BBN Systems and Technologies
A Division of Bolt Beranek and Newman Inc.

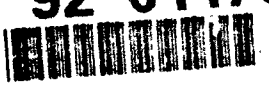
BBN Report No. 7628--Annex

SIMNET CVCC

Radio Performance Monitor (RADMON) CSCI

Volume II

Annex to Software Design Document: Source Code Reference



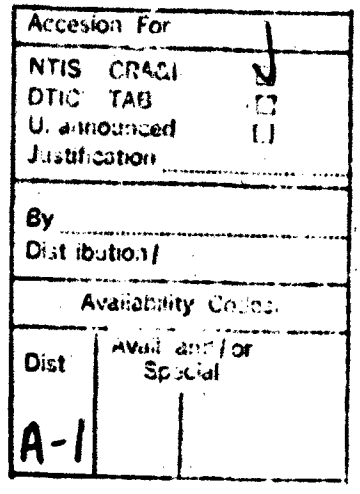
SIMNET
CVCC

Radio Performance Monitor (RADMON) CSCI

Volume II

Annex to Software Design Document: Source Code Reference

Contract No. MDA972-89-C-0060
Contract No. MDA972-90-C-0061
CDR Sequence No. 0002AB
December 1991



Prepared by:
Bolt Beranek and Newman Inc.
Systems and Technologies
Advanced Simulation
10 Moulton Street
Cambridge, MA 02138 USA

Prepared for:
Defense Advanced Research Projects Agency (DARPA)
3701 North Fairfax Street
Arlington, VA 22203-1714

This research was performed by BBN Systems and Technologies under Contract Nos. MDA972-89-C-0060 and MDA972-90-C-0061 to the Defense Advanced Research Projects Agency (DARPA). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policy, either expressed or implied, of DARPA, the U.S. Army, or the U.S. Government.

Table of Contents

0.1 display.c

0.1.1 Global Variables

0.1.1.1 function_button_widget

Synopsis: The main function button menu.
Type: WidgetRec *
Referenced by: void display_deselect()
void display_init()

0.1.1.2 general_area_widget

Synopsis: The widget for the general workspace area.
Type: WidgetRec *
Referenced by: static void clipResize()
void display_init()
static void map_create_widget()
static void traffic_create_widget()
static void create_status_widget()
static void state_create_widget()

0.1.1.3 general_widget

Synopsis: The scroll window containing the general workspace area.
Type: WidgetRec *
Referenced by: void display_init()

0.1.1.4 hierarchy_id

Synopsis: Identifier for UIL hierarchy.
Type: struct /* unnamed */ *
Referenced by: void display_init()
static void map_create_widget()
static void map_create_vehicle_widget()
static void traffic_create_widget()
static void create_status_widget()
static void state_create_widget()

0.1.1.5 label_widget

Synopsis: The text widget for labelling the overall user interface.
Type: WidgetRec *
Referenced by: void display_init()
static void set_identification_label()

0.1.1.6 main_widget

Synopsis: The main form widget.
Type: WidgetRec *
Referenced by: void display_init()
void mapInitWidgets()
void trafficInitWidgets()
void statusInitWidgets()
void stateInitWidgets()
void fileInitWidgets()

0.1.1.7 sub_function_widget

Synopsis: The frame widget for the sub–function menu.

Type: WidgetRec *
Referenced by: void display_init()
void trafficInit\Widgets()
void stateInit\Widgets()

0.1.1.8 top_widget

Synopsis: The top level application shell widget.
Type: WidgetRec *
Referenced by: static void dump_callback()
void display_init()

0.1.2 Local Variables

0.1.2.1 application_data

Synopsis: General data for the network monitor application extracted from the X-window resource manager.
Type: static APPLICATION_DATA
Referenced by: static void set_identification_label()

0.1.2.2 application_resources

Synopsis: Descriptions of resources.
Type: static XtResource
Referenced by: void display_init()

0.1.2.3 application_resources_2

Synopsis: Descriptions of additional resources fetched after fetching application_resources.
Type: static XtResource
Referenced by: void display_init()

0.1.2.4 buttonI_h_rcsid

Synopsis: RCS id for buttonI.h.
Type: static char *

0.1.2.5 cvt_args

Synopsis: Null argument list for resource converters.
Type: static struct /* unnamed */
Referenced by: void display_init()

0.1.2.6 function_button_closure

Synopsis: Wraps the function_buttons array in a closure suitable for passing to function_descend().
Type: static FUNCTION_BUTTON_CLOSURE
Referenced by: void display_deselect()
void display_init()
References to: static FUNCTION_BUTTON function_buttons[9]

0.1.2.7 function_buttons

Synopsis: Describes the buttons in the main function menu.
Type: static FUNCTION_BUTTON
Referenced by: static FUNCTION_BUTTON_CLOSURE function_button_closure
References to: FB_DATA map_function_closure
void file_read()
FB_DATA status_function_closure
FB_DATA traffic_sub_function_closure
void file_write()

void state_function()
static void quit_callback()
static void reset_callback()
FB_DATA state_sub_function_closure
void map_display()
static void dump_callback()

0.1.2.8 network_h_rcsid

Synopsis: The RCS id for network.h.
Type: static char

0.1.2.9 rcsid

Synopsis: The RCS id for this file.
Type: static char

0.1.3 Global Functions

0.1.3.1 Dispatch

Synopsis: Dispose of all X events.
Type: void
Arguments: <none>
Referenced by: <nothing>
References to: <nothing>

0.1.3.2 NameToWidget

Synopsis: Get a widget id for the named child of a widget. Checks for failure and aborts.
Type: WidgetRec *
Arguments:

parent	Widget	The parent widget.
name	char *	The name of the widget to find.

Referenced by: static void CvtStringToWidget()
void display_init()
void mapInitWidgets()
static void map_create_widget()
static void map_create_vehicle_widget()
void trafficInitWidgets()
static void traffic_create_widget()
void statusInitWidgets()
static void create_status_widget()
void stateInitWidgets()
static void state_create_widget()
void fileInitWidgets()
References to: char *strcpy()

0.1.3.3 display_deselect

Synopsis: Deselect the selected button if any.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: WidgetRec *function_button_widget
static FUNCTION_BUTTON_CLOSURE function_button_closure

0.1.3.4 display_init

Synopsis: Initialize lots of display stuff. Creates most of the widget tree.
Type: void
Arguments:

argcp	int *	Pointer to argument count (updated).

	argv	**char ****	Argument vector.

Referenced by: int main()
References to: void stateRegisterNames()
SimulationEnvironment simEnv
WidgetRec *NameToWidget()
void statusRegisterNames()
void statusInitWidgets()
void trafficRegisterNames()
void trafficInitWidgets()
void stateInitWidgets()
void mapRegisterNames()
void mapInitWidgets()
void fileRegisterNames()
void fileInitWidgets()
WidgetRec *function_button_widget
WidgetRec *sub_function_widget
WidgetRec *general_widget
WidgetRec *general_area_widget
WidgetRec *top_widget
WidgetRec *main_widget
WidgetRec *label_widget
struct /* unnamed */ *hierarchy_id
static APPLICATION_DATA application_data
static XtResource application_resources[1]
static XtResource application_resources_2[4]
static FUNCTION_BUTTON_CLOSURE function_button_closure
static void create_function_items()
static void clipResize()
static struct /* unnamed */ cvt_args[1]
static void CvtStringToWidget()
static void CvtStringToXmStringLtoR()
static void CvtStringToColor()

0.1.4 Local Functions

0.1.4.1 CvtStringToColor

Synopsis: Convert a string to an XColor.
Type: static void
Arguments:

args	**XrmValuePtr**	
nargs	**Int ***	
fromVal	**XrmValuePtr**	
toVal	**XrmValuePtr**	

Referenced by: void display_init()
References to: <nothing>

0.1.4.2 CvtStringToWidget

Synopsis: Resource converter from string to widget, where the string is the name of the widget relative to the parent.
Type: static void
Arguments:

args	**XrmValuePtr**	
nargs	**Int ***	
fromVal	**XrmValuePtr**	
toVal	**XrmValuePtr**	

Referenced by: void display_init()
References to: WidgetRec *NameToWidget()

0.1.4.3 CvtStringToXmStringLtoR

Synopsis: CvtStringToStringLtoR.
Convert a string to a compound converting \n to separators.
Type: static void
Arguments:

args	**XrmValuePtr**	
nargs	**int ***	
fromVal	**XrmValuePtr**	
toVal	**XrmValuePtr**	

Referenced by: void display_init()
References to: <nothing>

0.1.4.4 clipResize

Synopsis: Resize callback for the clipping widget of the the general_area_widget. Attempts to make the width of the Pack widget track the clipping widget. There are scenarios in which this can fail. It's not clear if these failures can be corrected given the way the X toolkit behaves.
Type: static void
Arguments:

w	**Widget**	Clipping widget.
data	**caddr_t**	Not used.
cback	**XmAnyCallbackStruct ***	Standard callback information.

Referenced by: void display_init()
References to: WidgetRec *general_area_widget

0.1.4.5 create_function_items

Synopsis: Construct a resource list and retrieve the label strings to be put on the buttons. Construct the item lists for the button lists. Recursively do the same for sub–function button lists.
Type: static void
Arguments:

w	**Widget**	widget for getting resources.
fbc	**FUNCTION_BUTTON_CLOSUREP**	Function button info.

Referenced by: static void create_function_items()
void display_init()
References to: static void create_function_items()

0.1.4.6 dump_callback

Synopsis: Activate callback for the dump button. Dumps the widget tree to stdout for debugging.
Type: static void
Arguments:

w	**Widget**	The button widget.

Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: void DisablePeriodic()
void EnablePeriodic()
WidgetRec *top_widget

0.1.4.7 function_callback

Synopsis: Activate callback for the function button list. If the selected function has sub–functions, the current sub–function widgets are unmanaged. The sub–function widget is activated and any previously active sub–sub–function widget are reactivated. Then the function attached to the button is executed.
Type: static void
Arguments:

w	**Widget**	The button widget.
fbc	**FUNCTION_BUTTON_CLOSUREP**	Information about context.
cback	**XmButtonLCallbackStruct ***	Standard callback information.

Referenced by: static XtCallbackRec callbacks[2]

static struct /* unnamed */ register_list[1]
References to: static void function_descend(FUNCTION_BUTTON_CLOSURE * void ()())

0.1.4.8 function_descend

Synopsis: Descends through selected buttons of the button list tree applying fn to the button lists.
Type: static void
Arguments:

fbc	FUNCTION_BUTTON_CLOSUREP	Sub-menu information.
fn	VOIDFNP	Function to apply.

Referenced by: static void function_callback()
References to: <nothing>

0.1.4.9 quit_callback

Synopsis: Activate callback for the quit button. Exits the program.
Type: static void
Arguments:

w	Widget	The button widget.

Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: void exit_gracefully()

0.1.4.10 reset_callback

Synopsis: Activate callback for the reset button. Returns to online mode if necessary and deletes all recorded data.
Type: static void
Arguments:

w	Widget	The button widget.

Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: unsigned char online_mode
void DrainPDUs()
void DisablePeriodic()
void EnablePeriodic()
void resetRadios()
void DestroyEverything()
void Online()
void traffic_reset()
void map_reset()
int resetTime()

0.2 file.c

0.2.1 Global Variables

0.2.2 Local Variables

0.2.2.1 file_error_widget

Synopsis: Popup for file errors.
Type: static WidgetRec *
Referenced by: static void file_error()
void fileInitWidgets()

0.2.2.2 file_inform_divisor

Synopsis: Divisor for scaling file progress.
Type: static int
Referenced by: static void file_inform()
char *file_gets()
void file_count()

0.2.2.3 file_inform_label_widget

Synopsis: The label widget in the file progress popup.
Type: static WidgetRec *
Referenced by: static void file_inform()
void fileInitWidgets()

0.2.2.4 file_inform_position

Synopsis: The current file progress position.
Type: static int
Referenced by: static void file_inform()
char *file_gets()
void file_count()

0.2.2.5 file_inform_scale_value

Synopsis: The current file progress scale value.
Type: static int
Referenced by: static void file_inform()
char *file_gets()
void file_count()

0.2.2.6 file_inform_scale_widget

Synopsis: The scale widget for file progress.
Type: static WidgetRec *
Referenced by: static void file_inform()
char *file_gets()
void file_count()
void fileInitWidgets()

0.2.2.7 file_inform_widget

Synopsis: The file progress popup widget.
Type: static WidgetRec *
Referenced by: static void file_flush()
static void file_inform()
static void file_inform_end()
char *file_gets()
void file_count()
void fileInitWidgets()

0.2.2.8 file_line_number

Synopsis: The current line number.
Type: static int
Referenced by: static void file_error()
static void file_inform()
char *file_gets()
void file_count()

0.2.2.9 file_widget

Synopsis: The file selection popup widget.
Type: static WidgetRec *
Referenced by: static void file_ok()
void file_write()
void file_read()
void fileInitWidgets()

0.2.2.10 network_h_rcsid

Synopsis: The RCS id of network.h.
Type: static char

0.2.2.11 rcsid

Synopsis: The RCS id of this file.
Type: static char

0.2.3 Global Functions

0.2.3.1 fileInitWidgets

Synopsis: Get widget IDs for file widgets.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: WidgetRec *NameToWidget()
WidgetRec *main_widget
static WidgetRec *file_error_widget
static WidgetRec *file_inform_widget
static WidgetRec *file_inform_label_widget
static WidgetRec *file_inform_scale_widget
static WidgetRec *file_widget

0.2.3.2 fileRegisterNames

Synopsis: Register names for Mrm related to the file widgets.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: <nothing>

0.2.3.3 file_count

Synopsis: Increment the lines which have been written to the output file. Update the file_inform_widget to indicate progress.
Type: void
Arguments:

n	int	The lines to increment by.

Referenced by: void tune_write_file()
void tlWriteFile()
void state_write_file()
void net_write_file()
static void file_write_file()
References to: static WidgetRec *file_inform_widget
static WidgetRec *file_inform_scale_widget
static int file_inform_divisor
static int file_inform_position
static int file_inform_scale_value
static int file_line_number

0.2.3.4 file_gets

Synopsis: Do fgets on the input file and count lines. Update the file_inform_widget as reading progresses.
Type: char *
Arguments:

buf	char *	The buffer into which to get the string.
size	int	The buffer size.
f	FILE *	The file to read from.

Referenced by: char *tune_read_file()
char *tlReadFile()

char *state_read_file()
char *net_read_file()
static void file_read_file()
References to: static WidgetRec *file_inform_widget
static WidgetRec *file_inform_scale_widget
static int file_inform_divisor
static int file_inform_position
static int file_inform_scale_value
static int file_line_number

0.2.3.5 file_read

Synopsis: Activate callback for the "read" button. Pops up a file selection widget to select a filename. The file_io_function is set to file_read_file.
Type: void
Arguments: <none>
Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: static WidgetRec *file_widget
static void ()()
static void file_popup()
static void file_read_file()

0.2.3.6 file_write

Synopsis: Activate callback for the "write" button. Pops up a file selection widget to select a filename. The file_io_function is set to file_write_file.
Type: void
Arguments: <none>
Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: static WidgetRec *file_widget
static void ()()
static void file_popup()
static void file_write_file()

0.2.4 Local Functions

0.2.4.1 file_error

Synopsis: Popup an error message for a file io operation.
Type: static void
Arguments:

fmt	char *	The format of the error message.
arg	char *	One sprintf argument.

Referenced by: static void file_write_file()
static void file_read_file()
References to: static WidgetRec *file_error_widget
static int file_line_number
static void file_popup()

0.2.4.2 file_flush

Synopsis: Bring the display up to date before continuing with the file operation.
Type: static void
Arguments: <none>
Referenced by: static void file_inform()
References to: static WidgetRec *file_inform_widget

0.2.4.3 file_inform

Synopsis: Popup the file progress widget containing the specified message. Initialize the scale widget.
Type: static void

Arguments:	fmt	char *	The message format.
	arg	char *	One sprint argument.
	maximum	int	The scale maximum (total file lines).
Referenced by:	static void file_write_file()		
	static void file_read_file()		
References to:	static WidgetRec *file_inform_widget		
	static WidgetRec *file_inform_label_widget		
	static WidgetRec *file_inform_scale_widget		
	static int file_inform_divisor		
	static int file_inform_position		
	static int file_inform_scale_value		
	static int file_line_number		
	static void file_popup()		
	static void file_flush()		

0.2.4.4 file_inform_end

Synopsis:	Popdown the file_inform_widget.
Type:	static void
Arguments:	<none>
Referenced by:	static void file_write_file()
	static void file_read_file()
References to:	static WidgetRec *file_inform_widget

0.2.4.5 file_ok

Synopsis:	File selection callback. Applies file_io_function to the selected file name.		
Type:	static void		
Arguments:	w	Widget	The file selection widget.
	data	caddr_t	Not used.
	cback	XmSelectionBoxCallbackStruct *	Standard callback information.
Referenced by:	static struct /* unnamed */ register_list[3]		
References to:	static WidgetRec *file_widget		
	static void ()()		

0.2.4.6 file_popup

Synopsis:	Pops up a widget on top of other widgets. For some reason, Motif doesn't always do this.		
Type:	static void		
Arguments:	w	Widget	The widget to pop up.
Referenced by:	static void file_error()		
	static void file_inform()		
	void file_write()		
	void file_read()		
References to:	<nothing>		

0.2.4.7 file_read_file

Synopsis:	file_io_function to read a radmon file. The selected file is opened and the various parts of the file read. A failure deletes all the information read and operation reverts to on–line mode.		
Type:	static void		
Arguments:	filename	char *	The name of the file to read.
Referenced by:	void file_read()		
References to:	char *strcpy()		
	int now		
	char *ethernet_address		
	char *host_name		
	char *file_name		
	int tod		

char *net_read_file()
void DrainPDUs()
void DisablePeriodic()
void DestroyEverything()
void Online()
void Offline()
char *state_read_file()
char *tune_read_file()
char *file_gets()
static void file_inform()
static void file_error()
static void file_inform_end()

0.2.4.8 file_write_file

Synopsis: file_io_function to write a radmon file. The selected file name is massaged to add a .radmon extension if necessary, the file is opened and the various parts of the file written.
Type: static void
Arguments: filename char * The name of the file to write.
Referenced by: void file_write()
References to: int now
char *ethernet_address
char *host_name
int tod
void net_write_file()
void DisablePeriodic()
void state_write_file()
void tune_write_file()
void file_count()
static void file_inform()
static void file_error()
static void file_inform_end()
int tune_count_file()
int state_count_file()
int net_count_file()

0.3 hTable.c

0.3.1 Global Variables

0.3.2 Local Variables

0.3.2.1 free_count

Synopsis: Count of entry items on free_list.
Type: static int
Referenced by: void htReplenish()
void htFreeInit()
void htFree()
HASHENTRY *htInsert()

0.3.2.2 free_list

Synopsis: List of available entry items.
Type: static HASHENTRY *
Referenced by: void htReplenish()
void htFreeInit()

void htFree()
HASHENTRY *htInsert()

0.3.2.3 rcsid

Synopsis: RCS id of this file.
Type: static char

0.3.3 Global Functions

0.3.3.1 htFind

Synopsis: Searchs the hash table for the specified key and returns the HASHENTRY found. Returns NULL if the entry cannot be found.
Type: HASHENTRY *

Arguments:			
	ht	HTABLEP	The hash table to look in.
	name	char *	The key to look for.
	value	hash_value_type	*Return the value here.

