

JAMscript-ESP32-port

1.0.0

Generated by Doxygen 1.11.0

1 Todo List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 <code>_cnode_t</code> Struct Reference	7
4.1.1 Field Documentation	7
4.1.1.1 <code>core_state</code>	7
4.1.1.2 <code>initialized</code>	7
4.1.1.3 <code>node_id</code>	7
4.1.1.4 <code>system_manager</code>	7
4.1.1.5 <code>zenoh</code>	8
4.2 <code>_corestate_t</code> Struct Reference	8
4.2.1 Field Documentation	8
4.2.1.1 <code>device_id</code>	8
4.2.1.2 <code>serial_num</code>	8
4.3 <code>_system_manager_t</code> Struct Reference	8
4.3.1 Field Documentation	8
4.3.1.1 <code>_connection_attempts</code>	8
4.3.1.2 <code>got_ip_event_handle</code>	8
4.3.1.3 <code>wifi_any_event_handle</code>	8
4.3.1.4 <code>wifi_connection</code>	9
4.4 <code>_zenoh_t</code> Struct Reference	9
4.4.1 Field Documentation	9
4.4.1.1 <code>z_pub</code>	9
4.4.1.2 <code>z_session</code>	9
4.4.1.3 <code>z_sub</code>	9
5 File Documentation	11
5.1 <code>inc/cnode.h</code> File Reference	11
5.1.1 Typedef Documentation	11
5.1.1.1 <code>cnode_t</code>	11
5.1.2 Function Documentation	11
5.1.2.1 <code>cnode_destroy()</code>	11
5.1.2.2 <code>cnode_init()</code>	12
5.1.2.3 <code>cnode_start()</code>	12
5.1.2.4 <code>cnode_stop()</code>	12
5.2 <code>cnode.h</code>	13
5.3 <code>inc/core.h</code> File Reference	13
5.3.1 Typedef Documentation	13

5.3.1.1 corestate_t	13
5.3.2 Function Documentation	13
5.3.2.1 core_destroy()	13
5.3.2.2 core_init()	14
5.3.2.3 core_setup()	14
5.4 core.h	14
5.5 inc/system_manager.h File Reference	15
5.5.1 Typedef Documentation	15
5.5.1.1 system_manager_t	15
5.5.2 Function Documentation	15
5.5.2.1 system_manager_destroy()	15
5.5.2.2 system_manager_init()	15
5.5.2.3 system_manager_wifi_init()	15
5.6 system_manager.h	16
5.7 inc/zenoh.h File Reference	16
5.7.1 Typedef Documentation	17
5.7.1.1 zenoh_callback_t	17
5.7.1.2 zenoh_t	17
5.7.2 Function Documentation	17
5.7.2.1 zenoh_declare_pub()	17
5.7.2.2 zenoh_declare_sub()	17
5.7.2.3 zenoh_destroy()	18
5.7.2.4 zenoh_init()	18
5.7.2.5 zenoh_publish()	18
5.7.2.6 zenoh_scout()	19
5.7.2.7 zenoh_start_lease_task()	19
5.7.2.8 zenoh_start_read_task()	19
5.8 zenoh.h	20
Index	21

Chapter 1

Todo List

Global `cnode_stop` (`cnode_t *cn`)

should we also stop wifi activity? then we would have to start wifi activity in `cnode_start`

Global `system_manager_wifi_init` (`system_manager_t *system_manager`)

How to set the SSID and Password for the wifi network?

Global `zenoh_init` ()

What configuration do we want for the zenoh session? Peer to peer, client?

