

# Host Communication Protocol

## 2.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Main</b>	<b>1</b>
1.1	FPC embedded stack and HCP . . . . .	1
1.2	Command flow specification . . . . .	1
<b>2</b>	<b>1_stack</b>	<b>3</b>
<b>3</b>	<b>FPC embedded stack</b>	<b>5</b>
3.1	Physical . . . . .	5
3.2	Link . . . . .	5
3.3	Transport . . . . .	6
3.4	Application . . . . .	6
3.5	HCP . . . . .	6
<b>4</b>	<b>2_hcpframe</b>	<b>7</b>
<b>5</b>	<b>HCP frame format</b>	<b>9</b>
5.1	Command . . . . .	9
5.2	Argument . . . . .	9
<b>6</b>	<b>4_biometrics</b>	<b>11</b>
<b>7</b>	<b>Biometrics</b>	<b>13</b>
7.1	Capture . . . . .	13
7.2	Extract . . . . .	13
7.3	Enroll . . . . .	13
7.4	Identify . . . . .	13

<b>8</b>	<b>5_image</b>	<b>15</b>
<b>9</b>	<b>Image handling</b>	<b>17</b>
9.1	Create . . . . .	17
9.2	Upload . . . . .	17
9.3	Download . . . . .	17
<b>10</b>	<b>6_template</b>	<b>19</b>
<b>11</b>	<b>Template handling</b>	<b>21</b>
11.1	Upload . . . . .	21
11.2	Download . . . . .	21
11.3	Save . . . . .	21
<b>12</b>	<b>7_storage</b>	<b>23</b>
<b>13</b>	<b>Storage handling</b>	<b>25</b>
13.1	Delete ID . . . . .	25
13.2	Delete All . . . . .	25
13.3	Upload . . . . .	25
13.4	Count . . . . .	25
13.5	Get IDs . . . . .	25
<b>14</b>	<b>8_sensor</b>	<b>27</b>
<b>15</b>	<b>Sensor operations</b>	<b>29</b>
15.1	Wait for finger up . . . . .	29
15.2	Wait for finger down . . . . .	29
15.3	Reset sensor . . . . .	29
<b>16</b>	<b>9_device</b>	<b>31</b>
<b>17</b>	<b>Device operations</b>	<b>33</b>
17.1	Reset device . . . . .	33

<b>18 Data Structure Index</b>	<b>35</b>
18.1 Data Structures . . . . .	35
<b>19 File Index</b>	<b>37</b>
19.1 File List . . . . .	37
<b>20 Data Structure Documentation</b>	<b>39</b>
20.1 fpc_com_chain Struct Reference . . . . .	39
20.1.1 Detailed Description . . . . .	40
20.1.2 Field Documentation . . . . .	40
20.1.2.1 app_mtu_buffer . . . . .	40
20.1.2.2 app_mtu_size . . . . .	41
20.1.2.3 app_overhead_get . . . . .	41
20.1.2.4 app_packet_size . . . . .	41
20.1.2.5 app_rx . . . . .	41
20.1.2.6 app_tx . . . . .	41
20.1.2.7 argument_allocator . . . . .	41
20.1.2.8 argument_free . . . . .	41
20.1.2.9 channel . . . . .	42
20.1.2.10 context . . . . .	42
20.1.2.11 crc_calc . . . . .	42
20.1.2.12 initialized . . . . .	42
20.1.2.13 link_overhead_get . . . . .	42
20.1.2.14 phy_mtu_buffer . . . . .	42
20.1.2.15 phy_mtu_size . . . . .	42
20.1.2.16 phy_rx . . . . .	43
20.1.2.17 phy_timeout_rx . . . . .	43
20.1.2.18 phy_timeout_tx . . . . .	43
20.1.2.19 phy_tx . . . . .	43
20.1.2.20 private_vars . . . . .	43
20.1.2.21 session . . . . .	43

20.1.2.22	<a href="#">tsp_overhead_get</a>	43
20.1.2.23	<a href="#">tsp_rx</a>	44
20.1.2.24	<a href="#">tsp_tx</a>	44
20.2	<a href="#">fpc_com_chain_private Struct Reference</a>	44
20.2.1	Detailed Description	45
20.2.2	Field Documentation	45
20.2.2.1	<a href="#">hcp_packet</a>	45
20.2.2.2	<a href="#">hcp_seq_len</a>	45
20.2.2.3	<a href="#">hcp_seq_nr</a>	45
20.3	<a href="#">fpc_com_packet_link Struct Reference</a>	45
20.3.1	Detailed Description	45
20.3.2	Field Documentation	46
20.3.2.1	<a href="#">channel</a>	46
20.3.2.2	<a href="#">crc</a>	46
20.3.2.3	<a href="#">data</a>	46
20.3.2.4	<a href="#">size</a>	46
20.4	<a href="#">fpc_com_packet_transport Struct Reference</a>	46
20.4.1	Detailed Description	46
20.4.2	Field Documentation	47
20.4.2.1	<a href="#">data</a>	47
20.4.2.2	<a href="#">seq_len</a>	47
20.4.2.3	<a href="#">seq_nr</a>	47
20.4.2.4	<a href="#">size</a>	47
20.5	<a href="#">fpc_hcp_arg_data Struct Reference</a>	47
20.5.1	Detailed Description	48
20.5.2	Field Documentation	48
20.5.2.1	<a href="#">arg</a>	48
20.5.2.2	<a href="#">data</a>	48
20.5.2.3	<a href="#">free_data</a>	48
20.5.2.4	<a href="#">size</a>	48
20.6	<a href="#">fpc_hcp_packet Struct Reference</a>	48
20.6.1	Detailed Description	49
20.6.2	Field Documentation	49
20.6.2.1	<a href="#">arguments</a>	49
20.6.2.2	<a href="#">id</a>	49
20.6.2.3	<a href="#">num_args</a>	49

<b>21 File Documentation</b>	<b>51</b>
21.1 doc/md/1_stack.md File Reference . . . . .	51
21.2 doc/md/2_hcpframe.md File Reference . . . . .	51
21.3 doc/md/4_biometrics.md File Reference . . . . .	51
21.4 doc/md/5_image.md File Reference . . . . .	51
21.5 doc/md/6_template.md File Reference . . . . .	51
21.6 doc/md/7_storage.md File Reference . . . . .	51
21.7 doc/md/8_sensor.md File Reference . . . . .	51
21.8 doc/md/9_device.md File Reference . . . . .	51
21.9 hcp.md File Reference . . . . .	51
21.10inc/fpc_com_chain.h File Reference . . . . .	51
21.10.1 Detailed Description . . . . .	53
21.10.2 Typedef Documentation . . . . .	53
21.10.2.1 fpc_com_chain_private_t . . . . .	53
21.10.2.2 fpc_com_chain_t . . . . .	53
21.10.3 Enumeration Type Documentation . . . . .	53
21.10.3.1 fpc_com_chain_dir_t . . . . .	53
21.11inc/fpc_com_link.h File Reference . . . . .	54
21.11.1 Detailed Description . . . . .	55
21.11.2 Function Documentation . . . . .	55
21.11.2.1 fpc_com_link_get_overhead(uint16_t *offset) . . . . .	55
21.11.2.2 fpc_com_link_receive(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain) . . . . .	55
21.11.2.3 fpc_com_link_transmit(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain) . . . . .	55
21.12inc/fpc_com_packets.h File Reference . . . . .	56
21.12.1 Detailed Description . . . . .	57
21.12.2 Macro Definition Documentation . . . . .	58
21.12.2.1 FPC_COM_ACK . . . . .	58
21.12.3 Typedef Documentation . . . . .	58
21.12.3.1 fpc_com_channel_t . . . . .	58
21.12.3.2 fpc_com_packet_link_t . . . . .	58

21.12.3.3 fpc_com_packet_tsp_t . . . . .	58
21.12.4 Enumeration Type Documentation . . . . .	58
21.12.4.1 fpc_com_channel . . . . .	58
21.13inc/fpc_com_result.h File Reference . . . . .	59
21.13.1 Detailed Description . . . . .	60
21.13.2 Typedef Documentation . . . . .	60
21.13.2.1 fpc_com_result_t . . . . .	60
21.13.3 Enumeration Type Documentation . . . . .	60
21.13.3.1 fpc_com_result . . . . .	60
21.14inc/fpc_com_transport.h File Reference . . . . .	60
21.14.1 Detailed Description . . . . .	61
21.14.2 Function Documentation . . . . .	61
21.14.2.1 fpc_com_transport_get_overhead(uint16_t *offset) . . . . .	61
21.14.2.2 fpc_com_transport_receive(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain) . . . . .	62
21.14.2.3 fpc_com_transport_transmit(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain) . . . . .	62
21.15inc/fpc_hcp.h File Reference . . . . .	63
21.15.1 Detailed Description . . . . .	64
21.15.2 Function Documentation . . . . .	64
21.15.2.1 fpc_hcp_arg_add(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t size, bool free_data, void *data) . . . . .	64
21.15.2.2 fpc_hcp_arg_check(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg) . . . . .	65
21.15.2.3 fpc_hcp_arg_copy_data(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t data_size, uint8_t *data) . . . . .	65
21.15.2.4 fpc_hcp_arg_get(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg) . . . . .	66
21.15.2.5 fpc_hcp_free(fpc_com_chain_t *chain, fpc_hcp_packet_t *packet) . . . . .	66
21.15.2.6 fpc_hcp_get_size(fpc_hcp_packet_t *packet, uint16_t *num_args) . . . . .	67
21.15.2.7 fpc_hcp_receive(fpc_hcp_packet_t *packet, fpc_com_chain_t *chain) . . . . .	67
21.15.2.8 fpc_hcp_transmit(fpc_hcp_packet_t *packet, fpc_com_chain_t *chain) . . . . .	67
21.16inc/fpc_hcp_common.h File Reference . . . . .	68
21.16.1 Detailed Description . . . . .	72



21.16.2 Macro Definition Documentation . . . . .	72
21.16.2.1 ARG_APP_BASE_VAL . . . . .	72
21.16.2.2 CMD_APP_BASE_VAL . . . . .	72
21.16.2.3 HCP_MIN . . . . .	72
21.16.3 Typedef Documentation . . . . .	72
21.16.3.1 fpc_hcp_arg_data_t . . . . .	72
21.16.3.2 fpc_hcp_arg_t . . . . .	72
21.16.3.3 fpc_hcp_cmd_t . . . . .	72
21.16.3.4 fpc_hcp_packet_t . . . . .	73
21.16.4 Enumeration Type Documentation . . . . .	73
21.16.4.1 fpc_hcp_arg . . . . .	73
21.16.4.2 fpc_hcp_cmd . . . . .	74
21.17src/fpc_com_link.c File Reference . . . . .	75
21.17.1 Detailed Description . . . . .	76
21.17.2 Function Documentation . . . . .	76
21.17.2.1 fpc_com_link_get_overhead(uint16_t *offset) . . . . .	76
21.17.2.2 fpc_com_link_receive(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain) . . . . .	77
21.17.2.3 fpc_com_link_transmit(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain) . . . . .	77
21.18src/fpc_com_transport.c File Reference . . . . .	78
21.18.1 Detailed Description . . . . .	78
21.18.2 Function Documentation . . . . .	79
21.18.2.1 fpc_com_transport_get_overhead(uint16_t *offset) . . . . .	79
21.18.2.2 fpc_com_transport_receive(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain) . . . . .	79
21.18.2.3 fpc_com_transport_transmit(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain) . . . . .	80
21.19src/fpc_hcp.c File Reference . . . . .	80
21.19.1 Detailed Description . . . . .	82
21.19.2 Macro Definition Documentation . . . . .	82
21.19.2.1 ARGUMENT_ARG_SIZE . . . . .	82
21.19.2.2 ARGUMENT_HEADER_SIZE . . . . .	82

21.19.2.3 ARGUMENT_SIZE_SIZE . . . . .	82
21.19.2.4 PACKET_HEADER_SIZE . . . . .	82
21.19.2.5 PACKET_ID_SIZE . . . . .	82
21.19.2.6 PACKET_NUM_ARGS_SIZE . . . . .	82
21.19.3 Function Documentation . . . . .	82
21.19.3.1 fpc_hcp_arg_add(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t size, bool free_data, void *data) . . . . .	82
21.19.3.2 fpc_hcp_arg_check(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg) . . . . .	83
21.19.3.3 fpc_hcp_arg_copy_data(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t data_size, uint8_t *data) . . . . .	83
21.19.3.4 fpc_hcp_arg_get(fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg) . . . . .	84
21.19.3.5 fpc_hcp_free(fpc_com_chain_t *chain, fpc_hcp_packet_t *packet) . . . . .	84
21.19.3.6 fpc_hcp_get_size(fpc_hcp_packet_t *packet, uint16_t *num_args) . . . . .	85
21.19.3.7 fpc_hcp_receive(fpc_hcp_packet_t *packet, fpc_com_chain_t *chain) . . . . .	85
21.19.3.8 fpc_hcp_transmit(fpc_hcp_packet_t *packet, fpc_com_chain_t *chain) . . . . .	86
21.19.3.9 recieve_chunks(fpc_com_chain_t *chain) . . . . .	86
21.19.3.10transmit_chunks(fpc_com_chain_t *chain) . . . . .	87
<b>Index</b>	<b>89</b>

# Chapter 1

## Main

Welcome to the documentation for the Host Communication Protocol (HCP).

The first part covers the physical method of sending messages and the second part covers the specification of the different command flows.

### 1.1 FPC embedded stack and HCP

- [FPC embedded stack](#)
- [HCP frame format](#)

### 1.2 Command flow specification

- [Biometrics](#)
  - [Capture](#)
  - [Extract](#)
  - [Enroll](#)
  - [Identify](#)
- [Image handling](#)
- [Template handling](#)
- [Storage handling](#)
- [Sensor operations](#)
- [Device operations](#)



## Chapter 2

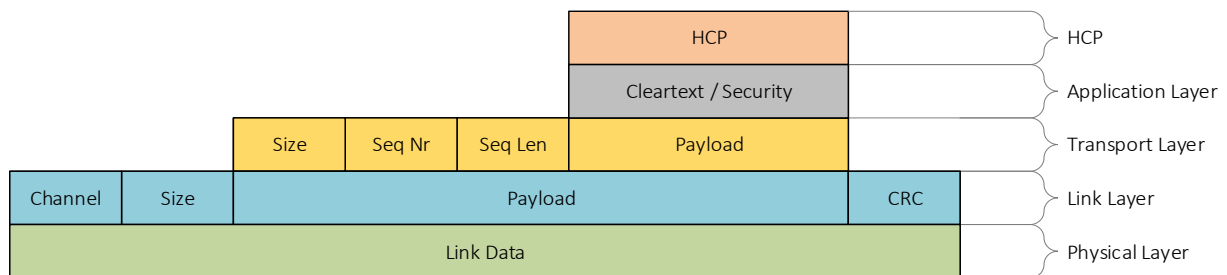
### 1\_stack



## Chapter 3

# FPC embedded stack

The communication stack implemented on the embedded devices by FPC follows the following specification.



**Figure 3.1 HCP embedded stack**

### 3.1 Physical

The physical layer have a fixed size buffer of 256 bytes.

### 3.2 Link

The link layer handles packet consistency.

Each packet received is acknowledged on the link layer, if an error occurs no retransmission is done on this level, instead the error is propagated upwards.

Channel	Size	Payload	CRC
2 bytes	2 bytes	size bytes	4 bytes

All fields are using unsigned data types.

### 3.3 Transport

The transport layer handles packet segmentation.

As the PHY MTU is 256 bytes the maximum payload per segment is 242 bytes.

Errors are propagated upwards.

Size	Seq Nr	Seq Len	Payload
2 bytes	2 bytes	2 bytes	size bytes

All fields are using unsigned data types.

### 3.4 Application

The application layer is a optional security layer, the default implementation is clear text (unsecure).

If a security solution is used it will be part of that products documentation.

### 3.5 HCP

The HCP frame is described in the [HCP frame format section](#).



