# AUDIOBOOK

## VS1010

**All information in this document is provided as-is without warranty. Features are subject to change without notice.**

**Revision History**

| Rev. | Date | Author | Description |
|---|---|---|---|
| 1.10 | 2025-10-21 | HH | Added UART terminals, typo and link corrections. |
| 1.00 | 2023-05-25 | MH | Initial public release. |
| 0.4 | 2023-05-22 | MH | Minor fixes and additions. |
| 0.3 | 2023-05-19 | MH | Updated specification. |
| 0.2 | 2023-04-13 | MH | Features updated. |
| 0.1 | 2023-02-10 | MH | Specification updated. |
| 0.01 | 2022-10-06 | HH | Initial pre-release. |

# Contents

# List of Figures

# 1 Introduction

This is the documentation for VLSI Solution's VS1010 Audiobook project.

The VS1010 Audiobook is a hardware/software project, that implements an audiobook player for VS1010 platform, which is suitable for:

- Talking Bibles as well as other religious texts.
- General audiobooks, e.g. novels.
- Language learning tools.

As an example, an excerpt from The Bible, which was used during this project, is provided along this package. This project is meant to be a demonstration, and it does not contain the whole Bible. This project's main purpose is to be a proof of concept, showing that this kind of implementation is possible. The parts of the Bible, that were used during this project were:

```
The Bible
-Old Testament
--Genesis
---1: The Beginning
----1-31
---2: Adam and Eve
----1-25
--Exodus
---1: The Isrealitis Oppressed
----1-22
-New Testament
--Matthew
---1: The Genealogy of Jesus the Messiah
----1-25
```

All of these parts are from public domain, and they are available to download from *https://www.wordproject.org/bibles/kj/index.htm* (King James Version (KJV)).

The following components are provided for the demonstration:

- VS1010 Audio Book PCB, including a microSdSd card with software, including source code..
- USB->UART cable to connected between DevBoard and your PC.
- USB-A <-> USB-C power cable.
- Use your favorite UART Terminal Emulation program installed on the PC. See in Chapter **??**, *Terminal Program Settings*) for details.

# 2 Terminal Emulation Programs and Their Settings

This chapter shows recommended settings for two example terminal programs, tested by VLSI. Other terminal programs can also be used, provided that the crucial parameters are set in a similar way.

## 2.1 Setting Up and Using Microcom / Linux

Install Microcom terminal emulator:

```
% sudo apt-get install microcom
```

Then run it at 115200 bps, 8N1:

```
% ls -la /dev/ttyUS*
crw-rw-rw- 1 root users 188, 0 Oct  3 10:11 /dev/ttyUSB0
crw-rw-rw- 1 root users 188, 1 Sep  4 16:03 /dev/ttyUSB1
crw-rw-rw- 1 root users 188, 2 Sep  4 16:04 /dev/ttyUSB2
crw-rw-rw- 1 root users 188, 3 Oct 20 15:25 /dev/ttyUSB3
% microcom -s 115200 -p /dev/ttyUSB3
```

Note:
Microcom's default speed is 115200 bps, so you may leave the `-s 115200` part out.

If you get the following message

```
Exitcode 2 - cannot open device /dev/ttyUSB0
```

it usually means that you are not in a group allowed to access the serial port. This group is usually called dialout. Verify this by running:

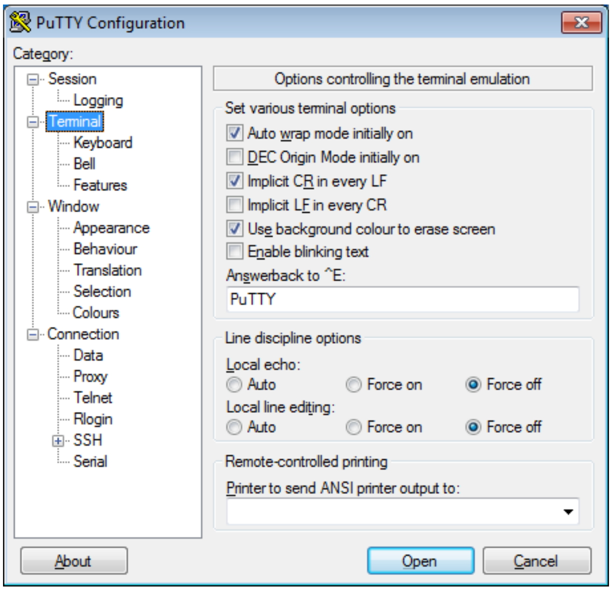
```
groups
```

Now, you can either run microcom as root with sudo, or, preferably, add yourself to the required dialout group (or whatever gives you access in your Linux/Unix flavour). Run:

```
sudo adduser myusername dialout
```

Replace `myusername` with your username. Then logout and login from the whole windowing system (or reboot) for the change to take effect.

## 2.2 Settings for PuTTY / Windows+Linux



Options controlling the terminal emulation

Set various terminal options

☑ Auto wrap mode initially on
☐ DEC Origin Mode initially on
☑ Implicit CR in every LF
☐ Implicit LF in every CR
☑ Use background colour to erase screen
☐ Enable blinking text

Answerback to ^E: PuTTY

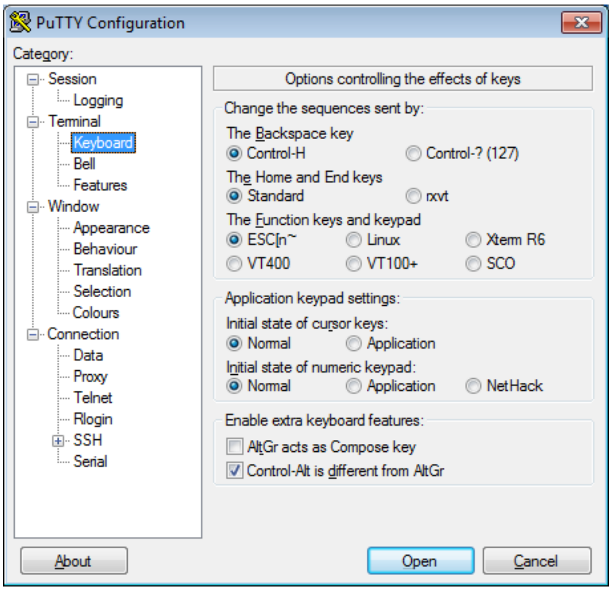
Line discipline options

Local echo: ○ Auto ○ Force on ● Force off
Local line editing: ○ Auto ○ Force on ● Force off

Remote-controlled printing

Printer to send ANSI printer output to:

Figure 1: PuTTY Configuration: Terminal



Options controlling the effects of keys

Change the sequences sent by:

The Backspace key: ● Control-H ○ Control-? (127)
The Home and End keys: ● Standard ○ rxvt
The Function keys and keypad: ● ESC[n~ ○ Linux ○ Xterm R6 ○ VT400 ○ VT100+ ○ SCO

Application keypad settings:

Initial state of cursor keys: ● Normal ○ Application
Initial state of numeric keypad: ● Normal ○ Application ○ NetHack

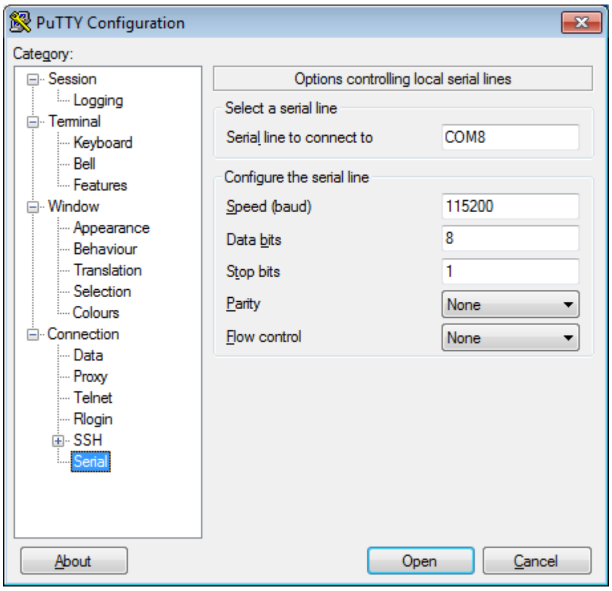
Enable extra keyboard features:

☐ AltGr acts as Compose key
☑ Control-Alt is different from AltGr

Figure 2: PuTTY Configuration: Keyboard

Figure 1 shows the terminal emulation settings for PuTTY. Make sure you check the “Implicit CR in every LF” box. In the keyboard settings in Figure 2, you may set Backspace key to either Control-H or Control-?.

Figure 3 shows the serial communication parameters for 115200 bps, 8N1. For binary file transfers to work, it is important to disable flow control.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM8

Configure the serial line

Speed (baud): 115200

Data bits: 8

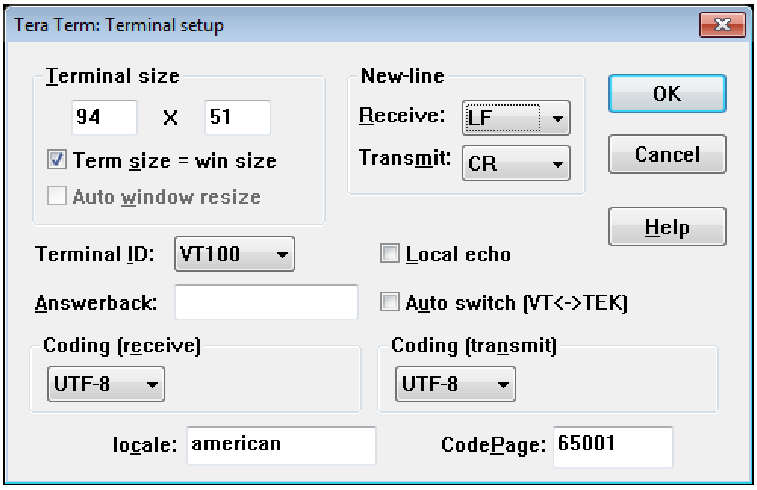
Stop bits: 1

Parity: None

Flow control: None

Figure 3: PuTTY Configuration: Serial

## 2.3 Settings for Tera Term / Windows



Terminal size: 94 X 51

☑ Term size = win size

☐ Auto window resize

New-line

Receive: LF

Transmit: CR

Terminal ID: VT100

☐ Local echo

Answerback:

☐ Auto switch (VT<->TEK)

Coding (receive): UTF-8

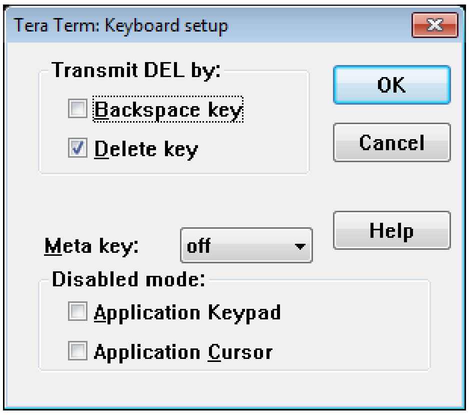
Coding (transmit): UTF-8

locale: american

CodePage: 65001

Figure 4: Tera Term: Terminal Setup

Figure 4 shows how to set line feeds and basic terminal emulation mode (VT100) in Tera Term.



