| File: | ALAMIN*MotorolaC550*howto.txt |

| Author: | Darek, darsliwa@poczta.onet.pl |

Date: August 2006

This howto is about:

- installation and configuration of ***Alamin*** software;
- connecting Motorola C550 GSM handset with built−in USB modem to PC.

Motorola C550 GSM phone (handset) has ability to work as GSM/GPRS modem and SMS terminal (for sending and receiving SMS messages via computer). Motorola C550 works perfectly with `Alamin` application after some fine tuning and software patching.

2006/08/10−08/26 (test)

2006/08/31 (production)

Alamin GSM SMS Gateway and Motorola C550 handset

ABOUT ALAMIN

Alamin is a wonderfull IP <−> SMS gateway aplication written in (my favourite :−) Perl. The main difference between Alamin and other alike programs is in the way SMS−es are sent. Other programs typically use *email−to−sms* or *web−to−sms* internet gateways managed by particular GSM operators. There are many drawback of such approach:

- there are no standard ways of interfacing with both gateways (*email−to−sms* and *web−to−sms*) − so change of GSM operator requires re−write of code;
- established de facto standard for particular operator is not valid anymore when, for instance, layout of *web−to−sms* web page gets changed;
- GSM operators often impose various limits (like hourly, daily, weekly limits of SMS sent from one IP) or apply additional delays in processing messages;
- your internet connection must be working − so you cannot use this method in conjunction with monitoring software for sending emergency notification about ... lost internet connection or router failure.

In contrast, Alamin either uses GSM handset/modem or direct IP connection to operator's SMS Center (SMSC in short) for SMS message sending and receiving. Thus GSM operator treats these SMS−es like any other messages written manually by you and delivers them without extra delay. The drawback of this method is that you have to invest in hardware (GSM modem or handset with modem functionality) and then incure charges for your SIM card (monthly subscriber's fee or prepaid charges). However, droping cost of SMS−es makes these expenses reasonable.

ALAMIN'S MODULES AND KEY FILES

The package consist of the following binary modules:

- gsgc −> Alamin GSM SMS Gateway Client. This is the ***end−user's*** command−line utility for preparing SMS messages.
- gsgcmd −> GSM SMS Gateway Core Module Daemon. It receives connections from the IP network (i.e. from `gsgc` clients) using the sms query protocol (smsqp) and saves messages in their queue.
- gsgdb2sms −> Alamin GSM SMS Gateway mysql reader. `gsgdb2sms` is the component that reads a mysql database table and calls gsgc (the client) to send messages. Database connect data, table name and fields can be configured using its config file. `gsgdb2sms` updates a field each time a message is correctly sent.
- gsgmdd −> GSM SMS Gateway Message Dispatcher Daemon. It periodically connects to the gsm device to check for new incoming messages, runs input message proccesor scripts (default location: `/usr/lib/alamin` directory) for each input message, and sends queued messages.
- gsgsmppin −> Alamin GSM SMS Gateway SMPP input interface. `gsgsmppin` is the component that retrieves messages queued into a SMSC with a SMPP interface, to be passed to the gsgimp defined. Connect parameters are configured using its config file.
- gsgsmppout −> Alamin GSM SMS Gateway SMPP output interface. `gsgsmppout` is the component that sends messages to a SMSC with a SMPP interface. Connect parameters are configured using its config file.
- gsgsmtpd −> GSM SMS Gateway SMTP interface. Alamin GSM SMS Gateway SMTP interface − DO NOT USE IT.

Alamin's diagram of operation:

```
           +----------+
MySQL ----> | gsgdb2sms|
INPUT       +----------+
                 |                                       alamin queue
                 \                                     /var/spool/alamin/
            +----------+           +----------------+      +------+
manual      |   gsgc   |  TCP/IP   |     gsgcmd     | ===> |  q1  |
INPUT   --> | (client) | --------> | (receiving and | ===> |  q2  |
(console)   +----------+           |   queueing)    | ===> |  q3  |
                                   +----------------+      |  q4  |
                                                           |  q5  |
                                                           |  q6  |
GSM Modem       RS232 or USB      +-----------------+      |  q7  |
IN/OUT    <---------------------> |     gsgmdd      | <=== |  q8  |
        TCP/IP +-----------+      |(SMS dispatching)| <=== |  q9  |
SMSC <======== | gsgsmppout| <--- |                 | <=== |      |
OUT            +-----------+      +-----------------+      +------+
                                           |
            one of input message scripts   |
APPLICAT.   +--------------------------+   |
OUT  <===== | /usr/lib/alamin/gsgimpxx |---+
            +--------------------------+
```

