

EEPROM Specification

Subject: Handling EEPROM parameters from PC

Revision: 0.1

Author: jmb

Last Updated: 07/03/2002 10:51

Last Updated By: jmb

Printout Date: 28/01/2004 09:55

File: C:\Documents and Settings\jmb\My Documents\handling_eeprom_specification.doc

1	Introduction	3
1.1	Phone Tool Overview.....	3
1.2	EEPROM Structure on the Mobile	3
2	File Formats on PC.....	4
2.1	File: eep*.cfg	4
2.1.1	Contents	4
2.1.2	Making the file.....	4
2.1.3	Usage.....	5
3	Using Phone Tool to Handle EEPROM Parameters.....	6
3.1	Setting up Communication.....	6
3.2	Loading a Configuration File.....	7
3.3	Browsing the parameters	7
3.4	Searching for a parameter	8
3.5	Change, read and write EEPROM values	9
3.6	Loading and Saving to File.....	10
3.7	EEPROM versions when Saving and Loading.....	10
3.7.1	Downloading new software	10
3.7.1.1	How Update is Done	11
3.7.1.2	How is the delta file made	11
3.7.1.3	How to be sure all parameters are preserved after a software download.....	11
4	Known Problems.....	12

1 Introduction

This document describes how to handle EEPROM parameters. It covers how to use the PC program *Phone Tool* in relation to EEPROM parameters, describes various file formats and how to generate them. Before going into details, a brief introduction of the *Phone Tool* architecture and the EEPROM structure on the mobile will be given. If you don't want much technical information go to chapter 3.

1.1 Phone Tool Overview

The *Phone Tool* is actually only a GUI front end on top of a DLL, DWDIO.DLL. This DLL encapsulates all the communication details between the mobile and the PC. More than 100 functions in DLL can be called to perform various tasks, e.g. getting the software version from the mobile, or writing a EEPROM parameter value. The DLL is all that is used in production to e.g. calibrate the mobiles (see the production test documentation for details on these functions). The DLL does not handle software download - this is done with a separate program.

The DLL makes it very easy to control the entire mobile from any Windows compiler or program that can call a DLL, this includes e.g. writing the IMEI to the mobile. Therefore the DLL is dongle protected. In the dongle a bitfield is stored, that tells the DLL which DLL functions are allowed by the user. Hence, the complete DLL is shipped to all users, but the dongle controls what is allowed.

1.2 EEPROM Structure on the Mobile

Many different variables are stored in non-volatile memory on the mobile. The term "EEPROM" is used in the meaning "non-volatile memory". The mobile does not actually have a physical EEPROM - it is emulated using blocks in the FLASH. The programmer defines what goes in the EEPROM in a header file: `eep.h`. In this file a big structure is defined, `eep_parms_type`, which defines exactly what the contents of the EEPROM is, and what the layout of the parameters is:

```
typedef struct
{
    eep_static_type  eep_static;
    eep_dynamic_type eep_dynamic;
} eep_parms_type;
```

The "static" and "dynamic" types are the two main types in the EEPROM. The static part contains all the calibration parameters, graphical user greeting and all other parameters set during production. These values are not written by the mobile software itself (except the one place where the PC tells the mobile to do it). This protects these values from accidental corruption (see EEPROM documentation for more details). The dynamic part contains user data, e.g. calendar events, that the mobile software reads/writes during normal operation. Should these values be corrupted, the mobile recovers by restoring the factory settings.

The external interface used when accessing EEPROM values from PC is:

Byte offset from beginning of static or dynamic part, and number of bytes to read/write. If some structures in `eep.h` are moved around, the PC needs to know this, and access a different offset. To keep track of the different structures, a define, `EEP_VERSION`, is stored in the `eep.h` file. This define is increased every time a change is made.

2 File Formats on PC

The chapter will briefly describe the various files used on PC, and how to make them.