Tera Term: Keyboard setup

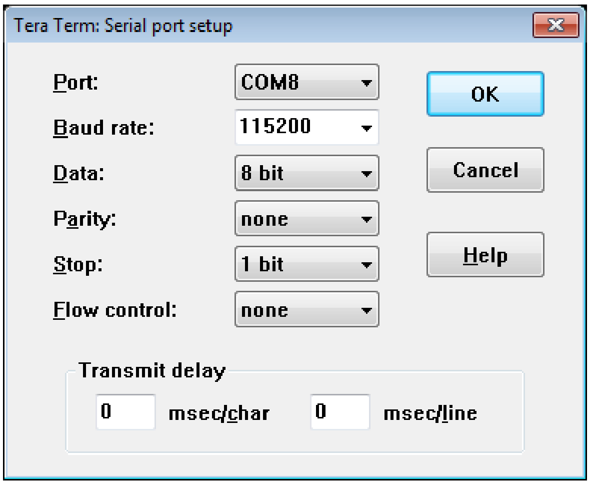
Transmit DEL by:
□ Backspace key
☑ Delete key

Meta key: off

Disabled mode:
□ Application Keypad
□ Application Cursor

Figure 5: Tera Term: Keyboard Setup

Figure 5 shows a working keyboard setup for Tera Term.



Tera Term: Serial port setup

Port: COM8
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Figure 6: Tera Term: Serial Port Setup

Figure 6 shows how to set serial parameters 115200 bps, 8N1, without flow control.

# 3 tEDAx

Audiobook's structure is described using the Trivial EDA eXchange (tEDAx) interchange format. Each of the books has its own `.txt`-file, which is written in tEDAx format. These text files contain all the necessary information about the audiobook, including:

- Book
  - Name of the book.
  - All levels and their names.
- Level items
  - Level number.
  - Full/Short name.
  - File's name. (Only necessary when file changes)
  - Start time.
  - Text. (If available)

To utilize this data on VS1010, this text file has to be converted to binary format (`.tbf`), so storing and accessing the data is as efficient as possible. This conversion can be done by using our tEDAx converter, which is called `tedaxbooktovsdsp`. This converter and its functionality, are described in Chapter 9.

For more information about tEDAx you can visit: *http://repo.hu/projects/tedax/*.

# 4 Audiobook's Data Structure

After generating the `.tbf`-file with the tEDAx converter, this data can now be utilized on VS1010. In this project, this information is then stored and accessed by using structs. These structs are described in the header file `book.h`. Each audiobook includes one Book-struct, and each level item has its own Level Item-struct. These structs contain lots of useful information:

Book-struct:

- Book's name
- Name of each level (e.g. "Section", "Book", "Chapter", "Verse")
- Number of level items
- Book's version
- Identifier

Level Item-struct:

- Parent level's index
- Index to the first item at sublevel
- Number of items at sublevel
- Item's full/short name (e.g. "Old Testament" and "OT")
- Text
- File name
- File length
- Timestamp
- Identifier

With these two structs, the data from the `.tbf`-file can be stored and utilized. This way every part of the book will be easily accessible.

Here is briefly explained, how all of this data is stored in the `.tbf`-file:

```
Part        : Content
--------------------------------------------------------
0x00 - 0x3f : struct Book
0x40 - 0x5f : struct LevelItem[0]
0x60 - 0x7f : struct LevelItem[1]
...
0x40+0x20*(n-1) - 0x40+0x20*n-1 : struct LevelItem[n-1]
0x40+0x20*n - end of the file : Strings
```

Each level item is connected to at least one other level item. All of these level items form a tree structure, which makes navigating to any part of the book fast and easy.