## Chapter 4

### 2\_hcpframe



## Chapter 5

# HCP frame format

The Host Communication Protocol (HCP) describes a general way of sending commands and information between devices.

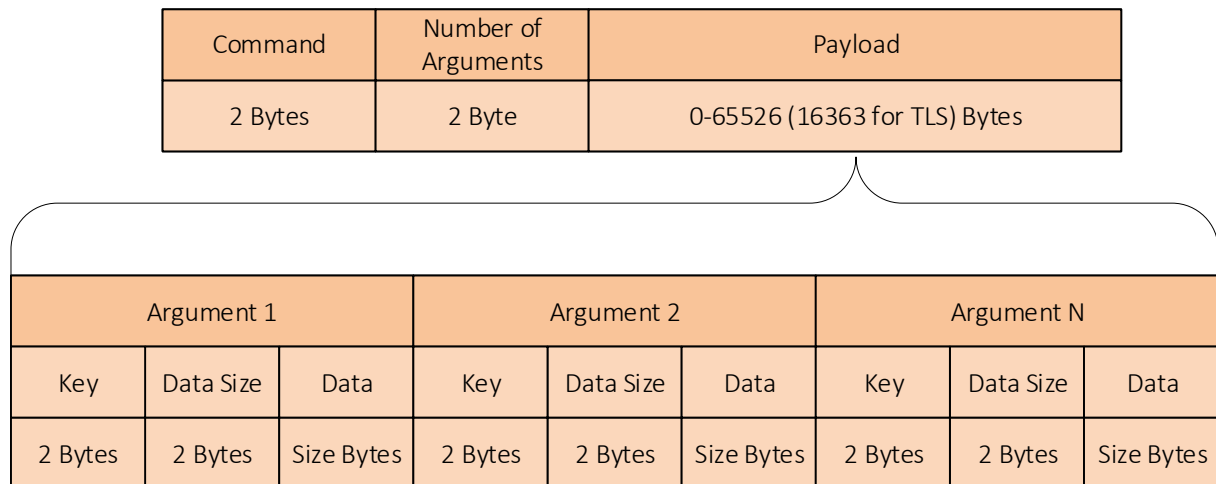


Figure 5.1 HCP frame format

### 5.1 Command

The Commands define the general action that is going to be executed. However, each command can have several Arguments each with data attached.

CMD	Num Args	Payload
2 bytes	2 bytes	xx bytes

All fields are using unsigned data types.

### 5.2 Argument

The Argument is used as a complement to the command if it is needed and can contain arbitrary data.

ARG	Size	Data
2 bytes	2 bytes	size bytes

All fields are using unsigned data types.

## **Chapter 6**

### **4\_biometrics**



## **Chapter 7**

# **Biometrics**

**7.1 Capture**

**7.2 Extract**

**7.3 Enroll**

**7.4 Identify**





## Chapter 8

### 5\_image



## **Chapter 9**

# **Image handling**

### **9.1 Create**

### **9.2 Upload**

### **9.3 Download**



## **Chapter 10**

### **6\_template**



## **Chapter 11**

# **Template handling**

**11.1 Upload**

**11.2 Download**

**11.3 Save**





## **Chapter 12**

### **7\_storage**



## **Chapter 13**

# **Storage handling**

**13.1 Delete ID**

**13.2 Delete All**

**13.3 Upload**

**13.4 Count**

**13.5 Get IDs**



## **Chapter 14**

### **8\_sensor**



## **Chapter 15**

# **Sensor operations**

**15.1 Wait for finger up**

**15.2 Wait for finger down**

**15.3 Reset sensor**





## **Chapter 16**

### **9\_device**



## **Chapter 17**

# **Device operations**

### **17.1 Reset device**



## Chapter 18

# Data Structure Index

### 18.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">fpc_com_chain</a>	39
<a href="#">fpc_com_chain_private</a>	44
<a href="#">fpc_com_packet_link</a>	45
<a href="#">fpc_com_packet_transport</a>	46
<a href="#">fpc_hcp_arg_data</a>	
Command Argument	47
<a href="#">fpc_hcp_packet</a>	
Application Command Packet	48



## Chapter 19

# File Index

### 19.1 File List

Here is a list of all files with brief descriptions:

<a href="#">inc/fpc_com_chain.h</a>		
	Communication chain type definitions . . . . .	51
<a href="#">inc/fpc_com_link.h</a>		
	Communication link interface . . . . .	54
<a href="#">inc/fpc_com_packets.h</a>		
	Communication packet type definitions . . . . .	56
<a href="#">inc/fpc_com_result.h</a>		
	Communication result type definitions . . . . .	59
<a href="#">inc/fpc_com_transport.h</a>		
	Communication transport interface . . . . .	60
<a href="#">inc/fpc_hcp.h</a>		
	Host Communication Protocol interface . . . . .	63
<a href="#">inc/fpc_hcp_common.h</a>		
	Host Communication Protocol common type definitions . . . . .	68
<a href="#">src/fpc_com_link.c</a>		
	Communication link layer implementation . . . . .	75
<a href="#">src/fpc_com_transport.c</a>		
	Communication transport layer implementation . . . . .	78
<a href="#">src/fpc_hcp.c</a>		
	Host Communication Protocol implementation . . . . .	80





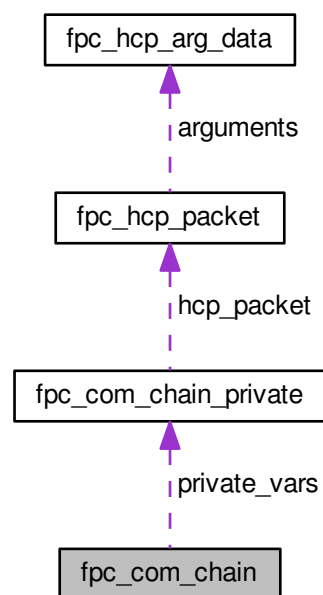
## Chapter 20

# Data Structure Documentation

### 20.1 fpc\_com\_chain Struct Reference

```
#include <fpc_com_chain.h>
```

Collaboration diagram for fpc\_com\_chain:



#### Data Fields

- bool `initialized`
- uint32\_t(\* `crc_calc`)(uint32\_t start, const void \*data, uint32\_t size)
- `fpc_com_chain_private_t` `private_vars`

- void \* [session](#)

User session pointer. User private stuff, to be able to pass necessary info from the layer that calls hcp down to the user's TX and RX functions (phy\_tx/rx), to enable multi threaded applications at the host side.

- void \* [context](#)

User context pointer. User private stuff, to be able to pass necessary context to argument\_allocator and argument\_free.

## HCP Layer

- void (\*)([argument\\_allocator](#))([fpc\\_hcp\\_cmd\\_t](#) cmd, [fpc\\_hcp\\_arg\\_t](#) arg, uint16\_t size, bool \*free\_data, void \*context)
- void([argument\\_free](#))([fpc\\_hcp\\_cmd\\_t](#) cmd, [fpc\\_hcp\\_arg\\_data\\_t](#) \*arg\_data, void \*context)

## Application Layer

- [fpc\\_com\\_result\\_t](#)(\* [app\\_tx](#))([fpc\\_com\\_chain\\_t](#) \*chain)
- [fpc\\_com\\_result\\_t](#)(\* [app\\_rx](#))([fpc\\_com\\_chain\\_t](#) \*chain)
- uint16\_t(\* [app\\_overhead\\_get](#))(uint16\_t \*offset)
- uint16\_t [app\\_packet\\_size](#) [2]
- uint16\_t [app\\_mtu\\_size](#) [2]
- uint8\_t \* [app\\_mtu\\_buffer](#) [2]

## Transport Layer

- [fpc\\_com\\_result\\_t](#)(\* [tsp\\_tx](#))([fpc\\_com\\_packet\\_tsp\\_t](#) \*packet, [fpc\\_com\\_chain\\_t](#) \*chain)
- [fpc\\_com\\_result\\_t](#)(\* [tsp\\_rx](#))([fpc\\_com\\_packet\\_tsp\\_t](#) \*packet, [fpc\\_com\\_chain\\_t](#) \*chain)
- uint16\_t(\* [tsp\\_overhead\\_get](#))(uint16\_t \*offset)

## Link Layer

- uint16\_t(\* [link\\_overhead\\_get](#))(uint16\_t \*offset)
- [fpc\\_com\\_channel\\_t](#) channel

## Physical Layer

- [fpc\\_com\\_result\\_t](#)(\* [phy\\_tx](#))(uint16\_t size, const uint8\_t \*buffer, uint32\_t timeout, void \*session)
- [fpc\\_com\\_result\\_t](#)(\* [phy\\_rx](#))(uint16\_t size, uint8\_t \*buffer, uint32\_t timeout, void \*session)
- uint16\_t [phy\\_mtu\\_size](#) [2]
- uint8\_t \* [phy\\_mtu\\_buffer](#) [2]
- uint32\_t [phy\\_timeout\\_tx](#)
- uint32\_t [phy\\_timeout\\_rx](#)

### 20.1.1 Detailed Description

Communication chain struct

Definition at line 50 of file fpc\_com\_chain.h.

### 20.1.2 Field Documentation

#### 20.1.2.1 uint8\_t\* fpc\_com\_chain::app\_mtu\_buffer[2]

Application MTU buffers

Definition at line 83 of file fpc\_com\_chain.h.

### 20.1.2.2 `uint16_t fpc_com_chain::app_mtu_size[2]`

Application MTU sizes

Definition at line 81 of file `fpc_com_chain.h`.

### 20.1.2.3 `uint16_t(* fpc_com_chain::app_overhead_get)(uint16_t *offset)`

Application layer overhead get interface function

Definition at line 77 of file `fpc_com_chain.h`.

### 20.1.2.4 `uint16_t fpc_com_chain::app_packet_size[2]`

Application packet sizes

Definition at line 79 of file `fpc_com_chain.h`.

### 20.1.2.5 `fpc_com_result_t(* fpc_com_chain::app_rx)(fpc_com_chain_t *chain)`

Application layer receive interface function

Definition at line 75 of file `fpc_com_chain.h`.

### 20.1.2.6 `fpc_com_result_t(* fpc_com_chain::app_tx)(fpc_com_chain_t *chain)`

Application layer transmit interface function

Definition at line 73 of file `fpc_com_chain.h`.

### 20.1.2.7 `void(* fpc_com_chain::argument_allocator)(fpc_hcp_cmd_t cmd, fpc_hcp_arg_t arg, uint16_t size, bool *free_data, void *context)`

Argument allocator interface function

Definition at line 59 of file `fpc_com_chain.h`.

### 20.1.2.8 `void(* fpc_com_chain::argument_free)(fpc_hcp_cmd_t cmd, fpc_hcp_arg_data_t *arg_data, void *context)`

Argument free interface function

Definition at line 62 of file `fpc_com_chain.h`.

#### 20.1.2.9 `fpc_com_channel_t fpc_com_chain::channel`

Communication channel

Definition at line 105 of file `fpc_com_chain.h`.

#### 20.1.2.10 `void* fpc_com_chain::context`

User context pointer. User private stuff, to be able to pass nessecary context to `argument_allocator` and `argument_free`.

Definition at line 143 of file `fpc_com_chain.h`.

#### 20.1.2.11 `uint32_t(* fpc_com_chain::crc_calc)(uint32_t start, const void *data, uint32_t size)`

CRC calculation interface function

Definition at line 66 of file `fpc_com_chain.h`.

#### 20.1.2.12 `bool fpc_com_chain::initialized`

Initialization status

Definition at line 52 of file `fpc_com_chain.h`.

#### 20.1.2.13 `uint16_t(* fpc_com_chain::link_overhead_get)(uint16_t *offset)`

Link layer overhead get interface function

Definition at line 103 of file `fpc_com_chain.h`.

#### 20.1.2.14 `uint8_t* fpc_com_chain::phy_mtu_buffer[2]`

Physical MTU buffers

Definition at line 121 of file `fpc_com_chain.h`.

#### 20.1.2.15 `uint16_t fpc_com_chain::phy_mtu_size[2]`

Physical MTU sizes

Definition at line 119 of file `fpc_com_chain.h`.

20.1.2.16 **fpc\_com\_result\_t**(\* fpc\_com\_chain::phy\_rx) (uint16\_t size, uint8\_t \*buffer, uint32\_t timeout, void \*session)

Physical layer receive interface function

Definition at line 116 of file fpc\_com\_chain.h.

20.1.2.17 **uint32\_t** fpc\_com\_chain::phy\_timeout\_rx

Physical receive timeout

Definition at line 125 of file fpc\_com\_chain.h.

20.1.2.18 **uint32\_t** fpc\_com\_chain::phy\_timeout\_tx

Physical transmit timeout

Definition at line 123 of file fpc\_com\_chain.h.

20.1.2.19 **fpc\_com\_result\_t**(\* fpc\_com\_chain::phy\_tx) (uint16\_t size, const uint8\_t \*buffer, uint32\_t timeout, void \*session)

Physical layer transmit interface function

Definition at line 113 of file fpc\_com\_chain.h.

20.1.2.20 **fpc\_com\_chain\_private\_t** fpc\_com\_chain::private\_vars

Communication change private variables

Definition at line 129 of file fpc\_com\_chain.h.

20.1.2.21 **void\*** fpc\_com\_chain::session

User session pointer. User private stuff, to be able to pass necessary info from the layer that calls hcp down to the user's TX and RX functions (phy\_tx/rx), to enable multi threaded applications at the host side.

Definition at line 137 of file fpc\_com\_chain.h.

20.1.2.22 **uint16\_t**(\* fpc\_com\_chain::tsp\_overhead\_get) (uint16\_t \*offset)

Transport layer overhead get interface function

Definition at line 95 of file fpc\_com\_chain.h.

20.1.2.23 `fpc_com_result_t(* fpc_com_chain::tsp_rx)(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`

Transport layer receive interface function

Definition at line 93 of file `fpc_com_chain.h`.

20.1.2.24 `fpc_com_result_t(* fpc_com_chain::tsp_tx)(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`

Transport layer transmit interface function

Definition at line 91 of file `fpc_com_chain.h`.

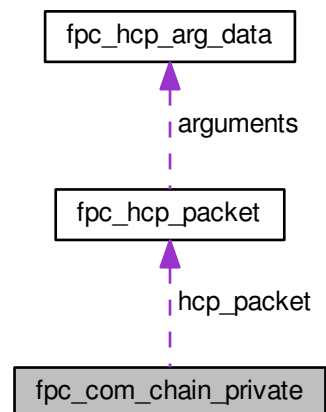
The documentation for this struct was generated from the following file:

- [inc/fpc\\_com\\_chain.h](#)

## 20.2 fpc\_com\_chain\_private Struct Reference

```
#include <fpc_com_chain.h>
```

Collaboration diagram for `fpc_com_chain_private`:



### Data Fields

- [fpc\\_hcp\\_packet\\_t \\* hcp\\_packet](#)
- `uint16_t hcp_seq_len`
- `uint16_t hcp_seq_nr`

### 20.2.1 Detailed Description

Communication chain private struct

Definition at line 36 of file fpc\_com\_chain.h.

### 20.2.2 Field Documentation

#### 20.2.2.1 fpc\_hcp\_packet\_t\* fpc\_com\_chain\_private::hcp\_packet

HCP packet

Definition at line 38 of file fpc\_com\_chain.h.

#### 20.2.2.2 uint16\_t fpc\_com\_chain\_private::hcp\_seq\_len

HCP sequence length

Definition at line 40 of file fpc\_com\_chain.h.

#### 20.2.2.3 uint16\_t fpc\_com\_chain\_private::hcp\_seq\_nr

HCP sequence number

Definition at line 42 of file fpc\_com\_chain.h.

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

- [inc/fpc\\_com\\_chain.h](#)

## 20.3 fpc\_com\_packet\_link Struct Reference

```
#include <fpc_com_packets.h>
```

### Data Fields

- [fpc\\_com\\_channel\\_t channel](#)
- [uint16\\_t size](#)
- [uint8\\_t \\* data](#)
- [uint32\\_t crc](#)

### 20.3.1 Detailed Description

Link layer packet

Definition at line 61 of file fpc\_com\_packets.h.

