File: CC2Guide-MacSound-file.PDF
Format: PDF
Date: July 12th, 2003
Author: Mafi; closecombat2@claranet.de; http://members.fortunecity.de/closecombat2/
Last revision: v3, May 3rd, 2006

Close Combat 2 "A Bridge Too Far"

The Mac-Sound-file

(Mac-version of CC2, with some remarks about the PC-version)

What it is

"Close Combat - A Bridge Too Far" (abreviated CC2, ABTF, CC2-ABTF) was the second game of the CloseCombat-series created by Atomic and presented by Microsoft to the Mac-community. It was also the last game of this series for the MacOS. The series was then continued by SSI, UbiSoft and Destineer for PCs only (up to day CC3, CC4, CC5, CCM, RtB, CCMRAFRegt). The game was released in 1997 on a hybrid-CD, running on PCs and under the MacOS 7.5 up to 9.2.2 / MacOS X (for PPC processors) as well.

Many thanks to PHIL LANE

Many thanks to PHIL LANE (PSLSoftware@hotmail.com) for his great work. He is the author of the CC2-Soundeditor for PCs 'CC2hack.exe'. So I was able to know for what I must look in the CC2 sound files.

Many thanks to TINTIN

Many thanks to TINTIN aka GERRY SHAW (http://www.organicbit.com/closecombat/) for his detailed information at his homepage about the CC3 sound file format.

What do you need

First of all you need the original CD “Close Combat : A Bridge Too Far” (hybrid PC / Mac) and the last available update from the internet (version 2.0b: `www.microsoft.com/games/closecombat/cc2/downloads.htm`) or the demo version of CC2 from the same site. For patching the PC sound file you can use PHIL LANE’s soundeditor 'CC2hack.exe' (valid for CC2, CC3, CC4 and CC5). For patching the CC2 Mac sound file 'CC sounds' you can use my tool for MacOS Classic 'myMook'.

The file “CC Sounds” (Mac) and “Sounds.cc2” (PC)

The handling of sounds in the Mac version of Close Combat 2 – A Bridge too far (abbreviated ABTF) differs from the PC version sligthly. Common is: all sounds of ABTF are stored in one single file using the scheme:

- information about the number of sounds which are in the file,
- a directory telling more about storing of the sounds in this file,
- the sound datas.

Later versions of the Close Combat series store their sounds in multiple files, but use all the the same sound file format like the PC-CC2-sound file. Those files of CC3, CC4, CC5 have the filename extension '`.sfx`'.

The name of the sound file of the PC version of ABTF is 'Sounds.cc2', the name of the sound file of the Mac version of ABTF is 'CC Sounds' and is part of the installation on the HD (path: 'NameOfYourHD: A Bridge Too Far ƒ:Sounds:CC Sounds), can be found on the CD (path: 'CloseCombat:Mac:Sounds:CC Sounds') and is part of the demo installation, too. This is essential if you want to get the sound file without having the CD. In clear words: if you have the Mac version of ABTF, then the sound file must reside in a sub-folder (named 'Sounds') of the folder where the executable of ABTF is located (the latest version of the full executable program can be found at the same site as the demo of ABTF).
Differences between the sound files of both versions of ABTF are:

- the directory entries of the Mac version contain only infomations about length and offset of the sound datas,
- the directory entries of the PC version contain informations about length, offset and wave table format of the sound datas (in other words: a part of a regular wave-file header resides in each directory entry),
- each sound data of the Mac version contains a header (it is, as I will show later on, the header of a valid MacOS 'snd '-resource fork) and the wave table,
- each sound data of the PC version contains only a wave table,
- the header entries of the PC version are encoded in reverse byte format LittleEndian (low-byte first), the Mac version is encoded in BigEndian format (high-byte first),
- **in later versions (for example: CC5) even the raw wave table datas are encoded in LittleEndian, too.**

The file format of “Sounds.cc2” (PC)

As described by TINTIN, the file format of every sound file of CC2, CC3, CC4 and CC5 is the same. This is what you will find in the original CC2 sound file 'Sounds.cc2' (improved version of TINTIN’s description):

// Header (16 bytes)
longint *count*; // number of directory entries following this header: 323
longint *unknown*; // is value 45h = 69
longint *padding1*; // is 0
longint *padding2*; // is 0

