cparser

1.0

Generated by Doxygen 1.9.0

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 Cp_Command_t Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Field Documentation 5

3.1.2.1 callback 5

3.1.2.2 name 5

3.1.2.3 numOfParams 6

3.1.2.4 params 6

3.2 Cp_Param_t Struct Reference 6

3.2.1 Detailed Description 6

3.2.2 Field Documentation 6

3.2.2.1 letter 6

3.2.2.2 type 7

3.3 Dictionary_Dictionary_t Struct Reference 7

3.3.1 Detailed Description 7

3.3.2 Field Documentation 7

3.3.2.1 keys 7

3.3.2.2 numberOfElements 7

3.3.2.3 values 7

4 File Documentation **9**

4.1 inc/cparser.h File Reference 9

4.1.1 Detailed Description 10

4.1.2 Typedef Documentation 10

4.1.2.1 Cp_ParsedCallback_t 10

4.1.3 Enumeration Type Documentation 10

4.1.3.1 _Cp_ParamType_t 10

4.1.4 Function Documentation 10

4.1.4.1 Cp_FeedLine() 10

4.2 inc/cparser_config.h File Reference 11

4.2.1 Detailed Description 11

4.2.2 Macro Definition Documentation 11

4.2.2.1 CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH 11

4.2.2.2 CPARSER_CONFIG_MAX_NUM_OF_COMMANDS 11

4.2.2.3 CPARSER_CONFIG_MAX_NUM_OF_PARAMS 11

4.3 inc/dictionary.h File Reference 12

4.3.1 Detailed Description 12

4.3.2 Function Documentation 12
4.3.2.1 Dictionary_Add() 12
4.3.2.2 Dictionary_Clear() 13
4.3.2.3 Dictionary_DoesExist() 13
4.3.2.4 Dictionary_Get() 13
4.3.2.5 Dictionary_Remove() 14
4.4 inc/generic.h File Reference 14
4.4.1 Detailed Description 14
4.4.2 Enumeration Type Documentation 14
4.4.2.1 _Bool_t 14
4.5 src/cparser.c File Reference 15
4.5.1 Detailed Description 15
4.5.2 Function Documentation 15
4.5.2.1 Cp_FeedLine() 16
4.5.2.2 cropJerk() 16
4.5.2.3 doesMatch() 16
4.5.2.4 getLength() 17
4.5.2.5 getSign() 17
4.5.2.6 parseFields() 17
4.5.2.7 parseFloat() 18
4.5.2.8 parseFractional() 18
4.5.2.9 parseSignedInteger() 19
4.5.2.10 parseUnsignedInteger() 19
4.5.2.11 parseValue() 20

Index **21**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Cp_Command_t 5
Cp_Param_t 6
Dictionary_Dictionary_t 7

Chapter 2

File Index

2.1 File List

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

inc/cparser.h 9
inc/cparser_config.h 11
inc/dictionary.h 12
inc/generic.h 14
src/cparser.c 15

Chapter 3

Data Structure Documentation

3.1 Cp_Command_t Struct Reference

```
#include <cparser.h>
```

Data Fields

- const char name [CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH+1]
- const Cp_Param_t params [CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
- const Cp_ParsedCallback_t callback
- const uint8_t numOfParams

3.1.1 Detailed Description

Command structure.

3.1.2 Field Documentation

3.1.2.1 callback

```
const Cp_ParsedCallback_t Cp_Command_t::callback
```

Parsed callback function pointer

3.1.2.2 name

```
const char Cp_Command_t::name[CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH+1]
```

Null terminated command name string

3.1.2.3 numOfParams

```
const uint8_t Cp_Command_t::numOfParams
```

Number of parameters

3.1.2.4 params

```
const Cp_Param_t Cp_Command_t::params[CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
```

Array of parameters

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

- inc/cparser.h

3.2 Cp_Param_t Struct Reference

```
#include <cparser.h>
```

Data Fields

- char letter
- Cp_ParamType_t type

3.2.1 Detailed Description

Parameter structure. Each parameter in a command structure is an instance of this struct.

