GCP (Generic Communications Protocol) Library

0.1

Generated by Doxygen 1.7.2

Tue May 22 2012 09:55:24

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

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

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

prev	The value calculated before the flush.
poly	The polynomial to use.
flip	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

data	The data being used to generate the checksum.
size	The size (in bytes) of the data being CRC'd.
params	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

prev	The previously calculated value.
byte	The byte being processed.
msb_first	Set to a non-zero value when the most significant bit of the data is to be processed first.
poly	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

val	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

val	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

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

prev	The value calculated before the flush.
poly	The polynomial to use.
flip	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

data	The data being used to generate the checksum.
size	The size (in bytes) of the data being CRC'd.
params	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

prev	The previously calculated value.
byte	The byte being processed.
msb_first	Set to a non-zero value when the most significant bit of the data is to be processed first.
poly	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 (GCPConn *c, uint8_t b)

  Reads the first byte of the preamble.

- static void recv_preamble2 (GCPConn *c, uint8_t b)

  Reads the second byte of the preamble.

- static void recv_size1 (GCPConn *c, uint8_t b)

  Reads the first byte of the data size.

- static void recv_size2 (GCPConn *c, uint8_t b)

  Reads the second byte of the data size.

- static void recv_payload (GCPConn *c, uint8_t b)

  Reads a byte of payload data.

- static void recv_crc1 (GCPConn *c, uint8_t b)

  Reads the first byte of the checksum.

- static void recv_crc2 (GCPConn *c, uint8_t b)

  Reads the second byte of the checksum.

- static uint8_t send_preamble1 (GCPConn *c)

  Calculates the first byte of the preamble to be sent.

- static uint8_t send_preamble2 (GCPConn *c)

  Calculates the second byte of the preamble to be sent.

- static uint8_t send_size1 (GCPConn *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_recv_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

c	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_recv_byte (GCPConn * *c*, uint8_t *b*)

Processes a byte from the stream.

Parameters

c	A pointer to the connection object.
b	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

c	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

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.
b	The byte being read.

4.3.2.6 void recv_payload (GCPConn * c, uint8_t b) [static]

Reads a byte of payload data.

Parameters

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.
b	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

c	A pointer to the connection object.

Returns

The first byte of the checksum.

4.3.2.12 uint8_t send_crc2 (GCPConn * c) [static]

Calculates the second byte of the checksum to be sent.

Parameters

c	A pointer to the connection object.

Returns

The second byte of the checksum.

4.3.2.13 uint8_t send_payload (GCPConn * c) [static]

Calculates the next byte of the payload to be sent.

Parameters

c	A pointer to the connection object.

Returns

The next byte of the payload.

4.3.2.14 uint8_t send_preamble1 (GCPConn * c) [static]

Calculates the first byte of the preamble to be sent.

Parameters

c	A pointer to the connection object.

Returns

The first byte of the preamble.

4.3.2.15 uint8_t send_preamble2 (GCPConn * c) [static]

Calculates the second byte of the preamble to be sent.

Parameters

c	A pointer to the connection object.

Returns

The second byte of the preamble.

4.3.2.16 uint8_t send_size1 (GCPConn * c) [static]

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

Parameters

c	A pointer to the connection object.

Returns

The first byte of the payload size.

4.3.2.17 uint8_t send_size2 (GCPConn * c) [static]

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

Parameters

c	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 GCPConn

  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_recv_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

c	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_recv_byte (GCPConn * c, uint8_t b)

Processes a byte from the stream.

Parameters

c	A pointer to the connection object.
b	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 (GCPConn * c)

Calculates the next byte to be sent to the stream.

Parameters

c	A pointer to the connection.

Returns

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

Index