

# SART Developer's Guide

1.00

Semantic Aware Real-Time scheduler

September 2010



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	list_t Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	6
3.2	SRT_CONJUNCTION Struct Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.3	SRT_DATA Struct Reference . . . . .	8
3.3.1	Detailed Description . . . . .	8
3.4	SRT_DISJUNCTION Struct Reference . . . . .	9
3.4.1	Detailed Description . . . . .	9
3.5	SRT_JOB Struct Reference . . . . .	10
3.5.1	Detailed Description . . . . .	10
3.6	srt_job_attr Struct Reference . . . . .	11
3.6.1	Detailed Description . . . . .	11
3.6.2	Field Documentation . . . . .	11
3.6.2.1	disabled . . . . .	11
3.6.2.2	policy . . . . .	11
3.6.2.3	prominence . . . . .	11
3.7	SRT_RESOURCE Struct Reference . . . . .	12
3.7.1	Detailed Description . . . . .	12
3.8	SRT_TASK Struct Reference . . . . .	13
3.8.1	Detailed Description . . . . .	13
3.9	srt_task_attr Struct Reference . . . . .	14
3.9.1	Detailed Description . . . . .	14

3.9.2	Field Documentation	14
3.9.2.1	constraint_disabled	14
3.9.2.2	constraint_forced	14
3.9.2.3	constraint_hard	14
<b>4</b>	<b>File Documentation</b>	<b>15</b>
4.1	graph.h File Reference	15
4.1.1	Detailed Description	16
4.1.2	Function Documentation	16
4.1.2.1	graph_add_edge	16
4.1.2.2	graph_add_vertex	16
4.1.2.3	graph_destroy	17
4.1.2.4	graph_edge_at	17
4.1.2.5	graph_init	17
4.1.2.6	graph_remove_all_edges	17
4.1.2.7	graph_remove_edge	18
4.1.2.8	graph_remove_edge_at	18
4.1.2.9	graph_remove_vertex	18
4.1.2.10	graph_remove_vertex_at	19
4.1.2.11	graph_vertex_at	19
4.1.2.12	graph_vertex_by_label	19
4.2	sart.h File Reference	20
4.2.1	Detailed Description	21
4.2.2	Typedef Documentation	21
4.2.2.1	schd_checker	21
4.2.3	Function Documentation	21
4.2.3.1	srt_chek_kb	21
4.2.3.2	srt_conj_add_conj	22
4.2.3.3	srt_conj_add_data	22
4.2.3.4	srt_conj_add_disj	22
4.2.3.5	srt_conj_create	22
4.2.3.6	srt_data_create	23
4.2.3.7	srt_data_set_attr	23
4.2.3.8	srt_disj_add_conj	23
4.2.3.9	srt_disj_add_data	23
4.2.3.10	srt_disj_add_disj	24
4.2.3.11	srt_disj_create	24

4.2.3.12	<code>srt_export_kb2dot</code>	24
4.2.3.13	<code>srt_finish</code>	24
4.2.3.14	<code>srt_get_entity_by_name</code>	24
4.2.3.15	<code>srt_init</code>	25
4.2.3.16	<code>srt_job_add_data</code>	25
4.2.3.17	<code>srt_job_add_task</code>	25
4.2.3.18	<code>srt_job_create</code>	25
4.2.3.19	<code>srt_job_get_attr</code>	26
4.2.3.20	<code>srt_job_set_attr</code>	26
4.2.3.21	<code>srt_res_add_provision</code>	26
4.2.3.22	<code>srt_res_create</code>	27
4.2.3.23	<code>srt_res_set_attr</code>	27
4.2.3.24	<code>srt_set_sched_checker</code>	27
4.2.3.25	<code>srt_switch_context</code>	27
4.2.3.26	<code>srt_task_add_conj_input</code>	28
4.2.3.27	<code>srt_task_add_data_input</code>	28
4.2.3.28	<code>srt_task_add_disj_input</code>	28
4.2.3.29	<code>srt_task_add_output</code>	28
4.2.3.30	<code>srt_task_create</code>	29
4.2.3.31	<code>srt_task_get_attr</code>	29
4.2.3.32	<code>srt_task_set_attr</code>	29
4.2.3.33	<code>srt_task_set_descriptor</code>	29
4.2.3.34	<code>srt_test_solution</code>	30
4.3	<code>srt_kernel.h</code> File Reference	31
4.3.1	Detailed Description	31
4.3.2	Function Documentation	31
4.3.2.1	<code>count_common_task</code>	31
4.3.2.2	<code>min_dependency_path</code>	32
4.3.2.3	<code>rm_sched_checker</code>	32
4.3.2.4	<code>select_task_set</code>	32
4.3.2.5	<code>update_kb_graph</code>	33
4.4	<code>srt_os_generic.h</code> File Reference	34
4.4.1	Detailed Description	34
4.4.2	Function Documentation	34
4.4.2.1	<code>__activate_task</code>	34
4.4.2.2	<code>__deactivate_task</code>	35

---

4.4.2.3	__is_task_active	35
4.4.2.4	__post_context_switch	35
4.4.2.5	__pre_context_switch	35
4.5	srterrno.h File Reference	36
4.5.1	Detailed Description	36
4.5.2	Define Documentation	36
4.5.2.1	__ASSERT	36
4.5.2.2	ERR_NO_DATAKIND	36
4.6	srttypes.h File Reference	37
4.6.1	Detailed Description	38
4.6.2	Define Documentation	38
4.6.2.1	OSROUTINE	38
4.6.2.2	SYSCALL	38
4.6.3	Enumeration Type Documentation	38
4.6.3.1	SRT_CONCEPT	38
4.6.3.2	SRT_POLICY	38

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">list_t</a>	5
<a href="#">SRT_CONJUNCTION</a>	7
<a href="#">SRT_DATA</a>	8
<a href="#">SRT_DISJUNCTION</a>	9
<a href="#">SRT_JOB</a>	10
<a href="#">srt_job_attr</a>	11
<a href="#">SRT_RESOURCE</a>	12
<a href="#">SRT_TASK</a>	13
<a href="#">srt_task_attr</a>	14





