback procedure	character set
digit_check	only digits 0-9
int_check	digits 0-9 and '-' in column 1
decimal_check	digits 0-9 and '.' and '-' in colunm 1
alpha_check	only letters a-z,A-Z
alpha_sp_check	letters a-z,A-Z and space
alphanum_check	letters a-z,A-Z and digits 0-9
alphanum_sp_check	letters a-z,A-Z and digits 0-9 and space
alpha_ul_sp_check	letters a-z,A-Z and digits 0-9 and '_' and space

If there are any other sets needed, please let me know.

There is also a new function, data_check, that checks the completed entry in the text box. It is called by the **XmNloosingFocusCallback**. Currently, it only checks integers. Later it will be expanded to check floating point and regular expressions. It requires that you enter a template in the callback tag field specifying ranges and values that are valid for that text box. The values and ranges are enclosed in brackets - []. Ranges are delimited by '<' and values and range sets by ','. Following are some examples:

[330]	match only the integer 330
[-200<200]	any integer from -200 to 200
[100,200,300,400]	only 100 or 200 or 300 or 400
[100<200,300<400]	any integer between 100 & 200 or between 300 & 400
[100<200,300,400]	any integer between 100 & 200 or 300 or 400

The template may be up to 80 characters including the brackets. If the value is outside the range(s), an error box informs the user that the value is out of range but allows keeping the value.

These callbacks should be added to all text boxes that should limit what the user can input.

Class

```
#define GraphDisplayOff                 0
#define GraphDisplayOn                  1
#define GraphDisplayRequestOn           2

typedef long GraphClass;
#define GraphClassVertexBus             0
#define GraphClassVertexName            1
#define GraphClassVertexBendPoint       2
#define GraphClassVertexGenerator       3
#define GraphClassEdgeSection           4
#define GraphClassEdgeComplexSection    5
#define GraphClassEdgeSubSection        6
#define GraphClassVertexGroup           7
#define GraphClassVertexDrawPoint       8
#define GraphClassEdgeDraw              9
#define GraphClassPaperVertex           10
#define GraphClassPaperEdge             11
#define GraphClassVertexComment         12
#define GraphClassVertexSubcomment      13
#define GraphClassVertexOrgComment      14

#define GraphClassVertexLegendCorner    15
#define GraphClassVertexLegendText      16
#define GraphClassEdgeLegend            17

#define GraphClassVertexLabelCorner     18
#define GraphClassVertexLabelText       19
#define GraphClassEdgeLabel             20
#define GraphClassVertexBorder          21
#define GraphClassEdgeBorder            22
#define GraphClassVertexFont            23

typedef long GraphType;
#define GraphTypeVertex                 0
#define GraphTypeEdge                   1
#define ANY_TYPE                        99

#define BUS_RADIUS              5
#define PT_RADIUS               2       /* NOTE: if changed, then re-do     */
                                        /*       the pt_pb pixmap (main.u ) */
#define PARTIAL_RESCALE         0
#define FULL_RESCALE            1

#define SCRN_BORDER             15
```

source: graph_data.h

NAME

clock - Reports CPU time used

LIBRARY

Standard C Library (libc.a)

SYNOPSIS

```
#include <time.h>

clock_t clock (void);
```

DESCRIPTION

The clock() function reports the amount of processor time used by the calling process.

NOTES

The clock() function is made obsolete by the getrusage() function; however, it is included for compatibility with older BSD programs.

AES Support Level:
Full use

RETURN VALUES

The clock() function returns the amount of processor time (in microseconds) used since the first call to clock(). To determine the time in seconds, divide the value returned by clock() by the value CLOCKS_PER_SEC. If the processor time used is not available or its value cannot be represented, the clock() function returns (clock_t)-1.

RELATED INFORMATION

Functions: getrusage(2), times(3), wait(2)

colorcell

one of 256 available colors to use.
has a RED-GREEN-BLUE component

use:

XQueryColor (XtDisplay (sl), wmap, &color)

to get the RGB values

where

color.pixel = color cell # (input)

color.red
color.green } values.
color.blue

R O U T I N E S

```
XAllocColor            ( dpy, cmap, colorcell_def )
XAllocColorCells       ( dpy, cmap, contig, plane_masks, nplanes,pexels, ncolors)
XAllocColorPlanes      ( dpy, cmap, contig, pixels, ncolors, nreds, ngreens,
                         nblues, rmask, gmark, bmask )
XAllocNamedColor       ( dpy, cmap, colorname, colorcell_def, rgb_db_def )
XAllocStandColormap    ()
XCopyColormapAndFree   ( dpy, cmap )
XCreateColorMap        ( dpy, w, visual, alloc )
XFreeColormap          ( dpy, cmap )
XFreeColors            ( dpy, cmap, pixels, npixels, planes )
XGetRGBColorMaps       ( dpy, w, std_colormap, count, property )
XGetStandardColormap   ( dpy, w, cmap_info, property )
XInstallColormap       ( dpy, cmap )
XLookupColor           ( dpy, cmap, colorname, rgb_db_def, hardware_def )
XParseColor            ( dpy, cmap, spec, rgb_db_def )
XQueryColor            ( dpy, cmap, colorcell_def )
XSetRGBColormaps       ( dpy, w, std_colormap, count, property )
XSetStandardColormap   ( dpy, w, cmap_info, property )
XSetWMColormapWindows  ( dpy, w, colormap_windows )
XSetWindowColormap     ( dpy, w, cmap )
XStoreColor            ( dpy, cmap, colorcell_def )
XStoreColors           ( dpy, cmap, colorcell_defs, ncolors )
XStoreNamedColor       ( dpy, cmap, colorname, pexel, flags )
XUninstallColormap     ( dpy, cmap )
```

K E Y W O R D S

```
Color
        XAllocColor
        XAllocNamedColor
        XFreeColors            ( dpy, cmap, pixels, npixels, planes )
        XLookupColor
        XParseColor              ( find colorcell to match color by string )
        XQueryColor              ( get RGB values of colorcell )
        XStoreColor
        XStoreColors
        XStoreNamedColor

ColorCell
        XAllocColorCell

Colormap
        XAllocStandColormap
        XCreateColorMap        ( dpy, w, visual, alloc )
        XFreeColormap          ( dpy, cmap )
        XSetWMColormapWindows
        XInstallColormap
        XSetRGBColormaps
        XSetStandardColormap
        XSetWindowColormap
        XUninstallColormap

Get
        XGetRGBColorMaps
        XGetStandardColormap

Install
        XInstallColormap
        XUninstallColormap

Named
```

```
        XAllocNamedColor
        XStoreNamedColor

Planes
        XAllocColorPlanes
        XAllocate( , , , plane_mask, nplane, , , )
        XFreeColors( , , , , planes )

RGB
        XSetRGBColormaps

Set
        XAllocColor
        XAllocColorCell
        XAllocNamedColor
        XAllocStandColormap
        XInstallColormap
        XSetRGBColormaps
        XSetStandardColormap
        XSetWMColormapWindows
        XSetWindowColormap

Standard
        XAllocStandColormap
        XSetStandardColormap
        XUninstallColormap

Store
        XAllocColor
        XStoreColor           ( dpy, cmap, colorcell_def )
        XStoreColors          ( dpy, cmap, colorcell_defs, ncolors )
        XStoreNamedColor      ( dpy, cmap, colorname, pexel, flags )
```

S T R U C T U E S

```
XtStandardColormap
  struct{
        Colormap colormap;
        unsgn long red_max;
        unsgn long red_mult;
        unsgn long green_max;
        unsgn long green_mult;
        unsgn long blue_max;
        unsgn long blue_mult;
        unsgn long base_pixel;
        VisualID visualid;
        XID killid;
        } XtStandardColormap;

XColor
  struct{
        unsgn long pixel;
        unsgn short red;
        unsgn short green;
        unsgn short blue;
        char pad;
        } XColor
```

color edit
slider bars

module: color edit.c

routine:

dialog: color_edit_dialog (color.u)

Command line arguements
(or options)

These are the options users include in the same line when starting gui.

Example

$$\underbrace{\text{GUI } \underbrace{\text{+AUTOSTART DEBUG 5 BG RED}}_{\text{command line options}}}_{\text{command line}}$$

see: ipfrsrc.c

COMMENTS

see ps[illegible]:Con[illegible]File (pscorbin.c)

[illegible] see comments

also see: DEFINE

[illegible]

[illegible]

Comments

(graph comments)
(not pF comment)

1

parseCommandWindow (graphpanic)

edit comments - [illegible] comments

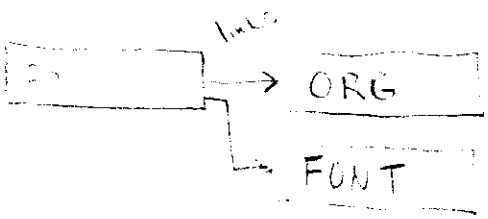
When parseGraphics is [illegible] it also sends DEFINES and COMMENTS to pF flow (see semi-commands [illegible] pF in graphpanic).

A DEFINES and COMMENTS are written out [illegible] and sent to pF flow.

1

When refreshGraph is called refresh_comments sends a GET_DATA command, asking for these DEFINED COMMENTS. They come [illegible] the [illegible] stream [illegible] clipped [illegible] and [illegible] the [illegible]

SRS: DEFINES
[illegible]



[illegible] graph [illegible]
(pF [illegible])

1

comments

(pF_comments)

see: PUT_DATA, TYPE=COMMENTS

pF_descrip.c

pscreateCommentVertexes (graphpscor.c)

Comments

(creation logic/sequences)

graphpscor.c

psDrawOptions

↓

pscreateCommentVertexes

↓

pscreateStringVertex

↓

graphdata.c

createGraphVertex

↓

(create GRAPHELEMENT only)

vertex.c

manageVertex

↓

createVertexLabelXY ("Temp-Str")

↓

XtVaCreateManagedWidget (x-str)

XtAddCallback (drawAreaButtonCB)

XtAddCallback (drawTextCB)

XtVaSetValues (labelString)

(width, height)

On Expose .....

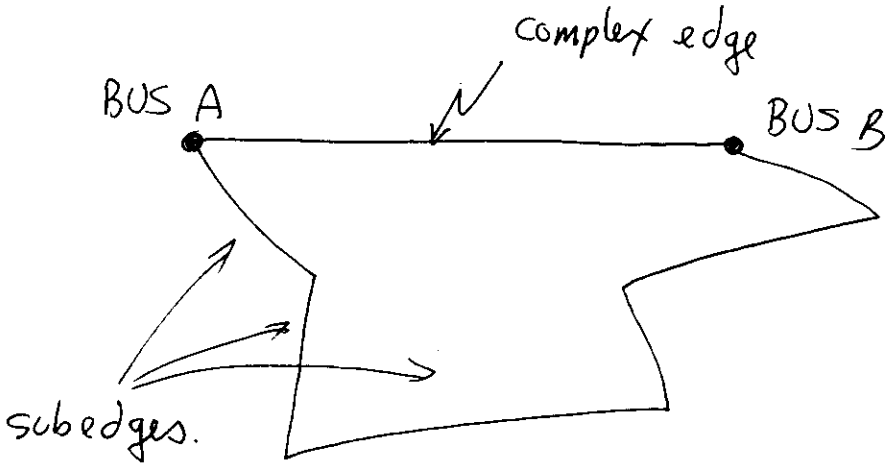
drawTextCB

XtVaGetValues

drawCharsX

XDrawRectangle

complex edge



compound strings

MOTIF text strings are COMPOUND STRINGS (not ascii format)

```
Widget  wid;
char *str[32];

                ( compound to ascii )

str[0] = XmTextGetString( wid )


                ( ascii to compound )

strcpy( str, "27.94" );
XmTextSetString( wid, str );

 - or -

XmTextSetString( wid, "27.94" );
```

Other commands:

```
xstr = XmStringCreateLtoR( in_str, XmSTRING_DEFAULT_CHARSET );

XmStringInitContext( &str, compound_str );

XmStringGetNextSegment( str, &text, &charset, &dir, &sep );

XmSTringCreateSimple

XmStringCreate(             )
```

Sample routine to convert compound string to ascii:

```
Main
{
        Widget wid;
        XmString xstr;
        Char * str;

        XtVaGetValues( wid, XmNlabelString, &xstr, NULL );

        str = cstring_to_string( xstr );

}
```

```
Char *cstring_to_string(XmString motif_string)
{

        XmStringContext     context;
        char                *text;
        XmStringCharSet     charset;
        XmStringDirection   dir;

        XmStringInitContext( &context, motif_string);

        XmStringGetNextSegment( context, &text, &charset, &dir, &separator );

        XmStringFreeContext( context );

        return( text );

}
```

See: cstring_to_string (utils.c)

```
#include <stdlib.h>
char *str;
f = atof(str);
i = atoi(str);
l = atol(str);
```

in stdlib_ext.c :

```
i = atoi_cnt_zero ( str, size)
i = atoi_cnt      ( str, size )
d = atof_cnt_zero ( str, size )
d = atof_cnt      ( str, size )
```

see: atof

convert.c

phased out - use reformat.c
instead

See reformat

coord branches

5.3.8 Branch Coordinate Data

The branch coordinate data describes the bending points in a branch and identifies which segment will show the flow and transformer symbol or compensation symbol. See Table 5-11 and Figure 5-4 for the format of the branch coordinate data record.

