
Zeta Documentation

Release 0.1.0-alpha

Lucas Peixoto, Rodrigo Peixoto

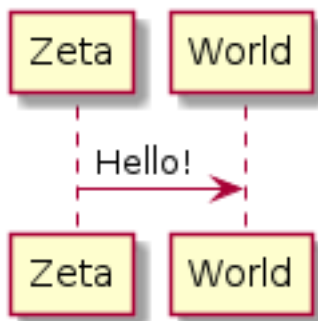
Aug 27, 2021

CONTENTS:

1	Getting started	1
2	Code documentation	3
3	License	31
4	Indices and tables	33
	Index	35

GETTING STARTED

This is the first step to do.



CODE DOCUMENTATION

This is the temporary code documentation.

```
zeta.zeta.Channel : public object
    Represents a channel with a http or https YAML file.
```

Public Functions

```
__init__ ()
__repr__ (self)
```

Public Members

```
name
read_only
on_changed
size
persistent
sem
id
initial_value
message
message_obj
pub_services_obj
sub_services_obj
zeta.zeta.Config : public object
```

Public Functions

```
__init__()
```

Public Members

```
sector_count
```

```
storage_partition
```

```
storage_period
```

```
union data
```

```
#include <zeta.template.h>
```

Public Members

```
zt_data_int8_t s8
```

```
zt_data_uint8_t u8
```

```
zt_data_int16_t s16
```

```
zt_data_uint16_t u16
```

```
zt_data_int32_t s32
```

```
zt_data_uint32_t u32
```

```
zt_data_int64_t s64
```

```
zt_data_uint64_t u64
```

```
zt_data_bytes_t bytes
```

```
zeta.cli.FileFactory class public object
Base class FileFactory class public object generate a file-based
in a template and your respective substitutions.
```

Subclassed by *HeaderFileFactory*, *SourceFileFactory*, *ZetaConf*

Public Functions

```
__init__()
```

```
create_substitutions(self) be implemented by classes inherited.
```

```
:returns: None
```

```
:rtype: None
```

```
generate_file(self) output file with the respective substitutions
assigned in create_substitutions function.
```

```
:returns: None
```

```
:rtype: None
```



```
run(self) the routine responsible for assigns substitutions and
write the output file.
```

```
:returns: None
:rtype: None
```

Public Members

destination_file

template_file

zeta

substitutions

union flag_data

#include <zeta.template.h> Define pendent options that a channel can have.

Public Members

uint8_t pend_persistent

Active represent that channel must be saved in flash by zeta_thread_nvs

uint8_t pend_callback

Active represent that services callbacks from subscribers must be called by zeta_thread

uint8_t on_changed

Active represent that the service callback will be called on change and not on update

uint8_t source_serial_isc

When active indicates the channel was changed by the serial isc (some host process) instead of a target service

struct flag_data::[anonymous] field

uint8_t data

Raw data

```
zeta.zeta_cli.HeaderFileFactory or public FileFactory
Represents a group of files that will
be used by Zeta.
```

Subclassed by *ZetaHeader*

Public Functions

__init__()

class zeta.zeta_isc.IPC

Public Functions

```
__init__()  
create_channels(self)  
digest_packet()  
read_channel(self, cid)  
set_channel(self, cid, msg)
```

Public Static Attributes

```
sem  
channel_changed_queue
```

Private Static Attributes

```
__zeta  
__channels
```

```
class zeta.zeta_isc.IPCChannel
```

Public Functions

```
__init__()  
__repr__(self)  
message_union_field(self)  
create_struct()
```

Public Members

```
metadata  
data
```

```
zeta_pkt_IPHeader{ : public LittleEndianStructure  
    uint8_t channel : 8; /*!> 256 channels available*/  
    uint8_t op : 3; /*!> 0: read, 1: write, 2: read response,  
                        3: write response, 4: update */  
    uint8_t status : 4; /*!> 0: ok, 1: failed */  
    uint8_t has_data : 1; /*!> 0: no data, 1: contains data */  
};
```

Private Static Attributes

`_pack_`
`_fields_`

```
zeta.zeta_pkt.IPHeaderDataInfo { public LittleEndianStructure
    uint8_t crc : 8;  /**!> CCITT 8, polynom 0x07, initial value 0x00 */
    uint8_t size : 8; /**!> data size */
};
```

Private Static Attributes

`_pack_`
`_fields_`

```
zeta.zeta_pkt.IPCPacket : public LittleEndianStructure
```

Public Functions

```
__init__ (self, args, kwargs)
set_data ()
set_data_with_struct (self, struct)
clear_data (self)
from_bytes (cls, raw_data)
to_bytes (self)
data (self)
set_header ()
set_data_info ()
set_header_from_bytes ()
set_data_info_from_bytes ()
header_to_bytes (self)
__repr__ (self)
struct_contents (self, struct)
struct_contents_set (self, struct)
```

Public Static Attributes

`OP_READ`
`OP_WRITE`
`OP_READ_RESPONSE`
`OP_WRITE_RESPONSE`
`OP_UPDATE`
`OP_DEBUG`
`DATA_UNAVAILABLE`
`DATA_AVAILABLE`
`STATUS_OK`
`STATUS_FAILED`

Private Members

`__data`

Private Static Attributes

`_pack_`
`_fields_`

```
class zeta.zeta.Message
    """zeta.zeta.Message is the Message defined by the Zeta yaml file.
```

Public Functions

`__init__()`
`__repr__(self)`

Public Members

`name`
`msg_format`

```
class zeta.zeta_isc.SerialDataHandler
```

Public Functions

```

__init__(self)
append(self, data)
send_command()
digest(self)
run(self)

```

Public Members

```

iqueue
oqueue

```

Public Static Attributes

```

STATE_DIGEST_HEADER_OP
STATE_DIGEST_HEADER_DATA_INFO
STATE_DIGEST_BODY

```

Private Members

```

__buffer
__state
__current_pkt

```

zeta **zeta.Service** **public object** Represents a service in the Zeta YAML file.

Public Functions

```

__init__()

```

Public Members

```

name
priority
stack_size
pub_channels_names
sub_channels_names
pub_channels_obj
sub_channels_obj

```

```
zetacli.SourceFileFactory = public FileFactory
"""Base class for SourceFileFactory classes that will
be used by Zeta.
```

Subclassed by [ZetaSource](#)

Public Functions

```
__init__()
```

```
zeta.messages.TestStringMethods : public TestCase
```

Public Functions

```
test_primitives(self)
```

```
test_struct_simple(self)
```

```
test_union_simple(self)
```

```
test_bitarray_simple(self)
```

```
test_array_simple(self)
```

```
test_complex_message(self)
```

```
zetacli.Zeta = public object
"""Base class for the public object that has access to services, channels
and config parameters specified in YAML file.
```

Public Functions

```
__init__()
```

Public Members

```
config
```

```
channels
```

```
services
```

```
messages
```

Private Functions

```
__check_service_channel_relation(self) or is used some
nonexistent channel.

:returns: None
:rtype: None
:raise ZetaCLIError: Channel name doesn't exists in channel list.
```

```

    check_channel_message_relation(self) or is used some
    nonexistent channel.

    :returns: None
    :rtype: None
    :raise ZetaCLIError: Channel name doesn't exists in channel list.

