DEVELOPER'S GUIDE
M2M TECHNOLOGY, GETTING STARTED

P&I M2M

Version1.0 • Version August 10, 2012 • Status Approved

LIFE'S FOR SHARING

OVERVIEW

Today, using mobile communications technologies to transmit data can network systems and machines cost-effectively. This opens up a number of options to integrate monitoring, control and remote maintenance services (M2M communication) into existing solutions or plan them for new products.

This guide is designed to introduce developers to this topic and make them aware of the special problems it may present.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This guide is intended for application developers and provides an application-oriented view into the world of data services and their use. It focuses on empirical experiences and assessments and does not go into a detailed description of the underlying technologies.

Developers working in the M2M area often come from two different educational backgrounds, either the IT world in which terms like IP, TCP, UDP, HTTP or SSL are fundamental, or the automation world that deals with point-to-point connections and low-level hardware access on a daily basis. For a uniform M2M solution, knowledge from both worlds is extremely helpful, and in some cases absolutely essential.

2. PREREQUISITES

To use this developer's guide, several prerequisites must be met.
You must have a GPRS terminal, a PC, documentation, and of course a SIM card.

2.1. SAMPLE HARDWARE

All the examples used in this document use a mobile communications terminal from MC Technologies GmbH, based on the EGS5 module from Cinterion. In particular, this module offers an integrated IP stack and the ability to program the engine itself in JavaTM. Figure 1 shows the most important function blocks of the current module. You can see that a module is extremely well equipped with two serial interfaces, SPI, I2C and GPIOs. Many modules still have voice communications capability.

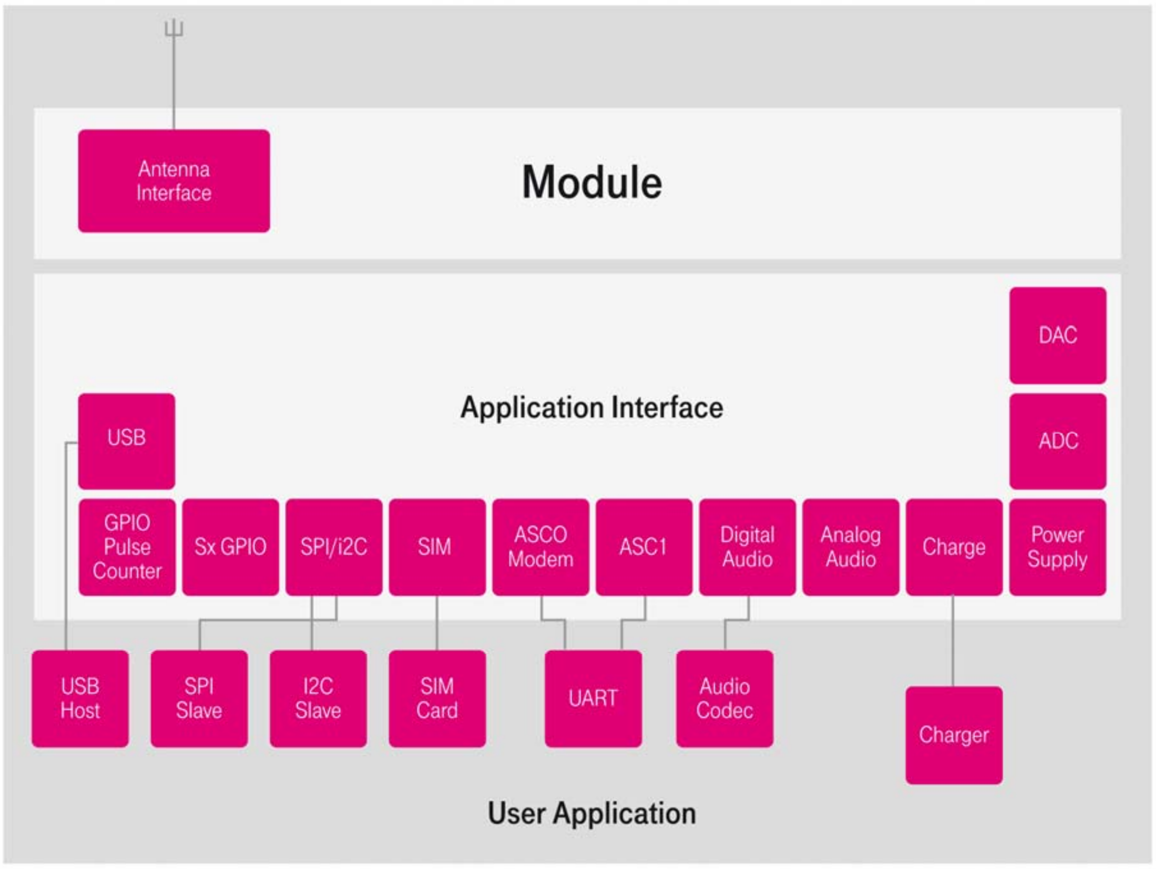


Figure 1 - Overview block diagram of GSM Module

The terminal can be obtained as a developer kit directly from Telekom:
www.developergarden.com/m2m/device-library/cinterion

Similar mobile communications terminals can be obtained from many other providers, such as Telit and Sierra Wireless.
For more information , see
www.developergarden.com/m2m/device-library/telit-wireless-solutions
and
www.developergarden.com/m2m/device-library/sierra-wireless

In general, the examples cited in this guide also function on these GPRS terminals. Please refer to the manufacturer's documentation.

The mobile communications terminal consists of the Cinterion module and a printed circuit board (PCB). Included on the PCB are:

- the power supply,
- driver components for the interfaces,
- a microcontroller as an external watchdog (in JavaTM mode) and for managing the life cycle of the modem.

This GPRS terminal has both an RS-232 and a USB interface. The RS-232 interface is fully equipped, i.e., not only with GND (ground) and the two data lines (RX and TX) but also with the signal lines and control lines CTS, RTS, DSR, DTR and RING.

Figure 2 shows a photograph with the three terminals in different colors, and for size comparison, a PCB that is installed in the terminal and a SIM card. The installed Cinterion module is the chip in the middle of the PCB above the RJ45 socket.

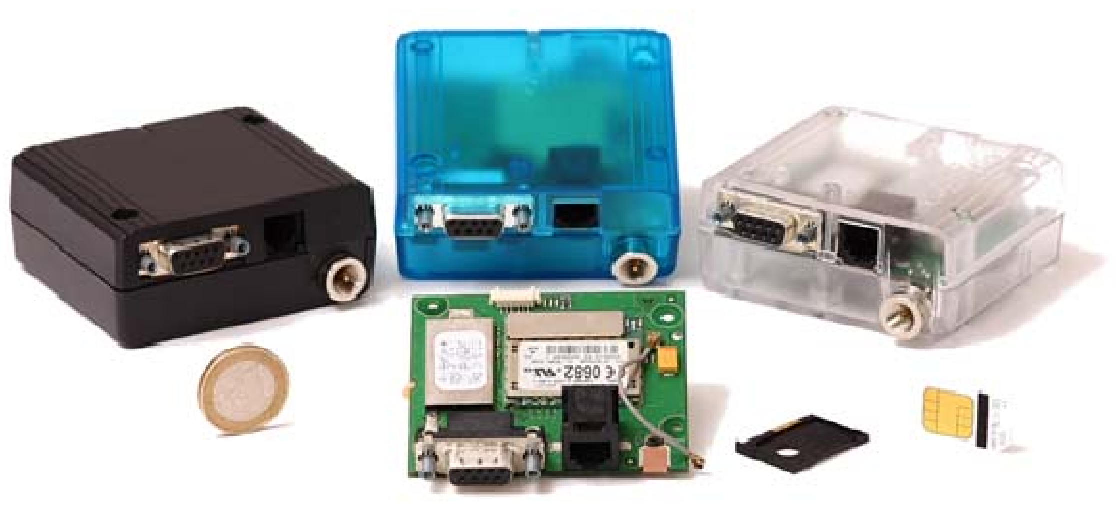


Figure 2 – Sample hardware

2.1.1. REFERENCE DESIGN

This document does not address hardware topics in depth, but there are a few important points that always need to be kept in mind when developing terminal hardware.

The Eval Kits usually provide all the documents needed for developing your own hardware. The manufacturers have usually published reference designs for the major circuits; follow the notes and instructions precisely to implement them. Generally important areas include:

- Module interfaces (e.g. SIM card interface)
  As seen in Figure 1, the modern GSM modules offer a number of interfaces that of course have different wiring requirements. Once a design is implemented and the solution out in the field, correcting any faulty circuitry is always expensive. One interface in constant use is the SIM interface. Since the SIM card is also a processor, the power supply and the data lines to the modem module must be taken into account. If they are not wired according to specification, an operating voltage shutdown could damage the SIM card over time due to the incorrect level on the data bus. This renders communication impossible.
- Power supply
  The power supply of the terminal should not be underestimated. Although the application is embedded, the power supply has to cover a wide range of tasks. Because most scenarios are low-cost, the power supply should not be designed incorrectly. GSM modules require a lot of power depending on the operating mode. Power also needs to be delivered through the power supply extremely quickly (typically from high capacitors); within milliseconds the requirement may climb from several mAs in standby to well over an ampere. If the power is not supplied, many modules reset (see section 5.3.1, starting on pg. 42), and communication is interrupted. Factors for different power requirements include operating mode (GSM/GPRS) and the antenna used (reception quality).

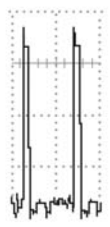
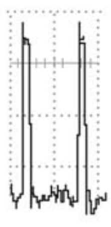
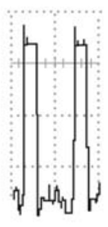
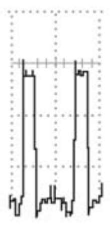
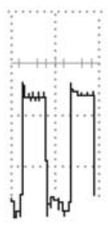
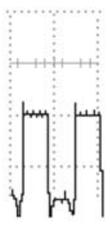
Mode	GSM call	GPRS Class 8	GPRS Class 10		GPRS Class 12	
Timeslot configuration	1 Tx / 1 Rx	1 Tx / 4 Rx	2 Tx / 3 Rx		4 Tx / 1 Rx	
RF power nominal	2W (33dBm)	2W (33dBm)	2W (33dBm)	1W (30 dBm)	1W (30 dBm)	0.5W (27dBm)
Radio output power reduction with AT ^SCFG, parameter >ropr<	>ropr<=1...3	>ropr<=1...3	>ropr<=1	>ropr<=2 or 3	>ropr<=1	>ropr<=2 or 3
Current characteristics						
Burst current @ 50Ω antenna (typ.)	1600 mA	1600 mA	1600 mA	1300 mA	1100 mA	880 mA
Burst current @ total mismatch	1600 mA	1600 mA	1600 mA	1300 mA	1100 mA	880 mA
Average current @ 50Ω antenna (typ.)	250 mA	260 mA	480 mA	400 mA	570 mA	480 mA
Average current @ total mismatch	250 mA	260 mA	480 mA	400 mA	570 mA	480 mA

Figure 3 – Power needs in various operating modes

2.2. PC

A PC is needed to communicate with the mobile communications terminal; the PC must have an RS-232 or USB interface to connect.

A terminal program is also required, i.e. minicom for Linux or Hyperterm for Windows. It is better to use a native RS-232 interface or a USB-to-serial adapter than a USB interface.

Direct use of a USB interface may require additional drivers from Cinterion and the fact that the virtual serial interfaces (USB descriptors) are allocated dynamically must be taken into account. If the terminal reboots with a terminal program open, the PC will assign another virtual serial interface to the terminal after the reboot (e.g., first COM22, then COM23). The same is true if the USB connection cable is briefly removed. Note that the operating system will reassign the interface accordingly.

2.3. DOCUMENTATION AND TOOLS

The following documents and tools are absolutely required:

- MC Technologies MC88(i)T User Guide
- Cinterion EGS5 AT Command Set
- Cinterion EGS5 Application Notes
- Cinterion EGS5 Modem Driver
- Cinterion EGS5 USB Modem Driver
- Cinterion EGS5 Release Notes
- Cinterion Install-CD EGS5 SDK Software Developer Kit
- Cinterion Module Exchange Suite MES

You may need to sign an NDA to obtain some documents and the developer CD from the manufacturer or distributor. The release notes from the previous modules, here TC65 or TC65i, are also important in this context.

