A6Lib

Generated by Doxygen 1.8.14

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 A6lib Class Reference 5

3.1.1 Detailed Description 6

3.1.2 Constructor & Destructor Documentation 6

3.1.2.1 A6lib() `[1/3]` 6

3.1.2.2 A6lib() `[2/3]` 6

3.1.2.3 A6lib() `[3/3]` 7

3.1.2.4 ∼A6lib() 7

3.1.3 Member Function Documentation 7

3.1.3.1 addHandler() 7

3.1.3.2 deleteSMS() 7

3.1.3.3 getDeviceStatus() 8

3.1.3.4 getFirmWareVer() 8

3.1.3.5 getIMEI() 8

3.1.3.6 getOperatorName() 9

3.1.3.7 getRealTimeClock() 9

3.1.3.8 getRealTimeClockString() 9

3.1.3.9 getRegisterStatus() 9

3.1.3.10 getRSSI() 10

3.1.3.11 getSignalQuality() 10

3.1.3.12 getSMSList() 10

3.1.3.13 getSMSSca() 10

3.1.3.14 handle() 11

3.1.3.15 hardReset() 11

3.1.3.16 onSMSReceived() 11

3.1.3.17 onSMSSent() 11

3.1.3.18 onSMSStorageFull() 12

3.1.3.19 powerUp() 12

3.1.3.20 readSMS() 12

3.1.3.21 sendCommand() 13

3.1.3.22 sendPDU() [1/2] 13

3.1.3.23 sendPDU() [2/2] 13

3.1.3.24 sendSMS() 14

3.1.3.25 sendUSSD() 14

3.1.3.26 setCharSet() 14

3.1.3.27 setSMSStorageArea() 15

3.1.3.28 softReset() 15

3.1.3.29 start() 15

3.1.3.30 waitForNetwork() 15

3.2 SMSInfo Class Reference 17

3.2.1 Constructor & Destructor Documentation 17

3.2.1.1 SMSInfo() 17

3.2.2 Member Data Documentation 17

3.2.2.1 dateTime 17

3.2.2.2 message 18

3.2.2.3 number 18

4 File Documentation **19**

4.1 A6lib.cpp File Reference 19

4.2 A6lib.h File Reference 19

4.2.1 Typedef Documentation 20

4.2.1.1 sms_full_cb_t 20

4.2.1.2 sms_rx_cb_t 20

4.2.1.3 sms_tx_cb_t 20

4.2.1.4 void_cb_t 20

4.2.2 Enumeration Type Documentation 20

4.2.2.1 CharSet 20

4.2.2.2 DeviceStatus 21

4.2.2.3 RegisterStatus 21

4.2.2.4 SMSRecordType 21

4.2.2.5 SMSStorageArea 22

4.3 pdu.h File Reference 22

4.3.1 Function Documentation 22

4.3.1.1 pdu_encode() 22

4.3.1.2 pdu_encodew() 23

Index **25**

Chapter 1

Class Index

1.1 Class List

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

A6lib
A library for controlling Ai-Thinker A6 GSM modem(also works with others like SIM800) 5
SMSInfo 17

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

A6lib.cpp 19
A6lib.h 19
pdu.h 22

Chapter 3

Class Documentation

3.1 A6lib Class Reference

A library for controlling Ai-Thinker A6 GSM modem(also works with others like SIM800).

```
#include <A6lib.h>
```

