TECHNICAL NOTE

USB AUDIO PLAYBACK PERIPHERAL (APP)
UDA1331H

EXTENDED
EVALUATION BOARD
AND
APPLICATION NOTE 2
v1.2

DML97002

TECHNICAL NOTE

USB AUDIO PLAYBACK PERIPHERAL (APP) UDA1331H

EXTENDED EVALUATION BOARD AND APPLICATION NOTE 2 v1.2

Author:

Sedat Serper
Philips Semiconductors, Inc.
BLITS, Digital Media Laboratory
Sunnyvale, CA
U.S.A.

Keywords:

Universal Serial Bus (USB)
Digital-Analog Converter (DAC)
UDA1321
UDA1331H
USB-DAC configuration editor
UniCoDes © 1999-2000
Windows™ 98
iMac™

Date: June 14th, 2000

Abstract

This application is an extension to accompany technical note DML97001 that describes the installation and application of the Philips Semiconductors USB-Digital to Analog Converter solution UDA1331H for the Universal Serial Bus (USB).

Whereas document DML97001 focused on basic information about the Universal Serial Bus itself and how to install and use the UDA1331H evaluation board, this document gives lots of IC related details for those customers that want to fully understand and customize the USB-APP UDA1331H for use in their own specific application.

This document does not include the basic information given in DML97001. For information on how to install the Philips Semiconductors USB-APP on your system please refer to the mentioned document.

Notice: *Windows™ 95, Windows ™ 98 and Windows™ NT are registered trademarks of Microsoft ® Corporation. Memphis is the code name used by Microsoft® for the Windows™ 98 development project.*

USB compliant conform USB specifications v1.0

Table of Contents

1. Introduction

1.1 General Information

This application note explains details about the specific and extended features of the Philips Semiconductors USB-APP UDA1331H. It is intended for customers that want to fully understand and customize the USB-APP for use in their own specific application.

This application note will not explain the Universal Serial Bus as such or the installation of the UDA1331H evaluation board. You can find this information in the principal application note *DML97001* of Philips Semiconductors and in the documentation that is described in Chapter 8 “References”.

The application of the USB-APP UDA1331H requires a USB enabled PC running one of the offcial releases of Windows 98.

NOTE: The UDA1331H operates under Windows 98. It can not be used with other (not USB streaming supporting) Windows operating systems. These operating systems do not provide the drivers and system support necessary to operate devices attached to the Universal Serial Bus.

To achieve full system functionality it is necessary that the PC hardware, the PC operating system, the installed driver software and the µController firmware of the chip work flawlessly together.

We strongly advise not to use earlier builds of Win98.

In any case, constant and quick efforts are being made to guarantee that the USB-APP will run without problems with any official Windows 98 releases. The firmware used with the USB-APP is based upon an internal µC80C51 core. It will be flexible/configurable enough to cover as many USB audio features as possible. The current evaluation systems allows the User to

- select between 4 different integrated ROM configurations for their own USB-APP application
- plug in an additional EEPROM that contains a custom configuration.

Legal usage of the Philips USB-AUDIO DACs:

It is very important that you are aware that use of one of the four internal maps is legal if you are the legal owner of the internal map. In any other case, you may **NOT** use any of the internal maps, other than for test purposes ! Otherwise you can and will be prosecuted to the maximum extent possible under law and you can count on severe civil and criminal penalties. So, if you are not a legal owner of one of the internal maps, you must implement your own configuration by means of an external I²C-EEPROM and program your companies VID (& PID) as will be explained in the following chapters. The internal maps are dedicated to either Philips products or customers of Philips who have a legal agreement / contract to use one of the internal maps.

2. System Startup

At Startup/Power-on the USB-APP completes a boot cycle during which all its internal registers are loaded and all pins and settings are configured. There are a total of 5 different configurations that a user can choose from at startup time. Four of these configurations are built into the UDA1331H firmware. They can be used without any change. If only the name descriptor needs to be changed this can be done by changing some system files. This has the advantage that there is no need for an EEPROM (see section 4.2of this application note). If you don't want to do that or you want another configuration, you can use the fifth configuration. At startup this fifth configuration can be loaded using the common I²C Bus protocol from an external 256 byte EEPROM (if present). The USB-APP has a number of General Purpose Input / Output pins that can serve various different purposes. These will be described in detail in chapter 2. For now it is only necessary to notice that two of the GPIO pins (GP0 and GP3) are used to select one of the four internal configurations (in case there is no dedicated I²C EEPROM present).

The basic boot sequence of the USB-APP is shown in Figure 2-1:

