DGV MANUAL

# Introduction

Dgv is a small program written in C that converts DAI programs (a vintage computer designed in 1978) into “optimized wav” files which can be loaded up to 7.7x faster than when playing original files from a cassette player. Files input can be “.wav” audio files or DAI “.dai” binary files.

Performances depends on the DAI version (the electronic design of the cassette input being different on a DAI V4 / V5 than on a DAI V7, see annex), on the player bandwidth and on the volume capabilities. In addition, speed factor will be limited to 4.8x unless the DAI is slightly modified at the hardware level.

WE CAN NOT CERTIFY THAT THE AUDIO DEVICE AND THE LEVEL YOU USE WILL NOT ARM YOUR DAI OR YOUR AUDIO DEVICE. USE THIS PROGRAM AT YOUR OWN RISK.

In particular, the old DAI operational amplifier IC14 used to process the analog signal, seems to be fragile. Therefore, it is recommended to control the audio player voltage output to avoid excessive levels.

See Bruno Vivien “Paradai” web site to get more information on audio format, “dai” files structure, program formats, DAI read functions, electronic schemes,..

<https://filedn.eu/l8fxcG16N8iQwXfMASdK9c4/DAI/plan.html>

# Functionalities

Dgv can convert real “wav” files into binary “dai” binary files or directly into optimized “wav” files. It can be run directly in windows with default options (converts all “wav” and “dai” files to optimized “wav” files) or within a terminal with options.

In a windows terminal, typical format will be:

**Dgv input\_file output\_file --options**

Detailed format can be obtained by typing “Dgv ?” to get help.

On windows 10 / 11, executing a program in a terminal requires to put “.\” before the name of the program.

## Converting “wav” files to “dai” files

### Valid input “wav” file

* “Wav” audio file with any sampling rate over 2400Hz, single or stereo channels (left channel is used), 8bits or 16bits
* From a K7 player or output by a DAI, by MAME emulator or by a Dgv program (“optimized wav” file). It automatically deals with the fact that MAME and DAI files polarity are inverted
* Supports the 3 DAI formats: Basic programs, Binary programs and Arrays

### Output “dai” file

* Binary “dai” format. This format contains all binary information of a program/data array plus the name and checksum information.

Ex: “Dgv Program.wav Pro.dai” will generate a “dai” Binary file “Pro.dai”; “Dgv \*.wav” or “Dgv \*.wav \*.dai” will convert all “wav” files in “dai” Binary files

## Converting “dai” files to optimized “wav” files

Input “dai” files correspond to binary file with the “.dai” extension.

Dgv “optimized wav” files can be played on the following DAI versions:

* DAI V4 / V5 or equivalent
* DAI V7 or equivalent
* Modified DAI V7, having only the LM324 amplifier IC14 replaced by a TLC274

Audio player: “optimized wav” signal depends largely on device output bandwidth (sampling rate being only an indication in practice) and driving power capabilities (o/w max voltage especially for the V4 version).

Output “optimized wav” files are generated according to the playing device and to the DAI hardware. By default, polarity of the signal is inverted for real devices and not inverted for the MAME emulator.

When output file name is not specified, or when “\*.wav” is given as a parameter, the output file name will be constructed with the input file name without its extension.

Default output file name structure is: “InputName\_DgvProfile--Options”, with the DgvProfile text depending on chosen profile version (see below). Options, which has the same structure as parameter options, is the result of profile default options and user requested options.

Ex: “Dgv Program.dai \*.wav --V4” will generate a “wav” file “Program\_DgvDaiV7C--V4MBIF384000.wav” optimized for a physical DAI V7 with the following characteristics: single channel, 1 byte per sample, inverted signal, sampling frequency of 384KHz.

# Dgv parameters

Format: “Dgv Source.xxx Destination.yyy –Options”

## File Source and destination names

Extensions xxx or yyy must be equal to “dai” or “wav”. Source & Destination names can be equal to ”\*.dai” or “\*.wav”.

If there is no InputName and no DestinationName, Dgv will convert any “wav” file to “dai” files and any “dai” file (including those created) to “optimized wav” files with all versions. Consequently, most conversions can be made without opening a terminal. This is equivalent to providing “\*.wav \*.wav” as parameters.

