EDIABAS

Electronic Diagnostic Basic System

ECU SIMULATOR

VERSION 6a

Contents

1. Revision history

Version 3.0	First release
Version 3.0A	Revised
Version 4.1	Revised for EDIABAS V4.1.0
Version 5	Revised for EDIABAS V5.1.0
Version 5a	Extended for EDIABAS V5.5.0
Version 5b	Extended for QNX
Version 6	Revised for EDIABAS V6.0.0
Version 6a	Revised for EDIABAS V6.4.4

2. Introduction

2.1. About this manual

This manual describes how to use the ECU (Electronic Control Unit) Simulator integrated in EDIABAS. General information about EDIABAS and the ECU description files can be found in [1].

2.2. Notational conventions

The following typographical conventions are used throughout this manual:

Example	Description
SAMPLE.C	Uppercase denotes file names, registers and operating system commands.
apiJob, APIREADY	Bold-faced type identifies keywords and operators of the language BEST/2 and BEST/s as well as the API functions. These words must be written exactly as specified in syntax descriptions.
expression	Italics designate placeholders for values to be entered by the programmer; e.g., file names..
[option]	Words enclosed in square brackets may be optionally specified.
{ **result** | **argument** }	Curvy braces and vertical strokes characterize entries from which only one must be selected, except when in square brackets.
[constant...] job...	An ellipsis (three dots) which directly follows an expression indicates that several expressions of the same type can follow.
`hallo="Test";`	This syntax designates examples, user entries, program outputs and error messages.

`while() {`
`.`
`.}`
	A column or a row comprising three dots indicates that a section of an example was intentionally omitted.

[1]	Reference to a document in References.

2.3. Special features, terms, acronyms

An explanation of the abbreviations used in this and all other EDIABAS documentation can be found in the publication "EDIABAS User Manual" in chapter "GLOSSARY".

3. General

The response behavior of the diagnostic bus interface and the ECUs can be simulated in EDIABAS. This scope of functions is designated ECU Simulator. The ECU Simulator is a constituent of EDIABAS. From the user and the application program view, EDIABAS behaves identically in simulation and normal mode. So-called "simulation files" control the ECU Simulator.
The ECU Simulator is an indispensable tool for developing and testing ECU description files and application programs. In simulation mode, jobs from the ECU description files can be executed without a real diagnostic bus interface and ECU.

4. Overview

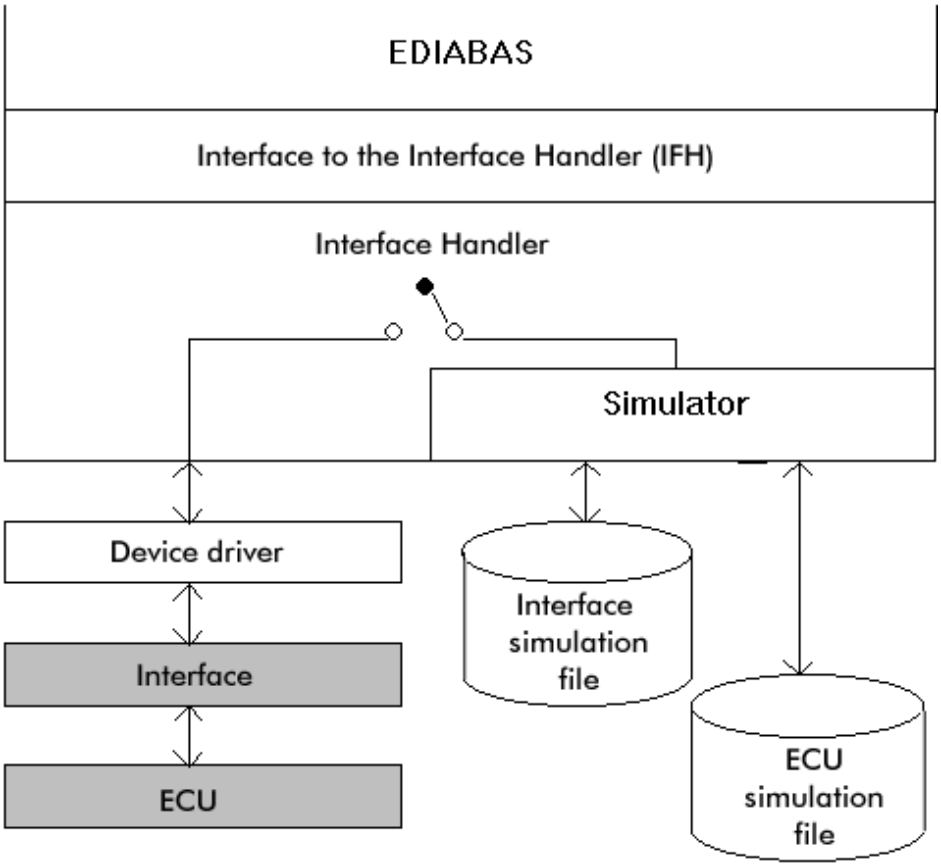
The ECU Simulator is available for all operating systems environment for which EDIABAS is supplied. These environments presently include:

