

VPI_TCP_SERVER



June 16, 2025

Jay Convertino

Contents

1 Usage	2
1.1 Introduction	2
1.2 Dependencies	2
1.2.1 fusesoc_info Depenecies	2
2 Architecture	3
3 Building	3
3.1 fusesoc	3
3.2 Source Files	4
3.2.1 fusesoc_info File List	4
3.3 Targets	4
3.3.1 fusesoc_info Targets	4
3.4 Directory Guide	5
4 Simulation	6
5 Code Documentation	7
5.1 tcp server code document	8

1 Usage

1.1 Introduction

VPI TCP Server is a library which allows for a Verilog simulation to interface with a TCP port. This library provides three functions.

- `setup_tcp_server(ADDRESS, PORT)`, RETURNS File Descriptor (FD)
- `recv_tcp_server(PORT, VECTOR)`, RETURNS number of bytes received (non-blocking, 0 is nothing available)
- `send_tcp_server(PORT, VECTOR)`, RETURNS number of bytes send (non-blocking, 0 is nothing written)

Library supports up to 256 TCP server instances. Each instance is setup by `setup_tcp_server`. This returns a descriptor for that instance. Then that descriptor is used for nothing. The field `PORT` is used to associate `setup_tcp_server` with a `recv_tcp_server` and a `send_tcp_server`. This can be done in multiple calls.

You can use the following for including the library in your project:

```
dep_vpi:
  depend:
    - AFRL:vpi:tcp_server:1.0.0

targets:
  default: &default
  description: Default file set.
  filesets: [src, dep, dep_vpi]
```

1.2 Dependencies

The following are the dependencies of the cores.

- fusesoc 2.X
- iverilog (simulation)

1.2.1 fusesoc_info Depenecies

- `dep_tb`
 - AFRL:utility:sim_helper
- `dep_gen`
 - AFRL:utility:generators:1.0.0

2 Architecture

This VPI library provides three functions for the user to use during simulation for creating a TCP server. They are `setup_tcp_server`, `recv_tcp_server`, and `send_tcp_server`. These are used to setup the server on a port, receive data, and send data respectively. These functions use ringbuffers and multithreading to separate server I/O from the simulation so TCP access will not slow down the simulation.

The `setup_tcp_server` is given an address, usually local or a ethernet port address, to use and a port. It will attempt to use this information and create an active connection. The connection is stored and is based upon its port number. Meaning all send and recvs use the port number to differentiate the connects. This obviously has its flaws, but provides a simple interface to the end user. This function has to be called first before `recv` or `send`.

The `recv_tcp_server` will read data from the socket setup by `setup_tcp_server`. The port must match the port given setup. It will read and return the number of bytes into the vector. This will also return the number of bytes read. Since this is a non-blocking function this can be zero.

The `send_tcp_server` will write data to the socket setup by `setup_tcp_server`. The port must match the port given in setup. It will write and return the number of bytes written to the socket. This will also return the number of bytes written. This is non-blocking and can write zero bytes if it is unable to.

Please see 5 for more information per target.

3 Building

The all VPI TCP Server source files are written in C to target the VPI API from Verilog 2001. They should simulate in any modern simulation tool that has VPI support. The library comes as a fusesoc packaged core and can be included in any other testbench. Be sure to make sure you have met the dependencies listed in the previous section.

3.1 fusesoc

Fusesoc is a system for building FPGA software without relying on the internal project management of the tool. Avoiding vendor lock in to Vivado or Quartus. These cores, when included in a project, can be easily integrated and targets created based upon the end developer needs. The core by itself is not a part of a system and should be integrated into a fusesoc based system. Simulations are setup to use fusesoc and are a part of its targets.

3.2 Source Files

3.2.1 fusesoc_info File List

- src
 - 'src/tcp_server.c': 'file_type': 'cSource'
 - 'src/send_tcp_server.c': 'file_type': 'cSource'
 - 'src/recv_tcp_server.c': 'file_type': 'cSource'
 - 'src/tcp_server.h': 'file_type': 'cSource', 'is_include_file': True
 - 'src/send_tcp_server.h': 'file_type': 'cSource', 'is_include_file': True
 - 'src/recv_tcp_server.h': 'file_type': 'cSource', 'is_include_file': True
 - 'src/tcp_server.sft': 'file_type': 'user'
- lib
 - 'lib_ringbuffer/build/libringBuffer.a': 'file_type': 'user', 'copyto':
,','
- header
 - 'lib_ringbuffer/ringBuffer.h': 'file_type': 'cSource', 'is_include_file':
True
- tb
 - 'tb/tb_vpi.v': 'file_type': 'verilogSource'

3.3 Targets

3.3.1 fusesoc_info Targets

- default
 - Info: Intergration default target for simulations.
- sim
 - Info: Test VPI TCP server.