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_cnode_t	7
_corestate_t	8
_system_manager_t	8
_zenoh_t	9

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

inc/ cnode.h	11
inc/ core.h	13
inc/ system_manager.h	15
inc/ zenoh.h	16

Chapter 4

Data Structure Documentation

4.1 `_cnode_t` Struct Reference

Data Fields

- `system_manager_t * system_manager`
- `char * node_id`
- `zenoh_t * zenoh`
- `corestate_t * core_state`
- `bool initialized`

4.1.1 Field Documentation

4.1.1.1 `core_state`

`corestate_t* core_state`

4.1.1.2 `initialized`

`bool initialized`

4.1.1.3 `node_id`

`char* node_id`

4.1.1.4 `system_manager`

`system_manager_t* system_manager`

4.1.1.5 zenoh

`zenoh_t* zenoh`

4.2 __corestate_t Struct Reference

Data Fields

- char * [device_id](#)
- int [serial_num](#)

4.2.1 Field Documentation

4.2.1.1 device_id

`char* device_id`

4.2.1.2 serial_num

`int serial_num`

4.3 __system_manager_t Struct Reference

Data Fields

- int [_connection_attempts](#)
- bool [wifi_connection](#)
- [esp_event_handler_instance_t](#) [wifi_any_event_handle](#)
- [esp_event_handler_instance_t](#) [got_ip_event_handle](#)

4.3.1 Field Documentation

4.3.1.1 _connection_attempts

`int _connection_attempts`

4.3.1.2 got_ip_event_handle

`esp_event_handler_instance_t got_ip_event_handle`

4.3.1.3 wifi_any_event_handle

`esp_event_handler_instance_t wifi_any_event_handle`

4.3.1.4 wifi_connection

```
bool wifi_connection
```

4.4 _zenoh_t Struct Reference

Data Fields

- z_owned_publisher_t * [z_pub](#)
- z_owned_subscriber_t * [z_sub](#)
- z_owned_session_t * [z_session](#)

4.4.1 Field Documentation

4.4.1.1 z_pub

```
z_owned_publisher_t* z_pub
```

4.4.1.2 z_session

```
z_owned_session_t* z_session
```

4.4.1.3 z_sub

```
z_owned_subscriber_t* z_sub
```


Chapter 5

File Documentation

5.1 inc/cnode.h File Reference

Data Structures

- struct [_cnode_t](#)

Typedefs

- typedef struct [_cnode_t](#) [cnode_t](#)

Functions

- [cnode_t](#) * [cnode_init](#) (int argc, char **argv)
Constructor. Initiates the cnode structure and initiates all of its components. E.g., we call [system_manager_init\(\)](#), [zenoh_init\(\)](#), ...
- void [cnode_destroy](#) ([cnode_t](#) *cn)
Frees memory allocated during [cnode_init\(\)](#)
- bool [cnode_start](#) ([cnode_t](#) *cn)
Starts listening thread.
- bool [cnode_stop](#) ([cnode_t](#) *cn)
Stops listening thread.

5.1.1 Typedef Documentation

5.1.1.1 cnode_t

```
typedef struct \_cnode\_t cnode\_t
```

5.1.2 Function Documentation

5.1.2.1 cnode_destroy()

```
void cnode_destroy (  
    cnode\_t * cn)
```

Frees memory allocated during [cnode_init\(\)](#)

Parameters

<i>cn</i>	- pointer to <code>cnode_t</code> struct
-----------	--

5.1.2.2 cnode_init()

```
cnode_t * cnode_init (
    int argc,
    char ** argv)
```

Constructor. Initiates the cnode structure and initiates all of its components. E.g., we call [system_manager_init\(\)](#), [zenoh_init\(\)](#), ...

Parameters

<i>argc</i>	- cmd line argument count
<i>argv</i>	- cmd line args

Returns

pointer to `cnode_t` struct

5.1.2.3 cnode_start()

```
bool cnode_start (
    cnode_t * cn)
```

Starts listening thread.

Parameters

<i>cn</i>	- pointer to <code>cnode_t</code> struct
-----------	--

5.1.2.4 cnode_stop()

```
bool cnode_stop (
    cnode_t * cn)
```

Stops listening thread.