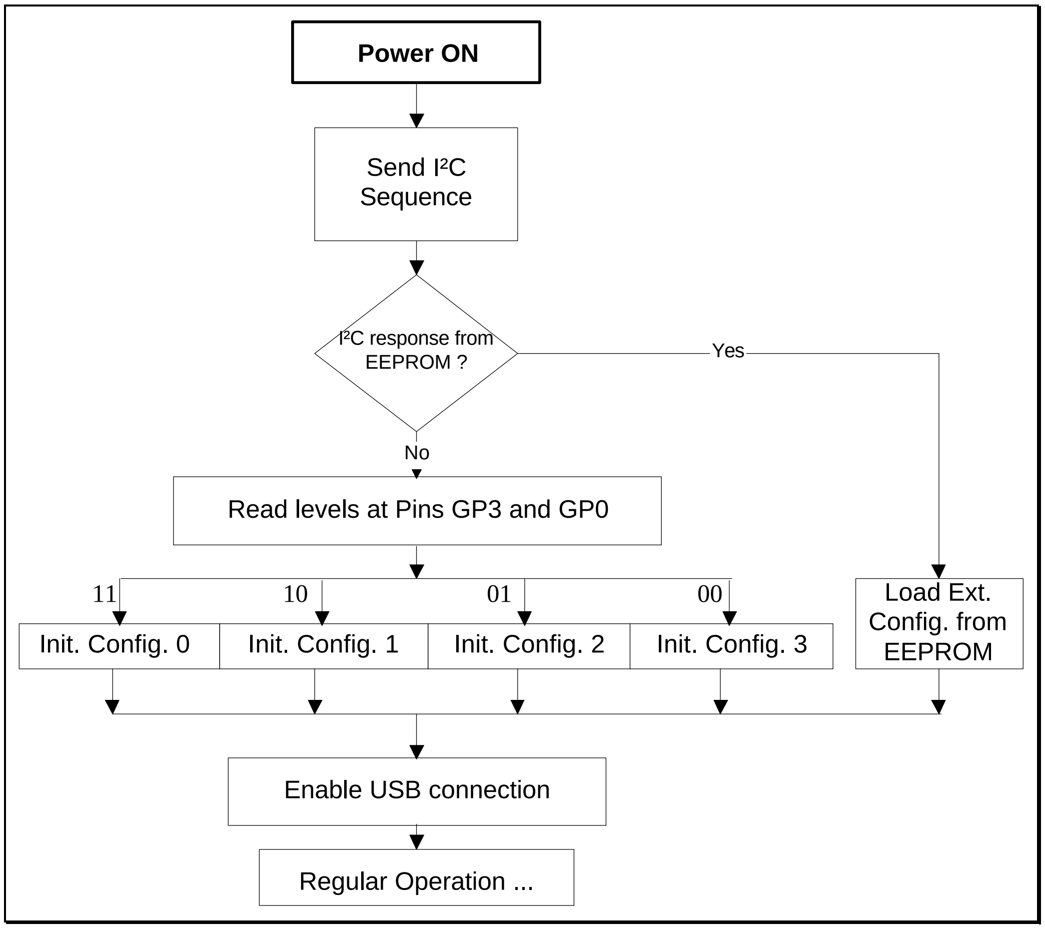


Figure 2-1 Boot Sequence Flow Chart

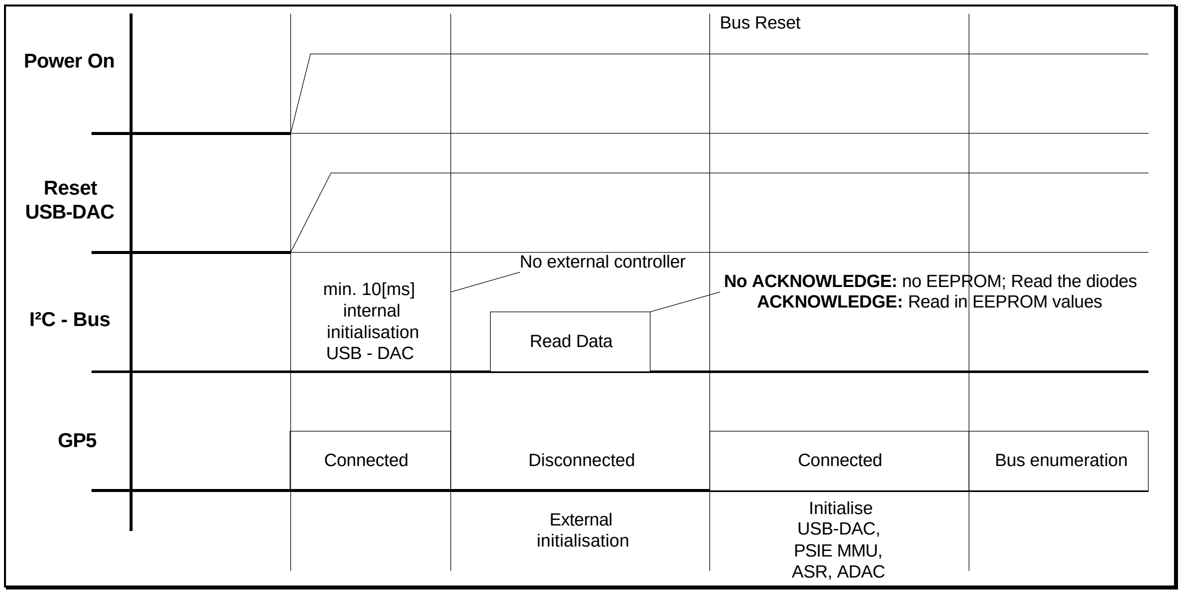

Power On
Reset USB-DAC
I²C - Bus
GP5
Bus Reset
No external controller
min. 10[ms] internal initialisation USB - DAC
Read Data
No ACKNOWLEDGE: no EEPROM; Read the diodes
ACKNOWLEDGE: Read in EEPROM values
Connected
Disconnected
Connected
Bus enumeration
External initialisation
Initialise USB-DAC, PSIE MMU, ASR, ADAC

Figure 2-2 Basic boot-sequence of the USB - APP

The USB-APP can only be used as a a self-powered device as defined in the USB Specification. For test purposes you may also use the bus-power though. The transitions from/to the states ‘power-up’, ‘power-down’ and ‘suspended’ will be handled at the hardware level. After the hardware power-up phase, the firmware initializes the USB-APP components. Initialization is done in a specific order:

To load data from an external I2C - EEPROM, at initialization time, the internal µController will send a read request to I²C slave address $08. It will attempt to read the device’s first two byte locations and compare these bytes with the sequence “$AA $55”. If a match occurs, it assumes that this I²C device is an EEPROM and that it is dedicated to the USB-APP. It will then read the stored configuration data. The exact structure of this I2C - EEPROM configuration data, is given in the datasheet. For more details about this configuration map, please read chapter 3.

The circuitry connected to the General Purpose Input/Output Pins serves a number of different purposes and will be described in more detail later in this chapter. For now (or for the simplest case) it will be assumed that they are simple pull-up/ pull-down resistors that determine the level. A detailed explanation will follow in chapter 2.

