Formatting ADAM Digital Data Packs Using MAME's 'castool'

Thomas Cherryhomes <thom.cherryhomes@gmail.com>

November 10, 2023

Abstract

This document shows the process of applying formatting to suitable media without needing a donor tape to copy the formatting information. This process can be applied to either first-party media or to third-party media after the appropriate mechanical modifications are made to the cassette tape casing. This method requires only software to generate the required signal, and a suitable recording device to transfer the resulting signal to tape. Once transferred, the resulting tape can be used as any other digital data pack, for any required application. Applications for this method include lowest-common-denominator software distribution and reconstruction of degraded data packs so that they can be used again.

This paper is split into two sections, the first providing an overview of the format of Coleco Adam Digital Data packs, while the second provides the procedure for formatting a data pack using the MAME castool. The first section may be skipped if the procedure is all that is needed.

This represents the long tail of collaborative work across multiple projects to solve a long term problem in the Adam community, and this paper would not have been possible without the efforts of many.

Contents

List of Figures

List of Tables

List of Algorithms

Part I
An Overview of Digital Data Packs

Digital Data Packs are the standard data storage medium for the Coleco Adam computer. While the medium itself has a familiar form factor similar to audio cassettes, they not only have slight mechanical differences, but they also possess a block oriented formatting of header information that is spread across the tape so that the data drive can find a specific unit of information[1].

While Coleco had told early reviewers of the Adam that the ability to format data packs would be implemented in an upgrade, this never happened, and likely [2] was never intended to happen. With the short production lifetime of the system, there were a limited number of digital data pack medium produced for the system, with many of them being thrown into waste, meaning there currently exists a limited number of viable media in its original form.

To counteract this, the remaining users of the system modified existing audio cassettes so that they would fit into digital data pack drives, while also modifying data packs so that they could fit into audio cassette recorders. Once both cassettes were modified, the original data pack is dubbed onto the new modified cassette, copying the required address marks in the process. This process, while effective, relies on the integrity of the original donor cassette, and the effectiveness of the recorder. Over time and use, the donor media will become less effective at making new copies, and the generational copying effect present in dubbed copies further degrades the effectiveness of the resulting copies, and the donor-copy method that was devised by the Adam enthusiast community will eventually be ineffective.

This paper defines a method of creating the required waveform to be recorded onto a recipient data pack that does not require a donor pack. This is achieved entirely in software, and only requires a cas-

[1] defined as an addressable block of 1024 bytes

[2] speculation

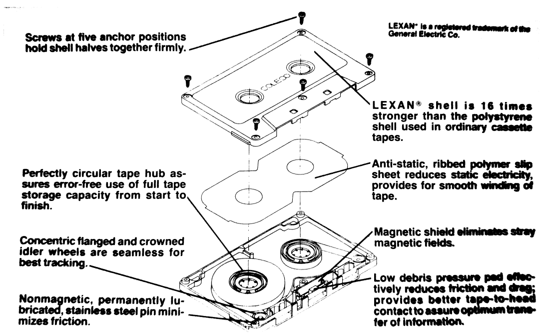

Figure 1: Physical Characteristics of a Digital Data Pack

sette recorder to transfer the resulting signal onto tape. The software takes as input, an image of data, and generates an audio file [3] which can be recorded. The software can produce data packs in either of the two possible formats [4], and the resulting pack can be used for any particular application.

But in order to understand how this is possible, an overview of the digital data pack is needed.

1 Physical Characteristics

The Adam's Digital Data Packs share physical characteristics with the Philips Compact Cassette. In particular, the size, shape, and mechanical tolerances are compatible. Because the data drive requires much faster transport of tape, the cassette shell was not only made of more duable Lexan material, but the mechanical tolerances of the rollers and spools are optimized for significantly less wow and flutter. Figure 1 is the insert that was packaged with each Coleco digital data pack, and shows the enhancements that Coleco added to deal with issues such as increased friction from increased speed.