3.2.2 Field Documentation

3.2.2.1 letter

```
char Cp_Param_t::letter
```

Letter of the parameter

3.2.2.2 type

```
Cp_ParamType_t Cp_Param_t::type
```

Type of the parameter

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

- inc/cparser.h

3.3 Dictionary_Dictionary_t Struct Reference

```
#include <dictionary.h>
```

Data Fields

- char keys [CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
- void $*$ values [CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
- uint8_t numberOfElements

3.3.1 Detailed Description

Dictionary structure.

3.3.2 Field Documentation

3.3.2.1 keys

```
char Dictionary_Dictionary_t::keys[CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
```

Letters paired with the values(keys)

3.3.2.2 numberOfElements

```
uint8_t Dictionary_Dictionary_t::numberOfElements
```

Number of elements

3.3.2.3 values

```
void* Dictionary_Dictionary_t::values[CPARSER_CONFIG_MAX_NUM_OF_PARAMS]
```

Pointers to the values

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

- inc/dictionary.h

Chapter 4

File Documentation

4.1 inc/cparser.h File Reference

```
#include "generic.h"
#include "cparser_config.h"
#include "dictionary.h"
```

Data Structures

- struct Cp_Param_t
- struct Cp_Command_t

Typedefs

- typedef void(∗ Cp_ParsedCallback_t) (Dictionary_Dictionary_t ∗dictionary)

  Callback function prototype for the command parsed callbacks.

Enumerations

- enum _Cp_ParamType_t { CP_PARAM_TYPE_LETTER = 0, CP_PARAM_TYPE_INTEGER = 1, CP_PARAM_TYPE_REAL = 2 }

Functions

- void Cp_Reset (void)

  Clears the command parser registry.
- void Cp_Register (Cp_Command_t ∗command)

  Registers a command.
- uint8_t Cp_FeedLine (char ∗input)

  Feeds line of string.

4.1.1 Detailed Description

Interface of the cparser library.

4.1.2 Typedef Documentation

4.1.2.1 Cp_ParsedCallback_t

```
typedef void(* Cp_ParsedCallback_t) (Dictionary_Dictionary_t *dictionary)
```

Callback function prototype for the command parsed callbacks.

Parameters

dictionary	Pointer to the dictionary of parameters.

4.1.3 Enumeration Type Documentation

4.1.3.1 _Cp_ParamType_t

```
enum _Cp_ParamType_t
```

Parameter type enumeration. Determines type of the parameter to be parsed.

Enumerator

CP_PARAM_TYPE_LETTER	Letter
CP_PARAM_TYPE_INTEGER	Signed integer
CP_PARAM_TYPE_REAL	Real number

4.1.4 Function Documentation

4.1.4.1 Cp_FeedLine()

```
uint8_t Cp_FeedLine (
            char * input )
```

Feeds line of string.

Parameters

input	Line string.

Return values

TRUE	or FALSE.

4.2 inc/cparser_config.h File Reference

Macros

- #define CPARSER_CONFIG_MAX_NUM_OF_PARAMS 5
- #define CPARSER_CONFIG_MAX_NUM_OF_COMMANDS 25
- #define CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH 5

4.2.1 Detailed Description

Configuration parameters.

4.2.2 Macro Definition Documentation

4.2.2.1 CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH

```
#define CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH 5
```

Maximum command name length(except null terminator)

4.2.2.2 CPARSER_CONFIG_MAX_NUM_OF_COMMANDS

```
#define CPARSER_CONFIG_MAX_NUM_OF_COMMANDS 25
```

Maximum number of commands which can be registered.

4.2.2.3 CPARSER_CONFIG_MAX_NUM_OF_PARAMS

```
#define CPARSER_CONFIG_MAX_NUM_OF_PARAMS 5
```

Maximum number of parameters which a command can have.