3.4 Directory Guide

Below highlights important folders from the root of the directory.

1. **docs** Contains all documentation related to this project.
 - **manual** Contains user manual and github page that are generated from the latex sources.
2. **src** Contains source files for tcp server vpi interface.
3. **tb** Contains test bench files.

4 Simulation

A barebones test bench for iverilog is included in `tb/tb_vpi.v` . This can be run from `fusesoc` with the following.

```
$ fusesoc run --target=sim AFRL:vpi:tcp_server:1.0.0
```

5 Code Documentation

- **TCP SERVER FILE SOURCE, DOXYGEN**

The next section documents the library.

TCP_SERVER

1.0

Generated by Doxygen 1.9.8

1 Data Structure Documentation	1
1.1 s_process_data Struct Reference	1
1.1.1 Field Documentation	1
1.1.1.1 arg1_handle	1
1.1.1.2 arg2_handle	1
1.1.1.3 array_byte_size	1
1.1.1.4 num_ab_val_pairs	1
1.1.1.5 p_ringbuffer	2
1.1.1.6 systf_handle	2
1.1.1.7 thread	2
1.2 s_send_tcp_server Struct Reference	2
1.2.1 Field Documentation	3
1.2.1.1 connection_thread	3
1.2.1.2 kill_thread	3
1.2.1.3 p_address	3
1.2.1.4 p_socket_info	3
1.2.1.5 poll_connection	3
1.2.1.6 port	3
1.2.1.7 recv_process_data	3
1.2.1.8 send_process_data	3
1.2.1.9 systf_handle	3
2 File Documentation	5
2.1 recv_tcp_server.c File Reference	5
2.1.1 Detailed Description	6
2.1.2 Function Documentation	6
2.1.2.1 recv_tcp_server_calltf()	6
2.1.2.2 recv_tcp_server_compiletf()	6
2.1.2.3 recv_tcp_server_end_sim_cb()	7
2.1.2.4 recv_tcp_server_start_sim_cb()	7
2.1.2.5 recv_thread()	7
2.2 recv_tcp_server.h File Reference	7
2.2.1 Detailed Description	8
2.2.2 Function Documentation	8
2.2.2.1 recv_tcp_server_calltf()	8
2.2.2.2 recv_tcp_server_compiletf()	9
2.3 recv_tcp_server.h	9
2.4 send_tcp_server.c File Reference	9
2.4.1 Detailed Description	10
2.4.2 Function Documentation	10
2.4.2.1 send_tcp_server_calltf()	10
2.4.2.2 send_tcp_server_compiletf()	10

2.4.2.3 send_tcp_server_end_sim_cb()	10
2.4.2.4 send_tcp_server_start_sim_cb()	11
2.4.2.5 send_thread()	11
2.5 send_tcp_server.h File Reference	11
2.5.1 Detailed Description	12
2.5.2 Function Documentation	12
2.5.2.1 send_tcp_server_calltf()	12
2.5.2.2 send_tcp_server_completf()	12
2.6 send_tcp_server.h	13
2.7 tcp_server.c File Reference	13
2.7.1 Function Documentation	14
2.7.1.1 connection_keep_alive()	14
2.7.1.2 recv_tcp_server_reg_systf()	14
2.7.1.3 send_tcp_server_reg_systf()	14
2.7.1.4 setup_tcp_server_calltf()	14
2.7.1.5 setup_tcp_server_completf()	14
2.7.1.6 setup_tcp_server_end_sim_cb()	14
2.7.1.7 setup_tcp_server_reg_systf()	15
2.7.1.8 setup_tcp_server_start_sim_cb()	15
2.7.1.9 tcp_server_sizetf()	15
2.7.2 Variable Documentation	15
2.7.2.1 g_num_of_connections	15
2.7.2.2 g_send_tcp_server	15
2.7.2.3 vlog_startup_routines	15
2.8 tcp_server.h File Reference	16
2.8.1 Detailed Description	17
2.8.2 Macro Definition Documentation	18
2.8.2.1 BUFFSIZE	18
2.8.2.2 DATAHUNK	18
2.8.2.3 MAX_CONNECTIONS	18
2.8.2.4 RECV_NAME	18
2.8.2.5 SEND_NAME	18
2.8.2.6 SETUP_NAME	18
2.8.3 Variable Documentation	18
2.8.3.1 g_send_tcp_server	18
2.9 tcp_server.h	19
Index	21