Because a traditional capstan is not used to guide the tape against the tape head, additional holes are needed on the bottom of the cassette shell to stabilize the tape path against the static roller and head assembly on the bottom of the drive. In addition, there are two additional holes on the top left and right of the cassette shell on the back side of the cassette which serve to not only stabilize the cassette shell in the drive, but to ensure the cassette is only inserted in the front-facing manner.

The length of first-party Digital Data Packs is roughly equivalent to 25.6 minutes of audio tape, meaning that an audio cassette with a side length of 30 minutes of audio is ideal for use.

1.1 Adapting an existing audio cassette

The process of adapting an audio cassette shell to be used as a digital data pack is described in [5] on page 4, using a dremel or a drill with the appropriately sized bit to add the holes on the top and bottom of the cassette shell. The process of adapting an existing DDP to act as a donor, also present in [5] on page 3 are not needed. Any suitable[5] 60 minute cassette may be modified into a data pack.

2 Recorded Signal Characteristics

The pack topology is divided into two tracks, A and B, each with a very wide track width. The width is substantial enough to require recording the two seperate tracks on the two seperate sides, reversing track B to have the correct orientation.

The two tracks each contain a phase modulated square wave, with a nominal bit cell width of 70 microseconds. The presence of a transition at 31 microseconds within the bit cell indicates the presence of a binary one, while the absence of this additional transition represents a binary zero.

The tape transport travels at 80 inches per second, at an approximate 15,000 bits per second transfer rate [6], with a bit density of approximately 714.3 bits per inch.

[3] in Microsoft WAV format

[4] right and center

[5] FeO_2 Type-I e.g. Sony HF-60

[6] [3] page 69

# of bytes	Description
1	Sync Byte (0x16)
8	Header Bytes
21	zero bytes for padding
1	Sync Byte (0x16)
1024	Block Payload
21	zero bytes for padding
1	Sync byte (0x16)
2	XOR checksum
2	zero bytes for padding
922	sync bytes (0xAA)
2	zero bytes for padding

Table 1: DDP Block Format

```
<— TAPE MOTION (block #s in hex)
0              40              7F
=================================
80             C0              FF
```

Figure 2: GW Right Directory Block Layout

```
<— TAPE MOTION (block #s in hex)
40         7F  0               3F
=================================
80         BF  C0              FF
```

Figure 3: HE Center Directory Block Layout

2.1 Block Format

The complete format of each data block and its associated header and synchronization bytes are described in [6]. Each data block consists of the items in Table 1. It is important to understand that the header containing the addressing information, and the data payload are stored together in the same track, but the data drive can not create the headers itself, so they must already be present on the data pack.

2.1.1 Header Bytes format

The header is present in each data block, and provides not only the block number, but also the pack type, and track information, so that the type of pack can be determined on each and every read, to allow for spontaneous insertion of different pack types, and is listed in Table 2. The two pack formats for Right and Center directory are defined entirely in this header.

The header ID determines the pack type. The letters 'G' 'W' indicate that the pack is a right-directory pack, with blocks allocated sequentially starting from the beginning of the pack being block 0, and the end of the pack being block 255. This format is used by the Super Game tapes such as "Buck Rogers: Planet of Zoom." See Figure 2.

Conversely, the letters 'H' 'E' indicate that the tape has block 0 in the middle of the tape, with blocks going across the tape from that point, and wrapping back around to the middle. This format was used in the SmartBASIC, productivity programs, and blank data packs because it placed the directory block[7] equidistant in the middle of the pack. See Figure 3.

Two bytes are reserved for the block number. While this does mean that up to 65535 possible blocks could be placed on a data pack, were there enough physical tape, in practice only 256 total blocks are defined, so the second byte is empty. Since the data drive is controlled by a 6801[3], the byte order is big-endian for any block numbers.

An XORed copy of the block number is placed next, and is done such that adding the block and !NOT block numbers together should sum to zero.

This is followed by a single byte indicating the total number of blocks in a given track, and is sub-

[7] block number 1

# of bytes	Description
2	Header ID
2	Block Number
2	!NOT Block Number
1	Total Number of Blocks in Track
1	XOR checksum of above

Table 2: DDP Block Header Format

tracted against the total # of blocks to determine if a block is on track A or B, by the data drive firmware.

