

Generic Trace Generator (GTG)

0.1

Generated by Doxygen 1.7.6.1

Tue Apr 24 2012 10:29:22

Contents

1	The GTG library	1
1.1	Presentation	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	Trace type handler	9
5.1.1	Enumeration Type Documentation	9
5.1.1.1	traceType	9
5.1.2	Function Documentation	10
5.1.2.1	bufferedModeActivated	10
5.1.2.2	getName	10
5.1.2.3	getTraceType	10
5.1.2.4	setTraceType	10
5.2	To init the generated trace file(s)	11
5.2.1	Function Documentation	11
5.2.1.1	addEntityValue	11
5.2.1.2	endTrace	11
5.2.1.3	initTrace	11
5.2.1.4	setCompress	12

5.3	Functions related to the containers	13
5.3.1	Function Documentation	13
5.3.1.1	AddComment	13
5.3.1.2	addContainer	13
5.3.1.3	addContType	14
5.3.1.4	destroyContainer	14
5.4	Functions related to the states	15
5.4.1	Function Documentation	15
5.4.1.1	addStateType	15
5.4.1.2	popState	15
5.4.1.3	pushState	16
5.4.1.4	setState	16
5.5	Functions related to the events	17
5.5.1	Function Documentation	17
5.5.1.1	addEvent	17
5.5.1.2	addEventType	17
5.6	Functions related to links	18
5.6.1	Function Documentation	18
5.6.1.1	addLinkType	18
5.6.1.2	endLink	18
5.6.1.3	startLink	19
5.7	Functions related to variables	20
5.7.1	Function Documentation	20
5.7.1.1	addVar	20
5.7.1.2	addVarType	20
5.7.1.3	setVar	21
5.7.1.4	subVar	21
5.8	Defined colors for GTG	22
5.8.1	Function Documentation	23
5.8.1.1	GTG_COLOR_GET_BLUE	23
5.8.1.2	GTG_COLOR_GET_GREEN	23
5.8.1.3	GTG_COLOR_GET_RED	23
5.8.1.4	GTG_COLOR_SET_COLOR	23
5.8.2	Variable Documentation	24

5.8.2.1	GTG_BLACK	24
5.8.2.2	GTG_BLUE	24
5.8.2.3	GTG_BROWN	24
5.8.2.4	GTG_DARKBLUE	24
5.8.2.5	GTG_DARKGREY	24
5.8.2.6	GTG_DARKPINK	24
5.8.2.7	GTG_GREEN	24
5.8.2.8	GTG_GRENAT	24
5.8.2.9	GTG_KAKI	24
5.8.2.10	GTG_LIGHTBROWN	24
5.8.2.11	GTG_LIGHTGREY	25
5.8.2.12	GTG_LIGHTPINK	25
5.8.2.13	GTG_MAUVE	25
5.8.2.14	GTG_ORANGE	25
5.8.2.15	GTG_PINK	25
5.8.2.16	GTG_PURPLE	25
5.8.2.17	GTG_RED	25
5.8.2.18	GTG_REDBLOOD	25
5.8.2.19	GTG_SEABLUE	25
5.8.2.20	GTG_TEAL	25
5.8.2.21	GTG_WHITE	26
5.8.2.22	GTG_YELLOW	26
5.9	Memory management	27
5.9.1	Function Documentation	27
5.9.1.1	gtg_block_free	27
5.9.1.2	gtg_block_malloc	27
5.9.1.3	gtg_block_memory_init	27
5.10	OTF interface in C of the traceGeneratorBasic API	28
5.10.1	Function Documentation	29
5.10.1.1	OTF_get_color	29
5.10.1.2	OTFAddComment	29
5.10.1.3	OTFAddContType	30
5.10.1.4	OTFAddEntityValue	30
5.10.1.5	OTFAddEvent	30

5.10.1.6	OTFAddEventType	31
5.10.1.7	OTFAddLinkType	31
5.10.1.8	OTFAddStateType	31
5.10.1.9	OTFAddVar	32
5.10.1.10	OTFAddVarType	32
5.10.1.11	OTFDestroyContainer	33
5.10.1.12	OTFEndLink	33
5.10.1.13	OTFEndTrace	33
5.10.1.14	OTFInitTrace	34
5.10.1.15	OTFPopState	34
5.10.1.16	OTFPushState	34
5.10.1.17	OTFSetCompress	35
5.10.1.18	OTFSetState	35
5.10.1.19	OTFSetVar	35
5.10.1.20	OTFStartContainer	36
5.10.1.21	OTFStartLink	36
5.10.1.22	OTFSubVar	36
5.11	Paje interface in C of the GTGBasic1 API	38
5.11.1	Function Documentation	39
5.11.1.1	Paje_get_color	39
5.11.1.2	pajeAddComment	39
5.11.1.3	pajeAddContainer	40
5.11.1.4	pajeAddContType	40
5.11.1.5	pajeAddEntityValue	41
5.11.1.6	pajeAddEvent	41
5.11.1.7	pajeAddEventType	41
5.11.1.8	pajeAddLinkType	42
5.11.1.9	pajeAddStateType	42
5.11.1.10	pajeAddVar	42
5.11.1.11	pajeAddVarType	43
5.11.1.12	pajeDestroyContainer	43
5.11.1.13	pajeEndLink	44
5.11.1.14	pajeEndTrace	44
5.11.1.15	pajeGetName	44

5.11.1.16	pajeInitTrace	44
5.11.1.17	pajePopState	45
5.11.1.18	pajePushState	45
5.11.1.19	pajeSeqAddContainer	46
5.11.1.20	pajeSetCompress	46
5.11.1.21	pajeSetState	46
5.11.1.22	pajeSetVar	47
5.11.1.23	pajeStartLink	47
5.11.1.24	pajeSubVar	47
5.11.1.25	viteEndTrace	48
5.12	Functions for postponing event-processing function calls	49
5.12.1	Function Documentation	49
5.12.1.1	gtg_record	49
5.12.1.2	gtg_write_events	49
5.13	Types used	50
5.13.1	Typedef Documentation	50
5.13.1.1	varPrec	50
5.13.2	Enumeration Type Documentation	50
5.13.2.1	trace_return_t	50
6	Data Structure Documentation	51
6.1	Container Struct Reference	51
6.1.1	Detailed Description	51
6.1.2	Field Documentation	51
6.1.2.1	alias	51
6.1.2.2	ctType	51
6.1.2.3	id	51
6.1.2.4	name	51
6.1.2.5	state_stack	51
6.1.2.6	token	52
6.2	ContainerType Struct Reference	52
6.2.1	Field Documentation	52
6.2.1.1	alias	52
6.2.1.2	id	52

6.2.1.3	name	52
6.2.1.4	token	52
6.3	EntityValue Struct Reference	52
6.3.1	Detailed Description	53
6.3.2	Field Documentation	53
6.3.2.1	alias	53
6.3.2.2	groupId	53
6.3.2.3	id	53
6.3.2.4	name	53
6.3.2.5	token	53
6.4	EventType Struct Reference	53
6.4.1	Detailed Description	53
6.4.2	Field Documentation	53
6.4.2.1	alias	53
6.4.2.2	contType	53
6.4.2.3	id	53
6.4.2.4	name	54
6.4.2.5	token	54
6.5	gtg_color Struct Reference	54
6.5.1	Detailed Description	54
6.5.2	Field Documentation	54
6.5.2.1	color_name	54
6.5.2.2	rgb	54
6.6	gtg_list Struct Reference	54
6.6.1	Field Documentation	55
6.6.1.1	next	55
6.6.1.2	prev	55
6.7	gtg_memory Struct Reference	55
6.7.1	Field Documentation	55
6.7.1.1	block_len	55
6.7.1.2	current_mem	55
6.7.1.3	first_free	55
6.7.1.4	first_mem	55
6.7.1.5	first_new	55

6.7.1.6	mem_len	55
6.7.1.7	nb_allocated	56
6.8	Link Struct Reference	56
6.8.1	Field Documentation	56
6.8.1.1	src	56
6.8.1.2	time	56
6.9	LinkType Struct Reference	56
6.9.1	Detailed Description	56
6.9.2	Field Documentation	57
6.9.2.1	alias	57
6.9.2.2	contType	57
6.9.2.3	destType	57
6.9.2.4	id	57
6.9.2.5	name	57
6.9.2.6	srcType	57
6.9.2.7	token	57
6.10	otf_color Struct Reference	57
6.10.1	Field Documentation	57
6.10.1.1	blue	57
6.10.1.2	colorID	57
6.10.1.3	green	57
6.10.1.4	red	57
6.11	State Struct Reference	58
6.11.1	Detailed Description	58
6.11.2	Field Documentation	58
6.11.2.1	cont	58
6.11.2.2	stateType	58
6.11.2.3	token	58
6.11.2.4	value	58
6.12	StateType Struct Reference	58
6.12.1	Detailed Description	59
6.12.2	Field Documentation	59
6.12.2.1	alias	59
6.12.2.2	groupId	59

6.12.2.3	id	59
6.12.2.4	name	59
6.12.2.5	token	59
6.13	Variable Struct Reference	59
6.13.1	Field Documentation	59
6.13.1.1	id	59
6.13.1.2	parent	59
6.13.1.3	token	59
6.13.1.4	type	59
6.13.1.5	value	59
6.14	VariableType Struct Reference	60
6.14.1	Detailed Description	60
6.14.2	Field Documentation	60
6.14.2.1	alias	60
6.14.2.2	contType	60
6.14.2.3	id	60
6.14.2.4	name	60
6.14.2.5	token	60
7	File Documentation	61
7.1	GTG.h File Reference	61
7.1.1	Detailed Description	61
7.2	GTGBasic.h File Reference	61
7.2.1	Detailed Description	64
7.2.2	Define Documentation	64
7.2.2.1	GTG_FLAG_NONE	64
7.2.2.2	GTG_FLAG_NOTBUF	64
7.2.2.3	GTG_FLAG_OUTOFORDER	64
7.2.2.4	GTG_FLAG_USE_MPI	64
7.2.3	Typedef Documentation	64
7.2.3.1	gtg_flag_t	64
7.2.3.2	traceType_t	65
7.3	GTGColor.h File Reference	65
7.3.1	Detailed Description	67

7.3.2	Define Documentation	67
7.3.2.1	GTG_COLOR_BLUE_MASK	67
7.3.2.2	GTG_COLOR_BLUE_POS	67
7.3.2.3	GTG_COLOR_GREEN_MASK	67
7.3.2.4	GTG_COLOR_GREEN_POS	67
7.3.2.5	GTG_COLOR_RED_MASK	67
7.3.2.6	GTG_COLOR_RED_POS	67
7.3.3	Typedef Documentation	67
7.3.3.1	gtg_color_t	67
7.3.3.2	gtg_rgb_color_t	67
7.3.4	Function Documentation	67
7.3.4.1	gtg_color_create	67
7.3.4.2	gtg_color_exit	67
7.3.4.3	gtg_color_free	67
7.3.4.4	gtg_color_init	67
7.4	GTGCompress.h File Reference	67
7.4.1	Function Documentation	68
7.4.1.1	gtg_compress_f2f	68
7.4.1.2	gtg_compress_f2m	68
7.4.1.3	gtg_compress_init	68
7.4.1.4	gtg_compress_m2f	68
7.4.1.5	gtg_compress_m2m	68
7.4.1.6	gtg_decompress_f2f	68
7.4.1.7	gtg_decompress_f2m	68
7.4.1.8	gtg_decompress_init	68
7.4.1.9	gtg_decompress_m2f	68
7.4.1.10	gtg_decompress_m2m	68
7.5	GTGList.h File Reference	68
7.5.1	Define Documentation	69
7.5.1.1	GTG_LIST	69
7.5.1.2	gtg_list_entry	70
7.5.1.3	gtg_list_for_each	70
7.5.1.4	gtg_list_for_each_entry	70
7.5.1.5	gtg_list_for_each_entry_safe	70

7.5.1.6	gtg_list_for_each_reverse	71
7.5.1.7	gtg_list_for_each_safe	71
7.5.1.8	GTG_LIST_INIT	71
7.5.2	Typedef Documentation	71
7.5.2.1	gtg_list_t	71
7.5.3	Function Documentation	71
7.5.3.1	__gtg_list_add	71
7.5.3.2	__gtg_list_del	71
7.5.3.3	gtg_list_add	72
7.5.3.4	gtg_list_add_tail	72
7.5.3.5	gtg_list_del	72
7.5.3.6	gtg_list_size	72
7.6	GTGMemory.h File Reference	72
7.6.1	Detailed Description	73
7.6.2	Typedef Documentation	73
7.6.2.1	gtg_memory_t	73
7.7	GTGOTF.h File Reference	73
7.7.1	Detailed Description	73
7.8	GTGOTF_Basic.h File Reference	74
7.8.1	Detailed Description	75
7.8.2	Function Documentation	76
7.8.2.1	OTFDefineContainer	76
7.9	GTGOTF_Structs.h File Reference	76
7.9.1	Detailed Description	77
7.9.2	Define Documentation	77
7.9.2.1	alloc_init_struct	77
7.9.2.2	alloc_State	78
7.9.2.3	alloc_struct	78
7.9.2.4	alloc_Variable	78
7.9.2.5	Container_NIL	78
7.9.2.6	ContainerType_NIL	78
7.9.2.7	EntityValue_NIL	78
7.9.2.8	EventType_NIL	79
7.9.2.9	free_struct	79

