GigaSet 307x Device Driver

1. Requirements

1.1. Hardware

This driver supports the connection of the Gigaset 307x/417x family of ISDN DECT bases via Gigaset M101 Data, Gigaset M105 Data or direct USB connection. The following devices are reported to be compatible:

Bases:
- Siemens Gigaset 3070/3075 isdn
- Siemens Gigaset 4170/4175 isdn
- Siemens Gigaset SX205/255
- Siemens Gigaset SX353
- T-Com Sinus 45 [AB] isdn
- T-Com Sinus 721X[A] [SE]
- Vox Chicago 390 ISDN (KPN Telecom)

RS232 data boxes:
- Siemens Gigaset M101 Data
- T-Com Sinus 45 Data 1

USB data boxes:
- Siemens Gigaset M105 Data
- Siemens Gigaset USB Adapter DECT
- T-Com Sinus 45 Data 2
- T-Com Sinus 721 data
- Chicago 390 USB (KPN)

See also http://www.erbze.info/sinus_gigaset.htm and
http://gigaset307x.sourceforge.net/

We had also reports from users of Gigaset M105 who could use the drivers with SX 100 and CX 100 ISDN bases (only in unimodem mode, see section 2.5.) If you have another device that works with our driver, please let us know.

Chances of getting an USB device to work are good if the output of
- lsusb

at the command line contains one of the following:
- ID 0681:0001
- ID 0681:0002
- ID 0681:0009
- ID 0681:0021
- ID 0681:0022

1.2. Software

The driver works with ISDN4linux and so can be used with any software which is able to use ISDN4linux for ISDN connections (voice or data). Experimental Kernel CAPI support is available as a compilation option.

There are some user space tools available at
http://sourceforge.net/projects/gigaset307x/
which provide access to additional device specific functions like SMS,

phonebook or call journal.

2. How to use the driver

2.1. Modules

For the devices to work, the proper kernel modules have to be loaded. This normally happens automatically when the system detects the USB device (base, M105) or when the line discipline is attached (M101). It can also be triggered manually using the modprobe(8) command, for example for troubleshooting or to pass module parameters.

The module ser_gigaset provides a serial line discipline N_GIGASET_M101 which uses the regular serial port driver to access the device, and must therefore be attached to the serial device to which the M101 is connected. The ldattach(8) command (included in util-linux-ng release 2.14 or later) can be used for that purpose, for example:

```
ldattach GIGASET_M101 /dev/ttyS1
```

This will open the device file, attach the line discipline to it, and then sleep in the background, keeping the device open so that the line discipline remains active. To deactivate it, kill the daemon, for example with

```
killall ldattach
```

before disconnecting the device. To have this happen automatically at system startup/shutdown on an LSB compatible system, create and activate an appropriate LSB startup script /etc/init.d/gigaset. (The init name 'gigaset' is officially assigned to this project by LANANA.) Alternatively, just add the 'ldattach' command line to /etc/rc.local.

The modules accept the following parameters:

Module	Parameter	Meaning
gigaset	debug	debug level (see section 3.2.)
bas_gigaset) ser_gigaset) usb_gigaset)	startmode	initial operation mode (see section 2.5.): 1=ISDN4linux/CAPI (default), 0=Unimodem
	cidmode	initial Call-ID mode setting (see section 2.5.): 1=on (default), 0=off

Depending on your distribution you may want to create a separate module configuration file /etc/modprobe.d/gigaset for these, or add them to a custom file like /etc/modprobe.conf.local.

2.2. Device nodes for user space programs

The device can be accessed from user space (eg. by the user space tools mentioned in 1.2.) through the device nodes:

- /dev/ttyGS0 for M101 (RS232 data boxes)
- /dev/ttyGU0 for M105 (USB data boxes)
- /dev/ttyGB0 for the base driver (direct USB connection)

If you connect more than one device of a type, they will get consecutive

device nodes, eg. /dev/ttyGU1 for a second M105.

You can also set a "default device" for the user space tools to use when no device node is given as parameter, by creating a symlink /dev/ttyG to one of them, eg.:

```
ln -s /dev/ttyGB0 /dev/ttyG
```

The devices accept the following device specific ioctl calls (defined in gigaset_dev.h):

ioctl(int fd, GIGASET_REDIR, int *cmd);
If cmd==1, the device is set to be controlled exclusively through the character device node; access from the ISDN subsystem is blocked.
If cmd==0, the device is set to be used from the ISDN subsystem and does not communicate through the character device node.

ioctl(int fd, GIGASET_CONFIG, int *cmd);
(ser_gigaset and usb_gigaset only)
If cmd==1, the device is set to adapter configuration mode where commands are interpreted by the M10x DECT adapter itself instead of being forwarded to the base station. In this mode, the device accepts the commands described in Siemens document "AT-Kommando Alignment M10x Data" for setting the operation mode, associating with a base station and querying parameters like field strengh and signal quality.
Note that there is no ioctl command for leaving adapter configuration mode and returning to regular operation. In order to leave adapter configuration mode, write the command ATO to the device.