4.3 inc/dictionary.h File Reference

```
#include "generic.h"
#include "cparser_config.h"
```

Data Structures

- struct Dictionary_Dictionary_t

Functions

- static void Dictionary_Add (Dictionary_Dictionary_t ∗dictionary, char key, void ∗value)

  Adds element to the dictionary.
- static void Dictionary_Remove (Dictionary_Dictionary_t ∗dictionary, char key)

  Removes element from the dictionary.
- static void Dictionary_Clear (Dictionary_Dictionary_t ∗dictionary)

  Clears all the elements of the dictionary.
- static Bool_t Dictionary_DoesExist (Dictionary_Dictionary_t ∗dictionary, char key)

  Check if the element exists in the given dictionary.
- static void ∗ Dictionary_Get (Dictionary_Dictionary_t ∗dictionary, char key)

  Parses pointer of the element value from the dictionary.

4.3.1 Detailed Description

Dictionary structure and related operations. These are used accross the parameter parsing process and passing them to the application.

4.3.2 Function Documentation

4.3.2.1 Dictionary_Add()

```
static void Dictionary_Add (
             Dictionary_Dictionary_t * dictionary,
             char key,
             void * value )  [inline], [static]
```

Adds element to the dictionary.

Parameters

dictionary	Pointer to the dictionary object.
key	Key of the element.
value	Pointer of the value of element.

4.3.2.2 Dictionary_Clear()

```
static void Dictionary_Clear (
            Dictionary_Dictionary_t * dictionary )  [inline], [static]
```

Clears all the elements of the dictionary.

Parameters

dictionary	Pointer to the dictionary object.

4.3.2.3 Dictionary_DoesExist()

```
static Bool_t Dictionary_DoesExist (
            Dictionary_Dictionary_t * dictionary,
            char key )  [inline], [static]
```

Check if the element exists in the given dictionary.

Parameters

dictionary	Pointer to the dictionary object.
key	Key of the element.

Return values

TRUE	or FALSE.

4.3.2.4 Dictionary_Get()

```
static void* Dictionary_Get (
            Dictionary_Dictionary_t * dictionary,
            char key )  [inline], [static]
```

Parses pointer of the element value from the dictionary.

Parameters

dictionary	Pointer to the dictionary object.
key	Key of the element.

Return values

NULL	or pointer to the element value.

4.3.2.5 Dictionary_Remove()

```
static void Dictionary_Remove (
            Dictionary_Dictionary_t * dictionary,
            char key )  [inline], [static]
```

Removes element from the dictionary.

Parameters

dictionary	Pointer to the dictionary object.
key	Key of the element.

4.4 inc/generic.h File Reference

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

Enumerations

- enum _Bool_t { FALSE = 0, TRUE = !FALSE }

4.4.1 Detailed Description

Definitions which are used globally are contained in this file.

4.4.2 Enumeration Type Documentation

4.4.2.1 _Bool_t

```
enum _Bool_t
```

Boolean enumeration.

Enumerator

FALSE	FALSE
TRUE	TRUE

4.5 src/cparser.c File Reference

```
#include "../inc/cparser.h"
#include "math.h"
```

Functions

- static Bool_t doesMatch (const char ∗cname, char ∗input, uint8_t length)

  Compare if the input matches to the command name.
- static void parseFields (char ∗input, uint8_t inputLength, Field_t ∗fields, uint8_t ∗numOfFields)

  Parse fields of the command string.
- static Bool_t parseValue (char ∗input, uint8_t inputLength, Cp_ParamType_t type, void ∗data, uint8_t ∗size)

  Parses the value of a given type.
- static Bool_t parseFloat (char ∗input, uint8_t start_idx, uint8_t length, float ∗value)

  Parses the floating point value.
- static Bool_t parseFractional (char ∗input, uint8_t start_idx, uint8_t length, float ∗value)

  Parses the fractional value.
- static Bool_t parseSignedInteger (char ∗input, uint8_t start_idx, uint8_t length, int32_t ∗value)

  Parses the signed integer value.
- static Bool_t parseUnsignedInteger (char ∗input, uint8_t start_idx, uint8_t length, uint32_t ∗value)