Column 27 requires additional explanation. Several alternative routes may be established for printing parallel circuits separately. The most preferred path is 1, next 2, etc. When the option to display parallel circuits separately is on and there are as many or more routings as circuits, the circuits are shown separately.

Table 5-11 Branch Coordinate Data Format

Column	Format	Description
1	A 1	L or T identifies a Line or Transformer.
2		Not used.
3-10	A 8	Bus1 name.
11-14	F 4.0	Bus1 kV.
15-22	A 8	Bus2 name.
23-26	F 4.0	Bus2 kV.
27	I1	Preference number for routing parallel circuits separately.
28		Not used.
29-30	I 2	Segment for annotation with flow. A negative number means do not show arrow and flow.
31-42	2F 6.2	X, Y coordinates for 1st bending point.
43-54	2F 6.2	X, Y coordinates for 2nd bending point.
55-66	2F 6.2	X, Y coordinates for 3rd bending point.
67-78	2F 6.2	X, Y coordinates for 4th bending point.
79-90	2F 6.2	X, Y coordinates for 5th bending point.

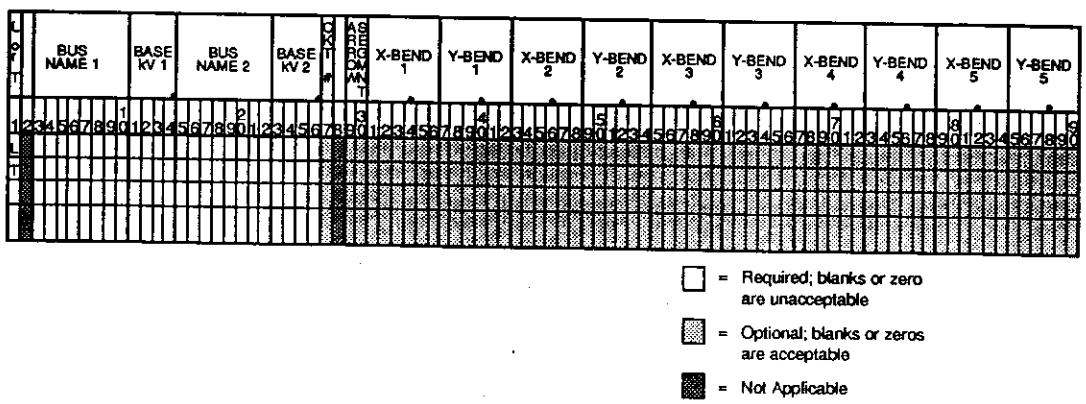


Figure 5-4 Branch Coordinate Data Record

coord_data.c

obsolete routine

replaced with pscordat.c

coordinates.

selection of coord file:

reading coordinate file to db: pcoord.t.c

also see: refference frame
MOTIF-cm conversion

Table B-2. General Appearance Options

Option	Description
SIze=XX.XX,YY.YY	DEFAULT = 8.5, 11.
ORientation=Landscape ORientation=Portrait	DEFAULT=Portrait.
OFfset=XX.XX,YY.YY	Lower left of diagram relative to lower left of page. DEFAULT = 0.0, 0.0.
TRansparency=Transparent TRansparency=Opaque	DEFAULT for insets is Opaque. Main diagram is always opaque.
SCale_factor=X.XX,Y.YY	DEFAULT = 1.0, 1.0.
BOrder=XX.XX,YY.YY	Locates upper right corner of border. DEFAULT = ~~no border~~ ~~border as close as provides per mil to edge of paper~~ Ok
BX=XX.UL,YY.UL,XX.LR,YY.LR	Locates an identification box. If XX.LR and YY.LR are zero (0), the box is positioned in the lower right corner of the diagram and XX.UL and YY.UL locate the upper left corner of the box. Default values are then assigned BOrder, CR (coordinate file), CAse_name, and COmments. These locations can be overridden. * DEFAULT = no identification box.
CAse_name=XX.XX,YY.YY	Locates case name from Powerflow program. DEFAULT = no case name.
COmments=XX.XX,YY.YY	Locates comments from user entry and Powerflow program. DEFAULT = no Powerflow comments.
~~CR=XX.XX,YY.YY~~ GONE FROM UC	~~Locates coordinate file name~~ ~~DEFAULT = relative to BX.~~
LG=XX.XX,YY.YY	Locates upper left corner of legend box. DEFAULT = no legend.

* The override capability has been deactivated by user request.

The options described in Table B-3 determine which Powerflow values will be displayed on a diagram.

The bus, branch, and flow options are all independent of each other. Those selections that are ON by DEFAULT may be turned off in one of two ways. Some, such as the bus name selection, may be toggled to the abbreviation or full bus name and base kv. Others, such as generation, may be turned off by preceding the value of interest with NO_. For example, to *not* show generation:

Table B-3. Powerflow Values Options

Option	Description
DIagram_type=Pq_flow	See flow detail; DEFAULT
DIagram_type=Mva/I	Maximum values
DIagram_type=Loss	P and/or Q set via P_S, Q_S
~~Diagram_type=Interchange~~	
DIagram_type=Coordinates	No flow data
VAlues=Normal	DEFAULT
VAlues=Difference	case1 - case2
FLow_detail=P_Sending_end	DEFAULT
FLow_detail=Q_Sending_end	DEFAULT
FLow_detail=P_Receiving	
FLow_detail=Q_Receiving	
BUs_detail=Bus_name,Abbreviation	DEFAULT
BUs_detail=Bus_name,Powerflow_name (and kV)	
BUs_detail=Voltage,kV	DEFAULT
BUs_detail=Voltage,Per Unit	
BUs_detail=Angle	DEFAULT
BUs_detail=Generation	DEFAULT
BUs_detail=Shunt	DEFAULT
BUs_detail=Load	
BUs_detail=~~Cut_branches~~	
BUs_detail=Outages	
BRanch_detail=Trans_taps	
BRanch_detail=Compensation	
BRanch_detail=Parallels,Combined	DEFAULT
BRanch_detail=Parallels,Separate	
BRanch_detail=Outages	

Table B-4. Option Recc

Column	Format	
1	A 1	O — identifies
3-90	A 88	Free field des

coordinate File

see: pscor_db

printopts routine

[illegible] (pscor[illegible])

[illegible]

[illegible]

[illegible] (pscor[illegible])

[illegible]

(common [illegible])

[illegible]

[illegible]

COORD FILE CONTENTS

[ID COORD	(standard header)
DEFINE	–
C x, y comments	(comments) –
Options	(options)
B	(Bus card)
L	(line card)
T	(transformer card)
D	(draw) –
(* EOR)	(end-of-file)

Reading the coordinate file is a lengthy and difficult process to trace, understand and debug! The follow is a summary and reference to some key steps:

***** PART I ****** Selecting the coord file ********

1) "File" pulldown is activated.

2) "Open" button on pulldown is selected, releasing the button caused the "open_file_dialog" to appear and calls routine set_default_files (in filedlgrtn.c).

3) In "FILE OPEN" dialog, coordinate file is selected and blue "Ready to Load" label comes on. Users pushes the blue

   [Load Selection] button to go on to next step.

***** PART II ******** Reading the coord data *******

4) Apply_files routine (in filedlgrtn.c) responds to this button, and determines if a new coord file is to be loaded.

5) psguiOpenCoordFile (pscordat.c) opens the file.

6) psreadCoordData (pscordat.c) reads entire coord file to coord_db.

7) init_print_opts (printopts.c) clears old options from MOTIF widgets.

8) read_print_options (printopts.c) filters the OPTIONS from coord_db and loads the MOTIF widgets.

9) psbuildGraphCoord (graphpscor.c) takes bus and line records from coord_db and begins the MERGE. (i.e. transfers data from coord_db to graph_db)

10) (After 4 above) After merge, the comments and other things must be created. psDrawOptions (graphpscor.c) is called which begins to created the following:
    a) paper edges
    b) border edges
    c) draw edges
    d) label box (if any)
    e) legend (if any)
    f) comments
    g) defines

NAME	KEY	DEFAULT	TEXT WIDGET	VARIABLE NAME
Plot Size	SIze	8.5 11.	print_size_x_text print_size_y_text	paper_x paper_y
Orientation	ORientation	P	print_portrait_rb print_landscape_rb	portrait_mode(T) (F)
Offset	OFfset	0.0 0.0	print_x_offset_text print_y_offset_text	offset_x offset_y
Transparencey	TRansparent Opaque	OP	print_opaque_rb print_transparent_rb	(T) (F)
Scaling	SCale_factor	SC	print_x_scale_text print_y_scale_text	scale_x scale_y
Border	BOrder			
Label Box	BX	blank	label_box_x label_box_y label_box_se_x label_box_se_y	(not used)
Case Name location	CA	blank	print_case_xpos_text print_case_ypos_text	(not used)
Comments location	COmments	blank	print_cmnt_xpos_text print_cmnt_ypos_text	(not used)
Legend	LG	blank	legend_x_text legend_y_text	(not used)

coord_db

data base which holds all coordinate files records.

also see:

coord_db_bus_search	(pscordat.c)
psreadCoordData	(pscordat.c)
read_coord_file	
psprintCoordData (prints all buses)	(pscordat.c)

Note: coord_data.c
has been replaced with pscordat.c

program abort creates a dump of the state of the system, all internal variables, registers & program counter.

example: gui

(segmentation fault) → core

dbx gui core

where

a parameter can be set in .cshrc file to make core file 0 & disable creation of program dump.

core can also be used to move a running program to another computer.

Create

see: create Graph Vertex (graphdata.c)

addGraph Link (graph data.c)

Create Vertex Gadget (0, 0, 0)
(vertex.c)

draw graph edges (edge.c)
(all line edges created)

ps create Graph Cor Edges (graphpscor.c)

dmgr - (db_create)

bends --- see breakEdge

ps create String Vertex ( ) (graphpscor.c)

from scratch - see NEW coord File
(create_from_scratch (filedlgrtn.c))

Create branches (any type)

Method 1:
- Use bus input icon
- Click on bus
- Use "Create Bus Comp" (below bus jacket)
  Release on desired type.
- Type 2nd BUS name, base
- Fill in branch data
- Push [ADD]

Method 2:
- Use line icon
- Click on BUS
- Drag line & click on 2nd BUS
- Use type palldown to set line type
- Fill branch data.
- Push [ADD]

Create buses

1) Click on Bus ICON (dialog appears)
2) If bus is ALREADY in PF skip to step 6
3) Fill out dialog boxes
4) Push [ADD] – to create new bus in powerflow
5) [CLOSE] – skip to 8
6) Bring up Alpha List (Edia, Alpha List)
7) Select Bus to put on coord file
8) Click on main window

create bus vertex

see: bus vertexes (create)

_riateGraphBUXXYXY (graphdata.c)

create_bus comp

Pull down menu which allows creation of any bus componet.

calls: create_cont_record
- pq_record
- reac_rec
- line_rec & set_title_mode_2namebase
- xfmr_rec
- xfmr_shift_rec
- equiv_rec
- regxfmr_rec
- dc_2_term_rec
- dc_mult_term_rec

create edge gadget

manageEdge

→ drawLine ( )

→ XtVaCreateManagedWidget

XtAddCallBack

create edges

for any edge

see: pscreateGraphCorEdge
pscreateGraphCorEdges } graphpscor.c

for T or L cards

Note: the above two routines do not create the <u>widget</u>. Must still call <u>manageEdge</u>!

