TURGEN SYSTEM DOCUMENTATION
version 8.3.17

Michael Kalouš

Contents

Part I
TURGEN SYSTEM

1 Introduction

1.1 Primary Functionality

TURGEN SYSTEM converts files to various turbo systems and also to standard tape records.
Output of the program is a WAVE file, an electric signal on the output of the sound card, or a tape image.
TURGEN SYSTEM has special support for conversion of segmented binary files which includes a wizard and a set of binary loaders.

1.2 Auxiliary Functionality

- Turbo decoder that can be used to retrieve information from tapes
- Reading of tape images
- Data transfer via serial port in collaboration with CAS COM utility
- Tool for merging segments of binary files
- Tool for embedding tokenized BASIC files to monolithic binary files
- Tape image extractor tool that extracts data from tape images

1.3 Support for Standard Tape Records

Standard tape records are fully supported. Tape images with baud, data, and fsk chunks can be read by the Tape image plugin. Monolithic binary files, segmented binary files, BASIC files, and plain data files can be converted to standard tape records by the Standard plugin. Refer to sections 17 and 18.

1.4 Hardware that can be used

Data recorders with or without turbo upgrades, data recorder replacement devices, MEGA-CD interface (CD-LINK), SIO2PC, ATART.

1.5 Target Audience

Owners of Atari 8-bit computers who use data recorders or replacement devices.

1.6 Program Characteristics

TURGEN SYSTEM is written in Java™ 2 programming language and it is a free software distributed under the terms of GNU General Public License, version 2.

1.7 System Requirements

- Java Runtime Environment 1.5.0 or newer
- Solaris, Microsoft Windows 98/2000/ME/XP/Vista/7/8, GNU/Linux, Mac OS

1.8 Glossary

Abbreviations

TS. TURGEN SYSTEM.

JRE. Java Runtime Environment.

JDK. Java Development Kit.

Terminology

Binary file. File primarily designed to accommodate a *program* for Atari 8-bit computers. Binary file has a defined internal structure.

Every binary file starts with a two-byte header (or eye-catcher). Both bytes have value of 255.

Binary file consist of data blocks called *segments*. A segment is a block of data that has a *segment header*. The segment header is four bytes long and accommodates two 16-bit addresses (start address and end address). These two addresses determine where to place the segment when it is being loaded from a peripheral device into memory.

There are four types of segments. All of them are described in table 1.

Segment header	Segment description
736, 737	**RUN segment.** The segment is always 2 bytes long. These two bytes accommodate an address that specifies where to jump after all segments of a given binary file are loaded.
738, 739	**INIT segment.** The segment is always 2 bytes long. These two bytes accommodate an address of a subroutine that is executed immediately after the INIT segment is loaded.
736, 739	**RUNINIT segment.** Combination of the previous jump segments. 4 Bytes long.
Other	**Data segment.** Such segment accommodates executable code or data.

Table 1: Segments

Monolithic binary file. Binary file with a very simple structure. It consists of exactly one data segment and at most one RUN segment.

Segmented binary file. Binary file that is not monolithic.

Binary loader. Program, or routine (usually part of an operating system) that loads and executes binary files.

Binary load. Name for the process of loading and running of a binary file. This process is performed by a binary loader.

Cassette recorder. Consumer electronics device designed to read or write electric signal to or from compact cassettes.

Data recorder. Device designed to read or write signal to or from compact cassettes, specially designed to be connected to computers, for example Atari XC-12.

Program directory. Directory or folder where TURGEN SYSTEM is installed to. Special symbol `<TSDIR>` is also used to reference this directory.

Configuration directory. Directory or folder where TURGEN SYSTEM stores user configuration files. Special symbol `<CFGDIR>` is also used to reference this directory.

Java home directory. Directory or folder where JRE (or JDK) resides. Special symbol `<JAVAHOME>` is also used to reference this directory.

Turbo system. This term denominates a system designed to speed up data transfer of the original Atari data recorder. A typical turbo system consist of three parts: 1. Hardware modification of data recorder, 2. Software exploiting such hardware modification, 3. File format. There is not necessarily 1:1:1 relation. For one hardware modification of data recorder, there can exist more software or more file formats.

TURGEN SYSTEM restricts this term to systems that use PWM (pulse width modulation) to encode data. Some examples of turbo systems are Turbo 2000 or Turbo Blizzard. Example of system that is not considered to be turbo system is the IRON TURBO.

2 Distribution and Installation

Preliminary Operations

Before installing TURGEN SYSTEM, do install the Java Runtime Environment. Download JRE from the following address: http://java.com/en/download/index.jsp. It is recommended to download the newest available version of JRE for your operating system.

Distribution

Program is distributed in two forms:

1. Compressed tar files. These files are devoted for users that do not use supported versions of Microsoft Windows or those who want to avoid overhead of automated installers for some reasons.
2. Automated installer for Microsoft Windows. This installer provides convenience and creates shortcuts in the Start menu or on the desktop. Shortcuts won't be created if JRE (or JDK) is not installed.

Installation and Directories.

To install TURGEN SYSTEM, decompress the compressed tar file or run the automated installer. During installation, the program directory is created and populated with files.

The program directory contains all necessary files required to run TURGEN SYSTEM. TS never writes to this directory; it is recommended to set read-only access to the directory in order to provide security.

The configuration directory is created by TS when needed (usually during the first run). Its location is operating system dependent and TS requires read-write access for full functionality, however it will still work with limitations even though read-write access is not granted. To display name of this directory, select *About* item from the *Turgen* menu.

3 Operations Guide

3.1 Starting TURGEN SYSTEM

The Hard Way

Program code of the TURGEN SYSTEM is located in the `turgen.jar` file. To run TURGEN SYSTEM, run Java launcher and specify that the program code is in the `turgen.jar` file, using the following commands.

`<JAVAHOME>/bin/java -jar <TSDIR>/turgen.jar` on Unix-like systems or
`<JAVAHOME>\bin\javaw.exe -jar <TSDIR>\turgen.jar` under Microsoft Windows.

Microsoft Windows

Use shortcuts created by the automated installer or double-click the `turgen.jar` file icon.

Other Operating Systems

Create launchers or shortcuts using means provided by your desktop environment.

Command Line Parameters

One command line parameter is accepted - a file that accommodates a playlist.

Log File

Exceptions, traces or other important diagnostic information is stored into the `turgen.log` file located in the configuration directory. This file is recycled when it exceeds size of 4 MB.

3.2 Program Controls

Main Menu and Playlist

The main menu is located on the top of the program window. Almost all functions are accessible from the main menu.
In the middle of the program window, there is *playlist* that accommodates information what files are to be converted and how they will be converted.

Main Control Buttons

Under the playlist, there is a panel, where the *main control buttons* are located. The most used functions of TS are reachable using these buttons. Every button has its icon and name. The buttons are depicted in table 2.

Icon	Name	Icon	Name
	Add item to playlist		Move 1 playlist item up
	Edit selected playlist item		Move 1 playlist item down
	Remove selected playlist items		Generate WAVE file
	Select all playlist items		Generate AUDIO directly
	Generate tape image		Wizard for binary files

Table 2: Main control buttons

Progress Monitoring Panel

This panel is located in the bottom part of the program window. Conversions of files can be monitored or stopped here. At the very bottom of the program window, there is a status bar.

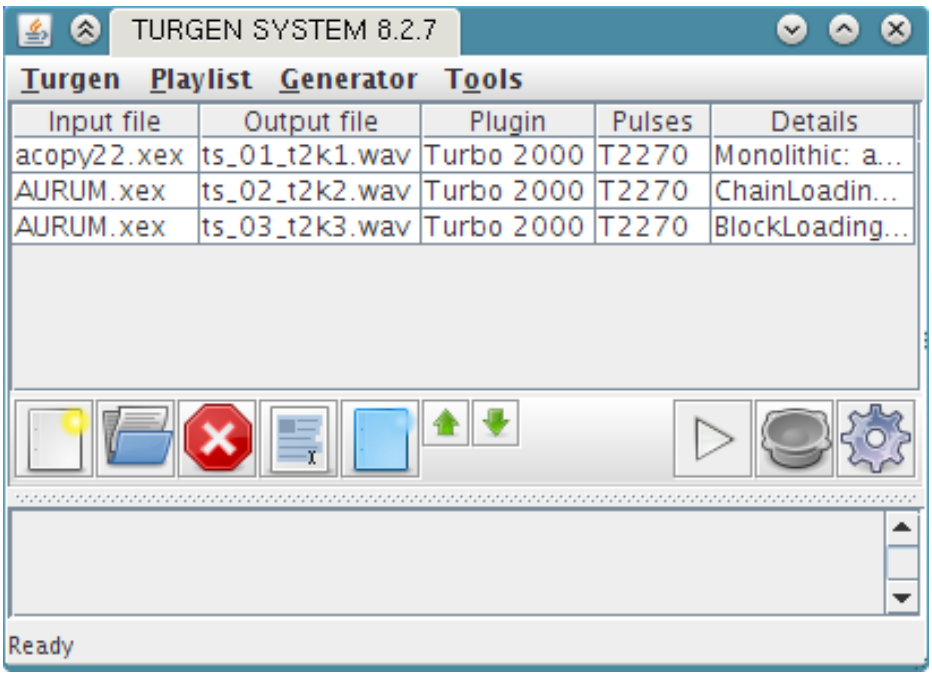

Figure 1: Program window

3.3 Conversion of Files

3.3.1 Playlist Items

In order to convert a file, TURGEN SYSTEM requires information about such conversion which includes input file name, input and output format, output file name, what loader to prepend and various other parameters. Such information is accommodated in *playlist items*. Playlist items are elements of the *playlist*.

3.3.2 Working with Playlist

To work with playlist, use the main control buttons or the items from the *Playlist* menu.

To add a playlist item to the playlist, click the *Add item to playlist* main control button. To edit playlist items, click the *Edit selected playlist item* main control button. To remove selected playlist items, click the *Remove selected playlist items* main control button.

To select playlist items, use keyboard or mouse. To select all playlist items, click *Select all playlist items* main control button.

To move single playlist items use *Move 1 playlist item up* and *Move 1 playlist item down* buttons.

To load or save a playlist, use the *Load* and *Save* items from the *Playlist* menu.

3.3.3 Playlist Item Manipulation

TURGEN SYSTEM provides a dialog for playlist item manipulation which is depicted on figure 2.

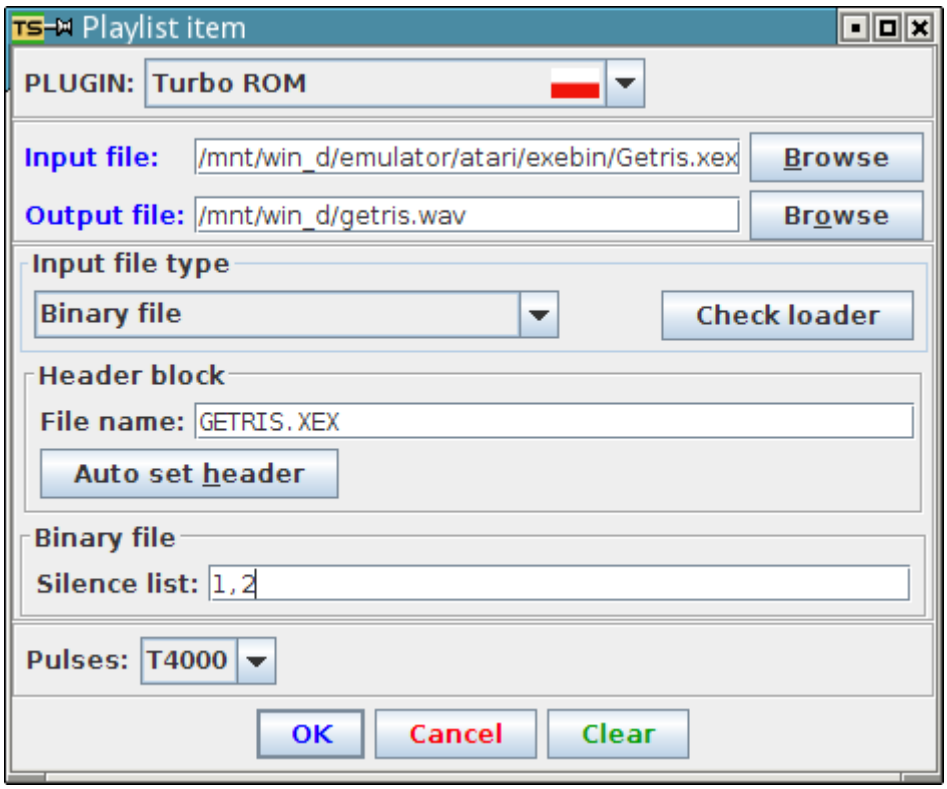

Figure 2: Manipulating playlist item

Use the Combo box *PLUGIN* to choose plugin that will be used for conversion. National flags can be used to distinguish from where a particular turbo system comes from.

In the top part of the dialog, there is a common panel that is used to set input and output file. To display a dialog with information about input file, click the *Input file* label. To fill the output file text field automatically, click the *Output file* label. To display a file chooser dialog, click the *Browse* buttons.

In the middle part of the dialog, there are controls specific for the selected plugin.

Use buttons at the bottom to commit or cancel playlist item manipulation. Click *Clear* button to reset controls in the dialog.

3.4 Wizard for Binary Files

Wizard for binary files provides convenience when converting binary files.

To start the wizard, click the *Wizard for binary files* main control button.

In the first step, specify a binary file that you want to convert and what are the capabilities of your data recorder. In the second step, choose one of the conversion methods offered by the wizard, output file name and pulses.

Offered conversion methods are ordered by the number of pilot tones or leaders. The wizard also checks (finitely of course) whether the binary file will destroy a built-in binary loader that will be prepended.