Referenced by: TUNE *tune_find()
STATE *state_find()
static int VehicleIDtoIndex()
References to: static int hash()

0.3.3.2 htFree

Synopsis: htFree(): Release hash table storage.
Type: void

Arguments:			
	ht	HTABLEP	The hash table to release.

Referenced by: void tune_destroy()
void state_destroy()
void net_destroy()
References to: static HASHENTRY *free_list
static int free_count
int free()

0.3.3.3 htFreeInit

Synopsis: Initialize the free list and count and do the initial replenishment.
Type: void
Arguments: <none>
Referenced by: int main()
References to: void htReplenish()
static HASHENTRY *free_list
static int free_count

0.3.3.4 htInit

Synopsis: Creates a hash table of the specified size and key size. Initializes the empty hash table.
Type: HTABLE *

Arguments:			
	size	int	Size of hash table (0 means default).
	key	int	Size of key (zero or neg for string).

Referenced by: void tune_init()
void tune_destroy()
void state_init()
void state_destroy()
void InitSimNetwork()
void net_destroy()
References to: <nothing>

0.3.3.5 htInsert

Synopsis: Search for an existing entry matching the key. If found, its value is changed. Otherwise, a new entry is created and chained onto the list for the proper bucket. In either case, the HASHENTRY is returned.

Type: HASHENTRY *

Arguments:

ht	HTABLEP	The hash table to use.
name	char *	The key to insert under.
value	int	The value to insert.

Referenced by: TUNE *tune_find()
STATE *state_find()
static int VehicleIDtoIndex()

References to: static HASHENTRY *free_list
static int free_count
static int hash()

0.3.3.6 htReplenish

Synopsis: Replenish storage for hashentries. htInsert may be called from interrupt level and is not allowed to allocate memory. We preallocate here and keep the entries on a free list.

Type: void

Arguments: <none>

Referenced by: int main()
static void slow_periodic()
char *tune_read_file()
char *state_read_file()
void htFreeInit()

References to: static HASHENTRY *free_list
static int free_count

0.3.4 Local Functions

0.3.4.1 hash

Synopsis: Combines together the bytes making up the key in to an integer index in to the hash table.

Type: static int

Arguments:

name	char *	String to hash.
ht	HTABLEP	Hash table.

Referenced by: HASHENTRY *htInsert()
HASHENTRY *htFind()

References to: <nothing>

0.4 map.c

0.4.1 Global Variables

0.4.1.1 map_function_closure

Synopsis: Map function buttons description.

Type: FB_DATA

Referenced by: static FUNCTION_BUTTON function_buttons[9]

References to: static WidgetRec *map_function_widget
static WidgetRec *map_function_attach_widget

0.4.2 Local Variables

0.4.2.1 buttonl_h_rcsid

Synopsis: RCS id of buttonl.h.
Type: static char *

0.4.2.2 map_anomaly_widget

Synopsis: The anomaly button widget.
Type: static WidgetRec *
Referenced by: static void map_freeze()
static void map_unfreeze()
static void map_create_widget()

0.4.2.3 map_body_widget

Synopsis: The drawing area portion.
Type: static WidgetRec *
Referenced by: static void map_color_animate()
static void map_draw()
static void map_erase()
static void map_display_move()
static void map_input()
static void map_allocate_colors()
static unsigned int map_animation_pixmap()
static void map_create_widget()
static void map_create_vehicle_widget()
void map_display()
void map_reset()
static void map_advance()

0.4.2.4 map_box_height

Synopsis: The outline box height.
Type: static unsigned int
Referenced by: static void map_display_move()

0.4.2.5 map_box_visible

Synopsis: True if we are displaying the box.
Type: static char
Referenced by: static void map_display_move()
static void map_input()
static void map_create_widget()

0.4.2.6 map_box_width

Synopsis: The outline box width.
Type: static unsigned int
Referenced by: static void map_display_move()

0.4.2.7 map_box_x

Synopsis: The outline box location (x.
Type: static int
Referenced by: static void map_display_move()

0.4.2.8 map_box_y

Synopsis: The outline box location (y.
Type: static int
Referenced by: static void map_display_move()

0.4.2.9 map_cmap

Synopsis: The color map for the map.
Type: static unsigned int
Referenced by: static void map_color_animate()
static void map_allocate_colors()

0.4.2.10 map_color_phase

Synopsis: Current animation phase.
Type: static int
Referenced by: static void map_color_animate()
static void map_allocate_colors()

0.4.2.11 map_color_timeout

Synopsis: Next animation timeout.
Type: static unsigned int
Referenced by: static void map_color_animate()
static void map_draw()
static void map_allocate_colors()

0.4.2.12 map_conflict_gc

Synopsis: Graphics context for drawing reception conflict connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static void map_create_widget()

0.4.2.13 map_connected_gc

Synopsis: Graphics context for drawing connected connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static void map_create_widget()

0.4.2.14 map_data

Synopsis: Resource data for the map display.
Type: static MAP_DATA
Referenced by: static void map_color_animate()
static unsigned char map_style_of_type()
static void map_draw()
static void map_color_radio()
static void map_color_vehicle()
static void map_input()
static struct /* unnamed */ *colorp[12]
static void map_allocate_colors()
static unsigned int map_animation_pixmap()
static void map_create_widget()
static char map_layout()

0.4.2.15 map_display_pending

Synopsis: Pending map_display.
Type: static unsigned int
Referenced by: void map_display()
void map_vehicle_moved()
static void map_more_radios()

0.4.2.16 map_display_pending_50

Synopsis: Pending quick above.

Type: static unsigned int
Referenced by: void map_display()
void map_vehicle_moved()
static void map_more_radios()

0.4.2.17 map_erase_gc

Synopsis: Graphics context for erasing.
Type: static XGC *
Referenced by: static char map_draw_bolt()
static void map_display_move()
static void map_create_widget()

0.4.2.18 map_exact_defs

Synopsis: Colors for animation.
Type: static struct /* unnamed */
Referenced by: static void map_color_animate()
static void map_allocate_colors()

0.4.2.19 map_freeze_widget

Synopsis: The freeze button.
Type: static WidgetRec *
Referenced by: static void map_freeze()
static void map_unfreeze()
static void map_create_widget()

0.4.2.20 map_frozen

Synopsis: True if the map is frozen.
Type: static char
Referenced by: static void map_freeze()
static void map_unfreeze()
void map_display()
void map_advance_display()
void map_destroy()

0.4.2.21 map_function_attach_widget

Synopsis: Widget to which buttons are added.
Type: static WidgetRec *
Referenced by: FB_DATA map_function_closure
void mapInitWidgets()
static void map_create_radio_buttons()
void map_destroy()

0.4.2.22 map_function_widget

Synopsis: Container of above.
Type: static WidgetRec *
Referenced by: FB_DATA map_function_closure
void mapInitWidgets()

0.4.2.23 map_height

Synopsis: Current height of the map.
Type: static unsigned int
Referenced by: static void map_input()
static void map_resize()
static char map_layout()

0.4.2.24 map_left_extension

Synopsis: How far a vehicle extends left.
Type: static int
Referenced by: static void map_measure()
static char map_layout()

0.4.2.25 map_move_vehicle

Synopsis: The vehicle being moved.
Type: static VehicleInfo *
Referenced by: static void map_display_move()
static void map_input()
static void map_create_widget()

0.4.2.26 map_move_widget

Synopsis: The vehicle widget being moved.
Type: static WidgetRec *
Referenced by: static void map_input()
static void map_create_widget()

0.4.2.27 map_move_x

Synopsis: Where the move started from (x.
Type: static int
Referenced by: static void map_display_move()
static void map_input()

0.4.2.28 map_move_y

Synopsis: Where the move started from (y.
Type: static int
Referenced by: static void map_display_move()
static void map_input()

0.4.2.29 map_n_radio_buttons

Synopsis: The current number of buttons.
Type: static int
Referenced by: static void map_select()
static void map_create_widget()
static void map_create_radio_buttons()
void map_advance_display()
void map_destroy()

0.4.2.30 map_oneway_gc

Synopsis: Graphics context for drawing oneway connectivity.
Type: static XGC *
Referenced by: static void map_create_widget()

0.4.2.31 map_outline_gc

Synopsis: Graphics context for drawing the outline.
Type: static XGC *
Referenced by: static char map_draw_bolt()
static void map_display_move()
static void map_create_widget()

0.4.2.32 map_pending_draw

Synopsis: Summary of pending draw operations on the map.
Type: static struct /* unnamed */

Referenced by: static void map_bounding_box()
static char map_draw_bolt()
static void map_reset_pending_draw()
static void map_draw()
static void map_start_draw()
static void map_erase()
static void map_start_erase()
static void map_input()
static void map_expose()
void mapRegisterNames()
static void map_update()

0.4.2.33 map_receiving_gc

Synopsis: Graphics context for drawing receiving connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static char map_draw_bolt()
static void map_create_widget()

0.4.2.34 map_resources

Synopsis: Map resource descriptions.
Type: static XtResource
Referenced by: static void map_create_widget()
References to: void BackgroundColorDefault()
void ForegroundColorDefault()

0.4.2.35 map_right_extension

Synopsis: How far a vehicle extends right.
Type: static int
Referenced by: static void map_measure()
static char map_layout()

0.4.2.36 map_some_animated

Synopsis: Some animated bolts exist.
Type: static char
Referenced by: static void map_color_animate()
static char map_draw_bolt()
static void map_draw()
static void map_allocate_colors()

0.4.2.37 map_step_widget

Synopsis: The step button.
Type: static WidgetRec *
Referenced by: static void map_freeze()
static void map_unfreeze()
static void map_create_widget()

0.4.2.38 map_time

Synopsis: The time text widget.
Type: static int
Referenced by: static void map_time_callback()
static void map_step()
static void map_step_to_anomaly()
static void map_freeze()
static void map_create_widget()

static void map_set_time()
static void map_advance()

0.4.2.39 map_time_widget

Synopsis: The time text widget.
Type: static WidgetRec *
Referenced by: static void map_freeze()
static void map_unfreeze()
static void map_create_widget()
static void map_set_time()

0.4.2.40 map_tuned_gc

Synopsis: Graphics context for drawing tuned connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static void map_create_widget()

0.4.2.41 map_unconnected_gc

Synopsis: Graphics context for drawing unconnected connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static void map_create_widget()

0.4.2.42 map_unfreeze_widget

Synopsis: The unfreeze button.
Type: static WidgetRec *
Referenced by: static void map_freeze()
static void map_unfreeze()
static void map_create_widget()

0.4.2.43 map_untuned_gc

Synopsis: Graphics context for drawing untuned connectivity.
Type: static XGC *
Referenced by: static void map_gc_of_type()
static void map_create_widget()

0.4.2.44 map_vehicle_height

Synopsis: How tall the vehicle is.
Type: static int
Referenced by: static void map_measure()
static char map_layout()

0.4.2.45 map_widget

Synopsis: The connectivity map.
Type: static WidgetRec *
Referenced by: static void map_close()
static void map_create_widget()
void map_display()
static void map_update()
void map_vehicle_moved()
static void map_more_radios()
void map_reset()
static void map_advance()
void map_advance_display()
void map_destroy()

0.4.2.46 map_width

Synopsis: Current width of the map.
Type: static unsigned int
Referenced by: static void map_input()
static void map_resize()
static char map_layout()

0.4.2.47 network_h_rcsId

Synopsis: The RCS id of network.h.
Type: static char

0.4.2.48 rcsId

Synopsis: The RCS id of this file.
Type: static char

0.4.3 Global Functions

0.4.3.1 BackgroundColorDefault

Synopsis: This routine is similar to _XmBackgroundColorDefault except that it returns an XColor instead of a Pixel. First _XmBackgroundColorDefault is called and then the resultant Pixel is queried to produce an XColor.
Type: void

Arguments:			
	widget	**Widget**	The Widget.
	offset	**int**	Offset in resource.
	valPtr	**XrmValuePtr**	Resource value.

Referenced by: static XtResource map_resources[27]
static XtResource traffic_resources[18]
References to: int bcopy()

0.4.3.2 ForegroundColorDefault

Synopsis: This routine is similar to _XmForegroundColorDefault except that it returns an XColor instead of a Pixel. First _XmForegroundColorDefault is called and then the resultant Pixel is queried to produce an XColor.
Type: void

Arguments:			
	widget	**Widget**	The Widget.
	offset	**int**	Offset in resource.
	valPtr	**XrmValuePtr**	Resource value.

Referenced by: static XtResource map_resources[27]
static XtResource traffic_resources[18]
References to: int bcopy()

0.4.3.3 mapInitWidgets

Synopsis: Get widget pointers.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: WidgetRec *NameToWidget()
static WidgetRec *map_function_attach_widget
static WidgetRec *map_function_widget
WidgetRec *main_widget

0.4.3.4 mapRegisterNames

Synopsis: Register callbacks and other values. Add converter for MapStyle.
Type: void

Arguments: <none>
Referenced by: void display_init()
References to: static struct /* unnamed */ map_pending_draw
static void CvtStringToMapStyle()

0.4.3.5 map_advance_display

Synopsis: Public interface to map_advance.
Type: void
Arguments:

up_to	MSEC	How far to advance.
max_advance	MSEC	The maximum to advance.
if_frozen	Boolean	True if the display is frozen.

Referenced by: static void slow_periodic()
static void medium_periodic()
static void traffic_input()
References to: int net_n_radios
void map_display()
static char map_frozen
static WidgetRec *map_widget
static int map_n_radio_buttons
static void map_advance()

0.4.3.6 map_destroy

Synopsis: Destroy the radio selection buttons and remove the map display from the screen.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: static char map_frozen
static WidgetRec *map_function_attach_widget
static WidgetRec *map_widget
static int map_n_radio_buttons

0.4.3.7 map_display

Synopsis: Display the map. First the map_widget is created, if necessary. Then new radio selection buttons are created, if necessary. Then new vehicle widgets are created, if necessary. Then vehicle widgets are remanaged as necessary. The map is remeasured and re-layed out. Then the map_widget is managed if it is not already on the screen otherwise it is redrawn if any of the preceding operations require it. The map is forced into the frozen state in offline mode.
Type: void
Arguments: <none>
Referenced by: static FUNCTION_BUTTON function_buttons[9]
void map_vehicle_moved()
static void map_more_radios()
static void map_advance()
void map_advance_display()
References to: VehicleInfo **net_vehicles
int net_n_vehicles
unsigned char online_mode
static char map_frozen
static WidgetRec *map_widget
static WidgetRec *map_body_widget
static unsigned int map_display_pending
static unsigned int map_display_pending_50
static void map_recolor()
static void map_freeze()
static void map_create_widget()
static void map_create_vehicle_widget()

static char map_remanage_vehicle()
static void map_measure()
static char map_layout()
static void map_create_radio_buttons()

0.4.3.8 map_init_radio

Synopsis: Initializes the map information for a radio.
Type: void
Arguments: rp RadioInfoP The radio to initialize.
Referenced by: void map_reset()
char *net_read_file()
void netChangeState()
References to: STATE *state_find()
int net_max_radios
STATE initial_state
int bzero()

0.4.3.9 map_reset

Synopsis: Resets the map. Called when a new set of data is started (read from file or restarting on-line collection).
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
void Offline()
void Online()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
unsigned char online_mode
void map_init_radio()
static WidgetRec *map_widget
static WidgetRec *map_body_widget
static void map_set_time()
static void map_freeze()
static void map_unfreeze()

0.4.3.10 map_vehicle_appeared

Synopsis: map_vehicle_vanished.
Marks the vehicle as needing recoloring.
Type: void
Arguments: vp1 VehicleInfoP The vehicle which appeared.
Referenced by: char *net_read_file()
static void ProcessVehicleAppearancePDU()
References to: <nothing>

0.4.3.11 map_vehicle_moved

Synopsis: Schedules a map_display operation for ten seconds from now if there is not already a map_display pending.
Type: void
Arguments: vp VehicleInfoP The vehicle which moved.
Referenced by: <nothing>
References to: void map_display()
static WidgetRec *map_widget
static unsigned int map_display_pending
static unsigned int map_display_pending_50

0.4.3.12 map_vehicle_vanished

Synopsis: Marks the vehicle as needing recoloring.
Type: void
Arguments: vp1 VehicleInfoP The vehicle which vanished.
Referenced by: static void TimeoutVehicles()
References to: <nothing>

0.4.4 Local Functions

0.4.4.1 CvtStringToMapStyle

Synopsis: Resource converter to convert map style keywords to the corresponding style constant.
Type: static void
Arguments: args XrmValue * Not used.
nargs int * Number of args (0).
from XrmValue * Source resource value.
to XrmValue * Converted value.
Referenced by: void mapRegisterNames()
References to: <nothing>

0.4.4.2 map_advance

Synopsis: Advances the map display one step. First find which radio has the earliest next state. If this state has a new tuning, change all connectivity to either con_tuned or con_untuned depending on whether the other radio is tuned the same or not.

Next consider whether the state transition is the initiation or termination of transmission or reception. In each case, update the connectivity accordingly. A complication is necessary due to the possibility that PDU transmission delays allow cause and effect to be reversed; the beginning of transmission may follow the beginning of reception. This is resolved by examining the following state of affected radios to avoid false indications of lack of connectivity.

Start of Transmission.

Scan all receivers tuned to this frequency. A receiver which is tuned the same as the transmitter should be receiving from that transmitter or should be doing so in the next state. If it is not, then there is either a conflict or a lack of connectivity. A conflict is indicated when the receiver is busy receiving from a different transmitter or is itself transmitting. These two connectivity changes are made now, otherwise we wait until the next state when the onset of reception will be recorded.

Start of Reception.

When a receiver begins receiving from a transmitter, the connectivity becomes con_receiving.

End of Transmission.

Nothing is done on this transition. All the work is done at the end of reception at the receivers.

End of Reception.

Reception can cease due to four causes: end of transmission, loss of signal, reception of a stronger signal (SC mode only), or start of transmission by the receiver (double push-to-talk). End of transmission is determined by examining the state of the transmitter (or its next state) to see if it is still transmitting on the same frequency. If not, then this is a normal end of reception and the connectivity is marked con_connected. If the transmitter is still transmitting then the next state of the receiver is examined. If that state is only a short time in the future and is ST_TRANSMITTING or ST_RECEIVING, then the connectivity becomes con_conflict. Otherwise the connectivity becomes con_unconnected.