2.1 General Purpose Input / Output Pins (GPIO's)

The USB-APP has a total of 6 General Purpose Input / Output (GPIO) pins which can be configured and used for different functions such as I^2S interfacing to an external DSP, Volume Up and Down, Tone Control or Mute.

The GPIO pins of the USB-APP µController can be used for:

- interface to user definable purposes.
- support a 'USB-APP configuration' map selection.
- Disconnect/Connect for bus-powered operation.

There are basically three different configurations for the GPIO pins possible that can be described as:

- No digital I/O communication.
- 4-pin digital I/O communication.
- 6-pin digital I/O communication.

2.1.1 No Digital I/O configurations

This port configuration can be chosen via the configuration map at start-up of the UDA1331H. The following table gives examples for the usage of the different functions for the "No Digital I/O" communication:

Table 2-1 Example : No Digital I/O communication

Pin	Input / Output	Function 1	Function 2
GP5	Output, not programmable (2)	Connect / Disconnect	Connect / Disconnect
GP4	Input, programmable	Alarm mute (3)	HID Input 3
GP3	Input, programmable	HID Input 2	HID Input 2
GP2	Output, programmable	Standby (4)	HID Output 2(6)
GP1	Output, programmable	Mute (5)	HID Output 1(6)
GP0	Input, programmable	HID Input 1	HID Input 1

Notes:
(1) GP0 to GP5 must have a pull-up resistor.
(2) Connect / Disconnect: Holds the USB disconnected as long as the initialization is not finished.
(3) Alarm mute: Input to switch the sound off; especially used if the USB host program does not respond to the control. This button acts directly on the sound and passes the mute to the USB host.
(4) Standby is Switched on (output becomes LOW) if the mute is active for 2 minutes programmable time.
(5) Mute is switched on (output becomes LOW) if the isochronous data flow is interrupted.
(6) For selection between HID/LED application see configuration map BYTE 11 (output is active HIGH).

2.1.1.1 Disconnect/Connect (GP5)

Pin GP5 is used to avoid malfunction during initialization phase, it is therefore important that you implement this function within your own design. While initializing, the Universal Serial Bus remains disconnected and the configuration selection diodes or the contents of the I²C EEPROM are read in. As soon as the µController is ready for regular operation, the Universal Serial Bus gets connected and the configuration selection diodes are switched off. In the time-scale we will see the following happening:

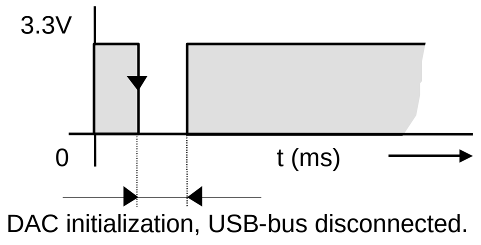


Figure 2-3 Connect/ disconnect phase after power-up.

See also **Figure 2- 2** for better understanding.

2.1.1.2 Possibilities with Function 1 and Function 2

**** Possibilities with Function 1***

It is a must to have a standby circuit or any other way of control to disable the energy consuming parts of your audio device when there is no processing of sound at all. Certainly in battery powered devices it is recommended to implement standby features. To support customers who are interested in this feauture the Philips USB-APP is equiped with a *Mute* and *Standby* output. These outputs can be used to switch other devices (like amplifiers) in their ON or OFF state. Address 17h defines the delay between mute and play in steps of 1 second and address 18h defines the delay between mute and standby in steps of 5 seconds.

Mute & Standby outputs:

Let's look to *function 1, table 2*-1.
GP1 will represent the Mute output and GP2 will represent the Standby output. Address 05h contains the neseccary settings for this. We will have to set address 05h to 00h in order to select *function 1*. The tool *UniCoDes* could be very handy right now, and we advise you to use this tool to win time.
If we set address 17h to 05h (equals 5 seconds delay = **T1**) and address 18h to 02h (equals 10 seconds = **T2**), we will see the following happen:

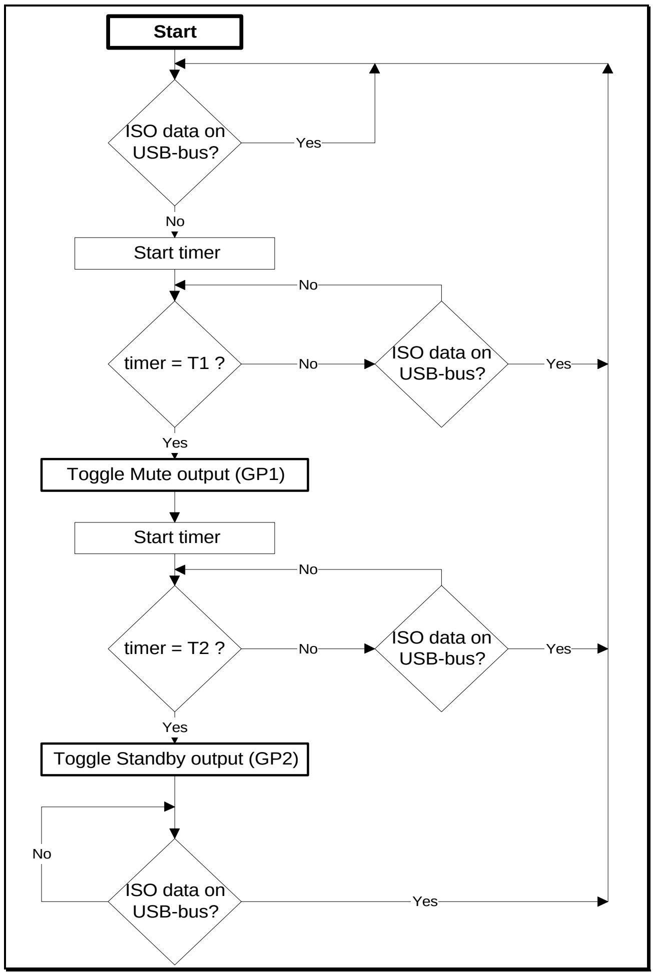


Figure 2-4 Flow-chart of the Standby and Mute functions

Alarm Mute & HID inputs:

In the same configuration you can control the *Alarm Mute* with general purpose pin GP4. GP0 and GP3 are programmable and can be used as a Human Interface Device (HID) input (1 and 2). Depending on the definitions of the addresses 07h and 0Bh you can define the inputs according the **HID usage table**. This document can be downloaded from an internet site, please read chapter 5 for more details. *UniCoDes* has these settings as a default ready for you to load and implement it into the I²C EEPROM.