Default location of key files and directories:

- gsgc /usr/bin
- gsgcmd, gsgmdd, gsgdb2sms, gsgsmppin, gsgsmppout −> /usr/sbin
- libraries for INPUT processing /usr/lib/alamin

- config files /etc/alamin
- doc /usr/share/doc/alamin−doc
- pid_file /var/run/alamin/gsgcmd.pid
- accounting file /var/log/alamin/gsgd−accounting.log
- spool directory /var/spool/alamin
- alias file (for gsgcmd and gsgstmpd) /etc/alamin/alias.conf
- users file (for gsgcmd and gsgstmpd) /etc/alamin/user.conf
- log rotation /etc/logrotate.d/alamin−server
- startup script /etc/init.d/alamin−server

ALAMIN INSTALLATION

`Makefile` file included in `alamin−0.3.7.tar.gz` is buggy. For some reason, certain file names and directory names are incorrect, so I had to manually edit the `makefile`. When it was corrected, it was enough to type (while being in `alamin−0.3.7` directory):

```
> make install
```

to have `alamin` binaries, config files and documentation installed all over the system. See above for location of key files.

NOTE 1 For some reason `makefile` does NOT CREATE spool directory tree! If you do not correct MANUALLY this omision, then your `alamin` WILL NOT START. Directory tree for `alamin` spool shall be as follows:

```
/var/spool/alamin/
          q1/
          q2/
          ...
          q9/
```

Subdirectories q1−q9 in directory `/var/spool/alamin/` are needed for queue storage (9 queues in total). So make them yourself:

```
mkdir /var/spool/alamin
mkdir /var/spool/alamin/q1
mkdir /var/spool/alamin/q2
    ...
mkdir /var/spool/alamin/q9
```

NOTE 2 For some reason `makefile` does NOT CREATE directories for `pid_file`, accounting and log files. Make them manually:

```
mkdir /var/run/alamin
mkdir /var/log/alamin
```

PERL MODULES INSTALLATION (DEPENDENCIES)

`Alamin` requires some other Perl modules to be installed before it can run properly. I found this task a bit challenging, because not all of them were found on installation CD−ROMs or DVDs for my Linux distributions (Mandriva 10.1 and 10.2). Some modules for Mandrake/Mandriva were available from Internet repositories, but they were compiled for higher Perl revision than I currently have (Perl 5.8.6 on Rain, 5.8.5 on

Cyclon). To overcome this, I've decided to install Perl modules prepared for distros other than Mandriva.

Required Perl modules:

- `Proc::Daemon`

The only Mandriva/Mandrake package I could find in internet archives was perl−Proc−Daemon−0.03−1mdv2007.0.i586.rpm. However, this package is for Perl 5.8.8, so it generates error when trying to install:

```
error: Failed dependencies:
    perl−base >= 2:5.8.8 is needed by perl−Proc−Daemon−0.03−1mdv2007.0
```

So I had to take RPM from different distro. Installation of Fedora 3 package perl−Proc−Daemon−0.03−1.1.fc3.rf.noarch.rpm went pretty succesfully. This package is for Perl 5.8.5 so it shall work on my other server (Cyclon) as well.

- `Net::SMPP`

This package is required for direct interfacing of Alamin to operator's SMS Center. I found 2 RPM packages:

- perl−Net−SMPP−1.03−8.noarch.rpm
- perl−Net−SMPP−1.03−8.noarch.rpm

but none of them wanted to install itself flawlessly. Because I am NOT going to interface with SMSC, I've decided to IGNORE `Net::SMPP` module.

CHAT INSTALLATION

One of binaries that *Alamin* depends on is `chat`. It took me considerable amount of time to locate RPM package containing `chat` program. It turned out that `chat` is the default Mandriva application for PC−to−modem conversation (`vmdial` is default for Red Hat and derivative distributions) and it is included in `ppp` package. So package `ppp−2.4.3−4mdk` has been installed. Binary file has been placed in `/usr/sbin/chat`.