Public Member Functions

- A6lib (HardwareSerial ∗port)
- A6lib (SoftwareSerial ∗port)
- A6lib (uint8_t rx_pin, uint8_t tx_pin)
- ∼A6lib ()
- void handle ()
- bool start (uint8_t max_retry)
- bool waitForNetwork (unsigned long baud, uint16_t time_out)
- void powerUp (int pin)
- void softReset ()
- void hardReset (uint8_t pin)
- DeviceStatus getDeviceStatus ()
- String getFirmWareVer ()
- int getRSSI ()
- uint8_t getSignalQuality ()
- time_t getRealTimeClock ()
- String getRealTimeClockString (const String &format=String())
- String getIMEI ()
- String getSMSSca ()
- RegisterStatus getRegisterStatus ()
- String getOperatorName ()
- String sendUSSD (const String &ussd_code, uint16_t timeout=-1)
- bool setSMSStorageArea (SMSStorageArea)
- bool setCharSet (CharSet)
- bool sendSMS (const String &number, const String &text)
- bool sendPDU (const String &number, const String &content)
- bool sendPDU (const String &number, uint16_t ∗content, uint8_t len)
- SMSInfo readSMS (uint8_t index)
- bool deleteSMS (uint8_t index, bool del_all=false)
- int8_t getSMSList (int8_t ∗buff, uint8_t len, SMSRecordType record)
- void addHandler (void_cb_t)
- void onSMSSent (sms_tx_cb_t)
- void onSMSReceived (sms_rx_cb_t)
- void onSMSStorageFull (sms_full_cb_t)
- String sendCommand (const String &command, uint16_t reply_timeout=2000)

3.1.1 Detailed Description

A library for controlling Ai-Thinker A6 GSM modem(also works with others like SIM800).

An Arduino library for communicating with the AI-Thinker A6 GSM modem, It currently supports ESP8266 and A↩ VR architectures. This small lib mainly intended for Ai-Thinker A6 modem but may possiblly work with other GSM modems supporting standard AT command set (e.g SIM800, SIM900 ,...). Using this lib is straightforward, you can create an object of A6lib via HardwareSerial, SoftwareSerial or just two pin number for built in SoftwareSerial. Then you usually should power up your module (A6lib::powerUp()) and initlize A6lib object to start communicating with modem at desired baud rate. from now on, use public APIs to control your modem and get informations from it. Also there's a rich debugging part inisde the library, to enable it define DEBUG in your environment.

This lib has been modified to be asynchronous, so currently you can pass your functions to register APIs to catch these events:

1. SMS sent
2. SMS recevied
3. Storage area is full

Note

A note about A6lib::addHandler(): When you have some important tasks in your code for example reading keypad etc, you can add a main function for running those tasks and pass it to A6lib::addHandler(), when you pass a valid function, lib will call it whenever it's in waiting state (waiting for modem to reply at some time) and thus it'll prevent locking in that precious time.

To get start you can check out examples directory.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 A6lib() [1/3]

```
A6lib::A6lib (
            HardwareSerial * port )
```

Constructs A6lib object with the given serial *port*.

Parameters

port	HardwareSerial object for use inside A6lib.

3.1.2.2 A6lib() [2/3]

```
A6lib::A6lib (
              SoftwareSerial * port )
```

Constructs A6lib object with the given serial *port*.

Parameters

port	SoftwareSerial object for use inside A6lib

3.1.2.3 A6lib() [3/3]

```
A6lib::A6lib (
              uint8_t rx_pin,
              uint8_t tx_pin )
```

Constructs A6lib object with the given pin numbers. this is done by creating new SoftwareSerial object.

Parameters

tx_pin	SoftwareSerial TX pin
rx_pin	SoftwareSerial RX pin

3.1.2.4 ∼A6lib()

```
A6lib::∼A6lib ( )
```

Destroys A6lib object.

3.1.3 Member Function Documentation

3.1.3.1 addHandler()

```
void A6lib::addHandler (
              void_cb_t cb )
```

Add the A6lib main handler callback. A6lib will call this handler when it is inside the waiting routine. it'll prevent lock in your code when you have some critical tasks to run. Note: The result of passing loop() to this function is undefined!

3.1.3.2 deleteSMS()

```
bool A6lib::deleteSMS (
              uint8_t index,
              bool del_all = false )
```

Delete a SMS from modem prefered storage area. Note: if del_all is true, index will be ignored and all SMS will be deleted in storage area.