3.5 Output

3.5.1 Output of Electric Signal into WAVE File

To output electric signal into a WAVE file, select at least one playlist item and click the *Generate WAVE file* main control button. A separate thread for each selected playlist item is created. These threads run in parallel and it is possible to stop them using the *Stop* buttons that will appear in the progress monitoring panel. The output files are overwritten without a warning.

3.5.2 Output of Electric Signal to Computer's Audio System.

To output electric signal to computer's audio system, select at least one playlist item and click the *Generate AUDIO directly* main control button. Playlist items are processed serially. It is possible to stop the output using *Stop* and *Stop all* buttons that will appear in the progress monitoring panel.

3.5.3 Output of Tape Image

To output a tape image, select at least one playlist item and click the *Generate tape image* main control button. Playlist items are processed serially. The output files are overwritten without a warning. The format of tape image can be configured using Program configuration facility described in section 4.

3.6 Input/Output matrix

Conversion capabilities of Turgen System are enumerated in the following table.

Input	Wave file	Audio system	Tape Image with **trXX** chunks	Tape image with **pwmX** chunks
Regular file	YES	YES	YES	YES
Binary file	YES	YES	YES	YES
Tape image (**trXX** chunks)	YES	YES	Passed through	YES
Tape image (**pwmX** chunks)	YES	YES	NO	Passed through
Tape image (**baud, fsk** chunks)	YES	YES	Passed through	Passed through

4 Program Configuration Facility

The Program configuration facility provides various program parts with configuration capabilities in a uniform way. Program configuration consists of *configuration entries* that are grouped in *configuration classes*. Program parts use API to retrieve contents of configuration entries, user is provided with *Preferences dialog* which he can use to modify configuration. To provide persistence, configuration is stored in a properties file.

Some configuration entries are used by the core of TURGEN SYSTEM, others are used by the plugins.

4.1 Modifying Configuration

There are two ways how the configuration can be modified:

1. Usage of the *Preferences* dialog. To make a modification permanent, it is necessary to save the configuration by selecting the *Save preferences* menu item from the *Tools* menu. This is the preferred way how to modify the configuration.
2. Manual adjustment of the `turgen.properties` file that can be found in the configuration directory. This is deprecated way and it is not described in this documentation.

4.2 General Configuration Entries

TURGEN SYSTEM/GUI Look and feel
Look and feel of the user interface, fully qualified class name. If the entry is empty, default look and feel is used.

Standard looks and feels:
`javax.swing.plaf.metal.MetalLookAndFeel` (all platforms, default)
`com.sun.java.swing.plaf.motif.MotifLookAndFeel` (all platforms)
`com.sun.java.swing.plaf.windows.WindowsLookAndFeel` (Microsoft Windows only)
`com.sun.java.swing.plaf.gtk.GTKLookAndFeel` (only when GTK+ 2.2 or newer is installed)
`com.sun.java.swing.plaf.mac.MacLookAndFeel` (only Mac, untested)
`com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel` (new since JDK 1.6 update 10)

TURGEN SYSTEM/favorite output directory
Favorite directory for output files.

4.3 Output of Electric Signal to Computer's Audio System

Audio output/Silence between files
Length of silence in milliseconds generated between playlist items.

Audio generator/Amplitude
Amplitude of the signal. 0-100%. It is not recommended to set the value too low.

Audio generator/Channels
Number of channels.

Audio generator/Bits per sample
Number of bits per one sample.

Audio generator/Signed samples
Indicates whether to use signed samples.

Audio generator/Initial silence
Length of silence generated after audio system initialization, unit is 0.1 of second. This may be needed, because the very first data sent to the audio system can result in unwanted signal.

Audio generator/Terminal silence
Length of silence generated before audio system termination. This may be needed, because the very last data sent to the audio system can be truncated.

Audio generator/Buffer size
Size of the buffer for data being sent to the audio system. Increasing this value can help when there are under runs and output from the audio system is repeatedly interrupted.

Audio generator/Signal in right channel only
If the number of channel is set to two, signal will be generated to the right channel only.

Audio generator/Do not modulate standard records
If set to true, standard tape records will be generated without modulation.

Audio generator/Use double buffering
If set to true, an internal buffer is used to store the electric signal before it is sent to the computer's audio system. This reduces number of round-trips between TS and operating system's native sound facilities.

Audio generator/Use harmonic pulses
If set to true, harmonic pulses are generated instead of rectangular pulses. This can be useful when the electric signal is passed through a device that distorts signal with steep edges.

4.4 Output of Electric Signal into WAVE File

WAVE generator/Amplitude
Amplitude of the signal. 0-100%. It is not recommended to set the value too low.

WAVE generator/Channels
Number of channels.

WAVE generator/Bits per sample
Number of bits per one sample.

WAVE generator/Postprocessing command
Command line for postprocessing. For more information refer to section 7.5.1.

WAVE generator/Change extension to .wav
Extension of output files will be changed to .wav in intelligent way if set to true.

WAVE generator/Signal in right channel only
If the number of channel is set to two, signal will be generated to the right channel only.

WAVE generator/Do not modulate standard records
If set to true, standard tape records will be generated without modulation

WAVE generator/Use harmonic pulses
If set to true, harmonic pulses are generated instead of rectangular pulses. This can be useful when the electric signal is passed through a device that distorts signal with steep edges.

4.5 Output of Tape Images

Tape image output/Change extension to .cas
Extension of output files will be changed to .wav in intelligent way if set to true.

Tape image generator/Postprocessing command
Command line for postprocessing. For more information refer to section 7.5.2.

Tape image generator/Auto create temporary files
If the output file is not specified and this entry is set to true, temporary file is created automatically. This is useful for postprocessing.

Tape image generator/Tape image chunks
This entry is used to specify, what type of chunks will be placed into generated tape images. trXX style chunks or pwmX can be specified. trXX style chunks are used by CASCOM , pwmX style chunks are used by A8CAS.

5 Advanced Settings

5.1 Active Plugins

The list of plugins that will be loaded is stored in the `plugins.list` file. To disable a plugin, remove or comment-out (using the # character) the corresponding line. To change the order of the plugins (that is what users from Poland may want to do), change order of the lines.

Note: Some plugins process trXX tape image chunks. If a plugin is disabled, given trXX tape image chunk cannot be processed.

5.2 Repository of Pulses

The repository of pulses is stored in `pulses/pulses.list` file. Repository can be modified, directions are present in the file itself as comments. It is not recommended to modify the repository without serious reasons. Changes take effect after TS is restarted.

6 Tools

6.1 Merging Segments of Binary Files

6.1.1 Introduction

TURGEN SYSTEM is equipped with a tool that merges segments of binary files into one segment and adds extra code that emulates the effect of the INIT segments. This allows to convert segmented binary files that meet certain conditions to monolithic binary files.

This tool was created in early times of TURGEN SYSTEM in order to partially cope with limitations of Czechoslovak Turbo 2000 and Super Turbo turbo systems.

Conditions that the segmented binary files must meet are the following: 1. Data segments must not overlap, 2. Routines called by the jump segments must work even when all data segments are already loaded, 3. There must be place for code that emulates effect of the INIT segments. Usually, the only way how to test this conditions is to run the resulting monolithic binary file in some emulator. It should be noted that capabilities of this tool must be considered limited.

6.1.2 Merging Segments Step by Step

Display the dialog for merging segments of binary file by selecting item *Merge segments of binary file* from the *Tools* menu.

Fill-in the *Input file* text field and press the *Analyze* button. The list of segments will be filled-in and you will be able to check whether the first condition is met. Then fill-in the *Output file* text field and specify address of the replacement code that will emulate the effect of jump segments. The replacement code is needed only when there are INIT or RUNINIT segments in the binary file.

Click the *Merge !* button to merge the segments and create the output binary file. Test the created binary file using some emulator.

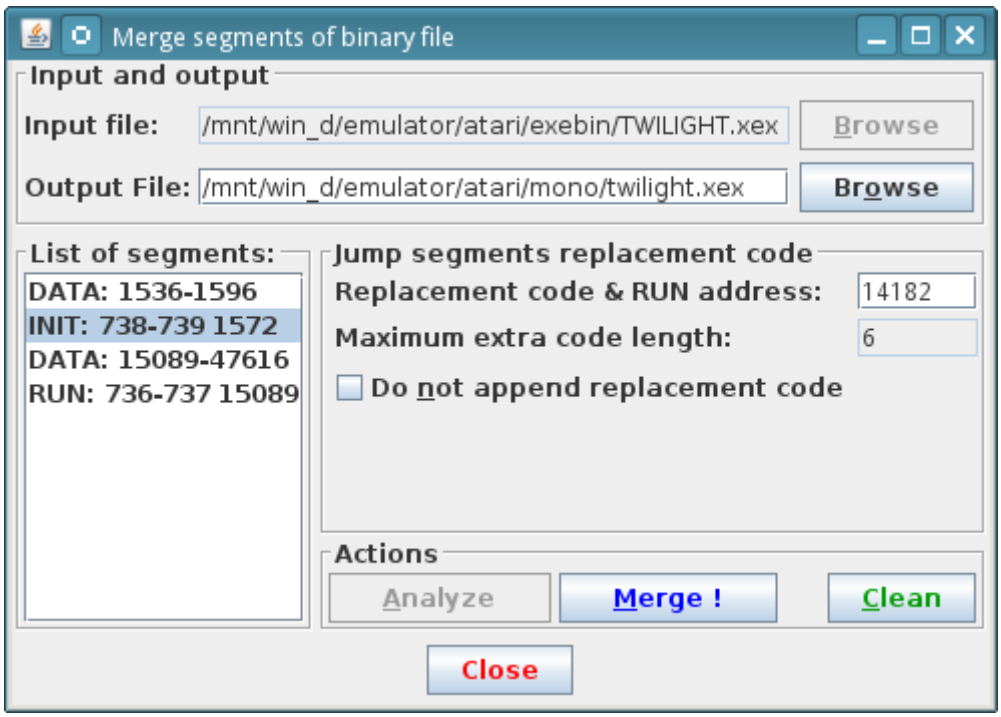

Figure 3: Merging segments of binary file

6.2 Turbo Decoder

6.2.1 Overview

Turbo decoder is a tool designed to retrieve data from compact cassettes. The decoding algorithm is a simple Java rewrite of turbo loaders from assembler 6502.

The electric signal can be read from WAVE files. When creating WAVE files, it is recommended to set the cassette recorder to get maximum signal amplitude. The WAVE file must be in the following format: PCM, 1 or 2 channels, 44100 - 96000 Hz, 8 or 16 bits per sample.

Another possibility is to read the electric signal directly from the computer's audio system. This source of electric signal is supported, but not recommended as it requires error-prone processing of the signal in real time.

Turbo systems supported by the decoder are the following: Turbo 2000, Turbo 2000 - kilobyte blocks, Super Turbo, Turbo Tape and B-TAPE, KSO Turbo 2000, Turbo Blizzard, Turbo ROM, Atari Super Turbo (AST format only), Hard Turbo, and Lower Silesia Turbo 2000.

The decoder can also work in so called "Monitor mode" that allows you to read raw turbo blocks.

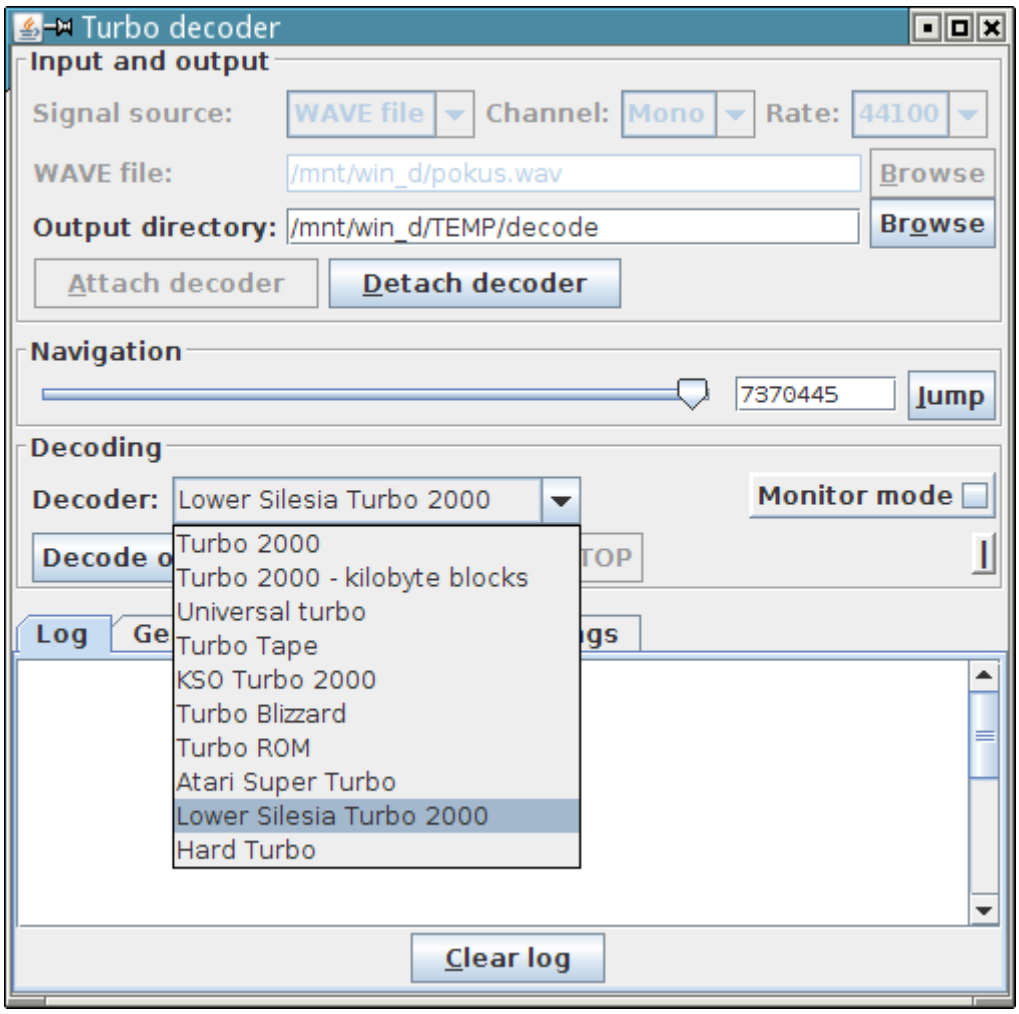

Figure 4: Turbo decoder

6.2.2 Preparing the Decoder

Open the decoder window by selecting the *Turbo decoder* item from the *Tools* menu.