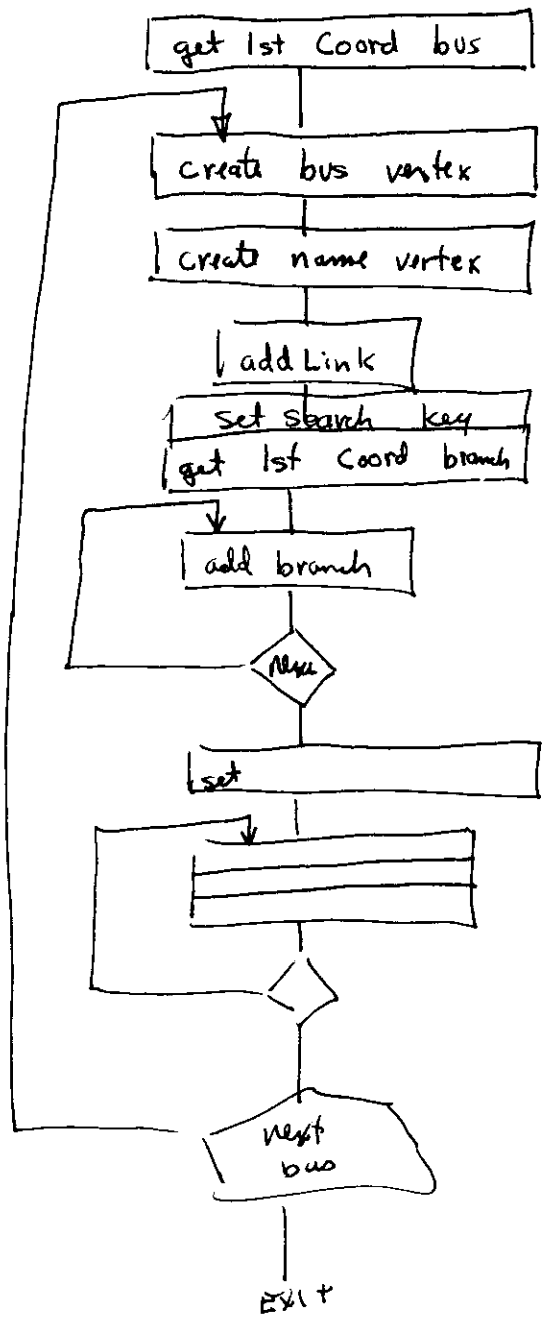
see: psbuildGraphCoord
(Loops thru coord file & calls above to build graph elements)

create from scratch
(create new)

sel: NEW COORD FILE

create graph_db

Process begins when apply_Files calls psbuildGraphCoord (in graphpscor.c), which begins reading the coordinatedb and loading the pscor_db



CreateGraphVertex

Source: graphdata.c

purpose: Add vertex to graph_db

Format:
```
CreateGraphVertex (char *key,
                   GraphClass class,
                   char *namebase,
                   int x,
                   int y,
                   int source,
                   GraphElement **pgraphrec )
```

valid class are: GraphClassVertexBus
GraphClassVertexName
GraphClassVertexBendPoint
} in graph_data.h

* Creating vertexes is a 2-step process.

* Do NOT confuse the following:
  1) Graphelement Vertex
  2) Gadget Vertex (often called graph vertex by Dan)

* Graphelement is ALWAYS created before a gadget.

* A gadget is not neccessarily created after creating a GraphElement

* Creating a gadget "turns on" the graphelement (makes it appear in MOTIF on the user's terminal screen).

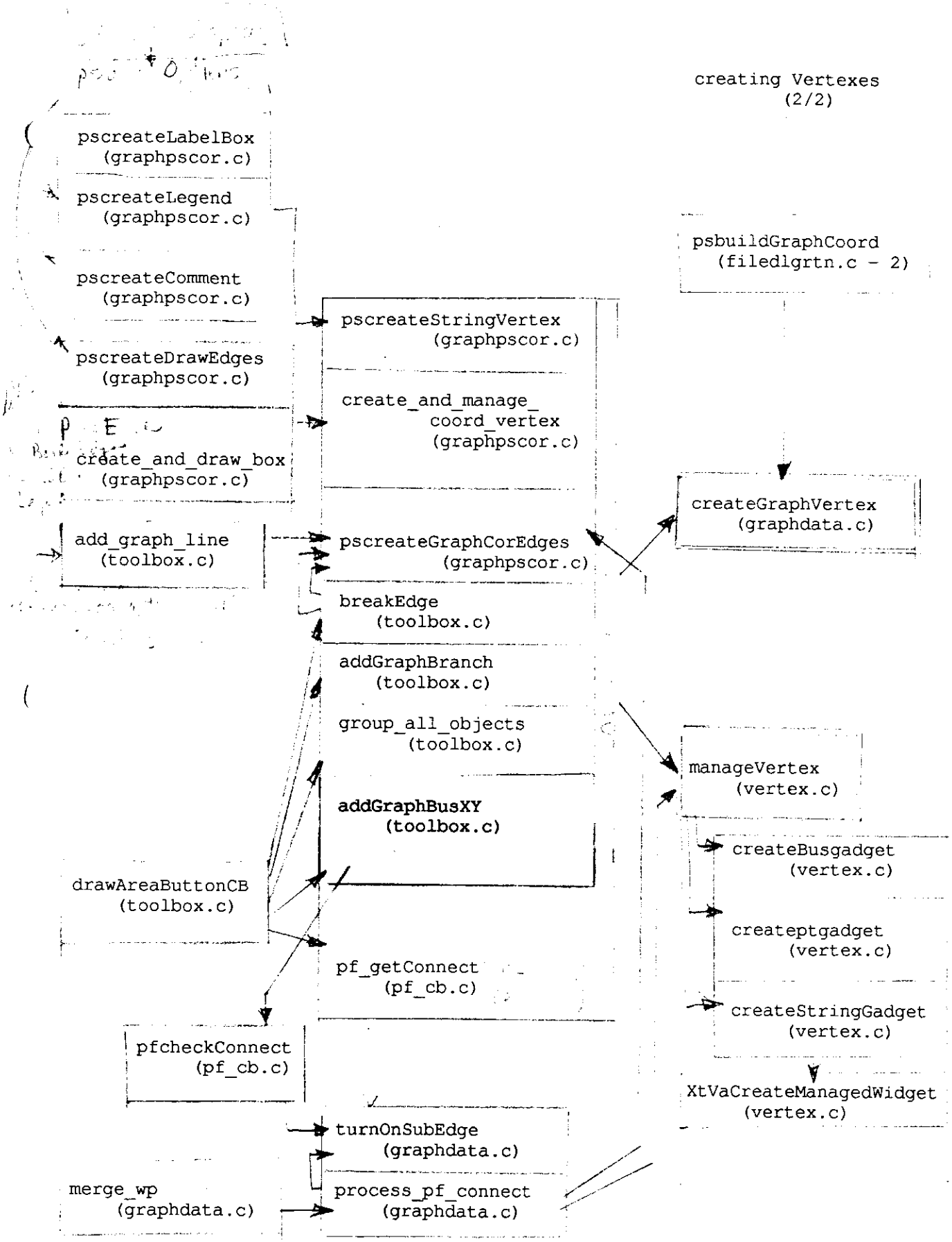
* Destroying a gadget "turns OFF" the graphelement.

* Simplest case - reading new (or replacing old) coord file creates graphelement for ALL coord file vertexes - (BUS, NAME, BENDS) (They are not TURNED ON until the merge.

* Dual case (where both graphelement and gadgets are created):
  explode
  group
  draw
  label
  legend
  comment
  user-created-bus
  breakEdge (new bend)
  drag-line (new branch)

* Other - turn on only occurs when:
  merge
  explode (graphelement already exists)

creating Vertexes
(2/2)



source: createVtx.lst

bend only logic

Create react icon
(from coord file)
method

1) apply_Files

2) psbuildGraphCoord
{ Loop thur Buses

3) createGraphBusXYXY

createGraphVertex (Bus)
creatiGraphVertex (Name)
create Graph GenReac (Gen)
create Graph Gen Reac (Reac)

4) **CreateGraphGen Reac** ([illegible])

add Gen-Bus Link

pscreateGraphCorEdge (edge, gen, bus)

creepage

Coordinate plots tend to creep during conversion from pixels to floating point and vice versa.

When a new coordinate file is saved, the program converts a pixel location to a floating point number, AND then converts that floating pt right back to a pixel location. If this last pixel ~~does~~ value does not match the original pixel value, then <u>corrective action is taken!</u>

see: MOTIF_to_ps.x
MOTIF_to_ps.y
pscoord_to_MOTIF_x
pscoord_to_MOTIF_y
} ingraphpscor.c

current bus

~~retu~~ last bus selected for any reason.

sel: getCurBus (curbus.c)
setCurBus

cursor

Screen cursor on gui is set when toolbar button is pushed.

XmNvalueChangedCallback must call
change_cursor_to (in utils.c)

utils.c has include (ipfcursor.h)

Cursors can be designed using bitmap
/usr/bin/X11/bitmap test

copy results to ipfcursors.h when done

background patter puts a white outline around black cursor.

[illegible] : hot spot
XCreatePixmapCursor
XDefineCursor
change_cursor (utils.c)
ipfcursors.h
[illegible]

cursor

(on [illegible])

event → xbutton, x
event → xbutton, y

location, mouse (cursor)

? data ?

try: get values
set values
text boxes

see:

pscor_db	(coord data)
graph_db	(graph element data)
ai_db	(area/intstic data)
db_valid_db (& db)	

staff test data at:

<u>shrunis/ipf/dat</u>

data checks

see: checks

checks MOTIF data for range, character correctiveness.

must add XmNmodifyVerifyCallback to text widget

data manager

see: dmgr

dbcreate

db

see: dmgr
data base

```
DB_STAT db_create (CONST SCHEMA *sch,
                   CONST long numFields,
                   CONST long partition,
                   DBID, *db)
```

example:

```
# define AI_KEY_CNT (int)(sizeof(ai_schema)/sizeof(ai_schema[0]))
# define AI_PARTITIONS 0
```

```
void ~~[illegible]~~ loadFrea (ai-data.c)
{DBID ai-db;
 DBSTAT stat,
      {
 stat = db-create( (SCHEMA *)ai_schema, AI_KEY_COUNT,
                   AI_PARTITIONS, ai_db);

 if (stat != D_OKAY)
      printf(" Error creating area/interting db \n");
      }
}
```

ref.

```
static SCHEMA ai_schema[] =
{
     { size ..  },
     { Key 1    },
     { Key 2    },
     { Key 3    },
     {          },

}
```

defaults

see: resource file

debug aids

try: printGraphElement (ptr)

display-bus_coord_data (pscordat.c)

display

X Toolkit Warning: Urm__CW_FixupCallback: .....
may imply that GUI did not compile correctly
(Unresolved routine)

debug
hints.

r -server -socketid -debug 1

sets some
debug options

default is 0

sets global flag ipfdebug (from ipfdebug.h)

also see: dbg, dbx gui

use: if (ipfdebug & DB_QuikExitMask)
dmgr debug

also see: core
trapdoor
system test
"multiply defined" error
"unresolved" error

debug MASK

trap door debugging used to test select parts

1> use:
#include ipfdebug.h

2> in any routine.
if (ipfdebug & DB_Mask) .....

3> to activate:
r -debug 2048

mask key

use as a pair

Oct 3 1994 11:42 **ipfdebug.h** Page 1

```
/*
* file of debug flags
*/
extern unsigned long ipfdebug;
#define ipfdbio stderr
#define DB_NoMask           (0L)
#define DB_TraceMask        (1L<<0)  /*    1 */
#define DB_LineTapMask      (1L<<1)  /*    2 */
#define DB_BusSectionMask   (1L<<2)  /*    4 */
#define DB_Toolbox          (1L<<3)  /*    8 */
#define DB_Edge             (1L<<4)  /*   16 */
#define DB_Vertex           (1L<<5)  /*   32 */
#define DB_Filedlgrtn       (1L<<6)  /*   64 */
#define DB_SelfTest         (1L<<7)  /*  128 */
#define DB_GraphPSCorMask   (1L<<8)  /*  256 */
#define DB_BusFilterMask    (1L<<9)  /*  512 */
#define DB_Pf_CbMask        (1L<<10) /* 1024 */
#define DB_QuikExitMask     (1L<<11) /* 2048 */
```

DEFINE

See: send_comments_and_defines_to_pf (graphpscor.c)
refresh_comments (pf_cb.c)
comment (graph comments)

[illegible]

FillDefineList (ipTdef.2.c)

[illegible]

[illegible]

see: deleteGraphBusVertex (namebase)
delete Bus Vertex (ptr)

rm Link (GraphElement, Graph Element, Link, userdata)

ll_detete ( )
db_delete (db, ptr)
rm Back Link ( )
deleteGraphLink ( )
delete Graph Element
destroy Gadgets (0, 0, 0)

outage (delete line or bus)

see:

PUT_DATA, TYPE=COMMENTS

pf_descrip.c

destroy

see: delete

destroyGadgets

source: graphdata.c

purpose: unmanages all visible gadgets in graph_db

Format: destoryGadgets (0, 0, 0)

must be done with care – 3 reasons to destroy gadgets

- load new coord OR base File
- changing from CoordOnly to CoordBase display
- change to blank screen & create from scatch

Care must be taken not to NULL GraphElement wid before the gadget is unmanaged (or destroyed)

see: main
solve dialog
printer dialog
boiler plate
page
plot
save
load
pqcurve
linezcalc

