Arduino Gsm Driver

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:06:46

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

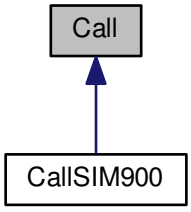
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Call Class Reference

`#include <Call.h>`

Inheritance diagram for Call:



Public Member Functions

- virtual unsigned char answer ()=0
- virtual unsigned char callNumber (unsigned char ∗number)=0
- virtual unsigned char callFromPhonebook (unsigned char position)=0
- virtual unsigned char callByPhonebookMatch (unsigned char ∗entry)=0
- virtual unsigned char redial ()=0
- virtual unsigned char disconnect ()=0

4.1.1 Detailed Description

Arduino - Gsm driver.

Call.h

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Call.h.

4.1.2 Member Function Documentation

4.1.2.1 virtual unsigned char Call::answer ( ) [pure virtual]

Answer an Incoming Call.

TA sends off-hook to the remote station.

Returns

Implemented in CallSIM900.

4.1.2.2 virtual unsigned char Call::callByPhonebookMatch (unsigned char * *entry*) [pure virtual]

Originate Call to Phone Number in Memory Which Corresponds to Field.

This Command make the TA attempts to set up an outgoing call to stored number which has the name matching with the entry string.

Parameters

entry	Phonebook entry.

Returns

Implemented in CallSIM900.

4.1.2.3 virtual unsigned char Call::callFromPhonebook (unsigned char *position*) [pure virtual]

Originate Call to Phone Number in Current Memory.

This Command can be used to dial a phone number from current phonebook.

Parameters

position	Phonebook position.

Returns

Implemented in CallSIM900.

4.1.2.4 virtual unsigned char Call::callNumber (unsigned char * *number*) [pure virtual]

Mobile Originated Call to Dial A Number.

This Command can be used to set up outgoing voice, data or fax calls. It also serves to control supplementary services.

Parameters

number	The number to make the call to.

Returns

Implemented in CallSIM900.

4.1.2.5 virtual unsigned char Call::disconnect ( ) [pure virtual]

Disconnect Existing Connection.

Returns

Implemented in CallSIM900.

4.1.2.6 virtual unsigned char Call::redial ( ) [pure virtual]

Redial Last Telephone Number Used.

This Command redials the last voice and data call number used.

Returns

Implemented in CallSIM900.

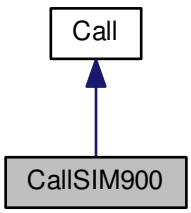
The documentation for this class was generated from the following file:

- Call.h

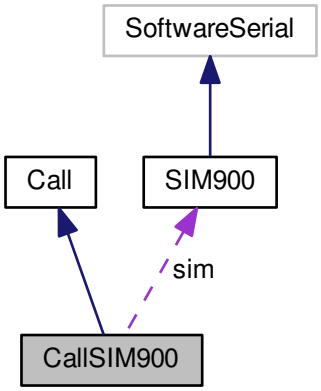
4.2 CallSIM900 Class Reference

`#include <CallSIM900.h>`

Inheritance diagram for CallSIM900:



Collaboration diagram for CallSIM900:



Public Member Functions

- CallSIM900 (SIM900 ∗sim)
- virtual unsigned char answer ()
- virtual unsigned char callNumber (unsigned char ∗number)
- virtual unsigned char callFromPhonebook (unsigned char position)
- virtual unsigned char callByPhonebookMatch (unsigned char ∗entry)
- virtual unsigned char redial ()
- virtual unsigned char disconnect ()

Private Attributes

- SIM900 ∗ sim

4.2.1 Detailed Description

Arduino - Gsm driver.

CallSIM900.h

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 17 of file CallSIM900.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CallSIM900::CallSIM900 (SIM900 ∗ *sim*)

Definition at line 16 of file CallSIM900.cpp.

4.2.3 Member Function Documentation

4.2.3.1 unsigned char CallSIM900::answer ( ) [virtual]

Answer an Incoming Call.

TA sends off-hook to the remote station.

Returns

Implements Call.

Definition at line 19 of file CallSIM900.cpp.

4.2.3.2 unsigned char CallSIM900::callByPhonebookMatch (unsigned char * *entry*) [virtual]

Originate Call to Phone Number in Memory Which Corresponds to Field.

This Command make the TA attempts to set up an outgoing call to stored number which has the name matching with the entry string.

Parameters

entry	Phonebook entry.

Returns

Implements Call.

Definition at line 32 of file CallSIM900.cpp.

4.2.3.3 unsigned char CallSIM900::callFromPhonebook (unsigned char *position*) [virtual]

Originate Call to Phone Number in Current Memory.

This Command can be used to dial a phone number from current phonebook.

Parameters

position	Phonebook position.

Returns

Implements Call.

Definition at line 28 of file CallSIM900.cpp.

4.2.3.4 unsigned char CallSIM900::callNumber (unsigned char * *number*) [virtual]

Mobile Originated Call to Dial A Number.

This Command can be used to set up outgoing voice, data or fax calls. It also serves to control supplementary services.

Parameters

number	The number to make the call to.

Returns

Implements Call.

Definition at line 24 of file CallSIM900.cpp.

4.2.3.5 unsigned char CallSIM900::disconnect ( ) [virtual]

Disconnect Existing Connection.

Returns

Implements Call.

Definition at line 40 of file CallSIM900.cpp.

4.2.3.6 unsigned char CallSIM900::redial ( ) [virtual]

Redial Last Telephone Number Used.

This Command redials the last voice and data call number used.

Returns

Implements Call.

Definition at line 36 of file CallSIM900.cpp.

4.2.4 Member Data Documentation

4.2.4.1 SIM900* CallSIM900::sim [private]

Definition at line 19 of file CallSIM900.h.

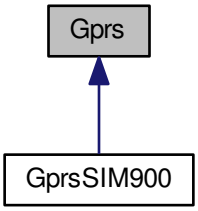
The documentation for this class was generated from the following files:

- CallSIM900.h
- CallSIM900.cpp

4.3 Gprs Class Reference

```
#include <Gprs.h>
```

Inheritance diagram for Gprs:



Public Member Functions

- virtual unsigned char useMultiplexer (bool use)=0
- virtual unsigned char attach (const char ∗apn, const char ∗login, const char ∗password)=0
- virtual unsigned char bringUp ()=0
- virtual unsigned char obtainIp (unsigned char ∗buf)=0
- virtual unsigned char status ()=0
- virtual unsigned char configureDns (const char ∗primary, const char ∗secondary)=0
- virtual unsigned char open (const char ∗mode, const char ∗address, unsigned int port)=0
- virtual unsigned char open (char connection, const char ∗mode, const char ∗address, unsigned int port)=0
- virtual unsigned char close (char connection)=0
- virtual unsigned char close ()=0
- virtual unsigned char resolve (const char ∗name, unsigned char ∗buf, unsigned int len)=0
- virtual unsigned char send (unsigned char ∗buf, unsigned int len)=0
- virtual unsigned char send (char connection, unsigned char ∗buf, unsigned int len)=0
- virtual unsigned char setUpServer (unsigned char mode, unsigned int port)=0
- virtual unsigned char shutdown ()=0

4.3.1 Detailed Description

Arduino - Gsm driver.