### 20.3.2 Field Documentation

#### 20.3.2.1 `fpc_com_channel_t fpc_com_packet_link::channel`

Communication channel

Definition at line 63 of file `fpc_com_packets.h`.

#### 20.3.2.2 `uint32_t fpc_com_packet_link::crc`

CRC of data

Definition at line 69 of file `fpc_com_packets.h`.

#### 20.3.2.3 `uint8_t* fpc_com_packet_link::data`

Packet data

Definition at line 67 of file `fpc_com_packets.h`.

#### 20.3.2.4 `uint16_t fpc_com_packet_link::size`

Size of packet

Definition at line 65 of file `fpc_com_packets.h`.

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

- [inc/fpc\\_com\\_packets.h](#)

## 20.4 `fpc_com_packet_transport` Struct Reference

```
#include <fpc_com_packets.h>
```

### Data Fields

- `uint16_t` [size](#)
- `uint16_t` [seq\\_len](#)
- `uint16_t` [seq\\_nr](#)
- `uint8_t *` [data](#)

#### 20.4.1 Detailed Description

Transport layer packet.

Definition at line 37 of file `fpc_com_packets.h`.



### 20.4.2 Field Documentation

#### 20.4.2.1 `uint8_t* fpc_com_packet_transport::data`

Packet data

Definition at line 45 of file `fpc_com_packets.h`.

#### 20.4.2.2 `uint16_t fpc_com_packet_transport::seq_len`

Sequence length

Definition at line 41 of file `fpc_com_packets.h`.

#### 20.4.2.3 `uint16_t fpc_com_packet_transport::seq_nr`

Sequence number

Definition at line 43 of file `fpc_com_packets.h`.

#### 20.4.2.4 `uint16_t fpc_com_packet_transport::size`

Size of packet

Definition at line 39 of file `fpc_com_packets.h`.

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

- [inc/fpc\\_com\\_packets.h](#)

## 20.5 fpc\_hcp\_arg\_data Struct Reference

Command Argument.

```
#include <fpc_hcp_common.h>
```

### Data Fields

- [fpc\\_hcp\\_arg\\_t arg](#)
- [uint16\\_t size](#)
- [bool free\\_data](#)
- [uint8\\_t \\* data](#)

### 20.5.1 Detailed Description

Command Argument.

Definition at line 196 of file fpc\_hcp\_common.h.

### 20.5.2 Field Documentation

#### 20.5.2.1 `fpc_hcp_arg_t fpc_hcp_arg_data::arg`

Argument

Definition at line 198 of file fpc\_hcp\_common.h.

#### 20.5.2.2 `uint8_t* fpc_hcp_arg_data::data`

Pointer to data

Definition at line 204 of file fpc\_hcp\_common.h.

#### 20.5.2.3 `bool fpc_hcp_arg_data::free_data`

Free data inside HCP

Definition at line 202 of file fpc\_hcp\_common.h.

#### 20.5.2.4 `uint16_t fpc_hcp_arg_data::size`

Size of data

Definition at line 200 of file fpc\_hcp\_common.h.

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

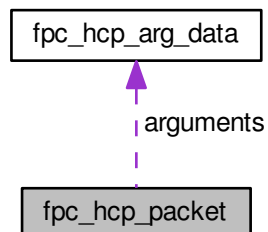
- [inc/fpc\\_hcp\\_common.h](#)

## 20.6 fpc\_hcp\_packet Struct Reference

Application Command Packet.

```
#include <fpc_hcp_common.h>
```

Collaboration diagram for fpc\_hcp\_packet:



## Data Fields

- [fpc\\_hcp\\_cmd\\_t id](#)
- [uint16\\_t num\\_args](#)
- [fpc\\_hcp\\_arg\\_data\\_t \\* arguments](#)

### 20.6.1 Detailed Description

Application Command Packet.

Definition at line 210 of file `fpc_hcp_common.h`.

### 20.6.2 Field Documentation

#### 20.6.2.1 `fpc_hcp_arg_data_t*` `fpc_hcp_packet::arguments`

Pointer to argument data

Definition at line 216 of file `fpc_hcp_common.h`.

#### 20.6.2.2 `fpc_hcp_cmd_t` `fpc_hcp_packet::id`

Command ID

Definition at line 212 of file `fpc_hcp_common.h`.

#### 20.6.2.3 `uint16_t` `fpc_hcp_packet::num_args`

Number of arguments

Definition at line 214 of file `fpc_hcp_common.h`.

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

- [inc/fpc\\_hcp\\_common.h](#)



## Chapter 21

# File Documentation

**21.1** `doc/md/1_stack.md` File Reference

**21.2** `doc/md/2_hcpframe.md` File Reference

**21.3** `doc/md/4_biometrics.md` File Reference

**21.4** `doc/md/5_image.md` File Reference

**21.5** `doc/md/6_template.md` File Reference

**21.6** `doc/md/7_storage.md` File Reference

**21.7** `doc/md/8_sensor.md` File Reference

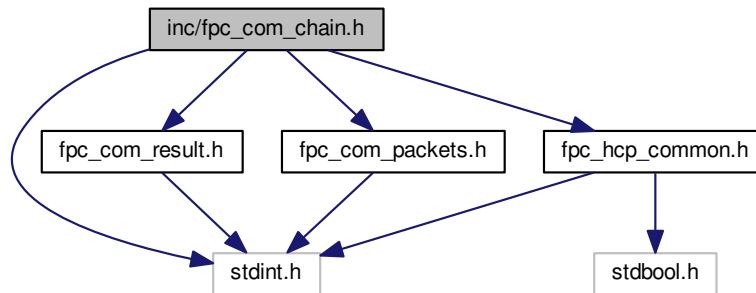
**21.8** `doc/md/9_device.md` File Reference

**21.9** `hcp.md` File Reference

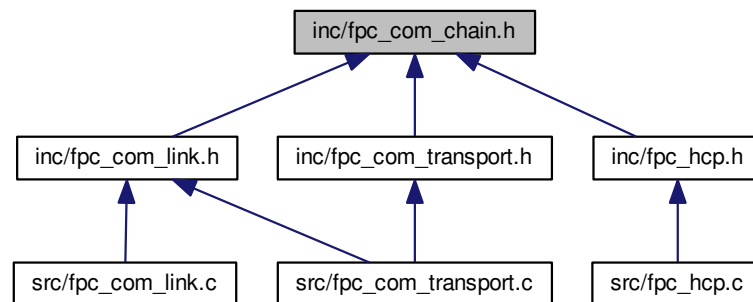
**21.10** `inc/fpc_com_chain.h` File Reference

Communication chain type definitions.

```
#include <stdint.h>
#include "fpc_com_result.h"
#include "fpc_hcp_common.h"
#include "fpc_com_packets.h"
Include dependency graph for fpc_com_chain.h:
```



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



## Data Structures

- struct `fpc_com_chain_private`
- struct `fpc_com_chain`

## Typedefs

- typedef struct `fpc_com_chain_private` `fpc_com_chain_private_t`  
*Communication chain private variables.*
- typedef struct `fpc_com_chain` `fpc_com_chain_t`  
*Communication chain.*

## Enumerations

- enum `fpc_com_chain_dir_t` {  
    `FPC_COM_CHAIN_TX` = 0,  
    `FPC_COM_CHAIN_RX` = 1 }

*Communication chain direction type.*

### 21.10.1 Detailed Description

Communication chain type definitions.

### 21.10.2 Typedef Documentation

#### 21.10.2.1 typedef struct `fpc_com_chain_private` `fpc_com_chain_private_t`

Communication chain private variables.

Definition at line 34 of file `fpc_com_chain.h`.

#### 21.10.2.2 typedef struct `fpc_com_chain` `fpc_com_chain_t`

Communication chain.

Definition at line 48 of file `fpc_com_chain.h`.

### 21.10.3 Enumeration Type Documentation

#### 21.10.3.1 enum `fpc_com_chain_dir_t`

Communication chain direction type.

#### Enumerator

**`FPC_COM_CHAIN_TX`**

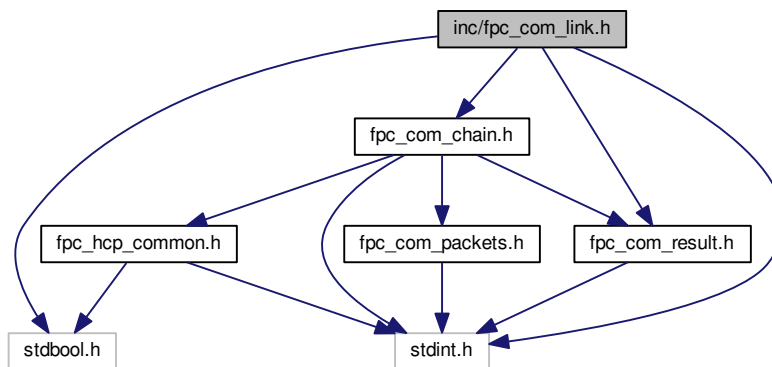
**`FPC_COM_CHAIN_RX`**

Definition at line 149 of file `fpc_com_chain.h`.

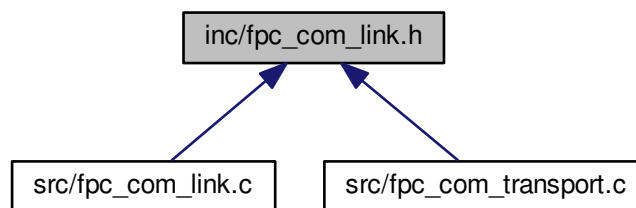
## 21.11 inc/fpc\_com\_link.h File Reference

Communication link interface.

```
#include <stdbool.h>
#include <stdint.h>
#include "fpc_com_result.h"
#include "fpc_com_chain.h"
Include dependency graph for fpc_com_link.h:
```



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



### Functions

- `fpc_com_result_t fpc_com_link_transmit(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain)`  
*Sends a packet over the physical link in blocking mode.*
- `fpc_com_result_t fpc_com_link_receive(fpc_com_packet_link_t *packet, fpc_com_chain_t *chain)`  
*Receives a packet from the physical link.*
- `uint16_t fpc_com_link_get_overhead(uint16_t *offset)`  
*Returns the overhead of the layer.*



### 21.11.1 Detailed Description

Communication link interface.

### 21.11.2 Function Documentation

#### 21.11.2.1 `uint16_t fpc_com_link_get_overhead ( uint16_t * offset )`

Returns the overhead of the layer.

##### Parameters

<code>out</code>	<code>offset</code>	The offset to the packet data.
------------------	---------------------	--------------------------------

##### Returns

Overhead size in bytes.

Definition at line 126 of file `fpc_com_link.c`.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, and `fpc_com_packet_link::size`.

#### 21.11.2.2 `fpc_com_result_t fpc_com_link_receive ( fpc_com_packet_link_t * packet, fpc_com_chain_t * chain )`

Receives a packet from the physical link.

##### Parameters

<code>in, out</code>	<code>packet</code>	Packet to populate.
<code>in</code>	<code>chain</code>	The communication chain to use.

##### Returns

[`fpc\_com\_result\_t`](#)

Definition at line 73 of file `fpc_com_link.c`.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, `fpc_com_chain::crc_calc`, `fpc_com_packet_link::data`, `FPC_COM_ACK`, `FPC_COM_CHAIN_RX`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_IO_ERROR`, `FPC_COM_RESULT_OK`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_mtu_size`, `fpc_com_chain::phy_rx`, `fpc_com_chain::phy_timeout_rx`, `fpc_com_chain::phy_timeout_tx`, `fpc_com_chain::phy_tx`, `fpc_com_chain::session`, and `fpc_com_packet_link::size`.

#### 21.11.2.3 `fpc_com_result_t fpc_com_link_transmit ( fpc_com_packet_link_t * packet, fpc_com_chain_t * chain )`

Sends a packet over the physical link in blocking mode.

**Parameters**

in	<i>packet</i>	Packet to transmit.
in	<i>chain</i>	The communication chain to use.

**Returns**

[fpc\\_com\\_result\\_t](#)

Definition at line 27 of file `fpc_com_link.c`.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, `fpc_com_chain::crc_calc`, `fpc_com_packet_link::data`, `FPC_COM_ACK`, `FPC_COM_CHAIN_TX`, `fpc_com_link_get_overhead()`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_IO_ERROR`, `FPC_COM_RESULT_OK`, `FPC_COM_RESULT_TIMEOUT`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_rx`, `fpc_com_chain::phy_timeout_rx`, `fpc_com_chain::phy_timeout_tx`, `fpc_com_chain::phy_tx`, `fpc_com_chain::session`, and `fpc_com_packet_link::size`.

Here is the call graph for this function:

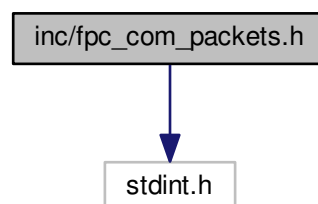


## 21.12 `inc/fpc_com_packets.h` File Reference

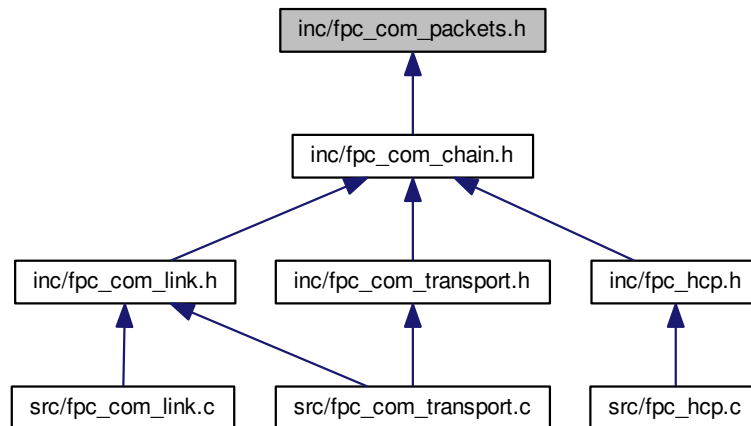
Communication packet type definitions.

```
#include <stdint.h>
```

Include dependency graph for `fpc_com_packets.h`:



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



## Data Structures

- struct [fpc\\_com\\_packet\\_transport](#)
- struct [fpc\\_com\\_packet\\_link](#)

## Macros

- `#define FPC\_COM\_ACK 0x7f01ff7f`

## Typedefs

- typedef struct [fpc\\_com\\_packet\\_transport](#) [fpc\\_com\\_packet\\_tsp\\_t](#)
- typedef uint16\_t [fpc\\_com\\_channel\\_t](#)
- typedef struct [fpc\\_com\\_packet\\_link](#) [fpc\\_com\\_packet\\_link\\_t](#)

## Enumerations

- enum [fpc\\_com\\_channel](#) {  
[FPC\\_COM\\_CHANNEL\\_NONE](#) = 0x00,  
[FPC\\_COM\\_CHANNEL\\_CLEAR](#) = 0x01,  
[FPC\\_COM\\_CHANNEL\\_TLS](#) = 0x02,  
[FPC\\_COM\\_CHANNEL\\_END](#) = 0xFF }

### 21.12.1 Detailed Description

Communication packet type definitions.

## 21.12.2 Macro Definition Documentation

### 21.12.2.1 `#define FPC_COM_ACK 0x7f01ff7f`

Communication acknowledge definition

Definition at line 32 of file `fpc_com_packets.h`.

## 21.12.3 Typedef Documentation

### 21.12.3.1 `typedef uint16_t fpc_com_channel_t`

Communication channel type

Definition at line 58 of file `fpc_com_packets.h`.

### 21.12.3.2 `typedef struct fpc_com_packet_link fpc_com_packet_link_t`

Link layer packet

### 21.12.3.3 `typedef struct fpc_com_packet_transport fpc_com_packet_tsp_t`

Transport layer packet.

## 21.12.4 Enumeration Type Documentation

### 21.12.4.1 `enum fpc_com_channel`

Transport packet channels.