# Chapter 2

## File Index

### 2.1 File List

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

<a href="#">graph.h</a>	15
<a href="#">sart.h</a>	20
<a href="#">srt_kernel.h</a>	31
<a href="#">srt_os_generic.h</a>	34
<a href="#">srtermo.h</a>	36
<a href="#">srtypes.h</a>	37



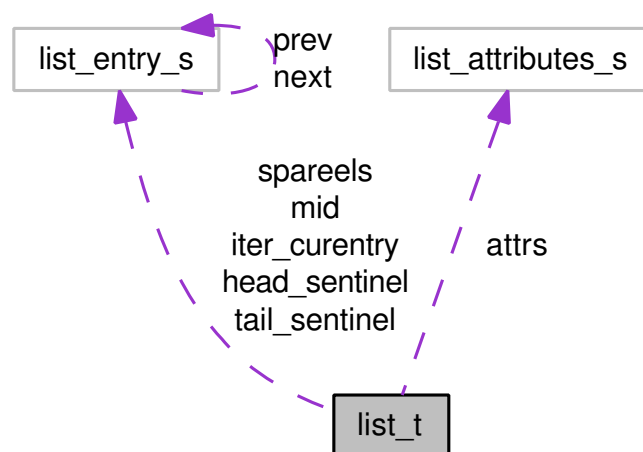
# Chapter 3

## Data Structure Documentation

### 3.1 list\_t Struct Reference

```
#include <simclist.h>
```

Collaboration diagram for list\_t:



#### Data Fields

- struct list\_entry\_s \* **head\_sentinel**
- struct list\_entry\_s \* **tail\_sentinel**
- struct list\_entry\_s \* **mid**
- unsigned int **numels**
- struct list\_entry\_s \*\* **spareels**
- unsigned int **spareelsnum**
- int **iter\_active**
- unsigned int **iter\_pos**
- struct list\_entry\_s \* **iter\_cureentry**
- struct list\_attributes\_s **attrs**

### 3.1.1 Detailed Description

list object

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

- [simclist.h](#)

## 3.2 SRT\_CONJUNCTION Struct Reference

```
#include <srctypes.h>
```

### Data Fields

- `srt_id` `id`

### 3.2.1 Detailed Description

SART Conjunction

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

- [srctypes.h](#)

## 3.3 SRT\_DATA Struct Reference

```
#include <srttpes.h>
```

### Data Fields

- `srt_id` `id`

### 3.3.1 Detailed Description

SART Data

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

- [srttpes.h](#)

## 3.4 SRT\_DISJUNCTION Struct Reference

```
#include <srttpes.h>
```

### Data Fields

- `srt_id` `id`

### 3.4.1 Detailed Description

SART Disjunction

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

- [srttpes.h](#)

## 3.5 SRT\_JOB Struct Reference

```
#include <srttpes.h>
```

### Data Fields

- `srt_id` `id`

### 3.5.1 Detailed Description

SART Job

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

- [srttpes.h](#)



## 3.6 srt\_job\_attr Struct Reference

```
#include <sart.h>
```

### Data Fields

- bool [disabled](#)
- u\_int [prominence](#)
- SRT\_POLICY [policy](#)

### 3.6.1 Detailed Description

Job attributes structure

### 3.6.2 Field Documentation

#### 3.6.2.1 bool srt\_job\_attr::disabled

Disabled Job

#### 3.6.2.2 SRT\_POLICY srt\_job\_attr::policy

Reasoning policy assigned to the Job

#### 3.6.2.3 u\_int srt\_job\_attr::prominence

Prominence of the Job

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

- [sart.h](#)

## 3.7 SRT\_RESOURCE Struct Reference

```
#include <srctypes.h>
```

### Data Fields

- `srt_id` `id`

### 3.7.1 Detailed Description

SART Resource

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

- [srctypes.h](#)

## 3.8 SRT\_TASK Struct Reference

```
#include <srctypes.h>
```

### Data Fields

- `srt_id` **id**
- `void *` **descriptor**

### 3.8.1 Detailed Description

SART Task

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

- [srctypes.h](#)

## 3.9 srt\_task\_attr Struct Reference

```
#include <sart.h>
```

### Data Fields

- bool [constraint\\_hard](#)
- bool [constraint\\_forced](#)
- bool [constraint\\_disabled](#)

### 3.9.1 Detailed Description

Task attributes structure

### 3.9.2 Field Documentation

#### 3.9.2.1 bool srt\_task\_attr::constraint\_disabled

Disabled task

#### 3.9.2.2 bool srt\_task\_attr::constraint\_forced

Forced run task

#### 3.9.2.3 bool srt\_task\_attr::constraint\_hard

Hard real time task

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

- [sart.h](#)