Select source of the electric signal using the *Signal source* combo box.

If the source of the electric signal is a WAVE file, fill-in the WAVE file name and select input channel using the *Channel* combo box.

If the source of the electric signal is the computer's audio system, select input channel using the *Channel* combo box and select the sample rate using the *Rate* combo box.

Fill-in the output directory and click the *Attach Decoder* button. The decoder is now ready to decode the electric signal.

6.2.3 Decoding Files

To set various decoding parameters, use controls on the *Settings* tab.

Use the *Navigation* panel to specify current position in the WAVE file.

The controls on the *Decoding* panel are devoted to perform decoding. Use the combo box *Decoder* to select a turbo system.

To start decoding of one file or all files until the end of the WAVE file, click *Decode one file* or *Decode until EOF* buttons. To stop the decoding process, press the *Stop* button.

You can see the results of decoding in the *Log* tab. If you use a digitized sound editor, you can use the sample numbers enclosed in the curly braces.

If you want to work with different WAVE file, or to select another source of the electric signal, detach the turbo decoder using the *Detach decoder* button.

If the decoder stops responding due to a bad audio system setup, press the *Stop* button while pressing the SHIFT key. This is called emergency stop. After an emergency stop, the decoder must be detached and attached to be ready again.

6.2.4 Monitor Mode

To work in the monitor mode, select the *Monitor mode* check box. In the Turbo monitor mode, turbo decoder reads raw blocks of the selected turbo system. If a block is found, it is decoded until an error occurs or end of block is encountered. No checksums are verified and internal format of the block is not validated.

To decode a single block, click the *Decode one file* button. To decode all blocks until end of file, click the *Decode until EOF* button.

The monitor mode is available for special purposes like decoding data stored in non-standard formats, retrieving corrupted files etc.

6.3 Embedding Tokenized BASIC Files to Binary Files

6.3.1 Motivation

Some turbo systems define file formats than cannot accommodate tokenized BASIC files. In order to circumvent such limitation, TS has a tool that embeds tokenized BASIC files in monolithic binary files. **The purpose of the tool is embedding, not compilation**.

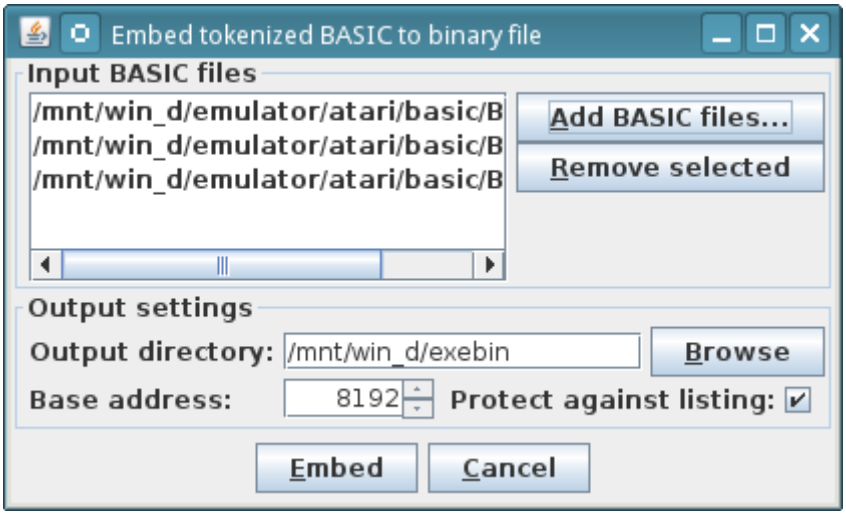

Figure 5: Embedding tokenized BASIC files to binary files

6.3.2 Operations

Open the embedding tool window by selecting the *Embed BASIC to binary file* item from the *Tools* menu.

Click the *Add BASIC files...* button to display a file chooser that allows you to select BASIC files to be embedded to binary files. Chosen files are added to a list. You can select multiple files to be embedded. Click the *Remove selected* button to remove BASIC files from the list.

Use the *Output directory* text field to specify into which directory the binary files will be placed. Use the *Browse* button to specify the directory using a file chooser.

Use the *Base address* spinner to specify address to which the BASIC files will be loaded (tokenized BASIC code is relocatable).

Use the *Protect against listing* check box to protect the BASIC program against listing by disabling the BREAK key and setting the COLDST flag. Note that such protection is only rudimentary and can be broken easily.

To embed the selected BASIC files in binary files, click the *Embed* button. All BASIC files in the list will be embedded in binary files. A report is displayed after embedding processing finishes.

6.4 Tape Image Extractor

Tape image extractor is a tool designed to extract data from tape images. Tape image extractor allows you to do the following:

1. Display basic information about tape image chunks
2. Select tape image chunks for data extraction
3. Determine which portions of tape image chunks will be extracted
4. Extract data from tape image chunks to plain data files
5. Extract boot files to binary files
6. Extract data from tape image chunks to binary files

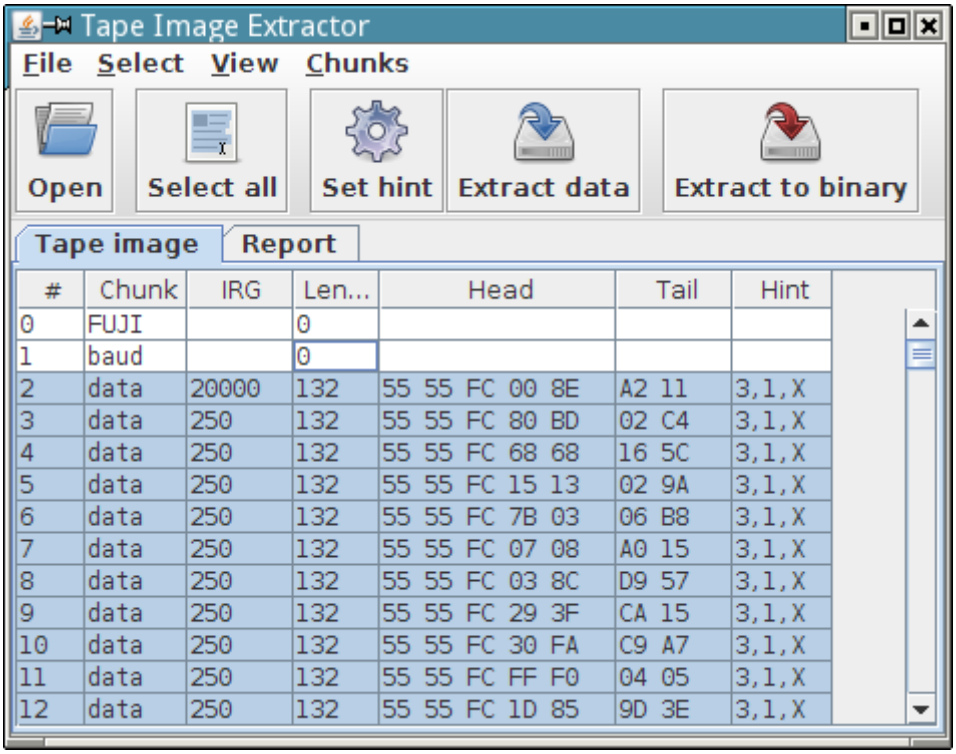

Figure 6: Tape Image Extractor

6.4.1 Operations

To open the Tape Image Extractor window, select the *Tape image extractor* item from the *Tools* menu.

To open a tape image, select the *Open tape image* item from the *File* menu.

To extract data from tape image chunks, select one or more tape image chunks. Then select the *Extract data* item from the *Chunks* menu. Select output file using the displayed file chooser. A report is displayed after the extraction.

To extract data from tape image chunks to a binary file, select one or more tape image chunks. Then select the *Extract data to binary file* item from the *Chunks* menu. A dialog that will allow you to select output file and parameters of the extraction will be displayed. Enter values and click the *Extract* button to perform the extraction.

To determine which portion of tape image chunk or chunks will be extracted, select one or more tape image chunks. Then select the *Set data extraction hint* from the *Chunk* menu. Enter values and click the *OK* button.

7 Appendices

7.1 Hints and Tips

- If the SHIFT key is pressed when the *Add item to playlist* main control button is pressed, the contents of the playlist item manipulation dialog will not be cleared. This can be used to duplicate playlist items.
- A good way how to reduce loading times is to reduce length of pilot tones.
- Don't forget that you can resize program windows as you want. Bounds of the windows are persistent, so if you exit the TS and restart it, you will find the windows exactly where you left them.
- If you, for some reason, store the generated electric signal on compact discs (CDs) and you use compact disc player, try to switch off various shock protections and avoid nonstandard rotational speeds. Such features sometimes cause sample looses.

7.2 Data Recorders

In case you are using data recorder and compact cassettes for data storage, use fast transfer rates with care. It is perfectly OK to use fast transfer rates for single data transfers, but it is not OK in case of data preservation, because compact cassettes tend to age and the signal can degrade or disappear.

You must use standard compact cassettes, these are usually marked as:
IEC I/TYPE I/NORMAL BIAS/NORMAL POSITION
Compact cassettes marked as CHROME, METAL or HIGH POSITION are not suitable for data recorders.

7.3 Data Recorder Replacements

During the time, various data recorder replacements were invented. Some of them are simple adapters that allow to connect cassette recorders or compact disc players, some of them use built-in portable music players.

Most of such devices lack the MOTOR CTRL signal handling. This makes sometimes the binary load operation difficult to accomplish, because execution of routines invoked from INIT segments can take some time. Partial remedy to this problem is usage of the silence lists that allow to generate silence after blocks that accommodate INIT segments.

7.4 MEGA-CD interface (CD-LINK)

You can use MEGA-CD interface to transfer Turbo 2000, Super Turbo, Turbo 2000 - kilobytes blocks, and B-TAPE records to Atari.

- Use silence lists that allow to generate silence after blocks that accommodate INIT segments.
- To transfer standard tape records, set *Do not modulate standard records* configuration entries to *true*.

7.5 Postprocessing

TURGEN SYSTEM allows to process its outputs by external programs. This is called *postprocessing*. After the output is created, an external program according to the given *command line* is executed.

To reference the TURGEN SYSTEM's output in the command line, some special symbols are introduced. Meaning of the symbols is described in table 3.

If the command line starts with the exclamation mark (!), output file is deleted after the external program terminates, if possible.

Processing of the command line is very limited. The command line is interpreted as a sequence of strings separated by spaces. The first string is name (possibly including path) of the external program, the remaining strings are parameters passed to it. Parameters containing spaces have to be enclosed in double quotes.

Command lines (after special symbol substitution) are always stored into the `turgen.log` file, so the correctness of command lines can be verified.

Symbol	Meaning
%OD%	Output directory
%ODS%	Output directory followed by file separator
%OFN%	Output file name without last extension
%OFNE%	Output file name with last extension

Table 3: Special symbols

7.5.1 Postprocessing of WAVE Files

To switch on postprocessing of the WAVE files, select the *Wave postprocessing* check box menu item from the *Generator* menu. Specify the command line in the Wave generator/Postprocessing command configuration entry.

7.5.2 Postprocessing of Tape Images

To switch on postprocessing of the tape images, select the *Tape image postprocessing* check box menu item from the *Generator* menu. Specify the command line in the Tape image generator/Postprocessing command configuration entry.

7.5.3 Examples

Example of special symbols for output file: `/mnt/win_d/emulator/atari/wav/river_raid.wav`
`%OD% = /mnt/win_d/emulator/atari/wav`
`%ODS% = /mnt/win_d/emulator/atari/wav/`
`%OFNE% = river_raid.wav`
`%OFN% = river_raid`
Example of conversion to MP3:
`lame %ODS%%OFNE% %ODS%%OFN%.mp3`
Example of conversion to OGG Vorbis with deletion of the original file:
`!oggenc -o %ODS%%OFN%.ogg %ODS%%OFNE%`
Conversion to MP3 using LAME in the terminal emulator and consequent playback by xmms media player. The original file is deleted:
`!konsole -e bash -c "lame %ODS%%OFNE% %ODS%%OFN%.mp3 && xmms %ODS%%OFN%.mp3"`

7.6 Data Transfer via Serial Port in Collaboration with CAS COM Utility

7.6.1 Introduction

CAS COM is a small utility for loading tape images to Atari via serial port. It supports standard and Turbo 2000 cassette images (namely tr2k tape image chunks). CAS COM works with SIO2PC, ATART or compatible hardware interfaces.

CAS COM can be used as postprocessor of tape images created by TURGEN SYSTEM. Since CAS COM is currently available only for Microsoft Windows operating systems, the following text presumes that such operating system is used.

CAS COM has a web page: http://sdq.czweb.org/old_computers/atari/projects/cascom/index.html.

7.6.2 Command Line Considerations

Let us presume that CAS COM resides in `C:\UTILS\ATARI` directory. The configuration entry Tape image generator/Postprocessing command should have the following value:

`cmd /x C:\UTILS\CASCOM\CASCOM.EXE %ODS%%OFNE%`

It is important to add command line interpreter to the command line, because CAS COM uses console I/O to display important information. Users of Microsoft Windows 95/98/98SE/ME should use `command` instead of `cmd`.

