

# GCP (Generic Communications Protocol) Library

## 0.1

Generated by Doxygen 1.7.2

Tue May 22 2012 09:55:24



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	CRC16Params Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	GCPCConn Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	7
<b>4</b>	<b>File Documentation</b>	<b>9</b>
4.1	crc16.c File Reference . . . . .	9
4.1.1	Detailed Description . . . . .	10
4.1.2	Function Documentation . . . . .	10
4.1.2.1	crc16_check . . . . .	10
4.1.2.2	crc16_flush . . . . .	10
4.1.2.3	crc16_gen . . . . .	10
4.1.2.4	crc16_process_byte . . . . .	11
4.1.2.5	flip_16bit . . . . .	11
4.1.2.6	flip_8bit . . . . .	11
4.2	crc16.h File Reference . . . . .	11
4.2.1	Detailed Description . . . . .	12
4.2.2	Function Documentation . . . . .	12
4.2.2.1	crc16_check . . . . .	12
4.2.2.2	crc16_flush . . . . .	13
4.2.2.3	crc16_gen . . . . .	13
4.2.2.4	crc16_process_byte . . . . .	13
4.3	gcp.c File Reference . . . . .	14
4.3.1	Detailed Description . . . . .	15
4.3.2	Function Documentation . . . . .	15
4.3.2.1	gcp_init . . . . .	15
4.3.2.2	gcp_recv_byte . . . . .	15
4.3.2.3	gcp_send_byte . . . . .	16
4.3.2.4	recv_crc1 . . . . .	16
4.3.2.5	recv_crc2 . . . . .	16
4.3.2.6	recv_payload . . . . .	16
4.3.2.7	recv_preamble1 . . . . .	17

---

4.3.2.8	<a href="#">recv_preamble2</a>	17
4.3.2.9	<a href="#">recv_size1</a>	17
4.3.2.10	<a href="#">recv_size2</a>	17
4.3.2.11	<a href="#">send_crc1</a>	17
4.3.2.12	<a href="#">send_crc2</a>	18
4.3.2.13	<a href="#">send_payload</a>	18
4.3.2.14	<a href="#">send_preamble1</a>	18
4.3.2.15	<a href="#">send_preamble2</a>	18
4.3.2.16	<a href="#">send_size1</a>	19
4.3.2.17	<a href="#">send_size2</a>	19
4.4	<a href="#">gcp.h File Reference</a>	19
4.4.1	<a href="#">Detailed Description</a>	20
4.4.2	<a href="#">Enumeration Type Documentation</a>	20
4.4.2.1	<a href="#">GCPFrameState</a>	20
4.4.3	<a href="#">Function Documentation</a>	20
4.4.3.1	<a href="#">gcp_init</a>	20
4.4.3.2	<a href="#">gcp_recv_byte</a>	21
4.4.3.3	<a href="#">gcp_send_byte</a>	21

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">CRC16Params</a> (CRC-16 parameters ) . . . . .	5
<a href="#">GCPConn</a> (GCP connection parameters and state ) . . . . .	6



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">crc16.c</a>	9
<a href="#">crc16.h</a>	11
<a href="#">gcp.c</a>	14
<a href="#">gcp.h</a>	19





## Chapter 3

# Data Structure Documentation

### 3.1 CRC16Params Struct Reference

CRC-16 parameters.

```
#include <crc16.h>
```

#### Data Fields

- uint16\_t [prefix](#)  
*Prefix to be added to the data.*
- uint16\_t [poly](#)  
*Polynomial to be used.*
- unsigned [flip\\_bits](#): 1  
*When true, indicates that the most significant bit of each byte should be processed first.*
- unsigned [flip\\_bytes](#): 1  
*When true, indicates that the the bytes at the highest index in the buffer should be processed first.*
- unsigned [flip\\_output](#): 1  
*When true, indicates that the bits in the output should be reversed after calculating.*

#### 3.1.1 Detailed Description

CRC-16 parameters.