- MS-WINDOWS 95/98/ME/NT4/2000/XP (WIN32)
- MS-WINDOWS 3.11/95/98/ME (WIN16)
- SCO-UNIX
- QNX

The ECU Simulator is a part of the Interface Handler (IFH) and, hence, EDIABAS. It is accessed EDIABAS-internally via the general procedural IFH interface. There are no special access functions for the ECU Simulator which exceed the normal functional scope of the API interface; i.e. no special adaptations have to be made to the application program. The ECU Simulator is controlled via a series of simulation files which determine how the ECU Simulator is to respond to the requests of jobs to the diagnostic interface or ECUs.

Neither device drivers nor diagnostic bus interface and ECU are required for simulation.

EDIABAS — ECU SIMULATOR



5. Control

The ECU Simulator is controlled via three different types of "simulation files". These files are text files and can therefore be edited using every text editor. Since the simulation files are closed again after each access, an ECU description file can be debugged in the development environment under Windows, and a simulation file can simultaneously be edited.

5.1. EDIABAS configurations for simulation

The ECU Simulator is enabled and disabled by means of the EDIABAS configuration element **Simulation**:

Simulation = 0 (Simulation disabled)
Simulation = 1 (Simulation enabled)

The path containing the ECU simulation files is determined using the EDIABAS configuration element **SimulationPath**:

SimulationPath = <*directory*>

The interface to be simulated is determined by the configuration element **Interface**:

Interface = <*Name of the hardware interface*>

5.2. Interface simulation file

There is an interface simulation file for each diagnostic bus interface supported by EDIABAS with which the behavior of the interface can be simulated (see section "Simulations files/Interface simulation file").

5.3. ECU simulation file

ECU simulation files can be created for each ECU by the user. These files can be used to simulated the response behavior of the ECUs (see section "Simulation files/ECU simulation file").

6. Error messages

In simulation mode, a series of error messages have a different cause as than in normal EDIABAS operation. These error messages are listed below:

IFH-0002: NO RESPONSE FROM INTERFACE

This error message is issued when attempt is made to access simulation data which do not exist in the file when accessing an interface simulation file (e.g., when a certain message or label is not found).

IFH-0009: NO RESPONSE FROM CONTROLUNIT

This message is issued when attempt is made to access data which do not exist in the file when accessing an ECU simulation file (e.g., when a certain message or label is not found).

IFH-0026: SIMULATION ERROR

This error message is issued whenever:

The interface simulation file is not found.
The ECU simulation is not found.
The simulation files under UNIX are not lowercase.
The simulation file has an incorrect file format (e.g., CR-LF instead of only LF at the end of the line under UNIX).
An error occurs when accessing (e.g., open, close, read, syntax) a simulation file.
A certain block in a simulation file cannot be found.

This error message does not occur in normal EDIABAS operation.

7. Syntax and contents of the simulation files

Each simulation file consists of both function blocks in which several labels (text marks) may exist and comments. Function blocks are described in the section following the next.

Each function block begins with a heading (title) which must be enclosed in square brackets and ends before the beginning of a new function block or with the file end. Related data (e.g., request messages to an ECU or response messages of an ECU) are both combined in a function block.