  Parses the unsigned integer value.
- static void getSign (char ∗input, uint8_t start_idx, uint8_t length, int8_t ∗sign, uint8_t ∗stop_idx)

  Gets the sign of a given value in the given char array.
- static void cropJerk (char ∗input, uint8_t start_idx, uint8_t length, uint8_t ∗stop_idx)

  Crops the meaningless data out of the char array.
- static uint8_t getLength (const char ∗input)

  Gets the length of a null terminated string(length except null termination)
- void Cp_Reset (void)

  Clears the command parser registry.
- void Cp_Register (Cp_Command_t ∗command)

  Registers a command.
- uint8_t Cp_FeedLine (char ∗input)

  Feeds line of string.

4.5.1 Detailed Description

Cparser library core module.

4.5.2 Function Documentation

4.5.2.1 Cp_FeedLine()

```
uint8_t Cp_FeedLine (
            char * input )
```

Feeds line of string.

Parameters

input	Line string.

Return values

TRUE	or FALSE.

4.5.2.2 cropJerk()

```
void cropJerk (
            char * input,
            uint8_t start_idx,
            uint8_t length,
            uint8_t * stop_idx )  [static]
```

Crops the meaningless data out of the char array.

Parameters

input	Pointer to the input char array.
start_idx	Start index of the process.
length	Window of the process.
stop_idx	Pointer to the meaningfull data.

4.5.2.3 doesMatch()

```
Bool_t doesMatch (
            const char * cname,
            char * input,
            uint8_t length )  [static]
```

Compare if the input matches to the command name.

Parameters

cname	Null terminated command name string.
input	Input char array.
length	Length of the input.

Return values

TRUE	or FALSE.

4.5.2.4 getLength()

```
uint8_t getLength (
             const char * input )  [static]
```

Gets the length of a null terminated string(length except null termination)

Parameters

input	Pointer to the string.

Return values

0xFF	if not found. Length if found.

4.5.2.5 getSign()

```
void getSign (
             char * input,
             uint8_t start_idx,
             uint8_t length,
             int8_t * sign,
             uint8_t * stop_idx )  [static]
```

Gets the sign of a given value in the given char array.

Parameters

input	Char array to be processed.
start_idx	Start index of the processing.
length	Window of the process.
sign	Pointer to return 1 or -1(positive or negative respectively).
stop_idx	Pointer to return index next to the sign element index.

4.5.2.6 parseFields()

```
void parseFields (
             char * input,
             uint8_t inputLength,
             Field_t * fields,
             uint8_t * numOfFields ) [static]
```

Parse fields of the command string.

Parameters

input	Command line string.
inputLength	Length of the input line string.
fields	Pointer to return fields of the line string.
numOfFields	Pointer to return number of fields.

4.5.2.7 parseFloat()

```
Bool_t parseFloat (
            char * input,
            uint8_t start_idx,
            uint8_t length,
            float * value ) [static]
```

Parses the floating point value.

Parameters

input	Input char array.
start_idx	Start index of the number.
length	Length of the char array.
value	Pointer to the return value.

Return values

TRUE	or FALSE.

4.5.2.8 parseFractional()

```
Bool_t parseFractional (
            char * input,
            uint8_t start_idx,
            uint8_t length,
            float * value ) [static]
```

Parses the fractional value.

Parameters

input	Input char array.
start_idx	Start index of the number.
length	Length of the char array.
value	Pointer to the return value.

Return values

TRUE	or FALSE.

4.5.2.9 parseSignedInteger()

```
Bool_t parseSignedInteger (
            char * input,
            uint8_t start_idx,
            uint8_t length,
            int32_t * value )  [static]
```

Parses the signed integer value.

Parameters

input	Input char array.
start_idx	Start index of the number.
length	Length of the char array.
value	Pointer to the return value.

Return values

TRUE	or FALSE.

4.5.2.10 parseUnsignedInteger()

```
Bool_t parseUnsignedInteger (
            char * input,
            uint8_t start_idx,
            uint8_t length,
            uint32_t * value )  [static]
```

Parses the unsigned integer value.

Parameters

input	Input char array.