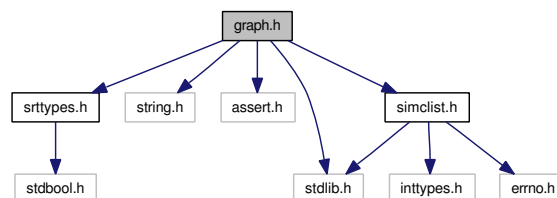
# Chapter 4

## File Documentation

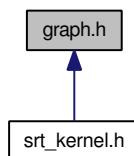
### 4.1 graph.h File Reference

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "srtpypes.h"
#include "simclist.h"
```

Include dependency graph for graph.h:



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



### Data Structures

- struct **vertex**
- struct **edge**
- struct **graph**

## Functions

- int `graph_init` (graph \*g)  
*initialize a graph object for use.*
- void `graph_destroy` (graph \*g)
- int `graph_add_vertex` (graph \*g, void \*data, SRT\_CONCEPT type, const char \*label)
- vertex \* `graph_vertex_at` (graph \*g, size\_t index)
- vertex \* `graph_vertex_by_label` (graph \*g, const char \*label)
- int `graph_remove_vertex_at` (graph \*g, size\_t index)
- int `graph_remove_vertex` (graph \*g, vertex \*v)
- int `graph_add_edge` (graph \*g, vertex \*v\_source, vertex \*v\_dest, float w)
- int `graph_remove_edge` (graph \*g, vertex \*v\_source, vertex \*v\_dest)
- int `graph_remove_edge_at` (graph \*g, vertex \*v, size\_t edge\_index)
- int `graph_remove_all_edges` (graph \*g, vertex \*v)
- edge \* `graph_edge_at` (graph \*g, vertex \*v, size\_t index)

### 4.1.1 Detailed Description

#### 4.1.2 Function Documentation

##### 4.1.2.1 int `graph_add_edge` (graph \* g, vertex \* v\_source, vertex \* v\_dest, float w)

adding an edge to a vertex of graph

##### Parameters:

*g* a pointer to an initialized graph object  
*v\_source* adding an edge to this vertex  
*v\_dest* adding this edge to the vertex  
*w* weight of the edge

##### Returns:

0 for success. other values for failure

adding an edge to an existed vertex

##### 4.1.2.2 int `graph_add_vertex` (graph \* g, void \* data, SRT\_CONCEPT type, const char \* label)

adding a vertex to the graph

##### Parameters:

*g* a pointer to an initialized graph object  
*data* a pointer to the vertex's relevant data  
*type* type of the vertex  
*label* label of the vertex

##### Returns:

0 for success. other values for failure

checking existence of the node

#### 4.1.2.3 void graph\_destroy (graph \* g)

completely remove the graph from memory.

**Parameters:**

*g* a pointer to an initialized graph object

**Returns:**

nothing!

#### 4.1.2.4 edge\* graph\_edge\_at (graph \* g, vertex \* v, size\_t index)

retrieving an edge of a vertex noted by its index

**Parameters:**

*g* a pointer to an initialized graph object

*v* the edge of the relevant vertex

*index* index of an edge in the vertex

**Returns:**

a pointer to the vertex for success. NULL for failure.

retrieving an edge of a vertex noted by its index

#### 4.1.2.5 int graph\_init (graph \* g)

initialize a graph object for use.

**Parameters:**

*g* a pointer to a graph object

**Returns:**

0 for success. other values for failure

graph initializing

#### 4.1.2.6 int graph\_remove\_all\_edges (graph \* g, vertex \* v)

removing all edges from a vertex of graph denoted by v

**Parameters:**

*g* a pointer to an initialized graph object

*v* removing the edges from this vertex

**Returns:**

0 for success. other values for failure

removing all edges from a vertex of graph denoted by v

**4.1.2.7 int graph\_remove\_edge (graph \* g, vertex \* v\_source, vertex \* v\_dest)**

removing an edge from a vertex of graph

**Parameters:**

- g* a pointer to an initialized graph object
- v\_source* removing the edge from this vertex
- v\_dest* corresponded edge to be removed

**Returns:**

- 0 for success. other values for failure

removing an edge from a vertex of graph

**4.1.2.8 int graph\_remove\_edge\_at (graph \* g, vertex \* v, size\_t edge\_index)**

removing an edge from a vertex of graph by its index

**Parameters:**

- g* a pointer to an initialized graph object
- v* removing the edge from this vertex
- edge\_index* corresponded edge by index

**Returns:**

- 0 for success. other values for failure

removing an edge from a vertex of graph by the index

**4.1.2.9 int graph\_remove\_vertex (graph \* g, vertex \* v)**

removing a vertex from the graph

**Parameters:**

- g* a pointer to an initialized graph object
- v* the vertex to be removed

**Returns:**

- 0 for success. other values for failure

removing a vertex from the graph

removing all the edges of g connected to vertex v



**4.1.2.10 int graph\_remove\_vertex\_at (graph \* g, size\_t index)**

removing a vertex from the graph by index

**Parameters:**

*g* a pointer to an initialized graph object  
*index* index of vertex in the graph

**Returns:**

0 for success. other values for failure

removing a vertex from the graph by index  
removing all the edges of g connected to vertex v

**4.1.2.11 vertex\* graph\_vertex\_at (graph \* g, size\_t index)**

retrieving a vertex of the graph by index

**Parameters:**

*g* a pointer to an initialized graph object  
*index* index of vertex in the graph

**Returns:**

a pointer to the vertex for success. NULL for failure.

retrieving a vertex of the graph by index

**4.1.2.12 vertex\* graph\_vertex\_by\_label (graph \* g, const char \* label)**

retrieving a vertex of the graph by its label

**Parameters:**

*g* a pointer to an initialized graph object  
*label* label of vertex in the graph

**Returns:**

a pointer to the vertex for success. NULL for failure.

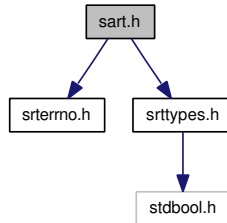
retrieving a vertex of the graph by its label

## 4.2 srt.h File Reference

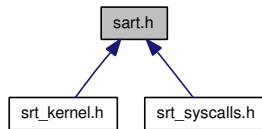
```
#include "srterrno.h"
```