Type: static void
Arguments: up_to MSEC How far to advance.
advance_mode MAP_ADVANCE_MODE
How to advance.
max_advance MSEC The maximum advancement.

Referenced by: static void map_time_callback()
static void map_step()
static void map_step_to_anomaly()
static void map_freeze()
static void map_unfreeze()
void map_advance_display()
References to: int now
struct RadioInfoStruct **net_radios
int net_n_radios
void map_display()
static WidgetRec *map_widget
static WidgetRec *map_body_widget
static int map_time
static void map_set_time()
static void map_update()
static int map_set_connectivity()
static int map_radio_transmit_time()
static void map_set_radio_time()

0.4.4.3 map_allocate_colors

Synopsis: Allocate colors for the map display. We actually only need four writable colors, but we allocate them all anyway.
Type: static void
Arguments: <none>
Referenced by: static void map_create_widget()
References to: static WidgetRec *map_body_widget
static MAP_DATA map_data
static struct /* unnamed */ map_exact_defs[4][4]
static unsigned int map_cmap
static char map_some_animated
static int map_color_phase
static unsigned int map_color_timeout

0.4.4.4 map_animation_pixmap

Synopsis: Create one of eight pixmaps containing a succesion of stripes in four colors which, when animated, will appear to move north, northeast, east, southeast, etc.
Type: static unsigned int

Arguments:			
	dir	int	Direction of motion.
	rootwindow	Window	Root window.

Referenced by: static void map_create_widget()
References to: static WidgetRec *map_body_widget
static MAP_DATA map_data

0.4.4.5 map_bounding_box

Synopsis: Updates the area which needs to be redrawn by virtue of erasing a lightning bolt. At the moment, this area is simpling the bounding rectangle covering all bolts which have been erased.
Type: static void

Arguments:			
	pts	register XPoint *	Array of points.
	npts	int	Number of points.

Referenced by: static char map_draw_bolt()
References to: static struct /* unnamed */ map_pending_draw

0.4.4.6 map_close

Synopsis: The map_widget is removed from the screen (unmanaged).
Type: static void
Arguments: <none>

Referenced by: static struct /* unnamed */ register_list[13]
References to: static WidgetRec *map_widget

0.4.4.7 map_color_animate

Synopsis: Interval timer callback for doing color-map animation. Switches the colors for the animated lightning bolts to the next phase. The timer is restarted if there are still some animated bolts present otherwise the animation stops.
Type: static void
Arguments: <none>
Referenced by: static void map_color_animate()
static void map_draw()
References to: static WidgetRec *map_body_widget
static MAP_DATA map_data
static struct /* unnamed */ map_exact_defs[4][4]
static unsigned int map_cmap
static char map_some_animated
static int map_color_phase
static unsigned int map_color_timeout
static void map_color_animate()

0.4.4.8 map_color_radio

Synopsis: Change the background color of the radio label widget according to the radio's state.
Type: static void
Arguments: **rp** **RadioInfoP** The radio to color.
Referenced by: static void map_input()
static void map_recolor()
References to: VehicleInfo **net_vehicles
static MAP_DATA map_data

0.4.4.9 map_color_vehicle

Synopsis: Change the background color of the vehicle label widget according to the vehicle's state.
Type: static void
Arguments: **vp** **VehicleInfoP** The vehicle to color.
Referenced by: static void map_input()
static void map_recolor()
References to: static MAP_DATA map_data

0.4.4.10 map_create_radio_buttons

Synopsis: Create selection buttons for radios which have recently appeared.
Type: static void
Arguments: <none>
Referenced by: void map_display()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
char *radio_name()
static WidgetRec *map_function_attach_widget
static int map_n_radio_buttons

0.4.4.11 map_create_vehicle_widget

Synopsis: Create the widget for a vehicle. The various widget ids are recorded in the vehicle or radio structures. Event handlers are added for dragging the widgets around. The radios are initially unselected.
Type: static void
Arguments: **vp** **VehicleInfoP** The vehicle to be created.
Referenced by: void map_display()

References to: WidgetRec *NameToWidget()
char *vehicle_name()
void set_widget_label()
static WidgetRec *map_body_widget
struct /* unnamed */ *hierarchy_id
static void map_input()

0.4.4.12 map_create_widget

Synopsis: Create the map widget and initialize things.
Type: static void
Arguments: <none>
Referenced by: void map_display()
References to: WidgetRec *NameToWidget()
static WidgetRec *map_widget
static WidgetRec *map_body_widget
static WidgetRec *map_step_widget
static WidgetRec *map_anomaly_widget
static WidgetRec *map_freeze_widget
static WidgetRec *map_unfreeze_widget
static WidgetRec *map_time_widget
static WidgetRec *map_move_widget
static char map_box_visible
static VehicleInfo *map_move_vehicle
static MAP_DATA map_data
static int map_time
static XGC *map_erase_gc
static XGC *map_outline_gc
static XGC *map_oneway_gc
static XGC *map_untuned_gc
static XGC *map_tuned_gc
static XGC *map_unconnected_gc
static XGC *map_connected_gc
static XGC *map_conflict_gc
static XGC *map_receiving_gc[8]
static int map_n_radio_buttons
struct /* unnamed */ *hierarchy_id
WidgetRec *general_area_widget
static void map_set_time()
static XtResource map_resources[27]
static void map_resize()
static void map_unfreeze()
static void map_allocate_colors()
static unsigned int map_animation_pixmap()

0.4.4.13 map_display_move

Synopsis: Update the position of a vehicle outline box while dragging the vehicle around.
Type: static void
Arguments:

visible	**Boolean**	True to draw, false to erase.
x	**Position**	Vehicle position x.
y	**Position**	Vehicle position y.

Referenced by: static void map_input()
References to: static WidgetRec *map_body_widget
static int map_move_x
static int map_move_y
static int map_box_x
static int map_box_y

static unsigned int map_box_wid:.
static unsigned int map_box_height
static char map_box_visible
static VehicleInfo *map_move_vehicle
static XGC *map_erase_gc
static XGC *map_outline_gc

0.4.4.14 map_draw

Synopsis: Draw the bolts on the map. This procedure can be interrupted by pending events so the state is kept in map_pending_draw.

The procedure is as follows: For every pair of selected radios, the connectivity types are examined to determine a drawing mode. If the types are marked as new, the mode is draw_mode_force. Otherwise if there is non-empty box, the drawing mode is draw_mode_box. Otherwise nothing is drawn. In this last case, the pair is examined further to determine if the bolt requires animation in order to maintain the map_some_animated flag otherwise this flag is maintained as a side-effect of redrawing the bolt.

If the bolt is to be drawn, its shape is determined and the bolt is drawn using map_draw_bolt. The type and shape are recorded in the connectivity arrays.

After all bolts have been (re)drawn, color-map animation is started if necessary.

Type: static void
Arguments: <none>
Referenced by: static void map_start_draw()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
static struct /* unnamed */ map_pending_draw
static WidgetRec *map_body_widget
static MAP_DATA map_data
static char map_some_animated
static unsigned int map_color_timeout
static void map_color_animate()
static unsigned char map_style_of_type()
static unsigned char map_shape_of_types()
static char map_draw_bolt()

0.4.4.15 map_draw_bolt

Synopsis: Draw a lightning bolt between two radios according the indicated connectivity types. The shape has already been determined by the caller.

Type: static char

Arguments:			
	dpy	**Display ***	The X display on which to draw.
	wind	**Window**	The X window on which to draw.
	rp1	**RadioInfoP**	The first radio.
	rp2	**RadioInfoP**	The second radio.
	type1	**unsigned char**	The first connectivity.
	type2	**unsigned char**	The second connectivity.
	shape	**unsigned char**	The shape to use.
	mode	**DRAW_MODE**	The drawing mode.

Referenced by: static void map_draw()
static void map_erase()
References to: VehicleInfo **net_vehicles
static struct /* unnamed */ map_pending_draw
static XGC *map_erase_gc
static XGC *map_outline_gc
static XGC *map_receiving_gc[8]
static char map_some_animated
static unsigned char map_style_of_type()

static void map_gc_of_type()
static void map_bounding_box()

0.4.4.16 map_erase

Synopsis: This routine is similar to map_draw except that bolts are erased according to their current shape instead of being drawn in their new shape. Erasure occurs in anticipation of a subsequent draw. Thus if the draw operation will completely cover the current bolt, no erasure is necessary. This is an important optimization because the shape being erased is frequently covered by the shape being drawn and if the erasure is done it forces redrawing of all the bolts which intersect the erased bolt.

As bolts are erased, the area erased is accumulated to determine which bolts will need to be redrawn.

Type: static void

Arguments: <none>

Referenced by: static void map_start_erase()

References to: struct RadioInfoStruct **net_radios
int net_n_radios
static struct /* unnamed */ map_pending_draw
static WidgetRec *map_body_widget
static unsigned char map_style_of_type()
static unsigned char map_shape_of_types()
static char map_draw_bolt()
static void map_start_draw()

0.4.4.17 map_expose

Synopsis: If a map_draw operation is in progress, it is cancelled. If a map_erase operation is in progress, it is left alone. Then the exposed area is added to the map_pending_draw area. If a map_erase operation is not in progress a map_draw is started.

Type: static void

Arguments:

w	Widget	The exposed widget.
a	caddr_t	not used.
cback	XmDrawingAreaCallbackStruct *	Standard callback info.

Referenced by: static struct /* unnamed */ register_list[13]

References to: static struct /* unnamed */ map_pending_draw
static void map_start_draw()

0.4.4.18 map_freeze

Synopsis: Callback function for activation of the freeze button. Desensitizes the freeze button and sensitizes the unfreeze, step, and anomaly buttons. It also allows input to the time widget. If the traffic display is already frozen and has a valid cursor, the map time is advanced to the traffic cursor time. Otherwise the traffic cursor time is set to the map time.

Type: static void

Arguments: <none>

Referenced by: static struct /* unnamed */ register_list[13]
void map_display()
void map_reset()

References to: unsigned char traffic_cursor_time()
void traffic_set_cursor_time()
static char map_frozen
static WidgetRec *map_step_widget
static WidgetRec *map_anomaly_widget
static WidgetRec *map_freeze_widget
static WidgetRec *map_unfreeze_widget
static WidgetRec *map_time_widget
static int map_time

static void map_advance()

0.4.4.19 map_gc_of_type

Synopsis: Converts the connectivity types between two radios to the graphics contexts for filling the two halves of a lightning bolt.

Type: static void

Arguments:

type	unsigned char	The major connectivity type.
lesser_type	unsigned char	The minor connectivity type.
gcp1	GC *	How to draw the major end.
gcp2	GC *	How to draw the minor end.

Referenced by: static char map_draw_bolt()

References to:
static XGC *map_untuned_gc
static XGC *map_tuned_gc
static XGC *map_unconnected_gc
static XGC *map_connected_gc
static XGC *map_conflict_gc
static XGC *map_receiving_gc[8]

0.4.4.20 map_input

Synopsis: The input callback for vehicle boxes. Input processing is only performed to permit a vehicle box to be dragged to a new location.

Type: static void

Arguments:

w	Widget	The vehicle box widget.
vp	VehicleInfoP	The vehicle.
event	XEvent *	The X (mouse) event.

Referenced by:
static struct /* unnamed */ register_list[13]
static XtCallbackRec input_callbacks[2]
static void map_create_vehicle_widget()

References to:
static struct /* unnamed */ map_pending_draw
static WidgetRec *map_body_widget
static WidgetRec *map_move_widget
static int map_move_x
static int map_move_y
static char map_box_visible
static VehicleInfo *map_move_vehicle
static MAP_DATA map_data
static unsigned int map_width
static unsigned int map_height
static void map_color_radio()
static void map_color_vehicle()
static void map_display_move()

0.4.4.21 map_layout

Synopsis: Layout the vehicle widgets on the map. Vehicles are placed on a grid such that vehicles are ordered according to their east--west and north--south positions.

Type: static char

Arguments: <none>

Referenced by: void map_display()

References to:
VehicleInfo **net_vehicles
int net_n_vehicles
static MAP_DATA map_data
static unsigned int map_width
static unsigned int map_height
static int map_left_extension
static int map_right_extension
static int map_vehicle_height

static int map_layout_compare_x()
static int map_layout_compare_y()

0.4.4.22 map_layout_compare_x

Synopsis: Function for qsort to compare the x coordinate of two vehicles. Ties are broken by the location in memory of the vehicle data.
Type: static int
Arguments:

vpp1	VehicleInfoP *	First vehicle.
vpp2	VehicleInfoP *	Second vehicle.

Referenced by: static char map_layout()
References to: <nothing>

0.4.4.23 map_layout_compare_y

Synopsis: Function for qsort to compare the y coordinate of two vehicles. Ties are broken by the location in memory of the vehicle data.
Type: static int
Arguments:

vpp1	VehicleInfoP *	First vehicle.
vpp2	VehicleInfoP *	Second vehicle.

Referenced by: static char map_layout()
References to: <nothing>

0.4.4.24 map_measure

Synopsis: Update the left and right extension and height measurements for all selected vehicles.
Type: static void
Arguments: <none>
Referenced by: void map_display()
References to: VehicleInfo **net_vehicles
int net_n_vehicles
static int map_left_extension
static int map_right_extension
static int map_vehicle_height

0.4.4.25 map_more_radios

Synopsis: Schedules a map_display operation for 50 milliseconds from now if there is not already one scheduled. If a later map_display is scheduled, that is cancelled.
Type: static void
Arguments: <none>
Referenced by: static void map_select()
References to: void map_display()
static WidgetRec *map_widget
static unsigned int map_display_pending
static unsigned int map_display_pending_50

0.4.4.26 map_radio_transmit_time

Synopsis: Finds the latest time before the indicated time when the radio transmitted.
Type: static int
Arguments:

rp	RadioInfoP	The radio to examine.
when	MSEC	The time to seek.

Referenced by: static void map_advance()
References to: char tlFind()
char tlPrev()

0.4.4.27 map_recolor

Synopsis: Recolor all vehicles and radios as needed.
Type: static void

Arguments: <none>
Referenced by: void map_display()
static void map_update()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
VehicleInfo **net_vehicles
int net_n_vehicles
static void map_color_radio()
static void map_color_vehicle()

0.4.4.28 map_remanage_vehicle

Synopsis: Change the vehicle appearance according to the new selection state for the radios of the vehicle. The vehicle is unmanaged if neither radio is selected. If a radio is selected, then the label and separator for that radio are managed, if necessary. If a radio is not selected, then the label and separator for that radio are unmanaged, if necessary. If either radio is selected, then the vehicle is managed, if necessary.
Type: static char
Arguments: vp VehicleInfoP The vehicle to re-manage.
Referenced by: void map_display()
References to: <nothing>

0.4.4.29 map_reset_pending_draw

Synopsis: Reset the map_pending_draw information to be empty.
Type: static void
Arguments: <none>
Referenced by: static void map_update()
References to: static struct /* unnamed */ map_pending_draw

0.4.4.30 map_resize

Synopsis: The new dimensions of the map_body_widget are recorded.
Type: static void
Arguments: w Widget The resized widget.
Referenced by: static struct /* unnamed */ register_list[13]
static void map_create_widget()
References to: static unsigned int map_width
static unsigned int map_height

0.4.4.31 map_select

Synopsis: This is the callback function for selecting radios to present in the connectivity map. The principal function is to set the r_map.selected boolean for every radio which is selected and to clear the boolean for every radio which is not selected. The manual placement of the vehicle of newly selected radios is cancelled and vehicle recoloring is scheduled as necessary. map_more_radios is called to schedule the map_display.
Type: static void
Arguments: w Widget The button widget.
a caddr_t Not used.
cback XmListCallbackStruct * Callback info.
Referenced by: static struct /* unnamed */ register_list[13]
References to: struct RadioInfoStruct **net_radios
VehicleInfo **net_vehicles
static int map_n_radio_buttons
static void map_more_radios()

0.4.4.32 map_set_connectivity

Synopsis: Set the connectivity of "rp1" from the "idx" transmitter to "new_connectivity". Filters out redundant connectivity changes. Marks the connectivity as "con_new". Returns bits indicating what kind of changes occurred.
Type: static int
Arguments:

rp1	RadioInfoP	The radio to set the connectivity of.
idx	int	The index of the other radio.
new_connectivity	unsigned char	The new connectivity.

Referenced by: static void map_advance()
References to: <nothing>

0.4.4.33 map_set_radio_time

Synopsis: Sets the pointer into a radio's data to the location corresponding to the specified time.
Type: static void
Arguments:

rp	RadioInfoP	The radio to set.
when	MSEC	The time to set.

Referenced by: static void map_advance()
References to: STATE *state_find()
int net_max_radios
char tlFind()
STATE initial_state
int bzero()

0.4.4.34 map_set_time

Synopsis: Change the value displayed in the map_time_widget.
Type: static void
Arguments:

t	MSEC	The new time to be displayed.

Referenced by: static void map_time_callback()
static void map_create_widget()
void map_reset()
static void map_advance()
References to: static WidgetRec *map_time_widget
static int map_time

0.4.4.35 map_shape_of_types

Synopsis: Returns the appropriate bolt shape for the connectivity types between a pair of radios.
Type: static unsigned char
Arguments:

type1	unsigned char	First connectivity.
type2	unsigned char	Second connectivity.

Referenced by: static void map_draw()
static void map_erase()
References to: <nothing>

0.4.4.36 map_start_draw

Synopsis: Reset the map_pending_draw state and start drawing.
Type: static void
Arguments: <none>
Referenced by: static void map_erase()
static void map_expose()
References to: static struct /* unnamed */ map_pending_draw
static void map_draw()

0.4.4.37 map_start_erase

Synopsis: Reset map_pending_draw and start erasing.
Type: static void

Arguments: <none>
Referenced by: static void map_update()
References to: static struct /* unnamed */ map_pending_draw
static void map_erase()

0.4.4.38 map_step

Synopsis: Callback function for activating the step button. Calls map_advance to advance one step.
Type: static void
Arguments: <none>
Referenced by: static struct /* unnamed */ register_list[13]
References to: int now
void traffic_set_cursor_time()
static int map_time
static void map_advance()

0.4.4.39 map_step_to_anomaly

Synopsis: Callback function for activating the anomaly button. Calls map_advance to advance to the next anomaly.
Type: static void
Arguments: <none>
Referenced by: static struct /* unnamed */ register_list[13]
References to: int now
void traffic_set_cursor_time()
static int map_time
static void map_advance()

0.4.4.40 map_style_of_type

Synopsis: Converts a connectivity type to a lightning bolt style.
Type: static unsigned char
Arguments: **type** **unsigned char** Connectivity type.
Referenced by: static char map_draw_bolt()
static void map_draw()
static void map_erase()
References to: static MAP_DATA map_data

0.4.4.41 map_time_callback

Synopsis: This is the callback function for typing in a new map time. The text of the time widget is parsed into a new time and the map display advanced to that time. If the time cannot be parsed, the time is reset to the current map time.
Type: static void
Arguments:

w	**Widget**	The text widget.
a	**caddr_t**	Not used.
cback	**XmAnyCallbackStruct ***	Callback info.