Parameters

start_idx	Start index of the number.
length	Length of the char array.
value	Pointer to the return value.

Return values

TRUE	or FALSE.

4.5.2.11 parseValue()

```
Bool_t parseValue (
            char * input,
            uint8_t inputLength,
            Cp_ParamType_t type,
            void * data,
            uint8_t * size ) [static]
```

Parses the value of a given type.

Parameters

input	Input char array.
inputLength	Length of the char array.
type	Type of the parameter.
data	Pointer to the return value.
size	Pointer to the return value memory size.

Return values

TRUE	or FALSE.

Index

_Bool_t
- generic.h, 14

_Cp_ParamType_t
- cparser.h, 10

callback
- Cp_Command_t, 5

Cp_Command_t, 5
- callback, 5
- name, 5
- numOfParams, 5
- params, 6

Cp_FeedLine
- cparser.c, 15
- cparser.h, 10

Cp_Param_t, 6
- letter, 6
- type, 6

CP_PARAM_TYPE_INTEGER
- cparser.h, 10

CP_PARAM_TYPE_LETTER
- cparser.h, 10

CP_PARAM_TYPE_REAL
- cparser.h, 10

Cp_ParsedCallback_t
- cparser.h, 10

cparser.c
- Cp_FeedLine, 15
- cropJerk, 16
- doesMatch, 16
- getLength, 17
- getSign, 17
- parseFields, 17
- parseFloat, 18
- parseFractional, 18
- parseSignedInteger, 19
- parseUnsignedInteger, 19
- parseValue, 20

cparser.h
- _Cp_ParamType_t, 10
- Cp_FeedLine, 10
- CP_PARAM_TYPE_INTEGER, 10
- CP_PARAM_TYPE_LETTER, 10
- CP_PARAM_TYPE_REAL, 10
- Cp_ParsedCallback_t, 10

cparser_config.h
- CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH, 11
- CPARSER_CONFIG_MAX_NUM_OF_COMMANDS, 11
- CPARSER_CONFIG_MAX_NUM_OF_PARAMS, 11

CPARSER_CONFIG_MAX_COMMAND_NAME_LENGTH
- cparser_config.h, 11

CPARSER_CONFIG_MAX_NUM_OF_COMMANDS
- cparser_config.h, 11

CPARSER_CONFIG_MAX_NUM_OF_PARAMS
- cparser_config.h, 11

cropJerk
- cparser.c, 16

dictionary.h
- Dictionary_Add, 12
- Dictionary_Clear, 13
- Dictionary_DoesExist, 13
- Dictionary_Get, 13
- Dictionary_Remove, 14

Dictionary_Add
- dictionary.h, 12

Dictionary_Clear
- dictionary.h, 13

Dictionary_Dictionary_t, 7
- keys, 7
- numberOfElements, 7
- values, 7

Dictionary_DoesExist
- dictionary.h, 13

Dictionary_Get
- dictionary.h, 13

Dictionary_Remove
- dictionary.h, 14

doesMatch
- cparser.c, 16

FALSE
- generic.h, 15

generic.h
- _Bool_t, 14
- FALSE, 15
- TRUE, 15

getLength
- cparser.c, 17

getSign
- cparser.c, 17

inc/cparser.h, 9

inc/cparser_config.h, 11

inc/dictionary.h, 12

inc/generic.h, 14

keys
    Dictionary_Dictionary_t, 7

letter
    Cp_Param_t, 6

name
    Cp_Command_t, 5
numberOfElements
    Dictionary_Dictionary_t, 7
numOfParams
    Cp_Command_t, 5

params
    Cp_Command_t, 6
parseFields
    cparser.c, 17
parseFloat
    cparser.c, 18
parseFractional
    cparser.c, 18
parseSignedInteger
    cparser.c, 19
parseUnsignedInteger
    cparser.c, 19
parseValue
    cparser.c, 20

src/cparser.c, 15

TRUE
    generic.h, 15
type
    Cp_Param_t, 6

values
    Dictionary_Dictionary_t, 7