2.1 File: eep*.cfg

2.1.1 Contents

As seen in the section 1.2 the PC needs to have byte for byte knowledge of all the parameters defined in `eep.h` on the mobile. This includes any fill bytes that the 16 bit target compiler will insert to ensure 16 bit word alignment. This information is stored in plain text files with the extension “`.cfg`”. These file are called EEPROM configuration files. An example, `eep014.cfg`. This is the configuration file for EEPROM version 14. It contains the following in plain ASCII text:

Addr	Type	Element name
0000	S_CHAR	eep_static.rf_comp.pa_ch_comp[0][0]
0001	S_CHAR	eep_static.rf_comp.pa_ch_comp[0][1]
0002	S_CHAR	eep_static.rf_comp.pa_ch_comp[0][2]
0003	S_CHAR	eep_static.rf_comp.pa_ch_comp[0][3]
0004	S_CHAR	eep_static.rf_comp.pa_ch_comp[1][0]
0005	S_CHAR	eep_static.rf_comp.pa_ch_comp[1][1]
0006	S_CHAR	eep_static.rf_comp.pa_ch_comp[1][2]
0007	S_CHAR	eep_static.rf_comp.pa_ch_comp[1][3]
0008	S_CHAR	eep_static.rf_comp.pa_temp_comp[0][0]
0009	S_CHAR	eep_static.rf_comp.pa_temp_comp[0][1]
000A	S_CHAR	eep_static.rf_comp.pa_temp_comp[0][2]
000B	S_CHAR	eep_static.rf_comp.pa_temp_comp[0][3]
000C	S_CHAR	eep_static.rf_comp.pa_temp_comp[0][4]
000D	S_CHAR	eep_static.rf_comp.pa_temp_comp[1][0]
000E	S_CHAR	eep_static.rf_comp.pa_temp_comp[1][1]
000F	S_CHAR	eep_static.rf_comp.pa_temp_comp[1][2]
0010	S_CHAR	eep_static.rf_comp.pa_temp_comp[1][3]
0011	S_CHAR	eep_static.rf_comp.pa_temp_comp[1][4]
0012	U_INT	eep_static.rf_comp.pa_vcc_comp[0].threshold
0014	U_INT	eep_static.rf_comp.pa_vcc_comp[0].offset_step
0016	U_INT	eep_static.rf_comp.pa_vcc_comp[1].threshold
0018	U_INT	eep_static.rf_comp.pa_vcc_comp[1].offset_step
001A	S_CHAR	eep_static.rf_comp.rxlev_temp_comp[0][0]
001B	S_CHAR	eep_static.rf_comp.rxlev_temp_comp[0][1]
...		
0A16	U_CHAR	eep_dynamic.ll_trace.setup[26]
0A17	U_CHAR	eep_dynamic.ll_trace.setup[27]
0A18	U_CHAR	eep_dynamic.ll_trace.setup[28]
0A19	U_CHAR	eep_dynamic.ll_trace.setup[29]

This is offsets in hex (on the left) of all the elements defined in the EEPROM structure, along with the basic types (signed/unsigned, char/int/long), and the c-syntax for accessing the all parameter elements. This file only defines the structure of the EEPROM, not the values of the element.

2.1.2 Making the file

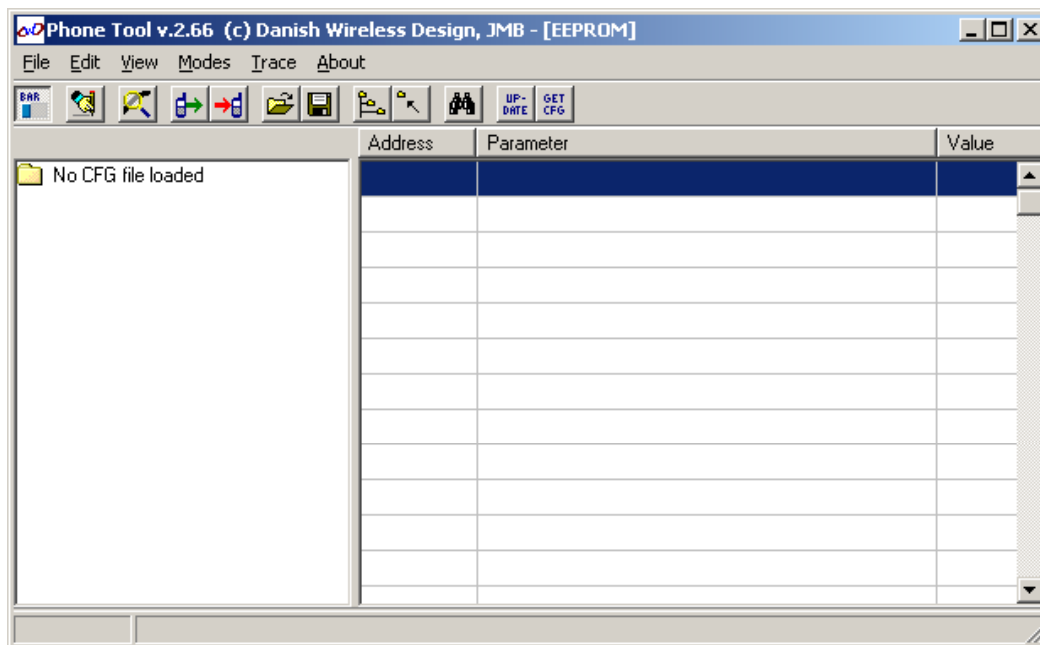
This file would take a long time to make manually, so a small tool has been made to do the tedious work. To make the file, copy the `eep.h` file to the folder `/tools/make_cfg/`, and run the batch file `go.bat`. This will generate a `eep.cfg` file, that must be manually renamed to correspond with the `EEP_VERSION` define in the `eep.h` file.

