Particle Photon software

2.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

JSON Parser and Generator

There are a number of JSON parsers and generators for Particle products including the popular `SparkJson` library and `JSMNSpark`.

I created yet another library because I wanted something lightweight. SparkJson creates piles of objects that are copies of the original data during parsing. `JSMN` is very lightweight, but is kind of a pain to use.

What I did was wrap JSMN with an easier to use C++ API, along with adding easy value accessors.

I also added a JSON generator that's nearly as efficient as using sprintf, but much easier to use. It takes care of escaping quotes and special characters, and converts UTF-8 to JSON UTF-16 entities.

The parser and generator are separated internally so if you only need one or the other the linker will remove the unnecessary code automatically to save space.

The `full API documentation can be found here`.

JSON Parser

The parser can be used in many situations, but it's particularly well-suited for handing responses from webhooks, including multi-part responses.

The parser can be used in two different ways: static allocation, where almost all of the memory location is done in advance, or dynamically.

To do it dynamically, just construct the `JsonParser` object as a global or local variable:

```
JsonParser parser;
```

To do it statically, you need to guess the maximum size of the data you want to receive and the maximum number of tokens it will have. Each object is one token, plus two tokens for each key/value pair. Each array is one token, plus one token for each value in the array.

This `JsonParserStatic` example creates a static parser to parse up to 1024 bytes of data and 50 tokens:

```
JsonParserStatic<1024, 50> parser;
```

You then typically add the data to parse using the `addData` or `addString` method. If you're getting the data from a subscribe handler, you'll probably use addString.

```
parser.addString(data);
```

If you have a pointer and length, the addData method can be used instead.

Then, once all of the data has been added, call `parse`. This is handy for webhooks where you may get a multipart response. Example 3 demonstrates this:

```
void subscriptionHandler(const char *event, const char *data) {
    int responseIndex = 0;

    const char *slashOffset = strrchr(event, '/');
    if (slashOffset) {
        responseIndex = atoi(slashOffset + 1);
    }

    if (responseIndex == 0) {
        jsonParser.clear();
    }
    jsonParser.addString(data);

    if (jsonParser.parse()) {
        // Looks valid (we received all parts)

        // This printing thing is just for testing purposes, you should use the commands to
        // process data
        printJson(jsonParser);
    }
}
```

Say you have this object:

```
{
  "t1":"abc",
  "t2":1234,
  "t3":1234.5,
  "t4":true,
  "t5":false,
  "t6":null,
  "t7":"\"quoted\""
}
```

You could read the value of t1 by using `getOuterValueByKey` and this code:

```
String strValue;
parser1.getOuterValueByKey("t1", strValue);
```

This also works for other data types:

```
int intValue;
parser1.getOuterValueByKey("t2", intValue)

float floatValue;
parser1.getOuterValueByKey("t3", floatValue);

bool boolValue;
parser1.getOuterValueByKey("t4", boolValue);
```

There's also a fluent-style API that can make reading complex JSON easier. For example, given this fragment of JSON:

```
{
    "response": {
        "version": "0.1",
        "termsofService": "http://www.wunderground.com/weather/api/d/terms.html",
        "features": {
            "forecast": 1
        }
    },
    "forecast": {
        "txt_forecast": {
            "date": "12:25 PM EST",
            "forecastday": {
                "period": 7,
                "icon": "nt_partlycloudy",
                "icon_url": "http://icons.wxug.com/i/c/k/nt_partlycloudy.gif",
                "title": "Saturday Night",
                "fcttext": "Partly cloudy early with increasing clouds overnight. Low 29F. Winds NW at 15
       to 25 mph.",
                "fcttext_metric": "Partly cloudy early with increasing clouds overnight. Low -2C. Winds NW
       at 25 to 40 km/h.",
                "pop": "20"
            }
        },

String s = parser.getReference().key("response").key("version").valueString();
// s == "0.1"

s = parser.getReference().key("forecast").key("txt_forecast").key("date").valueString();
// s = "12:25 PM EST"

int value =
       parser.getReference().key("forecast").key("txt_forecast").key("forecastday").key("period").valueInt();
// value == 7
```

If you have a complicated JSON file to decode, using the `JSON Parser Tool` makes it easy. You paste in your JSON and it formats it nicely. Click on a row and will generate the fluent accessor to get that value!

JSON Generator

The JSON Generator is used to build valid JSON strings. While you can build JSON using sprintf, the JSON generator is able to double-quote escape strings, and escape double quotes within strings. It can also generate correct JSON unicode characters.

The most common use is to construct a static buffer to hold the JSON data for Particle.publish. Since this data is limited to 256 bytes, this is a reasonable approach using `JsonWriterStatic`:

```
JsonWriterStatic<256> jw;
```

You can also dynamically allocate a buffer using the plain `JsonWriter`.

The JsonWriter handles nested objects and arrays, but does so without creating temporary copies of the objects. Because of this, it's necessary to use startObject(), startArray(), and finishObjectOrArray() so the objects are balanced properly.

To make this easier, the `JsonWriterAutoObject` can be instantiated on the stack. When the object goes out of scope, it will automatically close the object. You use it like this:

```
{
    JsonWriterAutoObject obj(&jw);

    // Add various types of data
    jw.insertKeyValue("a", true);
    jw.insertKeyValue("b", 1234);
    jw.insertKeyValue("c", "test");
}
```

This will output the JSON data:

```
{\"a\":true,\"b\":1234,\"c\":\"test\"}
```

If you are sending float or double values you may want to limit the number of decimal places to send. This is done using `setFloatPlaces`.

JsonModifier

The JsonModifier class (added in version 0.1.0) makes it possible to modify an existing object that has been parsed with JsonParser.

You will typically process a JSON object using a `JsonParser` object, `addString()` or `addData()` method, then `parse()`.

Assuming your `JsonParser` is in the variable `jp` you then construct a temporary modifier object on the stack like this:

```
JsonModifier mod(jp);
```

The most common thing to do is have a JSON object and you want to update the value, or insert the value if it does not exist:

```
mod.insertOrUpdateKeyValue(jp.getOuterObject(), "a", (int)1);
```

If the input JSON was empty, it would then be:

```
{"a":1}
```

You can add int, long, float, double, bool, and const char $*$ objects this way.

```
mod.insertOrUpdateKeyValue(jp.getOuterObject(), "b", "testing");
```

This would change the object to:

```
{"a":1,"b":"testing"}
```

Updating an object will remove it from its current location and add it at the end of the object.

Another common function is `appendArrayValue()` which appends to an array.

You can also use `removeKeyValue()` and `removeArrayIndex()` to remove keys or array entries.

Examples

There are three Particle devices examples.

1 - Parser

The parser example is a standalone test of parsing some JSON data. The data is built into the code, so just just run it and monitor the serial output to make sure the test passes.

It also demonstrates how to read simple values out of the JSON data.

2 - Generator

The generator example is a standalone test of generating some JSON data. The data is built into the code, so just just run it and monitor the serial output to make sure the test passes.

It also demonstrates how to write JSON data.

3 - Subscription

This example creates a subscription on the event jsonParserTest, so you can send it JSON data, and it will parse and print it to the debuggging serial. For example, if you published these three events:

```
particle publish jsonParserTest '{"a":1234}' --private
particle publish jsonParserTest '{"a":1234,"b":"test"}' --private
particle publish jsonParserTest '{"a":1234,"b":"test":"c":[1,2,3]}' --private
```

You'd get these three objects printed to debugging serial.

```
{
  "a":1234
}
{
  "a":1234,
  "b":"test"
}
{
  "a":1234,
  "b":"test",
  "c":  [
    1,
    2,
    3
  ]

}
```

It also demonstrates how to handle multi-part webhook responses.

Test code

The github repository also has code in the test directory. It can run an automated test of several sample data files to verify operation. It's run by doing something like:

```
cd test
make
```

On Linux only, if you have valgrind installed, it can also do a build with valgrind checking to check for memory leaks and buffer overruns. It's run by doing:

```
cd test
make check
```

The test code is also a reference of various ways you can call the API.

Version History

0.1.3 (2020-09-22)

- Added JsonWriter methods insertKeyArray() and insertKeyVector() to make it easier to add arrays.
- Added JsonWriter methods insertArray() and insertVector() to make it easier to add arrays.

0.1.1 (2020-05-14)

Fixed a bug where calling parse() on an empty buffer returns true. It should return false. See issue #7.

0.1.0 (2019-09-18)

Added support for JsonModifier, a class to modify an existing JSON object in place, without making a copy of it.

0.0.7 (2019-08-30)

Fixed a bug in the 3-subscription example. The check for the part number should use strrchr, not strchr, because it needs to find the last slash before the part number for webhook multi-part responses.

Chapter 2

RFID

Update for Libraries 2.0 by Paul Kourany, Jan 2017 - v1.0.3 Adapted for Spark Core by Paul Kourany, May 2014

v0.1.2 - SOS bug fixed, now compatible with all Particle devices

Read a card using a mfrc522 reader on your SPI interface on your Arduino

- Pin layout should be as follows (on Spark Core):
- MOSI: Pin A5
- MISO: Pin A4
- SCK : Pin A3
- SS : Pin A2 (Configurable)
- RST : Pin D2 (Configurable)
-

Arduino RFID Library for MFRC522

Read a card using a mfrc522 reader on your SPI interface on your Arduino

- Pin layout should be as follows (on Arduino Uno):
- MOSI: Pin 11 / ICSP-4
- MISO: Pin 12 / ICSP-1
- SCK : Pin 13 / ISCP-3
- SS : Pin 10 (Configurable)
- RST : Pin 9 (Configurable)
-
- Pin layout should be as follows (on Arduino Mega):
- MOSI: Pin 51 / ICSP-4
- MISO: Pin 50 / ICSP-1
- SCK : Pin 52 / ISCP-3
- SS : Pin 53 (Configurable)
- RST : Pin 5 (Configurable)

Chapter 3

MQTT for Photon, Spark Core

MQTT publish/subscribe library for Photon, Spark Core version 0.4.28.

Source Code

This lightweight library source code are only 2 files. firmware -> MQTT.cpp, MQTT.h.

Application can use QOS0,1,2 and retain flag when send a publish message.

Example

Some sample sketches for Spark Core and Photon included(firmware/examples/).

- mqtttest.ino : simple pub/sub sample.
- mqttqostest.ino : QoS1, QoS2 publish and callback sample.

developer examples

some applications use MQTT with Photon. here are developer's reference examples.

- Spark Core / Photon and CloudMQTT
- MQTT Publish-Subscribe Using Rpi, ESP and Photon
- Particle Photon on Watson IoT
- Connecting IoT devices to the Watson Conversation Car-Dashboard app
- ThingSpeak MQTT API
- HOW TO CONNECT A PARTICLE PHOTON TO THE LOSANT IOT PLATFORM
- How I Hacked my Humidor with Losant and a Particle Photon
- ARTIK as MQTT Message Broker
- Particle and Ubidots using MQTT

- USING TWILIO SYNC WITH MQTT ON A PARTICLE PHOTON

## sample source

```
#include "application.h"
#include "MQTT.h"

void callback(char* topic, byte* payload, unsigned int length);
MQTT client("iot.eclipse.org", 1883, callback);

// recieve message
void callback(char* topic, byte* payload, unsigned int length) {
    char p[length + 1];
    memcpy(p, payload, length);
    p[length] = NULL;

    if (!strcmp(p, "RED"))
        RGB.color(255, 0, 0);
    else if (!strcmp(p, "GREEN"))
        RGB.color(0, 255, 0);
    else if (!strcmp(p, "BLUE"))
        RGB.color(0, 0, 255);
    else
        RGB.color(255, 255, 255);
    delay(1000);
}


void setup() {
    RGB.control(true);

    // connect to the server(unique id by Time.now())
    client.connect("sparkclient_" + String(Time.now()));

    // publish/subscribe
    if (client.isConnected()) {
        client.publish("outTopic/message","hello world");
        client.subscribe("inTopic/message");
    }
}

void loop() {
    if (client.isConnected())
        client.loop();
}
```

FAQ

Can't connect/publish/subscribe to the MQTT server?

- Check your MQTT server and port(default 1883) is really working with the mosquitto_pub/sub command. And maybe your MQTT server can't connect from Internet because of firewall. Check your network environments.
- Check your subscribe/publish topic name is really matched.
- Perhaps device firmware network stack is failed. check your firmware version and bugs.
- If you use MQTT-TLS, check your RooT CA pem file, client key, certifications is okay or not.
- Several MQTT server will disconnect to the 1st connection when you use the same user_id. When the application call the connect method, use different user_id in every devices in connect method's 2nd argument. Use MAC address as a user_id will be better.

```
// device.1
client.connect("spark-client", "user_1", "password1");
// other devices...
client.connect("spark-client", "user_others", "password1");
```

I want to change MQTT keep alive timeout.

MQTT keepalive timeout is defined "MQTT_DEFAULT_KEEPALIVE 15"(15 sec) in header file. You can change the keepalive timeout in constructor.

```
MQTT client("server_name", 1883, callback); // default: send keepalive packet to MQTT server i
MQTT client("server_name", 1883, 30, callback); // keepliave timeout is 30sec.
```

Want to use over the 255 message size.

In this library, max MQTT message size is defined "MQTT_MAX_PACKET_SIZE 255" in header file. But If you want to use over 255bytes, use the constructor 4th argument.

```
MQTT client("server_name", 1883, callback); // default 255bytes
MQTT client("server_name", 1883, MQTT_DEFAULT_KEEPALIVE, callback, 512); // max 512bytes
```

Can I use on old firmware?

No, use default latest firmware. I test this library on default latest firmware or latest pre-release version. If you really want to use old firmware(I think don't need that case), maybe it can't work well and it is out of my assumption.

Bug or Problem?

First of all, check the `Particle community site. But still your problem will not clear, please send a bug-fixed diff and Pull request or problem details to issue. Pull Request If you have a bug or feature, please send a pull request. Thanks for all developer's pull request!`

Chapter 4

Particle Photon code

The Particle Photon subsystem software named 2020_photon_code: The entire '2020_photon_code' folder is a Visual Studio project that uses the Particle Workbench and dependencies to program the Photons (remotely).

Welcome to the project!

`/src` folder:

This is the source folder that contains the firmware files for the project. It should *not* be renamed. Anything that is in this folder when you compile your project will be sent to the Particle compile service and compiled into a firmware binary for the Particle device that you have targeted. The project is set up for Photon v2.0.1.

The main files are included in the `src` folder. The dependencies are specified in the `project.properties` file referenced below.

`.ino` file:

This file is the firmware that will run as the primary application on the Particle device. It contains a `setup()` and `loop()` function, and is written in C++.

`project.properties` file:

This is the file that specifies the name and version number of the libraries that the project depends on. Dependencies are added automatically to the `project.properties` file when you add a library to a project using the `particle library add` command in the CLI or add a library in the Desktop IDE.

Adding additional files to the project

Projects with multiple sources

If you would like add additional files to your application, they should be added to the `/src` folder. All files in the `/src` folder will be sent to the Particle Cloud to produce a compiled binary.

Projects with external libraries

If the project includes a library that has not been registered in the Particle libraries system, you should create a new folder named `/lib/<libraryname>/src` under `/<project dir>` and add the `.h`, `.cpp` & `library.properties` files for your library there.

Compiling the project

When you're ready to compile the project, make sure you have the correct Particle device target selected and run `particle compile <platform>` in the CLI or click the Compile button in the Desktop IDE. The following files in the project folder will be sent to the compile service:

- Everything in the `/src` folder, including your `.ino` application file
- The `project.properties` file for your project
- Any libraries stored under `lib/<libraryname>/src`

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Namespace Documentation

9.1 JsonParserGeneratorRK Namespace Reference

Classes

- struct jsmn_parser

    JSON parser.
- struct jsmntok_t

    JSON token description.

Enumerations

- enum jsmntype_t {
  JSMN_UNDEFINED = 0, JSMN_OBJECT = 1, JSMN_ARRAY = 2, JSMN_STRING = 3,
  JSMN_PRIMITIVE = 4 }

    JSON type identifier (object, array, string, primitive)
- enum jsmnerr { JSMN_ERROR_NOMEM = -1, JSMN_ERROR_INVAL = -2, JSMN_ERROR_PART = -3 }

    JSMN error codes.

Functions

- void jsmn_init (jsmn_parser ∗parser)

    Create JSON parser over an array of tokens.
- int jsmn_parse (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, unsigned int num_tokens)

    Run JSON parser.
- static jsmntok_t ∗ jsmn_alloc_token (jsmn_parser ∗parser, jsmntok_t ∗tokens, size_t num_tokens)
- static void jsmn_fill_token (jsmntok_t ∗token, jsmntype_t type, int start, int end)
- static int jsmn_parse_primitive (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, size_t num_tokens)
- static int jsmn_parse_string (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, size_t num↩ _tokens)

9.1.1 Enumeration Type Documentation

9.1.1.1 jsmnerr

```
enum JsonParserGeneratorRK::jsmnerr
```

JSMN error codes.

Enumerator

JSMN_ERROR_NOMEM	Not enough tokens were provided.
JSMN_ERROR_INVAL	Invalid character inside JSON string.
JSMN_ERROR_PART	The string is not a full JSON packet, more bytes expected.

Definition at line 30 of file JsonParserGeneratorRK.h.

9.1.1.2 jsmntype_t

```
enum JsonParserGeneratorRK::jsmntype_t
```

JSON type identifier (object, array, string, primitive)

Enumerator

JSMN_UNDEFINED	undefined JSON type
JSMN_OBJECT	JSON object.
JSMN_ARRAY	JSON array.
JSMN_STRING	JSON string.
JSMN_PRIMITIVE	JSON primitive (number, true, false, or null)

Definition at line 19 of file JsonParserGeneratorRK.h.

9.1.2 Function Documentation

9.1.2.1 jsmn_alloc_token()

```
static jsmntok_t* JsonParserGeneratorRK::jsmn_alloc_token (
            jsmn_parser * parser,
            jsmntok_t * tokens,
            size_t num_tokens )  [static]
```

Allocates a fresh unused token from the token pull.

Definition at line 1102 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParserGeneratorRK::jsmntok_t::size, JsonParser←GeneratorRK::jsmntok_t::start, and JsonParserGeneratorRK::jsmn_parser::toknext.

Referenced by jsmn_parse(), jsmn_parse_primitive(), and jsmn_parse_string().

9.1.2.2 jsmn_fill_token()

```
static void JsonParserGeneratorRK::jsmn_fill_token (
             jsmntok_t * token,
             jsmntype_t type,
             int start,
             int end )  [static]
```

Fills token type and boundaries.

Definition at line 1120 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParserGeneratorRK::jsmntok_t::size, JsonParser↩ GeneratorRK::jsmntok_t::start, and JsonParserGeneratorRK::jsmntok_t::type.

Referenced by jsmn_parse_primitive(), and jsmn_parse_string().

9.1.2.3 jsmn_init()

```
void JsonParserGeneratorRK::jsmn_init (
             jsmn_parser * parser )
```

Create JSON parser over an array of tokens.

Creates a new parser based over a given buffer with an array of tokens available.

Definition at line 1405 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmn_parser::pos, JsonParserGeneratorRK::jsmn_parser::toknext, and JsonParserGeneratorRK::jsmn_parser::toksuper.

Referenced by JsonParser::parse().

9.1.2.4 jsmn_parse()

```
int JsonParserGeneratorRK::jsmn_parse (
             jsmn_parser * parser,
             const char * js,
             size_t len,
             jsmntok_t * tokens,
             unsigned int num_tokens )
```

Run JSON parser.

It parses a JSON data string into and array of tokens, each describing a single JSON object.

Parse JSON string and fill tokens.

Definition at line 1247 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, jsmn_alloc_token(), JSMN_ARRAY, JSMN_ERROR_INVAL, JSMN_ERROR_NOMEM, JSMN_ERROR_PART, JSMN_OBJECT, jsmn_parse_primitive(), jsmn_parse_string(), JsonParserGeneratorRK::jsmn_parser::pos, JsonParserGeneratorRK::jsmntok_t::size, JsonParserGenerator↩ RK::jsmntok_t::start, JsonParserGeneratorRK::jsmn_parser::toknext, JsonParserGeneratorRK::jsmn_parser↩ ::toksuper, and JsonParserGeneratorRK::jsmntok_t::type.

Referenced by JsonParser::parse().

9.1.2.5 jsmn_parse_primitive()

```
static int JsonParserGeneratorRK::jsmn_parse_primitive (
            jsmn_parser * parser,
            const char * js,
            size_t len,
            jsmntok_t * tokens,
            size_t num_tokens )  [static]
```

Fills next available token with JSON primitive.

Definition at line 1131 of file JsonParserGeneratorRK.cpp.

References jsmn_alloc_token(), JSMN_ERROR_INVAL, JSMN_ERROR_NOMEM, jsmn_fill_token(), JSMN_PR↩ IMITIVE, and JsonParserGeneratorRK::jsmn_parser::pos.

Referenced by jsmn_parse().

9.1.2.6 jsmn_parse_string()

```
static int JsonParserGeneratorRK::jsmn_parse_string (
            jsmn_parser * parser,
            const char * js,
            size_t len,
            jsmntok_t * tokens,
            size_t num_tokens )  [static]
```

Fills next token with JSON string.

Definition at line 1180 of file JsonParserGeneratorRK.cpp.

References jsmn_alloc_token(), JSMN_ERROR_INVAL, JSMN_ERROR_NOMEM, JSMN_ERROR_PART, jsmn↩ _fill_token(), JSMN_STRING, and JsonParserGeneratorRK::jsmn_parser::pos.

Referenced by jsmn_parse().

Chapter 10

Class Documentation

10.1 JsonParserGeneratorRK::jsmn_parser Struct Reference

JSON parser.

```
#include <JsonParserGeneratorRK.h>
```

Public Attributes

- unsigned int pos

  offset in the JSON string
- unsigned int toknext

  next token to allocate
- int toksuper

  superior token node, e.g parent object or array

10.1.1 Detailed Description

JSON parser.

Contains an array of token blocks available. Also stores the string being parsed now and current position in that string.

Definition at line 55 of file JsonParserGeneratorRK.h.

10.1.2 Member Data Documentation

10.1.2.1 pos

```
unsigned int JsonParserGeneratorRK::jsmn_parser::pos
```

offset in the JSON string

Definition at line 56 of file JsonParserGeneratorRK.h.

Referenced by JsonParserGeneratorRK::jsmn_init(), JsonParserGeneratorRK::jsmn_parse(), JsonParser←GeneratorRK::jsmn_parse_primitive(), and JsonParserGeneratorRK::jsmn_parse_string().

10.1.2.2 toknext

```
unsigned int JsonParserGeneratorRK::jsmn_parser::toknext
```

next token to allocate

Definition at line 57 of file JsonParserGeneratorRK.h.

Referenced by JsonParserGeneratorRK::jsmn_alloc_token(), JsonParserGeneratorRK::jsmn_init(), and Json←ParserGeneratorRK::jsmn_parse().

10.1.2.3 toksuper

```
int JsonParserGeneratorRK::jsmn_parser::toksuper
```

superior token node, e.g parent object or array

Definition at line 58 of file JsonParserGeneratorRK.h.

Referenced by JsonParserGeneratorRK::jsmn_init(), and JsonParserGeneratorRK::jsmn_parse().

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

- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.2 JsonParserGeneratorRK::jsmntok_t Struct Reference

JSON token description.

```
#include <JsonParserGeneratorRK.h>
```

Public Attributes

- jsmntype_t type

    type (object, array, string etc.)
- int start

    start position in JSON data string
- int end

    end position in JSON data string
- int size

    size

10.2.1 Detailed Description

JSON token description.

Definition at line 39 of file JsonParserGeneratorRK.h.

10.2.2 Member Data Documentation

10.2.2.1 end

```
int JsonParserGeneratorRK::jsmntok_t::end
```

end position in JSON data string

Definition at line 42 of file JsonParserGeneratorRK.h.

Referenced by JsonParser::copyTokenValue(), JsonModifier::findRightComma(), JsonParser::getArraySize(), JsonParser::getKeyValueTokenByIndex(), JsonParser::getTokenByIndex(), JsonParser::getTokenJsonString(), JsonParser::getTokenValue(), JsonParser::getValueTokenByIndex(), JsonParserGeneratorRK::jsmn_alloc_token(), JsonParserGeneratorRK::jsmn_fill_token(), JsonParserGeneratorRK::jsmn_parse(), printJsonInner(), printToken(), JsonModifier::removeArrayIndex(), JsonModifier::removeKeyValue(), JsonParser::skipObject(), JsonModifier↩ ::startAppend(), JsonModifier::startModify(), and JsonModifier::tokenWithQuotes().

10.2.2.2 size

```
int JsonParserGeneratorRK::jsmntok_t::size
```

size

Definition at line 43 of file JsonParserGeneratorRK.h.

Referenced by JsonParserGeneratorRK::jsmn_alloc_token(), JsonParserGeneratorRK::jsmn_fill_token(), Json↩ ParserGeneratorRK::jsmn_parse(), and JsonModifier::startAppend().

10.2.2.3 start

```
int JsonParserGeneratorRK::jsmntok_t::start
```

start position in JSON data string

Definition at line 41 of file JsonParserGeneratorRK.h.

Referenced by JsonParser::copyTokenValue(), JsonModifier::findLeftComma(), JsonParser::getTokenJsonString(), JsonParser::getTokenValue(), JsonParserGeneratorRK::jsmn_alloc_token(), JsonParserGeneratorRK::jsmn_fill_↩ token(), JsonParserGeneratorRK::jsmn_parse(), printJsonInner(), printToken(), JsonModifier::removeArrayIndex(), JsonModifier::removeKeyValue(), JsonModifier::startModify(), and JsonModifier::tokenWithQuotes().

10.2.2.4 type

```
jsmntype_t JsonParserGeneratorRK::jsmntok_t::type
```

type (object, array, string etc.)

Definition at line 40 of file JsonParserGeneratorRK.h.

Referenced by JsonParser::getOuterArray(), JsonParser::getOuterObject(), JsonParser::getOuterToken(), Json↩ ParserGeneratorRK::jsmn_fill_token(), JsonParserGeneratorRK::jsmn_parse(), printJsonInner(), printToken(), and JsonModifier::tokenWithQuotes().

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

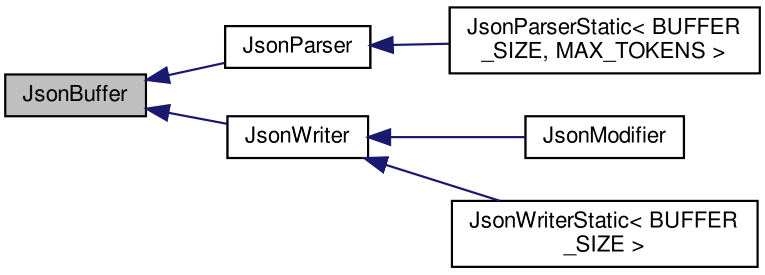
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.3 JsonBuffer Class Reference

Base class for managing a static or dynamic buffer, used by both JsonParser and JsonWriter.

```
#include <JsonParserGeneratorRK.h>
```

Inheritance diagram for JsonBuffer:


JsonBuffer
JsonParser
JsonParserStatic< BUFFER _SIZE, MAX_TOKENS >
JsonWriter
JsonModifier
JsonWriterStatic< BUFFER _SIZE >

Public Member Functions

- JsonBuffer ()

  Construct a JsonBuffer object with no external buffer specified.
- virtual ∼JsonBuffer ()

  Destructor. Destroying the object does not delete any underlying buffer!
- JsonBuffer (char ∗buffer, size_t bufferLen)

  Construct a JsonBuffer with an external buffer of a given size.
- void setBuffer (char ∗buffer, size_t bufferLen)

  Sets the buffers to the specified buffer and length.
- bool allocate (size_t len)

  Allocate the buffer using malloc/realloc.
- bool addString (const char ∗data)

  Add a c-string to the end of the buffer.
- bool addData (const char ∗data, size_t dataLen)

  Add a string to the end of the buffer.
- char ∗ getBuffer () const

  Gets a pointer to the internal buffer.
- size_t getOffset () const

  Gets the current offset for writing.
- void setOffset (size_t offset)

  swets the current offset for writing
- size_t getBufferLen () const

  Gets the current length of the buffer.
- void clear ()

  Clears the current buffer for writing.
- void nullTerminate ()

  Null terminates the buffer.

Protected Attributes

- char ∗ buffer

  The buffer to to read from or write to. This is not null-terminated.
- size_t bufferLen

  The length of the buffer in bytes,.
- size_t offset

  The read or write offset.
- bool staticBuffers

  True if the buffers were passed in and should not freed or reallocated.

10.3.1 Detailed Description

Base class for managing a static or dynamic buffer, used by both JsonParser and JsonWriter.

Definition at line 146 of file JsonParserGeneratorRK.h.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 JsonBuffer() [1/2]

```
JsonBuffer::JsonBuffer ( )
```

Construct a JsonBuffer object with no external buffer specified.

Definition at line 6 of file JsonParserGeneratorRK.cpp.

References buffer, bufferLen, offset, and staticBuffers.

Referenced by JsonParser::JsonParser(), and JsonWriter::JsonWriter().

10.3.2.2 ∼JsonBuffer()

```
JsonBuffer::∼JsonBuffer ( )  [virtual]
```

Destructor. Destroying the object does not delete any underlying buffer!

Definition at line 9 of file JsonParserGeneratorRK.cpp.

References buffer, and staticBuffers.

10.3.2.3 JsonBuffer() [2/2]

```
JsonBuffer::JsonBuffer (
             char * buffer,
             size_t bufferLen )
```

Construct a JsonBuffer with an external buffer of a given size.

Parameters

buffer	Pointer to the buffer
bufferLen	The length of the buffer

This buffer will not be deleted when the object is destructed.

Definition at line 15 of file JsonParserGeneratorRK.cpp.

References buffer, bufferLen, offset, and staticBuffers.

Referenced by JsonParser::JsonParser(), and JsonWriter::JsonWriter().

10.3.3 Member Function Documentation

10.3.3.1 addData()

```
bool JsonBuffer::addData (
            const char * data,
            size_t dataLen )
```

Add a string to the end of the buffer.

Parameters

data	Pointer to the string bytes. Does not need to be null-terminated
dataLen	Length of the data in bytes. For UTF-8, this is the number of bytes, not characters!

Definition at line 48 of file JsonParserGeneratorRK.cpp.

References allocate(), buffer, bufferLen, and offset.

Referenced by main().

10.3.3.2 addString()

```
bool JsonBuffer::addString (
            const char * data )  [inline]
```

Add a c-string to the end of the buffer.

Parameters

data	Pointer to a c-string (null terminated).

Definition at line 197 of file JsonParserGeneratorRK.h.

Referenced by main().

10.3.3.3 allocate()

```
bool JsonBuffer::allocate (
            size_t len )
```

Allocate the buffer using malloc/realloc.

Parameters

len	The length of the buffer in bytes

Returns

true if the allocation/reallocation was successful or false if there was not enough free memory.

There's also a version that takes a pointer and length to use a static buffer instead of a dynamically allocated one.

Definition at line 25 of file JsonParserGeneratorRK.cpp.

References buffer, bufferLen, and staticBuffers.

Referenced by addData(), and main().

10.3.3.4 clear()

```
void JsonBuffer::clear ( )
```

Clears the current buffer for writing.

This only sets the offset to 0, it does not clear the bytes.

Definition at line 62 of file JsonParserGeneratorRK.cpp.

References offset.

10.3.3.5 getBuffer()

```
char* JsonBuffer::getBuffer ( ) const  [inline]
```

Gets a pointer to the internal buffer.

Note: The internal buffer is not null-terminated!

Definition at line 213 of file JsonParserGeneratorRK.h.

References buffer.

Referenced by JsonModifier::findLeftComma(), JsonModifier::findRightComma(), JsonModifier::startAppend(), and JsonModifier::startModify().

10.3.3.6 getBufferLen()

```
size_t JsonBuffer::getBufferLen ( ) const  [inline]
```

Gets the current length of the buffer.

The buffer length is either the bufferLen passed to the constructor that takes a buffer and bufferLen or the length allocated using allocate(len).

Definition at line 231 of file JsonParserGeneratorRK.h.

References bufferLen.

Referenced by JsonModifier::startAppend(), and JsonModifier::startModify().

10.3.3.7 getOffset()

```
size_t JsonBuffer::getOffset ( ) const  [inline]
```

Gets the current offset for writing.

Definition at line 218 of file JsonParserGeneratorRK.h.

References offset.

Referenced by _assertJsonParserBuffer(), _assertJsonWriterBuffer(), JsonModifier::findRightComma(), Json↵ Modifier::finish(), JsonModifier::removeArrayIndex(), JsonModifier::removeKeyValue(), JsonModifier::startAppend(), and JsonModifier::startModify().

10.3.3.8 nullTerminate()

```
void JsonBuffer::nullTerminate ( )
```

Null terminates the buffer.

Definition at line 66 of file JsonParserGeneratorRK.cpp.

References buffer, bufferLen, and offset.

10.3.3.9 setBuffer()

```
void JsonBuffer::setBuffer (
            char * buffer,
            size_t bufferLen )
```

Sets the buffers to the specified buffer and length.

Parameters

buffer	Pointer to the buffer
bufferLen	The length of the buffer

This buffer will not be deleted when the object is destructed.

Definition at line 19 of file JsonParserGeneratorRK.cpp.

References buffer, bufferLen, and staticBuffers.

Referenced by JsonModifier::startAppend(), and JsonModifier::startModify().

10.3.3.10 setOffset()

```
void JsonBuffer::setOffset (
            size_t offset ) [inline]
```

swets the current offset for writing

Definition at line 223 of file JsonParserGeneratorRK.h.

References offset.

Referenced by JsonModifier::finish(), JsonModifier::removeArrayIndex(), and JsonModifier::removeKeyValue().

10.3.4 Member Data Documentation

10.3.4.1 buffer

```
char* JsonBuffer::buffer [protected]
```

The buffer to to read from or write to. This is not null-terminated.

Definition at line 246 of file JsonParserGeneratorRK.h.

Referenced by addData(), allocate(), JsonParser::copyTokenValue(), JsonWriter::finishObjectOrArray(), get↩ Buffer(), JsonParser::getTokenJsonString(), JsonParser::getTokenValue(), JsonWriter::insertChar(), JsonBuffer(), nullTerminate(), JsonParser::parse(), setBuffer(), and ∼JsonBuffer().

10.3.4.2 bufferLen

```
size_t JsonBuffer::bufferLen [protected]
```

The length of the buffer in bytes,.

Definition at line 247 of file JsonParserGeneratorRK.h.

Referenced by addData(), allocate(), JsonWriter::finishObjectOrArray(), getBufferLen(), JsonWriter::insertChar(), JsonWriter::insertString(), JsonWriter::insertvsprintf(), JsonBuffer(), nullTerminate(), and setBuffer().

10.3.4.3 offset

```
size_t JsonBuffer::offset [protected]
```

The read or write offset.

Definition at line 248 of file JsonParserGeneratorRK.h.

Referenced by addData(), clear(), JsonWriter::finishObjectOrArray(), getOffset(), JsonWriter::init(), JsonWriter↩ ::insertChar(), JsonWriter::insertString(), JsonWriter::insertvsprintf(), JsonBuffer(), nullTerminate(), JsonParser↩ ::parse(), and setOffset().

10.3.4.4 staticBuffers

`bool JsonBuffer::staticBuffers [protected]`

True if the buffers were passed in and should not freed or reallocated.

Definition at line 249 of file JsonParserGeneratorRK.h.

Referenced by allocate(), JsonParser::allocateTokens(), JsonBuffer(), JsonParser::parse(), setBuffer(), ∼Json↵ Buffer(), and JsonParser::∼JsonParser().