// Directory Entry (28 bytes per entry)
longint *length*; // length of this wave table in bytes
longint *offset*; // offset to sound data wave table from **start of file** in bytes
longint *stereomono*; // channel s: 0001-0002h or 0001-0001h (stereo or mono)
longint *rate1*; // playback rate of a single channel, is in PC-CC2 always 5622h = 22.050 kHz
longint *rate2*; // playback rate for all channels together, is often AC44h = 44.100 kHz
shortint *channels*; // number of channels: 1 (mono) or 2 (stereo)
shortint *sampling*; // sampling rate: is in PC-CC2 always 8-bit sampling
longint *padding7*; // 4 bytes: 6164:0000h = 'da ' (from 'data'), 0000:0000h or 4E49:0000h

Analyzing the directory entries of PC-file 'Sounds.cc2':

First to mention: it is not a MacOS-born file! The byte format is **LittleEndian**, the result is a reverse byte format (low-byte first).

The values in each directory entry and their meaning were revealed by Phil Lane and later on described by TinTin, too. The values correspond to the header of sound files of format 'WAVE'. It looks like each directory entry contains a part of a WAVE-header. The first 4 bytes contain the length of the sound datas (in bytes), the second 4 bytes contain the the offset to the sound datas from the beginning of the file (in bytes). The third 4 bytes can contain the value 0001-0001h meaning mono-recorded sounds or 0001-0002 meaning stereo-recorded sounds. If you count the entries from 0 to 322 (like Phil Lane did), you will find that 'mono' is the setting for the entries 97-100, 139-144, 146-148, 150-152, 154-158, 161-166, 168, 170-172, 174-176, 178-180, 182-192, 194-199, 201, 203-204, 206-296, 298-316, 318-322. This are mainly speech sounds where stereo is not needed. In all these entries the result of setting to 'mono' is that the following long-integers 'rate1' and 'rate2' are set to 5622h meaning 22.050 kHz. Stereo entries have set 'rate1' to 22.050 kHz and 'rate2' set to AC44h = 44.100 kHz! If the entry is 'mono', then the 'channels' entry is always 0001h and the 'sampling' entry is 0008h, meaning 'mono' and '8-bit sampling'. On the other hand all 'stereo' entries have the the 'channels' entry set to 0002h and the 'sampling' entry set to 0008h, meaning 'stereo' and '8-bit sampling'. I think that the meaning of the third and sixth long-integer are not 'unknown' as discribed by TinTin but contains information about 'mono/stereo' and the sampling.

The only long-integer which meaning is unknown is the last, the seventh long-integer: it is in most cases 6461:0000h = 'da ' (the beginning of the string 'data' out of a WAVE-header proceeding the entry where the length of the sound datas are defined). The entries of the file 'Sounds.cc2' No. 5, 7-14, 19, 22, 60-62, 64, 145-322 have here the value 0000:0000h (and the setting 'stereo, 8-bit'). The entries No. 25, 121-123, 103 contain here the value 494E:0000h = 'IN '. I think that is only a padding long-integer which can be ignored when extracting the sound datas.

The raw sound datas follow the directory immediately and are of variable length. These raw wave tables are identical to wave tables of a files of format 'WAVE':

The file format of “WAVE”-files (PC), which will be produced by my tool “myMook” (Mac)

To make exporting of sounds out of the sound file of the PC-version of ABTF easier, I have implemented the export of 'WAVE' sound files in my tool 'myMook', using the following header:

// Wave format (LittleEndian, low byte first)

string "RIFFP]"	// a string determining the WAVE-file format
shortint *value1*	// value 0001h
string "WAVEfmt "	// a string determining the WAVE-file format
longint *headerlength*	// length of following data description, value 00000010h
longint *stereomono*;	// channel s: 0001:0002h or 0001:0001h (stereo or mono)
longint *rate1*;	// playback rate of a single channel
longint *rate2*;	// playback rate for all channels together
shortint *channels*;	// number of channels: 1 (mono) or 2 (stereo)
shortint *sampling*;	// sampling rate: is in PC-CC2 always 8-bit sampling
string "data"	// hear ends the data description and the datas follows
longint *length*	// length of wavetable
variable bytes	// the raw wave table itself

The first 16 bytes defines the file format, the next 16 bytes describe the sound recording format, then follows the string 'data' and the length of the then following raw wave table (all in LittleEndian encoding). As you can see, you can take the the part of each directory entry in 'Sounds.cc2' concerning the sound data description unchanged to create a valid 'WAVE'-file header.