2.1.3 Usage

When the *Phone Tool* needs to read or write EEPROM parameters from the mobile, it asks the mobile what EEPROM version it uses, and try to load the corresponding configuration file. This means that the *Phone Tool* is shipped with many configuration files (one for each EEPROM version). When a new EEPROM version is made, a new configuration file is also made, and sent to *Phone Tool* users.

3 Using Phone Tool to Handle EEPROM Parameters

When the *Phone Tool* is set into EEPROM mode e.g. via the “Modes” menu, it will look as indicated in the following picture:




The program indicates that no EEPROM configuration file has been loaded. This is needed before any EEPROM parameters can be read or written to the mobile.

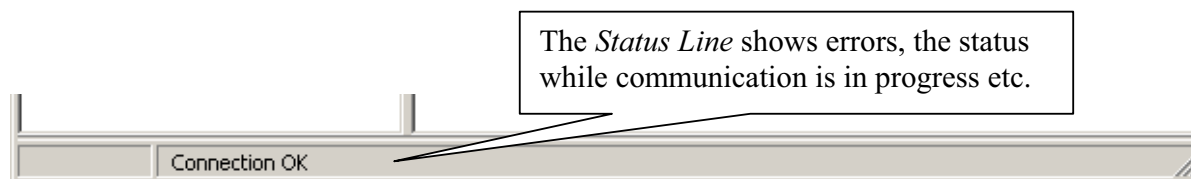
3.1 Setting up Communication

Before any communication can be made to the mobile, the correct COM port must be set up using the menu “Edit→Settings”. If the program can’t open the selected COM port, be sure to close any programs that may be using this COM port. Communication to the mobile is still not possible unless the mobile is connected via a level convert and booted up in test-mode. To set the mobile in test-mode do the following:

- Remove charger and battery to ensure that phone is off
- Re-inserting the battery (not the charger), press and hold in sequence: *, #, and finally power ON simultaneously
- Wait until the backlight turns on, then release the three keys.


Note, you can communicate with the mobile without first setting it in test-mode, but this is the easiest way.

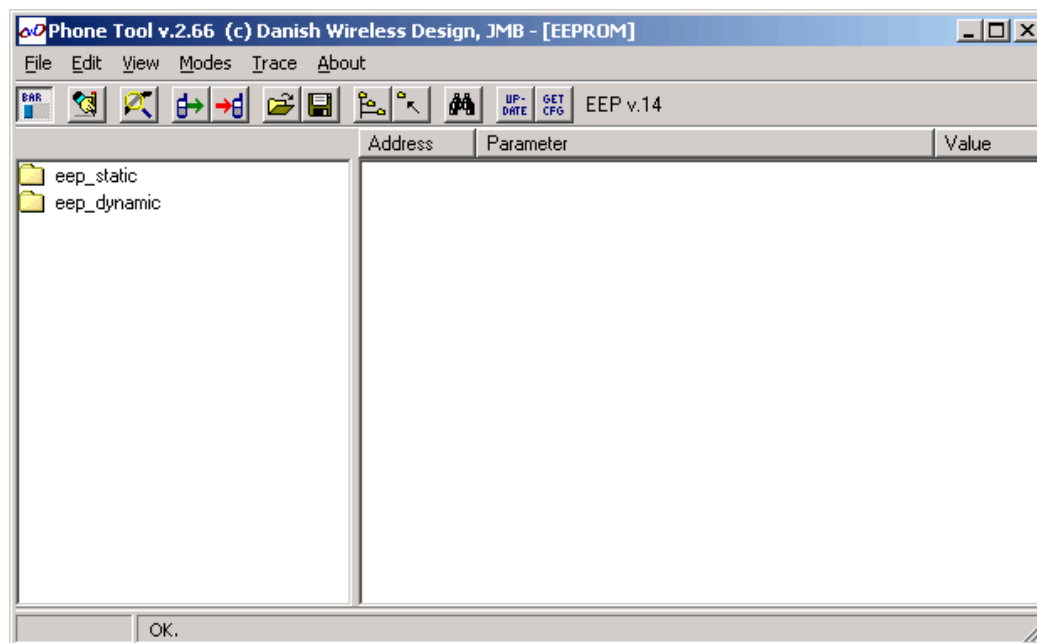
Once, everything is ready try pressing  (check connection). If the Phone Tool shows “Connection OK” on the bottom *status line* everything is ready:




A very common mistake, is to forget to supply the level-convert with power.

3.2 Loading a Configuration File


By pressing the  button, the program will ask the connected mobile what EEPROM version it uses, and load the correct configuration file, as indicated in the following picture:

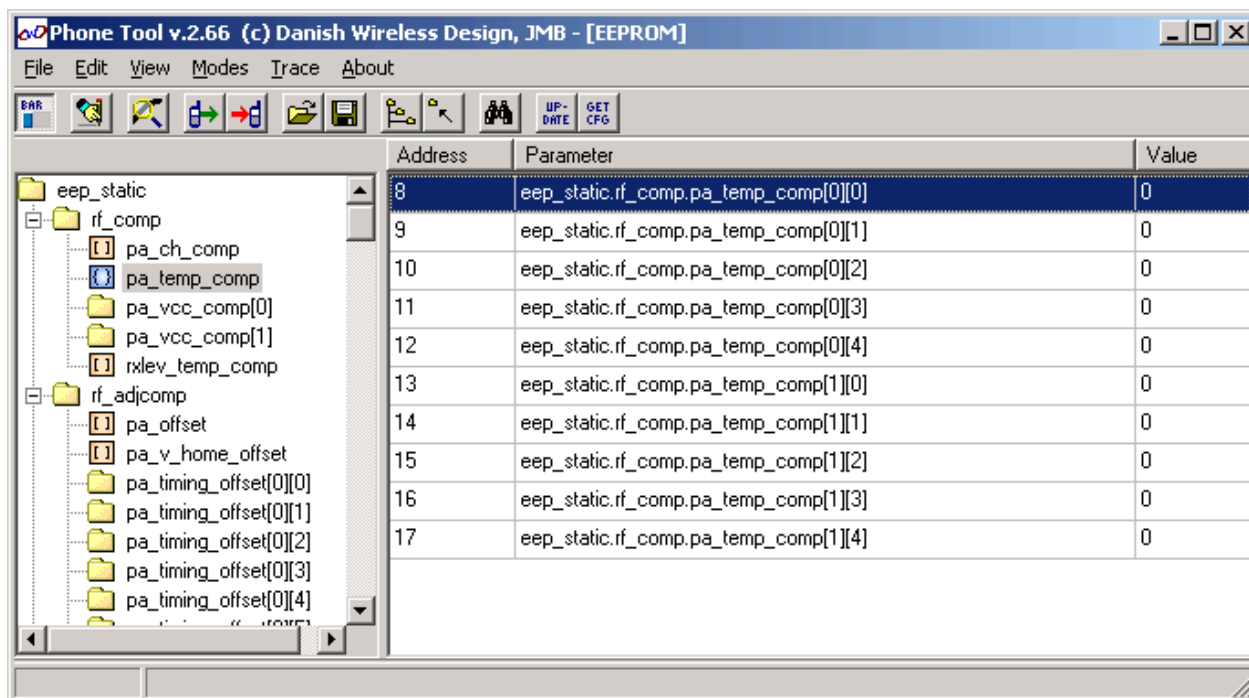




If you switch mobiles while running *Phone Tool*, it is important to let the tool know that you changed mobile, by pressing  again, although the tool will automatically detect new mobiles when reading all parameters from the mobile (see section 3.5 will explain this in further detail).