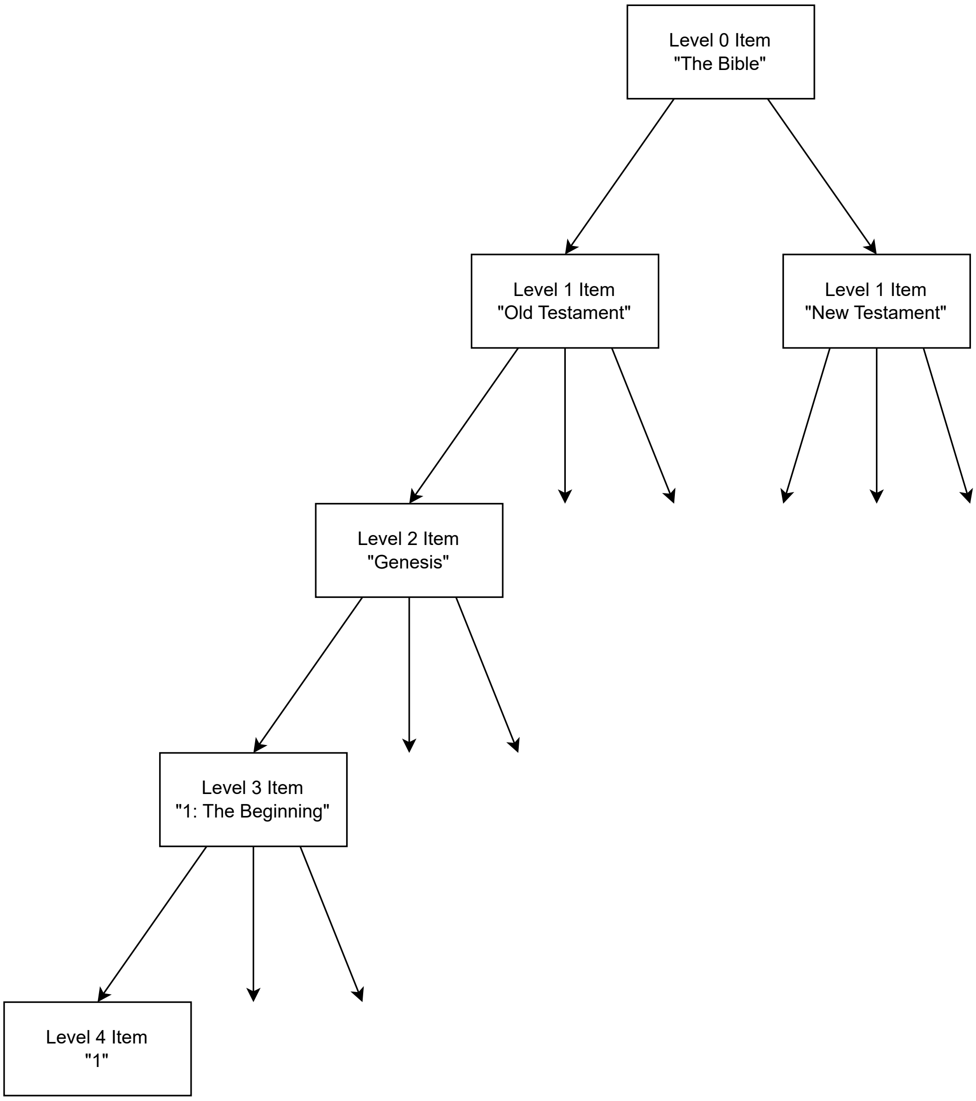


Figure 7: Audiobook's data structure using 4-level Bible as an example

Figure 7 illustrates how the level items are connected with each other in the Bible example.

## 5 Features

Audiobook program is capable of the following features:

- At least 4-Level playback structure, with optionally named levels. For example, in the Bible, the levels are:
  - Level 1: Sections: Old vs New Testament.
  - Level 2: Books: 39 in the Old Testament, 27 in the New Testament.
  - Level 3: Chapters: 929 in the Old Testament, 260 in the New Testament.
  - Level 4: Verses: 23145 in the Old Testament, 7957 in the New Testament.
- Playback from MP3 files. Any file can map to any level of the playback structure.
- Play point selection from any of the existing playback structure levels.
- Volume control.
- Ability to skip to next/previous part.
- Ability to rewind or fast forward.
- Display the text while playing the MP3 file, e.g. the relevant book portion in text form.
- Button/UART controls.