Chapter 1

Data Structure Documentation

1.1 `s_process_data` Struct Reference

```
#include <tcp_server.h>
```

Data Fields

- `PLI_INT32` [num_ab_val_pairs](#)
- `PLI_INT32` [array_byte_size](#)
- `struct s_ringBuffer *` [p_ringbuffer](#)
- `pthread_t` [thread](#)
- `vpiHandle` [systf_handle](#)
- `vpiHandle` [arg1_handle](#)
- `vpiHandle` [arg2_handle](#)

1.1.1 Field Documentation

1.1.1.1 `arg1_handle`

```
vpiHandle s_process_data::arg1_handle
```

1.1.1.2 `arg2_handle`

```
vpiHandle s_process_data::arg2_handle
```

1.1.1.3 `array_byte_size`

```
PLI_INT32 s_process_data::array_byte_size
```

1.1.1.4 `num_ab_val_pairs`

```
PLI_INT32 s_process_data::num_ab_val_pairs
```

1.1.1.5 p_ringbuffer

```
struct s_ringBuffer* s_process_data::p_ringbuffer
```

1.1.1.6 systf_handle

```
vpiHandle s_process_data::systf_handle
```

1.1.1.7 thread

```
pthread_t s_process_data::thread
```

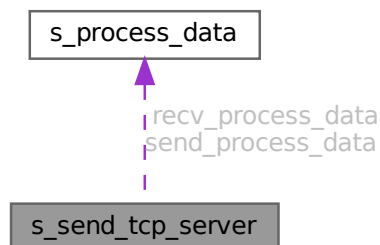
The documentation for this struct was generated from the following file:

- [tcp_server.h](#)

1.2 s_send_tcp_server Struct Reference

```
#include <tcp_server.h>
```

Collaboration diagram for s_send_tcp_server:



Data Fields

- int [kill_thread](#)
- struct pollfd [poll_connection](#)
- struct sockaddr_in * [p_socket_info](#)
- pthread_t [connection_thread](#)
- char * [p_address](#)
- unsigned short [port](#)
- vpiHandle [systf_handle](#)
- struct [s_process_data](#) [recv_process_data](#)
- struct [s_process_data](#) [send_process_data](#)

1.2.1 Field Documentation

1.2.1.1 connection_thread

pthread_t s_send_tcp_server::connection_thread

1.2.1.2 kill_thread

int s_send_tcp_server::kill_thread

1.2.1.3 p_address

char* s_send_tcp_server::p_address

1.2.1.4 p_socket_info

struct sockaddr_in* s_send_tcp_server::p_socket_info

1.2.1.5 poll_connection

struct pollfd s_send_tcp_server::poll_connection

1.2.1.6 port

unsigned short s_send_tcp_server::port

1.2.1.7 recv_process_data

struct [s_process_data](#) s_send_tcp_server::recv_process_data

1.2.1.8 send_process_data

struct [s_process_data](#) s_send_tcp_server::send_process_data

1.2.1.9 systf_handle

vpiHandle s_send_tcp_server::systf_handle

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

- [tcp_server.h](#)

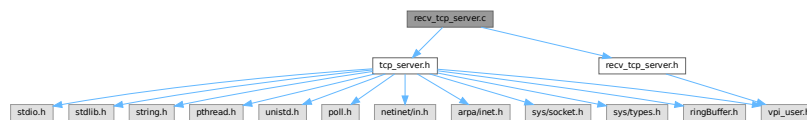
Chapter 2

File Documentation

2.1 recv_tcp_server.c File Reference

Functions for TCP server data receive.

```
#include "tcp_server.h"
#include "recv_tcp_server.h"
Include dependency graph for recv_tcp_server.c:
```



Functions

- void * [recv_thread](#) (void *data)
RECV TCP SETUP THREAD TO FILL RINGBUFFER.
- PLI_INT32 [recv_tcp_server_end_sim_cb](#) (p_cb_data data)
RECEIVE TCP SERVER DATA END SIM CALLBACK.
- PLI_INT32 [recv_tcp_server_start_sim_cb](#) (p_cb_data data)
RECEIVE TCP SERVER DATA START SIM CALLBACK.
- PLI_INT32 [recv_tcp_server_compiletf](#) (PLI_BYTE8 *user_data)
Compile time call, check the arguments for validity.
- PLI_INT32 [recv_tcp_server_calltf](#) (PLI_BYTE8 *user_data)
recv_tcp_server_calltf is a callback for the recv_tcp_server function.

2.1.1 Detailed Description

Functions for TCP server data receive.