The documentation for this class was generated from the following files:

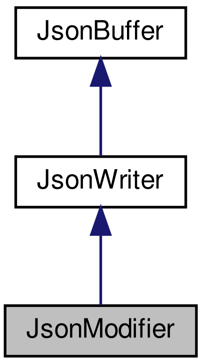
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.4 JsonModifier Class Reference

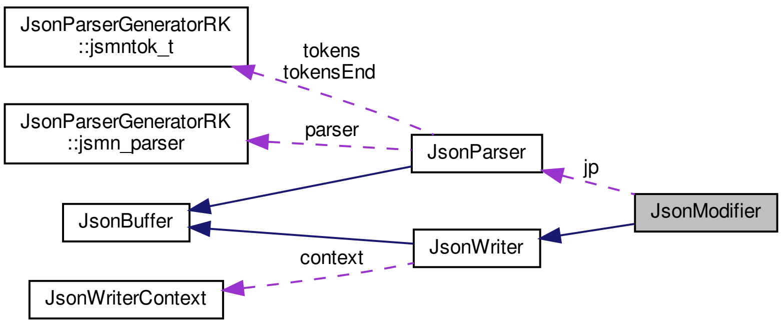
Class for modifying a JSON object in place, without needing to make a copy of it.

`#include <JsonParserGeneratorRK.h>`

Inheritance diagram for JsonModifier:



Collaboration diagram for JsonModifier:



Public Member Functions

- JsonModifier (JsonParser &jp)
- virtual ∼JsonModifier ()
- template<class T >
  void insertOrUpdateKeyValue (const JsonParserGeneratorRK::jsmntok_t ∗container, const char ∗key, T value)

  Inserts or updates a key/value pair into an object.
- template<class T >
  void appendArrayValue (const JsonParserGeneratorRK::jsmntok_t ∗arrayToken, T value)

  Appends a value to an array.
- bool removeKeyValue (const JsonParserGeneratorRK::jsmntok_t ∗container, const char ∗key)

  Removes a key and value from an object.
- bool removeArrayIndex (const JsonParserGeneratorRK::jsmntok_t ∗container, size_t index)

  Removes an entry from an array.
- bool startModify (const JsonParserGeneratorRK::jsmntok_t ∗token)

  Low level function to modify a token in place.
- bool startAppend (const JsonParserGeneratorRK::jsmntok_t ∗arrayOrObjectToken)

  Low level function to append to an object or array.
- void finish ()

  Finish modifying the object.
- JsonParserGeneratorRK::jsmntok_t tokenWithQuotes (const JsonParserGeneratorRK::jsmntok_t ∗tok) const

  Return a copy of tok, but moving so start and end include the double quotes for strings.
- int findLeftComma (const JsonParserGeneratorRK::jsmntok_t ∗tok) const

  Find the offset of the comma to the left of the token, or -1 if there isn't one.
- int findRightComma (const JsonParserGeneratorRK::jsmntok_t ∗tok) const

  Find the offset of the comma to the left of the token, or -1 if there isn't one.

Protected Attributes

- JsonParser & jp

  The JsonParser object passed to the constructor.
- int start = -1

  Start offset in the buffer. Set to -1 when startModify() or startAppend() is not in progress.
- int origAfter = 0

  Number of bytes after the insertion position, saved at saveLoc when start is in progress.
- int saveLoc = 0

  Location where data is temporarily saved until finish() is called.

Additional Inherited Members

10.4.1 Detailed Description

Class for modifying a JSON object in place, without needing to make a copy of it.

Make sure the underlying JsonParser is big enough to hold the modified object. If you use JsonParserStatic<> make sure you have enough bytes and tokens.

The most commonly used method is insertOrUpdateKeyValue(). This inserts or updates a key in an array. Another is appendArrayValue() which appends a value to an array. Both methods are templated so you can use them with any valid type supported by insertValue() in JsonWriter: bool, int, float, double, const char ∗.

This class is a subclass of JsonWriter, so you can also use the low-level functions and JsonWriter methods to do unusual object manipulations.

You can also use removeKeyValue() and removeArrayIndex() to remove keys or array entries.

Definition at line 1323 of file JsonParserGeneratorRK.h.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 JsonModifier()

```
JsonModifier::JsonModifier (
            JsonParser & jp )
```

Definition at line 881 of file JsonParserGeneratorRK.cpp.

References jp.

10.4.2.2 ∼JsonModifier()

```
JsonModifier::∼JsonModifier ( )  [virtual]
```

Definition at line 885 of file JsonParserGeneratorRK.cpp.

10.4.3 Member Function Documentation

10.4.3.1 appendArrayValue()

```
template<class T >
void JsonModifier::appendArrayValue (
            const JsonParserGeneratorRK::jsmntok_t * arrayToken,
            T value )  [inline]
```

Appends a value to an array.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char ∗.

To modify the outermost array, use jp.getOuterArray() for the arrayToken. You can also modify arrays in an object using getValueTokenByKey().

Note: This method call jp.parse() so any jsmntok_t may be changed by this method. If you've fetched one, such as by using getValueTokenByKey() be sure to fetch it again to be safe.

Definition at line 1363 of file JsonParserGeneratorRK.h.

References finish(), and startAppend().

10.4.3.2 findLeftComma()

```
int JsonModifier::findLeftComma (
             const JsonParserGeneratorRK::jsmntok_t * tok ) const
```

Find the offset of the comma to the left of the token, or -1 if there isn't one.

Used internally, you probably won't need to use this.

Definition at line 1058 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::getBuffer(), jp, JsonParserGeneratorRK::jsmntok_t::start, and tokenWithQuotes().

Referenced by removeArrayIndex(), and removeKeyValue().

10.4.3.3 findRightComma()

```
int JsonModifier::findRightComma (
             const JsonParserGeneratorRK::jsmntok_t * tok ) const
```

Find the offset of the comma to the left of the token, or -1 if there isn't one.

Used internally, you probably won't need to use this.

Definition at line 1077 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonBuffer::getBuffer(), JsonBuffer::getOffset(), jp, and tokenWithQuotes().

Referenced by removeArrayIndex(), and removeKeyValue().

10.4.3.4 finish()

```
void JsonModifier::finish ( )
```

Finish modifying the object.

Finish must be called after startModify or startAppend otherwise the object will be corrupted.

Note: This method call jp.parse() so any jsmntok_t may be changed by this method. If you've fetched one, such as by using getValueTokenByKey() be sure to fetch it again to be safe.

The high level function like insertOrUpdateKeyValue, appendArrayValue, removeKeyValue, and removeArrayIndex internally call finish so you should not call it again with those methods.

Definition at line 1033 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::getOffset(), jp, origAfter, JsonParser::parse(), JsonBuffer::setOffset(), and start.

Referenced by appendArrayValue(), insertOrUpdateKeyValue(), and main().

10.4.3.5 insertOrUpdateKeyValue()

```
template<class T >
void JsonModifier::insertOrUpdateKeyValue (
             const JsonParserGeneratorRK::jsmntok_t * container,
             const char * key,
             T value )  [inline]
```

Inserts or updates a key/value pair into an object.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char ∗.

To modify the outermost object, use jp.getOuterObject() for the container.

Note: This method call jp.parse() so any jsmntok_t may be changed by this method. If you've fetched one, such as by using getValueTokenByKey() be sure to fetch it again to be safe.

Definition at line 1340 of file JsonParserGeneratorRK.h.

References finish(), removeKeyValue(), and startAppend().

10.4.3.6 removeArrayIndex()

```
bool JsonModifier::removeArrayIndex (
             const JsonParserGeneratorRK::jsmntok_t * container,
             size_t index )
```

Removes an entry from an array.

Note: This method call jp.parse() so any jsmntok_t may be changed by this method. If you've fetched one, such as by using getValueTokenByKey() be sure to fetch it again to be safe.

Definition at line 944 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, findLeftComma(), findRightComma(), JsonBuffer::get↩ Offset(), JsonParser::getTokenByIndex(), jp, origAfter, JsonParser::parse(), JsonBuffer::setOffset(), JsonParser↩ GeneratorRK::jsmntok_t::start, and tokenWithQuotes().

Referenced by main().

10.4.3.7 removeKeyValue()

```
bool JsonModifier::removeKeyValue (
             const JsonParserGeneratorRK::jsmntok_t * container,
             const char * key )
```

Removes a key and value from an object.

Note: This method call jp.parse() so any jsmntok_t may be changed by this method. If you've fetched one, such as by using getValueTokenByKey() be sure to fetch it again to be safe.

Definition at line 890 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, findLeftComma(), findRightComma(), JsonBuffer::get↩ Offset(), JsonParser::getValueTokenByKey(), jp, origAfter, JsonParser::parse(), JsonBuffer::setOffset(), Json↩ ParserGeneratorRK::jsmntok_t::start, and tokenWithQuotes().

Referenced by insertOrUpdateKeyValue().

10.4.3.8 startAppend()

```
bool JsonModifier::startAppend (
            const JsonParserGeneratorRK::jsmntok_t * arrayOrObjectToken )
```

Low level function to append to an object or array.

Parameters

arrayOrObjectToken	the jsmntok_t to append to. This must be an object or array token.

You must call finish() after modification is done to restore the object to a valid state.

Definition at line 1009 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonBuffer::getBuffer(), JsonBuffer::getBufferLen(), Json↩Buffer::getOffset(), JsonWriter::init(), jp, origAfter, saveLoc, JsonBuffer::setBuffer(), JsonWriter::setIsFirst(), Json↩ParserGeneratorRK::jsmntok_t::size, and start.

Referenced by appendArrayValue(), insertOrUpdateKeyValue(), and main().

10.4.3.9 startModify()

```
bool JsonModifier::startModify (
            const JsonParserGeneratorRK::jsmntok_t * token )
```

Low level function to modify a token in place.

Parameters

token	the jsmntok_t to modify

You must call finish() after modification is done to restore the object to a valid state!

Note: insertOrUpdateKeyValue() does not use this. Instead it removes then appends the new value. The reason is that startModify does not work if you change the type of the data to or from a string. This is tricky to deal with correctly, so it's easier to just remove and add the item again.

Definition at line 988 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonBuffer::getBuffer(), JsonBuffer::getBufferLen(), Json↩Buffer::getOffset(), JsonWriter::init(), jp, origAfter, saveLoc, JsonBuffer::setBuffer(), JsonParserGeneratorRK↩::jsmntok_t::start, and start.

Referenced by main().

10.4.3.10 tokenWithQuotes()

```
JsonParserGeneratorRK::jsmntok_t JsonModifier::tokenWithQuotes (
            const JsonParserGeneratorRK::jsmntok_t * tok ) const
```

Return a copy of tok, but moving so start and end include the double quotes for strings.

Used internally, you probably won't need to use this.

Definition at line 1048 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParserGeneratorRK::JSMN_STRING, JsonParser↩ GeneratorRK::jsmntok_t::start, and JsonParserGeneratorRK::jsmntok_t::type.

Referenced by findLeftComma(), findRightComma(), removeArrayIndex(), and removeKeyValue().

10.4.4 Member Data Documentation

10.4.4.1 jp

```
JsonParser& JsonModifier::jp  [protected]
```

The JsonParser object passed to the constructor.

Definition at line 1447 of file JsonParserGeneratorRK.h.

Referenced by findLeftComma(), findRightComma(), finish(), JsonModifier(), removeArrayIndex(), removeKey↩ Value(), startAppend(), and startModify().

10.4.4.2 origAfter

```
int JsonModifier::origAfter = 0  [protected]
```

Number of bytes after the insertion position, saved at saveLoc when start is in progress.

Definition at line 1449 of file JsonParserGeneratorRK.h.

Referenced by finish(), removeArrayIndex(), removeKeyValue(), startAppend(), and startModify().

10.4.4.3 saveLoc

```
int JsonModifier::saveLoc = 0  [protected]
```

Location where data is temporarily saved until finish() is called.

Definition at line 1450 of file JsonParserGeneratorRK.h.

Referenced by startAppend(), and startModify().

10.4.4.4 start

```
int JsonModifier::start = -1  [protected]
```

Start offset in the buffer. Set to -1 when startModify() or startAppend() is not in progress.

Definition at line 1448 of file JsonParserGeneratorRK.h.

Referenced by finish(), startAppend(), and startModify().

The documentation for this class was generated from the following files:

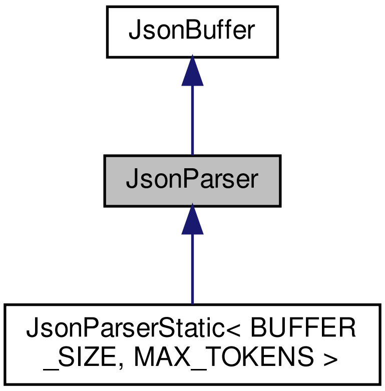
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.5 JsonParser Class Reference

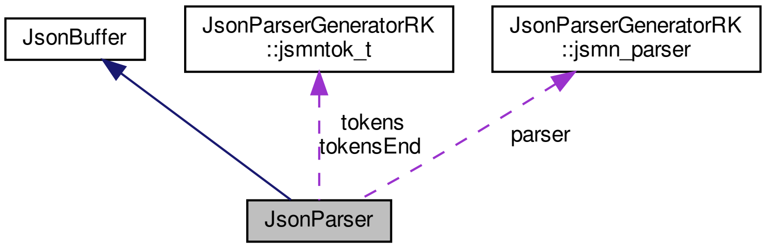
API to the JsonParser.

```
#include <JsonParserGeneratorRK.h>
```

Inheritance diagram for JsonParser:



Collaboration diagram for JsonParser:


JsonBuffer
JsonParserGeneratorRK ::jsmntok_t
JsonParserGeneratorRK ::jsmn_parser
tokens
tokensEnd
parser
JsonParser

Public Member Functions

- JsonParser ()

  Construct a parser object.
- virtual ∼JsonParser ()

  Destroy a parser object.
- JsonParser (char ∗buffer, size_t bufferLen, JsonParserGeneratorRK::jsmntok_t ∗tokens, size_t maxTokens)

  Static buffers constructor.
- bool allocateTokens (size_t maxTokens)

  Preallocates a specific number of tokens.
- bool parse ()

  Parses the data you have added using addData() or addString().
- JsonReference getReference () const

  Get a JsonReference object. This is used for fluent-style access to the data.
- const JsonParserGeneratorRK::jsmntok_t ∗ getOuterObject () const

  Gets the outer JSON object token.
- const JsonParserGeneratorRK::jsmntok_t ∗ getOuterArray () const

  Gets the outer JSON array token.
- const JsonParserGeneratorRK::jsmntok_t ∗ getOuterToken () const

  Gets the outer JSON object or array token.
- size_t getArraySize (const JsonParserGeneratorRK::jsmntok_t ∗arrayContainer) const

  Given a token for an JSON array in arrayContainer, gets the number of elements in the array.
- template<class T >
  bool getValueByKey (const JsonParserGeneratorRK::jsmntok_t ∗container, const char ∗name, T &result) const

  Given an object token in container, gets the value with the specified key name.
- template<class T >
  bool getOuterValueByKey (const char ∗name, T &result) const

  Gets the value with the specified key name out of the outer object.
- template<class T >
  bool getKeyValueByIndex (const JsonParserGeneratorRK::jsmntok_t ∗container, size_t index, String &key, T &result) const

  Gets the key/value pair of an object by index.
- template<class T >
  bool getOuterKeyValueByIndex (size_t index, String &key, T &result) const

  Gets the key/value pair of the outer object by index (0 = first, 1 = second, ...)
- template<class T >
  bool getValueByIndex (const JsonParserGeneratorRK::jsmntok_t ∗arrayContainer, size_t index, T &result) const

  Given an array token in arrayContainer, gets the value with the specified index.
- template<class T >
  bool getValueByColRow (const JsonParserGeneratorRK::jsmntok_t ∗arrayContainer, size_t col, size_t row, T &result) const

  This method is used to extract data from a 2-dimensional JSON array, an array of arrays of values.
- bool getValueTokenByKey (const JsonParserGeneratorRK::jsmntok_t ∗container, const char ∗key, const JsonParserGeneratorRK::jsmntok_t ∗&value) const

  Given an object token in container, gets the token value with the specified key name.
- bool getValueTokenByIndex (const JsonParserGeneratorRK::jsmntok_t ∗container, size_t desiredIndex, const JsonParserGeneratorRK::jsmntok_t ∗&value) const

  Given an array token in container, gets the token value with the specified index.
- bool getValueTokenByColRow (const JsonParserGeneratorRK::jsmntok_t ∗container, size_t col, size_t row, const JsonParserGeneratorRK::jsmntok_t ∗&value) const

  This method is used to extract data from a 2-dimensional JSON array, an array of arrays of values.

- const JsonParserGeneratorRK::jsmntok_t ∗ getTokenByIndex (const JsonParserGeneratorRK::jsmntok_↩ t ∗container, size_t desiredIndex) const

    Given a containing object, finds the nth token in the object. Internal use only.
- bool getKeyValueTokenByIndex (const JsonParserGeneratorRK::jsmntok_t ∗container, const JsonParser↩ GeneratorRK::jsmntok_t ∗&key, const JsonParserGeneratorRK::jsmntok_t ∗&value, size_t index) const

    Given a JSON object in container, gets the key/value pair specified by index. Internal use only.
- bool skipObject (const JsonParserGeneratorRK::jsmntok_t ∗container, const JsonParserGeneratorRK↩ ::jsmntok_t ∗&obj) const

    Used internally to skip over the token in obj.
- void copyTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, char ∗dst, size_t dstLen) const

    Copies the value of the token into a buffer, making it a null-terminated cstring.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, bool &result) const

    Gets a bool (boolean) value.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, int &result) const

    Gets an integer value.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, unsigned long &result) const

    Gets an unsigned long value.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, float &result) const

    Gets a float (single precision floating point) value.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, double &result) const

    Gets a double (double precision floating point) value.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, String &result) const

    Gets a String value into a Wiring String object.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, char ∗str, size_t &strLen) const

    Gets a string as a c-string into the specified buffer.
- bool getTokenValue (const JsonParserGeneratorRK::jsmntok_t ∗token, JsonParserString &str) const

    Gets a string as a JsonParserString object.
- bool getTokenJsonString (const JsonParserGeneratorRK::jsmntok_t ∗token, String &result) const

    Converts a token (object, array, string, or primitive) back into JSON in a Wiring String.
- bool getTokenJsonString (const JsonParserGeneratorRK::jsmntok_t ∗token, char ∗str, size_t &strLen) const

    Converts a token (object, array, string, or primitive) back into JSON in a buffer.
- bool getTokenJsonString (const JsonParserGeneratorRK::jsmntok_t ∗token, JsonParserString &str) const

    Gets a token as a JSON string.
- JsonParserGeneratorRK::jsmntok_t ∗ getTokens ()

    Used internally in the test suite for printing the token list.
- JsonParserGeneratorRK::jsmntok_t ∗ getTokensEnd ()

    Used internally in the test suite for printing the token list.
- size_t getMaxTokens () const

    Used internally in the test suite for printing the token list.

Static Public Member Functions

- static void appendUtf8 (uint16_t unicode, JsonParserString &str)

    Given a Unicode UTF-16 code point, converts it to UTF-8 and appends it to str.

Protected Attributes

- JsonParserGeneratorRK::jsmntok_t ∗ tokens

    Array of tokens after parsing.
- JsonParserGeneratorRK::jsmntok_t ∗ tokensEnd

    Pointer into tokens, points after last used token.
- size_t maxTokens

    Number of tokens that can be stored in tokens.
- JsonParserGeneratorRK::jsmn_parser parser

    The JSMN parser object.

Friends

- class JsonModifier

10.5.1 Detailed Description

API to the JsonParser.

This is a memory-efficient JSON parser based on jsmn. It only keeps one copy of the data in raw format and an array of tokens. You make calls to read values out.

Definition at line 262 of file JsonParserGeneratorRK.h.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 JsonParser() [1/2]

```
JsonParser::JsonParser ( )
```

Construct a parser object.

This version dynamically allocates the buffer and token storage. If you want to minimize memory allocations you can pass in a static buffer and array of tokens to use instead.

Definition at line 80 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::JsonBuffer(), maxTokens, tokens, and tokensEnd.

10.5.2.2 ∼JsonParser()

```
JsonParser::∼JsonParser ( ) [virtual]
```

Destroy a parser object.

If the buffer was allocated dynamically it will be deleted. If you passed in a static buffer the static buffer is not deleted.

Definition at line 89 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::staticBuffers, and tokens.

10.5.2.3 JsonParser() [2/2]

```
JsonParser::JsonParser (
            char * buffer,
            size_t bufferLen,
            JsonParserGeneratorRK::jsmntok_t * tokens,
            size_t maxTokens )
```

Static buffers constructor.

Definition at line 83 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::JsonBuffer(), maxTokens, and tokens.

10.5.3 Member Function Documentation

10.5.3.1 allocateTokens()

```
bool JsonParser::allocateTokens (
            size_t maxTokens )
```

Preallocates a specific number of tokens.

Optional: You should set this larger than the expected number of tokens for efficiency, but if you are not using the static allocator it will resize the token storage space if it's too small.

Definition at line 95 of file JsonParserGeneratorRK.cpp.

References maxTokens, JsonBuffer::staticBuffers, and tokens.

10.5.3.2 appendUtf8()

```
void JsonParser::appendUtf8 (
             uint16_t unicode,
             JsonParserString & str )  [static]
```

Given a Unicode UTF-16 code point, converts it to UTF-8 and appends it to str.

Definition at line 509 of file JsonParserGeneratorRK.cpp.

References JsonParserString::append().

10.5.3.3 copyTokenValue()

```
void JsonParser::copyTokenValue (
             const JsonParserGeneratorRK::jsmntok_t * token,
             char * dst,
             size_t dstLen ) const
```

Copies the value of the token into a buffer, making it a null-terminated cstring.

If the string is longer than dstLen - 1 bytes, it will be truncated and the result will still be a valid cstring.

This is used internally because the token data is not null-terminated, and doing things like sscanf or strtoul on it can read past the end of the buffer. This assures that only null-terminated data is passed to these functions.

Definition at line 327 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::buffer, JsonParserGeneratorRK::jsmntok_t::end, and JsonParserGeneratorRK::jsmntok_↩
t::start.

Referenced by getTokenValue().

10.5.3.4 getArraySize()

```
size_t JsonParser::getArraySize (
             const JsonParserGeneratorRK::jsmntok_t * arrayContainer ) const
```

Given a token for an JSON array in arrayContainer, gets the number of elements in the array.

0 = no elements, 1 = one element, ...

The index values for getValueByIndex(), etc. are 0-based, so the last index you pass in is less than getArraySize().

Definition at line 315 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, skipObject(), and tokensEnd.

Referenced by JsonReference::size().

10.5.3.5 getKeyValueByIndex()

```
template<class T >
bool JsonParser::getKeyValueByIndex (
             const JsonParserGeneratorRK::jsmntok_t * container,
             size_t index,
             String & key,
             T & result ) const  [inline]
```

Gets the key/value pair of an object by index.

Parameters

container	The object to look in (see getOuterKeyValueByIndex if you want to the outermost object you parsed)
index	0 = first, 1 = second, ...
key	Filled in with the name of the key
result	Filled in with the value. The value can be of type: bool, int, unsigned long, float, double, String, or (char ∗, size_t&).

Returns

true if the call succeeded or false if it failed.

Normally you get a value in an object by its key, but if you want to iterate all of the keys you can use this method. Call it until it returns false.

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within an object, use getValueTokenByKey() instead.

Definition at line 425 of file JsonParserGeneratorRK.h.

References getKeyValueTokenByIndex().

10.5.3.6 getKeyValueTokenByIndex()

```
bool JsonParser::getKeyValueTokenByIndex (
            const JsonParserGeneratorRK::jsmntok_t * container,
            const JsonParserGeneratorRK::jsmntok_t *& key,
            const JsonParserGeneratorRK::jsmntok_t *& value,
            size_t index ) const
```

Given a JSON object in container, gets the key/value pair specified by index. Internal use only.

Parameters

container	The array token to look in.
key	Filled in with the key token for nth key value pair.
value	Filled in with the value token for then nth key value pair.
index	The index to retrieve (0 = first, 1 = second, ...).

This is a low-level function; you will typically use getValueByIndex() or getValueByKey() instead.

Definition at line 250 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, skipObject(), and tokensEnd.

Referenced by getKeyValueByIndex(), getValueTokenByKey(), main(), and printJsonInner().

10.5.3.7 getMaxTokens()

```
size_t JsonParser::getMaxTokens ( ) const  [inline]
```

Used internally in the test suite for printing the token list.

Definition at line 743 of file JsonParserGeneratorRK.h.

References maxTokens.

10.5.3.8 getOuterArray()

```
const JsonParserGeneratorRK::jsmntok_t * JsonParser::getOuterArray ( ) const
```

Gets the outer JSON array token.

Sometimes the JSON will contain an array of values (or objects) instead of starting with an object. This gets the outermost array.

A token (JsonParserGeneratorRK::jsmntok_t) identifies a particular piece of data in the JSON data, such as an object, array, or element within an object or array, such as a string, integer, boolean, etc..

Definition at line 192 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::JSMN_ARRAY, tokens, tokensEnd, and JsonParserGeneratorRK::jsmntok↩_t::type.

10.5.3.9 getOuterKeyValueByIndex()

```
template<class T >
bool JsonParser::getOuterKeyValueByIndex (
             size_t index,
             String & key,
             T & result ) const  [inline]
```

Gets the key/value pair of the outer object by index (0 = first, 1 = second, ...)

Normally you get a value in an object by its key, but if you want to iterate all of the keys you can use this method.

Parameters

index	0 = first, 1 = second, ...
key	Filled in with the name of the key
result	Filled in with the value. The value can be of type: bool, int, unsigned long, float, double, String, or (char ∗, size_t&).

Returns

true if the call succeeded or false if it failed.

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within an object, use getValueTokenByKey() instead.

Definition at line 457 of file JsonParserGeneratorRK.h.

10.5.3.10 getOuterObject()

```
const JsonParserGeneratorRK::jsmntok_t * JsonParser::getOuterObject ( ) const
```

Gets the outer JSON object token.

Typically JSON will contain an object that contains values and possibly other objects. This method gets the token for the outer object.

A token (JsonParserGeneratorRK::jsmntok_t) identifies a particular piece of data in the JSON data, such as an object, array, or element within an object or array, such as a string, integer, boolean, etc..

Definition at line 218 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::JSMN_OBJECT, tokens, tokensEnd, and JsonParserGeneratorRK↩ ::jsmntok_t::type.

Referenced by getOuterValueByKey(), and main().

10.5.3.11 getOuterToken()

```
const JsonParserGeneratorRK::jsmntok_t * JsonParser::getOuterToken ( ) const
```

Gets the outer JSON object or array token.

A token (JsonParserGeneratorRK::jsmntok_t) identifies a particular piece of data in the JSON data, such as an object, array, or element within an object or array, such as a string, integer, boolean, etc..

Definition at line 227 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::JSMN_ARRAY, JsonParserGeneratorRK::JSMN_OBJECT, tokens, tokens↩ End, and JsonParserGeneratorRK::jsmntok_t::type.

Referenced by main(), and printJson().

10.5.3.12 getOuterValueByKey()

```
template<class T >
bool JsonParser::getOuterValueByKey (
            const char * name,
            T & result ) const [inline]
```

Gets the value with the specified key name out of the outer object.

Parameters

name	The name of the key to retrieve
result	The returned data.

Returns

true if the data was retrieved successfully, false if not (key not present or incompatible data type).

The outer object must be a JSON object, not an array.

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within an object, use getValueTokenByKey() instead.

Definition at line 393 of file JsonParserGeneratorRK.h.

References getOuterObject(), and getValueTokenByKey().

10.5.3.13 getReference()

```
JsonReference JsonParser::getReference ( ) const
```

Get a JsonReference object. This is used for fluent-style access to the data.

Definition at line 182 of file JsonParserGeneratorRK.cpp.

References JsonReference::JsonReference(), tokens, and tokensEnd.

10.5.3.14 getTokenByIndex()

```
const JsonParserGeneratorRK::jsmntok_t * JsonParser::getTokenByIndex (
             const JsonParserGeneratorRK::jsmntok_t * container,
             size_t desiredIndex ) const
```

Given a containing object, finds the nth token in the object. Internal use only.

Parameters

container	The array token to look in.
desiredIndex	The index to retrieve (0 = first, 1 = second, ...).

Returns

The token

This is used internally. It should not be used to get the nth array value, use getValueTokenByIndex instead.

Definition at line 201 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, skipObject(), and tokensEnd.

Referenced by JsonModifier::removeArrayIndex().

10.5.3.15 getTokenJsonString() [1/3]

```
bool JsonParser::getTokenJsonString (
            const JsonParserGeneratorRK::jsmntok_t * token,
            String & result ) const
```

Converts a token (object, array, string, or primitive) back into JSON in a Wiring String.

Parameters

token	The token to convert back to a string
result	Filled in with the string. Any previous contents in the string are cleared first.

Definition at line 487 of file JsonParserGeneratorRK.cpp.

References getTokenJsonString().

10.5.3.16 getTokenJsonString() [2/3]

```
bool JsonParser::getTokenJsonString (
            const JsonParserGeneratorRK::jsmntok_t * token,
            char * str,
            size_t & strLen ) const
```

Converts a token (object, array, string, or primitive) back into JSON in a buffer.

Parameters

token	The token to convert back to a string
str	The buffer to be written to
strLen	The length of the buffer on entry, set to the number of bytes written on exit.

Definition at line 495 of file JsonParserGeneratorRK.cpp.

References JsonParserString::getLength(), getTokenJsonString(), and JsonParserString::JsonParserString().

Referenced by main().

10.5.3.17 getTokenJsonString() [3/3]

```
bool JsonParser::getTokenJsonString (
            const JsonParserGeneratorRK::jsmntok_t * token,
            JsonParserString & str ) const
```

Gets a token as a JSON string.

Parameters

token	The token to convert back to a string
str	The JsonParserString object to write to

This overload is typically used internally, normally you'd use the version that takes a String& or char ∗, size_t.

Definition at line 502 of file JsonParserGeneratorRK.cpp.

References JsonParserString::append(), JsonBuffer::buffer, JsonParserGeneratorRK::jsmntok_t::end, and Json↩ ParserGeneratorRK::jsmntok_t::start.

Referenced by getTokenJsonString().

10.5.3.18 getTokens()

```
JsonParserGeneratorRK::jsmntok_t* JsonParser::getTokens ( ) [inline]
```

Used internally in the test suite for printing the token list.

Definition at line 733 of file JsonParserGeneratorRK.h.

References tokens.

Referenced by printTokens().

10.5.3.19 getTokensEnd()

```
JsonParserGeneratorRK::jsmntok_t* JsonParser::getTokensEnd ( ) [inline]
```

Used internally in the test suite for printing the token list.

Definition at line 738 of file JsonParserGeneratorRK.h.

References tokensEnd.

Referenced by printTokens().

10.5.3.20 getTokenValue() [1/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            bool & result ) const
```

Gets a bool (boolean) value.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is a bool variable.

Definition at line 338 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::buffer, JsonParserGeneratorRK::jsmntok_t::end, and JsonParserGeneratorRK::jsmntok_↩
t::start.

10.5.3.21 getTokenValue() [2/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            int & result ) const
```

Gets an integer value.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is an int variable.

Definition at line 360 of file JsonParserGeneratorRK.cpp.

References copyTokenValue().

10.5.3.22 getTokenValue() [3/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            unsigned long & result ) const
```

Gets an unsigned long value.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is an unsigned long variable.

Definition at line 373 of file JsonParserGeneratorRK.cpp.

References copyTokenValue().

10.5.3.23 getTokenValue() [4/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            float & result ) const
```

Gets a float (single precision floating point) value.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is a float variable.

Definition at line 388 of file JsonParserGeneratorRK.cpp.

References copyTokenValue().

10.5.3.24 getTokenValue() [5/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            double & result ) const
```

Gets a double (double precision floating point) value.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is a double variable.

Definition at line 397 of file JsonParserGeneratorRK.cpp.

References copyTokenValue().

10.5.3.25 getTokenValue() [6/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            String & result ) const
```

Gets a String value into a Wiring String object.

This will automatically decode Unicode character escapes in the data and the returned String will contain UTF-8.

Normally you'd use getValueByKey(), getValueByIndex() or getValueByColRow() which will automatically use this when the result parameter is a String variable.

Definition at line 408 of file JsonParserGeneratorRK.cpp.

References getTokenValue().

Referenced by main().