3. COMMUNICATION WITH A GPRS TERMINAL

The default settings of the Cinterion module for serial communication are:

- 115000 Baud
- 8 data bits
- 1 stop bit
- No flow control

The terminal is in command mode, i.e., it is waiting for commands. These commands must begin with the letter combination **AT** or **at**. In the factory default setting, the terminal confirms correctly received AT commands with an "OK" URC message (unsolicited response code). If the commands have incorrect syntax or are not correctly ended with a CR, an "ERROR" message is produced; in the command phase the terminal returns every character from the terminal device as an echo.

The hardware supports two alphabets (GSM and UCS2). All examples assume the use of the GSM alphabet.

3.1. COMMAND AND DATA MODE

In command mode, the terminal responds directly to AT commands. Special AT commands can open a data channel, switching the terminal into data mode. In this mode, data are transmitted directly in binary form and do not follow a protocol (see Example 1).

```
AT                         //In command mode
OK
ATD+49170123456            //Initial CSD call
CONNECT                    //Remote terminal accepted the call
                           //and the modem is in data mode
                           //Communication with the remote terminal
                           //Switch into command mode
                           //Connection remains
                           //open in the background
AT                         //Queries possible in command mode
OK
ATO                        //Resume data mode
CONNECT                    //Modem is in data mode
                           //Communication with the remote terminal
                           //Switch into command mode
ATH                        //Hang up
```

Example 1 - Switching between command and data mode

3.2. FLOW CONTROL

The PC and the mobile communications terminal both have buffer memory for serial communication. The buffer memory is used because data transmission between the terminal device and the GPRS terminal is asynchronous, i.e., the speed of data transfer may differ from the GPRS terminal and the GSM network. These buffers compensate for the different speeds. Furthermore, an error-controlled data connection between the GPRS terminal and the GSM network can be established so that if transmission errors occur, the error control will compensate for them. The data sent from the terminal device in the meantime are also temporarily stored in the send buffers. The same is true for the processing times in the terminal device, during which it cannot accept any data.

Flow control helps to inform the remote terminal that a buffer is almost full. If the application empties the buffer, communication is allowed to resume. There are two ways of implementing flow control:

- **Hardware-controlled flow control**
  Signal lines are used (usually CTS and RTS) to signal the remote terminal if data can be sent.
- **Software-controlled flow control**
  The data stream is modified such that two special data bytes (XON and XOFF) signal readiness for receipt. Note that for XON/XOFF, transmission of the transaction data should be ASCII, not binary, since a 0/1 sequence of the transaction data in binary could possibly be interpreted as XON/XOFF.

The default settings of the mobile communications terminal are **no flow control** in command mode and **hardware-controlled flow control** in data mode.

3.3. IMPORTANT AT COMMANDS

AT commands were first used for analog modems. With the advancement of the technology and its application in mobile communications, more and more commands were added. There are a great many AT commands, making it essential to obtain precise information on the command syntax from the Cinterion AT command documentation.

The following tables list several major AT commands with a brief description; commands that are largely manufacturer-independent are differentiated from Cinterion-specific commands. The manufacturer-independent commands should function in much the same way on all mobile communications terminals of all manufacturers.

COMMAND	DESCRIPTION
ATA	Accept call
ATE0	Turn off local echo
ATE1	Turn on local echo
ATD	Dial-in
ATH	Connection terminated, hang up
ATZ	Reset
ATI	Returns manufacturer, model and revision number
AT+CEER	Query of extended error status of last failed connection attempt (ATD)
AT+CGSN	Returns the IMEI (identification number of the terminal)
AT+CIMI	Returns the IMSI (identification number of the GSM network)
AT+CMEE=2	Set the extended error messages
AT+CMGF=1	Set text mode for SMS
AT+CMGL	Lists of existing SMS messages
AT+CMGR	Read an SMS
AT+CMGS	Send an SMS
AT+COPS?	Returns the carrier and registration procedure used (automatic or manual, i.e. permanent registration)
AT+COPS=?	Returns the list of received carriers and indicates whether the given carrier may be used
AT+CREG?	Returns registration status (registered in home network, roaming network, searching, registration not allowed)
AT+CSQ	Returns the RSSI value (also see glossary)
AT+CXXCID	Returns the card ID (identification number of the SIM card)

Table 1 – Manufacturer-independent AT commands

The commands described in table 2 are Cinterion AT commands. Most manufacturers have added their own commands to the standard AT command set. They do not function on all devices and can sometimes cause error messages on the incorrect hardware.

Also be aware that modules from the same manufacturer may not be compatible 1:1 where syntax or timing is concerned (internal processing of the AT commands).

COMMAND	DESCRIPTION
AT^MONI	Monitor command that allows reading a lot of important data on the currently used cell (in highly readable form)
AT^SMONDE	Most reception-rich monitor command, returns all major data on the currently used cell and up to six neighboring cells
AT^SCFG	Expanded configuration, extremely comprehensive and important command

Table 2 – Cinterion-specific AT commands

3.4. AN INITIAL APPLICATION EXAMPLE

The GPRS terminal is connected to the PC via RS-232 and a terminal program is open. If the Cinterion GPRS terminal is started now, it responds with the URC ˆSYSTART. In example 2, the output of the GPRS terminal is shown in red and the user input in black. Superfluous blank lines in the output of the GPRS terminal were deleted. Explanatory comments are preceded by //.

```
^SYSTART                        //Terminal is ready
AT                              //Query
OK                              //Response: OK
ATI                             //Query manufacturer, model and revision
Cinterion
EGS5 REVISION 01.000 OK
AT+CGSN                         //Query IMEI number of the terminal
356612020182202 OK
AT+CPIN?                        //Is the SIM card unlocked?
+CPIN: READY OK                 //Yes, the SIM card is unlocked
AT+COPS=?                       //Query the present providers (here in the border area)
+COPS: (2,"T-Mobile Austri",,"23203"),(1,"orange CH",,"22803"),
(1,"Swisscom",,"22801"),(1,"sunrise",,"22802"),(3,"one",,"23205"),
(1,"SwisscomFL",,"29501"),(3,"A1",,"23201"),(1,"Orange FL",,"29502"),
(1,"FL1",,"29505"),,(0-4),(0,2)
//Here the 1 means that the provider is present,
//the 2 displays the carrier used and
//the 3 means that this provider should be ignored.
```

Example 2 – Initial test with the EGS5

4. USING THE MOBILE COMMUNICATIONS NETWORK WITH A GPRS TERMINAL

This section explains, mainly through examples, how different communications channels can be used.

4.1. POINT-TO-POINT CONNECTIONS

Point-to-point (PtP) connections link two communication points, in this case the GPRS terminals. The advantage of these connections is that they are extremely plain and simple to explain and use based on the traditional telephone network. The biggest disadvantage of this type of communication is poor scalability, i.e., the complexity of an application increases markedly with the number of subscribers. This is due to the basic principle of circuit-switched services requiring a dedicated connection for each subscriber. So if multiple incoming messages need to be processed simultaneously, multiple modems are needed along with an application that manages them (terminal server).

In point-to-point connections, subscribers are identified and addressed via the telephone number (MSISDN).

4.1.1. CALLS

Two types of PtP calls can be made using a terminal:

- Voice call
- CSD call (circuit-switched data)

Note that the SIM card used needs to support the desired service. Standard M2M/SIM rates usually do not offer voice call options any longer, and many network operators no longer support CSD either.

During a call, a dedicated channel is set up with a remote terminal, a voice channel for a voice call and a data channel for a CSD call (9600 bps/14400 bps). These two operating modes also allow transitioning into the fixed network; for example, a CSD call is converted in the fixed network to an analog V.32 modem call. A terminal can initiate outgoing calls and accept incoming calls. Depending on the configuration, the terminal signals incoming calls via a signal on the RING line of ASC0 (the name of the serial RS-232 interface) or via different URCs (see Example 5).

The hardware in the example does not have any audio connections, but there are EGS5-based devices that can be used e.g. to connect a telephone receiver to the terminal.

```
ATD+49170123456;                      //Initiate outgoing voice call
                                      //(the ';' is important)
OK,NO CARRIER,NO DIALTONE,CONNECT,BUSY   //Possible responses from the GPRS terminal
                                      //after the call was set up
ATH                                   //to end the call
```

Example 3 – Initiating an outgoing voice call

```
ATD+49170123456                   //Initiate outgoing CSD call
                                  //(important: without the ';')
CONNECT <text> oder NO CARRIER    //Possible responses from the terminal,
                                  //after the call was set up.
                                  //If the call is successful
                                  //the terminal switches into data mode
                                  //which means that all data entered are
                                  //transmitted to the remote terminal.
Hello, remote terminal; this is an example of a data transmission
+++                               //Switch into command mode
ATH                               //End the data connection
```

Example 4 - Initiating a CSD call

Example 3 shows the setup of a voice call, while Example 4 shows the setup of a CSD call. There are many ways to configure the terminal to signal incoming calls; one simple configuration is shown in Example 5. A CSD call can be used to directly transmit data or as a data transport channel for the PPP protocol, which helps integration into an IP network.

As Examples 3 and 5 illustrate, handling PtP calls is relatively easy. That is also why these solutions are often found in productive environments in connection with extremely simple microcontrollers.

```
AT+CRC=1                            //Differentiation of incoming calls
OK                                  //by type
AT+CLIP=1                           //Display number of incoming call
OK
                                    //Wait for incoming calls
+CRING: VOICE                       //Incoming voice call
+CLIP: "+4917012345",145,,,,0       //Number of the caller
ATA
OK or NO CARRIER
ATH
//Wait for incoming calls
+CRING: REL ASYNC
+CLIP: "+4917012345",145,,,,0       //Number of the caller
ATA                                 //Accept or ignore incoming call
CONNECT <text> or NO CARRIER
+++                                 //If connect successful,
                                    //to switch into command mode
ATH                                 //to terminate
```

Example 5 – Configuration of signaling and acceptance of incoming calls

4.1.2. SMS

SMS stands for Short Message Service and describes a technology for sending short messages. When the GSM alphabet is used, SMS messages can be sent in 7-bit encoding (standard) or 8-bit encoding. In 7-bit encoding, an SMS can contain a maximum of 160 characters, in 8-bit a maximum of 140. The terminal can operate in two modes to handle SMS messages:

- **Text mode** – In this mode, a normal text message can be read or written very easily.
- **PDU mode** – In this mode, the SMS and all headers can be read or written in binary form. This mode is more powerful but considerably more complex. SMS headers contain e.g. the class, type and the encoding used. There are also extended headers that can be used to specify the sender and target port numbers, allowing an SMS to be used similarly to UDP packets.

