USB - CODEC
configuration editor
version 2.01

Philips Semiconductors

PHILIPS

USER MANUAL

USB - CODEC
configuration editor
version 2.01

Report No.: RNB-C/40/98XI - 1153

Sedat Serper

Consumer Systems Development – Design & Application,
Philips Semiconductors Nijmegen
The Netherlands

Date : December 1st, 1998

Preface

This paper describes the USB - CODEC configuration editor software consisting of a detailed user-manual, which describes the possibilities of this program. This tool is specially created for the UDA1325H and UDA1335H USB - CODEC IC's.

The UDA1325H and UDA1335H are USB compliant audio CODEC's. Both IC's are capable of replacing the whole sound-card in todays (1998-1999) PC systems which have Win98 as OS. This, enabling manufacturers of PC sound systems to integrate the USB - CODEC IC into their own product with all the benefits of a true USB product.

This software package runs only on Windows 95 and 98 ™ based operating systems.

CONTENTS

1 USB – CODEC configuration Editor v2.01

1.1 The software package

The software package consists of the 3 install disks. The first disk is holding the SETUP.EXE program together with datafiles.

Please use the installer, for compatibility with future versions of the *Configuration Editor* and to guarantee proper functioning of the program. Once setup has finished, there should be a directory been created which holds the data and executable file(s) of this program. This directory must contain the following files which setup has installed for you:

COD_201.EXE	=> The configuration editor executable program
EE_1024.EXE	=> DOS program for programming ADX formatted files into I²C-EEPROM.
EE_2048.EXE	=> DOS program for programming ADX formatted files into I²C-EEPROM.
RD_2048.EXE	=> DOS program for reading / downloading data of an I²C-EEPROM.
ST5UNST.LOG	=> Uninstall information for Windows.
REVISI~1.RTF	=> Help file
WELCOME.RTF	=> Help file
SHOWAL~1.RTF	=> Help file
SAVECO~1.RTF	=> Help file
EDITCO~1.RTF	=> Help file
READDE~1.RTF	=> Help file
LOADCO~1.RTF	=> Help file
PROGRA~1.RTF	=> Help file
~~XX~~.ADX	=> Dummy file
~~XX~~.XXX	=> Dummy file
REP~~.REP	=> Dummy file

1.2 Recommondations

A mouse is not strictly neseccary but advisable, since Windows © and programs within Windows © are mostly controlled by mouse.

1.3 Installing and starting the program

Press on the button 

and select the *Run* option.

This will pop-up a new window in which you can browse and select any program to start. In this case, select the executable file : SETUP.EXE from the first floppy-disk. This executable file will install the files to a drive and directory of your choice. Once this program is installed, do NOT rename or change this directory in case of incompatibility problems in the future with other (un)installers or other versions of *Configuration Editor*.

Run the program by selecting *USB-CODEC configuration editor v2.01* from the *Programs* menu after you selected the *Start* button.
You will be prompted to make a choice if you would like to add the shortcut of this program to your desktop. If you want to avoid this menu to appear every time you start-up the editor and also don't want to put the shortcut on your desktop, please select / check the respective check-box in the *Initialization* window.

If you decide to add the shortcut to the desktop you will notice the icon 

most likely at the upper left corner of your desktop. This icon is the shortcut to the executable. Next time you can doubleclick on this icon to start the program.

1.4 The software features

This version of *Configuration Editor* supports the following features:

- 1 file format is supported.
- User-friendly interface.
- 32-bit DLL libraries resulting in faster execution and disk-access.
- Easy address / configuration editing.
- Easy string editing.
- Easy descriptor editing.
- Progress bars are included for time consuming processes.
- Status bar to show the most important settings, like VID and topology selection.
- Pre-viewing of address values in form of a summary.
- Default option included for all four topologies.
- Topic search for easy access of the addresses / registers.
- Help for program - use included.
- Direct programming capability via a *Single Master Controller*.
- Direct reading capability via a *Single Master Controller*.

1.5 Using the program

1.5.1 General

After starting the program the startup screen will appear, as shown below:

Fig. 1

You can start either by directly selecting the *Edit Configuration* button, which loads the defaults configuration or select the *Load Configuration* button to load the supported file format(s) : **.adx* files. Beside default settings and file I/O, this version is equiped with the capability to read / download the contents of a (valid) I²C-EEPROM.
The easiest way to see the purpose of a button, is by simply trying out and see what happens. Nevertheless, to save you time, the following chapters describe briefly what the possibilities are of this editor and how you can load a file or EEPROM contents and edit it.