SYSLOG CONFIGURATION.

`~alamin/doc/LOGGING` file has some advices on customising syslog. I've coppied these sugestions to `/etc/syslog.conf` and restarted the syslog.

SMS SERVICE START (at boot time).

The `alamin` installation package includes sample `init.d` startup script. The script starts the deamons in the following order:

```
/usr/sbin/gsgcmd
/usr/sbin/gsgmdd
/usr/sbin/gsgsmppin
/usr/sbin/gsgsmppout
/usr/sbin/gsgdb2sms
```

It is user's choice to either use the default startup script without changes or write his own tailored script.

NOTE: the default `alamin` config relies on `/etc/services` for port assignment. Make sure that you either explicitely tell `alamin` (via config file or from command line) to use particular port OR update your `/etc/services` to include the following line:

```
smsqp   11201/tcp   # SMS queue protocol
```

USER RIGHTS.

`Alamin` offers flexible way of defining user rights to use available services (like sending and receiving SMS messages). User authentication is performed by two modules: `gsgcmd` and `gsgstmpd`. Users are authenticated from the client (`gsgc`) using a schema similar to CHAP. Passwords are not send in clear text over the network. Two scenarios (modes) are possible:

- user is ***anonymous*** (it means she/he didn't provide user name and password when invoking `gsgc` client). For such users `alamin` (or more precisely `gsgcmd`) grants ***DEFAULT*** user rights. DEFAULT rights are defined in `/etc/alamin/gsgd.conf` with key `defallowserv <default_allowed_services_list>`.
- user is ***recognized*** (it means her/his creditentials − user name and password − match one of users listed in `/etc/alamin/user.conf`). For such users `alamin` grants ***INDIVIDUAL*** user rights defined (per user) in `/etc/alamin/user.conf`. Example: `user:userpass::send_all:all`.

Assuming you want to secure your `alamin`, you have to:

- set `defallowserv` to `none` to disable anonymous access, and
- define all `alamin` users and their privileges in `/etc/alamin/user.conf`.

USAGE OF CLIENT APPLICATION.

Client application (`gsgc`) usage is pretty straightforward.

Anonymous mode:

```
> gsgc −−send +48601xxxyyy "message"
Message sent.
```

Authorized mode:

```
> gsgc −−user USERNAME −−password USERPASSWD −−send +48601xxxyyy "message"
Message sent.
```

NOTE Confirmation `Message sent` means that `alamin` (or more precisely `gsgcmd`) accepted message and placed it into spool. It ***DOES NOT*** mean that message was already sent via SMS! SMS sending is performed by `gsgmdd` and the client (`gsgc`) receives ***NO FEEDBACK*** on (un)succesfuly sent SMS messages.

NEW SMS SENDING MODE FOR Motorola C550: "STORAGE−AND−SEND"

GSM handsets and modems shall comply to ETSI−defined extensions for GSM phones, known also as HAYES AT+C commands. Complete list of AT+C commands is available form ETSI or other sources:

- Extended ETSI Hayes AT command parameters for SMS ;
- GSM modem / module − first steps .

That's all about theory. In reality, compatibility of particular GSM handsets or modems is far from perfect. There are even web pages that show compatibility matrix for particular handsets.

And here comes my GSM modem. I decided to use Motorola C550 − nice, cheap and compact GSM handset with USB modem. All available information on Internet sugested that C550 is AT+C compatible. There are even ppp/chat scripts to use C550 as GPRS modem under Linux. So far so good!

But ... Motorola C550 ***does not support*** `direct SMS` sending (`at+cmgs` command):

```
Aug 15 08:16:09 rain chat[12065]: send (AT+CMGS="+48601xxxyyy"^M)
Aug 15 08:16:09 rain chat[12065]: expect (>)
Aug 15 08:16:09 rain chat[12065]: AT+CMGS="+48601xxxyyy"
Aug 15 08:16:09 rain chat[12065]: ERROR
Aug 15 08:16:29 rain chat[12065]: alarm
Aug 15 08:16:29 rain chat[12065]: Failed
Aug 15 08:16:29 rain gsgmdd[12061]: ERROR sending to +48601xxxyyy using AT
```

or from another debuging (different tools):

```
at+cmgs="+48601xxxyyy"
+CME ERROR: operation not supported
at+cmsg=+48601xxxyyy
+CME ERROR: unknown
at+cmgs=+48601xxxyyy,145
+CME ERROR: unknown
```

From the above it is clear that Motorola C550 generates *ERROR* in response to valid `at+cmgs` command.