Author

Jay Convertino(johnathan.convertino.1@us.af.mil)

Date

2024-02-22

@LICENSE MIT Copyright 2024 Jay Convertino

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.

2.1.2 Function Documentation

2.1.2.1 `recv_tcp_server_calltf()`

```
PLI_INT32 recv_tcp_server_calltf (
    PLI_BYTE8 * user_data )
```

`recv_tcp_server_calltf` is a callback for the `recv_tcp_server` function.

`read_binary_calltf` is a callback for the `recv_tcp_server` function.

2.1.2.2 `recv_tcp_server_compiletf()`

```
PLI_INT32 recv_tcp_server_compiletf (
    PLI_BYTE8 * user_data )
```

Compile time call, check the arguments for validity.

RECEIVE TCP SERVER DATA COMPILE SETUP.

2.1.2.3 recv_tcp_server_end_sim_cb()

```
PLI_INT32 recv_tcp_server_end_sim_cb (  
    p_cb_data data )
```

RECEIVE TCP SERVER DATA END SIM CALLBACK.

2.1.2.4 recv_tcp_server_start_sim_cb()

```
PLI_INT32 recv_tcp_server_start_sim_cb (  
    p_cb_data data )
```

RECEIVE TCP SERVER DATA START SIM CALLBACK.

2.1.2.5 recv_thread()

```
void * recv_thread (  
    void * data )
```

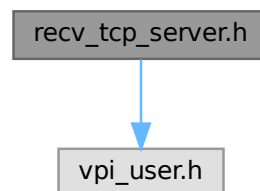
RECV TCP SETUP THREAD TO FILL RINGBUFFER.

2.2 recv_tcp_server.h File Reference

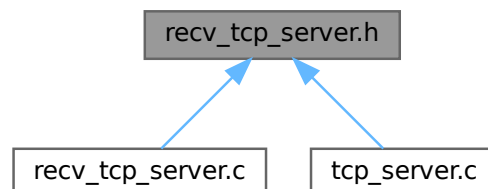
Functions for TCP server data receive.

```
#include <vpi_user.h>
```

Include dependency graph for recv_tcp_server.h:



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



Functions

- PLI_INT32 `recv_tcp_server_compiletf` (PLI_BYTE8 *user_data)
RECEIVE TCP SERVER DATA COMPILE SETUP.
- PLI_INT32 `recv_tcp_server_calltf` (PLI_BYTE8 *user_data)
read_binary_calltf is a callback for the recv_tcp_server function.

2.2.1 Detailed Description

Functions for TCP server data receive.

Author

Jay Convertino(johnathan.convertino.1@us.af.mil)

Date

2024-02-22

\$recv_tcp_server takes 2 arguments. First the fd returned from \$setup_tcp_server, and then a register for data in size bytes. The function returns the number of bytes read.

@LICENSE MIT Copyright 2024 Jay Convertino

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.

2.2.2 Function Documentation

2.2.2.1 `recv_tcp_server_calltf()`

```
PLI_INT32 recv_tcp_server_calltf (
    PLI_BYTE8 * user_data )
```

`read_binary_calltf` is a callback for the `recv_tcp_server` function.

`read_binary_calltf` is a callback for the `recv_tcp_server` function.

2.2.2.2 recv_tcp_server_compiletf()

```
PLI_INT32 recv_tcp_server_compiletf (
    PLI_BYTE8 * user_data )
```

RECEIVE TCP SERVER DATA COMPILE SETUP.

RECEIVE TCP SERVER DATA COMPILE SETUP.

2.3 recv_tcp_server.h

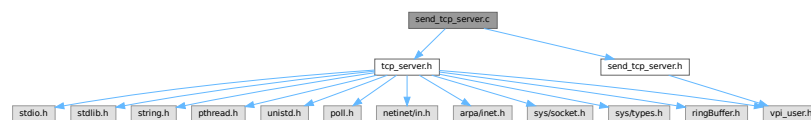
[Go to the documentation of this file.](#)

```
00001 //*****
00030 //*****
00031
00032 #ifndef __RECV_TCP_SERVER
00033 #define __RECV_TCP_SERVER
00034
00035 // Include the VPI library of routines (object based).
00036 #include <vpi_user.h>
00037
00038 //*****
00040 //*****
00041 PLI_INT32 recv_tcp_server_compiletf(PLI_BYTE8 *user_data);
00042
00043 //*****
00045 //*****
00046 PLI_INT32 recv_tcp_server_calltf(PLI_BYTE8 *user_data);
00047
00048 #endif
```