☞ To see what the purpose of a ICON-button is, you can move your mouse towards the respective button and wait until a *ToolTip* text appears which gives a short description of the function of the button.

1.5.2 Loading a configuration

Select the  button to initiate the window for loading files. You can also use your keyboard by pressing on the TAB key until the dashed marker appears on the button you want en strike the space bar to activate it (for more information about disabled buttons and functions of them, please read chapter 2.2).

Once the *load file* window is on your screen (like illustrated in **Fig. 2**), you can browse freely through the available drives and directories to select any configuration file.

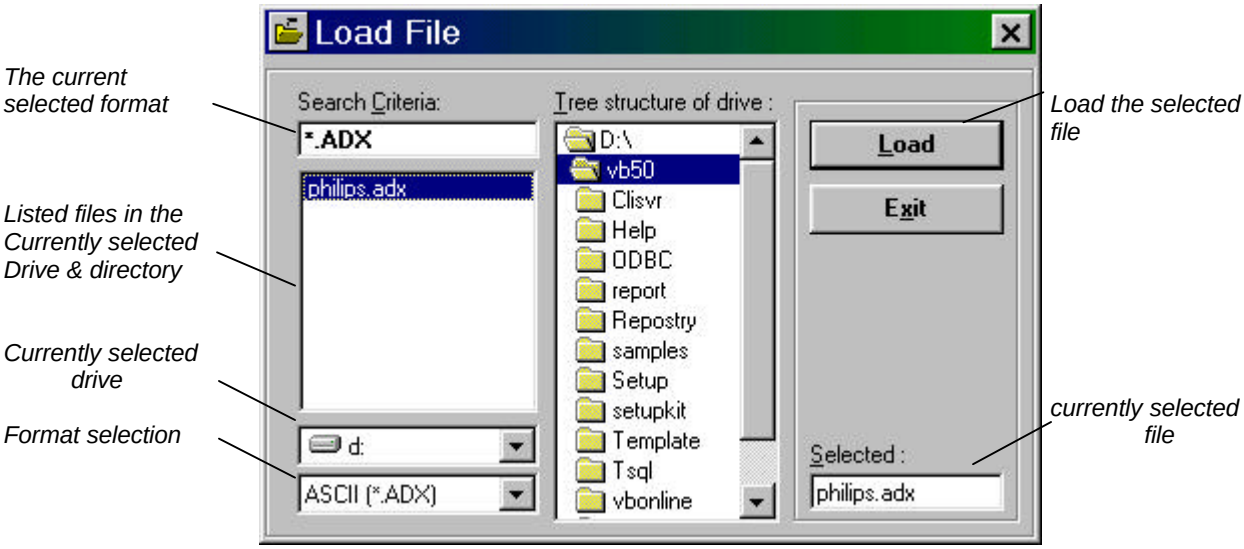

Fig. 2

Depending on the format you have selected, only the respective extensioned files will appear on the listed file sub-window. If you need to look for files with other formats, press on the corresponding spin-button control (see **Fig. 2**) to display the list of available formats and select the one you want.
However, you will discover that there is only one file format supported /selectable. If you finally found your file, select this and press on the *Load* button to load it and decode the contents.

☞ You can also double-click on the respective file to load.

1.5.3 Editing a configuration

It is not neseccary to load a file to be able to start the editing section with the button  . Once this button is pressed and released, the program will automaticly detect whether you have loaded a file, I²C-EEPROM or nothing and will decode and present the data if loaded or set it's default values to *topology 2* for all the addresses and show them instead.

The edit window looks similar as the one illustrated in figure 3 :