Parameters

<i>cn</i>	- pointer to <code>cnode_t</code> struct
-----------	--

Todo should we also stop wifi activity? then we would have to start wifi activity in `cnode_start`

5.2 cnode.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __CNODE_H__
00002 #define __CNODE_H__
00003
00004 #include "zenoh.h"
00005 #include "core.h"
00006 #include "system_manager.h"
00007 #include "utils.h"
00008
00009 /* STRUCTS & TYPEDEFS */
00010 typedef struct _cnode_t
00011 {
00012     system_manager_t* system_manager;
00013     char* node_id;
00014     zenoh_t* zenoh;
00015     corestate_t* core_state;
00016     bool initialized;
00017 } cnode_t;
00018
00019 /* FUNCTION PROTOTYPES */
00020
00021 cnode_t* cnode_init(int argc, char** argv);
00022
00023 void cnode_destroy(cnode_t* cn);
00024
00025 bool cnode_start(cnode_t* cn);
00026
00027 bool cnode_stop(cnode_t* cn);
00028 #endif

```

5.3 inc/core.h File Reference

Data Structures

- [struct _corestate_t](#)

Typedefs

- [typedef struct _corestate_t corestate_t](#)

Functions

- [corestate_t* core_init](#) (int serialNum)
Constructor. Initiates the core. Calls [core_setup\(\)](#) to generate serial & node ID.
- void [core_destroy](#) ([corestate_t](#)* cs)
Frees memory allocated during [core_init\(\)](#)
- void [core_setup](#) ([corestate_t](#)* cs)
Does the UUID4 generation (for node ID) and stores serial & node ID into flash memory.

5.3.1 Typedef Documentation

5.3.1.1 corestate_t

```
typedef struct _corestate_t corestate_t
```

5.3.2 Function Documentation

5.3.2.1 core_destroy()

```
void core_destroy (
    corestate_t * cs)
```

Frees memory allocated during [core_init\(\)](#)

Parameters

<code>cs</code>	pointer to <code>corestate_t</code> struct
-----------------	--

5.3.2.2 core_init()

```
corestate_t * core_init (
    int serialnum)
```

Constructor. Initiates the core. Calls [core_setup\(\)](#) to generate serial & node ID.

Parameters

<code>serialnum</code>	Serial number of the node
------------------------	---------------------------

Returns

pointer to `corestate_t` struct

5.3.2.3 core_setup()

```
void core_setup (
    corestate_t * cs)
```

Does the UUID4 generation (for node ID) and stores serial & node ID into flash memory.

Parameters

<code>cs</code>	pointer to <code>corestate_t</code> struct
-----------------	--

5.4 core.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __CORE_H__
00002 #define __CORE_H__
00003
00004 #include "utils.h"
00005
00006 /* STRUCTS & TYPEDEFS */
00007 typedef struct _corestate_t
00008 {
00009     char *device_id;
00010     int serial_num;
00011     // int default_port;
00012 } corestate_t;
00013
00014 /* FUNCTION PROTOTYPES */
00015
00021 corestate_t *core_init( int serialnum);
00022
00027 void core_destroy(corestate_t *cs);
00028
00033 void core_setup(corestate_t *cs);
00034
00035 #endif
```

5.5 inc/system_manager.h File Reference

Data Structures

- struct [_system_manager_t](#)

Typedefs

- typedef struct [_system_manager_t](#) [system_manager_t](#)

Functions

- [system_manager_t * system_manager_init \(\)](#)
Constructor. Initializes the system manager.
- bool [system_manager_destroy \(system_manager_t *system_manager\)](#)
Frees memory associated with the system_manager_t struct.
- bool [system_manager_wifi_init \(system_manager_t *system_manager\)](#)
Initializes the Wifi module and connects to a preset network.

5.5.1 Typedef Documentation

5.5.1.1 system_manager_t

```
typedef struct _system_manager_t system_manager_t
```

5.5.2 Function Documentation

5.5.2.1 system_manager_destroy()

```
bool system_manager_destroy (
    system_manager_t * system_manager)
```

Frees memory associated with the system_manager_t struct.

Parameters

<i>system_manager</i>	pointer to system_manager_t struct
-----------------------	------------------------------------

5.5.2.2 system_manager_init()

```
system_manager_t * system_manager_init ()
```

Constructor. Initializes the system manager.