Referenced by: static struct /* unnamed */ register_list[13]
References to: int now
void traffic_set_cursor_time()
static int map_time
static void map_advance()
static void map_set_time()

0.4.4.42 map_unfreeze

Synopsis: Callback function for activation of the unfreeze button. Desensitizes the step, anomaly, and unfreeze widgets, sensitizes the freeze button, and disables input to the time widget. The map time is advanced to the current time.

Type: static void
Arguments: <none>
Referenced by: static struct /* unnamed */ register_list[13]
static void map_create_widget()
void map_reset()
References to: int now
static char map_frozen
static WidgetRec *map_step_widget
static WidgetRec *map_anomaly_widget
static WidgetRec *map_freeze_widget
static WidgetRec *map_unfreeze_widget
static WidgetRec *map_time_widget
static void map_advance()

0.4.4.43 map_update

Synopsis: Update the map display to reflect its new state. Recolors everything and starts erasing bolts.
Type: static void
Arguments: <none>
Referenced by: static void map_advance()
References to: static struct /* unnamed */ map_pending_draw
static WidgetRec *map_widget
static void map_recolor()
static void map_reset_pending_draw()
static void map_start_erase()

0.5 net.c

0.5.1 Global Variables

0.5.1.1 net_connectivity

Synopsis: The connectivity arrays for every radio.
Type: unsigned char **
Referenced by: char *net_read_file()
void netReplenish()
void netChangeState()

0.5.1.2 net_max_radios

Synopsis: The total number of elements in net_radios.
Type: int
Referenced by: void map_init_radio()
static void map_set_radio_time()
char *net_read_file()
void netReplenish()

0.5.1.3 net_max_vehicles

Syncpsis: The total number of elements in net_vehicles.
Type: int
Referenced by: static int VehicleIDtoIndex()
char *net_read_file()
void netReplenish()

0.5.1.4 net_n_radios

Synopsis: The current number of used elements in net_radios.
Type: int

Referenced by: void resetRadios()
static void TimeoutRadios()
static void map_draw()
static void map_erase()
static void map_create_radio_buttons()
static void map_recolor()
void map_reset()
static void map_advance()
void map_advance_display()
void UpdateRadioTraffic()
void UpdateRadioStatus()
static void state_new_radio_buttons()
static void state_display()
void state_update()
void InitSimNetwork()
void net_destroy()
char *net_read_file()
int net_count_file()
void net_write_file()
void netReplenish()
void netChangeState()

0.5.1.5 net_n_vehicles

Synopsis: The current number of used elements in net_vehicles.
Type: int
Referenced by: void DestroyVehicles()
static void TimeoutVehicles()
static void map_measure()
static char map_layout()
void map_display()
static void map_recolor()
void net_destroy()
static int VehicleIDtoIndex()
char *net_read_file()
int net_count_file()
void net_write_file()
void netReplenish()

0.5.1.6 net_radios

Synopsis: An array of (pointers to) radio descriptions.
Type: struct RadioInfoStruct **
Referenced by: void resetRadios()
static void TimeoutRadios()
char *state_radio_name()
static void map_select()
static void map_draw()
static void map_erase()
static void map_create_radio_buttons()
static void map_recolor()
void map_reset()
static void map_advance()
void UpdateRadioTraffic()
void UpdateRadioStatus()
static void state_new_radio_buttons()
static void state_display()
char *net_read_file()

int net_count_file()
void net_write_file()
void netReplenish()
void netChangeState()

0.5.1.7 net_vehicles

Synopsis: An array of (pointers to) vehicle descriptions.
Type: VehicleInfo **
Referenced by: void DestroyVehicles()
static void TimeoutVehicles()
char *radio_name()
static void map_select()
static char map_draw_bolt()
static void map_color_radio()
static void map_measure()
static char map_layout()
void map_display()
static void map_recolor()
static int VehicleIDtoIndex()
char *net_read_file()
void net_write_file()
void netReplenish()
static void ProcessVehicleAppearancePDU()
static void ProcessDeactivatePDU()
static void ProcessStatusChangePDU()
static void ProcessVehicleStatusPDU()
static void ProcessTransmitterPDU()
static void ProcessReceiverPDU()

0.5.1.8 networkInterface

Synopsis: This is a handle on the network interface allowing access to information such as the ethernet address of the interface.
Type: int
Referenced by: static void slow_periodic()
static void medium_periodic()
int resetTime()
void InitSimNetwork()
void ReadPDUs()
char *net_geteaddr()

0.5.1.9 vehicle_ht

Synopsis: Hash table for translating vehicle ids into net_vehicles index.
Type: HTABLE *
Referenced by: void InitSimNetwork()
void net_destroy()
static int VehicleIDtoIndex()

0.5.2 Local Variables

0.5.2.1 network_h_rcsid

Synopsis: the RCS id of network.h.
Type: static char

0.5.2.2 ownAddress

Synopsis: .

Type: static struct /* unnamed */
Referenced by: void InitSimNetwork()

0.5.2.3 rcsid

Synopsis: The RCS id of this file.
Type: static char

0.5.3 Global Functions

0.5.3.1 DrainPDUs

Synopsis: Read and discard all pending PDUs. Eliminates old PDUs that might have arrived during prehistoric times.
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
static void file_read_file()
References to: <nothing>

0.5.3.2 InitSimNetwork

Synopsis: Initialize processing of Ethernet communications.
Type: void
Arguments: <none>
Referenced by: int main()
References to: HTABLE *htInit()
SimulationEnvironment simEnv
int net_n_radios
static struct /* unnamed */ ownAddress
int networkInterface
HTABLE *vehicle_ht

0.5.3.3 ReadPDUs

Synopsis: Read and process PDUs received from the simulation network.
Type: void
Arguments: <none>
Referenced by: static void fast_periodic()
References to: int now
int time_zero
SimulationEnvironment simEnv
int networkInterface
static void ProcessDeactivatePDU()
static void ProcessStatusChangePDU()
static void ProcessTransmitterPDU()
static void ProcessReceiverPDU()
static void ProcessVehicleAppearancePDU()
static void ProcessVehicleStatusPDU()

0.5.3.4 netChangeState

Synopsis: Record a new state for a radio. The first time a radio is mentioned, its initial state is created and its position in the net_radios vector is established.
Type: void

Arguments:			
	rp	RadioInfoP	The radio.
	st	STATEP	The new state.
	timestamp	MSEC	When the state change occured.

Referenced by: void resetRadios()
static void TimeoutRadios()

static void ProcessVehicleAppearancePDU()
static void ProcessStatusChangePDU()
static void ProcessVehicleStatusPDU()
static void ProcessTransmitterPDU()
static void ProcessReceiverPDU()

References to: STATE *state_find()
struct RadioInfoStruct **net_radios
int net_n_radios
unsigned char traffic_need_update
void tlChangeState()
void map_init_radio()
unsigned char **net_connectivity

0.5.3.5 netReplenish

Synopsis: Replenish resources to be consumed at "interrupt" level. Interrupt routines are not permitted to malloc memory so we preallocate memory here.
Type: void
Arguments: <none>
Referenced by: int main()
static void slow_periodic()
char *net_read_file()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
int net_max_radios
VehicleInfo **net_vehicles
int net_n_vehicles
int net_max_vehicles
void InitVehicle()
unsigned char **net_connectivity
int bzero()

0.5.3.6 net_count_file

Synopsis: Predict how many lines of output will be generated when writing the net information portion of a radmon file.
Type: int
Arguments: f FILE * Not used.
Referenced by: static void file_write_file()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
int net_n_vehicles
int tlCountFile()

0.5.3.7 net_destroy

Synopsis: Release net resources and reset to ground state.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: HTABLE *htInit()
void htFree()
int net_n_radios
int net_n_vehicles
HTABLE *vehicle_ht

0.5.3.8 net_geteaddr

Synopsis: Get a string for our net address.
Type: char *

Arguments: <none>
Referenced by: void Online()
References to: int networkInterface

0.5.3.9 net_read_file

Synopsis: Read the net data from a radmon file. The net data consists of the vehicle descriptions and the radio descriptions.
Type: char *
Arguments:

f	FILE *	The file to read from.
version	int	The file format version.

Referenced by: static void file_read_file()
References to:
char *strncpy()
struct RadioInfoStruct **net_radios
int net_n_radios
int net_max_radios
VehicleInfo **net_vehicles
int net_n_vehicles
int net_max_vehicles
void netReplenish()
char *tlReadFile()
void map_init_radio()
void map_vehicle_appeared()
char *file_gets()
unsigned char **net_connectivity
static int VehicleIDtoIndex()

0.5.3.10 net_write_file

Synopsis: Write the net portion of a radmon file.
Type: void
Arguments:

f	FILE *	The file to write to.

Referenced by: static void file_write_file()
References to:
struct RadioInfoStruct **net_radios
int net_n_radios
VehicleInfo **net_vehicles
int net_n_vehicles
char *vehicle_name()
void tlWriteFile()
void file_count()

0.5.4 Local Functions

0.5.4.1 ProcessDeactivatePDU

Synopsis: ProcessDeactivatePDU.
Type: static void
Arguments:

pdu	DeactivateResponseVariant *	The PDU.
timestamp	MSEC	When the PDU arrived.

Referenced by: void ReadPDUs()
References to:
SimulationEnvironment simEnv
VehicleInfo **net_vehicles
static int VehicleIDtoIndex()

0.5.4.2 ProcessReceiverPDU

Synopsis: Update what is known about a receiver from a Receiver PDU describing it.
Type: static void

Arguments: pdu **register ReceiverVariant ***
The PDU.
timestamp **MSEC** When the PDU arrived.
Referenced by: void ReadPDUs()
References to: VehicleInfo **net_vehicles
void netChangeState()
static int VehicleIDtoIndex()

0.5.4.3 ProcessStatusChangePDU

Synopsis: ProcessStatusChangePDU.
Type: static void
Arguments: **pdu** **StatusChangeVariant ***
The PDU.
timestamp **MSEC** When the PDU arrived.
Referenced by: void ReadPDUs()
References to: VehicleInfo **net_vehicles
void netChangeState()
static int VehicleIDtoIndex()

0.5.4.4 ProcessTransmitterPDU

Synopsis: Update what is known about a transmitter from a Transmitter PDU describing it.
Type: static void
Arguments: **pdu** **register TransmitterVariant ***
The PDU.
timestamp **MSEC** When the PDU arrived.
Referenced by: void ReadPDUs()
References to: TUNE *tune_find()
VehicleInfo **net_vehicles
void netChangeState()
static int VehicleIDtoIndex()

0.5.4.5 ProcessVehicleAppearancePDU

Synopsis: Mark a vahicle as existing. The first time a vehicle is seen its position in the net_vehicles vector is established.
Type: static void
Arguments: **pdu** **VehicleAppearanceVariant ***
The vehicle appearance PDU.
timestamp **MSEC** When the PDU arrived.
Referenced by: void ReadPDUs()
References to: VehicleInfo **net_vehicles
void netChangeState()
void map_vehicle_appeared()
static int VehicleIDtoIndex()

0.5.4.6 ProcessVehicleStatusPDU

Synopsis: ProcessVehicleStatusPDU.
Type: static void
Arguments: **pdu** **VehicleStatusVariant ***
The PDU.
timestamp **MSEC** When the PDU arrived.
Referenced by: void ReadPDUs()
References to: VehicleInfo **net_vehicles
void netChangeState()
static int VehicleIDtoIndex()

0.5.4.7 VehicleIDtoIndex

Synopsis: Looks up the vehicle id in the vehicle id hash table and returns the value found there, if any. If the vehicle is not found, an element of net_vehicles is allocated and entered into the hash table.
Type: static int
Arguments: vehicleID VehicleID The vehicle ID to look up.
Referenced by: char *net_read_file()
static void ProcessVehicleAppearancePDU()
static void ProcessDeactivatePDU()
static void ProcessStatusChangePDU()
static void ProcessVehicleStatusPDU()
static void ProcessTransmitterPDU()
static void ProcessReceiverPDU()
References to: HASHENTRY *htFind()
HASHENTRY *htInsert()
VehicleInfo **net_vehicles
int net_n_vehicles
int net_max_vehicles
HTABLE *vehicle_ht

0.6 radmon.c

0.6.1 Global Variables

0.6.1.1 ethernet_address

Synopsis: The string representation of our ethernet address.
Type: char *
Referenced by: static void set_identification_label()
void Online()
static void file_write_file()
static void file_read_file()

0.6.1.2 file_name

Synopsis: The name of the file of offline data.
Type: char *
Referenced by: static void set_identification_label()
static void file_read_file()

0.6.1.3 host_name

Synopsis: The name of this host.
Type: char *
Referenced by: static void set_identification_label()
void Online()
static void file_write_file()
static void file_read_file()

0.6.1.4 initial_state

Synopsis: A constant which is the initial state of any radio.
Type: STATE
Referenced by: void InitVehicle()
static void TimeoutRadios()
void map_init_radio()
static void map_set_radio_time()

0.6.1.5 now

Synopsis: The current (simulation) time.
Type: int
Referenced by: static void slow_periodic()
static void medium_periodic()
int resetTime()
void InitVehicle()
void DestroyVehicles()
static void TimeoutRadios()
static void TimeoutVehicles()
static void map_time_callback()
static void map_step()
static void map_step_to_anomaly()
static void map_unfreeze()
static void map_advance()
static void traffic_set_scrollbar()
unsigned char traffic_cursor_time()
static int traffic_draw_slot()
static void traffic_draw_grid()
static char traffic_left()
static void traffic_fit()
static void traffic_input()
static void traffic_scrollbar()
static void state_display()
void ReadPDUs()
static void file_write_file()
static void file_read_file()

0.6.1.6 online_mode

Synopsis: True if online data is being monitored.
Type: unsigned char
Referenced by: static void reset_callback()
int main()
static void slow_periodic()
static void medium_periodic()
static void set_identification_label()
void Offline()
void Online()
void map_display()
void map_reset()
static void traffic_unfreeze()

0.6.1.7 periodic_enabled

Synopsis: True if the periodic signal handler is enabled.
Type: unsigned char
Referenced by: int main()
void EnablePeriodic()
void DisablePeriodic()
static void fast_periodic()

0.6.1.8 simEnv

Synopsis: Information about the simulation being monitored.
Type: SimulationEnvironment
Referenced by: void display_init()
int main()
void InitSimNetwork()

void ReadPDUs()
static void ProcessDeactivatePDU()

0.6.1.9 time_zero

Synopsis: The time at the beginning of monitoring.
Type: int
Referenced by: static void slow_periodic()
static void medium_periodic()
int resetTime()
void ReadPDUs()

0.6.1.10 tod

Synopsis: Unix time of day.
Type: int
Referenced by: static void set_identification_label()
void Online()
static void file_write_file()
static void file_read_file()

0.6.1.11 traffic_need_update

Synopsis: True if the traffic display must be updated.
Type: unsigned char
Referenced by: void UpdateRadioTraffic()
void netChangeState()

0.6.2 Local Variables

0.6.2.1 network_h_rcsid

Synopsis: RCS id of network.h.
Type: static char

0.6.2.2 rcsid

Synopsis: RCS id of this file.
Type: static char

0.6.3 Global Functions

0.6.3.1 DestroyEverything

Synopsis: Call all the destroy routines to release all the resources associated with the current data.
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
static void file_read_file()
References to: void display_deselect()
void net_destroy()
void DestroyVehicles()
void status_destroy()
void traffic_destroy()
void state_destroy()
void tune_destroy()
void map_destroy()

0.6.3.2 DestroyVehicles

Synopsis: Reinitialize the net_vehicles to it's initial state. Resources are released.

Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: int now
VehicleInfo **net_vehicles
int net_n_vehicles
void tlFree()
void DestroyWidget()

0.6.3.3 DestroyWidget

Synopsis: Destroy a widget insuring that the widget exists and that the point is nullified afterward.
Type: void
Arguments: wp Widget * The widget to destroy.
Referenced by: void DestroyVehicles()
References to: <nothing>

0.6.3.4 DisablePeriodic

Synopsis: Turns the fast periodic processing off.
Type: void
Arguments: <none>
Referenced by: static void dump_callback()
static void reset_callback()
void tuneReplenishFreeList()
void tlReplenishFreeList()
void stateReplenishFreeList()
static void file_write_file()
static void file_read_file()
References to: unsigned char periodic_enabled

0.6.3.5 EnablePeriodic

Synopsis: Turns the fast periodic processing back on.
Type: void
Arguments: <none>
Referenced by: static void dump_callback()
static void reset_callback()
static void slow_periodic()
static void fast_periodic()
References to: unsigned char periodic_enabled
static void fast_periodic()

0.6.3.6 InitVehicle

Synopsis: Initialize a vehicle table entry. Called as net_vehicles grows to initialize new entries.
Type: void
Arguments: vp VehicleInfoP The vehicle to initialize.
vIdx Int Vehicle index for this vehicle.
Referenced by: void netReplenish()
References to: int now
STATE *state_find()
STATE initial_state

0.6.3.7 Offline

Synopsis: Perform various actions to switch the monitor from online mode to offline mode.
Type: void
Arguments: <none>
Referenced by: static void file_read_file()

References to: unsigned char online_mode
void traffic_reset()
void map_reset()
static void set_identification_label()

0.6.3.8 Online

Synopsis: Perform various actions to switch the monitor from offline mode to online mode.
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
int main()
static void file_read_file()
References to: unsigned char online_mode
char *ethernet_address
char *host_name
int tod
char *net_geteaddr()
void map_reset()
static void set_identification_label()

0.6.3.9 exit_gracefully

Synopsis: Clean up before exiting.
Type: void
Arguments: <none>
Referenced by: static void quit_callback()
int main()
References to: <nothing>

0.6.3.10 main

Synopsis: Program entry point.
Type: int
Arguments:

argc	int	Number of arguments.
argv	char **	The argument array.