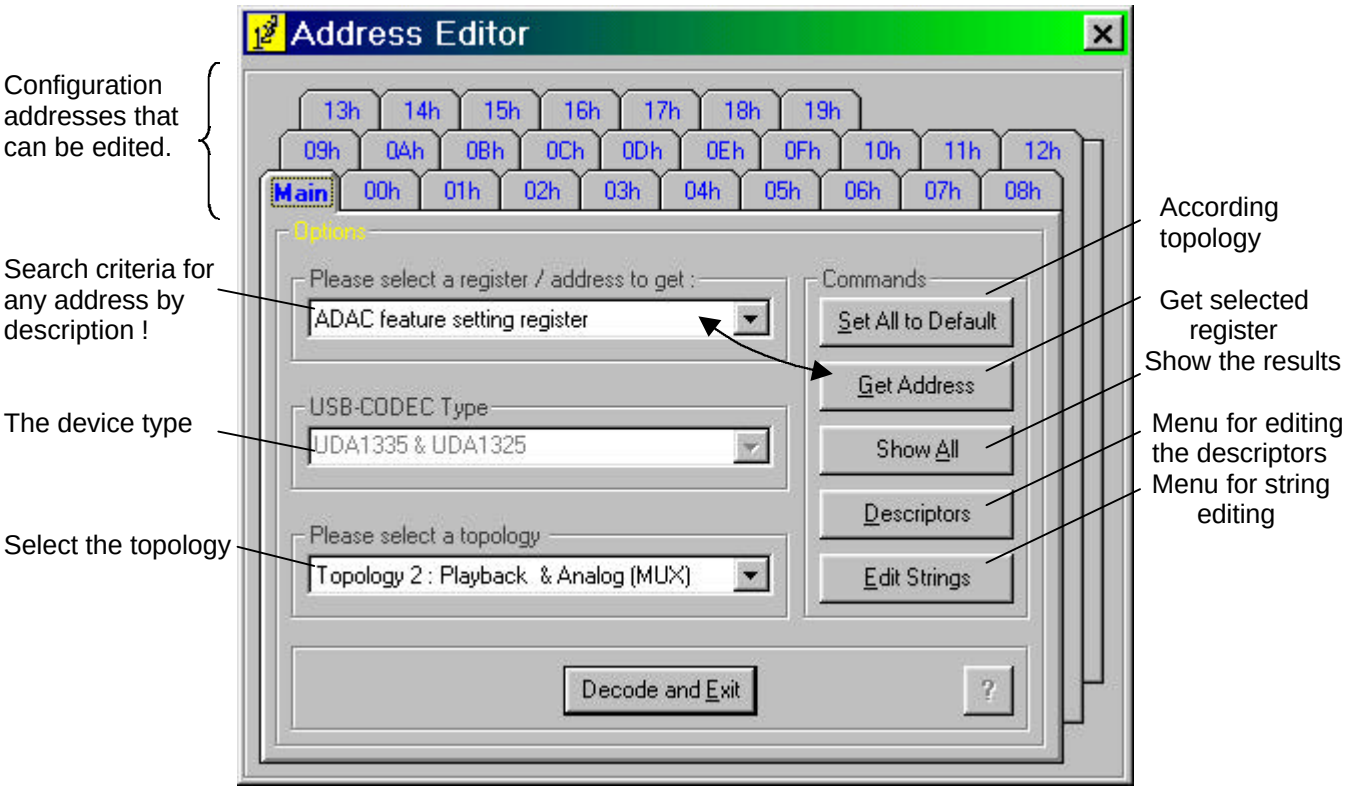

Fig. 3

It is rather difficult to know at which address what kind of settings are available, unless you have the datasheet with you ... Therefore, we have included a search topic by means of the description of the address (select criterea). Once you have selected a register description (in the example above : **ADAC feature setting register**), you can call this TAB field by pressing on the button *Get Address.*

The second combo-input box is to select the USB – CODEC type of device. As you can see, this section is disabled and at this point/ stage only one device type is supported (UDA1325x & UDA1335H). If neseccary, it can easily be expanded in the future.

The strings are put in a seperate window-menu which can be accessed by pressing on the button *Edit Strings*. You will see the four standard strings (Language, Manufacturer, Product and Serial number - strings) and also 27 other free programmable strings.

To edit the editable sections of the descriptors, press on the button *Descriptors*. At this point it is really advisable to get the latest version of the FRS document or one of the Application notes of the UDA1335H. Although this is not really neseccary if you are familiar with the descriptor settings and meanings of them.

Some settings in the first *17h* addresses do have affect on parts of the descriptor field. An example is address *0Fh*. If you decide not to use HID functionality, all HID – related addresses will be removed ! Another example is address *17h* which contains the sample frequency definition. Should you decide to use topology 3 or 4 and also change this address, then you will see that the 6 alternate settings will be modified to the sample frequency value set by you.

If you are finished editing you can either take a brief look at the results by pressing the button *Show All* or directly exit the editing window with *Decode and Exit*. More details will be given in the next chapter.