There are two areas on the terminal for saving SMS messages. One is in the flash of the terminal itself and the other on the SIM card. The AT+CPMS command can be used to configure the sequence and preference of these storage areas. For the following explanations and examples, it is important to retain the configuration selected at the beginning.

```
AT+CMGF=1           //Set text mode
OK
AT+CMGL             //List all unread SMS messages
+CMGL: 1,"REC UNREAD","+4917012234",,"12/05/06,09:54:26+08"
Test Sms 1
+CMGL: 2,"REC UNREAD","+49170123334",,"12/05/06,09:54:44+08"
Test Sms 2
OK                  //There are 2 unread SMS messages output;
                    //the first value indicates the position in the selected memory
                    //followed by the status, source number, date and finally the SMS
AT+CMGR=2           //Read the SMS at position 2 again
+CMGR: "REC READ","+4917023334",,"12/05/06,09:54:44+08"
Test Sms 2
OK
```

Example 6 – Reading an SMS

Just like for calls, the indication for incoming SMS messages has to be configured. Prior to sending, the mode to be used is set, text or PDU, then the destination phone number and the message is entered. SMS is a common communications channel; if the application frequently switches off the modem due to power-save mode, the provider keeps it available and then sends it after successful posting.

```
AT+CMGF=1             //Set text mode
OK
AT+CNMI=2,1           //URC, if a new SMS arrives
OK
                      //Wait for SMS arrival ...
+CMTI: "MT",1         //SMS saved in “MT” memory at position 1
AT+CMRG=1             //Read SMS at position 1
+CMGR: "REC UNREAD","+4917012234",,"12/05/06,09:54:26+08"
Test Sms 1
OK
```

Example 7 – Configuration of a URC for SMS receipt and read-out of received SMS

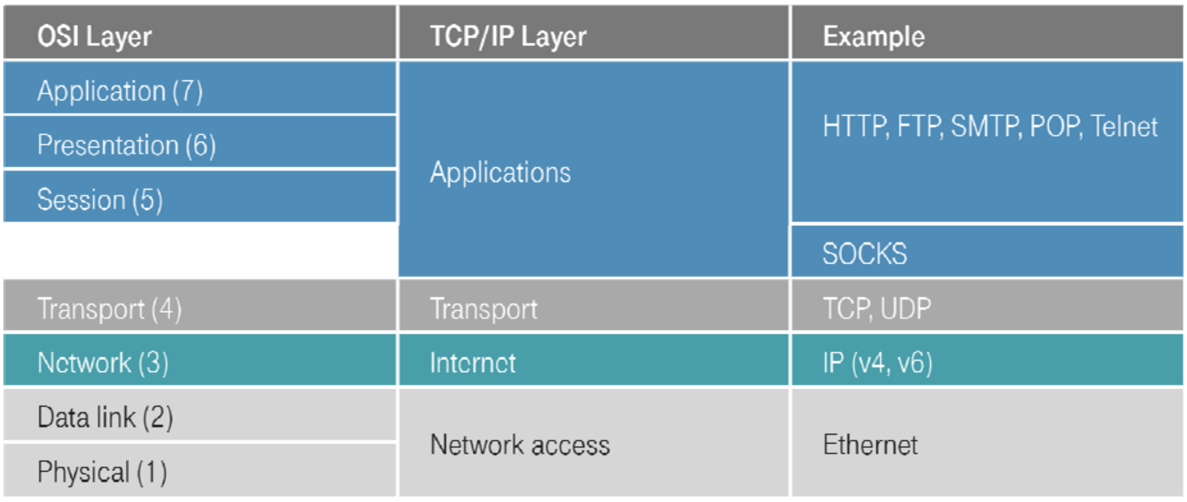
```
AT+CMGF=1             //Set text mode
OK
AT+CMGS="+4916123456" //Send an SMS to the number specified
>                     //Prompt to input SMS text
This is a sample SMS  //Enter text and end with CTRL-Z
+CMGS: 48             //If transmission successful, the
OK                    //message reference is output
```

Example 8 – Sending an SMS

4.2. COMMUNICATION IN THE IP NETWORK

IP, the Internet Protocol, or more precisely IPv4 and IPv6, are network protocols. In this document, IP always refers to IPv4.

IP corresponds to the network layer in the TCP/IP reference model; see Figure 4. Specifying an IP identifies or addresses a subscriber in the network. An IP address consists of 4 bytes, i.e. numbers from 0 to 255. Together with the subnet mask, a number that specifies which part of the IP address describes a network, associated logical subnetworks can be defined.



OSI Layer	TCP/IP Layer	Example
Application (7)	Applications	HTTP, FTP, SMTP, POP, Telnet
Presentation (6)		
Session (5)		
		SOCKS
Transport (4)	Transport	TCP, UDP
Network (3)	Internet	IP (v4, v6)
Data link (2)	Network access	Ethernet
Physical (1)		

Figure 4 – TCP/IP reference model

IP specifies the structure of an IP packet as a header and a payload. This header mainly consists of checksums and the sender and recipient address. IP also specifies how those packets are transported in a local network (LAN) or via gateways into other networks (routing).

In principle IP is robust to the failure of individual network nodes because the packets can simply be sent through other nodes. IP is independent of the actual (physical) transmission path of the data. The transmission media most frequently used are Ethernet, ISDN and mobile communications technologies.

4.2.1. INTRODUCTION TO GPRS

GPRS stands for General Packet Radio Service, e.g. for a packet-oriented data service in the GSM network. Other packet-oriented data services in mobile communications networks are, e.g., EDGE (also called 2.5 G, where the G stands for generation), UMTS (3G) or LTE (4G). With the exception of data throughput from an IP standpoint, how GPRS works can be directly transferred to the other services.

GPRS is not a call, strictly speaking, because unlike the circuit-switched services (PtP calls), mobile resources are reserved only when there is data. That is also the reason why most providers invoice GPRS by data volume and not connection time. In a PtP call, by contrast, the mobile resources needed are permanently allocated regardless of whether or not data are being transmitted. The main application of GPRS is providing an IP network.

GPRS is the precise specification for a data service built on the GSM standard (this technology is also used by the example hardware). The mobile modulation mechanisms are described in the same way as in the network structure. From the base station, to some extent different network infrastructures are used between 2G and 3G, but this has little impact on the following examples since they refer to IP services. Since 3G offers larger bandwidths, run/transmission times of the IP packets through the network are shorter; 3G is typically 5-7 times faster than 2G.

4.2.2. USING GPRS WITH LINUX

Unlike the examples in the previous sections, a Linux PC is explicitly required here since the terminal program alone is insufficient for communication. Furthermore, a Point-to-Point Protocol daemon (PPP daemon or PPPD) is needed. The PPP daemon sets up a connection with the remote terminal via a defined connection path using the Point-to-Point Protocol. The IP packets can then be forwarded via this remote terminal.

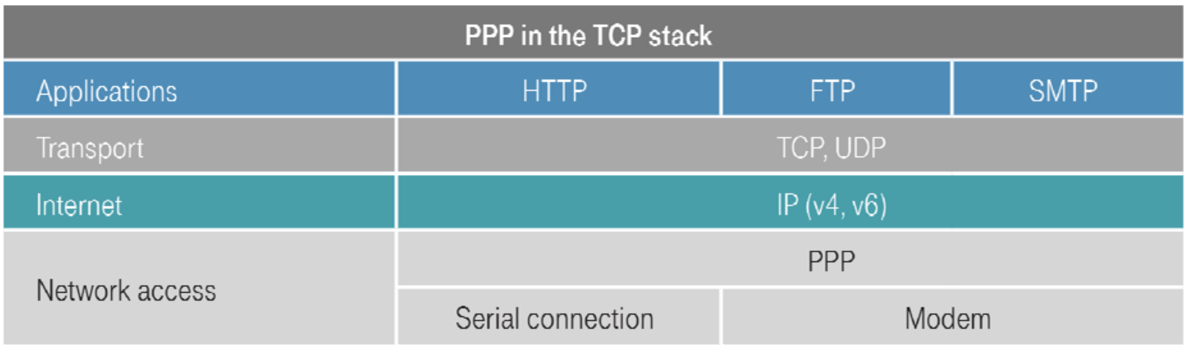


Figure 5 – PPP in the TCP stack

The Point-to-Point Protocol has to be supported in the kernel, therefore the PPP kernel module needs to be loaded and the PPP daemon has to be installed. Example 9 shows how a GPRS data tunnel can be opened to the provider using a terminal program. First a GPRS attachment is executed (connecting the terminal to the GPRS network), then the PDP context is configured, and the context is activated via the Dial-in command. The request for an Internet connection is passed to the network with the PDP context and includes the APN to be used (the Access Point Name is usually transmitted by the network operator with the rate information of the SIM). After the CONNECT output, the terminal switches into the transparent data mode and binary configuration messages from the remote terminal are received. After the CONNECT, the terminal program has to be ended so that it does not continue to block the serial interface.

Then the PPP daemon is started as shown in Example 10; the following parameters are used:

```
AT+CGATT=1                               //GPRS attachment
OK
AT+CGDCONT=1,"IP","internet.t-mobile"    //Set the standard APN
OK
ATD*99***1#
CONNECT <cryptic characters>             //Connect
```

Example 9 – Opening a GPRS connection

/dev/ttyS0: the interface to be used by the PPPD (terminal is connected to the first serial interface).

crtscts: setting the hardware-controlled flow control.

debug: activates the extended output mode of the daemon.

nodetach: means that the daemon should run in the foreground and the output written to stdout.

Example 10 shows the output of the PPP daemon. LCP (Link Control Protocol) messages are first exchanged to e.g. define the size of the MTU (Maximum Transfer Unit) or the authentication protocol (here PAP). Then the compression algorithm to be used is negotiated via a CCP (Compress Control Protocol). The IP configuration (device's own address, remote terminal address, DNS server) is negotiated via the following IPCP (Internet Protocol Control Protocol).

```
pppd /dev/ttyS0 crtscts debug nodetach
using channel 2
ppp0 <--> /dev/ttyS0
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x652e98b2> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x652e98b2> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0xa <mru 1600> <auth pap> <magic 0x3c08c4b3>
    <asyncmap 0x0> <pcomp> <accomp>]
sent [LCP ConfAck id=0xa <mru 1600> <auth pap> <magic 0x3c08c4b3>
    <asyncmap 0x0> <pcomp> <accomp>]
sent [PAP AuthReq id=0x1 user="frostfang" password=<hidden>]
rcvd [PAP AuthAck id=0x1 "TTP Com PPP - Password Verified OK"]
Remote message: TTP Com PPP - Password Verified OK
PAP authentication succeeded
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0>]
rcvd [LCP ProtRej id=0x0 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03 2f 32]
Protocol-Reject for 'Compression Control Protocol' (0x80fd) received
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <addr 10.0.0.1>]
sent [IPCP ConfAck id=0x1 <addr 10.0.0.1>]
rcvd [IPCP ConfNak id=0x2 <addr 10.210.170.19>]
sent [IPCP ConfReq id=0x3 <addr 10.210.170.19>]
rcvd [IPCP ConfAck id=0x3 <addr 10.210.170.19>]
not replacing existing default route via 172.16.1.1
local  IP address 10.210.170.19
remote IP address 10.0.0.1
Script /etc/ppp/ip-up started (pid 11638)
Script /etc/ppp/ip-up finished (pid 11638), status = 0x0
```

Example 10 – Opening a PPP session

```
route add www.heise.de ppp0 //Set explicit routing ¨over the modem ping www.heise.de
//ping, test the connection
PING www.heise.de (193.99.144.85) 56(84) bytes of data.
64 bytes from www.heise.de (193.99.144.85): icmp_req=1 ttl=240 time=701 ms
64 bytes from www.heise.de (193.99.144.85): icmp_req=2 ttl=240 time=336 ms
64 bytes from www.heise.de (193.99.144.85): icmp_req=3 ttl=240 time=496 ms
wget www.heise.de //HTTP Request
--2012-06-18 17:22:24--  http://www.heise.de/
Resolving www.heise.de... 193.99.144.85, 2a02:2e0:3fe:100::7
Connecting to www.heise.de|193.99.144.85|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: \index.html"
    [                             <=>       ] 84,061      6.55K/s   in 12s
2012-06-18 17:22:39 (6.78 KB/s) - \index.html" saved [84061]
```

Example 11 – Setting up IP routing and testing a PPP session

Now a network interface (ppp0) is set up and specified scripts called. The user can add other functionalities to the scripts, which should be invoked when the PPP connection is set up or terminated.

The important thing is that the routing of the PCs was not changed in this example. If IP packets are to be routed via the terminal, then the routing table has to be modified accordingly (see Example 11). Creating a PPP connection first generates just a routing entry for the remote terminal of the PPPD in the routing table, however the standard routing remains on the standard route. Pressing CTRL-C terminates the PPP daemon (see Example 12) and this in turn notifies the remote terminal that the connection was terminated and the network interface closed.

```
CTRL-C
^CTerminating on signal 2
Connect time 2.2 minutes.
Sent 4848 bytes, received 1610 bytes.
Script /etc/ppp/ip-down started (pid 11659)
sent [LCP TermReq id=0x2 "User request"]
Script /etc/ppp/ip-down finished (pid 11659), status = 0x0
rcvd [LCP TermAck id=0x2]
Connection terminated.
```

Example 12 – Closing a PPP session

Initially of course, Examples 8 through 10 have no practical significance, but they clearly show the two parts of the process. The GPRS tunnel to the provider infrastructure is one, and the PPPD process that maintains the connection with the remote terminal is the other. It is also obvious that the PPP process is only weakly coupled to the data tunnel used under it. This is how, just like via CSD, this type of PPP connection can be set up via an analog modem connection or even a null modem cable. What is special about GPRS is that this data tunnel is already intrinsically packet-oriented and requires resources from the provider only when data are flowing.