Gprs.h

GPRS connection using.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Gprs.h.

4.3.2 Member Function Documentation

4.3.2.1 virtual unsigned char Gprs::attach (const char ∗ *apn,* const char ∗ *login,* const char ∗ *password*) [pure virtual]

Start Task and Set APN, LOGIN, PASSWORD.

Each parameter must be \0 teminated.

Parameters

apn	The apn access point name.
login	The GPRS user name.
password	The GPRS password.

Returns

Implemented in GprsSIM900.

4.3.2.2 virtual unsigned char Gprs::bringUp ( ) [pure virtual]

Bring Up Wireless Connection with GPRS or CSD.

Connects to the GPRS network.

Returns

Implemented in GprsSIM900.

4.3.2.3 virtual unsigned char Gprs::close (char *connection*) [pure virtual]

Close TCP or UDP Connection.

Parameters

connection	

Returns

Implemented in GprsSIM900.

4.3.2.4 virtual unsigned char Gprs::close ( ) [pure virtual]

Close TCP or UDP Connection.

Returns

Implemented in GprsSIM900.

4.3.2.5 virtual unsigned char Gprs::configureDns (const char * *primary*, const char * *secondary*) [pure virtual]

Configure Domain Name Server.

Returns

Implemented in GprsSIM900.

4.3.2.6 virtual unsigned char Gprs::obtainIp (unsigned char * *buf*) [pure virtual]

Get Local IP Address.

Returns the the IP address assigned from GPRS or CSD in 4 bytes format.

Parameters

entry	Phonebook entry.

Returns

Implemented in GprsSIM900.

4.3.2.7 virtual unsigned char Gprs::open (const char * *mode*, const char * *address*, unsigned int *port*) [pure virtual]

Start Up TCP or UDP Connection.

Returns

Implemented in GprsSIM900.

4.3.2.8 virtual unsigned char Gprs::open (char *connection*, const char * *mode*, const char * *address*, unsigned int *port*) [pure virtual]

Start Up TCP or UDP Connection.

Returns

Implemented in GprsSIM900.

4.3.2.9 virtual unsigned char Gprs::resolve (const char * *name*, unsigned char * *buf*, unsigned int *len*) [pure virtual]

Query the IP Address of Given Domain Name.

Returns

Implemented in GprsSIM900.

4.3.2.10 virtual unsigned char Gprs::send (unsigned char * *buf*, unsigned int *len*) [pure virtual]

Send Data Through TCP or UDP Connection.

Returns

Implemented in GprsSIM900.

4.3.2.11 virtual unsigned char Gprs::send (char *connection*, unsigned char * *buf*, unsigned int *len*) [pure virtual]

Send Data Through TCP or UDP Connection.

Returns

Implemented in GprsSIM900.

4.3.2.12 virtual unsigned char Gprs::setUpServer (unsigned char *mode*, unsigned int *port*) [pure virtual]

Configure Module as Server.

Returns

Implemented in GprsSIM900.

4.3.2.13 virtual unsigned char Gprs::shutdown ( ) [pure virtual]

Deactivate GPRS PDP Context.

Returns

Implemented in GprsSIM900.

4.3.2.14 virtual unsigned char Gprs::status ( ) [pure virtual]

Query Current Connection Status.

Returns

Implemented in GprsSIM900.

4.3.2.15 virtual unsigned char Gprs::useMultiplexer (bool *use*) [pure virtual]

Start Up Multi-IP Connection.

Enable or disable multi IP connection.

Parameters

use	0 disables multi IP connection and 1 enables.

Returns

Implemented in GprsSIM900.

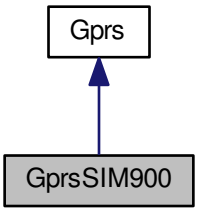
The documentation for this class was generated from the following file:

- Gprs.h

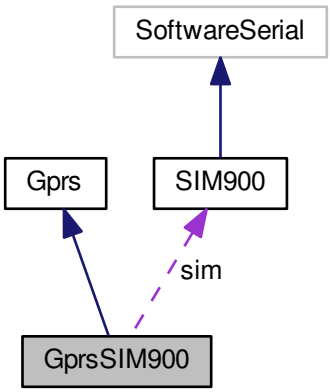
4.4 GprsSIM900 Class Reference

```
#include <GprsSIM900.h>
```

Inheritance diagram for GprsSIM900:



Collaboration diagram for GprsSIM900:



Public Types

- enum OperationResult { OK = 0, ERROR = 1, COMMAND_TOO_LONG = 2 }

Public Member Functions

- GprsSIM900 (SIM900 *sim)
- void begin (long bound)
- unsigned char useMultiplexer (bool use)
- unsigned char attach (const char *apn, const char *login, const char *password)
- unsigned char bringUp ()
- unsigned char obtainIp (unsigned char *buf)
- unsigned char status ()
- unsigned char configureDns (const char *primary, const char *secondary)
- unsigned char open (const char *mode, const char *address, unsigned int port)
- unsigned char open (char connection, const char *mode, const char *address, unsigned int port)

- unsigned char close (char connection)
- unsigned char close ()
- unsigned char resolve (const char *name, unsigned char *buf, unsigned int len)
- unsigned char send (unsigned char *buf, unsigned int len)
- unsigned char send (char connection, unsigned char *buf, unsigned int len)
- unsigned char setUpServer (unsigned char mode, unsigned int port)
- unsigned char shutdown ()

Private Attributes

- SIM900 * sim
- bool multiplexed

4.4.1 Detailed Description

Definition at line 20 of file GprsSIM900.h.

4.4.2 Member Enumeration Documentation

4.4.2.1 enum GprsSIM900::OperationResult

Enumerator

OK

ERROR

COMMAND_TOO_LONG

Definition at line 34 of file GprsSIM900.h.

4.4.3 Constructor & Destructor Documentation

4.4.3.1 GprsSIM900::GprsSIM900 (SIM900 * *sim*)

Public constructor.

Parameters

sim	The SIM900 pointer.

Definition at line 16 of file GprsSIM900.cpp.

4.4.4 Member Function Documentation

4.4.4.1 unsigned char GprsSIM900::attach (const char * *apn*, const char * *login*, const char * *password*) [virtual]

Start Task and Set APN, LOGIN, PASSWORD.

Each parameter must be \0 teminated.

Parameters

apn	The apn access point name.
login	The GPRS user name.

password	The GPRS password.

Returns

Implements Gprs.

Definition at line 35 of file GprsSIM900.cpp.

4.4.4.2 void GprsSIM900::begin (long *bound*)

Initializes the device.

Parameters

The	bound rate to be used.

Definition at line 20 of file GprsSIM900.cpp.

4.4.4.3 unsigned char GprsSIM900::bringUp ( ) [virtual]

Bring Up Wireless Connection with GPRS or CSD.

Connects to the GPRS network.

Returns

Implements Gprs.

Definition at line 47 of file GprsSIM900.cpp.

4.4.4.4 unsigned char GprsSIM900::close (char *connection*) [virtual]

Close TCP or UDP Connection.

Parameters

connection	

Returns

Implements Gprs.

Definition at line 116 of file GprsSIM900.cpp.

