userif software ver.001 設計書

A 版

2025-04-03

1 Introduction

1.1 Abstract

本書は userif software の設計書である。

userif（user interface）機能は、avisyslator にてシミュレーションを実施する際に、初期値設定／ロギング設定を行う機能、ログ出力を行う機能を提供する。
yaml 形式のファイルにて初期値設定／ロギング設定を行い、設定に従いシミュレーション実行、標準出力／text ファイル出力／ tsv ファイル出力へのログ出力を行う。

1.2 Reference

N/A

2 General Description

2.1 Glossary and Abbrebations

2.1.1 Glossary （用語集）

#	Name	Description
1	stdout	標準出力。
2	symbol_map_file	yaml 形式で実行ファイルの変数名とアドレスを定義した設定ファイル。
3	test_case_file	yaml 形式でテスト開始時の変数の初期値、time_series_log や cyclic_log、event_log 等を定義した設定ファイル。
4	init_input_file	yaml 形式でテスト開始時の変数の初期値を定義した設定ファイル。test_case_file と合わせて使用される。
5	std_out_log_input_file	yaml 形式で stdout に出力する cyclic_log や event_log に関連する情報を定義した設定ファイル。test_case_file と合わせて使用される。
6	time_series_log_input_file	yaml 形式で time_series_log に関連する情報を定義した設定ファイル。test_case_file と合わせて使用される。
7	std_out_log_file	stdout に出力した cyclic_log、event_log を保存したログファイル。
8	time_series_log_file	tsv 形式で test_case_file で定義した変数の値を周期的に出力したログファイル。
9	symbol_map_data	変数名とアドレスを格納したデータ。
10	init_data	変数名と初期値を格納したデータ。
11	cyclic_log_data	stdout に出力する変数情報を格納したデータ。
12	event_log_data	stdout に出力するイベント情報を格納したデータ。
13	time_series_log_data	time_series_log に出力する変数情報を格納したデータ。

2.1.2 Abbreviations （略語集）

N/A

2.2 Architecture

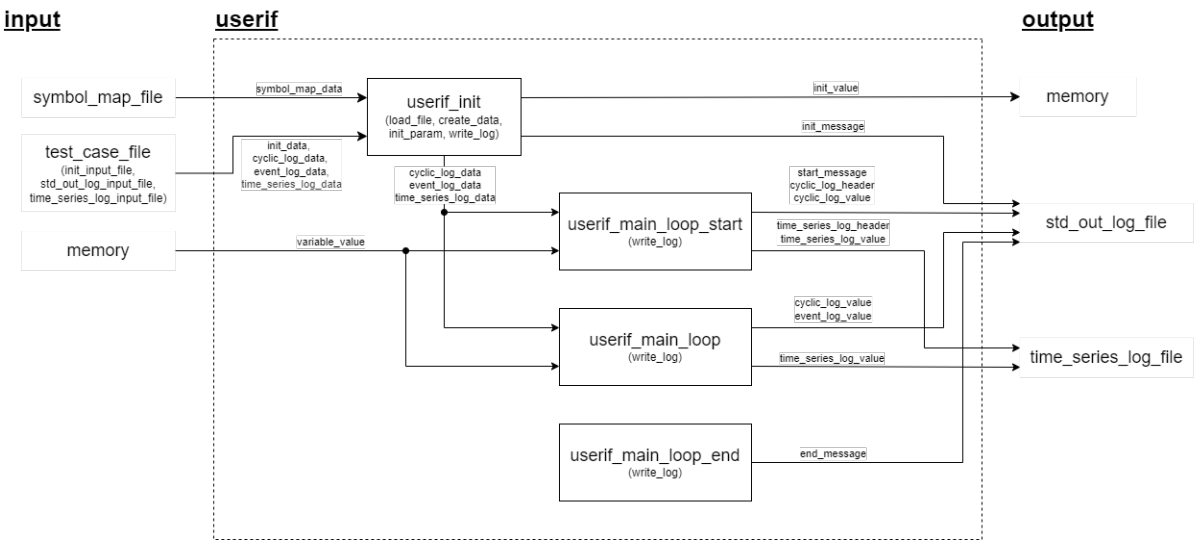
2.2.1 Structure （機能構成）

機能構成の概要を以下に示す。

#	Name	Description
1	userif ユーザーインターフェース	変数の初期化、ログ出力、イベント監視を行う外部関数を提供する。
2	load_file 設定ファイル読み取り	yaml 形式の設定ファイルを読み取る。読み取りに失敗した場合は（例えば指定したファイルがない、フォーマットに誤りがある等)、stdout にエラーを通知し、処理を中断する。読み取りに成功した場合は、読み取った情報を stdout に出力する。
3	create_data 内部データ作成	設定ファイルを読み込んだ情報から userif のログ作成、イベント監視用の内部データを作成する。内部データの個数は制限があり、制限を超えた場合はそれ以上のデータは作成しない。
4	write_log ログ出力	設定ファイルで指定した変数の値やイベント情報をログに出力する。イベントは条件不成立から条件成立時のみ出力する。
5	init_param 初期値設定	設定ファイルで指定した変数に初期値を設定する。同じ変数に２回初期値を設定した場合は、値を上書する。