```
#include "srtrtypes.h"
```

Include dependency graph for srt.h:



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



### Data Structures

- struct [srt\\_task\\_attr](#)
- struct [srt\\_job\\_attr](#)

### Typedefs

- typedef `bool(* schd\_checker)(const float u, const u_int n)`

### Functions

- `SRT\_DATA * srt\_data\_create (const char *label, const char *kind, u_int quality)`
- `srt_status srt\_data\_set\_attr (SRT\_DATA *data, u_int quality)`
- `SRT\_RESOURCE * srt\_res\_create (const char *label, bool available)`
- `srt_status srt\_res\_add\_provision (SRT\_RESOURCE *res, SRT\_DATA *data)`
- `srt_status srt\_res\_set\_attr (SRT\_RESOURCE *res, bool available)`
- `SRT\_JOB * srt\_job\_create (const char *label, u_int prominence, SRT\_POLICY policy)`
- `srt_status srt\_job\_add\_data (SRT\_JOB *job, SRT\_DATA *data)`
- `srt_status srt\_job\_add\_task (SRT\_JOB *job, SRT\_TASK *task)`
- `srt_status srt\_job\_set\_attr (SRT\_JOB *job, srt\_job\_attr attr)`
- `srt_status srt\_job\_get\_attr (SRT\_JOB *job, srt\_job\_attr *pattr)`
- `SRT\_DISJUNCTION * srt\_disj\_create (const char *label)`
- `srt_status srt\_disj\_add\_data (SRT\_DISJUNCTION *disj, SRT\_DATA *data)`
- `srt_status srt\_disj\_add\_disj (SRT\_DISJUNCTION *disj, SRT\_DISJUNCTION *disj2)`
- `srt_status srt\_disj\_add\_conj (SRT\_DISJUNCTION *disj, SRT\_CONJUNCTION *conj)`

- `SRT_CONJUNCTION * srt_conj_create` (const char \*label)
- `srt_status srt_conj_add_data` (`SRT_CONJUNCTION *conj`, `SRT_DATA *data`)
- `srt_status srt_conj_add_disj` (`SRT_CONJUNCTION *conj`, `SRT_DISJUNCTION *disj`)
- `srt_status srt_conj_add_conj` (`SRT_CONJUNCTION *conj`, `SRT_CONJUNCTION *conj2`)
- `SRT_TASK * srt_task_create` (const char \*label, u\_int period, u\_int consumption)
- `srt_status srt_task_set_attr` (`SRT_TASK *task`, `srt_task_attr attr`)
- `srt_status srt_task_get_attr` (`SRT_TASK *task`, `srt_task_attr *pattr`)
- `srt_status srt_task_add_disj_input` (`SRT_TASK *task`, `SRT_DISJUNCTION *disj`)
- `srt_status srt_task_add_conj_input` (`SRT_TASK *task`, `SRT_CONJUNCTION *conj`)
- `srt_status srt_task_add_data_input` (`SRT_TASK *task`, `SRT_DATA *data`)
- `srt_status srt_task_add_output` (`SRT_TASK *task`, `SRT_DATA *data`)
- `srt_status srt_task_set_descriptor` (`SRT_TASK *task`, void \*descriptor)
- void \* `srt_get_entity_by_name` (const char \*name)
- `srt_status srt_init` (void)
- `srt_status srt_finish` (void)
- `srt_status srt_chek_kb` (void)
- `srt_status srt_test_solution` (void)
- `srt_status srt_switch_context` (void \*os\_data)
- `srt_status srt_set_sched_checker` (`schd_checker check_func`)
- `srt_status srt_export_kb2dot` (const char \*filename)

### 4.2.1 Detailed Description

SART APIs for initialization, Knowledge base description and manipulation, context switching and scheduling.

### 4.2.2 Typedef Documentation

#### 4.2.2.1 typedef bool(\* schd\_checker)(const float u, const u\_int n)

User defined routine for schedulability test .

#### Parameters:

- u* utilization factor
- n* number of tasks

#### Returns:

true if it's feasible otherwise false.

### 4.2.3 Function Documentation

#### 4.2.3.1 srt\_status srt\_chek\_kb (void)

Checking for the consistency of the SART knowledge-base;

#### Returns:

ERR\_OK if success

#### 4.2.3.2 `srt_status srt_conj_add_conj (SRT_CONJUNCTION * conj, SRT_CONJUNCTION * conj2)`

Add CONJUNCTION entity to a DISJUNCTION

##### Parameters:

*conj* a pointer to [SRT\\_CONJUNCTION](#) structure  
*conj2* a pointer to [SRT\\_CONJUNCTION](#) structure

##### Returns:

ERR\_OK if success

#### 4.2.3.3 `srt_status srt_conj_add_data (SRT_CONJUNCTION * conj, SRT_DATA * data)`

Add DATA entity to a CONJUNCTION

##### Parameters:

*conj* a pointer to [SRT\\_CONJUNCTION](#) structure  
*data* a pointer to [SRT\\_DATA](#) structure

##### Returns:

ERR\_OK if success

#### 4.2.3.4 `srt_status srt_conj_add_disj (SRT_CONJUNCTION * conj, SRT_DISJUNCTION * disj)`

Add DISJUNCTION entity to a CONJUNCTION

##### Parameters:

*conj* a pointer to [SRT\\_CONJUNCTION](#) structure  
*disj* a pointer to [SRT\\_DISJUNCTION](#) structure

##### Returns:

ERR\_OK if success

#### 4.2.3.5 `SRT_CONJUNCTION* srt_conj_create (const char * label)`

Create a CONJUNCTION concept in SART Knowledge Base

##### Parameters:

*label* label

##### Returns:

a pointer to the [SRT\\_CONJUNCTION](#) structure if success; otherwise NULL.

#### 4.2.3.6 SRT\_DATA\* srt\_data\_create (const char \* *label*, const char \* *kind*, u\_int *quality*)

Create a DATA concept in SART Knowledge Base

**Parameters:**

*label* label

*kind* type of data

*quality* quality of data

**Returns:**

a pointer to the [SRT\\_DATA](#) structure if success; otherwise NULL.

#### 4.2.3.7 srt\_status srt\_data\_set\_attr (SRT\_DATA \* *data*, u\_int *quality*)

Set the attribute of a DATA

**Parameters:**

*data* a pointer to [SRT\\_DATA](#) structure

*quality* quality of data

**Returns:**

ERR\_OK if success

#### 4.2.3.8 srt\_status srt\_disj\_add\_conj (SRT\_DISJUNCTION \* *disj*, SRT\_CONJUNCTION \* *conj*)

Add CONJUNCTION entity to a DISJUNCTION

**Parameters:**

*disj* a pointer to [SRT\\_DISJUNCTION](#) structure

*conj* a pointer to [SRT\\_CONJUNCTION](#) structure

**Returns:**

ERR\_OK if success

#### 4.2.3.9 srt\_status srt\_disj\_add\_data (SRT\_DISJUNCTION \* *disj*, SRT\_DATA \* *data*)

Add DATA entity to a DISJUNCTION

**Parameters:**

*disj* a pointer to [SRT\\_DISJUNCTION](#) structure

*data* a pointer to [SRT\\_DATA](#) structure

**Returns:**

ERR\_OK if success

#### 4.2.3.10 `srt_status srt_disj_add_disj (SRT_DISJUNCTION * disj, SRT_DISJUNCTION * disj2)`

Add DISJUNCTION entity to a DISJUNCTION

**Parameters:**

*disj* a pointer to [SRT\\_DISJUNCTION](#) structure

*disj2* a pointer to [SRT\\_DISJUNCTION](#) structure

**Returns:**

ERR\_OK if success

#### 4.2.3.11 `SRT_DISJUNCTION* srt_disj_create (const char * label)`

Create a DISJUNCTION concept in SART Knowledge Base

**Parameters:**

*label* label

**Returns:**

a pointer to the [SRT\\_DISJUNCTION](#) structure if success; otherwise NULL.

#### 4.2.3.12 `srt_status srt_export_kb2dot (const char * filename)`

Export SART knowledge-base to a DOT graph file.

**Parameters:**

*filename* name of the output DOT file.