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

- [crc16.h](#)

## 3.2 GCPConn Struct Reference

GCP connection parameters and state.

```
#include <gcp.h>
```

### Data Fields

- `uint8_t * recv\_buf`  
*Pointer to the receive buffer.*
- `uint8_t * send\_buf`  
*Pointer to the send buffer.*
- `uint16_t recv\_size`  
*Size of the receive buffer.*
- `uint16_t send\_size`  
*Size of the data in the send buffer.*
- `uint16_t data\_size`  
*Size of the data in the receive buffer.*
- `uint16_t bytes\_rcvd`  
*Number of payload bytes received.*
- `uint16_t bytes\_sent`  
*Number of payload bytes sent.*
- `uint16_t recv\_crc`  
*The checksum received from the stream.*
- `uint16_t calc\_crc`  
*The calculated checksum of the data received.*
- `uint16_t send\_crc`  
*The checksum of the data being sent.*
- `GCPFrameState recv\_state`  
*The receive state.*
- `GCPFrameState send\_state`  
*The send state.*

- unsigned [recv\\_lock](#): 1

*When true, indicates that the receive buffer is being written to and should not be read from.*

- unsigned [send\\_lock](#): 1

*When true, indicates that the send buffer is being read from and should not be written to.*

### 3.2.1 Detailed Description

GCP connection parameters and state.

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

- [gcp.h](#)



## Chapter 4

# File Documentation

### 4.1 crc16.c File Reference

```
#include <config.h>
#include "crc16.h"
```

#### Functions

- static uint8\_t [flip\\_8bit](#) (uint8\_t val)  
*Reverses the bits in an 8-bit value.*
- static uint16\_t [flip\\_16bit](#) (uint16\_t val)  
*Reverses the bits in a 16-bit value.*
- uint16\_t [crc16\\_gen](#) (const uint8\_t \*data, uint16\_t size, const [CRC16Params](#) \*params)  
*Generates a CRC-16 checksum for a block of data.*
- int [crc16\\_check](#) (const uint8\_t \*data, uint16\_t size, const [CRC16Params](#) \*params, uint16\_t crc)  
*Validates a CRC-16 checksum for a block of data.*
- uint16\_t [crc16\\_process\\_byte](#) (uint16\_t prev, uint8\_t byte, uint16\_t poly, int msb\_first)  
*Process a single byte in a CRC-16 checksum calculation.*
- uint16\_t [crc16\\_flush](#) (uint16\_t prev, uint16\_t poly, int flip)  
*Processes the remaining 16 bits of padding in a CRC-16 checksum.*

### 4.1.1 Detailed Description

### 4.1.2 Function Documentation

**4.1.2.1** `int crc16_check ( const uint8_t * data, uint16_t size, const CRC16Params * params,  
uint16_t crc )`

Validates a CRC-16 checksum for a block of data.

#### Parameters

<i>data</i>	The data being checked.
<i>size</i>	The size (in bytes) of the data being checked.
<i>params</i>	A pointer to the checksum parameters.
<i>crc</i>	The checksum being checked.

#### Returns

0 if the checksum is valid; a negative value if data or params are NULL pointers; a positive value on an incorrect checksum.

**4.1.2.2** `uint16_t crc16_flush ( uint16_t prev, uint16_t poly, int flip )`

Processes the remaining 16 bits of padding in a CRC-16 checksum.

#### Parameters

<i>prev</i>	The value calculated before the flush.
<i>poly</i>	The polynomial to use.
<i>flip</i>	If non-zero, reverses the bit order of the output.

#### Returns

The calculated checksum.

**4.1.2.3** `uint16_t crc16_gen ( const uint8_t * data, uint16_t size, const CRC16Params * params  
)`

Generates a CRC-16 checksum for a block of data.

#### Parameters

<i>data</i>	The data being used to generate the checksum.
<i>size</i>	The size (in bytes) of the data being CRC'd.
<i>params</i>	A pointer to the checksum parameters.

#### Returns

The generated checksum on success; 0 on failure (data or params are NULL point-

ers).