2.2.2 Block Diagram （ブロック図）

ブロック図を以下に示す。



2.2.3 State Transition Diagram（状態遷移図）

N/A

2.3 Function

機能詳細を以下に示す。

2.3.1 userif

userif 機能を以下に示す。

#	Name	Description
1	userif_init userif の初期化処理	設定ファイルの読み込み、ログファイル作成、変数の初期化を行う。
2	userif_main_loop_start userif の loop の開始処理	シミュレーション開始を通知し、ログにヘッダー、シミュレーション開始時の変数の値を出力する。
3	userif_main_loop_end userif の loop の終了処理	シミュレーション終了を通知する。
4	userif_main userif の実行処理	ログ出力、イベント監視を行う。
5	userif_finish userif の終了処理	ログファイルを保存する。

2.3.2 load_file

load_file 機能を以下に示す。

#	Name	Description
1	load_symbol_map symbol_map_file の読み込み	symbol_map_file を読み込む。実行プロセスの開始アドレスは毎回異なるので、開始アドレスを mark_variable という変数名から計算する。読み込み時、フォーマットに異常がある場合はエラー情報をログに出力し、処理を中断する。
2	load_test_case test_case_file の読み込み	test_case_file を読み込む。読み込み時、フォーマットに異常がある場合はエラー情報をログに出力し、処理を中断する。
3	load_init_input init_input_file の読み込み	init_input_file を読み込む。読み込み時、フォーマットに異常がある場合はエラー情報をログに出力し、処理を中断する。
4	load_time_series_log_input time_series_log_input_file の読み込み	time_series_log_input_file を読み込む。読み込み時、フォーマットに異常がある場合はエラー情報をログに出力し、処理を中断する。
5	load_std_out_log_input std_out_log_input_file の読み込み	std_out_log_input_file を読み込む。読み込み時異常がある場合はエラー情報をログに出力し、処理を中断する。

2.3.3 create_data

create_data 機能を以下に示す。

#	Name	Description
1	create_time_series_log_data time_series_log_data の作成	time_series_log_data を作成する。作成時、symbol_map_data を元に変数のアドレスを計算する。対象ログのデータ数がTIME_SERIES_LOG↩_DATA_MAX を超えた場合、それ以降のデータを無効とする。
2	create_cyclic_log_data cyclic_log_data の作成	cyclic_log_data を作成する。作成時、symbol_map_data を元に変数のアドレスを計算する。対象ログのデータ数がCYCLIC_LOG_DATA_MAX を超えた場合、それ以降のデータを無効とする。
3	create_event_log_data event_log_data の作成	event_log_data を作成する。作成時、symbol_map_data を元にプロセスの変数のアドレスを計算する。イベントデータ数がEVENT_LOG_↩DATA_MAX を超えた場合、それ以降のデータを無効とする。また１つのイベントに対するモニタデータ数がEVENT_MONITOR_DATA_MAX を超えた場合、それ以降のデータを無効とする。
4	create_variable_data variable_data の作成	プロセスの base_address、変数名、symbol_map_data から、指定された変数のアドレスを計算し、variable_data を作成する。指定された変数が symbol_map_data にない場合は、variable_data の変数名は""、アドレスはNULL で返す。
5	get_symbol_address symbol のアドレス取得	指定した変数名に対応するアドレスを取得する。

2.3.4 write_log

write_log 機能を以下に示す。

#	Name	Description
1	write_time_series_log_header time_series_log のヘッダ作成	time_series_log_file に変数名のヘッダを出力する。変数名が symbol_map_data にない場合は処理をスキップする。
2	write_time_series_log_frame time_series_log の frame_data 出力	周期毎に time_series_log_file へ変数の値を出力する。変数名が symbol_map_data にない場合は処理をスキップする。
3	write_cyclic_log_header cyclic_log のヘッダ作成	stdout に変数名のヘッダを出力する。変数名が symbol_map_data にない場合は処理をスキップする。
4	write_cyclic_log_frame cyclic_log の frame_data 出力	周期毎に stdout へ変数の値を出力する。変数名が symbol_map↩_data にない場合は処理をスキップする。
5	write_event_int イベント出力 (型：int)	stdout へ対象イベントの種類との変数の値を出力する。出力の型は int。

#	Name	Description
6	write_event_double イベント出力 (型：double)	stdout へ対象イベントの種類と変数の値を出力する。出力の型は double。
7	write_event_monitor イベントに関連する変数の出力	stdout へイベントに関連する変数の値を出力する。
8	check_event イベント監視	イベントを監視し、イベント条件が不成立から成立した時にイベントログを出力する。
9	check_condition_int イベント条件判定 (型：init)	イベント条件が成立したか判定する。型は int。
10	check_condition_double イベント条件判定 (型：double)	イベント条件が成立したか判定する。型は double。
11	std_trace_init ログ初期化処理	std_out_log_file を作成する。test_case_file を読む前に初期化を行うため、ファイル名はテンポラリファイル"std_log_out_temp.txt"とする。
12	std_out_trace ログ出力	stdout と std_out_log_file にログを出力する。
13	stderr_out_trace エラーログ出力	stderr と std_out_log_file にログを出力する。
14	std_trace_term ログ終了処理	std_out_log_file を保存する。保存時にテンポラリファイルから test_case_file で指定されたファイル名に変更する。

2.3.5 init_param

init_param 機能を以下に示す。

#	Name	Description
1	init_param 変数の初期化	指定した変数を初期化する。同じ変数名を２回初期化した場合、値は上書きされ、警告を表示する。

2.3.6 Others

その他の機能を以下に示す。

#	Name	Description
1	get_exe_dir 実行ファイルのディレクトリ取得	実行しているファイルのディレクトリを取得する。

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

4 File Index

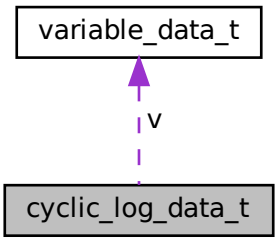
4.1 File List

Here is a list of all documented files with brief descriptions:

5 Data Structure Documentation

5.1 cyclic_log_data_t Struct Reference

Collaboration diagram for cyclic_log_data_t:



Data Fields

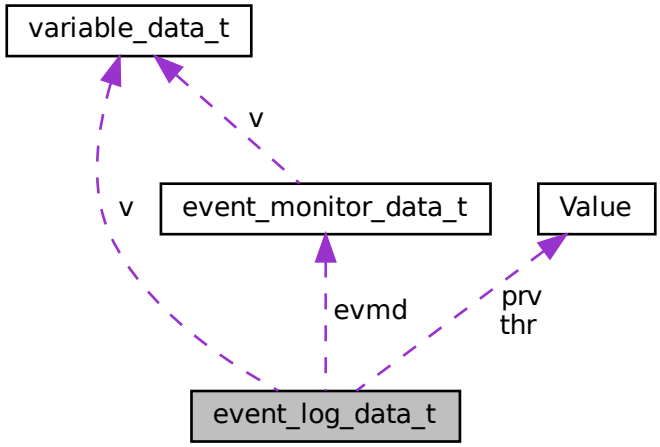
- variable_data_t v
  variable data
- ADDRESS_T next
  next address

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

- src/userif_ext.h

5.2 event_log_data_t Struct Reference

Collaboration diagram for event_log_data_t:



Data Fields

- variable_data_t v
  variable data
- union Value thr
  threshold value
- union Value prv
  previous value
- option_type_e opt_type
  event option type
- eval_type_e eval_type
  evaluation result
- ADDRESS_T **next**
- event_monitor_data_t evmd [EVENT_MONITOR_DATA_MAX]
  event monitor data

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

- src/userif_ext.h

5.3 event_log_t Struct Reference

Data Fields

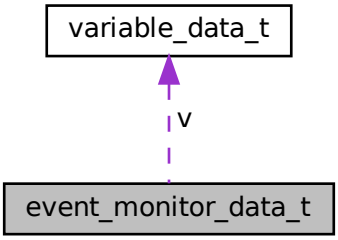
- const char * **variable**
- option_type_e **opt_type**
- const char * **threshold**
- uint32_t **monitors_count**
- const char ** **monitors**

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

- src/test_case_yaml.h

5.4 event_monitor_data_t Struct Reference

Collaboration diagram for event_monitor_data_t:



Data Fields

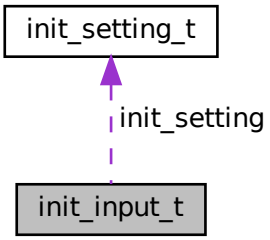
- variable_data_t v
  variable data
- ADDRESS_T next
  next address

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

- src/userif_ext.h

5.5 init_input_t Struct Reference

Collaboration diagram for init_input_t:



Data Fields

- const char * **title**
- init_setting_t * **init_setting**
- uint32_t **init_setting_count**

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

- src/test_case_yaml.h

5.6 init_setting_t Struct Reference

Data Fields

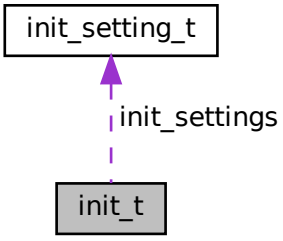
- const char * **variable**
- const char * **value**

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

- src/test_case_yaml.h

5.7 init_t Struct Reference

Collaboration diagram for init_t:



Data Fields

- const char ** **input_files**
- uint32_t **input_files_count**
- init_setting_t * **init_settings**
- uint32_t **init_settings_count**

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

- src/test_case_yaml.h

5.8 option_type_t Struct Reference

Data Fields

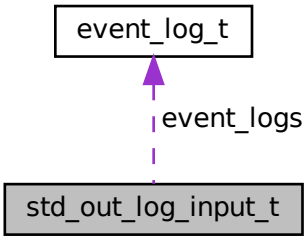
- const char * **str**
- option_type_e **type**

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

- src/userif_ext.h

5.9 std_out_log_input_t Struct Reference

Collaboration diagram for std_out_log_input_t:



Data Fields

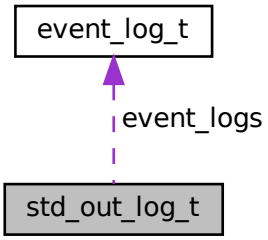
- const char * **title**
- uint32_t **cyclic_log_vairables_count**
- const char ** **cyclic_log_vairables**
- event_log_t * **event_logs**
- uint32_t **event_logs_count**

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

- src/test_case_yaml.h

5.10 std_out_log_t Struct Reference

Collaboration diagram for std_out_log_t:



Data Fields

- uint32_t **cycle**
- const char ∗∗ **input_files**
- uint32_t **input_files_count**
- const char ∗ **output_file**
- uint32_t **cyclic_log_vairables_count**
- const char ∗∗ **cyclic_log_vairables**
- event_log_t ∗ **event_logs**
- uint32_t **event_logs_count**

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

- src/test_case_yaml.h

5.11 symbol_data_t Struct Reference

Data Fields

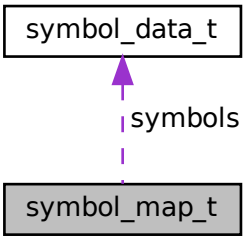
- const char ∗ **name**
- ADDRESS_T **address**
- symbol_type_e **type**