It is also possible to force the program to load a configuration file by selecting the menu “File→Load CFG file manually”, however do not do this unless there is a very good reason, because you may corrupt the mobile's EEPROM contents if you use a wrong configuration file.


3.3 Browsing the parameters

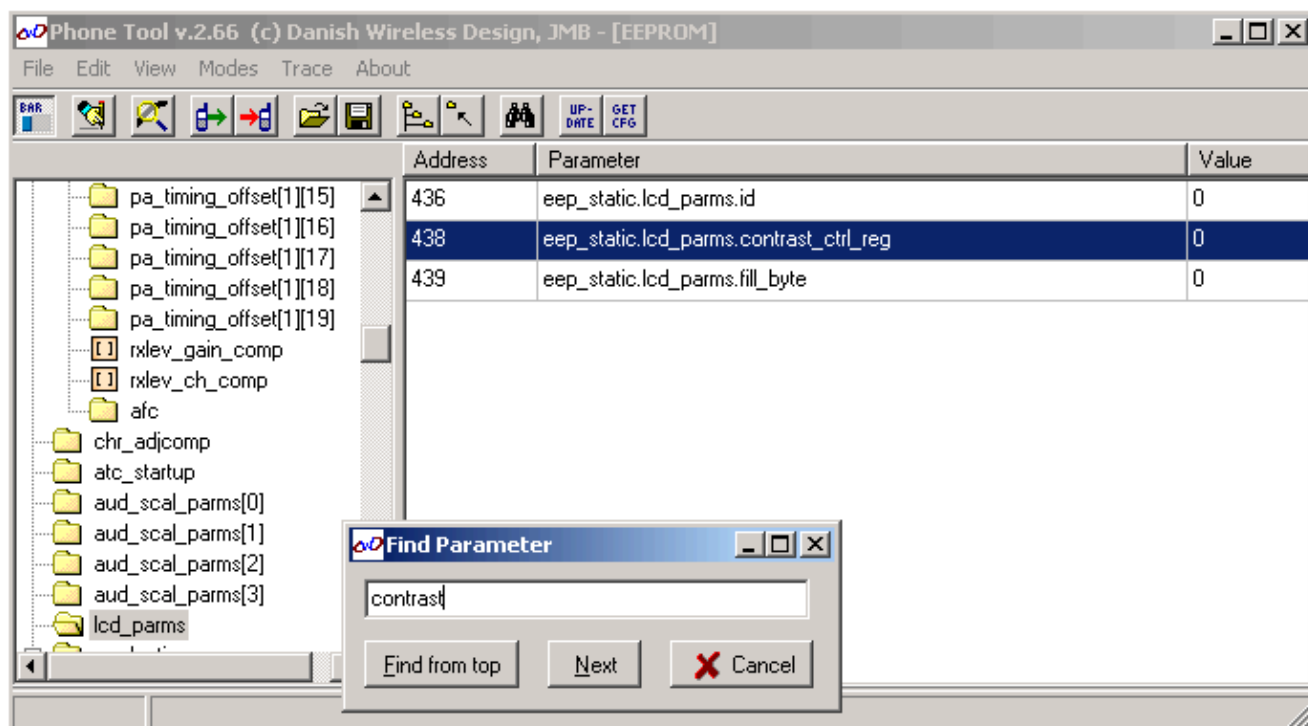
On the left side there is a directory structure of all the parameters. Double clicking on a folder, will open it and show what is inside it. Pressing  (expand tree) expands all folders as seen on the following picture:



Pressing  (collapse tree) hides all folders again (except the static and dynamic folders). Arrays are shown on the left as  icons. Pressing this icon shows the array contents. The blue array is the currently selected array. The elements of this array can be viewed on the right side.