A label is assigned to each date (e.g., a message, a version number or a voltage). The label represents the "Name" of the date:

```
;This is a comment
;Function block:
```

```
[Function block]

Label1 = 10000    ;Comment to Label1
Label2 = 00,01,02 ;Comment to Label2
```

7.1. Syntax

The names of function block headings and labels may consist of the following characters:

"A"-"Z", "a"-"z", "0"-"9" ,"_"

All names (both the name of the block heading and the name of a label)can be written in either upper- or lowercase (e.g., "Version" or "version"). The length of a name must not exceed 63 characters.

An assignment must follow each label (except in the block Response) (e.g., UBatt = 13000).

Valid delimiters are blanks and tab characters.

Binary data (interface messages or ECU messages) are written in hexadecimal format and are separated by a comma (e.g., "01,02,0A,0a,FF"). No delimiters may appear within the binary string except for the characters "X" or "x" (both upper- or lowercase is allowed) and "_". The upper- or lowercase characters "X" or "x" in a

block "REQUEST" may represent a number in a binary data string. (e.g., 0X for 00 to 0F, XA for 0A to FA and XX for 00 to FF). Agreements with arbitrary characters can be caused using this 'X' character in blocks which represent data from the interface handler to the interface or ECU. An underscore "_" represents a message with a length of null.

Example:

```
[REQUEST] ;Block heading

;Label with message assignment
id_lesen = 06,00,01,00,07,00

;Label with message assignment
ram_lesen = 06,00,01,XX,XX,XX
          ; Response is made to all messages
          ; beginning with 06,00,01 and 6 characters long

;For message of the length 0
empty = _

;Label with voltage assignment (in mV))
spannung  = 12000
```

Comments always begin with a semicolon (";"). Afterwards, all characters up to the end of the line comprise the comment. Comments may consist of all characters.

Example:

```
;One-line comment
[BLOCK]       ;Comment following a block title
Label = 1234  ;Comment following an assignment
```

7.2. Function blocks

The name of a function block heading is enclosed in square brackets, whereby delimiters are not allowed within. The sequence of the blocks within the simulation file is arbitrary. Section "Simulation files" describes which function blocks are to be entered in the matching simulation files.

Example:

```
[BLOCK]
```

The following function blocks exist:

7.2.1. VERSION

Block title: **VERSION**

Description: This block contains the version number of the interface firmware (EDIC and IDBSS) as 2 bytes in hexadecimal notation, separated by comma.
Two bytes in the sequence <Low byte> <High byte> come from the interface. The <High byte> has the value 00; t <Low byte> contains the version number.

Label: **Version**

Example:

Low byte	High byte	Dec. value	Version number
1E	00	30	3.0

```
Version = 1E,00
```

7.2.2.POWERSUPPLY

Block title: **POWERSUPPLY**

Description: This block contains the value for the battery voltage in millivolts as decimal notation. The values of the system results UBATTCURRENT and UBATTHISTORY can also be determined in this block.

Label: **Ubatt**

Example:

```
Ubatt = 12000 ; Millivolts
```

Label: **UbattCurrent**

Example:

```
UbattCurrent = 1 ; Ubatt is on
```

Label: **UbattHistory**

Example:

```
UbattHistory = 0 ; Ubatt was off
```

7.2.3.IGNITION

Block title: **IGNITION**

Description: This block contains the voltage value at the ignition in millivolts as decimal notation. The values of the system results IGNITIONCURRENT and IGNITIONHISTORY can also be determined in this block.

Label: **Ignition**

Example:

```
Ignition = 12000 ;Millivolts
```

Label: **IgnitionCurrent**

Example:

```
IgnitionCurrent = 0 ; Ignition is off
```

Label: **IgnitionHistory**

Example:

```
IgnitionHistory = 0 ; Ignition was off
```

7.2.4.PORT

Block title: **PORT**

Description: This block contains the values which are applied at the individual ports. The values at the ports zero to eight are the voltages on the analog inputs in millivolts (EDIC). The value at port nine is the digital value of the jumper field (EDIC).

Labels: **Port_0** to **Port_9**

Example:

```
Port_5 = 5000 ;Millivolts
```

7.2.5.REQUEST

Block title: **REQUEST**

Description: This block contains the requests (messages) to the interface or an ECU. A label is assigned to each request. The label names in this block can be selected by the user. For each label, the block "RESPONSE" must contain a label with the same name to which the response to the corresponding request is assigned. If the request contains an "X", this character represents an arbitrary number. If the request contains "_ , the character represents a message with the length 0.