Parameters

index	sms index in storage area

Returns

true on success

3.1.3.3 getDeviceStatus()

```
DeviceStatus A6lib::getDeviceStatus ( )
```

get the current modem working status.

Returns

on of the DeviceStatus value.

3.1.3.4 getFirmWareVer()

```
String A6lib::getFirmWareVer ( )
```

Get the revision identification or firmware version of modem.

Returns

If success a String contain firmware version, and if fail an empty string.

3.1.3.5 getIMEI()

```
String A6lib::getIMEI ( )
```

Get the modem IMEI(serial number identification).

Returns

if success a string contain IMEI number, if fail an empty string.

3.1.3.6 getOperatorName()

```
String A6lib::getOperatorName ( )
```

Get the Network operator name. note that the name is read from SIM card.

Returns

if success a String contain the operator name, else an empty String

3.1.3.7 getRealTimeClock()

```
time_t A6lib::getRealTimeClock ( )
```

Get the real time from modem(the return value is not necessary up to date).

Returns

if success a value contain time as time_t(epoch), if fail an invalid(-1) value.

3.1.3.8 getRealTimeClockString()

```
String A6lib::getRealTimeClockString (
              const String & format = String() )
```

Get the real time string from modem. please refer to `http://www.cplusplus.com/reference/ctime/strftime/` for format specifier.

Returns

if success a string contain local time in format yyyy.MM.dd hh:mm:ss, if fail an empty string.

3.1.3.9 getRegisterStatus()

```
RegisterStatus A6lib::getRegisterStatus ( )
```

Get the network registration status of modem.

Returns

on of the RegisterStatus value

3.1.3.10 getRSSI()

```
int A6lib::getRSSI ( )
```

Get the modem signal strength based on RSSI(measured as dBm).

Returns

If success a value between -113dBm and -51dBm and if fail 0.

3.1.3.11 getSignalQuality()

```
uint8_t A6lib::getSignalQuality ( )
```

Get the modem signal quality level.

Returns

if success a value between 0-100 and if fail 255.

3.1.3.12 getSMSList()

```
int8_t A6lib::getSMSList (
             int8_t * buff,
             uint8_t len,
             SMSRecordType record )
```

Get the list of available SMS in prefered storage area.

Parameters

buff	input buffer to store SMS indexes.
len	size of buff
record	on of the SMSRecordType.

Returns

if fail -1, otherwise number of founded SMS.

3.1.3.13 getSMSSca()

```
String A6lib::getSMSSca ( )
```

Get the current SMS service center address from modem.

Returns

if success a string contain SCA, if fail an empty string

3.1.3.14 handle()

```
void A6lib::handle ( )
```

the main handler of A6lib object. this function needs to be called inside main loop regularly, for callbacks to work correctly.

3.1.3.15 hardReset()

```
void A6lib::hardReset (
             uint8_t pin )
```

This function will do a hard reset on module. It's recommended to do this via an NMOS. Note: it will take some time for module to start + register for network. You may also need to reinitilize module with A6lib::start().

Parameters

pin	the pin number which is connected to modem reset(RST) pin.

3.1.3.16 onSMSReceived()

```
void A6lib::onSMSReceived (
             sms_rx_cb_t cb )
```

This function will register your callback and will call it when new SMS arrives.

Parameters

cb	pointer to callback function

3.1.3.17 onSMSSent()

```
void A6lib::onSMSSent (
             sms_tx_cb_t cb )
```

This function will register your callback and will call it when a SMS is sent.

Parameters

cb	pointer to callback function

3.1.3.18 onSMSStorageFull()

```
void A6lib::onSMSStorageFull (
             sms_full_cb_t cb )
```

This function will register your callback and will call it when modem prefered storage area is full.

Parameters

cb	pointer to callback function

3.1.3.19 powerUp()

```
void A6lib::powerUp (
             int pin )
```

this optional function will keep the PWR pin of modem in high TTL at start up to correctly powering the module. A6 modem needs this pin to be in high TTL for about 2 sec.