7.9.2.10	init_Container	79
7.9.2.11	init_ContainerType	79
7.9.2.12	init_EntityValue	79
7.9.2.13	init_EventType	80
7.9.2.14	init_LinkType	80
7.9.2.15	init_State	80
7.9.2.16	init_StateType	80
7.9.2.17	init_Variable	81
7.9.2.18	init_VariableType	81
7.9.2.19	LinkType_NIL	81
7.9.2.20	MAX_PROCESS	81
7.9.2.21	State_NIL	81
7.9.2.22	StateType_NIL	81
7.9.2.23	Variable_NIL	81
7.9.2.24	VariableType_NIL	81
7.9.3	Typedef Documentation	81
7.9.3.1	Container_t	81
7.9.3.2	ContainerType_t	81
7.9.3.3	EntityValue_t	81
7.9.3.4	EventType_t	82
7.9.3.5	Link_t	82
7.9.3.6	LinkType_t	82
7.9.3.7	otf_color_t	82
7.9.3.8	State_t	82
7.9.3.9	StateType_t	82
7.9.3.10	Variable_t	82
7.9.3.11	VariableType_t	82
7.10	GTGPaje.h File Reference	82
7.10.1	Detailed Description	83
7.10.2	Typedef Documentation	83
7.10.2.1	paje_color_t	83
7.11	GTGPaje_Basic.h File Reference	83
7.11.1	Detailed Description	85
7.11.2	Define Documentation	85

7.11.2.1	FMT_PAJE	85
7.11.2.2	FMT_VITE	86
7.11.3	Enumeration Type Documentation	86
7.11.3.1	gtg_paje_evtdef_e	86
7.11.3.2	gtg_paje_fieldtype_e	86
7.11.4	Function Documentation	87
7.11.4.1	pajeEventDefAddParam	87
7.12	GTGReplay.h File Reference	87
7.12.1	Detailed Description	87
7.12.2	Enumeration Type Documentation	87
7.12.2.1	event_type_t	87
7.13	GTGStack.h File Reference	88
7.13.1	Define Documentation	88
7.13.1.1	GTG_STACK	88
7.13.1.2	gtg_stack_entry	88
7.13.1.3	GTG_STACK_INIT	88
7.13.2	Typedef Documentation	88
7.13.2.1	gtg_stack	88
7.13.2.2	gtg_stack_t	88
7.13.3	Function Documentation	88
7.13.3.1	gtg_stack_empty	89
7.13.3.2	gtg_stack_pop	89
7.13.3.3	gtg_stack_push	89
7.13.3.4	gtg_stack_top	89
7.14	GTGTypes.h File Reference	89
7.14.1	Typedef Documentation	89
7.14.1.1	trace_return_t	89

Chapter 1

The GTG library

(\)
(*-*)
("")

1.1 Presentation

The GTG library provides a low level library to generate traces in various formats (Paje, OTF).

The use of the library is simple, you just need to include the [GTG.h](#) header and then you can use the library as you wish.

Some simple examples are available in the test directory.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Trace type handler	9
To init the generated trace file(s)	11
Functions related to the containers	13
Functions related to the states	15
Functions related to the events	17
Functions related to links	18
Functions related to variables	20
Defined colors for GTG	22
Memory management	27
OTF interface in C of the traceGeneratorBasic API	28
Paje interface in C of the GTGBasic1 API	38
Functions for postponing event-processing function calls	49
Types used	50

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Container	51
ContainerType	52
EntityValue	52
EventType	53
gtg_color	
This structure defines a color that can be used by GTG	54
gtg_list	54
gtg_memory	55
Link	56
LinkType	56
otf_color	57
State	58
StateType	58
Variable	59
VariableType	60

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

GTG.h	Generic header to include	61
GTGBasic.h	GTGBasic is a basic interface to generate trace in various formats	61
GTGColor.h	This file defines some useful colors to use in entity values for GTG	65
GTGCompress.h	67
GTGList.h	68
GTGMemory.h	This file defines a fast allocator for fixed-size blocks	72
GTGOTF.h	OTF is the global file for gtg interface using OTF	73
GTGOTF_Basic.h	OTF_GTGBasic1 is the OTF implementation of the basic interface to generate traces (GTGBasic1)	74
GTGOTF_Structs.h	OTF_Structs gives the global types and functions needed to have the OTF implementation	76
GTGPaje.h	PajeColor is a file that defines function that manipulate colors	82
GTGPaje_Basic.h	Paje_GTGBasic1 is the Paje implementation of the basic interface to generate traces (GTGBasic1)	83
GTGReplay.h	This file defines functions for postponing event-processing function calls	87
GTGStack.h	88
GTGTypes.h	89

Chapter 5

Module Documentation

5.1 Trace type handler

Enumerations

- enum `traceType` { `PAJE`, `VITE`, `OTF`, `TAU` }

The type of the output trace.

Functions

- void `setTraceType` (`traceType_t` type)
Set the type of output trace.
- `traceType_t` `getTraceType` ()
Get the type of the output trace.
- char * `getName` (int procRk)
To get the name of the file to give to the addCont function for processors.
- int `bufferedModeActivated` ()
Check whether the buffered-mode is activated.

5.1.1 Enumeration Type Documentation

5.1.1.1 enum `traceType`

The type of the output trace.

Enumerator:

- PAJE*** Paje trace format.
- VITE*** ViTE-specific trace format.
- OTF*** OTF trace format.
- TAU*** TAU Trace format.

5.1.2 Function Documentation

5.1.2.1 `int bufferedModeActivated ()`

Check wether the buffered-mode is activated.

Returns

1 is the buffered-mode is activate.
0 otherwise.

5.1.2.2 `traceType_t getName (int procRk)`

To get the name of the file to give to the addCont function for processors.

Parameters

<i>procRk</i>	Rank of the proc to get the file containing it
---------------	------------------------------------------------

Returns

The name of the file to give for a proc

5.1.2.3 `traceType_t getTraceType ()`

Get the type of the output trace.

Returns

The type of the trace

5.1.2.4 `void setTraceType (traceType_t type)`

Set the type of output trace.

Parameters

<i>type</i>	Type of trace to generate
-------------	---------------------------

5.2 To init the generated trace file(s)

Functions

- `trace_return_t initTrace` (const char *filename, int rank, `gtg_flag_t` flags)
Initialize a trace.
- `trace_return_t endTrace` ()
Finalize a trace.
- `trace_return_t setCompress` (int val)
Enable trace compression (only available for OTF traces).
- `trace_return_t addEntityValue` (const char *alias, const char *entType, const char *name, `gtg_color_t` p_color)
Add an Entity Value.

5.2.1 Function Documentation

5.2.1.1 `trace_return_t addEntityValue` (const char * *alias*, const char * *entType*, const char * *name*, `gtg_color_t` *p_color*)

Add an Entity Value.

Parameters

<i>alias</i>	Alias on the entity value
<i>entType</i>	Type of the entity that can have the value
<i>name</i>	Alternative name of the variable type
<i>p_color</i>	Color of the entity

Returns

TRACE_SUCCESS on success
An error code otherwise

5.2.1.2 `trace_return_t endTrace` ()

Finalize a trace.

Returns

TRACE_SUCCESS on success
An error code otherwise

5.2.1.3 `int initTrace` (const char * *filename*, int *rank*, `gtg_flag_t` *flags*)

Initialize a trace.

Parameters

<i>filename</i>	Root name of the file to create
<i>rank</i>	Process number of the file to create
<i>flags</i>	One of GTG_FLAG_NONE, GTG_FLAG_USE_MPI, GTG_FLAG_NO-TBUF.

Returns

TRACE_SUCCESS on success
An error code otherwise

5.2.1.4 trace_return_t setCompress (int val)

Enable trace compression (only available for OTF traces).

Parameters

<i>val</i>	0 means no compression, otherwise the output files will be compressed
------------	-----------------------------------------------------------------------

Returns

TRACE_SUCCESS on success
An error code otherwise

5.3 Functions related to the containers

Functions

- [trace_return_t addContType](#) (const char *alias, const char *contType, const char *name)
Add a [Container](#) Type.
- [trace_return_t addContainer](#) (varPrec time, const char *alias, const char *type, const char *container, const char *name, const char *file)
Add a [Container](#).
- [trace_return_t destroyContainer](#) (varPrec time, const char *name, const char *type)
Destroy a [Container](#).
- [trace_return_t AddComment](#) (const char *comment)
Add some Comment in Trace file.

5.3.1 Function Documentation

5.3.1.1 [trace_return_t AddComment](#) (const char * *comment*)

Add some Comment in Trace file.

Parameters

<i>comment</i>	Comment to be added
----------------	---------------------

Returns

TRACE_SUCCESS on success
An error code otherwise

5.3.1.2 [trace_return_t addContainer](#) (varPrec *time*, const char * *alias*, const char * *type*, const char * *container*, const char * *name*, const char * *file*)

Add a [Container](#).

Parameters

<i>time</i>	Time at which the container is added
<i>alias</i>	Alias of the new container
<i>type</i>	Type of the new container
<i>container</i>	Container parent
<i>name</i>	Alternative name of the variable type
<i>file</i>	File containing the container for vite format. Use "0" or "" chains for other formats.

Returns

TRACE_SUCCESS on success
An error code otherwise

5.3.1.3 `trace_return_t addContType (const char * alias, const char * contType, const char * name)`

Add a [Container](#) Type.

Parameters

<i>alias</i>	Alias on the container added
<i>contType</i>	Type of the parent container
<i>name</i>	Alternative name of the new container type

Returns

TRACE_SUCCESS on success
An error code otherwise

5.3.1.4 `trace_return_t destroyContainer (varPrec time, const char * name, const char * type)`

Destroy a [Container](#).

Parameters

<i>time</i>	Time at which the container is destroyed
<i>name</i>	Name of the container
<i>type</i>	Type of the container

Returns

TRACE_SUCCESS on success
An error code otherwise

5.4 Functions related to the states

Functions

- `trace_return_t addStateType` (const char *alias, const char *contType, const char *name)
Add a *State* Type.
- `trace_return_t setState` (varPrec time, const char *type, const char *cont, const char *val)
Set the *State* of a *Container*.
- `trace_return_t pushState` (varPrec time, const char *type, const char *cont, const char *val)
Save the current *State* on a stack and change the *State* of a *Container*.
- `trace_return_t popState` (varPrec time, const char *type, const char *cont)
Revert the *State* of a *Container* to its previous value.

5.4.1 Function Documentation

5.4.1.1 `trace_return_t addStateType (const char * alias, const char * contType, const char * name)`

Add a *State* Type.

Parameters

<i>alias</i>	Alias on the state type added
<i>contType</i>	Type of container of these states
<i>name</i>	Alternative name of the state type

Returns

TRACE_SUCCESS on success
An error code otherwise

5.4.1.2 `trace_return_t popState (varPrec time, const char * type, const char * cont)`

Revert the *State* of a *Container* to its previous value.

Parameters

<i>time</i>	Time the state changes
<i>type</i>	Type of the state
<i>cont</i>	<i>Container</i> whose state changes

Returns

TRACE_SUCCESS on success
An error code otherwise

5.4.1.3 `trace_return_t pushState (varPrec time, const char * type, const char * cont, const char * val)`

Save the current [State](#) on a stack and change the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time the state changes
<i>type</i>	Type of the state
<i>cont</i>	Container whose state changes
<i>val</i>	Value of state of container

Returns

TRACE_SUCCESS on success
An error code otherwise

5.4.1.4 `trace_return_t setState (varPrec time, const char * type, const char * cont, const char * val)`

Set the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time the state changes
<i>type</i>	Type of the state
<i>cont</i>	Container whose state changes
<i>val</i>	Value of new state of container

Returns

TRACE_SUCCESS on success
An error code otherwise

5.5 Functions related to the events

Functions

- `trace_return_t addEventType` (const char *alias, const char *contType, const char *name)

Add an Event Type.

- `trace_return_t addEvent` (varPrec time, const char *type, const char *cont, const char *val)

Add an Event.

5.5.1 Function Documentation

5.5.1.1 `trace_return_t addEvent` (varPrec time, const char * type, const char * cont, const char * val)

Add an Event.

Parameters

<i>time</i>	Time the event happens
<i>type</i>	Type of the event
<i>cont</i>	Container that produced the event
<i>val</i>	Value of the new event

Returns

TRACE_SUCCESS on success
An error code otherwise

5.5.1.2 `trace_return_t addEventType` (const char * alias, const char * contType, const char * name)

Add an Event Type.

Parameters

<i>alias</i>	Alias on the event type
<i>contType</i>	Type of container of these events
<i>name</i>	Alternative name of the event type

Returns

TRACE_SUCCESS on success
An error code otherwise

5.6 Functions related to links

Functions

- `trace_return_t addLinkType` (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)

Add a [Link](#) Type.

- `trace_return_t startLink` (varPrec time, const char *type, const char *cont, const char *src, const char *dest, const char *val, const char *key)

Start a [Link](#).

- `trace_return_t endLink` (varPrec time, const char *type, const char *cont, const char *src, const char *dest, const char *val, const char *key)

End a [Link](#).

5.6.1 Function Documentation

5.6.1.1 `trace_return_t addLinkType (const char * alias, const char * name, const char * contType, const char * srcContType, const char * destContType)`

Add a [Link](#) Type.

Parameters

<i>alias</i>	Alias on the link type
<i>name</i>	Alternative name of the link type
<i>contType</i>	Type of common ancestral container
<i>srcContType</i>	Type of the source container
<i>destCont-Type</i>	Type of the destination container

Returns

TRACE_SUCCESS on success
An error code otherwise

5.6.1.2 `trace_return_t endLink (varPrec time, const char * type, const char * cont, const char * src, const char * dest, const char * val, const char * key)`

End a [Link](#).

Parameters

<i>time</i>	Time the link ends
<i>type</i>	Type of the link
<i>cont</i>	Container containing the link (an ancestor of source and destination container)
<i>src</i>	Source container

<i>dest</i>	Destination container
<i>val</i>	Value of the link
<i>key</i>	Key to match the start link

Returns

TRACE_SUCCESS on success
An error code otherwise

5.6.1.3 `trace_return_t startLink (varPrec time, const char * type, const char * cont,
const char * src, const char * dest, const char * val, const char * key)`

Start a [Link](#).

Parameters

<i>time</i>	Time the link starts
<i>type</i>	Type of the link
<i>cont</i>	Container containing the link (an ancestor of source and destination container)
<i>src</i>	Source container
<i>dest</i>	Destination container
<i>val</i>	Value of the link
<i>key</i>	Key to match the end link

Returns

TRACE_SUCCESS on success
An error code otherwise

5.7 Functions related to variables

Functions

- `trace_return_t addVarType` (const char *alias, const char *name, const char *contType)

Add a *Variable* Type.