#### 4.1.2.4 uint16\_t crc16\_process\_byte ( uint16\_t *prev*, uint8\_t *byte*, uint16\_t *poly*, int *msb\_first* )

Process a single byte in a CRC-16 checksum calculation.

##### Parameters

<i>prev</i>	The previously calculated value.
<i>byte</i>	The byte being processed.
<i>msb_first</i>	Set to a non-zero value when the most significant bit of the data is to be processed first.
<i>poly</i>	The polynomial being used for the checksum.

##### Returns

The newly calculated value.

#### 4.1.2.5 uint16\_t flip\_16bit ( uint16\_t *val* ) [static]

Reverses the bits in a 16-bit value.

##### Parameters

<i>val</i>	The value to be flipped.
------------	--------------------------

##### Returns

The flipped value.

#### 4.1.2.6 uint8\_t flip\_8bit ( uint8\_t *val* ) [static]

Reverses the bits in an 8-bit value.

##### Parameters

<i>val</i>	The value to be flipped.
------------	--------------------------

##### Returns

The flipped value.

## 4.2 crc16.h File Reference

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

## Data Structures

- struct [CRC16Params](#)  
*CRC-16 parameters.*

## Functions

- [uint16\\_t crc16\\_gen](#) (const [uint8\\_t](#) \*data, [uint16\\_t](#) size, const [CRC16Params](#) \*params)  
*Generates a CRC-16 checksum for a block of data.*
- [int crc16\\_check](#) (const [uint8\\_t](#) \*data, [uint16\\_t](#) size, const [CRC16Params](#) \*params, [uint16\\_t](#) crc)  
*Validates a CRC-16 checksum for a block of data.*
- [uint16\\_t crc16\\_process\\_byte](#) ([uint16\\_t](#) prev, [uint8\\_t](#) byte, [uint16\\_t](#) poly, int msb\_ - first)  
*Process a single byte in a CRC-16 checksum calculation.*
- [uint16\\_t crc16\\_flush](#) ([uint16\\_t](#) prev, [uint16\\_t](#) poly, int flip)  
*Processes the remaining 16 bits of padding in a CRC-16 checksum.*

### 4.2.1 Detailed Description

### 4.2.2 Function Documentation

#### 4.2.2.1 `int crc16_check ( const uint8\_t * data, uint16\_t size, const CRC16Params * params, uint16\_t crc )`

Validates a CRC-16 checksum for a block of data.

#### Parameters

<i>data</i>	The data being checked.
<i>size</i>	The size (in bytes) of the data being checked.
<i>params</i>	A pointer to the checksum parameters.
<i>crc</i>	The checksum being checked.

#### Returns

0 if the checksum is valid; a negative value if data or params are NULL pointers; a positive value on an incorrect checksum.



#### 4.2.2.2 uint16\_t crc16\_flush ( uint16\_t *prev*, uint16\_t *poly*, int *flip* )

Processes the remaining 16 bits of padding in a CRC-16 checksum.

##### Parameters

<i>prev</i>	The value calculated before the flush.
<i>poly</i>	The polynomial to use.
<i>flip</i>	If non-zero, reverses the bit order of the output.

##### Returns

The calculated checksum.

#### 4.2.2.3 uint16\_t crc16\_gen ( const uint8\_t \* *data*, uint16\_t *size*, const CRC16Params \* *params* )

Generates a CRC-16 checksum for a block of data.

##### Parameters

<i>data</i>	The data being used to generate the checksum.
<i>size</i>	The size (in bytes) of the data being CRC'd.
<i>params</i>	A pointer to the checksum parameters.

##### Returns

The generated checksum on success; 0 on failure (data or params are NULL pointers).

#### 4.2.2.4 uint16\_t crc16\_process\_byte ( uint16\_t *prev*, uint8\_t *byte*, uint16\_t *poly*, int *msb\_first* )

Process a single byte in a CRC-16 checksum calculation.