Returns

pointer to system_manager_t struct

5.5.2.3 system_manager_wifi_init()

```
bool system_manager_wifi_init (
    system_manager_t * system_manager)
```

Initializes the Wifi module and connects to a preset network.

Return values

<i>true</i>	If wifi initiation successful
<i>false</i>	If error occurred during wifi init

Todo How to set the SSID and Password for the wifi network?

5.6 system_manager.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __SYSTEM_MANAGER_H__
00002 #define __SYSTEM_MANAGER_H__
00003
00004 #include <esp_event.h>
00005
00006 /* STRUCTS & TYPEDEFS */
00007 typedef struct _system_manager_t
00008 {
00009     // add more info if needed
00010     int _connection_attempts;
00011     bool wifi_connection;
00012
00013     esp_event_handler_instance_t wifi_any_event_handle;
00014     esp_event_handler_instance_t got_ip_event_handle;
00015
00016 } system_manager_t;
00017
00018 /* FUNCTION PROTOTYPES */
00019
00024 system_manager_t* system_manager_init();
00025
00030 bool system_manager_destroy(system_manager_t* system_manager);
00031
00038 bool system_manager_wifi_init(system_manager_t* system_manager);
00039 #endif

```

5.7 inc/zenoh.h File Reference

Data Structures

- struct [_zenoh_t](#)

Typedefs

- typedef struct [_zenoh_t](#) [zenoh_t](#)
- typedef void(* [zenoh_callback_t](#)) (z_loaned_sample_t *, void *)

Functions

- [zenoh_t](#) * [zenoh_init](#) ()
Constructor. Initializes zenoh objects and starts a Zenoh session.
- void [zenoh_destroy](#) ([zenoh_t](#) *zenoh)
Frees memory associated with the zenoh_t struct.
- bool [zenoh_scout](#) ()
Scouts for JNodes. Note that JNodes must be using Zenoh.
- bool [zenoh_declare_sub](#) ([zenoh_t](#) *zenoh, const char *key_expression, [zenoh_callback_t](#) *callback)

- *Declare a zenoh subscriber on a specific topic. Assign callback function.*
 • bool `zenoh_declare_pub` (`zenoh_t` *zenoh, const char *key_expression)
- *Declare a zenoh publisher on a specific topic.*
 • void `zenoh_start_read_task` (`zenoh_t` *zenoh)
 - *Start the zenoh read task by calling `zp_start_read_task()`*
- void `zenoh_start_lease_task` (`zenoh_t` *zenoh)
 - *Start the zenoh lease task by calling `zp_start_lease_task()`*
- bool `zenoh_publish` (`zenoh_t` *zenoh, const char *message)
 - *Publish a message over zenoh.*

5.7.1 Typedef Documentation

5.7.1.1 zenoh_callback_t

```
typedef void(* zenoh_callback_t) (z_loaned_sample_t *, void *)
```

5.7.1.2 zenoh_t

```
typedef struct _zenoh_t zenoh_t
```

5.7.2 Function Documentation

5.7.2.1 zenoh_declare_pub()

```
bool zenoh_declare_pub (
    zenoh_t * zenoh,
    const char * key_expression)
```

Declare a zenoh publisher on a specific topic.

Parameters

<i>zenoh</i>	pointer to <code>zenoh_t</code> struct
<i>key_expression</i>	string describing the 'subscription topic'

Return values

<i>true</i>	If publish declaration returned without error
<i>false</i>	If an error occurred

5.7.2.2 zenoh_declare_sub()

```
bool zenoh_declare_sub (
    zenoh_t * zenoh,
    const char * key_expression,
    zenoh_callback_t * callback)
```

Declare a zenoh subscriber on a specific topic. Assign callback function.

Parameters

<i>zenoh</i>	pointer to zenoh_t struct
<i>key_expression</i>	string describing the 'subscription topic'
<i>callback</i>	pointer to zenoh callback function

Return values

<i>true</i>	If subscription declaration returned without error
<i>false</i>	If an error occurred

5.7.2.3 zenoh_destroy()