Referenced by: <nothing>
References to: void tune_init()
void stateReplenishFreeList()
void state_init()
SimulationEnvironment simEnv
unsigned char online_mode
void display_init()
void InitSimNetwork()
void netReplenish()
void exit_gracefully()
void Online()
void tlReplenishFreeList()
void tuneReplenishFreeList()
void htFreeInit()
void htReplenish()
unsigned char periodic_enabled
static void print_banner()
static void medium_periodic()
static void slow_periodic()
int resetTime()

0.6.3.11 radio_name

Synopsis: Generate the name of a radio. A radio is named as <vehicle name>/[A|B] according to whether this is the first or second radio in the vehicle.
Type: char *
Arguments: rp RadioInfoP Radio to name.
Referenced by: char *state_radio_name()
static void map_create_radio_buttons()
static char traffic_add_pos()
static int traffic_create_radio_button()
static void create_status_widget()
static void create_status_select_button()
static void state_new_radio_buttons()
static int state_compile_statistics()
References to: VehicleInfo **net_vehicles
char *vehicle_name()

0.6.3.12 resetRadios

Synopsis: Discard state information for all radios. The radios are otherwise retained.
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
void netChangeState()
void tlFree()

0.6.3.13 resetTime

Synopsis: Called to reset the simulation clock time to zero.
Type: int
Arguments: <none>
Referenced by: static void reset_callback()
int main()
References to: int now
int time_zero
int networkInterface

0.6.3.14 state_radio_name

Synopsis: Generates the name of the radio which is transmitting in a particular state.
Type: char *
Arguments: st STATEP The state containing the transmitter.
Referenced by: void status_update()
static void state_format_status()
References to: struct RadioInfoStruct **net_radios
char *radio_name()

0.6.3.15 vehicle_name

Synopsis: Generate a vehicle name. A vehicle is named <battalion><company><bumper> if that information is known otherwise it is named with its vehicle ID.
Type: char *
Arguments: vp VehicleInfoP Vehicle to name.
Referenced by: char *radio_name()
static void map_create_vehicle_widget()
void net_write_file()
References to: <nothing>

0.6.4 Local Functions

0.6.4.1 TimeoutRadios

Synopsis: Mark radios for which no transmitterPDU has been received for 12 seconds as non–existant.
Type: static void
Arguments: <none>
Referenced by: static void slow_periodic()
References to: int now
struct RadioInfoStruct **net_radios
int net_n_radios
void netChangeState()
STATE initial_state

0.6.4.2 TimeoutVehicles

Synopsis: Mark vehicles for which no appearancePDU has been received for 12 seconds as non–existant.
Type: static void
Arguments: <none>
Referenced by: static void slow_periodic()
References to: int now
VehicleInfo **net_vehicles
int net_n_vehicles
void map_vehicle_vanished()

0.6.4.3 fast_periodic

Synopsis: This routine is invoked frequently via a alarm clock signal to process incoming PDUs.
Type: static void
Arguments: <none>
Referenced by: void EnablePeriodic()
References to: void ReadPDUs()
void EnablePeriodic()
unsigned char periodic_enabled

0.6.4.4 medium_periodic

Synopsis: Timer routine called every 1/4 second to advance the map display.
Type: static void
Arguments: <none>
Referenced by: int main()
static void medium_periodic()
References to: int now
int time_zero
unsigned char online_mode
void map_advance_display()
int networkInterface
static void medium_periodic()

0.6.4.5 print_banner

Synopsis: Print banner.
Type: static void
Arguments: <none>
Referenced by: int main()
References to: <nothing>

0.6.4.6 set_identification_label

Synopsis: Set the identification label widget to identify the data currently being presented. This may data from a file or it may be on–line currently being acquired from the simulation network.

Type: static void
Arguments: <none>
Referenced by: void Offline()
void Online()
References to: char *strcpy()
char *strncpy()
unsigned char online_mode
char *ethernet_address
char *host_name
char *file_name
int tod
void set_widget_label()
WidgetRec *label_widget

0.6.4.7 slow_periodic

Synopsis: Timer routine to update things such as the various displays about every second. Also replenishes resources used by AST level code.
Type: static void
Arguments: <none>
Referenced by: int main()
static void slow_periodic()
References to: int now
int time_zero
void stateReplenishFreeList()
void state_update()
unsigned char online_mode
void netReplenish()
void EnablePeriodic()
void UpdateRadioStatus()
void UpdateRadioTraffic()
void tlReplenishFreeList()
void tuneReplenishFreeList()
void map_advance_display()
void htReplenish()
int networkInterface
static void TimeoutRadios()
static void TimeoutVehicles()
static void slow_periodic()

0.7 state.c

0.7.1 Global Variables

0.7.1.1 state_sub_function_closure

Synopsis: Describes the state sub–function menu.
Type: FB_DATA
Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: static WidgetRec *state_sub_function_attach_widget
static WidgetRec *state_sub_function_widget
static FUNCTION_BUTTON_CLOSURE state_button_closure

0.7.2 Local Variables

0.7.2.1 buttonl_h_rcsid

Synopsis: RCS id of buttonl.h.
Type: static char *

0.7.2.2 network_h_rcsid

Synopsis: RCS id of network.h.
Type: static char

0.7.2.3 rcsid

Synopsis: RCS id of this file.
Type: static char

0.7.2.4 state_attribute_attach_closure

Synopsis: This constant structure describes the attribute selection menu.
Type: static FB_DATA
Referenced by: static FUNCTION_BUTTON state_buttons[3]
References to: static WidgetRec *state_attribute_widget
static WidgetRec *state_attribute_attach_widget

0.7.2.5 state_attribute_attach_widget

Synopsis: The attribute selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_attribute_attach_closure
void stateInitWidgets()

0.7.2.6 state_attribute_buttons

Synopsis: This constant structure describes the buttons used to compose the attribute filter.
Type: static STATE_FILTER
Referenced by: static void state_attr_select()
void stateRegisterNames()

0.7.2.7 state_attribute_widget

Synopsis: The frame around the attribute selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_attribute_attach_closure
void stateInitWidgets()

0.7.2.8 state_body_widget

Synopsis: The text widget in which the state display appears.
Type: static WidgetRec *
Referenced by: static void state_expose_callback()
static void state_create_widget()
static void state_display()

0.7.2.9 state_button_closure

Synopsis: The current sub-function state.
Type: static FUNCTION_BUTTON_CLOSURE
Referenced by: FB_DATA state_sub_function_closure
References to: static FUNCTION_BUTTON state_buttons[3]

0.7.2.10 state_buttons

Synopsis: This constant structure describes the state sub-function menu.

Type: static FUNCTION_BUTTON
Referenced by: static FUNCTION_BUTTON_CLOSURE state_button_closure
References to: static FB_DATA state_attribute_attach_closure
static FB_DATA state_radio_attach_closure
static FB_DATA state_status_attach_closure

0.7.2.11 state_data

Synopsis: Resource data pertaining to the state display.
Type: static STATE_DATE
Referenced by: static void state_expose_callback()
static void state_create_widget()
static void state_finish_com()
static void state_display()

0.7.2.12 state_derived_com_label1

Synopsis: The heading for derived information.
Type: static DERIVED_COM
Referenced by: static void state_expose_callback()
static void state_display()
References to: static STATE_COM state_state_com_label1

0.7.2.13 state_derived_com_label2

Synopsis: The second line of heading for derived information.
Type: static DERIVED_COM
Referenced by: static void state_expose_callback()
static void state_display()
References to: static STATE_COM state_state_com_label2

0.7.2.14 state_derived_com_max

Synopsis: The maxima of all derived state attributes found during construction of the state data.
Type: static DERIVED_COM
Referenced by: static void state_draw_one_com()
static int state_format_statistics()
static void state_display()
References to: static STATE_COM state_state_com_max

0.7.2.15 state_filter

Synopsis: The state filter bit mask.
Type: static int
Referenced by: static char state_is_distinct()
static void state_draw_one_com()
static void state_attr_select()
void stateRegisterNames()
static int state_derived_com_compare()
static void state_display()
void state_update()

0.7.2.16 state_font_height

Synopsis: The height of a line in the state display font.
Type: static unsigned int
Referenced by: static void state_expose_callback()
static void state_create_widget()
static void state_display()

0.7.2.17 state_gc

Synopsis: The graphics context used for rendering the state text.
Type: static XGC *
Referenced by: static void state_draw_s_com()
static void state_create_widget()

0.7.2.18 state_ht

Synopsis: The hash table for looking up states. Each distinct sate is recorded once and reused for every repetition of that state. This hash table allows previously recorded states to be found.
Type: static HTABLE *
Referenced by: void state_init()
STATE *state_find()
void state_destroy()

0.7.2.19 state_label_widget

Synopsis: The text widget for the headingon the state display.
Type: static WidgetRec *
Referenced by: static void state_create_widget()
static void state_display()

0.7.2.20 state_margin

Synopsis: Margin around the state text.
Type: static unsigned int
Referenced by: static void state_display()

0.7.2.21 state_max_radio_com

Synopsis: Total number of entries in state_radio_com.
Type: static int
Referenced by: void state_init()
static void state_expand_radio_com()

0.7.2.22 state_max_state_comp

Synopsis: Total number of entries in state_state_com.
Type: static int
Referenced by: void state_init()
static void state_expand_state_comp()
static void state_display()

0.7.2.23 state_max_vector

Synopsis: The total number of entries in state_vector.
Type: static int
Referenced by: void stateReplenishFreeList()
static STATE *stateAllocate()
void state_init()
char *state_read_file()

0.7.2.24 state_n_radio_buttons

Synopsis: The current number of button in the radio selection menu.
Type: static int
Referenced by: void state_init()
static void state_new_radio_buttons()
static void state_display()
void state_update()
void state_destroy()

0.7.2.25 state_n_radio_com

Synopsis: Number of entries in state_radio_com in use.
Type: static int
Referenced by: void state_init()
static void state_expose_callback()
static void state_display()
void state_update()
void state_destroy()

0.7.2.26 state_n_state_comp

Synopsis: Number of entries in state_state_com in use.
Type: static int
Referenced by: void state_init()
static void state_display()
void state_update()
void state_destroy()

0.7.2.27 state_n_vector

Synopsis: The number of entries in state_vector which are in use.
Type: static int
Referenced by: void stateReplenishFreeList()
static STATE *stateAllocate()
void state_init()
int state_count_file()
void state_write_file()
char *state_read_file()
static void state_clear_dci()
static void state_display()
void state_update()
void state_destroy()

0.7.2.28 state_radio_attach_closure

Synopsis: This constant structure describes the radio selection menu.
Type: static FB_DATA
Referenced by: static FUNCTION_BUTTON state_buttons[3]
References to: static WidgetRec *state_radio_widget
static WidgetRec *state_radio_attach_widget

0.7.2.29 state_radio_attach_widget

Synopsis: The radio selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_radio_attach_closure
void stateInitWidgets()
static void state_new_radio_buttons()
static void state_display()
void state_destroy()

0.7.2.30 state_radio_com

Synopsis: Vector of available RADIO_COM structures.
Type: static RADIO_COM *
Referenced by: void state_init()
static void state_expose_callback()
static void state_expand_radio_com()
static void state_display()

0.7.2.31 state_radio_com_label1

Synopsis: The heading for radio information.
Type: static RADIO_COM
Referenced by: static void state_expose_callback()
static void state_display()

0.7.2.32 state_radio_com_label2

Synopsis: The second line of heading for radio information.
Type: static RADIO_COM
Referenced by: static void state_expose_callback()
static void state_display()

0.7.2.33 state_radio_com_max

Synopsis: The radio information corresponding to the maxima.
Type: static RADIO_COM
Referenced by: static void state_draw_one_com()
static int state_compile_stati cs()
static void state_display()

0.7.2.34 state_radio_widget

Synopsis: The frame around the radio selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_radio_attach_closure
void stateInitWidgets()

0.7.2.35 state_resources

Synopsis: Describes how to initialize state_data.
Type: static XtResource
Referenced by: static void state_create_widget()

0.7.2.36 state_sep_widget

Synopsis: The separator between the label and body widgets.
Type: static WidgetRec *
Referenced by: static void state_create_widget()
static void state_display()

0.7.2.37 state_space_width

Synopsis: The width of a space in the state display font.
Type: static unsigned int
Referenced by: static void state_draw_s_com()
static void state_draw_one_com()
static void state_create_widget()
static void state_display()

0.7.2.38 state_state_com_label1

Synopsis: The heading for state information.
Type: static STATE_COM
Referenced by: static DERIVED_COM state_derived_com_label1

0.7.2.39 state_state_com_label2

Synopsis: The second line of heading for state information.
Type: static STATE_COM
Referenced by: static DERIVED_COM state_derived_com_label2

0.7.2.40 state_state_com_max

Synopsis: The maxima of all state attributes found during construction of the state data.
Type: static STATE_COM
Referenced by: static DERIVED_COM state_derived_com_max
static void state_format_status()
static int state_format_tuning()
static int state_format()
static void state_display()

0.7.2.41 state_state_comp

Synopsis: Vector of available STATE_COMP structures.
Type: static STATE_COM **
Referenced by: void state_init()
static void state_expand_state_comp()
static void state_fill_one_radio()
static void state_display()

0.7.2.42 state_status_attach_closure

Synopsis: This constant structure describes the status selection menu.
Type: static FB_DATA
Referenced by: static FUNCTION_BUTTON state_buttons[3]
References to: static WidgetRec *state_status_widget
static WidgetRec *state_status_attach_widget

0.7.2.43 state_status_attach_widget

Synopsis: The status selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_status_attach_closure
void stateInitWidgets()

0.7.2.44 state_status_buttons

Synopsis: This constant structure describes the buttons used to compose the status filter.
Type: static STATE_FILTER
Referenced by: static void state_status_select()
void stateRegisterNames()

0.7.2.45 state_status_filter

Synopsis: The state status filter bit mask.
Type: static int
Referenced by: static void state_status_select()
void stateRegisterNames()
static void state_fill_one_radio()

0.7.2.46 state_status_widget

Synopsis: The frame around the status selection menu for the state display.
Type: static WidgetRec *
Referenced by: static FB_DATA state_status_attach_closure
void stateInitWidgets()

0.7.2.47 state_sub_function_attach_widget

Synopsis: The state subfunction menu.
Type: static WidgetRec *
Referenced by: FB_DATA state_sub_function_closure
void stateInitWidgets()

0.7.2.48 state_sub_function_widget

Synopsis: The frame around the state subfunction menu.
Type: static WidgetRec *
Referenced by: FB_DATA state_sub_function_closure
void stateInitWidgets()

0.7.2.49 state_vector

Synopsis: An array of STATE structures.
Type: static STATE **
Referenced by: void stateReplenishFreeList()
static STATE *stateAllocate()
void state_init()
void state_write_file()
static void state_expand_state_comp()
static void state_clear_dci()
STATE *state_nth()

0.7.2.50 state_widget

Synopsis: The frame in which the state display appears.
Type: static WidgetRec *
Referenced by: static void state_close_callback()
static void state_create_widget()
static void state_display()
void state_update()
void state_destroy()

0.7.3 Global Functions

0.7.3.1 stateInitWidgets

Synopsis: Get widget ids for various state widgets.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: WidgetRec *NameToWidget()
WidgetRec *main_widget
WidgetRec *sub_function_widget
static WidgetRec *state_radio_widget
static WidgetRec *state_radio_attach_widget
static WidgetRec *state_attribute_widget
static WidgetRec *state_attribute_attach_widget
static WidgetRec *state_status_widget
static WidgetRec *state_status_attach_widget
static WidgetRec *state_sub_function_widget
static WidgetRec *state_sub_function_attach_widget

0.7.3.2 stateRegisterNames

Synopsis: Register names for Mrm.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: static int state_filter
static int state_status_filter
static STATE_FILTER state_attribute_buttons[13]
static STATE_FILTER state_status_buttons[6]

0.7.3.3 stateReplenishFreeList

Synopsis: Interrupt routines can't allocate memory so we preallocate storage for interrupt routine use.
Type: void
Arguments: <none>
Referenced by: int main()
static void slow_periodic()
void state_init()
char *state_read_file()
void state_destroy()
References to: void DisablePeriodic()
static STATE **state_vector
static int state_n_vector
static int state_max_vector

0.7.3.4 state_count_file

Synopsis: Count how many lines of output will be written for the state section of a radmon file.
Type: int
Arguments: f FILE * Not used.
Referenced by: static void file_write_file()
References to: static int state_n_vector

0.7.3.5 state_destroy

Synopsis: Release state related storage.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: HTABLE *htInit()
void htFree()
void stateReplenishFreeList()
static HTABLE *state_ht
static WidgetRec *state_radio_attach_widget
static WidgetRec *state_widget
static int state_n_vector
static int state_n_radio_com
static int state_n_state_comp
static int state_n_radio_buttons

0.7.3.6 state_find

Synopsis: Find the state vector entry for the given state. If the state is not found, a entry is made.
Type: STATE *
Arguments: st STATEP The state to find.
Referenced by: void InitVehicle()
void map_init_radio()
static void map_set_radio_time()
char *state_read_file()
void netChangeState()
References to: HASHENTRY *htFind()
HASHENTRY *htInsert()
static HTABLE *state_ht
static STATE *stateAllocate()

0.7.3.7 state_function

Synopsis: Activate callback for the state button. Generates the state display.
Type: void
Arguments: <none>

Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: static void state_display()

0.7.3.8 state_init

Synopsis: Initialize state things.
Type: void
Arguments: <none>
Referenced by: int main()
References to: HTABLE *htInit()
void stateReplenishFreeList()
static HTABLE *state_ht
static STATE **state_vector
static int state_n_vector
static int state_max_vector
static RADIO_COM *state_radio_com
static int state_n_radio_com
static int state_max_radio_com
static STATE_COM **state_state_comp
static int state_n_state_comp
static int state_max_state_comp
static int state_n_radio_buttons

0.7.3.9 state_nth

Synopsis: Return the nth state from the state_vector.
Type: STATE *
Arguments:

n	int	Index of desired state.

Referenced by: char *tlReadFile()
References to: static STATE **state_vector

0.7.3.10 state_read_file

Synopsis: Read the state section of a radmon file.
Type: char *
Arguments:

f	FILE *	The file to read.
version	int	The file format version.

Referenced by: static void file_read_file()
References to: void htReplenish()
void stateReplenishFreeList()
STATE *state_find()
char *file_gets()
static int state_n_vector
static int state_max_vector
TUNE *tune_nth()

0.7.3.11 state_update

Synopsis: Update the state display because new states or new radios exist.
Type: void
Arguments: <none>
Referenced by: static void slow_periodic()
References to: int net_n_radios
static WidgetRec *state_widget
static int state_n_vector
static int state_n_radio_com
static int state_n_state_comp
static int state_filter
static int state_n_radio_buttons
static void state_display()

0.7.3.12 state_write_file

Synopsis: Write the state section of a radmon file.
Type: void
Arguments:

f	FILE *	The file to write.