The data from the examples can very easily be incorporated into a functional system. Example 13 shows a chat script that assumes the functionality of the terminal program and sets up the GPRS data tunnel. The script is expecting the PIN for the SIM card in the PIN environment variable.

```
ABORT   ERROR
ABORT   ’NO CARRIER’
ABORT   ’NO DIALTONE’
ABORT   BUSY
TIMEOUT 3 ’’ AT
OK-AT-OK AT+CPIN?
TIMEOUT 30 READY-AT+CPIN=$PIN-OK AT
OK ’AT’
OK AT
OK ’AT+CGATT=1’
OK ’AT+CGDCONT=1,"IP","internet.t-mobile"’
OK ATD*99***1#
CONNECT
```

Example 13 – Chat script for standard SIM cards from Deutsche Telekom

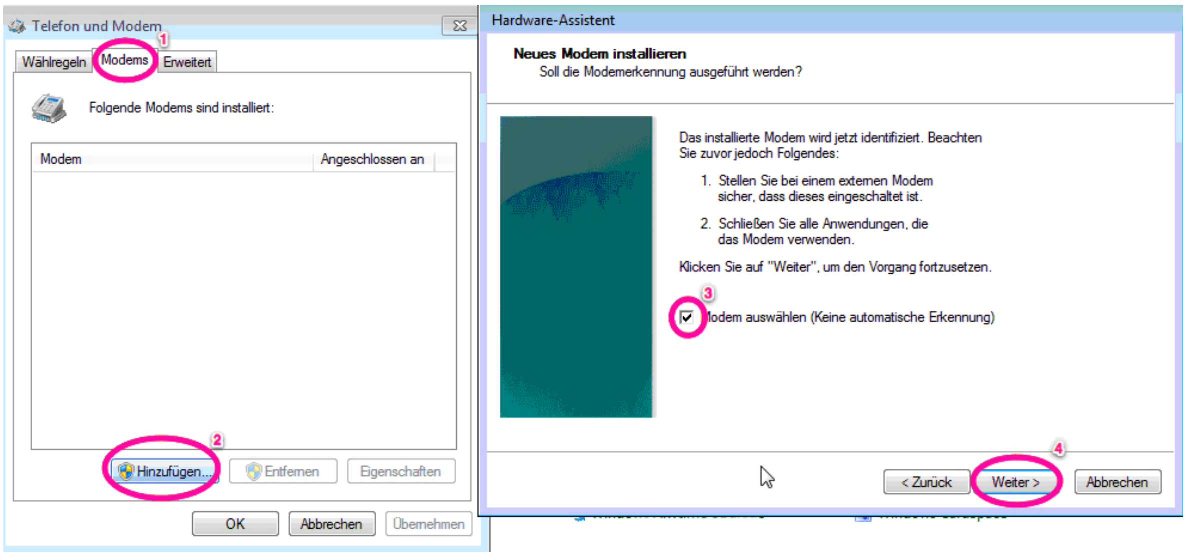
The PPP daemons are called by the pon/poff scripts and the configuration specified in Example 14.

```
/dev/ttyS0 crtscts noauth defaultroute
connect "chat -E -V -f /etc/chatscripts/chat.gprsD1"
```

Example 14 – PPP start script

4.2.3. USING GPRS WITH WINDOWS 7

Similar to the example in section 4.2.1, a GPRS connection can also be set up with Windows 7. The EGS5 terminal driver must be installed first. In the control panel, open the telephone and terminal item. A new device can be added in the Terminal tab. Select terminal must be selected in the hardware wizard (see Figure 6). Next select the RS-232 interface to which the terminal is connected (e.g. COM1). The system verifies that it can detect the terminal and adds the newly set-up terminal.



Folgende Modems sind installiert:

Neues Modem installieren
Soll die Modemerkennung ausgeführt werden?

Das installierte Modem wird jetzt identifiziert. Beachten Sie zuvor jedoch Folgendes:

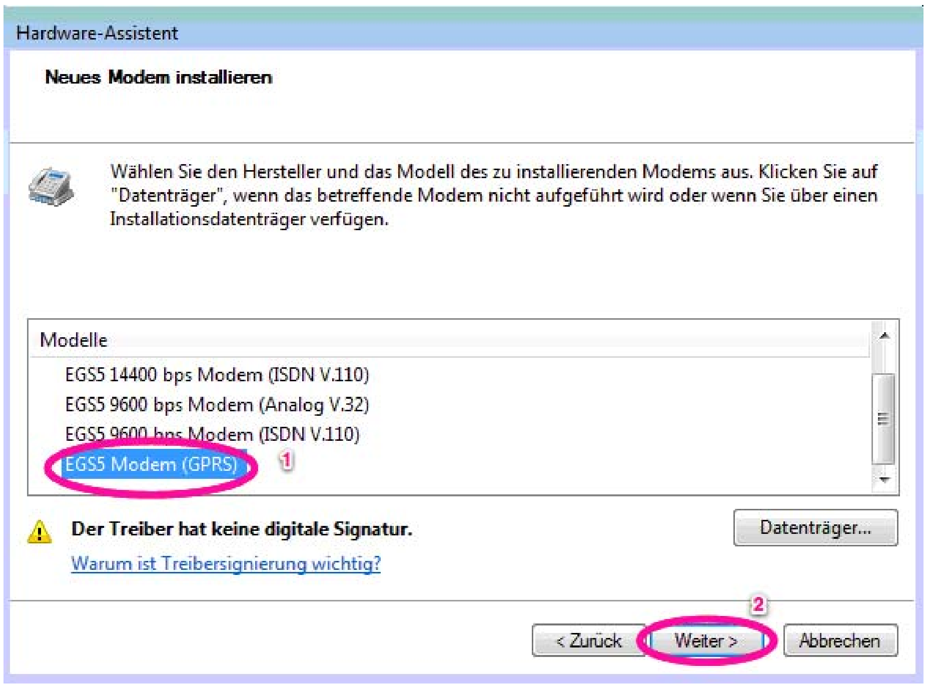
1. Stellen Sie bei einem externen Modem sicher, dass dieses eingeschaltet ist.
2. Schließen Sie alle Anwendungen, die das Modem verwenden.

Klicken Sie auf "Weiter", um den Vorgang fortzusetzen.

[✓] Modem auswählen (Keine automatische Erkennung)

Figure 6 – Adding a modem (GPRS terminal) via the Control Panel

In the dialog that appears, click the Data carrier button and select the EGS5 terminal driver. Under Models, EGS5 Terminal (GPRS) must be selected (see Figure 7).


Hardware-Assistent

Neues Modem installieren

Wählen Sie den Hersteller und das Modell des zu installierenden Modems aus. Klicken Sie auf "Datenträger", wenn das betreffende Modem nicht aufgeführt wird oder wenn Sie über einen Installationsdatenträger verfügen.

Modelle
EGS5 14400 bps Modem (ISDN V.110)
EGS5 9600 bps Modem (Analog V.32)
EGS5 9600 bps Modem (ISDN V.110)
[✓ EGS5 Modem (GPRS)] 1

Der Treiber hat keine digitale Signatur.

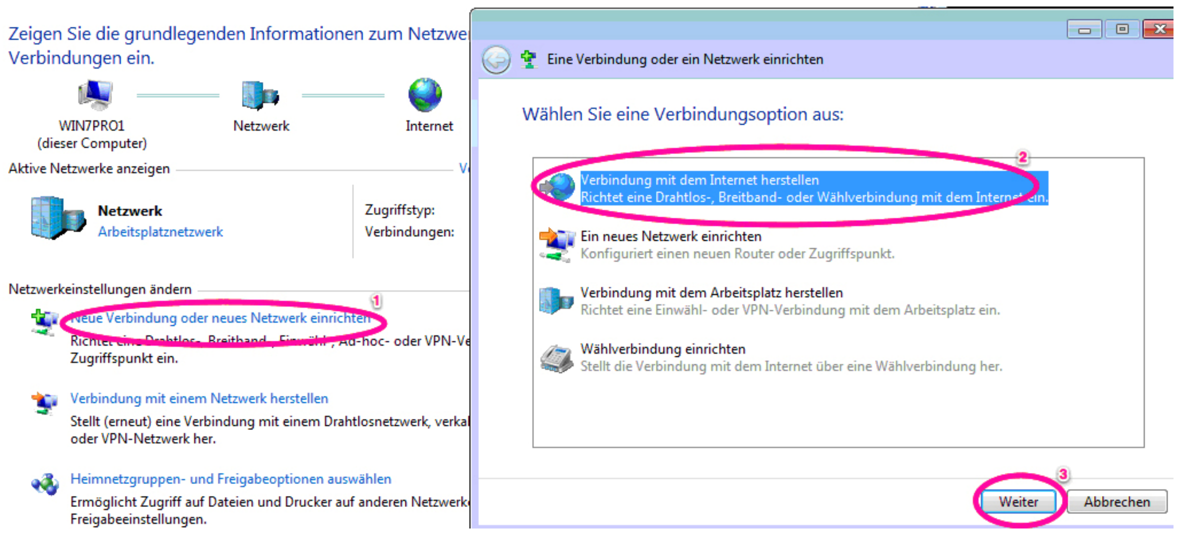
Warum ist Treibersignierung wichtig?

Figure 7 – Selecting the modem driver

Under System Settings, open the Network and Sharing Center.

Add a new connection via Set up a new connection or network. Now select Connect to the Internet (see Figure 8).

If you are notified that a connection to the Internet already exists, click Set Up A New Connection Anyway. For the type of connection, select Dial-in. Specify *99***1# as the dial-in number (like Example 9, see Figure 9).


Zeigen Sie die grundlegenden Informationen zum Netzwe
Verbindungen ein.

WIN7PRO1 (dieser Computer) — Netzwerk — Internet

Aktive Netzwerke anzeigen

Netzwerk
Arbeitsplatznetzwerk

Zugriffstyp:
Verbindungen:

Netzwerkeinstellungen ändern

[✓ Neue Verbindung oder neues Netzwerk einrichten] 1
Richtet eine Drahtlos-, Breitband-, Einwähl-, Ad-hoc- oder VPN-Ve
Zugriffspunkt ein.

Verbindung mit einem Netzwerk herstellen
Stellt (erneut) eine Verbindung mit einem Drahtlosnetzwerk, verkal
oder VPN-Netzwerk her.

Heimnetzgruppen- und Freigabeoptionen auswählen
Ermöglicht Zugriff auf Dateien und Drucker auf anderen Netzwerk
Freigabeeinstellungen.

Eine Verbindung oder ein Netzwerk einrichten

Wählen Sie eine Verbindungsoption aus:

[✓ Verbindung mit dem Internet herstellen
Richtet eine Drahtlos-, Breitband- oder Wählverbindung mit dem Internet ein.] 2

Ein neues Netzwerk einrichten
Konfiguriert einen neuen Router oder Zugriffspunkt.

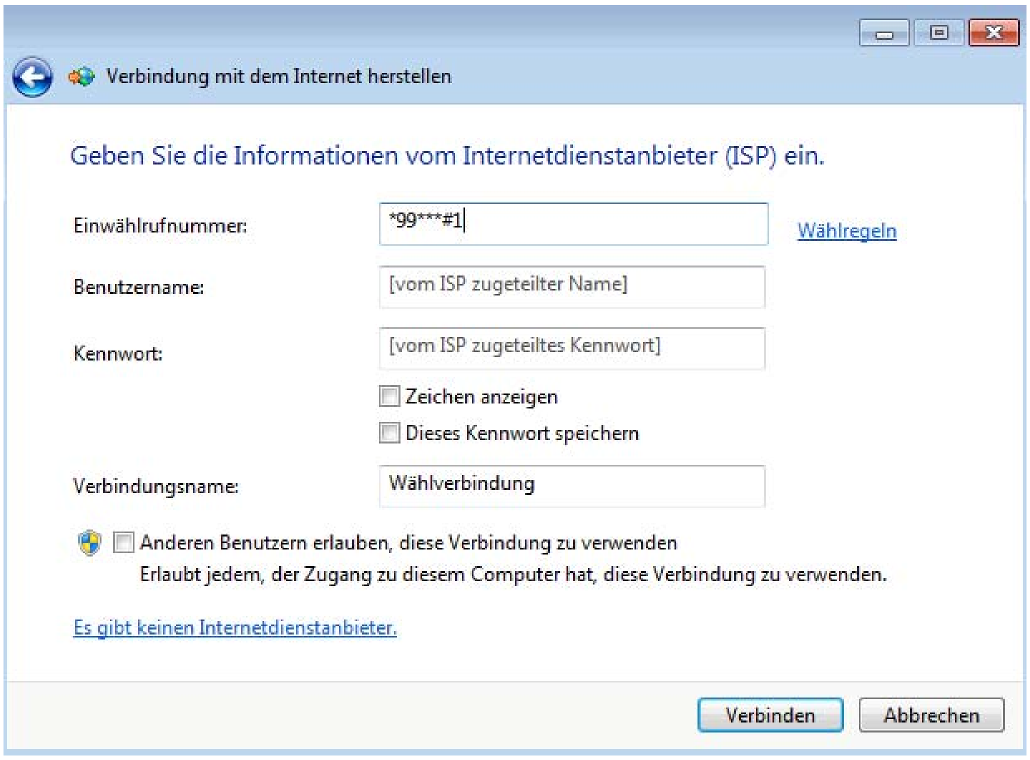
Verbindung mit dem Arbeitsplatz herstellen
Richtet eine Einwähl- oder VPN-Verbindung mit dem Arbeitsplatz ein.