2.4 send_tcp_server.c File Reference

Functions for TCP server data send.

```
#include "tcp_server.h"
#include "send_tcp_server.h"
Include dependency graph for send_tcp_server.c:
```



Functions

- void * [send_thread](#) (void *data)
SEND TCP SERVER THREAD TO EMPTY RINGBUFFER.
- PLI_INT32 [send_tcp_server_end_sim_cb](#) (p_cb_data data)
SEND TCP SERVER DATA END SIM CALLBACK.
- PLI_INT32 [send_tcp_server_start_sim_cb](#) (p_cb_data data)
SEND TCP SERVER DATA START SIM CALLBACK.
- PLI_INT32 [send_tcp_server_compiletf](#) (PLI_BYTE8 *user_data)
Compile time call, check the arguments for validity.
- PLI_INT32 [send_tcp_server_calltf](#) (PLI_BYTE8 *user_data)
Called by the simulator, each time it is requested.

2.4.1 Detailed Description

Functions for TCP server data send.

Author

Jay Convertino(johnathan.convertino.1@us.af.mil)

Date

2024-23-02

@LICENSE MIT Copyright 2024 Jay Convertino

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.

2.4.2 Function Documentation

2.4.2.1 send_tcp_server_calltf()

```
PLI_INT32 send_tcp_server_calltf (
    PLI_BYTE8 * user_data )
```

Called by the simulator, each time it is requested.

2.4.2.2 send_tcp_server_compiletf()

```
PLI_INT32 send_tcp_server_compiletf (
    PLI_BYTE8 * user_data )
```

Compile time call, check the arguments for validity.

SEND TCP SERVER DATA COMPILE SETUP.

2.4.2.3 send_tcp_server_end_sim_cb()

```
PLI_INT32 send_tcp_server_end_sim_cb (
    p_cb_data data )
```

SEND TCP SERVER DATA END SIM CALLBACK.

2.4.2.4 send_tcp_server_start_sim_cb()

```
PLI_INT32 send_tcp_server_start_sim_cb (  
    p_cb_data data )
```

SEND TCP SERVER DATA START SIM CALLBACK.

2.4.2.5 send_thread()

```
void * send_thread (  
    void * data )
```

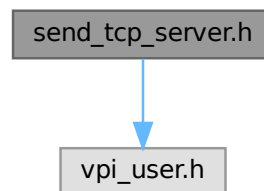
SEND TCP SERVER THREAD TO EMPTY RINGBUFFER.

2.5 send_tcp_server.h File Reference

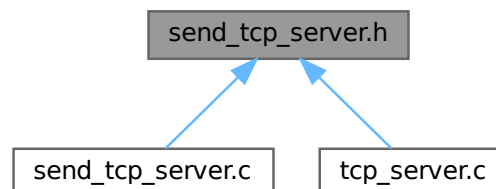
Function to send data over a tcp server.

```
#include <vpi_user.h>
```

Include dependency graph for send_tcp_server.h:



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



Functions

- PLI_INT32 [send_tcp_server_compiletf](#) (PLI_BYTE8 *user_data)
SEND TCP SERVER DATA COMPILE SETUP.
- PLI_INT32 [send_tcp_server_calltf](#) (PLI_BYTE8 *user_data)
Called by the simulator, each time it is requested.

2.5.1 Detailed Description

Function to send data over a tcp server.

Author

Jay Convertino(johnathan.convertino.1@us.af.mil)

Date

2024-24-2

@LICENSE MIT Copyright 2024 Jay Convertino

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.

2.5.2 Function Documentation

2.5.2.1 send_tcp_server_calltf()

```
PLI_INT32 send_tcp_server_calltf (
    PLI_BYTE8 * user_data )
```

Called by the simulator, each time it is requested.

2.5.2.2 send_tcp_server_compiletf()

```
PLI_INT32 send_tcp_server_compiletf (
    PLI_BYTE8 * user_data )
```

SEND TCP SERVER DATA COMPILE SETUP.

SEND TCP SERVER DATA COMPILE SETUP.