However, C550 **SUPPORTS `store−and−send` mode (SAS), in which message needs to be stored in Motorola memory first, and then be sent from there. Here you have some debuging logs from real C550:

a.) Write message to memory:

```
at+cmgw="+48601xxxyyy"
> alamakoty3        <− there was [Ctrl]−Z at the end
+CMGW: 2454     −> phone replies with memory_index
OK
```

b.) Send SMS message from storage:

```
at+cmss=[memory_index]  <− the memory_index given by `+cmgw` command
```

c.) Delete message:

```
at+cmgd=[memory_index]
```

NOTE: by default, Motorola C550 writes messages to "STO_SENT" (stored sent) folder. Syntax of `+cmgw` also allows write operations to other folders: "REC UNREAD" "REC READ" "STO UNSENT" "STO SENT". I wasn't able to get this working on Motorola C550:

```
at+cmgw="+48601xxxyyy","REC UNREAD"
ERROR
at+cmgw="+48601xxxyyy", "REC READ"
ERROR
at+cmgw="+48601xxxyyy",145,"REC READ"
ERROR
```

THEORY − SMPP PROTOCOL

`Alamin` has the capability to talk directly (it means without GSM modem) to operator's SMSC server via Short Message Peer to Peer Protocol. Here is some more description of SMPP taken from Perl's Net−SMPP module README:

Despite its name, SMPP protocol defines a client (ESME) and a server (often called SMSC in the mobile operator world). Client usually initiates the TCP connection and does bind to log in. After binding, a series of request response pairs, called PDUs (protocol data units) is exchanged. Request can be initiated by either end (hence "peer−to−peer"?) and the other end reponds. Requests are numbered with a sequence number and each response has corresponding sequence number. This allows several requests to be pending at the same time. Conceptually this is similar to IMAP or LDAP message IDs. Usually the $smpp object maintains the sequence numbers by itself and the programmer need not concern himself with their exact values, but should a need to override them arise, the seq argument can be supplied to any request or response method.

Motorola C550 − TECHNICAL DETAILS

- embeded digital camera (640x480pix, 2x digital zoom)
- MMS
- GPRS
- java
- games
- polyfonic ring tones (32 voices)
- MotoMixer functionality
- WAP 2.0
- Display: 4096 colours, 96x65pix
- Stanby time (MAX): 230 hours
- Battery: Li−Ion 750 mAh

It looks that C550 is a follower of older C330 and C350. Why? Because command `lsusb` shows in return `C330` (see below), and some web sources write about C550 and C350 similarities.

HOW LINUX KERNEL SEES Moto C550

Plain Mandriva 10.2 kernel:

```
> dmesg
[...]
usb 1−2.1.4: new full speed USB device using uhci_hcd and address 6
```

```
> lsusb
Bus 001 Device 006: ID 22b8:3802 Motorola PCS C330 GSM Phone
Bus 001 Device 004: ID 04b4:6560 Cypress Semiconductor Corp. CY7C65640\
     USB−2.0 "TetraHub"
Bus 001 Device 003: ID 058f:9254 Alcor Micro Corp. Hub
Bus 001 Device 002: ID 046d:c03f Logitech, Inc.
Bus 001 Device 001: ID 0000:0000
```

Loading `cdc_acm` module (USB Modem) fixes the problem. Now new device `ttyACM0` is available.

```
> modprobe cdc_acm

> dmesg
[...]
usb 1−2.1.4: new full speed USB device using uhci_hcd and address 6
cdc_acm 1−2.1.4:1.0: ttyACM0: USB ACM device
usbcore: registered new driver cdc_acm
drivers/usb/class/cdc−acm.c: v0.23:USB Abstract Control Model driver \
    for USB modems and ISDN adapters

> ls −l /dev/ttyACM*
crw−rw−−−−  1 root root 166, 0 Aug  5 22:04 /dev/ttyACM0
```