10.5.3.26 getTokenValue() [7/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            char * str,
            size_t & strLen ) const
```

Gets a string as a c-string into the specified buffer.

If the token specifies too large of a string it will be truncated. This will automatically decode Unicode character escapes in the data and the returned string will contain UTF-8.

Definition at line 417 of file JsonParserGeneratorRK.cpp.

References JsonParserString::getLength(), getTokenValue(), and JsonParserString::JsonParserString().

10.5.3.27 getTokenValue() [8/8]

```
bool JsonParser::getTokenValue (
            const JsonParserGeneratorRK::jsmntok_t * token,
            JsonParserString & str ) const
```

Gets a string as a JsonParserString object.

This is used internally by getTokenValue() overloads that take a String or buffer and length; you will normally not need to use this directly.

This will automatically decode Unicode character escapes in the data and the returned string will contain UTF-8.

Definition at line 425 of file JsonParserGeneratorRK.cpp.

References JsonParserString::append(), JsonBuffer::buffer, JsonParserGeneratorRK::jsmntok_t::end, and Json↵ ParserGeneratorRK::jsmntok_t::start.

Referenced by getTokenValue().

10.5.3.28 getValueByColRow()

```
template<class T >
bool JsonParser::getValueByColRow (
            const JsonParserGeneratorRK::jsmntok_t * arrayContainer,
            size_t col,
            size_t row,
            T & result ) const [inline]
```

This method is used to extract data from a 2-dimensional JSON array, an array of arrays of values.

Parameters

arrayContainer	A token for an array containing another array
col	The column (outer array index, 0 = first column, 1 = second column, ...)
row	The row (inner array index, 0 = first row, 1 = second row, ...)
result	Filled in with the value. The value can be of type: bool, int, unsigned long, float, double, String, or (char *, size_t&).

Returns

true if the call succeeded or false if it failed. You can call this repeatedly until it returns false to iterate the array.

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within a two-dimensional array, use getValueTokenByColRow() instead.

Definition at line 509 of file JsonParserGeneratorRK.h.

References getValueTokenByColRow().

10.5.3.29 getValueByIndex()

```
template<class T >
bool JsonParser::getValueByIndex (
            const JsonParserGeneratorRK::jsmntok_t * arrayContainer,
            size_t index,
            T & result ) const  [inline]
```

Given an array token in arrayContainer, gets the value with the specified index.

Parameters

arrayContainer	A token for an array
index	The index in the array. 0 = first item, 1 = second item, ...
result	Filled in with the value. The value can be of type: bool, int, unsigned long, float, double, String, or (char ∗, size_t&).

Returns

true if the call succeeded or false if it failed. You can call this repeatedly until it returns false to iterate the array.

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within an array, use getValueTokenByIndex() instead.

Definition at line 479 of file JsonParserGeneratorRK.h.

References getValueTokenByIndex().

10.5.3.30 getValueByKey()

```
template<class T >
bool JsonParser::getValueByKey (
            const JsonParserGeneratorRK::jsmntok_t * container,
            const char * name,
            T & result ) const  [inline]
```

Given an object token in container, gets the value with the specified key name.

Parameters

container	The token for the object to obtain the data from.
name	The name of the key to retrieve
result	The returned data. The value can be of type: bool, int, unsigned long, float, double, String, or (char ∗, size_t&).

Returns

true if the data was retrieved successfully, false if not (key not present or incompatible data type).

This should only be used for things like string, numbers, booleans, etc.. If you want to get a JSON array or object within an object, use getValueTokenByKey() instead.

Definition at line 367 of file JsonParserGeneratorRK.h.

References getValueTokenByKey().

10.5.3.31 getValueTokenByColRow()

```
bool JsonParser::getValueTokenByColRow (
            const JsonParserGeneratorRK::jsmntok_t * container,
            size_t col,
            size_t row,
            const JsonParserGeneratorRK::jsmntok_t *& value ) const
```

This method is used to extract data from a 2-dimensional JSON array, an array of arrays of values.

Parameters

container	A token for an array containing another array
col	The column (outer array index, 0 = first column, 1 = second column, ...)
row	The row (inner array index, 0 = first row, 1 = second row, ...)
value	Filled in with the token for the value for key.

Returns

true if the index row and column are valid or false if either is out of range.

This can be used for 2-dimensional arrays whose values are arrays or objects, to get the token for the container. It can also be used for values, but normally you'd use getValueByColRow() instead, which is generally more convenient.

Definition at line 301 of file JsonParserGeneratorRK.cpp.

References getValueTokenByIndex().

Referenced by getValueByColRow().

10.5.3.32 getValueTokenByIndex()

```
bool JsonParser::getValueTokenByIndex (
             const JsonParserGeneratorRK::jsmntok_t * container,
             size_t desiredIndex,
             const JsonParserGeneratorRK::jsmntok_t *& value ) const
```

Given an array token in container, gets the token value with the specified index.

Parameters

container	The array token to look in.
desiredIndex	The index to retrieve (0 = first, 1 = second, ...).
value	Filled in with the token for the value for key.

Returns

true if the index is valid or false if the index exceeds the size of the array.

This can be used for arrays whose values are arrays or objects, to get the token for the container. It can also be used for values, but normally you'd use getValueByIndex() instead, which is generally more convenient.

Definition at line 285 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, skipObject(), and tokensEnd.

Referenced by getValueByIndex(), getValueTokenByColRow(), JsonReference::index(), and printJsonInner().

10.5.3.33 getValueTokenByKey()

```
bool JsonParser::getValueTokenByKey (
             const JsonParserGeneratorRK::jsmntok_t * container,
             const char * key,
             const JsonParserGeneratorRK::jsmntok_t *& value ) const
```

Given an object token in container, gets the token value with the specified key name.

Parameters

container	The object token to look in.
key	The key to look for.
value	Filled in with the token for the value for key.

Returns

true if the key is found or false if not.

This can be used for objects whose keys are arrays or objects, to get the token for the container. It can also be used for values, but normally you'd use getValueByKey() instead, which is generally more convenient.

Definition at line 272 of file JsonParserGeneratorRK.cpp.

References getKeyValueTokenByIndex().

Referenced by getOuterValueByKey(), getValueByKey(), JsonReference::key(), main(), and JsonModifier::remove↩ KeyValue().

10.5.3.34 parse()

```
bool JsonParser::parse ( )
```

Parses the data you have added using addData() or addString().

When parsing data split into multiple chunks as a webhook response you can call addString() in your webhook subscription handler and call parse after each chunk. Only on the last chunk will parse return true, and you'll know the entire reponse has been received.

Definition at line 118 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::buffer, JsonParserGeneratorRK::JSMN_ERROR_NOMEM, JsonParserGeneratorRK↩ ::jsmn_init(), JsonParserGeneratorRK::jsmn_parse(), maxTokens, JsonBuffer::offset, parser, JsonBuffer::static↩ Buffers, tokens, and tokensEnd.

Referenced by JsonModifier::finish(), main(), JsonModifier::removeArrayIndex(), and JsonModifier::removeKey↩ Value().

10.5.3.35 skipObject()

```
bool JsonParser::skipObject (
            const JsonParserGeneratorRK::jsmntok_t * container,
            const JsonParserGeneratorRK::jsmntok_t *& obj ) const
```

Used internally to skip over the token in obj.

Parameters

container	The array token to look in.
obj	Object within the token, updated to the next object if true is returned

Returns

true if there was a next object, false if not.

For simple primitives and strings, this is equivalent to obj++. For objects and arrays, however, this skips over the entire object or array, including any nested objects within them.

Definition at line 237 of file JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, and tokensEnd.

Referenced by getArraySize(), getKeyValueTokenByIndex(), getTokenByIndex(), and getValueTokenByIndex().

10.5.4 Friends And Related Function Documentation

10.5.4.1 JsonModifier

```
friend class JsonModifier  [friend]
```

Definition at line 756 of file JsonParserGeneratorRK.h.

10.5.5 Member Data Documentation

10.5.5.1 maxTokens

```
size_t JsonParser::maxTokens  [protected]
```

Number of tokens that can be stored in tokens.

Definition at line 753 of file JsonParserGeneratorRK.h.

Referenced by allocateTokens(), getMaxTokens(), JsonParser(), and parse().

10.5.5.2 parser

```
JsonParserGeneratorRK::jsmn_parser JsonParser::parser  [protected]
```

The JSMN parser object.

Definition at line 754 of file JsonParserGeneratorRK.h.

Referenced by parse().

10.5.5.3 tokens

```
JsonParserGeneratorRK::jsmntok_t* JsonParser::tokens  [protected]
```

Array of tokens after parsing.

Definition at line 751 of file JsonParserGeneratorRK.h.

Referenced by allocateTokens(), getOuterArray(), getOuterObject(), getOuterToken(), getReference(), getTokens(), JsonParser(), parse(), and ∼JsonParser().

10.5.5.4 tokensEnd

```
JsonParserGeneratorRK::jsmntok_t* JsonParser::tokensEnd [protected]
```

Pointer into tokens, points after last used token.

Definition at line 752 of file JsonParserGeneratorRK.h.

Referenced by getArraySize(), getKeyValueTokenByIndex(), getOuterArray(), getOuterObject(), getOuterToken(), getReference(), getTokenByIndex(), getTokensEnd(), getValueTokenByIndex(), JsonParser(), parse(), and skip↩ Object().

The documentation for this class was generated from the following files:

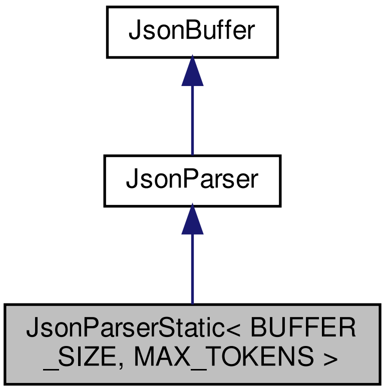
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.6 JsonParserStatic< BUFFER_SIZE, MAX_TOKENS > Class Template Reference

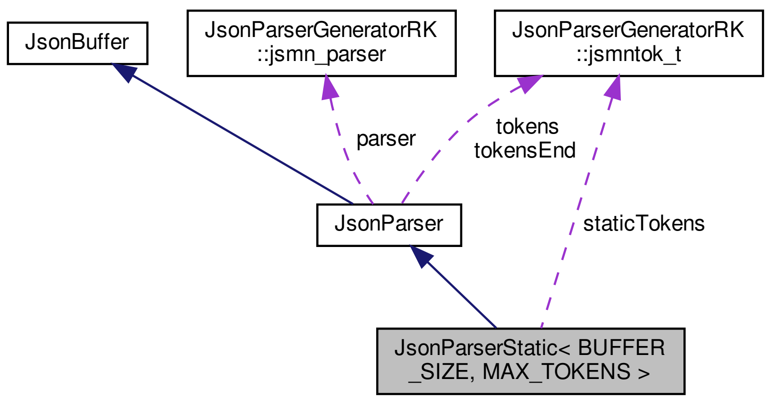
Creates a JsonParser with a static buffer.

```
#include <JsonParserGeneratorRK.h>
```

Inheritance diagram for JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >:



Collaboration diagram for JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >:


JsonBuffer
JsonParserGeneratorRK ::jsmn_parser
JsonParserGeneratorRK ::jsmntok_t
parser
tokens
tokensEnd
JsonParser
staticTokens
JsonParserStatic< BUFFER _SIZE, MAX_TOKENS >

Public Member Functions

- JsonParserStatic ()

  Construct a JsonParser using a static buffer and static maximum number of tokens.

Private Attributes

- char staticBuffer [BUFFER_SIZE]

  The static buffer to hold the data.
- JsonParserGeneratorRK::jsmntok_t staticTokens [MAX_TOKENS]

  The static buffer to hold the tokens.

Additional Inherited Members

10.6.1 Detailed Description

template<size_t BUFFER_SIZE, size_t MAX_TOKENS>
class JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >

Creates a JsonParser with a static buffer.

You normally use this when you're creating a parser as a global variable. For small data (under around 256 bytes so) you can also allocate one on the stack.

Parameters

BUFFER_SIZE	The maximum size of the data to be parsed, in bytes. If you are parsing a webhook response split into parts, this is the total size of all parts.
MAX_TOKENS	The maximum number of tokens you expect. Each object has a token and two for each key/value pair. Each array is a token and one for each element in the array.

Definition at line 772 of file JsonParserGeneratorRK.h.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 JsonParserStatic()

```
template<size_t BUFFER_SIZE, size_t MAX_TOKENS>
JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >::JsonParserStatic ( )  [inline], [explicit]
```

Construct a JsonParser using a static buffer and static maximum number of tokens.

Definition at line 777 of file JsonParserGeneratorRK.h.

10.6.3 Member Data Documentation

10.6.3.1 staticBuffer

```
template<size_t BUFFER_SIZE, size_t MAX_TOKENS>
char JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >::staticBuffer[BUFFER_SIZE]  [private]
```

The static buffer to hold the data.

Definition at line 777 of file JsonParserGeneratorRK.h.

10.6.3.2 staticTokens

```
template<size_t BUFFER_SIZE, size_t MAX_TOKENS>
JsonParserGeneratorRK::jsmntok_t JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >::staticTokens[M←
AX_TOKENS]  [private]
```

The static buffer to hold the tokens.

Definition at line 781 of file JsonParserGeneratorRK.h.

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

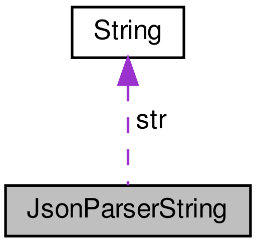
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.7 JsonParserString Class Reference

Class used internally for writing to strings.

```
#include <JsonParserGeneratorRK.h>
```

Collaboration diagram for JsonParserString:



Public Member Functions

- JsonParserString (String ∗str)

  Construct a JsonParserString wrapping a Wiring String.
- JsonParserString (char ∗buf, size_t bufLen)

  Construct a JsonParserString wrapping a buffer and length.
- void append (char ch)

  Append a single char to the underlying string.
- void append (const char ∗str, size_t len)

  Append a buffer and length to the underlying string.
- size_t getLength () const

  Get the length of the string.

Protected Attributes

- String ∗ str

  When writing to a String, the String object.
- char ∗ buf

  When writing to a buffer, the pointer to the buffer. Not used for String.
- size_t bufLen

  When writing to a buffer, the length of the buffer in bytes. Not used for String.
- size_t length

  The current offset being written to.

10.7.1 Detailed Description

Class used internally for writing to strings.

This is a wrapper around either String (the Wiring version) or a buffer and length. This allows writing to a static buffer with no dynamic memory allocation at all.

One of the things about String is that while you can pre-allocate reserve space for data, you can't get access to the internal length field, so you can't write to raw bytes then resize it to the correct size. This wrapper is that allows appending to either a String or buffer to get around this limitation of String.

You can also use it for sizing only by passing NULL for buf.

Definition at line 92 of file JsonParserGeneratorRK.h.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 JsonParserString() [1/2]

```
JsonParserString::JsonParserString (
             String * str )
```

Construct a JsonParserString wrapping a Wiring String.

Parameters

str	A pointer Wiring String object to write to.

Definition at line 623 of file JsonParserGeneratorRK.cpp.

References buf, bufLen, and length.

10.7.2.2 JsonParserString() [2/2]

```
JsonParserString::JsonParserString (
             char * buf,
             size_t bufLen )
```

Construct a JsonParserString wrapping a buffer and length.

Parameters

buf	A pointer to a buffer
bufLen	The length of the buffer in bytes

Definition at line 626 of file JsonParserGeneratorRK.cpp.

References buf, bufLen, and length.

Referenced by JsonParser::getTokenJsonString(), and JsonParser::getTokenValue().

10.7.3 Member Function Documentation

10.7.3.1 append() [1/2]

```
void JsonParserString::append (
            char ch )
```

Append a single char to the underlying string.

Parameters

ch	The char to append.

Definition at line 632 of file JsonParserGeneratorRK.cpp.

References buf, bufLen, and length.

Referenced by append(), JsonParser::appendUtf8(), and JsonParser::getTokenValue().

10.7.3.2 append() [2/2]

```
void JsonParserString::append (
            const char * str,
            size_t len )
```

Append a buffer and length to the underlying string.

Parameters

str	A pointer to the character to add. Does not need to be null-terminated.
len	Length of the string to append in bytes.

Definition at line 645 of file JsonParserGeneratorRK.cpp.

References append().

Referenced by JsonParser::getTokenJsonString().

10.7.3.3 getLength()

```
size_t JsonParserString::getLength ( ) const  [inline]
```

Get the length of the string.

Returns

The string length in bytes. If the string contains UTF-8 characters, it will be the number of bytes, not characters.

For buffer and bufLenb, the maximum string length will be bufLen - 1 to leave room for the null terminator.

Definition at line 134 of file JsonParserGeneratorRK.h.

References length.

Referenced by JsonParser::getTokenJsonString(), and JsonParser::getTokenValue().

10.7.4 Member Data Documentation

10.7.4.1 buf

```
char* JsonParserString::buf  [protected]
```

When writing to a buffer, the pointer to the buffer. Not used for String.

Definition at line 138 of file JsonParserGeneratorRK.h.

Referenced by append(), and JsonParserString().

10.7.4.2 bufLen

```
size_t JsonParserString::bufLen  [protected]
```

When writing to a buffer, the length of the buffer in bytes. Not used for String.

Definition at line 139 of file JsonParserGeneratorRK.h.

Referenced by append(), and JsonParserString().

10.7.4.3 length

```
size_t JsonParserString::length [protected]
```

The current offset being written to.

Definition at line 140 of file JsonParserGeneratorRK.h.

Referenced by append(), getLength(), and JsonParserString().

10.7.4.4 str

```
String* JsonParserString::str [protected]
```

When writing to a String, the String object.

Definition at line 137 of file JsonParserGeneratorRK.h.

The documentation for this class was generated from the following files:

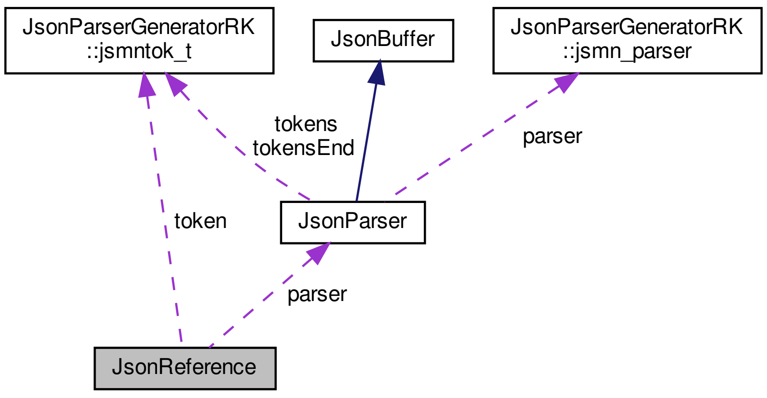
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.8 JsonReference Class Reference

This class provides a fluent-style API for easily traversing a tree of JSON objects to find a value.

```
#include <JsonParserGeneratorRK.h>
```

Collaboration diagram for JsonReference:


JsonParserGeneratorRK ::jsmntok_t
JsonBuffer
JsonParserGeneratorRK ::jsmn_parser
tokens tokensEnd
parser
token
JsonParser
parser
JsonReference

Public Member Functions

- JsonReference (const JsonParser ∗parser)

  Constructs an object. Normally you use the JsonParser getReference() method to get one of these instead of constructing one.
- virtual ∼JsonReference ()

  Destructor. This does not affect the lifecycle of the JsonParser.
- JsonReference (const JsonParser ∗parser, const JsonParserGeneratorRK::jsmntok_t ∗token)

  Constructs are JsonReference for a specific token within a JsonParser.
- JsonReference key (const char ∗name) const

  For JsonReference that refers to a JSON object, gets a new JsonReference to a value with the specified key name.
- JsonReference index (size_t index) const

  For a JsonReference that refers to a JSON array, gets a new JsonReference to a value in the array by index.
- size_t size () const

  For a JsonReference that refers to a JSON array, gets the size of the array.
- template<class T >
  bool value (T &result) const

  Get a value of the specified type for a given value for a specified key, or index for an array.
- bool valueBool (bool defaultValue=false) const

  Returns a boolean (bool) value for an object value for key, or array index.
- int valueInt (int defaultValue=0) const

  Returns a integer (int) value for an object value for key, or array index.
- unsigned long valueUnsignedLong (unsigned long defaultValue=0) const

  Returns a unsigned long integer for an object value for key, or array index.
- float valueFloat (float defaultValue=0.0) const

  Returns a float value for an object value for key, or array index.
- double valueDouble (double defaultValue=0.0) const

  Returns a double value for an object value for key, or array index.
- String valueString () const

  Returns a String value for an object value for key, or array index.

Private Attributes

- const JsonParser ∗ parser
- const JsonParserGeneratorRK::jsmntok_t ∗ token

10.8.1 Detailed Description

This class provides a fluent-style API for easily traversing a tree of JSON objects to find a value.

Definition at line 788 of file JsonParserGeneratorRK.h.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 JsonReference() [1/2]

```
JsonReference::JsonReference (
            const JsonParser ∗ parser )
```

Constructs an object. Normally you use the JsonParser getReference() method to get one of these instead of constructing one.

Parameters

parser	The JsonParser object you're traversing

Definition at line 546 of file JsonParserGeneratorRK.cpp.

References parser, and token.

Referenced by JsonParser::getReference(), index(), and key().

10.8.2.2 ∼JsonReference()

```
JsonReference::∼JsonReference ( )  [virtual]
```

Destructor. This does not affect the lifecycle of the JsonParser.

Definition at line 550 of file JsonParserGeneratorRK.cpp.

10.8.2.3 JsonReference() [2/2]

```
JsonReference::JsonReference (
            const JsonParser * parser,
            const JsonParserGeneratorRK::jsmntok_t * token )
```

Constructs are JsonReference for a specific token within a JsonParser.

Definition at line 553 of file JsonParserGeneratorRK.cpp.

References parser, and token.

Referenced by JsonParser::getReference(), index(), and key().

10.8.3 Member Function Documentation

10.8.3.1 index()

```
JsonReference JsonReference::index (
            size_t index ) const
```

For a JsonReference that refers to a JSON array, gets a new JsonReference to a value in the array by index.

Parameters

index	The index to retrieve (0 = first item, 1 = second item, ...).

Returns

A JsonReference to the value for this index.

Definition at line 567 of file JsonParserGeneratorRK.cpp.

References JsonParser::getValueTokenByIndex(), JsonReference(), parser, and token.

10.8.3.2 key()

```
JsonReference JsonReference::key (
                const char * name ) const
```

For JsonReference that refers to a JSON object, gets a new JsonReference to a value with the specified key name.

Parameters

name	of the key to look for.

Returns

A JsonReference to the value for this key.

Definition at line 556 of file JsonParserGeneratorRK.cpp.

References JsonParser::getValueTokenByKey(), JsonReference(), parser, and token.

10.8.3.3 size()

```
size_t JsonReference::size ( ) const
```

For a JsonReference that refers to a JSON array, gets the size of the array.

Returns

0 = an empty array, 1 = one element, ...

Definition at line 578 of file JsonParserGeneratorRK.cpp.

References JsonParser::getArraySize(), parser, and token.

10.8.3.4 value()

```
template<class T >
bool JsonReference::value (
            T & result ) const  [inline]
```

Get a value of the specified type for a given value for a specified key, or index for an array.

Parameters

result	Filled in with the value. The value can be of type: bool, int, unsigned long, float, double, String, or (char ∗, size_t&).

There are also type-specific versions like valueBool that return the value, instead of having to pass an object to hold the value, as in this call.

Definition at line 843 of file JsonParserGeneratorRK.h.

References parser, and token.

10.8.3.5 valueBool()

```
bool JsonReference::valueBool (
            bool defaultValue = false ) const
```

Returns a boolean (bool) value for an object value for key, or array index.

Parameters

defaultValue	Optional value to use if the key or array index is not found. Default: false.

Definition at line 587 of file JsonParserGeneratorRK.cpp.

10.8.3.6 valueDouble()

```
double JsonReference::valueDouble (
            double defaultValue = 0.0 ) const
```

Returns a double value for an object value for key, or array index.

Parameters

defaultValue	Optional value to use if the key or array index is not found. Default: 0.0.

Definition at line 607 of file JsonParserGeneratorRK.cpp.

10.8.3.7 valueFloat()

```
float JsonReference::valueFloat (
            float defaultValue = 0.0 ) const
```

Returns a float value for an object value for key, or array index.

Parameters

defaultValue	Optional value to use if the key or array index is not found. Default: 0.0.

Definition at line 602 of file JsonParserGeneratorRK.cpp.

10.8.3.8 valueInt()

```
int JsonReference::valueInt (
             int defaultValue = 0 ) const
```

Returns a integer (int) value for an object value for key, or array index.

Parameters

defaultValue	Optional value to use if the key or array index is not found. Default: 0.

Definition at line 592 of file JsonParserGeneratorRK.cpp.

10.8.3.9 valueString()

```
String JsonReference::valueString ( ) const
```

Returns a String value for an object value for key, or array index.

Returns

The string value, or an empty string if the key or array index is not found.

Definition at line 612 of file JsonParserGeneratorRK.cpp.

10.8.3.10 valueUnsignedLong()

```
unsigned long JsonReference::valueUnsignedLong (
             unsigned long defaultValue = 0 ) const
```

Returns a unsigned long integer for an object value for key, or array index.

Parameters

defaultValue	Optional value to use if the key or array index is not found. Default: 0.

Definition at line 597 of file JsonParserGeneratorRK.cpp.

10.8.4 Member Data Documentation

10.8.4.1 parser

```
const JsonParser* JsonReference::parser  [private]
```

Definition at line 895 of file JsonParserGeneratorRK.h.

Referenced by index(), JsonReference(), key(), size(), and value().

10.8.4.2 token

```
const JsonParserGeneratorRK::jsmntok_t* JsonReference::token  [private]
```

Definition at line 896 of file JsonParserGeneratorRK.h.

Referenced by index(), JsonReference(), key(), size(), and value().

The documentation for this class was generated from the following files:

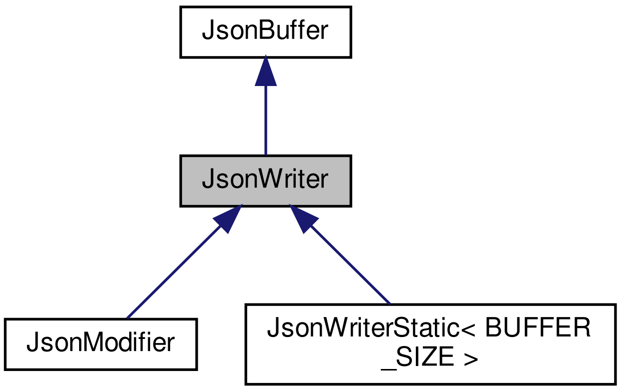
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.9 JsonWriter Class Reference

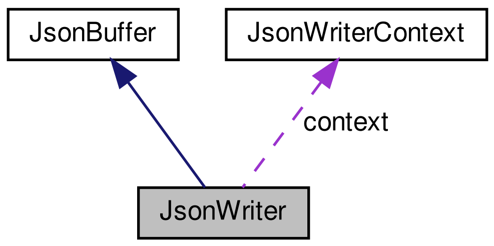
Class for building a JSON string.

```
#include <JsonParserGeneratorRK.h>
```

Inheritance diagram for JsonWriter:



Collaboration diagram for JsonWriter:



Public Member Functions

- JsonWriter ()

    Construct a JsonWriter with a dynamically allocated buffer.
- virtual ∼JsonWriter ()

    Destroy the object. If the buffer was dynamically allocated it will be freed.
- JsonWriter (char ∗buffer, size_t bufferLen)

    Construct a JsonWriter to write to a static buffer.
- void init ()

    Reset the writer, clearing all data.
- bool startObject ()

    Start a new JSON object. Make sure you finish it with finishObjectOrArray()
- bool startArray ()

    Start a new JSON array. Make sure you finish it with finishObjectOrArray()
- void finishObjectOrArray ()

    Finsh an object or array started with startObject() or startArray()
- void insertValue (bool value)

    Inserts a boolean value ("true" or "false").
- void insertValue (int value)

    Inserts an integer value.
- void insertValue (unsigned int value)

    Inserts an unsigned integer value.
- void insertValue (long value)

    Inserts a long integer value.
- void insertValue (unsigned long value)

    Inserts an unsigned long integer value.
- void insertValue (float value)

    Inserts a floating point value.
- void insertValue (double value)

    Inserts a floating point double value.
- void insertValue (const char ∗value)

    Inserts a quoted string value. This escapes special characters and encodes utf-8.
- void insertValue (const String &value)

    Inserts a quoted string value.

- void insertKeyObject (const char ∗key)

    Inserts a new key and empty object. You must close the object using finishObjectOrArray()!
- void insertKeyArray (const char ∗key)

    Inserts a new key and empty array. You must close the object using finishObjectOrArray()!
- template<class T >
  void insertKeyValue (const char ∗key, T value)

    Inserts a key/value pair into an object.
- template<class T >
  void insertArrayValue (T value)

    Inserts a value into an array.
- template<class T >
  void insertArray (T ∗pArray, size_t numElem)

    Inserts an array of values into an array.
- template<class T >
  void insertKeyArray (const char ∗key, T ∗pArray, size_t numElem)

    Inserts a new key and vector of values.
- template<class T >
  void insertVector (std::vector< T > vec)

    Inserts an array of values into an array.
- template<class T >
  void insertKeyVector (const char ∗key, std::vector< T > vec)

    Inserts a new key and vector of values.
- bool isTruncated () const
- void setFloatPlaces (int floatPlaces)

    Sets the number of digits for formatting float and double values.
- void insertCheckSeparator ()

    Check to see if a separator needs to be inserted. Used internally.
- bool startObjectOrArray (char startChar, char endChar)

    Used internally to start an object or array.
- void insertChar (char ch)

    Used internally to insert a character.
- void insertString (const char ∗s, bool quoted=false)

    Used internally to insert a string, quoted or not.
- void insertsprintf (const char ∗fmt,...)

    Used internally to insert using snprintf formatting.
- void insertvsprintf (const char ∗fmt, va_list ap)

    Used internally to insert using snprintf formatting with a va_list.
- void setIsFirst (bool isFirst=true)

    Used internally to set the current isFirst flag in the context.

Static Public Attributes

- static const size_t MAX_NESTED_CONTEXT = 9

Protected Attributes

- size_t contextIndex

    Index into the context for the current level of nesting.
- JsonWriterContext context [MAX_NESTED_CONTEXT]

    Structure for managing nested objects.
- bool truncated

    true if data was added that didn't fit and was truncated
- int floatPlaces

    default number of places to display for floating point numbers (default is -1, the default for sprintf)

10.9.1 Detailed Description

Class for building a JSON string.

Definition at line 910 of file JsonParserGeneratorRK.h.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 JsonWriter() [1/2]

```
JsonWriter::JsonWriter ( )
```

Construct a JsonWriter with a dynamically allocated buffer.

The buffer will be resized as necessary but you can improve efficiency by using the allocate() method of JsonBuffer to pre-allocate space rather than have to incrementally make it bigger as it's written to.

Use getBuffer() to get the pointer to the buffer and getOffset() to get the buffer pointer and size. The buffer is not null-terminated!

Definition at line 655 of file JsonParserGeneratorRK.cpp.

References floatPlaces, init(), and JsonBuffer::JsonBuffer().

10.9.2.2 ∼JsonWriter()

```
JsonWriter::∼JsonWriter ( )  [virtual]
```

Destroy the object. If the buffer was dynamically allocated it will be freed.

If the buffer was passed in using the buffer, bufferLen constructor the buffer is not freed by this call as it's likely statically allocated.

Definition at line 659 of file JsonParserGeneratorRK.cpp.

10.9.2.3 JsonWriter() [2/2]

```
JsonWriter::JsonWriter (
         char * buffer,
         size_t bufferLen )
```

Construct a JsonWriter to write to a static buffer.

Parameters

buffer	Pointer to the buffer
bufferLen	Length of the buffer in bytes

Definition at line 663 of file JsonParserGeneratorRK.cpp.

References floatPlaces, init(), and JsonBuffer::JsonBuffer().

Referenced by main().

10.9.3 Member Function Documentation

10.9.3.1 finishObjectOrArray()

```
void JsonWriter::finishObjectOrArray ( )
```

Finsh an object or array started with startObject() or startArray()

Definition at line 695 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::buffer, JsonBuffer::bufferLen, context, contextIndex, insertChar(), JsonBuffer::offset, and JsonWriterContext::terminator.

Referenced by insertKeyArray(), insertKeyVector(), main(), JsonWriterAutoArray::∼JsonWriterAutoArray(), and JsonWriterAutoObject::∼JsonWriterAutoObject().

10.9.3.2 init()

```
void JsonWriter::init ( )
```

Reset the writer, clearing all data.

You do not need to call init() as it's called from the two constructors. You can call it again if you want to reset the writer and reuse it, such as when you use JsonWriterStatic in a global variable.

Definition at line 667 of file JsonParserGeneratorRK.cpp.

References context, contextIndex, JsonWriterContext::isFirst, JsonBuffer::offset, JsonWriterContext::terminator, and truncated.

Referenced by JsonWriter(), JsonModifier::startAppend(), and JsonModifier::startModify().

10.9.3.3 insertArray()

```
template<class T >
void JsonWriter::insertArray (
            T * pArray,
            size_t numElem )  [inline]
```

Inserts an array of values into an array.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char ∗.

Definition at line 1091 of file JsonParserGeneratorRK.h.

10.9.3.4 insertArrayValue()

```
template<class T >
void JsonWriter::insertArrayValue (
            T value )  [inline]
```

Inserts a value into an array.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char ∗.

Definition at line 1079 of file JsonParserGeneratorRK.h.

References insertCheckSeparator().

10.9.3.5 insertChar()

```
void JsonWriter::insertChar (
            char ch )
```

Used internally to insert a character.

Used internally. You should use insertKeyValue() or insertArrayValue() with a string instead.

Definition at line 712 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::buffer, JsonBuffer::bufferLen, JsonBuffer::offset, and truncated.

Referenced by finishObjectOrArray(), insertCheckSeparator(), insertKeyArray(), insertKeyObject(), insertKey←Value(), insertString(), and startObjectOrArray().

10.9.3.6 insertCheckSeparator()

```
void JsonWriter::insertCheckSeparator ( )
```

Check to see if a separator needs to be inserted. Used internally.

You normally don't need to use this as it's called by insertKeyValue() and insertArrayValue().

Definition at line 823 of file JsonParserGeneratorRK.cpp.

References context, contextIndex, insertChar(), and JsonWriterContext::isFirst.

Referenced by insertArrayValue(), insertKeyArray(), insertKeyObject(), insertKeyValue(), main(), and startObject↩ OrArray().

10.9.3.7 insertKeyArray() [1/2]

```
void JsonWriter::insertKeyArray (
             const char * key )
```

Inserts a new key and empty array. You must close the object using finishObjectOrArray()!

Parameters

key	the key name to insert

Definition at line 867 of file JsonParserGeneratorRK.cpp.

References insertChar(), insertCheckSeparator(), insertValue(), setIsFirst(), and startArray().

Referenced by insertKeyArray(), insertKeyVector(), and main().

10.9.3.8 insertKeyArray() [2/2]

```
template<class T >
void JsonWriter::insertKeyArray (
             const char * key,
             T * pArray,
             size_t numElem )  [inline]
```

Inserts a new key and vector of values.

Parameters

key	the key name to insert
vec	the vector to insert

Definition at line 1105 of file JsonParserGeneratorRK.h.

References finishObjectOrArray(), and insertKeyArray().

10.9.3.9 insertKeyObject()

```
void JsonWriter::insertKeyObject (
             const char * key )
```

Inserts a new key and empty object. You must close the object using finishObjectOrArray()!

Parameters

key	the key name to insert

Definition at line 859 of file JsonParserGeneratorRK.cpp.

References insertChar(), insertCheckSeparator(), insertValue(), setIsFirst(), and startObject().

10.9.3.10 insertKeyValue()

```
template<class T >
void JsonWriter::insertKeyValue (
             const char * key,
             T value )  [inline]
```

Inserts a key/value pair into an object.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char *.

Definition at line 1065 of file JsonParserGeneratorRK.h.

References insertChar(), insertCheckSeparator(), and insertValue().

10.9.3.11 insertKeyVector()

```
template<class T >
void JsonWriter::insertKeyVector (
             const char * key,
             std::vector< T > vec )  [inline]
```

Inserts a new key and vector of values.

Parameters

key	the key name to insert
vec	the vector to insert

Definition at line 1132 of file JsonParserGeneratorRK.h.

References finishObjectOrArray(), and insertKeyArray().

10.9.3.12 insertsprintf()

```
void JsonWriter::insertsprintf (
             const char * fmt,
              ...  )
```

Used internally to insert using snprintf formatting.

Used internally. You should use insertKeyValue() or insertArrayValue() with a string, float, or double instead.

This method does not quote or escape the string - it's used mainly for formatting numbers.

Definition at line 802 of file JsonParserGeneratorRK.cpp.

Referenced by insertValue().

10.9.3.13 insertString()

```
void JsonWriter::insertString (
             const char * s,
             bool quoted = false )
```

Used internally to insert a string, quoted or not.

Used internally. You should use insertKeyValue() or insertArrayValue() with a string instead.

Definition at line 721 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::bufferLen, insertChar(), and JsonBuffer::offset.

Referenced by insertValue(), and main().

10.9.3.14 insertValue() [1/9]

```
void JsonWriter::insertValue (
            bool value )
```

Inserts a boolean value ("true" or "false").

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separtators between items.

Definition at line 832 of file JsonParserGeneratorRK.cpp.

References insertString().

10.9.3.15 insertValue() [2/9]

```
void JsonWriter::insertValue (
            int value )  [inline]
```

Inserts an integer value.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separators between items.

Definition at line 979 of file JsonParserGeneratorRK.h.

References insertsprintf().

Referenced by main().

10.9.3.16 insertValue() [3/9]

```
void JsonWriter::insertValue (
            unsigned int value )  [inline]
```

Inserts an unsigned integer value.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separators between items.

Definition at line 987 of file JsonParserGeneratorRK.h.

References insertsprintf().

10.9.3.17 insertValue() [4/9]

```
void JsonWriter::insertValue (
             long value )  [inline]
```

Inserts a long integer value.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separators between items.

Definition at line 995 of file JsonParserGeneratorRK.h.

References insertsprintf().

10.9.3.18 insertValue() [5/9]

```
void JsonWriter::insertValue (
             unsigned long value )  [inline]
```

Inserts an unsigned long integer value.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separators between items.

Definition at line 1003 of file JsonParserGeneratorRK.h.

References insertsprintf().

10.9.3.19 insertValue() [6/9]

```
void JsonWriter::insertValue (
             float value )
```

Inserts a floating point value.

Use setFloatPlaces() to set the number of decimal places to include.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separtators between items.

Definition at line 841 of file JsonParserGeneratorRK.cpp.

References floatPlaces, and insertsprintf().

10.9.3.20 insertValue() [7/9]

```
void JsonWriter::insertValue (
             double value )
```

Inserts a floating point double value.

Use setFloatPlaces() to set the number of decimal places to include.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separtators between items.

Definition at line 849 of file JsonParserGeneratorRK.cpp.

References floatPlaces, and insertsprintf().

Referenced by main().

10.9.3.21 insertValue() [8/9]

```
void JsonWriter::insertValue (
             const char * value )   [inline]
```

Inserts a quoted string value. This escapes special characters and encodes utf-8.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separtators between items.

Definition at line 1031 of file JsonParserGeneratorRK.h.

References insertString().

Referenced by insertKeyArray(), insertKeyObject(), and insertKeyValue().

10.9.3.22 insertValue() [9/9]