You will see from the **HID usage table** that GP0 and GP3 are defined by default as respectively *Volume Increment* and *Volume Decrement*.

To test these three input functions, please read chapter 7 under the topic :

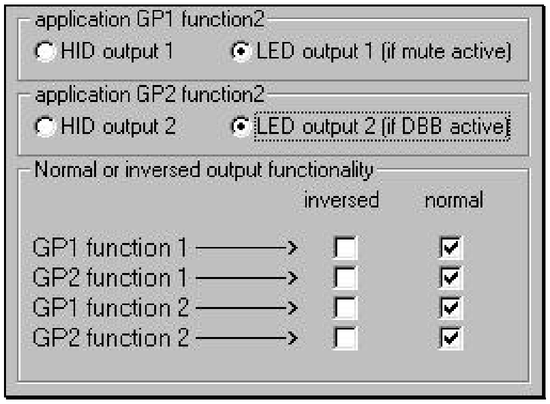
How can I setup the USB-APP to test Volume control and (Alarm) Mute?

** Possibilities with Function 2*

HID/LED outputs:

As can be seen in table 2-1, we have two outputs available. We can select between two possible functions for the outputs. One is as an **HID output** the other is as a **LED output**.

* Let us take a look how we can set the outputs to **LED outputs**. To accomplish this we must programm address 05h, 11h and 19h. To do this, load the default values in *UniCoDes* and select the illustrated options here below and on the next page: (read chapter 3.1 for details about address 19h).

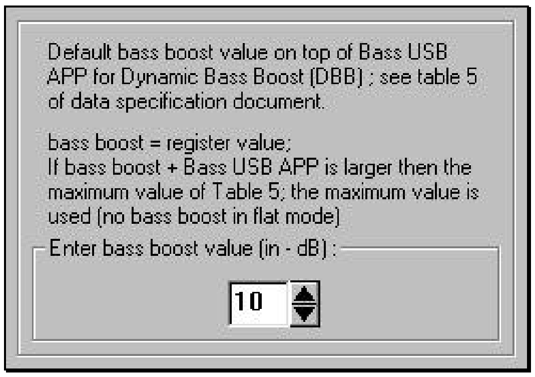


application GP1 function2: HID output 1 [✓ LED output 1 (if mute active)]

application GP2 function2: HID output 2 [✓ LED output 2 (if DBB active)]

Normal or inversed output functionality

	inversed	normal
GP1 function 1 ——>	☐	☑
GP2 function 1 ——>	☐	☑
GP1 function 2 ——>	☐	☑
GP2 function 2 ——>	☐	☑

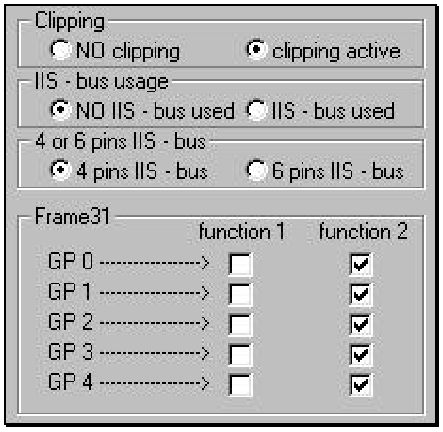


Default bass boost value on top of Bass USB APP for Dynamic Bass Boost (DBB) ; see table 5 of data specification document.

bass boost = register value;
If bass boost + Bass USB APP is larger then the maximum value of Table 5; the maximum value is used (no bass boost in flat mode)

Enter bass boost value (in - dB) : 10

Figure 2-5 Address 11h settings **Figure 2-6** Address 19h settings


Clipping
NO clipping [✓ clipping active]

IIS - bus usage
[✓ NO IIS - bus used] IIS - bus used

4 or 6 pins IIS - bus
[✓ 4 pins IIS - bus] 6 pins IIS - bus

Frame31

	function 1	function 2
GP 0 ----------------->	☐	✓
GP 1 ----------------->	☐	✓
GP 2 ----------------->	☐	✓
GP 3 ----------------->	☐	✓
GP 4 ----------------->	☐	✓

Figure 2-7 Address 05h settings

As you probably recognized, the figures 2-5, 2-6 and 2-7 are snap-shots of *UniCoDes*. Select the same settings and look which hexadecimal value has resulted with the command “Show All”. These values have to be programmed in the I²C EEPROM and after that remove and insert the USB - cable once. This way the UDA1331H will read the newly programmed values of the I²C EEPROM. After you have programmed the I²C EEPROM, you can apply two LEDs with each a resistor of 1[kΩ] in serie to the output pins GP1 and GP2 to test these functions, or use a Volt-meter to measure the outputs.

Start the *Speaker Control* of Win98 and toggle the *Mute all* check-box of the *Speaker* section. This should simultaneously toggle the corresponding LED = *GP1 output*. Now select the *Advanced options* in the menu *File*. You will see a check-box named as *Bass Boost.* Obviously, you can control the bass boost with this box. Check the state ot the bass boost with the second LED = *GP2 output*.

* Now that we have tested one function of the GP1 and GP2 outputs, we will take a quick look at the second function of these pins : programmed as **HID outputs**.

Figure 2-5 shows two selectable options for HID outputs. You can select these outputs and look at the hexadecimal result to program it into your I²C EEPROM. Depending on the definitions of the addresses 12h and 13h you can define the outputs according the **HID usage table**. This document, which describes the possible values with it’s corresponding function, can be downloaded from an internet site. Please read chapter 5 for more details about this internet site.