Referenced by: static void file_write_file()
References to: void file_count()
static STATE **state_vector
static int state_n_vector

0.7.4 Local Functions

0.7.4.1 stateAllocate

Synopsis: Allocate a state_vector entry.
Type: static STATE *
Arguments: <none>
Referenced by: STATE *state_find()
References to: static STATE **state_vector
static int state_n_vector
static int state_max_vector

0.7.4.2 state_attr_select

Synopsis: Activate callback for the attribute button list. The complication here is that certain buttons are coupled together. Selecting certain buttons forces other buttons to be selected as well and deselecting certain buttons forces other buttons to be deselected. The procedure is as follows:

1. Build a mask of the filter bits for all selected buttons.
2. Compute the mask of buttons which became unselected (clr_bits).
3. Compute the mask of buttons which became selected (set_bits).
4. Compute the mask of buttons which should also be selected by or'ing together the set_coupled_bits for the newly selected buttons.
5. Similarly, compute the mask of buttons which should also be unselected by or'ing together the clr_coupled_bits for the newly deselected buttons.
6. Augment clr_bits by the also_clr bits and augment set_bits by the also_set bits.
7. Remove any bits being cleared from the bits being set.
8. Change the selection state of the buttons which are also_clr and also_set.
9. Remove the clr_bits from state_filter and add set_bits.

Type: static void
Arguments:

w	Widget	The button widget.
a	caddr_t	Not used.
cback	XmButtonLCallbackStruct *	Standard callback information.

Referenced by: static struct /* unnamed */ register_list[9]
References to: static int state_filter
static STATE_FILTER state_attribute_buttons[13]
static void state_display()

0.7.4.3 state_clear_dci

Synopsis: Clear the st_dci field of all states.
Type: static void
Arguments: <none>
Referenced by: static void state_display()
References to: static STATE **state_vector

static int state_n_vector

0.7.4.4 state_close_callback

Synopsis: Activate callback for the close button of the state display. Removes the state display from the screen.
Type: static void
Arguments: <none>
Referenced by: static struct /* unnamed */ register_list[9]
References to: static WidgetRec *state_widget

0.7.4.5 state_compile_statistics

Synopsis: Format the statistics for a radio (or radios), compute the radio field. Returns the number of distinct states.
Type: static int
Arguments:

rc	RADIO_COMP	Radio stuff.
total_time	MSEC	Clock.

Referenced by: static void state_display()
References to: char *radio_name()
static RADIO_COM state_radio_com_max
static void state_finish_com()
static int state_format_statistics()

0.7.4.6 state_create_widget

Synopsis: Create a new state widget.
Type: static void
Arguments: <none>
Referenced by: static void state_display()
References to: WidgetRec *NameToWidget()
static WidgetRec *state_widget
static WidgetRec *state_body_widget
static WidgetRec *state_sep_widget
static WidgetRec *state_label_widget
static unsigned int state_space_width
static unsigned int state_font_height
static XGC *state_gc
static STATE_DATE state_data
static XtResource state_resources[2]
struct /* unnamed */ *hierarchy_id
WidgetRec *general_area_widget

0.7.4.7 state_derived_com_compare

Synopsis: Compare two DERIVED_COMs for sorting.
Type: static int
Arguments:

dc1	DERIVED_COMP	First DERIVED_COM.
dc2	DERIVED_COMP	Second DERIVED_COM.

Referenced by: static void state_display()
References to: int tune_compare()
static int state_filter

0.7.4.8 state_display

Synopsis: Compute a new state display.
Type: static void
Arguments:

force	Boolean	True to force redisplay.

Referenced by: <nothing>
References to: <nothing>

0.7.4.9 state_draw_one_com

Synopsis: Draws one line of the state display. rc and dc give the S_COMs to be drawn.
Type: static void

Arguments:			
	d	Display *	The X display on which to draw.
	w	Window	The X window on which to draw.
	y	Position	Where to draw.
	rc	RADIO_COMP	What to draw.
	dc	DERIVED_COMP	What to draw.

Referenced by: static void state_expose_callback()
References to: static unsigned int state_space_width
static DERIVED_COM state_derived_com_max
static RADIO_COM state_radio_com_max
static int state_filter
static void state_draw_s_com()

0.7.4.10 state_draw_s_com

Synopsis: Draw an S_COM (state component). This is basically a string. The com_max argument gives width information to achieve a columnar format.
Type: static void

Arguments:			
	d	Display *	The X display on which to draw.
	w	Window	The X window on which to draw.
	xp	Position *	Where to draw (updated).
	y	Position	Where to draw.
	com	S_COM	What to draw.
	com_max	S_COM	Layout information.
	left_justify	Boolean	True if information is to be left justified.

Referenced by: static void state_draw_one_com()
References to: static unsigned int state_space_width
static XGC *state_gc

0.7.4.11 state_expand_derived_com

Synopsis: Increase the size of a DERIVED_COM vector.
Type: static void

Arguments:			
	rc	RADIO_COMP	RADIO_COM to expand.
	n	Int	New size.

Referenced by: static void state_fill_one_radio()
References to: int bzero()

0.7.4.12 state_expand_radio_com

Synopsis: Increase the size of the state_radio_com vector.
Type: static void

Arguments:			
	n_radios	Int	New size.

Referenced by: static void state_display()
References to: static RADIO_COM *state_radio_com
static int state_max_radio_com
int bzero()

0.7.4.13 state_expand_state_comp

Synopsis: Increase the size of the state_state_comp vector to accomodate more states.
Type: static void

Arguments:			
	n	Int	New size.

Referenced by: static void state_display()
References to: static STATE **state_vector
static STATE_COM **state_state_comp
static int state_max_state_comp

int bzero()

0.7.4.14 state_expose_callback

Synopsis: Expose callback for the state display. Loops through all states drawing the distinct ones.
Type: static void
Arguments: <none>
Referenced by: static struct /* unnamed */ register_list[9]
References to: static WidgetRec *state_body_widget
static unsigned int state_font_height
static DERIVED_COM state_derived_com_label1
static RADIO_COM state_radio_com_label1
static DERIVED_COM state_derived_com_label2
static RADIO_COM state_radio_com_label2
static RADIO_COM *state_radio_com
static int state_n_radio_com
static STATE_DATE state_data
static void state_draw_one_com()

0.7.4.15 state_fill_one_radio

Synopsis: Process the states of one radio and compute the derived state information. All the states the radio has ever been in are examined. The first time a state is examined, the corresponding derived information is initialized and attached to the state. In any case, the derived information is accumulated according to how many times the state is entered and how long the radio is in that state.
Type: static void
Arguments:

rc	RADIO_COMP	RADIO_COM to fill.
rp	RadioInfoP	Radio information.
state_now	MSEC	State clock.

Referenced by: static void state_display()
References to: static STATE_COM **state_state_comp
static int state_status_filter
static void state_expand_derived_com()

0.7.4.16 state_finish_com

Synopsis: Store a string as the value of an S_COM. Reallocate the space for the value if necessary. Update the maximum width in max_cm as needed.
Type: static void
Arguments:

cm	register S_COMP	The S_COMP to update.
max_cm	S_COMP	Layout information.
s	char *	The string to store.

Referenced by: static void state_format_status()
static int state_format_tuning()
static int state_format()
static int state_format_statistics()
static int state_compile_statistics()
static void state_display()
References to: char *strcpy()
static STATE_DATE state_data

0.7.4.17 state_format

Synopsis: Format the fields of a state.
Type: static int
Arguments:

sc	STATE_COMP	The state format.

Referenced by: static void state_display()
References to: static STATE_COM state_state_com_max
static void state_finish_com()

static void state_format_status()
static int state_format_tuning()

0.7.4.18 state_format_statistics

Synopsis: The statistics components of the state line is formatted.
Type: static int
Arguments:

dc	DERIVED_COMP	Derived information.
n	int	Length of information.
overall_time	MSEC	State clock.

Referenced by: static int state_compile_statistics()
References to: static DERIVED_COM state_derived_com_max
static char state_is_distinct()
static void state_finish_com()

0.7.4.19 state_format_status

Synopsis: Format the "status", "from", and "at" fields for a state.
Type: static void
Arguments:

st	STATEP	The state to be formatted.
sc	STATE_COMP	The format information.

Referenced by: static int state_format()
References to: char *state_radio_name()
static STATE_COM state_state_com_max
static void state_finish_com()

0.7.4.20 state_format_tuning

Synopsis: Format the tuning and TOD fields of the state display.
Type: static int
Arguments:

tune	TUNEP	The tuning to be formatted.
sc	STATE_COMP	The format information.

Referenced by: static int state_format()
References to: static STATE_COM state_state_com_max
static void state_finish_com()

0.7.4.21 state_is_distinct

Synopsis: Tests whether two states are distinct given the current state_filter.
Type: static char
Arguments:

st1	STATEP	The first state.
st2	STATEP	The second state.

Referenced by: static int state_format_statistics()
References to: static int state_filter

0.7.4.22 state_new_radio_buttons

Synopsis: Create new radio buttons.
Type: static void
Arguments: <none>
Referenced by: static void state_display()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
char *radio_name()
static WidgetRec *state_radio_attach_widget
static int state_n_radio_buttons

0.7.4.23 state_radio_com_compare

Synopsis: state_com_compare.
Compare two RADIO_COMPs for sorting.

Type: static int
Arguments: rc1 RADIO_COMP First RADIO_COM.
rc2 RADIO_COMP Second RADIO_COM.
Referenced by: static void state_display()
References to: <nothing>

0.7.4.24 state_radio_select

Synopsis: Change for which radios the state display is generated. Note that state_display test the selected status of the radio selection buttons directly, so this function only calls state_display to regenerate the display according to the new set of selected radios.
Type: static void
Arguments: w Widget The button widget.
a caddr_t Not used.
cback XmButtonLCallbackStruct * Standard callback information.
Referenced by: static struct /* unnamed */ register_list[9]
References to: static void state_display()

0.7.4.25 state_status_select

Synopsis: Activate callback for the status selection buttons. Compute the new state_status_filter from the selected buttons.
Type: static void
Arguments: w Widget The button widget.
a caddr_t Not used.
cback XmButtonLCallbackStruct * Standard callback information.
Referenced by: static struct /* unnamed */ register_list[9]
References to: static int state_status_filter
static STATE_FILTER state_status_buttons[6]
static void state_display()

0.8 status.c

0.8.1 Global Variables

0.8.1.1 status_function_attach_widget

Synopsis: The status function button list widget.
Type: WidgetRec *
Referenced by: FB_DATA status_function_closure
static void status_close_callback()
void statusInitWidgets()
static void create_status_select_button()
void status_destroy()

0.8.1.2 status_function_closure

Synopsis: Status function description.
Type: FB_DATA
Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: WidgetRec *status_function_widget
WidgetRec *status_function_attach_widget

0.8.1.3 status_function_widget

Synopsis: The status function frame widget.
Type: WidgetRec *

Referenced by: FB_DATA status_function_closure
void statusInitWidgets()

0.8.1.4 terrainMap

Synopsis: Hardwired UTM mapping information for the Ft. Knox terrain database.
Type: TerrainEle

0.8.2 Local Variables

0.8.2.1 buttonI_h_rcsId

Synopsis: RCS id of buttonI.h.
Type: static char *

0.8.2.2 network_h_rcsId

Synopsis: RCS id of network.h.
Type: static char

0.8.2.3 rcsId

Synopsis: RCS id of this file.
Type: static char

0.8.2.4 status_data

Synopsis: Resource data for the status display.
Type: static STATUS_DATA
Referenced by: static void set_status_label()
static void create_status_widget()
static void unhilite_widget()

0.8.2.5 status_first_widget

Synopsis: True the first time a status widget is created. Resource information is fetched the first time only.
Type: static char
Referenced by: void statusInitWidgets()
static void create_status_widget()

0.8.2.6 status_labels

Synopsis: A pool of available label widgets for building a status display.
Type: static char **
Referenced by: static void create_status_select_button()
void status_destroy()

0.8.2.7 status_max_labels

Synopsis: The total number of status_labels.
Type: static int
Referenced by: static void create_status_select_button()

0.8.2.8 status_n_labels

Synopsis: The current number of status_labels.
Type: static int
Referenced by: void status_select()
static void create_status_select_button()
void UpdateRadioStatus()
void status_destroy()

0.8.2.9 status_resources

Synopsis: The resource descriptions for status displays.
Type: static XtResource
Referenced by: static void create_status_widget()

0.8.2.10 status_rps

Synopsis: Array of radios currently having active status displays.
Type: static struct RadioInfoStruct **
Referenced by: void status_select()
static void create_status_select_button()

0.8.3 Global Functions

0.8.3.1 UpdateRadioStatus

Synopsis: Interface to status_new_radio and status_update to create all new radio selection buttons and update all radio status displays.
Type: void
Arguments: <none>
Referenced by: static void slow_periodic()
static void traffic_cursor_valid()
static void traffic_cursor_invalid()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
void status_new_radio()
void status_update()
static int status_n_labels

0.8.3.2 set_widget_label

Synopsis: Essentially SPRINTF for label widgets.
Type: void
Arguments: <none>
Referenced by: static void set_identification_label()
static void map_create_vehicle_widget()
static void traffic_create_widget()
static void create_status_widget()
void status_update()
References to: <nothing>

0.8.3.3 statusInitWidgets

Synopsis: Get widget IDs for status widgets.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: WidgetRec *NameToWidget()
static char status_first_widget
WidgetRec *status_function_widget
WidgetRec *status_function_attach_widget
WidgetRec *main_widget

0.8.3.4 statusRegisterNames

Synopsis: Register names for Mrm.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: <nothing>

0.8.3.5 status_destroy

Synopsis: Deletes all status buttons and status widgets.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: static char **status_labels
static int status_n_labels
WidgetRec *status_function_attach_widget

0.8.3.6 status_new_radio

Synopsis: Called when a new radio appears in order to create its selection button.
Type: void
Arguments:

rp	**RadioInfoP**	The new radio.

Referenced by: void UpdateRadioStatus()
References to: static void create_status_widget()
static void create_status_select_button()

0.8.3.7 status_select

Synopsis: Called when the set of selected radios changes. Changes the set of managed status boxes to reflect the selected radios.
Type: void
Arguments:

w	**Widget**	The button widget.
a	**caddr_t**	Not used.
cback	**XmButtonLCallbackStruct ***	Standard callback information.

Referenced by: static struct /* unnamed */ register_list[1]
References to: static struct RadioInfoStruct **status_rps
static int status_n_labels

0.8.3.8 status_update

Synopsis: Updates the contents of a status box according to the new state of a radio. If the traffic widget has a valid cursor, the time of that cursor is used to find the state to be displayed. In this case, the time is displayed as "Now" to avoid extra repainting. If the traffic cursor is not valid, the status display is update to reflect the current state of the radio. Only changed fields are drawn.
Type: void
Arguments:

rp	**RadioInfoP**	The radio to update.

Referenced by: void UpdateRadioStatus()
References to: char *strcat()
char *state_radio_name()
void set_widget_label()
unsigned char traffic_cursor_time()
char *traffic_format_traffic_time()
char tlFind()
static void set_status_label()
static int getUpdateFlags()
static char *UTMString()
static void unhilite_widget()

0.8.4 Local Functions

0.8.4.1 UTMString

Synopsis: Get the coordinate string for the given position.
Note: returns static storage.
Type: static char *
Arguments:

x	**double**	Map position (meters).

	y	**double**	Map position.

Referenced by: void status_update()
References to: <nothing>

0.8.4.2 create_status_select_button

Synopsis: Adds a button to select the status of a new radio.
Type: static void

Arguments:	**rp**	**RadioInfoP**	The radio.

Referenced by: void status_new_radio()
References to: char *radio_name()
static char **status_labels
static struct RadioInfoStruct **status_rps
static int status_max_labels
static int status_n_labels
WidgetRec *status_function_attach_widget

0.8.4.3 create_status_widget

Synopsis: Create a new status widget.
Type: static void

Arguments:	**rp**	**RadioInfoP**	The radio.

Referenced by: void status_new_radio()
References to: WidgetRec *NameToWidget()
char *radio_name()
void set_widget_label()
static char status_first_widget
static STATUS_DATA status_data
static XtResource status_resources[2]
struct /* unnamed */ *hierarchy_id
WidgetRec *general_area_widget

0.8.4.4 getUpdateFlags

Synopsis: Computes a mask of status fields which differ between two states.
Type: static int

Arguments:	**old_state**	**register STATEP**	The old state.
	new_state	**register STATEP**	The new state.

Referenced by: void status_update()
References to: <nothing>

0.8.4.5 set_status_label

Synopsis: Same as set_widget_label except the foreground color is changed to the hilited color.
Type: static void
Arguments: <none>
Referenced by: void status_update()
References to: static STATUS_DATA status_data

0.8.4.6 status_close_callback

Synopsis: Activate callback for close button in status widget. Remove this status box from the screen.
Type: static void

Arguments:	**w**	**Widget**	The button widget.
	rp	**RadioInfoP**	The radio.
	b	**caddr_t**	Standard callback information.

Referenced by: static XtCallbackRec callbacks[2]
References to: WidgetRec *status_function_attach_widget

0.8.4.7 unhilite_widget

Synopsis: Restores the foreground color of a widget to the unhilited color.
Type: static void
Arguments: w Widget The widget to unhilite.
Referenced by: void status_update()
References to: static STATUS_DATA status_data

0.9 timeline.c

0.9.1 Global Variables

0.9.1.1 tlFreeList

Synopsis: Free list of timeline segments.
Type: TLQ
Referenced by: void tlReplenishFreeList()
void tlInit()
void tlChangeState()
char *tlReadFile()
void tlFree()

0.9.2 Local Variables

0.9.2.1 network_h_rcsId

Synopsis: RCS id of network.h.
Type: static char

0.9.2.2 rcsId

Synopsis: RCS id of this file.
Type: static char

0.9.3 Global Functions

0.9.3.1 tlChangeState

Synopsis: Add a new state transition to a radio. If the current segment is full, another is allocated.
Type: void
Arguments: rp RadioInfoP The radio to change the state of.
timestamp MSEC When the change occurred.
Referenced by: void netChangeState()
References to: TLQ tlFreeList
int tlEnqueue()
static TLS *tlDequeue()

0.9.3.2 tlCountFile

Synopsis: Count how many lines are required to write a timeline.
Type: int
Arguments: rp RadioInfoP The radio to count.
Referenced by: int net_count_file()
References to: <nothing>

0.9.3.3 tlEnqueue

Synopsis: Add a segment to the indicated queue.
Type: int
Arguments: tlq register TLQP The queue on which to put the segment.

	tls	**register TLSP**	The segment to add.
Referenced by:	void tlReplenishFreeList()		
	void tlChangeState()		
	char *tlReadFile()		
	void tlFree()		
References to:	<nothing>		

0.9.3.4 tlFind

Synopsis:	Find the element which embraces the given time and return its segment and element pointers. Returns False if there is no such element.		
Type:	char		
Arguments:	**rp**	**RadioInfoP**	The radio to search.
	when	**long**	What time to look for.
	tlsp	**TLSP ***	Return the segment here.
	tlep	**TLEP ***	Return the element here.
Referenced by:	static int map_radio_transmit_time()		
	static void map_set_radio_time()		
	static int traffic_draw_slot()		
	void status_update()		
References to:	<nothing>		

0.9.3.5 tlFree

Synopsis:	Release timeline information for a radio.		
Type:	void		
Arguments:	**rp**	**RadioInfoP**	The radio to release the timeline of.
Referenced by:	void DestroyVehicles()		
	void resetRadios()		
References to:	TLQ tlFreeList		
	int tlEnqueue()		
	static TLS *tlDequeue()		

0.9.3.6 tlInit

Synopsis:	Initialize timeline things.		
Type:	void		
Arguments:	<none>		
Referenced by:	<nothing>		
References to:	void tlReplenishFreeList()		
	TLQ tlFreeList		

0.9.3.7 tlNext

Synopsis:	Advance the given segment and element pointers to the next element. Returns false if there are no more.		
Type:	char		
Arguments:	**tlsp**	**register TLSP ***	Return the segment here.
	tlep	**register TLEP ***	Return the element here.
Referenced by:	<nothing>		
References to:	<nothing>		

0.9.3.8 tlPrev

Synopsis:	Step the given segment and element pointers to the previous element. Returns false if there are no more.		
Type:	char		
Arguments:	**tlsp**	**register TLSP ***	Return the segment here.
	tlep	**register TLEP ***	Return the element here.
Referenced by:	static int map_radio_transmit_time()		

References to: <nothing>

0.9.3.9 tlReadFile

Synopsis: Read a timeline portion of a radmon file.
Type: char *
Arguments:

f	FILE *	The file to read.
rp	RadioInfoP	The radio to read about.