DIALOGS

DIRECTORY OF LIST OF DIALOGS

By order of appearance in VUIT

		figure
xmMainWindow:	gui_main	1
XmQuestionDialog:	file_new_message_dia	
XmQuestionDialog:	create_new_coord_file_dlg	
XmMessageDialog:	exit_warning_box	36
XmFormDialog:	debug_dmgr_dialog	
XmFormDialog:	display_menu_dialog	
XmSelectionDialog:	define_selection_dialog	
XmForm:	bus_type_pics	
XmSelectionDialog:	area_selection_dialog	2
XmFormDialog:	area_interchange_box	2a
XmFormDialog:	area_type_add_dialog	2b
XmFormDialog:	bus_sect_dialog	3
XmFormDialog:	bus_edit_dialog	4
XmFormDialog:	modify_bus_coord_dia	5
XmFormDialog:	cflow_selection_dialog	6
XmFormDialog:	open_file_dialog	8
XmInformationDialog:	command_warning_dia	9
XmFormDialog:	bus_front_box	10
XmFormDialog:	bus_jackets	
XmFormDialog:	save_base_file_error_box	
XmFormDialog:	save_file_dialog	12
XmFormDialog:	save_network_dialog	12a
XmFormDialog:	stability_save_form	12b
XmFileSelectionDialog	file_save_select_dia	
XmFormDialog:	write_protected_file_msg	
XmFormDialog:	help_dialog	13
XmFormDialog:	help_annotate_dialog	13a
XmFormDialog:	error_message_dialog	14
XmWarningDialog:	text_input_error_dialog	15
XmInformationDialog:	unimplemented_feature_box	16
XmFormDialog:	ipc_command_board	17
XmFormDialog:	ipf_report_list_dialog	18
XmFormDialog:	ipf_alpha_bus_list_dialog	19
XmFormDialog:	line_tap_dialog	20
XmFormDialog:	linetap2	
XmFormDialog:	line_Z_calc_dialog	22
XmFileSelectionDialog:	line_z_filesel	23
XmFormDialog:	line_z_save_dialog	24
XmFormDialog:	print_opt_page_dialog	25
XmFormDialog:	plot_options_dialog	26
XmFormDialog:	user_comment_dialog	27
XmFormDialog:	printer_select_dialog	28
XmFormDialog:	select_reports_dialog	29
XmFormDialog:	pf_report_dialog (unused)	30
XmFileSelectionDialog:	reports_file_select_dia	31
XmFormDialog:	solve_dialog	33
XmFormDialog:	bus_help_dialog	34
XmFormDialog:	pf_descp_form	35
XmFormDialog:	nextwork_edit_dialog	
XmFormDialog:	test_continue_dialog	35
	userCancel	40

source: ds5005::/shr5/eohbber/pfi/doc/dialog.lst

Quick refference key:

alpha search	ipf_alpha_bus_list_dialog	19
area	area_selection_dialog	2
area interchange	area_interchange_box	2a
area	area_type_add_dialog	2b
bus edit (coord)	modify_bus_coord_dia	5
bus edit (network edit)	bus_edit_dialog	4
bus edit (pf)	bus_front_box	10
bus front box	bus_front_box	10
bus (help create)	bus_help_dialog	34
bus section	bus_sect_dialog	3
cflow	cflow_selection_dialog	6
command dialog	ipc_command_board	17
comments (powerflow)	pf_descp_form	35
comment (user)	user_comment	27
coord edit (bus)	modify_bus_coord_dia	5
error dialog	error_message_dialog	14
error (text input)	text_input_error_dialog	15
exit gui	exit_warning_box	36
file open (general)	open_file_dialog	8
file(s) save	save_file_dialog	12
file save	save_network_dialog	12a
file select (reports)	reports_file_select_dia	31
front box	bus_front_box	10
gui	gui_main	1
help	help_dialog	13
help annotate	help_annotate_dialog	13a
help (bus create)	bus_help_dialog	34
ipc_command	ipc_command_board	17
ipf_alpha_bus_list_	ipf_alpha_bus_list_dialog	19
line impedence (main)	line_Z_calc_dialog	22
line impedence (file)	line_z_filesel	23
line impedence (save)	line_z_save_dialog	24
line tap	line_tap_dialog	20
main	gui_main	1
network editor	bus_edit_dialog	4
open file	open_file_dialog	8
options (plot)	print_opt_page_dialog	25
options (pf data)	plot_options_dialog	26
plot options	print_opt_page_dialog	25
plot options (pf data)	plot_options_dialog	26
powerflow (case_id)	pf_descp_form	35
powerflow (comments)	pf_descp_form	35
powerflow (description)	pf_descp_form	35
print options	print_opt_page_dialog	25
printer (select)	printer_select_dialog	28
report (file select)	reports_file_select_dia	31
report (list)	ipf_report_list_dialog	18
report (select type)	select_reports_dialog	29
save file(s)	save_file_dialog	12
save file	save_network_dialog	12a
save line impedence	line_z_save_dialog	24
save stability	stability_save_form	12b
select (printer)	printer_select_dialog	28
select (report)	select_reports_dialog	29
solve	solve_dialog	33
stability (save)	stability_save_form	12b
test	test_continue_dialog	35
text_input_error_	text_input_error_dialog	15
unimplemented	unimplemented_feature_box	16
user comment	user_comment	27
warning dialog	command_warning_dia	9
warning (unimplemented)	unimplemented_feature_box	16

move to process an EVENT by calling correct routine:

see MainLoop

XtDispatchEvent - Book 5, pp 133

see: graphBaseOn
graphCorOn
graphMergeOn
manageEdge
turnOnSubEdge
displayOn

see: GraphElement - Display
graph display

display a vertex - see:

display solution

Dave Sygzmanski's data manager routines.

setting up: `#include "dmgr.h"`

⋮

schema:

```
static SCHEMA name[] =
{
    { 'n', 'c', sizeof(AIREC), 0 }
    { 'd', 'c', sizeof(ptr_a_rec -> area),
                offsetof (AREA , area) },
    { 'd', 'c', sizeof(                )

}
```

header file:

also see: data base
SCHEMA

dmgr components

record

key field

SCHEMA

---

key

type

len

offset

keynum

user compare routine ptr

dmgr debug

Under "Process", select "Dmgr Debug Dialog"

Does not work in dbx or tv (total view)

manages "debug_dmgr - dialog"

See autostart.c which enables/disables the ability to push this button

-debug 2048 required to activate this (so users don't get to use it)

buttons	calls
First	get_db_First
Next	get_db_next
Set Search Key	(manage_graphelement_search_dlg
1st Link	get_db_1st_link
Next Link	get_db_next_link
Link to Prev Link	/

dmgr search keys.

Where searching for records, we may want to search for match only in specific fields. These search keys are defined as:

graph_data.h

0 GRAPH_DFN
1 GRAPH_DISPLAY
2 GRAPH_VERTEX_ID
3 GRAPH_TYPE_VERTEX_ID
4 GRAPH_~~DISPLAY~~
5 GRAPH_TYPE_VERTEX_FARVERTEX
6 GRAPH_CLASS_IDX
7 GRAPH_VERTEX_LINK
8 GRAPH_EDGE_LINK

coord_data.h

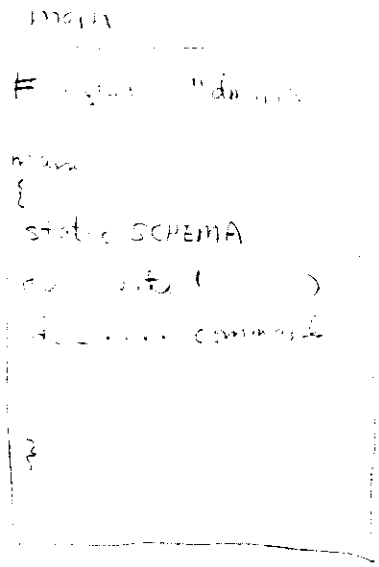
0 COORD_NONE
1 COORD_KEY
2 COORD_NAME1_BASE1
3 COORD_NAME2_BASE2
4 COORD_KEY_NAME1_BASE1
5 COORD_IDX
6 COORD_KEY_IDX
7 COORD_NAME1_BASE1_NAME2_BASE2

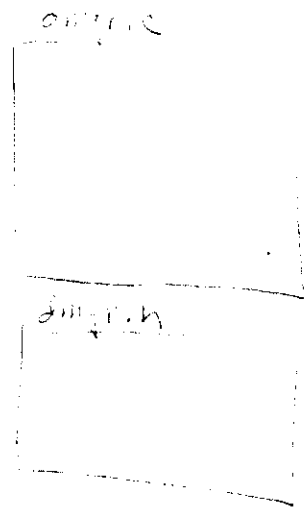
The above numbers coorespond 1-to-1 with the fields as defined in the SCHEMA

some multi-search fields are defined by having sort key longer than field, such that it overruns adjorning fields.

other multi-search are defined by specifiy a subroutine which makes the comparisons.

dmgr setup





[illegible] steps to building a [illegible]

1. [illegible] # include "dmgr.h" [illegible]
2. Build the SCHEMA (see SCHEMA setup)
3. Build the [illegible] array [illegible]
4. use appropriate call to [illegible] that dmgr data (see Data Manager [illegible]

the data manager consists of RECORDS which are sorted & access as a unit.

(fields & keys are discussed in SCHEMA)

MAIN ROUTINES:

toolbox_move_toggle	(toolbox.c)
drawAreaButtonCB	(toolbox.c)
start_rubberband	(toolmove.c)
drag_rubberband	(toolmove.c)
stop_rubberband	(toolmove.c)
draw_outline	(toolmove.c)
XDrawLine	(MOTIF)
XDrawRectangle	(MOTIF)
getobject	(toolmove.c)
setobject	(toolmove.c)
Move_object	(toolmove.c)
move_object_XY	(toolmove.c)
edge_update	(toolmove.c)
update_edge	(edge.c)
getPerimeterPts	(edge.c)
nearend	(toolmove.c)
XtVaGetValues	(MOTIF)
XtVaSetValues	(MOTIF)

FOUR MAJOR AREAS OF CONSIDERATION:

1) Setting up
2) Start drag
3) Dragging
4) Stop drag

MAIN VARIABLES:

global:

object		GraphElement vertex being dragged
BUS_RADIUS		set to 5
xorg yorg		where GraphElement object was picked up
xdelta ydelta	(move_object_)	used to computer final location of object
	(stop_rubberband)	final distance moved from original location
x_cursor y_cursor		last button.event location (set at start, drag & stop)

local:

newx newy	(move_object_)	new location after delta
curx cury	(draw_outline)	1) object widget initial loction 2) farend vertex widget location
	(move_object_)	original widget location (symbol or name)
xdraw ydraw	(draw_outline)	where rectangle (or line) is drawn
height width	(draw_outline)	size of widget (and outline) bus symbol is square, name is rectangle.

D E S C R I P T I O N O F D R A G P R O C E S S

i) CALLBACK SETUP: When widget is created.

ii) SETTING UP: Push the toolbox icon:
Which activate 2 proceedures: toolbox_move_toggle
change_cursor_to (no futhur coverage)

toolbox_move_toggle (toolmove.c)

```
move_status = True              (toggles)
graphics contents are set
XGrabButton event handlers are set
drag_cursor = (cursor changes shape)
```

1) PUSHING ON VERTEX GADGET:

calls: drawAreaButtonCB (toolbox.c)

find_button determines "tb_item" ("tb_Move")

client_data (graphElement) passed from MOTIF clues to the type of gadget selected. (Usually GraphTypeVertex)

setobject
object = GraphElement pushed on

setCurBus = buskey (Name and Kv)

BIG SWITCH statement looks at "tb_item" which skips to "tb_Move"

set up EVENTHANDLER which allows the drag process.

calls START_RUBBERBAND (toolmove.c) passes on the GraphElement

2) START_RUBBERBAND: (toolmove.c)

save location of original pushbutton event

```
xorg            (used in stop rubberband to compute delta)
                (used in draw_outline to compute location)
yorg
```

equates

```
x_cursor = xorg (used later in DRAG for undraw)

y_cursor = yorg
```

draw_outline is called, passing in GraphElement curobject

ButtonMotion Mask is called to set up call to DRAG_RUBBERBAND

3) DRAW_OUTLINE:

from curobject, wid_id is used to get:

```
curx            (gadget's location)

cury

xheight         (from XtVaGetValues)

xwidth
```

```
        location to draw the outline is determined:
                xdraw = xfig -xorg +curx
                ydraw =
        outline is DRAWN ( x, y, height, width )
                xdraw += BUS_RADIUS     (adjust location to draw lines to:)
                ydraw += BUS_RADIUS
        { loop thur vertex links }
                        draw_outline
        { loop thur edge lines }
                        curx = far edge GraphElement x
                        cury = far edge GraphElement y
                        Xdraw_line      ( xdraw, ydraw, curx, cury )
4) DRAG_RUBBERBAND:
        get curobject
        undraw it and related stuff (draw_outline)  (using x_cursor, y_cursor)
                x_cursor = cursor arrow location
                y_cursor = cursor arrow location
        redraw it (draw_outline)
5) STOP_RUBBERBAND:
        undraw curobject & related stuff
                x_cursor = cursor arrow location
                y_cursor = cursor arrow location
        disable ButtonMotionMask and ButtonReleaseMask
                xdelta =
                ydelta =
        call MOVE_OBJECT_AND_ATTHMTS (passing curobject)
                object = null   ensure drag is done ???
6) MOVE_OBJECT_AND_ATTHMTS:
        Get GraphElement location:
                curx =          (where gadget is now)
                cury =
                newx =          (where it will be repositioned)
                newy =
```

```
        Change widget location with XtVaSetValues ( newx, newy )

        Change GraphElement location with newx, newy

                db_update()

        { loop thur vertexes }

                move_object()    (by xdelta, ydelta)

        { loop thur lines }

                updateEdge       (passing in edge GraphElement)
```