Enumerator

***FPC\_COM\_CHANNEL\_NONE***  
***FPC\_COM\_CHANNEL\_CLEAR***  
***FPC\_COM\_CHANNEL\_TLS***  
***FPC\_COM\_CHANNEL\_END***

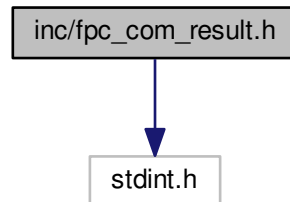
Definition at line 51 of file `fpc_com_packets.h`.

## 21.13 inc/fpc\_com\_result.h File Reference

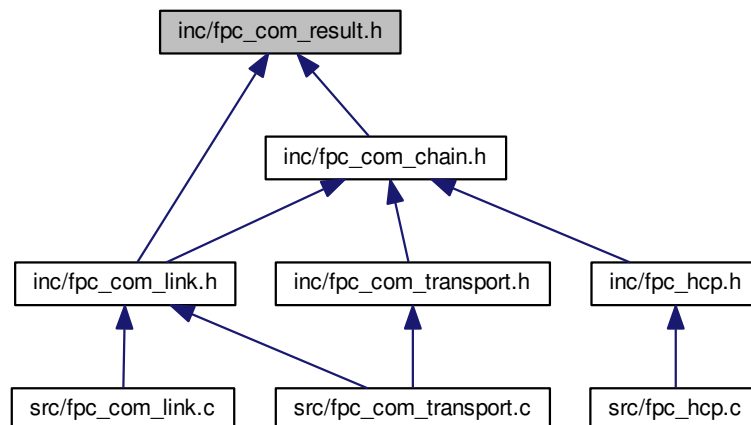
Communication result type definitions.

```
#include <stdint.h>
```

Include dependency graph for fpc\_com\_result.h:



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



### Typedefs

- typedef uint8\_t [fpc\\_com\\_result\\_t](#)

### Enumerations

- enum [fpc\\_com\\_result](#) {  
[FPC\\_COM\\_RESULT\\_OK](#),  
[FPC\\_COM\\_RESULT\\_NO\\_MEMORY](#),  
[FPC\\_COM\\_RESULT\\_INVALID\\_ARGUMENT](#),  
[FPC\\_COM\\_RESULT\\_NOT\\_IMPLEMENTED](#),  
[FPC\\_COM\\_RESULT\\_IO\\_ERROR](#),  
[FPC\\_COM\\_RESULT\\_TIMEOUT](#) }

### 21.13.1 Detailed Description

Communication result type definitions.

### 21.13.2 Typedef Documentation

#### 21.13.2.1 typedef uint8\_t fpc\_com\_result\_t

Communication result type

Definition at line 41 of file fpc\_com\_result.h.

### 21.13.3 Enumeration Type Documentation

#### 21.13.3.1 enum fpc\_com\_result

Communication result codes

Enumerator

```
FPC_COM_RESULT_OK
FPC_COM_RESULT_NO_MEMORY
FPC_COM_RESULT_INVALID_ARGUMENT
FPC_COM_RESULT_NOT_IMPLEMENTED
FPC_COM_RESULT_IO_ERROR
FPC_COM_RESULT_TIMEOUT
```

Definition at line 32 of file fpc\_com\_result.h.

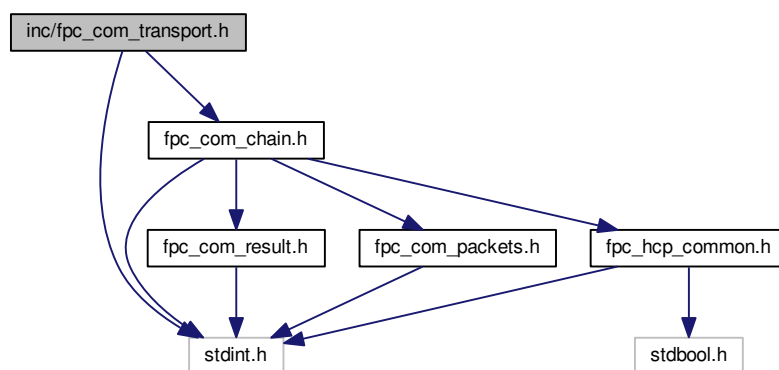
## 21.14 inc/fpc\_com\_transport.h File Reference

Communication transport interface.

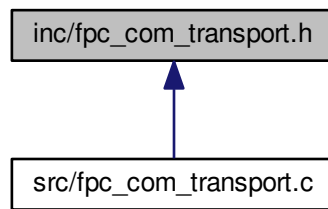
```
#include <stdint.h>
```

```
#include "fpc_com_chain.h"
```

Include dependency graph for fpc\_com\_transport.h:



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



## Functions

- `fpc_com_result_t fpc_com_transport_transmit (fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`  
*Transmit a transport layer packet.*
- `fpc_com_result_t fpc_com_transport_receive (fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`  
*Receive a transport layer packet.*
- `uint16_t fpc_com_transport_get_overhead (uint16_t *offset)`  
*Returns the overhead of the layer.*

### 21.14.1 Detailed Description

Communication transport interface.

### 21.14.2 Function Documentation

#### 21.14.2.1 `uint16_t fpc_com_transport_get_overhead ( uint16_t * offset )`

Returns the overhead of the layer.

#### Parameters

out	offset	The offset to the packet data.
-----	--------	--------------------------------

#### Returns

Overhead size in bytes.

Definition at line 88 of file `fpc_com_transport.c`.

References `fpc_com_packet_transport::seq_len`, `fpc_com_packet_transport::seq_nr`, and `fpc_com_packet_↔transport::size`.

21.14.2.2 **fpc\_com\_result\_t** fpc\_com\_transport\_receive ( **fpc\_com\_packet\_tsp\_t** \* *packet*, **fpc\_com\_chain\_t** \* *chain* )

Receive a transport layer packet.

#### Parameters

in, out	<i>packet</i>	The packet to populate.
in	<i>chain</i>	The chain to use.

#### Returns

[fpc\\_com\\_result\\_t](#)

Definition at line 60 of file fpc\_com\_transport.c.

References [fpc\\_com\\_packet\\_transport::data](#), [fpc\\_com\\_packet\\_link::data](#), [fpc\\_com\\_link\\_receive\(\)](#), [FPC\\_COM\\_RESULT\\_INVALID\\_ARGUMENT](#), [FPC\\_COM\\_RESULT\\_OK](#), [fpc\\_com\\_packet\\_transport::seq\\_len](#), [fpc\\_com\\_packet\\_transport::seq\\_nr](#), and [fpc\\_com\\_packet\\_transport::size](#).

Here is the call graph for this function:



21.14.2.3 **fpc\_com\_result\_t** fpc\_com\_transport\_transmit ( **fpc\_com\_packet\_tsp\_t** \* *packet*, **fpc\_com\_chain\_t** \* *chain* )

Transmit a transport layer packet.

#### Parameters

in	<i>packet</i>	The packet to transmit.
in	<i>chain</i>	The chain to use.

#### Returns

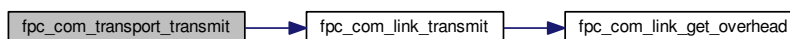
[fpc\\_com\\_result\\_t](#)

Definition at line 28 of file fpc\_com\_transport.c.

References [fpc\\_com\\_packet\\_link::channel](#), [fpc\\_com\\_chain::channel](#), [fpc\\_com\\_packet\\_link::data](#), [FPC\\_COM\\_CHAIN\\_TX](#), [fpc\\_com\\_link\\_transmit\(\)](#), [FPC\\_COM\\_RESULT\\_INVALID\\_ARGUMENT](#), [fpc\\_com\\_chain::link\\_overhead\\_get](#), [fpc\\_com\\_chain::phy\\_mtu\\_buffer](#), [fpc\\_com\\_packet\\_transport::seq\\_len](#), [fpc\\_com\\_packet\\_transport::seq\\_nr](#), [fpc\\_com\\_packet\\_transport::size](#), [fpc\\_com\\_packet\\_link::size](#), and [fpc\\_com\\_chain::tsp\\_overhead\\_get](#).



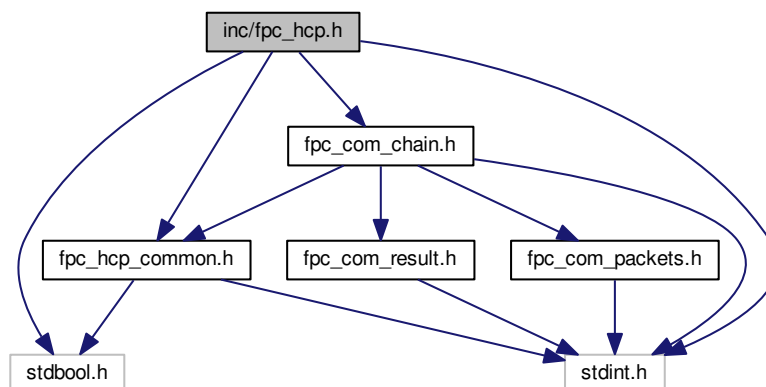
Here is the call graph for this function:



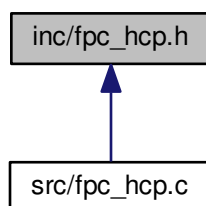
## 21.15 inc/fpc\_hcp.h File Reference

Host Communication Protocol interface.

```
#include <stdbool.h>
#include <stdint.h>
#include "fpc_hcp_common.h"
#include "fpc_com_chain.h"
Include dependency graph for fpc_hcp.h:
```



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



## Functions

- `fpc_com_result_t fpc_hcp_transmit (fpc_hcp_packet_t *packet, fpc_com_chain_t *chain)`  
*Transmits an application packet through the supplied transmit chain.*
- `fpc_com_result_t fpc_hcp_receive (fpc_hcp_packet_t *packet, fpc_com_chain_t *chain)`  
*Receives an application packet through the supplied transmit chain.*
- `bool fpc_hcp_arg_add (fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t size, bool free_data, void *data)`  
*Add argument to packet.*
- `bool fpc_hcp_arg_check (fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg)`  
*Check if command contains selected argument key.*
- `fpc_hcp_arg_data_t * fpc_hcp_arg_get (fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg)`  
*Get Argument with specified key.*
- `bool fpc_hcp_arg_copy_data (fpc_hcp_packet_t *packet, fpc_hcp_arg_t arg, uint16_t data_size, uint8_t *data)`  
*Copy data from an argument with specified key.*
- `void fpc_hcp_free (fpc_com_chain_t *chain, fpc_hcp_packet_t *packet)`  
*Frees the resources held by the packet i.e. the dynamic data held in the arguments.*
- `uint16_t fpc_hcp_get_size (fpc_hcp_packet_t *packet, uint16_t *num_args)`  
*Calculate serialized packet size.*

### 21.15.1 Detailed Description

Host Communication Protocol interface.

### 21.15.2 Function Documentation

21.15.2.1 `bool fpc_hcp_arg_add ( fpc_hcp_packet_t * packet, fpc_hcp_arg_t arg, uint16_t size, bool free_data, void * data )`

Add argument to packet.

#### Note

This function does not allocate any memory, it will only set the argument variables.

#### Parameters

in	<i>packet</i>	Packet to add to.
in	<i>arg</i>	Argument id.
in	<i>size</i>	Size of argument data.
in	<i>free_data</i>	Set to true if data should be owned by the argument, false if user still owns data.
in	<i>data</i>	Pointer to argument data.

#### Returns

true = success, false = failure.

Definition at line 145 of file fpc\_hcp.c.

References `fpc_hcp_arg_data::arg`, `ARG_NONE`, `fpc_hcp_packet::arguments`, `fpc_hcp_arg_data::data`, `fpc_hcp_arg_data::free_data`, `fpc_hcp_packet::num_args`, and `fpc_hcp_arg_data::size`.

### 21.15.2.2 `bool fpc_hcp_arg_check ( fpc_hcp_packet_t * packet, fpc_hcp_arg_t arg )`

Check if command contains selected argument key.

#### Parameters

in	<i>packet</i>	The packet to scan.
in	<i>arg</i>	Argument to look for.

#### Returns

true if found, false if not found.

Definition at line 169 of file fpc\_hcp.c.

References `fpc_hcp_arg_get()`.

Here is the call graph for this function:



### 21.15.2.3 `bool fpc_hcp_arg_copy_data ( fpc_hcp_packet_t * packet, fpc_hcp_arg_t arg, uint16_t data_size, uint8_t * data )`

Copy data from an argument with specified key.

Argument data will be copied to specified data buffer. Remaining bytes in data will be cleared if the argument data size is less than data size when the argument contains data.

#### Parameters

in	<i>packet</i>	The packet to operate on.
in	<i>arg</i>	The arg to retrieve data from.
in	<i>data_size</i>	Number of bytes to copy.
in, out	<i>data</i>	Pointer to data buffer.

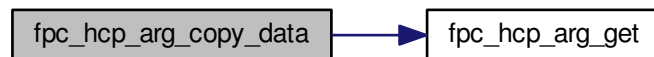
**Returns**

True if argument found, false if not found.

Definition at line 183 of file fpc\_hcp.c.

References fpc\_hcp\_arg\_data::data, fpc\_hcp\_arg\_get(), and fpc\_hcp\_arg\_data::size.

Here is the call graph for this function:



#### 21.15.2.4 fpc\_hcp\_arg\_data\_t\* fpc\_hcp\_arg\_get ( fpc\_hcp\_packet\_t \* *packet*, fpc\_hcp\_arg\_t *arg* )

Get Argument with specified key.

**Parameters**

in	<i>packet</i>	The packet to operate on.
in	<i>arg</i>	The arg to retrieve.

**Returns**

Pointer to [fpc\\_hcp\\_arg\\_data\\_t](#) is successful, otherwise NULL.

Definition at line 173 of file fpc\_hcp.c.

References fpc\_hcp\_arg\_data::arg, fpc\_hcp\_packet::arguments, and fpc\_hcp\_packet::num\_args.

#### 21.15.2.5 void fpc\_hcp\_free ( fpc\_com\_chain\_t \* *chain*, fpc\_hcp\_packet\_t \* *packet* )

Frees the resources held by the packet i.e. the dynamic data held in the arguments.

**Parameters**

in	<i>chain</i>	Pointer to the communication chain used to retrieve the packet.
in	<i>packet</i>	Pointer to packet.

Definition at line 198 of file fpc\_hcp.c.

References fpc\_hcp\_arg\_data::arg, ARG\_NONE, fpc\_com\_chain::argument\_free, fpc\_hcp\_packet::arguments, CMD\_NONE, fpc\_com\_chain::context, fpc\_hcp\_packet::id, and fpc\_hcp\_packet::num\_args.

### 21.15.2.6 `uint16_t fpc_hcp_get_size ( fpc_hcp_packet_t * packet, uint16_t * num_args )`

Calculate serialized packet size.

#### Parameters

in	<i>packet</i>	Packet to calculate.
in, out	<i>num_args</i>	Will return number of arguments held by the command can be set to NULL.

#### Returns

Serialized size.

Definition at line 64 of file `fpc_hcp.c`.

References `fpc_hcp_arg_data::arg`, `ARG_NONE`, `ARGUMENT_HEADER_SIZE`, `fpc_hcp_packet::arguments`, `fpc_hcp_packet::num_args`, `PACKET_HEADER_SIZE`, and `fpc_hcp_arg_data::size`.

### 21.15.2.7 `fpc_com_result_t fpc_hcp_receive ( fpc_hcp_packet_t * packet, fpc_com_chain_t * chain )`

Receives an application packet through the supplied transmit chain.

#### Parameters

in, out	<i>packet</i>	Pointer to pre-allocated packet struct.
in	<i>chain</i>	The chain to use.

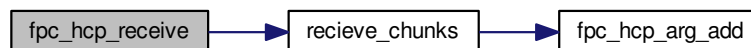
#### Returns

`fpc_com_result_t`

Definition at line 117 of file `fpc_hcp.c`.

References `fpc_com_chain::app_mtu_buffer`, `fpc_com_chain::app_mtu_size`, `FPC_COM_CHAIN_RX`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `fpc_com_chain_private::hcp_packet`, `fpc_com_chain::initialized`, `fpc_com_chain::link_overhead_get`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_mtu_size`, `fpc_com_chain::private_vars`, `recieve_chunks()`, and `fpc_com_chain::tsp_overhead_get`.