Wählverbindung einrichten
Stellt die Verbindung mit dem Internet über eine Wählverbindung her.

Figure 8 – Adding a new connection

After you click Connect the system attempts to establish a connection. The attempt will fail, however, because an APN has not yet been specified or a GPRS attachment executed. You therefore need to click Skip during the connection test. In the next question regarding connection type, select Public network.

To enter the APN data, open a terminal program (e.g., HyperTerminal) and access the RS-232 interface to which the terminal is connected. Using the terminal program, enter the first two commands from Example 9.



Geben Sie die Informationen vom Internetdienstanbieter (ISP) ein.

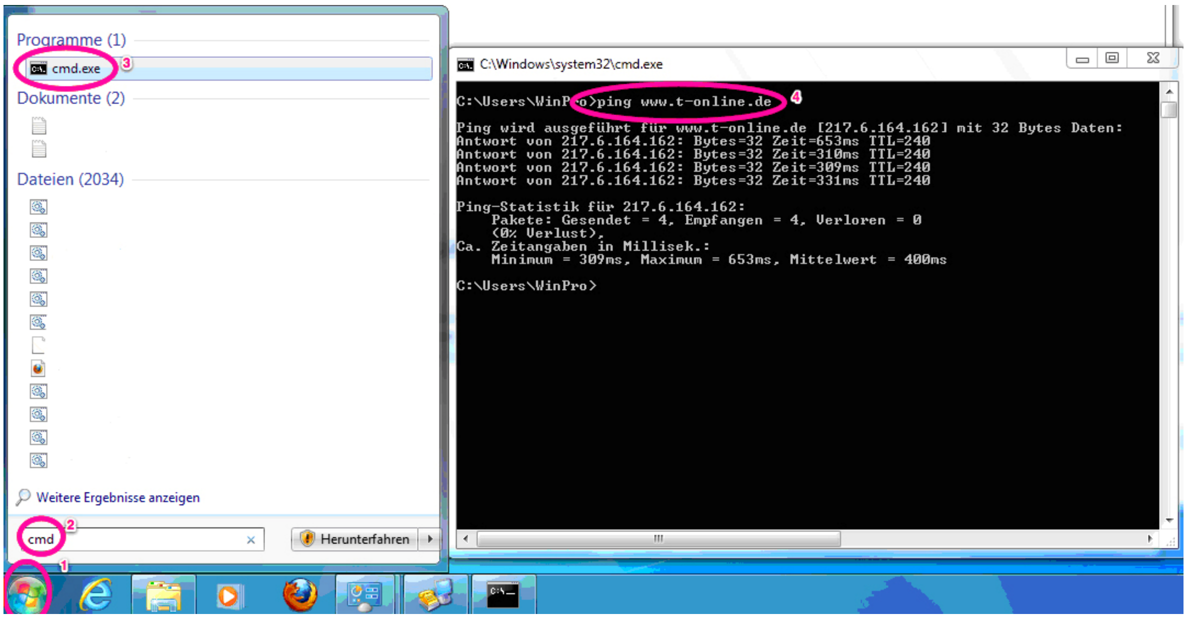
Einwählrufnummer:	*99***#1
Benutzername:	[vom ISP zugeteilter Name]
Kennwort:	[vom ISP zugeteiltes Kennwort]
	☐ Zeichen anzeigen
	☐ Dieses Kennwort speichern
Verbindungsname:	Wählverbindung

☐ Anderen Benutzern erlauben, diese Verbindung zu verwenden
Erlaubt jedem, der Zugang zu diesem Computer hat, diese Verbindung zu verwenden.

Figure 9 – Entering the dial-in number

Close the terminal program and restart the dial-in (in the Network and Sharing Center and adapter settings). Now the network connection via GPRS should succeed.

Now you can test the Internet access using the ping command. Open the cmd command line and enter ping t-mobile.de (see Figure 10).



```
C:\Users\WinPro>ping www.t-online.de

Ping wird ausgeführt für www.t-online.de [217.6.164.162] mit 32 Bytes Daten:
Antwort von 217.6.164.162: Bytes=32 Zeit=653ms TTL=240
Antwort von 217.6.164.162: Bytes=32 Zeit=310ms TTL=240
Antwort von 217.6.164.162: Bytes=32 Zeit=309ms TTL=240
Antwort von 217.6.164.162: Bytes=32 Zeit=331ms TTL=240

Ping-Statistik für 217.6.164.162:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
    (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 309ms, Maximum = 653ms, Mittelwert = 400ms

C:\Users\WinPro>
```

Figure 10 – Testing the network connection

4.2.4. TCP, UDP AND ICMP

The most well known IP-based protocols are TCP, UDP and ICMP.

TCP stands for Transmission Control Protocol and provides the user with a logical data channel over the Internet, one that is very robust against data loss. To emphasize its relationship to IP, it is often called TCP/IP.

TCP implements socket abstraction. A socket is a data endpoint. This type of data endpoint can be connected to any other data endpoint, providing the programmer with a virtual data tunnel or another data endpoint. The only prerequisite is that this endpoint be reachable in the network.

UDP stands for User Datagram Protocol and is a mechanism for sending/addressing data packets without data backup.

Datagram sockets, which are sockets based on UDP, are much more simple than TCP sockets because they offer only the option to exchange datagrams, i.e. data packets, with other data endpoints. A data endpoint can be configured to accept connections or data packets (listener) or can be actively connected to such a listener.

```
AT^SICS=0,conType,GPRS0                   //GPRS as bearer, Profile 0
OK
AT^SICS=0,apn,internet.t-mobile           //Set APN for Profile 0
OK
AT^SICO=0                                 //Open bearer for Profile 0
OK
AT^SICI?                                  //Query information
^SICI: 0,2,0,"10.217.103.105"             //Profile 0, Service is up, IP Address
AT^SISX=Ping,0,heise.de,3           //Send 3 ICMP echo request packets to heise.de
^SISX: "Ping",1,0,"193.99.144.80",609     //DNS resolution successful,
^SISX: "Ping",1,0,"193.99.144.80",291     //i.e., the name server is reachable
^SISX: "Ping",1,0,"193.99.144.80",351
^SISX: "Ping",2,0,3,3,0,0
^SISX: "Ping",3,0,291,609,417
OK                                  //Summarized statistics on the 3 responses
AT^SICC=0                                 //Close Profile 0
OK
AT^SICI?                                  //Query information

^SICI: 0,0,0,"0.0.0.0"              //Profile 0, Service is Down, no IP address
```

Example 15 – Configuration and test of embedded IP stack

ICMP is the abbreviation for Internet Control Message Protocol. It is used by IP for signaling. The most well known use of ICMP is in the ping program, which uses the ICMP Echo Request and ICMP Echo Reply messages to check the reachability of a node.

Many of today's GPRS modules, like the EGS5 in our examples, offer other options in addition to GPRS. Also on board are a TCP/IP and a UDP stack and protocol implementations for HTTP, FTP, SMTP and POP3.

In Example 15, this integrated IP stack is configured and the bearer service started. A ping is also invoked to check the state of the connection. This example shows how an application can monitor its state in the IP network (see section 5.2) and how an analysis can be performed if communication problems arise (see section 5.3). A simple ping call can determine if there is access to the domain name server (UDP functionality and reachability of the DNS in the IP network of the provider) and whether a selected server is responding.

4.2.5. HTTP

There are many protocols based on TCP for widely varying purposes. Three of the most common and important are HTTP, SMTP and FTP.

HTTP stands for Hypertext Transfer Protocol. It is a protocol that was developed for transporting Web page content. It is based on TCP/IP and uses only one TCP connection. HTTP is used as a transport layer by many higher protocols; for example, nearly all Web service implementations are based on HTTP or HTTPS.

HTTPS is HTTP with an SSL-encrypted channel. SSL stands for Secure Socket Layer and provides an encrypted and authenticated data channel. Many embedded systems that record measurement data or supply control functionalities offer a Web server as a user interface, where a browser is used for entering data or control commands. Since HTTP is implemented on the EGS5, HTTP access can be realized with just a few AT commands via the embedded IP stack. This is shown in Example 16.

```
AT^SICS=0,conType,GPRS0                //GPRS as bearer, Profile 0
OK
AT^SICS=0,apn,internet.t-mobile        //Set APN for Profile 0
OKa
AT^SISS=0,srvtype,http                 //Set HTTP protocol, Service 0
OK
AT^SISS=0,hcmethod,0                   //Use HTTP-GET method

OK
AT^SISS=0,conid,0                      //Use Profile 0, Service 0

OK
AT^SISS=0,address,http://www.telekom.de //Set URL, Service 0
OK
AT^SISO=0                              //Open Service 0
OK ^SIS: 0, 0, 2201, "HTTP/1.1 200 OK"  //HTTP Request executed
^SISR: 0, 1                            //Data are available from Service 0
AT^SISR=0,1500                         //Read maximum 1500 bytes from Service 0
 ... Data from Web server ...
^SISR: 0, 1500                         //Other data can be read
...  ...
^SISR: 0, -2                           //No other data present
AT^SISI?                               //Invoke service data
^SISI: 0,6,8085,0,0,0                  //Service 0, Down, 8085 Bytes received, 0
sent
AT^SISC=0                              //Close Service 0
OK
```

Example 16 – Access to a Web page via EGS5 TCP stack

4.2.6. SMTP

SMTP is the abbreviation for Simple Mail Transfer Protocol and is used for sending e-mail. An SMTP client is implemented via the internal protocol stack of the terminal, so e-mail, too, can be sent using just a few AT commands.

This is shown in Example 17. Note that when e-mail is sent, either an SMTP server (see Example 17) or an SMTP relay of the provider is used to send the e-mail to the recipient.

Direct delivery is also possible, but the recipient's SMTP server has to be known or taken from the MX entry of the related domain. Today, however, many servers face tight anti-spam requirements that the terminal often does not meet if a SIM card with a dynamic IP address is used.

```
AT^SICS=0,conType,GPRS0                //GPRS as bearer, Profile 0
OK
AT^SICS=0,apn,internet.t-mobile        //Set APN for Profile 0
OK
AT^SISS=0,srvtype,smtp                 //Set SMPT protocol, Service 0
OK
AT^SISS=0,conid,0                      //Use Profile 0, Service 0
OK
AT^SISS=0,alphabet,1                   //Important, use ASCII alphabet!

OK
AT^SISS=0,address,smtp.testserver.de   //Set outbox server
OK
AT^SISS=0,user,testuser@testmail.de    //Authentication on outbox server
OK
AT^SISS=0,passwd,testpwd               //Password on the outbox server
OK
AT^SISS=0,smFrom,karl@telekom.de       //Sender of the e-mail
OK
AT^SISS=0,smRcpt,klaus@telekom.de      //Recipient of the e-mail
OK
AT^SISS=0,smSubj,Test                  //Reference line of the e-mail
OK
AT^SISO=0                              // Open Service 0

OK
^SISW: 0, 1                            //Service opened, write can proceed

AT^SISW=0,23                           //23 bytes should be written
^SISW: 0, 23, 0                        //23 bytes can be entered

This is a test mail
OK
^SISW=0,0,1                            //E-mail text is entered

^SISW: 0, 0, 0                         //All data were sent

OK
^SISW: 0, 2                            //E-mail was send without errors

AT^SISC=0                              //Close Service 0
OK
```

Example 17 – Sending an e-mail via TCP stack of the EGS5

4.2.7. FTP

FTP stands for File Transfer Protocol and is used to quickly transfer files over a network. Unlike HTTP or SMTP, FTP uses multiple TCP connections: a control channel that transmits the commands, and a new TCP connection for each file transfer. The control channel is always initiated by the FTP client, and the data channels are initiated by the server if FTP is active and by the client if FTP is passive.