```
void JsonWriter::insertValue (
             const String & value )   [inline]
```

Inserts a quoted string value.

This escapes special characters and encodes utf-8. See also the version that takes a plain const char ∗.

You would normally use insertKeyValue() or insertArrayValue() instead of calling this directly as those functions take care of inserting the separtators between items.

Definition at line 1042 of file JsonParserGeneratorRK.h.

10.9.3.23 insertVector()

```
template<class T >
void JsonWriter::insertVector (
            std::vector< T > vec )  [inline]
```

Inserts an array of values into an array.

Uses templates so you can pass any type object that's supported by insertValue() overloads, for example: bool, int, float, double, const char ∗.

Definition at line 1118 of file JsonParserGeneratorRK.h.

10.9.3.24 insertvsprintf()

```
void JsonWriter::insertvsprintf (
            const char * fmt,
            va_list ap )
```

Used internally to insert using snprintf formatting with a va_list.

Used internally. You should use insertKeyValue() or insertArrayValue() with a string, float, or double instead.

This method does not quote or escape the string - it's used mainly for formatting numbers.

Definition at line 809 of file JsonParserGeneratorRK.cpp.

References JsonBuffer::bufferLen, JsonBuffer::offset, and truncated.

10.9.3.25 isTruncated()

```
bool JsonWriter::isTruncated ( ) const  [inline]
```

If you try to insert more data than will fit in the buffer, the isTruncated flag will be set, and the buffer will likely not be valid JSON and should not be used.

Definition at line 1142 of file JsonParserGeneratorRK.h.

References truncated.

10.9.3.26 setFloatPlaces()

```
void JsonWriter::setFloatPlaces (
            int floatPlaces )  [inline]
```

Sets the number of digits for formatting float and double values.

Parameters

floatPlaces	The number of decimal places for float and double. Set it to -1 to use the default for snprintf. -1 is the default value if you don't call setFloatPlaces.

Definition at line 1150 of file JsonParserGeneratorRK.h.

References floatPlaces.

10.9.3.27 setIsFirst()

```
void JsonWriter::setIsFirst (
            bool isFirst = true )
```

Used internally to set the current isFirst flag in the context.

Definition at line 875 of file JsonParserGeneratorRK.cpp.

References context, contextIndex, and JsonWriterContext::isFirst.

Referenced by insertKeyArray(), insertKeyObject(), and JsonModifier::startAppend().

10.9.3.28 startArray()

```
bool JsonWriter::startArray ( )  [inline]
```

Start a new JSON array. Make sure you finish it with finishObjectOrArray()

Definition at line 958 of file JsonParserGeneratorRK.h.

References startObjectOrArray().

Referenced by insertKeyArray(), and JsonWriterAutoArray::JsonWriterAutoArray().

10.9.3.29 startObject()

```
bool JsonWriter::startObject ( )  [inline]
```

Start a new JSON object. Make sure you finish it with finishObjectOrArray()

Definition at line 953 of file JsonParserGeneratorRK.h.

References startObjectOrArray().

Referenced by insertKeyObject(), JsonWriterAutoObject::JsonWriterAutoObject(), and main().

10.9.3.30 startObjectOrArray()

```
bool JsonWriter::startObjectOrArray (
            char startChar,
            char endChar )
```

Used internally to start an object or array.

Used internally; you should use startObject() or startArray() instead. Make sure you finish any started object or array using finishObjectOrArray().

Definition at line 679 of file JsonParserGeneratorRK.cpp.

References context, contextIndex, insertChar(), insertCheckSeparator(), JsonWriterContext::isFirst, MAX_NEST←ED_CONTEXT, and JsonWriterContext::terminator.

Referenced by startArray(), and startObject().

10.9.4 Member Data Documentation

10.9.4.1 context

```
JsonWriterContext JsonWriter::context[MAX_NESTED_CONTEXT] [protected]
```

Structure for managing nested objects.

Definition at line 1217 of file JsonParserGeneratorRK.h.

Referenced by finishObjectOrArray(), init(), insertCheckSeparator(), setIsFirst(), and startObjectOrArray().

10.9.4.2 contextIndex

```
size_t JsonWriter::contextIndex [protected]
```

Index into the context for the current level of nesting.

Definition at line 1216 of file JsonParserGeneratorRK.h.

Referenced by finishObjectOrArray(), init(), insertCheckSeparator(), setIsFirst(), and startObjectOrArray().

10.9.4.3 floatPlaces

```
int JsonWriter::floatPlaces  [protected]
```

default number of places to display for floating point numbers (default is -1, the default for sprintf)

Definition at line 1219 of file JsonParserGeneratorRK.h.

Referenced by insertValue(), JsonWriter(), and setFloatPlaces().

10.9.4.4 MAX_NESTED_CONTEXT

```
const size_t JsonWriter::MAX_NESTED_CONTEXT = 9  [static]
```

This constant is the maximum number of nested objects that are supported; the actual number is one less than this so when set to 9 you can have eight objects nested in each other.

Overhead is 8 bytes per nested context, so 9 elements is 72 bytes.

Definition at line 1213 of file JsonParserGeneratorRK.h.

Referenced by startObjectOrArray().

10.9.4.5 truncated

```
bool JsonWriter::truncated  [protected]
```

true if data was added that didn't fit and was truncated

Definition at line 1218 of file JsonParserGeneratorRK.h.

Referenced by init(), insertChar(), insertvsprintf(), and isTruncated().

The documentation for this class was generated from the following files:

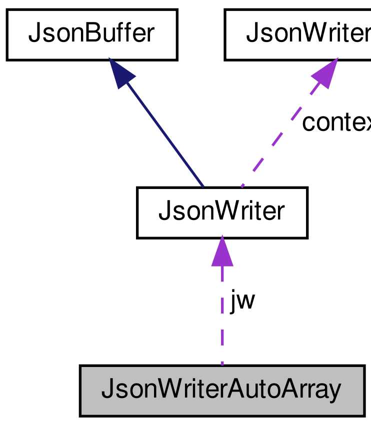
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp

10.10 JsonWriterAutoArray Class Reference

Class for creating a JSON array with JsonWriter.

```
#include <JsonParserGeneratorRK.h>
```

Collaboration diagram for JsonWriterAutoArray:



Public Member Functions

- JsonWriterAutoArray (JsonWriter ∗jw)

  Start a new array.
- ∼JsonWriterAutoArray ()

  End the array.

Protected Attributes

- JsonWriter ∗ jw

  JsonWriter to write to.

10.10.1 Detailed Description

Class for creating a JSON array with JsonWriter.

When you create an object, you must call startArray() to start and finishObjectOrArray() to complete it.

This class is instantiated on the stack to automatically start and finish for you.

Definition at line 1285 of file JsonParserGeneratorRK.h.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 JsonWriterAutoArray()

```
JsonWriterAutoArray::JsonWriterAutoArray (
            JsonWriter * jw )  [inline]
```

Start a new array.

Parameters

jw	The JsonWriter object to insert the array into

Definition at line 1292 of file JsonParserGeneratorRK.h.

References jw, and JsonWriter::startArray().

10.10.2.2 ∼JsonWriterAutoArray()

```
JsonWriterAutoArray::∼JsonWriterAutoArray ( )  [inline]
```

End the array.

Definition at line 1299 of file JsonParserGeneratorRK.h.

References JsonWriter::finishObjectOrArray(), and jw.

10.10.3 Member Data Documentation

10.10.3.1 jw

```
JsonWriter* JsonWriterAutoArray::jw  [protected]
```

JsonWriter to write to.

Definition at line 1304 of file JsonParserGeneratorRK.h.

Referenced by JsonWriterAutoArray(), and ∼JsonWriterAutoArray().

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

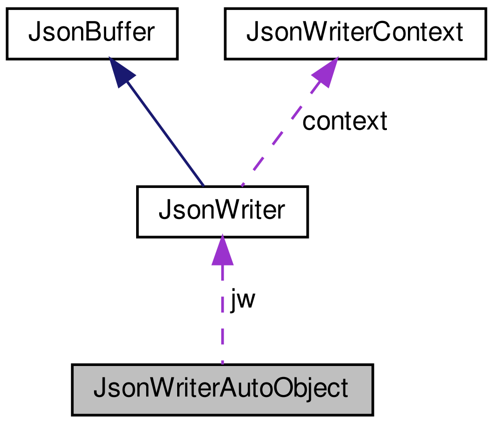
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.11 JsonWriterAutoObject Class Reference

Class for creating a JSON object with JsonWriter.

```
#include <JsonParserGeneratorRK.h>
```

Collaboration diagram for JsonWriterAutoObject:



Public Member Functions

- JsonWriterAutoObject (JsonWriter ∗jw)

  Start a new object.
- ∼JsonWriterAutoObject ()

  End the object.

Protected Attributes

- JsonWriter ∗ jw

  JsonWriter to write to.

10.11.1 Detailed Description

Class for creating a JSON object with JsonWriter.

When you create an object, you must call startObject() to start and finishObjectOrArray() to complete it.

This class is instantiated on the stack to automatically start and finish for you.

Definition at line 1256 of file JsonParserGeneratorRK.h.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 JsonWriterAutoObject()

```
JsonWriterAutoObject::JsonWriterAutoObject (
            JsonWriter * jw )  [inline]
```

Start a new object.

Parameters

jw	The JsonWriter object to insert the object into

Definition at line 1263 of file JsonParserGeneratorRK.h.

References jw, and JsonWriter::startObject().

10.11.2.2 ∼JsonWriterAutoObject()

```
JsonWriterAutoObject::∼JsonWriterAutoObject ( )  [inline]
```

End the object.

Definition at line 1270 of file JsonParserGeneratorRK.h.

References JsonWriter::finishObjectOrArray(), and jw.

10.11.3 Member Data Documentation

10.11.3.1 jw

```
JsonWriter* JsonWriterAutoObject::jw  [protected]
```

JsonWriter to write to.

Definition at line 1275 of file JsonParserGeneratorRK.h.

Referenced by JsonWriterAutoObject(), and ∼JsonWriterAutoObject().

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

- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.12 JsonWriterContext Struct Reference

Used internally by JsonWriter.

```
#include <JsonParserGeneratorRK.h>
```

Public Attributes

- bool isFirst

  True if this the first element in this object or array and doesn't need a comma before it.
- char terminator

  The character that will terminate the object or array when ended.

10.12.1 Detailed Description

Used internally by JsonWriter.

Definition at line 902 of file JsonParserGeneratorRK.h.

10.12.2 Member Data Documentation

10.12.2.1 isFirst

```
bool JsonWriterContext::isFirst
```

True if this the first element in this object or array and doesn't need a comma before it.

Definition at line 903 of file JsonParserGeneratorRK.h.

Referenced by JsonWriter::init(), JsonWriter::insertCheckSeparator(), JsonWriter::setIsFirst(), and JsonWriter↩
::startObjectOrArray().

10.12.2.2 terminator

```
char JsonWriterContext::terminator
```

The character that will terminate the object or array when ended.

Definition at line 904 of file JsonParserGeneratorRK.h.

Referenced by JsonWriter::finishObjectOrArray(), JsonWriter::init(), and JsonWriter::startObjectOrArray().

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

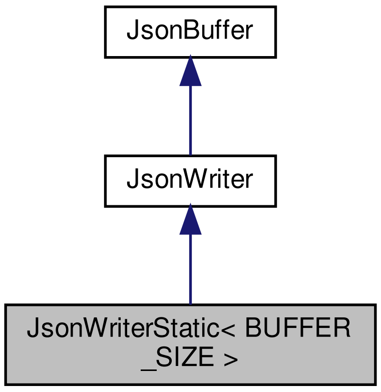
- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.13 JsonWriterStatic< BUFFER_SIZE > Class Template Reference

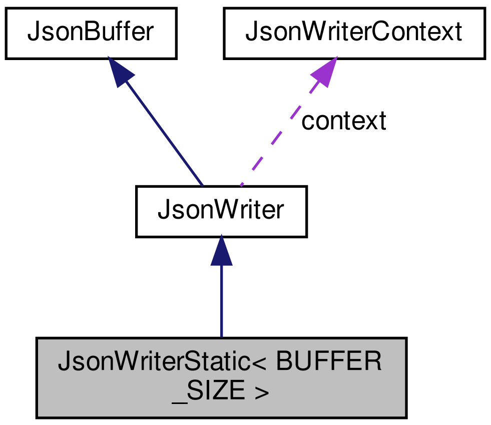
Creates a JsonWriter with a statically allocated buffer.

```
#include <JsonParserGeneratorRK.h>
```

Inheritance diagram for JsonWriterStatic< BUFFER_SIZE >:



Collaboration diagram for JsonWriterStatic< BUFFER_SIZE >:



Public Member Functions

- JsonWriterStatic ()

Private Attributes

- char staticBuffer [BUFFER_SIZE]

  static buffer to write to

Additional Inherited Members

10.13.1 Detailed Description

template<size_t BUFFER_SIZE>
class JsonWriterStatic< BUFFER_SIZE >

Creates a JsonWriter with a statically allocated buffer.

You typically do this when you want to create a buffer as a global variable.

Example:

```
JsonWriterStatic<256> jsonWriter;
```

Creates a 256 byte buffer to write JSON to. You'd normally do this as a global variable, but for smaller buffers (256 and smaller should be fine) in the loop thread, you can allocate one on the stack as a local variable.

Parameters

BUFFER_SIZE	The size of the buffer to reserve.

Definition at line 1241 of file JsonParserGeneratorRK.h.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 JsonWriterStatic()

```
template<size_t BUFFER_SIZE>
JsonWriterStatic< BUFFER_SIZE >::JsonWriterStatic ( )  [inline], [explicit]
```

Definition at line 1243 of file JsonParserGeneratorRK.h.

10.13.3 Member Data Documentation

10.13.3.1 staticBuffer

```
template<size_t BUFFER_SIZE>
char JsonWriterStatic< BUFFER_SIZE >::staticBuffer[BUFFER_SIZE]  [private]
```

static buffer to write to

Definition at line 1243 of file JsonParserGeneratorRK.h.

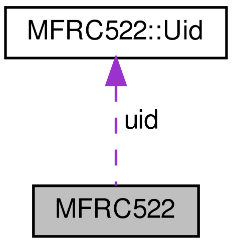
The documentation for this class was generated from the following file:

- lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h

10.14 MFRC522 Class Reference

```
#include <MFRC522.h>
```

Collaboration diagram for MFRC522:



Classes

- struct MIFARE_Key
- struct Uid

Public Types

- enum PCD_Register {
CommandReg = 0x01 << 1, ComIEnReg = 0x02 << 1, DivIEnReg = 0x03 << 1, ComIrqReg = 0x04 << 1,
DivIrqReg = 0x05 << 1, ErrorReg = 0x06 << 1, Status1Reg = 0x07 << 1, Status2Reg = 0x08 << 1,
FIFODataReg = 0x09 << 1, FIFOLevelReg = 0x0A << 1, WaterLevelReg = 0x0B << 1, ControlReg = 0x0C << 1,
BitFramingReg = 0x0D << 1, CollReg = 0x0E << 1, ModeReg = 0x11 << 1, TxModeReg = 0x12 << 1,
RxModeReg = 0x13 << 1, TxControlReg = 0x14 << 1, TxASKReg = 0x15 << 1, TxSelReg = 0x16 << 1,
RxSelReg = 0x17 << 1, RxThresholdReg = 0x18 << 1, DemodReg = 0x19 << 1, MfTxReg = 0x1C << 1,
MfRxReg = 0x1D << 1, SerialSpeedReg = 0x1F << 1, CRCResultRegH = 0x21 << 1, CRCResultRegL =

0x22 << 1,
ModWidthReg = 0x24 << 1, RFCfgReg = 0x26 << 1, GsNReg = 0x27 << 1, CWGsPReg = 0x28 << 1,
ModGsPReg = 0x29 << 1, TModeReg = 0x2A << 1, TPrescalerReg = 0x2B << 1, TReloadRegH = 0x2C << 1,
TReloadRegL = 0x2D << 1, TCounterValueRegH = 0x2E << 1, TCounterValueRegL = 0x2F << 1, Test↩
Sel1Reg = 0x31 << 1,
TestSel2Reg = 0x32 << 1, TestPinEnReg = 0x33 << 1, TestPinValueReg = 0x34 << 1, TestBusReg = 0x35 << 1,
AutoTestReg = 0x36 << 1, VersionReg = 0x37 << 1, AnalogTestReg = 0x38 << 1, TestDAC1Reg = 0x39 << 1,
TestDAC2Reg = 0x3A << 1, TestADCReg = 0x3B << 1 }

- enum PCD_Command {
  PCD_Idle = 0x00, PCD_Mem = 0x01, PCD_GenerateRandomID = 0x02, PCD_CalcCRC = 0x03,
  PCD_Transmit = 0x04, PCD_NoCmdChange = 0x07, PCD_Receive = 0x08, PCD_Transceive = 0x0C,
  PCD_MFAuthent = 0x0E, PCD_SoftReset = 0x0F }
- enum PCD_RxGain {
  RxGain_18dB = 0x00 << 4, RxGain_23dB = 0x01 << 4, RxGain_18dB_2 = 0x02 << 4, RxGain_23dB_2 = 0x03 << 4,
  RxGain_33dB = 0x04 << 4, RxGain_38dB = 0x05 << 4, RxGain_43dB = 0x06 << 4, RxGain_48dB = 0x07 << 4,
  RxGain_min = 0x00 << 4, RxGain_avg = 0x04 << 4, RxGain_max = 0x07 << 4 }
- enum PICC_Command {
  PICC_CMD_REQA = 0x26, PICC_CMD_WUPA = 0x52, PICC_CMD_CT = 0x88, PICC_CMD_SEL_CL1 = 0x93,
  PICC_CMD_SEL_CL2 = 0x95, PICC_CMD_SEL_CL3 = 0x97, PICC_CMD_HLTA = 0x50, PICC_CMD_M↩
  F_AUTH_KEY_A = 0x60,
  PICC_CMD_MF_AUTH_KEY_B = 0x61, PICC_CMD_MF_READ = 0x30, PICC_CMD_MF_WRITE = 0xA0,
  PICC_CMD_MF_DECREMENT = 0xC0,
  PICC_CMD_MF_INCREMENT = 0xC1, PICC_CMD_MF_RESTORE = 0xC2, PICC_CMD_MF_TRANSFER = 0xB0, PICC_CMD_UL_WRITE = 0xA2 }
- enum MIFARE_Misc { MF_ACK = 0xA, MF_KEY_SIZE = 6 }
- enum PICC_Type {
  PICC_TYPE_UNKNOWN = 0, PICC_TYPE_ISO_14443_4 = 1, PICC_TYPE_ISO_18092 = 2, PICC_TYP↩
  E_MIFARE_MINI = 3,
  PICC_TYPE_MIFARE_1K = 4, PICC_TYPE_MIFARE_4K = 5, PICC_TYPE_MIFARE_UL = 6, PICC_TYP↩
  E_MIFARE_PLUS = 7,
  PICC_TYPE_TNP3XXX = 8, PICC_TYPE_NOT_COMPLETE = 255 }
- enum StatusCode {
  STATUS_OK = 1, STATUS_ERROR = 2, STATUS_COLLISION = 3, STATUS_TIMEOUT = 4,
  STATUS_NO_ROOM = 5, STATUS_INTERNAL_ERROR = 6, STATUS_INVALID = 7, STATUS_CRC_W↩
  RONG = 8,
  STATUS_MIFARE_NACK = 9 }

Public Member Functions

- MFRC522 (byte chipSelectPin, byte resetPowerDownPin)
- void setSPIConfig ()
- void PCD_WriteRegister (byte reg, byte value)
- void PCD_WriteRegister (byte reg, byte count, byte ∗values)
- byte PCD_ReadRegister (byte reg)
- void PCD_ReadRegister (byte reg, byte count, byte ∗values, byte rxAlign=0)
- void setBitMask (unsigned char reg, unsigned char mask)
- void PCD_SetRegisterBitMask (byte reg, byte mask)
- void PCD_ClearRegisterBitMask (byte reg, byte mask)
- byte PCD_CalculateCRC (byte ∗data, byte length, byte ∗result)
- void PCD_Init ()

- void PCD_Reset ()
- void PCD_AntennaOn ()
- void PCD_AntennaOff ()
- byte PCD_GetAntennaGain ()
- void PCD_SetAntennaGain (byte mask)
- byte PCD_TransceiveData (byte ∗sendData, byte sendLen, byte ∗backData, byte ∗backLen, byte ∗valid↩ Bits=NULL, byte rxAlign=0, bool checkCRC=false)
- byte PCD_CommunicateWithPICC (byte command, byte waitIRq, byte ∗sendData, byte sendLen, byte ∗backData=NULL, byte ∗backLen=NULL, byte ∗validBits=NULL, byte rxAlign=0, bool checkCRC=false)
- byte PICC_RequestA (byte ∗bufferATQA, byte ∗bufferSize)
- byte PICC_WakeupA (byte ∗bufferATQA, byte ∗bufferSize)
- byte PICC_REQA_or_WUPA (byte command, byte ∗bufferATQA, byte ∗bufferSize)
- byte PICC_Select (Uid ∗uid, byte validBits=0)
- byte PICC_HaltA ()
- byte PCD_Authenticate (byte command, byte blockAddr, MIFARE_Key ∗key, Uid ∗uid)
- void PCD_StopCrypto1 ()
- byte MIFARE_Read (byte blockAddr, byte ∗buffer, byte ∗bufferSize)
- byte MIFARE_Write (byte blockAddr, byte ∗buffer, byte bufferSize)
- byte MIFARE_Decrement (byte blockAddr, long delta)
- byte MIFARE_Increment (byte blockAddr, long delta)
- byte MIFARE_Restore (byte blockAddr)
- byte MIFARE_Transfer (byte blockAddr)
- byte MIFARE_Ultralight_Write (byte page, byte ∗buffer, byte bufferSize)
- byte MIFARE_GetValue (byte blockAddr, long ∗value)
- byte MIFARE_SetValue (byte blockAddr, long value)
- byte PCD_MIFARE_Transceive (byte ∗sendData, byte sendLen, bool acceptTimeout=false)
- const char ∗ GetStatusCodeName (byte code)
- byte PICC_GetType (byte sak)
- const char ∗ PICC_GetTypeName (byte type)
- void PICC_DumpToSerial (Uid ∗uid)
- void PICC_DumpMifareClassicToSerial (Uid ∗uid, byte piccType, MIFARE_Key ∗key)
- void PICC_DumpMifareClassicSectorToSerial (Uid ∗uid, MIFARE_Key ∗key, byte sector)
- void PICC_DumpMifareUltralightToSerial ()
- void MIFARE_SetAccessBits (byte ∗accessBitBuffer, byte g0, byte g1, byte g2, byte g3)
- bool MIFARE_OpenUidBackdoor (bool logErrors)
- bool MIFARE_SetUid (byte ∗newUid, byte uidSize, bool logErrors)
- bool MIFARE_UnbrickUidSector (bool logErrors)
- bool PICC_IsNewCardPresent ()
- bool PICC_ReadCardSerial ()

Public Attributes

- Uid uid

Static Public Attributes

- static const byte FIFO_SIZE = 64

Private Member Functions

- byte MIFARE_TwoStepHelper (byte command, byte blockAddr, long data)

Private Attributes

- byte _chipSelectPin
- byte _resetPowerDownPin

10.14.1 Detailed Description

Definition at line 86 of file MFRC522.h.

10.14.2 Member Enumeration Documentation

10.14.2.1 MIFARE_Misc

`enum MFRC522::MIFARE_Misc`

Enumerator

MF_ACK	
MF_KEY_SIZE	

Definition at line 221 of file MFRC522.h.

10.14.2.2 PCD_Command

`enum MFRC522::PCD_Command`

Enumerator

PCD_Idle	
PCD_Mem	
PCD_GenerateRandomID	
PCD_CalcCRC	
PCD_Transmit	
PCD_NoCmdChange	
PCD_Receive	
PCD_Transceive	
PCD_MFAuthent	
PCD_SoftReset	

Definition at line 165 of file MFRC522.h.

10.14.2.3 PCD_Register

```
enum MFRC522::PCD_Register
```

Enumerator

CommandReg	
ComIEnReg	
DivIEnReg	
ComIrqReg	
DivIrqReg	
ErrorReg	
Status1Reg	
Status2Reg	
FIFODataReg	
FIFOLevelReg	
WaterLevelReg	
ControlReg	
BitFramingReg	
CollReg	
ModeReg	
TxModeReg	
RxModeReg	
TxControlReg	
TxASKReg	
TxSelReg	
RxSelReg	
RxThresholdReg	
DemodReg	
MfTxReg	
MfRxReg	
SerialSpeedReg	
CRCResultRegH	
CRCResultRegL	
ModWidthReg	
RFCfgReg	
GsNReg	
CWGsPReg	
ModGsPReg	
TModeReg	
TPrescalerReg	
TReloadRegH	
TReloadRegL	
TCounterValueRegH	
TCounterValueRegL	
TestSel1Reg	
TestSel2Reg	
TestPinEnReg	
TestPinValueReg	

Enumerator

TestBusReg	
AutoTestReg	
VersionReg	
AnalogTestReg	
TestDAC1Reg	
TestDAC2Reg	
TestADCReg	

Definition at line 90 of file MFRC522.h.

10.14.2.4 PCD_RxGain

```
enum MFRC522::PCD_RxGain
```

Enumerator

RxGain_18dB	
RxGain_23dB	
RxGain_18dB←_2	
RxGain_23dB←_2	
RxGain_33dB	
RxGain_38dB	
RxGain_43dB	
RxGain_48dB	
RxGain_min	
RxGain_avg	
RxGain_max	

Definition at line 180 of file MFRC522.h.

10.14.2.5 PICC_Command

```
enum MFRC522::PICC_Command
```

Enumerator

PICC_CMD_REQA	
PICC_CMD_WUPA	
PICC_CMD_CT	
PICC_CMD_SEL_CL1	
PICC_CMD_SEL_CL2	

Enumerator

PICC_CMD_SEL_CL3	
PICC_CMD_HLTA	
PICC_CMD_MF_AUTH_KEY↩ _A	
PICC_CMD_MF_AUTH_KEY↩ _B	
PICC_CMD_MF_READ	
PICC_CMD_MF_WRITE	
PICC_CMD_MF_DECREMENT	
PICC_CMD_MF_INCREMENT	
PICC_CMD_MF_RESTORE	
PICC_CMD_MF_TRANSFER	
PICC_CMD_UL_WRITE	

Definition at line 195 of file MFRC522.h.

10.14.2.6 PICC_Type

```
enum MFRC522::PICC_Type
```

Enumerator

PICC_TYPE_UNKNOWN	
PICC_TYPE_ISO_14443_4	
PICC_TYPE_ISO_18092	
PICC_TYPE_MIFARE_MINI	
PICC_TYPE_MIFARE_1K	
PICC_TYPE_MIFARE_4K	
PICC_TYPE_MIFARE_UL	
PICC_TYPE_MIFARE_PLUS	
PICC_TYPE_TNP3XXX	
PICC_TYPE_NOT_COMPLETE	

Definition at line 227 of file MFRC522.h.

10.14.2.7 StatusCode

```
enum MFRC522::StatusCode
```

Enumerator

STATUS_OK	
STATUS_ERROR	

Enumerator

STATUS_COLLISION	
STATUS_TIMEOUT	
STATUS_NO_ROOM	
STATUS_INTERNAL_ERROR	
STATUS_INVALID	
STATUS_CRC_WRONG	
STATUS_MIFARE_NACK	

Definition at line 241 of file MFRC522.h.

10.14.3 Constructor & Destructor Documentation

10.14.3.1 MFRC522()

```
MFRC522::MFRC522 (
            byte chipSelectPin,
            byte resetPowerDownPin )
```

Constructor. Prepares the output pins.

Parameters

chipSelectPin	Arduino pin connected to MFRC522's SPI slave select input (Pin 24, NSS, active low)
resetPowerDownPin	Arduino pin connected to MFRC522's reset and power down input (Pin 6, NRSTPD, active low)

Definition at line 18 of file MFRC522.cpp.

10.14.4 Member Function Documentation

10.14.4.1 GetStatusCodeName()

```
const char * MFRC522::GetStatusCodeName (
            byte code )
```

Returns a string pointer to a status code name.

Parameters

code	One of the StatusCode enums.

Definition at line 1077 of file MFRC522.cpp.

10.14.4.2 MIFARE_Decrement()

```
byte MFRC522::MIFARE_Decrement (
             byte blockAddr,
             long delta )
```

MIFARE Decrement subtracts the delta from the value of the addressed block, and stores the result in a volatile memory. For MIFARE Classic only. The sector containing the block must be authenticated before calling this function. Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001]. Use MIFARE_↩Transfer() to store the result in a block.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	The block (0-0xff) number.
delta	This number is subtracted from the value of block blockAddr.

Definition at line 877 of file MFRC522.cpp.

10.14.4.3 MIFARE_GetValue()

```
byte MFRC522::MIFARE_GetValue (
             byte blockAddr,
             long * value )
```

Helper routine to read the current value from a Value Block.

Only for MIFARE Classic and only for blocks in "value block" mode, that is: with access bits [C1 C2 C3] = [110] or [001]. The sector containing the block must be authenticated before calling this function.

Parameters

in	*blockAddr*	The block (0x00-0xff) number.
out	*value*	Current value of the Value Block.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Definition at line 975 of file MFRC522.cpp.

10.14.4.4 MIFARE_Increment()

```
byte MFRC522::MIFARE_Increment (
            byte blockAddr,
            long delta )
```

MIFARE Increment adds the delta to the value of the addressed block, and stores the result in a volatile memory. For MIFARE Classic only. The sector containing the block must be authenticated before calling this function. Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001]. Use MIFARE_Transfer() to store the result in a block.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	The block (0-0xff) number.
delta	This number is added to the value of block blockAddr.

Definition at line 891 of file MFRC522.cpp.

10.14.4.5 MIFARE_OpenUidBackdoor()

```
bool MFRC522::MIFARE_OpenUidBackdoor (
            bool logErrors )
```

Performs the "magic sequence" needed to get Chinese UID changeable Mifare cards to allow writing to sector 0, where the card UID is stored.

Note that you do not need to have selected the card through REQA or WUPA, this sequence works immediately when the card is in the reader vicinity. This means you can use this method even on "bricked" cards that your reader does not recognise anymore (see MFRC522::MIFARE_UnbrickUidSector).

Of course with non-bricked devices, you're free to select them before calling this function.

Definition at line 1445 of file MFRC522.cpp.

Referenced by MIFARE_SetUid(), and MIFARE_UnbrickUidSector().

10.14.4.6 MIFARE_Read()

```
byte MFRC522::MIFARE_Read (
             byte blockAddr,
             byte * buffer,
             byte * bufferSize )
```

Reads 16 bytes (+ 2 bytes CRC_A) from the active PICC.

For MIFARE Classic the sector containing the block must be authenticated before calling this function.

For MIFARE Ultralight only addresses 00h to 0Fh are decoded. The MF0ICU1 returns a NAK for higher addresses. The MF0ICU1 responds to the READ command by sending 16 bytes starting from the page address defined by the command argument. For example; if blockAddr is 03h then pages 03h, 04h, 05h, 06h are returned. A roll-back is implemented: If blockAddr is 0Eh, then the contents of pages 0Eh, 0Fh, 00h and 01h are returned.

The buffer must be at least 18 bytes because a CRC_A is also returned. Checks the CRC_A before returning STATUS_OK.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	MIFARE Classic: The block (0-0xff) number. MIFARE Ultralight: The first page to return data from.
buffer	The buffer to store the data in
bufferSize	Buffer size, at least 18 bytes. Also number of bytes returned if STATUS_OK.

Definition at line 774 of file MFRC522.cpp.

References STATUS_NO_ROOM.

10.14.4.7 MIFARE_Restore()

```
byte MFRC522::MIFARE_Restore (
             byte blockAddr )
```

MIFARE Restore copies the value of the addressed block into a volatile memory. For MIFARE Classic only. The sector containing the block must be authenticated before calling this function. Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001]. Use MIFARE_Transfer() to store the result in a block.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	The block (0-0xff) number.

Definition at line 905 of file MFRC522.cpp.

10.14.4.8 MIFARE_SetAccessBits()

```
void MFRC522::MIFARE_SetAccessBits (
             byte * accessBitBuffer,
             byte g0,
             byte g1,
             byte g2,
             byte g3 )
```

Calculates the bit pattern needed for the specified access bits. In the [C1 C2 C3] tupples C1 is MSB (=4) and C3 is LSB (=1).

Parameters

accessBitBuffer	Pointer to byte 6, 7 and 8 in the sector trailer. Bytes [0..2] will be set.
g0	Access bits [C1 C2 C3] for block 0 (for sectors 0-31) or blocks 0-4 (for sectors 32-39)
g1	Access bits C1 C2 C3] for block 1 (for sectors 0-31) or blocks 5-9 (for sectors 32-39)
g2	Access bits C1 C2 C3] for block 2 (for sectors 0-31) or blocks 10-14 (for sectors 32-39)
g3	Access bits C1 C2 C3] for the sector trailer, block 3 (for sectors 0-31) or block 15 (for sectors 32-39)

Definition at line 1419 of file MFRC522.cpp.

10.14.4.9 MIFARE_SetUid()

```
bool MFRC522::MIFARE_SetUid (
             byte * newUid,
             byte uidSize,
             bool logErrors )
```

Reads entire block 0, including all manufacturer data, and overwrites that block with the new UID, a freshly calculated BCC, and the original manufacturer data.

It assumes a default KEY A of 0xFFFFFFFFFFFF. Make sure to have selected the card before this function is called.

Definition at line 1515 of file MFRC522.cpp.

References MIFARE_OpenUidBackdoor(), PCD_StopCrypto1(), PICC_IsNewCardPresent(), and PICC_Read↵ CardSerial().

10.14.4.10 MIFARE_SetValue()

```
byte MFRC522::MIFARE_SetValue (
             byte blockAddr,
             long value )
```

Helper routine to write a specific value into a Value Block.

Only for MIFARE Classic and only for blocks in "value block" mode, that is: with access bits [C1 C2 C3] = [110] or [001]. The sector containing the block must be authenticated before calling this function.

Parameters

in	*blockAddr*	The block (0x00-0xff) number.
in	*value*	New value of the Value Block.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Definition at line 1000 of file MFRC522.cpp.

10.14.4.11 MIFARE_Transfer()

```
byte MFRC522::MIFARE_Transfer (
            byte blockAddr )
```

MIFARE Transfer writes the value stored in the volatile memory into one MIFARE Classic block. For MIFARE Classic only. The sector containing the block must be authenticated before calling this function. Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001].

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	The block (0-0xff) number.

Definition at line 948 of file MFRC522.cpp.

References STATUS_OK.

10.14.4.12 MIFARE_TwoStepHelper()

```
byte MFRC522::MIFARE_TwoStepHelper (
            byte command,
            byte blockAddr,
            long data )  [private]
```

Helper function for the two-step MIFARE Classic protocol operations Decrement, Increment and Restore.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

command	The command to use
blockAddr	The block (0-0xff) number.
data	The data to transfer in step 2

Definition at line 917 of file MFRC522.cpp.

References STATUS_OK.

10.14.4.13 MIFARE_Ultralight_Write()

```
byte MFRC522::MIFARE_Ultralight_Write (
            byte page,
            byte * buffer,
            byte bufferSize )
```

Writes a 4 byte page to the active MIFARE Ultralight PICC.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

page	The page (2-15) to write to.
buffer	The 4 bytes to write to the PICC
bufferSize	Buffer size, must be at least 4 bytes. Exactly 4 bytes are written.

Definition at line 844 of file MFRC522.cpp.

References STATUS_INVALID, and STATUS_OK.

10.14.4.14 MIFARE_UnbrickUidSector()

```
bool MFRC522::MIFARE_UnbrickUidSector (
            bool logErrors )
```

Resets entire sector 0 to zeroes, so the card can be read again by readers.

Definition at line 1617 of file MFRC522.cpp.

References MIFARE_OpenUidBackdoor().

10.14.4.15 MIFARE_Write()