```

```

__process_file()

```

zetacli.ZetaCLI public object. Represents the callbacks that will be called when the user type zeta on the terminal.

Public Functions

```

__init__(self) constructor.

```

```

:returns: None
:rtype: None

```

init(self) when the user type "zeta init" and is responsible for generates the minimum requirements in order to Zeta works properly.

```

:returns: Exit code
:rtype: int
:raise ZetaCLIError: Error in zeta.cmake or zeta.yaml file path

```

version(self) when the user type "zeta version" and is responsible for sends out the ZetaCLI version.

```

:returns: Exit code
:rtype: int

```

check_files(self)

check_code(self, src_dir)

check(self) when the user type "zeta check" and is responsible for checks if the needed steps were made by user in order to Zeta works properly.

```

:returns: Exit code
:rtype: int

```

services(self) when the user type "zeta services" and is responsible for generates files template-based with the services initialized.

```

:returns: Exit code
:rtype: int
:raise ZetaCLIError: Error opening or reading files

```

gen(self) generate all the internal files that represents Zeta system like channels, Zeta threads, Zeta API, and others.

(continues on next page)

(continued from previous page)

```
:returns: Exit code
:rtype: int
```

sniffer(self) all the internal files that represents Zeta system like channels, Zeta threads, Zeta API, and others.

```
:returns: Exit code
:rtype: int
```

isc(self) starts a high level ISC on the host. It takes all the yaml file details to run the proper environment.

```
:returns: Exit code
:rtype: int
```

zeta.BzetaClass for ZetaCLIError in Zeta public Exception

Public Functions

__init__(self, message, errcode)

```
:param message: Message to be displayed printing error
:param errcode: Error code
:returns: None
:rtype: None
```

handle(self) possible for handling the raise call.

```
:returns: None
:rtype: None
```

__str__(self)

Public Members

errcode

message

zeta.sniffer ZetaCLIEventListWalker public ListWalker
 Rsniffer is ZetaCLIEventListWalker used by ListWalker show the events occurred.

Public Functions

`__init__()`

```
create_table(self, title='{0}/{1}')

:param title: the table's title
:returns: the table container (LineBox)
:rtype: urwid.LineBox
```

`__len__(self)` iterator overloading

```
:returns: the table number of lines
:rtype: int
```

`append(self, item)` method overloading

```
:param item: dictionary with the data of the new item to be added
:returns: None
:rtype: None
```

`create_data_viewer(self)` viewer.

```
:returns: the data viewer
:rtype: urwid.Text
```

`__getitem__(self, index)` overloading

```
:param index: index of the desired element
:returns: the element at index
:rtype: urwid.Columns
```

`set_modified_callback(self, callback)` **MonitoredList is not**
implemented in SimpleFocusListWalker.
Use `connect_signal(list_walker, "modified", ...)` instead.

`set_focus(self, position)` n.

`next_position(self, position)` start_from.

`prev_position(self, position)` start_from.

`positions(self, reverse=False)` returning an iterable of positions.

Public Members

`event_list`
`table_rows`
`focus`
`table_title`
`data_viewer`

Private Functions

```
__create_table_header(self)  
  
:param event_list: list of event items  
:returns: table header  
:rtype: urwid.AttrMap
```

`zeta.zeta_cli.ZetaConf` : public FileFactory

Public Functions

`__init__()`
`create_substitutions(self)`

`zeta.zeta_cli.ZetaHeader` : public HeaderFileFactory
Represents the ZetaHeader class. It has the goal to assigns all the substitutions needed to Zeta works properly.

Public Functions

`__init__()`
`create_substitutions(self)` the needed substitutions to be written on the output file.

:returns: None
:rtype: None

Public Members

`services_reference`
`max_channel_size`

`class ZetaMessage` : represents the ZetaMessage. It holds the messages' information for code generation

Public Functions

`__init__()`

`__repr__(self)`

`digest_type(self, mtype)` and mounts the necessary data to the code generation.

:returns: **None**
:rtype: **None**

`code(self)` generates the code for the message.

:returns: the code generated that represents the message
:rtype: **str**

Public Members

name

size

level

description

parent

mtype_obj

mtype_base_type

mtype

fields

zeta_sniffer.ZetaSerialMonitor **public Protocol** implements the interface data related to the income information.

Public Functions

`__init__()`

`connection_made(self, transport)` connection **is** made

`digest_is_message(self, msg)` added message to extract the message information.

`data_received(self, data)` an a new data **is** available to be read.

:param data: **bytes** received at the serial.
:returns: **None**
:rtype: **None**

`connection_lost(self, exc)` the connection **is** lost

```
pause_writing(self) when the writing is paused
```

```
resume_writing(self) when the writing is resumed
```

```
send(self) four newline-terminated messages, one byte at a time.
```

Public Members

`event_list`

`logging_ui`

`line_buffer`

`zeta`

`transport`

```
class ZetaSniffer:
    """ZetaSniffer class that implements the sniffer execution function.
```

Public Functions

```
run(self, serial_port: str, baudrate: int)
    """Run the sniffer. It opens the serial at the
    specified baud rate passed as argument.

    :param serial_port: the serial port description. For example "/dev/ttyUSB0".
    :param baudrate: the communication frequency. For example 115200.
    :returns: None
    :rtype: None"""
```

```
zetacli.ZetaSource: public SourceFileFactory
    """ZetaSource class that has the goal to assigns all the
    substitutions needed to Zeta works properly.
```

Public Functions

```
__init__()
```

```
gen_sems(self)
    """Generate the semaphore for assigns the channel semaphores.

    :returns: None
    :rtype: None"""
```

```
gen_channels(self)
    """Generate the channels for creates all the channels that will be used by
    Zeta.

    :returns: None
    :rtype: None"""
```

```
gen_nvscfg(self)
    """Generate the nvscfg for assigns the nvs config.

    :returns: None
    :rtype: None"""
```

```
create_substitutions(self) the needed substitutions to be written
on the output file.
```

```
:returns: None
:rtype: None
```

Public Members

```
channels_creation
channels_sems
sector_size
sector_count
storage_offset
set_publishers
set_subscribers
arrays_init
services_array_init
run_services
storage_partition
```

```
zeta.ZetaYamlLoader: public SafeLoader a correct
dictionary-based in the YAML file.
```

Public Functions

```
__init__(self, stream) constructor.

:param stream: default param used for yaml.SafeLoader
:returns: None
:rtype: None
```

```
ref(self, node) ref statements in YAML file and generate a valid
reference

:param node: Ref statement with the reference name
:returns: Reference name
:rtype: str
```

```
include(self, node) include statements in YAML file and generate a
valid reference

:param node: Include statement with the reference name
:returns: Content of the file passed by include
:rtype: dict
```

```
struct zt_channel
#include <zeta.template.h> Define Zeta channel type.
```