2.6 send_tcp_server.h

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

```

00001 //*****
00027 //*****
00028
00029 #ifndef __SEND_TCP_SERVER
00030 #define __SEND_TCP_SERVER
00031
00032 // Include the VPI library of routines (object based).
00033 #include <vpi_user.h>
00034
00035 //*****
00037 //*****
00038 PLI_INT32 send_tcp_server_compiletf (PLI_BYTE8 *user_data);
00039
00040 //*****
00042 //*****
00043 PLI_INT32 send_tcp_server_calltf (PLI_BYTE8 *user_data);
00044
00045 #endif

```

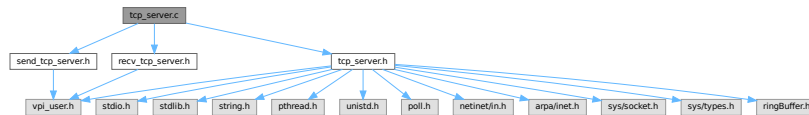
2.7 tcp_server.c File Reference

```

#include "send_tcp_server.h"
#include "recv_tcp_server.h"
#include "tcp_server.h"

```

Include dependency graph for tcp_server.c:



Functions

- void * [connection_keep_alive](#) (void *p_data)
- PLI_INT32 [setup_tcp_server_start_sim_cb](#) (p_cb_data data)
SETUP TCP SERVER DATA START SIM CALLBACK.
- PLI_INT32 [setup_tcp_server_end_sim_cb](#) (p_cb_data data)
SETUP TCP SERVER END SIM CALLBACK.
- PLI_INT32 [tcp_server_sizetf](#) (PLI_BYTE8 *user_data)
Returns the size, in bits, of the function return type.
- PLI_INT32 [setup_tcp_server_compiletf](#) (PLI_BYTE8 *user_data)
Compile time call, check the arguments for validity.
- PLI_INT32 [setup_tcp_server_calltf](#) (PLI_BYTE8 *user_data)
setup_tcp_server_calltf is the callback for the setup_tcp_server function.
- void [recv_tcp_server_reg_systf](#) (void)
Setup recv_tcp_server function.
- void [send_tcp_server_reg_systf](#) (void)
Setup send_tcp_server function.
- void [setup_tcp_server_reg_systf](#) (void)
Setup setup_tcp_server function.

Variables

- unsigned int `g_num_of_connections` = 0
- struct `s_send_tcp_server` `g_send_tcp_server` [MAX_CONNECTIONS]
- void(* `vlog_startup_routines` [])(void)
register the new file functions

2.7.1 Function Documentation

2.7.1.1 connection_keep_alive()

```
void * connection_keep_alive (
    void * p_data )
```

2.7.1.2 recv_tcp_server_reg_systf()

```
void recv_tcp_server_reg_systf (
    void )
```

Setup `recv_tcp_server` function.

2.7.1.3 send_tcp_server_reg_systf()

```
void send_tcp_server_reg_systf (
    void )
```

Setup `send_tcp_server` function.

2.7.1.4 setup_tcp_server_calltf()

```
PLI_INT32 setup_tcp_server_calltf (
    PLI_BYTE8 * user_data )
```

`setup_tcp_server_calltf` is the callback for the `setup_tcp_server` function.

2.7.1.5 setup_tcp_server_compiletf()

```
PLI_INT32 setup_tcp_server_compiletf (
    PLI_BYTE8 * user_data )
```

Compile time call, check the arguments for validity.

2.7.1.6 setup_tcp_server_end_sim_cb()

```
PLI_INT32 setup_tcp_server_end_sim_cb (
    p_cb_data data )
```

SETUP TCP SERVER END SIM CALLBACK.

2.7.1.7 setup_tcp_server_reg_systf()

```
void setup_tcp_server_reg_systf (
    void )
```

Setup setup_tcp_server function.

2.7.1.8 setup_tcp_server_start_sim_cb()

```
PLI_INT32 setup_tcp_server_start_sim_cb (
    p_cb_data data )
```

SETUP TCP SERVER DATA START SIM CALLBACK.

2.7.1.9 tcp_server_sizetf()

```
PLI_INT32 tcp_server_sizetf (
    PLI_BYTE8 * user_data )
```

Returns the size, in bits, of the function return type.

2.7.2 Variable Documentation

2.7.2.1 g_num_of_connections

```
unsigned int g_num_of_connections = 0
```

2.7.2.2 g_send_tcp_server

```
struct s_send_tcp_server g_send_tcp_server[MAX_CONNECTIONS]
```

2.7.2.3 vlog_startup_routines

```
void(* vlog_startup_routines[]) (void) (
    void )
```