- `trace_return_t setVar` (varPrec time, const char *type, const char *cont, varPrec val)

Set a *Variable* value.

- `trace_return_t addVar` (varPrec time, const char *type, const char *cont, varPrec val)

Add a value to a *Variable*.

- `trace_return_t subVar` (varPrec time, const char *type, const char *cont, varPrec val)

Subtract a value from a *Variable*.

5.7.1 Function Documentation

5.7.1.1 `trace_return_t addVar` (varPrec time, const char * type, const char * cont, varPrec val)

Add a value to a *Variable*.

Parameters

<i>time</i>	Time the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	<i>Container</i> containing the variable
<i>val</i>	Value added

Returns

TRACE_SUCCESS on success
An error code otherwise

5.7.1.2 `trace_return_t addVarType` (const char * alias, const char * name, const char * contType)

Add a *Variable* Type.

Parameters

<i>alias</i>	Alias on the variable type
<i>contType</i>	Type of container
<i>name</i>	Alternative name of the variable type

Returns

TRACE_SUCCESS on success
An error code otherwise

5.7.1.3 `trace_return_t setVar (varPrec time, const char * type, const char * cont, varPrec val)`

Set a [Variable](#) value.

Parameters

<i>time</i>	Time the variable changes
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	New value of the variable

Returns

TRACE_SUCCESS on success
An error code otherwise

5.7.1.4 `trace_return_t subVar (varPrec time, const char * type, const char * cont, varPrec val)`

Subtract a value from a [Variable](#).

Parameters

<i>time</i>	Time the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value subtracted

Returns

TRACE_SUCCESS on success
An error code otherwise

5.8 Defined colors for GTG

Data Structures

- struct [gtg_color](#)

This structure defines a color that can be used by GTG.

Functions

- static uint8_t [GTG_COLOR_GET_BLUE](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the blue component of a rgb color.
- static uint8_t [GTG_COLOR_GET_GREEN](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the green component of a rgb color.
- static uint8_t [GTG_COLOR_GET_RED](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the red component of a rgb color.
- static [gtg_rgb_color_t](#) [GTG_COLOR_SET_COLOR](#) (uint8_t r, uint8_t g, uint8_t b)
Return the 4-bytes RGB color from 3 1-byte components.

Variables

- [gtg_color_t](#) [GTG_BLACK](#)
Default black color. (R,G,B) = (0, 0, 0)
- [gtg_color_t](#) [GTG_RED](#)
Default red color. (R,G,B) = (255, 0, 0)
- [gtg_color_t](#) [GTG_GREEN](#)
Default green color. (R,G,B) = (0, 255, 0)
- [gtg_color_t](#) [GTG_BLUE](#)
Default blue color. (R,G,B) = (0, 0, 255)
- [gtg_color_t](#) [GTG_WHITE](#)
Default white color. (R,G,B) = (255, 255, 255)
- [gtg_color_t](#) [GTG_TEAL](#)
Default teal color. (R,G,B) = (0, 255, 255)
- [gtg_color_t](#) [GTG_DARKGREY](#)
Default dark grey color. (R,G,B) = (85, 85, 85)
- [gtg_color_t](#) [GTG_YELLOW](#)
Default yellow color. (R,G,B) = (255, 255, 0)
- [gtg_color_t](#) [GTG_PURPLE](#)
Default purple color. (R,G,B) = (153, 25, 230)
- [gtg_color_t](#) [GTG_LIGHTBROWN](#)
Default light brown color. (R,G,B) = (170, 130, 130)
- [gtg_color_t](#) [GTG_LIGHTGREY](#)
Default light grey color. (R,G,B) = (200, 200, 200)

- `gtg_color_t` GTG_DARKBLUE
Default dark blue color. $(R,G,B) = (0, 0, 80)$
- `gtg_color_t` GTG_PINK
Default pink color. $(R,G,B) = (255, 0, 255)$
- `gtg_color_t` GTG_DARKPINK
Default dark pink color. $(R,G,B) = (180, 80, 180)$
- `gtg_color_t` GTG_SEABLUE
Default sea blue color. $(R,G,B) = (25, 128, 200)$
- `gtg_color_t` GTG_KAKI
Default kaki color. $(R,G,B) = (80, 100, 25)$
- `gtg_color_t` GTG_REDBLOOD
Default red blood color. $(R,G,B) = (200, 25, 25)$
- `gtg_color_t` GTG_BROWN
Default brown color. $(R,G,B) = (100, 25, 25)$
- `gtg_color_t` GTG_GRENAT
Default grenat color. $(R,G,B) = (100, 0, 80)$
- `gtg_color_t` GTG_ORANGE
Default orange color. $(R,G,B) = (255, 160, 0)$
- `gtg_color_t` GTG_MAUVE
Default mauve color. $(R,G,B) = (128, 0, 255)$
- `gtg_color_t` GTG_LIGHTPINK
Default light pink color. $(R,G,B) = (255, 128, 255)$

5.8.1 Function Documentation

5.8.1.1 GTG_COLOR_GET_BLUE (`gtg_rgb_color_t rgb`) [inline, static]

Return the 1-byte value of the blue component of a rgb color.

5.8.1.2 GTG_COLOR_GET_GREEN (`gtg_rgb_color_t rgb`) [inline, static]

Return the 1-byte value of the green component of a rgb color.

5.8.1.3 GTG_COLOR_GET_RED (`gtg_rgb_color_t rgb`) [inline, static]

Return the 1-byte value of the red component of a rgb color.

5.8.1.4 GTG_COLOR_SET_COLOR (`uint8_t r, uint8_t g, uint8_t b`) [inline, static]

Return the 4-bytes RGB color from 3 1-byte components.

5.8.2 Variable Documentation

5.8.2.1 GTG_BLACK

Default black color. (R,G,B) = (0, 0, 0)

5.8.2.2 GTG_BLUE

Default blue color. (R,G,B) = (0, 0, 255)

5.8.2.3 GTG_BROWN

Default brown color. (R,G,B) = (100, 25, 25)

5.8.2.4 GTG_DARKBLUE

Default dark blue color. (R,G,B) = (0, 0, 80)

5.8.2.5 GTG_DARKGREY

Default dark grey color. (R,G,B) = (85, 85, 85)

5.8.2.6 GTG_DARKPINK

Default dark pink color. (R,G,B) = (180, 80, 180)

5.8.2.7 GTG_GREEN

Default green color. (R,G,B) = (0, 255, 0)

5.8.2.8 GTG_GRENAT

Default grenat color. (R,G,B) = (100, 0, 80)

5.8.2.9 GTG_KAKI

Default kaki color. (R,G,B) = (80, 100, 25)

5.8.2.10 GTG_LIGHTBROWN

Default light brown color. (R,G,B) = (170, 130, 130)

5.8.2.11 GTG_LIGHTGREY

Default light grey color. (R,G,B) = (200, 200, 200)

5.8.2.12 GTG_LIGHTPINK

Default light pink color. (R,G,B) = (255, 128, 255)

5.8.2.13 GTG_MAUVE

Default mauve color. (R,G,B) = (128, 0, 255)

5.8.2.14 GTG_ORANGE

Default orange color. (R,G,B) = (255, 160, 0)

5.8.2.15 GTG_PINK

Default pink color. (R,G,B) = (255, 0, 255)

5.8.2.16 GTG_PURPLE

Default purple color. (R,G,B) = (153, 25, 230)

5.8.2.17 GTG_RED

Default red color. (R,G,B) = (255, 0, 0)

5.8.2.18 GTG_REDBLOOD

Default red blood color. (R,G,B) = (200, 25, 25)

5.8.2.19 GTG_SEABLUE

Default sea blue color. (R,G,B) = (25, 128, 200)

5.8.2.20 GTG_TEAL

Default teal color. (R,G,B) = (0, 255, 255)

5.8.2.21 GTG_WHITE

Default white color. (R,G,B) = (255, 255, 255)

5.8.2.22 GTG_YELLOW

Default yellow color. (R,G,B) = (255, 255, 0)

5.9 Memory management

Functions

- void [gtg_block_memory_init](#) ([gtg_memory_t](#) *memory, size_t block_size, long initial_block_number)
Initialize the allocator.
- void * [gtg_block_malloc](#) ([gtg_memory_t](#) memory)
Allocate a block of data.
- void [gtg_block_free](#) ([gtg_memory_t](#) memory, void *ptr)
Free a block of data.

5.9.1 Function Documentation

5.9.1.1 void [gtg_block_free](#) ([gtg_memory_t](#) memory, void * ptr)

Free a block of data.

Parameters

<i>memory</i>	The memory describer
<i>ptr</i>	The block of data to free

5.9.1.2 void * [gtg_block_malloc](#) ([gtg_memory_t](#) memory)

Allocate a block of data.

Parameters

<i>memory</i>	The memory describer
---------------	----------------------

Returns

A pointer to a block or NULL if allocation failed

5.9.1.3 void [gtg_block_memory_init](#) ([gtg_memory_t](#) * memory, size_t block_size, long initial_block_number)

Initialize the allocator.

Parameters

<i>memory</i>	A memory describer
<i>block_size</i>	The block size to be allocated when malloc is called
<i>initial_block-number</i>	The number of blocks to allocate initially

5.10 OTF interface in C of the traceGeneratorBasic API

Functions

- `const otf_color_t OTF_get_color (gtg_color_t color)`
Converts a GTG color into a OTF color.
- `trace_return_t OTFInitTrace (const char *filename, gtg_flag_t flags)`
Initialize an OTF trace.
- `trace_return_t OTFSetCompress (int val)`
Enable trace compression.
- `trace_return_t OTFAddContType (const char *alias, const char *contType, const char *name)`
Add a [Container](#) Type.
- `trace_return_t OTFAddStateType (const char *alias, const char *contType, const char *name)`
Add a [State](#) Type.
- `trace_return_t OTFAddEventType (const char *alias, const char *contType, const char *name)`
Add an Event Type.
- `trace_return_t OTFAddLinkType (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)`
Add a [Link](#) Type.
- `trace_return_t OTFAddVarType (const char *alias, const char *name, const char *contType)`
Add a [Variable](#) Type.
- `trace_return_t OTFAddEntityValue (const char *alias, const char *entType, const char *name, const otf_color_t color)`
Add an Entity Value.
- `trace_return_t OTFStartContainer (varPrec time, const char *alias, const char *type, const char *container, const char *name, const char *file)`
Start a [Container](#).
- `trace_return_t OTFDestroyContainer (varPrec time, const char *name, const char *type)`
Destroy a [Container](#).
- `trace_return_t OTFSetState (varPrec time, const char *type, const char *cont, const char *val)`
Set the [State](#) of a [Container](#).
- `trace_return_t OTFPushState (varPrec time, const char *type, const char *cont, const char *val)`
Save the current [State](#) on a stack and change the [State](#) of a [Container](#).
- `trace_return_t OTFPopState (varPrec time, const char *type, const char *cont)`
Revert the [State](#) of a [Container](#) to its previous value.
- `trace_return_t OTFAddEvent (varPrec time, const char *type, const char *cont, const char *val)`
Add an Event.

- `trace_return_t OTFStartLink` (`varPrec` time, `const char *type`, `const char *src`, `const char *dest`, `const char *val`, `const char *key`)
Start a [Link](#).
- `trace_return_t OTFEndLink` (`varPrec` time, `const char *type`, `const char *src`, `const char *dest`, `const char *val`, `const char *key`)
End a [Link](#).
- `trace_return_t OTFSetVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)
Set a [Variable](#) value.
- `trace_return_t OTFAddVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)
Add a value to a [Variable](#).
- `trace_return_t OTFSubVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)
Subtract a value from a [Variable](#).
- `trace_return_t OTFAddComment` (`const char *comment`)
Add some Comment in Trace file.
- `trace_return_t OTFEndTrace` ()
Finalize an OTF trace.

5.10.1 Function Documentation

5.10.1.1 `const char * OTF_get_color (gtg_color_t color)`

Converts a GTG color into a OTF color.

Parameters

<code>color</code>	GTG color to convert
--------------------	----------------------

Returns

The OTF color

5.10.1.2 `trace_return_t OTFAddComment (const char * comment)`

Add some Comment in Trace file.

Parameters

<code>comment</code>	Comment to be added
----------------------	---------------------

Returns

TRACE_SUCCESS on success
An error code otherwise

5.10.1.3 **trace_return_t** OTFAddContType (const char * *alias*, const char * *contType*, const char * *name*)

Add a [Container](#) Type.

Parameters

<i>alias</i>	Alias on the container
<i>contType</i>	Type of container
<i>name</i>	Name of the container type

Returns

0 if success
An error code otherwise

5.10.1.4 **trace_return_t** OTFAddEntityValue (const char * *alias*, const char * *entType*, const char * *name*, const otf_color_t *color*)

Add an Entity Value.

Parameters

<i>alias</i>	Alias on the entity value
<i>entType</i>	Type of the entity
<i>name</i>	Name of the variable type
<i>color</i>	Color of the entity

Returns

0 if success
An error code otherwise

5.10.1.5 **trace_return_t** OTFAddEvent (varPrec *time*, const char * *type*, const char * *cont*, const char * *val*)

Add an Event.

Parameters

<i>time</i>	Time at which the event happens
<i>type</i>	Type of the event
<i>cont</i>	Container in this event
<i>val</i>	Entity value of the event of the container

Returns

0 if success
An error code otherwise

5.10.1.6 `trace_return_t OTFAddEventType (const char * alias, const char * contType,
const char * name)`

Add an Event Type.

Parameters

<i>alias</i>	Alias on the event type
<i>contType</i>	Type of container
<i>name</i>	Name of the event type

Returns

0 if success
An error code otherwise

5.10.1.7 `trace_return_t OTFAddLinkType (const char * alias, const char * name, const
char * contType, const char * srcContType, const char * destContType)`

Add a [Link](#) Type.