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

- src/symbol_map_yaml.h

5.12 symbol_map_t Struct Reference

Collaboration diagram for symbol_map_t:



Data Fields

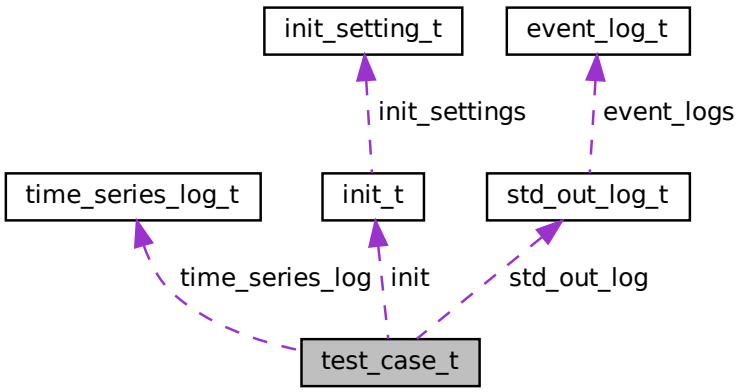
- const char * **title**
- ADDRESS_T **mark_address**
- symbol_data_t * **symbols**
- uint32_t **symbols_count**

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

- src/symbol_map_yaml.h

5.13 test_case_t Struct Reference

Collaboration diagram for test_case_t:



Data Fields

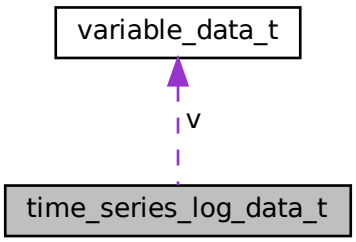
- const char * **title**
- init_t * **init**
- std_out_log_t * **std_out_log**
- time_series_log_t * **time_series_log**

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

- src/test_case_yaml.h

5.14 time_series_log_data_t Struct Reference

Collaboration diagram for time_series_log_data_t:



Data Fields

- variable_data_t v
  variable data
- ADDRESS_T next
  next address

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

- src/userif_ext.h

5.15 time_series_log_input_t Struct Reference

Data Fields

- const char * **title**
- const char ** **time_series_log_variables**
- uint32_t **time_series_log_variables_count**

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

- src/test_case_yaml.h

5.16 time_series_log_t Struct Reference

Data Fields

- uint32_t **cycle**
- const char ∗∗ **input_files**
- uint32_t **input_files_count**
- const char ∗ **output_file**
- const char ∗∗ **time_series_log_variables**
- uint32_t **time_series_log_variables_count**

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

- src/test_case_yaml.h

5.17 user_interface_params_t Struct Reference

Data Fields

- int time_series_log_cycle
  time series log cycle
- int std_out_log_cycle
  stdout log cycle

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

- src/userif_ext.h

5.18 Value Union Reference

Data Fields

- int32_t i_value
  int value
- double d_value
  double value

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

- src/userif_ext.h

5.19 variable_data_t Struct Reference

Data Fields

- char name [LOG_NAME_SIZE]
  variable name
- ADDRESS_T adr
  variable address
- log_type_e type
  variable type

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

- src/userif_ext.h

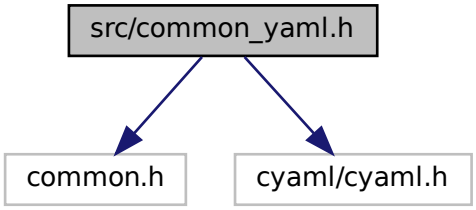
6 File Documentation

6.1 src/common_yaml.h File Reference

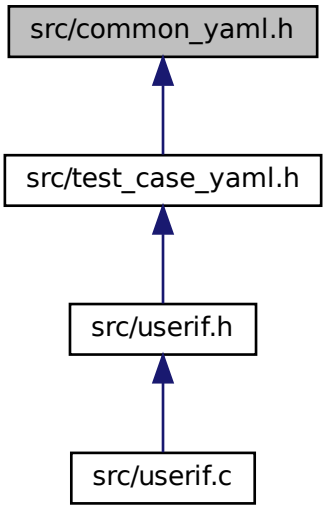
This file contains the common yaml schema.

```
#include "common.h"
#include <cyaml/cyaml.h>
```

Include dependency graph for common_yaml.h:



This graph shows which files directly or indirectly include this file:



6.1.1 Detailed Description

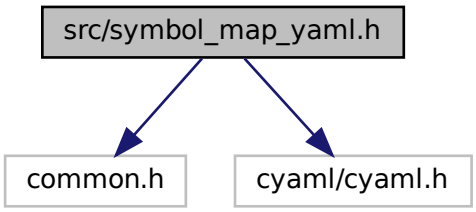
This file contains the common yaml schema.