**Returns:**

ERR\_OK if success.

#### 4.2.3.13 `srt_status srt_finish (void)`

Ending SART;

**Returns:**

ERR\_OK if success

NOTE: [srt\\_finish\(\)](#) must be called at the end of you program.

#### 4.2.3.14 `void* srt_get_entity_by_name (const char * name)`

Retrieve a SART entity by its name;

**Parameters:**

*name* label of the entity

**Returns:**

a pointer to a SART entity if success otherwise NULL;

NOTE: User must be aware of the type of the entity requested by its name!

**4.2.3.15 srt\_status srt\_init (void)**

Initializing SART;

**Returns:**

ERR\_OK if success

NOTE: [srt\\_init\(\)](#) must be called before any other SART APIs.

**4.2.3.16 srt\_status srt\_job\_add\_data (SRT\_JOB \**job*, SRT\_DATA \**data*)**

Add DATA necessity to a JOB

**Parameters:**

*job* a pointer to [SRT\\_JOB](#) structure

*data* a pointer to [SRT\\_DATA](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.17 srt\_status srt\_job\_add\_task (SRT\_JOB \**job*, SRT\_TASK \**task*)**

Add TASK necessity to a JOB

**Parameters:**

*job* a pointer to [SRT\\_JOB](#) structure

*task* a pointer to [SRT\\_TASK](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.18 SRT\_JOB\* srt\_job\_create (const char \**label*, u\_int *prominence*, SRT\_POLICY *policy*)**

Create a JOB concept in SART Knowledge Base

**Parameters:**

*label* label

*prominence* prominence or priority of JOB

*policy* ploicy concerning reasoning of job

**Returns:**

a pointer to the [SRT\\_JOB](#) structure if success; otherwise NULL.

**4.2.3.19** `srt_status srt_job_get_attr (SRT_JOB *job, srt_job_attr *pattr)`

Set the attribute of Job

**Parameters:**

*job* a pointer to [SRT\\_JOB](#) structure

*pattr* a pointer to [srt\\_job\\_attr](#) structure to save the attributes

**Returns:**

ERR\_OK if success

**4.2.3.20** `srt_status srt_job_set_attr (SRT_JOB *job, srt_job_attr attr)`

Set the attribute of Job

**Parameters:**

*job* a pointer to [SRT\\_JOB](#) structure

*attr* [srt\\_job\\_attr](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.21** `srt_status srt_res_add_provision (SRT_RESOURCE *res, SRT_DATA *data)`

Add provision to a RESOURCE

**Parameters:**

*res* a pointer to [SRT\\_RESOURCE](#) structure

*data* a pointer to [SRT\\_DATA](#) structure

**Returns:**

ERR\_OK if success



**4.2.3.22 SRT\_RESOURCE\* srt\_res\_create (const char \* label, bool available)**

Create a RESOURCE concept in SART Knowledge Base

**Parameters:**

*label* label

*available* availability of resource

**Returns:**

a pointer to the [SRT\\_RESOURCE](#) structure if success; otherwise NULL.

**4.2.3.23 srt\_status srt\_res\_set\_attr (SRT\_RESOURCE \* res, bool available)**

Set availability attribute of a resource

**Parameters:**

*res* a pointer to [SRT\\_RESOURCE](#) structure

*available* attribute of resource

**Returns:**

ERR\_OK if success

**4.2.3.24 srt\_status srt\_set\_sched\_checker (schd\_checker check\_func)**

Set a user defined routine for schedulability test.

**Parameters:**

*check\_func* a pointer to schd\_checker function

**Returns:**

ERR\_OK if success.

**4.2.3.25 srt\_status srt\_switch\_context (void \* os\_data)**

Reason the current SART knowledge-base and switch to new context if any solution exists.

**Parameters:**

*os\_data* OS dependent required data for low-level task manipulating. please refer to the document for the specific OS in the /arch folder.

**Returns:**

ERR\_OK if success

**4.2.3.26** `srt_status srt_task_add_conj_input (SRT_TASK * task, SRT_CONJUNCTION * conj)`

Add CONJUNCTION entity to the input of the task

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*conj* a pointer to [SRT\\_CONJUNCTION](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.27** `srt_status srt_task_add_data_input (SRT_TASK * task, SRT_DATA * data)`

Add DATA entity to the input of the task

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*data* a pointer to [SRT\\_DATA](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.28** `srt_status srt_task_add_disj_input (SRT_TASK * task, SRT_DISJUNCTION * disj)`

Add DISJUNCTION entity to the input of the task

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*disj* a pointer to [SRT\\_DISJUNCTION](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.29** `srt_status srt_task_add_output (SRT_TASK * task, SRT_DATA * data)`

Add Data entity to the output of a task

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*data* a pointer to [SRT\\_DATA](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.30 SRT\_TASK\* srt\_task\_create (const char \* *label*, u\_int *period*, u\_int *consumption*)**

Create a TASK concept in SART Knowledge Base

**Parameters:**

*label* label  
*period* period of the periodic task  
*consumption* task worst case execution time

**Returns:**

a pointer to the [SRT\\_TASK](#) structure if success; otherwise NULL.

**4.2.3.31 srt\_status srt\_task\_get\_attr (SRT\_TASK \* *task*, srt\_task\_attr \* *pattr*)**

Retrieve the attribute of a TASK

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*pattr* a pointer to [srt\\_task\\_attr](#) structure to save the attributes

**Returns:**

ERR\_OK if success

**4.2.3.32 srt\_status srt\_task\_set\_attr (SRT\_TASK \* *task*, srt\_task\_attr *attr*)**

Set the attribute of TASK

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*attr* [srt\\_task\\_attr](#) structure

**Returns:**

ERR\_OK if success

**4.2.3.33 srt\_status srt\_task\_set\_descriptor (SRT\_TASK \* *task*, void \* *descriptor*)**

Set the OS dependent task descriptor. This descriptor will be used by SART core to make relation between TASK concepts defined in the SART knowledge-base and real operating system tasks.

**Parameters:**

*task* a pointer to [SRT\\_TASK](#) structure  
*descriptor* a pointer to a OS dependent task descriptor. for detailed information about the type and structure of this descriptor, please refer to the document for the specific OS in the /arch folder.

**Returns:**

ERR\_OK if success

**4.2.3.34 srt\_status srt\_test\_solution (void)**

Checks whether any solution exist in SART knowledge-base by the reasoner or not.

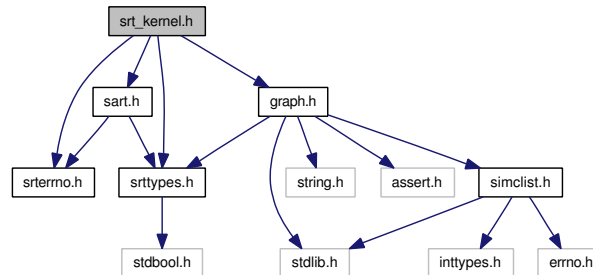
**Returns:**

ERR\_OK if success

## 4.3 srt\_kernel.h File Reference