When no DestinationName is provided, it will be inferred from the Source name and the complementary extension will be used (“wav” -> “dai” or “dai” -> “wav”).

When generating “optimized wav” file, Dgv will add information to the name including files characteristics, unless a Destination name was provided.

## Options for “wav” output files

An optional argument can be used to modify “optimized wav” files characteristics and is formed with '--' followed by several options in any order: profile version depending on player and DAI version, “wav” sample length (1 or 2 bytes), “wav” sample frequency, signal polarity.

Optional argument example: “--V4MBIF384000”.

Providing the profile version is normally sufficient (format “Vx”). Other options are mainly used to deal with audio device constrains (Ex: an Olympus voice recorder LS-P2 cans only play 16bit stere files; some devices may invert the signal).

### Version, format “Vx”

‘x’ = number corresponding to the optimized “wav” file version associated with the DAI version and the player sampling frequency. If not specified, default value is V7.

Some audio players may not work at the given sampling frequency because actual bandwidth is much lower than it should. As “speed” / sample frequency increases, minimum electric signal level requirement increases. DAI V4 / V5 minimum electric level is significantly higher than for DAI V7.

Profiles have been tested with an USB-C Dongle “FiiO KA11 USB-C DAC dongle” on a PC and a Galaxy Note 10+ smartphone (no driver is required).

Profile version V0 corresponds approximately to the signal output of a real DAI.

Profile to be read on a DAI V4 or above, min +/-1.4V 0, 59% on PC with a dongle:

* V0 = DaiK7, mono, 8 bits, 24KHz, inversed polarity -> speed gain 1.0x
* V1 = DaiV4, mono, 8 bits, 48KHz, inversed polarity -> speed gain 3.7x

Profile to be read on a DAI V7 or above, min +/-0.45V, 31% on PC with a dongle:

* V2 = DAIV7A, mono, 8 bits, 48KHz, inversed polarity -> speed gain 3.7x
* V3 = DAIV7B, mono, 8 bits, 96KHz, inversed polarity -> speed gain 4.3x
* V4 = DAIV7C, mono, 8 bits, 384KHz, inversed polarity -> speed gain 4.8x

Profile to be read on a modified[[1]](#footnote-1) DAI V7, min +/-0.70V, 39% on PC with a dongle

* V5 = DAIV7T, mono, 8 bits, 192KHz, inversed polarity -> speed gain 6.3x
* V6 = DAIV7U, mono, 8 bits, 384KHz, inversed polarity -> speed gain 7.7x

Profile to be read on MAME emulator:

* V7 = MameA, mono, 8 bits, 96KHz, normal polarity -> speed gain 9.4x

### Channel count, format ‘M’ or ‘S’

‘M’ for Mono, ‘S’ for Stereo

### Sample size, format ‘B’ or ‘W’

‘B’ for 1 byte / 8 bits by sample, ‘W’ for 2 bytes / 16 bits by sample

### Signal polarity, format ‘N’ or ‘I’

‘N’ for normal polarity, i.e. TTL level / MAME polarity. ‘I’ for Inversed polarity, to be used on physical DAI as the amplification stage inverts the cassette signal

### Sampling frequency, format “Fxxxx” to “Fxxxxxxx”

“xxxxx” is the sampling frequency for output signal in Hz (4 to 7 characters). Specifying a higher frequency than the profile’s frequency (a lower one will not generate a functional file) is only useful for very specific reasons such making it playable on devices reading only specific formats.

### Profile version default optimized “wav” names

“Optimized wav” files are generated using profiles. Each versions have a default format which can be modified using options. When not specified by user, Dgv will add a text made of a profile version related text and a “wav” characteristics related text. This latter related text has the same structure as the optional argument.