The file format of “CC Sounds” (Mac)

First: it is a MacOS-born file, byte-format is **BigEndian**, that means high-byte first!

Number of Sound entries is 323 as reported by PHIL LANE in 1998, the same as in the PC-file 'Sounds.cc2'. The reason why the programmers at Atomic did not stored the sounds in a resource fork (as usually done with MacOS programs) is the limit of the size of resource-forks: they are limited to 16 MB. So they stored it in the data fork of a file, but using the same data format as for 'snd '-resources as usual.

Now here is the file format of 'CC sounds':

- Header: first 4 bytes: number of entries = 323 = 00000143h
- 2584 bytes: directory with 323 entries à 8 bytes.
- each entry of the format: offset of the sound (4 bytes) from the **start of the datas** (NOT from start of the file), length of the sound (4 bytes) (this length does include the resource header length),
- 323 'snd '-resources in stereo, but written into the data-fork. (the original file contains only stereo sounds, but mono sounds are allowed, too).

The beginning of the sound datas is at byte 2588 (= hex offset from start of file 0A1Ch). In the original file the data format is: sampling rate 8-bit, 2 channel stereo, 44.1 kHz or 22.050 kHz. Header length of each data entry is 84 bytes (that means: it is an extended 'snd '-resource header), then the wave table of the sound datas follows. The header is a valid 'snd '-format-1 resource, in the original file always an extended sound header because all sounds are in stereo. It is unusual that resources are stored in the data fork, but in this case the programmers at Atomic Inc. were forced to do so.

You can find a suitable description of this resource format in Apple's book 'New Inside Macintosh: Sound' on page 2-74, 2-77, 2-104ff and 2-155.

The description we need can be found on page 2-77, page 2-104 and page 2-106:

A format 1 'snd ' resource containing sampled-sound data with extended sound header (as shown with the first sound out of the file 'CC2 sounds' as an example):

```
data 'snd ' (ID=1000, "CC2Sound0", purgeable) {
/*the sound resource header, BigEndian byte format*/
$"0001"         /*format type, should be 1, in CC2: must be 1*/
$"0001"         /*number of data types, in CC2: must be 1*/
$"0005"         /*sampled-sound data, in CC2: must be 5*/
$"000000C0"     /*initialization option: initMono = 0080h, */
                /* (initStereoCC2 = 00C0h, initMonoCC2 = 00A0h)*/
                /* (initStereo16bit = 00E0h is not valid for CC2!)
/*the sound commands*/
$"0001"         /*number of sound commands that follow (1)*/
$"8051"         /*command 1--bufferCmd*/
$"0000"         /*param1 = 0*/
$"00000014"     /*param2 = offset to sound header (20 bytes)*/
/*the sampled sound header*/
$"00000000"     /*pointer to data (it follows immediately)*/
$"00000002"     /*number of channels in sample (here: 2)*/
$"56220000"     /*sampling rate of this sound (here: 22.050 kHz). can be: */
                /*this value must be in LittleEndian format !!! */
                /* rate44khz     = $AC440000;     {44100.00000 Fixed} */
                /* rate22khz     = $56EE8BA3;     {22254.54545 Fixed} */
                /* rate11khz     = $2B7745D1;     {11127.27273 Fixed} */
$"00000000"     /*starting of the sample's loop point, in CC2: must be 0 = start of datas*/
$"0000022F"     /*ending of the sample's loop point, in CC2: is the length of the */
                /*sampled sound datas = end of datas if mono, half of the length if stereo*/
$"FF"               /*extended sample encoding. can be: */
                    /*                    stdSH = $00;          {standard sound header} */
                    /*                    extSH = $FF;          {extended sound header} */
                    /*                    cmpSH = $FE;          {compressed sound header} */
$"3C"               /*baseFrequency: the pitch at which the original sample was taken.*/
                    /* in MIDI-note: 3Ch (= 60) means the 'C' from octave 6 */
/* the extension to the sound header */
$"0000022F"         /*numFrames  {total number of frames = length / number of channels} */
$"400DAC44000000000000"
                    /* AIFFSampleRate:    Extended80 = 10 bytes; {rate of original sample} */
$"00000000"         /* markerChunk:       Ptr = 4 bytes;        {reserved} */
$"00000000"         /* instrumentChunks:  Ptr = 4 bytes; {pointer to instrument info} */
$"00000000"         /* AESRecording:      Ptr = 4 bytes; {pointer to audio info} */
$"0008"             /* sampleSize:        ShortInteger = 2 bytes;{number of bits per sample} */
$"0000"             /* futureUse1:        ShortInteger = 2 bytes; {reserved} */
$"00000000"         /* futureUse2:        LongInt = 4 bytes;    {reserved} */
$"00000000"         /* futureUse3:        LongInt = 4 bytes;    {reserved} */
$"00000000"         /* futureUse4:        LongInt = 4 bytes;    {reserved} */
/*the sampled-sound data (raw wave table)*/
$"7C 7C 7B 7B ....
                    /*rest of data omitted in this example*/
};
```