6.2 src/symbol_map_yaml.h File Reference

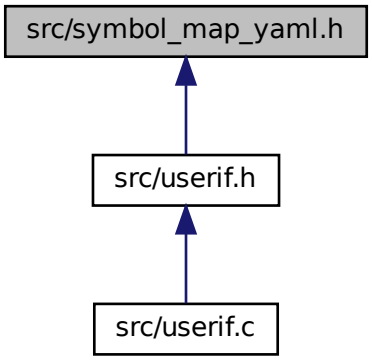
This file contains the symbol map yaml schema.

```
#include "common.h"
#include <cyaml/cyaml.h>
```

Include dependency graph for symbol_map_yaml.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct symbol_data_t
- struct symbol_map_t

Macros

- #define **SYMBOL_MAP_FILE_NAME** "symbol_map.yaml"

Enumerations

- enum **symbol_type_e** {
  SYMBOL_TYPE_INT = 1 , **SYMBOL_TYPE_DBL** , **SYMBOL_TYPE_VECTOR** , **SYMBOL_TYPE_QUAT** , **SYMBOL_TYPE_MATRIX** }

6.2.1 Detailed Description

This file contains the symbol map yaml schema.

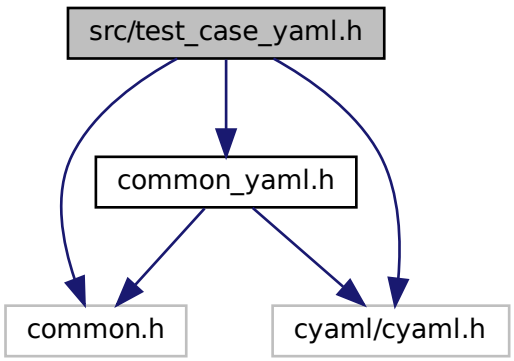
6.3 src/test_case_yaml.h File Reference

This file contains the test case yaml schema.

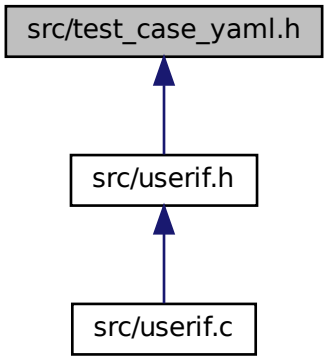
```
#include "common.h"
#include "common_yaml.h"
```

```
#include <cyaml/cyaml.h>
```

Include dependency graph for test_case_yaml.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct init_setting_t
- struct event_log_t
- struct std_out_log_t

- struct time_series_log_t
- struct init_t
- struct test_case_t
- struct init_input_t
- struct time_series_log_input_t
- struct std_out_log_input_t

6.3.1 Detailed Description

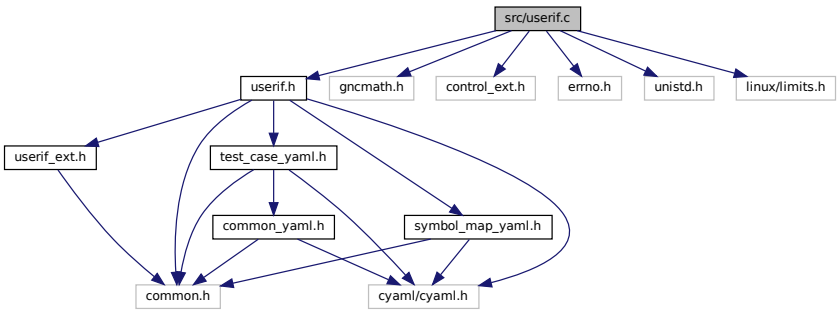
This file contains the test case yaml schema.

6.4 src/userif.c File Reference

This file contains the user interface functions.

```
#include "userif.h"
#include "gncmath.h"
#include "control_ext.h"
#include <errno.h>
#include <unistd.h>
#include <linux/limits.h>
```

Include dependency graph for userif.c:



Macros

- #define STD_OUT_LOG_TEMP_FILE_NAME "std_log_out_temp.txt"
  std out log temp file name
- #define **LTE**(val, thr) (thr >= val)

- #define **LT**(val, thr) (thr > val)
- #define **GTE**(val, thr) (thr <= val)
- #define **GT**(val, thr) (thr < val)
- #define **EQ**(src, dest) (src == dest)
- #define **NEQ**(src, dest) (src != dest)
- #define **CHG**(src, dest) (src != dest)

Functions

- int std_trace_init (char *trace_file_path)
  Initialize trace log file.
- void std_out_trace (char *format,...)
  Output log stream to stdout and log file.
- void stderr_out_trace (char *format,...)
  Output log stream to stderr and log file.
- int std_trace_term ()
  Terminates trace log file.
- void get_exe_dir (char *dir, int dir_len)
  Get the directory of the executable process.
- int get_symbol_address (const char *_name_target, symbol_map_t *symbol_map, ADDRESS_T *address, symbol_type_e *type)
  Get the address of the specified variable.
- int load_symbol_map (char *file_path, ADDRESS_T *p_adr_base, symbol_map_t **pp_sm)
  Load symbol map file.
- int load_test_case (char *file_path, test_case_t **pp_tc)
  Load test case file.
- int load_init_input (char *file_path, init_input_t **pp_in)
  Load input init file.
- int load_time_series_log_input (char *file_path, time_series_log_input_t **pp_tsli)
  Load time series log input file.
- int load_std_out_log_input (char *file_path, std_out_log_input_t **pp_soli)
  Log the std out log data.
- void init_param (ADDRESS_T adr_base, symbol_map_t *sm, init_setting_t *settings, uint32_t count, uint32_t *offset)
  Initialize the user interface.
- void create_variable_data (ADDRESS_T adr_base, symbol_map_t *sm, const char *name, variable_data_t *var)
  Create variable data.
- void create_time_series_log_data (ADDRESS_T adr_base, symbol_map_t *sm, const char **variables, uint32_t variables_count, uint32_t *offset)
  Create time series log data.
- void create_cyclic_log_data (ADDRESS_T adr_base, symbol_map_t *sm, const char **variables, uint32_t variables_count, uint32_t *offset)
  Create cyclic log data.
- void create_event_log_data (ADDRESS_T adr_base, symbol_map_t *sm, event_log_t *event_logs, uint32_t event_logs_count, uint32_t *offset)

Create event log data.