Here is the call graph for this function:



### 21.15.2.8 `fpc_com_result_t fpc_hcp_transmit ( fpc_hcp_packet_t * packet, fpc_com_chain_t * chain )`

Transmits an application packet through the supplied transmit chain.

**Parameters**

in	<i>packet</i>	Application packet to send.
in	<i>chain</i>	The chain to use.

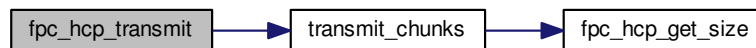
**Returns**

fpc\_com\_result\_t

Definition at line 89 of file fpc\_hcp.c.

References fpc\_com\_chain::app\_mtu\_buffer, fpc\_com\_chain::app\_mtu\_size, FPC\_COM\_CHAIN\_TX, FPC\_COM\_RESULT\_INVALID\_ARGUMENT, fpc\_com\_chain\_private::hcp\_packet, fpc\_com\_chain::initialized, fpc\_com\_chain::link\_overhead\_get, fpc\_com\_chain::phy\_mtu\_buffer, fpc\_com\_chain::phy\_mtu\_size, fpc\_com\_chain::private\_vars, transmit\_chunks(), and fpc\_com\_chain::tsp\_overhead\_get.

Here is the call graph for this function:



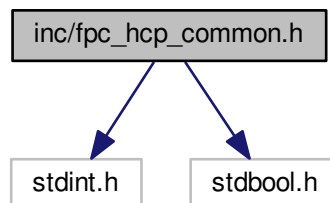
## 21.16 inc/fpc\_hcp\_common.h File Reference

Host Communication Protocol common type definitions.

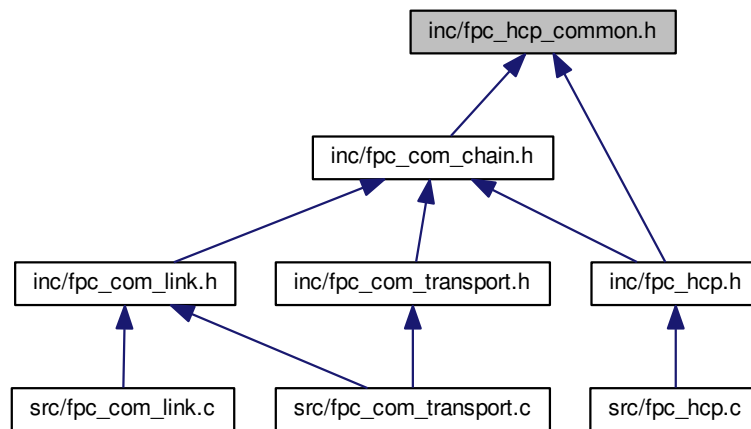
```
#include <stdint.h>
```

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

Include dependency graph for fpc\_hcp\_common.h:



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



## Data Structures

- struct [fpc\\_hcp\\_arg\\_data](#)  
*Command Argument.*
- struct [fpc\\_hcp\\_packet](#)  
*Application Command Packet.*

## Macros

- #define [HCP\\_MIN](#)(x, y) (((x) < (y)) ? (x) : (y))
- #define [CMD\\_APP\\_BASE\\_VAL](#) 0xE000
- #define [ARG\\_APP\\_BASE\\_VAL](#) 0x7000

## Typedefs

- typedef uint16\_t [fpc\\_hcp\\_cmd\\_t](#)
- typedef uint16\_t [fpc\\_hcp\\_arg\\_t](#)
- typedef struct [fpc\\_hcp\\_arg\\_data](#) [fpc\\_hcp\\_arg\\_data\\_t](#)  
*Command Argument.*
- typedef struct [fpc\\_hcp\\_packet](#) [fpc\\_hcp\\_packet\\_t](#)  
*Application Command Packet.*

## Enumerations

- enum `fpc_hcp_cmd` {  
    `CMD_NONE` = 0x0000,  
    `CMD_CAPTURE` = 0x0001,  
    `CMD_ENROLL` = 0x0002,  
    `CMD_IDENTIFY` = 0x0003,  
    `CMD_MATCH` = 0x0004,  
    `CMD_IMAGE` = 0x0005,  
    `CMD_TEMPLATE` = 0x0006,  
    `CMD_WAIT` = 0x0007,  
    `CMD_SETTINGS` = 0x0008,  
    `CMD_NAVIGATE` = 0x1001,  
    `CMD_SENSOR` = 0x1002,  
    `CMD_DEADPIXELS` = 0x1003,  
    `CMD_CONNECT` = 0x2001,  
    `CMD_RECONNECT` = 0x2002,  
    `CMD_RESET` = 0x3002,  
    `CMD_CANCEL` = 0x3003,  
    `CMD_INFO` = 0x3004,  
    `CMD_STORAGE_TEMPLATE` = 0x4002,  
    `CMD_STORAGE_CALIBRATION` = 0x4003,  
    `CMD_STORAGE_LOG` = 0x4004,  
    `CMD_STORAGE_SETTINGS` = 0x4005,  
    `CMD_TEST` = 0x5001,  
    `CMD_MCU` = 0x5002,  
    `CMD_GPIO` = 0x5003,  
    `CMD_COMMUNICATION` = 0x6001,  
    `CMD_APP_BASE` = `CMD_APP_BASE_VAL`,  
    `CMD_DIAG` = 0xF003,  
    `CMD_FFFF` = 0xFFFF }

- enum `fpc_hcp_arg` {



```

ARG_NONE = 0x0000,
ARG_FINGER_DOWN = 0x0001,
ARG_FINGER_UP = 0x0002,
ARG_START = 0x0003,
ARG_ADD = 0x0004,
ARG_FINISH = 0x0005,
ARG_ID = 0x0006,
ARG_ALL = 0x0007,
ARG_EXTRACT = 0x0008,
ARG_MATCH_IMAGE = 0x0009,
ARG_MATCH = 0x000A,
ARG_ACQUIRE = 0x1001,
ARG_RELEASE = 0x1002,
ARG_SET = 0x1003,
ARG_GET = 0x1004,
ARG_UPLOAD = 0x1005,
ARG_DOWNLOAD = 0x1006,
ARG_CREATE = 0x1007,
ARG_SAVE = 0x1008,
ARG_DELETE = 0x1009,
ARG_DATA = 0x100A,
ARG_UPDATE = 0x100B,
ARG_SEQ_NR = 0x100C,
ARG_SEQ_LEN = 0x100D,
ARG_RESULT = 0x2001,
ARG_COUNT = 0x2002,
ARG_SIZE = 0x2003,
ARG_LEVEL = 0x2004,
ARG_FORMAT = 0x2005,
ARG_FLAG = 0x2006,
ARG_PROPERTIES = 0x2007,
ARG_SPEED = 0x2008,
ARG_PROD_TEST = 0x2009,
ARG_SENSOR_TYPE = 0x3001,
ARG_WIDTH = 0x3002,
ARG_HEIGHT = 0x3003,
ARG_RESET = 0x3004,
ARG_DPI = 0x3005,
ARG_MAX_SPI_CLOCK = 0x3006,
ARG_NUM_SUB_AREAS_WIDTH = 0x3007,
ARG_NUM_SUB_AREAS_HEIGHT = 0x3008,
ARG_IRQ_STATUS = 0x3009,
ARG_RESET_HARD = 0x300A,
ARG_IDLE = 0x4001,
ARG_SLEEP = 0x4002,
ARG_DEEP_SLEEP = 0x4003,
ARG_POWER_MODE = 0x4004,
ARG_BUSY_WAIT = 0x4005,
ARG_TIMEOUT = 0x5001,
ARG_DONE = 0x5002,
ARG_BOOT = 0x6001,
ARG_STATUS = 0x6002,
ARG_VERSION = 0x6003,
ARG_UNIQUE_ID = 0x6004,
ARG_APP_BASE = ARG_APP_BASE_VAL,
ARG_NONCE = 0x8001,
ARG_MAC = 0x8002,
ARG_RANDOM = 0x8003,
ARG_CLAIM = 0x8004,
ARG_PUBLIC_KEY = 0x8005,
ARG_CRYPTHERTEXT = 0x8006,
ARG_MTU = 0x9001,
ARG_STACK = 0xE001,
ARG_FILL = 0xE002,
ARG_HEAR = 0xF002

```

```
ARG_FFFF = 0xFFFF }
```

### 21.16.1 Detailed Description

Host Communication Protocol common type definitions.

### 21.16.2 Macro Definition Documentation

#### 21.16.2.1 #define ARG\_APP\_BASE\_VAL 0x7000

Program specific arguments base number

Definition at line 39 of file fpc\_hcp\_common.h.

#### 21.16.2.2 #define CMD\_APP\_BASE\_VAL 0xE000

Program specific commands base number

Definition at line 36 of file fpc\_hcp\_common.h.

#### 21.16.2.3 #define HCP\_MIN( x, y ) (((x) < (y)) ? (x) : (y))

Returns the smallest of two values.

Definition at line 33 of file fpc\_hcp\_common.h.

### 21.16.3 Typedef Documentation

#### 21.16.3.1 typedef struct fpc\_hcp\_arg\_data fpc\_hcp\_arg\_data\_t

Command Argument.

#### 21.16.3.2 typedef uint16\_t fpc\_hcp\_arg\_t

HCP Argument type

Definition at line 191 of file fpc\_hcp\_common.h.

#### 21.16.3.3 typedef uint16\_t fpc\_hcp\_cmd\_t

HCP Command type

Definition at line 92 of file fpc\_hcp\_common.h.

21.16.3.4 typedef struct fpc\_hcp\_packet fpc\_hcp\_packet\_t

Application Command Packet.

## 21.16.4 Enumeration Type Documentation

21.16.4.1 enum fpc\_hcp\_arg

HCP Argument definitions

Enumerator

***ARG\_NONE***  
***ARG\_FINGER\_DOWN***  
***ARG\_FINGER\_UP***  
***ARG\_START***  
***ARG\_ADD***  
***ARG\_FINISH***  
***ARG\_ID***  
***ARG\_ALL***  
***ARG\_EXTRACT***  
***ARG\_MATCH\_IMAGE***  
***ARG\_MATCH***  
***ARG\_ACQUIRE***  
***ARG\_RELEASE***  
***ARG\_SET***  
***ARG\_GET***  
***ARG\_UPLOAD***  
***ARG\_DOWNLOAD***  
***ARG\_CREATE***  
***ARG\_SAVE***  
***ARG\_DELETE***  
***ARG\_DATA***  
***ARG\_UPDATE***  
***ARG\_SEQ\_NR***  
***ARG\_SEQ\_LEN***  
***ARG\_RESULT***  
***ARG\_COUNT***  
***ARG\_SIZE***  
***ARG\_LEVEL***  
***ARG\_FORMAT***  
***ARG\_FLAG***  
***ARG\_PROPERTIES***  
***ARG\_SPEED***  
***ARG\_PROD\_TEST***

***ARG\_SENSOR\_TYPE***  
***ARG\_WIDTH***  
***ARG\_HEIGHT***  
***ARG\_RESET***  
***ARG\_DPI***  
***ARG\_MAX\_SPI\_CLOCK***  
***ARG\_NUM\_SUB\_AREAS\_WIDTH***  
***ARG\_NUM\_SUB\_AREAS\_HEIGHT***  
***ARG\_IRQ\_STATUS***  
***ARG\_RESET\_HARD***  
***ARG\_IDLE***  
***ARG\_SLEEP***  
***ARG\_DEEP\_SLEEP***  
***ARG\_POWER\_MODE***  
***ARG\_BUSY\_WAIT***  
***ARG\_TIMEOUT***  
***ARG\_DONE***  
***ARG\_BOOT***  
***ARG\_STATUS***  
***ARG\_VERSION***  
***ARG\_UNIQUE\_ID***  
***ARG\_APP\_BASE***  
***ARG\_NONCE***  
***ARG\_MAC***  
***ARG\_RANDOM***  
***ARG\_CLAIM***  
***ARG\_PUBLIC\_KEY***  
***ARG\_CIPHERTEXT***  
***ARG\_MTU***  
***ARG\_STACK***  
***ARG\_FILL***  
***ARG\_HEAP***  
***ARG\_MODE***  
***ARG\_DEBUG***  
***ARG\_FFFF***

Definition at line 95 of file fpc\_hcp\_common.h.

#### 21.16.4.2 enum fpc\_hcp\_cmd

HCP Command definitions

Enumerator

***CMD\_NONE***  
***CMD\_CAPTURE***

***CMD\_ENROLL***  
***CMD\_IDENTIFY***  
***CMD\_MATCH***  
***CMD\_IMAGE***  
***CMD\_TEMPLATE***  
***CMD\_WAIT***  
***CMD\_SETTINGS***  
***CMD\_NAVIGATE***  
***CMD\_SENSOR***  
***CMD\_DEADPIXELS***  
***CMD\_CONNECT***  
***CMD\_RECONNECT***  
***CMD\_RESET***  
***CMD\_CANCEL***  
***CMD\_INFO***  
***CMD\_STORAGE\_TEMPLATE***  
***CMD\_STORAGE\_CALIBRATION***  
***CMD\_STORAGE\_LOG***  
***CMD\_STORAGE\_SETTINGS***  
***CMD\_TEST***  
***CMD\_MCU***  
***CMD\_GPIO***  
***CMD\_COMMUNICATION***  
***CMD\_APP\_BASE***  
***CMD\_DIAG***  
***CMD\_FFFF***

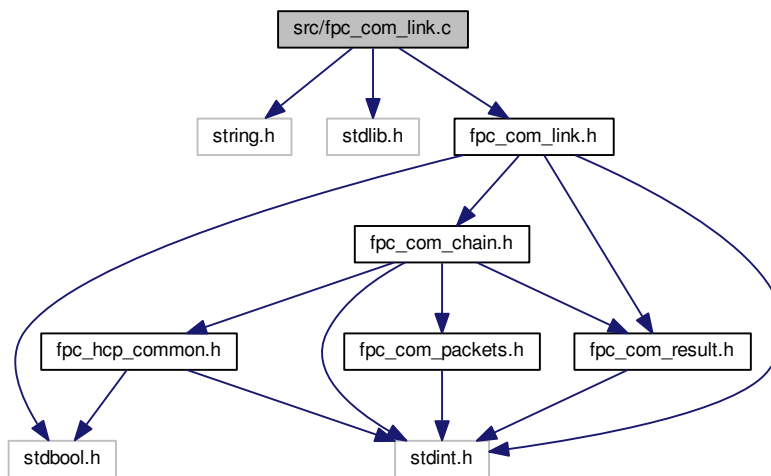
Definition at line 42 of file fpc\_hcp\_common.h.

## 21.17 src/fpc\_com\_link.c File Reference

Communication link layer implementation.

```
#include <string.h>
#include <stdlib.h>
#include "fpc_com_link.h"
```

Include dependency graph for `fpc_com_link.c`:



## Functions

- `fpc_com_result_t fpc_com_link_transmit (fpc_com_packet_link_t *packet, fpc_com_chain_t *chain)`  
*Sends a packet over the physical link in blocking mode.*
- `fpc_com_result_t fpc_com_link_receive (fpc_com_packet_link_t *packet, fpc_com_chain_t *chain)`  
*Receives a packet from the physical link.*
- `uint16_t fpc_com_link_get_overhead (uint16_t *offset)`  
*Returns the overhead of the layer.*

### 21.17.1 Detailed Description

Communication link layer implementation.

### 21.17.2 Function Documentation

#### 21.17.2.1 `uint16_t fpc_com_link_get_overhead ( uint16_t * offset )`

Returns the overhead of the layer.

#### Parameters

out	<i>offset</i>	The offset to the packet data.
-----	---------------	--------------------------------

#### Returns

Overhead size in bytes.

Definition at line 126 of file fpc\_com\_link.c.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, and `fpc_com_packet_link::size`.