Finally, a one byte ones-complement checksum, the same as used by AdamNet[2] for each of its packets, is placed at the end of the header.

2.2 Track Topology

Each of the 256 possible blocks is assigned onto one of two possible tracks, A or B, with each track containing 128 blocks. The beginning of each track has approximately 2,753 leading zeroes to act both as padding and synchronization to find the beginning of a data track.

2.3 Tape layouts and block numbering

The different tape layouts all sequentially number the blocks in exactly the same way, with block 0 being at the beginning of the tape, and block 255 being at the end of the tape. The difference lies in not only the block header (HE vs GW), but also the fact that the tape firmware has an additional subroutine called MANGLE_NUM[8] which divides the total number of blocks in half, adds that to the current block number, and throws away the carry, causing the value to wrap back around to zero. See the listing for Algorithm 1

2.4 Signal level considerations

To best match the signal level emitted by the data drive when writing data payload, the sample level for both tracks is set to -12 decibels [6] and is static for the duration of the recording.

8[3] page 70

Algorithm 1 MANGLE_NUM

```
MANGLE_NUM
          TST      TAPE_TYPE_,D
; SEE WHERE THE DIRECTORY IS –
          BEQ      MANGL_END
; AT BEGINNING, GO AWAY
          LDD      BLOCKS_TRACK,D
          LSRD
; DIVIDE BY 2
          ADDD     WANTED_BLOCK,D
          STD      WANTED_BLOCK,D
          SUBD     BLOCKS_TRACK,D  ; NON-EXTANT?
          BHS      MANGL_HI
          RTS
; WE'RE OK, JUST RETURN
MANGL_HI
          LDD      WANTED_BLOCK,D  ; SUBTRACT
          SUBD     BLOCKS_TRACK,D
          STD      WANTED_BLOCK,D  ; NOPE
MANGL_END
          RTS
```

Part II
Data Pack Formatting Procedure

Formatting a data pack, while not trivial, is a straightforward procedure. It involves creating or using a source data pack image, with each of the 256 blocks sequentially placed in a file. This is known as .ddp format in the Coleco Adam community, and you can use any of the data pack image files present on sites such as the Adam Archive[8] as the source material for creating the signal file. Once the signal file is sourced, the 'castool[6]' present in MAME can be used to generate the initial signal file. An audio editing tool such as 'Audacity[7]' can then be used to do necessary edits before finally using a device such as a USB enabled cassette recorder[1] to transfer the data tracks onto a suitable cassette [9].

9[5] page 6

Linux is used in the examples. It is assumed that mame installed via your package manager, or placed somewhere within your search path.

3 Requirements

The requirements to perform the procedure are:

- Windows, Mac or Linux. Linux is shown here in this procedure
- The ability to compile C or C++ code. Windows can use the MAME windows binaries[4].
- A copy of MAME[4].
- An audio editing program, such as Audacity[7].
- A suitable cassette recording device[1].
- A dremel or drill with a 5/32 drill bit as specified in [5] page 4.
- A source DDP image, or a program to create an empty one, such as 'dd' in Linux.
- A blank USB stick large enough to hold the generated audio files from castool.

4 Modifying an Audio Cassette

Starting with a suitable 60 minute audio cassette, [10] drill out the required two holes on the back side of the tape, as specified in "Audio Cassette Tape to a Adam DDP Pack!!" [5] on page 4.

Verify that the newly drilled cassette will fit in your data pack drive, before proceeding.

5 Preparing the source image

A DDP image is required. At a minimum a file containing all zeroes for 262,144 bytes is sufficient, and can be created with the following command:

[10] e.g. Sony HF-60

```
$ dd if=/dev/zero of=blank.ddp bs=1024 count=256
```

On Windows, Programs such as Adam Image Manager, which is available on the Adam Archive[8] can be used instead.