# 6 Demo Board

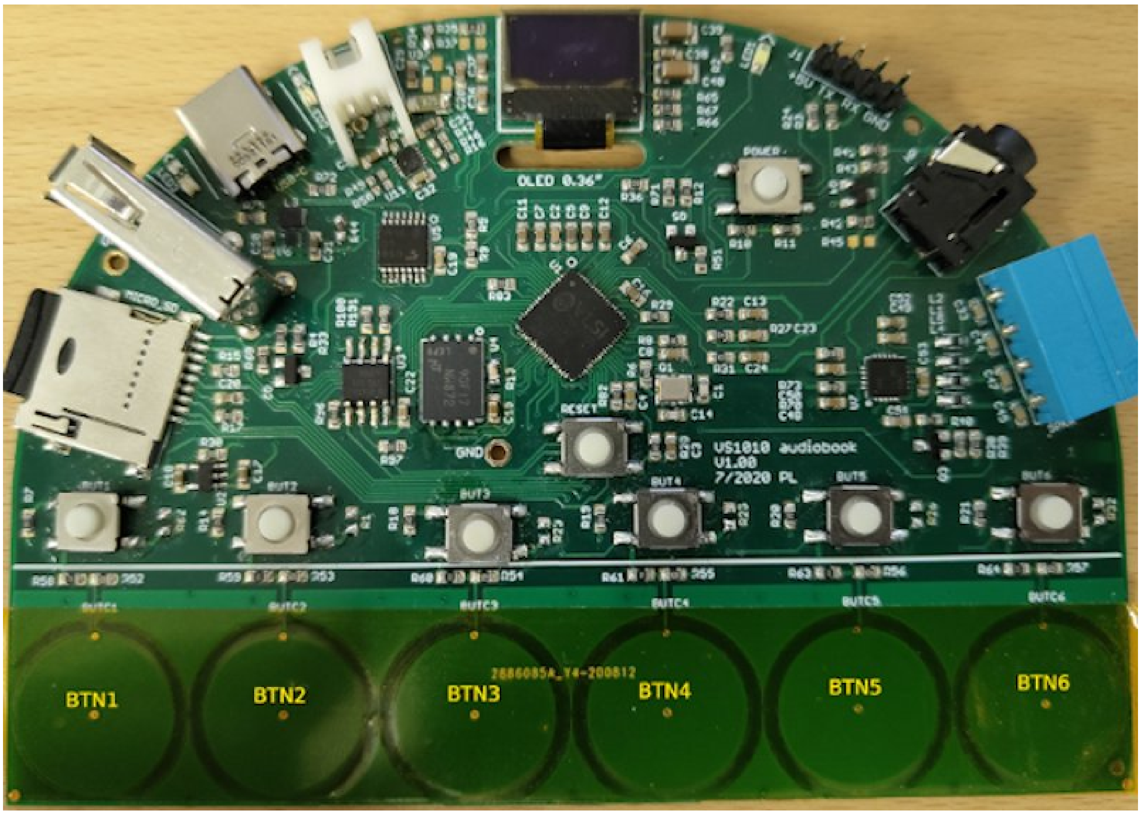


Figure 8: Demo board used in this project

Figure 8 presents the PCB that was used in this project.

This demo board supports both UART and button controls. You are able to choose which controls you would like to use in your own implementation. If you decide to use button controls, make sure the buttons are bound to the correct GPIO pins.

By default, both UART and Button controls are enabled in the demo.

# 7 Controls

Audiobook player can be controlled via UART or with button presses. Each control has been described below:

| **Button/UART Controls** | | |
|---|---|---|
| **Menu Controls** | | |
| **Button** | **UART** | **Description** |
| Button 1, short | ’p’ | Move cursor up. |
| Button 2, short | ’n’ | Move cursor down. |
| Button 5, short | ’<’ | Return to previous menu. |
| Button 6, short | ’>’ | Choose the item, which the cursor is currently pointing at. |
| **Player Controls** | | |
| **Button** | **UART** | **Description** |
| Button 1, short | ’p’ | Go back to previous verse. |
| Button 2, short | ’n’ | Skip to next verse. |
| Button 3, long | ’1’ | Rewind 60 seconds. |
| Button 3, short | ’2’ | Rewind 10 seconds. |
| Button 4, short | ’3’ | Fast forward 10 seconds. |
| Button 4, long | ’4’ | Fast forward 60 seconds. |
| Button 5, short | ’-’ | Lower the volume by 9dB. |
| Button 6, short | ’+’ | Increase the volume by 9dB. |
| Button 5, long | ’<’ | Return to table of contents. |

[1] See Chapter 8.1, *Example: Picking A Verse in The Bible via UART*, for an example of how the level selection works via UART.