### 21.17.2.2 `fpc_com_result_t fpc_com_link_receive ( fpc_com_packet_link_t * packet, fpc_com_chain_t * chain )`

Receives a packet from the physical link.

#### Parameters

in, out	<i>packet</i>	Packet to populate.
in	<i>chain</i>	The communication chain to use.

#### Returns

`fpc_com_result_t`

Definition at line 73 of file fpc\_com\_link.c.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, `fpc_com_chain::crc_calc`, `fpc_com_packet_link::data`, `FPC_COM_ACK`, `FPC_COM_CHAIN_RX`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_IO_ERROR`, `FPC_COM_RESULT_OK`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_mtu_size`, `fpc_com_chain::phy_rx`, `fpc_com_chain::phy_timeout_rx`, `fpc_com_chain::phy_timeout_tx`, `fpc_com_chain::phy_tx`, `fpc_com_chain::session`, and `fpc_com_packet_link::size`.

### 21.17.2.3 `fpc_com_result_t fpc_com_link_transmit ( fpc_com_packet_link_t * packet, fpc_com_chain_t * chain )`

Sends a packet over the physical link in blocking mode.

#### Parameters

in	<i>packet</i>	Packet to transmit.
in	<i>chain</i>	The communication chain to use.

#### Returns

`fpc_com_result_t`

Definition at line 27 of file fpc\_com\_link.c.

References `fpc_com_packet_link::channel`, `fpc_com_packet_link::crc`, `fpc_com_chain::crc_calc`, `fpc_com_packet_link::data`, `FPC_COM_ACK`, `FPC_COM_CHAIN_TX`, `fpc_com_link_get_overhead()`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_IO_ERROR`, `FPC_COM_RESULT_OK`, `FPC_COM_RESULT_TIMEOUT`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_rx`, `fpc_com_chain::phy_timeout_rx`, `fpc_com_chain::phy_timeout_tx`, `fpc_com_chain::phy_tx`, `fpc_com_chain::session`, and `fpc_com_packet_link::size`.

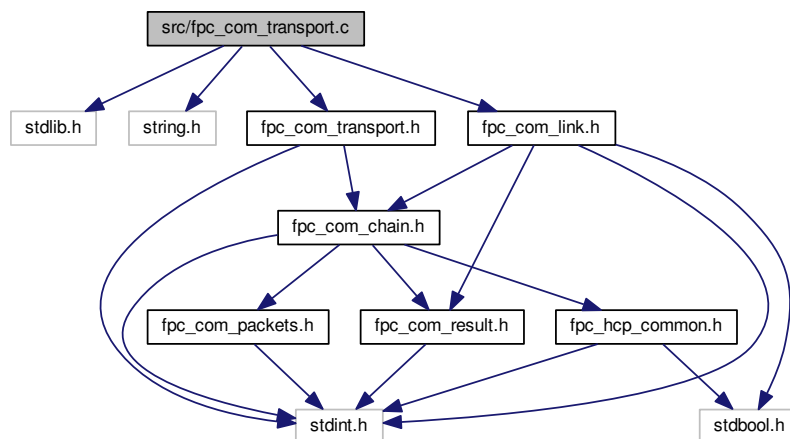
Here is the call graph for this function:



## 21.18 src/fpc\_com\_transport.c File Reference

Communication transport layer implementation.

```
#include <stdlib.h>
#include <string.h>
#include "fpc_com_link.h"
#include "fpc_com_transport.h"
Include dependency graph for fpc_com_transport.c:
```



### Functions

- `fpc_com_result_t fpc_com_transport_transmit(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`  
*Transmit a transport layer packet.*
- `fpc_com_result_t fpc_com_transport_receive(fpc_com_packet_tsp_t *packet, fpc_com_chain_t *chain)`  
*Receive a transport layer packet.*
- `uint16_t fpc_com_transport_get_overhead(uint16_t *offset)`  
*Returns the overhead of the layer.*

#### 21.18.1 Detailed Description

Communication transport layer implementation.



## 21.18.2 Function Documentation

### 21.18.2.1 uint16\_t fpc\_com\_transport\_get\_overhead ( uint16\_t \* *offset* )

Returns the overhead of the layer.

#### Parameters

out	<i>offset</i>	The offset to the packet data.
-----	---------------	--------------------------------

#### Returns

Overhead size in bytes.

Definition at line 88 of file fpc\_com\_transport.c.

References `fpc_com_packet_transport::seq_len`, `fpc_com_packet_transport::seq_nr`, and `fpc_com_packet_transport::size`.

### 21.18.2.2 fpc\_com\_result\_t fpc\_com\_transport\_receive ( fpc\_com\_packet\_tsp\_t \* *packet*, fpc\_com\_chain\_t \* *chain* )

Receive a transport layer packet.

#### Parameters

in, out	<i>packet</i>	The packet to populate.
in	<i>chain</i>	The chain to use.

#### Returns

[fpc\\_com\\_result\\_t](#)

Definition at line 60 of file fpc\_com\_transport.c.

References `fpc_com_packet_transport::data`, `fpc_com_packet_link::data`, `fpc_com_link_receive()`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_OK`, `fpc_com_packet_transport::seq_len`, `fpc_com_packet_transport::seq_nr`, and `fpc_com_packet_transport::size`.

Here is the call graph for this function:



21.18.2.3 `fpc_com_result_t fpc_com_transport_transmit ( fpc_com_packet_tsp_t * packet, fpc_com_chain_t * chain )`

Transmit a transport layer packet.

#### Parameters

in	<i>packet</i>	The packet to transmit.
in	<i>chain</i>	The chain to use.

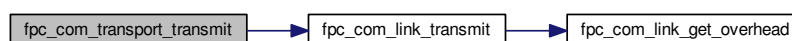
#### Returns

[fpc\\_com\\_result\\_t](#)

Definition at line 28 of file `fpc_com_transport.c`.

References `fpc_com_packet_link::channel`, `fpc_com_chain::channel`, `fpc_com_packet_link::data`, `FPC_COM_CHAIN_TX`, `fpc_com_link_transmit()`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `fpc_com_chain::link_overhead_get`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_packet_transport::seq_len`, `fpc_com_packet_transport::seq_nr`, `fpc_com_packet_transport::size`, `fpc_com_packet_link::size`, and `fpc_com_chain::tsp_overhead_get`.

Here is the call graph for this function:



## 21.19 `src/fpc_hcp.c` File Reference

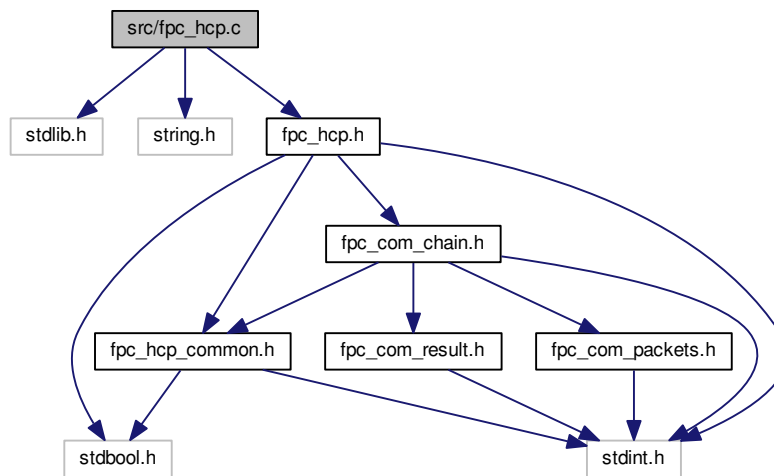
Host Communication Protocol implementation.

```

#include <stdlib.h>
#include <string.h>
#include "fpc_hcp.h"

```

Include dependency graph for fpc\_hcp.c:



## Macros

### HCP Packet Member Sizes

Macros for packet member sizes.

- #define `PACKET_ID_SIZE` `sizeof(((fpc_hcp_packet_t*)0)->id)`
- #define `PACKET_NUM_ARGS_SIZE` `sizeof(((fpc_hcp_packet_t*)0)->num_args)`
- #define `PACKET_HEADER_SIZE` `(PACKET_ID_SIZE + PACKET_NUM_ARGS_SIZE)`

### HCP Argument Member Sizes

Macros for argument member sizes.

- #define `ARGUMENT_ARG_SIZE` `sizeof(((fpc_hcp_arg_data_t*)0)->arg)`
- #define `ARGUMENT_SIZE_SIZE` `sizeof(((fpc_hcp_arg_data_t*)0)->size)`
- #define `ARGUMENT_HEADER_SIZE` `(ARGUMENT_ARG_SIZE + ARGUMENT_SIZE_SIZE)`

## Functions

- static `fpc_com_result_t` `recieve_chunks` (`fpc_com_chain_t` \*chain)  
*Handle receive chunks.*
- static `fpc_com_result_t` `transmit_chunks` (`fpc_com_chain_t` \*chain)  
*Handle transmit chunks.*
- uint16\_t `fpc_hcp_get_size` (`fpc_hcp_packet_t` \*packet, uint16\_t \*num\_args)  
*Calculate serialized packet size.*
- `fpc_com_result_t` `fpc_hcp_transmit` (`fpc_hcp_packet_t` \*packet, `fpc_com_chain_t` \*chain)  
*Transmits an application packet through the supplied transmit chain.*
- `fpc_com_result_t` `fpc_hcp_receive` (`fpc_hcp_packet_t` \*packet, `fpc_com_chain_t` \*chain)  
*Receives an application packet through the supplied transmit chain.*
- bool `fpc_hcp_arg_add` (`fpc_hcp_packet_t` \*packet, `fpc_hcp_arg_t` arg, uint16\_t size, bool free\_data, void \*data)

- Add argument to packet.*
- bool `fpc_hcp_arg_check` (`fpc_hcp_packet_t` \*packet, `fpc_hcp_arg_t` arg)  
*Check if command contains selected argument key.*
- `fpc_hcp_arg_data_t` \* `fpc_hcp_arg_get` (`fpc_hcp_packet_t` \*packet, `fpc_hcp_arg_t` arg)  
*Get Argument with specified key.*
- bool `fpc_hcp_arg_copy_data` (`fpc_hcp_packet_t` \*packet, `fpc_hcp_arg_t` arg, `uint16_t` data\_size, `uint8_t` \*data)  
*Copy data from an argument with specified key.*
- void `fpc_hcp_free` (`fpc_com_chain_t` \*chain, `fpc_hcp_packet_t` \*packet)  
*Frees the resources held by the packet i.e. the dynamic data held in the arguments.*

### 21.19.1 Detailed Description

Host Communication Protocol implementation.

### 21.19.2 Macro Definition Documentation

21.19.2.1 `#define ARGUMENT_ARG_SIZE sizeof(((fpc_hcp_arg_data_t*)0)->arg)`

Definition at line 44 of file `fpc_hcp.c`.

21.19.2.2 `#define ARGUMENT_HEADER_SIZE (ARGUMENT_ARG_SIZE + ARGUMENT_SIZE_SIZE)`

Definition at line 46 of file `fpc_hcp.c`.

21.19.2.3 `#define ARGUMENT_SIZE_SIZE sizeof(((fpc_hcp_arg_data_t*)0)->size)`

Definition at line 45 of file `fpc_hcp.c`.

21.19.2.4 `#define PACKET_HEADER_SIZE (PACKET_ID_SIZE + PACKET_NUM_ARGS_SIZE)`

Definition at line 35 of file `fpc_hcp.c`.

21.19.2.5 `#define PACKET_ID_SIZE sizeof(((fpc_hcp_packet_t*)0)->id)`

Definition at line 33 of file `fpc_hcp.c`.

21.19.2.6 `#define PACKET_NUM_ARGS_SIZE sizeof(((fpc_hcp_packet_t*)0)->num_args)`

Definition at line 34 of file `fpc_hcp.c`.

### 21.19.3 Function Documentation

21.19.3.1 `bool fpc_hcp_arg_add ( fpc_hcp_packet_t * packet, fpc_hcp_arg_t arg, uint16_t size, bool free_data, void * data )`

Add argument to packet.

#### Note

This function does not allocate any memory, it will only set the argument variables.

## Parameters

in	<i>packet</i>	Packet to add to.
in	<i>arg</i>	Argument id.
in	<i>size</i>	Size of argument data.
in	<i>free_data</i>	Set to true if data should be owned by the argument, false if user still owns data.
in	<i>data</i>	Pointer to argument data.

## Returns

true = success, false = failure.

Definition at line 145 of file fpc\_hcp.c.

References `fpc_hcp_arg_data::arg`, `ARG_NONE`, `fpc_hcp_packet::arguments`, `fpc_hcp_arg_data::data`, `fpc_hcp_arg_data::free_data`, `fpc_hcp_packet::num_args`, and `fpc_hcp_arg_data::size`.

### 21.19.3.2 bool fpc\_hcp\_arg\_check ( fpc\_hcp\_packet\_t \* *packet*, fpc\_hcp\_arg\_t *arg* )

Check if command contains selected argument key.

## Parameters

in	<i>packet</i>	The packet to scan.
in	<i>arg</i>	Argument to look for.

## Returns

true if found, false if not found.

Definition at line 169 of file fpc\_hcp.c.

References `fpc_hcp_arg_get()`.

Here is the call graph for this function:



### 21.19.3.3 bool fpc\_hcp\_arg\_copy\_data ( fpc\_hcp\_packet\_t \* *packet*, fpc\_hcp\_arg\_t *arg*, uint16\_t *data\_size*, uint8\_t \* *data* )

Copy data from an argument with specified key.

Argument data will be copied to specified data buffer. Remaining bytes in data will be cleared if the argument data size is less than data size when the argument contains data.

**Parameters**

in	<i>packet</i>	The packet to operate on.
in	<i>arg</i>	The arg to retrieve data from.
in	<i>data_size</i>	Number of bytes to copy.
in, out	<i>data</i>	Pointer to data buffer.

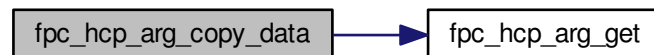
**Returns**

True if argument found, false if not found.

Definition at line 183 of file `fpc_hcp.c`.

References `fpc_hcp_arg_data::data`, `fpc_hcp_arg_get()`, and `fpc_hcp_arg_data::size`.

Here is the call graph for this function:



#### 21.19.3.4 `fpc_hcp_arg_data_t* fpc_hcp_arg_get ( fpc_hcp_packet_t * packet, fpc_hcp_arg_t arg )`

Get Argument with specified key.

**Parameters**

in	<i>packet</i>	The packet to operate on.
in	<i>arg</i>	The arg to retrieve.

**Returns**

Pointer to [fpc\\_hcp\\_arg\\_data\\_t](#) is successful, otherwise NULL.

Definition at line 173 of file `fpc_hcp.c`.

References `fpc_hcp_arg_data::arg`, `fpc_hcp_packet::arguments`, and `fpc_hcp_packet::num_args`.

#### 21.19.3.5 `void fpc_hcp_free ( fpc_com_chain_t * chain, fpc_hcp_packet_t * packet )`

Frees the resources held by the packet i.e. the dynamic data held in the arguments.

## Parameters

in	<i>chain</i>	Pointer to the communication chain used to retrieve the packet.
in	<i>packet</i>	Pointer to packet.

Definition at line 198 of file fpc\_hcp.c.

References `fpc_hcp_arg_data::arg`, `ARG_NONE`, `fpc_com_chain::argument_free`, `fpc_hcp_packet::arguments`, `CMD_NONE`, `fpc_com_chain::context`, `fpc_hcp_packet::id`, and `fpc_hcp_packet::num_args`.

### 21.19.3.6 `uint16_t fpc_hcp_get_size ( fpc_hcp_packet_t * packet, uint16_t * num_args )`

Calculate serialized packet size.

## Parameters

in	<i>packet</i>	Packet to calculate.
in, out	<i>num_args</i>	Will return number of arguments held by the command can be set to NULL.

## Returns

Serialized size.

Definition at line 64 of file fpc\_hcp.c.

References `fpc_hcp_arg_data::arg`, `ARG_NONE`, `ARGUMENT_HEADER_SIZE`, `fpc_hcp_packet::arguments`, `fpc_hcp_packet::num_args`, `PACKET_HEADER_SIZE`, and `fpc_hcp_arg_data::size`.