Because it uses multiple sockets, FTP can cause problems in mobile communications networks. The ports needed for the data channels in the mobile communications network could be reserved or blocked by a firewall, for example. FTP can also be mapped directly through the terminal; a simple example follows showing how to read out a directory (Example 18).

```
AT^SICS=0,conType,GPRS0                          //GPRS as bearer, Profile 0
OK
AT^SICS=0,apn,internet.t-mobile                  //Set APN for Profile 0
OK
AT^SISS=0,srvtype,ftp                            //Set FTP protocol, Service 0
OK
AT^SISS=0,conid,0                                //Use Profile 0, Service 0
OK
AT^SISS=0,address,"ftp://ftp.debian.org;type=d"  //Access debian.org
OK
AT^SISO=0                                        //Open Service 0
OK
^SISR: 0, 1                                      //Service opened, data are available
AT^SISR=0,1500                                   //Read up to 1500 bytes
^SISR: 0, 8                                      //8 bytes read
debian                                           //Only one subdirectory exists
OK
^SISR=0, 2                                       //All data read
AT^SISC=0                                        //Close Service 0
OK
```

Example 18 – Downloading a directory via FTP

4.2.8. POP3

POP3 is the abbreviation for Post Office Protocol Version 3. This is a protocol for picking up e-mail from a server and managing it. The exact specification can be obtained from RFC 1939. This protocol was also implemented in the embedded IP stack of the EGS5. There are two examples provided.

The first example (19) shows configuration and querying the status of the mailbox (the number of e-mails present and the total size).

```
AT^SICS=0,conType,GPRS0                    //GPRS as bearer, Profile 0
OK
AT^SICS=0,apn,internet.t-mobile            //Set APN for Profile 0
OK
AT^SISS=0,alphabet,1                       //Important, use ASCII alphabet!
OK
AT^SISS=0,srvtype,pop3                     //Set POP3 protocol, Service 0
OK
AT^SISS=0,conid,0                          //Use Profile 0, Service 0
OK
AT^SISS=0,address,"pop3.telekom.de"        //Address of the POP3 server
OK
AT^SISS=0,user,testuser@testmail.de        //User name on the POP3 Server
OK
AT^SISS=0,passwd,testpwd                   //Password on the POP3 server
OK
AT^SISS=0,pcmd,1                           //Command: STATUS
OK
AT^SISO=0                                  //Open Service 0
OK
^SISR: 0, 1                                //Service opened, data are available
AT^SISR=0,1500                             //Read up to 1500 bytes
^SISR: 0, 11                               //11 bytes read
3 9864                                     //3 e-mails with 9864 bytes
.                                          //A dot to close
AT^SISC=0                                  //Close Service 0
OK
```

Example 19 – Configuration of a POP3 service and status query

Example 20 uses the configuration from Example 19 to call a list of existing e-mail messages and then uses the same configuration to retrieve one e-mail.

```
<Configuration as in the previous example>
AT^SISS=0,pcmd,2                            //Command: LIST
OK
AT^SISS=0,pnumber,0                         //List all existing E-mails
OK
AT^SISO=0                                   //Open Service 0

OK
^SISR: 0, 1                                 //Service is open, data are available
AT^SISR=0,1500                              //Read up to 1500 bytes
^SISR: 0, 27                                //27 bytes read
1 5376                                      //E-mail No. 1 with 5376 bytes
2 1057                                      //E-mail No. 2 with 1057 bytes
3 3431                                      //E-mail No. 3 with 3431 bytes
.
OK
^SISR: 0, 2                                 //All data read
AT^SISC=0                                   //Close Service 0
OK
AT^SISS=0,pcmd,3                            //Command: RETRIEVE
AT^SISS=0,pnumber,2                         //E-mail No. 2 read
AT^SISO=0                                   //Open Service 0
OK
^SISR: 0, 1                                 //Service is open, data are available
AT^SISR=0,1500                              //Read up to 1500 bytes
^SISR: 0, 1060                              //1060 bytes read
X-Envelope-From: <karl.mustermann@test.de> //The e-mail content
...
OK
AT^SISC=0                                   //Close Service 0
OK
```

Example 20 – List of existing e-mail messages and e-mail download via POP3 and embedded IP stack

5. GUIDELINES FOR APPLICATION PROGRAMMING

Programming applications that use mobile communications technologies requires adhering to basic principles that to some extent differ greatly from programming general network applications.

For one thing, the quantity of data is a cost factor and for another, many applications in the mobile world are autonomous. Such autonomous applications, such as pump control and monitoring, need to work robustly over long periods without human intervention. In the office or at home, problems in data transmission over mobile telephony networks can be easily solved by removing the USB stick or pressing the reset button on the router. The autonomous application needs to address any problems itself.

It is especially important for the application to constantly monitor its state in the network (GSM as well as IP). Section 5.2 is devoted to this topic because in normal applications the network layer (TCP/IP) and the underlying physical transmission path are encapsulated and therefore strictly decoupled. Section 5.3 covers the various ways to respond to a problem.

5.1. TRANSMITTED DATA VOLUMES AND ESTIMATING THEM

When programming applications for mobile communications, the quantity of data involved and the cost must always be kept in mind. A realistic estimate of the data quantity is therefore important.

If the operating system is fully monitored, a firewall can record the exact amount of data. Even the PPPD daemon is a good source for the data quantities accumulated per connection.

On the application side, however, you can also calculate an estimate by counting the payloads and the number of packets presumably required to transmit them. A TCP packet has a header of at least 40 bytes (in the factory default setting, the modem uses a variant with a 52-byte header with the embedded IP stack), and usually there is an acknowledgment for each packet. So to send a packet with a 20-byte payload (e.g. five 32-bit whole numbers) via TCP, at least a total of 100 bytes accrue.

For invoicing however, the mobile communications provider uses these numbers as a basis only. It is important for users to know the exact terms of the invoice, like block sizes and rounding.

Network operators use the setup and termination of the PDP context as the event for determining the data volume that is billed. If a GPRS connection aborts or the terminal is simply shut off after transmission, the PDP context persists in the network for a time. A timeout in the network (also see 5.2.1) triggers a reset, individual IP and PDP connections are closed and the data volumes of the connections are calculated. Network operators offer different rate plans for events, making it advisable to ensure the communication behavior of the M2M terminal application fits well with the rate. The network infrastructure element on which the data volumes are determined is the GGSN.

Example:
Rates typically round blocks to 1k/10k.
If the user selected 1k block rounding and a 20-byte message was sent one time, it is described as effectively comprising 100 bytes. If the terminal logs off after successful transmission, the network operator takes that as a billing event and in this case rounds it to 1k, which means nearly 900 bytes are "given away". These bytes could have been filled with additional user data or statistics. If the application made it possible to remain online (GPRS logged in), these individual transmissions would be added together. Unless the rate specified otherwise and if the application stayed on the entire day, all the open connections would be calculated and rounded at night, when the network operators often force a disconnection. In our example, only one session would be rounded, and if the user knew the exact time of the forced disconnection, the transmission volume could be targeted to avoid unnecessary rounding.

5.1.1. HIGHER PROTOCOLS AND DATA TRANSMISSION VOLUMES

As described in Section 4.2, today's M2M modules offer a lot of functions especially in the area of IP protocols. Applications that were written on simple microcontrollers can exchange data and control commands with a central server system via IP, for example, using only simple AT commands, which is the big advantage of this setup. These protocols take over authentication and reliable transmission and can be centrally integrated into user administration, plus much more.

But keep one thing in mind. Protocols like SMTP and POP3 were not developed to efficiently transmit user data from M2M applications. Yet there are scenarios in which this type of solution setup makes sense. If there are numerous events in which data need to be sent or received, transmission could be converted. In this case it might make more sense to define a separate transmission protocol that is optimally suited to the application and communication requirement. This could be further combined with the embedded IP stack of the module, such as using a TCP or UDP socket via AT commands, which would transparently transfer the raw data (the protocol that was individually defined).

The following example shows the messages exchanged between the modem and the mail server when the transmission uses SMTP.

```
|Time      | 192.168.1.20                          |
|         |                   | 109.84.0.86       |
|15.824957|         S: 220 m2m-test.tel           |SMTP: 116 S: 220 m2m-test.telekom.com ESMTP server ready.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|16.486316|         C: EHLO [10.57.127.           |SMTP:  82 C: EHLO [10.57.127.214]
|         |(25)     <------------------  (53862)  |
|         |                                       |
|16.487517|         S: 250-m2m-test.tel           |SMTP: 139 S: 250-m2m-test.telekom.com Hello 10.57.127.214
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|16.540105|         S: 250-SIZE 0 | 250           |SMTP: 129 S: 250-SIZE 0 | 250-AUTH CRAM-MD5 LOGIN |
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|17.127395|         C: AUTH LOGIN                 |SMTP:  78 C: AUTH LOGIN
|         |(25)     <------------------  (53862)  |
|         |                                       |
|17.128397|         S: 334 VXNlcm5hbWU6           |SMTP:  84 S: 334 VXNlcm5hbWU6
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|17.566254|         C: ZWdzNQ==                   |SMTP:  76 C: ZWdzNQ==
|         |(25)     <------------------  (53862)  |
|         |                                       |
|17.566391|         S: 334 UGFzc3dvcmQ6           |SMTP:  84 S: 334 UGFzc3dvcmQ6
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|18.086500|         C: ZWdzNQ==                   |SMTP:  76 C: ZWdzNQ==
|         |(25)     <------------------  (53862)  |
|         |                                       |
|18.086971|         S: 235 Authenticati           |SMTP:  98 S: 235 Authentication successful.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|18.446947|         C: MAIL FROM:<egs5@           |SMTP: 110 C: MAIL FROM:<egs5@m2m-test.telekom.com>
|         |(25)     <------------------  (53862)  |
|         |                                       |
|18.470672|         S: 250 Sender OK -            |SMTP:  95 S: 250 Sender OK - send RCPTs.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|18.806059|         C: RCPT TO:<Thomas>           |SMTP: 110 C: RCPT TO:<Thomas>
|         |(25)     <------------------  (53862)  |
|         |                                       |
|18.807836|         S: 250 Recipient OK           |SMTP: 105 S: 250 Recipient OK - send RCPT or DATA.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|19.127286|         C: DATA   |                   |SMTP:  72 C: DATA
|         |(25)     <------------------  (53862)  |
|         |                                       |
|19.128270|         S: 354 OK, send dat           |SMTP: 397 S: 354 OK, send data, end with CRLF.CRLF
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|19.709779|         C: DATA fragment, 3           |SMTP: 122 C: DATA fragment, 331 bytes
|         |(25)     <------------------  (53862)  |
|         |                                       |
|20.106195|         subject: MAIL-TEST            |IMF:   69 subject: MAIL-TEST EGA5, from: egs5 <egs5@m2m-te
|         |(25)     <------------------  (53862)  |
|         |                                       |
|20.107113|         S: 250 Data receive           |SMTP:  89 S: 250 Data received OK.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
|20.425911|         C: QUIT   |                   |SMTP:  72 C: QUIT
|         |(25)     <------------------  (53862)  |
|         |                                       |
|20.426786|         S: 221 m2m-test.tel           |SMTP: 121 S: 221 m2m-test.telekom.com Service closing
channel.
|         |(25)     ------------------>  (53862)  |
|         |                                       |
```

Example 21 – SMTP protocol flow between terminal and mail server

The protocol flow for a complete mail transmission is shown here. The blue arrows show packets traveling between the terminal and the mail server, and the green arrows show packets from the mail server in the direction of the terminal. (25) is the SMTP port of the mail server, (53862) the port used on the terminal, and the red numbers indicate the number of bits in the given packet. In this case the actual user data was 49 bytes, which were transmitted in line 16. If you add up the number of bytes used here, you could say that SMTP is not exactly suitable for transmitting small amounts of data. HTTP or a socket connection is better here for transmission to a server, although the server does need to prepare the data received itself in order to send it on as e-mail.