7) UPDATEEDGE: (edge.c)

```
        get graphnode1

        get graphnode2

        getPerimeterPts          (x, y, x2, y2)

        rebox data (call line2rect)

        XtVaSetValues ( using rebox data )
```

DRAW CARDS

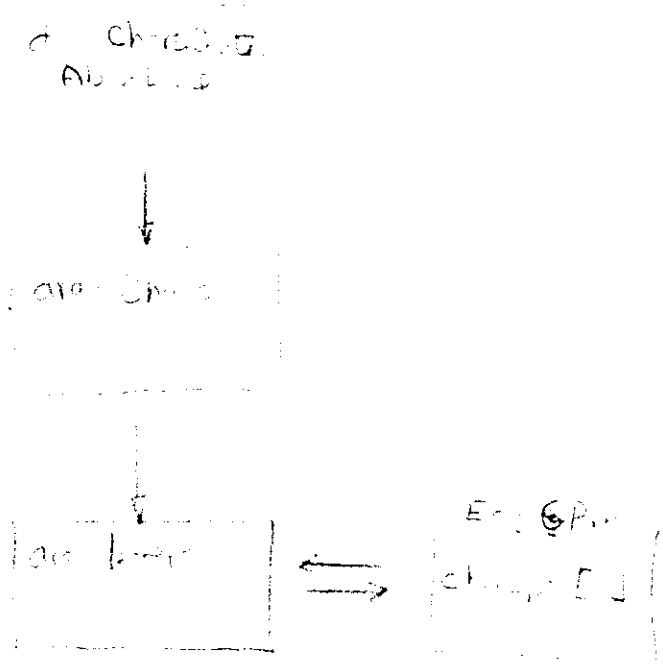
APPENDIX E

DRAW RECORD FORMAT

Draw cards may be used with or without comment cards. They may be used to underline parts of comments, draw a box for a title block, or draw attention to portions of the diagram.

Column	FMT	Description
1	A1	D - Identifies
3-6	F4.2	X - X coordinate of first point on line
7-10	F4.2	Y - Y coordinate of first point on line
11	I1	PEN - Pen position for moving to first X,Y coordinate 0 Ignore this point 1 Move to this point with the pen down (drawing) 2 Move to this point with the pen up Drawn figures may be continued from card to card. Hence, the first as well as all the rest of the points must have the pen position (up or down) specified.
12-20	2F4.2,I1	X, Y, PEN - for next point
21-29	2F4.2,I1	X, Y, PEN - for next point
30-38	2F4.2,I1	XPEN - for next point
39-47	2F4.2,I1	X, Y, PEN - for next point
48-56	2F4.2,I1	X, Y, PEN - for next point
57-65	2F4.2,I1	X, Y, PEN - for next point
66-74	2F4.2,I1	X, Y, PEN - for next point

also see: psCreateDrawEdges
(graphpscor.c)



drawGraphEdge

source:

[illegible] (...c)

[illegible] (graph.[illegible])

[illegible]

DrawLines

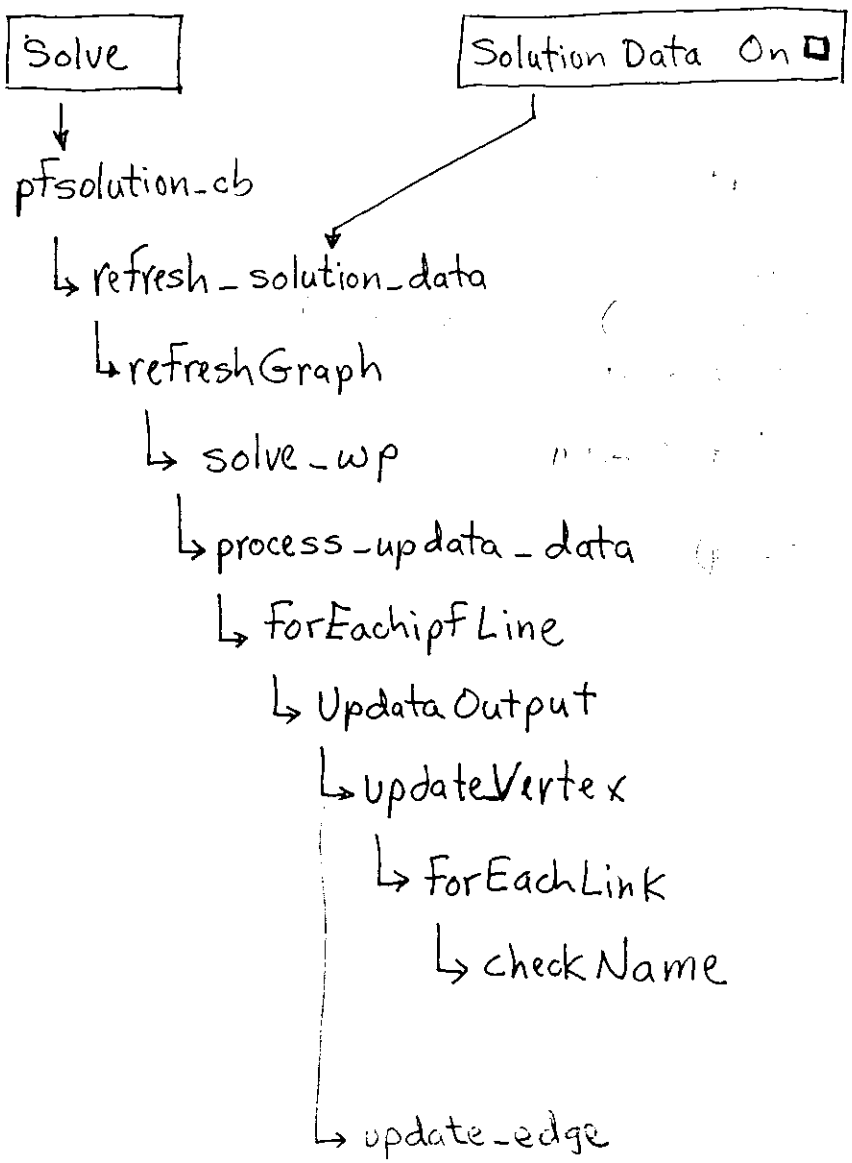
See: addDrawLine (toolbox.c)
pscreateDrawEdges (graphpscor.c)

draw line segments

whenever an <u>expose</u> event happens to a widget which has a line segment (an edge widget) the subroutine <u>Redisplay</u> is called which draws a segment between the two opposite corners.

edge widgets are <u>created</u> by drawline in manageEdge (module edge.c)

draw Vertex data



each

see: ForEachLink

see: FindGraphEdge

edge.c

one of two graph element components
(vertexes & edges)

also see: complex edge
sub edge
pscreateGraphCorEdge

collection of routines to handle
the edge widgets.

Note: No segments are drawn inside these
box widgets at this time.

EdgeG.c makes line segment appear
whenever an expose occurs.

EdgeG.c
(now edgeg.c)

this module won't compile correctly on VMS? machine

add following line to bottom of make file.

```
"gui.m" 74 lines, 1115 characters
clean:
        rm -f *.o *.uid  core $(IMAGE)

#
# new suffix rule for compiling uil files.
#

EdgeG.o:
        $(CC) $(CINCLUDES) $(DEFINES) -c $*.c
.uil.uid:
        $(UIL) $(UILFLAGS) $(UILINC) -o $*.uid $<


# DO NOT DELETE THIS LINE -- make depend depends on it.
~
~
~
~
~
~
~
```

must be TAB!

add

Bug in C89 compiler, get rid of -std & -g options and compiler works fine.

C89 ~~-std~~ -g

Resources of each point

ine: EdgeGraph.h

_EdgePoint

inputCallback

inputCallbackDone

D

x

y

x2

y2

lineIntersect

Error

B

drawOn

B

symbolSegment ←

drawOn

anyDrawOn

textNumberOn

P

drawPixel

P

drawPoint

P

lineType

P

pointOn

X

label

textLabel

GG

drawGC

GC

lineGC

P

EdgePoint

Source: graphdata.c

purpose: moves widget from x, y to x2, y2

Format: edgeupdate (widget, x1, y1, x2, y2)

edit coord Vertexes

display_bus_coord_data

See: display_coord_data (pscordat.c)

purpose: puts an error message in a dialog box for IPF users to see

usage:

```
# include "em.h"
em_init
:
char   errmsg [133];
:
:
:
sprintf ( errmsg,  "Error in routine \n");
err.line  = EM-LINE;
err.msg  = errmsg;
err.link  = "creategraphtbl";
err.type  =  WARNING;     (or INFO, or FATAL)
err.ident =
:
em_show ( & err);
:
:
:
```

error dialog box

see em

error messages

see em

debug

unresolved error

EVENT is some action caused by user interaction with programe. see Vol 5 pag 435-490

see: KeyPress } keyboard
KeyRelease }
ButtonPress } mouse button
ButtonRelease }
MotionNotify – mouse move
EnterNotify } arrow enters & leaves window
LeaveNotify }
FocusIn } highlights window Frame as "active"
FocusOut }
KeyMapNotify
Expose – causes redraw
GraphicsExpose – e.g. new dialog
NoExpose
VisibilityNotify
CreateNotify
DestroyNotify
UnmapNotify – e.g. close dialog
MapRequest
ReparentyNotify e.g. new dialog box
ConFigureNotify e.g. change window size
GravityNotify
ResizeRequest
CirculateNotify
CirculateRequest see: Mar. Lay.
PropertyNotify
SlectClear
SelectionRequest
SelectimNotify
ClientMessage
MappingNotify

event loop

see: Main Loop

exists

See

db-val.d_db - for dump files

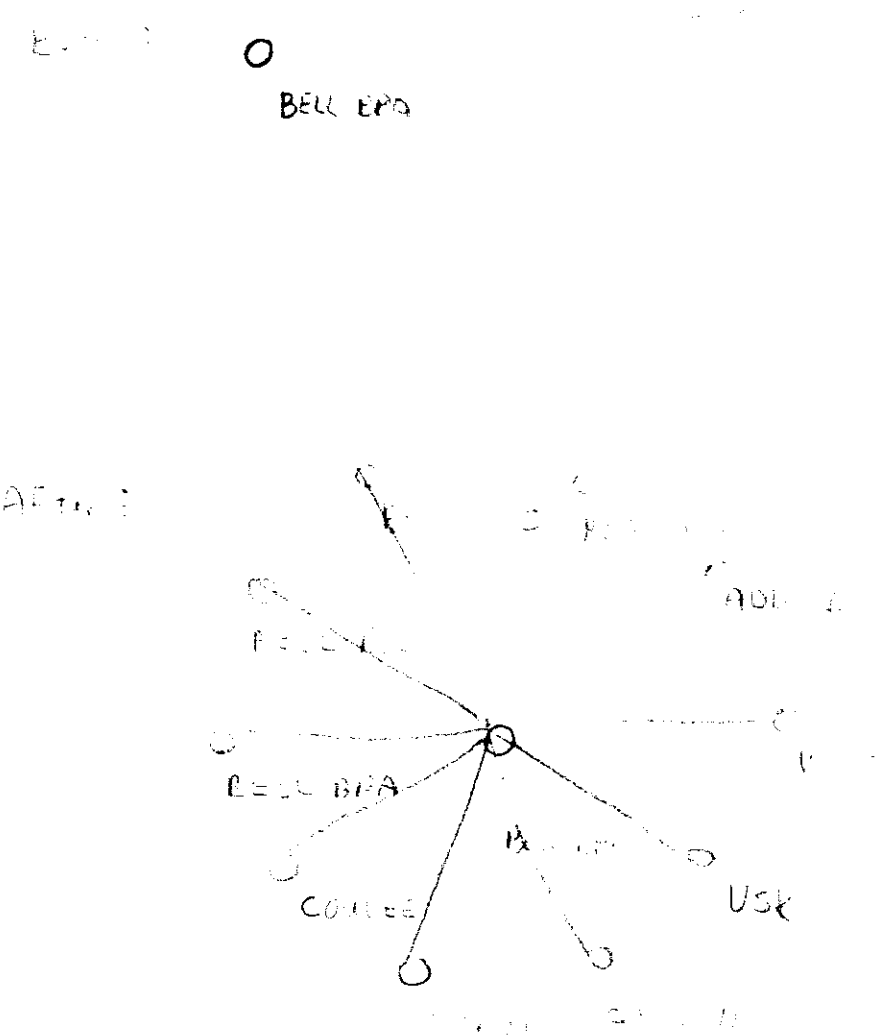
pt_file_exists

File_exists

/get_data/type=bus_exists

explode

This situation occurs when "explode" toolbox icon is active, and user clicks on an existing bus. GUI will look up (from powerflow) all missing lines which are not drawn on screen, create the adjacent lines and buses.