### 21.19.3.7 `fpc_com_result_t fpc_hcp_receive ( fpc_hcp_packet_t * packet, fpc_com_chain_t * chain )`

Receives an application packet through the supplied transmit chain.

## Parameters

in, out	<i>packet</i>	Pointer to pre-allocated packet struct.
in	<i>chain</i>	The chain to use.

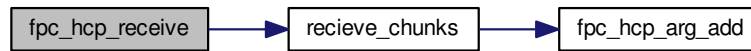
## Returns

[fpc\\_com\\_result\\_t](#)

Definition at line 117 of file fpc\_hcp.c.

References `fpc_com_chain::app_mtu_buffer`, `fpc_com_chain::app_mtu_size`, `FPC_COM_CHAIN_RX`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `fpc_com_chain_private::hcp_packet`, `fpc_com_chain::initialized`, `fpc_com_chain::link_overhead_get`, `fpc_com_chain::phy_mtu_buffer`, `fpc_com_chain::phy_mtu_size`, `fpc_com_chain::private_vars`, `recieve_chunks()`, and `fpc_com_chain::tsp_overhead_get`.

Here is the call graph for this function:



#### 21.19.3.8 fpc\_com\_result\_t fpc\_hcp\_transmit ( fpc\_hcp\_packet\_t \* packet, fpc\_com\_chain\_t \* chain )

Transmits an application packet through the supplied transmit chain.

##### Parameters

in	<i>packet</i>	Application packet to send.
in	<i>chain</i>	The chain to use.

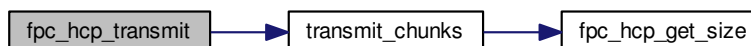
##### Returns

fpc\_com\_result\_t

Definition at line 89 of file fpc\_hcp.c.

References fpc\_com\_chain::app\_mtu\_buffer, fpc\_com\_chain::app\_mtu\_size, FPC\_COM\_CHAIN\_TX, FPC\_COM\_RESULT\_INVALID\_ARGUMENT, fpc\_com\_chain\_private::hcp\_packet, fpc\_com\_chain::initialized, fpc\_com\_chain::link\_overhead\_get, fpc\_com\_chain::phy\_mtu\_buffer, fpc\_com\_chain::phy\_mtu\_size, fpc\_com\_chain::private\_vars, transmit\_chunks(), and fpc\_com\_chain::tsp\_overhead\_get.

Here is the call graph for this function:



#### 21.19.3.9 static fpc\_com\_result\_t recieve\_chunks ( fpc\_com\_chain\_t \* chain ) [static]

Handle receive chunks.

##### Parameters

<i>chain</i>	Comminucation chain. return <a href="#">fpc_com_result_t</a>
--------------	--



Definition at line 211 of file fpc\_hcp.c.

References `fpc_com_chain::app_mtu_buffer`, `fpc_com_chain::app_mtu_size`, `fpc_com_chain::app_overhead_get`, `fpc_com_chain::app_packet_size`, `fpc_com_chain::app_rx`, `fpc_hcp_arg_data::arg`, `fpc_com_chain::argument_↵` allocator, `ARGUMENT_ARG_SIZE`, `fpc_com_chain::argument_free`, `ARGUMENT_SIZE_SIZE`, `fpc_com_chain_↵` ::context, `fpc_hcp_arg_data::data`, `FPC_COM_CHAIN_RX`, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_↵` COM\_RESULT\_NO\_MEMORY, `FPC_COM_RESULT_OK`, `fpc_hcp_arg_add()`, `fpc_hcp_arg_data::free_data`, `H_↵` CP\_MIN, `fpc_com_chain_private::hcp_packet`, `fpc_hcp_packet::id`, `PACKET_HEADER_SIZE`, `PACKET_ID_SIZE`, `PACKET_NUM_ARGS_SIZE`, `fpc_com_chain::private_vars`, and `fpc_hcp_arg_data::size`.

Here is the call graph for this function:



**21.19.3.10** `static fpc_com_result_t transmit_chunks ( fpc_com_chain_t *chain ) [static]`

Handle transmit chunks.

Parameters

<i>chain</i>	Comminucation chain. return <a href="#">fpc_com_result_t</a>
--------------	--

Definition at line 355 of file fpc\_hcp.c.

References `fpc_com_chain::app_mtu_buffer`, `fpc_com_chain::app_mtu_size`, `fpc_com_chain::app_overhead_get`, `fpc_com_chain::app_packet_size`, `fpc_com_chain::app_tx`, `fpc_hcp_arg_data::arg`, `ARGUMENT_ARG_SIZE`, `A_↵` RGUMENT\_HEADER\_SIZE, `ARGUMENT_SIZE_SIZE`, `fpc_hcp_packet::arguments`, `fpc_hcp_arg_data::data`, `F_↵` PC\_COM\_CHAIN\_TX, `FPC_COM_RESULT_INVALID_ARGUMENT`, `FPC_COM_RESULT_OK`, `fpc_hcp_get_↵` size(), `HCP_MIN`, `fpc_com_chain_private::hcp_packet`, `fpc_com_chain_private::hcp_seq_len`, `fpc_com_chain_↵` private::hcp\_seq\_nr, `fpc_hcp_packet::id`, `PACKET_HEADER_SIZE`, `PACKET_ID_SIZE`, `PACKET_NUM_ARGS_↵` \_SIZE, `fpc_com_chain::phy_mtu_size`, `fpc_com_chain::private_vars`, and `fpc_hcp_arg_data::size`.

Here is the call graph for this function:





# Index

ARG\_ACQUIRE  
fpc\_hcp\_common.h, [73](#)

ARG\_ADD  
fpc\_hcp\_common.h, [73](#)

ARG\_ALL  
fpc\_hcp\_common.h, [73](#)

ARG\_APP\_BASE\_VAL  
fpc\_hcp\_common.h, [72](#)

ARG\_APP\_BASE  
fpc\_hcp\_common.h, [74](#)

ARG\_BOOT  
fpc\_hcp\_common.h, [74](#)

ARG\_BUSY\_WAIT  
fpc\_hcp\_common.h, [74](#)

ARG\_CIPHERTEXT  
fpc\_hcp\_common.h, [74](#)

ARG\_CLAIM  
fpc\_hcp\_common.h, [74](#)

ARG\_COUNT  
fpc\_hcp\_common.h, [73](#)

ARG\_CREATE  
fpc\_hcp\_common.h, [73](#)

ARG\_DATA  
fpc\_hcp\_common.h, [73](#)

ARG\_DEBUG  
fpc\_hcp\_common.h, [74](#)

ARG\_DEEP\_SLEEP  
fpc\_hcp\_common.h, [74](#)

ARG\_DELETE  
fpc\_hcp\_common.h, [73](#)

ARG\_DONE  
fpc\_hcp\_common.h, [74](#)

ARG\_DOWNLOAD  
fpc\_hcp\_common.h, [73](#)

ARG\_DPI  
fpc\_hcp\_common.h, [74](#)

ARG\_EXTRACT  
fpc\_hcp\_common.h, [73](#)

ARG\_FFFF  
fpc\_hcp\_common.h, [74](#)

ARG\_FILL  
fpc\_hcp\_common.h, [74](#)

ARG\_FINGER\_DOWN  
fpc\_hcp\_common.h, [73](#)

ARG\_FINGER\_UP  
fpc\_hcp\_common.h, [73](#)

ARG\_FINISH  
fpc\_hcp\_common.h, [73](#)

ARG\_FLAG  
fpc\_hcp\_common.h, [73](#)

ARG\_FORMAT  
fpc\_hcp\_common.h, [73](#)

ARG\_GET  
fpc\_hcp\_common.h, [73](#)

ARG\_HEAP  
fpc\_hcp\_common.h, [74](#)

ARG\_HEIGHT  
fpc\_hcp\_common.h, [74](#)

ARG\_IDLE  
fpc\_hcp\_common.h, [74](#)

ARG\_IRQ\_STATUS  
fpc\_hcp\_common.h, [74](#)

ARG\_ID  
fpc\_hcp\_common.h, [73](#)

ARG\_LEVEL  
fpc\_hcp\_common.h, [73](#)

ARG\_MATCH\_IMAGE  
fpc\_hcp\_common.h, [73](#)

ARG\_MATCH  
fpc\_hcp\_common.h, [73](#)

ARG\_MAX\_SPI\_CLOCK  
fpc\_hcp\_common.h, [74](#)

ARG\_MAC  
fpc\_hcp\_common.h, [74](#)

ARG\_MODE  
fpc\_hcp\_common.h, [74](#)

ARG\_MTU  
fpc\_hcp\_common.h, [74](#)

ARG\_NONCE  
fpc\_hcp\_common.h, [74](#)

ARG\_NONE  
fpc\_hcp\_common.h, [73](#)

ARG\_NUM\_SUB\_AREAS\_HEIGHT  
fpc\_hcp\_common.h, [74](#)

ARG\_NUM\_SUB\_AREAS\_WIDTH  
fpc\_hcp\_common.h, [74](#)

ARG\_POWER\_MODE  
fpc\_hcp\_common.h, [74](#)

ARG\_PROD\_TEST  
fpc\_hcp\_common.h, [73](#)

ARG\_PROPERTIES  
fpc\_hcp\_common.h, [73](#)

ARG\_PUBLIC\_KEY  
fpc\_hcp\_common.h, [74](#)

ARG\_RANDOM  
fpc\_hcp\_common.h, [74](#)

ARG\_RELEASE  
fpc\_hcp\_common.h, [73](#)

ARG\_RESET\_HARD  
     fpc\_hcp\_common.h, 74  
 ARG\_RESET  
     fpc\_hcp\_common.h, 74  
 ARG\_RESULT  
     fpc\_hcp\_common.h, 73  
 ARG\_SAVE  
     fpc\_hcp\_common.h, 73  
 ARG\_SENSOR\_TYPE  
     fpc\_hcp\_common.h, 73  
 ARG\_SEQ\_LEN  
     fpc\_hcp\_common.h, 73  
 ARG\_SEQ\_NR  
     fpc\_hcp\_common.h, 73  
 ARG\_SET  
     fpc\_hcp\_common.h, 73  
 ARG\_SIZE  
     fpc\_hcp\_common.h, 73  
 ARG\_SLEEP  
     fpc\_hcp\_common.h, 74  
 ARG\_SPEED  
     fpc\_hcp\_common.h, 73  
 ARG\_STACK  
     fpc\_hcp\_common.h, 74  
 ARG\_START  
     fpc\_hcp\_common.h, 73  
 ARG\_STATUS  
     fpc\_hcp\_common.h, 74  
 ARG\_TIMEOUT  
     fpc\_hcp\_common.h, 74  
 ARG\_UNIQUE\_ID  
     fpc\_hcp\_common.h, 74  
 ARG\_UPDATE  
     fpc\_hcp\_common.h, 73  
 ARG\_UPLOAD  
     fpc\_hcp\_common.h, 73  
 ARG\_VERSION  
     fpc\_hcp\_common.h, 74  
 ARG\_WIDTH  
     fpc\_hcp\_common.h, 74  
 ARGUMENT\_ARG\_SIZE  
     fpc\_hcp.c, 82  
 ARGUMENT\_HEADER\_SIZE  
     fpc\_hcp.c, 82  
 ARGUMENT\_SIZE\_SIZE  
     fpc\_hcp.c, 82  
 app\_mtu\_buffer  
     fpc\_com\_chain, 40  
 app\_mtu\_size  
     fpc\_com\_chain, 40  
 app\_overhead\_get  
     fpc\_com\_chain, 41  
 app\_packet\_size  
     fpc\_com\_chain, 41  
 app\_rx  
     fpc\_com\_chain, 41  
 app\_tx  
     fpc\_com\_chain, 41  
 arg  
     fpc\_hcp\_arg\_data, 48  
 argument\_allocator  
     fpc\_com\_chain, 41  
 argument\_free  
     fpc\_com\_chain, 41  
 arguments  
     fpc\_hcp\_packet, 49  
 CMD\_APP\_BASE\_VAL  
     fpc\_hcp\_common.h, 72  
 CMD\_APP\_BASE  
     fpc\_hcp\_common.h, 75  
 CMD\_CANCEL  
     fpc\_hcp\_common.h, 75  
 CMD\_CAPTURE  
     fpc\_hcp\_common.h, 74  
 CMD\_COMMUNICATION  
     fpc\_hcp\_common.h, 75  
 CMD\_CONNECT  
     fpc\_hcp\_common.h, 75  
 CMD\_DEADPIXELS  
     fpc\_hcp\_common.h, 75  
 CMD\_DIAG  
     fpc\_hcp\_common.h, 75  
 CMD\_ENROLL  
     fpc\_hcp\_common.h, 74  
 CMD\_FFFF  
     fpc\_hcp\_common.h, 75  
 CMD\_GPIO  
     fpc\_hcp\_common.h, 75  
 CMD\_IDENTIFY  
     fpc\_hcp\_common.h, 75  
 CMD\_IMAGE  
     fpc\_hcp\_common.h, 75  
 CMD\_INFO  
     fpc\_hcp\_common.h, 75  
 CMD\_MATCH  
     fpc\_hcp\_common.h, 75  
 CMD\_MCU  
     fpc\_hcp\_common.h, 75  
 CMD\_NAVIGATE  
     fpc\_hcp\_common.h, 75  
 CMD\_NONE  
     fpc\_hcp\_common.h, 74  
 CMD\_RECONNECT  
     fpc\_hcp\_common.h, 75  
 CMD\_RESET  
     fpc\_hcp\_common.h, 75  
 CMD\_SENSOR  
     fpc\_hcp\_common.h, 75  
 CMD\_SETTINGS  
     fpc\_hcp\_common.h, 75  
 CMD\_STORAGE\_CALIBRATION  
     fpc\_hcp\_common.h, 75  
 CMD\_STORAGE\_LOG  
     fpc\_hcp\_common.h, 75  
 CMD\_STORAGE\_SETTINGS  
     fpc\_hcp\_common.h, 75