5.2. DISTRIBUTION OF TASKS BETWEEN THE MOBILE COMMUNICATIONS INFRASTRUCTURE AND APPLICATION

One important question in the design of autonomous applications is which functionalities will the application cover versus which will be provided as infrastructure by the provider.

First we will explain how IP networks are usually made available by the providers. Providers assume customers who use client applications such as smart phones or laptops.

5.2.1. IP TOPOLOGIES IN MOBILE TELEPHONY NETWORKS

Usually private IP addresses are assigned in mobile telephony networks. Only rarely do providers assign public dynamic IP addresses to standard SIM cards. Using add-on products, however, providers can also post public fixed IP addresses, or more frequently a virtual private network.

IP addresses in the address spaces 10.0.0.0/8, 192.168.0.0/16 and172.16.0.0/12 are known as private IP addresses. These addresses are often used in local networks and are characterized by the fact that they are not forwarded by the routers in the global Internet. In principle, this means that such an IP address is not reachable outside of a LAN.

To enable nodes with these IP addresses to access to the global Internet or other networks, such networks are connected to the Internet through a router with NAT functionality (NAT: Network Address Translation) or a so-called masquerade. The masquerader takes the original IP addresses of the IP packets and transcribes them to a public IP address, and writes back the target addresses in the associate response packets (see Figure 11).

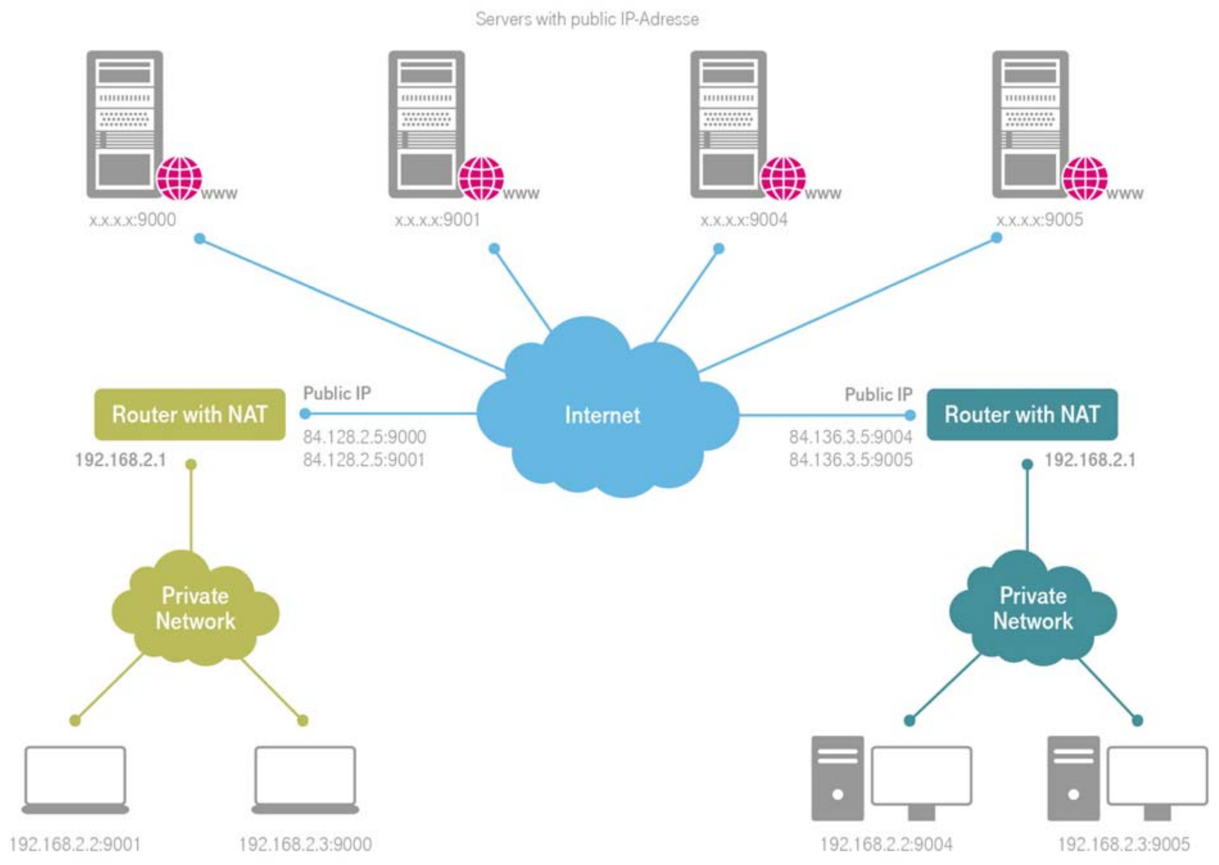


Figure 11 – Private IP addresses and NAT

Color is used to indicate how the assignment to the devices in the private network is established via a call to a specific port number in the router. This initiates the communication from the terminal device in the private network to the server in the public network. The response is assigned back to the terminal device through port mapping.

Furthermore, the IP addresses are typically reassigned to the devices dynamically with each registration. Note the different timeouts in this context:

GPRS Session Timeout: Validity of IP address without data traffic usually 24 hours.

TCP Socket Timeout: Time period in which the provider does not touch an open TCP connection, usually between 5 and 30 minutes. Afterward, the TCP connection is closed normally by the provider or simply invalidated without notifying either of the data endpoints. This occurs at the firewall between the mobile network provider and the Internet. These timeouts can be canceled through extended functions or add-on products.

UDP Timeout: In a network with private IP addresses, the time period within which a response packet from the public network to a UDP query from the private network is routed back before NAT drops the related routing entry (usually between 30 and 90 seconds).

How mobile communications providers behave when the given timeouts are reached often varies widely and sometimes changes; the integrity of a TCP connection can therefore be assumed only at the time data are being exchanged with the remote terminal.

5.2.2. REACHABILITY IN THE MOBILE COMMUNICATIONS NETWORK

The use of private IP addresses makes reaching devices from the public network impossible. The blocking NAT router in this case is the GGSN, on which the firewall for accessing the Internet is also located. Even if the terminal is assigned dynamic, public IP addresses, the problem is still that a new IP address is assigned every time a connection is made to a specific device. In this case the terminal is reachable from the Internet through the public IP, but an application on the customer server has to assign the terminal e.g. to a specific connected machine. This setup does not allow direct terminal-to-terminal communication. The network infrastructure of the operator blocks it. So if data need to be exchanged from terminal to terminal or between their terminal devices, they have to go through the customer server. In this situation, providers help by offering a mobile communications infrastructure, connecting it with a virtual network for the customer and organizing it with fixed IP addresses if necessary.

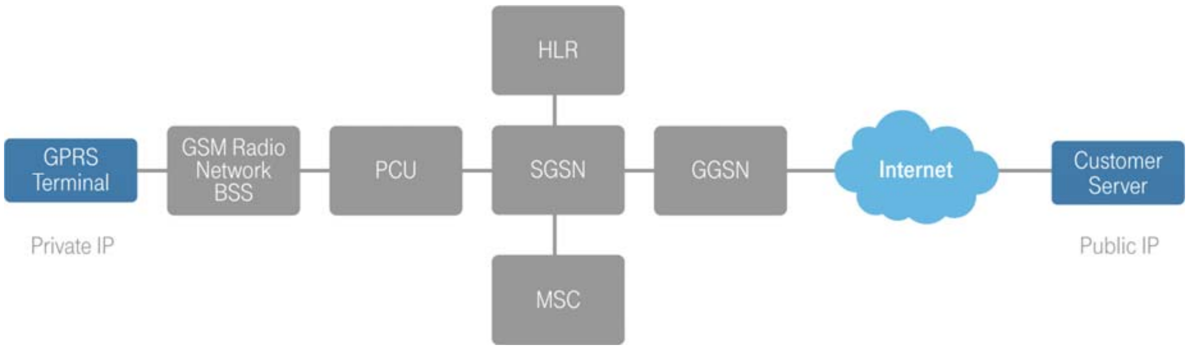


Figure 12 – Private IP address assignment in the mobile communications network

This type of virtual network (VPN: Virtual Private Network) allows dial-in via mobile communications technologies as well as direct IP access. Traditional server applications, like an HTTP or FTP server, are able to work in a VPN, i.e., they can be reached from every node in the VPN. These servers can thus provide their services in the entire VPN. If this type of application is to run autonomously, extra effort is required for monitoring because the application needs to check if the assigned IP address is still valid and reachable (see Section 5.2.2).

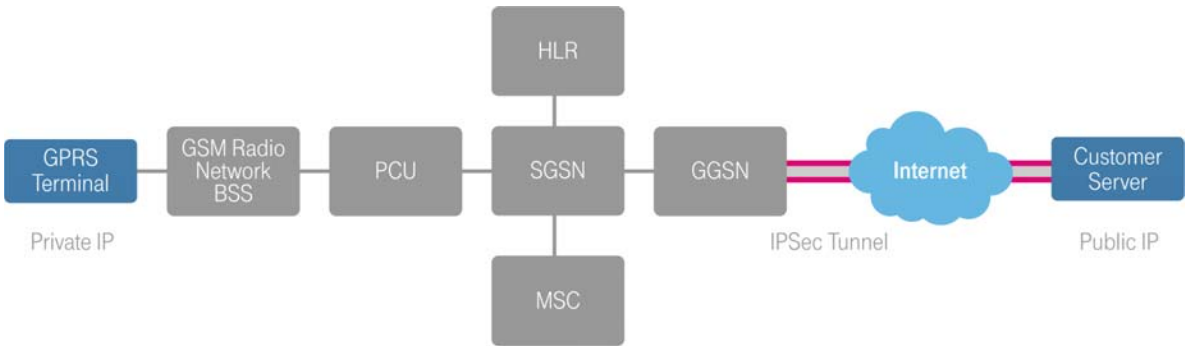


Figure 13 – Private IP address assignment with VPN tunnel

To get by without such a VPN infrastructure, the application has to provide additional functionalities as well as additional infrastructure.

- In an environment with dynamic public IP addresses, an intermediate server that assigns the current IP address to each device is sufficient. One known service of this type is dyndns.org. In this type of service, the terminals are always reachable from the Internet using the same DNS name, even though the IP address changes. This allows the traditional server to continue operating, except for the monitoring modifications. Keep in mind the license terms of such a service, which to some extent exclude commercial use and often do not guarantee availability at all. Other disadvantages of public IP addresses on the terminal are:
- The device is reachable by every Internet user. This considerably increases the security demands on the firewall and the programming of the terminal application.
- The Internet is constantly being searched for valid IP addresses and domains. If a valid IP is located, e.g. via ping, ports and possible functions of the terminal device there are scanned. Even if the application is robust and securely programmed, the IP packets are still calculated as data volumes. Here the network operator is unfortunately unable to differentiate between the application user data and the unauthorized packets.
- In an environment with private IP addresses, the application has to be adapted so it does not function as a server. As a client, it needs to accept the connection to a publicly accessible relay server and forward its services there. This requires considerable effort. As already mentioned, a terminal device with private IP addresses is not reachable from the Internet, therefore the client, in this case the terminal, must always initiate the communication and maintain the connection to the server to preserve the bidirectionality of the data transmission.

5.3. MONITORING

All applications that use AT commands should do so with the extended error codes. They are simple to turn on using the AT+CMEE=2 command. The extended error codes are much more helpful during troubleshooting and analyzing error situations than the generic ERROR output.

Also keep in mind that it is possible for a GPRS terminal to turn off (e.g, due to undervoltage or high temperatures) or – even worse – that a software error could cause it to crash. The application therefore needs to check the GPRS terminal at regular intervals to ensure that the corresponding program sections and communication elements are responding at all. The hardware in our example is able to run M2M applications in JavaTM on an application processor of the GPRS module. In this case, the JavaTM Midlet has to do this and operate the watchdog located in the terminal.

5.3.1. MONITORING THE GSM AND GPRS TERMINAL STATE

In applications that use calls or SMS as a communications channel, monitoring is relatively easy. It usually suffices to monitor the state in the GSM network. This essentially involves the following factors:

- the cell used (base station),
- the signal strength, and along with that,
- whether the GPRS terminal is even logged into a network.