Referenced by: char *net_read_file()
References to: STATE *state_nth()
void tlReplenishFreeList()
char *file_gets()
TLQ tlFreeList
int tlEnqueue()
static TLS *tlDequeue()

0.9.3.10 tlReplenishFreeList

Synopsis: Replenish the free list. Interrupt level routines can't allocate memory so we keep a list of available segments to be used at interrupt level.
Type: void
Arguments: <none>
Referenced by: int main()
static void slow_periodic()
void tlInit()
char *tlReadFile()
References to: void DisablePeriodic()
TLQ tlFreeList
int tlEnqueue()

0.9.3.11 tlWriteFile

Synopsis: Write a timeline portion of a radmon file.
Type: void
Arguments:

f	FILE *	The file to write.
rp	RadioInfoP	The radio to write about.

Referenced by: void net_write_file()
References to: void file_count()

0.9.4 Local Functions

0.9.4.1 tlDequeue

Synopsis: Remove a segment from the indicated queue.
Type: static TLS *
Arguments:

tlq	register TLQP	The queue from which to get the segment.

Referenced by: void tlChangeState()
char *tlReadFile()
void tlFree()
References to: <nothing>

0.10 traffic.c

0.10.1 Global Variables

0.10.1.1 traffic

Synopsis: The (only) traffic display information.
Type: TRAFFIC_INFO

Referenced by: TRAFFIC_INFO traffic
static struct /* unnamed */ register_list[3]
void traffic_set_cursor_time()
unsigned char traffic_cursor_time()
char *traffic_format_traffic_time()
void traffic_reset()
void traffic_new_tune()
void traffic_new_radio()
void traffic_update()
void traffic_destroy()
References to: TRAFFIC_INFO traffic

0.10.1.2 traffic_radio_function_attach_widget

Synopsis: The traffic radio button list widget.
Type: WidgetRec *
Referenced by: static FB_DATA radio_function_closure
void trafficInitWidgets()
static void traffic_create_widget()

0.10.1.3 traffic_radio_function_widget

Synopsis: The traffic radio frame widget.
Type: WidgetRec *
Referenced by: static FB_DATA radio_function_closure
void trafficInitWidgets()
static void traffic_create_widget()

0.10.1.4 traffic_sub_function_attach_widget

Synopsis: The traffic sub–functions button list widget.
Type: WidgetRec *
Referenced by: FB_DATA traffic_sub_function_closure
void trafficInitWidgets()

0.10.1.5 traffic_sub_function_closure

Synopsis: The traffic sub–functions description.
Type: FB_DATA
Referenced by: static FUNCTION_BUTTON function_buttons[9]
References to: static FUNCTION_BUTTON_CLOSURE traffic_button_closure
WidgetRec *traffic_sub_function_attach_widget
WidgetRec *traffic_sub_function_widget

0.10.1.6 traffic_sub_function_widget

Synopsis: The traffic sub–functions frame widget.
Type: WidgetRec *
Referenced by: FB_DATA traffic_sub_function_closure
void trafficInitWidgets()

0.10.1.7 traffic_tuning_function_attach_widget

Synopsis: The traffic tuning button list widget.
Type: WidgetRec *
Referenced by: static FB_DATA tuning_function_closure
void trafficInitWidgets()
static void traffic_create_widget()

0.10.1.8 traffic_tuning_function_widget

Synopsis: The traffic tuning frame widget.

Type: WidgetRec *
Referenced by: static FB_DATA tuning_function_closure
void trafficInitWidgets()
static void traffic_create_widget()

0.10.2 Local Variables

0.10.2.1 buttonI_h_rcsId

Synopsis: RCS id of buttonI.h.
Type: static char *

0.10.2.2 network_h_rcsId

Synopsis: RCS id of network.h.
Type: static char

0.10.2.3 radio_function_closure

Synopsis: The traffic radio buttons description.
Type: static FB_DATA
Referenced by: static FUNCTION_BUTTON traffic_buttons[2]
References to: WidgetRec *traffic_radio_function_widget
WidgetRec *traffic_radio_function_attach_widget

0.10.2.4 rcsId

Synopsis: RCS id of this file.
Type: static char

0.10.2.5 traffic_button_closure

Synopsis: The traffic buttons description.
Type: static FUNCTION_BUTTON_CLOSURE
Referenced by: FB_DATA traffic_sub_function_closure
References to: static FUNCTION_BUTTON traffic_buttons[2]

0.10.2.6 traffic_buttons

Synopsis: The list of traffic sub-function buttons.
Type: static FUNCTION_BUTTON
Referenced by: static FUNCTION_BUTTON_CLOSURE traffic_button_closure
References to: static FB_DATA radio_function_closure
static FB_DATA tuning_function_closure

0.10.2.7 traffic_data

Synopsis: The resource data for traffic displays.
Type: static TRAFFIC_DATA
Referenced by: static void traffic_erase_labels()
static void traffic_draw_label()
static int traffic_draw_slot()
static char traffic_measure_grid()
static void traffic_draw_grid()
static void traffic_redraw_labels()
static void traffic_redraw()
static struct /* unnamed */ *colorp[8]
static void traffic_allocate_colors()
static void traffic_create_widget()
static void traffic_new_height()

0.10.2.8 traffic_n_radios

Synopsis: The number of traffic radio buttons.
Type: static int
Referenced by: void trafficInitWidgets()
void UpdateRadioTraffic()
void traffic_destroy()

0.10.2.9 traffic_resources

Synopsis: The traffic resources description.
Type: static XtResource
Referenced by: static void traffic_create_widget()
References to: void BackgroundColorDefault()
void ForegroundColorDefault()

0.10.2.10 tuning_function_closure

Synopsis: The traffic tuning buttons description.
Type: static FB_DATA
Referenced by: static FUNCTION_BUTTON traffic_buttons[2]
References to: WidgetRec *traffic_tuning_function_widget
WidgetRec *traffic_tuning_function_attach_widget

0.10.3 Global Functions

0.10.3.1 UpdateRadioTraffic

Synopsis: Add new radios if necessary and keep the traffic display up to date.
Type: void
Arguments: <none>
Referenced by: static void slow_periodic()
References to: struct RadioInfoStruct **net_radios
int net_n_radios
unsigned char traffic_need_update
void traffic_new_radio()
void traffic_update()
static int traffic_n_radios

0.10.3.2 trafficInitWidgets

Synopsis: Get widget ids for the various button lists.
Type: void
Arguments: <none>
Referenced by: void display_init()
References to: WidgetRec *NameToWidget()
WidgetRec *main_widget
WidgetRec *sub_function_widget
WidgetRec *traffic_sub_function_widget
WidgetRec *traffic_sub_function_attach_widget
WidgetRec *traffic_radio_function_widget
WidgetRec *traffic_radio_function_attach_widget
WidgetRec *traffic_tuning_function_widget
WidgetRec *traffic_tuning_function_attach_widget
static int traffic_n_radios

0.10.3.3 trafficRegisterNames

Synopsis: Register names to be used by mrm.
Type: void
Arguments: <none>

Referenced by: void display_init()
References to: <nothing>

0.10.3.4 traffic_cursor_time

Synopsis: Return the time of the current traffic cursor. If the cursor is not valid, the present time (now) is returned and the function value is FALSE.
Type: unsigned char
Arguments: **p** **MSEC *** Return the time here.
Referenced by: static void map_freeze()
void status_update()
References to: int now
TRAFFIC_INFO traffic

0.10.3.5 traffic_destroy

Synopsis: Destroy resources allocated for traffic display.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: TRAFFIC_INFO traffic
static int traffic_n_radios
static void traffic_delete_pos()

0.10.3.6 traffic_format_traffic_time

Synopsis: Format a time value for the main traffic widget.
Type: char *
Arguments: **time** **MSEC** The time to convert.
Referenced by: void status_update()
References to: TRAFFIC_INFO traffic
static char *traffic_format_time()

0.10.3.7 traffic_new_radio

Synopsis: Add a new radio selection button.
Type: void
Arguments: **rp** **RadioInfoP** The new radio.
Referenced by: void UpdateRadioTraffic()
References to: TRAFFIC_INFO traffic
static void traffic_create_widget()
static int traffic_create_radio_button()

0.10.3.8 traffic_new_tune

Synopsis: Add a new tuning button.
Type: void
Arguments: **tune** **TUNEP** The new tuning.
Referenced by: static void tune_new()
References to: TRAFFIC_INFO traffic
static void traffic_create_widget()
static int traffic_create_tune_button()

0.10.3.9 traffic_reset

Synopsis: Reset the traffic display back to its ground state.
Type: void
Arguments: <none>
Referenced by: static void reset_callback()
void Offline()
References to: TRAFFIC_INFO traffic

static char traffic_measure_grid()
static void traffic_unfreeze()
static void traffic_cursor_invalid()
static void traffic_set_scrollbar()
static void traffic_erase_drawing()

0.10.3.10 traffic_set_cursor_time

Synopsis: Move the traffic cursor to a specific time. The traffic display is frozen if it was not previously. The displayed is scrolled as necessary to bring the selected time into view.
Type: void
Arguments:

time	MSEC	The time to set.

Referenced by: static void map_time_callback()
static void map_step()
static void map_step_to_anomaly()
static void map_freeze()
References to: TRAFFIC_INFO traffic
static char traffic_left()
static char traffic_right()
static void traffic_draw_cursor()
static void traffic_erase_cursor()
static void traffic_freeze()
static int TIME_TO_WINDOW_COORD()
static void traffic_cursor_valid()
static void traffic_set_scrollbar()

0.10.3.11 traffic_update

Synopsis: Update the traffic display. Create the traffic widget, if necessary. Scroll if necessary and permitted to keep the present time on the screen.
Type: void
Arguments: <none>
Referenced by: void UpdateRadioTraffic()
References to: TRAFFIC_INFO traffic
static char traffic_left()
static void traffic_set_scrollbar()
static void traffic_create_widget()

0.10.4 Local Functions

0.10.4.1 TIME_TO_WINDOW_COORD

Synopsis: Function to convert from time to position in the traffic window.
Type: static int
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
t	MSEC	Time to convert from.

Referenced by: void traffic_set_cursor_time()
static int traffic_draw_slot()
static void traffic_draw_grid()
static void traffic_zoom_out()
static char traffic_left()
References to: <nothing>

0.10.4.2 TIME_TO_WINDOW_DELTA

Synopsis: Function to convert from a time dimension to a window dimension.
Type: static int
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
t	MSEC	Time delta to convert from.

Referenced by: static char traffic_measure_grid()
static void traffic_zoom_in()
static void traffic_zoom_out()
static void traffic_scrollbar()
References to: <nothing>

0.10.4.3 WINDOW_TO_TIME_COORD

Synopsis: Function to convert from window position to time.
Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	x	Int	Position to convert from.

Referenced by: static void traffic_set_scrollbar()
static int traffic_draw_slot()
static char traffic_measure_grid()
static void traffic_draw_grid()
static void traffic_zoom_in()
static void traffic_zoom_out()
static void traffic_input()
static void traffic_scrollbar()
References to: <nothing>

0.10.4.4 WINDOW_TO_TIME_DELTA

Synopsis: Function to convert from a window dimension to a time dimension.
Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	x	Int	Position delta to convert from.

Referenced by: static void traffic_set_scrollbar()
References to: <nothing>

0.10.4.5 traffic_add_pos

Synopsis: Add a new slot to the traffic display for the radio button in the indicated position.
Type: static char

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	position	Int	The button position.

Referenced by: static void traffic_sub_select()
References to: char *radio_name()
static void traffic_new_height()

0.10.4.6 traffic_allocate_colors

Synopsis: Allocates colors for the various states represented in the traffic display. Allocates one plane for the cursor so it can be drawn independently of the traffic lines.
Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.

Referenced by: static void traffic_create_widget()
References to: static TRAFFIC_DATA traffic_data

0.10.4.7 traffic_close

Synopsis: Activate callback for the "close" button. Remove the traffic widget from the screen.
Type: static void

Arguments:			
	w	Widget	The close button widget.
	trf	TRAFFIC_INFOP	Traffic information.
	cback	caddr_t	Standard callback information.

Referenced by: static XtCallbackRec close_callbacks[2]
References to: <nothing>

0.10.4.8 traffic_continue_pan_left

Synopsis: Timer routine to perform another step of panning to the left. Restarts another timer if scrolling occurs

Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	id	XtIntervalId	Not used.

Referenced by: static void traffic_continue_pan_left()
static void traffic_pan_left()

References to: static char traffic_left()
static void traffic_set_scrollbar()
static void traffic_continue_pan_left()

0.10.4.9 traffic_continue_pan_right

Synopsis: Timer routine to perform another step of panning to the right. Restarts another timer if scrolling occurs.

Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	id	XtIntervalId	Not used.

Referenced by: static void traffic_continue_pan_right()
static void traffic_pan_right()

References to: static char traffic_right()
static void traffic_set_scrollbar()
static void traffic_continue_pan_right()

0.10.4.10 traffic_create_radio_button

Synopsis: Add a new radio selection button to the button list. Called when a new radio appears.

Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	rp	RadioInfoP	The radio.

Referenced by: void traffic_new_radio()

References to: char *radio_name()

0.10.4.11 traffic_create_tune_button

Synopsis: Add a new tuning button to the button list. Called when a new radio tuning appears.

Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	tune	TUNEP	The tuning.

Referenced by: void traffic_new_tune()

References to: <nothing>

0.10.4.12 traffic_create_widget

Synopsis: Create a traffic widget and fill in its associated data structure.

Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.

Referenced by: void traffic_new_tune()
void traffic_new_radio()
void traffic_update()

References to: WidgetRec *NameToWidget()
void set_widget_label()
WidgetRec *traffic_radio_function_widget
WidgetRec *traffic_radio_function_attach_widget
WidgetRec *traffic_tuning_function_widget
WidgetRec *traffic_tuning_function_attach_widget
struct /* unnamed */ *hierarchy_id
WidgetRec *general_area_widget

static char traffic_measure_grid()
static TRAFFIC_DATA traffic_data
static XtResource traffic_resources[18]
static void traffic_cursor_invalid()
static void traffic_graphics_expose()
static void traffic_motion()
static void traffic_allocate_colors()

0.10.4.13 traffic_cursor_invalid

Synopsis: Make the current traffic cursor invalid. The status widgets are updated to reflect the present time.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: static void traffic_zoom_in()
static void traffic_zoom_out()
static void traffic_unfreeze()
static void traffic_create_widget()
void traffic_reset()
References to: void UpdateRadioStatus()

0.10.4.14 traffic_cursor_valid

Synopsis: Make the current traffic cursor valid. The status widgets are updated to reflect the new time.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: void traffic_set_cursor_time()
static void traffic_input()
References to: void UpdateRadioStatus()

0.10.4.15 traffic_delete_pos

Synopsis: Delete the slot for the radio button at the indicated position.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
position	int	The button position.

Referenced by: static void traffic_sub_select')
void traffic_destroy()
References to: static void traffic_new_height()

0.10.4.16 traffic_draw_cursor

Synopsis: Draw the cursor and mark.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: void traffic_set_cursor_time()
static char traffic_left()
static char traffic_right()
static void traffic_input()
static void traffic_motion()
References to: <nothing>

0.10.4.17 traffic_draw_grid

Synopsis: Draw and label the traffic label.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
left	int	Left edge of the area to draw.
width	int	Width of the area to draw.

Referenced by: static void traffic_draw_slots()

References to: int now
static TRAFFIC_DATA traffic_data
static int TIME_TO_WINDOW_COORD()
static int WINDOW_TO_TIME_COORD()
static char *traffic_format_time()

0.10.4.18 traffic_draw_label

Synopsis: Draw the label for the given slot.
Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	slot	int	The slot to draw.

Referenced by: static void traffic_draw_labels()
References to: static TRAFFIC_DATA traffic_data

0.10.4.19 traffic_draw_labels

Synopsis: Draw the labels for all slots.
Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	first_slot	int	Draw labels from here.
	last_slot	int	to here.

Referenced by: static void traffic_redraw_labels()
References to: static void traffic_draw_label()

0.10.4.20 traffic_draw_slot

Synopsis: Draw the data for the given slot starting a position "left" on the screen and continuing for "width" pixels.

For each state change, a GC is selected according to the previous state and a line drawn using that GC from the previous position to the position of the state change. Transmissions are emphasized by drawing the lines for transmission one pixel further than for other states.

Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	slot	int	The slot to draw.
	left	int	The left side of the area to redraw.
	width	int	The width of the area to redraw.

Referenced by: static void traffic_draw_slots()
References to: int now
char tlFind()
static TRAFFIC_DATA traffic_data
static int TIME_TO_WINDOW_COORD()
static int WINDOW_TO_TIME_COORD()
static char tune_selected()

0.10.4.21 traffic_draw_slots

Synopsis: Draw slots in the indicated region.
Type: static void

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	first_slot	int	First slot to draw.
	last_slot	int	Last slot to draw.
	left	int	Left edge of area to draw.
	width	int	Width of area to draw.

Referenced by: static void traffic_redraw()
static char traffic_left()
static char traffic_right()
References to: static void traffic_draw_grid()
static int traffic_draw_slot()

0.10.4.22 traffic_erase_cursor

Synopsis: Erase the cursor and mark.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: void traffic_set_cursor_time()
static char traffic_left()
static char traffic_right()
static void traffic_unfreeze()
static void traffic_input()
static void traffic_motion()
References to: <nothing>

0.10.4.23 traffic_erase_drawing

Synopsis: Erase (and then redraw) the traffic drawing.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: static void traffic_zoom_in()
static void traffic_zoom_out()
static char traffic_left()
static char traffic_right()
static void traffic_fit()
static void traffic_sub_select()
void traffic_reset()
References to: static void traffic_set_scrollbar()

0.10.4.24 traffic_erase_labels

Synopsis: Erase (and then redraw) the labels for the given slots.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
first_slot	int	Erase from here.
last_slot	int	to here.