Parameters

pin	the pin number which is connected to modem PWR pin(or PWR_KEY pin).

3.1.3.20 readSMS()

```
SMSInfo A6lib::readSMS (
             uint8_t index )
```

Read a SMS in modem prefered storage area

Parameters

index	sms index in storage area

Returns

a SMSInfo object contain SMS information(number+date+timestamp) on success, and if fail an empty SMSInfo object.

3.1.3.21 sendCommand()

```
String A6lib::sendCommand (
             const String & command,
             uint16_t reply_timeout = 2000 )
```

Send new command to modem. command should be a valid AT command, otherwise modem will return error with corresponding error code(this function will append
to command). Note: you may want to check modem is busy or not with A6lib::isBusy().

Parameters

command	the valid command to be sent with AT prefix
reply_timeout	the amount of time(as ms) we wait for reply

Returns

if success an string contain modem reply, otherwise contain error code

3.1.3.22 sendPDU() [1/2]

```
bool A6lib::sendPDU (
             const String & number,
             const String & content )
```

Send an ASCII SMS in PDU mode.

Parameters

number	the detination phone number which should begin with international code
content	the SMS content in ASCII and up to 160 chars

Returns

true on success

3.1.3.23 sendPDU() [2/2]

```
bool A6lib::sendPDU (
             const String & number,
             uint16_t * content,
             uint8_t len )
```

Send a UCS2 SMS in PDU mode.

Parameters

number	the detination phone number which should begin with international code
content	the SMS content coded in UCS2 format and up to 70 chars.
len	the number of UCS2 chars in *content*

Returns

true on success

3.1.3.24 sendSMS()

```
bool A6lib::sendSMS (
            const String & number,
            const String & text )
```

Send SMS (in text mode) to specified number.

Parameters

number	valid destination number without +
text	SMS content in ascii encoding

Returns

true on success

3.1.3.25 sendUSSD()

```
String A6lib::sendUSSD (
            const String & ussd_code,
            uint16_t timeout = -1 )
```

3.1.3.26 setCharSet()

```
bool A6lib::setCharSet (
            CharSet set )
```

set the module charset.

Parameters

charset	the required charset. could be on of the ::Charset value

Returns

true on success.

3.1.3.27 setSMSStorageArea()

```
bool A6lib::setSMSStorageArea (
             SMSStorageArea  )
```

3.1.3.28 softReset()

```
void A6lib::softReset ( )
```

This function implement a software restart on module(if suppoerted). Note: it will take some time for module to start + register for network. You may also need to reinitilize module with A6lib::start().

3.1.3.29 start()

```
bool A6lib::start (
             uint8_t max_retry )
```

This is the A6lib object initlizer routine. you must usually call this after restarting your modem following by A6lib↩
:waitForNetwork().

Parameters

max_retry	the maximum number of time A6lib object try to setup modem.

Returns

true on success

3.1.3.30 waitForNetwork()

```
bool A6lib::waitForNetwork (
             unsigned long baud,
             uint16_t time_out )
```

This method will wait for modem to trigger the registration indication which is the result of correct netowrk registration. you must call this usually before A6lib::start().

Parameters

baud	the desired baud rate to start with
time_out	the maximum amount of time A6lib object wait for network registration indication.

Returns

true on success

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

- A6lib.h
- A6lib.cpp

3.2 SMSInfo Class Reference

```
#include <A6lib.h>
```

Public Member Functions

- SMSInfo ()

Public Attributes

- String number
- String dateTime
- String message

3.2.1 Constructor & Destructor Documentation

3.2.1.1 SMSInfo()

```
SMSInfo::SMSInfo ( )  [inline]
```

3.2.2 Member Data Documentation

3.2.2.1 dateTime

```
String SMSInfo::dateTime
```

3.2.2.2 message

```
String SMSInfo::message
```

3.2.2.3 number

```
String SMSInfo::number
```

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

- A6lib.h

Chapter 4

File Documentation

4.1 A6lib.cpp File Reference

```
#include <stdio.h>
#include <stdarg.h>
#include "pdu.h"
#include "A6lib.h"
```

4.2 A6lib.h File Reference

```
#include <time.h>
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <HardwareSerial.h>
```

Classes

- class SMSInfo
- class A6lib

  A library for controlling Ai-Thinker A6 GSM modem(also works with others like SIM800).