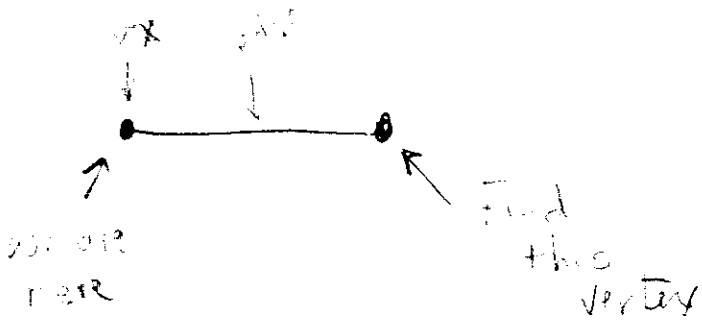
expose

also see: draw line segments

<u>expose</u> is any event which covers or uncovers any (or all) portion of a display box:

Far end

Sec: Find far vertex (graph data)



Fetch

used in vuit_main_template_c
(which eventually go to gui.c file)

Fetch_widget ("save_file_dialog")

Keeps dialog from appearing on screen until needed.

also: assures they get registered promptly - or program will bomb.

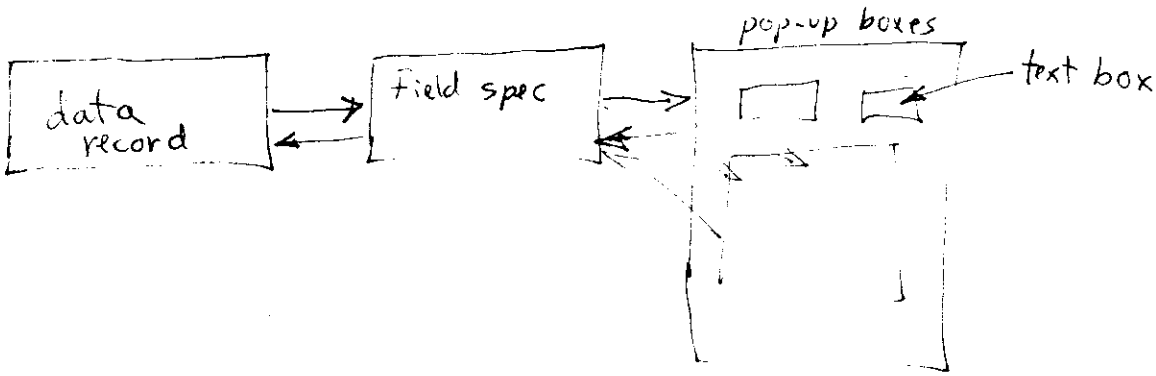
Normally dialog is not registered until it appears on screen. If C-code tries to access any widgets in dialogs not registered, program bombs.

also see: Fetch_File_windows (Filedlgrtn.c)
Fetch_print_windows (printopts.c)

see: text Field behavior
Field_spec
checks

Field_spec
(selection.c)

This scheme is used to minimize programming errors. It ensures that "one-fix-fixes-all" and eliminates the need to keep two mutually section of code working together.



field spec takes ANY data record, ~~cuts~~ ~~spe~~ Finds the type, seperates the fields and xfers it to it correct text box. It ALSO takes records from pop-up boxes, retrieves all the text box data and <u>builds</u> a <u>data record.</u>

It handles 12 bus-type records, continuation records 16 branch-type records, 3 types of area control/interchange records.

These routines are SMART ENOUGH to find the type of record that is being processed and take appropiate action. to build or dissemble data.

File opening

[illegible]

see: bosFilter.c

Find

see: FindGraphEdgebyNames (by bus names)

Findgraphedge (by x, y)

~~FindNameVertex~~

FindBusGraphVertexRec (by name)

FarendGraphVertex (vertex, edge, far_vertex)

FindEdgeVertexes

Findnextedge (

Findnextvertex

FindFirstcoordbusrec

FindFirstbusrec

FindVertexLinksToVertex

[illegible] NameVertex

deleteEdgebyName

[illegible] VertexbyName

turn[illegible] EdgebyName

FindFirst

see: psFindFirstcoordbusrec (pscordat.c)

see: FindFirstbusrec (base_data.c)

see: FindFirstcoordbusrec (coord_data.c)

FindFirstcoordbusrec

(~~base.data.c~~)
(pscordat.c)

routine to start reading bus records from coord db.

FindGraphEdge

source: graphdata.c

purpose: given 2 bus names & voltage returns a pointer to the edge in graph_db

Format:

Findgraphedge

source: edge.c

purpose: Find edge nearest to x,y

Format findgraphedge (int x, int y, GraphElement **pFound)

source: graphdata.c

purpose: Find next edge connected to this edge

Format: Findnextedge (GraphElement vertex,
GraphElement edge,
GraphElement nextedge)

Source: graphdata.c

purpose: find far end of edge

Format: Findnextedge (GraphElement, edge,
GraphElement, vertex
GraphElement nextvertex)

First

see: Find first

Flow segment

graph-data.h has dply-seg

complex edge, dply-seg = sub edge that line Flow data appears.
(counting from bus vertex which is lowest alphabetically)

sub-edge, dply-seg = 0 not flow segment
= 1 flow segment

Flow segment is the SUB EDGE which Flow data, ARROW (and) TRANSF SYMBOL will be on.

also see: arrow

xlsfonts

gives list of fonts available for a particular terminal

/usr/bin/X11/xfontsel

is program that lets you play with the settings

Font

(Coord File)

"p" record in coord file is a font specification & is treated similiar to a <u>Comment</u> card.

see: [illegible]

ForEachLink

source: graphdata.c

```
ForEachLink ( db,
              pel,
              GRAPH_VERTEX_LINK,
              rmLink,        ⇒ routine to call
              *pel )
```

Foreground colors

see: SetNormalGC (drawgodget.c)
getForegroundGC (vertex.c)

see: cursor location

see: cursor location

GET DATA

3.5 GET_DATA

This command with its many different forms fetches data from the Powerflow process. It calls `p_gtdata.f` with the following parameters.

```
integer function p_gtdata (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains any of the following commands.

```
/GET_DATA, TYPE = A_DATA
/GET_DATA, TYPE = AREA_DATA
/GET_DATA, TYPE = AREA_LIST
/GET_DATA, TYPE = BSEKV_LIST
/GET_DATA, TYPE = BUS_EXISTS
/GET_DATA, TYPE = BUS_LIST
/GET_DATA, TYPE = BUS_VOLTAGES
/GET_DATA, TYPE = COMMENTS
/GET_DATA, TYPE = CONNECTION
/GET_DATA, TYPE = COUNT
/GET_DATA, TYPE = FILE_EXISTS
/GET_DATA, TYPE = I_DATA
/GET_DATA, TYPE = INITIALIZE_DEF
/GET_DATA, TYPE = INPUT
/GET_DATA, TYPE = LINE_IMPEDANCE_CALCULATION
/GET_DATA, TYPE = LOAD_AREA
/GET_DATA, TYPE = LOAD_DEFINE
/GET_DATA, TYPE = LOAD_REF_AREA
/GET_DATA, TYPE = LOAD_REF_BASE
/GET_DATA, TYPE = NETWORK_DATA
/GET_DATA, TYPE = OUTAGES
/GET_DATA, TYPE = OUTPUT
/GET_DATA, TYPE = OWNER_LIST
/GET_DATA, TYPE = RECORD_LIST
/GET_DATA, TYPE = REF_AREA_DATA
/GET_DATA, TYPE = REF_OUTPUT
/GET_DATA, TYPE = SOL_PAR
/GET_DATA, TYPE = STATUS
/GET_DATA, TYPE = SUB_DEFINE
/GET_DATA, TYPE = SYSTEM
/GET_DATA, TYPE = ZONE_LIST
```

The routine `p_gtdata.f` parses these commands and calls a subroutine to perform the specific task, according to the type of data indicated.

GET_DATA, TYPE = A_DATA

This command retrieves in `out_buffer` all type A input data records in WSCC format. It calls

GET_DATA
SOL_PAR

GET_DATA, TYPE = SOL_PAR

This command obtains solution tolerances, controls, or switiches which influence the proecssing of the case in residence.

The obtained system data is idential to the set of data modified by the related command `/SOLUTION`

The returned values are encoded in the character array `in_buffer` in free field, C-formatted strings. The quantities enclosed in angle brackets "< ... >" denote variables quantitied, < status > denotes a logical on or off, < value > denotes an integer, floating point, or character quantity.

```
/GET_DATA, TYPE = SOL_PAR,
> AI_CONTROL = < value >          { CON | MOD | OFF }
> BASE_SOLUTION = < status >
> DEBUG_TX = < status >
> DEBUG_BUS = < status >
> DEBUG_AI = < status >
> DEBUG_DC = < status >
> LIMITS_QRES = < value >
> LIMITS_PHA = < value >
> LIMITS_DA = < value >
> LIMITS_DV = < value >
> LTC = < value>                  { ON | ON_NV | ON_NPS | OFF | ON_DCONLY }
> MISC_XBUS = < value >           { BPA | VMAX | WSCC }
> MISC_DCLP = < status >
> MISC_VFLAT = < status >
> MISC_TSTART = < value >
> MISC_ITER_SUM = < status >
> MISC_PHA_SHIFT_BIAS = < value >    { BPA | WSCC }
> SOL_ITER_DECOUP = < value >
> SOL_ITER_NEWTON = < value >
> TOL_BUSV = < value >
> TOL_AIPOWER = < value >
> TOL_TX = < value >
> TOL_Q = < value >
```

return status: status = 0 : success
1 : errors

########

use # each time the "solution" dialog is popped up to set (initialize) values according to what is in power flow

A.4 PROCESSES

SOLUTION

############ made changes

This command causes the Powerflow process to solve the currently resident base case. It calls the FORTRAN module `p_solton.f` with the following parameters.

```
integer function p_solton (in_buffer, out_buffer)
parameter (MAXBUFFER = 6600)
character in_buffer * (MAXBUFFER)
character out_buffer * (MAXBUFFER)
```

The character array `in_buffer` contains the following information. Normal defaults are shown, optional items are in [ ].

```
/SOLUTION
[ > AI_CONTROL = CON ]
[ > BASE_SOLUTION ]
[ > DEBUG,TX=OFF,BUS=OFF,AI=OFF,DCMODEL=OFF ]
[ > LIMITS,QRES=<value>,PHA=<value>,DEL_ANG=<value>,DEL_VOLT=<value> ]
[ > LTC = ON ]
[ > MISC_CNTRL, -
    X_BUS = BPA, -
    DCLP = ON, -
    VFLATSTART = ON, -
    TSTART = 0.5, -
    ITER_SUM = OFF, -
    NUMVSTEPS = 3, -
    PHASE_SHIFTER_BIAS = BPA ]
[ > SOL_ITER, DECOUPLED = 2, NEWTON = 30 ]
[ > TOLERANCE, BUSV = 0.005, AIPOWER = 0.001, TX = 0.001, Q = 0.005 ]
```

~~enhan~~ enhance solution command to include all new parameters

```
##############################
```

get text data

see: lookup_and_get_field (in selection.c)
checks

text boxes

XmText

get values

XtVaGetValues ($\underbrace{\text{, ptr ,}}_{\text{item}}$ $\underbrace{\text{, ptr ,}}_{\text{item}}$ NULL)

Note: Null value [illegible]

examples:

```
XtVaGetValues (edit_item[cnt].list, XmNselectedItemCount,
               &selcnt, XmNselectedItem, &strings, NULL)
```

see: XtVaSetValues (, , , , , NULL)

obsolete

now called graph-base-obsolete.c

graphBaseOn

Source: graphdata.c

purpose: displays all coord data ANDed with base data <u>PLUS</u> extra line data (no. coord lines displayed)

graphcor.c

obsolete routine

replaced with graphpscor.c

graphCorOn

(graphdata.c)

source:

purpose: display only graphdata from Coord File

(This routine sets the display flag) of all vertexes and edges that come from the coordinate file.

Format: graphCorOn (0, 0, 0)

graphdata.c - collection of support routines

See: findGraphEdge
createGraphVertex
turnOnSubEdge

<u>graphelement</u>

psbuildGraphCoord (in graphpscor.c)

Graphdisplay (OFF)
(On)
(RequestOn)

This is a key set in GraphElement which indicates of the coorresponding gadget exists (visible on screen)

createGraphVertex – initially sets it to GraphDisplay OFF
pscreateGraphCorEdge – " " " " "

RequestOn set: pscreateStringVertex (graphpscor.c)
(when graphVertexes are created)
creati-and-manage-coord-vertex
" " " " edge

pF-checkConnect (pF-cb.c)
(when new edges are created)

addGraphBranch (toolbox.c)
~~vertex.c~~
lookat at: ~~create Vertex Endpts~~ (vertex.c)
manageEdge (edge.c)

graph display

GraphDisplay OFF 0

(indicates graph element is not a managed widget)

- create Graph Vertex sets
- ~~graph elements are created with~~ GraphDisplay OFF
- destroy Gadgets sets GraphDisplay OFF
- turnOff GraphElement sets "

GraphDisplay On 1

- turnOnSubEdge set (2)
- create Bend Array ck
- display On
- graph Bend Toggle Label
- Switch edge

(indicates graph element has been managed)

GraphDisplay Request On 2

(

turnOff Element

graph_db key → #include "graph_data.h"

display	level	type	class	x	y	wid	primitive	attributes	transformation matrix	source	vertex_id	forvertex id	id	style	name1	base1	name2	base2
i	i	i	i	i	i	W	i	i	i	i	char [12]	char [12]	c	c	char [8]	[4]	[8]	[4]

coord (CoordKey)

type	subtype	chgcde	owner	name1	base1	zone
c	c	c	[3]	char [8]	[4]	

pfbus (PFBusKey)

type	subtype	chgcde	owner	name1	base1	meter	name2	base2	circuit	section
c	c	c	[3]	char [8]	[4]	c				

pfbranch (PFBranchKey)

id	display	name	base	abbr	v-flag
c	c	char [8]	[4]	[8]	c

coordbus (CoordBusKey)

```
1 GraphElement  graphrecord
2 GraphElement  *pgraphrecord = &graphrecord
3 pgraphrecord = &graphrecord;
```

examples of usage:

```
strncpy(pgraphrecord -> vertex_id, name.base,
        sizeof(pgraphrecord -> vertex_id));

graphrecord.display = GraphDisplayOff;
graphrecord.x = x;

strncpy( pgraphrecord -> key.coord.id, key,
         sizeof ( CoordKey ));
```

GRAPHELMENT

GRAPHELEMENT is a <u>record</u> from the graph-db, and has a format as defined by graph-data.h. To access the data, a pointer is obtained to the record, then with a typedef statement, the record can be broken down into the following components.

graphrec.display	key whether or now item is visible on screen
. level	normally "0"
. type	
. Class	,
. x	integer (pixels)
. y	integer (pixels)
. Widget	int
. primates	
. attr	
. trans Matrix	
. source	
. vertex_id	12-character name (bus) or branch)
. farvertex_id	12-charater name (branch) only)
. ~~id~~	

organized according to 1 of 4 different types.

GRAPHELEMENT

(accessing data)

Procedure to access any GraphElement

```
#include graph-data.h
void create
{
GraphElemint graphrecord;        /* define & allocate space for data */
GraphElement *pgraphrecord;      /* allocate space for p/r           */

    graphrecord.class = 0;                                  (int or char)
                                                            (byte)
    strncpy (pgraphrecord -> vertex-id , name bus, 12)      (string)

    strncpy (pgraphrecord -> key.coord.il ,).....)          (string to a sub-
                                                            template)
                  ↑
                - or - pfbus
                - or - pfbrnch
                - or - coordbus

}
```

also see: printGraphElement

GRAPHELEMENT
(class)

XXX.class =	(0)	GraphClassVertexBus
	(1)	GraphClassVertexName
	(2)	" BendPoint
	(3)	" Generator
	(4)	EdgeSection
	(5)	~~Edge~~ ComplexSection
	(6)	~~Edge~~ SubSection
	(7)	" VertexGroup

GRAPHELEMENT

(create edge)

GRAPHELEMENT edges are always created by a call to pscreateGraphCorEdges (in graphpscor.c)

GRAPHELEMENT

(create) vertex

GRAPHELEMENT vertexes are always created by a call to <u>create GraphVertex</u> (in graphdata.c).

The following data must be supplied to create a vertex

1) class (Graph Class Vertex Bus
Graph Class Vertex, Name, etc.

2) vertex_id (12 char name & base)

3) x, y (in MOTIF ref frame - pixels)

4) source (GraphSource Coord,

5) key ("B" or "L" char from coord file

GRAPHELEMENT

(display)

ge.display [illegible] 0 Graphelement OFF
1 Graphelement On
2 Graphelement [illegible]

display is OFF when first created.

GRAPHELEMENT
(links)

almost all vertexes & edges have links to another vertex or edge. When a buse vertex is created, a name vertex is also created AND two links ~~are~~ created – one points to the other.

Edges have links to vertexes, and vertexes have links to other edges. This system is vitial to ensuring that when buses or lines are moved on screen, the edges maintain their connectivity.

links are established with the call:

addGraphLink (rec1 $\xrightarrow{\text{(points to)}}$, rec2);

where rec1, rec2 are ptrs to graph elements

also see:

- replaceGraphLink
- deleteGraphLink
- addGraphLink

Graph Element print
(debug aid)

Source: graphdata.c

Format: printGraphElement (ptr)

---

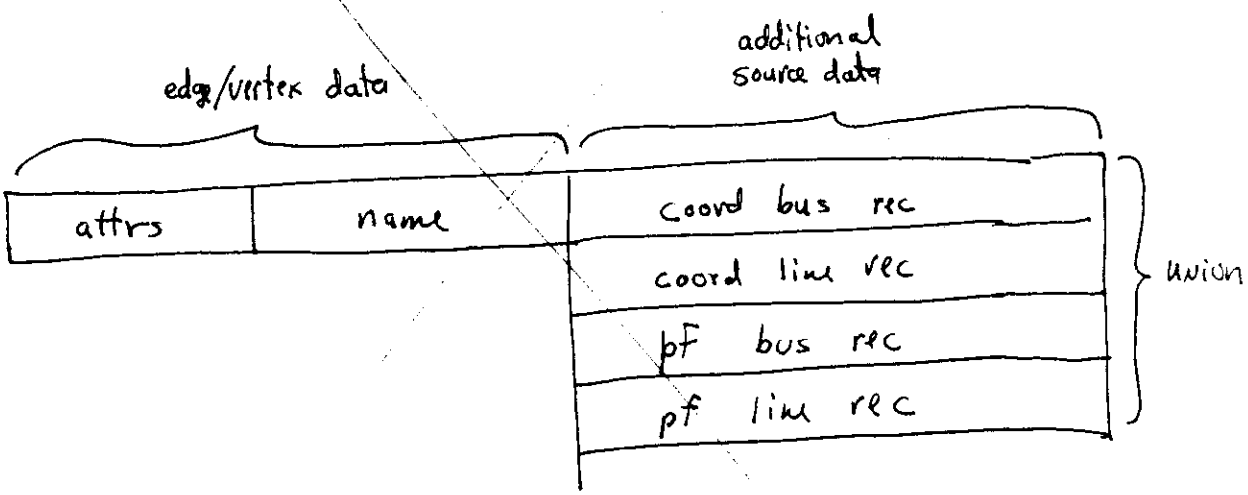
draw_outline (display, window, gc, ptr)

GRAPHELEMENT

(search logic)

GRAPHELEMENT
(structure)

The graphelement structure is defined by graph_data.h and has 4 possible formats. All graphelement records are contained within graph_db. By manipulating the graph_db, the changes are supposed to be automatically updated on the screen.



Filling or creating a new element requires filling in the two halves - the 2nd half has a structure that depends on the source & type of data.

Graph Element
(type)

xxx.type	=(0)	Graph Type Vertex
	=(1)	" Edge

GraphElement

(print)

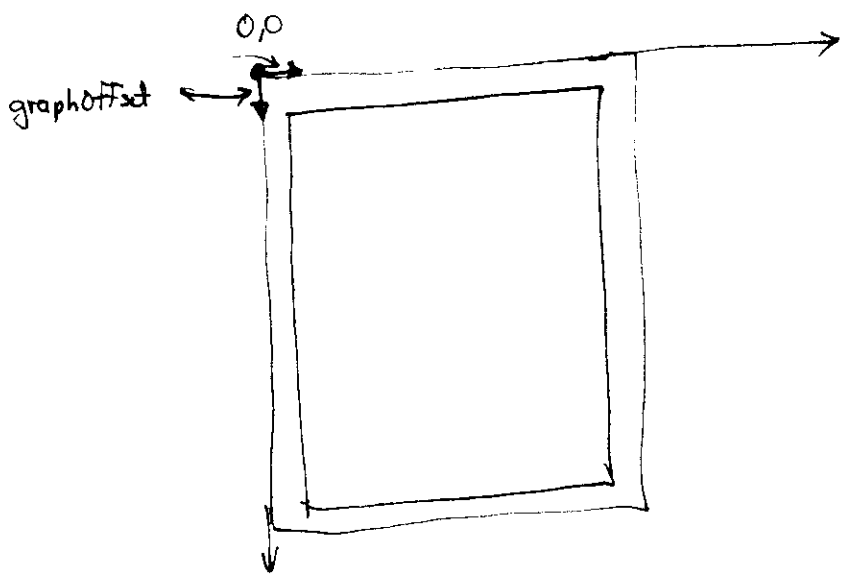
see: printGraphElement

source: graphdata.c

purpose:

format: graphMergeOn (0,0,0)

This variable is used to prevent clashes with the screen plots. Usually the border drawn around the plot will cause a bomb if the value goes negative.



do not confuse with <u>offset</u> !

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

graphOffset is the <u>final</u> process after ~~scaling and~~ offsetting and scaling.

(Its also the first process adjustment before saving a .cor file.)

see: offset
border

gray out

set: sensitive

disable_pushbutton (utils.c)
enable_pushbutton

group

a group of bus nodes that are moved together as a group on screen.

GUI

(params)

GUI -servername (/shrunis/exe/ipFsrv)

-socketid (1040)

-autostart (+)

-display (5005:0)

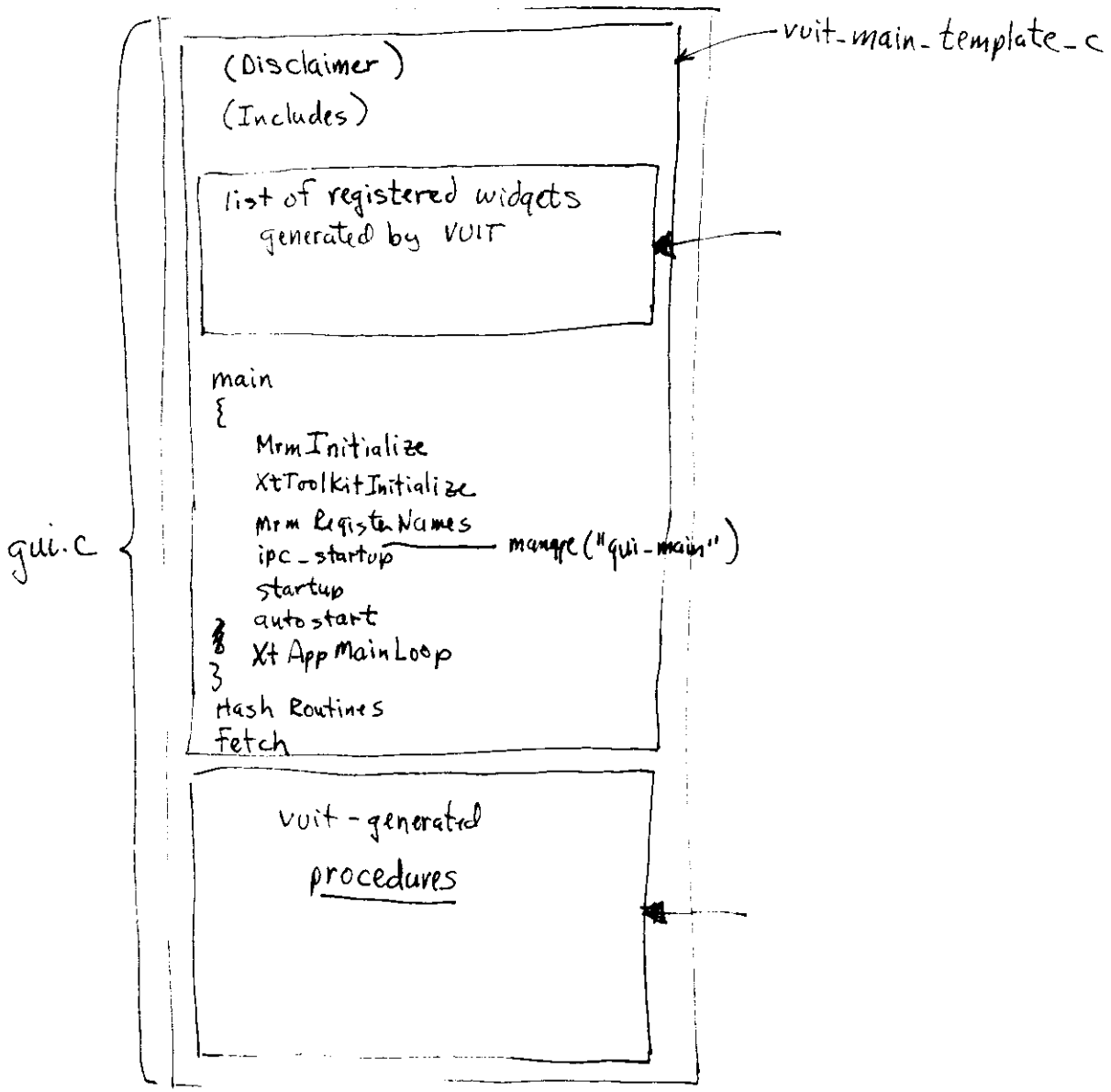
-autostart

-fg (blue)

-debug

This file is normally generated by VUIT under the menu bar item Activities, Generate Application Files, in C. The "Generate Build Procedure" button must be UP (not shaded), with the following boxes filled as:

Application File Name: gui.c
Include Editor File Name: gui.h
Build Procedure Name: make file



start

↓

MrmInitialize (Prepare Mrm Library - See pp 656 book CA)

XtToolkitInitialize (Top level widget)
XtCreateApplicationContext
XtOpenDisplay

XtCreateShell

getXtResources (Read Resource file XGUI)

MrmOpenHierarchy (open UID file "gui.uid)

MrmRegisterNames (register all WIT names from UID)

Manage "gui_main"
XtRealizeWidget } bring up main window

ipc_startup()
startup()
autostart()
← actions performed w/ user keystroke

XtAppMainLoop (user mouse & clicks now cause action

Routines called by GUI.C procedures

GUI: F.

2/10/96

	C-ROUTINE TO CALL:	OFFICAL VUIT NAME:
1	alpha_check	***
2	alphanum_check	***
3	alphanum_sp_check	***
4	apply_files	***
5	apply_files	***
6	cancel_bus_settings	***
7	cflow_debug_cb	***
8	cflow_kill_cb	***
9	cflow_launch_cb	***
10	change_cursor_to	***
11	change_print_plot_opts	***
12	clear_solution_data	***
13	cor_selection_edit	***
14	create_cont_record	***
15	create_dc_2_term_rec	***
16	create_dc_multi_term_rec	***
17	create_equiv_rec	***
18	create_equiv_rec	***
19	create_from_scratch	***
20	create_line_rec	***
21	create_pq_record	***
22	create_reac_rec	***
23	create_regxfmr_rec	***
24	create_xfmr_rec	***
25	creategraphtbl	***
26	data_check	***
27	decimal_check	***
28	digit_check	***
29	draw_area_input	***
30	edit_apply	***
31	edit_bus	***
32	edit_bus_close	***
33	edit_init	***
34	edit_reset	***
35	edit_send_to_pf	***
36	error_dia_help_cb	***
37	exit_ipf	***
38	exit_ipf	!***VUIT_Action
39	exit_ipf_quick	***
40	file_check_and_save_cb	***
41	file_default_set	***
42	file_name_set	***
43	file_save_cb	***
44	fill_area_selection_box2	***
45	fill_branch_jacket_cb_sb	***
46	fill_bus_dialog_cb	***
47	get_bus_alpha_select	***
48	get_bus_selection	***
49	graphBendToggleLabel	***
50	help_annotate_get	file_save_proc;
51	help_annotate_get	***
52	help_annotate_remove	***
53	help_annotate_save	***
54	help_dialog_page_down	***
55	help_dialog_page_up	***
56	help_expose_callback	***
57	help_file_name_set	***
58	help_input_callback	***
59	ipc_commandString_xtoc	***
60	ipf_alpha_srch_value_chg	***
61	ipf_bus_list_select	alpha_bus_list_select;
62	line_pq_edit_delete	***
63	line_pq_edit_insert	***
64	line_pq_edit_replace	***
65	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      ***
```

```
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
```