DEBUGGING − *HAYES AT+C* COMMANDS

The so−called `Hayes AT+C` commands are defined by ETSI extensions for GSM phones:

- Extended ETSI Hayes AT command parameters for SMS
- GSM modem / module − first steps
- Kody AT dla roznych producentow
- Phone / modem compatibility list

However, Motorola handsets support also another set of (Motorola proprietary) commands called `AT+M`:

- Motorola AT+M extended commands Extended mode MP on (mode=2):

  at+mode=0 OK at+mode=2 OK +MBAN: Copyright 2000−2002 Motorola, Inc.

Misc. links:

- Interesting article on SMS, text mode, PDU, SMSC, etc.
- ASCII and HEX codes
- How MMS works

DEBUGGING − MOTOROLA C550 SUPPORTED COMMANDS:

The commands below were manually checked by me. I used `minicom` to test the AT commands and watch phone answers.

Available phonebooks:

```
at+cpbs?    − shows currently selected
+CPBS: "AD"
```

```
OK
at+cpbs=?   – shows all available phone books
+CPBS: ("ME","SM","MT","ON","DC","MC","RC","AD","QD","FD")
OK
```

Interpretation of phone answer (http://www.csparks.com/MotoBackup/MotorolaAT.xhtml)

```
AD: The main combined phonebook (phone, email, all in one)
RC: Received calls list
LD: Last–dialed phonebook
QD: Quick dial numbers
ME: Not sure. Seems to select the main phone book.
MT: Not sure. Seems to select the main phone book.
ON: Not sure. Contains the phone's own number and one other entry.
DC: Dialed calls. Same as LD on the V60
MC: Last dialed number
RC: Received calls
```

Describe the current phonebook:

```
at+mpbr=?
+MPBR: 1–200,40,18,8,0–1,50,(0–73,76,255),(0),(0–1),(1–30),(255),25,(255),263,(0)
OK
```

Interpretation of phone answer (http://www.csparks.com/MotoBackup/MotorolaAT.xhtml)

```
1–200   Number of possible entries in the phone book.
40  Maximum length of phone number
18  Maximum length of the name
8   Number of entry types
0–1 Index range for recorded voice patterns
50  Maximum size of an email address
()  Parametry w nawiasach – przeznaczenie nieznane
```

List phonebook entries

- Show a selected entry: `at+mpbr=indexValue`
- Show a range of entries: `at+mpbr=firstIndex,lastIndex`

  at+mpbr=1 – show entry no 1 +MPBR: 1,"605590694",129,"Marta Z",0,0,255,0,1,1,255,255,0,"",0,0
  OK

Stored phonebook number types:

```
128 Email address or mailing list
    Phone "number" contains an "@" character.
129 National number
145 International
    Phone number has a leading "+" character.
```

Stored phonebook entry types:

```
0 Work
1 Home
2 Main
3 Mobile
```

```
4 Fax
5 Pager
6 Email
7 Mailing list
```

Find an entry by name:

```
at+mpbf="Search string"
```

Write, modify or delete an entry:

- General format: `at+mpbw=index,"number or email",etype,"name",type`
- Write or modify a phone number entry: `at+mpbw=10,"4015524025",129,"Joseph Curwin",1`
- Write or modify an email entry: `at+mpbw=9,"euler@bbaw.de",129,"Leonhard Euler",6`
- Erase entry number 9: `at+mpbw=9`

Show the serial number:

```
at+cgsn
+CGSN: IMEI352718xxxxyyyy
```

Show the revision number:

```
at+cgmr
+CGMR: "R321_G_0A.10.27R"
```

Show the model description:

```
at+cgmm
+CGMM: "GSM900","GSM1800","GSM1900","GSM850","MODEL=C550"
```

Show available character sets:

```
at+cscs=?
+CSCS: ("8859−1","ASCII","GSM","UCS2","UTF8")
OK
at+cscs?
+CSCS: "ASCII"
OK
```

SMS Message format:

```
at+cmgf?
+CMGF: 1
```

Answer:

- 0 − PDU
- 1 − text

NOTE: Motorola C550 accepts SMS−es in text format ONLY:

```
at+cmgf=?
+CMGF: (1)
OK
```

Service Center Address:

```
at+csca?
+CSCA: "+48601000310",145
```

SCA number format:

- 128: unknown
- 129: national
- 145: international
- 161: national

SMS message read:

```
at+cmgr=1
+CMGR: "REC UNREAD", "+48601xxxyyy", "2006/8/8,8:42:48"
Ala ma kota
OK

at+cmgr=1
+CMGR: "REC READ", "+48601xxxyyy", "2006/8/8,8:42:48"
Ala ma kota
OK
```

Write SMS message to memory:

```
at+cmgw="+48601xxxyyy"
> alamakoty3        <− there was [Ctrl]−Z at the end
+CMGW: 2454     −> phone replies with memory_index
OK
```

NOTE: by default, Motorola C550 writes messages to "STO_SENT" (stored sent) folder. Syntax of `+cmgw` allows write operations to other folders: "REC UNREAD" "REC READ" "STO UNSENT" "STO SENT", but I wasn't able to get this working:

```
at+cmgw="+48601xxxyyy","REC UNREAD"
ERROR
at+cmgw="+48601xxxyyy", "REC READ"
ERROR
at+cmgw="+48601xxxyyy",145,"REC READ"
ERROR
```

Delete message:

```
at+cmgd=[memory_index]
```

Send SMS:

```
at+cmgs=
```

C550 generates errors in response to `+cmgs` command:

```
at+cmgs="+48601370317"
+CME ERROR: operation not supported
at+cmsg=+48601370317
+CME ERROR: unknown
at+cmgs=+48601370317,145
+CME ERROR: unknown
```

Send SMS message from storage:

```
at+cmss=[memory_index]  <- the memory_index given by `+cmgw` command
```

Preferred Message Storage:

Selects memory storage spaces to be used for reading.

```
at+cpms?
+CPMS: MT,0,352,OM,0,88,IM,0,88
```

Phone answer values:

- "ME": ME message store − PHONE memory
- "SM": SIM message store
- "MT": any of the storages associated with ME
- "BM": CBM message store (in volatile memory)
- "TA": TA message storage
- "SR": Status report storage
- "PT": ???
- "CT": ???
- "TL": template message store
- "IM": ???
- "OM": ???
- "DM": ???

  at+cpms=? +CPMS: (IM,OM,BM,MT,DM),(OM,DM),(IM)

Field strenght:

```
at+csq
+CSQ: 27,99
```

NOTE: `at+csq` works ***ONLY*** in `mode=0`:

```
at+mode=0
OK
at+csq
+CSQ: 27,99 <- CORRECT ANSWER
OK
```

```
at+mode=2
OK
+MBAN: Copyright 2000−2002 Motorola, Inc.
at+csq
ERROR       <- ERROR
```

Battery charge:

```
at+cbc?
+CBC: 0,60
```

Extended mode MP on (mode=2):

```
at+mode=0
OK
at+mode=2
OK
+MBAN: Copyright 2000−2002 Motorola, Inc.
```

NOTE: *Extended mode* is a special mode for MOTOROLA GSM phones, where `at+m` extended commands are available.

DEBUGGING − COMMANDS NOT SUPPORTED BY Mot C550:

- at+cpin − entry/check PIN
- at+creg − Register Network 2=log off, 1=log in, 0=don´t know
- at+cmgf=0 − SMS in format PDU (C550 supports only TEXT: at+cmgf=1)
- at+cops? − sprawdza, czy siec zwiazana z SIM−em jest dostepna

SERVICE MENU FOR C350 (so possibly also for C550)

Source: http://www.diamondelectric.ru/secart55.html

Press fast: Menu048263*

Display: Opcode:__

Then enter one of the following codes. Example: 54*18*1 resets phone. 54* is the obligatory PREFIX for ALL COMMANDS.