```
void zenoh_destroy (  
    zenoh_t * zenoh)
```

Frees memory associated with the zenoh_t struct.

Parameters

<i>zenoh</i>	pointer to zenoh_t struct
--------------	---------------------------

5.7.2.4 zenoh_init()

```
zenoh_t * zenoh_init ()
```

Constructor. Initializes zenoh objects and starts a Zenoh session.

Returns

pointer to zenoh_t struct

Todo What configuration do we want for the zenoh session? Peer to peer, client?

5.7.2.5 zenoh_publish()

```
bool zenoh_publish (  
    zenoh_t * zenoh,  
    const char * message)
```

Publish a message over zenoh.

Parameters

<i>zenoh</i>	pointer to zenoh_t struct
<i>message</i>	string consisting of message

Return values

<i>true</i>	If publish successful
<i>false</i>	If an error occurred

5.7.2.6 zenoh_scout()

```
bool zenoh_scout ()
```

Scouts for JNodes. Note that JNodes must be using Zenoh.

Return values

<i>true</i>	If a JNode is found
<i>false</i>	If a JNode is not found

Note

Can be called even before calling [zenoh_init\(\)](#) as long as wifi has been initiated

5.7.2.7 zenoh_start_lease_task()

```
void zenoh_start_lease_task (  
    zenoh\_t * zenoh)
```

Start the zenoh lease task by calling `zp_start_lease_task()`

Parameters

<i>zenoh</i>	pointer to <code>zenoh_t</code> struct
--------------	--

5.7.2.8 zenoh_start_read_task()

```
void zenoh_start_read_task (  
    zenoh\_t * zenoh)
```

Start the zenoh read task by calling `zp_start_read_task()`

Parameters

<i>zenoh</i>	pointer to <code>zenoh_t</code> struct
--------------	--

5.8 zenoh.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __ZENOH_H__
00002 #define __ZENOH_H__
00003
00004 #include <zenoh-pico.h>
00005 #include "utils.h"
00006
00007 /* STRUCTS & TYPEDEFS */
00008 typedef struct _zenoh_t
00009 {
00010     z_owned_publisher_t z_pub;
00011     z_owned_subscriber_t z_sub;
00012     z_owned_session_t z_session;
00013 } zenoh_t;
00014
00015 typedef void (*zenoh_callback_t)(z_loaned_sample_t*, void*);
00016
00017 /* FUNCTION PROTOTYPES */
00018
00024 zenoh_t* zenoh_init();
00025
00030 void zenoh_destroy(zenoh_t* zenoh);
00031
00038 bool zenoh_scout();
00039
00048 bool zenoh_declare_sub(zenoh_t* zenoh, const char* key_expression, zenoh_callback_t* callback);
00049
00057 bool zenoh_declare_pub(zenoh_t* zenoh, const char* key_expression);
00058
00063 void zenoh_start_read_task(zenoh_t* zenoh); // do we really need this
00064
00069 void zenoh_start_lease_task(zenoh_t* zenoh); // do we really need this
00070
00078 bool zenoh_publish(zenoh_t* zenoh, const char* message);
00079 #endif
```