```
byte MFRC522::MIFARE_Write (
             byte blockAddr,
             byte * buffer,
             byte bufferSize )
```

Writes 16 bytes to the active PICC.

For MIFARE Classic the sector containing the block must be authenticated before calling this function.

For MIFARE Ultralight the opretaion is called "COMPATIBILITY WRITE". Even though 16 bytes are transferred to the Ultralight PICC, only the least significant 4 bytes (bytes 0 to 3) are written to the specified address. It is recommended to set the remaining bytes 04h to 0Fh to all logic 0.

Returns

- STATUS_OK on success, STATUS_??? otherwise.

Parameters

blockAddr	MIFARE Classic: The block (0-0xff) number. MIFARE Ultralight: The page (2-15) to write to.
buffer	The 16 bytes to write to the PICC
bufferSize	Buffer size, must be at least 16 bytes. Exactly 16 bytes are written.

Definition at line 809 of file MFRC522.cpp.

References STATUS_INVALID, and STATUS_OK.

10.14.4.16 PCD_AntennaOff()

```
void MFRC522::PCD_AntennaOff ( )
```

Turns the antenna off by disabling pins TX1 and TX2.

Definition at line 250 of file MFRC522.cpp.

10.14.4.17 PCD_AntennaOn()

```
void MFRC522::PCD_AntennaOn ( )
```

Turns the antenna on by enabling pins TX1 and TX2. After a reset these pins disabled.

Definition at line 240 of file MFRC522.cpp.

Referenced by PCD_Init().

10.14.4.18 PCD_Authenticate()

```
byte MFRC522::PCD_Authenticate (
             byte command,
             byte blockAddr,
             MIFARE_Key * key,
             Uid * uid )
```

Executes the MFRC522 MFAuthent command. This command manages MIFARE authentication to enable a secure communication to any MIFARE Mini, MIFARE 1K and MIFARE 4K card. The authentication is described in the MFRC522 datasheet section 10.3.1.9 and http://www.nxp.com/documents/data_sheet/MF1↵S503x.pdf section 10.1. For use with MIFARE Classic PICCs. The PICC must be selected - ie in state ACTIVE(∗) - before calling this function. Remember to call PCD_StopCrypto1() after communicating with the authenticated P↵ICC - otherwise no new communications can start.

All keys are set to FFFFFFFFFFFFh at chip delivery.

Returns

STATUS_OK on success, STATUS_??? otherwise. Probably STATUS_TIMEOUT if you supply the wrong key.

Parameters

command	PICC_CMD_MF_AUTH_KEY_A or PICC_CMD_MF_AUTH_KEY_B
blockAddr	The block number. See numbering in the comments in the .h file.
key	Pointer to the Crypteo1 key to use (6 bytes)
uid	Pointer to Uid struct. The first 4 bytes of the UID is used.

Definition at line 727 of file MFRC522.cpp.

10.14.4.19 PCD_CalculateCRC()

```
byte MFRC522::PCD_CalculateCRC (
             byte * data,
             byte length,
             byte * result )
```

Use the CRC coprocessor in the MFRC522 to calculate a CRC_A.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

data	In: Pointer to the data to transfer to the FIFO for CRC calculation.
length	In: The number of bytes to transfer.
result	Out: Pointer to result buffer. Result is written to result[0..1], low byte first.

Definition at line 160 of file MFRC522.cpp.

References STATUS_OK, and STATUS_TIMEOUT.

10.14.4.20 PCD_ClearRegisterBitMask()

```
void MFRC522::PCD_ClearRegisterBitMask (
             byte reg,
             byte mask )
```

Clears the bits given in mask from register reg.

Parameters

reg	The register to update. One of the PCD_Register enums.
mask	The bits to clear.

Definition at line 146 of file MFRC522.cpp.

10.14.4.21 PCD_CommunicateWithPICC()

```
byte MFRC522::PCD_CommunicateWithPICC (
             byte command,
             byte waitIRq,
             byte * sendData,
             byte sendLen,
             byte * backData = NULL,
             byte * backLen = NULL,
             byte * validBits = NULL,
             byte rxAlign = 0,
             bool checkCRC = false )
```

Transfers data to the MFRC522 FIFO, executes a commend, waits for completion and transfers data back from the FIFO. CRC validation can only be done if backData and backLen are specified.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

command	The command to execute. One of the PCD_Command enums.
waitIRq	The bits in the ComIrqReg register that signals successful completion of the command.
sendData	Pointer to the data to transfer to the FIFO.
sendLen	Number of bytes to transfer to the FIFO.
backData	NULL or pointer to buffer if data should be read back after executing the command.
backLen	In: Max number of bytes to write to ∗backData. Out: The number of bytes returned.
validBits	In/Out: The number of valid bits in the last byte. 0 for 8 valid bits.
rxAlign	In: Defines the bit position in backData[0] for the first bit received. Default 0.
checkCRC	In: True => The last two bytes of the response is assumed to be a CRC_A that must be validated.

Definition at line 305 of file MFRC522.cpp.

References STATUS_COLLISION, STATUS_CRC_WRONG, STATUS_ERROR, STATUS_MIFARE_NACK, ST↵ATUS_NO_ROOM, STATUS_OK, and STATUS_TIMEOUT.

10.14.4.22 PCD_GetAntennaGain()

```
byte MFRC522::PCD_GetAntennaGain ( )
```

Get the current MFRC522 Receiver Gain (RxGain[2:0]) value. See 9.3.3.6 / table 98 in http://www.nxp.↵com/documents/data_sheet/MFRC522.pdf NOTE: Return value scrubbed with (0x07<<4)=01110000b as RCFfgReg may use reserved bits.

Returns

Value of the RxGain, scrubbed to the 3 bits used.

Definition at line 261 of file MFRC522.cpp.

10.14.4.23 PCD_Init()

```
void MFRC522::PCD_Init ( )
```

Initializes the MFRC522 chip.

Definition at line 198 of file MFRC522.cpp.

References PCD_AntennaOn(), and PCD_Reset().

10.14.4.24 PCD_MIFARE_Transceive()

```
byte MFRC522::PCD_MIFARE_Transceive (
             byte * sendData,
             byte sendLen,
             bool acceptTimeout = false )
```

Wrapper for MIFARE protocol communication. Adds CRC_A, executes the Transceive command and checks that the response is MF_ACK or a timeout.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

sendData	Pointer to the data to transfer to the FIFO. Do NOT include the CRC_A.
sendLen	Number of bytes in sendData.
acceptTimeout	True => A timeout is also success

Definition at line 1032 of file MFRC522.cpp.

References STATUS_ERROR, STATUS_INVALID, STATUS_MIFARE_NACK, and STATUS_OK.

10.14.4.25 PCD_ReadRegister() [1/2]

```
byte MFRC522::PCD_ReadRegister (
            byte reg )
```

Reads a byte from the specified register in the MFRC522 chip. The interface is described in the datasheet section 8.1.2.

Parameters

reg	The register to read from. One of the PCD_Register enums.

Definition at line 83 of file MFRC522.cpp.

10.14.4.26 PCD_ReadRegister() [2/2]

```
void MFRC522::PCD_ReadRegister (
            byte reg,
            byte count,
            byte * values,
            byte rxAlign = 0 )
```

Reads a number of bytes from the specified register in the MFRC522 chip. The interface is described in the datasheet section 8.1.2.

Parameters

reg	The register to read from. One of the PCD_Register enums.
count	The number of bytes to read
values	Byte array to store the values in.
rxAlign	Only bit positions rxAlign..7 in values[0] are updated.

Definition at line 97 of file MFRC522.cpp.

10.14.4.27 PCD_Reset()

```
void MFRC522::PCD_Reset ( )
```

Performs a soft reset on the MFRC522 chip and waits for it to be ready again.

Definition at line 224 of file MFRC522.cpp.

Referenced by PCD_Init().

10.14.4.28 PCD_SetAntennaGain()

```
void MFRC522::PCD_SetAntennaGain (
            byte mask )
```

Set the MFRC522 Receiver Gain (RxGain) to value specified by given mask. See 9.3.3.6 / table 98 in `http://www.nxp.com/documents/data_sheet/MFRC522.pdf` NOTE: Given mask is scrubbed with (0x07<<4)=01110000b as RCFfgReg may use reserved bits.

Definition at line 270 of file MFRC522.cpp.

10.14.4.29 PCD_SetRegisterBitMask()

```
void MFRC522::PCD_SetRegisterBitMask (
            byte reg,
            byte mask )
```

Sets the bits given in mask in register reg.

Parameters

reg	The register to update. One of the PCD_Register enums.
mask	The bits to set.

Definition at line 135 of file MFRC522.cpp.

10.14.4.30 PCD_StopCrypto1()

```
void MFRC522::PCD_StopCrypto1 ( )
```

Used to exit the PCD from its authenticated state. Remember to call this function after communicating with an authenticated PICC - otherwise no new communications can start.

Definition at line 753 of file MFRC522.cpp.

Referenced by MIFARE_SetUid(), and PICC_DumpMifareClassicToSerial().

10.14.4.31 PCD_TransceiveData()

```
byte MFRC522::PCD_TransceiveData (
            byte * sendData,
            byte sendLen,
            byte * backData,
            byte * backLen,
            byte * validBits = NULL,
            byte rxAlign = 0,
            bool checkCRC = false )
```

Executes the Transceive command. CRC validation can only be done if backData and backLen are specified.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

sendData	Pointer to the data to transfer to the FIFO.
sendLen	Number of bytes to transfer to the FIFO.
backData	NULL or pointer to buffer if data should be read back after executing the command.
backLen	In: Max number of bytes to write to *backData. Out: The number of bytes returned.
validBits	In/Out: The number of valid bits in the last byte. 0 for 8 valid bits. Default NULL.
rxAlign	In: Defines the bit position in backData[0] for the first bit received. Default 0.
checkCRC	In: True => The last two bytes of the response is assumed to be a CRC_A that must be validated.

Definition at line 287 of file MFRC522.cpp.

10.14.4.32 PCD_WriteRegister() [1/2]

```
void MFRC522::PCD_WriteRegister (
            byte reg,
            byte value )
```

Writes a byte to the specified register in the MFRC522 chip. The interface is described in the datasheet section 8.1.2.

Parameters

reg	The register to write to. One of the PCD_Register enums.
value	The value to write.

Definition at line 54 of file MFRC522.cpp.

10.14.4.33 PCD_WriteRegister() [2/2]

```
void MFRC522::PCD_WriteRegister (
            byte reg,
            byte count,
            byte * values )
```

Writes a number of bytes to the specified register in the MFRC522 chip. The interface is described in the datasheet section 8.1.2.

Parameters

reg	The register to write to. One of the PCD_Register enums.
count	The number of bytes to write to the register
values	The values to write. Byte array.

Definition at line 67 of file MFRC522.cpp.

10.14.4.34 PICC_DumpMifareClassicSectorToSerial()

```
void MFRC522::PICC_DumpMifareClassicSectorToSerial (
            Uid * uid,
            MIFARE_Key * key,
            byte sector )
```

Dumps memory contents of a sector of a MIFARE Classic PICC. Uses PCD_Authenticate(), MIFARE_Read() and PCD_StopCrypto1. Always uses PICC_CMD_MF_AUTH_KEY_A because only Key A can always read the sector trailer access bits.

Parameters

uid	Pointer to Uid struct returned from a successful PICC_Select().
key	Key A for the sector.
sector	The sector to dump, 0..39.

Definition at line 1249 of file MFRC522.cpp.

10.14.4.35 PICC_DumpMifareClassicToSerial()

```
void MFRC522::PICC_DumpMifareClassicToSerial (
            Uid * uid,
            byte piccType,
            MIFARE_Key * key )
```

Dumps memory contents of a MIFARE Classic PICC. On success the PICC is halted after dumping the data.

Parameters

uid	Pointer to Uid struct returned from a successful PICC_Select().
piccType	One of the PICC_Type enums.
key	Key A used for all sectors.

Definition at line 1208 of file MFRC522.cpp.

References PCD_StopCrypto1().

10.14.4.36 PICC_DumpMifareUltralightToSerial()

```
void MFRC522::PICC_DumpMifareUltralightToSerial ( )
```

Dumps memory contents of a MIFARE Ultralight PICC.

Definition at line 1383 of file MFRC522.cpp.

10.14.4.37 PICC_DumpToSerial()

```
void MFRC522::PICC_DumpToSerial (
            Uid * uid )
```

Dumps debug info about the selected PICC to Serial. On success the PICC is halted after dumping the data. For MIFARE Classic the factory default key of 0xFFFFFFFFFFFF is tried.

Parameters

uid	Pointer to Uid struct returned from a successful PICC_Select().

Definition at line 1154 of file MFRC522.cpp.

10.14.4.38 PICC_GetType()

```
byte MFRC522::PICC_GetType (
            byte sak )
```

Translates the SAK (Select Acknowledge) to a PICC type.

Returns

PICC_Type

Parameters

sak	The SAK byte returned from PICC_Select().

Definition at line 1100 of file MFRC522.cpp.

References PICC_TYPE_ISO_14443_4, PICC_TYPE_ISO_18092, PICC_TYPE_NOT_COMPLETE, and PICC_↩ TYPE_UNKNOWN.

10.14.4.39 PICC_GetTypeName()

```
const char * MFRC522::PICC_GetTypeName (
             byte piccType )
```

Returns a string pointer to the PICC type name.

Parameters

piccType	One of the PICC_Type enums.

Definition at line 1132 of file MFRC522.cpp.

10.14.4.40 PICC_HaltA()

```
byte MFRC522::PICC_HaltA ( )
```

Instructs a PICC in state ACTIVE(∗) to go to state HALT.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Definition at line 682 of file MFRC522.cpp.

References STATUS_ERROR, and STATUS_OK.

10.14.4.41 PICC_IsNewCardPresent()

```
bool MFRC522::PICC_IsNewCardPresent ( )
```

Returns true if a PICC responds to PICC_CMD_REQA. Only "new" cards in state IDLE are invited. Sleeping cards in state HALT are ignored.

Returns

bool

Definition at line 1643 of file MFRC522.cpp.

Referenced by MIFARE_SetUid().

10.14.4.42 PICC_ReadCardSerial()

```
bool MFRC522::PICC_ReadCardSerial ( )
```

Simple wrapper around PICC_Select. Returns true if a UID could be read. Remember to call PICC_IsNewCard↩Present(), PICC_RequestA() or PICC_WakeupA() first. The read UID is available in the class variable uid.

Returns

bool

Definition at line 1658 of file MFRC522.cpp.

Referenced by MIFARE_SetUid().

10.14.4.43 PICC_REQA_or_WUPA()

```
byte MFRC522::PICC_REQA_or_WUPA (
            byte command,
            byte * bufferATQA,
            byte * bufferSize )
```

Transmits REQA or WUPA commands. Beware: When two PICCs are in the field at the same time I often get STATUS_TIMEOUT - probably due do bad antenna design.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

command	The command to send - PICC_CMD_REQA or PICC_CMD_WUPA
bufferATQA	The buffer to store the ATQA (Answer to request) in
bufferSize	Buffer size, at least two bytes. Also number of bytes returned if STATUS_OK.

Definition at line 428 of file MFRC522.cpp.

References STATUS_ERROR, STATUS_NO_ROOM, and STATUS_OK.

10.14.4.44 PICC_RequestA()

```
byte MFRC522::PICC_RequestA (
            byte * bufferATQA,
            byte * bufferSize )
```

Transmits a REQuest command, Type A. Invites PICCs in state IDLE to go to READY and prepare for anticollision or selection. 7 bit frame. Beware: When two PICCs are in the field at the same time I often get STATUS_TIMEOUT - probably due do bad antenna design.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

bufferATQA	The buffer to store the ATQA (Answer to request) in
bufferSize	Buffer size, at least two bytes. Also number of bytes returned if STATUS_OK.

Definition at line 404 of file MFRC522.cpp.

10.14.4.45 PICC_Select()

```
byte MFRC522::PICC_Select (
            Uid * uid,
            byte validBits = 0 )
```

Transmits SELECT/ANTICOLLISION commands to select a single PICC. Before calling this function the PICCs must be placed in the READY(∗) state by calling PICC_RequestA() or PICC_WakeupA(). On success:

- The chosen PICC is in state ACTIVE(∗) and all other PICCs have returned to state IDLE/HALT. (Figure 7 of the ISO/IEC 14443-3 draft.)
- The UID size and value of the chosen PICC is returned in ∗uid along with the SAK.

A PICC UID consists of 4, 7 or 10 bytes. Only 4 bytes can be specified in a SELECT command, so for the longer UIDs two or three iterations are used: UID size Number of UID bytes Cascade levels Example of PICC ======== ================== ============= =============== single 4 1 MIFARE Classic double 7 2 MIFARE Ultralight triple 10 3 Not currently in use?

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

uid	Pointer to Uid struct. Normally output, but can also be used to supply a known UID.
validBits	The number of known UID bits supplied in ∗uid. Normally 0. If set you must also supply uid->size.

Definition at line 467 of file MFRC522.cpp.

References STATUS_COLLISION, STATUS_CRC_WRONG, STATUS_ERROR, STATUS_INTERNAL_ERROR, STATUS_INVALID, and STATUS_OK.

10.14.4.46 PICC_WakeupA()

```
byte MFRC522::PICC_WakeupA (
            byte * bufferATQA,
            byte * bufferSize )
```

Transmits a Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to go to READY(*) and prepare for anticollision or selection. 7 bit frame. Beware: When two PICCs are in the field at the same time I often get STATUS_TIMEOUT - probably due do bad antenna design.

Returns

STATUS_OK on success, STATUS_??? otherwise.

Parameters

bufferATQA	The buffer to store the ATQA (Answer to request) in
bufferSize	Buffer size, at least two bytes. Also number of bytes returned if STATUS_OK.

Definition at line 416 of file MFRC522.cpp.

10.14.4.47 setBitMask()

```
void MFRC522::setBitMask (
            unsigned char reg,
            unsigned char mask )
```

10.14.4.48 setSPIConfig()

```
void MFRC522::setSPIConfig ( )
```

Set SPI bus to work with MFRC522 chip. Please call this function if you have changed the SPI config since the MFRC522 constructor was run.

Definition at line 39 of file MFRC522.cpp.

10.14.5 Member Data Documentation

10.14.5.1 _chipSelectPin

```
byte MFRC522::_chipSelectPin  [private]
```

Definition at line 349 of file MFRC522.h.

10.14.5.2 _resetPowerDownPin

```
byte MFRC522::_resetPowerDownPin  [private]
```

Definition at line 350 of file MFRC522.h.

10.14.5.3 FIFO_SIZE

```
const byte MFRC522::FIFO_SIZE = 64  [static]
```

Definition at line 269 of file MFRC522.h.

10.14.5.4 uid

```
Uid MFRC522::uid
```

Definition at line 266 of file MFRC522.h.

The documentation for this class was generated from the following files:

- lib/MFRC522/src/MFRC522.h
- lib/MFRC522/src/MFRC522.cpp

10.15 MFRC522::MIFARE_Key Struct Reference

```
#include <MFRC522.h>
```

Public Attributes

- byte keyByte [MF_KEY_SIZE]

10.15.1 Detailed Description

Definition at line 261 of file MFRC522.h.

10.15.2 Member Data Documentation

10.15.2.1 keyByte

```
byte MFRC522::MIFARE_Key::keyByte[MF_KEY_SIZE]
```

Definition at line 262 of file MFRC522.h.

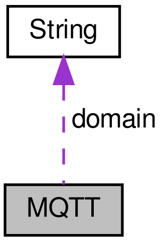
The documentation for this struct was generated from the following file:

- lib/MFRC522/src/MFRC522.h

10.16 MQTT Class Reference

```
#include <MQTT.h>
```

Collaboration diagram for MQTT:



Public Types

- enum EMQTT_QOS { QOS0 = 0, QOS1 = 1, QOS2 = 2 }
- enum MQTT_VERSION { MQTT_V31 = 3, MQTT_V311 = 4 }
- enum EMQTT_CONNACK_RESPONSE {
  CONN_ACCEPT = 0, CONN_UNACCEPTABLE_PROCOTOL = 1, CONN_ID_REJECT = 2, CONN_SER↩
  VER_UNAVAILALE = 3,
  CONN_BAD_USER_PASSWORD = 4, CONN_NOT_AUTHORIZED = 5 }

Public Member Functions

- MQTT ()
- MQTT (char ∗domain, uint16_t port, void(∗callback)(char ∗, uint8_t ∗, unsigned int))
- MQTT (char ∗domain, uint16_t port, void(∗callback)(char ∗, uint8_t ∗, unsigned int), int maxpacketsize)
- MQTT (uint8_t ∗ip, uint16_t port, void(∗callback)(char ∗, uint8_t ∗, unsigned int))
- MQTT (uint8_t ∗ip, uint16_t port, void(∗callback)(char ∗, uint8_t ∗, unsigned int), int maxpacketsize)
- MQTT (char ∗domain, uint16_t port, int keepalive, void(∗callback)(char ∗, uint8_t ∗, unsigned int))
- MQTT (char ∗domain, uint16_t port, int keepalive, void(∗callback)(char ∗, uint8_t ∗, unsigned int), int max-packetsize)

- MQTT (uint8_t ∗ip, uint16_t port, int keepalive, void(∗callback)(char ∗, uint8_t ∗, unsigned int))
- MQTT (uint8_t ∗ip, uint16_t port, int keepalive, void(∗callback)(char ∗, uint8_t ∗, unsigned int), int maxpack-etsize)
- ∼MQTT ()
- void setBroker (char ∗domain, uint16_t port)
- void setBroker (uint8_t ∗ip, uint16_t port)
- bool connect (const char ∗id)
- bool connect (const char ∗id, const char ∗user, const char ∗pass)
- bool connect (const char ∗id, const char ∗user, const char ∗pass, const char ∗willTopic, EMQTT_QOS will←Qos, uint8_t willRetain, const char ∗willMessage, bool cleanSession, MQTT_VERSION version=MQTT_←V311)
- void disconnect ()
- void clear ()
- bool publish (const char ∗topic, const char ∗payload)
- bool publish (const char ∗topic, const char ∗payload, bool retain)
- bool publish (const char ∗topic, const char ∗payload, EMQTT_QOS qos, uint16_t ∗messageid=NULL)
- bool publish (const char ∗topic, const char ∗payload, EMQTT_QOS qos, bool dup, uint16_t ∗messageid=N←ULL)
- bool publish (const char ∗topic, const uint8_t ∗pyaload, unsigned int plength)
- bool publish (const char ∗topic, const uint8_t ∗payload, unsigned int plength, EMQTT_QOS qos, uint16_t ∗messageid=NULL)
- bool publish (const char ∗topic, const uint8_t ∗payload, unsigned int plength, EMQTT_QOS qos, bool dup, uint16_t ∗messageid=NULL)
- bool publish (const char ∗topic, const uint8_t ∗payload, unsigned int plength, bool retain)
- bool publish (const char ∗topic, const uint8_t ∗payload, unsigned int plength, bool retain, EMQTT_QOS qos, uint16_t ∗messageid=NULL)
- bool publish (const char ∗topic, const uint8_t ∗payload, unsigned int plength, bool retain, EMQTT_QOS qos, bool dup, uint16_t ∗messageid)
- void addQosCallback (void(∗qoscallback)(unsigned int))
- bool subscribe (const char ∗topic)
- bool subscribe (const char ∗topic, EMQTT_QOS)
- bool unsubscribe (const char ∗topic)
- bool loop ()
- bool isConnected ()

Private Member Functions

- uint16_t readPacket (uint8_t ∗)
- uint8_t readByte ()
- bool write (uint8_t header, uint8_t ∗buf, uint16_t length)
- uint16_t writeString (const char ∗string, uint8_t ∗buf, uint16_t pos)
- void initialize (char ∗domain, uint8_t ∗ip, uint16_t port, int keepalive, void(∗callback)(char ∗, uint8_t ∗, un-signed int), int maxpacketsize)
- bool publishRelease (uint16_t messageid)
- bool publishComplete (uint16_t messageid)

Private Attributes

- TCPClient _client
- uint8_t ∗ buffer = NULL
- uint16_t nextMsgId
- unsigned long lastOutActivity
- unsigned long lastInActivity

- bool pingOutstanding
- void(∗ callback)(char ∗, uint8_t ∗, unsigned int)
- void(∗ qoscallback)(unsigned int)
- String domain
- uint8_t ∗ ip = NULL
- uint16_t port
- int keepalive
- uint16_t maxpacketsize

10.16.1 Detailed Description

Definition at line 105 of file MQTT.h.

10.16.2 Member Enumeration Documentation

10.16.2.1 EMQTT_CONNACK_RESPONSE

```
enum MQTT::EMQTT_CONNACK_RESPONSE
```

Enumerator

CONN_ACCEPT	
CONN_UNACCEPTABLE_PROCOTOL	
CONN_ID_REJECT	
CONN_SERVER_UNAVAILALE	
CONN_BAD_USER_PASSWORD	
CONN_NOT_AUTHORIZED	

Definition at line 119 of file MQTT.h.

10.16.2.2 EMQTT_QOS

```
enum MQTT::EMQTT_QOS
```

types

Enumerator

QOS0	
QOS1	
QOS2	

Definition at line 108 of file MQTT.h.

10.16.2.3 MQTT_VERSION

```
enum MQTT::MQTT_VERSION
```

Enumerator

MQTT_V31	
MQTT_V311	

Definition at line 114 of file MQTT.h.

10.16.3 Constructor & Destructor Documentation

10.16.3.1 MQTT() [1/9]

```
MQTT::MQTT ( )  [inline]
```

Definition at line 152 of file MQTT.h.

10.16.3.2 MQTT() [2/9]

```
MQTT::MQTT (
             char * domain,
             uint16_t port,
             void(*)(char *, uint8_t *, unsigned int) callback )
```

Definition at line 12 of file MQTT.cpp.

References initialize().

10.16.3.3 MQTT() [3/9]

```
MQTT::MQTT (
             char * domain,
             uint16_t port,
             void(*)(char *, uint8_t *, unsigned int) callback,
             int maxpacketsize )
```

Definition at line 16 of file MQTT.cpp.

References initialize().

10.16.3.4 MQTT() [4/9]

```
MQTT::MQTT (
            uint8_t * ip,
            uint16_t port,
            void(*)(char *, uint8_t *, unsigned int) callback )
```

Definition at line 20 of file MQTT.cpp.

References initialize().

10.16.3.5 MQTT() [5/9]

```
MQTT::MQTT (
            uint8_t * ip,
            uint16_t port,
            void(*)(char *, uint8_t *, unsigned int) callback,
            int maxpacketsize )
```

Definition at line 24 of file MQTT.cpp.

References initialize().

10.16.3.6 MQTT() [6/9]

```
MQTT::MQTT (
            char * domain,
            uint16_t port,
            int keepalive,
            void(*)(char *, uint8_t *, unsigned int) callback )
```

Definition at line 28 of file MQTT.cpp.

References initialize().

10.16.3.7 MQTT() [7/9]

```
MQTT::MQTT (
            char * domain,
            uint16_t port,
            int keepalive,
            void(*)(char *, uint8_t *, unsigned int) callback,
            int maxpacketsize )
```

Definition at line 32 of file MQTT.cpp.

References initialize().

10.16.3.8 MQTT() [8/9]

```
MQTT::MQTT (
            uint8_t * ip,
            uint16_t port,
            int keepalive,
            void(*)(char *, uint8_t *, unsigned int) callback )
```

Definition at line 36 of file MQTT.cpp.

References initialize().

10.16.3.9 MQTT() [9/9]

```
MQTT::MQTT (
            uint8_t * ip,
            uint16_t port,
            int keepalive,
            void(*)(char *, uint8_t *, unsigned int) callback,
            int maxpacketsize )
```

Definition at line 40 of file MQTT.cpp.

References initialize().

10.16.3.10 ∼MQTT()

```
MQTT::∼MQTT ( )
```

Definition at line 44 of file MQTT.cpp.

References buffer, disconnect(), and isConnected().

10.16.4 Member Function Documentation

10.16.4.1 addQosCallback()

```
void MQTT::addQosCallback (
            void(*)(unsigned int) qoscallback )
```

Definition at line 89 of file MQTT.cpp.

References qoscallback.

10.16.4.2 clear()

```
void MQTT::clear ( )
```

Definition at line 534 of file MQTT.cpp.

10.16.4.3 connect() [1/3]

```
bool MQTT::connect (
             const char * id )
```

Definition at line 94 of file MQTT.cpp.

References connect(), and QOS0.

Referenced by reconnect().

10.16.4.4 connect() [2/3]

```
bool MQTT::connect (
             const char * id,
             const char * user,
             const char * pass )
```

Definition at line 98 of file MQTT.cpp.

References connect(), and QOS0.

10.16.4.5 connect() [3/3]

```
bool MQTT::connect (
             const char * id,
             const char * user,
             const char * pass,
             const char * willTopic,
             EMQTT_QOS willQos,
             uint8_t willRetain,
             const char * willMessage,
             bool cleanSession,
             MQTT_VERSION version = MQTT_V311 )
```

Definition at line 102 of file MQTT.cpp.

References buffer, CONN_ACCEPT, isConnected(), keepalive, MQTT_V31, MQTT_V311, nextMsgId, ping↩
Outstanding, readPacket(), write(), and writeString().

Referenced by connect().

10.16.4.6 disconnect()

```
void MQTT::disconnect ( )
```

Definition at line 506 of file MQTT.cpp.

References buffer.

Referenced by setBroker(), and ∼MQTT().

10.16.4.7 initialize()

```
void MQTT::initialize (
             char * domain,
             uint8_t * ip,
             uint16_t port,
             int keepalive,
             void(*)(char *, uint8_t *, unsigned int) callback,
             int maxpacketsize )  [private]
```

Definition at line 53 of file MQTT.cpp.

References buffer, callback, domain, ip, keepalive, maxpacketsize, String::operator=(), port, and qoscallback.

Referenced by MQTT().

10.16.4.8 isConnected()

```
bool MQTT::isConnected ( )
```

Definition at line 528 of file MQTT.cpp.

Referenced by connect(), loop(), loop(), publish(), publishComplete(), publishRelease(), reconnect(), setBroker(), subscribe(), unsubscribe(), and ∼MQTT().

10.16.4.9 loop()

```
bool MQTT::loop ( )
```

Definition at line 240 of file MQTT.cpp.

References buffer, callback, isConnected(), keepalive, lastInActivity, lastOutActivity, pingOutstanding, publish←Complete(), publishRelease(), qoscallback, and readPacket().

Referenced by loop().

10.16.4.10 publish() [1/10]

```
bool MQTT::publish (
             const char * topic,
             const char * payload )
```

Definition at line 339 of file MQTT.cpp.

Referenced by allowUser_callback(), maxCurrentC1_test(), and maxCurrentC2_test().

10.16.4.11 publish() [2/10]

```
bool MQTT::publish (
             const char * topic,
             const char * payload,
             bool retain )
```

Definition at line 343 of file MQTT.cpp.

10.16.4.12 publish() [3/10]

```
bool MQTT::publish (
             const char * topic,
             const char * payload,
             EMQTT_QOS qos,
             uint16_t * messageid = NULL )
```

Definition at line 351 of file MQTT.cpp.

10.16.4.13 publish() [4/10]

```
bool MQTT::publish (
             const char * topic,
             const char * payload,
             EMQTT_QOS qos,
             bool dup,
             uint16_t * messageid = NULL )
```

Definition at line 347 of file MQTT.cpp.

10.16.4.14 publish() [5/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * pyaload,
             unsigned int plength )
```

Definition at line 355 of file MQTT.cpp.

References publish(), and QOS0.

10.16.4.15 publish() [6/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * payload,
             unsigned int plength,
             EMQTT_QOS qos,
             uint16_t * messageid = NULL )
```

Definition at line 363 of file MQTT.cpp.

References publish().

10.16.4.16 publish() [7/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * payload,
             unsigned int plength,
             EMQTT_QOS qos,
             bool dup,
             uint16_t * messageid = NULL )
```

Definition at line 359 of file MQTT.cpp.

References publish().

10.16.4.17 publish() [8/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * payload,
             unsigned int plength,
             bool retain )
```

Definition at line 367 of file MQTT.cpp.

References publish(), and QOS0.

10.16.4.18 publish() [9/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * payload,
             unsigned int plength,
             bool retain,
             EMQTT_QOS qos,
             uint16_t * messageid = NULL )
```

Definition at line 371 of file MQTT.cpp.

References publish().

Referenced by publish().

10.16.4.19 publish() [10/10]

```
bool MQTT::publish (
             const char * topic,
             const uint8_t * payload,
             unsigned int plength,
             bool retain,
             EMQTT_QOS qos,
             bool dup,
             uint16_t * messageid )
```

Definition at line 375 of file MQTT.cpp.

References buffer, isConnected(), maxpacketsize, nextMsgId, QOS1, QOS2, write(), and writeString().

Referenced by publish().

10.16.4.20 publishComplete()

```
bool MQTT::publishComplete (
             uint16_t messageid )  [private]
```

Definition at line 429 of file MQTT.cpp.

References buffer, and isConnected().

Referenced by loop().

10.16.4.21 publishRelease()

```
bool MQTT::publishRelease (
             uint16_t messageid )  [private]
```

Definition at line 416 of file MQTT.cpp.

References buffer, and isConnected().

Referenced by loop().

10.16.4.22 readByte()

```
uint8_t MQTT::readByte ( )  [private]
```

Definition at line 190 of file MQTT.cpp.

Referenced by readPacket().

10.16.4.23 readPacket()

```
uint16_t MQTT::readPacket (
             uint8_t * lengthLength )  [private]
```

Definition at line 195 of file MQTT.cpp.

References buffer, maxpacketsize, and readByte().

Referenced by connect(), and loop().

10.16.4.24 setBroker() [1/2]

```
void MQTT::setBroker (
             char * domain,
             uint16_t port )
```

Definition at line 70 of file MQTT.cpp.

References disconnect(), domain, ip, isConnected(), String::operator=(), and port.

10.16.4.25 setBroker() [2/2]

```
void MQTT::setBroker (
             uint8_t * ip,
             uint16_t port )
```

Definition at line 79 of file MQTT.cpp.

References disconnect(), domain, ip, isConnected(), String::operator=(), and port.

10.16.4.26 subscribe() [1/2]

```
bool MQTT::subscribe (
             const char * topic )
```

Definition at line 469 of file MQTT.cpp.

References QOS0, and subscribe().

Referenced by reconnect().

10.16.4.27 subscribe() [2/2]

```
bool MQTT::subscribe (
             const char * topic,
             EMQTT_QOS qos )
```

Definition at line 473 of file MQTT.cpp.

References buffer, isConnected(), nextMsgId, write(), and writeString().

Referenced by subscribe().

10.16.4.28 unsubscribe()

```
bool MQTT::unsubscribe (
             const char * topic )
```

Definition at line 491 of file MQTT.cpp.

References buffer, isConnected(), nextMsgId, write(), and writeString().

10.16.4.29 write()

```
bool MQTT::write (
            uint8_t header,
            uint8_t * buf,
            uint16_t length )  [private]
```

Definition at line 442 of file MQTT.cpp.

Referenced by connect(), publish(), subscribe(), and unsubscribe().

10.16.4.30 writeString()

```
uint16_t MQTT::writeString (
            const char * string,
            uint8_t * buf,
            uint16_t pos )  [private]
```

Definition at line 514 of file MQTT.cpp.

References maxpacketsize.

Referenced by connect(), publish(), subscribe(), and unsubscribe().

10.16.5 Member Data Documentation

10.16.5.1 _client

```
TCPClient MQTT::_client  [private]
```

Definition at line 129 of file MQTT.h.

10.16.5.2 buffer

```
uint8_t* MQTT::buffer = NULL  [private]
```

Definition at line 130 of file MQTT.h.

Referenced by connect(), disconnect(), initialize(), loop(), publish(), publishComplete(), publishRelease(), read↩
Packet(), subscribe(), unsubscribe(), and ∼MQTT().