4.4.4.5 unsigned char GprsSIM900::close ( ) [virtual]

Close TCP or UDP Connection.

Returns

Implements Gprs.

Definition at line 127 of file GprsSIM900.cpp.

4.4.4.6 unsigned char GprsSIM900::configureDns (const char * *primary*, const char * *secondary*) [virtual]

Configure Domain Name Server.

Returns

Implements Gprs.

Definition at line 85 of file GprsSIM900.cpp.

4.4.4.7 unsigned char GprsSIM900::obtainIp (unsigned char * *buf*) [virtual]

Get Local IP Address.

Returns the the IP address assigned from GPRS or CSD in 4 bytes format.

Parameters

entry	Phonebook entry.

Returns

Implements Gprs.

Definition at line 53 of file GprsSIM900.cpp.

4.4.4.8 unsigned char GprsSIM900::open (const char * *mode*, const char * *address*, unsigned int *port*) [virtual]

Start Up TCP or UDP Connection.

Returns

Implements Gprs.

Definition at line 95 of file GprsSIM900.cpp.

4.4.4.9 unsigned char GprsSIM900::open (char *connection*, const char * *mode*, const char * *address*, unsigned int *port*) [virtual]

Start Up TCP or UDP Connection.

Returns

Implements Gprs.

Definition at line 99 of file GprsSIM900.cpp.

4.4.4.10 unsigned char GprsSIM900::resolve (const char * *name*, unsigned char * *buf*, unsigned int *len*) [virtual]

Query the IP Address of Given Domain Name.

Returns

Implements Gprs.

Definition at line 131 of file GprsSIM900.cpp.

4.4.4.11 unsigned char GprsSIM900::send (unsigned char * *buf,* unsigned int *len*) [virtual]

Send Data Through TCP or UDP Connection.

Returns

Implements Gprs.

Definition at line 138 of file GprsSIM900.cpp.

4.4.4.12 unsigned char GprsSIM900::send (char *connection,* unsigned char * *buf,* unsigned int *len*) [virtual]

Send Data Through TCP or UDP Connection.

Returns

Implements Gprs.

Definition at line 143 of file GprsSIM900.cpp.

4.4.4.13 unsigned char GprsSIM900::setUpServer (unsigned char *mode,* unsigned int *port*) [virtual]

Configure Module as Server.

Returns

Implements Gprs.

Definition at line 147 of file GprsSIM900.cpp.

4.4.4.14 unsigned char GprsSIM900::shutdown ( ) [virtual]

Deactivate GPRS PDP Context.

Returns

Implements Gprs.

Definition at line 150 of file GprsSIM900.cpp.

4.4.4.15 unsigned char GprsSIM900::status ( ) [virtual]

Query Current Connection Status.

Returns

Implements Gprs.

Definition at line 81 of file GprsSIM900.cpp.

4.4.4.16 unsigned char GprsSIM900::useMultiplexer (bool *use*) [virtual]

Start Up Multi-IP Connection.

Enable or disable multi IP connection.

Parameters

use	0 disables multi IP connection and 1 enables.

Returns

Implements Gprs.

Definition at line 26 of file GprsSIM900.cpp.

4.4.5 Member Data Documentation

4.4.5.1 bool GprsSIM900::multiplexed [private]

Multi connection.

Definition at line 30 of file GprsSIM900.h.

4.4.5.2 SIM900* GprsSIM900::sim [private]

SIM900 pointer.

Definition at line 25 of file GprsSIM900.h.

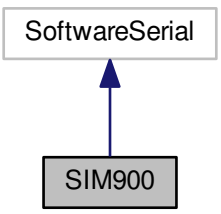
The documentation for this class was generated from the following files:

- GprsSIM900.h
- GprsSIM900.cpp

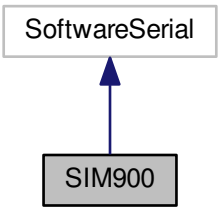
4.5 SIM900 Class Reference

`#include <SIM900.h>`

Inheritance diagram for SIM900:



Collaboration diagram for SIM900:



Public Member Functions

- SIM900 (unsigned char receivePin, unsigned char transmitPin)
- void begin (long bound)
- unsigned char * getLastResponse ()
- bool sendCommandExpecting (const char *command, const char *expectation, bool append, unsigned long timeout)
- bool sendCommandExpecting (const char *command, const char *expectation, bool append)
- bool sendCommandExpecting (const char *command, const char *expectation, unsigned long timeout)
- bool sendCommandExpecting (const char *command, const char *expectation)
- bool doesResponseContains (const char *expectation)
- int sendCommand (const char *command, bool append, unsigned long timeout)
- int sendCommand (const char *command, bool append)
- int sendCommand (const char *command, unsigned long timeout)
- int sendCommand (const char *command)
- int readResponse (unsigned long timeout)
- void setCommandEcho (bool echo)

Private Attributes

- unsigned char rxBuffer [SIM900_RX_BUFFER_SIZE]
- unsigned char * rxPointer
- bool echo

4.5.1 Detailed Description

Definition at line 20 of file SIM900.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 SIM900::SIM900 (unsigned char *receivePin,* unsigned char *transmitPin*)

Public constructor.

Parameters

serial	

Definition at line 16 of file SIM900.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void SIM900::begin (long *bound*)

Initializes the device.

Parameters

The	bound rate to be used.

Definition at line 21 of file SIM900.cpp.

4.5.3.2 bool SIM900::doesResponseContains (const char * *expectation*)

Checks if the last response contains the given sub-string.

Parameters

expectation	The expectation string.

Returns

Definition at line 42 of file SIM900.cpp.

4.5.3.3 unsigned char* SIM900::getLastResponse ( ) [inline]

Get a pointer to the last response.

Returns

Definition at line 57 of file SIM900.h.

4.5.3.4 int SIM900::readResponse (unsigned long *timeout*)

Reads the response from the device.

Parameters

timeout	The maximum time to perform the op.

Returns

How many bytes was received. 0 if timeout.

Definition at line 57 of file SIM900.cpp.

4.5.3.5 int SIM900::sendCommand (const char * *command*, bool *append*, unsigned long *timeout*)

Sends a command to the device.

Parameters

command	The command string, should be \0 ended.
timeout	The maximum time to perform the op.

Returns

Definition at line 48 of file SIM900.cpp.

4.5.3.6 int SIM900::sendCommand (const char * *command*, bool *append*) [inline]

Sends a command to the device.

DEFAULT TIMEOUT

Parameters

command	The command string, should be \0 ended.
append	Boolean saying if the AT must be appended.

Returns

Definition at line 138 of file SIM900.h.

4.5.3.7 int SIM900::sendCommand (const char * *command*, unsigned long *timeout*) [inline]

Sends a command to the device.

DEFAULT APPEND

Parameters

command	The command string, should be \0 ended.
append	Boolean saying if the AT must be appended.

Returns

Definition at line 151 of file SIM900.h.

4.5.3.8 int SIM900::sendCommand (const char * *command*) [inline]

Sends a command to the device.

DEFAULT TIMEOUT DEFAULT APPEND

Parameters

command	The command string, should be \0 ended.

Returns

Definition at line 164 of file SIM900.h.