Label: **Any**

Example:

```
Telegramm1= _
Telegramm2= 01,02,XX,03
```

7.2.6.RESPONSE

Block title: **RESPONSE**

Description: This block contains the responses to the requests contained in block "REQUEST". This block may also contain labels without

assignments; in this case, 0 bytes are returned as a (valid) response message.

Label: See **REQUEST**

Example:

```
Telegramm1= 11,22,33,44
Telegramm2= 55,66,77,88
diagEnde =
```

7.2.7.LOOPTEST

Block title: **LOOPTEST**

Description: This block contains the result of the line test. 0 designates that the test is not OK; 1 designates that the test is OK.

Label: **Looptest**

Example:

```
Looptest = 1 ;Test OK
```

7.2.8.KEYBYTES

Block title: **KEYBYTES**

Description: This block returns the key bytes and the identification data of an ECU (when provided in ECU concept).

Label: **Keybytes**

Example:

```
Keybytes = 01,02,04,07 ;....
```

8. Simulation files

8.1. Interface simulation file

A separate simulation file exists for each interface to be simulated. The file name consists of the interface name with the extension ".SIM" (e.g. EDIC.SIM). The name of the simulation file must be written in lowercase under UNIX. The simulation file must exist in the correct file format (under MS-DOS/WINDOWS CR-LF, under UNIX only LF at the end of the line. The setting of the EDIABAS configuration element is to be used as the interface name. **Interface**. The interface simulation files contain the interface-specific return values. The following function blocks are to be entered in this file:

- VERSION
- POWERSUPPLY
- IGNITION
- PORT
- LOOPTEST
- REQUEST
- RESPONSE

A special mechanism was implemented for the interface EDIC and IDBSS for sending and receiving an ECU message in raw mode; i.e., the data are sent 1:1 to the diagnostic bus interface without interpretation. Sending and receiving an ECU message is conducted in the following manner:

1. Send a message to an ECU (control bytes 04,00 + message data). For this purpose, the request message is entered with leading underscore in the block REQUEST.

```
[REQUEST] ;Request messages to the interface
_send_id_lesen     = 04,00,01,00,00,03,00,00
```

The interface responds with BUSY for acknowledgment (control bytes 01,00). This response is entered under RESPONSE.

```
[RESPONSE] ;Responses from interface
_send_id_lesen     = 01,00
```

2. In order to read the ECU response from the diagnostic bus interface, the interface status must be interrogated (control bytes 07,00). This request is entered again under REQUEST.

```
[REQUEST] ;Request message to the interface
_send_id_lesen     = 04,00,01,00,00,03,00,00
```

```
requestState        = 07,00
```

If the control bytes 07,00 are sent directly after an ECU message request, the response does not appear under the same label as the request (in the example "requestState"). Instead, however, it under the label of the ECU message request (now, however, with a leading "X" (in the example "Xsend_id_lesen").

```
[RESPONSE] ;Interface response
_send_id_lesen      = 01,00
Xsend_id_lesen      =
02,00,0D,01,F6,30,32,38,35,30,30,36,30,30,34;,...
requestState        = 00,00
```

The response appears under the same label as the request (in example "requestState" only when the control bytes 07,00 are sent without pervious ECU message request.