Default profile version added text (i.e. when user only provides “--Vx" as option) is:

* V0 -> “\_DgvDaiK7--V0MBIF24000”
* V1 -> “\_DgvDaiV4--V0MBIF48000”
* V2 -> “\_DgvDAIV7A--V0MBIF48000”
* V3 -> “\_DgvDaiV7B--V0MBIF48000”
* V4 -> “\_DgvDaiV7C--V0MBIF384000”
* V5 -> “\_DgvDaiV7T--V0MBIF192000”
* V6 -> “\_DgvDaiV7U--V0MBIF384000”
* V7 -> “\_DgvMameA--V0MBIF96000”

These profiles have been validated on a single DAI V4 with a module to behave like a V7. This is not sufficient to be sure that it will work with all DAI (due notably to components dispersion) and audio players. Experiments have been limited and more efficient profiles can probably be found.

## Examples

* “Dgv Pacman.wav \*.dai”, generates a “Pacman\_Dgv.dai” binary file
* “Dgv Pacman.dai Pac.wav”, generates a“Pac.wav” wav file with profile V7 for MAME
* “Dgv \*.wav \*.wav --V4I” will convert all valid “wav” files (not having a profile name) to “optimized wav” files using the profile V4, to be normally loaded on a DAI V4, and make it readable on a MAME emulator, MAME having an inversed polarity.
* “Dgv Pacman.wav \*.wav --SWIV3F192000”, will create the “optimized wav” file “Pacman\_Dgv--V3SWIF192000.wav”, which corresponds to a V3 profile saved in stereo, 16 bits at 192KHz (instead of mono, 8 bits, 96KHz)
* “Dgv ?” will display help

# Dgv internal general principles

## Introduction

Following principles are just an introduction to Dgv internal functioning.

Please read code comments to have more details. A global understanding of DAI read functions and program structure is also necessary (see Bruno Vivien Paradai web site or DAI manuals).

To read “wav” files or to write optimized files, Dgv uses the main following principles:

* Simulate the exact timing of DAI read functions at the CPU cycle level. This is equivalent to what MAME emulator is doing.
* Adjust timing delays to offset DAI analog electronic (at least part of it) and audio player impacts
* Synchronize “wav” samples level to DAI read functions

This implies not only to simulate the DAI Read bit function, which reads the 4 TTL levels (i.e logic) of a bit (“Dai bit” in Dgv comments), but also functions at byte / block levels with delays between functions.

Depending on the used hardware (DAI version but also audio player), each TTL level transition is detected in a DAI using 4 loops. A minimum number of loops is required for different reasons.

Timings have also to be adjusted according to the analog input stage, notably due to the poor performances of the DAI operational amplifier (LM324) and to TTL trigger voltages (see below).

The minimum number of loops, and analog related timing offsets can be modified in the “Profile versions” using structures (DaiHW\_Profile[].DaiBitPeriods\_MinLoops[][] & DaiHW\_Profile[].PeriodsOffset\_HwDelay[]).

Interrupts, which only impact the “wav” Leader part of the program, are not simulated by default in Dgv (some basic simulation remains possible).

## Reading “wav” files

When reading a “wav” file into Dgv, it simulates DAI TTL input delays and read functions timings and assumes that the input stage does not introduce any delay. This is close to the behavior of a DAI V7 with a very performant operational amplifier.

Information is then stored in a structure which is equivalent to a “dai” file. In a debug mode, using Visual studio, it can be useful to find and correct reading errors from an analog cassette.

## Generating optimized “wav” files

“Wav” optimized files are generated using an internal representation of “dai” files.

Output required characteristics, are defined using a profile version and users’ options. However, output is driven essentially by the sampling frequency.

Dgv simulates DAI analog input delays and read functions timing in order to predict when “wav” signal level should be changed in order to be sure that previous TTL transition will be detected by the read function of a real DAI. Analog input delay is the consequence of the electronic input stage and of the performance of the audio device playing at a specific sampling frequency.