Public Members

const char ***name**
Channel name

uint8_t ***data**
Channel raw data

uint8_t **read_only**

uint8_t **size**
Channel size

uint8_t **persistent**
Persistent type

zt_channel_e **id**
Channel Id

union *flag_data* **flag**
Options

struct k_sem ***sem**
Preserve shared-memory

zt_service_t ****publishers**
Publishers

zt_service_t ****subscribers**
Subscribers

struct **zt_data_bytes_t**
#include <zeta.template.h>

Public Members

size_t **size**

uint8_t **value**[]

struct **zt_data_int16_t**
#include <zeta.template.h>

Public Members

size_t **size**

int16_t **value**

struct **zt_data_int32_t**
#include <zeta.template.h>

Public Members

size_t **size**

int32_t **value**

```
struct zt_data_int64_t  
    #include <zeta.template.h>
```

Public Members

size_t **size**

int64_t **value**

```
struct zt_data_int8_t  
    #include <zeta.template.h>
```

Public Members

size_t **size**

int8_t **value**

```
struct zt_data_uint16_t  
    #include <zeta.template.h>
```

Public Members

size_t **size**

uint16_t **value**

```
struct zt_data_uint32_t  
    #include <zeta.template.h>
```

Public Members

size_t **size**

uint32_t **value**

```
struct zt_data_uint64_t  
    #include <zeta.template.h>
```

Public Members

size_t **size**

uint64_t **value**

```
struct zt_data_uint8_t  
    #include <zeta.template.h>
```

Public Members

```
size_t size
uint8_t value
struct zt_isc_net_pkt
```

Public Members

```
struct zt_isc_net_pkt_header header
struct zt_isc_net_pkt_header_message_info message_info
uint8_t *message
struct zt_isc_net_pkt_header
```

Public Members

```
uint8_t channel
uint8_t op
    !> 256 channels available
uint8_t status
    !> 0: read, 1: write, 2: read response, 3: write response, 4: update
uint8_t has_data
    !> 0: ok, 1: failed
struct zt_isc_net_pkt_header_message_info
```

Public Members

```
uint8_t crc
uint8_t size
    !> CCITT 8, polynom 0x07, initial value 0x00
struct zt_isc_packet
    #include <zeta.template.h>
```

Public Members

```
uint32_t id
uint8_t service_id
uint8_t channel_id
uint8_t op
uint8_t size
uint8_t message[$max_channel_size]
zeta_isc_zt_msg public Union
zeta_isc_zt_msg public Union
```

A zeta_isc_zt_msg config file will be filled up here


```
struct zt_service
    #include <zeta.template.h> Define Zeta service type.
```

Public Members

```
zt_service_e id
    Service ID

const char *name
    Service name

struct k_thread thread
    Service RTOS thread

zt_callback_f cb
    Service callback

k_thread_entry_t entry_point
    Service thread function (entry point)

k_thread_stack_t *stack
    Service thread stack

size_t stack_size
    Service thread stack size
```

```
module zeta
```

```
module __main__
```

```
module zeta._version
```

Variables

```
str
```

```
module zeta.messages
```

Variables

```
MsgType = namedtuple('MsgCode', "statement separator begin end sizable")
```

```
module sniffer
```

```
module zeta.zeta
```

Variables

```
ZETA_MODULE_DIR = "."
ZETA_TEMPLATES_DIR = "."
PROJECT_DIR = "."
ZETA_DIR = "."
ZETA_SRC_DIR = "."
ZETA_INCLUDE_DIR = "."
```

```
module zeta.zeta_cli
```

Functions

```
run()
```

Variables

```
ZETA_MODULE_DIR = "."
```

```
ZETA_TEMPLATES_DIR = "."
```

```
PROJECT_DIR = str(Path(".").resolve())
```

```
ZETA_DIR = "."
```

```
ZETA_SRC_DIR = "."
```

```
ZETA_INCLUDE_DIR = "."
```

```
OK_COLORED = "\033[0;42m \033[1;97mOK \033[0m"
```

```
FAIL_COLORED = "\033[0;41m \033[1;97mFAIL \033[0m"
```

```
WARNING_COLORED = "\033[1;43m \033[1;97mWARNING \033[0m"
```

```
module zeta.zeta_errors
```

Variables

```
EZTFIELD = 1
```

```
EZTFIELD = 2
```

```
EZTCHECKFAILED = 3
```

```
EZTINVREF = 4
```

```
EZTUNEXP = 10
```

```
module zeta.zeta_isc
```

Functions

```
repr(self)
```

```
create_base_message()
```

```
uart_write_handler()
```

```
uart_read_handler()
```

```
struct_contents(struct)
```

```
struct_contents_set(struct, raw_data)
```

```
callback_handler()
```

```
pub_read_handler()
```

```
isc_run()
```

Variables

```
ipc = None
context = Context.instance()
target_attached = False

module zeta_pkt

file __init__.py
file __main__.py
file _version.py
file messages.py
file sniffer.py
file zeta.template.c
    #include "zeta.h"#include <drivers/flash.h>#include <fs/nvs.h>#include <logging/log.h>#include <storage/flash_map.h>#include <string.h>#include <zephyr.h>#include "zt_serial_ipc.h"#include "device-tree_fixups.h"
```

Defines

```
NVS_SECTOR_COUNT
NVS_STORAGE_PARTITION
```

Functions

```
LOG_MODULE_REGISTER (zeta, CONFIG_ZETA_LOG_LEVEL)

static void __zt_monitor_thread (void)

K_THREAD_DEFINE (zt_monitor_thread_id, ZT_MONITOR_THREAD_STACK_SIZE, __zt_monitor_thread, 0, 0, 0)

K_MSGQ_DEFINE (zt_channels_changed_msgq, sizeof(uint8_t), 30, 4)

const char *zt_channel_name (zt_channel_e id, int *error)
    Return the channel name.
```

Parameters

- **id** – Channel Id
- **error** – Handle possible errors

Returns Channel name

```
size_t zt_channel_size (zt_channel_e id, int *error)
    Return the channel size.
```

Parameters

- **id** – Channel Id
- **error** – Handle possible errors

Returns Channel size

int **zt_chan_read** (zt_channel_e *id*, *zt_data_t* **channel_data*)

Read channel value.

Parameters

- **id** – Channel Id
- **channel_data** – pointer to a *zt_data_t* where the data will be retrieved.

Returns -ENODATA The channel was not found

Returns -EFAULT Channel value is NULL

Returns -EPERM Channel hasn't read function implemented

Returns -EINVAL Size passed is different to channel size

Returns Error code

int **zt_chan_pub** (zt_channel_e *id*, *zt_data_t* **channel_data*)

Publish channel value.