```
#include "srtpypes.h"
#include "srterrno.h"
#include "graph.h"
#include "sart.h"
```

Include dependency graph for srt\_kernel.h:



### Data Structures

- struct `srt_data_t`
- struct `srt_conjunction_t`
- struct `srt_disjunction_t`
- struct `srt_task_t`
- struct `srt_resource_t`
- struct `srt_job_t`
- struct `sart_kb`

### Functions

- `srt_status update_kb_graph` (void)
- `bool rm_sched_checker` (const float u, const u\_int n)
- `srt_status select_task_set` (void)
- `float min_dependency_path` (graph \*g, vertex \*initial, list\_t \*path)
- `u_int count_common_task` (list\_t \*l1, list\_t \*l2, SRT\_CONCEPT rt)

### Variables

- `sart_kb srt_kb`

#### 4.3.1 Detailed Description

#### 4.3.2 Function Documentation

##### 4.3.2.1 `u_int count_common_task` (list\_t \*l1, list\_t \*l2, SRT\_CONCEPT rt)

count the number of common tasks in two lists

#### 4.3.2.2 float min\_dependency\_path (graph \* g, vertex \* initial, list\_t \* path)

TODO: 1) Needs to find the second and other feasibly schedulable task sets. 2) For optimization purpose, change all the list exploration by iteration method using list\_iterator\_start/next()

If this is an unavailable resource or a disabled task return with cost = -1 which says to not explore into this branch any more.

We are visiting this node. "visited" flag MUST always be cleared before returning from the routine

if this is a leaf, return with the cost = zero

Preventing getting stuck in a loop Do not meet a node which has already been visited.

A Soft-constrained task has no cost!!!

Explore deeper into graph by going toward one branch

The return cost of bottom leaves should be positive. A negative value means a task constraint or resource availability are not met.

sum up costs of the branches of a conjunction node.

chose the minimum branch.

In case of having two different path with the same costs, we should chose the path which most of its tasks has already been chosen for the other job.

TODO: Needs to be changed to support soft real-time tasks too.

An empty result occurs only and only if the constraint for the bottom leaves are not met. Therefore the current leaf should also be ignored.

adding initial job to the path

Adding tasks to the main path

#### 4.3.2.3 bool rm\_sched\_checker (const float u, const u\_int n)

Default schedubility checker routine (Rate Monotonic).

#### 4.3.2.4 srt\_status select\_task\_set (void)

Selecting the most suitable task set to perform user jobs. TODO: 1) MAX\_QUALITY policy for the job has not been implemented. [select\\_task\\_set\(\)](#) and [min\\_dependency\\_path\(\)](#) need to be modified to support different job policies. 2) In order to have more efficient reasoner which allows to check all the combination of the tasks set and aims to schedule most possible hard real-time jobs, first it's needed to find all the possible dependency paths (task set) which are independently schedulable. Then try to find the combination of tasks which allow to schedule more jobs within hard real-time constraint.

checking for the consistency of the KB

defining a list to keep all the nodes in the minimum dependency path including data, con/disjunction and etc.

Adding all the Forced task to the list

If the task set is not feasibly schedulable return with failure!

finding minimum dependency tasks set for each job

saving the beginning position of next task set

negative cost means no feasible path is found!  $\text{max\_cost} = \text{cost} + \text{forced\_cost}$ .

if there is no feasible path for this job, remove it from the task set.

if no feasible path is found, return with failure

checking if its needed to update the task set.

need to update the task\_set list

select tasks which are needed to be deactivated

#### 4.3.2.5 srt\_status update\_kb\_graph (void)

sart core internal routines

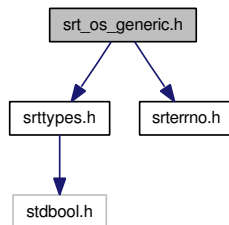
TODO: 1) two data input from different task should not be connected to each other!

## 4.4 srt\_os\_generic.h File Reference

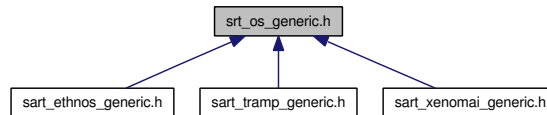
```
#include "sratypes.h"
```