6 Generating the audio signal file

With the source image in hand, the castool[6] from MAME[4] can be used to generate the resulting audio signal file in WAV format, with the following command:

```
$ castool convert ddp blank.ddp blank.wav
```

By default, castool will generate a center-directory audio image, with block 0 being in the middle of the data pack. Should you wish to generate a right-directory audio image, with block 0 being at the beginning of the tape, you will need to obtain the source code to mame, and modify lib/src/formats/adam_cas.cpp on line 137 from:

```
int layout_type = TYPE_HE;
```

to:

```
int layout_type = TYPE_GW;
```

and subsequently recompile mame with

```
$ make
```

7 Editing the audio signal file to fit on tape

The audio signal file, as it currently exists fresh out of castool, is formatted as a stereo audio file approximately 25.6 minutes long. Figure 4 shows the zoomed out waveform as it exists when imported into Audacity. It needs to be split into two seperate mono files and silence added to both ends of the file. In addition, the audio file for the second track must be reversed so that it is in the correct orientation for the data

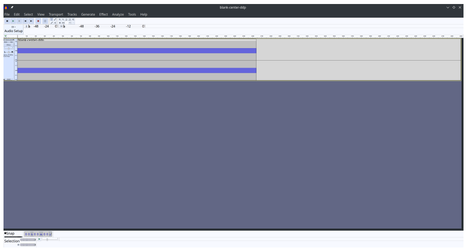

Figure 4: Imported audio signal file

drive's abnormally wide playback head, with the following procedure:

1. In the File menu, select Import > Audio...
2. Find and select blank.wav to open.
3. Once open, split the stereo file, into mono using the track menu, see Figure 5.
4. Click on the first track inside the waveform window. Then press Shift-Home to skip to beginning
5. From the Generate menu, select Silence... Specify 20 seconds. This will add the appropriate delay to pass the initial leader, and enough slew to allow the data drive to properly sense the beginning of the track.
6. Press Shift-End to skip to the end of the first track.
7. Repeat Generate > Silence..., and this time specify 6 minutes. This will add the appropriate padding to the end of the tape to fill out the 30 minutes.
8. Repeat steps 4-8 with the second track.
9. Press the Select button on the second track. See Figure X. This will select the entirety of track B.
10. In the Effect menu, select Special > Reverse. This will reverse the audio signal, and the second track, should be in the opposite orientation of the first.
11. Press the Select button of the first track.
12. From the File menu, select Export > Export Selected Audio
13. Ensure that the file format is 16-bit WAV, it should be, by default.
14. Save the file as track_a.wav.
15. Repeat steps 9 to 14, saving the resulting file as track_b.wav

The resulting image files are ready for recording, and can be loaded individually back into Audacity to verify, before recording. Verify that the correct amount of silence (20 seconds and 6 minutes) is present, and that it is reversed on track B.

The audio files track_a.wav and track_b.wav should now be readied to be recorded. If the USB enabled recorder is used as cited, the track_a.wav and track_b.wav files will be placed and recorded onto the recipient tape, one at a time.

8 Recording the Audio Signal

> The following procedure assumes that the cited QFX USB enabled tape recorder is being used. If another recorder is used, the instructions must be adjusted accordingly.

1. Ensure that the cassette recorder is plugged in, and switch is in the OFF position. Ensure that the volume knob is above 0. It does not matter where, as it is only monitoring the signal and not affecting how it is recorded.
2. Copy track_a.wav onto the USB stick.
3. Unmount the USB stick.
4. Slide the cassette into the recorder, side A up. Rewind to the beginning if necessary.
5. Place the USB stick into the USB port on the cassette recorder.
6. Press the record and play buttons to arm it for recording.