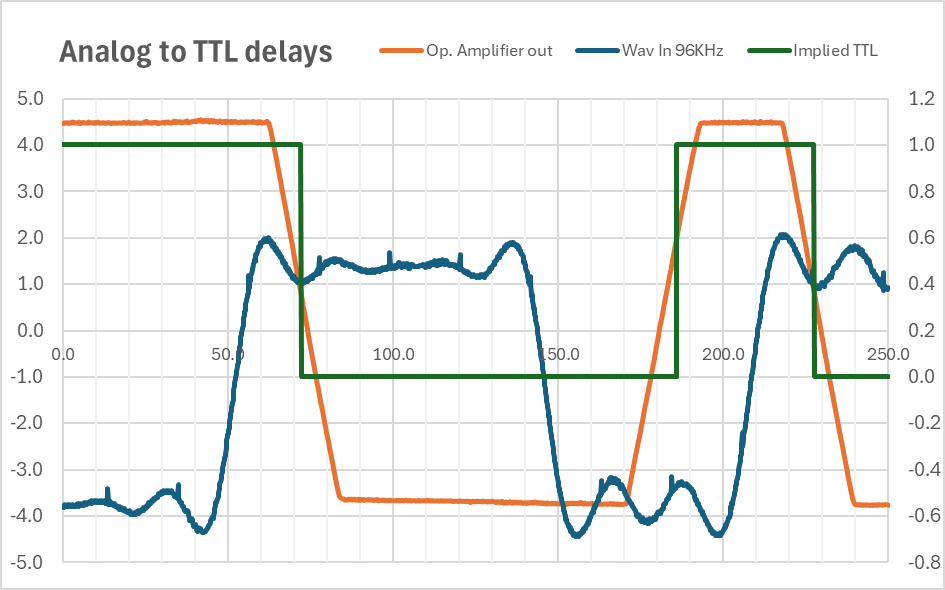
Reading high speed information requires to increase “wav” sampling frequency and to adjust the required loop count to be sure that all TTL transitions can be changed. This loop counts are specific to profile versions and are determined experimentally (using electronic simulation or a real circuit).

## Analog delay explanations

Following graph represents real measures on a DAI V7 with a LM324, when playing an optimized “wav” file at 48KHz.

Estimating precisely the timing of transition level depends mainly on the following stages:

1. “Wav” file sampling frequency and in some cases limitations of the playing device (o/w slew rate, output filter)
2. Transition delay of the DAI analog circuitry which is mainly linked to its operational amplifier slew rate (max voltage increase / us)
3. TTL transition voltage level, typically 0.8V for High to low and 2.0V for Low to TTL High



Time is in us, voltage in Volts. Sampling parasites are visible every 20us.

Another limitation to increasing speed comes from the fact that output stage needs to become negative (-3.8V) before being able to reach the 2.0V TTL transition level. LM324 is too slow for this and needs to be replaced by a faster version.

# Annex - schemes

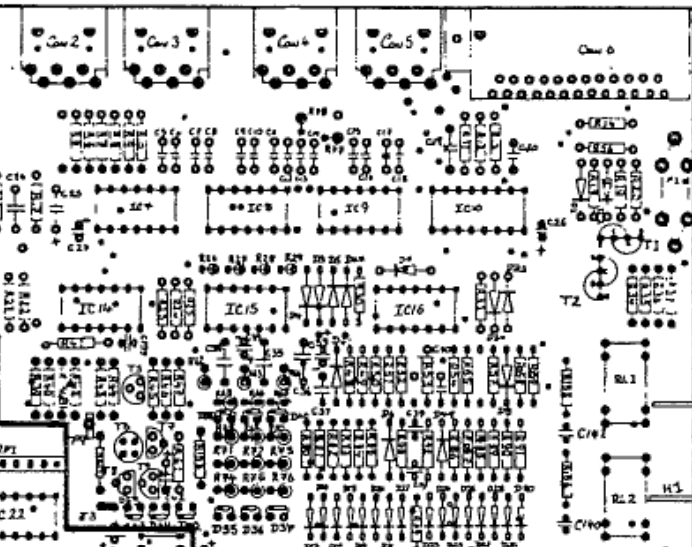
DAI version is written on the DAI PCB, on the right of the keyboard.

## A diagram of a circuit Description automatically generatedA computer circuit board with many wires Description automatically generatedDAI V4 scheme

## DAI V7 scheme

A diagram of a circuit

Description automatically generated



1. IC14 LM324 needs to be replaced by a more performant TLC274 [↑](#footnote-ref-1)