<u>HID inputs:</u>

* The remaining parts are the *HID inputs* GP0, GP3 and GP4. You can freely program these inputs according the previously mentioned **HID usage table** into the respective addresses 07h, 0Bh and 0Fh. The programming of these inputs is on the same way as described for *HID inputs* with **Function 1.**

2.1.2 Four-pins I/O configurations

We will discuss the table below in this chapter:

Table 2-2 Example : Four pin Digital I/O communication

Pin	Input / Output	Function 1	Function 2
GP5	Output, not programmable (2)	Connect / Disconnect	Connect / Disconnect
GP4	Digital I/O	BCK Output	BCK Output
GP3	Digital I/O	WS Output	WS Output
GP2	Digital I/O	DATA Output	DATA Output
GP1	Digital I/O	DATA Input	DATA Input
GP0	Input, programmable	HID Input 1	Alarm Mute (3)

Notes:
(1) Connect / Disconnect: Holds the USB disconnected as long as the initialization is not finished.
(2) Alarm mute: Input to switch the sound off; especially used if the USB host program does not respond to the control. This button acts directly on the sound and passes the mute to the USB host.

2.1.2.1 Possibilities with Function 1 and Function 2

**** Possibilities with Function 1***

As we can see in the table above, we have the same Disconnect/Connect function in GP5. We also can find similarity in GP0 concerning HID input 1. Therefore, see the previous chapter which discusses the GP0 and GP5 functionality.

The main difference is the fact that the Data stream in I²S format is available to you by means of GP2, GP3 and GP4. There is also an I²S input available (GP1). The I²S ouput can be wired to a DSP which can manipulate the data and return it back to the I²S input of the UDA1331H (see figure 2-8).

**** Possibilities with Function 2***

The only difference with function 1 is that GP0 is predefined as an *Alarm Mute* input. Toggling this input will toggle the Mute check-box in Windows and will finally toggle the sound at your speakers.

2.1.3 Six-pins I/O configurations

We will discuss the table below in this chapter:

Table 2-3 Example : 6-pin Digital I/O communication

Pin	Input / Output	Function 1
GP5	Digital I/O	WS Input
GP4	Digital I/O	BCK Output
GP3	Digital I/O	WS Output
GP2	Digital I/O	DATA Output
GP1	Digital I/O	DATA Input
GP0	Digital I/O	BCK Input

As we can see in the table above, we have no Disconnect/Connect function in GP5. Therefore it is important that the USB-APP has finished it's initialization before it is connected to the USB-bus. Please read chapter 7 how to implement 6-pins Digital I/O.

The main difference with the previous I/O configurations, is the fact that the I²S output is totally seperated from the I²S input. This also opens the possibility to couple a DSP unit between these two general purpose pins. This way you can manupilate the I²S data and send it back to the UDA1331H like illustrated in figure 2-8:

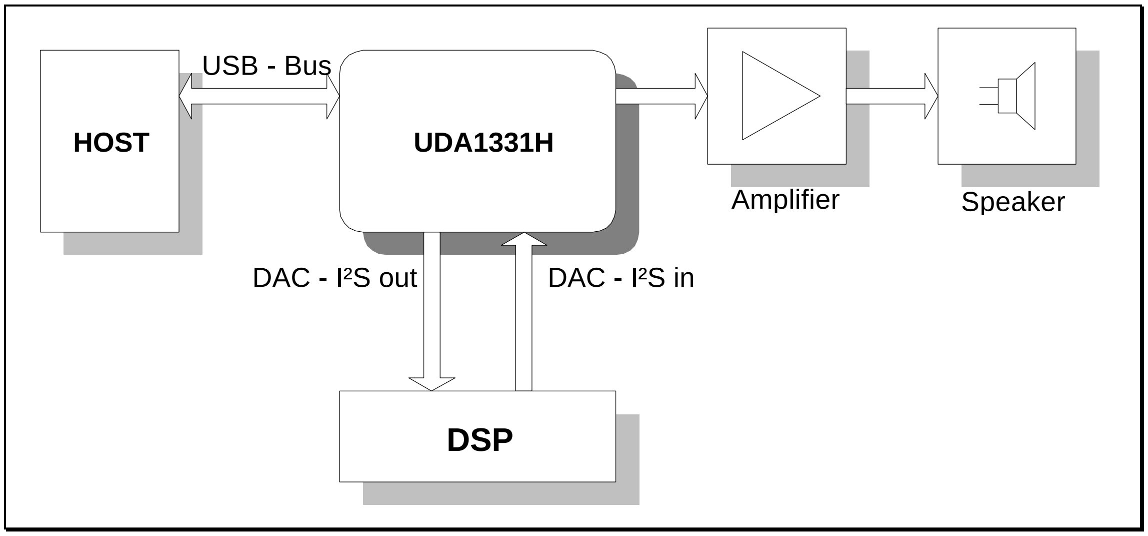


Figure 2-8 Example for purpose of the 4-pins & 6-pins I/O configuration.