- void write_time_series_log_header ()
  Write time series log header.
- void write_time_series_log_frame (int cycle, double t)
  Write time series log frame.
- void write_cyclic_log_header ()
  Write cyclic log header.
- void write_cyclic_log_frame (int cycle, double t)
  Write cyclic log frame.
- void write_event_monitor (event_log_data_t *_evld)
  Write event log frame.
- void write_event_int (option_type_e opt_type, double t, char *name, int cur, int prv)
  Write event log. (int version)
- void write_event_double (option_type_e opt_type, double t, char *name, double cur, double prv)
  Write event log. (double version)
- eval_type_e check_condition_int (option_type_e opt_type, int src, int dest, int thr)
  Check the event condition.(int version)
- eval_type_e check_condition_double (option_type_e opt_type, double src, double dest, double thr)
  Check the event condition.(double version)
- void check_event (double t)
  Check the event.
- int userif_init (double t, char *test_case_file)
  Initialize userif.
- int userif_main_loop_start (double t)
  Initialize main loop.
- int userif_main_loop_end ()
  Terminate main loop.
- int userif_main (int cycle, double t)
  Execute main loop.
- int userif_finish (void)
  Terminate userif.

Variables

- int32_t mark_variable = 0
  mark variable for calculating the base address of the symbol map
- user_interface_params_t uip = {0}
  user interface parameters
- time_series_log_data_t tsld [TIME_SERIES_LOG_DATA_MAX] = {0}
  time series log data array
- cyclic_log_data_t cold [CYCLIC_LOG_DATA_MAX] = {0}
  cyclic log data array
- event_log_data_t evld [EVENT_LOG_DATA_MAX] = {0}
  event log data array
- char init_vars [INIT_DATA_MAX][LOG_NAME_SIZE] ={0}
  initialized variables
- FILE * tslfp = NULL
  time series log file pointer

- FILE * solfp = NULL
  std out log file pointer
- char std_log_out_temp_file_path [PATH_MAX] = {0}
  std out log temp file path
- char std_out_log_file_path [PATH_MAX] = {0}
  std out log file path

6.4.1 Detailed Description

This file contains the user interface functions.

6.4.2 Function Documentation

6.4.2.1 check_condition_double()

```
eval_type_e check_condition_double (
        option_type_e opt_type,
        double src,
        double dest,
        double thr )
```

Check the event condition.(double version)

Parameters

in	*opt_type*	option type
in	*src*	source value
in	*dest*	destination value
in	*thr*	threshold value

Returns

EVAL_TRUE: true, EVAL_FALSE: false, EVAL_NONE: none

6.4.2.2 check_condition_int()

```
eval_type_e check_condition_int (
        option_type_e opt_type,
        int src,
        int dest,
        int thr )
```

Check the event condition.(int version)

Parameters

in	*opt_type*	option type
in	*src*	source value
in	*dest*	destination value
in	*thr*	threshold value

Returns

EVAL_TRUE: true, EVAL_FALSE: false, EVAL_NONE: none

6.4.2.3 check_event()

```
void check_event (
        double t )
```

Check the event.

Parameters

in	*t*	simulation time

6.4.2.4 create_cyclic_log_data() `void create_cyclic_log_data (`

```
ADDRESS_T adr_base,
symbol_map_t * sm,
const char ** variables,
uint32_t variables_count,
uint32_t * offset )
```

Create cyclic log data.

Parameters

in	*adr_base*	base address of the executable process
in	*sm*	symbol map data
in	*variables*	variable name buffer
in	*variables_count*	number of variable name buffer
in,out	*offset*	offset of the cyclic log data buffer

6.4.2.5 create_event_log_data() `void create_event_log_data (`

```
ADDRESS_T adr_base,
symbol_map_t * sm,
event_log_t * event_logs,
uint32_t event_logs_count,
uint32_t * offset )
```

Create event log data.

Parameters

in	*adr_base*	base address of the executable process
in	*sm*	symbol map data
in	*event_logs*	event log data buffer
in	*event_logs_count*	number of event log data buffer
in,out	*offset*	offset of the event log data buffer

6.4.2.6 create_time_series_log_data() `void create_time_series_log_data (`

```
ADDRESS_T adr_base,
symbol_map_t * sm,
const char ** variables,
uint32_t variables_count,
uint32_t * offset )
```

Create time series log data.

Parameters

in	*adr_base*	base address of the executable process
in	*sm*	symbol map data
in	*variables*	variable name buffer
in	*variables_count*	number of variable name buffer
in,out	*offset*	offset of the time series log data buffer

6.4.2.7 create_variable_data() `void create_variable_data (`

```
ADDRESS_T adr_base,
symbol_map_t * sm,
const char * name,
variable_data_t * var )
```

Create variable data.

Parameters

in	*adr_base*	base address of the executable process
in	*sm*	symbol map data
in	*name*	variable name
out	*var*	variable data

6.4.2.8 get_exe_dir() `void get_exe_dir (`

```
char * dir,
int dir_len )
```

Get the directory of the executable process.