1.5.4 Saving a configuration

If you want to save your configuration file then select the button  . This will pop-up a new window as shown below:

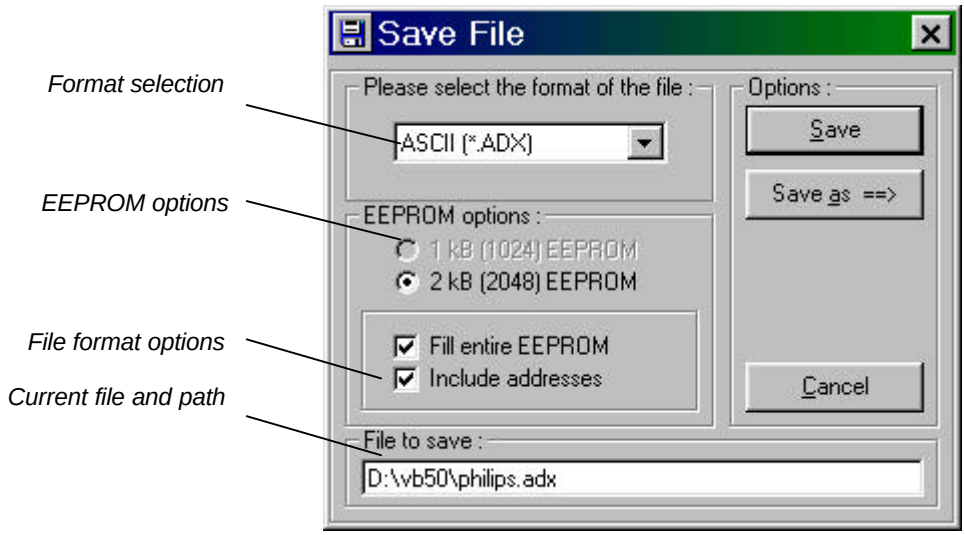

Fig. 4

As we can see, there are beside format selection also 4 other options available. These are specially related to the I²C - EEPROM –size that you will be using for the CODEC. In case your total datasize is greater than 1024 bytes, the option-button *1kB (1024) EEPROM* will be disabled and the program will automaticly select the 2*kB (2048) EEPROM* option.

☞ The descriptor fields / datasize is for every topology a predetermined value. This is not the case for the strings, since the user defines it's own amount of strings and also determines the size of each of those strings. So mainly the strings will determine which size EEPROM you will need for your application.

If you need the file for documentation on paper or any other means where paper-space is critical, you can disable the check-box *Fill entire EEPROM*. If this would be checked, the file would be filled with the value 0 for the unused addresses at the end. This is specially done for I²C programmers which can read **.adx* formats, like **EE_1024.EXE** and **EE_2048.EXE** provided with this package.

Before you save your configuration map with the *Save* command, you may change the format of the destination file if this is neseccary.

If you prefer another name for the file afterall, simply click on the text-box and edit the file name or path to your own wishes. As you can see, the command *Save as* ==> is enabled in this version, and therefore you could also use this option to browse through your system's database to select / find another path and/or filenames.

Furthermore, it is really advisable NOT to alter the format, since this will cause unpredictable results when loaded and decoded by this program. There is not (yet) included any error checking by any means.
But you could change the hexadecimal values of the addresses or corresponding values if needed, this should be no problem.

1.5.5 Scrolling through results

You don't really have to save your configuration into a file to *look* what the results are in hexadecimal format. The command *Show All* in the address editor can do this for you. Once activated you will see a similar window like presented here below:

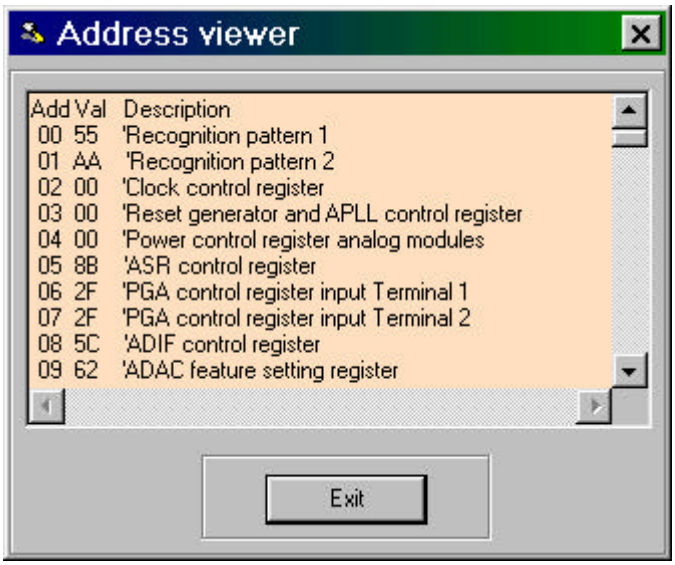

Fig. 5

Each line (except the first one) represents an address with its value and a short description of the address in regard. You can scroll through all the addresses by moving the vertical and horizontal scroll bars in the desired direction.
This format is more or less the same format of **.adx* files listed in the available format box in the Load and Save command windows.