2.1.4 Diode Selection

The schematic here below gives the diode delections for selecting a configuration in combination with using two HID inputs:

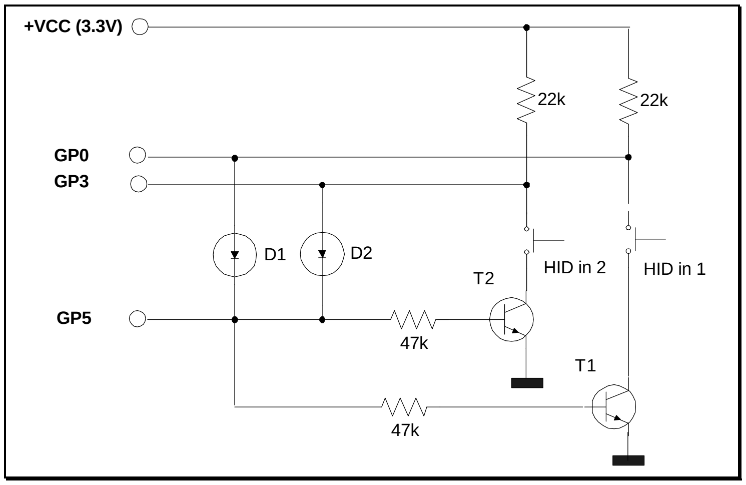


Figure 2-9 Diode Selections for selecting Configuration in combination with using 2 HID inputs

At initialization phase (GP5 low), both transistors are not conducting and it is not possible to read in a wrong diode value by (accidently or by purpose) pressing one or two of the external function (HID) buttons. After initialization the transistors are conducting. Including an optional additional diode requires another additional transistor:

0 diodes	= no transistors,
1 diode	= 1 transistor
2 diodes	= 2 transistors.

Configuration Number	Diode #	Comment	level during initialization	
			GP0	GP3
0	--	No diodes or transistors needed	high	high
1	1	1 transistor needed	low	high
2	2	1 transistor needed	high	low
3	1 and 2	2 transistors needed	low	low

Table 2-5 Diode Selections

Note:

If no HID is used in the specific option or a wrong selection doesn’t matter, the transistors are not needed.

In the general USB concept the ‘USB-Host’ needs to be notified of all activity in it’s USB topology. This includes the use of the GPIO pins. For this purpose the USB HID specifications were defined. The HID Entities describe to the ‘Host’ the type of inputs/outputs that can be controlled.

There are two HID concepts /phases defined in the HID entity description. Phase one defines that ‘all’ control parameter changes need to be sent as requests to the Host first. Upon reception it will take action (phase two) and send a control change request command when e.g. a button was pushed.

Accordingly the action flow after pressing a button is:

1. the Device µController reports to the Host that a push button has been pressed and what function it represents,

2. the Host will initiate the correct request via the correct Class (Audio Device Class or HID), according to the pressed push button.

The Device µController will translate this request into a physical action by e.g. controlling the Volume level.

2.2 Additional External Circuitry

2.2.1 Oscillator Circuit

The USB-APP runs at 48MHz. An external signal can be applied to the XTAL input (pin 37). If the USB-APP is used with a crystal, the circuit described in figure 2-10 should be used.

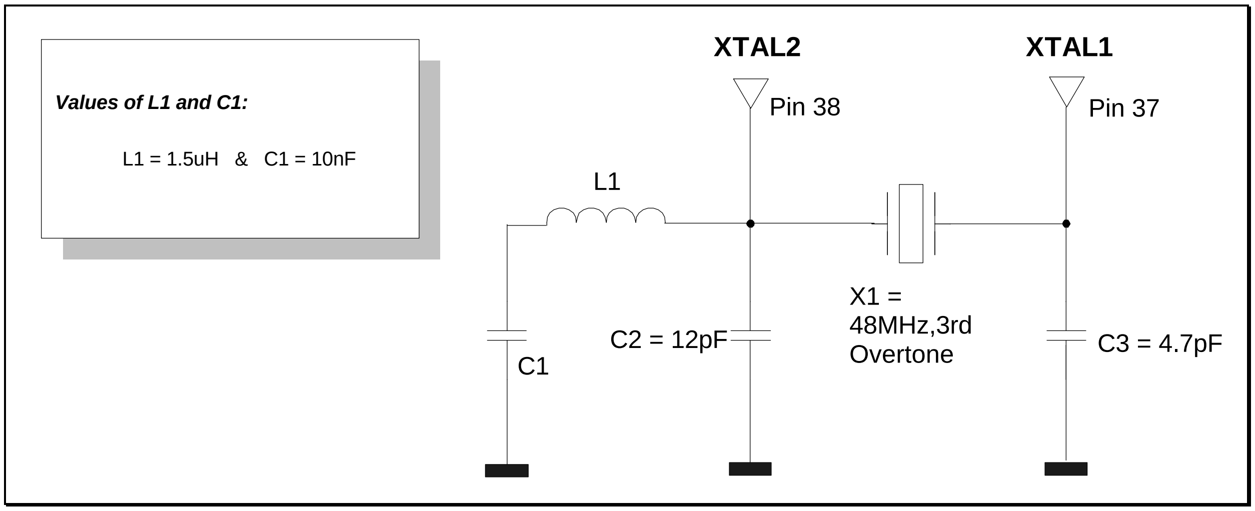


Figure 2-10 Crystal circuitry with 48MHz filtering components

The crystal specifications (summarized):

* Frequency stability in a temperature range of :
    - 55 Deg. Celcius to 105 Deg. Celcius => 50ppm typ.
    - 10 Deg. Celcius to 60 Deg. Celcius => 10ppm typ.
    - Resonance impedance = 40..45 Ohm
    - Static capacitance (between electrodes) = 5pF typ.
    - Load capacitance = 8pF typ.

* Third overtone crystal.

It is important that the capacitors C2 and C3 have the value as mentioned in figure 2-10. These values are beside defining crystal frequency output also important for the UDA1331H internal hardware. There are no special requirements regarding to the deviation of these capacitors and therefore standard cap's can be used (maximum of 25% spread).

There are no special requirements for the inductor.

2.2.2 Coils at the input of the USB

To meet the EMC compatibility rules,standardized by the ISO orginization, it could be neseccary to use the coils in combination with the capacitors as shown in the schematic at the input of the USB. Though, removing these components should not affect the functionality of the UDA1331H in any way. It will depend on your PCB design whether this EMC counter measure is needed or not.
If you plan to use this circuitry it is important that you use a 4-in one package coil and not 4 physically separated coils. This is to meet the specifications given in the datasheet. The isolators in the 4-in one packaged coil are designed to minimize the cross-over signals between the coils which is hard to accomplish if you use 4 separate coils (depending on the placement of these coils).