Parameters

out	*dir*	directory of the executable file
in	*dir_len*	length of the directory

6.4.2.9 get_symbol_address() `int get_symbol_address (`

```
const char * _name_target,
symbol_map_t * symbol_map,
ADDRESS_T * address,
symbol_type_e * type )
```

Get the address of the specified variable.

Parameters

in	*symbol_map*	symbol map of the target executable file
out	*address*	address of the target variable
out	*type*	type of the target variable

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.10 init_param() `void init_param (`

```
ADDRESS_T adr_base,
symbol_map_t * sm,
init_setting_t * settings,
uint32_t count,
uint32_t * offset )
```

Initialize the user interface.

Parameters

in	*adr_base*	base address of the executable process
in	*sm*	symbol map data
in	*setting*	initialzation setting buffer
in	*count*	number of initialization setting buffer
in,out	*offset*	offset of the initialized variable buffer

Note

If sama variable is initialized, the value is overwritten and output warining message.

6.4.2.11 load_init_input() `int load_init_input (`

```
char * file_path,
init_input_t ** pp_in )
```

Load input init file.

Parameters

in	*file_path*	init input file path
out	*pp_in*	init input data

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.12 load_std_out_log_input() `int load_std_out_log_input (`

```
char * file_path,
std_out_log_input_t ** pp_soli )
```

Log the std out log data.

Parameters

in	*file_path*	std out log file path
out	*pp_soli*	std out log input data

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.13 load_symbol_map() `int load_symbol_map (`

```
char * file_path,
ADDRESS_T * p_adr_base,
symbol_map_t ** pp_sm )
```

Load symbol map file.

Parameters

in	*file_path*	symbol map file path
in	*p_adr_base*	base address of the executable process
out	*pp_sm*	symbol map data

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.14 load_test_case() `int load_test_case (`

```
char * file_path,
test_case_t ** pp_tc )
```

Load test case file.

Parameters

in	*file_path*	test case file path
out	*pp_tc*	test case data

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.15 load_time_series_log_input() `int load_time_series_log_input (`

```
char * file_path,
time_series_log_input_t ** pp_tsli )
```

Load time series log input file.

Parameters

in	*file_path*	time series log input file path
out	*pp_tsli*	time series log input data

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.16 std_out_trace() `void std_out_trace (`

```
            char * format,
             ...  )
```

Output log stream to stdout and log file.

Parameters

in	*format*	format of log output

6.4.2.17 std_trace_init() `int std_trace_init (`

```
            char * trace_file_path )
```

Initialize trace log file.

Parameters

in	*trace_file_path*	log file path

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.18 std_trace_term() `int std_trace_term ( )`

Terminates trace log file.

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.19 stderr_out_trace()

```
void stderr_out_trace (
        char * format,
         ...  )
```

Output log stream to stderr and log file.

Parameters

in	*format*	format of log output

6.4.2.20 userif_finish()

```
int userif_finish (
        void  )
```

Terminate userif.

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

Note

If the process exits without calling this function, the log file will not be saved.

6.4.2.21 userif_init()

```
int userif_init (
        double t,
        char * test_case_file )
```

Initialize userif.

Parameters

in	*t*	simulation time
in	*test_case_file*	test case file path

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

Note

The param (t) is not used.

6.4.2.22 userif_main() `int userif_main (`

```
int cycle,
double t )
```

Execute main loop.

Parameters

in	*cycle*	cycle count
in	*t*	simulation time

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.23 userif_main_loop_end() `int userif_main_loop_end ( )`

Terminate main loop.

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.24 userif_main_loop_start() `int userif_main_loop_start (`

`double t )`

Initialize main loop.

Parameters

in	*t*	simulation time

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.4.2.25 write_cyclic_log_frame() `void write_cyclic_log_frame (`

`int cycle,`

`double t )`

Write cyclic log frame.

Parameters

in	*cycle*	cycle count
in	*t*	simulation time

6.4.2.26 write_event_double() `void write_event_double (`

```
option_type_e opt_type,
double t,
char * name,
double cur,
double prv )
```

Write event log. (double version)

Parameters

in	*opt_type*	option type
in	*t*	simulation time
in	*name*	variable name
in	*cur*	current value
in	*prv*	previous value

6.4.2.27 write_event_int() `void write_event_int (`

```
option_type_e opt_type,
double t,
char * name,
int cur,
int prv )
```

Write event log. (int version)

Parameters

in	*opt_type*	option type
in	*t*	simulation time
in	*name*	variable name
in	*cur*	current value
in	*prv*	previous value

6.4.2.28 write_event_monitor() `void write_event_monitor (`

`event_log_data_t * _evld )`

Write event log frame.

Parameters

in	*_evld*	event log data

6.4.2.29 write_time_series_log_frame() `void write_time_series_log_frame (`

`int cycle,`

`double t )`

Write time series log frame.