4.5.3.9 bool SIM900::sendCommandExpecting (const char * *command*, const char * *expectation*, bool *append*, unsigned long *timeout*)

Sends a command expecting some result.

Parameters

command	The command string, should be \0 ended.
expectation	The expectation string.
timeout	The maximum time to perform the op.

Returns

Definition at line 35 of file SIM900.cpp.

4.5.3.10 bool SIM900::sendCommandExpecting (const char * *command*, const char * *expectation*, bool *append*) [inline]

Sends a command expecting some result.

DEFAULT TIMEOUT

Parameters

command	The command string, should be \0 ended.
expectation	The expectation string.

Returns

Definition at line 81 of file SIM900.h.

4.5.3.11 bool SIM900::sendCommandExpecting (const char * *command*, const char * *expectation*, unsigned long *timeout*) [inline]

Sends a command expecting some result.

DEFAULT APPEND

Parameters

command	The command string, should be \0 ended.
expectation	The expectation string.

Returns

Definition at line 94 of file SIM900.h.

4.5.3.12 bool SIM900::sendCommandExpecting (const char * *command*, const char * *expectation*) [inline]

Sends a command expecting some result.

DEFAULT TIMEOUT DEFAULT APPEND

Parameters

command	The command string, should be \0 ended.
expectation	The expectation string.

Returns

Definition at line 108 of file SIM900.h.

4.5.3.13 void SIM900::setCommandEcho (bool *echo*)

Configures echo mode.

Parameters

echo	

Definition at line 81 of file SIM900.cpp.

4.5.4 Member Data Documentation

4.5.4.1 bool SIM900::echo [private]

Using echo.

Definition at line 34 of file SIM900.h.

4.5.4.2 unsigned char SIM900::rxBuffer[SIM900_RX_BUFFER_SIZE] [private]

RX buffer.

Definition at line 24 of file SIM900.h.

4.5.4.3 unsigned char* SIM900::rxPointer [private]

Pointer to the RX buffer.

Definition at line 29 of file SIM900.h.

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

- SIM900.h
- SIM900.cpp

4.6 Sms Class Reference

`#include <Sms.h>`

Public Member Functions

- virtual unsigned char remove (unsigned char index, unsigned char flags)=0
- virtual unsigned char format (bool format)=0
- virtual unsigned char bringUp ()=0
- virtual unsigned char obtainIp (unsigned char *buf)=0
- virtual unsigned char status ()=0
- virtual unsigned char configureDns (const char *primary, const char *secondary)=0
- virtual unsigned char open (unsigned char mode, unsigned char *address, unsigned char port)=0
- virtual unsigned char open (char connection, unsigned char mode, unsigned char *address, unsigned char port)=0
- virtual unsigned char close (char connection)=0
- virtual unsigned char resolve (unsigned char *name, unsigned char *buf, unsigned int len)=0
- virtual unsigned char send (unsigned char *buf)=0
- virtual unsigned char send (char connection, unsigned char *buf, unsigned int len)=0
- virtual unsigned char setUpServer (unsigned char mode, unsigned int port)=0
- virtual unsigned char shutdown ()=0

4.6.1 Detailed Description

Arduino - Gsm driver.

Sms.h

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Sms.h.

4.6.2 Member Function Documentation

4.6.2.1 virtual unsigned char Sms::bringUp ( ) [pure virtual]

Bring Up Wireless Connection with Sms or CSD.

Connects to the Sms network.

Returns

4.6.2.2 virtual unsigned char Sms::close (char *connection*) [pure virtual]

Close TCP or UDP Connection.

Returns

4.6.2.3 virtual unsigned char Sms::configureDns (const char * *primary*, const char * *secondary*) [pure virtual]

Configure Domain Name Server.

Returns

4.6.2.4 virtual unsigned char Sms::format (bool *format*) [pure virtual]

Select SMS Message Format.

Parameters

format	Message format.

Returns

4.6.2.5 virtual unsigned char Sms::obtainIp (unsigned char * *buf*) [pure virtual]

Get Local IP Address.

Returns the the IP address assigned from Sms or CSD in 4 bytes format.

Parameters

entry	Phonebook entry.

Returns

4.6.2.6 virtual unsigned char Sms::open (unsigned char *mode,* unsigned char * *address,* unsigned char *port*) [pure virtual]

Start Up TCP or UDP Connection.

Returns

4.6.2.7 virtual unsigned char Sms::open (char *connection,* unsigned char *mode,* unsigned char * *address,* unsigned char *port*) [pure virtual]

Start Up TCP or UDP Connection.

Returns

4.6.2.8 virtual unsigned char Sms::remove (unsigned char *index,* unsigned char *flags*) [pure virtual]

Delete SMS Message.

Parameters

index	Message location.
flags	Deletion flags.

Returns

4.6.2.9 virtual unsigned char Sms::resolve (unsigned char * *name,* unsigned char * *buf,* unsigned int *len*) [pure virtual]

Query the IP Address of Given Domain Name.

Returns

4.6.2.10 virtual unsigned char Sms::send (unsigned char * *buf*) [pure virtual]

Send Data Through TCP or UDP Connection.

Returns

4.6.2.11 virtual unsigned char Sms::send (char *connection,* unsigned char * *buf,* unsigned int *len*) [pure virtual]

Send Data Through TCP or UDP Connection.

Returns

4.6.2.12 virtual unsigned char Sms::setUpServer (unsigned char *mode,* unsigned int *port*) [pure virtual]

Configure Module as Server.

Returns

4.6.2.13 virtual unsigned char Sms::shutdown ( ) [pure virtual]

Deactivate Sms PDP Context.

Returns

4.6.2.14 virtual unsigned char Sms::status ( ) [pure virtual]

Query Current Connection Status.

Returns

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

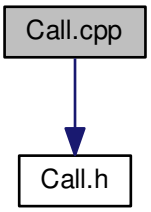
- Sms.h

5 File Documentation

5.1 Call.cpp File Reference

```
#include "Call.h"
```

Include dependency graph for Call.cpp:



Macros

- #define __ARDUINO_DRIVER_GSM_CALL_CPP__ 1

5.1.1 Macro Definition Documentation

5.1.1.1 #define __ARDUINO_DRIVER_GSM_CALL_CPP__ 1

Arduino - Gsm driver.

Call.cpp

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

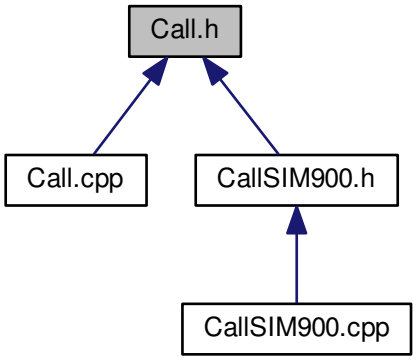
Definition at line 12 of file Call.cpp.