CAS COM accepts other useful command line parameters. `/S` starts tape image loading immediately, `/e` terminates program after data transfer. Moreover `/1`, `/2`, `/3`, `/4` or `/c<num>` parameters allow to choose serial port.

Part II
BUILT-IN PLUGINS

8 Turbo 2000 and Super Turbo

8.1 Characteristics

Plugins convert input files to Czechoslovak turbo systems **Turbo 2000** and **Super Turbo**. Both plugins also provide means to circumvent main limitation of both systems - the incapability to accommodate segmented binary files. User interface is depicted on figure 7.

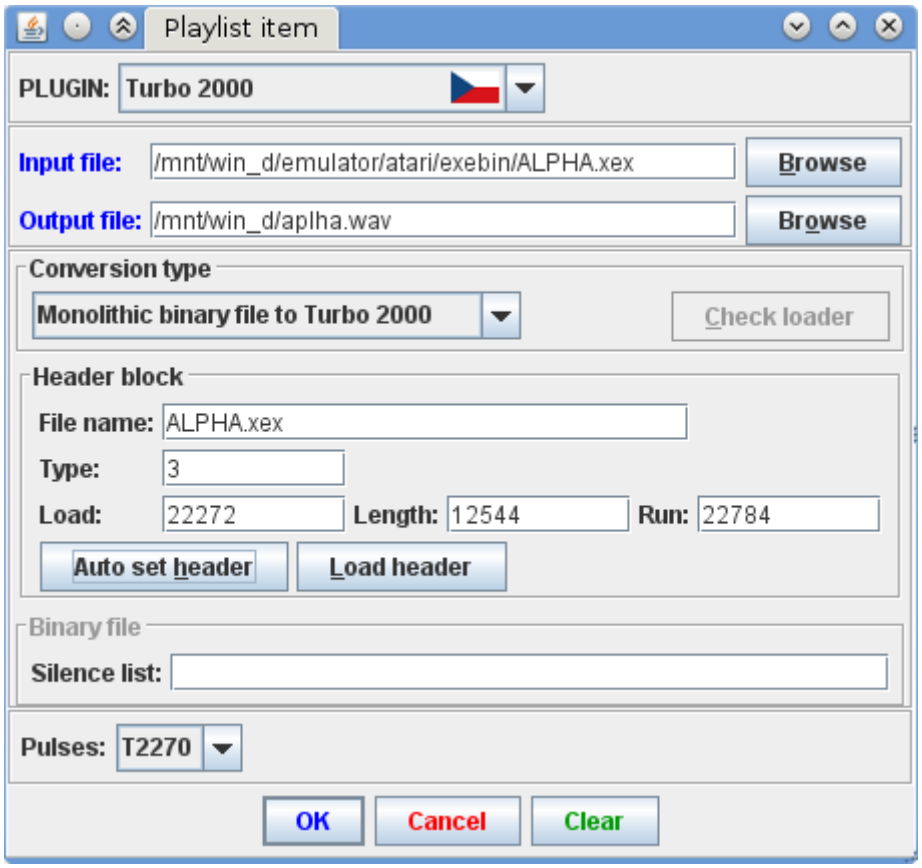

Figure 7: Turbo 2000, Super Turbo

8.2 Conversion types

Turbo 2000 and Super Turbo support up to 6 conversion types. Choose conversion type using the *Conversion type* combo box.

Monolithic binary file to Turbo 2000 or Super Turbo. Conversion of a monolithic binary file (see 1.8). The input file must be a monolithic binary file. This is a natural usage of Turbo 2000 or Super Turbo systems.

ChainLoading. Conversion of segmented binary file to *chain of Turbo 2000 or Super Turbo files*. A special binary loader that loads the files is prepended before the chain. The binary file can have up to 252 segments. Use the *Check loader* button to determine whether the special binary loader will be destroyed by the binary file.

BlockLoading. Conversion of segmented binary file to *chain of Turbo 2000 or Super Turbo blocks*. A special binary loader that loads the blocks (and accommodates information where to place the blocks in the memory) is prepended before the chain. The binary file can have up to 62 segments. Use the *Check loader* button to determine whether the special binary loader will be destroyed by the binary file.

Tokenized BASIC to Turbo 2000 or Super Turbo. Conversion of tokenized BASIC program. Note that many loaders cannot load such program directly. To circumvent such limitation, you can embed tokenized BASIC to binary file. See section 6.3.

Plain DATA to Turbo 2000 or Super Turbo. Conversion of plain data.

Binary file to binary turbo. Conversion of binary file to Binary turbo (file type 4) format. This conversion type is available only for the Turbo 2000 plugin. VisiCopy III loader is able to load and execute such file, but it must not contain any INIT segments.

8.3 Header Block

Use the text fields *File name, Type, Load, Length* and *Run* to specify entries of the Turbo 2000 or Super Turbo header block.

Click the *Auto set header* button to automatically enter values in the text fields mentioned above. The information is obtained by analysis of the input file and the selected conversion type. Furthermore, if the input file is not a monolithic binary file and according to the selected conversion type it should be, a warning dialog with recommendations is displayed.

Click the *Load header* to load the turbo header from an external file. The header is loaded automatically if it is stored in a file that is named same as the input file followed by a `.theader` suffix. In other cases, a file chooser is displayed.

8.4 Inserting Silence After INIT Segments

Fill-in the text field *Silence list* with comma separated decimal numbers to specify generated silence. Every number is interpreted as number of seconds of silence to be inserted after corresponding INIT segments. This is applicable only for ChainLoading and BlockLoading conversion types.

8.5 Prepending Universal Turbo Loader

If you do not have a cartridge with Turbo 2000 or Universal Turbo loader, you can prepend a cassette version of such loader. Set the *Prepend Universal Turbo loader* configuration entry to *true*.

8.6 Configuration Entries

Turbo 2000/Header block pilot tone length
Number of pilot tone pulses of the header block (256-8192)

Turbo 2000/Data block pilot tone length
Number of pilot tone pulses of the data block (256-8192)

Turbo 2000/Pilot tone length for BlockLoading
Number of pilot tone pulses of data blocks for BlockLoading conversion type (256-8192)

Turbo 2000/Prepend Universal Turbo loader
Prepend Universal Turbo loader as a standard cassette boot file

Super Turbo/Header block pilot tone length
Number of pilot tone pulses of the header block (256-8192)

Super Turbo/Data block pilot tone length
Number of pilot tone pulses of the data block (256-8192)

Super Turbo/Pilot tone length for BlockLoading
Number of pilot tone pulses of data blocks for BlockLoading conversion type (256-8192)

Super Turbo/Prolongate pilot tone
Increase number of pilot tone pulses with increasing approximate baud rate

Super Turbo/Prolongate pilot tone for BlockLoading
Increase number of pilot tone pulses with increasing approximate baud rate for BlockLoading conversion type

Super Turbo/Prepend Universal Turbo loader
Prepend Universal Turbo loader as a standard cassette boot file

9 Turbo 2000 - Kilobyte Blocks

9.1 Characteristics

Plugin converts files to **Turbo 2000 - kilobyte blocks** Czechoslovak turbo system. There are no special restrictions on input files.

9.2 User Interface

The user interface is depicted on figure 8.

The text field *File name* corresponds to the same field of the Turbo 2000 - kilobyte blocks header block. The purpose of the text field *Silence list* is described in section 8.4.

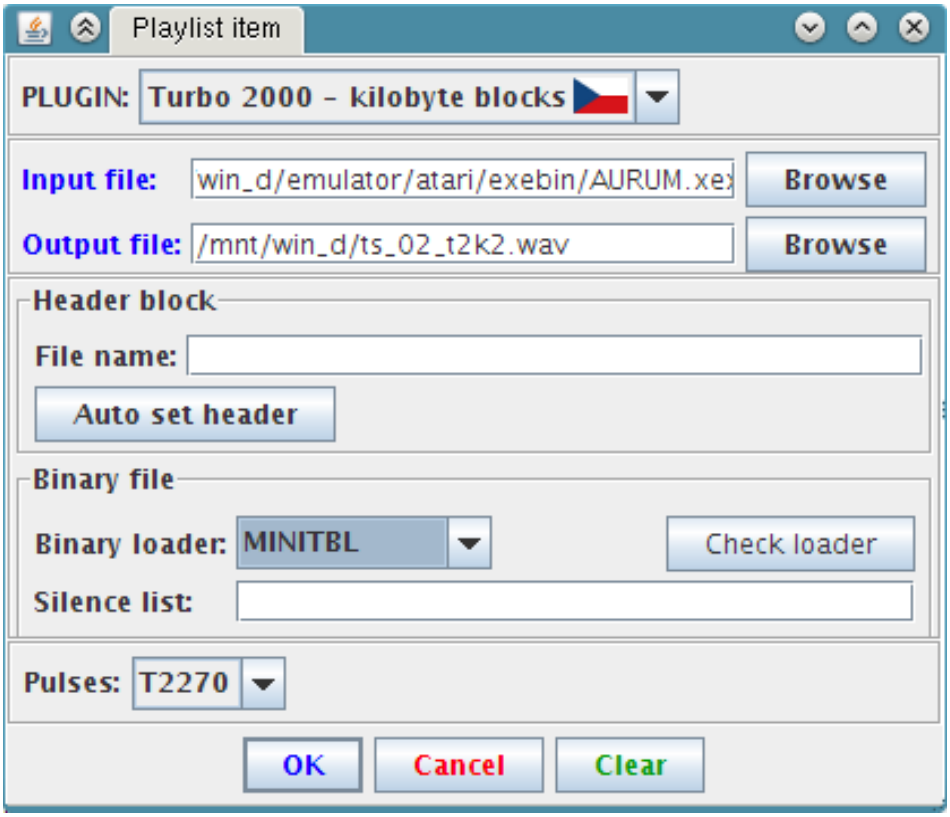

Figure 8: Turbo 2000 - kilobyte blocks

9.3 Binary Loaders

Turbo 2000 - kilobyte blocks turbo system can accommodate binary files. You can prepend one of the special miniature built-in binary loaders. For this turbo system, you can choose MINITBL, NANOTBL, NANOTBL[UR] or NANOTBL[U2] loader by using the *Binary loader* combo box. The loader is converted to the Turbo 2000 system.

MINITBL loader is a combination of stripped"T:" device handler that allows only READ operation and code for binary load using CIO. NANOTBL loaders are single-purpose binary loaders reading data blocks, not

using CIO at all. NANOTBL stores data blocks to the freely available RAM. NANOTBL[UR] stores the data blocks to the beginning of the "RAM under ROM", NANOTBL[U2] to the end of "RAM under ROM".

Use the *Check loader* button to verify whether the binary file will destroy the selected binary loader.

9.4 Prepending Universal Turbo Loader

If you don't have a cartridge with Turbo 2000 or Universal Turbo loader, you can prepend a cassette version of such loader. Set the *Prepend Universal Turbo loader* configuration entry to *true*.

9.5 Configuration Entries

Turbo 2000 - kilobyte blocks/Header block pilot tone length
Number of pilot tone pulses of the header block (256-8192)

Turbo 2000 - kilobyte blocks/Data block pilot tone length
Number of pilot tone pulses of the data blocks (256-8192)

Turbo 2000 - kilobyte blocks/Loader header block pilot tone length
Number of pilot tone pulses of the header block of the binary loader (256-8192)

Turbo 2000 - kilobyte blocks/Loader data block pilot tone length
Number of pilot tone pulses of the data block of the binary loader (256-8192)

Turbo 2000 - kilobyte blocks/Silence after header
Number of seconds of the silence inserted after the header block (0-30)

Turbo 2000 - kilobyte blocks/Silence after loader
Number of seconds of the silence inserted after the binary loader (0-30)

Turbo 2000 - kilobyte blocks/Name loader same as file
The binary loader will have name same as the converted file

Turbo - kilobyte blocks/Prepend Universal Turbo loader
Prepend Universal Turbo loader as a standard cassette boot file

10 B-TAPE

10.1 Characteristics

Plugin converts files to **B-TAPE** Czechoslovak turbo system. There are no special restrictions on input files. B-TAPE system is backward compatible with Turbo Tape system used by TT-DOS operating system.

10.2 User Interface

The user interface is depicted on figure 9.

The text field *File name* corresponds to the same field of the B-TAPE data block. The purpose of the text field *Silence list* is described in section 8.4. Use the *Tape mode* combo box to select tape mode.

10.3 Binary Loaders

B-TAPE turbo system can accommodate binary files. You can prepend one of the special miniature built-in binary loaders. For this turbo system, you can choose NANOBTAPE, NANOBTAPE[UR] or NANOBTAPE[U2] loader using the *Binary loader* combo box. The loader is converted to the Turbo 2000 system.

NANOBTAPE loaders are single-purpose binary loaders reading blocks, not using CIO at all. NANOBTAPE stores the blocks to freely available RAM. NANOBTAPE[UR] stores the blocks to the beginning of the "RAM under ROM", NANOBTAPE[U2] to the end of "RAM under ROM".

Use the *Check loader* button to verify whether the binary file will destroy the selected binary loader.

10.4 Prepending Universal Turbo Loader

If you don't have a cartridge with Turbo 2000 or Universal Turbo loader, you can prepend a cassette version of such loader. Set the *Prepend Universal Turbo loader* configuration entry to *true*.