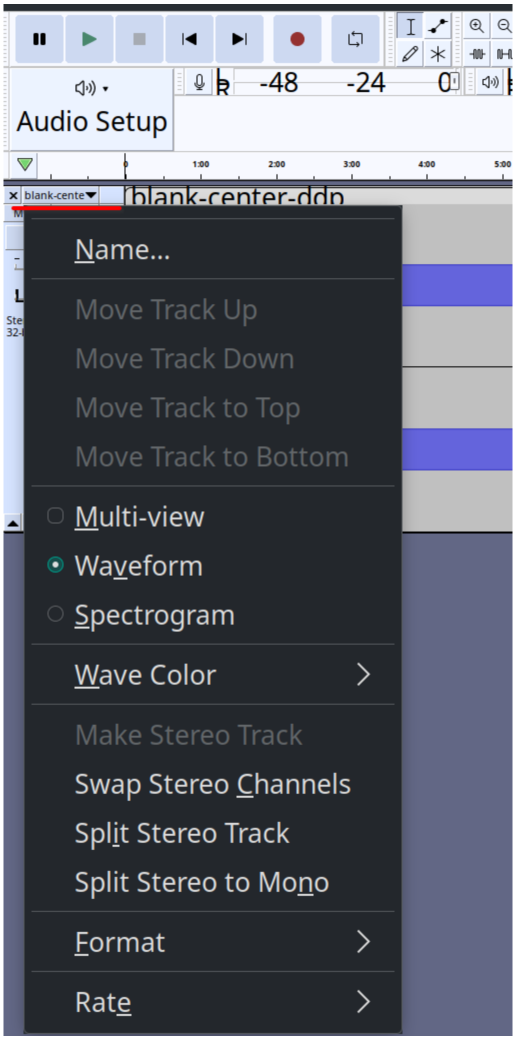

Figure 5: Splitting from Stereo to Mono

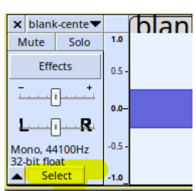

Figure 6: Selecting all audio on a track

7. Slide the recorder switch into the USB position. POWER should light, and the cassette transport should be moving. After approximately 30 seconds, the digital audio signal for the tape formatting should be heard.

8. The process of recording side A should take approximately 30 minutes, and the recorder will automatically stop.

9. Slide the power switch to OFF.

10. Remove the USB stick and place back in computer.

11. Delete track_a.wav from the USB stick, and replace with track_b.wav

12. Repeat steps 3 to 9, for side B.

The cassette is now ready to be used as a digital data pack. If you followed the instructions for a blank image, the data pack will need to be formatted e.g. using SmartBASIC or AJM File Manager, or a utility like Coleco TAPEUTL should be used to copy the image of a super game onto the data pack.

References

[1] Amazon. QFX RETRO-39 Portable Shoebox Tape Recorder, Analog Cassette Tape Deck with USB 2.0, Built-in Microphone, 3.5" Bluetooth Speaker. https://www.amazon.com/dp/B076W2F1GV. [Online; accessed 10-November-2023].

[2] Thomas Cherryhomes. All About AdamNet. http://adamarchive.org/archive/Technical/ADAM%20Tech%20Manuals/All%20About%20ADAMnet%20%28Thomas%20Cherryhomes%29.pdf, 2021. [Online; accessed 10-November-2023].

[3] Coleco. ADAM 6801 Digital Data Drive Listing. https://archive.org/details/adam-6801-digital-data-drive-listing, 1983. [Online; accessed 10-November-2023].

[4] MAMEDev. www.mamedev.org. https://www.mamedev.org/. [Online; accessed 10-November-2023].

[5] Mark. Audio Cassette Tape to a ADAM DDP Pack!! http://adamarchive.org/archive/Technical/ADAM%20Mods/Audio%20Cassette%20Tape%20to%20a%20ADAM%20DDP%20Pack%21%21%20v2.2.pdf, 2019. [Online; accessed 10-November-2023].

[6] Miodrag Milanovic. adam cas.cpp. https://github.com/mamedev/mame/blob/master/src/lib/formats/adam_cas.cpp, 2012. [Online; accessed 10-November-2023].

[7] Audacity Team. www.audacityteam.org. https://www.audacityteam.org/download/. [Online; accessed 10-November-2023].

[8] et al. William Hicks Jr., James Notini. The Coleco Adam Archive . http://www.adamarchive.org/, ongoing. [Online; accessed 10-November-2023].