Parameters

- **id** – Channel Id
- **channel_data** – pointer to a *zt_data_t* where the data is.

Returns -ENODATA The channel was not found

Returns -EACCESS Current thread hasn't permission to publish this channel

Returns -EFAULT Channel value is NULL

Returns -EPERM Channel is read only

Returns -EINVAL Size passed is different to channel size

Returns -EAGAIN Valid function returns false

Returns Error code

file **zeta.template.h**

#include <stddef.h>#include <zephyr.h>#include <zephyr/types.h>

Defines

ZT_MONITOR_THREAD_STACK_SIZE

Stack size that is used in Zeta thread that manages channels callback calls.

ZT_STORAGE_SLEEP_TIME

Storage sleep time.

ZT_MONITOR_THREAD_PRIORITY

Channels thread priority.

ZT_SERVICE_DECLARE (*_name*, *_task*, *_cb*)

Declare a zeta service.

Parameters

- **_name** – Service name
- **_task** – Task pointer function
- **_cb** – Callback to be called when some subscribed channel change

ZT_SERVICE_RUN (*_name*)

Run a zeta service.

Parameters

- **_name** – Service name

ZT_VARIABLE_REF_SIZE (*x*)

Read variable reference and size easily \ to use in Zeta API. \ .

\

Parameters

- **x** – variable name \ \

ZT_CHECK_VAL (*_p, _e, _err, ...*)

Check if _v value is equal to _c, otherwise _err will be returned and a message will be sent to LOG.

Parameters

- **_v** – Value
- **_c** – Condition
- **_err** – Error code

ZT_CHECK (*_p, _err, ...*)

Check if _v is true, otherwise _err will be returned and a message will be sent to LOG.

Parameters

- **_v** – Value
- **_err** – Error code

Returns

ZT_DATA_S8 (*data*)

ZT_DATA_U8 (*data*)

ZT_DATA_S16 (*data*)

ZT_DATA_U16 (*data*)

ZT_DATA_S32 (*data*)

ZT_DATA_U32 (*data*)

ZT_DATA_S64 (*data*)

ZT_DATA_U64 (*data*)

ZT_DATA_BYTES (*_size, data, ...*)

Typedefs

```
typedef union data zt_data_t
typedef void (*zt_callback_f) (zt_channel_e id)
    zeta_callback_f define the callback function type of Zeta.

    Parameters id – Channel Id.

typedef struct zt_service zt_service_t
typedef struct zt_isc_packet zt_isc_packet_t
typedef struct zt_channel zt_channel_t
```

Functions

```
struct zt_isc_packet __attribute__((packed))
size_t zt_channel_size (zt_channel_e id, int *error)
    Return the channel size.
```

Parameters

- **id** – Channel Id
- **error** – Handle possible errors

Returns Channel size

```
const char *zt_channel_name (zt_channel_e id, int *error)
    Return the channel name.
```

Parameters

- **id** – Channel Id
- **error** – Handle possible errors

Returns Channel name

```
int zt_chan_read (zt_channel_e id, zt_data_t *channel_data)
    Read channel value.
```

Parameters

- **id** – Channel Id
- **channel_data** – pointer to a *zt_data_t* where the data will be retrieved.

Returns -ENODATA The channel was not found

Returns -EFAULT Channel value is NULL

Returns -EPERM Channel hasn't read function implemented

Returns -EINVAL Size passed is different to channel size

Returns Error code

```
int zt_chan_pub (zt_channel_e id, zt_data_t *channel_data)
    Publish channel value.
```

Parameters

- **id** – Channel Id

- **channel_data** – pointer to a `zt_data_t` where the data is.

Returns -ENODATA The channel was not found

Returns -EACCESS Current thread hasn't permission to publish this channel

Returns -EFAULT Channel value is NULL

Returns -EPERM Channel is read only

Returns -EINVAL Size passed is different to channel size

Returns -EAGAIN Valid function returns false

Returns Error code

Variables

`uint32_t id`

`uint8_t service_id`

`uint8_t channel_id`

`uint8_t op`

`uint8_t size`

`uint8_t message[$max_channel_size]`

`union flag_data __attribute__`

file `zt_serial_ipc.h`

`#include <zephyr.h>#include "zeta.h"`

Defines

`ZT_SERIAL_IPC_RX_THREAD_STACK_SIZE`

`ZT_SERIAL_RX_THREAD_PRIORITY`

`ZT_SERIAL_IPC_TX_THREAD_STACK_SIZE`

`ZT_SERIAL_TX_THREAD_PRIORITY`

Functions

`int zt_serial_ipc_send_update_to_host (zt_channel_e id)`

`k_tid_t zt_serial_ipc_thread ()`

file `zt_uart.h`

`#include <device.h>#include <zephyr.h>`

Functions

```
int uart_open (char *dev_label)
int uart_write (uint8_t *data, size_t size)
int uart_write_str (char *str)
int uart_write_byte (uint8_t *byte)
struct k_msgq *uart_get_input_msgq ()
struct k_msgq *uart_get_output_msgq ()
```

file **zt_serial_ipc.c**

```
#include "zt_serial_ipc.h" #include <string.h> #include <sys/crc.h> #include <sys/printk.h> #include
<zephyr.h> #include <zt_uart.h> #include "devicetree.h" #include "kernel.h" #include "zeta.h"
```

Defines

```
UART_DEVICE_NAME
    SPDX-License-Identifier: Apache-2.0
MAX_MESSAGE_LENGTH
ZT_ISC_PKT_ASSERT_EQ (actual, expected, ret)
ZT_ISC_PKT_ASSERT (cond, ret)
SELF_TEST_INIT ()
SELF_TEST_ASSERT (cond)
SELF_TEST_FINISH ()
NUMBER_OF_CHANNELS
ZT_ISC_NET_PKT_WITH_MSG_SIZE (_size)
```

Enums

```
enum zt_isc_net_pkt_op_t
    Values:
        enumerator OP_READ
        enumerator OP_WRITE
        enumerator OP_READ_RESPONSE
        enumerator OP_WRITE_RESPONSE
        enumerator OP_UPDATE
        enumerator OP_DEBUG
enum pkt_data_availability_t
    Values:
        enumerator DATA_UNAVAILABLE
        enumerator DATA_AVAILABLE
```



```
enum pkt_status_t
```