Referenced by: static void traffic_sub_select()
References to: static TRAFFIC_DATA traffic_data
static void traffic_set_scrollbar()

0.10.4.25 traffic_expose

Synopsis: Expose callback for the traffic display. Redraws the exposed slots.
Type: static void
Arguments:

w	Widget	The exposed drawing area widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	XmDrawingAreaCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec expose_callbacks[2]
References to: static void traffic_redraw()

0.10.4.26 traffic_fit

Synopsis: Callback for fit button. Adjust the scale and offset to fit the entire data in the window.
Type: static void
Arguments:

w	Widget	The fit button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	caddr_t	Standard callback information.

Referenced by: static XtCallbackRec fit_callbacks[2]
References to: int now
static char traffic_measure_grid()
static void traffic_unfreeze()

static void traffic_set_scrollbar()
static void traffic_erase_drawing()

0.10.4.27 traffic_format_time

Synopsis: Constructs a label for the fiven time. tr_grid_first_format and tr_grid_last_format give the range of fields required for the current scaling. These values essentially discard high-order fields which are always zero in the current scaling and discard low-order fields which are not necessary to resolve the grid markings. Thus if the display must represent non-zero hours and the grid markings are multiples of 5 seconds, then the tr_grid_first_format will by DATE_HOURS and tr_grid_last_format will be DATE_SECONDS.

The first_formats array gives the format of the first field. Generally, this discards leading zeros. The middle_formats array gives the format of intermediate fields. Generally, this retains all leading zeros in the field. The final_formats array gives the format of the last field. Generally, this includes suffixes which disambiguate the result (e.g. hh:mm is augmented to hh:mm:00 to distinguish if from mm:ss). The only_formats array combines the first_formats and last_formats.

Type: static char *

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	time	long	The time to convert.

Referenced by: char *traffic_format_traffic_time()
static char traffic_measure_grid()
static void traffic_draw_grid()

References to: char *strcat()

0.10.4.28 traffic_freeze

Synopsis: Callback for freeze button. Prevents the traffic display from scrolling. Sensitizes the unfreeze button and desensitizes the freeze button.

Type: static void

Arguments:			
	w	Widget	The freeze button widget.
	trf	TRAFFIC_INFOP	Traffic information.
	cback	caddr_t	Standard callback information.

Referenced by: void traffic_set_cursor_time()
static void traffic_zoom_in()
static void traffic_zoom_out()
static char traffic_right()
static void traffic_input()
static XtCallbackRec freeze_callbacks[2]

References to: <nothing>

0.10.4.29 traffic_graphics_expose

Synopsis: Graphics expose callback for the traffic display. Redraws the exposed slots.

Type: static void

Arguments:			
	w	Widget	The exposed widget.
	trf	TRAFFIC_INFOP	Traffic information.
	event	XEvent *	The X event information.

Referenced by: static void traffic_create_widget()

References to: static void traffic_redraw()

0.10.4.30 traffic_input

Synopsis: Input callback for the traffic drawing widget. Sets the cursor and mark to where the left button is pressed and released respectively.

Type: static void

Arguments:			
	w	Widget	The drawing widget.
	trf	TRAFFIC_INFOP	Traffic information.
	cback	XmDrawingAreaCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec input_callbacks[2]
References to: int now
void map_advance_display()
static void traffic_draw_cursor()
static void traffic_erase_cursor()
static void traffic_freeze()
static int WINDOW_TO_TIME_COORD()
static void traffic_cursor_valid()

0.10.4.31 traffic_left

Synopsis: Scroll the traffic display left by the indicated amount. The scroll distance is limited so that the right data extreme is not less than "limit". The cursor is move so it stays with the data. The grid is remeasured. The function returns False if no scrolling could be done.
Type: static char
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
delta	int	The amount to move.
limit	int	The maximum position.

Referenced by: void traffic_set_cursor_time()
static void traffic_continue_pan_left()
static int traffic_set_offset()
void traffic_update()
References to: int now
static char traffic_measure_grid()
static void traffic_draw_cursor()
static void traffic_erase_cursor()
static int TIME_TO_WINDOW_COORD()
static void traffic_erase_drawing()
static void traffic_draw_slots()

0.10.4.32 traffic_measure_grid

Synopsis: Figure out how to layout and label the grid on the traffic display. The grid is constrained to increment by the listed amounts and the grid interval is selected to place no more than 8 ticks on the display. If the number of ticks chosen would result in the labels overlapping, the number of ticks is reduced.
Type: static char
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: static void traffic_resize()
static void traffic_zoom_in()
static void traffic_zoom_out()
static char traffic_left()
static char traffic_right()
static void traffic_fit()
static void traffic_create_widget()
void traffic_reset()
References to: static TRAFFIC_DATA traffic_data
static int WINDOW_TO_TIME_COORD()
static int TIME_TO_WINDOW_DELTA()
static char *traffic_format_time()
static enum /* unnamed */ traffic_time_to_date_field()

0.10.4.33 traffic_motion

Synopsis: Updates the mark position as the mouse is dragged across the traffic display.
Type: static void
Arguments:

w	Widget	The drawing widget.
trf	TRAFFIC_INFOP	Traffic information.
event	XEvent *	X event (mouse event).

Referenced by: static void traffic_create_widget()
References to: static void traffic_draw_cursor()
static void traffic_erase_cursor()

0.10.4.34 traffic_new_height

Synopsis: Compute a new height for the traffic widget and change it necessary.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: static char traffic_add_pos()
static void traffic_delete_pos()
References to: static TRAFFIC_DATA traffic_data

0.10.4.35 traffic_pan_left

Synopsis: Initiate or terminate panning left. Arming the pan left button starts panning. Disarming terminates panning.
Type: static void
Arguments:

w	Widget	The pan left button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	XmAnyCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec pan_left_callbacks[2]
References to: static void traffic_continue_pan_left()

0.10.4.36 traffic_pan_right

Synopsis: Initiate or terminate panning right. Arming the pan right button starts panning. Disarming terminates panning.
Type: static void
Arguments:

w	Widget	The pan right button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	XmAnyCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec pan_right_callbacks[2]
References to: static void traffic_continue_pan_right()

0.10.4.37 traffic_redraw

Synopsis: Redraw traffic slots.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.
left	int	Left edge of area to redraw.
top	int	Top edge of area to redraw.
width	int	Width of area to redraw.
height	int	Height of area to redraw.

Referenced by: static void traffic_expose()
static void traffic_graphics_expose()
References to: static TRAFFIC_DATA traffic_data
static void traffic_draw_slots()

0.10.4.38 traffic_redraw_labels

Synopsis: Expose callback for the label area. Redraws the labels included in the exposed area.
Type: static void
Arguments:

w	Widget	Drawing area widget to paint.
trf	TRAFFIC_INFOP	Traffic information.
cback	XmDrawingAreaCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec expose_labels_callbacks[2]

References to: static TRAFFIC_DATA traffic_data
static void traffic_draw_labels()

0.10.4.39 traffic_resize

Synopsis: Resize callback for traffic drawing widget. Records the new dimensions, sets the scrollbar parameters, and recomputes the grid parameters. The concomitant redrawing is initiated by the expose event.

Type: static void

Arguments:			
	w	Widget	The resized widget.
	trf	TRAFFIC_INFOP	Traffic information.
	cback	XmDrawingAreaCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec resize_callbacks[2]
References to: static char traffic_measure_grid()
static void traffic_set_scrollbar()

0.10.4.40 traffic_right

Synopsis: Scroll the traffic display right by the indicated amount. The scroll distance is limited so that the left data extreme is not greater than 10. The cursor is move so it stays with the data. The grid is remeasured. The function returns False if no scrolling could be done.

Type: static char

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	delta	int	The amount to move.

Referenced by: void traffic_set_cursor_time()
static void traffic_continue_pan_right()
static int traffic_set_offset()
References to: static char traffic_measure_grid()
static void traffic_draw_cursor()
static void traffic_erase_cursor()
static void traffic_freeze()
static void traffic_erase_drawing()
static void traffic_draw_slots()

0.10.4.41 traffic_scrollbar

Synopsis: Callback for the traffic scrollbar. Calls traffic_set_offset to set the new position as selected by the scrollbar.

Type: static void

Arguments:			
	w	Widget	The scrollbar widget.
	trf	TRAFFIC_INFOP	Traffic information.
	cback	XmScrollBarCallbackStruct *	Standard callback info.

Referenced by: static XtCallbackRec scrollbar_callbacks[2]
References to: int now
static int WINDOW_TO_TIME_COORD()
static int TIME_TO_WINDOW_DELTA()
static void traffic_set_scrollbar()
static int traffic_set_offset()

0.10.4.42 traffic_set_offset

Synopsis: Changes the drawing offset to a new value. Calls traffic_left or traffic_right as needed to accomplish this.

Type: static int

Arguments:			
	trf	TRAFFIC_INFOP	Traffic information.
	new_offset	int	The new traffic offset.

Referenced by: static void traffic_scrollbar()
References to: static char traffic_left()

static char traffic_right()

0.10.4.43 traffic_set_scrollbar

Synopsis: Update scrollbar variables to reflect a new position or size.
Type: static void
Arguments:

trf	TRAFFIC_INFOP	Traffic information.

Referenced by: void traffic_set_cursor_time()
static void traffic_erase_labels()
static void traffic_erase_drawing()
static void traffic_resize()
static void traffic_zoom_in()
static void traffic_zoom_out()
static void traffic_continue_pan_left()
static void traffic_continue_pan_right()
static void traffic_fit()
static void traffic_scrollbar()
void traffic_reset()
void traffic_update()
References to: int now
static int WINDOW_TO_TIME_COORD()
static int WINDOW_TO_TIME_DELTA()

0.10.4.44 traffic_sub_select

Synopsis: Callback for traffic selection buttons. Adds and deletes radios from the display as indicated by the selections.
Type: static void
Arguments:

w	Widget	widget.
bi	BUTTON_INFOP	Button information.
cback	XmListCallbackStruct *	Standard callback info.

Referenced by: static struct /* unnamed */ register_list[3]
References to: static void traffic_erase_labels()
static void traffic_erase_drawing()
static char traffic_add_pos()
static void traffic_delete_pos()

0.10.4.45 traffic_time_to_date_field

Synopsis: Returns the ceiling date type for the specified time.
Type: static enum /* unnamed */
Arguments:

time	long	The time being converted.

Referenced by: static char traffic_measure_grid()
References to: <nothing>

0.10.4.46 traffic_unfreeze

Synopsis: Callback for unfreeze button. Allows the traffic display to scrolling. Sensitizes the freeze button and desensitizes the unfreeze button.
Type: static void
Arguments:

w	Widget	The unfreeze button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	caddr_t	Standard callback information.

Referenced by: static void traffic_fit()
static XtCallbackRec unfreeze_callbacks[2]
void traffic_reset()
References to: unsigned char online_mode
static void traffic_erase_cursor()
static void traffic_cursor_invalid()

0.10.4.47 traffic_zoom_in

Synopsis: Activate callback for the zoomin button. Adjusts the scale and offset of the drawing so the area between the mark and cursor falls between the left and right extremes of the display. If the cursor is not valid, the scale is doubles keeping the left edge constant.
Type: static void
Arguments:

w	Widget	The zoomin button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	caddr_t	Standard callback information.

Referenced by: static XtCallbackRec zoom_in_callbacks[2]
References to: static char traffic_measure_grid()
static void traffic_freeze()
static int WINDOW_TO_TIME_COORD()
static int TIME_TO_WINDOW_DELTA()
static void traffic_cursor_invalid()
static void traffic_set_scrollbar()
static void traffic_erase_drawing()

0.10.4.48 traffic_zoom_out

Synopsis: Activate callback for the zoom out button. Adjusts the scale and offset of the drawing so the area between the left and right extremes of the display falls between the mark and cursor. If the cursor is not valid, the scale is halved keeping th > left edge constant.
Type: static void
Arguments:

w	Widget	The zoomout button widget.
trf	TRAFFIC_INFOP	Traffic information.
cback	caddr_t	Standard callback information.

Referenced by: static XtCallbackRec zoom_out_callbacks[2]
References to: static char traffic_measure_grid()
static void traffic_freeze()
static int TIME_TO_WINDOW_COORD()
static int WINDOW_TO_TIME_COORD()
static int TIME_TO_WINDOW_DELTA()
static void traffic_cursor_invalid()
static void traffic_set_scrollbar()
static void traffic_erase_drawing()

0.10.4.49 tune_selected

Synopsis: Test if the specified tuning is currently selected. The buttons given by "bi" are scanned looking for the one which matches the tuning and returning its selected state.
Type: static char
Arguments:

bi	BUTTON_INFOP	Information about the button being tested.
tune	TUNEP	The tuning to match.

Referenced by: static int traffic_draw_slot()
References to: <nothing>

0.11 tune.c

0.11.1 Global Variables

0.11.2 Local Variables

0.11.2.1 network_h_rcsid

Synopsis: RCS id of network.h.
Type: static char

0.11.2.2 rcsid

Synopsis: RCS id of this file.
Type: static char

0.11.2.3 tune_ht

Synopsis: A hash table containing all tunings ever encountered.
Type: static HTABLE *
Referenced by: void tune_init()
TUNE *tune_find()
void tune_destroy()

0.11.2.4 tune_list

Synopsis: The array of all tunings.
Type: static TUNE **
Referenced by: void tune_write_file()
void tuneReplenishFreeList()
static TUNE *tuneAllocate()
void tune_init()
static void tune_set_name()
static void tune_new()
TUNE *tune_nth()

0.11.2.5 tune_max_list

Synopsis: The total size of tune_list.
Type: static int
Referenced by: char *tune_read_file()
void tuneReplenishFreeList()
static TUNE *tuneAllocate()
void tune_init()

0.11.2.6 tune_n_list

Synopsis: The number of entries in tune_list.
Type: static int
Referenced by: int tune_count_file()
void tune_write_file()
char *tune_read_file()
void tuneReplenishFreeList()
static TUNE *tuneAllocate()
void tune_init()
static void tune_set_name()
static void tune_new()
void tune_destroy()

0.11.2.7 tune_n_pending

Synopsis: Count of unprocessed new tunings.
Type: static int
Referenced by: void tune_init()
static void tune_new()
void tune_destroy()

0.11.2.8 tune_new_pending

Synopsis: The ID of the timer used for defered update of tuning lists.
Type: static unsigned int
Referenced by: void tune_init()
TUNE *tune_find()

static void tune_new()

0.11.3 Global Functions

0.11.3.1 tuneReplenishFreeList

Synopsis: Replenish the list of free tunings. We preallocate tunings and keep in a list so interrupt level code doesn't have to allocate storage.
Type: void
Arguments: <none>
Referenced by: int main()
static void slow_periodic()
char *tune_read_file()
void tune_init()
References to: void DisablePeriodic()
static TUNE **tune_list
static int tune_n_list
static int tune_max_list

0.11.3.2 tune_compare

Synopsis: Compare two tunings. Used by the state display to sort states.
Type: int
Arguments:

t1	register TUNEP	The first tuning.
t2	register TUNEP	The second tuning.

Referenced by: static int state_derived_com_compare()
References to: <nothing>

0.11.3.3 tune_count_file

Synopsis: Count the number of output lines for the tuning section of a radmon file.
Type: int
Arguments:

f	FILE *	Not used.

Referenced by: static void file_write_file()
References to: static int tune_n_list

0.11.3.4 tune_destroy

Synopsis: Release tuning storage.
Type: void
Arguments: <none>
Referenced by: void DestroyEverything()
References to: HTABLE *htInit()
void htFree()
static HTABLE *tune_ht
static int tune_n_list
static int tune_n_pending

0.11.3.5 tune_find

Synopsis: Find a tuning for the given mode and frequency or hopinfo. If the tuning does not already exist, it is created.
Type: TUNE *
Arguments:

mode	unsigned char	The tuning mode (FH or SC).
frequency	long	The frequence if SC.
hopinfo	FHParameters *	The hop info if FH.

Referenced by: char *tune_read_file()
static void ProcessTransmitterPDU()
References to: HASHENTRY *htFind()
HASHENTRY *htInsert()

static HTABLE *tune_ht
static unsigned int tune_new_pending
static void tune_new()
static TUNE *tuneAllocate()
static void tune_set_name()
int bzero()

0.11.3.6 tune_init

Synopsis: Initialize tuning stuff, perform initial replenishment.
Type: void
Arguments: <none>
Referenced by: int main()
References to: HTABLE *htInit()
void tuneReplenishFreeList()
static HTABLE *tune_ht
static TUNE **tune_list
static int tune_n_list
static int tune_max_list
static int tune_n_pending
static unsigned int tune_new_pending

0.11.3.7 tune_nth

Synopsis: Return the nth tuning.
Type: TUNE *

Arguments:			
	n	int	The index to retrieve.

Referenced by: char *state_read_file()
References to: static TUNE **tune_list

0.11.3.8 tune_read_file

Synopsis: Read the tuning section of a radmon file.
Type: char *

Arguments:			
	f	FILE *	The file to read.
	version	int	File format version.

Referenced by: static void file_read_file()
References to: void htReplenish()
TUNE *tune_find()
void tuneReplenishFreeList()
char *file_gets()
static int tune_n_list
static int tune_max_list

0.11.3.9 tune_write_file

Synopsis: Write the tuning section of a radmon file.
Type: void

Arguments:			
	f	FILE *	Not used.

Referenced by: static void file_write_file()
References to: void file_count()
static TUNE **tune_list
static int tune_n_list

0.11.4 Local Functions

0.11.4.1 tuneAllocate

Synopsis: Get a tuning from the free list.
Type: static TUNE *

Arguments: <none>
Referenced by: TUNE *tune_find()
References to: static TUNE **tune_list
static int tune_n_list
static int tune_max_list

0.11.4.2 tune_new

Synopsis: Called when new tunings have been created to update the tune_list and create tuning buttons for the traffic display.
Type: static void
Arguments: <none>
Referenced by: TUNE *tune_find()
References to: void traffic_new_tune()
static TUNE **tune_list
static int tune_n_list
static int tune_n_pending
static unsigned int tune_new_pending

0.11.4.3 tune_set_name

Synopsis: Set the q_name of a tuning from the frequency or hopset id. We do this once when the tuning is created as the name is readily available.
Type: static void
Arguments: tune1 TUNEP The tuning to set.
Referenced by: TUNE *tune_find()
References to: static TUNE **tune_list
static int tune_n_list