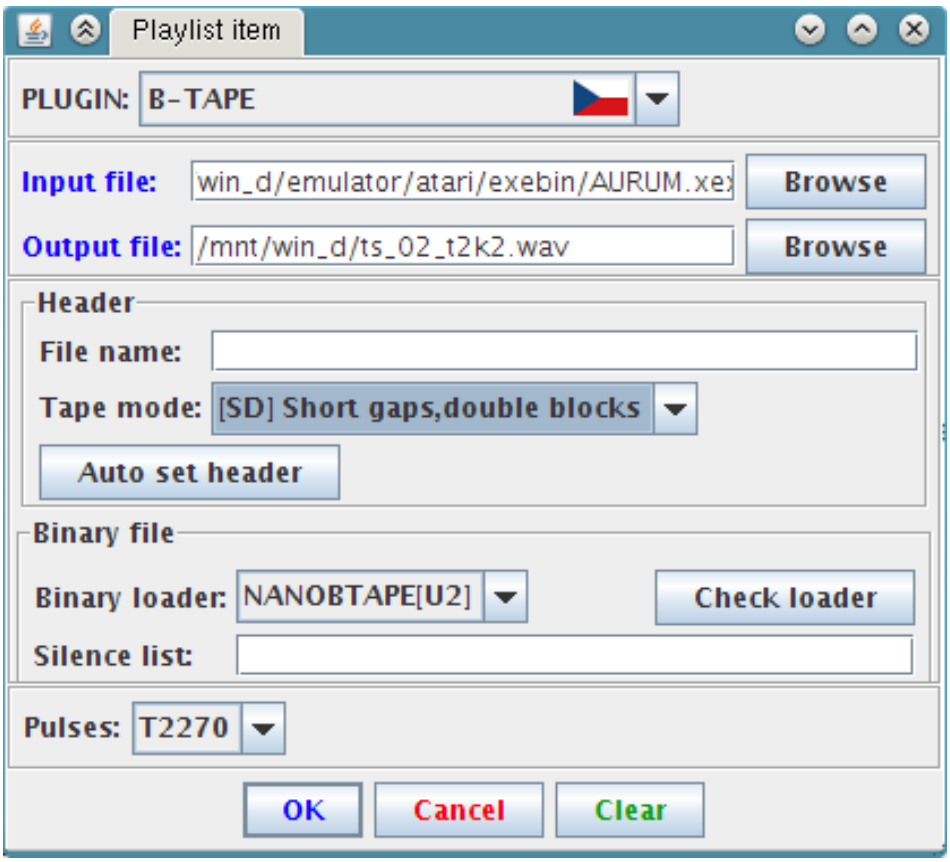

Figure 9: B-TAPE

10.5 Configuration Entries

B-TAPE/Pilot tone length
Number of pilot tone pulses of the blocks (256-8192)

B-TAPE/Prolongate pilot tone
Increase number of pilot tone pulses with increasing approximate baud rate

B-TAPE/Loader header block pilot tone length
Number of pilot tone pulses of the header block of the binary loader (256-8192)

B-TAPE/Loader data block pilot tone length
Number of pilot tone pulses of the data block of the binary loader (256-8192)

B-TAPE/Silence after first block
Number of seconds of the silence inserted after the first B-TAPE block (0-30)

B-TAPE/Silence after loader
Number of seconds of the silence inserted after the binary loader (0-30)

B-TAPE/Name loader same as file
The binary loader will have name same as the converted file

B-TAPE/Force tr2k tape image chunks
The generated trXX type tape image chunks will be `tr2k` instead of `trbt`. This allows the CAS COM utility to transfer files in the B-TAPE format.

B-TAPE/Prepend Universal Turbo loader
Prepend Universal Turbo loader as a standard cassette boot file

11 KSO Turbo 2000

11.1 Characteristics

Plugin converts files to **KSO Turbo 2000** Polish turbo system. There are no special restrictions on input files. KSO Turbo 2000 system is compatible with various similar turbo systems (e.g. Turbo 2000F).

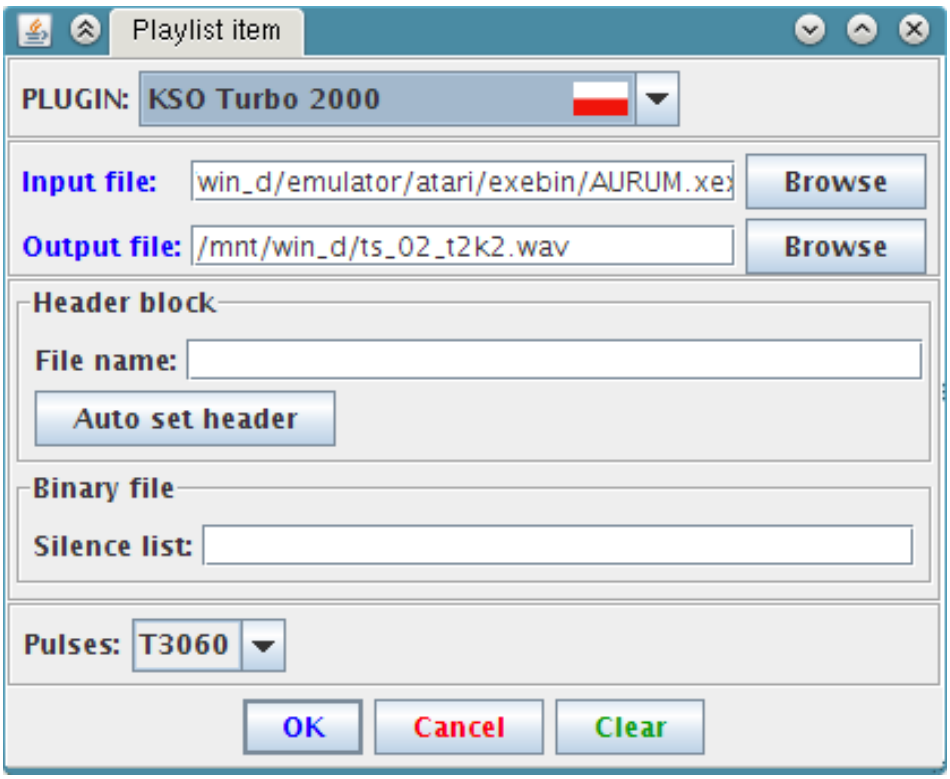

Figure 10: KSO Turbo 2000

11.2 User Interface

The user interface is depicted on figure 10.

The text field *File name* corresponds to the same field of the header block. The purpose of the text field *Silence list* is described in section 8.4.

11.3 Configuration Entries

KSO Turbo 2000/Header block pilot tone length
Number of pilot tone pulses of the header block (256-8192)

KSO Turbo 2000/Data block pilot tone length
Number of pilot tone pulses of the data blocks (256-8192)

KSO Turbo 2000/Invert polarity of pulses
Invert polarity of pulses

12 Turbo Blizzard

12.1 Characteristics

Plugin converts files to **Turbo Blizzard** Polish turbo system. There are no special restrictions on input files.

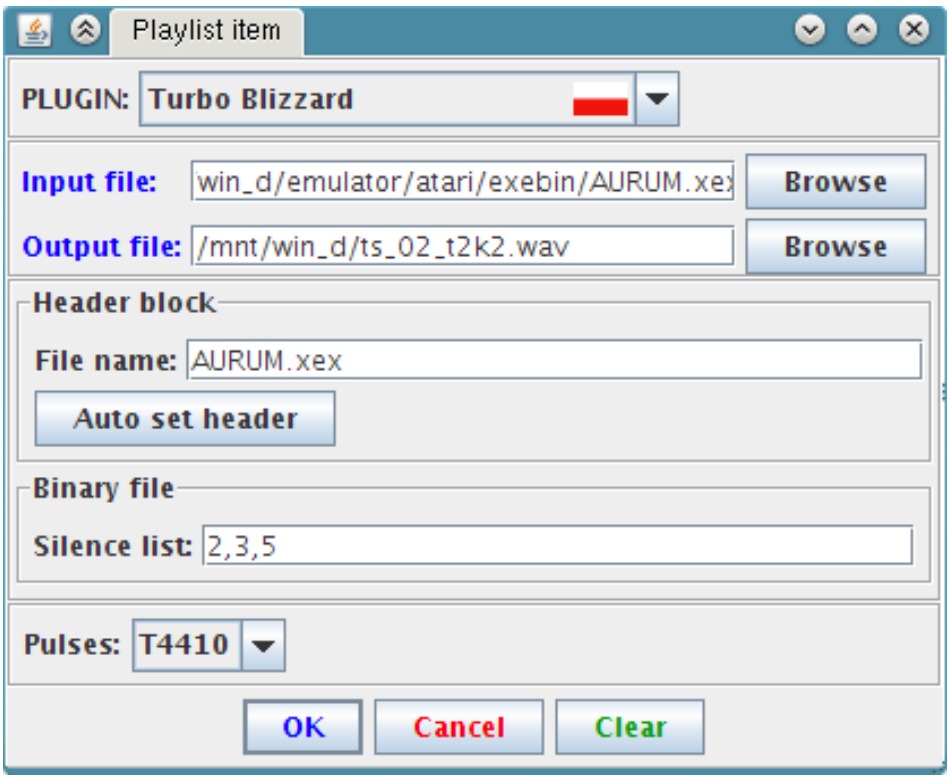

Figure 11: Turbo Blizzard

12.2 User Interface

The user interface is depicted on figure 11.

The text field *File name* corresponds to the same field of the header block. The purpose of the text field *Silence list* is described in section 8.4.

12.3 Configuration Entries

Turbo Blizzard/Invert polarity of pulses
Invert polarity of pulses

Turbo Blizzard/Long gaps between blocks
Make long gaps between blocks

13 Turbo ROM

13.1 Characteristics

Plugin converts files to **Turbo ROM** Polish turbo system. The following input files are supported:

- Turbo ROM compatible binary files
- Binary files
- Tokenized BASIC files

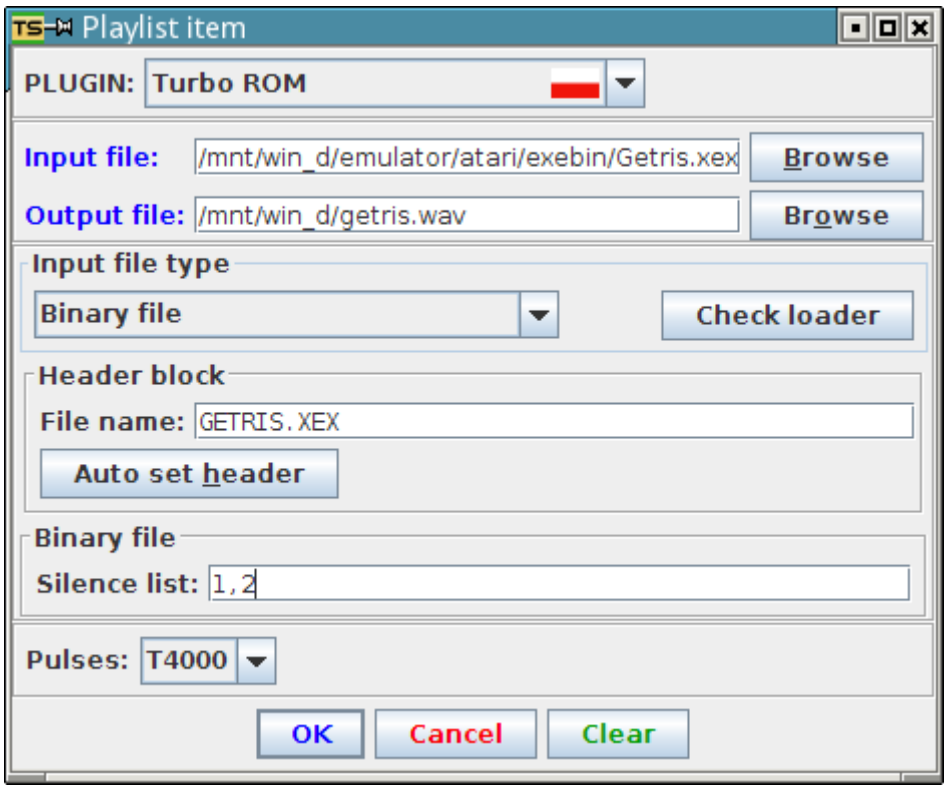

Figure 12: Turbo ROM

13.2 User Interface

The user interface is depicted on figure 12.

The text field *File name* corresponds to the same field of the header block. Use the *Auto set header* button to automatically set file name and also to verify whether the input file is Turbo ROM compatible binary file.

Use the *Input file type* combo box to select input file type.

13.3 Conversion of Turbo ROM Compatible Binary Files

These binary files consist of exactly one DATA segment and at most one RUN segment and at most one INIT segment. Files are converted to the Turbo ROM natural format.

13.4 Conversion of Binary Files

13.4.1 Processing

Binary Files are converted to a special format. A special binary loader in Turbo ROM natural format is prepended. Binary files are normalized before conversion (RUNINIT segments are split, RUN segment is moved to the end of the file).

13.4.2 User Interface

Use the *Check loader* button to verify whether the binary file will destroy the binary loader. The purpose of the text field *Silence list* is described in section 8.4.

13.5 Conversion of Tokenized BASIC Files

Files are converted to Turbo ROM natural format for tokenized BASIC files.

13.6 Prepending Loaders

If you do not have a cartridge with Turbo ROM loader, you can prepend a cassette version of such loader. Set the *Prepend Turbo ROM loader*, and *Convert binary loader to a standard cassette boot file* configuration entries to *true*.

13.7 Configuration Entries

Turbo ROM/Invert polarity of pulses
Invert polarity of pulses

Turbo ROM/Header pilot tone length
Number of pilot tone pulses of the header block (1024-8192)

Turbo ROM/Data block pilot tone length
Number of pilot tone pulses of the data block (128-2048)

Turbo ROM/Binary load pilot tone length
Number of pilot tone pulses of blocks when binary files are converted

Turbo ROM/Silence after header
Number of seconds of the silence inserted after the header block

Turbo ROM/Silence after binary loader
Number of seconds of the silence inserted after binary loader

Turbo ROM/Convert binary loader to a standard cassette boot file
If set to true, the binary loader is converted to a standard cassette boot file instead of a Tubo ROM compatible binary file.