# 8 Usage Examples

The user of the Bible audiobook should be able to navigate to any verse with a limited amount of button presses.

For instance, Old Testament, Genesis 1:8 is: Section: Old Testament, Book: Genesis, Chapter: 1, Verse: 8.

When user picks this verse, the book will start playing the correct file where the chosen verse is located at.

Additionally the program will display the part of the book it is currently playing:

```
‘‘Old Testament, Genesis 1:8’’ or ‘‘OT Genesis 1:8’’.
```

Also if there is text available, and if there is enough space to display that text, the program will also print the text of this current verse. After the current verse has ended, the program will switch automatically to the next verse.

## 8.1 Example: Picking A Verse in The Bible via UART

Audiobook program has an ability to be controlled via UART or with button presses, and you are able to choose which controls you would like to use. When using button controls make sure the buttons are bound to the correct GPIO pins to prevent any strange behaviours. In this example we will be using UART controls.

Let’s first assume, that the audiobook player has just started. After the program has done all the necessary initializations, it will list all the available audiobooks in the target directory (in my case this directory is S:/books/. Make sure to change this, if you have different target directory). The program also prints ’<’ cursor to the end of the first book’s name, so the user knows which book he is going to pick. By using ’n’ or ’p’, user can move this cursor up/down in the menu. With ’>’, user can pick the book, where the cursor is currently pointing at. In this example let’s assume the user picks The Holy Bible.

After choosing the bible, the program lists all level 1 items (in this demo case “Old Testament” and “New testament”). By using the same commands as before (’n’, ’p’, and ’>’), the user is able to choose which part of the book he would like to listen. Also by using ’<’, user can return to previous menu.

If the user picks, for example the “Old Testament”, the program will print all available books in the Old Testament (in this case “Genesis” and “Exodus”).

If the user picks, for example the “Genesis”, the program prints all available chapters in Genesis (in this case “The Beginning” and “Adam and Eve”).

After choosing one of the chapters, for example “The Beginning”, the program prints every available verse in that specific chapter. The user interface should now look something like this:

```
Choose a Verse:
The Bible
-Old Testament
--Genesis
---1: The Beginning
----1
----2
----3
----4
----5
----6
----7
----8 <
----9
----10
----11
----12
----13
----14
----15
----16
----17
----18
----19
----20
----21
----22
----23
----24
----25
----26
----27
----28
----29
----30
----31
```

Now the user can decide, which verse he would like to listen. After picking one of the verses, the audio player starts playing the correct MP3 file from the point, where the chosen verse starts at. If available, the program also prints the text of the verse it is currently playing:

```
Old Testament: Genesis 1:8
And God called the firmament Heaven. And the evening and the morning were the second day.
```

After audio player has finished playing the current verse, it starts automatically playing the next verse, and displays it onto the screen the same way as before. While in player, the user can use ’+’ or ’-’ for volume control, ’n’ or ’p’ to skip to next/previous verse, and also ’1’, ’2’, ’3’ or ’4’ to rewind or fast forward. Also user can use ’<’ to exit the player and return to the menu screen. All of these controls are also explained in chapter 7.

# 9 tEDAx converter

As mentioned in chapter 3, to utilize the data from tEDAx text file, it first has to be converted to binary format. This is done by using our tEDAx converter (`tedaxbooktovsdsp`). `Tedaxbooktovsdsp` is a command line tool, which takes any tEDAx format `.txt`-file as an input and converts it to `.tbf`-file. The `.txt`-file must be placed in the same folder as the `tedaxbooktovsdsp` converter in order it to work. Additionally every audio file, which are part of the audiobook, has to be stored in the folder called “audio”. This folder must also be placed in the same folder as the tEDAx converter.