Parameters

<i>alias</i>	Alias on the link type
<i>name</i>	Name of the link type
<i>contType</i>	Type of container
<i>srcContType</i>	Type of the source container
<i>destCont- Type</i>	Type of the destination container

Returns

0 if success
An error code otherwise

5.10.1.8 `trace_return_t OTFAddStateType (const char * alias, const char * contType,
const char * name)`

Add a [State](#) Type.

Parameters

<i>alias</i>	Alias on the state type
<i>contType</i>	Type of container
<i>name</i>	Name of the state type

Returns

0 if success
An error code otherwise

5.10.1.9 **trace_return_t OTFAddVar (varPrec *time*, const char * *type*, const char * *cont*, varPrec *val*)**

Add a value to a [Variable](#).

Parameters

<i>time</i>	Time at which the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value added

Returns

0 if success
An error code otherwise

5.10.1.10 **trace_return_t OTFAddVarType (const char * *alias*, const char * *contType*, const char * *name*)**

Add a [Variable](#) Type.

Parameters

<i>alias</i>	Alias on the variable type
<i>contType</i>	Type of container
<i>name</i>	Name of the variable type

Returns

0 if success
An error code otherwise

5.10.1.11 `trace_return_t OTFDestroyContainer (varPrec time, const char * name,
const char * type)`

Destroy a [Container](#).

Parameters

<i>time</i>	Time at which the container is destroyed
<i>name</i>	Name of the container
<i>type</i>	Type of the container

Returns

0 if success
An error code otherwise

5.10.1.12 `trace_return_t OTFEndLink (varPrec time, const char * type, const char *
cont, const char * dest, const char * val, const char * key)`

End a [Link](#).

Parameters

<i>time</i>	Time at which the link ends
<i>type</i>	Type of the link
<i>cont</i>	Container containing the link
<i>dest</i>	Container destination
<i>val</i>	Entity value of the link
<i>key</i>	Key to identify the link

Returns

0 if success
An error code otherwise

5.10.1.13 `OTFEndTrace ()`

Finalize an OTF trace.

Returns

0 if success
An error code otherwise

5.10.1.14 `trace_return_t OTFInitTrace (const char * filename, gtg_flag_t flags)`

Initialize an OTF trace.

Parameters

<i>filename</i>	Root name of the file to create
<i>flags</i>	One of GTG_FLAG_NONE, GTG_FLAG_USE_MPI, GTG_FLAG_NO-TBUF.

Returns

0 if success An error code otherwise

5.10.1.15 `trace_return_t OTFPopState (varPrec time, const char * type, const char * cont)`

Revert the [State](#) of a [Container](#) to its previous value.

Parameters

<i>time</i>	Time at which the state is popped
<i>type</i>	Type of the state
<i>cont</i>	Container in this state

Returns

0 if success
An error code otherwise

5.10.1.16 `trace_return_t OTFPushState (varPrec time, const char * type, const char * cont, const char * val)`

Save the current [State](#) on a stack and change the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time at which the state is pushed
<i>type</i>	Type of the state
<i>cont</i>	Container in this state
<i>val</i>	Entity value of the state of the container

Returns

0 if success
An error code otherwise

5.10.1.17 `trace_return_t OTFSetCompress (int val)`

Enable trace compression.

Parameters

<i>val</i>	0 means no compression, otherwise the output files will be compressed.
------------	------------------------------------------------------------------------

Returns

0 if success
An error code otherwise

5.10.1.18 `trace_return_t OTFSetState (varPrec time, const char * type, const char * cont, const char * val)`

Set the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time at which the state is set
<i>type</i>	Type of the state
<i>cont</i>	Container in this state
<i>val</i>	Entity value of the state of the container

Returns

0 if success
An error code otherwise

5.10.1.19 `trace_return_t OTFSetVar (varPrec time, const char * type, const char * cont, varPrec val)`

Set a [Variable](#) value.

Parameters

<i>time</i>	Time at which the variable is set
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value of the variable

Returns

0 if success
An error code otherwise

5.10.1.20 `trace_return_t OTFStartContainer (varPrec time, const char * alias, const char * type, const char * container, const char * name, const char * file)`

Start a [Container](#).

Parameters

<i>time</i>	Time at which the container is added
<i>alias</i>	Alias of the new container
<i>type</i>	Type of the container
<i>container</i>	Container parent
<i>name</i>	Name of the variable type
<i>file</i>	File containing the container trace

Returns

0 if success
An error code otherwise

5.10.1.21 `trace_return_t OTFStartLink (varPrec time, const char * type, const char * cont, const char * src, const char * val, const char * key)`

Start a [Link](#).

Parameters

<i>time</i>	Time at which the link starts
<i>type</i>	Type of the link
<i>cont</i>	Container containing the link
<i>src</i>	Container source
<i>val</i>	Entity value of the link
<i>key</i>	Key to identify the link

Returns

0 if success
An error code otherwise

5.10.1.22 `trace_return_t OTFSubVar (varPrec time, const char * type, const char * cont, varPrec val)`

Subtract a value from a [Variable](#).

Parameters

<i>time</i>	Time at which the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value subtracted

Returns

0 if success
An error code otherwise

5.11 Paje interface in C of the GTGBasic1 API

Functions

- `const paje_color_t Paje_get_color (gtg_color_t p_color)`
Converts a GTG color into a PAJE color.
- `trace_return_t pajelnitTrace (const char *filename, int rank, gtg_flag_t flags, int fmt)`
*Initialize a VITE trace (*.ept)*
- `char * pajeGetName (int rk)`
Function to get the name of the file containing all the data for the proc of rank rk.
- `trace_return_t pajeSetCompress (int val)`
Enable trace compression.
- `trace_return_t pajeAddContType (const char *alias, const char *contType, const char *name)`
Add a [Container](#) Type.
- `trace_return_t pajeAddStateType (const char *alias, const char *contType, const char *name)`
Add a [State](#) Type.
- `trace_return_t pajeAddEventType (const char *alias, const char *contType, const char *name)`
Add an Event Type.
- `trace_return_t pajeAddLinkType (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)`
Add a [Link](#) Type.
- `trace_return_t pajeAddVarType (const char *alias, const char *name, const char *contType)`
Add a [Variable](#) Type.
- `trace_return_t pajeAddEntityValue (const char *alias, const char *entType, const char *name, const char *color)`
Add an Entity Value.
- `trace_return_t pajeAddContainer (varPrec time, const char *alias, const char *type, const char *container, const char *name, const char *file)`
Add a [Container](#) (VITE format).
- `trace_return_t pajeSeqAddContainer (varPrec time, const char *alias, const char *type, const char *container, const char *name)`
Add a [Container](#) (PAJE format).
- `trace_return_t pajeDestroyContainer (varPrec time, const char *name, const char *type)`
Destroy a [Container](#).
- `trace_return_t pajeSetState (varPrec time, const char *type, const char *cont, const char *val)`
Set the [State](#) of a [Container](#).
- `trace_return_t pajePushState (varPrec time, const char *type, const char *cont, const char *val)`

Save the current *State* on a stack and change the *State* of a *Container*.

- `trace_return_t pajePopState` (`varPrec` time, `const char *type`, `const char *cont`)

Revert the *State* of a *Container* to its previous value.

- `trace_return_t pajeAddEvent` (`varPrec` time, `const char *type`, `const char *cont`, `const char *val`)

Add an Event.

- `trace_return_t pajeStartLink` (`varPrec` time, `const char *type`, `const char *cont`, `const char *src`, `const char *val`, `const char *key`)

Start a link.

- `trace_return_t pajeEndLink` (`varPrec` time, `const char *type`, `const char *cont`, `const char *dest`, `const char *val`, `const char *key`)

Start a link.

- `trace_return_t pajeSetVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)

Set a *Variable* value.

- `trace_return_t pajeAddVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)

Add a value to a *Variable*.

- `trace_return_t pajeSubVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec val`)

Subtract a value from a *Variable*.

- `trace_return_t pajeAddComment` (`const char *comment`)

Add some Comment in Trace file.

- `trace_return_t pajeEndTrace` ()

Finalize a PAJE trace.

- `trace_return_t viteEndTrace` ()

Finalize a VITE trace.

5.11.1 Function Documentation

5.11.1.1 `const paje_color_t Paje_get_color (gtg_color_t color)`

Converts a GTG color into a PAJE color.

Parameters

<i>color</i>	GTG color to convert
--------------	----------------------

Returns

The PAJE color

5.11.1.2 `trace_return_t pajeAddComment (const char * comment)`

Add some Comment in Trace file.

Parameters

<i>comment</i>	Comment to be added
----------------	---------------------

Returns

TRACE_SUCCESS on success
An error code otherwise

5.11.1.3 **trace_return_t** **pajeAddContainer** (**varPrec** *time*, **const** **char** * *alias*, **const** **char** * *type*, **const** **char** * *container*, **const** **char** * *name*, **const** **char** * *file*)

Add a [Container](#) (VITE format).

Parameters

<i>time</i>	Time at which the container is added
<i>alias</i>	Alias on the new container
<i>type</i>	Type of the container
<i>container</i>	Container parent
<i>name</i>	Name of the variable type
<i>file</i>	File containing the container trace

Returns

0 if success
An error code otherwise

5.11.1.4 **trace_return_t** **pajeAddContType** (**const** **char** * *alias*, **const** **char** * *contType*, **const** **char** * *name*)

Add a [Container](#) Type.

Parameters

<i>alias</i>	Alias on the container
<i>contType</i>	Type of container
<i>name</i>	Name of the container type

Returns

0 if success
An error code otherwise

5.11.1.5 **trace_return_t** pajeAddEntityValue (const char * *alias*, const char * *entType*,
const char * *name*, const char * *color*)

Add an Entity Value.

Parameters

<i>alias</i>	Alias on the entity value
<i>entType</i>	Type of the entity
<i>name</i>	Name of the variable type
<i>color</i>	Color of the entity

Returns

0 if success
An error code otherwise

5.11.1.6 **trace_return_t** pajeAddEvent (varPrec *time*, const char * *type*, const char *
cont, const char * *val*)

Add an Event.

Parameters

<i>time</i>	Time at which the event happens
<i>type</i>	Type of the event
<i>cont</i>	Container in this event
<i>val</i>	Entity value of the event of the container

Returns

0 if success
An error code otherwise

5.11.1.7 **trace_return_t** pajeAddEventType (const char * *alias*, const char * *contType*,
const char * *name*)

Add an Event Type.

Parameters

<i>alias</i>	Alias on the event type
<i>contType</i>	Type of container
<i>name</i>	Name of the event type

Returns

0 if success
An error code otherwise

5.11.1.8 `trace_return_t pajeAddLinkType (const char * alias, const char * name, const char * contType, const char * srcContType, const char * destContType)`

Add a [Link](#) Type.

Parameters

<i>alias</i>	Alias on the link type
<i>name</i>	Name of the link type
<i>contType</i>	Type of container
<i>srcContType</i>	Type of the source container
<i>destContType</i>	Type of the destination container

Returns

0 if success
An error code otherwise

5.11.1.9 `trace_return_t pajeAddStateType (const char * alias, const char * contType, const char * name)`

Add a [State](#) Type.

Parameters

<i>alias</i>	Alias on the state type
<i>contType</i>	Type of container
<i>name</i>	Name of the state type

Returns

0 if success
An error code otherwise

5.11.1.10 `trace_return_t pajeAddVar (varPrec time, const char * type, const char * cont, varPrec val)`

Add a value to a [Variable](#).

Parameters

<i>time</i>	Time at which the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value added

Returns

0 if success
An error code otherwise

5.11.1.11 **trace_return_t** pajeAddVarType (const char * *alias*, const char * *contType*,
const char * *name*)

Add a [Variable](#) Type.

Parameters

<i>alias</i>	Alias on the variable type
<i>contType</i>	Type of container
<i>name</i>	Name of the variable type

Returns

0 if success
An error code otherwise

5.11.1.12 **trace_return_t** pajeDestroyContainer (varPrec *time*, const char * *name*,
const char * *type*)

Destroy a [Container](#).

Parameters

<i>time</i>	Time at which the container is destroyed
<i>name</i>	Name on the container to destroy
<i>type</i>	Type of the container

Returns

0 if success
An error code otherwise

5.11.1.13 `trace_return_t pajeEndLink (varPrec time, const char * type, const char * cont, const char * dest, const char * val, const char * key)`

Start a link.

Parameters

<i>time</i>	Time at which the link starts
<i>type</i>	Type of the link
<i>cont</i>	Container parent of the source and destination containers containing the link
<i>dest</i>	Source container
<i>val</i>	Value of the link
<i>key</i>	Key used to match start link with end link

Returns

0 if success
An error code otherwise

5.11.1.14 `pajeEndTrace ()`

Finalize a PAJE trace.

Returns

0 if success
An error code otherwise

5.11.1.15 `char * pajeGetName (int rk)`

Function to get the name of the file containing all the data for the proc of rank *rk*.

Parameters

<i>rk</i>	Rank of the proc you want the filename containing it
-----------	------------------------------------------------------

Returns

Name of the file.

5.11.1.16 `trace_return_t pajelnitTrace (const char * filename, int rank, gtg_flag_t flags, int fmt)`

Initialize a VITE trace (*.ept)

Parameters

<i>filename</i>	Root name of the file to create
<i>rank</i>	Rank of the processor
<i>flags</i>	One of GTG_FLAG_NONE, GTG_FLAG_USE_MPI, GTG_FLAG_NO-TBUF.
<i>fmt</i>	Format, paje or vite

Returns

0 if success
An error code otherwise

5.11.1.17 **trace_return_t pajePopState (varPrec *time*, const char * *type*, const char * *cont*)**

Revert the [State](#) of a [Container](#) to its previous value.

Parameters

<i>time</i>	Time at which the state is popped
<i>type</i>	Type of the state
<i>cont</i>	Container in this state

Returns

0 if success
An error code otherwise

5.11.1.18 **trace_return_t pajePushState (varPrec *time*, const char * *type*, const char * *cont*, const char * *val*)**

Save the current [State](#) on a stack and change the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time at which the state is pushed
<i>type</i>	Type of the state
<i>cont</i>	Container in this state
<i>val</i>	Entity value of the state of the container

Returns

0 if success
An error code otherwise

5.11.1.19 **trace_return_t** pajeSeqAddContainer (*varPrec time*, *const char * alias*,
*const char * type*, *const char * container*, *const char * name*)

Add a [Container](#) (PAJE format).