10.16.5.3 callback

```
void(* MQTT::callback) (char *, uint8_t *, unsigned int)  [private]
```

Definition at line 135 of file MQTT.h.

Referenced by initialize(), and loop().

10.16.5.4 domain

```
String MQTT::domain  [private]
```

Definition at line 141 of file MQTT.h.

Referenced by initialize(), and setBroker().

10.16.5.5 ip

```
uint8_t* MQTT::ip = NULL  [private]
```

Definition at line 142 of file MQTT.h.

Referenced by initialize(), and setBroker().

10.16.5.6 keepalive

```
int MQTT::keepalive  [private]
```

Definition at line 144 of file MQTT.h.

Referenced by connect(), initialize(), and loop().

10.16.5.7 lastInActivity

```
unsigned long MQTT::lastInActivity  [private]
```

Definition at line 133 of file MQTT.h.

Referenced by loop().

10.16.5.8 lastOutActivity

```
unsigned long MQTT::lastOutActivity [private]
```

Definition at line 132 of file MQTT.h.

Referenced by loop().

10.16.5.9 maxpacketsize

```
uint16_t MQTT::maxpacketsize [private]
```

Definition at line 145 of file MQTT.h.

Referenced by initialize(), publish(), readPacket(), and writeString().

10.16.5.10 nextMsgId

```
uint16_t MQTT::nextMsgId [private]
```

Definition at line 131 of file MQTT.h.

Referenced by connect(), publish(), subscribe(), and unsubscribe().

10.16.5.11 pingOutstanding

```
bool MQTT::pingOutstanding [private]
```

Definition at line 134 of file MQTT.h.

Referenced by connect(), and loop().

10.16.5.12 port

```
uint16_t MQTT::port [private]
```

Definition at line 143 of file MQTT.h.

Referenced by initialize(), and setBroker().

10.16.5.13 qoscallback

```
void(* MQTT::qoscallback) (unsigned int) [private]
```

Definition at line 136 of file MQTT.h.

Referenced by addQosCallback(), initialize(), and loop().

The documentation for this class was generated from the following files:

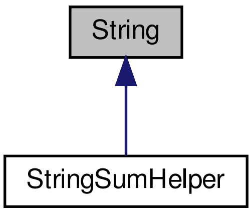
- lib/MQTT/src/MQTT.h
- lib/MQTT/src/MQTT.cpp

10.17 String Class Reference

Wiring String: A class to hold and manipulate a dynamically allocated string.

```
#include <spark_wiring_string.h>
```

Inheritance diagram for String:



Public Member Functions

- String (const char ∗cstr="")

  Construct a String object from a c-string (null-terminated)
- String (const char ∗cstr, unsigned int length)

  Construct a String object from a pointer and length.
- String (const String &str)

  Construct a String object as a copy of another string.
- String (const __FlashStringHelper ∗pstr)
- String (const Printable &printable)

  Construct a String object from any Printable object.
- String (char c)

  Construct a String containing a single character.
- String (unsigned char b, unsigned char base=10)

  Construct a String from a unsigned char (uint8_t) value, expressed as a number.

- String (int value, unsigned char base=10)

  Construct a String from a int (32 bit signed integer) value, expressed as a number.
- String (unsigned int value, unsigned char base=10)

  Construct a String from a unsigned int (32 bit unsigned integer) value, expressed as a number.
- String (long value, unsigned char base=10)

  Construct a String from a long (32 bit signed integer) value, expressed as a number.
- String (unsigned long value, unsigned char base=10)

  Construct a String from a unsigned long (32 bit unsigned integer) value, expressed as a number.
- String (float value, int decimalPlaces=6)

  Construct a String from a float (32 bit single precision floating point) value, expressed as a number.
- String (double value, int decimalPlaces=6)

  Construct a String from a double (64 bit double precision floating point) value, expressed as a number.
- ∼String (void)

  Destructor. Also deletes the underlying dynamically allocated string.
- unsigned char reserve (unsigned int size)

  Reserves a buffer of size.
- unsigned int length (void) const

  Returns the length of the string in bytes.
- String & operator= (const String &rhs)

  Assigns this string to have a copy of String rhs.
- String & operator= (const char ∗cstr)

  Assigns this string to have a copy of c-string (null-terminated) cstr.
- String & operator= (const __FlashStringHelper ∗pstr)
- operator const char ∗ () const

  Returns the contents this String as a c-string (null-terminated)
- unsigned char concat (const String &str)

  Append (concatenate) a String object to the end of this String, modifying this string in place.
- unsigned char concat (const char ∗cstr)

  Append (concatenate) a c-string (null-terminated) to the end of this String, modifying this string in place.
- unsigned char concat (const __FlashStringHelper ∗str)
- unsigned char concat (char c)

  Append (concatenate) a single character to the end of this String, modifying this string in place.
- unsigned char concat (unsigned char c)

  Append (concatenate) the byte value c to the end of this String as a decimal number 0 - 255, modifying this string in place.
- unsigned char concat (int num)

  Append (concatenate) the integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.
- unsigned char concat (unsigned int num)

  Append (concatenate) the unsigned integer value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.
- unsigned char concat (long num)

  Append (concatenate) the long integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.
- unsigned char concat (unsigned long num)

  Append (concatenate) the unsigned long value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.
- unsigned char concat (float num)

  Append (concatenate) the float n to the end of this String as a decimal number (base 10), modifying this string in place.
- unsigned char concat (double num)

Append (concatenate) the double precision float n to the end of this String as a decimal number (base 10), modifying this string in place.

- String & operator+= (const String &rhs)

  Appends (concatenate) a String object to the end of this String, modifying this string in place.
- String & operator+= (const char ∗cstr)

  Appends (concatenate) a c-string (null-terminated) to the end of this String, modifying this string in place.
- String & operator+= (char c)

  Appends (concatenate) a single character to the end of this String, modifying this string in place.
- String & operator+= (unsigned char num)

  Append (concatenate) the byte value num to the end of this String as a decimal number 0 - 255, modifying this string in place.
- String & operator+= (int num)

  Append (concatenate) the integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.
- String & operator+= (unsigned int num)

  Append (concatenate) the unsigned integer value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.
- String & operator+= (long num)

  Append (concatenate) the long integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.
- String & operator+= (unsigned long num)

  Append (concatenate) the unsigned long value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.
- operator StringIfHelperType () const
- int compareTo (const String &s) const

  Compares this string to another string using strcmp (case-sensitive)
- unsigned char equals (const String &s) const

  Returns true if this string is equal to another string (case-sensitive)
- unsigned char equals (const char ∗cstr) const

  Returns true if this string equal to another string (case-sensitive)
- unsigned char operator== (const String &rhs) const

  Returns true if this string is equal to another string (case-sensitive)
- unsigned char operator== (const char ∗cstr) const

  Returns true if this string equal to another string (case-sensitive)
- unsigned char operator!= (const String &rhs) const

  Returns true if this string is greater than to another string (case-sensitive)
- unsigned char operator!= (const char ∗cstr) const

  Returns true if this string not equal to another string (case-sensitive)
- unsigned char operator< (const String &rhs) const

  Returns true if this string is less than to another string (case-sensitive)
- unsigned char operator> (const String &rhs) const

  Returns true if this string is greater than to another string (case-sensitive)
- unsigned char operator<= (const String &rhs) const

  Returns true if this string is less than or equal to another string (case-sensitive)
- unsigned char operator>= (const String &rhs) const

  Returns true if this string is greater than or equal to another string (case-sensitive)
- unsigned char equalsIgnoreCase (const String &s) const

  Returns true if this string equals another string (case-insensitive)
- unsigned char startsWith (const String &prefix) const

  Returns true if this string starts with prefix (case-sensitive)
- unsigned char startsWith (const String &prefix, unsigned int offset) const

  Returns true if this string contains prefix at specified offset (case-sensitive)

- unsigned char endsWith (const String &suffix) const

  Returns true if this string ends with suffix (case-sensitive)
- char charAt (unsigned int index) const

  Gets the character at offset index.
- void setCharAt (unsigned int index, char c)

  Set the character at offset index.
- char operator[ ] (unsigned int index) const

  Gets the character at offset index.
- char & operator[ ] (unsigned int index)

  Set the character at offset index.
- void getBytes (unsigned char ∗buf, unsigned int bufsize, unsigned int index=0) const

  Copy the data out of this String into another buffer.
- void toCharArray (char ∗buf, unsigned int bufsize, unsigned int index=0) const

  Copy the data out of this String into another buffer.
- const char ∗ c_str () const

  Returns a c-string (null-terminated)
- int indexOf (char ch) const

  Search this string for a given character.
- int indexOf (char ch, unsigned int fromIndex) const

  Search this string for a given character starting at an offset.
- int indexOf (const String &str) const

  Search this string for a given String.
- int indexOf (const String &str, unsigned int fromIndex) const

  Search this string for a given String starting at an offset.
- int lastIndexOf (char ch) const

  Search this string for a given character, starting at the end.
- int lastIndexOf (char ch, unsigned int fromIndex) const

  Search this string for a given character, starting at the fromIndex and going toward the beginning.
- int lastIndexOf (const String &str) const

  Search this string for a last occurrence of str.
- int lastIndexOf (const String &str, unsigned int fromIndex) const

  Search this string for a last occurrence of str starting at fromIndex.
- String substring (unsigned int beginIndex) const

  Returns a String object with a copy of the characters starting at beginIndex through the end of the string.
- String substring (unsigned int beginIndex, unsigned int endIndex) const

  Returns a String object with a copy of the characters in the specified range.
- String & replace (char find, char replace)

  Replaces every occurrence of a character in the string with another character, modifying it in place.
- String & replace (const String &find, const String &replace)

  Replaces every occurrence of a String with another String, modifying it in place.
- String & remove (unsigned int index)

  Removes characters from the String, modifying it in place.
- String & remove (unsigned int index, unsigned int count)

  Removes characters from the String, modifying it in place.
- String & toLowerCase (void)

  Converts this String to lower case, modifying it in place.
- String & toUpperCase (void)

  Converts this String to upper case, modifying it in place.
- String & trim (void)

  Removes leading an trailing white spaces from this string, modifying it in place.
- long toInt (void) const

  Converts this string to a signed integer (32-bit)
- float toFloat (void) const

  Converts this string to a float (single precision floating point value)

Static Public Member Functions

- static String format (const char ∗format,...)

  Uses sprintf-style formatting to build a String object [static].

Protected Member Functions

- void init (void)
- void invalidate (void)
- unsigned char changeBuffer (unsigned int maxStrLen)
- unsigned char concat (const char ∗cstr, unsigned int length)
- String & copy (const char ∗cstr, unsigned int length)
- String & copy (const __FlashStringHelper ∗pstr, unsigned int length)

Protected Attributes

- char ∗ buffer

  The buffer containing the data. It is always null-terminated.
- unsigned int capacity

  The capacity of the buffer. The longest string is one byte less than this.
- unsigned int len

  The String length (not counting the null terminator).
- unsigned char flags

  Unused, for future features.

Private Types

- typedef void(String::∗ StringIfHelperType) () const

Private Member Functions

- void StringIfHelper () const

Friends

- class StringPrintableHelper
- StringSumHelper & operator+ (const StringSumHelper &lhs, const String &rhs)

  Append (concatenate) a String to the end of lhs.
- StringSumHelper & operator+ (const StringSumHelper &lhs, const char ∗cstr)

  Append (concatenate) a c-string (null-terminated) to the end of lhs.
- StringSumHelper & operator+ (const StringSumHelper &lhs, char c)

  Append (concatenate) the character c the end of lhs a.
- StringSumHelper & operator+ (const StringSumHelper &lhs, unsigned char num)

  Append (concatenate) the unsigned char num to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, int num)

  Append (concatenate) the signed int num to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, unsigned int num)

  Append (concatenate) the unsigned int num to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, long num)

  Append (concatenate) the long integer num to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, unsigned long num)

  Append (concatenate) the unsigned long integer to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, float num)

  Append (concatenate) the float num to the end of lhs as a decimal number (base 10)
- StringSumHelper & operator+ (const StringSumHelper &lhs, double num)

  Append (concatenate) the double precision float num to the end of lhs as a decimal number (base 10)

10.17.1 Detailed Description

Wiring String: A class to hold and manipulate a dynamically allocated string.

Definition at line 54 of file spark_wiring_string.h.

10.17.2 Member Typedef Documentation

10.17.2.1 StringIfHelperType

```
typedef void(String::* String::StringIfHelperType) () const  [private]
```

Definition at line 59 of file spark_wiring_string.h.

10.17.3 Constructor & Destructor Documentation

10.17.3.1 String() [1/13]

```
String::String (
             const char * cstr = "" )
```

Construct a String object from a c-string (null-terminated)

Parameters

cstr	The string to copy, optional. If not specified, starts with an empty string

Referenced by StringSumHelper::StringSumHelper().

10.17.3.2 String() [2/13]

```
String::String (
             const char * cstr,
             unsigned int length )
```

Construct a String object from a pointer and length.

Parameters

cstr	Pointer to a bytes, typically ASCII or UTF-8. Does not need to be null-terminated.
length	Length in bytes of the string.

10.17.3.3 String() [3/13]

```
String::String (
             const String & str )
```

Construct a String object as a copy of another string.

Parameters

str	The string to copy. Changes made to str in the future won't be reflected in this copy.

Referenced by StringSumHelper::StringSumHelper().

10.17.3.4 String() [4/13]

```
String::String (
             const __FlashStringHelper * pstr )
```

10.17.3.5 String() [5/13]

```
String::String (
             const Printable & printable )
```

Construct a String object from any Printable object.

Parameters

printable	The Printable object. The toPrint() method will be called on it to print to this String the textual representation of the object.

For example, IPAddress is printable, so you can pass an IPAddress to this constructor and this string will contain a textual representation of the IPAddress (dotted quad).

10.17.3.6 String() [6/13]

```
String::String (
             char c )   [explicit]
```

Construct a String containing a single character.

Parameters

c	The character to set the String to

Referenced by StringSumHelper::StringSumHelper().

10.17.3.7 String() [7/13]

```
String::String (
            unsigned char b,
            unsigned char base = 10 )  [explicit]
```

Construct a String from a unsigned char (uint8_t) value, expressed as a number.

Parameters

b	The value.
base	The number base, default is 10 (decimal). Other values include 8 (octal) and 16 (hexadecimal).

Referenced by StringSumHelper::StringSumHelper().

10.17.3.8 String() [8/13]

```
String::String (
            int value,
            unsigned char base = 10 )  [explicit]
```

Construct a String from a int (32 bit signed integer) value, expressed as a number.

Parameters

value	The value.
base	The number base, default is 10 (decimal). Other values include 8 (octal) and 16 (hexadecimal).

Referenced by StringSumHelper::StringSumHelper().

10.17.3.9 String() [9/13]

```
String::String (
            unsigned int value,
            unsigned char base = 10 )  [explicit]
```

Construct a String from a unsigned int (32 bit unsigned integer) value, expressed as a number.

Parameters

value	The value.
base	The number base, default is 10 (decimal). Other values include 8 (octal) and 16 (hexadecimal).

Referenced by maxCurrentC1_test(), maxCurrentC2_test(), and StringSumHelper::StringSumHelper().

10.17.3.10 String() [10/13]

```
String::String (
             long value,
             unsigned char base = 10 )  [explicit]
```

Construct a String from a long (32 bit signed integer) value, expressed as a number.

Parameters

value	The value.
base	The number base, default is 10 (decimal). Other values include 8 (octal) and 16 (hexadecimal).

Referenced by StringSumHelper::StringSumHelper().

10.17.3.11 String() [11/13]

```
String::String (
             unsigned long value,
             unsigned char base = 10 )  [explicit]
```

Construct a String from a unsigned long (32 bit unsigned integer) value, expressed as a number.

Parameters

value	The value.
base	The number base, default is 10 (decimal). Other values include 8 (octal) and 16 (hexadecimal).

Referenced by StringSumHelper::StringSumHelper().

10.17.3.12 String() [12/13]

```
String::String (
             float value,
             int decimalPlaces = 6 )  [explicit]
```

Construct a String from a float (32 bit single precision floating point) value, expressed as a number.

Parameters

value	The value.
decimalPlaces	The number of decimal places to show. Default = 6.

10.17.3.13 String() [13/13]

```
String::String (
             double value,
             int decimalPlaces = 6 )  [explicit]
```

Construct a String from a double (64 bit double precision floating point) value, expressed as a number.

Parameters

value	The value.
decimalPlaces	The number of decimal places to show. Default = 6.

10.17.3.14 ∼String()

```
String::∼String (
             void  )
```

Destructor. Also deletes the underlying dynamically allocated string.

10.17.4 Member Function Documentation

10.17.4.1 c_str()

```
const char* String::c_str ( ) const  [inline]
```

Returns a c-string (null-terminated)

This allows the String object to be passed to anything that requires a c-string. See also operator const char ∗.

One place where you need to explicitly use c_str() or cast is when passing a String as a variable argument to sprintf:

```
String str;
snprintf(buf, sizeof(buf), "string=%s", str.c_str());
```

If you leave off the c_str() the value won't be printed as string. This also applies to things that use sprintf internally, like Log:

```
Log.info("string=%s", str.c_str());
```

This method returns a pointer to the internal buffer. If the underlying string is reallocated because the string is appended to, this pointer will be invalid.

Definition at line 819 of file spark_wiring_string.h.

References buffer.

Referenced by operator const char ∗().

10.17.4.2 changeBuffer()

```
unsigned char String::changeBuffer (
             unsigned int maxStrLen )  [protected]
```

10.17.4.3 charAt()

```
char String::charAt (
             unsigned int index ) const
```

Gets the character at offset index.

Parameters

index	The index to set (0 = first character)

Returns

The character is 0 if the index is larger than the length of the string.

10.17.4.4 compareTo()

```
int String::compareTo (
             const String & s ) const
```

Compares this string to another string using strcmp (case-sensitive)

Parameters

s	the string to compare to

Returns

$<$ 0 if s is less than this, == 0 is s equals this, or $>$ 0 if s is greater than this

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.5 concat() [1/12]

```
unsigned char String::concat (
             const String & str )
```

Append (concatenate) a String object to the end of this String, modifying this string in place.

Parameters

str	The string to copy from. It is not modified.

Returns

true if the append succeeded or false if there was not enough memory or the parameter was invalid.

Referenced by operator+=().

10.17.4.6 concat() [2/12]

```
unsigned char String::concat (
             const char * cstr )
```

Append (concatenate) a c-string (null-terminated) to the end of this String, modifying this string in place.

Parameters

cstr	The string to copy from. It is not modified.

Returns

true if the append succeeded or false if there was not enough memory or the parameter was invalid.

Referenced by operator+=().

10.17.4.7 concat() [3/12]

```
unsigned char String::concat (
             const __FlashStringHelper * str )
```

10.17.4.8 concat() [4/12]

```
unsigned char String::concat (
             char c )
```

Append (concatenate) a single character to the end of this String, modifying this string in place.

Parameters

c	The character to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by operator+=().

10.17.4.9 concat() [5/12]

```
unsigned char String::concat (
             unsigned char c )
```

Append (concatenate) the byte value c to the end of this String as a decimal number 0 - 255, modifying this string in place.

Parameters

c	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by operator+=().

10.17.4.10 concat() [6/12]

```
unsigned char String::concat (
             int num )
```

Append (concatenate) the integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by allowUser_callback(), maxCurrentC1_test(), maxCurrentC2_test(), and operator+=().

10.17.4.11 concat() [7/12]

```
unsigned char String::concat (
             unsigned int num )
```

Append (concatenate) the unsigned integer value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by operator+=().

10.17.4.12 concat() [8/12]

```
unsigned char String::concat (
             long num )
```

Append (concatenate) the long integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by operator+=().

10.17.4.13 concat() [9/12]

```
unsigned char String::concat (
             unsigned long num )
```

Append (concatenate) the unsigned long value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

Referenced by operator+=().

10.17.4.14 concat() [10/12]

```
unsigned char String::concat (
             float num )
```

Append (concatenate) the float n to the end of this String as a decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

10.17.4.15 concat() [11/12]

```
unsigned char String::concat (
             double num )
```

Append (concatenate) the double precision float n to the end of this String as a decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

true if the append succeeded or false if there was not enough memory.

10.17.4.16 concat() [12/12]

```
unsigned char String::concat (
             const char * cstr,
             unsigned int length )  [protected]
```

10.17.4.17 copy() [1/2]

```
String& String::copy (
             const char * cstr,
             unsigned int length )  [protected]
```

10.17.4.18 copy() [2/2]

```
String& String::copy (
             const __FlashStringHelper * pstr,
             unsigned int length )  [protected]
```

10.17.4.19 endsWith()

```
unsigned char String::endsWith (
             const String & suffix ) const
```

Returns true if this string ends with suffix (case-sensitive)

Parameters

suffix	the string containing the suffix to test

Uses the C standard library function strcmp which is case-sensitive and may not work properly with UTF-8 characters.

10.17.4.20 equals() [1/2]

```
unsigned char String::equals (
            const String & s ) const
```

Returns true if this string is equal to another string (case-sensitive)

Parameters

s	the string to compare to

Returns

true if the other string is equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Referenced by operator!=(), and operator==().

10.17.4.21 equals() [2/2]

```
unsigned char String::equals (
            const char * cstr ) const
```

Returns true if this string equal to another string (case-sensitive)

Parameters

cstr	the c-string (null-terminated) to compare to

Returns

true if the other string is equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Referenced by operator!=(), and operator==().

10.17.4.22 equalsIgnoreCase()

```
unsigned char String::equalsIgnoreCase (
             const String & s ) const
```

Returns true if this string equals another string (case-insensitive)

Parameters

s	the string to compare to

Returns

true if equal, false if not

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.23 format()

```
static String String::format (
             const char * format,
              ... ) [static]
```

Uses sprintf-style formatting to build a String object [static].

Parameters

format	The formatting string
...	Variable arguments corresponding to the formatting string

Returns

Returns a String object formatted as specified

10.17.4.24 getBytes()

```
void String::getBytes (
             unsigned char * buf,
             unsigned int bufsize,
             unsigned int index = 0 ) const
```

Copy the data out of this String into another buffer.

Parameters

buf	The buffer to copy into
bufsize	The size of the buffer. The buffer will contain a null-terminted string so the maximum string length is bufsize - 1.
index	The index to start copying from (0 = first character). Optional. Default is from 0, the start of the string.

If bufsize is smaller than the string the string will be truncated and still null-terminated. If the string is truncated and UTF-8, it may break a multi-byte character sequence in the middle, resulting in invalid UTF-8.

Referenced by toCharArray().

10.17.4.25 indexOf() [1/4]

```
int String::indexOf (
            char ch ) const
```

Search this string for a given character.

Parameters

ch	The ASCII character to search for

Returns

index of the character or -1 if not found. 0 = the first character.

This uses the C standard library function strchr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings.

10.17.4.26 indexOf() [2/4]

```
int String::indexOf (
            char ch,
            unsigned int fromIndex ) const
```

Search this string for a given character starting at an offset.

Parameters

ch	The ASCII character to t search for
fromIndex	The index to start from (0 = first character)

Returns

index of the character or -1 if not found. 0 = the first character.

This uses the C standard library function strchr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings.

10.17.4.27 indexOf() [3/4]

```
int String::indexOf (
            const String & str ) const
```

Search this string for a given String.

Parameters

str	The string to search for

Returns

index of the string or -1 if not found. 0 = the first character.

This uses the C standard library function strstr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings. It is case-sensitive.

10.17.4.28 indexOf() [4/4]

```
int String::indexOf (
             const String & str,
             unsigned int fromIndex ) const
```

Search this string for a given String starting at an offset.

Parameters

str	The string to search for
fromIndex	The index to start from (0 = first character)

Returns

index of the string or -1 if not found. 0 = the first character.

This uses the C standard library function strstr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings. It is case-sensitive.

10.17.4.29 init()

```
void String::init (
             void ) [protected]
```

10.17.4.30 invalidate()

```
void String::invalidate (
             void ) [protected]
```

10.17.4.31 lastIndexOf() [1/4]

```
int String::lastIndexOf (
             char ch ) const
```

Search this string for a given character, starting at the end.

Parameters

ch	The ASCII character to search for

Returns

index of the character or -1 if not found. 0 = the first character.

This uses the C standard library function strrchr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings.

10.17.4.32 lastIndexOf() [2/4]

```
int String::lastIndexOf (
             char ch,
             unsigned int fromIndex ) const
```

Search this string for a given character, starting at the fromIndex and going toward the beginning.

Parameters

ch	The ASCII character to search for
fromIndex	The index to start from (0 = first character)

Returns

index of the character or -1 if not found. 0 = the first character.

This uses the C standard library function strrchr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings.

10.17.4.33 lastIndexOf() [3/4]

```
int String::lastIndexOf (
             const String & str ) const
```

Search this string for a last occurrence of str.

Parameters

str	The string to search for

Returns

index of the start of the string or -1 if not found. 0 = the first character.

This uses the C standard library function strstr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings. It is case-sensitive.

10.17.4.34 lastIndexOf() [4/4]

```
int String::lastIndexOf (
            const String & str,
            unsigned int fromIndex ) const
```

Search this string for a last occurrence of str starting at fromIndex.

Parameters

str	The string to search for
fromIndex	The index to start from (0 = first character)

Returns

index of the start of the string or -1 if not found. 0 = the first character.

This uses the C standard library function strstr and is only compatible with ASCII characters. It can return invalid results for UTF-8 strings. It is case-sensitive.

10.17.4.35 length()

```
unsigned int String::length (
            void ) const [inline]
```

Returns the length of the string in bytes.

Note that for UTF-8 strings, this is the number of bytes, not characters.

Definition at line 208 of file spark_wiring_string.h.

References len.

10.17.4.36 operator const char ∗()

```
String::operator const char * ( ) const [inline]
```

Returns the contents this String as a c-string (null-terminated)

See also c_str() which is another way to do this.

Definition at line 241 of file spark_wiring_string.h.

References c_str().

10.17.4.37 operator StringIfHelperType()

```
String::operator StringIfHelperType ( ) const  [inline]
```

Definition at line 536 of file spark_wiring_string.h.

References buffer, and StringIfHelper().

10.17.4.38 operator"!=() [1/2]

```
unsigned char String::operator!= (
             const String & rhs ) const  [inline]
```

Returns true if this string is greater than to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is greater than this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Definition at line 610 of file spark_wiring_string.h.

References equals().

10.17.4.39 operator"!=() [2/2]

```
unsigned char String::operator!= (
             const char * cstr ) const  [inline]
```

Returns true if this string not equal to another string (case-sensitive)

Parameters

cstr	the c-string (null-terminated) to compare to

Returns

true if the other string is not equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Definition at line 622 of file spark_wiring_string.h.

References equals().

10.17.4.40 operator+=() [1/8]

```
String& String::operator+= (
             const String & rhs )  [inline]
```

Appends (concatenate) a String object to the end of this String, modifying this string in place.

Parameters

rhs	The string to copy from. It is not modified.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 352 of file spark_wiring_string.h.

References concat().

10.17.4.41 operator+=() [2/8]

```
String& String::operator+= (
            const char * cstr )  [inline]
```

Appends (concatenate) a c-string (null-terminated) to the end of this String, modifying this string in place.

Parameters

cstr	The string to copy from. It is not modified.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 362 of file spark_wiring_string.h.

References concat().

10.17.4.42 operator+=() [3/8]

```
String& String::operator+= (
            char c )  [inline]
```

Appends (concatenate) a single character to the end of this String, modifying this string in place.

Parameters

c	The character to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 372 of file spark_wiring_string.h.

References concat().

10.17.4.43 operator+=() [4/8]

```
String& String::operator+= (
            unsigned char num )  [inline]
```

Append (concatenate) the byte value num to the end of this String as a decimal number 0 - 255, modifying this string in place.

Parameters

num	The value to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 382 of file spark_wiring_string.h.

References concat().

10.17.4.44 operator+=() [5/8]

```
String& String::operator+= (
            int num )  [inline]
```

Append (concatenate) the integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 392 of file spark_wiring_string.h.

References concat().

10.17.4.45 operator+=() [6/8]

```
String& String::operator+= (
            unsigned int num )  [inline]
```

Append (concatenate) the unsigned integer value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 402 of file spark_wiring_string.h.

References concat().

10.17.4.46 operator+=() [7/8]

```
String& String::operator+= (
            long num )  [inline]
```

Append (concatenate) the long integer value num to the end of this String as a signed decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 412 of file spark_wiring_string.h.

References concat().

10.17.4.47 operator+=() [8/8]

```
String& String::operator+= (
             unsigned long num )  [inline]
```

Append (concatenate) the unsigned long value num to the end of this String as a unsigned decimal number (base 10), modifying this string in place.

Parameters

num	The value to append.

Returns

This string to you can chain operations together. If there was not enough memory or other error occurs, this String will be left unmodified.

Definition at line 422 of file spark_wiring_string.h.

References concat().

10.17.4.48 operator<()

```
unsigned char String::operator< (
             const String & rhs ) const
```

Returns true if this string is less than to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is less than this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.49 operator<=()

```
unsigned char String::operator<= (
             const String & rhs ) const
```

Returns true if this string is less than or equal to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is less than or equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.50 operator=() [1/3]

```
String& String::operator= (
            const String & rhs )
```

Assigns this string to have a copy of String rhs.

Parameters

rhs	The string to copy from.

10.17.4.51 operator=() [2/3]

```
String& String::operator= (
            const char * cstr )
```

Assigns this string to have a copy of c-string (null-terminated) cstr.

Parameters

cstr	The string to copy from.

Referenced by allowUser_callback(), callback(), charToString(), MQTT::initialize(), loop(), and MQTT::setBroker().

10.17.4.52 operator=() [3/3]

```
String& String::operator= (
            const __FlashStringHelper * pstr )
```

10.17.4.53 operator==() [1/2]

```
unsigned char String::operator== (
            const String & rhs ) const  [inline]
```

Returns true if this string is equal to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Definition at line 585 of file spark_wiring_string.h.

References equals().

10.17.4.54 operator==() [2/2]

```
unsigned char String::operator== (
            const char * cstr ) const  [inline]
```

Returns true if this string equal to another string (case-sensitive)

Parameters

cstr	the c-string (null-terminated) to compare to

Returns

true if the other string is equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

Definition at line 597 of file spark_wiring_string.h.

References equals().

Referenced by switchTest().

10.17.4.55 operator>()

```
unsigned char String::operator> (
            const String & rhs ) const
```

Returns true if this string is greater than to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is greater than this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.56 operator>=()

```
unsigned char String::operator>= (
             const String & rhs ) const
```

Returns true if this string is greater than or equal to another string (case-sensitive)

Parameters

rhs	the string to compare to

Returns

true if the other string is greater than or equal to this string.

Uses the C standard library function strcmp which is case-sensitive and does not correctly compare UTF-8 characters.

10.17.4.57 operator[]() [1/2]

```
char String::operator[] (
             unsigned int index ) const
```

Gets the character at offset index.

Parameters

index	The index to set (0 = first character)

Returns

The character is 0 if the index is larger than the length of the string.

10.17.4.58 operator[]() [2/2]

```
char& String::operator[] (
            unsigned int index )
```

Set the character at offset index.

Parameters

index	The index to set (0 = first character)

Returns

A reference to set.

If index is greater than the length of the string, a dummy reference is returned instead. This allows operation to execute without error, but also discards the change. In other words, you cannot use this to append to the string, only modify an existing character.

10.17.4.59 remove() [1/2]

```
String& String::remove (
            unsigned int index )
```

Removes characters from the String, modifying it in place.

Parameters

index	Index to start removing from, inclusive. 0 = first character of the string through the end of the string.

Returns

this String, so you can chain multiple operations

10.17.4.60 remove() [2/2]

```
String& String::remove (
            unsigned int index,
            unsigned int count )
```

Removes characters from the String, modifying it in place.

Parameters

index	Index to start removing from, inclusive. 0 = first character of the string.
count	Number of characters to remove. Typically 1 (remove one character) or more. Removes to the end of the string if count is larger than the size of the string.

Returns

this String, so you can chain multiple operations

10.17.4.61 replace() [1/2]

```
String& String::replace (
            char find,
            char replace )
```

Replaces every occurrence of a character in the string with another character, modifying it in place.

Parameters

find	the character to look for
replace	the character to replace it with

Returns

this String, so you can chain multiple operations

10.17.4.62 replace() [2/2]

```
String& String::replace (
            const String & find,
            const String & replace )
```

Replaces every occurrence of a String with another String, modifying it in place.

Parameters

find	the string to look for (case-sensitive)
replace	the string to replace it with

Returns

this String, so you can chain multiple operations

10.17.4.63 reserve()

```
unsigned char String::reserve (
            unsigned int size )
```

Reserves a buffer of size.

This can improve the efficiency if you know approximately how big your string will be. Otherwise, the string is made larger in increments, which is much less efficient.

If, for example you reserve 100 bytes in a new empty string, the length will still be 0 until you append characters to it. It just will be able to append 100 bytes until it has to expand the internal dynamically allocated buffer.

10.17.4.64 setCharAt()

```
void String::setCharAt (
             unsigned int index,
             char c )
```

Set the character at offset index.

Parameters

index	The index to set (0 = first character)
c	The value to set the character to.

If index is greater than the length of the string, nothing is done. In other words, you cannot use this to append to the string, only modify an existing character.

10.17.4.65 startsWith() [1/2]

```
unsigned char String::startsWith (
             const String & prefix ) const
```

Returns true if this string starts with prefix (case-sensitive)

Parameters

prefix	the string containing the string to test against

Uses the C standard library function strcmp which is case-sensitive and may not work properly with UTF-8 characters.

10.17.4.66 startsWith() [2/2]

```
unsigned char String::startsWith (
             const String & prefix,
             unsigned int offset ) const
```

Returns true if this string contains prefix at specified offset (case-sensitive)

Parameters

prefix	the string containing the string to test against
offset	the offset to check at (0 = first characters)

Uses the C standard library function strcmp which is case-sensitive and may not work properly with UTF-8 characters.

10.17.4.67 StringIfHelper()

```
void String::StringIfHelper ( ) const  [inline], [private]
```

Definition at line 60 of file spark_wiring_string.h.

Referenced by operator StringIfHelperType().

10.17.4.68 substring() [1/2]

```
String String::substring (
             unsigned int beginIndex ) const
```

Returns a String object with a copy of the characters starting at beginIndex through the end of the string.

Parameters

beginIndex	The index to start copying from, inclusive (0 = first byte, 1 = second byte, ...)

Returns

A copy of the specified substring

Note: If the String contains UTF-8 characters, beginIndex and endIndex are in bytes, not characters! It does not prevent splitting a UTF-8 multi-byte sequence.

Referenced by readRFIDCard().

10.17.4.69 substring() [2/2]

```
String String::substring (
             unsigned int beginIndex,
             unsigned int endIndex ) const
```

Returns a String object with a copy of the characters in the specified range.

Parameters

beginIndex	The index to start copying from, inclusive (0 = first byte, 1 = second byte, ...)
endIndex	The index to stop at, exclusive. The last character copied is the one before this one.

Returns

A copy of the specified substring

Note: If the String contains UTF-8 characters, beginIndex and endIndex are in bytes, not characters! It does not prevent splitting a UTF-8 multi-byte sequence.

10.17.4.70 toCharArray()

```
void String::toCharArray (
             char * buf,
             unsigned int bufsize,
             unsigned int index = 0 ) const  [inline]
```

Copy the data out of this String into another buffer.

Parameters

buf	The buffer to copy into
bufsize	The size of the buffer. The buffer will contain a null-terminted string so the maximum string length is bufsize - 1.
index	The index to start copying from (0 = first character). Optional. Default is from 0, the start of the string.

If bufsize is smaller than the string the string will be truncated and still null-terminated. If the string is truncated and UTF-8, it may break a multi-byte character sequence in the middle, resulting in invalid UTF-8.

Definition at line 792 of file spark_wiring_string.h.

References getBytes().