Values:

```
enumerator STATUS_OK
```

```
enumerator STATUS_FAILED
```

Functions

```
K_MSGQ_DEFINE (__channel_updated_queue, 1, 16, 1)
```

```
void zt_isc_net_pkt_clear(struct zt_isc_net_pkt *self)
```

```
void zt_isc_net_pkt_send(struct zt_isc_net_pkt *self)
```

```
int zt_isc_net_pkt_set_message(struct zt_isc_net_pkt *self, char *message, size_t message_size)
```

```
void zt_isc_net_pkt_calc_crc(struct zt_isc_net_pkt *self)
```

```
uint8_t zt_isc_net_pkt_header_is_valid(struct zt_isc_net_pkt_header *header)
```

```
uint8_t zt_isc_net_pkt_message_is_valid(struct zt_isc_net_pkt *pkt)
```

```
int digest_byte(uint8_t data, struct zt_isc_net_pkt *pkt)
```

```
uint64_t digest_byte_test()
```

```
int zt_serial_ipc_send_update_to_host(zt_channel_e id)
```

```
void __zt_serial_ipc_tx_thread(void)
```

```
k_tid_t zt_serial_ipc_thread()
```

```
void __zt_serial_ipc_rx_thread(void)
```

```
K_THREAD_DEFINE (zt_serial_ipc_rx_id, ZT_SERIAL_IPC_RX_THREAD_STACK_SIZE, __zt_serial_
```

```
K_THREAD_DEFINE (zt_serial_ipc_tx_id, ZT_SERIAL_IPC_TX_THREAD_STACK_SIZE, __zt_serial_
```

Variables

```
static k_tid_t __current_rx_thread
```

file **zt_uart.c**

```
#include "zt_uart.h"#include <drivers/uart.h>#include <zephyr.h>
```

Functions

```
K_MSGQ_DEFINE (__input_msgq, sizeof(uint8_t), 512, 1)
```

```
K_MSGQ_DEFINE (__output_msgq, sizeof(uint8_t), 512, 1)
```

```
static void uart_fifo_callback(const struct device *dev, void *user_data)
```

```
struct k_msgq *uart_get_input_msgq()
```

```
struct k_msgq *uart_get_output_msgq()
```

```
int uart_open(char *dev_label)
```

```
int uart_write(uint8_t *data, size_t size)
```

```
int uart_write_str(char *str)
```

```
int uart_write_byte (uint8_t *byte)
```

Variables

```
static const struct device *__uart_dev = 0
```

```
const struct uart_config uart_cfg = {.baudrate = 115200, .parity = UART_CFG_PARITY_
```

```
file zeta.py
```

```
file zeta_cli.py
```

```
file zeta_errors.py
```

```
file zeta_isc.py
```

```
file zeta_pkt.py
```

```
dir /home/docs/checkouts/readthedocs.org/user_builds/zeta-middleware/checkouts/latest/zeta/t
```

```
dir /home/docs/checkouts/readthedocs.org/user_builds/zeta-middleware/checkouts/latest/zeta/t
```

```
dir /home/docs/checkouts/readthedocs.org/user_builds/zeta-middleware/checkouts/latest/zeta/t
```

```
dir /home/docs/checkouts/readthedocs.org/user_builds/zeta-middleware/checkouts/latest/zeta/t
```

```
dir /home/docs/checkouts/readthedocs.org/user_builds/zeta-middleware/checkouts/latest/zeta
```

LICENSE

MIT Copyright 2020 Lucas Peixoto and Rodrigo Peixoto

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

Symbols

__buffer (*zeta.zeta_isc.SerialDataHandler attribute*), 9
 __channels (*zeta.zeta_isc.IPC attribute*), 6
 __check_channel_message_relation()
 built-in function, 10
 __check_service_channel_relation()
 built-in function, 10
 __create_table_header()
 built-in function, 14
 __current_pkt (*zeta.zeta_isc.SerialDataHandler attribute*), 9
 __current_rx_thread (*C++ member*), 29
 __data, 8
 __getitem__()
 built-in function, 13
 __init__()
 built-in function, 3–5, 7, 9–17
 __len__()
 built-in function, 13
 __process_file()
 built-in function, 11
 __repr__()
 built-in function, 3, 7
 __state (*zeta.zeta_isc.SerialDataHandler attribute*), 9
 __str__()
 built-in function, 12
 __uart_dev (*C++ member*), 30
 __zeta (*zeta.zeta_isc.IPC attribute*), 6
 __zt_monitor_thread (*C++ function*), 23
 __zt_serial_ipc_rx_thread (*C++ function*), 29
 __zt_serial_ipc_tx_thread (*C++ function*), 29
 fields, 7, 8
 pack, 7, 8

A

append()
 built-in function, 13
 arrays_init, 17

B

built-in function

__check_channel_message_relation(), 10
 __check_service_channel_relation(), 10
 __create_table_header(), 14
 __getitem__(), 13
 __init__(), 3–5, 7, 9–17
 __len__(), 13
 __process_file(), 11
 __repr__(), 3, 7
 __str__(), 12
 append(), 13
 check(), 11
 check_code(), 11
 check_files(), 11
 clear_data(), 7
 connection_lost(), 15
 connection_made(), 15
 create_data_viewer(), 13
 create_substitutions(), 4, 14, 16
 create_table(), 13
 data(), 7
 data_received(), 15
 digest_isc_message(), 15
 from_bytes(), 7
 gen(), 11
 gen_creation(), 16
 gen_nvs_config(), 16
 gen_sems(), 16
 generate_file(), 4
 handle(), 12
 header_to_bytes(), 7
 include(), 17
 init(), 11
 isc(), 12
 next_position(), 13
 pause_writing(), 15
 positions(), 13
 prev_position(), 13
 ref(), 17
 resume_writing(), 16
 run(), 4

[send\(\)](#), 16
[services\(\)](#), 11
[set_data\(\)](#), 7
[set_data_info\(\)](#), 7
[set_data_info_from_bytes\(\)](#), 7
[set_data_with_struct\(\)](#), 7
[set_focus\(\)](#), 13
[set_header\(\)](#), 7
[set_header_from_bytes\(\)](#), 7
[set_modified_callback\(\)](#), 13
[sniffer\(\)](#), 12
[struct_contents\(\)](#), 7
[struct_contents_set\(\)](#), 7
[test_array_simple\(\)](#), 10
[test_bitarray_simple\(\)](#), 10
[test_complex_message\(\)](#), 10
[test_primitives\(\)](#), 10
[test_struct_simple\(\)](#), 10
[test_union_simple\(\)](#), 10
[to_bytes\(\)](#), 7
[version\(\)](#), 11
[zeta.messages.ZetaMessage.__init__\(\)](#), 15
[zeta.messages.ZetaMessage.__repr__\(\)](#), 15
[zeta.messages.ZetaMessage.code\(\)](#), 15
[zeta.messages.ZetaMessage.digest_type\(\)](#), 15
[zeta.sniffer.ZetaSniffer.run\(\)](#), 16
[zeta.zeta.Message.__init__\(\)](#), 8
[zeta.zeta.Message.__repr__\(\)](#), 8
[zeta.zeta_cli.run\(\)](#), 22
[zeta.zeta_isc.callback_handler\(\)](#), 22
[zeta.zeta_isc.create_base_message\(\)](#), 22
[zeta.zeta_isc.IPC.__init__\(\)](#), 6
[zeta.zeta_isc.IPC.create_channels\(\)](#), 6
[zeta.zeta_isc.IPC.digest_packet\(\)](#), 6
[zeta.zeta_isc.IPC.read_channel\(\)](#), 6
[zeta.zeta_isc.IPC.set_channel\(\)](#), 6
[zeta.zeta_isc.IPCChannel.__init__\(\)](#), 6
[zeta.zeta_isc.IPCChannel.__repr__\(\)](#), 6
[zeta.zeta_isc.IPCChannel.create_struct\(\)](#), 6
[zeta.zeta_isc.IPCChannel.message_unionfield\(\)](#), 6
[zeta.zeta_isc.isc_run\(\)](#), 22
[zeta.zeta_isc.pub_read_handler\(\)](#), 22
[zeta.zeta_isc.repr\(\)](#), 22
[zeta.zeta_isc.SerialDataHandler.__init__\(\)](#), 9
[zeta.zeta_isc.SerialDataHandler.append\(\)](#), 9
[zeta.zeta_isc.SerialDataHandler.digest\(\)](#), 9
[zeta.zeta_isc.SerialDataHandler.run\(\)](#), 9
[zeta.zeta_isc.SerialDataHandler.send_command\(\)](#), 9
[zeta.zeta_isc.struct_contents\(\)](#), 22
[zeta.zeta_isc.struct_contents_set\(\)](#), 22
[zeta.zeta_isc.uart_read_handler\(\)](#), 22
[zeta.zeta_isc.uart_write_handler\(\)](#), 22