Turbo ROM/Prepend Turbo ROM loader
Prepend Turbo ROM loader as a standard cassette boot file

Turbo ROM/Use diagnostic Turbo ROM loader
If set to true, a diagnostic version of the Turbo ROM loader will be prepended. The diagnostic version of the loader displays check sum, last used buffer address, and hexadecimal dump of the header block. Value of this configuration entry is honored only when the Prepend Turbo ROM loader entry is set to true.

Turbo ROM/LOAD address of a BASIC program
Base address to which BASIC programs will be loaded

Turbo ROM/RUN address of a BASIC program
Address of a routine that executes BASIC programs (default is 41086)

14 Atari Super Turbo

14.1 Characteristics

Plugin converts files to **Atari Super Turbo** Polish turbo system. Conversion of binary files to the following formats is supported:

- AST format that can accommodate binary files that have up to 44 segments and no INIT segments.
- BUT format that can accommodate binary files that have up to 254 segments of any type. A generic BUT loader is used to load such files. The loader occupies addresses 1926 - 2185.

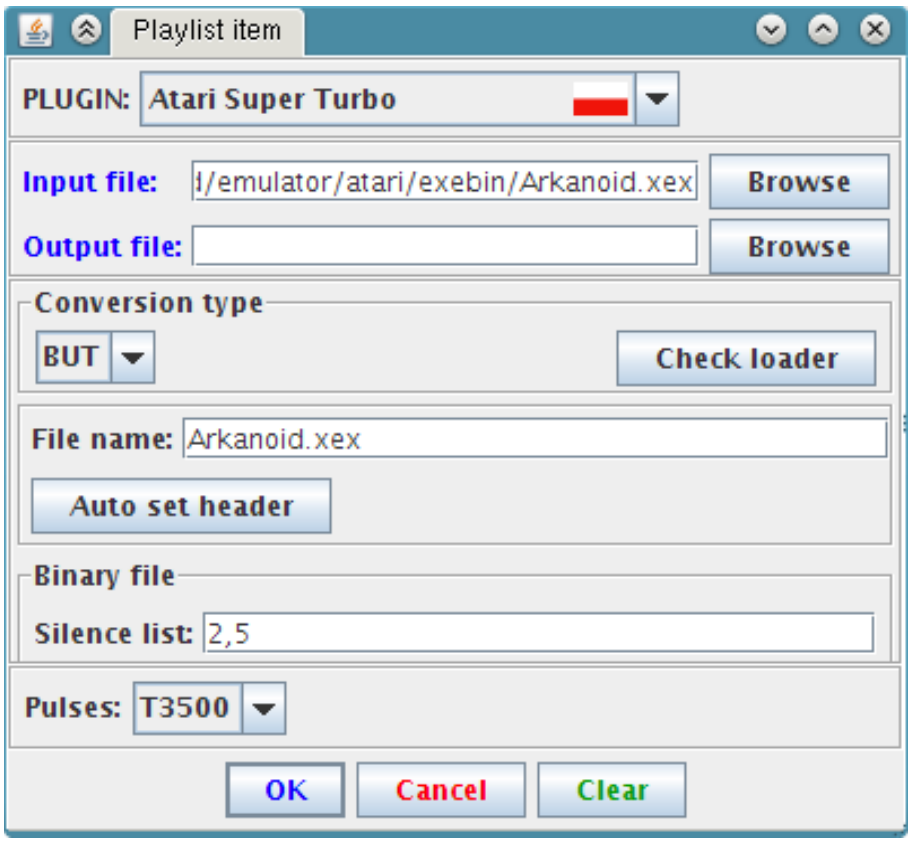

Figure 13: Atari Super Turbo

14.2 User Interface

The user interface is depicted on figure 13.

The text field *File name* corresponds to the same field of the header block. The *Auto set header* button can be used to automatically set file name and also to verify whether the input file is compatible with the selected format (AST or BUT). The purpose of the text field *Silence list* is described in section 8.4.

14.3 Configuration Entries

Atari Super Turbo/Invert polarity of pulses
Invert polarity of pulses

Atari Super Turbo/AST header pilot tone length
Number of pilot tone pulses of the AST header block (256-16384)

Atari Super Turbo/Data block pilot tone length
Number of pilot tone pulses of data blocks (256-16384)

Atari Super Turbo/BUT loader will wait for any key
Determines whether the BUT loader will require a key press to proceed

Atari Super Turbo/Silence after BUT loader
Duration of silence generated after BUT loader

15 Hard Turbo

15.1 Characteristics

Plugin converts files to the **Hard Turbo** Polish turbo system. According to the nature of this turbo system, input files must be binary files.

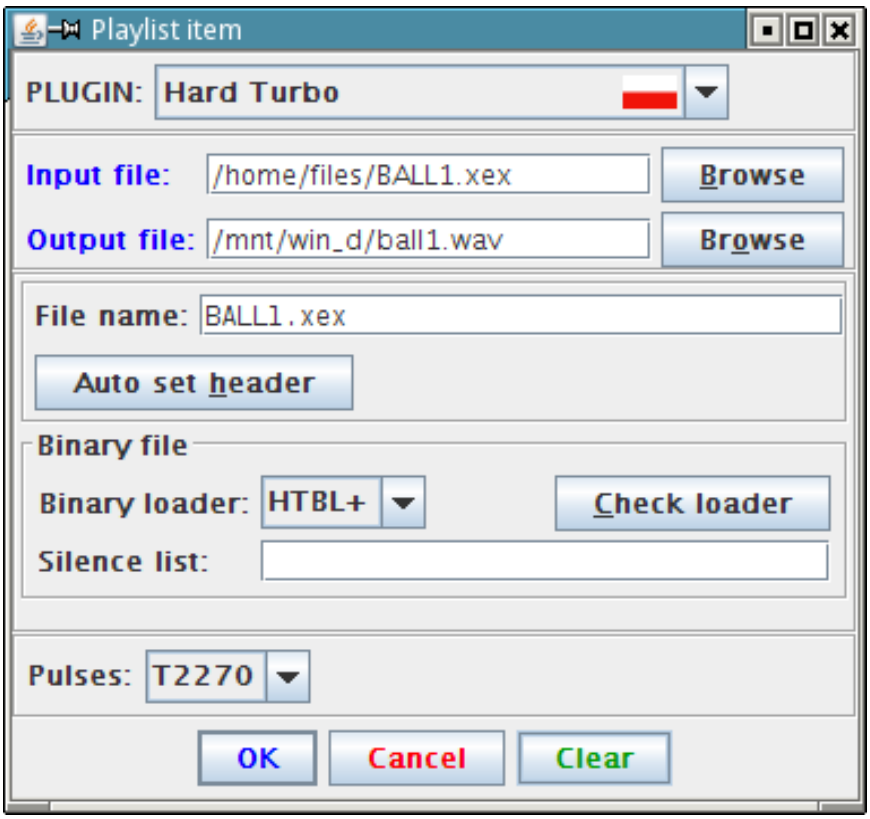

Figure 14: Hard Turbo

15.2 User Interface

The user interface is depicted on figure 14.
The text field *File name* corresponds to the same field of the main header block. Use the *Auto set header* to automatically set file name. The purpose of the text field *Silence list* is described in section 8.4.

15.3 HTBL+ Loader

You can add HTBL+ loader before the binary file converted.
HTBL+ is an extended binary loader that has the following enhancements

- Data recorder motor is switched off after segment data is loaded. This enables loading of binary files that execute lot of code from INIT segments
- RUNAD is set in a way that allows RUN segment to be the first segment of the binary file
- The loader never waits for any key under normal circumstances. This makes the loader easier to use with devices that do not support MOTOR CTRL signal
- The loader forces warm start during its initialization. This allows to load and run programs that rely on the display configuration set up by the warm start routines
- BASIC is automatically disabled

To add the HTBL+ loader, select HTBL+ from the *Binary loader* check box. Use the *Check loader* button to verify whether the binary file will destroy the HTBL+ loader.

15.4 Configuration Entries

Hard Turbo/Invert polarity of pulses
Invert polarity of pulses
Hard Turbo/Header block pilot tone length
Number of pilot tone pulses of the header block (256-16384)

Hard Turbo/Data block pilot tone length
Number of pilot tone pulses of data blocks (256-16384)

Hard Turbo/Name loader same as file
The binary loader will have name same as the converted file

16 Lower Silesia Turbo 2000

16.1 Characteristics

Plugin converts binary files to the **Lower Silesia Turbo 2000** turbo system from Poland.

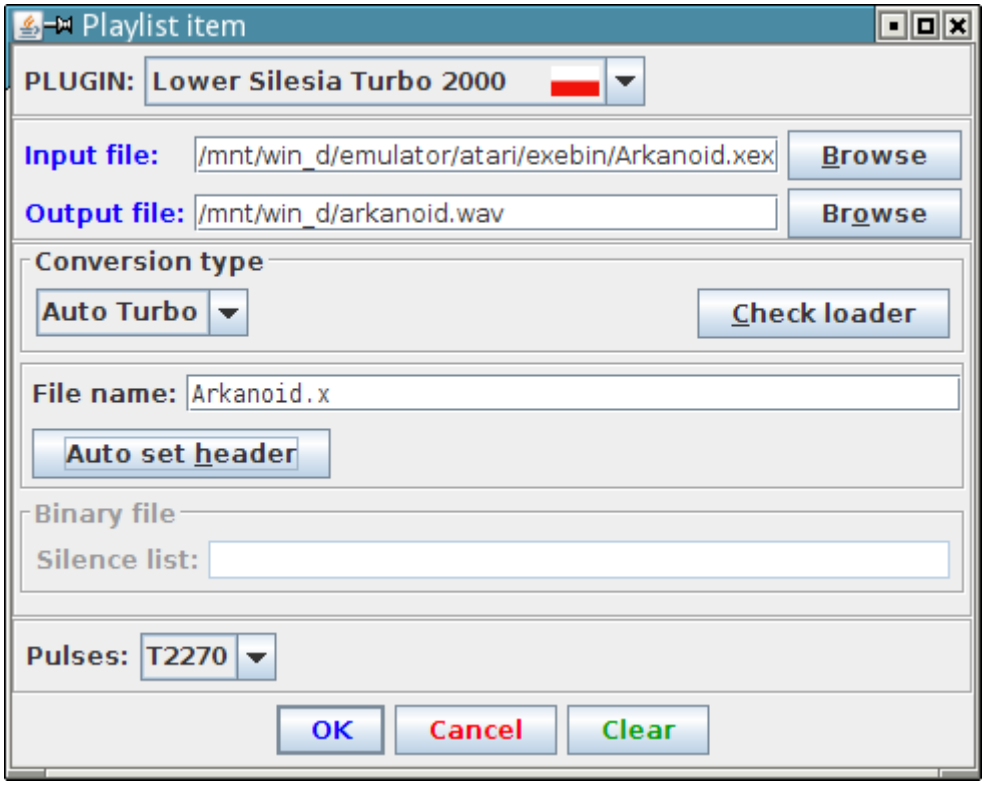

Figure 15: Lower Silesia Turbo 2000

16.2 Conversion Types

Two conversion types are supported.

Auto Turbo. Conversion of input binary file to the Auto Turbo format (see section 22.6.2). The Auto Turbo format can accommodate any binary file, but only binary files that have no INIT segments can be loaded and executed.

UE Protected Binary. Conversion of input binary file to the Protected Binary Format (see section 22.6.3) use by the "Unknown Exterminator" copier. A binary loader is converted to the Auto Turbo format. Then the input binary file is converted to the Protected Binary Format.

16.3 User Interface

The user interface is depicted on figure 15.

The text field *File name* corresponds to the same field of the header block. Click the *Auto set header* button to automatically set the file name. Select conversion type using the *Conversion type* box. The purpose of the text field *Silence list* is described in section 8.4.

Click the *Check loader* button to check whether the selected input file is compatible with Auto Turbo or binary loader.

16.4 Configuration Entries

Lower Silesia Turbo 2000/Header block pilot tone length
Number of pilot tone pulses of the header block (256-8192)

Lower Silesia Turbo 2000/Data block pilot tone length
Number of pilot tone pulses of the data block (256-8192)

Lower Silesia Turbo 2000/Segment header pilot tone length
Number of pilot tone pulses of the segment header block (128-768)

Lower Silesia Turbo 2000/Silence after loader
Duration of silence generated after binary loader

17 Standard

17.1 Characteristics

Plugin converts input files to standard tape records. Transfer speed is always 600 bd. The following input files are supported:

- Monolithic binary files
- Segmented binary files
- Tokenized BASIC files and BASIC source code files
- Plain data files

17.2 User Interface

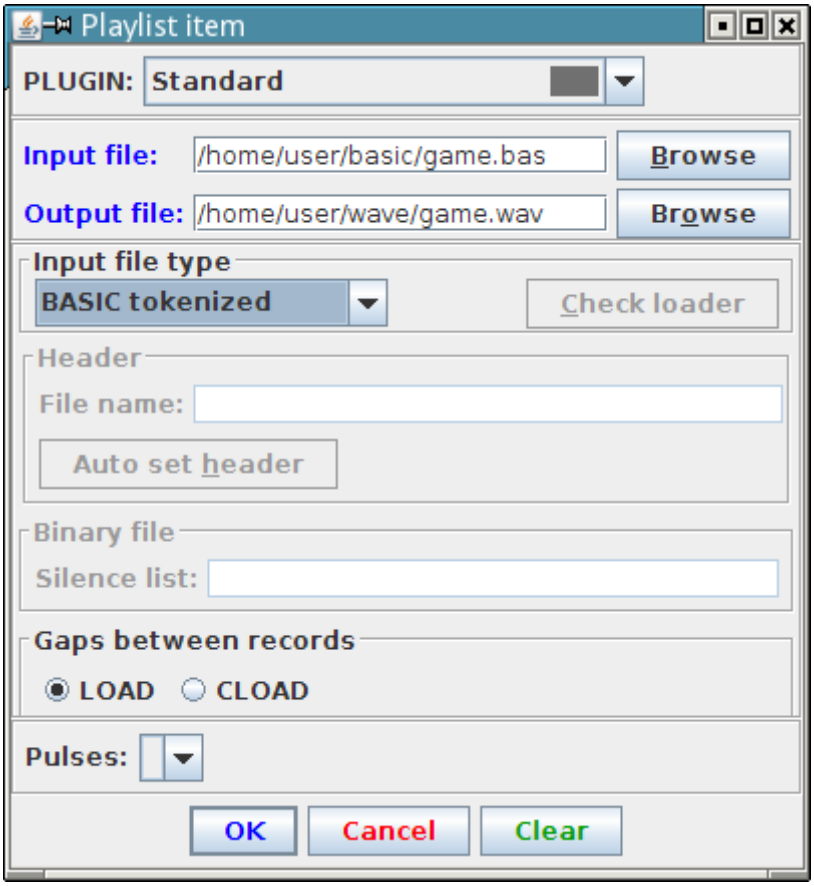

Figure 16: Standard

User interface is depicted on figure 16.

