# JAMscript-ESP32-port

1.0.0

Generated by Doxygen 1.11.0

# 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:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

# 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**

| | |
|---|---|
| *cn* | - pointer to cnode_t 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**

| | |
|---|---|
| *argc* | - cmd line argument count |
| *argv* | - 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**

| | |
|---|---|
| *cn* | - pointer to cnode_t struct |

#### 5.1.2.4 cnode_stop()

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

Stops listening thread.

**Parameters**

| | |
|---|---|
| *cn* | - pointer to cnode_t 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
00028 cnode_t*    cnode_init(int argc, char** argv);
00029
00034 void        cnode_destroy(cnode_t* cn);
00035
00040 bool        cnode_start(cnode_t* cn);
00041
00047 bool        cnode_stop(cnode_t* cn);
00048 #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**

| | |
|---|---|
| *cs* | pointer to corestate_t 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**

| | |
|---|---|
| *serialnum* | 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**

| | |
|---|---|
| *cs* | pointer to corestate_t 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**

| | |
|---|---|
| *system_manager* | 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**

| | |
|---|---|
| *true* | If wifi initiation successful |
| *false* | If error occured 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**

| | |
|---|---|
| *zenoh* | pointer to zenoh_t struct |
| *key_expression* | string describing the 'subscription topic' |

**Return values**

| | |
|---|---|
| *true* | If publish declaration returned without error |
| *false* | If an error occured |

### 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**

| | |
|---|---|
| *zenoh* | pointer to zenoh_t struct |
| *key_expression* | string describing the 'subscription topic' |
| *callback* | pointer to zenoh callback function |

**Return values**

| | |
|---|---|
| *true* | If subscription declaration returned without error |
| *false* | If an error occured |

#### 5.7.2.3 zenoh_destroy()

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

Frees memory associated with the zenoh_t struct.

**Parameters**

| | |
|---|---|
| *zenoh* | 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**

| | |
|---|---|
| *zenoh* | pointer to zenoh_t struct |
| *message* | string consisting of message |

**Return values**

| | |
|---|---|
| *true* | If publish successful |
| *false* | If an error occured |

#### 5.7.2.6 zenoh_scout()

```
bool zenoh_scout ()
```

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

**Return values**

| | |
|---|---|
| *true* | If a JNode is found |
| *false* | 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**

| | |
|---|---|
| *zenoh* | pointer to zenoh_t 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**

| | |
|---|---|
| *zenoh* | pointer to zenoh_t 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