##### Parameters

<i>prev</i>	The previously calculated value.
<i>byte</i>	The byte being processed.
<i>msb_first</i>	Set to a non-zero value when the most significant bit of the data is to be processed first.
<i>poly</i>	The polynomial being used for the checksum.

##### Returns

The newly calculated value.

## 4.3 gcp.c File Reference

```
#include <config.h>
#include "gcp.h"
#include "crc16.h"
```

### Defines

- #define [POLY](#) 0x8005  
*The polynomial used by the GCP protocol for CRC calculation.*

### Functions

- static void [recv\\_preamble1](#) (GCPCConn \*c, uint8\_t b)  
*Reads the first byte of the preamble.*
- static void [recv\\_preamble2](#) (GCPCConn \*c, uint8\_t b)  
*Reads the second byte of the preamble.*
- static void [recv\\_size1](#) (GCPCConn \*c, uint8\_t b)  
*Reads the first byte of the data size.*
- static void [recv\\_size2](#) (GCPCConn \*c, uint8\_t b)  
*Reads the second byte of the data size.*
- static void [recv\\_payload](#) (GCPCConn \*c, uint8\_t b)  
*Reads a byte of payload data.*
- static void [recv\\_crc1](#) (GCPCConn \*c, uint8\_t b)  
*Reads the first byte of the checksum.*
- static void [recv\\_crc2](#) (GCPCConn \*c, uint8\_t b)  
*Reads the second byte of the checksum.*
- static uint8\_t [send\\_preamble1](#) (GCPCConn \*c)  
*Calculates the first byte of the preamble to be sent.*
- static uint8\_t [send\\_preamble2](#) (GCPCConn \*c)  
*Calculates the second byte of the preamble to be sent.*
- static uint8\_t [send\\_size1](#) (GCPCConn \*c)  
*Calculates the first byte of the payload size to be sent.*

- static uint8\_t [send\\_size2](#) (GCPConn \*c)  
*Calculates the second byte of the payload size to be sent.*
- static uint8\_t [send\\_payload](#) (GCPConn \*c)  
*Calculates the next byte of the payload to be sent.*
- static uint8\_t [send\\_crc1](#) (GCPConn \*c)  
*Calculates the first byte of the checksum to be sent.*
- static uint8\_t [send\\_crc2](#) (GCPConn \*c)  
*Calculates the second byte of the checksum to be sent.*
- int [gcp\\_init](#) (GCPConn \*c)  
*Initializes a connection object.*
- int [gcp\\_rcv\\_byte](#) (GCPConn \*c, uint8\_t b)  
*Processes a byte from the stream.*
- uint8\_t [gcp\\_send\\_byte](#) (GCPConn \*c)  
*Calculates the next byte to be sent to the stream.*

### 4.3.1 Detailed Description

### 4.3.2 Function Documentation

#### 4.3.2.1 int gcp\_init ( GCPConn \* c )

Initializes a connection object.

##### Parameters

<i>c</i>	A pointer to the connection object to be initialized.
----------	---

##### Returns

0 on success; a non-zero value on failure (c is a NULL pointer).

#### 4.3.2.2 int gcp\_rcv\_byte ( GCPConn \* c, uint8\_t b )

Processes a byte from the stream.

##### Parameters

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte read from the stream.

**Returns**

0 on success; a non-zero value on failure (*c* is a NULL pointer).

**4.3.2.3 uint8\_t gcp\_send\_byte ( GCPConn \* *c* )**

Calculates the next byte to be sent to the stream.

**Parameters**

<i>c</i>	A pointer to the connection.
----------	------------------------------

**Returns**

The next byte on success; 0 on failure (*c* is a NULL pointer).

**4.3.2.4 void recv\_crc1 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the first byte of the checksum.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.5 void recv\_crc2 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the second byte of the checksum.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.6 void recv\_payload ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads a byte of payload data.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.7 void recv\_preamble1 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the first byte of the preamble.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.8 void recv\_preamble2 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the second byte of the preamble.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.9 void recv\_size1 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the first byte of the data size.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.10 void recv\_size2 ( GCPConn \* *c*, uint8\_t *b* ) [static]**

Reads the second byte of the data size.