Initial value:

```
= {
    recv_tcp_server_reg_systf,
    send_tcp_server_reg_systf,
    setup_tcp_server_reg_systf,
    0
}
```

register the new file functions

2.8 tcp_server.h File Reference

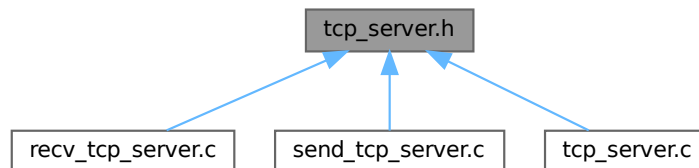
Functions to write raw binary files properly in verilog.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
#include <poll.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <vpi_user.h>
#include "ringBuffer.h"
```

Include dependency graph for tcp_server.h:



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



Data Structures

- struct [s_process_data](#)
- struct [s_send_tcp_server](#)

Macros

- #define [BUFFSIZE](#) (1 << 23)
- #define [DATAHUNK](#) (1 << 21)
- #define [MAX_CONNECTIONS](#) 256
- #define [RCV_NAME](#) "\$recv_tcp_server"
- #define [SEND_NAME](#) "\$send_tcp_server"
- #define [SETUP_NAME](#) "\$setup_tcp_server"

Variables

- struct `s_send_tcp_server` `g_send_tcp_server` [`MAX_CONNECTIONS`]

2.8.1 Detailed Description

Functions to write raw binary files properly in verilog.

Functions to create multiple TCP servers.

Author

Jay Convertino(`johnathan.convertino.1@us.af.mil`)

Date

2024-22-02

@LICENSE MIT Copyright 2024 Jay Convertino

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.

Author

Jay Convertino(`johnathan.convertino.1@us.af.mil`)

Date

2024-23-02

@LICENSE MIT Copyright 2024 Jay Convertino

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.

2.8.2 Macro Definition Documentation

2.8.2.1 BUFSIZE

```
#define BUFSIZE (1 << 23)
```

2.8.2.2 DATACHUNK

```
#define DATACHUNK (1 << 21)
```

2.8.2.3 MAX_CONNECTIONS

```
#define MAX_CONNECTIONS 256
```

2.8.2.4 RECV_NAME

```
#define RECV_NAME "$recv_tcp_server"
```

2.8.2.5 SEND_NAME

```
#define SEND_NAME "$send_tcp_server"
```

2.8.2.6 SETUP_NAME

```
#define SETUP_NAME "$setup_tcp_server"
```

2.8.3 Variable Documentation

2.8.3.1 g_send_tcp_server

```
struct s_send_tcp_server g_send_tcp_server[MAX_CONNECTIONS] [extern]
```

2.9 tcp_server.h

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