2.2.3 Coils at the Vdd lines of the UDA331H

If we look at the application schematic we see different coils in combination with capacitors at the Vdd lines of the UDA1331H. This is done to minimize distortion and noise at the Vdd lines.Without using these components, the USB-APP will function correctly but we do not guarantee the high performance specifications of the UDA1331H.

Note:
These extra decoupling components in combination with your specific application will help to get an EMC compatible design.

3. Configuration Data Structure

This chapter describes the most interesting features of the USB-APP in combination with the possibilities of Win98.

3.1 Working with Dynamic Bass Boost (DBB)

As you can read from the datasheet, address 19h is the register which contains the value for the DBB value.
If this value is equal to 00h = 0 [dB], bass boost will not be reported to the USB host by the device. You will not see any DBB control box if this is the case. Should you enter any value not equal to 00h, the bass boost will be reported to the host and you will see a check-box in the *Speaker Control* of Win98.
Without any detection you could create clipping of the output signal. The USB-APP can detect this sort of situations and reduce the bass to prevent clipping. Address 05h includes this clipping prevention mode if this is desired. This way we create a *dynamic bass boost* control.

Important note:

If you activate the *bass boost* and your *volume slide-bar* is at it's maximum in combination with *clipping prevention mode* ON, then this will result in no audible effect when the slide-bar of the bass is increased. To be able to hear the effect of the bass you must regulate the *volume* to a medium level.

3.2 Working with Default Volume

It is possible that you desire a specific volume at startup of the USB-APP. This is an option which the USB-APP does support by means of address 1Ah. The value in this address will be loaded the first time Windows starts up and this wil be applied to the volume slide-bar in the *Speaker Control* menu.
Every time you re-plug the USB-APP application board, the value in address 1Ah will be ignored and overruled by the most recent value of Windows (set by the user earlier).

3.3 Product ID (idProduct) and Vendor ID (idVendor)

The Philips USB-APP provides a way to customers to define their specific Vendor ID and Product ID if this is desired. In that case, the customer has to contact *Microsoft* to make an official entry in the product and drivers listing of Microsoft. The result will be that the values entered in address 1Bh, 1Ch (regarding VID) and 1Dh, 1Eh (regarding PID) will be read and recognized by Win98 the first time that the USB-APP is connected to the USB-host & Win98. This will activate Win98 to set the defined VID and PID in the systems registers.

4. Customizing your design

4.1 Handling Device Configurations

As has been explained in the previous chapter, there are in principle 5 different configurations that can be used to configure the UDA1331H for use within a USB Audio System:

- 4 internal configurations that can be selected via external circuitry
- 1 external configuration in an EEPROM that is downloaded via I²C protocol

The external configuration basically leaves all possibilities open for specific settings but it involves the use of a small external EEPROM of which the contents can be generated with *UniCoDes* (see chapter 4.2).

4.2 Configuring the USB-APP

Philips Semiconductors provides a program called the Universal Configuration Designer, shortened to *UniCoDes*, that greatly simplifies the generation & editing of a configuration data set. This tool is available on the USB-AUDIO website. See chapter 5, USB Contact Addresses for URL's.

4.3 The USB-I²C example driver

This driver allows the control of any additional I²C devices through the USB path.
A software driver is available from Philips Semiconductors (currently only for Windows 98, Windows NT5 under development) that offers a simple API for software packages that need to setup or control these additional devices using the industry standard I²C bus.

5. USB Contact addresses

A lot of the above and additional useful information can further be found on the following Internet web site that is fully dedicated to USB support and information:

http://www.usb.org

You can e.g. request a new manufacturer's ID to use in your USB product, you can find recent news and you can of course download latest USB specifications.

Information about other Philips USB-AUDIO products can be found at :

http://www.semiconductors.philips.com/usb/products/audio

and

http://www.usbaudio.ce.philips.com

For information about I²C definitions/documentation see:

http://www.semiconductors.philips.com/i2c/facts/

For information about I²S definitions see datasheet page 27.

For information on EMI vs. PCB design issues see:

http://developer.intel.com/design/USB/papers/emi_apps.pdf

6. Glossary

The following explains a number of terms that are used in this document:

Term	Explanation
ADAC	Asynchronous Digital Audio Converter
ASR	Audio Sample Rate Redistributor
DAC	Digital to Analog Converter
Descriptor	Set of registers (hard- or software) that describes the device capabilities to the USB host.
Device Class	Class of Devices that share certain features e.g. Audio Device Class, Human Interface Device Class …)
EEPROM	Electrically Erasable Programmable Read Only Memory
Endpoint	Actual data sink/source in a device. A device can have several endpoints.
GPIO	General Purpose Input / Output
Hub	A USB device that has one upstream connector (towards the Host) and a number of downstream ports (usually between 4 and 7). A Hub Device enhances the number of USB devices (including other hubs) that can be connected to the USB bus. It does not add any further functionality to the bus.
MMU	Memory Management Unit
Port	USB connection on a Hub Device, enabling one USB Device (possibly another Hub) to be connected.
PSIE	Philips Serial Interface Engine
UDAO	Universal Digital Audio Output
USB	Universal Serial Bus, max. speed is 12 Mbit/sec
Windows™ 98	Microsoft ® Operating System, upgrade to Windows™ 95, scheduled for release in Q2/1998, beta version have been available to developers since about 7/1997

Table 6-1 Glossary listing

7. FREQUENTLY ASKED QUESTIONS

* In case of Problems

Should you still experience problems in defining the most appropriate application for the Philips Semiconductors UDA1331H demonstration board, we recommend that you get into contact with your local Philips Semiconductors Field Applications Engineer. The local Philips FAE will be able to help you or will get you into contact with a USB expert within Philips Semiconductor.