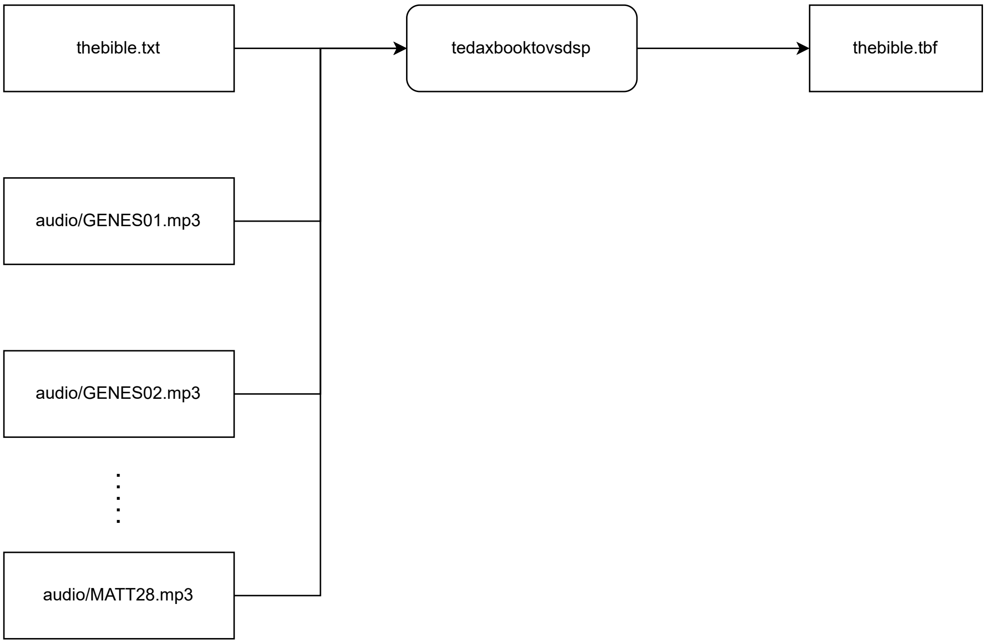


Figure 9: Illustration of tEDAx converter’s input and output files

As an example, to convert a tEDAx file called `thebible.txt` to binary format, on Windows platform it can be achieved by using these two commands on terminal:

```
NOTE: make sure to replace the directory/folder of the CD command with the
actual directory on your system.

cd c:\audiobook\tedaxbooktovsdsp
tedaxbooktovsdsp thebible.txt
```

On Linux platform, running the converter works almost the same way. Just remember to add “./” before the program’s name:

```
./tedaxbooktovsdsp thebible.txt
```

After using these two commands the program creates a new file called `thebible.tbf`, which can then be copied to the VS1010.

# 10 Latest Document Version Changes

This chapter describes the latest changes to this document.

### Version 1.10, 2025-10-21

- Added Chapter 2, *Terminal Emulation Programs and Their Settings*.
- Typo and link corrections.

### Version 1.00, 2023-05-25

- Initial public release.

### Version 0.4, 2023-05-22

- Minor changes to the documentation. Also added a new figure of the tEDAx converter.

### Version 0.3, 2023-05-19

- Updated information about tEDAx and audiobook's data structure.

### Version 0.2, 2023-04-13

- Updated Rewind/Fast forward feature. Added file length to level items.

### Version 0.1, 2023-02-10

- Updated audiobook specs.

### Version 0.01, 2022-10-06

- First, very early draft / suggestion.

## 11 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

URL: http://www.vlsi.fi/
Phone: +358-50-462-3200
Commercial e-mail: sales@vlsi.fi

For technical support or suggestions regarding this document, please participate at
http://www.vsdsp-forum.com/
For confidential technical discussions, contact
support@vlsi.fi