The result: in case of the CC2 file the header of a stereo 'snd '-resource with **initialization option "000000C0h"** has a fixed length of **84 bytes**. In case of a 'snd '-resource with **initialization option "000000E0h"** (stereo 16bit) the header is longer: in this case the header length **is 112 bytes**! "E0h" encoded stereo sound resources will cause trouble. You must avoid them!

To import PC-mono sounds you should use instead a so-called standard sound header (shown with the last sound out of the PC file 'Sounds.cc2' as an example):

```
data 'snd ' (ID=3322, "PCSound322mono", purgeable) {
/*the sound resource header, in BigEndian*/
$"0001"             /*format type, should be 1, in CC2: must be 1*/
$"0001"             /*number of data types, in CC2: must be 1*/
$"0005"             /*sampled-sound data, in CC2: must be 5*/
$"000000A0"         /*initialization option: initMono = 0080h, */
                    /* (initStereoCC2 = 00C0h, initMonoCC2 = 00A0h)*/
/*the sound commands*/
$"0001"             /*number of sound commands that follow (1)*/
$"8051"             /*command 1--bufferCmd*/
$"0000"             /*param1 = 0*/
$"00000014"         /*param2 = offset to sound header (20 bytes)*/
                    /*the sampled sound header*/
$"00000000"         /*pointer to data (it follows immediately)*/
$"00008C40"         /*number of bytes in sample (here: 35904)*/
$"56220000"         /*sampling rate of this sound (here: 22.050 kHz).*/
                    /*this value must be in LittleEndian format !!! */
$"00000000"         /*starting of the sample's loop point, in CC2: must be 0 = start of datas*/
$"00008C3F"         /*ending of the sample's loop point, in CC2: is the length of the*/
                    /* sampled sound datas = end of datas if mono*/
$"00"               /*standard sample encoding.*/
$"3C"               /*baseFrequency: the pitch at which the original sample was taken.*/
/*the sampled-sound data*/
$"80 80 80 80 ....
                    /*rest of data omitted in this example*/
};
```

The result: in case of the CC2 file the header of a **mono 'snd '-resource** should have a fixed length of **42 bytes**.

This 'snd ' resource indicates that the sound is defined using sampled-sound data. The resource includes a call to a single sound command, the bufferCmd command. The offset bit of the command number is set to indicate that the sound data is contained in the resource itself. Following the command and its two parameters is the sampled sound header, the first part of which contains important information about the sample. The second parameter to the bufferCmd command indicates the offset from the beginning of the resource to the sampled sound header, in this case 20 bytes (taken from APPLE’s book Sound). CC2 uses originally extended sound headers, because all sound entries are in stereo (different to the PC version)! But with the header definitions above we are able to convert an entire PC-'Sounds.cc2' file to MacOS format.

“myMook” (Mac)

In July 2003 I made a little program running under MacOS 8.5 – 9.2.2 using the data definitions described above. This tool can convert an entire PC-sound file of ABTF or of a custom made sound file of the ABTF mods PacificFront (by TAKI, sound file by MASAHIKO MIZUCHI), Kreta (by KYLE SCOTT, sound file by TIM CATHERALL), GreatWar patch (by TIM CATHERALL) and others from PC format into Mac format. Not much to say about it: it works.

The program can extract the PC-sound datas in 'WAVE'-file format from all sound files of the CC series from CC2 to CC5. Furthermore the program can extract the PC-sound datas out of the CC2 PC-sound file 'Sounds.cc2' in the file format of System-7-sounds to play them directly from the desktop. These System-7-sounds simply contain the sound datas in their resource fork as 'snd '-resources. The program can repack a CC2 PC-sound file from 'WAVE'-files in the same format as it exports them.