5.2 Call.cpp

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_CALL_CPP__
00012 #define __ARDUINO_DRIVER_GSM_CALL_CPP__ 1
00013
00014 #include "Call.h"
00015
00016 #endif /* __ARDUINO_DRIVER_GSM_CALL_CPP__ */
```

5.3 Call.h File Reference

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



Classes

- class Call

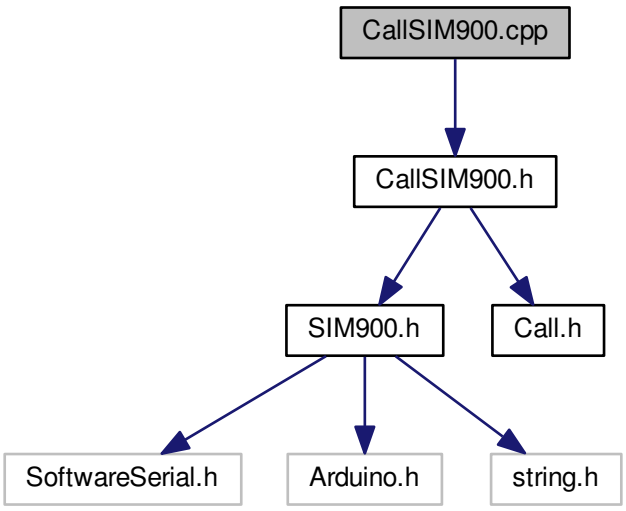
5.4 Call.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_CALL_H__
00012 #define __ARDUINO_DRIVER_GSM_CALL_H__ 1
00013
00014 class Call {
00015
00016 public:
00017
00025     virtual unsigned char answer() = 0;
00026
00036     virtual unsigned char callNumber(unsigned char *number) = 0;
00037
00046     virtual unsigned char callFromPhonebook(unsigned char position) = 0;
00047
00057     virtual unsigned char callByPhonebookMatch(unsigned char *entry) = 0;
00058
00066     virtual unsigned char redial() = 0;
00067
```

```
00073     virtual unsigned char disconnect() = 0;
00074 };
00075
00076 #endif /* __ARDUINO_DRIVER_GSM_CALL_H__ */
```

5.5 CallSIM900.cpp File Reference

`#include "CallSIM900.h"`
Include dependency graph for CallSIM900.cpp:



Macros

- #define __ARDUINO_DRIVER_GSM_CALL_SIM900_CPP__ 1

5.5.1 Macro Definition Documentation

5.5.1.1 #define __ARDUINO_DRIVER_GSM_CALL_SIM900_CPP__ 1

Arduino - Gsm driver.

CallSIM900.cpp

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file CallSIM900.cpp.

5.6 CallSIM900.cpp

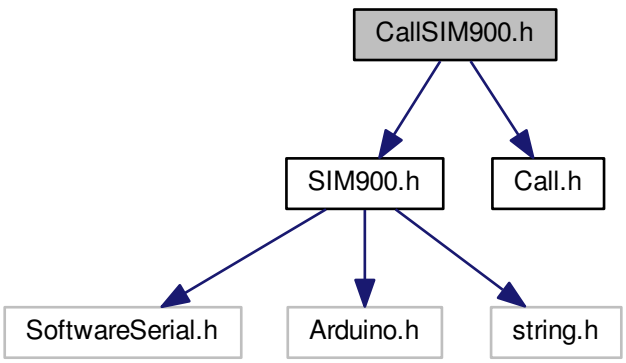
```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_CALL_SIM900_CPP__
```

```
00012 #define __ARDUINO_DRIVER_GSM_CALL_SIM900_CPP__ 1
00013
00014 #include "CallSIM900.h"
00015
00016 CallSIM900::CallSIM900(SIM900 *sim) : sim(sim) {
00017 }
00018
00019 unsigned char CallSIM900::answer() {
00020 //    sim->sendCommand((const char*) "ATA\n", 4);
00021 //    return sim->receiveStatus();
00022 }
00023
00024 unsigned char CallSIM900::callNumber(unsigned char *number) {
00025
00026 }
00027
00028 unsigned char CallSIM900::callFromPhonebook(unsigned char position) {
00029
00030 }
00031
00032 unsigned char CallSIM900::callByPhonebookMatch(unsigned char *entry) {
00033
00034 }
00035
00036 unsigned char CallSIM900::redial() {
00037
00038 }
00039
00040 unsigned char CallSIM900::disconnect() {
00041
00042 }
00043
00044 #endif /* __ARDUINO_DRIVER_GSM_CALL_SIM900_CPP__ */
```

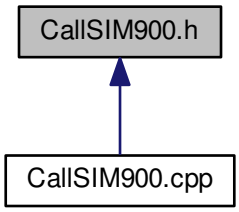
5.7 CallSIM900.h File Reference

```
#include <SIM900.h>
#include <Call.h>
```

Include dependency graph for CallSIM900.h:



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



Classes

- class CallSIM900

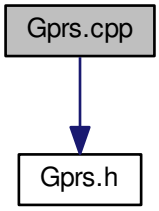
5.8 CallSIM900.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_CALL_SIM900_H__
00012 #define __ARDUINO_DRIVER_GSM_CALL_SIM900_H__ 1
00013
00014 #include <SIM900.h>
00015 #include <Call.h>
00016
00017 class CallSIM900 : public Call {
00018
00019     SIM900 *sim;
00020
00021 public:
00022
00023     CallSIM900(SIM900 *sim);
00024
00032     virtual unsigned char answer();
00033
00043     virtual unsigned char callNumber(unsigned char *number);
00044
00053     virtual unsigned char callFromPhonebook(unsigned char position);
00054
00064     virtual unsigned char callByPhonebookMatch(unsigned char *entry);
00065
00073     virtual unsigned char redial();
00074
00080     virtual unsigned char disconnect();
00081 };
00082
00083 #endif /* __ARDUINO_DRIVER_GSM_CALL_SIM900_H__ */
```

5.9 Gprs.cpp File Reference

```
#include "Gprs.h"
```

Include dependency graph for Gprs.cpp:



Macros

- #define __ARDUINO_DRIVER_GSM_GPRS_CPP__ 1

5.9.1 Macro Definition Documentation

5.9.1.1 #define __ARDUINO_DRIVER_GSM_GPRS_CPP__ 1

Arduino - Gsm driver.

Gprs.cpp

Interface to calls.

Author

Dalmir da Silva dalmirdasilva@gmail.com

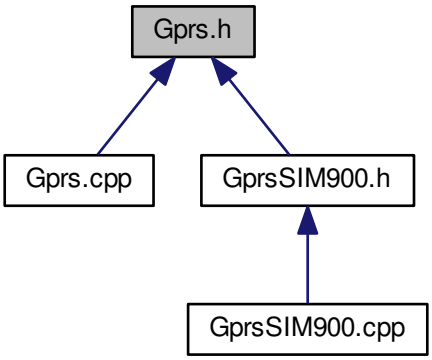
Definition at line 12 of file Gprs.cpp.

5.10 Gprs.cpp

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_GPRS_CPP__
00012 #define __ARDUINO_DRIVER_GSM_GPRS_CPP__ 1
00013
00014 #include "Gprs.h"
00015
00016 #endif /* __ARDUINO_DRIVER_GSM_GPRS_CPP__ */
```