- 0*0*0 Select tone 0
- 0*0*1 Select tone 1
- 0*0*2 Select tone 2
- 0*0*3 Select tone 3
- 0*0*4 Select tone 4
- 0*0*5 Select tone 5
- 0*0*6 Select tone 6
- 0*0*7 Select tone 7
- 0*0*8 Select tone 8
- 0*0*9 Select tone 9
- 0*0*124 Select tone 1 KHz

- 0*0*125 Select tone 2 KHz
- 0*0*126 Select tone 3 KHz
- 0*0*127 Select tone 4 KHz
- 0*1*X Disable tone X
- 3*0*1 Enable vibrator
- 3*0*0 Disable vibrator
- 4*3*1 Enable speech coder full rate Audio loopback
- 4*3*0 Disable speech coder full rate
- 4*4*1 Enable speech coder enhanced full rate
- 4*4*0 Disable speech coder enhanced full rate
- 4*5*1 Enable speech coder half rate
- 4*5*0 Disable speech coder half rate
- 5*0*0 Set audio level 0 Audio level
- 5*0*1 Set audio level 1
- 5*0*2 Set audio level 2
- 5*0*3 Set audio level 3
- 5*0*4 Set audio level 4
- 5*0*5 Set audio level 5
- 5*0*6 Set audio level 6
- 5*0*7 Set audio level 7
- 5*0*8 Set audio level 8
- 5*0*9 Set audio level 9
- 5*0*10 Set audio level 10
- 5*0*11 Set audio level 11
- 5*0*12 Set audio level 12
- 5*0*13 Set audio level 13
- 5*0*14 Set audio level 14
- 5*0*15 Set audio level 15
- 6*220*0 Set Audio Path. Int Mic, IntSpk, RX unmute, TX unmute
- 6*460*0 Set Audio Path. Boom Mic, Boom Spk, RX unmute, TX unmute
- 10*03* Set band GSM 900
- 10*04* Set band DCS 1800
- 10*05* Set band PCS 1900
- 10*06* Set dual band GSM 900 / 1800
- 10*1*0 Read band 3= GSM 4= DCS 5= PCS 6 =GSM/DCS
- 18*0 Initialize non−volatile memory (Master Reset)
- 18*1 Initialize Non−volatile memory (Master Clear)
- 20*X*0 Load Channel number X Select Channel (Used for debugging Rx mode)
- 20*1*0 Load channel number 1 GSM Low channel
- 20*62*0 Load channel number 62 GSM Mid channel
- 20*124*0 Load channel number 124 GSM High channel
- 20*512*0 Load channel number 512 DCS Low channel
- 20*700*0 Load channel number 700 DCS Mid channel
- 20*885*0 Load channel number 885 CDS High channel
- 20*512*0 Load channel number 512 PCS Low channel
- 20*661*0 Load channel number 661 PCS Mid channel
- 20*810*0 Load channel number 810 PCS High channel
- 55*2*001 Test Display. All pixels ON
- 55*2*000 Test Display. All pixels OFF
- 55*2*002 Test Display. Checkerboard pattern A
- 55*2*003 Test Display. Checkerboard pattern B

- 552004 Test Display. Border pixels ON
- *#06# IMEI Check No Test Mode Required

NETmonitor:

- menu00*83786633* − ON
- menu00*8378633* − OFF

ADDITIONAL APPLICATIONS FOR C550

- KMobileTools dla C350 and Motorola C650. 2006/08/06 − instalacja kmobiletools−0.4.3.1−2mdk (578 KB) z repozytorium "sunsite.mandrake.10.2.contrib".

UNSUCCESFULL ATTEMPTS

Before I found `alamin`, I was testing some other applications. You will find my feedback on them in this section.

'SMSlink' INSTALLATION (2006/08/08)

SMSLink and SMS Link Add On'S are two web sites with more details on `SMSLink`. Despite heavy googling, I couldn't find RPM packages for Mandriva. I decided to use RedHat *SMSlink* package. I downloaded `smslink` and required (dependency) `libmodem`.

```
libmodem−1.5−1.i386.rpm (RedHat Other)
    /etc/modems
    /usr/lib/libmodem.so

smslink−0.52b2−1.i386.rpm
    /usr/sbin/sms_serv
    /usr/sbin/sms2mailgw
```

The installation was trivial:

```
> rpm −i libmodem−1.5−1.i386.rpm
> rpm −i smslink−0.52b2−1.i386.rpm
```

CONFIGURATION

Hmm, it looks that RPM package is missing the config files:

```
/etc/gsmdevices
/etc/gsmcaps
```

but luckily some config files are available from:
http://cvs.uwc.ac.za/viewcvs/viewcvs.cgi/smsdiary/server/smslink−0.56b/?cvsroot=AVOIR#dirlist

'SMSlink' REMOVAL (2006/08/14)

I removed `SMSlink` i `libmodem`.