These states can all be checked using the following AT commands:

- AT+CREG=2 configures the GPRS terminal so that it delivers URCs as soon as the login state changes or the GPRS terminal registers at another base station (cell change). The following login states exist:
- 0: not logged in, this can be the case between search periods. If this state persists, it usually indicates a serious problem.
- 1: searching, i.e., the GPRS terminal is looking for a permitted provider to log in. If this state persists, reception is often insufficient.
- 2: logged into home network
- 3: log-in prohibited This state requires contacting the provider used.
- 5: roaming
- AT+CSQ returns the RSSI value (0-31 and 99, if unknown)
- AT+COPS? returns the provider used (if logged in)
- AT+COPS=? returns all available providers (see Example 2)
- AT^SMOND returns a lot of data on the cell in use and up to six neighboring cells

In addition, temperature and voltage should always be monitored. For this task, the GPRS terminal can be configured with the following AT commands:

- AT^SCTM=1 configures the GPRS terminal to issue temperature warnings as URCs; this command can also be used to read out the temperature of the module. The temperature URCs appear as follows:
- ^SCTM B: < m >, m=-2, -1, 0, 1, 2, where 0 is the normal temperature range, 1 and -1 are underage and overage warnings and 2 and -2 are output just before switch-off.
- URCs for critical voltages are always configured and appear as follows:
  - ^SBC: Undervoltage
  - ^SBC: Overvoltage warning
  - ^SBC: Overvoltage shutdown

From these commands and URCs it is obvious that even during normal operations an emergency shutdown of the GPRS terminal can occur. Therefore an application should always be able to restart the GPRS terminal (also see 5.2).

5.3.2. MONITORING THE STATE IN THE IP NETWORK

Besides monitoring the GPRS terminal state and the login state, an IP-based application also needs to regularly determine the validity of the IP address. The following points need to be considered:

- If a PPP daemon is used, the application should be tied into the ip-up.d and ip-down.d scripts to find out when the PPP connection is being set up and terminated.
- If the IP stack of the GPRS terminal is used, the AT^SICI? command can be used to query the state of the IP stack.
- If the application is a client from a communications technology standpoint, pay attention to timeouts during data delivery because the standard TCP settings will not meet many requirements. These timeouts are a very clear indicator of problems or an invalid IP address.
- A mobile server application (the GPRS terminal represents the server) should check for incoming requests. If there are none for some time, the validity of the IP address needs checking, e.g, by pinging a DNS server or even a destination on the public Internet. If the application needs to run in a VPN, a target computer in the remote IP segment – not on the provider infrastructure side – needs to be defined that can be used for these tests.
- If the GPRS terminal is not logged into the GSM network, the IP services are interrupted and should not be used by the application since they cannot be resumed once the GSM connection is reestablished.

5.4. ESCALATION IN THE EVENT OF COMMUNICATIONS PROBLEMS

In this context, escalation means responding to communications problems with increasingly more far reaching measures. Their precise sequence and use are determined by the demands on the application and should be developed on that basis. This escalation scenario refers primarily to applications that use IPs as a communications channel, but also apply in principle to point-to-point connections.

A communications problem is detected when an error is reported from the network layer or the application detects an exceeded time limit. The most common ways to address communications problems are the following:

1. Try to recontact the remote terminal: resend a new TCP socket or UDP packet
2. If possible, use a different remote terminal to see if the original may have simply failed
3. Independent testing of the network functionality through:
   - ICPM Echo Request to a server in the network that is known to respond to such pings
   - DNS query on the name server assigned by the provider
4. Explicit termination and re-setup of the PPP connection
5. Explicit GPRS detachment of the GPRM terminal (using AT+CGATT=0) and reattempt an attachment (AT+CGATT=1). If this is unsuccessful, an analysis of the extended error codes is recommended.
6. APN change
7. Initiate automatic re-login into the GSM network
8. Manual (application-controlled) login with another allowed provider
9. For SIM cards with an embedded application, communication with same (e.g. to change the IMSI)
10. Switch the GPRS terminal into Airplane Mode (AT^SCFG=MEopMode/Airplane,on to turn on and AT^SCFG=MEopMode/Airplane,off to turn off)
11. A reboot of the GPRS terminal (AT+CFUN=0,1)
12. Turn off the GPRS terminal (AT^SMSO), then interrupt the power supply and restart, e.g., including via a watchdog
13. Replace the SIM card

The important thing is that the application records all these attempts to aid the later analysis. It is also conceivable for the application to vary its response in the above scenarios based on the specific problem. It is also important to build increasingly longer pauses into the escalation process to avoid overburdening the mobile communications infrastructure, to save costs and to keep the device from being identified as a problem device by the provider and excluded.

If the application does not succeed in reaching the remote terminal, sending an emergency message via a point-to-point communications channel is always an option. There are certainly situations in which IP services are not available, e.g. temporary node overload during a highway traffic jam if there is only one cell tower in the vicinity, although the base GSM functionality may be available in this case.

The application also needs to monitor any existing GPRS connection for quality and event messages. For example, if the cell changes frequently with respect to the base station used, this can be due to the load ratios at the base station and the downstream base station controllers if the application detects poor or changing signal values/RSSI at the same time. The error rate in the transmission channel would also be quite high here, which error correction in e.g. TCP would attempt to compensate, thereby increasing the required data transmission volumes again.

5.5. MODEM AS MASTER OR SLAVE?

M2M is often used in an environment of existing technologies and systems to achieve additional benefits or process optimization. This requires adding M2M to the existing technology. In cases where terminal equipment must be connected, you need to decide whether to enhance the legacy system or use an external, autonomous solution that is precisely optimized to the M2M requirements.

One of the major advantages of higher-quality M2M modules, like the EGS5 in our example, is that the modem itself can be programmed. Applications can be programmed, in this case in JavaTM. This enables autonomous applications on a component that is needed in any case: the M2M modem module. The existing hardware can be accessed from there via the many different interfaces. Compared to microcontroller programming in C or even Assembler, programming the modem in JavaTM is very quick and easy. Other module manufacturers use e.g. PythonTM or OpenATTM as the application script language. The modem also offers local and remote update processes for software developed in-house, which means changing the software in the field is not a problem. Another advantage is access to TLS/SSL via JavaTM, which has proven itself over many years and is technically mature.

If new terminal equipment is being developed, there are of course other criteria that determine the design of the new solution; often it is not necessary to use two separate application processors (the one in the terminal device and the one in the modem module). Naturally everything can run on the new main CPU, which requires more effort to implement the functions that come standard from the factory with the higher-quality modem modules.

5.6. SECURITY

The security of an application in the mobile communications network involves a variety of issues that are addressed in the following subsections.

5.6.1. SIM CARD SECURITY

The PIN needs to be set on the SIM to prevent misuse. Administering individual PIN numbers does require considerably more work; project-specific and even standard PINs can be coordinated with the provider.

When PIN2 is activated on the GPRS terminal, other security functions can be configured, e.g. a restricted list of allowed telephone numbers or a maximum number of units to be spent.

It is also possible to have the provider lock or release the SIM card for roaming or individual services (SMS, GPRS).

5.6.2. COMMUNICATION SECURITY

Communication in the GSM network is generally encrypted. However the level of data security needed should be precisely analyzed based on the given application.

The VPN solutions offered by mobile communications providers are securely encrypted, but only starting at the point of transition to the Internet in the direction of the end customer (see Figure 13, Page 38); the transmission path and the provider-internal communication are not additionally secured.

If critical data are being transmitted, additional end-to-end encryption and authentication between the subscribers should always be in place. SSL, or more specifically TLS, is the state of the art here. Also keep in mind that encryption will add up to 30% to the data volume transmitted, depending on the transmission quantities and protocol.

5.6.3. PHYSICAL SECURITY OF THE GPRS TERMINAL

If confidential data are being exchanged, the application and the hardware must be physically secure. If the example hardware is programmed in JavaTM and used as an autonomous unit, the following security features can be activated:

- Password protection of the autostart functionality
- Password protection of the update functionality
- Control over the update functionality in the application
- Switch-off of OBEX access to the flash file system
- Option to load a certificate (RSA) on the GPRS terminal and thereby ensure that:
- Only signed software is executed
- Only HTTPS connections to servers with the same certificate or the same certificate chain are allowed
- Changes to security settings are possible only with signed AT commands.
- The SIM card should be securely installed and have a PIN, as already discussed.
- Mobile communications providers also offer a SIM card that works only in the HW assigned and can log into the network only using it. This requires that the IMEI of the terminal or the modem module be permanently assigned to the SIM card (like SIM LOCK mobile telephones).

6. SUMMARY

The core message and most important piece of advice in this guide can be summed up as follows:

An application in the mobile communications network must always check and re-check its state within that network.

A GLOSSARY

- **APN** – Access Point Name: Access point for a GPRS connection
- **CSD** – Circuit Switched Data: Standard for transmitting data calls. Depending on the network operator, transmission rates from 9,600 or 14,400 bps are supported; at connection transitions into the analog fixed network, they are offered with V.32 and in ISDN with V.110.
- **CTS** – Clear To Send: Control message to signal readiness to send
- **DTR** – Data Terminal Ready: Control message to signal connected terminal equipment is ready to receive
- **GSM** – Global System for Mobile Communications: Digital transmission standard for mobile communications networks for voice and data communication
- **GPRS** – General Packet Radio Service: A standard for transmitting data over the mobile communications network
- **HSCSD** – High-Speed Circuit-Switched Data: Permits considerably greater speeds (up to 57,600 bps) than CSD
- **IMEI** – International Mobile Station Equipment Identity: A unique 15-digit serial number for uniquely identifying every GSM and UMTP terminal
- **LTE** – Long Term Evolution: Fourth generation of fast data communication in mobile communications networks, also called 4G
- **PIN** – Personal Identification Number: Helps identify the owner. It protects the SIM card from unauthorized access and must be entered after the device is turned on.
- **PPP** – Point-to-Point Protocol: A network protocol for setting up a connection via dial-up connections
- **PUK** – Personal Unblocking Key: If the PIN is entered incorrectly three times in a row, the PUK is used to unlock the SIM card. If the PUK is entered incorrectly three times, the SIM card can no longer be unlocked.
- **RTS** – Request to Send: Signals that the modem wants to send data
- **RSSI** – Received Signal Strength Indicator: While operating, every GSM module/terminal measures the strength of the signal received by the BTS and reports this value in dedicated mode to the network. The reception strength is measured in dBm. All devices can be queried for this RSSI value using the AT+CSQ command; some modules and terminals even offer direct dBm values: The following RSSI to dBM values are defined (also see the documents for the specific HW in use):

FROM:	TO:	EQUALS:
Less than	-110dBm	RSSI 0
-110dBm	-109dBm	RSSI 1
-109dBm	-108dBm	RSSI 2
...	...	...
-49dBm	-48dBm	RSSI 62
-48dBm	And better	RSSI 63
Value unknown		RSSI 99

- **SMS** – Short Message Service: This service is used to send short text messages of up to 160 characters between mobile telephone and/or GPRS terminals. Current devices support sending more than 160 characters, but the message is split in the network and the cost is higher.
- **SIM Card** – Subscriber Identity Module: Chip card to identify the subscriber
- **XOFF**: Control characters to stop the flow of data on a serial interface
- **XON**: Control characters to continue the flow of data on a serial interface

B USEFUL LINKS

- http://tools.ietf.org/html/rfc791 First RFC on IP and the update, highly recommended
- http://tools.ietf.org/html/rfc793 RFC on TCP and updates
- http://tools.ietf.org/html/rfc767 RFC on UDP
- http://tools.ietf.org/html/rfc194 RFC on HTTP and the updates
- http://tools.ietf.org/html/rfc959 RFC on FTP and the updates
- http://tools.ietf.org/html/rfc821 RFC on SMTP with updates and replacements
- http://tools.ietf.org/html/rfc1081 RFC on POP3 with updates and replacements
- http://tools.ietf.org/html/rfc2757 RFC Long Thin Networks, fundamentals on TCP/IP over networks with high latencies and low bandwidths
- http://de.wikipedia.org search for GPRS, PPP, TCP, UDP, IP and see related English pages
- http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html Introduction to basic terms, recommended reading in this context
- http://ppp.samba.org/documentation.html Various documents on pppd