5.11 Gprs.h File Reference

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



Classes

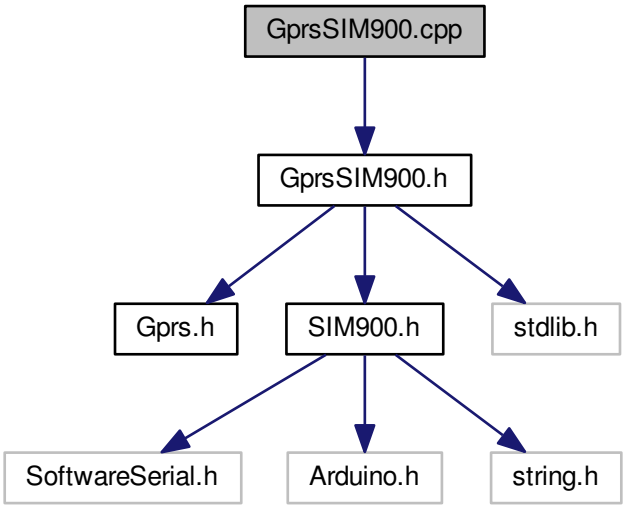
- class Gprs

5.12 Gprs.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_GPRS_H__
00012 #define __ARDUINO_DRIVER_GSM_GPRS_H__ 1
00013
00014 class Gprs {
00015 public:
00016
00025     virtual unsigned char useMultiplexer(bool use) = 0;
00026
00037     virtual unsigned char attach(const char *apn, const char *login, const char *password) = 0;
00038
00046     virtual unsigned char bringUp() = 0;
00047
00057     virtual unsigned char obtainIp(unsigned char *buf) = 0;
00058
00064     virtual unsigned char status() = 0;
00065
00071     virtual unsigned char configureDns(const char *primary, const char *secondary) = 0;
00072
00078     virtual unsigned char open(const char *mode, const char *address, unsigned int port) = 0;
00079
00085     virtual unsigned char open(char connection, const char *mode, const char *address, unsigned int
      port) = 0;
00086
00093     virtual unsigned char close(char connection) = 0;
00094
00100     virtual unsigned char close() = 0;
00101
00107     virtual unsigned char resolve(const char *name, unsigned char *buf, unsigned int len) = 0;
00108
00114     virtual unsigned char send(unsigned char *buf, unsigned int len) = 0;
00115
00121     virtual unsigned char send(char connection, unsigned char *buf, unsigned int len) = 0;
00122
00128     virtual unsigned char setUpServer(unsigned char mode, unsigned int port) = 0;
00129
00135     virtual unsigned char shutdown() = 0;
00136 };
00137
00138 #endif /* __ARDUINO_DRIVER_GSM_GPRS_H__ */
```

5.13 GprsSIM900.cpp File Reference

`#include "GprsSIM900.h"`
Include dependency graph for GprsSIM900.cpp:



Macros

- #define __ARDUINO_DRIVER_GSM_GPRS_SIM900_CPP__ 1

5.13.1 Macro Definition Documentation

5.13.1.1 #define __ARDUINO_DRIVER_GSM_GPRS_SIM900_CPP__ 1

Arduino - Gsm driver.

GprsSIM900.h

GPRS connection using SIM900.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file GprsSIM900.cpp.

5.14 GprsSIM900.cpp

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_GPRS_SIM900_CPP__
00012 #define __ARDUINO_DRIVER_GSM_GPRS_SIM900_CPP__ 1
00013
00014 #include "GprsSIM900.h"
00015
00016 GprsSIM900::GprsSIM900(SIM900 *sim) : sim(sim) {
00017     multiplexed = false;
00018 }
```

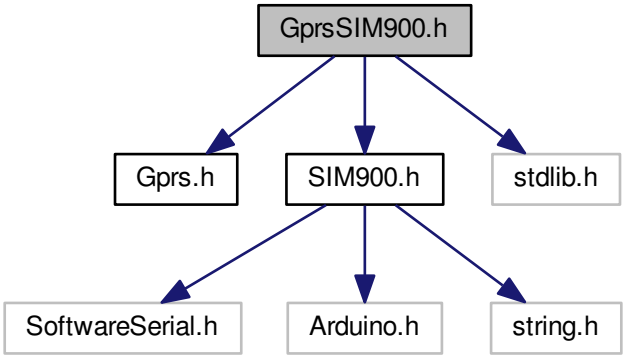
```
00019
00020 void GprsSIM900::begin(long bound) {
00021     sim->begin(bound);
00022     sim->setCommandEcho(false);
00023     sim->sendCommand("+CIPSHUT", (bool) true);
00024 }
00025
00026 unsigned char GprsSIM900::useMultiplexer(bool use) {
00027     char command[] = "+CIPMUX=0";
00028     if (use) {
00029         command[8] = '1';
00030     }
00031     multiplexed = use;
00032     return (unsigned char) sim->sendCommandExpecting(command, "OK", (bool) true);
00033 }
00034
00035 unsigned char GprsSIM900::attach(const char *apn, const char *login, const char *password
      ) {
00036     bool expected;
00037     sim->write("AT+CSTT=\"");
00038     sim->write(apn);
00039     sim->write("\",\"");
00040     sim->write(login);
00041     sim->write("\",\"");
00042     sim->write(password);
00043     expected = sim->sendCommandExpecting("\"", "OK");
00044     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00045 }
00046
00047 unsigned char GprsSIM900::bringUp() {
00048     bool expected;
00049     expected = sim->sendCommandExpecting("+CIICR", "OK", true, 20000);
00050     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00051 }
00052
00053 unsigned char GprsSIM900::obtainIp(unsigned char *buf) {
00054     sim->sendCommand("+CIFSR", (bool) true);
00055     if (!sim->doesResponseContains("ERROR")) {
00056         char n = 0, j, i = 0, part[4] = {0};
00057         unsigned char* p = sim->getLastResponse();
00058         while (*p != '\0' && n < 4) {
00059             if (*p >= '0' && *p <= '9') {
00060                 part[i++ % 3] = *p;
00061             }
00062             if (*p == '.') {
00063                 n++;
00064                 *(buf++) = (unsigned char) atoi(part);
00065                 for (j = 0; j < 4; j++) {
00066                     part[j] = 0;
00067                 }
00068                 i = 0;
00069             }
00070             p++;
00071         }
00072         if (i > 0) {
00073             n++;
00074             *(buf++) = (unsigned char) atoi(part);
00075         }
00076         return (n == 4) ? GprsSIM900::OK : GprsSIM900::ERROR;
00077     }
00078     return GprsSIM900::ERROR;
00079 }
00080
00081 unsigned char GprsSIM900::status() {
00082     sim->sendCommand("+CIFSR", (bool) true);
00083 }
00084
00085 unsigned char GprsSIM900::configureDns(const char *primary, const char *secondary)
      {
00086     bool expected;
00087     sim->write("AT+CDNSCFG=\"");
00088     sim->write(primary);
00089     sim->write("\",\"");
00090     sim->write(secondary);
00091     expected = sim->sendCommandExpecting("\"", "OK");
00092     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00093 }
00094
00095 unsigned char GprsSIM900::open(const char *mode, const char *address, unsigned int port) {
00096     return open(-1, mode, address, port);
00097 }
00098
00099 unsigned char GprsSIM900::open(char connection, const char *mode, const char *address,
      unsigned int port) {
```

```
00100     bool expected;
00101     sim->write("AT+CIPSTART=");
00102     if (connection != -1) {
00103         sim->write('0' + connection);
00104         sim->write(',');
00105     }
00106     sim->write('"');
00107     sim->write(mode);
00108     sim->write("\",\"");
00109     sim->write(address);
00110     sim->write("\",\"");
00111     sim->print(port, DEC);
00112     expected = sim->sendCommandExpecting("\"", "OK");
00113     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00114 }
00115
00116 unsigned char GprsSIM900::close(char connection) {
00117     bool expected;
00118     sim->write("AT+CIPCLOSE=1");
00119     if (connection != -1) {
00120         sim->write(',');
00121         sim->write('0' + connection);
00122     }
00123     expected = sim->sendCommandExpecting("", "OK");
00124     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00125 }
00126
00127 unsigned char GprsSIM900::close() {
00128     return close(-1);
00129 }
00130
00131 unsigned char GprsSIM900::resolve(const char *name, unsigned char *buf, unsigned int len
      ) {
00132     bool expected;
00133     sim->write("AT+CDNSGIP=");
00134     expected = sim->sendCommandExpecting(name, "OK");
00135     return (unsigned char) (expected ? GprsSIM900::OK :
      GprsSIM900::ERROR);
00136 }
00137
00138 unsigned char GprsSIM900::send(unsigned char *buf, unsigned int len) {
00139     //AT+CIPSEND=<length>
00140     // sim->write(buf, len);
00141 }
00142
00143 unsigned char GprsSIM900::send(char connection, unsigned char *buf, unsigned int len) {
00144     sim->readBytes((char *)buf, len);
00145 }
00146
00147 unsigned char GprsSIM900::setUpServer(unsigned char mode, unsigned int port) {
00148 }
00149
00150 unsigned char GprsSIM900::shutdown() {
00151 }
00152
00153 #endif /* __ARDUINO_DRIVER_GSM_GPRS_SIM900_CPP__ */
00154
```