OTHER APPLICATIONS

- SMS Server Tools
- GSM::SMS − Perl Modules For Smart Messaging
- mail2sms

mail2sms converts a single (large) mail to a tiny text with contents from the mail. Perfectly suitable to send as an SMS message to a GSM telephone.

GPRS CONNECTION (pppd)

EasyConnect GPRS.

In the present mini how−to I'm reporting how I connected my laptop to internet using a Motorola c350 mobile phone (gprs).

Make sure the package 'ppp' is installed on your machine; then,

---

Motorola c350

Create the file '/etc/chatscripts/phone' and, if your server is Vodafone, fill it with:

```
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT DELAYED
'' ATZ
OK−AT−OK AT+CGDCONT=1,"IP","web.omnitel.it"
OK−AT−OK AT+CGQREQ=1,2,4,3,6,31
OK−AT−OK AT+CGQMIN=1,2,4,3,6,31
OK−AT−OK AT+CGATT=1 OK−AT−OK ATD*99#
CONNECT ''
```

; instead, if your server is TIM, replace "web.omnitel.it" with "ibox.tim.it".

Create the file '/etc/ppp/resolv/phone' and fill it with:

```
nameserver 194.185.97.134
```

Create the file '/etc/ppp/peers/phone' and fill it with:

```
hide−password
noauth
connect "/usr/sbin/chat −v −f /etc/chatscripts/phone"
debug
−crtscts
/dev/ttyACM0
57600
defaultroute
noipdefault
```

```
remotename phone
ipparam phone
usepeerdns
```

At the end of each file leave a blank line.

At the end of '/etc/syslog.conf' add the following lines:

```
local2.*                                /var/log/ppp
daemon.*                                /var/log/ppp
```

Inny plik konfiguracyjny dla `ppp` ze strony fabiux.

I've created a file named /etc/ppp/peers/gprs with the following content:

```
115200
crtscts
modem
lock
defaultroute
noipdefault
hide-password
noauth
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs"
debug
/dev/ttyACM0
user "your-username"
password "your-password"
remotename gprs
ipparam gprs
lcp-echo-interval 90000
usepeerdns
receive-all
nopcomp
noaccomp
nomagic
noccp
novj
novjccomp
nodetach
nobsdcomp
nodeflate
usepeerdns
logfile "/var/log/pppd.log"
```

I also created a file named /etc/chatscripts/gprs with the following content:

```
ABORT BUSY ABORT 'NO CARRIER' ABORT VOICE ABORT 'NO DIALTONE' ABORT 'NO DIAL TONE' ABORT 'NO AN
'' 'ATZ;AT+CGDCONT=1,"IP","your-provider-domain","";'
OK-AT-OK "ATDT*99***1#"
CONNECT \d\c
```

MODEM IN NOKIA HANDSETS

Web pages on PySMS project inform about some bugs in Nokia handsets working in modem mode:

- The first manifests itself as corruption of the currently transmitting character when a CR is Rxd. This causes bad characters to be transmitted and may result in framing errors, possibly corrupting subsequent characters too. This is presumably because the Tx routines are software driven, but the bit−rate gets disrupted because the latency of the CR processing is too high. This bug can be largely avoided by turning off echo, so that Tx is not occurring during Rx on the phone.
- The second bug is more difficult to avoid. There seems to be a bug in the Tx buffer, which appears to wrap around when sending large amounts of data. For example, if you list > 40 messages, the data stream will have holes in it, resulting in lost data and a breaking of the comms protocol. If messages are processed and deleted as they come in, the number of messages on the phone will never be large enough to cause this problem. But there is no good fix for this bug.

Similar info is found on Gnokii web page:

- gnokii supports most Nokia phones from the 3110/3180, 5110/6110, 7110/6210 and 6510/6310 series, details including bugs specific to each series appear in the files README−3810, README−6110, README−7110 and README−6510 respectively. Also all AT compatible phones are supported.