Index

- [_cnode_t, 7](#)
 - [core_state, 7](#)
 - [initialized, 7](#)
 - [node_id, 7](#)
 - [system_manager, 7](#)
 - [zenoh, 7](#)
- [_connection_attempts](#)
 - [_system_manager_t, 8](#)
- [_corestate_t, 8](#)
 - [device_id, 8](#)
 - [serial_num, 8](#)
- [_system_manager_t, 8](#)
 - [_connection_attempts, 8](#)
 - [got_ip_event_handle, 8](#)
 - [wifi_any_event_handle, 8](#)
 - [wifi_connection, 8](#)
- [_zenoh_t, 9](#)
 - [z_pub, 9](#)
 - [z_session, 9](#)
 - [z_sub, 9](#)
- [cnode.h](#)
 - [cnode_destroy, 11](#)
 - [cnode_init, 12](#)
 - [cnode_start, 12](#)
 - [cnode_stop, 12](#)
 - [cnode_t, 11](#)
- [cnode_destroy](#)
 - [cnode.h, 11](#)
- [cnode_init](#)
 - [cnode.h, 12](#)
- [cnode_start](#)
 - [cnode.h, 12](#)
- [cnode_stop](#)
 - [cnode.h, 12](#)
- [cnode_t](#)
 - [cnode.h, 11](#)
- [core.h](#)
 - [core_destroy, 13](#)
 - [core_init, 14](#)
 - [core_setup, 14](#)
 - [corestate_t, 13](#)
- [core_destroy](#)
 - [core.h, 13](#)
- [core_init](#)
 - [core.h, 14](#)
- [core_setup](#)
 - [core.h, 14](#)
- [core_state](#)
 - [_cnode_t, 7](#)
- [corestate_t](#)
 - [core.h, 13](#)
- [device_id](#)
 - [_corestate_t, 8](#)
- [got_ip_event_handle](#)
 - [_system_manager_t, 8](#)
- [inc/cnode.h, 11, 13](#)
- [inc/core.h, 13, 14](#)
- [inc/system_manager.h, 15, 16](#)
- [inc/zenoh.h, 16, 20](#)
- [initialized](#)
 - [_cnode_t, 7](#)
- [node_id](#)
 - [_cnode_t, 7](#)
- [serial_num](#)
 - [_corestate_t, 8](#)
- [system_manager](#)
 - [_cnode_t, 7](#)
- [system_manager.h](#)
 - [system_manager_destroy, 15](#)
 - [system_manager_init, 15](#)
 - [system_manager_t, 15](#)
 - [system_manager_wifi_init, 15](#)
- [system_manager_destroy](#)
 - [system_manager.h, 15](#)
- [system_manager_init](#)
 - [system_manager.h, 15](#)
- [system_manager_t](#)
 - [system_manager.h, 15](#)
- [system_manager_wifi_init](#)
 - [system_manager.h, 15](#)
- [Todo List, 1](#)
- [wifi_any_event_handle](#)
 - [_system_manager_t, 8](#)
- [wifi_connection](#)
 - [_system_manager_t, 8](#)
- [z_pub](#)
 - [_zenoh_t, 9](#)
- [z_session](#)
 - [_zenoh_t, 9](#)
- [z_sub](#)
 - [_zenoh_t, 9](#)
- [zenoh](#)

- [_cnode_t](#), [7](#)
- [zenoh.h](#)
 - [zenoh_callback_t](#), [17](#)
 - [zenoh_declare_pub](#), [17](#)
 - [zenoh_declare_sub](#), [17](#)
 - [zenoh_destroy](#), [18](#)
 - [zenoh_init](#), [18](#)
 - [zenoh_publish](#), [18](#)
 - [zenoh_scout](#), [19](#)
 - [zenoh_start_lease_task](#), [19](#)
 - [zenoh_start_read_task](#), [19](#)
 - [zenoh_t](#), [17](#)
- [zenoh_callback_t](#)
 - [zenoh.h](#), [17](#)
- [zenoh_declare_pub](#)
 - [zenoh.h](#), [17](#)
- [zenoh_declare_sub](#)
 - [zenoh.h](#), [17](#)
- [zenoh_destroy](#)
 - [zenoh.h](#), [18](#)
- [zenoh_init](#)
 - [zenoh.h](#), [18](#)
- [zenoh_publish](#)
 - [zenoh.h](#), [18](#)
- [zenoh_scout](#)
 - [zenoh.h](#), [19](#)
- [zenoh_start_lease_task](#)
 - [zenoh.h](#), [19](#)
- [zenoh_start_read_task](#)
 - [zenoh.h](#), [19](#)
- [zenoh_t](#)
 - [zenoh.h](#), [17](#)