10.17.4.71 toFloat()

```
float String::toFloat (
             void  ) const
```

Converts this string to a float (single precision floating point value)

Returns

a float value or 0.0 if a parsing error occurs (not a float).

10.17.4.72 toInt()

```
long String::toInt (
             void  ) const
```

Converts this string to a signed integer (32-bit)

Returns

An integer value or 0 if a parsing error occurs (not an integer).

Referenced by maxCurrentC1(), and maxCurrentC2().

10.17.4.73 toLowerCase()

```
String& String::toLowerCase (
            void )
```

Converts this String to lower case, modifying it in place.

Returns

this String, so you can chain multiple operations

This is done using the C standard library function tolower() on each character. It only works with 7-bit ASCII characters and will corrupt UTF-8 data.

10.17.4.74 toUpperCase()

```
String& String::toUpperCase (
            void )
```

Converts this String to upper case, modifying it in place.

Returns

this String, so you can chain multiple operations

This is done using the C standard library function toupper() on each character. It only works with 7-bit ASCII characters and will corrupt UTF-8 data.

10.17.4.75 trim()

```
String& String::trim (
            void )
```

Removes leading an trailing white spaces from this string, modifying it in place.

Returns

this String, so you can chain multiple operations

Whitespace is determined by the C standard library function isspace().

10.17.5 Friends And Related Function Documentation

10.17.5.1 operator+ [1/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            const String & rhs ) [friend]
```

Append (concatenate) a String to the end of lhs.

Parameters

lhs	The string to append to. String lhs is not modified.
rhs	The value to append.

Returns

the combined string

10.17.5.2 operator+ [2/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            const char * cstr )  [friend]
```

Append (concatenate) a c-string (null-terminated) to the end of lhs.

Parameters

lhs	The string to append to. String lhs is not modified.
cstr	The value to append.

Returns

the combined string

10.17.5.3 operator+ [3/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            char c )  [friend]
```

Append (concatenate) the character c the end of lhs a.

Parameters

lhs	The string to append to. String lhs is not modified.
c	The character to append

Returns

the combined string

10.17.5.4 operator+ [4/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            unsigned char num )  [friend]
```

Append (concatenate) the unsigned char num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.5 operator+ [5/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            int num )  [friend]
```

Append (concatenate) the signed int num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.6 operator+ [6/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            unsigned int num )  [friend]
```

Append (concatenate) the unsigned int num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.7 operator+ [7/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            long num )  [friend]
```

Append (concatenate) the long integer num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.8 operator+ [8/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            unsigned long num )  [friend]
```

Append (concatenate) the unsigned long integer to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.9 operator+ [9/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            float num )  [friend]
```

Append (concatenate) the float num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.10 operator+ [10/10]

```
StringSumHelper& operator+ (
            const StringSumHelper & lhs,
            double num )  [friend]
```

Append (concatenate) the double precision float num to the end of lhs as a decimal number (base 10)

Parameters

lhs	The string to append to. String lhs is not modified.
num	The value to append.

Returns

the combined string

10.17.5.11 StringPrintableHelper

```
friend class StringPrintableHelper  [friend]
```

Definition at line 1078 of file spark_wiring_string.h.

10.17.6 Member Data Documentation

10.17.6.1 buffer

```
char* String::buffer  [protected]
```

The buffer containing the data. It is always null-terminated.

Definition at line 1058 of file spark_wiring_string.h.

Referenced by c_str(), and operator StringIfHelperType().

10.17.6.2 capacity

```
unsigned int String::capacity  [protected]
```

The capacity of the buffer. The longest string is one byte less than this.

Definition at line 1059 of file spark_wiring_string.h.

10.17.6.3 flags

```
unsigned char String::flags  [protected]
```

Unused, for future features.

Definition at line 1061 of file spark_wiring_string.h.

10.17.6.4 len

```
unsigned int String::len  [protected]
```

The String length (not counting the null terminator).

Definition at line 1060 of file spark_wiring_string.h.

Referenced by length().

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

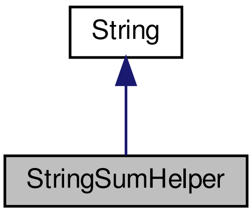
- lib/JsonParserGeneratorRK/docs/src/spark_wiring_string.h

10.18 StringSumHelper Class Reference

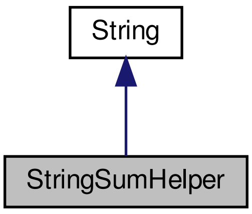
Class used when appending mutiple String and other values using +.

```
#include <spark_wiring_string.h>
```

Inheritance diagram for StringSumHelper:



Collaboration diagram for StringSumHelper:



Public Member Functions

- StringSumHelper (const String &s)
  Append a String object.
- StringSumHelper (const char ∗p)
  Append a const char ∗ (c-string, null terminated)
- StringSumHelper (char c)
  Append a single character.
- StringSumHelper (unsigned char num)
  Append a byte as a decimal number 0 - 255.
- StringSumHelper (int num)
  Append a 32-bit signed integer as a decimal number.
- StringSumHelper (unsigned int num)
  Append a 32-bit unsigned integer as a decimal number.
- StringSumHelper (long num)
  Append a 32-bit long integer as a decimal number.
- StringSumHelper (unsigned long num)
  Append a 32-bit unsigned long as a decimal number.

Additional Inherited Members

10.18.1 Detailed Description

Class used when appending mutiple String and other values using +.

Definition at line 1085 of file spark_wiring_string.h.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 StringSumHelper() [1/8]

```
StringSumHelper::StringSumHelper (
            const String & s )  [inline]
```

Append a String object.

Parameters

s	The string to append.

Returns

StringSumHelper object that encapsulates a copy of that string for appending to another string.

Definition at line 1095 of file spark_wiring_string.h.

References String::String().

10.18.2.2 StringSumHelper() [2/8]

```
StringSumHelper::StringSumHelper (
            const char * p )  [inline]
```

Append a const char ∗ (c-string, null terminated)

Parameters

p	The string to append.

Returns

StringSumHelper object that encapsulates a copy of that string for appending to another string.

Definition at line 1104 of file spark_wiring_string.h.

References String::String().

10.18.2.3 StringSumHelper() [3/8]

```
StringSumHelper::StringSumHelper (
            char c )  [inline]
```

Append a single character.

Parameters

c	The character to append.

Returns

StringSumHelper object that encapsulates a copy of that character for appending to another string.

Definition at line 1113 of file spark_wiring_string.h.

References String::String().

10.18.2.4 StringSumHelper() [4/8]

```
StringSumHelper::StringSumHelper (
             unsigned char num )  [inline]
```

Append a byte as a decimal number 0 - 255.

Parameters

num	The byte value to append.

Returns

StringSumHelper object that encapsulates the textual representation of the number for appending to another string.

Definition at line 1122 of file spark_wiring_string.h.

References String::String().

10.18.2.5 StringSumHelper() [5/8]

```
StringSumHelper::StringSumHelper (
             int num )  [inline]
```

Append a 32-bit signed integer as a decimal number.

Parameters

num	The byte value to append.

Returns

StringSumHelper object that encapsulates the textual representation of the number for appending to another string.

Definition at line 1131 of file spark_wiring_string.h.

References String::String().

10.18.2.6 StringSumHelper() [6/8]

```
StringSumHelper::StringSumHelper (
             unsigned int num )  [inline]
```

Append a 32-bit unsigned integer as a decimal number.

Parameters

num	The byte value to append.

Returns

StringSumHelper object that encapsulates the textual representation of the number for appending to another string.

Definition at line 1140 of file spark_wiring_string.h.

References String::String().

10.18.2.7 StringSumHelper() [7/8]

```
StringSumHelper::StringSumHelper (
             long num )  [inline]
```

Append a 32-bit long integer as a decimal number.

Parameters

num	The byte value to append.

Returns

StringSumHelper object that encapsulates the textual representation of the number for appending to another string.

Definition at line 1149 of file spark_wiring_string.h.

References String::String().

10.18.2.8 StringSumHelper() [8/8]

```
StringSumHelper::StringSumHelper (
             unsigned long num )  [inline]
```

Append a 32-bit unsigned long as a decimal number.

Parameters

num	The byte value to append.

Returns

StringSumHelper object that encapsulates the textual representation of the number for appending to another string.

Definition at line 1158 of file spark_wiring_string.h.

References String::String().

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

- lib/JsonParserGeneratorRK/docs/src/spark_wiring_string.h

10.19 MFRC522::Uid Struct Reference

```
#include <MFRC522.h>
```

Public Attributes

- byte size
- byte uidByte [10]
- byte sak

10.19.1 Detailed Description

Definition at line 254 of file MFRC522.h.

10.19.2 Member Data Documentation

10.19.2.1 sak

```
byte MFRC522::Uid::sak
```

Definition at line 257 of file MFRC522.h.

10.19.2.2 size

```
byte MFRC522::Uid::size
```

Definition at line 255 of file MFRC522.h.

10.19.2.3 uidByte

```
byte MFRC522::Uid::uidByte[10]
```

Definition at line 256 of file MFRC522.h.

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

- lib/MFRC522/src/MFRC522.h

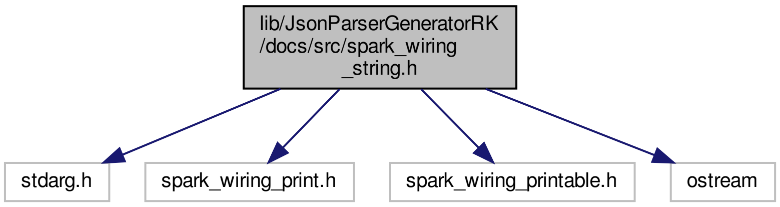
Chapter 11

File Documentation

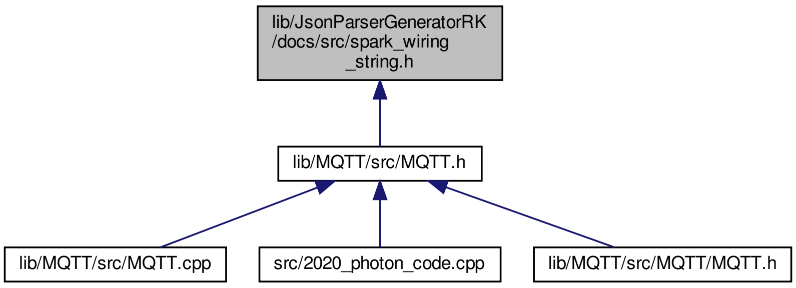
11.1 lib/JsonParserGeneratorRK/docs/src/spark_wiring_string.h File Reference

```
#include <stdarg.h>
#include "spark_wiring_print.h"
#include "spark_wiring_printable.h"
#include <ostream>
```

Include dependency graph for spark_wiring_string.h:



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



Classes

- class String

  Wiring String: A class to hold and manipulate a dynamically allocated string.
- class StringSumHelper

  Class used when appending mutiple String and other values using +.

Macros

- #define F(X) (X)

Functions

- std::ostream & operator<< (std::ostream &os, const String &value)

11.1.1 Macro Definition Documentation

11.1.1.1 F

```
#define F(
              X ) (X)
```

Definition at line 44 of file spark_wiring_string.h.

11.1.2 Function Documentation

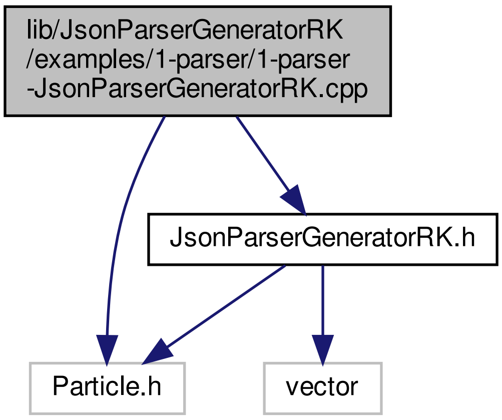
11.1.2.1 operator<<()

```
std::ostream& operator<< (
             std::ostream & os,
             const String & value )
```

11.2 lib/JsonParserGeneratorRK/examples/1-parser/1-parser-JsonParserGenerator←RK.cpp File Reference

```
#include "Particle.h"
#include "JsonParserGeneratorRK.h"
```

Include dependency graph for 1-parser-JsonParserGeneratorRK.cpp:


lib/JsonParserGeneratorRK
/examples/1-parser/1-parser
-JsonParserGeneratorRK.cpp
JsonParserGeneratorRK.h
Particle.h
vector

Functions

- void runTest ()
- void setup ()

  Inital setup for pin assignments and serial links start.
- void loop ()

  Main running function that executes all other functions; runs over 5times/second.

Variables

- const unsigned long TEST_RUN_PERIOD_MS = 10000
- unsigned long lastRun = 0
- const char ∗const test2 = "{\"t1\":\"abc\",\"t2\":1234,\"t3\":1234.5,\"t4\":true,\"t5\":false,\"t6\":null, \"t7\" ←: \"\\\"quoted\\\"\" } "
- JsonParserStatic< 256, 20 > parser1

11.2.1 Function Documentation

11.2.1.1 loop()

```
void loop ( )
```

Main running function that executes all other functions; runs over 5times/second.

Definition at line 19 of file 1-parser-JsonParserGeneratorRK.cpp.

References runTest().

11.2.1.2 runTest()

```
void runTest ( )
```

Definition at line 26 of file 1-parser-JsonParserGeneratorRK.cpp.

11.2.1.3 setup()

```
void setup ( )
```

Inital setup for pin assignments and serial links start.

Definition at line 15 of file 1-parser-JsonParserGeneratorRK.cpp.

11.2.2 Variable Documentation

11.2.2.1 lastRun

```
unsigned long lastRun = 0
```

Definition at line 6 of file 1-parser-JsonParserGeneratorRK.cpp.

11.2.2.2 parser1

```
JsonParserStatic<256, 20> parser1
```

Definition at line 13 of file 1-parser-JsonParserGeneratorRK.cpp.

11.2.2.3 test2

```
const char* const test2 = "{\"t1\":\"abc\",\"t2\":1234,\"t3\":1234.5,\"t4\":true,\"t5\"↩
:false,\"t6\":null, \"t7\" :  \"\\\"quoted\\\"\" } "
```

Definition at line 10 of file 1-parser-JsonParserGeneratorRK.cpp.

11.2.2.4 TEST_RUN_PERIOD_MS

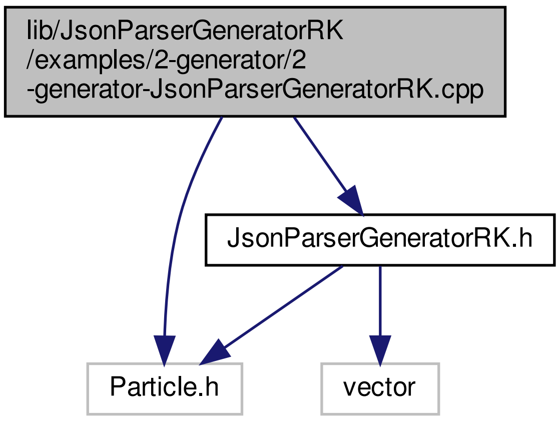
```
const unsigned long TEST_RUN_PERIOD_MS = 10000
```

Definition at line 5 of file 1-parser-JsonParserGeneratorRK.cpp.

11.3 lib/JsonParserGeneratorRK/examples/2-generator/2-generator-JsonParserGenerator↩ RK.cpp File Reference

```
#include "Particle.h"
#include "JsonParserGeneratorRK.h"
```

Include dependency graph for 2-generator-JsonParserGeneratorRK.cpp:


lib/JsonParserGeneratorRK /examples/2-generator/2 -generator-JsonParserGeneratorRK.cpp
JsonParserGeneratorRK.h
Particle.h
vector

Functions

- void runTest ()
- void setup ()
- void loop ()

Variables

- const unsigned long TEST_RUN_PERIOD_MS = 10000
- unsigned long lastRun = 0

11.3.1 Function Documentation

11.3.1.1 loop()

```
void loop ( )
```

Definition at line 15 of file 2-generator-JsonParserGeneratorRK.cpp.

References runTest().

11.3.1.2 runTest()

```
void runTest ( )
```

Definition at line 22 of file 2-generator-JsonParserGeneratorRK.cpp.

Referenced by loop().

11.3.1.3 setup()

```
void setup ( )
```

Definition at line 11 of file 2-generator-JsonParserGeneratorRK.cpp.

11.3.2 Variable Documentation

11.3.2.1 lastRun

```
unsigned long lastRun = 0
```

Definition at line 6 of file 2-generator-JsonParserGeneratorRK.cpp.

11.3.2.2 TEST_RUN_PERIOD_MS

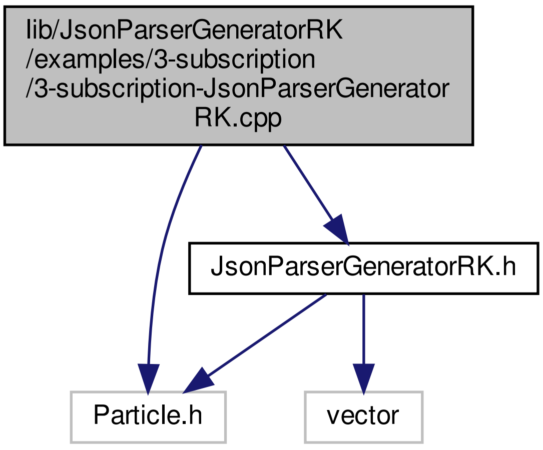
```
const unsigned long TEST_RUN_PERIOD_MS = 10000
```

Definition at line 5 of file 2-generator-JsonParserGeneratorRK.cpp.

11.4 lib/JsonParserGeneratorRK/examples/3-subscription/3-subscription-JsonParser↩GeneratorRK.cpp File Reference

```
#include "Particle.h"
#include "JsonParserGeneratorRK.h"
```

Include dependency graph for 3-subscription-JsonParserGeneratorRK.cpp:


lib/JsonParserGeneratorRK /examples/3-subscription /3-subscription-JsonParserGenerator RK.cpp
JsonParserGeneratorRK.h
Particle.h
vector

Functions

- void subscriptionHandler (const char ∗event, const char ∗data)
- void printJson (JsonParser &jp)
- void setup ()
- void loop ()
- void printIndent (size_t indent)
- void printString (const char ∗str)
- void printJsonInner (JsonParser &jp, const JsonParserGeneratorRK::jsmntok_t ∗container, size_t indent)

Variables

- JsonParserStatic< 2048, 100 > jsonParser

11.4.1 Function Documentation

11.4.1.1 loop()

```
void loop ( )
```

Definition at line 22 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.4.1.2 printIndent()

```
void printIndent (
             size_t indent )
```

Definition at line 47 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.4.1.3 printJson()

```
void printJson (
             JsonParser & jp )
```

Definition at line 149 of file 3-subscription-JsonParserGeneratorRK.cpp.

References JsonParser::getOuterToken(), and printJsonInner().

11.4.1.4 printJsonInner()

```
void printJsonInner (
             JsonParser & jp,
             const JsonParserGeneratorRK::jsmntok_t * container,
             size_t indent )
```

Definition at line 75 of file 3-subscription-JsonParserGeneratorRK.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParser::getKeyValueTokenByIndex(), JsonParser::get↩ ValueTokenByIndex(), JsonParserGeneratorRK::JSMN_ARRAY, JsonParserGeneratorRK::JSMN_OBJECT, Json↩ ParserGeneratorRK::JSMN_PRIMITIVE, JsonParserGeneratorRK::JSMN_STRING, JsonParserGeneratorRK::J↩ SMN_UNDEFINED, printIndent(), printJsonInner(), JsonParserGeneratorRK::jsmntok_t::start, and JsonParser↩ GeneratorRK::jsmntok_t::type.

11.4.1.5 printString()

```
void printString (
             const char * str )
```

Definition at line 53 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.4.1.6 setup()

```
void setup ( )
```

Definition at line 17 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.4.1.7 subscriptionHandler()

```
void subscriptionHandler (
             const char * event,
             const char * data )
```

Definition at line 25 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.4.2 Variable Documentation

11.4.2.1 jsonParser

```
JsonParserStatic<2048, 100> jsonParser
```

Definition at line 9 of file 3-subscription-JsonParserGeneratorRK.cpp.

11.5 lib/JsonParserGeneratorRK/README.md File Reference

11.6 lib/MFRC522/README.md File Reference

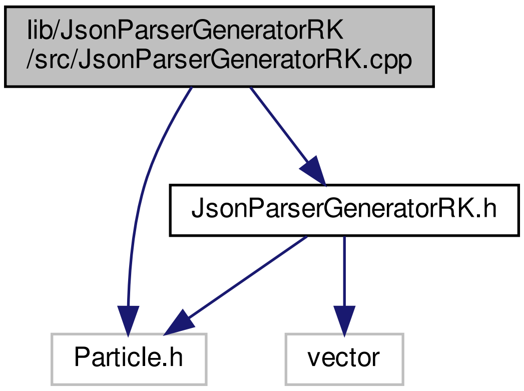
11.7 lib/MQTT/README.md File Reference

11.8 README.md File Reference

11.9 lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.cpp File Reference

```
#include "Particle.h"
#include "JsonParserGeneratorRK.h"
```

Include dependency graph for JsonParserGeneratorRK.cpp:



Namespaces

- JsonParserGeneratorRK

Functions

- static jsmntok_t ∗ JsonParserGeneratorRK::jsmn_alloc_token (jsmn_parser ∗parser, jsmntok_t ∗tokens, size_t num_tokens)
- static void JsonParserGeneratorRK::jsmn_fill_token (jsmntok_t ∗token, jsmntype_t type, int start, int end)
- static int JsonParserGeneratorRK::jsmn_parse_primitive (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, size_t num_tokens)
- static int JsonParserGeneratorRK::jsmn_parse_string (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, size_t num_tokens)
- int JsonParserGeneratorRK::jsmn_parse (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, unsigned int num_tokens)

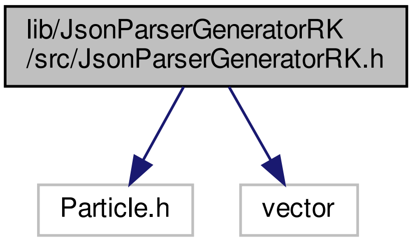
  Run JSON parser.
- void JsonParserGeneratorRK::jsmn_init (jsmn_parser ∗parser)

  Create JSON parser over an array of tokens.

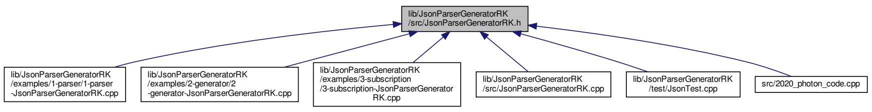
11.10 lib/JsonParserGeneratorRK/src/JsonParserGeneratorRK.h File Reference

```
#include "Particle.h"
#include <vector>
```

Include dependency graph for JsonParserGeneratorRK.h:



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


lib/JsonParserGeneratorRK
/src/JsonParserGeneratorRK.h
lib/JsonParserGeneratorRK
/examples/1-parser/1-parser
-JsonParserGeneratorRK.cpp
lib/JsonParserGeneratorRK
/examples/2-generator/2
-generator-JsonParserGeneratorRK.cpp
lib/JsonParserGeneratorRK
/examples/3-subscription
/3-subscription-JsonParserGenerator
RK.cpp
lib/JsonParserGeneratorRK
/src/JsonParserGeneratorRK.cpp
lib/JsonParserGeneratorRK
/test/JsonTest.cpp
src/2020_photon_code.cpp

Classes

- struct JsonParserGeneratorRK::jsmntok_t

    JSON token description.
- struct JsonParserGeneratorRK::jsmn_parser

    JSON parser.
- class JsonParserString

    Class used internally for writing to strings.
- class JsonBuffer

    Base class for managing a static or dynamic buffer, used by both JsonParser and JsonWriter.
- class JsonParser

    API to the JsonParser.
- class JsonParserStatic< BUFFER_SIZE, MAX_TOKENS >

    Creates a JsonParser with a static buffer.
- class JsonReference

    This class provides a fluent-style API for easily traversing a tree of JSON objects to find a value.
- struct JsonWriterContext

    Used internally by JsonWriter.
- class JsonWriter

    Class for building a JSON string.
- class JsonWriterStatic< BUFFER_SIZE >

Creates a JsonWriter with a statically allocated buffer.

- class JsonWriterAutoObject

  Class for creating a JSON object with JsonWriter.

- class JsonWriterAutoArray

  Class for creating a JSON array with JsonWriter.

- class JsonModifier

  Class for modifying a JSON object in place, without needing to make a copy of it.

Namespaces

- JsonParserGeneratorRK

Enumerations

- enum JsonParserGeneratorRK::jsmntype_t {
  JsonParserGeneratorRK::JSMN_UNDEFINED = 0, JsonParserGeneratorRK::JSMN_OBJECT = 1, Json↩
  ParserGeneratorRK::JSMN_ARRAY = 2, JsonParserGeneratorRK::JSMN_STRING = 3,
  JsonParserGeneratorRK::JSMN_PRIMITIVE = 4 }

  JSON type identifier (object, array, string, primitive)

- enum JsonParserGeneratorRK::jsmnerr { JsonParserGeneratorRK::JSMN_ERROR_NOMEM = -1, Json↩
  ParserGeneratorRK::JSMN_ERROR_INVAL = -2, JsonParserGeneratorRK::JSMN_ERROR_PART = -3 }

  JSMN error codes.

Functions

- void JsonParserGeneratorRK::jsmn_init (jsmn_parser ∗parser)

  Create JSON parser over an array of tokens.

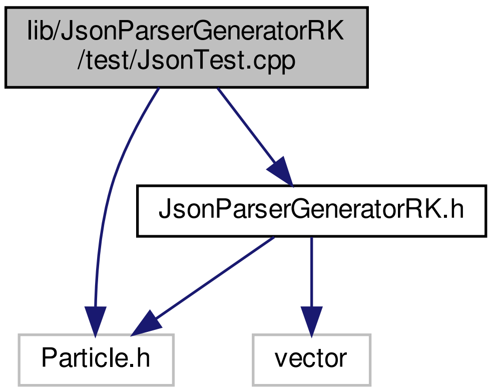
- int JsonParserGeneratorRK::jsmn_parse (jsmn_parser ∗parser, const char ∗js, size_t len, jsmntok_t ∗tokens, unsigned int num_tokens)

  Run JSON parser.

11.11 lib/JsonParserGeneratorRK/test/JsonTest.cpp File Reference

```
#include "Particle.h"
#include "JsonParserGeneratorRK.h"
```

Include dependency graph for JsonTest.cpp:



Macros

- #define assertJsonParserBuffer(jp, expected) _assertJsonParserBuffer(jp, expected, __LINE__)
- #define assertJsonWriterBuffer(jw, expected) _assertJsonWriterBuffer(jw, expected, __LINE__)

Functions

- void printTokens (JsonParser &jp)
- void printToken (JsonParser &jp, const JsonParserGeneratorRK::jsmntok_t ∗tok)
- void printJson (JsonParser &jp)
- char ∗ readTestData (const char ∗filename)
- void _assertJsonParserBuffer (JsonParser &jp, const char ∗expected, size_t line)
- void _assertJsonWriterBuffer (JsonWriter &jw, const char ∗expected, size_t line)
- int main (int argc, char ∗argv[ ])
- void printIndent (size_t indent)
- void printString (const char ∗str)
- void printJsonInner (JsonParser &jp, const JsonParserGeneratorRK::jsmntok_t ∗container, size_t indent)

11.11.1 Macro Definition Documentation

11.11.1.1 assertJsonParserBuffer

```
#define assertJsonParserBuffer(
            jp,
            expected ) _assertJsonParserBuffer(jp, expected, __LINE__)
```

Definition at line 44 of file JsonTest.cpp.

11.11.1.2 assertJsonWriterBuffer

```
#define assertJsonWriterBuffer(
              jw,
              expected ) _assertJsonWriterBuffer(jw, expected, __LINE__)
```

Definition at line 61 of file JsonTest.cpp.

11.11.2 Function Documentation

11.11.2.1 _assertJsonParserBuffer()

```
void _assertJsonParserBuffer (
             JsonParser & jp,
             const char * expected,
             size_t line )
```

Definition at line 30 of file JsonTest.cpp.

References JsonBuffer::getOffset().

11.11.2.2 _assertJsonWriterBuffer()

```
void _assertJsonWriterBuffer (
             JsonWriter & jw,
             const char * expected,
             size_t line )
```

Definition at line 47 of file JsonTest.cpp.

References JsonBuffer::getOffset().

11.11.2.3 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 65 of file JsonTest.cpp.

References JsonBuffer::addData(), JsonBuffer::addString(), JsonBuffer::allocate(), JsonModifier::finish(), Json↵Writer::finishObjectOrArray(), JsonParser::getKeyValueTokenByIndex(), JsonParser::getOuterObject(), Json↵Parser::getOuterToken(), JsonParser::getTokenJsonString(), JsonParser::getTokenValue(), JsonParser::get↵ValueTokenByKey(), JsonWriter::insertCheckSeparator(), JsonWriter::insertKeyArray(), JsonWriter::insertString(), JsonWriter::insertValue(), JsonWriter::JsonWriter(), JsonParser::parse(), readTestData(), JsonModifier::remove↵ArrayIndex(), JsonModifier::startAppend(), JsonModifier::startModify(), and JsonWriter::startObject().

11.11.2.4 printIndent()

```
void printIndent (
            size_t indent )
```

Definition at line 1861 of file JsonTest.cpp.

Referenced by printJsonInner().

11.11.2.5 printJson()

```
void printJson (
            JsonParser & jp )
```

Definition at line 1963 of file JsonTest.cpp.

References JsonParser::getOuterToken(), and printJsonInner().

11.11.2.6 printJsonInner()

```
void printJsonInner (
            JsonParser & jp,
            const JsonParserGeneratorRK::jsmntok_t * container,
            size_t indent )
```

Definition at line 1889 of file JsonTest.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParser::getKeyValueTokenByIndex(), JsonParser::get←ValueTokenByIndex(), JsonParserGeneratorRK::JSMN_ARRAY, JsonParserGeneratorRK::JSMN_OBJECT, Json←ParserGeneratorRK::JSMN_PRIMITIVE, JsonParserGeneratorRK::JSMN_STRING, JsonParserGeneratorRK::J←SMN_UNDEFINED, printIndent(), printJsonInner(), JsonParserGeneratorRK::jsmntok_t::start, and JsonParser←GeneratorRK::jsmntok_t::type.

Referenced by printJson(), and printJsonInner().

11.11.2.7 printString()

```
void printString (
            const char * str )
```

Definition at line 1867 of file JsonTest.cpp.

11.11.2.8 printToken()

```
void printToken (
            JsonParser & jp,
            const JsonParserGeneratorRK::jsmntok_t * tok )
```

Definition at line 1827 of file JsonTest.cpp.

References JsonParserGeneratorRK::jsmntok_t::end, JsonParserGeneratorRK::JSMN_ARRAY, JsonParser←GeneratorRK::JSMN_OBJECT, JsonParserGeneratorRK::JSMN_PRIMITIVE, JsonParserGeneratorRK::JSMN←_STRING, JsonParserGeneratorRK::JSMN_UNDEFINED, JsonParserGeneratorRK::jsmntok_t::start, and Json←ParserGeneratorRK::jsmntok_t::type.

Referenced by printTokens().

11.11.2.9 printTokens()

```
void printTokens (
            JsonParser & jp )
```

Definition at line 1818 of file JsonTest.cpp.

References JsonParser::getTokens(), JsonParser::getTokensEnd(), and printToken().

11.11.2.10 readTestData()

```
char* readTestData (
            const char * filename )
```

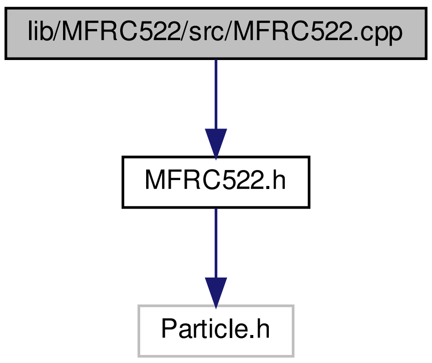
Definition at line 8 of file JsonTest.cpp.

Referenced by main().

11.12 lib/MFRC522/src/MFRC522.cpp File Reference

```
#include "MFRC522.h"
```

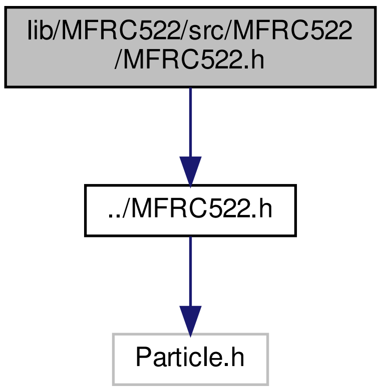
Include dependency graph for MFRC522.cpp:



11.13 lib/MFRC522/src/MFRC522/MFRC522.h File Reference

```
#include "../MFRC522.h"
```

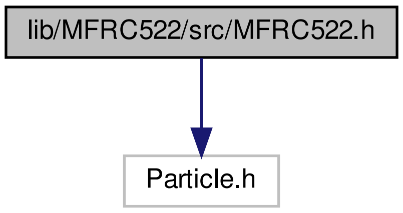
Include dependency graph for MFRC522.h:



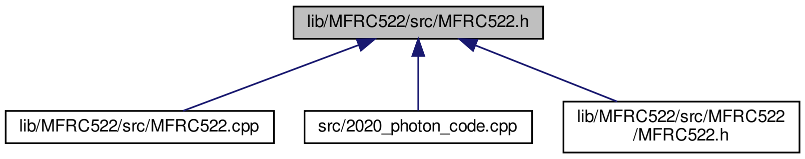
11.14 lib/MFRC522/src/MFRC522.h File Reference

```
#include "Particle.h"
```

Include dependency graph for MFRC522.h:



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



Classes

- class MFRC522
- struct MFRC522::Uid
- struct MFRC522::MIFARE_Key

Typedefs

- typedef uint16_t word

11.14.1 Typedef Documentation

11.14.1.1 word