Parameters

<i>time</i>	Time at which the container is added
<i>alias</i>	Alias on the new container
<i>type</i>	Type of the container
<i>container</i>	Container parent
<i>name</i>	Name of the variable type

Returns

0 if success
 An error code otherwise

5.11.1.20 **trace_return_t** pajeSetCompress (*int val*)

Enable trace compression.

Parameters

<i>val</i>	0 means no compression, otherwise the output files will be compressed.
------------	------------------------------------------------------------------------

Returns

0 if success
 An error code otherwise

5.11.1.21 **trace_return_t** pajeSetState (*varPrec time*, *const char * type*, *const char * cont*, *const char * val*)

Set the [State](#) of a [Container](#).

Parameters

<i>time</i>	Time at which the state is set
<i>type</i>	Type of the state
<i>cont</i>	Container in this state
<i>val</i>	Entity value of the state of the container

Returns

0 if success
 An error code otherwise

5.11.1.22 `trace_return_t pajeSetVar (varPrec time, const char * type, const char * cont, varPrec val)`

Set a [Variable](#) value.

Parameters

<i>time</i>	Time at which the variable is set
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value of the variable

Returns

0 if success
An error code otherwise

5.11.1.23 `trace_return_t pajeStartLink (varPrec time, const char * type, const char * cont, const char * src, const char * val, const char * key)`

Start a link.

Parameters

<i>time</i>	Time at which the link starts
<i>type</i>	Type of the link
<i>cont</i>	Container parent of the source and destination containers containing the link
<i>src</i>	Source container
<i>val</i>	Value of the link
<i>key</i>	Key used to match start link with end link

Returns

0 if success
An error code otherwise

5.11.1.24 `trace_return_t pajeSubVar (varPrec time, const char * type, const char * cont, varPrec val)`

Subtract a value from a [Variable](#).

Parameters

<i>time</i>	Time at which the variable is incremented
<i>type</i>	Type of the variable
<i>cont</i>	Container containing the variable
<i>val</i>	Value subtracted

Returns

0 if success
An error code otherwise

5.11.1.25 viteEndTrace ()

Finalize a VITE trace.

Returns

0 if success
An error code otherwise

5.12 Functions for postponing event-processing function calls

Functions

- void `gtg_record` (enum `event_type_t` type, varPrec time,...)
postpone the recording of an event
- void `gtg_write_events` (long nb_events_to_write)
run the first nb_events_to_write events

5.12.1 Function Documentation

5.12.1.1 void `gtg_record` (enum `event_type_t` type, varPrec time, ...)

postpone the recording of an event

Parameters

<i>type</i>	The type of function to postpone
<i>time</i>	The time at which the event happens

5.12.1.2 void `gtg_write_events` (long nb_events_to_write)

run the first nb_events_to_write events

Parameters

<i>nb_events_ - to_write</i>	The number of functions to process (-1 for all functions)
----------------------------------	-----------------------------------------------------------

5.13 Types used

Typedefs

- typedef double `varPrec`
Use the double precision type for time and value.

Enumerations

- enum `trace_return_t` { `TRACE_SUCCESS`, `TRACE_ERR_OPEN`, `TRACE_ERR_CLOSE`, `TRACE_ERR_WRITE`, `TRACE_ERR_NOT_IMPL` }
Define various return values.

5.13.1 Typedef Documentation

5.13.1.1 typedef double `varPrec`

Use the double precision type for time and value.

5.13.2 Enumeration Type Documentation

5.13.2.1 enum `trace_return_t`

Define various return values.

Enumerator:

`TRACE_SUCCESS` Success of the call.
`TRACE_ERR_OPEN` Failed to open files to write.
`TRACE_ERR_CLOSE` Failed to close file.
`TRACE_ERR_WRITE` Failed to write trace.
`TRACE_ERR_NOT_IMPL` Function not impleneted.

Chapter 6

Data Structure Documentation

6.1 Container Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- char * [name](#)
- char * [alias](#)
- int [ctType](#)
- int [id](#)
- struct [gtg_list](#) token
- [State_t](#) [state_stack](#)

6.1.1 Detailed Description

Containers

6.1.2 Field Documentation

6.1.2.1 char* Container::alias

6.1.2.2 int Container::ctType

6.1.2.3 int Container::id

6.1.2.4 char* Container::name

6.1.2.5 State_t Container::state_stack

6.1.2.6 struct `gtg_list` `Container::token`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.2 ContainerType Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- char * [name](#)
- char * [alias](#)
- int [id](#)
- struct [gtg_list](#) [token](#)

6.2.1 Field Documentation

6.2.1.1 char* `ContainerType::alias`

6.2.1.2 int `ContainerType::id`

6.2.1.3 char* `ContainerType::name`

6.2.1.4 struct `gtg_list` `ContainerType::token`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.3 EntityValue Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- char * [name](#)
- char * [alias](#)
- int [groupId](#)
- int [id](#)
- struct [gtg_list](#) [token](#)

6.3.1 Detailed Description

[EntityValue](#), contains the name of the functions/states

6.3.2 Field Documentation

6.3.2.1 `char* EntityValue::alias`

6.3.2.2 `int EntityValue::groupId`

6.3.2.3 `int EntityValue::id`

6.3.2.4 `char* EntityValue::name`

6.3.2.5 `struct gtg_list EntityValue::token`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.4 EventType Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- `char * name`
- `char * alias`
- `int contType`
- `int id`
- `struct gtg_list token`

6.4.1 Detailed Description

Events/Markers

6.4.2 Field Documentation

6.4.2.1 `char* EventType::alias`

6.4.2.2 `int EventType::contType`

6.4.2.3 `int EventType::id`

6.4.2.4 `char* EventType::name`

6.4.2.5 `struct gtg_list EventType::token`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.5 `gtg_color` Struct Reference

This structure defines a color that can be used by GTG.

```
#include <GTGColor.h>
```

Data Fields

- `char * color_name`
- `gtg_rgb_color_t rgb`

6.5.1 Detailed Description

This structure defines a color that can be used by GTG.

6.5.2 Field Documentation

6.5.2.1 `char* gtg_color::color_name`

The name of the color (ie. "RED" or "Black",...)

6.5.2.2 `gtg_rgb_color_t gtg_color::rgb`

RGB code of the color. It should be obtained by calling `GTG_COLOR_SET_COLOR(r, g, b)`.

The documentation for this struct was generated from the following file:

- [GTGColor.h](#)

6.6 `gtg_list` Struct Reference

```
#include <GTGList.h>
```

Data Fields

- struct [gtg_list](#) * [prev](#)
- struct [gtg_list](#) * [next](#)

6.6.1 Field Documentation

6.6.1.1 struct [gtg_list](#)* [gtg_list::next](#)

6.6.1.2 struct [gtg_list](#)* [gtg_list::prev](#)

The documentation for this struct was generated from the following file:

- [GTGList.h](#)

6.7 gtg_memory Struct Reference

```
#include <GTGMemory.h>
```

Data Fields

- void * [first_mem](#)
- void * [current_mem](#)
- size_t [block_len](#)
- long [mem_len](#)
- void * [first_free](#)
- long [first_new](#)
- long [nb_allocated](#)

6.7.1 Field Documentation

6.7.1.1 size_t [gtg_memory::block_len](#)

6.7.1.2 void* [gtg_memory::current_mem](#)

6.7.1.3 void* [gtg_memory::first_free](#)

6.7.1.4 void* [gtg_memory::first_mem](#)

6.7.1.5 long [gtg_memory::first_new](#)

6.7.1.6 long [gtg_memory::mem_len](#)

6.7.1.7 long gtg_memory::nb_allocated

The documentation for this struct was generated from the following file:

- [GTGMemory.h](#)

6.8 Link Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- [varPrec time](#)
- [int src](#)

6.8.1 Field Documentation

6.8.1.1 int Link::src

6.8.1.2 varPrec Link::time

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.9 LinkType Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- [char * name](#)
- [char * alias](#)
- [int contType](#)
- [int srcType](#)
- [int destType](#)
- [int id](#)
- [struct gtg_list token](#)

6.9.1 Detailed Description

Links/Messages

6.9.2 Field Documentation

6.9.2.1 `char* LinkType::alias`

6.9.2.2 `int LinkType::contType`

6.9.2.3 `int LinkType::destType`

6.9.2.4 `int LinkType::id`

6.9.2.5 `char* LinkType::name`

6.9.2.6 `int LinkType::srcType`

6.9.2.7 `struct gtf_list LinkType::token`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.10 otf_color Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- `char * colorID`
- `uint8_t red`
- `uint8_t green`
- `uint8_t blue`

6.10.1 Field Documentation

6.10.1.1 `uint8_t otf_color::blue`

6.10.1.2 `char* otf_color::colorID`

6.10.1.3 `uint8_t otf_color::green`

6.10.1.4 `uint8_t otf_color::red`

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.11 State Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- int [value](#)
- int [cont](#)
- int [stateType](#)
- [gtg_stack](#) token

6.11.1 Detailed Description

States

6.11.2 Field Documentation

6.11.2.1 int State::cont

6.11.2.2 int State::stateType

6.11.2.3 [gtg_stack](#) State::token

6.11.2.4 int State::value

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.12 StateType Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- char * [name](#)
- char * [alias](#)
- int [groupId](#)
- int [id](#)
- struct [gtg_list](#) token

6.12.1 Detailed Description

StateTypes

6.12.2 Field Documentation

6.12.2.1 char* StateType::alias

6.12.2.2 int StateType::groupId

6.12.2.3 int StateType::id

6.12.2.4 char* StateType::name

6.12.2.5 struct gtg_list StateType::token

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.13 Variable Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- int [parent](#)
- int [type](#)
- uint64_t [value](#)
- int [id](#)
- struct [gtg_list](#) [token](#)

6.13.1 Field Documentation

6.13.1.1 int Variable::id

6.13.1.2 int Variable::parent

6.13.1.3 struct gtg_list Variable::token

6.13.1.4 int Variable::type

6.13.1.5 uint64_t Variable::value

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

6.14 VariableType Struct Reference

```
#include <GTGOTF_Structs.h>
```

Data Fields

- char * [name](#)
- char * [alias](#)
- int [contType](#)
- int [id](#)
- struct [gtg_list](#) token

6.14.1 Detailed Description

Variables/Counters

6.14.2 Field Documentation

6.14.2.1 char* VariableType::alias

6.14.2.2 int VariableType::contType

6.14.2.3 int VariableType::id

6.14.2.4 char* VariableType::name

6.14.2.5 struct gtg_list VariableType::token

The documentation for this struct was generated from the following file:

- [GTGOTF_Structs.h](#)

Chapter 7

File Documentation

7.1 GTG.h File Reference

Generic header to include.

```
#include <stdint.h> #include "GTGTypes.h" #include "GTG-Color.h" #include "GTGBasic.h"
```

7.1.1 Detailed Description

Generic header to include.

Authors

Developpers are :

Francois Rue - francois.rue@labri.fr

Francois Trahay - francois.trahay@labri.fr

Johnny Jazeix - jazeix@enseirb-matmeca.fr

Kevin Coulomb - kevin.coulomb@gmail.com

Mathieu Faverge - faverge@labri.fr

Olivier Lagrasse - lagrasse@enseirb-matmeca.fr

7.2 GTGBasic.h File Reference

GTGBasic is a basic interface to generate trace in various formats.

```
#include <stdlib.h> #include <string.h> #include "GTG-Color.h" #include "GTGTypes.h"
```

Defines

- #define [GTG_FLAG_NONE](#) 0

No flag specified.

- #define `GTG_FLAG_USE_MPI` 1

Several MPI processes are currently using GTG.

- #define `GTG_FLAG_NOTBUF` 2

For writing the traces in a non-buffered mode.

- #define `GTG_FLAG_OUTOFORDER` 4

Allow the application to record events out of order.

Typedefs

- typedef uint8_t `gtg_flag_t`

Flags that can be specified to GTG.

- typedef enum `traceType` `traceType_t`

Enumerations

- enum `traceType` { `PAJE`, `VITE`, `OTF`, `TAU` }

The type of the output trace.

Functions

- void `setTraceType` (`traceType_t` type)

Set the type of output trace.

- `traceType_t` `getTraceType` ()

Get the type of the output trace.

- char * `getName` (int procRk)

To get the name of the file to give to the addCont function for processors.

- int `bufferedModeActivated` ()

Check whether the buffered-mode is activated.

- `trace_return_t` `initTrace` (const char *filename, int rank, `gtg_flag_t` flags)

Initialize a trace.

- `trace_return_t` `endTrace` ()

Finalize a trace.

- `trace_return_t` `setCompress` (int val)

Enable trace compression (only available for OTF traces).

- `trace_return_t` `addContType` (const char *alias, const char *contType, const char *name)

*Add a *Container* Type.*

- `trace_return_t` `addStateType` (const char *alias, const char *contType, const char *name)

*Add a *State* Type.*

- `trace_return_t` `addEventType` (const char *alias, const char *contType, const char *name)

Add an Event Type.

- `trace_return_t addLinkType` (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)

Add a [Link](#) Type.

- `trace_return_t addVarType` (const char *alias, const char *name, const char *contType)

Add a [Variable](#) Type.

- `trace_return_t addEntityValue` (const char *alias, const char *entType, const char *name, `gtg_color_t` p_color)

Add an Entity Value.

- `trace_return_t addContainer` (`varPrec` time, const char *alias, const char *type, const char *container, const char *name, const char *file)

Add a [Container](#).

- `trace_return_t destroyContainer` (`varPrec` time, const char *name, const char *type)

Destroy a [Container](#).

- `trace_return_t setState` (`varPrec` time, const char *type, const char *cont, const char *val)

Set the [State](#) of a [Container](#).

- `trace_return_t pushState` (`varPrec` time, const char *type, const char *cont, const char *val)

Save the current [State](#) on a stack and change the [State](#) of a [Container](#).