**Parameters**

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte being read.

**4.3.2.11 uint8\_t send\_crc1 ( GCPConn \* *c* ) [static]**

Calculates the first byte of the checksum to be sent.

**Parameters**

<i>c</i>	A pointer to the connection object.
----------	-------------------------------------

**Returns**

The first byte of the checksum.

**4.3.2.12** `uint8_t send_crc2 ( GCPCConn * c ) [static]`

Calculates the second byte of the checksum to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The second byte of the checksum.

**4.3.2.13** `uint8_t send_payload ( GCPCConn * c ) [static]`

Calculates the next byte of the payload to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The next byte of the payload.

**4.3.2.14** `uint8_t send_preamble1 ( GCPCConn * c ) [static]`

Calculates the first byte of the preamble to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The first byte of the preamble.

**4.3.2.15** `uint8_t send_preamble2 ( GCPCConn * c ) [static]`

Calculates the second byte of the preamble to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The second byte of the preamble.

**4.3.2.16** `uint8_t send_size1 ( GCPCConn * c ) [static]`

Calculates the first byte of the payload size to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The first byte of the payload size.

**4.3.2.17** `uint8_t send_size2 ( GCPCConn * c ) [static]`

Calculates the second byte of the payload size to be sent.

**Parameters**

<code>c</code>	A pointer to the connection object.
----------------	-------------------------------------

**Returns**

The second byte of the payload size.

## 4.4 gcp.h File Reference

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

**Data Structures**

- struct [GCPCConn](#)  
*GCP connection parameters and state.*

**Enumerations**

- enum [GCPFrameState](#) {  
    [gcp\\_preamble1](#), [gcp\\_preamble2](#), [gcp\\_size1](#), [gcp\\_size2](#),  
    [gcp\\_payload](#), [gcp\\_crc1](#), [gcp\\_crc2](#) }  
*Communication state.*

## Functions

- `int gcp_init (GCPConn *c)`  
*Initializes a connection object.*
- `int gcp_rcv_byte (GCPConn *c, uint8_t b)`  
*Processes a byte from the stream.*
- `uint8_t gcp_send_byte (GCPConn *c)`  
*Calculates the next byte to be sent to the stream.*

### 4.4.1 Detailed Description

### 4.4.2 Enumeration Type Documentation

#### 4.4.2.1 enum GCPFrameState

Communication state.

Enumerator:

***gcp\_preamble1*** Reading first byte of the preamble.  
***gcp\_preamble2*** Reading second byte of the preamble.  
***gcp\_size1*** Reading first byte of the payload size.  
***gcp\_size2*** Reading second byte of the payload size.  
***gcp\_payload*** Reading payload data.  
***gcp\_crc1*** Reading first byte of the checksum.  
***gcp\_crc2*** Reading second byte of the checksum.

### 4.4.3 Function Documentation

#### 4.4.3.1 `int gcp_init ( GCPConn * c )`

Initializes a connection object.

Parameters

<code>c</code>	A pointer to the connection object to be initialized.
----------------	---

Returns

0 on success; a non-zero value on failure (c is a NULL pointer).



#### 4.4.3.2 `int gcp_rcv_byte ( GCPCConn * c, uint8_t b )`

Processes a byte from the stream.

##### Parameters

<i>c</i>	A pointer to the connection object.
<i>b</i>	The byte read from the stream.

##### Returns

0 on success; a non-zero value on failure (*c* is a NULL pointer).

#### 4.4.3.3 `uint8_t gcp_send_byte ( GCPCConn * c )`

Calculates the next byte to be sent to the stream.

##### Parameters

<i>c</i>	A pointer to the connection.
----------	------------------------------

##### Returns

The next byte on success; 0 on failure (*c* is a NULL pointer).