ioctl(int fd, GIGASET_BRKCHARS, unsigned char brkchars[6]);
(usb_gigaset only)
Set the break characters on an M105's internal serial adapter to the six bytes stored in brkchars[]. Unused bytes should be set to zero.

ioctl(int fd, GIGASET_VERSION, unsigned version[4]);
Retrieve version information from the driver. version[0] must be set to one of:
- GIGVER_DRIVER: retrieve driver version
- GIGVER_COMPAT: retrieve interface compatibility version
- GIGVER_FWBASE: retrieve the firmware version of the base
Upon return, version[] is filled with the requested version information.

2.3. ISDN4linux

This is the "normal" mode of operation. After loading the module you can set up the ISDN system just as you'd do with any ISDN card supported by the ISDN4Linux subsystem. Most distributions provide some configuration utility. If not, you can use some HOWTOs like
    http://www.linuxhaven.de/dlhp/HOWTO/DE-ISDN-HOWTO-5.html
If this doesn't work, because you have some device like SX100 where debug output (see section 3.2.) shows something like this when dialing
```
    CMD Received: ERROR
    Available Params: 0
    Connection State: 0, Response: -1
    gigaset_process_response: resp_code -1 in ConState 0 !
    Timeout occurred
```

you probably need to use unimodem mode. (see section 2.5.)

2.4. CAPI

If the driver is compiled with CAPI support (kernel configuration option GIGASET_CAPI, experimental) it can also be used with CAPI 2.0 kernel and user space applications. For user space access, the module capi.ko must be loaded. The capiinit command (included in the capi4k-utils package) does this for you.

The CAPI variant of the driver supports legacy ISDN4Linux applications via the capidrv compatibility driver. The kernel module capidrv.ko must be loaded explicitly with the command

```
   modprobe capidrv
```

if needed, and cannot be unloaded again without unloading the driver first. (These are limitations of capidrv.)

The note about unimodem mode in the preceding section applies here, too.

2.5. Unimodem mode

This is needed for some devices [e.g. SX100] as they have problems with the "normal" commands.

If you have installed the command line tool gigacontr, you can enter unimodem mode using

```
    gigacontr --mode unimodem
```

You can switch back using

```
    gigacontr --mode isdn
```

You can also put the driver directly into Unimodem mode when it's loaded, by passing the module parameter startmode=0 to the hardware specific module, e.g.

```
   modprobe usb_gigaset startmode=0
```

or by adding a line like

```
   options usb_gigaset startmode=0
```

to an appropriate module configuration file, like /etc/modprobe.d/gigaset or /etc/modprobe.conf.local.

In this mode the device works like a modem connected to a serial port (the /dev/ttyGU0, ... mentioned above) which understands the commands

```
    ATZ                 init, reset
        => OK or ERROR
    ATD
    ATDT                dial
        => OK, CONNECT,
           BUSY,
           NO DIAL TONE,
           NO CARRIER,
           NO ANSWER
    <pause>+++<pause>   change to command mode when connected
    ATH                 hangup
```

You can use some configuration tool of your distribution to configure this "modem" or configure pppd/wvdial manually. There are some example ppp configuration files and chat scripts in the gigaset-VERSION/ppp directory

in the driver packages from http://sourceforge.net/projects/gigaset307x/.
Please note that the USB drivers are not able to change the state of the control lines. This means you must use "Stupid Mode" if you are using wvdial or you should use the nocrtscts option of pppd.
You must also assure that the ppp_async module is loaded with the parameter flag_time=0. You can do this e.g. by adding a line like

```
options ppp_async flag_time=0
```

to an appropriate module configuration file, like /etc/modprobe.d/gigaset or /etc/modprobe.conf.local.

2.6. Call-ID (CID) mode

Call-IDs are numbers used to tag commands to, and responses from, the Gigaset base in order to support the simultaneous handling of multiple ISDN calls. Their use can be enabled ("CID mode") or disabled ("Unimodem mode"). Without Call-IDs (in Unimodem mode), only a very limited set of functions is available. It allows outgoing data connections only, but does not signal incoming calls or other base events.

DECT cordless data devices (M10x) permanently occupy the cordless connection to the base while Call-IDs are activated. As the Gigaset bases only support one DECT data connection at a time, this prevents other DECT cordless data devices from accessing the base.

During active operation, the driver switches to the necessary mode automatically. However, for the reasons above, the mode chosen when the device is not in use (idle) can be selected by the user.
- If you want to receive incoming calls, you can use the default settings (CID mode).
- If you have several DECT data devices (M10x) which you want to use in turn, select Unimodem mode by passing the parameter "cidmode=0" to the appropriate driver module (ser_gigaset or usb_gigaset).