Typedefs

- typedef void(∗ void_cb_t) (void)
- typedef void(∗ sms_rx_cb_t) (uint8_t indx, const SMSInfo &)
- typedef void(∗ sms_tx_cb_t) (void)
- typedef void_cb_t sms_full_cb_t

Enumerations

- enum DeviceStatus { Status_Ready = 0, Status_Unknown = 2, Status_Ringing = 3, Status_Call_In_Progress = 4 }
- enum CharSet { Gsm, Ucs2, Hex, Pccp936 }
- enum RegisterStatus {
  NotRegistered = 0, Registered_HomeNetwork = 1, Searching_To_Register = 2, Register_Denied = 3,
  Unknown = 4, Registered_Roaming = 5 }
- enum SMSStorageArea {
  ME = 1, SM, MT, SM_P,
  ME_P }
- enum SMSRecordType { All, Unread, Read }

4.2.1 Typedef Documentation

4.2.1.1 sms_full_cb_t

```
typedef void_cb_t sms_full_cb_t
```

4.2.1.2 sms_rx_cb_t

```
typedef void(* sms_rx_cb_t) (uint8_t indx, const SMSInfo &)
```

4.2.1.3 sms_tx_cb_t

```
typedef void(* sms_tx_cb_t) (void)
```

4.2.1.4 void_cb_t

```
typedef void(* void_cb_t) (void)
```

4.2.2 Enumeration Type Documentation

4.2.2.1 CharSet

```
enum CharSet
```

Enumerator

Gsm	
Ucs2	
Hex	
Pccp936	

4.2.2.2 DeviceStatus

```
enum DeviceStatus
```

Enumerator

Status_Ready	
Status_Unknown	
Status_Ringing	
Status_Call_In_Progress	

4.2.2.3 RegisterStatus

```
enum RegisterStatus
```

Enumerator

NotRegistered	
Registered_HomeNetwork	
Searching_To_Register	
Register_Denied	
Unknown	
Registered_Roaming	

4.2.2.4 SMSRecordType

```
enum SMSRecordType
```

Enumerator

All	
Unread	
Read	

4.2.2.5 SMSStorageArea

```
enum SMSStorageArea
```

Enumerator

ME	
SM	
MT	
SM↩_P	
ME↩_P	

4.3 pdu.h File Reference

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

Functions

- int pdu_encode (const char ∗sca, const char ∗phone, const char ∗text, uint8_t text_len, uint8_t ∗pdu, uint8_t pdu_size)

  Encode input SMS text (which is coded in ASCII) into a SMS-SUBMIT pdu.
- int pdu_encodew (const char ∗sca, const char ∗phone, const uint16_t ∗text, uint8_t text_len, uint8_t ∗pdu, uint8_t pdu_size)

  Encode input SMS text (which is coded in UCS2) into a SMS-SUBMIT pdu.

4.3.1 Function Documentation

4.3.1.1 pdu_encode()

```
int pdu_encode (
            const char ∗ sca,
            const char ∗ phone,
            const char ∗ text,
            uint8_t text_len,
            uint8_t ∗ pdu,
            uint8_t pdu_size )
```

Encode input SMS *text* (which is coded in ASCII) into a SMS-SUBMIT pdu.