☞ The button  on the main screen (see page 7) has the same function as the *Show All* command in the editor window.

1.5.6 Help command

To complete the program, a small but hopefully effective *help* option is included. In general sence, it gives you the same information as described in this user manual and some extras.

The main window of *Help* looks as illustrated below:

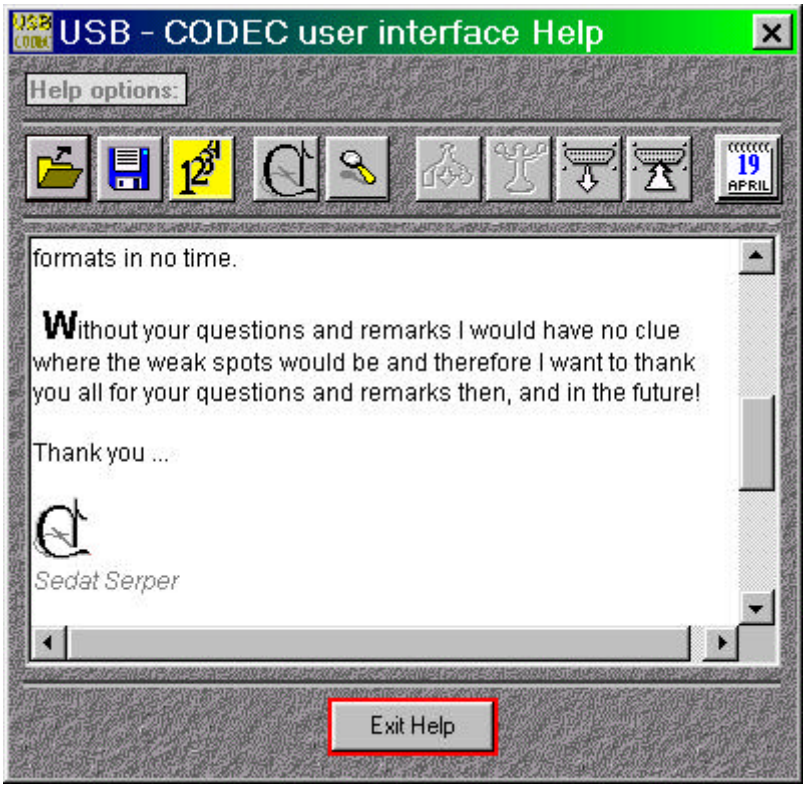

Fig. 6

You have 8 topics at your service. The 8 topic buttons give you each a description of the function when it is pressed in the main menu.
Use the vertical and horizontal scroll bars to browse through the text.

1.5.7 Reading and Writing a configuration into an I²C-EEPROM

This version of the editor is equiped with the capability of *Reading* an I²C-EEPROM. The *Write* capability was already introduced in the previous version of the editor. Both functions need a *Single Master Controller* to be able to setup the I²C -communication. A schematic of this controller and some other information regarding the subject is given in detail in the second application note of the UDA1335H (*DML98012.PDF*). Ask your nearest FAE for this document.

The buttons of the respective functions :

 To read the I²C-EEPROM

 To write an I²C-EEPROM

Reading an I²C-EEPROM has the risc of reading invalid / incorrect data which can cause run-time errors in the editor and causing it to termintate. Therefore, we have introduced some important check's on the data to ensure that the data is valid and editable. Each read-session is therefore finished with a small summary of these checks as shown in figure 7.