* Frequently asked questions listing

We have provided a listing of frequently asked questions how to setup a specific configuration or solve a solution. Some of them might be interesting for you and maybe could answer a question you had in mind. The listing here below gives a summarize of the questions which will be answered in the folllowing pages:

Page	Question
24	**How can I setup a 6-pins I²S configuration ?**
25	**How can I setup the USB-APP to test Volume control and (Alarm) Mute?**
26	**How can I select an internal configuration map?**

Table 7-1 *Frequently asked questions* summarize.

*** Question** **: How can I setup a 6-pins I²S configuration ?**

*** Answer/solution** **:**

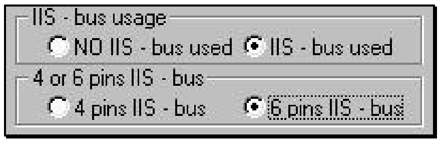
Follow the steps below to setup the UDA1331H QFP64 PCB for 6-pins I²S configuration.

Step 1. If *UniCoDes* is installed you can directly continue with Step 2.
Otherwise, please download the software tool from the USB-AUDIO webpage and install it on your Win98 PC.

Step 2. From the File-menu, select the command "New Design" and select the USB-DAC /N101 release.

Step 3. From the File-menu, select the command "Edit via Palett". This will pop-up the configuration editor.

Step 4. Select address 05h and activate the following items:



IIS - bus usage
NO IIS - bus used [✓ IIS - bus used]

4 or 6 pins IIS - bus
4 pins IIS - bus [✓ 6 pins IIS - bus]

Step 5. Go to the Design-menu and select "Device Specific Configuration".You will see that address 05h has the value E0h.

Step 6. You may decide to save your design into a file before continuing.

Step 7. Program the hexadecimal values to your I²C EEPROM on the DAC board with your personel I²C programmer or use the embedded programmer within UniCoDes. See the Help-menu for additional hartdware that you need to use the embedded programmers.

Step 8. Unplug the USB-cable from the DAC - board (In case you have a USB-bus-powered board).

Step 9. Plug-in the USB cable and wait until enumeration has finished. Then you must ***first*** remove jumper J7 and **then** connect the *DO to DI, WSO to WSI and finally BCKO to BCKI* (we recommend you to use a flat-cable for this). J7 disables the *connect/ disconnect* circuit, because this is not usable when the USB-APP is programmed for 6-pins I²S I/O (see data-sheet and/or table 2-2).

Step 10. Play a WAF-file to check your setup. You should hear the sound you are playing.

*** Question : How can I setup the USB-APP to test Volume control and Alarm Mute?**

*** Answer/solution :**

Step 1. If *UniCoDes* is installed you can directly continue with Step 2.
Otherwise, please download the software tool from the USB-AUDIO webpage and install it on your Win98 PC.

Step 2. From the File-menu, select the command “New Design” and select the USB-DAC /N101 release.

Step 3. From the File-menu, select the command “Edit via Palett”. This will pop-up the configuration editor.

Step 4. Now that the program has loaded the neseccary items you probably want to know which hexadecimal value you have to program in your I²C EEPROM.

From the Design-menu select “Browse through map”.

You see a summarize of all the addresses with it’s corresponding value. Browse through it to read & double-check the values of the corresponding addresses.

Step 5. Now that you have the neseccary values you can program your I²C EEPROM.

Step 6. To save your setting, see the File-menu. You have the choice to save the file in *.bin format if needed.

Step 7. Once you have programmed your I²C EEPROM, you can continue by checking if the board is provided with pull-up resistors at the GP0, GP3 and GP4 pins. If not present, please solder a resistor of 22k to the corresponding pin(s).

Step 8. Unplug the USB-cable and plug-in again to reset the USB-APP and reload the newly programmed external configuration. Play a WAF-file of your choice in a loop-mode (repeat).

Step 9. While the sound is playing, get an electrical wire which is connected to ground (0V) of the PCB on one end. The other end of the wire will act as a toggle control.

Step 10. If this wire is connected shortly to GP4, it will toggle the *mute* status.
If this wire is connected shortly to GP3, the *Volume Down* control will be activated.
And finally, if this wire is connected shortly to GP0, the *Volume Up* control will be activated.
Start the *Speaker Control* of Win98 to pop-up the menu for the volume control bars. This will visually confirm your control actions.

*** Question : How can I select an internal configuration map?**

*** Answer/solution :**

Step 1. MAKE SURE YOU ARE UPDATED ON THE LEGAL USAGE OF THE INTERNAL MAPS. PLEASE READ CHAPTER *INTRODUCTION* !

Step 2. If you plug-in your USB-cable to the USB-APP board, by default, the configuration inputs GP0 and GP3 are both on a high level (pull-up's). The corresponding internal map will be loaded if there is no external I²C EEPROM inserted with it's first two byte addresses to AAh and 55h.

Step 3. To select any other internal map, setup a circuit like illustrated in figure 2-9.

Step 4. Connecting a diode to one of the two general purpose pins, GP0 and/or GP3, will select a low level on start-up. See data-sheet or the document SW2117.PDF for the contents of the internal map that corresponds with the selected digital levels (HIGH or LOW).

8. References

More information on different USB-(AUDIO) topics can be found in the following publications:

[1] UDA1331H datasheet

[2] UDA1321 data sheet

[3] USB-Specification version 1.0 November 1995

[4] USB Device Class Definition for Audio Devices

[5] USB Human interface Device Class specification

[6] USB Human Interface Device Class Usage Table

[7] The following documents are close related to the USB-DAC UDA1331H device:

Title	Description
SW2117.pdf	Internal map descriptions of the UDA1321x / N101 devices.
DML97001.pdf	General application note for using the UDA1331H USB-APP.
DML97002.pdf	More detailed application note for programming the UDA1331H USB-APP.
DML98001.pdf	General application note for using the UDA1321x /N101 USB-DAC.
DML98002.pdf	Detailed application note for programming the UDA1321x /N101 USB-DAC.

See also chapter 5, “USB Contact Addresses” for USB-AUDIO related information sources !