Enabling Future Computer Applications Using GSM Phones

Graeme J. Proudler
Networks and Communications Laboratory
HP Laboratories Bristol

HPL-95-42
April, 1995

AT commands, GSM, portable computing

This paper describes the kind of simple computer applications that are enabled by a recently agreed command interface to GSM mobile phones. Some less obvious requirements emerged during the development of the commands, and they are described.

1 Introduction

In 1994 a working group from Ericsson, Nokia and Hewlett Packard defined a set of commands[1] for interaction between computers and digital cellular phones, particularly GSM phones. The commands have been accepted as a work item in ETSI SMG4.

The intention behind the commands is to stimulate growth of computer applications that operate with a variety of phones. No existing command set with the necessary functions could be found, and the set of commands was developed to try and prevent market fragmentation. The commands are written according to the established standards for the 'AT' style commands widely used in modems. They allow the computer to find out what is happening at the phone, to control the phone functions normally controlled by a human user, and to control the sending and receiving of messages.

2 Simple applications using mobile phones

Applications can make it easier for the user to operate the radio, or use the radio to provide mobile communications for the computer.

Apart from fax and data, where the phone is used as a modem, the simple applications seem to be:

2.1 Phonebooks

Many manufacturers have been trying to encourage the use of laptop or palmtop computers, or Personal Digital Assistants, as the place where phonebook information is held. They generally have a generous user interface and can store other information associated with a phone number (such as address and context). Unfortunately a user has to read the desired phone number from the computer screen and type it into the phone ! This invokes the response from many potential users that they'd rather use paper - it's cheaper and doesn't need batteries.

However, the electronic version clearly has the advantage if the user can select an entry and the application automatically dials the associated number. It provides greater justification for the phonebook in the computer, and makes it trivial for the user to dial and manage the phonebook in the phone.

[1] The document defining the commands is available free [1]. No charge is made for use of the command set as such, and no other obligations to pay royalties or licences have been identified so far.

2.2 Expanded user interface

Ergonomic restraints and cost usually restrict the size of screen and keypad on the phone. There are often too many functions to allow single-key-press access, so multiple key presses are used for less common functions, and the information on the display is often terse.

On the other hand, a computer, even a small one, usually has a bigger display and larger number of keys, and offers the potential of an easier phone interface if the phone controls and display can be emulated on the computer. This requires remote access to phone controls, to the phone display, and to other phone indicators not on the display. Then the computer can track events at the phone, learn keystrokes for less common functions, and even provide full control of the phone from the computer.

In particular, an expanded user interface would make it easier to send and receive short textual messages over the phone.

2.3 Voice messaging

For most users, a phone is primarily a mechanism for voice communication. This functionality can be improved by the use of a computer. The radio transmitter, radio receiver, phone microphone, phone loudspeaker, and computer memory can be interconnected in several ways. The computer can be used to record a conversation and to act as an answer phone, using the phone as both the communication means and the audio input and output.

3 Less obvious commands

Details of command sets are not particularly inspiring, but some of the design decisions in constructing the set were more interesting than others ! This section describes a few of the more interesting decisions:

3.1 Turning the phone on and off

It's obviously necessary to turn the phone on and off. However, it became clear that 'on' and 'off' meant different things to different people - a phone may have several degrees of 'on' and 'off'. In the end the relevant command referred to functionality, rather than on and off, so the phone is switched on by selecting 'full functionality' and switched off by selecting 'minimum functionality'. It's up to individual phone manufacturers to decide what this means for them. Other degrees of functionality are also provided - various rf circuits may be disabled.

3.2 Configuring the channel

It eventually became clear that if the phone supports data communications, the network must be notified of the required land-based service that will be used to transport the data. This involves selecting a particular land-based service and the characteristics of that land-based service.

At the moment the command set gives a choice of either the PSTN or ISDN, since the PSTN is the most common bearer and ISDN is supposed to be coming ! It's perfectly possible to add other bearers such as ATM when they become widespread.

Another command then selects the characteristics of that service. At present this is simply a choice of synchronous or asynchronous services appropriate to the selected bearer, and the speed of that service.

3.3 Configuring the phone

How should the phone deal with unsolicited reporting of events ? An application may be able to deal with immediate indications if the computer is switched on, but need to request buffering if the computer is about to be switched off (and is unable to wake up in response to external signals). When buffering is turned off and the phone is allowed to pass results to the computer, what happens to items stored in the buffer ?

Eventually it was decided to allocate two command parameters to deal with this situation. Hopefully the resulting choice is rich enough for most needs:

- The parameter 'buffer' determines what happens when the command is executed. If the other parameter in the command allows result codes to be passed to the application, the choice is:
  - clear existing result codes from the buffer.
  - flush existing result codes to the computer.
- The 'mode' parameter determines what happens to the result codes. The choice is:
  - buffer the result codes, so that they are stored for later access by the application. These buffered codes can be flushed to the application by a subsequent use of the command that sets the 'buffer' parameter to flush the buffer and does not select this 'mode' option.
  - pass result codes directly to the application, but throw away result codes that occur when the link between computer and phone is busy.