CMD\_STORAGE\_TEMPLATE  
     fpc\_hcp\_common.h, 75  
 CMD\_TEMPLATE  
     fpc\_hcp\_common.h, 75  
 CMD\_TEST  
     fpc\_hcp\_common.h, 75  
 CMD\_WAIT  
     fpc\_hcp\_common.h, 75  
 channel  
     fpc\_com\_chain, 41  
     fpc\_com\_packet\_link, 46  
 context  
     fpc\_com\_chain, 42  
 crc  
     fpc\_com\_packet\_link, 46  
 crc\_calc  
     fpc\_com\_chain, 42  
 data  
     fpc\_com\_packet\_link, 46  
     fpc\_com\_packet\_transport, 47  
     fpc\_hcp\_arg\_data, 48  
 doc/md/1\_stack.md, 51  
 doc/md/2\_hcpframe.md, 51  
 doc/md/4\_biometrics.md, 51  
 doc/md/5\_image.md, 51  
 doc/md/6\_template.md, 51  
 doc/md/7\_storage.md, 51  
 doc/md/8\_sensor.md, 51  
 doc/md/9\_device.md, 51  
 FPC\_COM\_ACK  
     fpc\_com\_packets.h, 58  
 FPC\_COM\_CHAIN\_RX  
     fpc\_com\_chain.h, 53  
 FPC\_COM\_CHAIN\_TX  
     fpc\_com\_chain.h, 53  
 FPC\_COM\_CHANNEL\_CLEAR  
     fpc\_com\_packets.h, 58  
 FPC\_COM\_CHANNEL\_END  
     fpc\_com\_packets.h, 58  
 FPC\_COM\_CHANNEL\_NONE  
     fpc\_com\_packets.h, 58  
 FPC\_COM\_CHANNEL\_TLS  
     fpc\_com\_packets.h, 58  
 FPC\_COM\_RESULT\_INVALID\_ARGUMENT  
     fpc\_com\_result.h, 60  
 FPC\_COM\_RESULT\_IO\_ERROR  
     fpc\_com\_result.h, 60  
 FPC\_COM\_RESULT\_NO\_MEMORY  
     fpc\_com\_result.h, 60  
 FPC\_COM\_RESULT\_NOT\_IMPLEMENTED  
     fpc\_com\_result.h, 60  
 FPC\_COM\_RESULT\_OK  
     fpc\_com\_result.h, 60  
 FPC\_COM\_RESULT\_TIMEOUT  
     fpc\_com\_result.h, 60  
 fpc\_com\_chain, 39  
     app\_mtu\_buffer, 40  
     app\_mtu\_size, 40  
     app\_overhead\_get, 41  
     app\_packet\_size, 41  
     app\_rx, 41  
     app\_tx, 41  
     argument\_allocator, 41  
     argument\_free, 41  
     channel, 41  
     context, 42  
     crc\_calc, 42  
     initialized, 42  
     link\_overhead\_get, 42  
     phy\_mtu\_buffer, 42  
     phy\_mtu\_size, 42  
     phy\_rx, 42  
     phy\_timeout\_rx, 43  
     phy\_timeout\_tx, 43  
     phy\_tx, 43  
     private\_vars, 43  
     session, 43  
     tsp\_overhead\_get, 43  
     tsp\_rx, 43  
     tsp\_tx, 44  
 fpc\_com\_chain.h  
     FPC\_COM\_CHAIN\_RX, 53  
     FPC\_COM\_CHAIN\_TX, 53  
     fpc\_com\_chain\_dir\_t, 53  
     fpc\_com\_chain\_private\_t, 53  
     fpc\_com\_chain\_t, 53  
 fpc\_com\_chain\_dir\_t  
     fpc\_com\_chain.h, 53  
 fpc\_com\_chain\_private, 44  
     hcp\_packet, 45  
     hcp\_seq\_len, 45  
     hcp\_seq\_nr, 45  
 fpc\_com\_chain\_private\_t  
     fpc\_com\_chain.h, 53  
 fpc\_com\_chain\_t  
     fpc\_com\_chain.h, 53  
 fpc\_com\_channel  
     fpc\_com\_packets.h, 58  
 fpc\_com\_channel\_t  
     fpc\_com\_packets.h, 58  
 fpc\_com\_link.c  
     fpc\_com\_link\_get\_overhead, 76  
     fpc\_com\_link\_receive, 77  
     fpc\_com\_link\_transmit, 77  
 fpc\_com\_link.h  
     fpc\_com\_link\_get\_overhead, 55  
     fpc\_com\_link\_receive, 55  
     fpc\_com\_link\_transmit, 55  
 fpc\_com\_link\_get\_overhead  
     fpc\_com\_link.c, 76  
     fpc\_com\_link.h, 55  
 fpc\_com\_link\_receive  
     fpc\_com\_link.c, 77  
     fpc\_com\_link.h, 55  
 fpc\_com\_link\_transmit

- fpc\_com\_link.c, 77
  - fpc\_com\_link.h, 55
- fpc\_com\_packet\_link, 45
  - channel, 46
  - crc, 46
  - data, 46
  - size, 46
- fpc\_com\_packet\_link\_t
  - fpc\_com\_packets.h, 58
- fpc\_com\_packet\_transport, 46
  - data, 47
  - seq\_len, 47
  - seq\_nr, 47
  - size, 47
- fpc\_com\_packet\_tsp\_t
  - fpc\_com\_packets.h, 58
- fpc\_com\_packets.h
  - FPC\_COM\_ACK, 58
  - FPC\_COM\_CHANNEL\_CLEAR, 58
  - FPC\_COM\_CHANNEL\_END, 58
  - FPC\_COM\_CHANNEL\_NONE, 58
  - FPC\_COM\_CHANNEL\_TLS, 58
  - fpc\_com\_channel, 58
  - fpc\_com\_channel\_t, 58
  - fpc\_com\_packet\_link\_t, 58
  - fpc\_com\_packet\_tsp\_t, 58
- fpc\_com\_result
  - fpc\_com\_result.h, 60
- fpc\_com\_result.h
  - FPC\_COM\_RESULT\_INVALID\_ARGUMENT, 60
  - FPC\_COM\_RESULT\_IO\_ERROR, 60
  - FPC\_COM\_RESULT\_NO\_MEMORY, 60
  - FPC\_COM\_RESULT\_NOT\_IMPLEMENTED, 60
  - FPC\_COM\_RESULT\_OK, 60
  - FPC\_COM\_RESULT\_TIMEOUT, 60
  - fpc\_com\_result, 60
  - fpc\_com\_result\_t, 60
- fpc\_com\_result\_t
  - fpc\_com\_result.h, 60
- fpc\_com\_transport.c
  - fpc\_com\_transport\_get\_overhead, 79
  - fpc\_com\_transport\_receive, 79
  - fpc\_com\_transport\_transmit, 79
- fpc\_com\_transport.h
  - fpc\_com\_transport\_get\_overhead, 61
  - fpc\_com\_transport\_receive, 61
  - fpc\_com\_transport\_transmit, 62
- fpc\_com\_transport\_get\_overhead
  - fpc\_com\_transport.c, 79
  - fpc\_com\_transport.h, 61
- fpc\_com\_transport\_receive
  - fpc\_com\_transport.c, 79
  - fpc\_com\_transport.h, 61
- fpc\_com\_transport\_transmit
  - fpc\_com\_transport.c, 79
  - fpc\_com\_transport.h, 62
- fpc\_hcp.c
  - ARGUMENT\_ARG\_SIZE, 82
  - ARGUMENT\_HEADER\_SIZE, 82
  - ARGUMENT\_SIZE\_SIZE, 82
  - fpc\_hcp\_arg\_add, 82
  - fpc\_hcp\_arg\_check, 83
  - fpc\_hcp\_arg\_copy\_data, 83
  - fpc\_hcp\_arg\_get, 84
  - fpc\_hcp\_free, 84
  - fpc\_hcp\_get\_size, 85
  - fpc\_hcp\_receive, 85
  - fpc\_hcp\_transmit, 86
  - PACKET\_HEADER\_SIZE, 82
  - PACKET\_ID\_SIZE, 82
  - PACKET\_NUM\_ARGS\_SIZE, 82
  - recieve\_chunks, 86
  - transmit\_chunks, 87
- fpc\_hcp.h
  - fpc\_hcp\_arg\_add, 64
  - fpc\_hcp\_arg\_check, 65
  - fpc\_hcp\_arg\_copy\_data, 65
  - fpc\_hcp\_arg\_get, 66
  - fpc\_hcp\_free, 66
  - fpc\_hcp\_get\_size, 66
  - fpc\_hcp\_receive, 67
  - fpc\_hcp\_transmit, 67
- fpc\_hcp\_arg
  - fpc\_hcp\_common.h, 73
- fpc\_hcp\_arg\_add
  - fpc\_hcp.c, 82
  - fpc\_hcp.h, 64
- fpc\_hcp\_arg\_check
  - fpc\_hcp.c, 83
  - fpc\_hcp.h, 65
- fpc\_hcp\_arg\_copy\_data
  - fpc\_hcp.c, 83
  - fpc\_hcp.h, 65
- fpc\_hcp\_arg\_data, 47
  - arg, 48
  - data, 48
  - free\_data, 48
  - size, 48
- fpc\_hcp\_arg\_data\_t
  - fpc\_hcp\_common.h, 72
- fpc\_hcp\_arg\_get
  - fpc\_hcp.c, 84
  - fpc\_hcp.h, 66
- fpc\_hcp\_arg\_t
  - fpc\_hcp\_common.h, 72
- fpc\_hcp\_cmd
  - fpc\_hcp\_common.h, 74
- fpc\_hcp\_cmd\_t
  - fpc\_hcp\_common.h, 72
- fpc\_hcp\_common.h
  - ARG\_ACQUIRE, 73
  - ARG\_ADD, 73
  - ARG\_ALL, 73
  - ARG\_APP\_BASE\_VAL, 72
  - ARG\_APP\_BASE, 74
  - ARG\_BOOT, 74

ARG\_BUSY\_WAIT, [74](#)  
ARG\_CIPHERTEXT, [74](#)  
ARG\_CLAIM, [74](#)  
ARG\_COUNT, [73](#)  
ARG\_CREATE, [73](#)  
ARG\_DATA, [73](#)  
ARG\_DEBUG, [74](#)  
ARG\_DEEP\_SLEEP, [74](#)  
ARG\_DELETE, [73](#)  
ARG\_DONE, [74](#)  
ARG\_DOWNLOAD, [73](#)  
ARG\_DPI, [74](#)  
ARG\_EXTRACT, [73](#)  
ARG\_FFFF, [74](#)  
ARG\_FILL, [74](#)  
ARG\_FINGER\_DOWN, [73](#)  
ARG\_FINGER\_UP, [73](#)  
ARG\_FINISH, [73](#)  
ARG\_FLAG, [73](#)  
ARG\_FORMAT, [73](#)  
ARG\_GET, [73](#)  
ARG\_HEAP, [74](#)  
ARG\_HEIGHT, [74](#)  
ARG\_IDLE, [74](#)  
ARG\_IRQ\_STATUS, [74](#)  
ARG\_ID, [73](#)  
ARG\_LEVEL, [73](#)  
ARG\_MATCH\_IMAGE, [73](#)  
ARG\_MATCH, [73](#)  
ARG\_MAX\_SPI\_CLOCK, [74](#)  
ARG\_MAC, [74](#)  
ARG\_MODE, [74](#)  
ARG\_MTU, [74](#)  
ARG\_NONCE, [74](#)  
ARG\_NONE, [73](#)  
ARG\_NUM\_SUB\_AREAS\_HEIGHT, [74](#)  
ARG\_NUM\_SUB\_AREAS\_WIDTH, [74](#)  
ARG\_POWER\_MODE, [74](#)  
ARG\_PROD\_TEST, [73](#)  
ARG\_PROPERTIES, [73](#)  
ARG\_PUBLIC\_KEY, [74](#)  
ARG\_RANDOM, [74](#)  
ARG\_RELEASE, [73](#)  
ARG\_RESET\_HARD, [74](#)  
ARG\_RESET, [74](#)  
ARG\_RESULT, [73](#)  
ARG\_SAVE, [73](#)  
ARG\_SENSOR\_TYPE, [73](#)  
ARG\_SEQ\_LEN, [73](#)  
ARG\_SEQ\_NR, [73](#)  
ARG\_SET, [73](#)  
ARG\_SIZE, [73](#)  
ARG\_SLEEP, [74](#)  
ARG\_SPEED, [73](#)  
ARG\_STACK, [74](#)  
ARG\_START, [73](#)  
ARG\_STATUS, [74](#)  
ARG\_TIMEOUT, [74](#)  
ARG\_UNIQUE\_ID, [74](#)  
ARG\_UPDATE, [73](#)  
ARG\_UPLOAD, [73](#)  
ARG\_VERSION, [74](#)  
ARG\_WIDTH, [74](#)  
CMD\_APP\_BASE\_VAL, [72](#)  
CMD\_APP\_BASE, [75](#)  
CMD\_CANCEL, [75](#)  
CMD\_CAPTURE, [74](#)  
CMD\_COMMUNICATION, [75](#)  
CMD\_CONNECT, [75](#)  
CMD\_DEADPIXELS, [75](#)  
CMD\_DIAG, [75](#)  
CMD\_ENROLL, [74](#)  
CMD\_FFFF, [75](#)  
CMD\_GPIO, [75](#)  
CMD\_IDENTIFY, [75](#)  
CMD\_IMAGE, [75](#)  
CMD\_INFO, [75](#)  
CMD\_MATCH, [75](#)  
CMD\_MCU, [75](#)  
CMD\_NAVIGATE, [75](#)  
CMD\_NONE, [74](#)  
CMD\_RECONNECT, [75](#)  
CMD\_RESET, [75](#)  
CMD\_SENSOR, [75](#)  
CMD\_SETTINGS, [75](#)  
CMD\_STORAGE\_CALIBRATION, [75](#)  
CMD\_STORAGE\_LOG, [75](#)  
CMD\_STORAGE\_SETTINGS, [75](#)  
CMD\_STORAGE\_TEMPLATE, [75](#)  
CMD\_TEMPLATE, [75](#)  
CMD\_TEST, [75](#)  
CMD\_WAIT, [75](#)  
fpc\_hcp\_arg, [73](#)  
fpc\_hcp\_arg\_data\_t, [72](#)  
fpc\_hcp\_arg\_t, [72](#)  
fpc\_hcp\_cmd, [74](#)  
fpc\_hcp\_cmd\_t, [72](#)  
fpc\_hcp\_packet\_t, [72](#)  
HCP\_MIN, [72](#)  
fpc\_hcp\_free  
    fpc\_hcp.c, [84](#)  
    fpc\_hcp.h, [66](#)  
fpc\_hcp\_get\_size  
    fpc\_hcp.c, [85](#)  
    fpc\_hcp.h, [66](#)  
fpc\_hcp\_packet, [48](#)  
    arguments, [49](#)  
    id, [49](#)  
    num\_args, [49](#)  
fpc\_hcp\_packet\_t  
    fpc\_hcp\_common.h, [72](#)  
fpc\_hcp\_receive  
    fpc\_hcp.c, [85](#)  
    fpc\_hcp.h, [67](#)  
fpc\_hcp\_transmit  
    fpc\_hcp.c, [86](#)

- fpc\_hcp.h, [67](#)
- free\_data
  - fpc\_hcp\_arg\_data, [48](#)
- HCP\_MIN
  - fpc\_hcp\_common.h, [72](#)
- hcp.md, [51](#)
- hcp\_packet
  - fpc\_com\_chain\_private, [45](#)
- hcp\_seq\_len
  - fpc\_com\_chain\_private, [45](#)
- hcp\_seq\_nr
  - fpc\_com\_chain\_private, [45](#)
- id
  - fpc\_hcp\_packet, [49](#)
- inc/fpc\_com\_chain.h, [51](#)
- inc/fpc\_com\_link.h, [54](#)
- inc/fpc\_com\_packets.h, [56](#)
- inc/fpc\_com\_result.h, [59](#)
- inc/fpc\_com\_transport.h, [60](#)
- inc/fpc\_hcp.h, [63](#)
- inc/fpc\_hcp\_common.h, [68](#)
- initialized
  - fpc\_com\_chain, [42](#)
- link\_overhead\_get
  - fpc\_com\_chain, [42](#)
- num\_args
  - fpc\_hcp\_packet, [49](#)
- PACKET\_HEADER\_SIZE
  - fpc\_hcp.c, [82](#)
- PACKET\_ID\_SIZE
  - fpc\_hcp.c, [82](#)
- PACKET\_NUM\_ARGS\_SIZE
  - fpc\_hcp.c, [82](#)
- phy\_mtu\_buffer
  - fpc\_com\_chain, [42](#)
- phy\_mtu\_size
  - fpc\_com\_chain, [42](#)
- phy\_rx
  - fpc\_com\_chain, [42](#)
- phy\_timeout\_rx
  - fpc\_com\_chain, [43](#)
- phy\_timeout\_tx
  - fpc\_com\_chain, [43](#)
- phy\_tx
  - fpc\_com\_chain, [43](#)
- private\_vars
  - fpc\_com\_chain, [43](#)
- recieve\_chunks
  - fpc\_hcp.c, [86](#)
- seq\_len
  - fpc\_com\_packet\_transport, [47](#)
- seq\_nr
  - fpc\_com\_packet\_transport, [47](#)
- session
  - fpc\_com\_chain, [43](#)
- size
  - fpc\_com\_packet\_link, [46](#)
  - fpc\_com\_packet\_transport, [47](#)
  - fpc\_hcp\_arg\_data, [48](#)
- src/fpc\_com\_link.c, [75](#)
- src/fpc\_com\_transport.c, [78](#)
- src/fpc\_hcp.c, [80](#)
- transmit\_chunks
  - fpc\_hcp.c, [87](#)
- tsp\_overhead\_get
  - fpc\_com\_chain, [43](#)
- tsp\_rx
  - fpc\_com\_chain, [43](#)
- tsp\_tx
  - fpc\_com\_chain, [44](#)