And the program can transfer the Mac-sound file 'CC Sounds' into two independend files (file #1 contains the sounds #0 .. #160; file #2 contains the sounds #161 .. #322), where the sounds are stored in the resource fork. From this resource fork(s) the program can play the sounds, and it can play single System-7-sound files. A description of the CC2-sounds is included in a popup menu (thanks to PHIL LANE for his work). To complete the little program it is able to import single sounds from the resource fork of a System-7-sound file into the two independend files and it is able to **rebuild a new file 'CC Sounds'** from this two files (simple: creating a new header and copying the resource fork datas into the data fork of the new file).

In 2006 I added the functionality to convert a PC sound file into a MAC sound file and vice versa.

Furthermore the program is able to play sounds. Usually it should be able to play MacOS-System-7 sounds (when running under MacOS 8.5-9.2.2), WAVE-files (when running on PCs) or nearly every sound file supported by QuickTime® when running under MacOS-X (AIFF, WAVE).

If you want to identify the sounds by name inside the files 'Sounds.cc2' / 'CC Sounds', you can load an external sound index file. This file must be a normal text file, each line containing a text entry (one line for every 323 sounds). I recommend to use the original index file made by Phil Lane.

Sound converting

Because of the LittleEndian encoding of the wave tables of the newer CC sound files (CC5), you will need an additional tool to convert the extracted 'WAVE'-files into System-7-sound files before you can import them into 'myMook'. You can use shareware like 'Amadeus II' or the good old 'SoundEdit 16' tool (trademark by MACROMEDIA) for this purpose. To extract 'snd '-resources from any resource fork you can use APPLE’s 'ResEdit 2.1.3' or the shareware tool 'SoundExtractor 1.31' by Alberto Ricci (dated 1992). For import of resource fork datas you can use APPLE’s 'ResEdit 2.1.3', too.

For handling of large files (like CC5’s 'Music.sfx') 'myMook' will needs at least 48 MB of RAM, because it takes the whole sound in one piece into memory. If any conversion fails, try first to increase 'myMook's memory in the 'Finder' please. If you run into trouble with the sounds after conversion (only strange noise appears), the original sound datas migth be quadrophonic or better, which 'myMook' and/or ABTF migth be unable to handle. You can use conventional sound editors under MacOS to downsize those sounds to 44kHz, 2 channel, 8-bit sampling. I think then the sounds will be recognized properly.

Having multiple sound files on your HD

You can have more than one sound file at the same time in your ABTF sound file folder on your HD. ABTF will always take the one named 'CC Sounds'. In the data fork of the executable file 'A Bridge Too Far' is only one occurence of this string: in version v2.0b at 1577DCh (offset from start of data fork). If you have seperate executables 'A Bridge Too Far xxxx' for each CC2-mod you want to play you can patch these entry in the executable, naming the different sound files accordingly and your game will start up with the desired sounds. Don’t forget to turn on sound and music in the game.

Specials of the Mac-Installation of ABTF

During Installation of the game on your HD the installer will create two more files which are not on your original CD. In the Mac-version of ABTF the installer will extract the sound #102 (CC theme) from the file 'CC Sounds' and will store it as a valid QuickTime (trademark by Apple) movie sound file (created by SoundEditor16 (trademark of Macromedia), creator 'Nqst', type 'MooV', 8-bit sampling, 22.050 kHz, stereo) in the files 'amusic' (meaning Allied music of the main screen) and 'gmusic' (meaning German music of the main screen) in the folder 'A Bridge Too Far ƒ:Videos'. ABTF will not use the sound #102 out of the file 'CC Sounds' during runtime! So you can patch this sound entry in this file, but if you will hear a new sound in the main screen, you can easily replace these files by your own sound files in the folder 'Videos'. If you remove these sounds from the folder, you will have no sounds in the main screen at all (but it will cause no error message)!

You can patch the name of the main screen sound files in the data fork of the executable file of ABTF. In the Mac version v2.0b (latest update released) the names of the files reside at 157936hex ('amusic') and 15793Ehex ('gmusic'). If you have multiple executables of ABTF you can have different main screen sound files in the folder 'Videos' if you patch these entries, too.

MAFI
closecombat2@claranet.de -
http://members.fortunecity.de/closecombat2/
http://www.closecombat2.claranet.de/
http://www.geocities.com/cc2revival/
http://www.cc2.claranet.de/
http://www.dieppe.claranet.de/
http://www.afrika.claranet.de/