gui.h ≡ vuit_include_template_c

File which "makes" the gui executable

Format: make -F gui.m

see make

GUI starts it's execution (after typing 'GUI') in gui.c

Below are the major highlights (in gui.c) of this startup:

```
main(   )
{
     ..
     ..
     ..
  MrmInitialize
  XtToolkitInitialize          (Read resource file)
     ..
  Version #                    (version # appears here)
     ..
     ..
     ..
  VUIT_manage("gui_main");     (makes MAIN dialog appear on screen)
     ..
     ..
  fetches.......               ('registers' some dialogs)
     ..
     ..
     ..
  ipc_startup();               (starts the interactive powerflow (ipf)
  start();
  autostart();
     ..
     ..
     ..
  XtApplMainLoop();            (starts MOTIF process)
     ..
     ..
     ..
}
```

generated during a <u>make</u>

NOTICE: if gui.uid exists, then new gui.uid is <u>not</u> generated.

Always delete gui.uid every time a .u File has been changed.

UID =

Contains mostly the includes for all the .u files required to define the MOTIF stuff for GUI.

(list of gui.uil file as of Jun/1994)

```
module pfi
    names = case_sensitive
    version = "v1.0"

include file 'main.u';
include file 'procs.u';
include file 'values.u';
include file 'acbusform.u';
include file 'aclineform.u';
include file 'areaselect.u';
include file 'bussectn.u';
include file 'buseditn.u';
include file 'buscoord.u';
include file 'cflow.u';
include file 'contform.u';
include file 'dcbusform.u';
include file 'dclineform.u';
include file 'equivform.u';
include file 'fopendia.u';
include file 'frontdia.u';
include file 'fsavedia.u';
include file 'help.u';
include file 'ipcwindow.u';
include file 'ipfreport.u';
include file 'linetap.u';
include file 'linetap2.u';
include file 'linezcalc.u';
include file 'pixmaps.u';
include file 'pqcrvform.u';
include file 'printopt.u';
include file 'regxfmr.u';
include file 'reportdia.u';
include file 'solve.u';
include file 'swrreac.u';
include file 'xfmrdia.u';
include file 'bushelp.u';
include file 'pf_descrip.u';
include file 'systemtest.u';
end module;
```

Goal: generate a location in a 2000 integer array for
the name 'gui_main'

Step one: call to hashFunction copies name into locname array

```
locname.charname = 'g' 'u' 'i' '_' 'm' 'a' 'i' 'n'
locname.intname  =  67  75  69  5F  6D  61  69  6E
```

namelen is determinded to be 8 characters long.

namelen is shifted right by one bit to reduce the 8 to 4.

namelen is ANDs 8 with 1 to get the remainder namextr = 0.

Loop namelen (4) times to convert pairs of letters to integers:

```
'gu' = {6775}            (actual memory allocation)
        7567             (hexidemical equivalent  )
       (30055)           ( base 10 integer        )

7567 << 0 = 7567         (shift left by 0 )
7567 EOR 0 = 7567        (EOR to last number )
```

2nd iteration:

```
'i_' = {695f}

5f69 << 1 = bed2         (shift left by 1)
7567 EOR bed2 = cbb5     (EOR to last number)
```

3rd iternation

```
'ma' = {6d61}
616d << 2 = 185b4        (shift left by 2)
cbb5 EOR 185b4 = 014e01  (EOR to last number)
```

4th iteration:

```
'in' = {696e}
6e69 << 3 = 037348       (shift left by 3)
014e01 EOR 037348 = 023d49      (EOR to last number)
```

If there HAD been an ODD number of characters in the name:

the last character is removed by AND'ing with 00FF
then EOR'd same as above.

Final adjustments:

```
023d49 AND 7fff = 3d49       (AND'ed with 7FFF)
```

Dividing by array size:

```
3d49 / (2000) = (15689)/(2000) = 7.8445     (Divide by array size)
```

Remainder is .8445

```
2000 * .8445 = 1689
```

Thus the hashFunction indicates that the data should go at

location number 1689 (out of 2000).

hash lookup

quick lookup talbes & method to find widgets
routines are contained in vuit_main_template_c
which, in turn are put in the gui.c
during the VUIT generate process.

code is part of <u>vuit_main_template_c</u> &

includes the routines HashLookup
Hash Function
Hash Register

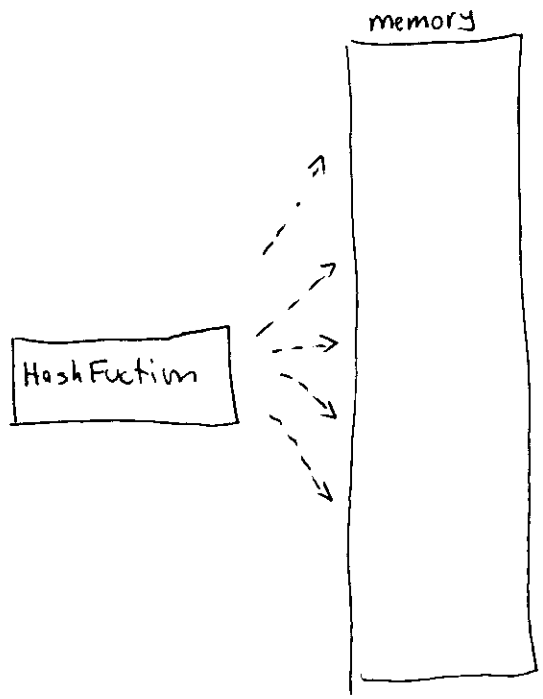
hashlookup

Can be thought of as 3 parts.

Hash Register
Hash Lookup
Hash Function

Hash Function generates the initial storage location

Hash Register AND Hash Lookup call Hash Function,



hashlookup theory

hashlookup begins with a call to HashFunction. HashFunction generates a number by juggling the character string in a preditable order which always results in particular number being generated.

Most of the time this generated number is unique. Using this generated number as a pointer, the spelling of the hash table name is compared with the name to find.

headers

example: #include <Xm.h> (provided by system)

include "em.h" (provided by programers)

some known headers.

base-data.h
coord-data.h
graph-data.h
fm.h
ipf-ipc.h
dmgr.h
em.h

hexidecimal

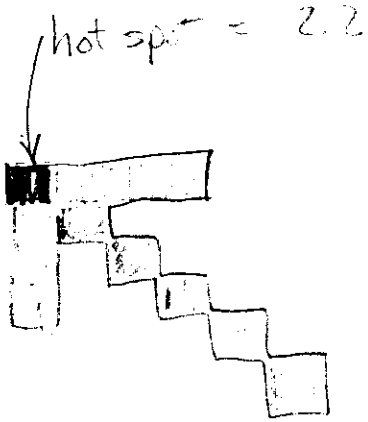
```
printf (" value %04.4x \n", hex value )
```

hide

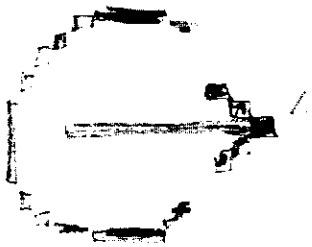
Some buses are "hidden" on graph
i.e. no icon is displayed
name is displayed
no voltage data is displayed

see: checkName (pf_cb.c) decision to display units.
psbuildGraphCoord (graphpscor.c) set value from coord f.4
printPSBusRecord (graphdata.c)
display_bus_coord_data (pscordat.c) set Radio buttons.
createBusGadget (vertex.c) sets correct icon

hot spot



hot spot on cursor
gives exact x, y
location (mouse)
is pointing to.



= 7,14

See: cursor
bit map
bit map Editor

icon

Term used to discribe pixmap or symbol on screen

See: setVertexIcon (pf-cb.c)

include option

includes.

I == look in this directory if not found in local directory.

-I. (back dir)

-I/usr/include

-I/usr/include/sys

-I/usr/include/Mrm

-I/usr/include/Xm

-I/usr/include/Xt

-I/urs/include/X11

-I/shrunis/ipf/src/gui/incref

-I/shrunis/ipf/src/ipc/incref

~~Flags~~

~~Std~~

For mat C89 -Idirectory directory directory etc. -(other options)

see: case_id

PUT_DATA, TYPE = COMMENTS

pf_descrip.c

initialize

see: autostart
resource file
init_print_opts
psgetHtWdScale

see: man **send**
ipcsend

sends a message to other processes

presently there are 3 versions of ipcsend – one for each platform.

```
>sl
    (listing)
>4
```

gets to the ipc library
(so sget will work etc.)

see: ipc_synch_rw (ipc_cb.c)

IPC

See: rf. (launch)

inter-process-communications

ipf
(launch)

Per-Flow [illegible] process

by a call to launch_srv (in launch_srv.c)

Contents of [illegible] is:

run_ipfserv ipfserv -socket id 1024

executes this line

contents of this file => S*

which means execute any string following run_ipfsrv ...

jacket

This is Bill Rogers' personal terminology to describe the 10 individual forms stacked inside the BUS EDIT dialog. Rather than build 10 individual dialogs, these "jackets" are managed & unmanaged depending on what type of data is being edited.