C

[channel_changed_queue](#) (*zeta.zeta_isc.IPC attribute*), 6
[channel_id](#) (*C++ member*), 27
[channels](#), 10
[channels_creation](#), 17
[channels_sems](#), 17
[built-in function](#), 11
[check_code\(\)](#)
[check_files\(\)](#)
[built-in function](#), 11
[clear_data\(\)](#)
[built-in function](#), 7
[config](#), 10
[connection_lost\(\)](#)
[built-in function](#), 15
[connection_made\(\)](#)
[built-in function](#), 15
[context](#) (*zeta.zeta_isc attribute*), 23
[create_data_viewer\(\)](#)
[built-in function](#), 13
[create_substitutions\(\)](#)
[built-in function](#), 4, 14, 16
[create_table\(\)](#)
[built-in function](#), 13

D

[data](#) (*C++ union*), 4
[data](#) (*zeta.zeta_isc.IPCChannel attribute*), 6
[datafile\(\)](#)
[built-in function](#), 7
[data::bytes](#) (*C++ member*), 4
[data::s16](#) (*C++ member*), 4
[data::s32](#) (*C++ member*), 4
[data::s64](#) (*C++ member*), 4
[data::s8](#) (*C++ member*), 4

data::u16 (C++ member), 4
 data::u32 (C++ member), 4
 data::u64 (C++ member), 4
 data::u8 (C++ member), 4
 DATA_AVAILABLE, 8
 data_received()
 built-in function, 15
 DATA_UNAVAILABLE, 8
 data_viewer, 14
 description (*zeta.messages.ZetaMessage attribute*), 15
 destination_file, 5
 digest_byte (C++ function), 29
 digest_byte_test (C++ function), 29
 digest_isc_message()
 built-in function, 15

E

errcode, 12
 event_list, 14, 16
 EZTCHECKFAILED (*zeta.zeta_errors attribute*), 22
 EZTFIELD (*zeta.zeta_errors attribute*), 22
 EZTFILE (*zeta.zeta_errors attribute*), 22
 EZTINVREF (*zeta.zeta_errors attribute*), 22
 EZTUNEXP (*zeta.zeta_errors attribute*), 22

F

FAIL_COLORED (*zeta.zeta_cli attribute*), 22
 fields (*zeta.messages.ZetaMessage attribute*), 15
 flag_data (C++ union), 5
 flag_data::data (C++ member), 5
 flag_data::field (C++ member), 5
 flag_data::on_changed (C++ member), 5
 flag_data::pend_callback (C++ member), 5
 flag_data::pend_persistent (C++ member), 5
 flag_data::source_serial_isc (C++ member), 5
 focus, 14
 from_bytes()
 built-in function, 7

G

gen()
 built-in function, 11
 gen_creation()
 built-in function, 16
 gen_nvs_config()
 built-in function, 16
 gen_sems()
 built-in function, 16
 generate_file()
 built-in function, 4

H

handle()
 built-in function, 12
 header_to_bytes()
 built-in function, 7

I

id, 3
 id (C++ member), 27
 include()
 built-in function, 17
 init()
 built-in function, 11
 initial_value, 3
 ipc (*zeta.zeta_isc attribute*), 23
 iqueue (*zeta.zeta_isc.SerialDataHandler attribute*), 9
 isc()
 built-in function, 12

L

level (*zeta.messages.ZetaMessage attribute*), 15
 line_buffer, 16
 LOG_MODULE_REGISTER (C++ function), 23
 logging_ui, 16

M

max_channel_size, 14
 MAX_MESSAGE_LENGTH (C macro), 28
 message, 3, 12
 message (C++ member), 27
 message_obj, 3
 messages, 10
 metadata (*zeta.zeta_isc.IPCChannel attribute*), 6
 msg_format (*zeta.zeta.Message attribute*), 8
 MsgType (*zeta.messages attribute*), 21
 mtype (*zeta.messages.ZetaMessage attribute*), 15
 mtype_base_type (*zeta.messages.ZetaMessage attribute*), 15
 mtype_obj (*zeta.messages.ZetaMessage attribute*), 15

N

name, 3, 9
 name (*zeta.messages.ZetaMessage attribute*), 15
 name (*zeta.zeta.Message attribute*), 8
 next_position()
 built-in function, 13
 NUMBER_OF_CHANNELS (C macro), 28
 NVS_SECTOR_COUNT (C macro), 23
 NVS_STORAGE_PARTITION (C macro), 23

O

OK_COLORED (*zeta.zeta_cli attribute*), 22
 on_changed, 3

op (C++ member), 27
OP_DEBUG, 8
OP_READ, 8
OP_READ_RESPONSE, 8
OP_UPDATE, 8
OP_WRITE, 8
OP_WRITE_RESPONSE, 8
oqueue (*zeta.zeta_isc.SerialDataHandler* attribute), 9

P

parent (*zeta.messages.ZetaMessage* attribute), 15
pause_writing()
 built-in function, 15
persistent, 3
pkt_data_availability_t (C++ enum), 28
pkt_data_availability_t::DATA_AVAILABLE
 (C++ enumerator), 28
pkt_data_availability_t::DATA_UNAVAILABLE
 (C++ enumerator), 28
pkt_status_t (C++ enum), 28
pkt_status_t::STATUS_FAILED (C++ enumerator), 29
pkt_status_t::STATUS_OK (C++ enumerator), 29
positions()
 built-in function, 13
prev_position()
 built-in function, 13
priority, 9
PROJECT_DIR (*zeta.zeta* attribute), 21
PROJECT_DIR (*zeta.zeta_cli* attribute), 22
pub_channels_names, 9
pub_channels_obj, 9
pub_services_obj, 3