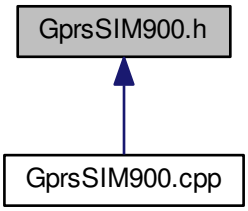
5.15 GprsSIM900.h File Reference

```
#include <Gprs.h>
#include <SIM900.h>
#include <stdlib.h>
```

Include dependency graph for GprsSIM900.h:



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



Classes

- class GprsSIM900

Macros

- #define GPRS_SIM900_MAX_COMMAND_LENGHT 64

5.15.1 Macro Definition Documentation

5.15.1.1 #define GPRS_SIM900_MAX_COMMAND_LENGHT 64

Arduino - Gsm driver.

GprsSIM900.h

GPRS connection using SIM900.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file GprsSIM900.h.

5.16 GprsSIM900.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_GPRS_SIM900_H__
00012 #define __ARDUINO_DRIVER_GSM_GPRS_SIM900_H__ 1
00013
00014 #define GPRS_SIM900_MAX_COMMAND_LENGHT 64
00015
00016 #include <Gprs.h>
00017 #include <SIM900.h>
00018 #include <stdlib.h>
00019
00020 class GprsSIM900 : public Gprs {
00021
00025     SIM900 *sim;
00026
00030     bool multiplexed;
00031
00032 public:
00033
00034     enum OperationResult {
00035         OK = 0,
00036         ERROR = 1,
00037         COMMAND_TOO_LONG = 2
00038     };
00039
00045     GprsSIM900(SIM900 *sim);
00046
00052     void begin(long bound);
00053
00062     unsigned char useMultiplexer(bool use);
00063
00074     unsigned char attach(const char *apn, const char *login, const char *password);
00075
00083     unsigned char bringUp();
00084
00094     unsigned char obtainIp(unsigned char *buf);
00095
00101     unsigned char status();
00102
00108     unsigned char configureDns(const char *primary, const char *secondary);
00109
00115     unsigned char open(const char *mode, const char *address, unsigned int port);
00116
00122     unsigned char open(char connection, const char *mode, const char *address, unsigned int port);
00123
00130     unsigned char close(char connection);
00131
00137     unsigned char close();
00138
00144     unsigned char resolve(const char *name, unsigned char *buf, unsigned int len);
00145
00151     unsigned char send(unsigned char *buf, unsigned int len);
00152
00158     unsigned char send(char connection, unsigned char *buf, unsigned int len);
00159
00165     unsigned char setUpServer(unsigned char mode, unsigned int port);
00166
00172     unsigned char shutdown();
00173 };
00174
00175 #endif /* __ARDUINO_DRIVER_GSM_GPRS_SIM900_H__ */
```

5.17 GsmSIM900.cpp File Reference

5.18 GsmSIM900.cpp

```
00001
```

5.19 GsmSIM900.h File Reference

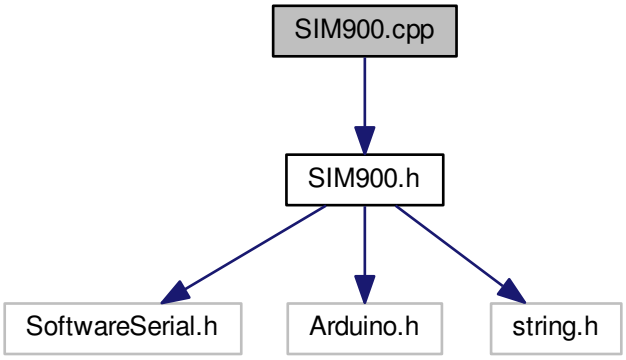
5.20 GsmSIM900.h

```
00001
```

5.21 SIM900.cpp File Reference

```
#include "SIM900.h"
```

Include dependency graph for SIM900.cpp:



Macros

- #define __ARDUINO_DRIVER_GSM_SIM900_CPP__ 1

5.21.1 Macro Definition Documentation

5.21.1.1 #define __ARDUINO_DRIVER_GSM_SIM900_CPP__ 1

Arduino - Gsm driver.

SIM900.cpp

SIM900 implementation of the SIM900 modem.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file SIM900.cpp.

5.22 SIM900.cpp

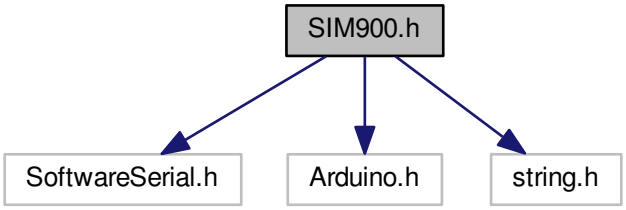
```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_SIM900_CPP__
00012 #define __ARDUINO_DRIVER_GSM_SIM900_CPP__ 1
00013
00014 #include "SIM900.h"
00015
00016 SIM900::SIM900(unsigned char receivePin, unsigned char transmitPin) : SoftwareSerial(
      receivePin, transmitPin) {
00017     rxBuffer[0] = '\0';
```