Parameters

sca	a null terminated string contain SMS service center address
phone	a null terminated string contain destination phone number
text	the SMS content in ASCII
text_len	the number of chars in SMS content(could be up to 160 char long)
pdu	the input buffer which is going to hold the final pdu
pdu_size	the size of input pdu buffer

Returns

if success a positive value represent number of pdu octets written, if fail a negative value represent error code

4.3.1.2 pdu_encodew()

```
int pdu_encodew (
            const char * sca,
            const char * phone,
            const uint16_t * text,
            uint8_t text_len,
            uint8_t * pdu,
            uint8_t pdu_size )
```

Encode input SMS *text* (which is coded in UCS2) into a SMS-SUBMIT pdu.

Parameters

sca	a null terminated string contain SMS service center address
phone	a null terminated string contain destination phone number
text	the SMS content coded in UCS2 coding scheme
text_len	the number of UCS2 chars in SMS content(could be up to 70 char long)
pdu	the input buffer which is going to hold the final pdu
pdu_size	the size of input pdu buffer

Returns

if success a positive value represent number of pdu octets written, if fail a negative value represent error code

Index

∼A6lib
    A6lib, 7

A6lib, 5
    ∼A6lib, 7
    A6lib, 6, 7
    addHandler, 7
    deleteSMS, 7
    getDeviceStatus, 8
    getFirmWareVer, 8
    getIMEI, 8
    getOperatorName, 8
    getRSSI, 9
    getRealTimeClock, 9
    getRealTimeClockString, 9
    getRegisterStatus, 9
    getSMSList, 10
    getSMSSca, 10
    getSignalQuality, 10
    handle, 11
    hardReset, 11
    onSMSReceived, 11
    onSMSSent, 11
    onSMSStorageFull, 12
    powerUp, 12
    readSMS, 12
    sendCommand, 13
    sendPDU, 13
    sendSMS, 14
    sendUSSD, 14
    setCharSet, 14
    setSMSStorageArea, 15
    softReset, 15
    start, 15
    waitForNetwork, 15
A6lib.cpp, 19
A6lib.h, 19
    CharSet, 20
    DeviceStatus, 21
    RegisterStatus, 21
    SMSRecordType, 21
    SMSStorageArea, 22
    sms_full_cb_t, 20
    sms_rx_cb_t, 20
    sms_tx_cb_t, 20
    void_cb_t, 20
addHandler
    A6lib, 7

CharSet
    A6lib.h, 20

dateTime
    SMSInfo, 17
deleteSMS
    A6lib, 7
DeviceStatus
    A6lib.h, 21

getDeviceStatus
    A6lib, 8
getFirmWareVer
    A6lib, 8
getIMEI
    A6lib, 8
getOperatorName
    A6lib, 8
getRSSI
    A6lib, 9
getRealTimeClock
    A6lib, 9
getRealTimeClockString
    A6lib, 9
getRegisterStatus
    A6lib, 9
getSMSList
    A6lib, 10
getSMSSca
    A6lib, 10
getSignalQuality
    A6lib, 10

handle
    A6lib, 11
hardReset
    A6lib, 11

message
    SMSInfo, 17

number
    SMSInfo, 18

onSMSReceived
    A6lib, 11
onSMSSent
    A6lib, 11
onSMSStorageFull
    A6lib, 12

pdu.h, 22

pdu_encode, 22
pdu_encodew, 23
pdu_encode
pdu.h, 22
pdu_encodew
pdu.h, 23
powerUp
A6lib, 12

readSMS
A6lib, 12
RegisterStatus
A6lib.h, 21

SMSInfo, 17
dateTime, 17
message, 17
number, 18
SMSInfo, 17
SMSRecordType
A6lib.h, 21
SMSStorageArea
A6lib.h, 22
sendCommand
A6lib, 13
sendPDU
A6lib, 13
sendSMS
A6lib, 14
sendUSSD
A6lib, 14
setCharSet
A6lib, 14
setSMSStorageArea
A6lib, 15
sms_full_cb_t
A6lib.h, 20
sms_rx_cb_t
A6lib.h, 20
sms_tx_cb_t
A6lib.h, 20
softReset
A6lib, 15
start
A6lib, 15

void_cb_t
A6lib.h, 20

waitForNetwork
A6lib, 15