R

read_only, 3
ref()
 built-in function, 17
resume_writing()
 built-in function, 16
run()
 built-in function, 4
run_services, 17

S

sector_count, 4, 17
sector_size, 17
SELF_TEST_ASSERT (C macro), 28
SELF_TEST_FINISH (C macro), 28
SELF_TEST_INIT (C macro), 28
sem, 3
sem (*zeta.zeta_isc.IPC* attribute), 6
send()

 built-in function, 16
service_id (C++ member), 27
services, 10
services()
 built-in function, 11
services_array_init, 17
services_reference, 14
set_data()
 built-in function, 7
set_data_info()
 built-in function, 7
set_data_info_from_bytes()
 built-in function, 7
set_data_with_struct()
 built-in function, 7
set_focus()
 built-in function, 13
set_header()
 built-in function, 7
set_header_from_bytes()
 built-in function, 7
set_modified_callback()
 built-in function, 13
set_publishers, 17
set_subscribers, 17
size, 3
size (C++ member), 27
size (*zeta.messages.ZetaMessage* attribute), 15
sniffer()
 built-in function, 12
stack_size, 9
STATE_DIGEST_BODY
 (*zeta.zeta_isc.SerialDataHandler* attribute), 9
STATE_DIGEST_HEADER_DATA_INFO
 (*zeta.zeta_isc.SerialDataHandler* attribute), 9
STATE_DIGEST_HEADER_OP
 (*zeta.zeta_isc.SerialDataHandler* attribute), 9
STATUS_FAILED, 8
STATUS_OK, 8
storage_offset, 17
storage_partition, 4, 17
storage_period, 4
str (*zeta._version* attribute), 21
struct_contents()
 built-in function, 7
struct_contents_set()
 built-in function, 7
sub_channels_names, 9
sub_channels_obj, 9
sub_services_obj, 3
substitutions, 5

T

table_rows, 14

table_title, 14
 target_attached (*zeta.zeta_isc attribute*), 23
 template_file, 5
 test_array_simple()
 built-in function, 10
 test_bitarray_simple()
 built-in function, 10
 test_complex_message()
 built-in function, 10
 test_primitives()
 built-in function, 10
 test_struct_simple()
 built-in function, 10
 test_union_simple()
 built-in function, 10
 to_bytes()
 built-in function, 7
 transport, 16

U

UART_DEVICE_NAME (*C macro*), 28
 uart_fifo_callback (*C++ function*), 29
 uart_get_input_msgq (*C++ function*), 28, 29
 uart_get_output_msgq (*C++ function*), 28, 29
 uart_open (*C++ function*), 28, 29
 uart_write (*C++ function*), 28, 29
 uart_write_byte (*C++ function*), 28, 29
 uart_write_str (*C++ function*), 28, 29

V

version()
 built-in function, 11

W

WARNING_COLORED (*zeta.zeta_cli attribute*), 22

Z

zeta, 5, 16
 zeta (*built-in class*), 21
 zeta.__main__ (*built-in class*), 21
 zeta._version (*built-in class*), 21
 zeta.messages (*built-in class*), 21
 zeta.messages.ZetaMessage (*built-in class*), 14
 zeta.messages.ZetaMessage.__init__()
 built-in function, 15
 zeta.messages.ZetaMessage.__repr__()
 built-in function, 15
 zeta.messages.ZetaMessage.code()
 built-in function, 15
 zeta.messages.ZetaMessage.digest_type()
 built-in function, 15
 zeta.sniffer (*built-in class*), 21
 zeta.sniffer.ZetaSniffer (*built-in class*), 16
 zeta.sniffer.ZetaSniffer.run()
 built-in function, 16
 zeta.zeta (*built-in class*), 21
 zeta.zeta.Message (*built-in class*), 8
 zeta.zeta.Message.__init__()
 built-in function, 8
 zeta.zeta.Message.__repr__()
 built-in function, 8
 zeta.zeta_cli (*built-in class*), 21
 zeta.zeta_cli.run()
 built-in function, 22
 zeta.zeta_errors (*built-in class*), 22
 zeta.zeta_isc (*built-in class*), 22
 zeta.zeta_isc.callback_handler()
 built-in function, 22
 zeta.zeta_isc.create_base_message()
 built-in function, 22
 zeta.zeta_isc.IPC (*built-in class*), 5
 zeta.zeta_isc.IPC.__init__()
 built-in function, 6
 zeta.zeta_isc.IPC.create_channels()
 built-in function, 6
 zeta.zeta_isc.IPC.digest_packet()
 built-in function, 6
 zeta.zeta_isc.IPC.read_channel()
 built-in function, 6
 zeta.zeta_isc.IPC.set_channel()
 built-in function, 6
 zeta.zeta_isc.IPCChannel (*built-in class*), 6
 zeta.zeta_isc.IPCChannel.__init__()
 built-in function, 6
 zeta.zeta_isc.IPCChannel.__repr__()
 built-in function, 6
 zeta.zeta_isc.IPCChannel.create_struct()
 built-in function, 6
 zeta.zeta_isc.IPCChannel.message_union_field()
 built-in function, 6
 zeta.zeta_isc.isc_run()
 built-in function, 22
 zeta.zeta_isc.pub_read_handler()
 built-in function, 22
 zeta.zeta_isc.repr()
 built-in function, 22
 zeta.zeta_isc.SerialDataHandler (*built-in class*), 8
 zeta.zeta_isc.SerialDataHandler.__init__()
 built-in function, 9
 zeta.zeta_isc.SerialDataHandler.append()
 built-in function, 9
 zeta.zeta_isc.SerialDataHandler.digest()
 built-in function, 9
 zeta.zeta_isc.SerialDataHandler.run()
 built-in function, 9
 zeta.zeta_isc.SerialDataHandler.send_command()
 built-in function, 9