- `trace_return_t popState` (`varPrec` time, const char *type, const char *cont)

Revert the [State](#) of a [Container](#) to its previous value.

- `trace_return_t addEvent` (`varPrec` time, const char *type, const char *cont, const char *val)

Add an Event.

- `trace_return_t startLink` (`varPrec` time, const char *type, const char *cont, const char *src, const char *dest, const char *val, const char *key)

Start a [Link](#).

- `trace_return_t endLink` (`varPrec` time, const char *type, const char *cont, const char *src, const char *dest, const char *val, const char *key)

End a [Link](#).

- `trace_return_t setVar` (`varPrec` time, const char *type, const char *cont, `varPrec` val)

Set a [Variable](#) value.

- `trace_return_t addVar` (`varPrec` time, const char *type, const char *cont, `varPrec` val)

Add a value to a [Variable](#).

- `trace_return_t subVar` (`varPrec` time, const char *type, const char *cont, `varPrec` val)

Subtract a value from a [Variable](#).

- `trace_return_t AddComment` (const char *comment)

Add some Comment in Trace file.

7.2.1 Detailed Description

GTGBasic is a basic interface to generate trace in various formats.

Version

0.1

Authors

Developers are :

Francois Rue - francois.rue@labri.fr

Francois Trahay - francois.trahay@labri.fr

Johnny Jazeix - jazeix@enseirb-matmeca.fr

Kevin Coulomb - kevin.coulomb@gmail.com

Mathieu Faverge - faverge@labri.fr

Olivier Lagrasse - lagrasse@enseirb-matmeca.fr

It has been initiated in 2010 by *eztrace* and *ViTE* projects that both needs a good library to generate traces.

7.2.2 Define Documentation

7.2.2.1 #define GTG_FLAG_NONE 0

No flag specified.

7.2.2.2 #define GTG_FLAG_NOTBUF 2

For writing the traces in a non-buffered mode.

7.2.2.3 #define GTG_FLAG_OUTOFORDER 4

Allow the application to record events out of order.

7.2.2.4 #define GTG_FLAG_USE_MPI 1

Several MPI processes are currently using GTG.

7.2.3 Typedef Documentation

7.2.3.1 typedef uint8_t gtg_flag_t

Flags that can be specified to GTG.

7.2.3.2 typedef enum traceType traceType_t

7.3 GTGColor.h File Reference

This file defines some useful colors to use in entity values for GTG.

```
#include <stdint.h>
```

Data Structures

- struct [gtg_color](#)

This structure defines a color that can be used by GTG.

Defines

- #define [GTG_COLOR_BLUE_POS](#) 0
- #define [GTG_COLOR_GREEN_POS](#) 8
- #define [GTG_COLOR_RED_POS](#) 16
- #define [GTG_COLOR_BLUE_MASK](#) (0x000000ff << GTG_COLOR_BLUE_POS)
- #define [GTG_COLOR_GREEN_MASK](#) (0x000000ff << GTG_COLOR_GREEN_POS)
- #define [GTG_COLOR_RED_MASK](#) (0x000000ff << GTG_COLOR_RED_POS)

Typedefs

- typedef uint32_t [gtg_rgb_color_t](#)
- typedef struct [gtg_color](#) * [gtg_color_t](#)

Functions

- static uint8_t [GTG_COLOR_GET_BLUE](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the blue component of a rgb color.
- static uint8_t [GTG_COLOR_GET_GREEN](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the green component of a rgb color.
- static uint8_t [GTG_COLOR_GET_RED](#) ([gtg_rgb_color_t](#) rgb)
Return the 1-byte value of the red component of a rgb color.
- static [gtg_rgb_color_t](#) [GTG_COLOR_SET_COLOR](#) (uint8_t r, uint8_t g, uint8_t b)
Return the 4-bytes RGB color from 3 1-byte components.
- void [gtg_color_init](#) ()
- void [gtg_color_exit](#) ()
- [gtg_color_t](#) [gtg_color_create](#) (const char *name, uint8_t r, uint8_t g, uint8_t b)
- void [gtg_color_free](#) ([gtg_color_t](#) color)

Variables

- `gtg_color_t GTG_BLACK`
Default black color. $(R,G,B) = (0, 0, 0)$
- `gtg_color_t GTG_RED`
Default red color. $(R,G,B) = (255, 0, 0)$
- `gtg_color_t GTG_GREEN`
Default green color. $(R,G,B) = (0, 255, 0)$
- `gtg_color_t GTG_BLUE`
Default blue color. $(R,G,B) = (0, 0, 255)$
- `gtg_color_t GTG_WHITE`
Default white color. $(R,G,B) = (255, 255, 255)$
- `gtg_color_t GTG_TEAL`
Default teal color. $(R,G,B) = (0, 255, 255)$
- `gtg_color_t GTG_DARKGREY`
Default dark grey color. $(R,G,B) = (85, 85, 85)$
- `gtg_color_t GTG_YELLOW`
Default yellow color. $(R,G,B) = (255, 255, 0)$
- `gtg_color_t GTG_PURPLE`
Default purple color. $(R,G,B) = (153, 25, 230)$
- `gtg_color_t GTG_LIGHTBROWN`
Default light brown color. $(R,G,B) = (170, 130, 130)$
- `gtg_color_t GTG_LIGHTGREY`
Default light grey color. $(R,G,B) = (200, 200, 200)$
- `gtg_color_t GTG_DARKBLUE`
Default dark blue color. $(R,G,B) = (0, 0, 80)$
- `gtg_color_t GTG_PINK`
Default pink color. $(R,G,B) = (255, 0, 255)$
- `gtg_color_t GTG_DARKPINK`
Default dark pink color. $(R,G,B) = (180, 80, 180)$
- `gtg_color_t GTG_SEABLUE`
Default sea blue color. $(R,G,B) = (25, 128, 200)$
- `gtg_color_t GTG_KAKI`
Default kaki color. $(R,G,B) = (80, 100, 25)$
- `gtg_color_t GTG_REDBLOOD`
Default red blood color. $(R,G,B) = (200, 25, 25)$
- `gtg_color_t GTG_BROWN`
Default brown color. $(R,G,B) = (100, 25, 25)$
- `gtg_color_t GTG_GRENAT`
Default grenat color. $(R,G,B) = (100, 0, 80)$
- `gtg_color_t GTG_ORANGE`
Default orange color. $(R,G,B) = (255, 160, 0)$
- `gtg_color_t GTG_MAUVE`
Default mauve color. $(R,G,B) = (128, 0, 255)$
- `gtg_color_t GTG_LIGHTPINK`
Default light pink color. $(R,G,B) = (255, 128, 255)$

7.3.1 Detailed Description

This file defines some useful colors to use in entity values for GTG.

Version

0.1

7.3.2 Define Documentation

7.3.2.1 `#define GTG_COLOR_BLUE_MASK (0x000000ff << GTG_COLOR_BLUE_POS)`

7.3.2.2 `#define GTG_COLOR_BLUE_POS 0`

7.3.2.3 `#define GTG_COLOR_GREEN_MASK (0x000000ff << GTG_COLOR_GREEN_POS)`

7.3.2.4 `#define GTG_COLOR_GREEN_POS 8`

7.3.2.5 `#define GTG_COLOR_RED_MASK (0x000000ff << GTG_COLOR_RED_POS)`

7.3.2.6 `#define GTG_COLOR_RED_POS 16`

7.3.3 Typedef Documentation

7.3.3.1 `typedef struct gtg_color* gtg_color_t`

7.3.3.2 `typedef uint32_t gtg_rgb_color_t`

7.3.4 Function Documentation

7.3.4.1 `gtg_color_t gtg_color_create (const char * name, uint8_t r, uint8_t g, uint8_t b)`

7.3.4.2 `void gtg_color_exit ()`

7.3.4.3 `void gtg_color_free (gtg_color_t color)`

7.3.4.4 `void gtg_color_init ()`

7.4 GTGCompress.h File Reference

```
#include <stdint.h> #include <stdio.h> #include <zlib.h>
```

Functions

- int [gtg_compress_m2m](#) (z_stream *z, void *in_buf, uint32_t len, void *out_buf, uint32_t out_max_len)

- int [gtg_compress_m2f](#) (z_stream *z, void *in_buf, uint32_t len, FILE *file_out)
- int [gtg_compress_f2m](#) (z_stream *z, FILE *file_in, void *out_buf, uint32_t out_max_len)
- int [gtg_compress_f2f](#) (z_stream *z, FILE *file_in, FILE *file_out)
- int [gtg_decompress_m2m](#) (z_stream *z, void *in_buf, uint32_t len, void *out_buf, uint32_t out_max_len)
- int [gtg_decompress_m2f](#) (z_stream *z, void *in_buf, uint32_t len, FILE *file_out)
- int [gtg_decompress_f2m](#) (z_stream *z, FILE *file_in, void *out_buf, uint32_t out_max_len)
- int [gtg_decompress_f2f](#) (z_stream *z, FILE *file_in, FILE *file_out)
- int [gtg_compress_init](#) (z_stream *z, int compression_ratio)
- int [gtg_decompress_init](#) (z_stream *z)

7.4.1 Function Documentation

7.4.1.1 int [gtg_compress_f2f](#) (z_stream * z, FILE * *file_in*, FILE * *file_out*)

7.4.1.2 int [gtg_compress_f2m](#) (z_stream * z, FILE * *file_in*, void * *out_buf*, uint32_t *out_max_len*)

7.4.1.3 int [gtg_compress_init](#) (z_stream * z, int *compression_ratio*)

7.4.1.4 int [gtg_compress_m2f](#) (z_stream * z, void * *in_buf*, uint32_t *len*, FILE * *file_out*)

7.4.1.5 int [gtg_compress_m2m](#) (z_stream * z, void * *in_buf*, uint32_t *len*, void * *out_buf*, uint32_t *out_max_len*)

7.4.1.6 int [gtg_decompress_f2f](#) (z_stream * z, FILE * *file_in*, FILE * *file_out*)

7.4.1.7 int [gtg_decompress_f2m](#) (z_stream * z, FILE * *file_in*, void * *out_buf*, uint32_t *out_max_len*)

7.4.1.8 int [gtg_decompress_init](#) (z_stream * z)

7.4.1.9 int [gtg_decompress_m2f](#) (z_stream * z, void * *in_buf*, uint32_t *len*, FILE * *file_out*)

7.4.1.10 int [gtg_decompress_m2m](#) (z_stream * z, void * *in_buf*, uint32_t *len*, void * *out_buf*, uint32_t *out_max_len*)

7.5 GTGList.h File Reference

Data Structures

- struct [gtg_list](#)

Defines

- #define [GTG_LIST_INIT](#)(ptr)
initialize a list.
- #define [GTG_LIST](#)(name)
declare and initialize a list.
- #define [gtg_list_entry](#)(ptr, type, member) ((type *)((char *)(ptr) - (char *)&((type *)0)->member)))
get the structure corresponding to a list entry
- #define [gtg_list_for_each](#)(pos, head) for (pos = (head)->next; pos != (head); pos = pos->next)
- #define [gtg_list_for_each_reverse](#)(pos, head) for (pos = (head)->prev; pos != (head); pos = pos->prev)
- #define [gtg_list_for_each_safe](#)(pos, n, head)
- #define [gtg_list_for_each_entry](#)(pos, head, member)
iterate over list of given type
- #define [gtg_list_for_each_entry_safe](#)(pos, n, head, member)
iterate over list of given type safe against removal of list entry

Typedefs

- typedef struct [gtg_list](#) * [gtg_list_t](#)

Functions

- static void [__gtg_list_add](#) ([gtg_list_t](#) lnew, [gtg_list_t](#) prev, [gtg_list_t](#) next)
- static void [gtg_list_add](#) ([gtg_list_t](#) lnew, [gtg_list_t](#) head)
Insert a new entry after the specified head.
- static void [gtg_list_add_tail](#) ([gtg_list_t](#) lnew, [gtg_list_t](#) head)
Insert a new entry before the specified head (ie. at the tail of the list).
- static void [__gtg_list_del](#) ([gtg_list_t](#) prev, [gtg_list_t](#) next)
- static void [gtg_list_del](#) ([gtg_list_t](#) entry)
delete an entry from its list and reinitialize it.
- static int [gtg_list_size](#) ([gtg_list_t](#) l)

7.5.1 Define Documentation

7.5.1.1 GTG_LIST(*name*)

Value:

```
struct gtg_list name; \
    GTG_LIST_INIT(&name)
```

declare and initialize a list.

Parameters

<i>name</i>	Name of the variable
-------------	----------------------

7.5.1.2 **gtg_list_entry**(*ptr*, *type*, *member*) ((type *)((char *)(ptr) - (char *)&((type *)0)->member)))

get the structure corresponding to a list entry

Parameters

<i>ptr</i>	pointer to the list entry (gtg_list_t)
<i>type</i>	the type of the struct this is embedded in.
<i>member</i>	the name of the struct gtg_list member within the struct.

7.5.1.3 **#define gtg_list_for_each**(*pos*, *head*) for (pos = (head)->next; pos != (head); pos = pos->next)

7.5.1.4 **#define gtg_list_for_each_entry**(*pos*, *head*, *member*)

Value:

```
for (pos = gtg_list_entry((head)->next, sizeof(*pos), member); \
    &pos->member != (head); \
    pos = gtg_list_entry(pos->member.next, sizeof(*pos), member))
```

iterate over list of given type

[gtg_list_for_each_entry\(pos, head, member\)](#)

Parameters

<i>pos</i>	the type * to use as a loop counter.
<i>head</i>	the head for the list.
<i>member</i>	the name of the struct gtg_list member within the struct.

7.5.1.5 **#define gtg_list_for_each_entry_safe**(*pos*, *n*, *head*, *member*)

Value:

```
for (pos = gtg_list_entry((head)->next, sizeof(*pos), member), \
    n = gtg_list_entry(pos->member.next, sizeof(*pos), member); \
    &pos->member != (head); \
    pos = n, n = gtg_list_entry(n->member.next, sizeof(*n), member))
```

iterate over list of given type safe against removal of list entry

[gtg_list_for_each_entry_safe\(pos, n, head, member\)](#)

Parameters

<i>pos</i>	the type * to use as a loop counter.
<i>n</i>	another type * to use as temporary storage
<i>head</i>	the head for the list.
<i>member</i>	the name of the struct gtg_list member within the struct.