If you want both of these at once, you are out of luck.

You can also use the tty class parameter "cidmode" of the device to change its CID mode while the driver is loaded, eg.

```
echo 0 > /sys/class/tty/ttyGU0/cidmode
```

2.7. Unregistered Wireless Devices (M101/M105)

The main purpose of the ser_gigaset and usb_gigaset drivers is to allow the M101 and M105 wireless devices to be used as ISDN devices for ISDN connections through a Gigaset base. Therefore they assume that the device is registered to a DECT base.

If the M101/M105 device is not registered to a base, initialization of the device fails, and a corresponding error message is logged by the driver. In that situation, a restricted set of functions is available which includes, in particular, those necessary for registering the device to a base or for switching it between Fixed Part and Portable Part modes. See the gigacontr(8) manpage for details.

3. Troubleshooting

3.1. Solutions to frequently reported problems

Problem:
You have a slow provider and isdn4linux gives up dialing too early.
Solution:
Load the isdn module using the dialtimeout option. You can do this e.g. by adding a line like

```
options isdn dialtimeout=15
```

to /etc/modprobe.d/gigaset, /etc/modprobe.conf.local or a similar file.

Problem:
The isdnlog program emits error messages or just doesn't work.
Solution:
Isdnlog supports only the HiSax driver. Do not attempt to use it with other drivers such as Gigaset.

Problem:
You have two or more DECT data adapters (M101/M105) and only the first one you turn on works.
Solution:
Select Unimodem mode for all DECT data adapters. (see section 2.5.)

Problem:
Messages like this:

```
usb_gigaset 3-2:1.0: Could not initialize the device.
```

appear in your syslog.
Solution:
Check whether your M10x wireless device is correctly registered to the Gigaset base. (see section 2.7.)

3.2. Telling the driver to provide more information

Building the driver with the "Gigaset debugging" kernel configuration option (CONFIG_GIGASET_DEBUG) gives it the ability to produce additional information useful for debugging.

You can control the amount of debugging information the driver produces by writing an appropriate value to /sys/module/gigaset/parameters/debug, e.g.

```
echo 0 > /sys/module/gigaset/parameters/debug
```

switches off debugging output completely,

```
echo 0x302020 > /sys/module/gigaset/parameters/debug
```

enables a reasonable set of debugging output messages. These values are bit patterns where every bit controls a certain type of debugging output. See the constants DEBUG_* in the source file gigaset.h for details.

The initial value can be set using the debug parameter when loading the module "gigaset", e.g. by adding a line

```
options gigaset debug=0
```

to your module configuration file, eg. /etc/modprobe.d/gigaset or /etc/modprobe.conf.local.

Generated debugging information can be found
- as output of the command

dmesg
- in system log files written by your syslog daemon, usually
  in /var/log/, e.g. /var/log/messages.

3.3. Reporting problems and bugs

If you can't solve problems with the driver on your own, feel free to use one of the forums, bug trackers, or mailing lists on
    http://sourceforge.net/projects/gigaset307x
or write an electronic mail to the maintainers.

Try to provide as much information as possible, such as
- distribution
- kernel version (uname -r)
- gcc version (gcc --version)
- hardware architecture (uname -m, ...)
- type and firmware version of your device (base and wireless module, if any)
- output of "lsusb -v" (if using an USB device)
- error messages
- relevant system log messages (it would help if you activate debug output as described in 3.2.)

For help with general configuration problems not specific to our driver, such as isdn4linux and network configuration issues, please refer to the appropriate forums and newsgroups.

3.4. Reporting problem solutions

If you solved a problem with our drivers, wrote startup scripts for your distribution, ... feel free to contact us (using one of the places mentioned in 3.3.). We'd like to add scripts, hints, documentation to the driver and/or the project web page.

4. Links, other software

- Sourceforge project developing this driver and associated tools
    http://sourceforge.net/projects/gigaset307x
- Yahoo! Group on the Siemens Gigaset family of devices
    http://de.groups.yahoo.com/group/Siemens-Gigaset
- Siemens Gigaset/T-Sinus compatibility table
    http://www.erbze.info/sinus_gigaset.htm

5. Credits

Thanks to

Karsten Keil
   for his help with isdn4linux
Deti Fliegl
   for his base driver code
Dennis Dietrich
   for his kernel 2.6 patches
Andreas Rummel

   for his work and logs to get unimodem mode working
Andreas Degert
   for his logs and patches to get cx 100 working
Dietrich Feist
   for his generous donation of one M105 and two M101 cordless adapters
Christoph Schweers
   for his generous donation of a M34 device

and all the other people who sent logs and other information.