- pass result codes directly to the application, but buffer result codes that occur when the link between computer and phone is busy and flush them when the link becomes free.
- pass result codes directly to the application, but use the standard in-band embedded technique to pass result codes that occur when the link between computer and phone is busy.

3.4 Making a call

This turned out to be complicated because of differences between traditional methods and the requirements of more modern networks. In the traditional PSTN, the call was established and then the information type (voice, data, fax) was declared. In newer networks such as GSM, the network needs to know the information type before the call can be established, because different network resources are required for different information types.

The peculiar method of call control used in this command set is the result of reconciling these contradictory requirments, so that traditional applications will still operate.

The procedure is as follows: before starting a connection, the nature of the connection must be declared. This is necessary so that appropriate network resources can be allocated. The decaration is whether the connection will use a single type of information, or will alternate between voice and one other type of information. This non-voice type of information is declared by a conventional '+FCLASS' command. Then the destination number is dialled. If the dial string finishes with a semicolon (;) then the call is initiated as a voice call. Otherwise it is intiated as a data or fax call, depending on the previous +FCLASS command.

When the first 'call' is finished, the connection can be terminated or (if network resources were allocated) a data or fax 'call' can be started using the same connection. When the data or fax 'call' is finished, the connection can be terminated or a new voice 'call' started. And so on.

Dialling is done using the standard ATD... command. If the connection is a single voice call, or is a dual type call and the first call over the connection is voice, the dial command must end in a semicolon (ATDT...;). If the connection is a single non-voice call, or is a dual type and the first call is not voice, the dial command must not end in a semicolon (ATD...).

A new 'hangup' command is defined because conventional hangup commands cannot be guaranteed to have the desired result in this new method of call control.

3.4.1 Dialling from a phonebook

The application can request dialling of a phone number in a particular location in phone memory or the phone number corresponding to a name in the phone memory.

The commands **D=....** cause the phone to dial a number stored in phone memory. Since a phone is logically composed of three separate parts, it is possible to specify the memory in either the Mobile Terminal (the phone *per se*), the Subscriber Identification Module (the SIM card), or the Terminal Adaptor (an attachment plugged onto the phone body to deal with access to a computer). It is also possible to dial a number matched to a particular alphanumeric string.

The command may or may not end with a semicolon (;). If it ends with a semicolon, it causes the phone to dial the appropriate number and then dial a semicolon. Obviously, this is used in the call control method described previously to indicate that the first call of a connection is a voice call.

3.5 Dealing with mobile messages

3.5.1 Message types

Can all messages be processed in the same way ? Probably not.

Users may deal with different message services, have different types of message, and need to deal differently with individual services and types. And a typical phone consists of three parts which can be physically separated (the Mobile Terminal, the SIM card, and the Terminal Adaptor). Each of these could have memory.

A user might want to see personal messages immediately, and store them in his SIM card. He might want only indications of news messages, and not care where they are stored.

So where should messages be stored ? The command set allows a choice of the MT, the SIM, the TA, or the Mobile Equipment (meaning any of MT, SIM, TA). In addition, broadcast messages are stored in their own special memory: Broadcast Memory.

It follows that a user will need several different profiles of options to deal with different message types, so the command set provides the ability to save a profile and recall a profile. The options allow the specification of message processing, including the exact memory to be used for storage.

3.5.2 Message assembly

How will an application wish to send a message ?

Phones have the ability to send messages when operating as a stand alone unit. So they can already assemble message fields to construct entire messages. Some applications might want to use this ability, and send individual parts of message fields to a phone, but others will want to send **P**rotocol **D**ata **U**nits (entire assembled messages) to the phone.

The two options have different implications for the set of commands between the computer and the phone. Assembly in the phone requires commands that deal with every individual field that exists in every messaging system to be supported. Using entire PDUs means that a generic command can be used for all messaging systems. However, note that in both cases the computer has to understand how to construct the message in any particular messaging system.

The command set supports message assembly for GSM plus the sending of PDUs. One command specifies which method the application wishes to use. In PDU mode, some related commands operate on a data structure that includes all the fields associated with a message, and a phone does not need to parse the message. In text mode, some related commands operate only on the textual part of the message (the field that is the purpose for the message). If text mode has been selected, a command also specifies the character set that defines the symbols represented by individual bytes. Several sets are supported, including 'RAW' (where no mapping to symbols is defined).

4 References

[1] Copies of the command set may be obtained from its editor:

name	Petri Heinonen
address	Nokia Mobile Phones P.O.Box 68 33721 Tampere FINLAND
tel.	+358 31 285 6804
fax.	+358 31 285 6888
email	phei@nmptre.nmp.nokia.com

Copies are also available from a bulletin board at tel. +358-31-285-6739 or +358-10-505-6739:

- The document is a ZIPped MS Word 2.0 document named GSMAT.ZIP. It is paginated for A4 sized paper and the HP LaserJet 4Si/4SiMX printer.
- Use your own name to log into the BBS and choose your own password. When you log-in the first time BBS asks a few questions. In the Main menu choose the Files menu and then Download. Enter the name of the file and then the transfer protocol you want to use (use ZModem if your equipment supports it).