```
00018     echo = true;
00019 }
00020
00021 void SIM900::begin(long bound) {
00022     bool ready = false;
00023     unsigned char tries = 3;
00024     SoftwareSerial::begin(bound);
00025     do {
00026         ready = sendCommandExpecting((const char *)"AT", (const char *)"OK", (unsigned
      long) 100);
00027         if (!ready) {
00028             delay(100);
00029         } else {
00030             break;
00031         }
00032     } while (tries--);
00033 }
00034
00035 bool SIM900::sendCommandExpecting(const char *command, const char *expectation,
       bool append, unsigned long timeout) {
00036     if (sendCommand(command, append, timeout) == 0) {
00037         return false;
00038     }
00039     return doesResponseContains(expectation);
00040 }
00041
00042 bool SIM900::doesResponseContains(const char *expectation) {
00043     rxPointer = &rxBuffer[0];
00044     bool does = strstr((const char*) rxPointer, (const char*) expectation) != NULL;
00045     return does;
00046 }
00047
00048 int SIM900::sendCommand(const char *command, bool append, unsigned long timeout) {
00049     rxBuffer[0] = '\0';
00050     if (append) {
00051         print("AT");
00052     }
00053     println(command);
00054     readResponse(timeout);
00055 }
00056
00057 int SIM900::readResponse(unsigned long timeout) {
00058     int availableBytes;
00059     unsigned long start = millis();
00060     int pointer = 0;
00061     while (!available() && (millis() - start) < timeout);
00062     start = millis();
00063     do {
00064         availableBytes = available();
00065         if (availableBytes > 0) {
00066             if (pointer + availableBytes >= SIM900_RX_BUFFER_SIZE) {
00067                 availableBytes = SIM900_RX_BUFFER_SIZE - (pointer + 1);
00068             }
00069             if (availableBytes == 0) {
00070                 flush();
00071             } else {
00072                 readBytes((char *) &rxBuffer[pointer], availableBytes);
00073                 pointer += availableBytes;
00074                 rxBuffer[pointer] = 0;
00075             }
00076         }
00077     } while ((millis() - start) < timeout && availableBytes);
00078     return pointer;
00079 }
00080
00081 void SIM900::setCommandEcho(bool echo) {
00082     this->echo = echo;
00083     char command[] = "E0";
00084     if (echo) {
00085         command[1] = '1';
00086     }
00087     sendCommand(command, true, 100);
00088 }
00089
00090 #endif /* __ARDUINO_DRIVER_GSM_SIM900_CPP__ */
```

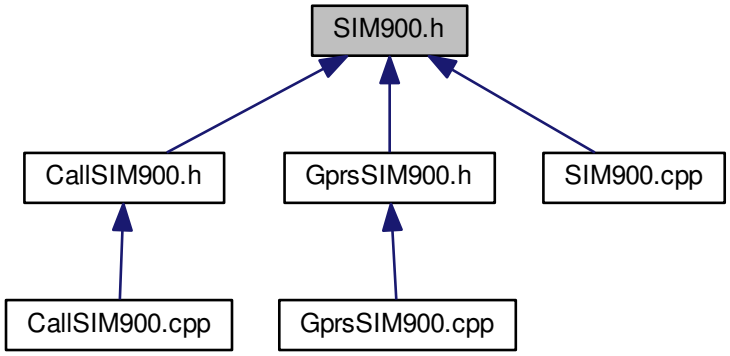
5.23 SIM900.h File Reference

```
#include <SoftwareSerial.h>
#include <Arduino.h>
#include <string.h>
```

Include dependency graph for SIM900.h:



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



Classes

- class SIM900

Macros

- #define SIM900_RX_BUFFER_SIZE 256

5.23.1 Macro Definition Documentation

5.23.1.1 #define SIM900_RX_BUFFER_SIZE 256

Arduino - Gsm driver.

SIM900.h

SIM900 implementation of the SIM900 modem.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 18 of file SIM900.h.

5.24 SIM900.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_SIM900_H__
00012 #define __ARDUINO_DRIVER_GSM_SIM900_H__ 1
00013
00014 #include <SoftwareSerial.h>
00015 #include <Arduino.h>
00016 #include <string.h>
00017
00018 #define SIM900_RX_BUFFER_SIZE 256
00019
00020 class SIM900 : public SoftwareSerial {
00024     unsigned char rxBuffer[SIM900_RX_BUFFER_SIZE];
00025
00029     unsigned char *rxPointer;
00030
00034     bool echo;
00035
00036 public:
00037
00043     SIM900(unsigned char receivePin, unsigned char transmitPin);
00044
00050     void begin(long bound);
00051
00057     unsigned char *getLastResponse() {
00058         rxPointer = &rxBuffer[0];
00059         return rxPointer;
00060     }
00061
00070     bool sendCommandExpecting(const char *command, const char *expectation, bool append
      , unsigned long timeout);
00071
00081     inline bool sendCommandExpecting(const char *command, const char *expectation, bool
       append) {
00082         return sendCommandExpecting(command, expectation, append, 1000);
00083     }
00084
00094     inline bool sendCommandExpecting(const char *command, const char *expectation,
      unsigned long timeout) {
00095         return sendCommandExpecting(command, expectation, false, timeout);
00096     }
00097
00108     inline bool sendCommandExpecting(const char *command, const char *expectation) {
00109         return sendCommandExpecting(command, expectation, (bool) false);
00110     }
00111
00118     bool doesResponseContains(const char *expectation);
00119
00127     int sendCommand(const char *command, bool append, unsigned long timeout);
00128
00138     inline int sendCommand(const char *command, bool append) {
00139         return sendCommand(command, append, 1000);
00140     }
00141
00151     inline int sendCommand(const char *command, unsigned long timeout) {
00152         return sendCommand(command, (bool) false, timeout);
00153     }
00154
00164     inline int sendCommand(const char *command) {
00165         return sendCommand(command, (bool) false);
00166     }
00167
00174     int readResponse(unsigned long timeout);
00175
00181     void setCommandEcho(bool echo);
00182 };
00183
00184 #endif /* __ARDUINO_DRIVER_GSM_SIM900_H__ */
```

5.25 Sms.cpp File Reference

5.26 Sms.cpp

```
00001
```

5.27 Sms.h File Reference

Classes

- class Sms

5.28 Sms.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_GSM_SMS_H__
00012 #define __ARDUINO_DRIVER_GSM_SMS_H__ 1
00013
00014 class Sms {
00015
00016 public:
00017
00026     virtual unsigned char remove(unsigned char index, unsigned char flags) = 0;
00027
00034     virtual unsigned char format(bool format) = 0;
00035
00043     virtual unsigned char bringUp() = 0;
00044
00054     virtual unsigned char obtainIp(unsigned char *buf) = 0;
00055
00061     virtual unsigned char status() = 0;
00062
00068     virtual unsigned char configureDns(const char *primary, const char *secondary) = 0;
00069
00075     virtual unsigned char open(unsigned char mode, unsigned char *address, unsigned char port) = 0;
00076
00082     virtual unsigned char open(char connection, unsigned char mode, unsigned char *address, unsigned
      char port) = 0;
00083
00089     virtual unsigned char close(char connection) = 0;
00090
00096     virtual unsigned char resolve(unsigned char *name, unsigned char *buf, unsigned int len) = 0;
00097
00103     virtual unsigned char send(unsigned char *buf) = 0;
00104
00110     virtual unsigned char send(char connection, unsigned char *buf, unsigned int len) = 0;
00111
00117     virtual unsigned char setUpServer(unsigned char mode, unsigned int port) = 0;
00118
00124     virtual unsigned char shutdown() = 0;
00125 };
00126
00127 #endif /* __ARDUINO_DRIVER_GSM_SMS_H__ */
```

Index