```

00001 //*****
00027 //*****
00028
00029 #ifndef __VPI_TCP_SERVER
00030 #define __VPI_TCP_SERVER
00031
00032 // c standard libraries
00033 #include <stdio.h>
00034 #include <stdlib.h>
00035 #include <string.h>
00036 // other libs
00037 // threads
00038 #include <pthread.h>
00039 // tcp
00040 #include <unistd.h>
00041 #include <poll.h>
00042 #include <netinet/in.h>
00043 #include <arpa/inet.h>
00044 #include <sys/socket.h>
00045 #include <sys/types.h>
00046 // Include the VPI library of routines (object based).
00047 #include <vpi_user.h>
00048 // include ringbuffer library
00049 #include "ringBuffer.h"
00050
00051 //ring buffer sizes
00052 // 4 MB
00053 #define BUFFSIZE (1 « 23)
00054 // 1 MB
00055 #define DATACHUNK (1 « 21)
00056
00057 #define MAX_CONNECTIONS 256
00058
00059 #define RECV_NAME "$recv_tcp_server"
00060 #define SEND_NAME "$send_tcp_server"
00061 #define SETUP_NAME "$setup_tcp_server"
00062
00063 struct s_process_data
00064 {
00065     // PLI_INT32 error;
00066     PLI_INT32 num_ab_val_pairs;
00067     PLI_INT32 array_byte_size;
00068
00069     struct s_ringBuffer *p_ringbuffer;
00070
00071     pthread_t thread;
00072
00073     vpiHandle systf_handle;
00074     vpiHandle arg1_handle;
00075     vpiHandle arg2_handle;
00076 };
00077
00078 struct s_send_tcp_server
00079 {
00080     int kill_thread;
00081
00082     struct pollfd poll_connection;
00083     struct sockaddr_in *p_socket_info;
00084
00085     pthread_t connection_thread;
00086
00087     char *p_address;
00088     unsigned short port;
00089
00090     vpiHandle systf_handle;
00091
00092     struct s_process_data recv_process_data;
00093     struct s_process_data send_process_data;
00094 };
00095
00096 extern struct s_send_tcp_server g_send_tcp_server[MAX_CONNECTIONS];
00097
00098 #endif

```


Index

arg1_handle
 s_process_data, 1
arg2_handle
 s_process_data, 1
array_byte_size
 s_process_data, 1

BUFSIZE
 tcp_server.h, 18

connection_keep_alive
 tcp_server.c, 14
connection_thread
 s_send_tcp_server, 3

DATAHUNK
 tcp_server.h, 18

g_num_of_connections
 tcp_server.c, 15
g_send_tcp_server
 tcp_server.c, 15
 tcp_server.h, 18

kill_thread
 s_send_tcp_server, 3

MAX_CONNECTIONS
 tcp_server.h, 18

num_ab_val_pairs
 s_process_data, 1

p_address
 s_send_tcp_server, 3
p_ringbuffer
 s_process_data, 1
p_socket_info
 s_send_tcp_server, 3
poll_connection
 s_send_tcp_server, 3
port
 s_send_tcp_server, 3

RECV_NAME
 tcp_server.h, 18
recv_process_data
 s_send_tcp_server, 3
recv_tcp_server.c, 5
 recv_tcp_server_calltf, 6
 recv_tcp_server_completf, 6
 recv_tcp_server_end_sim_cb, 6
 recv_tcp_server_start_sim_cb, 7
 recv_thread, 7
recv_tcp_server.h, 7
 recv_tcp_server_calltf, 8
 recv_tcp_server_completf, 8
recv_tcp_server_calltf
 recv_tcp_server.c, 6
 recv_tcp_server.h, 8
recv_tcp_server_completf
 recv_tcp_server.c, 6
 recv_tcp_server.h, 8
recv_tcp_server_end_sim_cb
 recv_tcp_server.c, 6
recv_tcp_server_reg_systf
 tcp_server.c, 14
recv_tcp_server_start_sim_cb
 recv_tcp_server.c, 7
recv_thread
 recv_tcp_server.c, 7

s_process_data, 1
 arg1_handle, 1
 arg2_handle, 1
 array_byte_size, 1
 num_ab_val_pairs, 1
 p_ringbuffer, 1
 systf_handle, 2
 thread, 2
s_send_tcp_server, 2
 connection_thread, 3
 kill_thread, 3
 p_address, 3
 p_socket_info, 3
 poll_connection, 3
 port, 3
 recv_process_data, 3
 send_process_data, 3
 systf_handle, 3
SEND_NAME
 tcp_server.h, 18
send_process_data
 s_send_tcp_server, 3
send_tcp_server.c, 9
 send_tcp_server_calltf, 10
 send_tcp_server_completf, 10
 send_tcp_server_end_sim_cb, 10
 send_tcp_server_start_sim_cb, 10
 send_thread, 11
send_tcp_server.h, 11

- send_tcp_server_calltf, 12
 - send_tcp_server_compiletf, 12
- send_tcp_server_calltf
 - send_tcp_server.c, 10
 - send_tcp_server.h, 12
- send_tcp_server_compiletf
 - send_tcp_server.c, 10
 - send_tcp_server.h, 12
- send_tcp_server_end_sim_cb
 - send_tcp_server.c, 10
- send_tcp_server_reg_systf
 - tcp_server.c, 14
- send_tcp_server_start_sim_cb
 - send_tcp_server.c, 10
- send_thread
 - send_tcp_server.c, 11
- SETUP_NAME
 - tcp_server.h, 18
- setup_tcp_server_calltf
 - tcp_server.c, 14
- setup_tcp_server_compiletf
 - tcp_server.c, 14
- setup_tcp_server_end_sim_cb
 - tcp_server.c, 14
- setup_tcp_server_reg_systf
 - tcp_server.c, 14
- setup_tcp_server_start_sim_cb
 - tcp_server.c, 15
- systf_handle
 - s_process_data, 2
 - s_send_tcp_server, 3
- tcp_server.c, 13
 - connection_keep_alive, 14
 - g_num_of_connections, 15
 - g_send_tcp_server, 15
 - recv_tcp_server_reg_systf, 14
 - send_tcp_server_reg_systf, 14
 - setup_tcp_server_calltf, 14
 - setup_tcp_server_compiletf, 14
 - setup_tcp_server_end_sim_cb, 14
 - setup_tcp_server_reg_systf, 14
 - setup_tcp_server_start_sim_cb, 15
 - tcp_server_sizetf, 15
 - vlog_startup_routines, 15
- tcp_server.h, 16
 - BUFFSIZE, 18
 - DATACHUNK, 18
 - g_send_tcp_server, 18
 - MAX_CONNECTIONS, 18
 - RECV_NAME, 18
 - SEND_NAME, 18
 - SETUP_NAME, 18
- tcp_server_sizetf
 - tcp_server.c, 15
- thread
 - s_process_data, 2
- vlog_startup_routines