Parameters

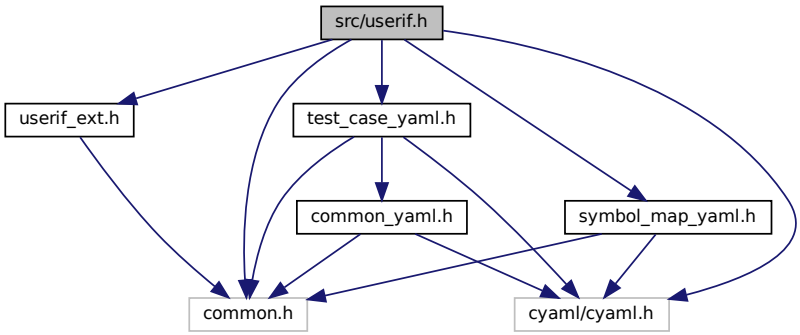
in	*cycle*	cycle count
in	*t*	simulation time

6.5 src/userif.h File Reference

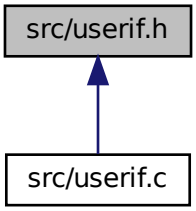
This file contains the user interface.

```
#include "common.h"
#include "userif_ext.h"
#include "symbol_map_yaml.h"
#include "test_case_yaml.h"
#include <cyaml/cyaml.h>
```

Include dependency graph for userif.h:



This graph shows which files directly or indirectly include this file:



Functions

- int userif_init (double t, char *test_case_file)
  Initialize userif.
- int userif_main_loop_start (double t)
  Initialize main loop.
- int userif_main_loop_end ()
  Terminate main loop.
- int userif_main (int cycle, double t)
  Execute main loop.
- int userif_finish (void)
  Terminate userif.

6.5.1 Detailed Description

This file contains the user interface.

6.5.2 Function Documentation

6.5.2.1 userif_finish() `int userif_finish (`

```
void  )
```

Terminate userif.

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

Note

If the process exits without calling this function, the log file will not be saved.

6.5.2.2 userif_init() `int userif_init (`

```
double t,
char * test_case_file )
```

Initialize userif.

Parameters

in	*t*	simulation time
in	*test_case_file*	test case file path

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

Note

The param (t) is not used.

6.5.2.3 userif_main() `int userif_main (`

```
int cycle,
double t )
```

Execute main loop.

Parameters

in	*cycle*	cycle count
in	*t*	simulation time

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.5.2.4 userif_main_loop_end() `int userif_main_loop_end ( )`

Terminate main loop.

Returns

EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.5.2.5 userif_main_loop_start() `int userif_main_loop_start ( double t )`

Initialize main loop.

Parameters

in	*t*	simulation time

Returns

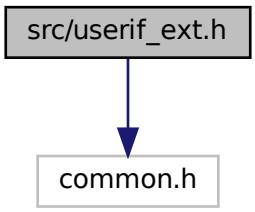
EXIT_SUCCESS: Success, EXIT_FAILURE: Failure

6.6 src/userif_ext.h File Reference

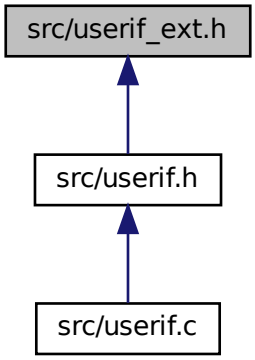
This file contains the user interface extension.

```
#include "common.h"
```

Include dependency graph for userif_ext.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct option_type_t
- struct user_interface_params_t
- struct variable_data_t
- struct time_series_log_data_t
- struct cyclic_log_data_t
- struct event_monitor_data_t
- union Value
- struct event_log_data_t

Macros

- #define **LOG_NAME_SIZE** (64)
- #define **INIT_DATA_MAX** (256)
- #define **TIME_SERIES_LOG_DATA_MAX** (256)
- #define **CYCLIC_LOG_DATA_MAX** (256)
- #define **EVENT_LOG_DATA_MAX** (256)
- #define **EVENT_MONITOR_DATA_MAX** (16)
- #define **MONTE_CARLO_DATA_MAX** (256)
- #define **DELAY_DATA_MAX** (256)
- #define **DELAY_BUF_MAX** (128)
- #define **INPUT_FILE_NAME_SIZE** (64)

Enumerations

- enum option_type_e {
  OPT_TYPE_NONE = 0 , OPT_TYPE_CHG , OPT_TYPE_LTE , OPT_TYPE_LT ,
  OPT_TYPE_GTE , OPT_TYPE_GT , OPT_TYPE_EQ , OPT_TYPE_NEQ }
- enum eval_type_e { EVAL_NONE = 0 , EVAL_FALSE , EVAL_TRUE }
- enum log_type_e { LOG_TYPE_UNKNOWN = 0 , LOG_TYPE_INT , LOG_TYPE_DBL }

Variables

- user_interface_params_t uip
  user interface parameters
- time_series_log_data_t tsld [TIME_SERIES_LOG_DATA_MAX]
  time series log data array
- cyclic_log_data_t **sold** [CYCLIC_LOG_DATA_MAX]
- event_log_data_t evld [EVENT_LOG_DATA_MAX]
  event log data array

6.6.1 Detailed Description

This file contains the user interface extension.

6.6.2 Enumeration Type Documentation

6.6.2.1 eval_type_e `enum eval_type_e`

Enumerator

EVAL_NONE	default
EVAL_FALSE	event condition is false
EVAL_TRUE	event condition is true

6.6.2.2 log_type_e `enum log_type_e`

Enumerator

LOG_TYPE_UNKNOWN	default
LOG_TYPE_INT	int
LOG_TYPE_DBL	double

6.6.2.3 option_type_e `enum option_type_e`

Enumerator

OPT_TYPE_NONE	defalut
OPT_TYPE_CHG	change
OPT_TYPE_LTE	less than or equal
OPT_TYPE_LT	less than
OPT_TYPE_GTE	greater than or equal
OPT_TYPE_GT	greater than
OPT_TYPE_EQ	equal
OPT_TYPE_NEQ	not equal