```
typedef uint16_t word
```

MFRC522.h - Library to use ARDUINO RFID MODULE KIT 13.56 MHZ WITH TAGS SPI W AND R BY COOQR↩OBOT. Based on code Dr.Leong (WWW.B2CQSHOP.COM) Created by Miguel Balboa (circuitito.com), Jan, 2012. Rewritten by Søren Thing Andersen (access.thing.dk), fall of 2013 (Translation to English, refactored, comments, anti collision, cascade levels.) Released into the public domain.

Please read this file for an overview and then MFRC522.cpp for comments on the specific functions. Search for "mf-rc522" on ebay.com to purchase the MF-RC522 board.

There are three hardware components involved: 1) The micro controller: An Arduino 2) The PCD (short for Proximity Coupling Device): NXP MFRC522 Contactless Reader IC 3) The PICC (short for Proximity Integrated Circuit Card): A card or tag using the ISO 14443A interface, eg Mifare or NTAG203.

The microcontroller and card reader uses SPI for communication. The protocol is described in the MFRC522 datasheet: `http://www.nxp.com/documents/data_sheet/MFRC522.pdf`

The card reader and the tags communicate using a 13.56MHz electromagnetic field. The protocol is defined in ISO/IEC 14443-3 Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anticollision". A free version of the final draft can be found at `http://wg8.de/wg8n1496_17n3613_↩Ballot_FCD14443-3.pdf` Details are found in chapter 6, Type A – Initialization and anticollision.

If only the PICC UID is wanted, the above documents has all the needed information. To read and write from MIFA↩
RE PICCs, the MIFARE protocol is used after the PICC has been selected. The MIFARE Classic chips and protocol is described in the datasheets: 1K: `http://www.nxp.com/documents/data_sheet/MF1S503x.↩
pdf` 4K: `http://www.nxp.com/documents/data_sheet/MF1S703x.pdf` Mini: `http://www.↩
idcardmarket.com/download/mifare_S20_datasheet.pdf` The MIFARE Ultralight chip and protocol is described in the datasheets: Ultralight: `http://www.nxp.com/documents/data_sheet/MF0IC↩
U1.pdf` Ultralight C: `http://www.nxp.com/documents/short_data_sheet/MF0ICU2_SDS.pdf`

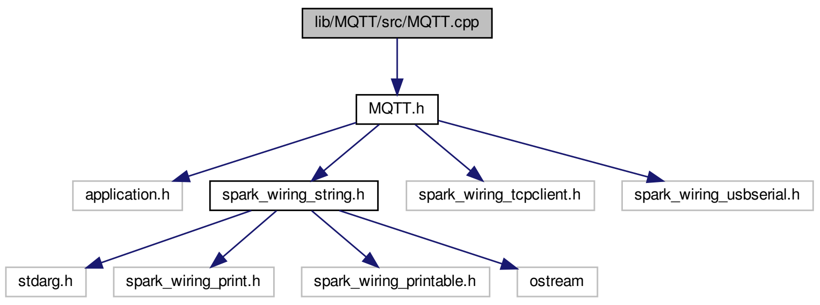
MIFARE Classic 1K (MF1S503x): Has 16 sectors $*$ 4 blocks/sector $*$ 16 bytes/block = 1024 bytes. The blocks are numbered 0-63. Block 3 in each sector is the Sector Trailer. See `http://www.nxp.↩
com/documents/data_sheet/MF1S503x.pdf` sections 8.6 and 8.7: Bytes 0-5: Key A Bytes 6-8: Access Bits Bytes 9: User data Bytes 10-15: Key B (or user data) Block 0 is read only manufacturer data. To access a block, an authentication using a key from the block's sector must be performed first. Example: To read from block 10, first authenticate using a key from sector 3 (blocks 8-11). All keys are set to FFFFFFFFFFFFh at chip delivery. Warning: Please read section 8.7 "Memory Access". It includes this text: if the PICC detects a format violation the whole sector is irreversibly blocked. To use a block in "value block" mode (for Increment/Decrement operations) you need to change the sector trailer. Use PICC_SetAccessBits() to calculate the bit patterns. MIFARE Classic 4K (MF1S703x): Has (32 sectors $*$ 4 blocks/sector + 8 sectors $*$ 16 blocks/sector) $*$ 16 bytes/block = 4096 bytes. The blocks are numbered 0-255. The last block in each sector is the Sector Trailer like above. MIFARE Classic Mini (MF1 IC S20): Has 5 sectors $*$ 4 blocks/sector $*$ 16 bytes/block = 320 bytes. The blocks are numbered 0-19. The last block in each sector is the Sector Trailer like above.

MIFARE Ultralight (MF0ICU1): Has 16 pages of 4 bytes = 64 bytes. Pages 0 + 1 is used for the 7-byte UID. Page 2 contains the last chech digit for the UID, one byte manufacturer internal data, and the lock bytes (see `http://www.nxp.com/documents/data_sheet/MF0ICU1.pdf` section 8.5.2) Page 3 is OTP, One Time Programmable bits. Once set to 1 they cannot revert to 0. Pages 4-15 are read/write unless blocked by the lock bytes in page 2. MIFARE Ultralight C (MF0ICU2): Has 48 pages of 4 bytes = 64 bytes. Pages 0 + 1 is used for the 7-byte UID. Page 2 contains the last chech digit for the UID, one byte manufacturer internal data, and the lock bytes (see `http://www.nxp.com/documents/data_sheet/MF0ICU1.pdf` section 8.5.2) Page 3 is OTP, One Time Programmable bits. Once set to 1 they cannot revert to 0. Pages 4-39 are read/write unless blocked by the lock bytes in page 2. Page 40 Lock bytes Page 41 16 bit one way counter Pages 42-43 Authentication configuration Pages 44-47 Authentication key

Definition at line 83 of file MFRC522.h.

11.15 lib/MQTT/src/MQTT.cpp File Reference

```
#include "MQTT.h"
```
Include dependency graph for MQTT.cpp:



Macros

- #define LOGGING
- #define MQTTQOS0_HEADER_MASK (0 << 1)
- #define MQTTQOS1_HEADER_MASK (1 << 1)
- #define MQTTQOS2_HEADER_MASK (2 << 1)
- #define DUP_FLAG_OFF_MASK (0<<3)
- #define DUP_FLAG_ON_MASK (1<<3)

11.15.1 Macro Definition Documentation

11.15.1.1 DUP_FLAG_OFF_MASK

```
#define DUP_FLAG_OFF_MASK (0<<3)
```

Definition at line 9 of file MQTT.cpp.

11.15.1.2 DUP_FLAG_ON_MASK

```
#define DUP_FLAG_ON_MASK (1<<3)
```

Definition at line 10 of file MQTT.cpp.

11.15.1.3 LOGGING

```
#define LOGGING
```

Definition at line 3 of file MQTT.cpp.

11.15.1.4 MQTTQOS0_HEADER_MASK

```
#define MQTTQOS0_HEADER_MASK (0 << 1)
```

Definition at line 5 of file MQTT.cpp.

11.15.1.5 MQTTQOS1_HEADER_MASK

```
#define MQTTQOS1_HEADER_MASK (1 << 1)
```

Definition at line 6 of file MQTT.cpp.

11.15.1.6 MQTTQOS2_HEADER_MASK

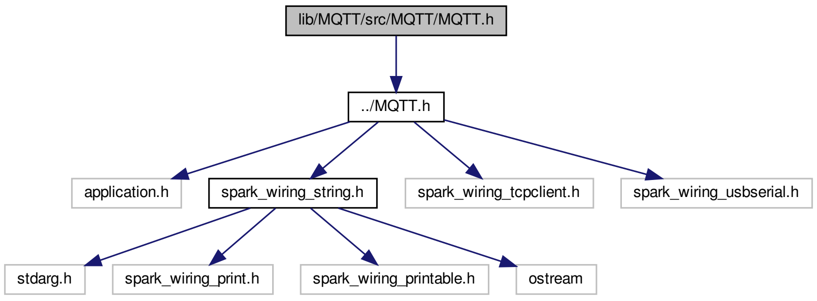
```
#define MQTTQOS2_HEADER_MASK (2 << 1)
```

Definition at line 7 of file MQTT.cpp.

11.16 lib/MQTT/src/MQTT/MQTT.h File Reference

```
#include "../MQTT.h"
```

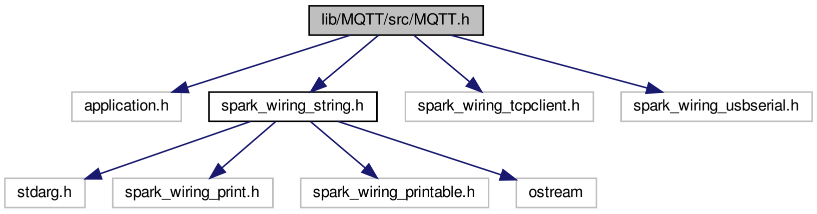
Include dependency graph for MQTT.h:



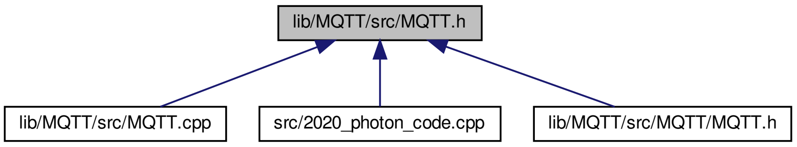
11.17 lib/MQTT/src/MQTT.h File Reference

```
#include "application.h"
#include "spark_wiring_string.h"
#include "spark_wiring_tcpclient.h"
#include "spark_wiring_usbserial.h"
```

Include dependency graph for MQTT.h:



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



Classes

- class MQTT

Macros

- #define MQTT_MAX_PACKET_SIZE 255
- #define MQTT_DEFAULT_KEEPALIVE 15
- #define MQTTPROTOCOLVERSION 3
- #define MQTTCONNECT 1 << 4
- #define MQTTCONNACK 2 << 4
- #define MQTTPUBLISH 3 << 4
- #define MQTTPUBACK 4 << 4
- #define MQTTPUBREC 5 << 4
- #define MQTTPUBREL 6 << 4
- #define MQTTPUBCOMP 7 << 4
- #define MQTTSUBSCRIBE 8 << 4
- #define MQTTSUBACK 9 << 4
- #define MQTTUNSUBSCRIBE 10 << 4
- #define MQTTUNSUBACK 11 << 4
- #define MQTTPINGREQ 12 << 4
- #define MQTTPINGRESP 13 << 4
- #define MQTTDISCONNECT 14 << 4
- #define MQTTReserved 15 << 4
- #define debug_print(fmt, ...) ((void)0)

11.17.1 Macro Definition Documentation

11.17.1.1 debug_print

```
#define debug_print(
              fmt,
              ... ) ((void)0)
```

Definition at line 101 of file MQTT.h.

11.17.1.2 MQTT_DEFAULT_KEEPALIVE

```
#define MQTT_DEFAULT_KEEPALIVE 15
```

Definition at line 76 of file MQTT.h.

11.17.1.3 MQTT_MAX_PACKET_SIZE

```
#define MQTT_MAX_PACKET_SIZE 255
```

Definition at line 73 of file MQTT.h.

11.17.1.4 MQTTCONNACK

```
#define MQTTCONNACK 2 << 4
```

Definition at line 80 of file MQTT.h.

11.17.1.5 MQTTCONNECT

```
#define MQTTCONNECT 1 << 4
```

Definition at line 79 of file MQTT.h.

11.17.1.6 MQTTDISCONNECT

```
#define MQTTDISCONNECT 14 << 4
```

Definition at line 92 of file MQTT.h.

11.17.1.7 MQTTPINGREQ

```
#define MQTTPINGREQ 12 << 4
```

Definition at line 90 of file MQTT.h.

11.17.1.8 MQTTPINGRESP

```
#define MQTTPINGRESP 13 << 4
```

Definition at line 91 of file MQTT.h.

11.17.1.9 MQTTPROTOCOLVERSION

```
#define MQTTPROTOCOLVERSION 3
```

Definition at line 78 of file MQTT.h.

11.17.1.10 MQTTPUBACK

```
#define MQTTPUBACK 4 << 4
```

Definition at line 82 of file MQTT.h.

11.17.1.11 MQTTPUBCOMP

```
#define MQTTPUBCOMP 7 << 4
```

Definition at line 85 of file MQTT.h.

11.17.1.12 MQTTPUBLISH

```
#define MQTTPUBLISH 3 << 4
```

Definition at line 81 of file MQTT.h.

11.17.1.13 MQTTPUBREC

```
#define MQTTPUBREC 5 << 4
```

Definition at line 83 of file MQTT.h.

11.17.1.14 MQTTPUBREL

```
#define MQTTPUBREL 6 << 4
```

Definition at line 84 of file MQTT.h.

11.17.1.15 MQTTReserved

```
#define MQTTReserved 15 << 4
```

Definition at line 93 of file MQTT.h.

11.17.1.16 MQTTSUBACK

```
#define MQTTSUBACK 9 << 4
```

Definition at line 87 of file MQTT.h.

11.17.1.17 MQTTSUBSCRIBE

```
#define MQTTSUBSCRIBE 8 << 4
```

Definition at line 86 of file MQTT.h.

11.17.1.18 MQTTUNSUBACK

```
#define MQTTUNSUBACK 11 << 4
```

Definition at line 89 of file MQTT.h.

11.17.1.19 MQTTUNSUBSCRIBE

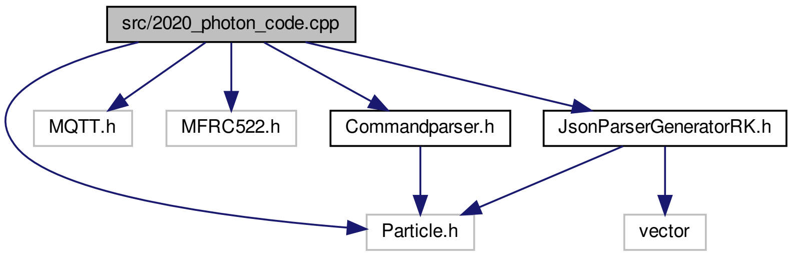
```
#define MQTTUNSUBSCRIBE 10 << 4
```

Definition at line 88 of file MQTT.h.

11.18 src/2020_photon_code.cpp File Reference

```
#include "Particle.h"
#include <MQTT.h>
#include <MFRC522.h>
#include "Commandparser.h"
#include <JsonParserGeneratorRK.h>
```

Include dependency graph for 2020_photon_code.cpp:



Macros

- #define CHARGEROFFSET 0

    constant that sets for which Photon this program is intended
- #define DEBUGPORT Serial
- #define SIZEOFUSERLIST 2
- #define SS_PIN_CHARGER1 A1
- #define SS_PIN_CHARGER2 A2
- #define RST_PIN A0
- #define EXTRA_DIGITAL_BREAKOUT_1 D0
- #define EXTRA_DIGITAL_BREAKOUT_2 D1
- #define EXTRA_DIGITAL_BREAKOUT_3 D3
- #define WAKEUP_OLIMEX D2
- #define RESET_OLIMEX D4
- #define PILOT_FEEDBACK_CAR_1 A6
- #define PILOT_FEEDBACK_CAR_2 A7
- #define AUTHENTICATION_CAR1 D5
- #define AUTHENTICATION_CAR2 D6
- #define EXTRA D7

Functions

- int resetOlimex (String input)

    Sends reset signal to EV charger controller.
- int WifiSignal (String input)

    Return wifi strength.
- int resetParticle (String input)

    Resets Photon.
- int progModeOlmx (String input)

Sets Olimex into programming mode.
- void blinkRFIDled (int charger, int action)

  unused function to blink the Photon LED
- int activeCharger ()

  Return 1 if socket 1 is used, 2 if socket 2 is used, and 3 if both are in use.
- int switchTest (String valueString)

  Switches between renewable mode (-input "true") and manual setpoint mode.
- int maxCurrentC1 (String setPointStr)

  Sets max Current output at socket 1 in manual mode.
- int maxCurrentC2 (String setPointStr)

  Sets max Current output at socket 2 in manual mode.
- int maxCurrentC1_test (unsigned int setPoint)

  Sets max Current output at socket 1/3 in renewable mode and publishes new setpoint at "HANevse/photonMaxC1" or C3.
- int maxCurrentC2_test (unsigned int setPoint)

  Sets max Current output at socket 2/4 in renewable mode and publishes new setpoint at "HANevse/photonMaxC2" or C4.
- String getUserIdAtSocket (int socket)

  Returns RFID tag at the asked socket.
- void allowUser_callback (byte ∗payload, unsigned int length)

  Callback function to process and execute approval or denial to charge from Pi, then MQTT publish reason to website GUI.
- int initRFID (String input)

  Initialises RFID reader.
- bool readRFIDCard (int Charger)

  Checks and reads RFID tag at the asked socket, then MQTT publishes it for Pi.
- void setup ()

  Inital setup for pin assignments and serial links start.
- void loop ()

  Main running function that executes all other functions; runs over 5times/second.
- int readSerialOlimex ()
- void reconnect (void)

  Function to reconnect to MQTT server if not connected and subscribe to needed topics.
- void callback (char ∗topic, byte ∗payload, unsigned int length)

  Main function for MQTT client to check for new messages and execute callback functions.
- void charToString (const char in[ ], String &out)

  Deprecated function to convert char to String - the String class already has one.
- void getMeasure_callback (byte ∗payload, unsigned int length)

  Callback function to automatically set max Currents from MQTT message if in renewable mode.
- STARTUP (WiFi.selectAntenna(ANT_EXTERNAL))
- void add_Measurement (float phaseVoltageL1, float phaseVoltageL2, float phaseVoltageL3, float currentL1, float currentL2, float currentL3, float Frequency, unsigned long Timestamp, int socketId=0, String userId="00")

  Function ran for each socket every 30s in main loop to send measurements through MQTT.

Variables

- float Current [2][3]
- float Power [2][3]
- float PhaseVoltage [2][3]
- float LineVoltage [2][3]
- float Energy [2]

- float Frequency [2]
- float CurrentList [20]
- int numberOfZeroReadings [2]
- String UIDtagCharger1 ="No ID"

  var to hold swiped RFID tag at first socket
- String UIDtagCharger2 ="No ID"

  var to hold swiped RFID tag at second socket
- MQTT client ("broker.hivemq.com", 1883, MQTT_DEFAULT_KEEPALIVE, callback, 512)

  MQTT client details; do not set last number to over 512!
- String test = "0"
- int counter =1
- MFRC522 mfrc522_Charger1 (SS_PIN_CHARGER1, RST_PIN)
- MFRC522 mfrc522_Charger2 (SS_PIN_CHARGER2, RST_PIN)
- unsigned long LatestStartTime [2] ={0,0}

  Holds latest start of new charge if charger is in use.
- bool handledCharger =0

  Holds last handled socket (0 for first socket)
- String ShareVar
- bool TESTCASE = false

  var that holds the charging mode (TRUE = renewable)
- ushort Pianswer =0

  var that holds answer from Pi but is unused now
- String currentStr =""
- unsigned int nextTime [2] = {30000,30000}

  Next timestamp to publish measurements in ms.

11.18.1 Macro Definition Documentation

11.18.1.1 AUTHENTICATION_CAR1

```
#define AUTHENTICATION_CAR1 D5
```

Definition at line 76 of file 2020_photon_code.cpp.

11.18.1.2 AUTHENTICATION_CAR2

```
#define AUTHENTICATION_CAR2 D6
```

Definition at line 77 of file 2020_photon_code.cpp.

11.18.1.3 CHARGEROFFSET

```
#define CHARGEROFFSET 0
```

constant that sets for which Photon this program is intended

For Photon 1 set it to 0, for Photon 2 set to 2. Any more and program would need to be edited.

Definition at line 59 of file 2020_photon_code.cpp.

11.18.1.4 DEBUGPORT

```
#define DEBUGPORT Serial
```

Definition at line 60 of file 2020_photon_code.cpp.

11.18.1.5 EXTRA

```
#define EXTRA D7
```

Definition at line 78 of file 2020_photon_code.cpp.

11.18.1.6 EXTRA_DIGITAL_BREAKOUT_1

```
#define EXTRA_DIGITAL_BREAKOUT_1 D0
```

Definition at line 69 of file 2020_photon_code.cpp.

11.18.1.7 EXTRA_DIGITAL_BREAKOUT_2

```
#define EXTRA_DIGITAL_BREAKOUT_2 D1
```

Definition at line 70 of file 2020_photon_code.cpp.

11.18.1.8 EXTRA_DIGITAL_BREAKOUT_3

```
#define EXTRA_DIGITAL_BREAKOUT_3 D3
```

Definition at line 71 of file 2020_photon_code.cpp.

11.18.1.9 PILOT_FEEDBACK_CAR_1

```
#define PILOT_FEEDBACK_CAR_1 A6
```

Definition at line 74 of file 2020_photon_code.cpp.

11.18.1.10 PILOT_FEEDBACK_CAR_2

```
#define PILOT_FEEDBACK_CAR_2 A7
```

Definition at line 75 of file 2020_photon_code.cpp.

11.18.1.11 RESET_OLIMEX

```
#define RESET_OLIMEX D4
```

Definition at line 73 of file 2020_photon_code.cpp.

11.18.1.12 RST_PIN

```
#define RST_PIN A0
```

Definition at line 67 of file 2020_photon_code.cpp.

11.18.1.13 SIZEOFUSERLIST

```
#define SIZEOFUSERLIST 2
```

Definition at line 61 of file 2020_photon_code.cpp.

11.18.1.14 SS_PIN_CHARGER1

```
#define SS_PIN_CHARGER1 A1
```

Definition at line 65 of file 2020_photon_code.cpp.

11.18.1.15 SS_PIN_CHARGER2

```
#define SS_PIN_CHARGER2 A2
```

Definition at line 66 of file 2020_photon_code.cpp.

11.18.1.16 WAKEUP_OLIMEX

```
#define WAKEUP_OLIMEX D2
```

Definition at line 72 of file 2020_photon_code.cpp.

11.18.2 Function Documentation

11.18.2.1 activeCharger()

```
int activeCharger ( )
```

Return 1 if socket 1 is used, 2 if socket 2 is used, and 3 if both are in use.

Definition at line 195 of file 2020_photon_code.cpp.

References Current.

11.18.2.2 add_Measurement()

```
void add_Measurement (
             float phaseVoltageL1,
             float phaseVoltageL2,
             float phaseVoltageL3,
             float currentL1,
             float currentL2,
             float currentL3,
             float Frequency,
             unsigned long Timestamp,
             int socketId = 0,
             String userId = "00" )
```

Function ran for each socket every 30s in main loop to send measurements through MQTT.

Definition at line 510 of file 2020_photon_code.cpp.

11.18.2.3 allowUser_callback()

```
void allowUser_callback (
             byte * payload,
             unsigned int length )
```

Callback function to process and execute approval or denial to charge from Pi, then MQTT publish reason to website GUI.

Definition at line 438 of file 2020_photon_code.cpp.

References client, String::concat(), String::operator=(), Pianswer, MQTT::publish(), UIDtagCharger1, and UIDtag←Charger2.

11.18.2.4 blinkRFIDled()

```
void blinkRFIDled (
             int charger,
             int action )
```

unused function to blink the Photon LED

Definition at line 179 of file 2020_photon_code.cpp.

11.18.2.5 callback()

```
void callback (
             char * topic,
             byte * payload,
             unsigned int length )
```

Main function for MQTT client to check for new messages and execute callback functions.

Definition at line 671 of file 2020_photon_code.cpp.

References maxCurrentC1(), maxCurrentC2(), String::operator=(), resetOlimex(), resetParticle(), switchTest(), and test.

11.18.2.6 charToString()

```
void charToString (
             const char in[],
             String & out )
```

Deprecated function to convert char to String - the String class already has one.

Definition at line 140 of file 2020_photon_code.cpp.

References String::operator=().

11.18.2.7 getMeasure_callback()

```
void getMeasure_callback (
            byte * payload,
            unsigned int length )
```

Callback function to automatically set max Currents from MQTT message if in renewable mode.

Definition at line 316 of file 2020_photon_code.cpp.

References maxCurrentC1_test(), and maxCurrentC2_test().

11.18.2.8 getUserIdAtSocket()

```
String getUserIdAtSocket (
            int socket )
```

Returns RFID tag at the asked socket.

Definition at line 307 of file 2020_photon_code.cpp.

References UIDtagCharger1, and UIDtagCharger2.

11.18.2.9 initRFID()

```
int initRFID (
            String input )
```

Initialises RFID reader.

Definition at line 558 of file 2020_photon_code.cpp.

Referenced by setup().

11.18.2.10 loop()

```
void loop ( )
```

Main running function that executes all other functions; runs over 5times/second.

Definition at line 879 of file 2020_photon_code.cpp.

References client, handledCharger, MQTT::isConnected(), LatestStartTime, MQTT::loop(), String::operator=(), readRFIDCard(), readSerialOlimex(), reconnect(), UIDtagCharger1, and UIDtagCharger2.

11.18.2.11 maxCurrentC1()

```
int maxCurrentC1 (
            String setPointStr )
```

Sets max Current output at socket 1 in manual mode.

Definition at line 229 of file 2020_photon_code.cpp.

References TESTCASE, and String::toInt().

Referenced by callback(), and switchTest().

11.18.2.12 maxCurrentC1_test()

```
int maxCurrentC1_test (
            unsigned int setPoint )
```

Sets max Current output at socket 1/3 in renewable mode and publishes new setpoint at "HANevse/photonMaxC1" or C3.

Definition at line 257 of file 2020_photon_code.cpp.

References client, String::concat(), MQTT::publish(), String::String(), and TESTCASE.

Referenced by getMeasure_callback().

11.18.2.13 maxCurrentC2()

```
int maxCurrentC2 (
            String setPointStr )
```

Sets max Current output at socket 2 in manual mode.

Definition at line 243 of file 2020_photon_code.cpp.

References TESTCASE, and String::toInt().

Referenced by callback(), and switchTest().

11.18.2.14 maxCurrentC2_test()

```
int maxCurrentC2_test (
            unsigned int setPoint )
```

Sets max Current output at socket 2/4 in renewable mode and publishes new setpoint at "HANevse/photonMaxC2" or C4.

Definition at line 273 of file 2020_photon_code.cpp.

References client, String::concat(), MQTT::publish(), String::String(), and TESTCASE.

Referenced by getMeasure_callback().

11.18.2.15 progModeOlmx()

```
int progModeOlmx (
            String input )
```

Sets Olimex into programming mode.

Definition at line 169 of file 2020_photon_code.cpp.

References resetOlimex().

11.18.2.16 readRFIDCard()

```
bool readRFIDCard (
            int Charger )
```

Checks and reads RFID tag at the asked socket, then MQTT publishes it for Pi.

Definition at line 581 of file 2020_photon_code.cpp.

References Pianswer, String::substring(), UIDtagCharger1, and UIDtagCharger2.

Referenced by loop().

11.18.2.17 readSerialOlimex()

```
int readSerialOlimex ( )
```

Function to read from Olimex serial port and run stringParse() Returns the last charger socket it received data from.

Definition at line 271 of file Commandparser.h.

References buff, bufpos, and stringParse().

Referenced by loop().

11.18.2.18 reconnect()

```
void reconnect (
            void  )
```

Function to reconnect to MQTT server if not connected and subscribe to needed topics.

Definition at line 785 of file 2020_photon_code.cpp.

References client, MQTT::connect(), MQTT::isConnected(), and MQTT::subscribe().

Referenced by loop().

11.18.2.19 resetOlimex()

```
int resetOlimex (
            String input )
```

Sends reset signal to EV charger controller.

Definition at line 151 of file 2020_photon_code.cpp.

Referenced by callback(), and progModeOlmx().

11.18.2.20 resetParticle()

```
int resetParticle (
            String input )
```

Resets Photon.

Definition at line 164 of file 2020_photon_code.cpp.

Referenced by callback().

11.18.2.21 setup()

```
void setup ( )
```

Inital setup for pin assignments and serial links start.

Definition at line 830 of file 2020_photon_code.cpp.

References initRFID().

11.18.2.22 STARTUP()

```
STARTUP (
             WiFi.  selectAntennaANT_EXTERNAL )
```

11.18.2.23 switchTest()

```
int switchTest (
             String valueString )
```

Switches between renewable mode (-input "true") and manual setpoint mode.

Definition at line 215 of file 2020_photon_code.cpp.

References maxCurrentC1(), maxCurrentC2(), String::operator==(), and TESTCASE.

Referenced by callback().

11.18.2.24 WifiSignal()

```
int WifiSignal (
             String input )
```

Return wifi strength.

Definition at line 159 of file 2020_photon_code.cpp.

11.18.3 Variable Documentation

11.18.3.1 client

```
MQTT client("broker.hivemq.com", 1883, MQTT_DEFAULT_KEEPALIVE, callback, 512)
```

MQTT client details; do not set last number to over 512!

Referenced by allowUser_callback(), loop(), maxCurrentC1_test(), maxCurrentC2_test(), and reconnect().

11.18.3.2 counter

```
int counter =1
```

Definition at line 96 of file 2020_photon_code.cpp.

11.18.3.3 Current

```
float Current[2][3]
```

Definition at line 30 of file Commandparser.h.

Referenced by activeCharger(), and stringParse().

11.18.3.4 CurrentList

```
float CurrentList[20]
```

Definition at line 36 of file Commandparser.h.

Referenced by stringParse().

11.18.3.5 currentStr

```
String currentStr =""
```

Definition at line 135 of file 2020_photon_code.cpp.

11.18.3.6 Energy

```
float Energy[2]
```

Definition at line 34 of file Commandparser.h.

Referenced by stringParse().

11.18.3.7 Frequency

```
float Frequency[2]
```

Definition at line 35 of file Commandparser.h.

Referenced by stringParse().

11.18.3.8 handledCharger

```
bool handledCharger =0
```

Holds last handled socket (0 for first socket)

Definition at line 102 of file 2020_photon_code.cpp.

Referenced by loop().

11.18.3.9 LatestStartTime

```
unsigned long LatestStartTime[2] ={0,0}
```

Holds latest start of new charge if charger is in use.

Definition at line 100 of file 2020_photon_code.cpp.

Referenced by loop().

11.18.3.10 LineVoltage

```
float LineVoltage[2][3]
```

Definition at line 33 of file Commandparser.h.

Referenced by stringParse().

11.18.3.11 mfrc522_Charger1

```
MFRC522 mfrc522_Charger1(SS_PIN_CHARGER1, RST_PIN)
```

11.18.3.12 mfrc522_Charger2

```
MFRC522 mfrc522_Charger2(SS_PIN_CHARGER2, RST_PIN)
```

11.18.3.13 nextTime

```
unsigned int nextTime[2] = {30000,30000}
```

Next timestamp to publish measurements in ms.

Definition at line 137 of file 2020_photon_code.cpp.

11.18.3.14 numberOfZeroReadings

```
int numberOfZeroReadings[2]
```

Definition at line 37 of file Commandparser.h.

Referenced by stringParse().

11.18.3.15 PhaseVoltage

```
float PhaseVoltage[2][3]
```

Definition at line 32 of file Commandparser.h.

Referenced by stringParse().

11.18.3.16 Pianswer

```
ushort Pianswer =0
```

var that holds answer from Pi but is unused now

Definition at line 110 of file 2020_photon_code.cpp.

Referenced by allowUser_callback(), and readRFIDCard().

11.18.3.17 Power

```
float Power[2][3]
```

Definition at line 31 of file Commandparser.h.

Referenced by stringParse().

11.18.3.18 ShareVar

```
String ShareVar
```

Definition at line 103 of file 2020_photon_code.cpp.

11.18.3.19 test

```
String test = "0"
```

Definition at line 93 of file 2020_photon_code.cpp.

Referenced by callback().

11.18.3.20 TESTCASE

```
bool TESTCASE = false
```

var that holds the charging mode (TRUE = renewable)

Definition at line 107 of file 2020_photon_code.cpp.

Referenced by maxCurrentC1(), maxCurrentC1_test(), maxCurrentC2(), maxCurrentC2_test(), and switchTest().

11.18.3.21 UIDtagCharger1

```
String UIDtagCharger1 ="No ID"
```

var to hold swiped RFID tag at first socket

Definition at line 51 of file 2020_photon_code.cpp.

Referenced by allowUser_callback(), getUserIdAtSocket(), loop(), and readRFIDCard().

11.18.3.22 UIDtagCharger2

```
String UIDtagCharger2 ="No ID"
```

var to hold swiped RFID tag at second socket

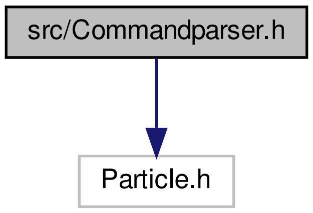
Definition at line 53 of file 2020_photon_code.cpp.

Referenced by allowUser_callback(), getUserIdAtSocket(), loop(), and readRFIDCard().

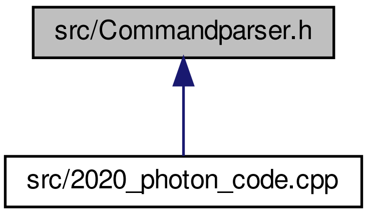
11.19 src/Commandparser.h File Reference

`#include "Particle.h"`

Include dependency graph for Commandparser.h:



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



Macros

- #define BUFSIZE 350
- #define RSTTIMEOUT 300000
- #define DEBUGPORT Serial

Functions

- void Send (String test)
- float bytesToFloat (unsigned char b0, unsigned char b1, unsigned char b2, unsigned char b3)
- bool bytesArrToFloatArr (char ∗Arr, unsigned int ArrLen, float ∗OutputArr, unsigned int FloatLen)
- int stringParse (char ∗buf, int buflen)
- int readSerialOlimex ()

Variables

- bool readnextLine = false
- char buff [BUFSIZE]
- int bufpos = 0
- unsigned long lastUpload = 0
- float Current [2][3] ={{0,0,0},{0,0,0}}
- float Power [2][3] ={{0,0,0},{0,0,0}}
- float PhaseVoltage [2][3] ={{0,0,0},{0,0,0}}
- float LineVoltage [2][3] ={{0,0,0},{0,0,0}}
- float Energy [2] ={0,0}
- float Frequency [2] ={0,0}
- float CurrentList [20]
- int numberOfZeroReadings [2] ={0,0}

11.19.1 Macro Definition Documentation

11.19.1.1 BUFSIZE

```
#define BUFSIZE 350
```

Definition at line 7 of file Commandparser.h.

11.19.1.2 DEBUGPORT

```
#define DEBUGPORT Serial
```

Definition at line 9 of file Commandparser.h.

11.19.1.3 RSTTIMEOUT

```
#define RSTTIMEOUT 300000
```

Definition at line 8 of file Commandparser.h.

11.19.2 Function Documentation

11.19.2.1 bytesArrToFloatArr()

```
bool bytesArrToFloatArr (
             char * Arr,
             unsigned int ArrLen,
             float * OutputArr,
             unsigned int FloatLen )
```

Function to convert an array of Olimex 4-byte values to float variables

Definition at line 65 of file Commandparser.h.

Referenced by stringParse().

11.19.2.2 bytesToFloat()

```
float bytesToFloat (
             unsigned char b0,
             unsigned char b1,
             unsigned char b2,
             unsigned char b3 )
```

Function to convert Olimex 4-byte value to float variable

Definition at line 50 of file Commandparser.h.

Referenced by stringParse().

11.19.2.3 readSerialOlimex()

```
int readSerialOlimex ( )
```

Function to read from Olimex serial port and run stringParse() Returns the last charger socket it received data from.

Definition at line 271 of file Commandparser.h.

References buff, bufpos, and stringParse().

Referenced by loop().

11.19.2.4 Send()

```
void Send (
             String test )
```

11.19.2.5 stringParse()

```
int stringParse (
              char * buf,
              int buflen )
```

Function to parse Olimex message into energy measurements Returns the charger socket it received data from.

Definition at line 107 of file Commandparser.h.

References bytesArrToFloatArr(), bytesToFloat(), Current, CurrentList, Energy, Frequency, LineVoltage, number←
OfZeroReadings, PhaseVoltage, and Power.

Referenced by readSerialOlimex().

11.19.3 Variable Documentation

11.19.3.1 buff

```
char buff[BUFSIZE]
```

Definition at line 14 of file Commandparser.h.

Referenced by readSerialOlimex().

11.19.3.2 bufpos

```
int bufpos = 0
```

Definition at line 15 of file Commandparser.h.

Referenced by readSerialOlimex().

11.19.3.3 Current

```
float Current[2][3] ={{0,0,0},{0,0,0}}
```

Definition at line 30 of file Commandparser.h.

Referenced by activeCharger(), and stringParse().

11.19.3.4 CurrentList

```
float CurrentList[20]
```

Definition at line 36 of file Commandparser.h.

Referenced by stringParse().

11.19.3.5 Energy

```
float Energy[2] ={0,0}
```

Definition at line 34 of file Commandparser.h.

Referenced by stringParse().

11.19.3.6 Frequency

```
float Frequency[2] ={0,0}
```

Definition at line 35 of file Commandparser.h.

Referenced by stringParse().

11.19.3.7 lastUpload

```
unsigned long lastUpload = 0
```

Definition at line 18 of file Commandparser.h.

11.19.3.8 LineVoltage

```
float LineVoltage[2][3] ={{0,0,0},{0,0,0}}
```

Definition at line 33 of file Commandparser.h.

Referenced by stringParse().

11.19.3.9 numberOfZeroReadings

```
int numberOfZeroReadings[2] ={0,0}
```

Definition at line 37 of file Commandparser.h.

Referenced by stringParse().

11.19.3.10 PhaseVoltage

```
float PhaseVoltage[2][3] ={{0,0,0},{0,0,0}}
```

Definition at line 32 of file Commandparser.h.

Referenced by stringParse().

11.19.3.11 Power

```
float Power[2][3] ={{0,0,0},{0,0,0}}
```

Definition at line 31 of file Commandparser.h.

Referenced by stringParse().

11.19.3.12 readnextLine

```
bool readnextLine = false
```

Definition at line 13 of file Commandparser.h.

Index