3.4 Searching for a parameter

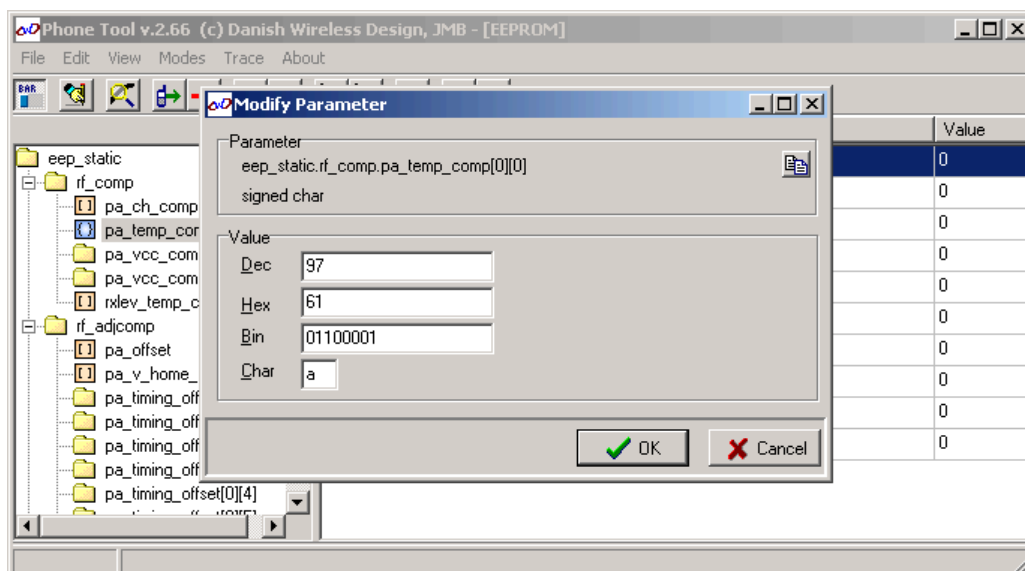
If you want to search for a parameter containing a specific string, press the find button: . In the example below, the user clicked the find button, wrote “contrast” and pressed enter:




This will find the first parameter containing the word “contrast”. On the right, the actual EEPROM parameter names and values are displayed. Nothing has been loaded from a file or from the phone at this time, so all values are zero.



3.5 Change, read and write EEPROM values


Double clicking on a parameter name (or pressing enter while a parameter has focus) brings up the “Modify Parameter” window as illustrated in the following picture:






It is possible to enter the elements value in many different formats, and for convenience it is also possible to copy the C-syntax of the parameter to the clipboard.

If  is pressed the value is stored in the PC’s memory. The default for all values on PC is zero, until parameters are read from the mobile or a from a file.



Before changing any values it is recommended to read all parameters from the mobile by pressing  (read all button). It is then possible to do all the wanted changes as described above. When all changes have been done, it is possible to write them all in one go to the mobile by pressing  (write all button). Note, the write all button really does write *all* parameters, not just the ones that are changed (that why it is recommended always to read all parameters first, so the parameters that are not modified remain untouched).

Every time you press  (read all) the tool automatically detects what EEPROM version the currently connected mobile uses, and loads the corresponding configuration file if not already loaded.

It is also possible to write the parameters that are changed to the mobile automatically as you change them: If the user setting “Write EEPROM parms on the fly” is checked (found in the menu: Edit→Settings), then the parameter is changed on the mobile immediately after you press:  in the “Modify Parameter” window shown above.

The normal procedure when you want to change a few parameters is to read all parameters () , change what you want to change, and if you are not writing parameters “on the fly”, then write them to the mobile again ().

3.6 Loading and Saving to File

Parameter values can be stored in a file, and loaded back. This is done by pressing  (save) or  (load), and selecting the file type “Eeprom File (*.epp)” in both cases. The files you save in this way are plain text files that can easily be edited:





```

...
0010 eep_static.rf_comp.pa_temp_comp[1][3] 0
0011 eep_static.rf_comp.pa_temp_comp[1][4] -40
0012 eep_static.rf_comp.pa_vcc_comp[0].threshold 33
0014 eep_static.rf_comp.pa_vcc_comp[0].offset_step 10
0016 eep_static.rf_comp.pa_vcc_comp[1].threshold 33
0018 eep_static.rf_comp.pa_vcc_comp[1].offset_step 10
001A eep_static.rf_comp.rxlev_temp_comp[0][0] 0
...

```


The offset in hexadecimal on the left side in a “.epp” file is not used when loading, only the parameter name and value are used. The values are always in decimal in this file format.