zeta.zeta_isc.struct_contents()
 built-in function, 22
zeta.zeta_isc.struct_contents_set()
 built-in function, 22
zeta.zeta_isc.uart_read_handler()
 built-in function, 22
zeta.zeta_isc.uart_write_handler()
 built-in function, 22
zeta.zeta_pkt (*built-in class*), 23
ZETA_DIR (*zeta.zeta attribute*), 21
ZETA_DIR (*zeta.zeta_cli attribute*), 22
ZETA_INCLUDE_DIR (*zeta.zeta attribute*), 21
ZETA_INCLUDE_DIR (*zeta.zeta_cli attribute*), 22
ZETA_MODULE_DIR (*zeta.zeta attribute*), 21
ZETA_MODULE_DIR (*zeta.zeta_cli attribute*), 22
ZETA_SRC_DIR (*zeta.zeta attribute*), 21
ZETA_SRC_DIR (*zeta.zeta_cli attribute*), 22
ZETA_TEMPLATES_DIR (*zeta.zeta attribute*), 21
ZETA_TEMPLATES_DIR (*zeta.zeta_cli attribute*), 22
zt_callback_f (*C++ type*), 26
zt_chan_pub (*C++ function*), 24, 26
zt_chan_read (*C++ function*), 23, 26
zt_channel (*C++ struct*), 17
zt_channel::data (*C++ member*), 18
zt_channel::flag (*C++ member*), 18
zt_channel::id (*C++ member*), 18
zt_channel::name (*C++ member*), 18
zt_channel::persistent (*C++ member*), 18
zt_channel::publishers (*C++ member*), 18
zt_channel::read_only (*C++ member*), 18
zt_channel::sem (*C++ member*), 18
zt_channel::size (*C++ member*), 18
zt_channel::subscribers (*C++ member*), 18
zt_channel_name (*C++ function*), 23, 26
zt_channel_size (*C++ function*), 23, 26
zt_channel_t (*C++ type*), 26
ZT_CHECK (*C macro*), 25
ZT_CHECK_VAL (*C macro*), 25
ZT_DATA_BYTES (*C macro*), 25
zt_data_bytes_t (*C++ struct*), 18
zt_data_bytes_t::size (*C++ member*), 18
zt_data_bytes_t::value (*C++ member*), 18
zt_data_int16_t (*C++ struct*), 18
zt_data_int16_t::size (*C++ member*), 18
zt_data_int16_t::value (*C++ member*), 18
zt_data_int32_t (*C++ struct*), 18
zt_data_int32_t::size (*C++ member*), 19
zt_data_int32_t::value (*C++ member*), 19
zt_data_int64_t (*C++ struct*), 19
zt_data_int64_t::size (*C++ member*), 19
zt_data_int64_t::value (*C++ member*), 19
zt_data_int8_t (*C++ struct*), 19
zt_data_int8_t::size (*C++ member*), 19
zt_data_int8_t::value (*C++ member*), 19
ZT_DATA_S16 (*C macro*), 25
ZT_DATA_S32 (*C macro*), 25
ZT_DATA_S64 (*C macro*), 25
ZT_DATA_S8 (*C macro*), 25
zt_data_t (*C++ type*), 26
ZT_DATA_U16 (*C macro*), 25
ZT_DATA_U32 (*C macro*), 25
ZT_DATA_U64 (*C macro*), 25
ZT_DATA_U8 (*C macro*), 25
zt_data_uint16_t (*C++ struct*), 19
zt_data_uint16_t::size (*C++ member*), 19
zt_data_uint16_t::value (*C++ member*), 19
zt_data_uint32_t (*C++ struct*), 19
zt_data_uint32_t::size (*C++ member*), 19
zt_data_uint32_t::value (*C++ member*), 19
zt_data_uint64_t (*C++ struct*), 19
zt_data_uint64_t::size (*C++ member*), 19
zt_data_uint64_t::value (*C++ member*), 19
zt_data_uint8_t (*C++ struct*), 19
zt_data_uint8_t::size (*C++ member*), 20
zt_data_uint8_t::value (*C++ member*), 20
zt_isc_net_pkt (*C++ struct*), 20
zt_isc_net_pkt::header (*C++ member*), 20
zt_isc_net_pkt::message (*C++ member*), 20
zt_isc_net_pkt::message_info (*C++ member*), 20
zt_isc_net_pkt_calc_crc (*C++ function*), 29
zt_isc_net_pkt_clear (*C++ function*), 29
zt_isc_net_pkt_header (*C++ struct*), 20
zt_isc_net_pkt_header::channel (*C++ member*), 20
zt_isc_net_pkt_header::has_data (*C++ member*), 20
zt_isc_net_pkt_header::op (*C++ member*), 20
zt_isc_net_pkt_header::status (*C++ member*), 20
zt_isc_net_pkt_header_is_valid (*C++ function*), 29
zt_isc_net_pkt_header_message_info (*C++ struct*), 20
zt_isc_net_pkt_header_message_info::crc (*C++ member*), 20
zt_isc_net_pkt_header_message_info::size (*C++ member*), 20
zt_isc_net_pkt_message_is_valid (*C++ function*), 29
zt_isc_net_pkt_op_t (*C++ enum*), 28
zt_isc_net_pkt_op_t::OP_DEBUG (*C++ enumerator*), 28
zt_isc_net_pkt_op_t::OP_READ (*C++ enumerator*), 28
zt_isc_net_pkt_op_t::OP_READ_RESPONSE (*C++ enumerator*), 28

zt_isc_net_pkt_op_t::OP_UPDATE (C++ *enumerator*), 28
 zt_isc_net_pkt_op_t::OP_WRITE (C++ *enumerator*), 28
 zt_isc_net_pkt_op_t::OP_WRITE_RESPONSE (C++ *enumerator*), 28
 zt_isc_net_pkt_send (C++ *function*), 29
 zt_isc_net_pkt_set_message (C++ *function*), 29
 ZT_ISC_NET_PKT_WITH_MSG_SIZE (C *macro*), 28
 zt_isc_packet (C++ *struct*), 20
 zt_isc_packet::channel_id (C++ *member*), 20
 zt_isc_packet::id (C++ *member*), 20
 zt_isc_packet::message (C++ *member*), 20
 zt_isc_packet::op (C++ *member*), 20
 zt_isc_packet::service_id (C++ *member*), 20
 zt_isc_packet::size (C++ *member*), 20
 zt_isc_packet_t (C++ *type*), 26
 ZT_ISC_PKT_ASSERT (C *macro*), 28
 ZT_ISC_PKT_ASSERT_EQ (C *macro*), 28
 ZT_MONITOR_THREAD_PRIORITY (C *macro*), 24
 ZT_MONITOR_THREAD_STACK_SIZE (C *macro*), 24
 ZT_SERIAL_IPC_RX_THREAD_STACK_SIZE (C *macro*), 27
 zt_serial_ipc_send_update_to_host (C++ *function*), 27, 29
 zt_serial_ipc_thread (C++ *function*), 27, 29
 ZT_SERIAL_IPC_TX_THREAD_STACK_SIZE (C *macro*), 27
 ZT_SERIAL_RX_THREAD_PRIORITY (C *macro*), 27
 ZT_SERIAL_TX_THREAD_PRIORITY (C *macro*), 27
 zt_service (C++ *struct*), 20
 zt_service::cb (C++ *member*), 21
 zt_service::entry_point (C++ *member*), 21
 zt_service::id (C++ *member*), 21
 zt_service::name (C++ *member*), 21
 zt_service::stack (C++ *member*), 21
 zt_service::stack_size (C++ *member*), 21
 zt_service::thread (C++ *member*), 21
 ZT_SERVICE_DECLARE (C *macro*), 24
 ZT_SERVICE_RUN (C *macro*), 24
 zt_service_t (C++ *type*), 26
 ZT_STORAGE_SLEEP_TIME (C *macro*), 24
 ZT_VARIABLE_REF_SIZE (C *macro*), 25