# Index

- crc16.c, [9](#)
  - crc16\_check, [10](#)
  - crc16\_flush, [10](#)
  - crc16\_gen, [10](#)
  - crc16\_process\_byte, [11](#)
  - flip\_16bit, [11](#)
  - flip\_8bit, [11](#)
- crc16.h, [11](#)
  - crc16\_check, [12](#)
  - crc16\_flush, [12](#)
  - crc16\_gen, [13](#)
  - crc16\_process\_byte, [13](#)
- crc16\_check
  - crc16.c, [10](#)
  - crc16.h, [12](#)
- crc16\_flush
  - crc16.c, [10](#)
  - crc16.h, [12](#)
- crc16\_gen
  - crc16.c, [10](#)
  - crc16.h, [13](#)
- crc16\_process\_byte
  - crc16.c, [11](#)
  - crc16.h, [13](#)
- CRC16Params, [5](#)
- flip\_16bit
  - crc16.c, [11](#)
- flip\_8bit
  - crc16.c, [11](#)
- gcp.c, [14](#)
  - gcp\_init, [15](#)
  - gcp\_rcv\_byte, [15](#)
  - gcp\_send\_byte, [16](#)
  - recv\_crc1, [16](#)
  - recv\_crc2, [16](#)
  - recv\_payload, [16](#)
  - recv\_preamble1, [16](#)
  - recv\_preamble2, [17](#)
  - recv\_size1, [17](#)
  - recv\_size2, [17](#)
  - send\_crc1, [17](#)
  - send\_crc2, [18](#)
  - send\_payload, [18](#)
  - send\_preamble1, [18](#)
  - send\_preamble2, [18](#)
  - send\_size1, [19](#)
  - send\_size2, [19](#)
- gcp.h, [19](#)
  - gcp\_crc1, [20](#)
  - gcp\_crc2, [20](#)
  - gcp\_payload, [20](#)
  - gcp\_preamble1, [20](#)
  - gcp\_preamble2, [20](#)
  - gcp\_size1, [20](#)
  - gcp\_size2, [20](#)
  - gcp\_init, [20](#)
  - gcp\_rcv\_byte, [20](#)
  - gcp\_send\_byte, [21](#)
  - GCPFrameState, [20](#)
- gcp\_crc1
  - gcp.h, [20](#)
- gcp\_crc2
  - gcp.h, [20](#)
- gcp\_payload
  - gcp.h, [20](#)
- gcp\_preamble1
  - gcp.h, [20](#)
- gcp\_preamble2
  - gcp.h, [20](#)
- gcp\_size1
  - gcp.h, [20](#)
- gcp\_size2
  - gcp.h, [20](#)
- gcp\_init
  - gcp.c, [15](#)
  - gcp.h, [20](#)
- gcp\_rcv\_byte
  - gcp.c, [15](#)
  - gcp.h, [20](#)
- gcp\_send\_byte
  - gcp.h, [20](#)

- [gcp.c](#), [16](#)
  - [gcp.h](#), [21](#)
- [GCPConn](#), [6](#)
- [GCPFrameState](#)
  - [gcp.h](#), [20](#)
- [recv\\_crc1](#)
  - [gcp.c](#), [16](#)
- [recv\\_crc2](#)
  - [gcp.c](#), [16](#)
- [recv\\_payload](#)
  - [gcp.c](#), [16](#)
- [recv\\_preamble1](#)
  - [gcp.c](#), [16](#)
- [recv\\_preamble2](#)
  - [gcp.c](#), [17](#)
- [recv\\_size1](#)
  - [gcp.c](#), [17](#)
- [recv\\_size2](#)
  - [gcp.c](#), [17](#)
- [send\\_crc1](#)
  - [gcp.c](#), [17](#)
- [send\\_crc2](#)
  - [gcp.c](#), [18](#)
- [send\\_payload](#)
  - [gcp.c](#), [18](#)
- [send\\_preamble1](#)
  - [gcp.c](#), [18](#)
- [send\\_preamble2](#)
  - [gcp.c](#), [18](#)
- [send\\_size1](#)
  - [gcp.c](#), [19](#)
- [send\\_size2](#)
  - [gcp.c](#), [19](#)