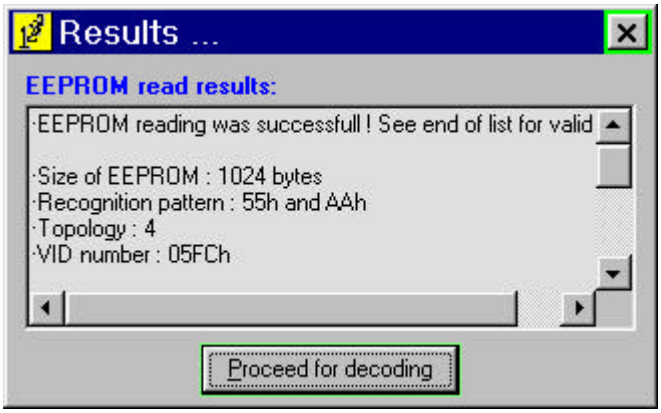

Fig. 7

The check-procedure will be extended end in more detail in the near future.

The read-processing is rather quick (2..3 seconds) in comparison with the write-procedure. Writing of any configuration is time-consuming and can take up to 2 minutes to write a 2k sized EEPROM. This is partly of the method used in the DOS program. It is processing data byte-per-byte and every cycle is programmed for a delay of 10 ms. Some I²C-EEPROM's have a much shorter write-cycle delay, but to ensure that every I²C-EEPROM can be written without worrying about cyclus-timing, we have set this value on default at 10ms.
By the way, there are Windows-based DLL files to setup the I²C-communication via a *Single Master Controller*, but these are still under test. Ultimately, this will replace the DOS-based programs once available …

1.6 Closing the program

To close / exit this program, use the button: 

It is important to know that you will NOT be prompted if you want to save any changes (if they are made) when you select this button. So be carefull.
This will be changed in one of the next versions of this program, a small dialog window will then appear when these sort of situations should occur.

2 Future plans

2.1 File formats

Like mentioned before, only one file-format is included which is supported by this program. It is an ASCII formatted file so that you can read this file in every ASCII-editor and get the information you want from it.
It is strongly recommend **NOT** to alter the format of the file. If the format is changed you can expect unpredictable results, even program termination (NON-FATAL).
There is an error checking included while loading and saving files by this program. Though format errors detection is not yet included.
As long as you change only the hexadecimal values of the corresponding addresses, you can expect normal functionality.

2.2 More features

You probably recognized the disabled buttons on the main screen (see page 7). These buttons will be able to program and read your USB-CODEC in a direct fashion via the USB – bus ! This means that you don't need any I²C programmer at all and you use only one tool to configure your USB – CODEC !!
This version is released with the capability to program (read and write) your I²C EEPROM via the parallel port of your PC in combination with a *Single or Multi Master* I²C-controller. It is a time consuming and non-professional method but will be appreciated to those who have no other tools available …

3 Revision history

3.1 Version 1.00, first release

Since this is the first release, no revisions are made.

If you have suggestions, found bugs or have advises please don't hesitate to contact me.
Email me at : *sedat.serper@nym.sc.philips.com*

Many thanks in regard.

3.2 Version 1.01, second release

- A faulty constant definition caused error when address 0Fh or higher was edited and changed. This caused a loss of a large data-field on *Decode & Exit*. O.K. now.

- When you have decided not to use HID functions, the editor should remove all HID related descriptors and change the total size of the *Configuration Descriptor* field and also change the total interfaces from 4 to 3. This was clearly not implemented in version 1.00. This is fixed now => All the above mentioned actions are taken care of now …

- When the *Show All* command was activated, the commectary text regarding the words **MSB** and **LSB** of the *pointers to descriptors,* were mixed. O.K. now.

- To make access of the strings in the menu *Edit Strings* easier we have provided two extra buttons. The arrow indicates the sequence in which the strings are called on screen.

- The name of the command *Pointers & Strings* have been changed to *Edit Strings.*

- When accessing the *help menu* on some PC configurations, it happend that Windows gave a message that the path was not correct. This is fixed now.

- The Device Descriptor combo box was restricted to the ID numbers of Philips components. We have included some other ID's.

- In order to ephasize the fact that the previously used *.adr files are not interchangable with *USB-DAC configuration editor*, we have decided to use an other extension for the CODEC **:** *.adx. The format is still the same as before (ASCII), only the extension has changed.

- The combo-list in the address editor did not include the D*efault Dynamic Bass Boost* selection. Inserted now, and accessable by the command button *Get Address.*

- One of our customers also requested to have the choice whether the addressess should be included in the *.adx file or not. Therefore, we have included a check-box in the *Save* menu in order to do just so. Check it out.

- The idProduct combo box is replaced by a free configurable text box in which every existing or yet to be created Product ID between 0 and FFFFh can be set.

Thank you all for your input !