Example:

```
;************************************************
;******** Simulation file for EDIC ***************
;************************************************

[VERSION] ;Version number of the interface
Version = 5

[POWERSUPPLY] ;Supply voltage
Ubatt = 12000 ;Supply voltage in millivolts
UbattCurrent = 1 ;Supply voltage is on
UbattHistory = 1 ;Supply voltage was on

[IGNITION] ;Ignition
Ignition = 12000 ;Ignition voltage in millivolts
IgnitionCurrent = 1 ;Voltage in on
IgnitionHistory = 0 ;Voltage was off

[PORT] ;Input values for the ports 0 to 8
Port_0 = 10
Port_1 = 21
Port_2 = 32
Port_3 = 43
Port_4 = 54
Port_5 = 65
Port_6 = 76
Port_7 = 87
Port_8 = 98

[REQUEST] ;Request messages to the interface
reset              = 01,00
resetParameter     = 02,00
setParameter       = 03,00,03,00,02,10,01,00,00,D0,07,F4,01,01
stopFrequent       = 06,00
requestState       = 07,00
requestIdent       = 08,00
requestLastMsg     = 09,00
version            = 0A,00
loopTest           = 0B,00
baudrate           = 20,00,00       ;Baud rate is 9600 baud
readJumper         = 21,00
setDigiOut         = 22,00,FF,FF    ;Set all outputs
setProgVlt         = 23,00,10,27,01 ;Voltage at 10 volts
getAnalog          = 24,00,00       ;Analog input 0
switchSiR          = 26,00,E8,03    ;1000 ms
_send_id_lesen     = 0X,00,01,00,00,03,00,00
_send_fs_loeschen  = 0X,00,01,00,00,03,00,05
_send_diagende     = 0X,00,01,00,00,03,00,06
_send_fs_lesen       = 0X,00,0A,00,00,03,00,07
_send_ack          = 0X,00,01,00,00,03,00,09

[RESPONSE] ;Responses from interface
reset              = 00,00
resetParameter     = 00,00
setParameter       = 00,00
```

```
stopFrequent       = 00,00
requestState       = 00,00
requestIdent       = 00,00,04,01,09,03,0D,01,F6,30,32,38,31,30,30;,...
requestLastMsg     = 00,00
version            = 00,00,03,00
loopTest           = 0A,00
baudrate           = 00,00
readJumper         = 00,00,3F
setDigiOut         = 00,00
setProgVlt         = 00,00
getAnalog          = 00,00,E8,03    ;1000 millivolts
switchSiR          = 00,00
_send_id_lesen     = 01,00
Xsend_id_lesen     = 02,00,0D,01,F6,30,32,38,35,30,30,36,30,30,34;,...
_send_fs_loeschen  = 01,00
Xsend_fs_loeschen  = 02,00,03,00,09
_send_diagende     = 01,00
Xsend_diagende     = 02,00
_send_fs_lesen     = 01,00
Xsend_fs_lesen     = 02,00,10,DB,FC,B2,04,8A,00,05,A5,03,1A,01,02;,...
_send_ack          = 01,00
Xsend_ack          = 02,00,03,00,09

[LOOPTEST] ;Result of the Looptest
Looptest = 1
```

8.2. ECU simulation file

A simulation file exists for each ECU for which contains an own diagnostic bus address. This file contains the request and the response messages of the ECU. Except for the extension ".SIM" (e.g. DME31.SIM), this file has the same name as the ECU description file. The name of the simulation file must be written is lowercase under UNIX. The simulation file must exist in the correct file format (under MS-DOS/WINDOWS CR-LF, under UNIX only LF at the end of the line). The following function blocks are to be entered in these files:

- REQUEST
- RESPONSE
- KEYBYTES

Example:

```
;*************************************************
;********  Simulation file for the DME31  ***********
;*************************************************

[REQUEST] ;Request message to the ECU

empty       = _
id_lesen    = 03,00,00
ram_lesen   = 06,00,01,00,00,00
fs_loeschen = 03,00,05
diagende    = 03,00,06
fs_lesen    = 03,00,07
ack         = 03,00,09
adr_lesen   = 03,00,0B

[RESPONSE] ;Response from the ECU

empty       =
id_lesen    = 0D,00,F6,31,31,31,31,31,31,31,31,31,0D,00,F6,32;,...
ram_lesen   = 04,00,FE,64
fs_loeschen = 03,00,09
diagende    =   ;No response received from the DME
fs_lesen    = 04,00,FC,00,03,00,09 ;No error
ack         = 03,00,09
adr_lesen   = 0F,00,FA,5E,88,FF,FF,FF,FF,FF,FF,FF,FF,E5,0E,0F,00;,...

[KEYBYTES] ;Keybytes of the ECU
keybytes= 01,02,00,00,0D,00,F6,31,31,31,31,31,31,31,31,31,31,03;,...
```

A. References

[1] EDIABAS: BEST/2 Function Reference

[2] EDIABAS: ECU Simulator

[3] EDIABAS: BEST/1 - Language and Interpreter

[4] EDIABAS: User Manual

[5] EDIABAS: BEST/2 - Language Description

[6] EDIABAS: API User Manual