```
#include "srterrno.h"
```

Include dependency graph for srt\_os\_generic.h:



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



### Functions

- void [\\_\\_pre\\_context\\_switch](#) (void \*os\_data)
- void [\\_\\_post\\_context\\_switch](#) (void \*os\_data)
- int [\\_\\_is\\_task\\_active](#) (void \*task\_descriptor, void \*os\_data)
- int [\\_\\_activate\\_task](#) (void \*task\_descriptor, void \*os\_data)
- int [\\_\\_deactivate\\_task](#) (void \*task\_descriptor, void \*os\_data)

#### 4.4.1 Detailed Description

#### 4.4.2 Function Documentation

##### 4.4.2.1 int [\\_\\_activate\\_task](#) (void \* *task\_descriptor*, void \* *os\_data*)

This function will be called by sart to activate a task.

#### Parameters:

*task\_descriptor* pointer to an os task descriptor.

*os\_data* OS dependent data

#### Returns:

0 for success; a non-zero value for failure.



#### 4.4.2.2 int \_\_deactivate\_task (void \* *task\_descriptor*, void \* *os\_data*)

This function will be called by sart to deactivate a task.

**Parameters:**

*task\_descriptor* pointer to an os task descriptor.  
*os\_data* OS dependent data

**Returns:**

0 for success; a non-zero value for failure.

#### 4.4.2.3 int \_\_is\_task\_active (void \* *task\_descriptor*, void \* *os\_data*)

This function will be called by sart to indicate whether a task is activated or not.

**Parameters:**

*task\_descriptor* pointer to an os task descriptor  
*os\_data* OS dependent data

**Returns:**

0 if the task is not activated; otherwise a non-zero value should be returned.

#### 4.4.2.4 void \_\_post\_context\_switch (void \* *os\_data*)

Finalizing context switch. this function will be called by sart context switch routine after activating/deactivating task sets. You might put any post processing and configuring OS.

**Parameters:**

*os\_data* OS dependent data

#### 4.4.2.5 void \_\_pre\_context\_switch (void \* *os\_data*)

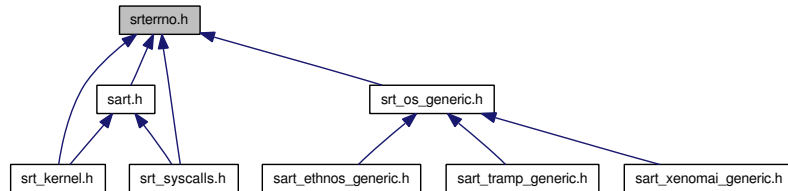
This header file includes the os (architecture) dependent routines which are needed to implemented as wrappers to SART in ./arch folder. Preparing for switching context. this function will be called by sart context switch routine before activating/deactivating task sets. You might put any initializing or configuring OS before actually switching to new context.

**Parameters:**

*os\_data* OS dependent data

## 4.5 srterrno.h File Reference

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



### Defines

- #define `__ASSERT(_cond)`
- #define `__CHECK_NULLPTR(_ptr)` if( !\_ptr ) return ERR\_NULL;
- #define `__DEBUG_MSG(msg)`
- #define `ERR_OK` 0
- #define `ERR_NULL` 1
- #define `ERR_FAILED` 2
- #define `ERR_NEED_UPDATE` 3
- #define `ERR_NO_DATAKIND` 100
- #define `ERR_NO_CONCITENCY` 101
- #define `ERR_NO_TASK_DESCRIPTOR` 102

### 4.5.1 Detailed Description

### 4.5.2 Define Documentation

#### 4.5.2.1 #define `__ASSERT(_cond)`

Enable debug mode from here

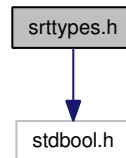
#### 4.5.2.2 #define `ERR_NO_DATAKIND` 100

error codes representing knowledge base constraints

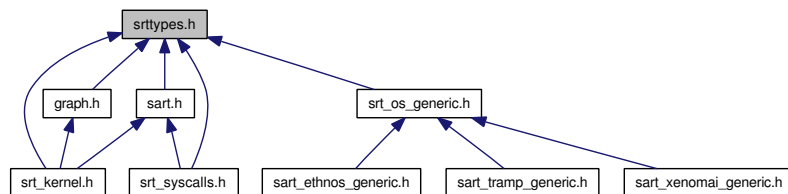
## 4.6 srttypes.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for srttypes.h:



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



### Data Structures

- struct [SRT\\_TASK](#)
- struct [SRT\\_DATA](#)
- struct [SRT\\_RESOURCE](#)
- struct [SRT\\_JOB](#)
- struct [SRT\\_CONJUNCTION](#)
- struct [SRT\\_DISJUNCTION](#)

### Defines

- #define `__sart_name__` "SART"
- #define `true` (int)1
- #define `false` (int)0
- #define `NULL` ((void\*)0)
- #define `DATA_KIND_SIZE` 128
- #define `LABLE_SIZE` 128
- #define `SYSCALL`(rettype) rettype
- #define `OSROUTINE`(rettype) rettype

### Typedefs

- typedef unsigned int `bool`
- typedef unsigned char `byte`
- typedef unsigned char `u_char`
- typedef unsigned int `u_int`

- typedef unsigned long **u\_long**
- typedef unsigned long long **u\_llong**
- typedef u\_int **srt\_id**
- typedef u\_int **srt\_status**

## Enumerations

- enum **SRT\_CONCEPT** {  
    **CONSTRAINT, DATA, TASK, RESOURCE,**  
    **JOB, HARD, SOFT, FORCED,**  
    **DISABLED, CONJUNCTION, DISJUNCTION }**
- enum **SRT\_POLICY** { **MIN\_COST, MAX\_QUALITY }**

### 4.6.1 Detailed Description

### 4.6.2 Define Documentation

#### 4.6.2.1 #define OSROUTINE(rettype) rettype

OS dependent routine

#### 4.6.2.2 #define SYSCALL(rettype) rettype

SART system call routine

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 enum SRT\_CONCEPT

SART KB related concepts

#### 4.6.3.2 enum SRT\_POLICY

SART reasoning policy

# Index

`__ASSERT`  
    `srterrno.h`, 36

`__activate_task`  
    `srt_os_generic.h`, 34

`__deactivate_task`  