Use the *Input file type* combo box to select input file type. Use the *Gaps between records* box to set duration of gaps between records. Use the *Check loader* button to verify whether the binary file will destroy a binary loader.

17.3 Conversion of Monolithic Binary Files

Monolithic binary files are converted to boot files. The following limitations apply for monolithic binary files:

- The DATA segment cannot occupy memory below address 1536
- Length of the DATA segment must not exceed 32649 bytes

17.4 Conversion of Segmented Binary Files

17.4.1 Processing

Segmented binary files are converted as follows:

- A binary loader skeleton is customized and converted to a boot file. Binary loader switches motor off after every INIT segment.
- The binary file is normalized (RUNINIT segments are split, RUN segment is moved to the end of the file)
- Segments of the binary file are converted to a sequence of files. A new file with long IRG is created after every INIT segment. This allows proper execution of the INIT segments.

- Duration of inter-record gaps is 375 ms by default. This gives the loader time to process the data loaded.

The following limitations apply for segmented binary files:

- Maximum number of segments is 254
- Binary files must not destroy the binary loader

17.4.2 User Interface

Use the *File name* text field to set name that will be displayed by the binary loader. The purpose of the text field *Silence list* is described in section 8.4. Use the *Check loader* button to verify whether the binary file will destroy the binary loader.

17.5 Configuration entries

Standard/Leader duration
Duration of the leader signal in seconds (10-30).

Standard/Binary loader IRG duration
Duration of the inter-record gaps in milliseconds for standard tape records generated when a segmented binary file is converted. Default value is 375.

Standard/IRG after INIT segment duration
Duration of the inter-record gaps after INIT segments in seconds when a a segmented binary file is converted. Default value is 12.

18 Tape Image

Plugin interprets tape images.

18.1 Turbo Records

TURGEN SYSTEM supports **trXX** chunks and **pwmX** chunks.

18.2 Standard Tape Records

Since version 8.3.7, **baud** and dependent **data** chunks are supported. Since version 8.3.8, **fsk** chunks are supported.

Part III
TURBO SYSTEMS

19 Introduction

This part contains information about various turbo systems that have been created in Czechoslovakia and Poland. Descriptions of the systems are sufficient to provide ability to write turbo generators, loaders, copiers and turbo decoders.

20 Information encoding

Information is encoded using pulse width modulation (PWM). In this document, width of pulse is defined as a distance between two transitions from logical zero to logical one. See figure 17.

Figure 17: Pulse and its width

21 Turbo Systems from Former Czechoslovakia

21.1 Common Information

This common information is related only to turbo systems described in this document. These turbo systems are or were widespread in former Czechoslovakia, but of course, there are or were also others.

21.1.1 Switching Data Recorder to Turbo Mode

Switching data recorder to turbo mode is done using electronic switch, that switches to turbo mode if signals COMMAND and MOTOR CTRL are active. Reading of data is done by monitoring signal at pin DATA-IN of the SIO connector. Writing of data is performed by direct change of logical value at DATA-OUT pin of the SIO connector.

21.1.2 Single Purpose or CIO

Two main kinds of Czechoslovak turbo systems can be distinguished:

1. Systems that use single purpose loaders and simple file format that can accommodate only one contiguous block of data (Turbo 2000, Super Turbo). These systems are unable to support binary load operation, although capable to accommodate binary load files themselves.
2. Systems that were created together with tape operating systems: Turbo 2000 - kilobyte blocks, Turbo Tape, B-TAPE. Aim of these systems is to partially or fully replace the disk drive.

21.1.3 Pilot Tone and Data Separation

All Czechoslovak turbo systems use same way how to separate pilot tone and data. One sync pulse (which is very narrow pulse) or narrow pulse.

21.2 Turbo 2000

21.2.1 Description

Turbo 2000 was the first turbo system available in former Czechoslovakia. It was developed in 1987 by Jiří Richter, student of Czech Technical University in Prague and member of Prague Atari user club.

Four types of pulses are distinguished: Narrow pulse, wide pulse, pilot tone pulse and sync pulse. Bits are stored in MSB to LSB order.

File stored using Turbo 2000 system consists of two blocks - header block (HEADER) and data block (DATA).

Both blocks are preceded by pilot tone (series of at least 256 pilot tone pulses) which is followed by sync pulse (which is very narrow pulse). Recommended number of pilot tone pulses is at least 2000 in order to provide compatibility with "Universal turbo" loaders.

21.2.2 Header block

Offset	Description
0	Always 0
1	File type (1 - plain data, 3 - program, 4 - binary file, 255, 254 - tokenized BASIC)
2-11	File name
12-13	Load address
14-15	Length of file
16-17	Start address
18	Check sum = (HEADER[0]) xor ... xor (HEADER[17])

21.2.3 Data block

Offset	Description
0	Always 255
1-?	Data itself
Last	Check sum = (DATA[0]) xor ... xor (DATA[?])

21.2.4 Timing

Standard transfer speed is approximately 2270 bauds. There is a big tolerance for width of all pulses.

Pulse	Center width	Tolerated range
Pilot tone	32/44100 s	(25-47)/44100 s
Wide	26/44100 s	(20-40)/44100 s
Narrow	13/44100 s	(6-19)/44100 s
Sync	10/44100 s	(4-17)/44100 s

21.2.5 Loaders

In early times, Turbo 2000 loaders were distributed on tapes, stored using standard 600 baud system. Then loaders on cartridges became widespread.

21.3 Super Turbo

21.3.1 Description

Super turbo system is an enhancement of Turbo 2000 system, developed by Jiří Richter. Supported transfer speeds are approximately from 2725 to 6411 bauds.

Two types of pulses are distinguished: Narrow pulse and wide pulse. Bits are stored in MSB to LSB order.

File stored using Super Turbo consists of two blocks - header block (HEADER) and data block (DATA). Both blocks are preceded with pilot tone (series of at least 1200 wide pulses) which is followed by narrow pulse. Recommended number of pilot tone pulses is at least 2000 in order to provide compatibility with available loaders.

21.3.2 Header block

Offset	Description
0	Always 183
1	File type (1 - plain data, 3 - program, 4 - binary file, 255 - tokenized basic)
2-21	File name
22-23	Load address
24-25	Length of file
26-27	Start address
28	Check sum = (HEADER[0]) xor ... xor (HEADER[27])

21.3.3 Data block

Offset	Description
0	Always 237
1-?	Data itself
Last	Check sum = (DATA[0]) xor ... xor (DATA[?])

21.3.4 Timing

Various transfer speeds are supported. For given speed, width of wide pulse is width of narrow pulse simply doubled.

Pulse	Width
Wide	(6/44100 - 22/44100) s
Narrow	(3/44100 - 11/44100) s

21.3.5 Loaders

Special loaders for Super Turbo only are very rare. So called “Universal turbo” loaders capable to load both Turbo 2000 and Super Turbo are widespread, mostly on cartridges. Universal turbo loaders measure speed using examination of three consequent pilot tone pulses. Measured speed is also used to distinguish between Turbo 2000 and Super Turbo.

During the time, so called “Visiloader” has been invented. This loader is able to display progress of loading using PMG.

21.4 Turbo 2000 - kilobyte blocks

21.4.1 Description

Turbo system designed together with various versions of "Turbo operating system" (TOS).

Four types of pulses are distinguished: Narrow pulse, wide pulse, pilot tone pulse and sync pulse. Bits are stored in MSB to LSB order.

File stored using this turbo system consists of many blocks. First block is a header block (HEADER), remaining blocks (BLOCK) are data blocks that have 1026 bytes. Every block is preceded by pilot tone (series of at least 256 pilot tone pulses) which is followed by one sync pulse. Recommended number of pilot tone pulses is at least 2000 in order to provide compatibility with available loaders.

21.4.2 Header block

Offset	Description
0-1	Always 0
2-17	File name
18	Check sum = HEADER[0] xor HEADER[1] xor ... xor HEADER[17]

21.4.3 Data blocks

Offset	Description
0	255=Full block, 250=EOF block. Numbers 251-254 indicate partially filled block. If we subtract 251 from this number, we obtain a difference that will be denoted as Z. This difference can be 0,1,2 or 3 and represents higher byte of number of valid data bytes in the block.
1-1024	Data itself. If the block is a full block, there is 1024 bytes of data. If the block is EOF block, there are all zeros. If the block is a partially filled block, there is data padded with zeros up to the length of 1023 bytes. Last byte is lower byte of number of valid data bytes in the block. If we denote this lower byte as X, the number of valid bytes in the block is $Z * 256 + X$
1025	Check sum = BLOCK[0] xor BLOCK[1] xor ... xor BLOCK[1024]

21.4.4 Timing

Standard transfer speed is approximately 2270 bauds. There is a big tolerance for width of pulses.

Pulse	Center width	Tolerated range
Pilot tone	32/44100 s	(25-47)/44100 s
Wide	26/44100 s	(20-40)/44100 s
Narrow	13/44100 s	(6-19)/44100 s
Sync	10/44100 s	(4-17)/44100 s

21.4.5 Loaders

This turbo system is integrated into to various version of "Turbo operating system" (TOS) and also to one special version of TURBO BASIC XL widely used in former Czechoslovakia. Usual CIO device installed is T: or D:. CIO functions supported are OPEN, READ, WRITE and CLOSE.

21.5 Turbo Tape

21.5.1 Description

Advanced turbo system introduced with TT-DOS operating system sold by JRC company. Aim of this turbo system is to fully replace disk drive. Usual CIO device installed is B: or D:.

Four types of pulses are distinguished: Narrow pulse, wide pulse, pilot tone pulse and sync pulse. Bits are stored in MSB to LSB order.

File stored using Turbo Tape consists of blocks (BLOCK) that are 1026 bytes long. Every block is preceded by pilot tone (series of at least 256 pilot tone or wide pulses) which is followed by one sync or narrow pulse. Recommended number of pilot tone or wide pulses is at least 2000 in order to provide compatibility with available loaders.

21.5.2 Tape modes

Mode	Description
SS	Short gaps between blocks. First block is written twice, others once.
SD	Short gaps between blocks. All blocks are written twice.
LS	Long gaps between blocks. First block is written twice, others once.
LD	Long gaps between blocks. All blocks are written twice.

D modes provide data redundancy for safe data storage. First block is always written twice in order to provide convenient support for READ DIRECTORY CIO function.

21.5.3 Structure of the blocks

Offset	Description
0	Sequential number of the block. Numbering starts from 1. In case of D modes, pairs of blocks have same sequential number.
1	Tape mode: SS=128, LS=0, SD=192, LD=64
2,3	Bits 0-11: Offset of last valid byte in the block (16-1024), this offset will be denoted as B. Bit 15: Last block flag. Byte at offset 2: $B\%256$ Byte at offset 3: $[B/256] + [128 * (EOF\ is\ true)]$
4	Undefined number. All blocks of one file should have same number here.
5	Undefined.
6-16	File name and extension padded with spaces. First 8 bytes are devoted to file name, last 3 bytes are devoted to extension.
17-1024	Data itself (1008 bytes). Data must be padded with any bytes.
1025	Check sum = BLOCK[0] xor BLOCK[1] xor ... xor BLOCK[1025]

21.5.4 Timing

Timing is compatible with Turbo 2000 and Super Turbo.

21.6 B-TAPE

21.6.1 Description

Advanced turbo system introduced with B-TAPE extension for operating system BW-DOS. Aim of this extension is to fully replace disk drive. B-TAPE allows to use both CIO and SIO to exploit data recorder turbo modification. The disadvantage is a big size of the device handler. B-TAPE was designed as improvement of Turbo Tape system.

Four types of pulses are distinguished: Narrow pulse, wide pulse, pilot tone pulse and sync pulse. Bits are stored in MSB to LSB order.

File stored using B-TAPE consists of blocks (BLOCK) that are 1026 bytes long. Every block is preceded by pilot tone (series of at least 256 pilot tone or wide pulses) which is followed by one sync or narrow pulse.

Recommended number of pilot tone or wide pulses is at least 2000 in order to provide compatibility with available loaders.

21.6.2 Tape modes

Mode	Description
SS	Short gaps between blocks. First block is written twice, others once.
SD	Short gaps between blocks. All blocks are written twice.
LS	Long gaps between blocks. First block is written twice, others once.
LD	Long gaps between blocks. All blocks are written twice.

D modes provide data redundancy for safe data storage. First block is always written twice in order to provide convenient support for READ DIRECTORY CIO function.