7.5.1.6 **#define** `gtg_list_for_each_reverse(pos, head)` for (`pos = (head)->prev; pos != (head); pos = pos->prev`)

7.5.1.7 **#define** `gtg_list_for_each_safe(pos, n, head)`

Value:

```
for (pos = (head)->next, n = pos->next; pos != (head); \
    pos = n, n = pos->next)
```

7.5.1.8 **GTG_LIST_INIT(ptr)**

Value:

```
do {
    (ptr)->prev = (ptr);
    (ptr)->next = (ptr);
} while(0)
```

initialize a list.

Parameters

<i>ptr</i>	pointer to the list (<code>gtg_list_t</code>).
------------	--------------------------------------------------

7.5.2 Typedef Documentation

7.5.2.1 **typedef** struct `gtg_list*` `gtg_list_t`

7.5.3 Function Documentation

7.5.3.1 **static void** `__gtg_list_add (gtg_list_t lnew, gtg_list_t prev, gtg_list_t next)` `[inline, static]`

7.5.3.2 **static void** `__gtg_list_del (gtg_list_t prev, gtg_list_t next)` `[inline, static]`

Delete a list entry by making the prev/next entries point to each other.

This is only for internal list manipulation where we know the prev/next entries already!

7.5.3.3 `void gtg_list_add (gtg_list_t lnew, gtg_list_t head)` `[inline, static]`

Insert a new entry after the specified head.

Parameters

<i>lnew</i>	new entry to be added
<i>head</i>	list head to add it after

7.5.3.4 `void gtg_list_add_tail (gtg_list_t lnew, gtg_list_t head)` `[inline, static]`

Insert a new entry before the specified head (ie. at the tail of the list).

Parameters

<i>lnew</i>	new entry to be added
<i>head</i>	list head to add it after

7.5.3.5 `void gtg_list_del (gtg_list_t entry)` `[inline, static]`

delete an entry from its list and reinitialize it.

Parameters

<i>entry</i>	the element to delete from the list.
--------------	--------------------------------------

7.5.3.6 `static int gtg_list_size (gtg_list_t l)` `[inline, static]`

7.6 GTGMemory.h File Reference

This file defines a fast allocator for fixed-size blocks.

```
#include <stdlib.h>
```

Data Structures

- struct [gtg_memory](#)

Typedefs

- typedef struct [gtg_memory](#) * [gtg_memory_t](#)

Functions

- void [gtg_block_memory_init](#) ([gtg_memory_t](#) *memory, size_t block_size, long initial_block_number)
Initialize the allocator.
- void * [gtg_block_malloc](#) ([gtg_memory_t](#) memory)
Allocate a block of data.
- void [gtg_block_free](#) ([gtg_memory_t](#) memory, void *ptr)
Free a block of data.

7.6.1 Detailed Description

This file defines a fast allocator for fixed-size blocks.

Version

0.1

7.6.2 Typedef Documentation

7.6.2.1 typedef struct [gtg_memory](#)* [gtg_memory_t](#)

7.7 GTGOTF.h File Reference

OTF is the global file for gtg interface using OTF.

```
#include <stdint.h> #include "GTGOTF_Structs.h" #include  
"GTGOTF_Basic.h"
```

7.7.1 Detailed Description

OTF is the global file for gtg interface using OTF.

Version

0.1

Authors

Developers are :
Francois Rue - francois.rue@labri.fr
Francois Trahay - francois.trahay@labri.fr
Johnny Jazeix - jazeix@enseirb-matmeca.fr
Kevin Coulomb - kevin.coulomb@gmail.com
Mathieu Faverge - faverge@labri.fr
Olivier Lagrasse - lagrasse@enseirb-matmeca.fr

7.8 GTGOTF_Basic.h File Reference

OTF_GTGBasic1 is the OTF implementation of the basic interface to generate traces (GTGBasic1).

```
#include "GTGTypes.h" #include "GTGBasic.h" #include "GTG-OTF_Structs.h"
```

Functions

- `const otf_color_t OTF_get_color (gtg_color_t color)`
Converts a GTG color into a OTF color.
- `trace_return_t OTFInitTrace (const char *filename, gtg_flag_t flags)`
Initialize an OTF trace.
- `trace_return_t OTFSetCompress (int val)`
Enable trace compression.
- `trace_return_t OTFAddContType (const char *alias, const char *contType, const char *name)`
Add a [Container](#) Type.
- `trace_return_t OTFAddStateType (const char *alias, const char *contType, const char *name)`
Add a [State](#) Type.
- `trace_return_t OTFAddEventType (const char *alias, const char *contType, const char *name)`
Add an [Event](#) Type.
- `trace_return_t OTFAddLinkType (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)`
Add a [Link](#) Type.
- `trace_return_t OTFAddVarType (const char *alias, const char *name, const char *contType)`
Add a [Variable](#) Type.
- `trace_return_t OTFAddEntityValue (const char *alias, const char *entType, const char *name, const otf_color_t color)`
Add an [Entity](#) Value.
- `trace_return_t OTFDefineContainer (const char *alias, const char *type, const char *container, const char *name, const char *file)`
- `trace_return_t OTFStartContainer (varPrec time, const char *alias, const char *type, const char *container, const char *name, const char *file)`
Start a [Container](#).
- `trace_return_t OTFDestroyContainer (varPrec time, const char *name, const char *type)`
Destroy a [Container](#).
- `trace_return_t OTFSetState (varPrec time, const char *type, const char *cont, const char *val)`
Set the [State](#) of a [Container](#).

- `trace_return_t OTFPushState` (`varPrec` time, `const char *type`, `const char *cont`, `const char *val`)
Save the current [State](#) on a stack and change the [State](#) of a [Container](#).
- `trace_return_t OTFPopState` (`varPrec` time, `const char *type`, `const char *cont`)
Revert the [State](#) of a [Container](#) to its previous value.
- `trace_return_t OTFAddEvent` (`varPrec` time, `const char *type`, `const char *cont`, `const char *val`)
Add an Event.
- `trace_return_t OTFStartLink` (`varPrec` time, `const char *type`, `const char *src`, `const char *dest`, `const char *val`, `const char *key`)
Start a [Link](#).
- `trace_return_t OTFEndLink` (`varPrec` time, `const char *type`, `const char *src`, `const char *dest`, `const char *val`, `const char *key`)
End a [Link](#).
- `trace_return_t OTFSetVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)
Set a [Variable](#) value.
- `trace_return_t OTFAddVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)
Add a value to a [Variable](#).
- `trace_return_t OTFSubVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)
Subtract a value from a [Variable](#).
- `trace_return_t OTFAddComment` (`const char *comment`)
Add some Comment in Trace file.
- `trace_return_t OTFEndTrace` ()
Finalize an OTF trace.

7.8.1 Detailed Description

OTF_GTGBasic1 is the OTF implementation of the basic interface to generate traces (GTGBasic1).

Version

0.1

Authors

Developers are :

Francois Rue - francois.rue@labri.fr

Francois Trahay - francois.trahay@labri.fr

Johnny Jazeix - jazeix@enseirb-matmeca.fr

Kevin Coulomb - kevin.coulomb@gmail.com

Mathieu Faverge - faverge@labri.fr

Olivier Lagrasse - lagrasse@enseirb-matmeca.fr

7.8.2 Function Documentation

7.8.2.1 `trace_return_t OTFDefineContainer (const char * alias, const char * type,
const char * container, const char * name, const char * file)`

7.9 GTGOTF_Structs.h File Reference

OTF_Structs gives the global types and functions needed to have the OTF implementation.

```
#include <stdint.h>  #include "GTGList.h"  #include "GTG-  
Stack.h"
```

Data Structures

- struct [StateType](#)
- struct [State](#)
- struct [ContainerType](#)
- struct [Container](#)
- struct [EntityValue](#)
- struct [EventType](#)
- struct [LinkType](#)
- struct [Link](#)
- struct [VariableType](#)
- struct [Variable](#)
- struct [otf_color](#)

Defines

- #define [MAX_PROCESS](#) 64
- #define [ContainerType_NIL](#) 0
- #define [Container_NIL](#) 0
- #define [StateType_NIL](#) 0
- #define [State_NIL](#) 0
- #define [EntityValue_NIL](#) 0
- #define [EventType_NIL](#) 0
- #define [LinkType_NIL](#) 0
- #define [VariableType_NIL](#) 0
- #define [Variable_NIL](#) 0
- #define [init_ContainerType](#)(var)
- #define [init_Container](#)(var)
- #define [init_StateType](#)(var)
- #define [init_EntityValue](#)(var)
- #define [init_EventType](#)(var)
- #define [init_LinkType](#)(var)
- #define [init_VariableType](#)(var)

- `#define init_Variable(var)`
- `#define init_State(var)`
- `#define alloc_struct(ptr, type, list_head)`
- `#define alloc_init_struct(type, ptr, list_head, _name_, _alias_)`
- `#define alloc_Variable(_ptr_, _id_, _parent_, _type_, _value_)`
- `#define alloc_State(_ptr_, _value_, _cont_, _stateType_)`
- `#define free_struct(_type_, _list_head_)`

Typedefs

- `typedef struct StateType StateType_t`
- `typedef struct State State_t`
- `typedef struct ContainerType ContainerType_t`
- `typedef struct Container Container_t`
- `typedef struct EntityValue EntityValue_t`
- `typedef struct EventType EventType_t`
- `typedef struct LinkType LinkType_t`
- `typedef struct Link Link_t`
- `typedef struct VariableType VariableType_t`
- `typedef struct Variable Variable_t`
- `typedef struct otf_color * otf_color_t`

7.9.1 Detailed Description

OTF_Structs gives the global types and functions needed to have the OTF implementation.

Version

0.1

Authors

Developers are :

Francois Rue - francois.rue@labri.fr

Francois Trahay - francois.trahay@labri.fr

Johnny Jazeix - jazeix@enseirb-matmeca.fr

Kevin Coulomb - kevin.coulomb@gmail.com

Mathieu Faverge - faverge@labri.fr

Olivier Lagrasse - lagrasse@enseirb-matmeca.fr

7.9.2 Define Documentation

7.9.2.1 `#define alloc_init_struct(type, ptr, list_head, _name_, _alias_)`

Value:

```

do {
    alloc_struct(ptr, type, list_head);
    (ptr)->name = (char *)malloc(sizeof(char)*(strlen(_name_)+1));
    strcpy((ptr)->name, _name_);
    (ptr)->alias = (char *)malloc(sizeof(char)*(strlen(_alias_)+1));
    strcpy((ptr)->alias, _alias_);
}while(0)

```

7.9.2.2 #define alloc_State(_ptr_, _value_, _cont_, _stateType_)

Value:

```

do {
    _ptr_ = (State_t*) malloc(sizeof(State_t));
    init_State(*(_ptr_));
    (_ptr_)->value = _value_;
    (_ptr_)->cont = _cont_;
    (_ptr_)->stateType = _stateType_;
}while(0)

```

7.9.2.3 #define alloc_struct(ptr, type, list_head)

Value:

```

do {
    ptr = (type*) malloc(sizeof(type));
    GTG_LIST_INIT(&(ptr->token));
    ptr->id = (gtg_list_entry((list_head)->prev, type, token)->id) + 1;
    gtg_list_add_tail(&(ptr->token), list_head);
} while(0)

```

7.9.2.4 #define alloc_Variable(_ptr_, _id_, _parent_, _type_, _value_)

Value:

```

do {
    (_ptr_) = (Variable_t*) malloc(sizeof(Variable_t));
    init_Variable(*(_ptr_));
    (_ptr_)->id = _id_;
    (_ptr_)->parent = _parent_;
    (_ptr_)->type = _type_;
    (_ptr_)->value = _value_;
}while(0)

```

7.9.2.5 #define Container_NIL 0

7.9.2.6 #define ContainerType_NIL 0

7.9.2.7 #define EntityValue_NIL 0

7.9.2.8 #define EventType_NIL 0**7.9.2.9 #define free_struct(_type_, _list_head_)****Value:**

```

do{ \
    _type_ *ptr, *tmp; \
    gtg_list_for_each_entry_safe(ptr, tmp, &(_list_head_.token, token) { \
        \
        gtg_list_del(&(ptr->token)); \
        free(ptr->name); \
        free(ptr->alias); \
        free(ptr); \
    } \
}while(0)

```

7.9.2.10 #define init_Container(var)**Value:**

```

do { \
    (var).name = NULL; \
    (var).alias = NULL; \
    (var).ctType = ContainerType_NIL; \
    (var).id = Container_NIL; \
    GTG_LIST_INIT(&(var).token); \
    GTG_STACK_INIT(&(var).state_stack.token); \
}while(0)

```

7.9.2.11 #define init_ContainerType(var)**Value:**

```

do { \
    (var).name = NULL; \
    (var).alias = NULL; \
    (var).id = ContainerType_NIL; \
    GTG_LIST_INIT(&(var).token); \
}while(0)

```

7.9.2.12 #define init_EntityValue(var)**Value:**

```

do { \
    (var).name = NULL; \
    (var).alias = NULL; \
    (var).groupId = 0; \
    (var).id = EntityValue_NIL; \
    GTG_LIST_INIT(&(var).token); \
}while(0)

```

7.9.2.13 #define init_EventType(var)**Value:**

```

do {
    (var).name      = NULL;
    (var).alias     = NULL;
    (var).contType  = ContainerType_NIL;
    (var).id        = EventType_NIL;
    GTG_LIST_INIT(&(var).token);
}while(0)

```

7.9.2.14 #define init_LinkType(var)**Value:**

```

do {
    (var).name      = NULL;
    (var).alias     = NULL;
    (var).contType  = ContainerType_NIL;
    (var).srcType   = ContainerType_NIL;
    (var).destType  = ContainerType_NIL;
    (var).id        = LinkType_NIL;
    GTG_LIST_INIT(&(var).token);
}while(0)