    `srt_os_generic.h`, 34

`__is_task_active`  
    `srt_os_generic.h`, 35

`__post_context_switch`  
    `srt_os_generic.h`, 35

`__pre_context_switch`  
    `srt_os_generic.h`, 35

`constraint_disabled`  
    `srt_task_attr`, 14

`constraint_forced`  
    `srt_task_attr`, 14

`constraint_hard`  
    `srt_task_attr`, 14

`count_common_task`  
    `srt_kernel.h`, 31

`disabled`  
    `srt_job_attr`, 11

`ERR_NO_DATAKIND`  
    `srterrno.h`, 36

`graph.h`, 15  
    `graph_add_edge`, 16  
    `graph_add_vertex`, 16  
    `graph_destroy`, 16  
    `graph_edge_at`, 17  
    `graph_init`, 17  
    `graph_remove_all_edges`, 17  
    `graph_remove_edge`, 17  
    `graph_remove_edge_at`, 18  
    `graph_remove_vertex`, 18  
    `graph_remove_vertex_at`, 18  
    `graph_vertex_at`, 19  
    `graph_vertex_by_label`, 19

`graph_add_edge`  
    `graph.h`, 16

`graph_add_vertex`  
    `graph.h`, 16

`graph_destroy`  
    `graph.h`, 16

`graph_edge_at`  
    `graph.h`, 17

`graph_init`  
    `graph.h`, 17

`graph_remove_all_edges`  
    `graph.h`, 17

`graph_remove_edge`  
    `graph.h`, 17

`graph_remove_edge_at`  
    `graph.h`, 18

`graph_remove_vertex`  
    `graph.h`, 18

`graph_remove_vertex_at`  
    `graph.h`, 18

`graph_vertex_at`  
    `graph.h`, 19

`graph_vertex_by_label`  
    `graph.h`, 19

`list_t`, 5

`min_dependency_path`  
    `srt_kernel.h`, 31

`OSROUTINE`  
    `srttypes.h`, 38

`policy`  
    `srt_job_attr`, 11

`prominence`  
    `srt_job_attr`, 11

`rm_sched_checker`  
    `srt_kernel.h`, 32

`sart.h`, 20  
    `schd_checker`, 21  
    `srt_chek_kb`, 21  
    `srt_conj_add_conj`, 21  
    `srt_conj_add_data`, 22  
    `srt_conj_add_disj`, 22  
    `srt_conj_create`, 22  
    `srt_data_create`, 22  
    `srt_data_set_attr`, 23  
    `srt_disj_add_conj`, 23

- srt\_disj\_add\_data, 23
- srt\_disj\_add\_disj, 23
- srt\_disj\_create, 24
- srt\_export\_kb2dot, 24
- srt\_finish, 24
- srt\_get\_entity\_by\_name, 24
- srt\_init, 25
- srt\_job\_add\_data, 25
- srt\_job\_add\_task, 25
- srt\_job\_create, 25
- srt\_job\_get\_attr, 26
- srt\_job\_set\_attr, 26
- srt\_res\_add\_provision, 26
- srt\_res\_create, 26
- srt\_res\_set\_attr, 27
- srt\_set\_sched\_checker, 27
- srt\_switch\_context, 27
- srt\_task\_add\_conj\_input, 27
- srt\_task\_add\_data\_input, 28
- srt\_task\_add\_disj\_input, 28
- srt\_task\_add\_output, 28
- srt\_task\_create, 28
- srt\_task\_get\_attr, 29
- srt\_task\_set\_attr, 29
- srt\_task\_set\_descriptor, 29
- srt\_test\_solution, 29
- sched\_checker
  - sart.h, 21
- select\_task\_set
  - srt\_kernel.h, 32
- srt\_chek\_kb
  - sart.h, 21
- SRT\_CONCEPT
  - srttypes.h, 38
- srt\_conj\_add\_conj
  - sart.h, 21
- srt\_conj\_add\_data
  - sart.h, 22
- srt\_conj\_add\_disj
  - sart.h, 22
- srt\_conj\_create
  - sart.h, 22
- SRT\_CONJUNCTION, 7
- SRT\_DATA, 8
- srt\_data\_create
  - sart.h, 22
- srt\_data\_set\_attr
  - sart.h, 23
- srt\_disj\_add\_conj
  - sart.h, 23
- srt\_disj\_add\_data
  - sart.h, 23
- srt\_disj\_add\_disj
  - sart.h, 23
- srt\_disj\_create
  - sart.h, 24
- SRT\_DISJUNCTION, 9
- srt\_export\_kb2dot
  - sart.h, 24
- srt\_finish
  - sart.h, 24
- srt\_get\_entity\_by\_name
  - sart.h, 24
- srt\_init
  - sart.h, 25
- SRT\_JOB, 10
- srt\_job\_add\_data
  - sart.h, 25
- srt\_job\_add\_task
  - sart.h, 25
- srt\_job\_attr, 11
  - disabled, 11
  - policy, 11
  - prominence, 11
- srt\_job\_create
  - sart.h, 25
- srt\_job\_get\_attr
  - sart.h, 26
- srt\_job\_set\_attr
  - sart.h, 26
- srt\_kernel.h, 31
  - count\_common\_task, 31
  - min\_dependency\_path, 31
  - rm\_sched\_checker, 32
  - select\_task\_set, 32
  - update\_kb\_graph, 33
- srt\_os\_generic.h, 34
  - \_\_activate\_task, 34
  - \_\_deactivate\_task, 34
  - \_\_is\_task\_active, 35
  - \_\_post\_context\_switch, 35
  - \_\_pre\_context\_switch, 35
- SRT\_POLICY
  - srttypes.h, 38
- srt\_res\_add\_provision
  - sart.h, 26
- srt\_res\_create
  - sart.h, 26
- srt\_res\_set\_attr
  - sart.h, 27
- SRT\_RESOURCE, 12
- srt\_set\_sched\_checker
  - sart.h, 27
- srt\_switch\_context
  - sart.h, 27
- SRT\_TASK, 13
- srt\_task\_add\_conj\_input
  - sart.h, 27

---

srt\_task\_add\_data\_input  
sart.h, 28

srt\_task\_add\_disj\_input  
sart.h, 28

srt\_task\_add\_output  
sart.h, 28

srt\_task\_attr, 14  
constraint\_disabled, 14  
constraint\_forced, 14  
constraint\_hard, 14

srt\_task\_create  
sart.h, 28

srt\_task\_get\_attr  
sart.h, 29

srt\_task\_set\_attr  
sart.h, 29

srt\_task\_set\_descriptor  
sart.h, 29

srt\_test\_solution  
sart.h, 29

srterrno.h, 36  
\_\_ASSERT, 36  
ERR\_NO\_DATAKIND, 36

srttypes.h, 37  
OSROUTINE, 38  
SRT\_CONCEPT, 38  
SRT\_POLICY, 38  
SYSCALL, 38

SYSCALL  
srttypes.h, 38

update\_kb\_graph  
srt\_kernel.h, 33