21.6.3 Structure of the blocks

Offset	Description
0	Sequential number of the block. Numbering starts from 1. In case of D modes, pairs of blocks have same sequential number.
1	Tape mode: SS=128, LS=0, SD=192, LD=64
2,3	Bits 0-11: Offset of last valid byte in the block (16-1024), this offset will be denoted as B. Bit 15: Last block flag. Byte at offset 2: $B\%256$ Byte at offset 3: $[B/256] + [128 * (EOF\ is\ true)]$
4	Undefined number. All blocks of one file should have same number here.
5	Random number. All blocks of one file must have same number here. This random number allows to distinguish files with same file name.
6-16	File name and extension padded with spaces. First 8 bytes are devoted to file name, last 3 bytes are devoted for extension.
17-1024	Data itself (1008 bytes). Data must be padded with zeros if needed.
1025	Check sum = BLOCK[0] xor BLOCK[1] xor ... xor BLOCK[1025]

21.6.4 Timing

Timing is compatible with Turbo 2000 and Super Turbo.

21.6.5 Notes

B-TAPE device handler is able to read files stored using Turbo Tape system. In order to circumvent problems with big size of the device handler, special minimalistic binary loader called MICROB was shipped with B-TAPE.

22 Turbo systems from Poland

22.1 KSO Turbo 2000 and Turbo 2000F

22.1.1 Description

Turbo systems used in Poland, originally designed together with KSO Turbo 2000 (Tape operating system) and then adopted for other tape operating systems.

Three types of pulses are distinguished: Pilot tone pulse, wide pulse and narrow pulse. Bits are stored in MSB to LSB order.

File stored using KSO Turbo 2000 system consists of blocks. First block is a header block (HEADER), other blocks are data blocks (DATA).

Every block is preceded by pilot tone (series of pilot tone pulses).

Signal is expected on some pin of joystick port (KSO Turbo 2000) or on DATA-IN pin of the SIO connector (Turbo 2000F).

22.1.2 Header block

Offset	Description
0	Always 0
1	Always 255
2-11	File name (10 characters)
12	Check sum = (HEADER[0] + HEADER[1] + ... HEADER[11]) mod 256

22.1.3 Data block

Offset	Description
0-1	Number of valid bytes in the block, 0-3072.
2-3073	Up to 3072 bytes of data padded with zeros if needed.
3074	Check sum = (BLOCK[0] + ... + BLOCK[3073]) mod 256

File ends with data block that has less than 3072 valid bytes. If the total file size can be divided by 3072 without a remainder, file must end with block that has 0 valid bytes.

22.1.4 Timing

Pulse	Width
Pilot tone	44/44100 s
Wide	22/44100 s
Narrow	11/44100 s

22.2 Atari Super Turbo (AST)

22.2.1 Description

Turbo system used in Poland suitable for accommodation of binary files. Please note that the following information is incomplete.

Three types of pulses are distinguished: Wide pulse, narrow pulse and STOP pulse. Bits are stored in LSB to MSB order.

There are three main file formats: AST, BUT, and BOT. All three formats use blocks that have the following structure:

1. Pilot tone (narrow pulses)
2. Narrow pulse
3. Data (wide and narrow pulses)
4. Terminator 1(4 narrow pulses)
5. Terminator 2 (128 STOP pulses)

22.2.2 AST File Format

This file format can be used to store binary files that have up to 44 segments and no INIT segments. File consists of several blocks.

1. Header block, 256 bytes of data
2. Data blocks. For each segment of the binary file, there is one data block.

22.2.3 BUT File Format

This file format can be used to store binary files. File consists of several blocks.

1. BUT loader header block. 6 bytes of data. It appears to be a boot header.
2. BUT loader data block. 576 bytes of data.
3. Pairs of data blocks for each segment. First block in a pair is a segment header, second block in a pair holds segment data.
4. Termination segment header[1]

22.2.4 AST Header Block

Offset	Description
0	Number of segments (1-44)
1	Header check sum Check sum = (HEADER[2] xor HEADER[3] +xor... HEADER[255])
2-177	44 segment headers (start and end addresses)
178	Always 102
179	Always 85
180-199	File name. Internal code is used.
200	Always 0
201-244	Check sums for each segment block Check sum = (DATA[1] xor DATA[2] +xor... DATA[?])
245-255	Unknown. $70, $4b, $00, $d6, $47, $b4, $cf, $4b, $00, $d6, $41.

22.2.5 BUT Loader Header Block

Offset	Description
0	Always 0
1	Length of the BUT loader data block in 128 bytes blocks. Always 5.
2,3	Loader load address. 1920.
4,5	Unknown. 182, 9.

22.2.6 BUT Loader Data Block

Offset	Description
0-575	BUT loader code

22.2.7 AST and BUT Data Block

Offset	Description
0-?	Data

22.2.8 BUT Termination Segment Header

Offset	Description
0-3	255, 255, 255, 255

[1] TS is using slightly different format. There is no termination segment header, but the loader knows the number of segments.

22.2.9 Timing

Pulse	Width
Wide	22/44100 s
Narrow	12/44100 s
Stop	72/44100 s

22.3 Turbo Blizzard

22.3.1 Description

Turbo system used in Poland, suitable for accommodation of binary files (small blocks, high transfer speed).

Three types of pulses are distinguished: Pilot tone pulse, wide pulse and narrow pulse. Bits are stored in MSB to LSB order.

File stored using Turbo Blizzard system consists of the following blocks.

1. Synchronization block. This block is preceded with pilot tone (3072 pilot tone pulses) and two narrow pulses. Block does not accommodate any data. Block is followed with silence lasting for 0.1 second.
2. Header block (HEADER). This block is preceded with pilot tone (302 pilot tone pulses) and two narrow pulses. Then 78 bytes of data follow. Block is followed with silence lasting for at least 3 seconds.
3. One or more data blocks. These blocks are preceded with pilot tone (302 pilot tone pulses) and two narrow pulses. Then 1029 bytes of data follow. Data blocks are separated by short gaps.

22.3.2 Header block

Offset	Description
0-75	File name
76	Check sum = (HEADER[0] + HEADER[1] + ... HEADER[75]) mod 256
77	Spare byte, always 0. This byte is never read by the loaders.

22.3.3 Data block

Offset	Description
0-1	Number of valid bytes in the block, 0-1024.
2-1025	Up to 1024 bytes of data padded with zeros if needed.
1026	Always 0
1027	Check sum = (HEADER[0] + HEADER[1] + ... HEADER[1026]) mod 256
1028	Spare byte, always 0. This byte is never read by the loaders.

22.3.4 Timing

Pulse	Width
Pilot tone	24/44100 s
Wide	12/44100 s
Narrow	8/44100 s

22.4 Turbo ROM

22.4.1 Description

Turbo system used in Poland, with limitations similar to Czechoslovak Turbo 2000 and Super Turbo Systems. This turbo system can accommodate Turbo ROM compatible binary files (these binary files consist of exactly one DATA segment at most one RUN segment and at most one INIT segment) or tokenized BASIC programs.

Three types of pulses are distinguished: Wide pulse, narrow pulse and stop pulse. Bits are stored in LSB to MSB order.

File stored using Turbo ROM system consists of two blocks.

1. Header block (HEADER). This block is preceded with pilot tone (4884 wide pulses) and one narrow pulse. Then 41 bytes of data follow. After header block, there are four wide pulses and one stop pulse.
2. Data block. This block is preceded with pilot tone (516 wide pulses) and one narrow pulse. Then data follow. Data blocks are separated by short gaps. After data block, there are four wide pulses and one stop pulse.

22.4.2 Header Block for Binary Files

Offset	Description
0	Header block check sum = (HEADER[1] ... xor ... HEADER[40])
1-2	Header load address (1537)
3-4	Header length excluding first byte (40)
5	Data block check sum = (DATA[0] xor DATA[1] xor ... DATA[?])
6-7	RUN address
8-9	INIT address
10-11	Load address
12-13	Data block length
14	Padding 0
15-34	File name. Internal code is used.
35	Program type flag. For binary files there is 1.
36	0 - JSR to the INIT address, 1 - No JSR to the INIT address
37-39	Padding zeros
40	RTS opcode (96)

22.4.3 Data Block for Binary Files

Offset	Description
0-?	Data bytes

22.4.4 Header block for BASIC files

A BASIC file consist of two parts. A header part (14 bytes long) and a main part. Data from the header part is stored in the header block and data from the main part is stored in the data block. The tokenized form of the BASIC file is relocatable and can be loaded to any address. Such address is denoted as base address.

Offset	Description
0	Header block check sum = (HEADER[1] ... xor ... HEADER[77])
1-2	Header load address (1537)
3-4	Header length excluding first byte (77)
5	Data block check sum = (DATA[0] xor DATA[1] xor ... DATA[?])
6-7	RUN address (41086). Points to a routine in the BASIC ROM.
8-9	INIT address. Always 0 for a BASIC program.
10-11	LOAD address. Calculated as base address + VNT.
12-13	Data block length. Length of the main part of the BASIC file.
14	Padding 0
15-34	File name. Internal code is used.
35	Program type flag. Always 0 for a BASIC program.
36	1 - No JSR to the INIT address
37-39	Padding zeros
40-59	Routine that copies the header part to the zero page `0xA2, 0x11, 0xBD, 0x3C, 0x06, 0x95,0x80, 0xCA, 0x10,0xF8,0x60`
60	Header part of the BASIC file (LOMEM, VNT, VNTE, VVT, STMTAB, STMCUR, STARP)
74-75	Address right past the loaded BASIC file (LOAD address + length of the main part)
76-77	Address right past the loaded BASIC file (LOAD address + length of the main part)

22.4.5 Data Block for BASIC Files

Offset	Description
0-?	Main part of the BASIC file

22.4.6 Timing

Pulse	Width
Wide	16/44100 s
Narrow	6/44100 s
Stop	48/44100 s

22.5 Hard Turbo

22.5.1 Description

Turbo system used in Poland that can accommodate only binary files.

Three types of pulses are distinguished: Pilot tone pulse, wide pulse, and narrow pulse. Bits are stored in MSB to LSB order.

File stored using Hard Turbo system consists of many blocks. There are three types of blocks.

1. Main header block (HEADER). This block is preceded with pilot tone (at least 512 pilot tone pulses) and one narrow pulse. Then 41 bytes of data follow.
2. Segment header block (SEGHEAD). This block is preceded with pilot tone (at least 512 pilot tone pulses) and one narrow pulse. Then 6 bytes of data follow.
3. Segment data block (DATA). This block is preceded with pilot tone (at least 512 pilot tone pulses) and one narrow pulse. Then data bytes follow.

A Hard Turbo file starts with the main header block. Then pairs of segment header and segment data blocks follow. The file ends with one special segment header block.

22.5.2 Main Header Block

Offset	Description
0	Identification byte, always 0
1-39	File name. ATASCII string terminated with EOL (155)
40	Check sum = (HEADER[0] xor HEADER[1] xor ... HEADER[39])

22.5.3 Segment Header Block

Offset	Description
0	Identification byte, always 255
1,2	First address of the following segment. Value of 65535 (255 255) indicates end of file
3,4	Last address of the following segment (increased by one)
5	Check sum = (SEGHEAD[0] xor SEGHEAD[1] xor ... SEGHEAD[4])

22.5.4 Segment Data Block

Offset	Description
0	Identification byte, always 255
1-?	Segment data
Last	Check sum = (DATA[0] xor DATA[1] xor ... DATA[?])

22.5.5 Timing

Pulse	Width
Pilot	34/44100 s
Wide	22/44100 s
Narrow	12/44100 s

22.6 Lower Silesia Turbo 2000

22.6.1 Description

This turbo system appears to be very similar to Turbo 2000 from former Czechoslovakia.

Four types of pulses are distinguished: Narrow pulse, wide pulse, pilot tone pulse and sync pulse. Bits are stored in MSB to LSB order.

There are several file formats used.

22.6.2 Auto Turbo Format

This format is used to store binary files with limitations. Binary files without INIT segments can be stored and executed. Binary files with INIT segments can be stored, but not executed.

File stored using Auto Turbo format consists of two blocks - header block (HEADER) and data block (DATA).

Both blocks are preceded by pilot tone (series of at least 256 pilot tone pulses) which is followed by a sync pulse (which is very narrow pulse). Recommended number of pilot tone pulses is at least 2000.

22.6.3 Unknown Exterminator Unprotected Binary Format and Protected Binary Format (UE UBF and UE PBF)

These file formats can be used to store binary files. A binary loader is prepended before the file. File consists of several blocks.

1. Dummy RUN block. 2 bytes of data (UE PBF only).
2. Pairs of data blocks for each segment. First block in a pair is a segment header (SEGHEAD), second block in a pair holds segment data (SEGDATA).
3. Termination segment header

22.6.4 Auto Turbo Header block

Offset	Description
0	Always 0
1	File type. Always 4.
2-11	File name
12-13	Always 0
14-15	Length of file
16-17	Always 0
18	Check sum = (HEADER[0]) xor ... xor (HEADER[17])

22.6.5 Auto Turbo Data block

Offset	Description
0	Always 255
1-?	Data of the binary file (includes segment headers)
Last	Check sum = (DATA[0] xor ... xor DATA[?])

22.6.6 UE PBF Dummy RUN block

Offset	Description
0,1	0x02 0xE0
2	Check sum = 0xE2

22.6.7 UE UBF and UE PBF Segment Header Block

Offset	Description
0,1	First address of the following segment
2,3	Last address of the following segment
4	Check sum = (SEGHEAD[0] xor ... xor SEGHEAD[3])

22.6.8 UE UBF and UE PBF Segment Data Block

Offset	Description
0-?	Data
Last	Check sum = (SEGDATA[0] xor ... xor SEGDATA[?])

22.6.9 UE UBF and UE PBF Termination Segment Header

Offset	Description
0-3	0x00,0x00,0x00,0x00
4	Check sum = 0x00

22.6.10 Timing

Standard transfer speed is approximately 2270 bauds. There is a big tolerance for width of all pulses.

Pulse	Center width	Tolerated range
Pilot tone	32/44100 s	(25-47)/44100 s
Wide	26/44100 s	(20-40)/44100 s
Narrow	13/44100 s	(6-19)/44100 s
Sync	10/44100 s	(4-17)/44100 s