You can delete lines in a “.epp” file with a text editor if you don’t want all values stored. When the file is loaded again, the parameters in the file are merged with the parameters already stored on the PC. This can be used if only certain values should be updated on the mobile. The procedure for doing this is:

- 1) Generate a “.epp” file with the only values you want changed (e.g. user graphical greeting).
- 2) Connect the new mobile to the PC, and press , to load the correct configuration file for this mobile
- 3) Read all parameters from this mobile by pressing .
- 4) Merge the parameters from the new mobile with the previously saved “.epp” file, by pressing  and load the file.
- 5) Press  to write all parameters stored on the PC to the new mobile. Now only the parameters in the saved “.epp” file have been changed in the new mobile.

3.7 EEPROM versions when Saving and Loading

What happens if I save parameters from a mobile, and want to download them to another mobile that uses a different EEPROM version? The example above with the user greeting, would actually work regardless of what EEPROM versions were used. Here is how it works:


After pressing , the tool asks the mobile what EEPROM version it uses and loads the corresponding EEPROM configuration file. Now, *Phone Tool* knows exactly what parameters are used in the current mobile and where they are located. When the tool loads a “.epp” file, it reads the parameter names in the file line by line, and places the read values at the offsets indicated by the current configuration file. This way, it doesn’t matter where the parameters were located originally, as long as the names remain the same.

If some read parameters can’t be found in the current configuration file, an error list of them all is generated. The parameters in this list could not be loaded, because they do not exist on the currently connected mobile. This happens e.g. when an “.epp” file from a new mobile is loaded into an older mobile. Usually, this can be disregarded, but the user is told that something could be wrong.


3.7.1 Downloading new software

All *dynamic* parameters are reset to their default values after a software download even if you don’t download the EEPROM along with the software. This is done by the software in the mobile on the first power up after a download (see subsection 3.7.1.3 on how to work around this problem).

What happens when I download new software that uses new *static* parameters? These parameters may be important for the mobile to work, so it is important that these new parameters are assigned to a default value.

All you do is press  regardless of what software is downloaded, and the tool will update any new *static* parameters for you.

3.7.1.1 How Update is Done

All new static parameters are added to a file called “delta.eep”. This is a standard “.eep” file (although the extension is wrong). The tool does the following when  is pressed:






- 1) Asks the mobile what EEPROM version it uses and loads the correct configuration file.
- 2) Reads all parameters from mobile
- 3) Open (merges) all parameters with the delta file (without warnings if parameters are missing)
- 4) Writes all parameters to the phone.

3.7.1.2 How is the delta file made

As it is now, all static parameters that need a default value have been added to the delta file ever since EEPROM version 3 (approximately). Hence the parameters in the delta file are accumulated. They are made either by manually editing a saved “.eep” file, or a EEPROM configuration file. There is no need to write anything in the file, other than the full parameter name and its value (offsets or title are not needed).



3.7.1.3 How to be sure all parameters are preserved after a software download

If you want all dynamic and static parameters to be preserved after a software download, the safe way is:

- 1) Read all parameters from the mobile *before* downloading new software (press: ) , and store it in a “.eep” file (press: )
- 2) Download the desired software
- 3) Power up in test-mode and press 
- 4) Load the saved “.eep” file
- 5) Press  (update).
- 6) Write all parameters to mobile () .

The above procedure is safe even if all parameters are reshuffled, and some are missing or added in one of the EEPROM versions.

4 Known Problems

- If the connection to the mobile is reported as okay after pressing  (as specified in section 3.1), but the tool reports communication errors when trying to write all/read all parameters, then check if a program on your PC is using almost all of the CPU power. Some Commercial programs like *RealPlayer* have been known to cause such problems. Close those programs and try again.
- If you save EEPROM parameters as “Flash File (*.eep)” type and try to use it during download, it does not work.
- If the connection jack to the mobile is inserted while the mobile is ready to receive, the first command will most likely fail. To avoid this press  after insertion.
- If the dongle is not inserted the program may wait forever while trying to decrypt the DLL. Exit the program, insert the dongle, and re-start the program.
- If the program startup up, and the connection is fine, but you get strange errors when trying to do different things like memory read, you may not have the rights to perform this operation, or the HASP dongle drivers are not properly installed. Try re-installing the HASP drivers, and let the program auto detect what parallel port you have (if you are not using USB).
- The USB version of the dongle does not work with NT. You will need a parallel version of the dongle for NT (NT does not support USB devices).