3.3 Version 1.02, third release

- The mute function was set to active in all four topologies. We have disabled this function when the default values are loaded and also provided user control for this function in address 09h.

- In address 0Fh the user seemed to have the choice to select a topology, while this combo-box was always disabled. We have replaced the combo-box with a normal text-box to indicate that this is only for information and not for selecting purposes.

- If we look to address 0Fh again, we see in the previous version of the editor that when the I²S – selection was disabled, the user still could select the 4-pins or 6-pins configuration. This is not logical, so we disabled the 4-pins and 6-pins options when the *NO I²S - bus used* is selected.

- The *Time between mute an play* address was settable to 0, which is not supported by the firmware. Lower-level set to 1.

- When the user selects 0 for the *Time between Mute and Standby*, this will autom. tell the state-machine of the firmware that this function is disabled. The user will be reminded to the status when he/she selects 0 by means of a text-label.

- The range of the Default value for DBB is 24 dB not 255, as was implemented in the previous editor.

- Range limitation is also implemented for address 15h, where the Startup default volume setting is located. The maximum settable value is –60 dB.

- The definitions of the output-polarity of the HID outputs in addr. 0Bh were inverse implemented. Corrected bug.

- The title of address 0Fh in the datasheet was changed into *I²S and Topology selection*. To stay tuned we changed the title in the editor too.

- The PID number in the Device Descriptor field was not settable to any desired value, instead there were only a few selectable via a combo-box. To be flexible and after customers request, we changed the control, in which the user can set any value for PID, VID and bcdDevice.

- Address 2C8h had a typo. Instead of the value C0h it had the value CDh which caused the inability to select the stereo / 8 bits recording option.

- When 0 dB was selected for the DBB value, it should change the descriptor bit regarding this function to 0, indicating that this function is not used. This was not implemented in the previous version of this editor. O.K. now.

- Topology 3 had some errors regarding discriptor length and faulty unit reference. Corrected this bug.

- When the user selects topology 3 or 4, this means that the descriptor *wMaxPackageSize* has to be recalculated and filled in the respective interface (=2=recording), regarding the sample-frequency that is set in address 17h.

- Changing the divide factor in address 02h will also cause the *wMaxPackageSize* to be updated in the respective descriptor fields.

- We have finally enabled the *Save as* command in the *Save Config.* menu. Hope that it will be usefull. At the same time we have put extra safety measures for faulty file-names and warnings for overwriting existing files.

- We have also put extra safety measures for situations when the editor is unable to locate the helpfile. This situation occurs when the user has installed the program and runs the editor for the first time without the use of the shortcut.

- In order to provide the most important information at one place, we have also provided a *status bar* at the main-screen of the editor.

3.4 Version 2.00, fourth release

- Some customers requested us to enable the ability to set the internal PLL frequency of the UDA device. Address 03h is the correct address for this to set. We have put the four possible frequency settings, though keep in mind that this value wil be overide by Win98 after of during enumeration.

- The string of the *Start-up Default volume* was not correct in the combo-box for the *Get Adress* functionality. The string "Default" was missing. O.K. now.

- If the user sets the value of *Default Bass Boost value* to zero, it should reset the respective descriptor-bit too. This was partly done, the LSB part was incorrect => it showed 55h instead of 15h permanently. O.K. now.

- The help-menu is conform the main menu's layout.

- For users who have a *single master controller,* we have included two DOS programs together with the total software package, called EE_2048.EXE and EE_1024.EXE. The programs are automatically called when the button *Write Configuration* is activated. Don't be suprised if it takes about 2 minutes to program an EEPROM. I know it is not really professional neither a fast method, but it is only preliminary until the USB-IIC communication is finally implemented. We are currently working on it…

- In the previous version we have implemented the automatic calculation of the *wMaxPackageSize* descriptor. This seemed to cause problems when Topology 1 was selected. The problem is ofcause the fact that this descriptor is only applicable for topology 3 and 4. When the calculation was done in topology 1 or 2, the wrong offset was used causing enumeration errors …
Corrected this bug too.

3.5 Version 2.01, fifth release

- The pointer to the HID descriptor field was not correct. This is corrected in all four topologies.

- Address 08h contains an extra label informing the user of the selected input channel. This will be autom. updated according the selected topology at the *Main* tab-field.

- Added the *read* capability of I²C data from the EEPROM to the menu & icons at the start-up menu.

Thank you all for your continues input !