```

7.9.2.15 #define init_State(var)**Value:**

```

do {
    (var).value     = EntityValue_NIL;
    (var).cont      = Container_NIL;
    (var).stateType = StateType_NIL;
    GTG_STACK_INIT(&(var).token);
}while(0)

```

7.9.2.16 #define init_StateType(var)**Value:**

```

do {
    (var).name      = NULL;
    (var).alias     = NULL;
    (var).groupId   = 0;
    (var).id        = StateType_NIL;
    GTG_LIST_INIT(&(var).token);
}while(0)

```

7.9.2.17 #define init_Variable(var)**Value:**

```

do {
    (var).parent = Container_NIL;
    (var).parent = VariableType_NIL;
    (var).value = 0;
    (var).id = Variable_NIL;
    GTG_LIST_INIT(&(var).token);
}while(0)

```

7.9.2.18 #define init_VariableType(var)**Value:**

```

do {
    (var).name = NULL;
    (var).alias = NULL;
    (var).contType = ContainerType_NIL;
    (var).id = VariableType_NIL;
    GTG_LIST_INIT(&(var).token);
}while(0)

```

7.9.2.19 #define LinkType_NIL 0**7.9.2.20 #define MAX_PROCESS 64****7.9.2.21 #define State_NIL 0****7.9.2.22 #define StateType_NIL 0****7.9.2.23 #define Variable_NIL 0****7.9.2.24 #define VariableType_NIL 0****7.9.3 Typedef Documentation****7.9.3.1 typedef struct Container Container_t**

Containers

7.9.3.2 typedef struct ContainerType ContainerType_t**7.9.3.3 typedef struct EntityValue EntityValue_t**[EntityValue](#), contains the name of the functions/states

7.9.3.4 typedef struct EventType EventType_t

Events/Markers

7.9.3.5 typedef struct Link Link_t

7.9.3.6 typedef struct LinkType LinkType_t

Links/Messages

7.9.3.7 typedef struct otf_color* otf_color_t

7.9.3.8 typedef struct State State_t

States

7.9.3.9 typedef struct StateType StateType_t

StateTypes

7.9.3.10 typedef struct Variable Variable_t

7.9.3.11 typedef struct VariableType VariableType_t

Variables/Counters

7.10 GTGPaje.h File Reference

pajeColor is a file that defines function that manipulate colors.

```
#include "GTGPaje_Basic.h"
```

Typedefs

- typedef char * [paje_color_t](#)

Functions

- const [paje_color_t](#) [Paje_get_color](#) ([gtg_color_t](#) p_color)
Converts a GTG color into a PAJE color.

7.10.1 Detailed Description

pajeColor is a file that defines function that manipulate colors.

Version

0.1

7.10.2 Typedef Documentation

7.10.2.1 `typedef char* paje_color_t`

7.11 GTGPaje_Basic.h File Reference

paje_GTGBasic1 is the Paje implementation of the basic interface to generate traces (GTGBasic1).

```
#include "GTGBasic.h"
```

Defines

- `#define FMT_PAJE 0`
Constant to create a paje trace.
- `#define FMT_VITE 1`
Constant to create a vite trace.

Enumerations

- enum `gtg_paje_evtdef_e` { `GTG_PAJE_EVTDEF_DefineContainerType`, `GTG_PAJE_EVTDEF_DefineStateType`, `GTG_PAJE_EVTDEF_DefineEventType`, `GTG_PAJE_EVTDEF_DefineEntityValue`, `GTG_PAJE_EVTDEF_CreateContainer`, `GTG_PAJE_EVTDEF_DestroyContainer`, `GTG_PAJE_EVTDEF_SetState`, `GTG_PAJE_EVTDEF_PushState`, `GTG_PAJE_EVTDEF_PopState`, `GTG_PAJE_EVTDEF_NewEvent`, `GTG_PAJE_EVTDEF_DefineLinkType`, `GTG_PAJE_EVTDEF_StartLink`, `GTG_PAJE_EVTDEF_EndLink`, `GTG_PAJE_EVTDEF_DefineVariableType`, `GTG_PAJE_EVTDEF_SetVariable`, `GTG_PAJE_EVTDEF_AddVariable`, `GTG_PAJE_EVTDEF_SubVariable`, `GTG_PAJE_EVTDEF_NBR` }
- enum `gtg_paje_fieldtype_e` { `GTG_PAJE_FIELDTYPE_Int`, `GTG_PAJE_FIELDTYPE_Hex`, `GTG_PAJE_FIELDTYPE_Date`, `GTG_PAJE_FIELDTYPE_Double`, `GTG_PAJE_FIELDTYPE_String`, `GTG_PAJE_FIELDTYPE_Color`, `GTG_PAJE_FIELDTYPE_NBR` }

Functions

- `trace_return_t pajeInitTrace` (const char *filename, int rank, `gtg_flag_t` flags, int fmt)

*Initialize a VITE trace (*.ept)*

- char * [pajeGetName](#) (int rk)

Function to get the name of the file containing all the data for the proc of rank rk.

- [trace_return_t pajeSetCompress](#) (int val)

Enable trace compression.

- [trace_return_t pajeAddContType](#) (const char *alias, const char *contType, const char *name)

Add a [Container](#) Type.

- [trace_return_t pajeAddStateType](#) (const char *alias, const char *contType, const char *name)

Add a [State](#) Type.

- [trace_return_t pajeAddEventType](#) (const char *alias, const char *contType, const char *name)

Add an Event Type.

- [trace_return_t pajeAddLinkType](#) (const char *alias, const char *name, const char *contType, const char *srcContType, const char *destContType)

Add a [Link](#) Type.

- [trace_return_t pajeAddVarType](#) (const char *alias, const char *name, const char *contType)

Add a [Variable](#) Type.

- [trace_return_t pajeAddEntityValue](#) (const char *alias, const char *entType, const char *name, const char *color)

Add an Entity Value.

- [trace_return_t pajeAddContainer](#) (varPrec time, const char *alias, const char *type, const char *container, const char *name, const char *file)

Add a [Container](#) (VITE format).

- [trace_return_t pajeSeqAddContainer](#) (varPrec time, const char *alias, const char *type, const char *container, const char *name)

Add a [Container](#) (PAJE format).

- [trace_return_t pajeDestroyContainer](#) (varPrec time, const char *name, const char *type)

Destroy a [Container](#).

- [trace_return_t pajeSetState](#) (varPrec time, const char *type, const char *cont, const char *val)

Set the [State](#) of a [Container](#).

- [trace_return_t pajePushState](#) (varPrec time, const char *type, const char *cont, const char *val)

Save the current [State](#) on a stack and change the [State](#) of a [Container](#).

- [trace_return_t pajePopState](#) (varPrec time, const char *type, const char *cont)

Revert the [State](#) of a [Container](#) to its previous value.

- [trace_return_t pajeAddEvent](#) (varPrec time, const char *type, const char *cont, const char *val)

Add an Event.

- [trace_return_t pajeStartLink](#) (varPrec time, const char *type, const char *cont, const char *src, const char *val, const char *key)

Start a link.

- `trace_return_t pajeEndLink` (`varPrec` time, `const char *type`, `const char *cont`, `const char *dest`, `const char *val`, `const char *key`)

Start a link.

- `trace_return_t pajeSetVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)

Set a `Variable` value.

- `trace_return_t pajeAddVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)

Add a value to a `Variable`.

- `trace_return_t pajeSubVar` (`varPrec` time, `const char *type`, `const char *cont`, `varPrec` val)

Subtract a value from a `Variable`.

- `trace_return_t pajeAddComment` (`const char *comment`)

Add some Comment in Trace file.

- `trace_return_t pajeEndTrace` ()

Finalize a PAJE trace.

- `trace_return_t viteEndTrace` ()

Finalize a VITE trace.

- `void pajeEventDefAddParam` (`enum gtg_paje_evtdef_e` event, `const char *name`, `enum gtg_paje_fieldtype_e` type)

7.11.1 Detailed Description

`paje_GTGBasic1` is the Paje implementation of the basic interface to generate traces (`GTGBasic1`).

Version

0.1

Authors

Developers are :

Francois Rue - `francois.rue@labri.fr`

Francois Trahay - `francois.trahay@labri.fr`

Johnny Jazeix - `jazeix@enseirb-matmeca.fr`

Kevin Coulomb - `kevin.coulomb@gmail.com`

Mathieu Faverge - `faverge@labri.fr`

Olivier Lagrasse - `lagrasse@enseirb-matmeca.fr`

7.11.2 Define Documentation

7.11.2.1 `#define FMT_PAJE 0`

Constant to create a paje trace.

7.11.2.2 #define FMT_VITE 1

Constant to create a vite trace.

7.11.3 Enumeration Type Documentation

7.11.3.1 enum gtg_paje_evtdef_e

Enumerator:

GTG_PAJE_EVTDEF_DefineContainerType
GTG_PAJE_EVTDEF_DefineStateType
GTG_PAJE_EVTDEF_DefineEventType
GTG_PAJE_EVTDEF_DefineEntityValue
GTG_PAJE_EVTDEF_CreateContainer
GTG_PAJE_EVTDEF_DestroyContainer
GTG_PAJE_EVTDEF_SetState
GTG_PAJE_EVTDEF_PushState
GTG_PAJE_EVTDEF_PopState
GTG_PAJE_EVTDEF_NewEvent
GTG_PAJE_EVTDEF_DefineLinkType
GTG_PAJE_EVTDEF_StartLink
GTG_PAJE_EVTDEF_EndLink
GTG_PAJE_EVTDEF_DefineVariableType
GTG_PAJE_EVTDEF_SetVariable
GTG_PAJE_EVTDEF_AddVariable
GTG_PAJE_EVTDEF_SubVariable
GTG_PAJE_EVTDEF_NBR

7.11.3.2 enum gtg_paje_fieldtype_e

Enumerator:

GTG_PAJE_FIELDTYPE_Int
GTG_PAJE_FIELDTYPE_Hex
GTG_PAJE_FIELDTYPE_Date
GTG_PAJE_FIELDTYPE_Double
GTG_PAJE_FIELDTYPE_String
GTG_PAJE_FIELDTYPE_Color
GTG_PAJE_FIELDTYPE_NBR

7.11.4 Function Documentation

- 7.11.4.1 void `pajeEventDefAddParam` (enum `gtg_paje_evtdef_e` *event*, const char *
name, enum `gtg_paje_fieldtype_e` *type*)

7.12 GTGReplay.h File Reference

This file defines functions for postponing event-processing function calls.

Enumerations

- enum `event_type_t` { `event_addContainer`, `event_destroyContainer`, `event_setState`, `event_pushState`, `event_popState`, `event_addEvent`, `event_startLink`, `event_endLink`, `event_setVar`, `event_addVar`, `event_subVar` }

Functions

- void `gtg_record` (enum `event_type_t` *type*, `varPrec` *time*,...)
postpone the recording of an event
- void `gtg_write_events` (long *nb_events_to_write*)
run the first nb_events_to_write events

7.12.1 Detailed Description

This file defines functions for postponing event-processing function calls.

Version

0.1

7.12.2 Enumeration Type Documentation

7.12.2.1 enum `event_type_t`

Enumerator:

`event_addContainer`
`event_destroyContainer`
`event_setState`
`event_pushState`
`event_popState`
`event_addEvent`
`event_startLink`
`event_endLink`

event_setVar

event_addVar

event_subVar

7.13 GTGStack.h File Reference

```
#include "GTGList.h"
```

Defines

- #define [GTG_STACK_INIT](#)(ptr) [GTG_LIST_INIT](#)(ptr)
- #define [GTG_STACK](#)(ptr) [GTG_LIST](#)(ptr)
- #define [gtg_stack_entry](#)(ptr, type, member) [gtg_list_entry](#)(ptr, type, member)

Typedefs

- typedef struct [gtg_list](#) [gtg_stack](#)
- typedef [gtg_stack](#) * [gtg_stack_t](#)

Functions

- static void [gtg_stack_push](#) ([gtg_stack_t](#) lnew, [gtg_stack_t](#) p_stack)
- static void [gtg_stack_pop](#) ([gtg_stack_t](#) p_stack)
- static [gtg_stack_t](#) [gtg_stack_top](#) ([gtg_stack_t](#) p_stack)
- static int [gtg_stack_empty](#) ([gtg_stack_t](#) p_stack)

7.13.1 Define Documentation

7.13.1.1 #define [GTG_STACK](#)(ptr) [GTG_LIST](#)(ptr)

7.13.1.2 #define [gtg_stack_entry](#)(ptr, type, member) [gtg_list_entry](#)(ptr, type, member)

7.13.1.3 #define [GTG_STACK_INIT](#)(ptr) [GTG_LIST_INIT](#)(ptr)

7.13.2 Typedef Documentation

7.13.2.1 typedef struct [gtg_list](#) [gtg_stack](#)

7.13.2.2 typedef [gtg_stack](#)* [gtg_stack_t](#)

7.13.3 Function Documentation

7.13.3.1 `static int gtg_stack_empty (gtg_stack_t p_stack)` `[inline, static]`

7.13.3.2 `static void gtg_stack_pop (gtg_stack_t p_stack)` `[inline, static]`

7.13.3.3 `static void gtg_stack_push (gtg_stack_t lnew, gtg_stack_t p_stack)`
`[inline, static]`

7.13.3.4 `static gtg_stack_t gtg_stack_top (gtg_stack_t p_stack)` `[inline, static]`

7.14 GTGTypes.h File Reference

Typedefs

- typedef double [varPrec](#)
Use the double precision type for time and value.
- typedef enum [trace_return_t](#) [trace_return_t](#)

Enumerations

- enum [trace_return_t](#) { [TRACE_SUCCESS](#), [TRACE_ERR_OPEN](#), [TRACE_ERR_CLOSE](#), [TRACE_ERR_WRITE](#), [TRACE_ERR_NOT_IMPL](#) }
Define various return values.

7.14.1 Typedef Documentation

7.14.1.1 typedef enum [trace_return_t](#) [trace_return_t](#)