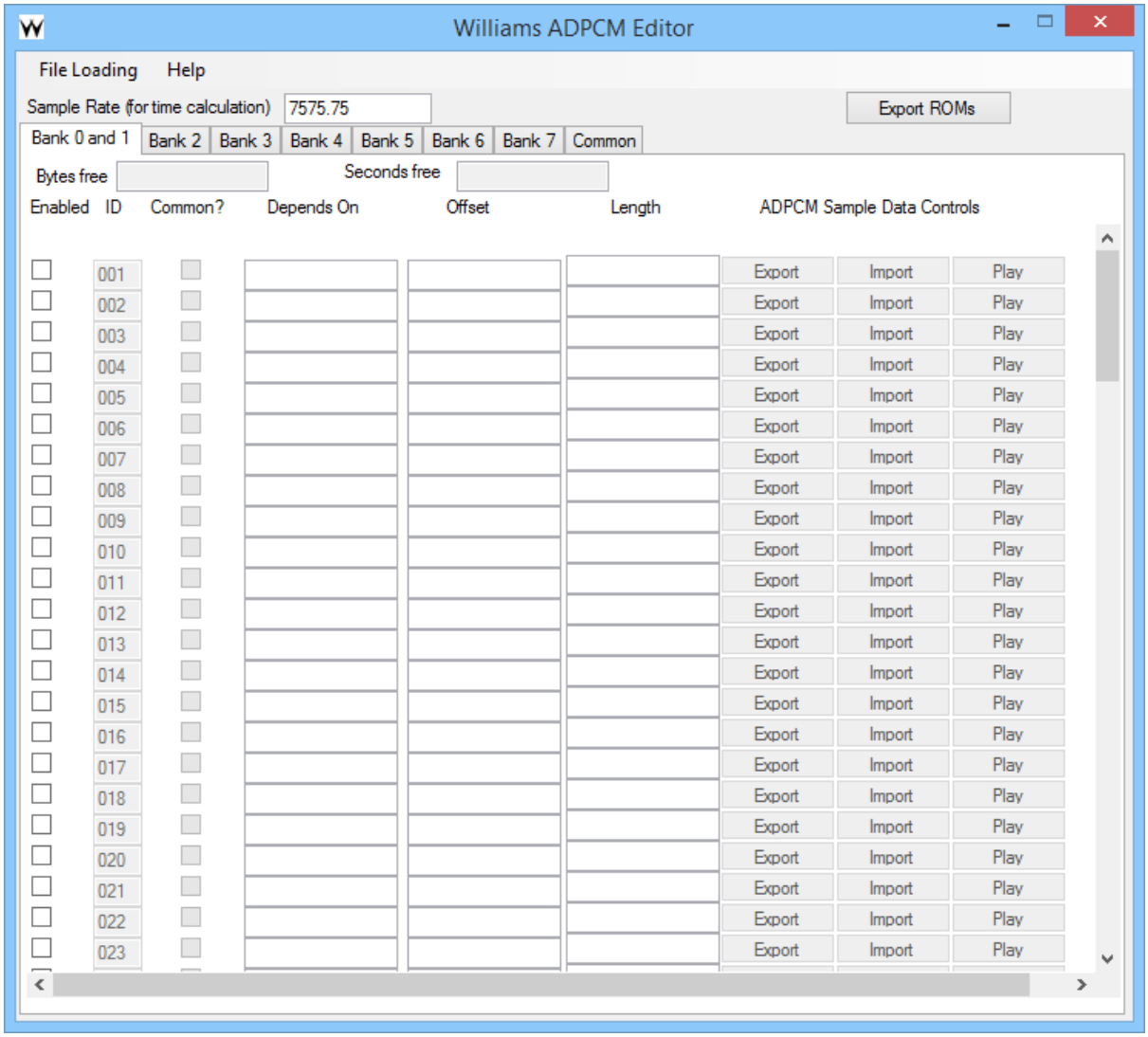
Williams OKI Editor

The Williams OKI Editor can be daunting at first, due to some of the complexity behind how it operates but for most operations this guide should cover everything.

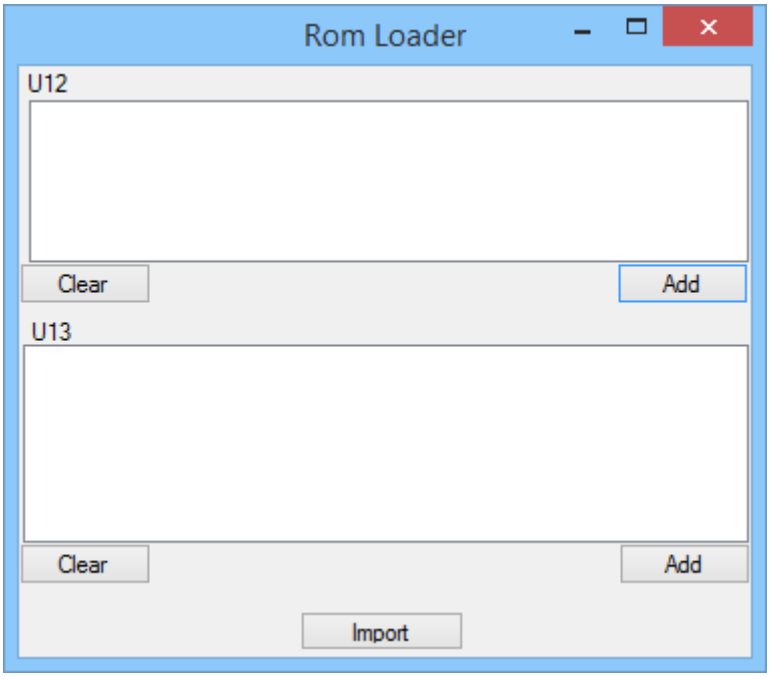
Main Window



The Menu Bar at the top contains the File Loading (Import ROMs at the moment, but there may be more in future), and a Help section that links to an 'About' screen.

Loading a file

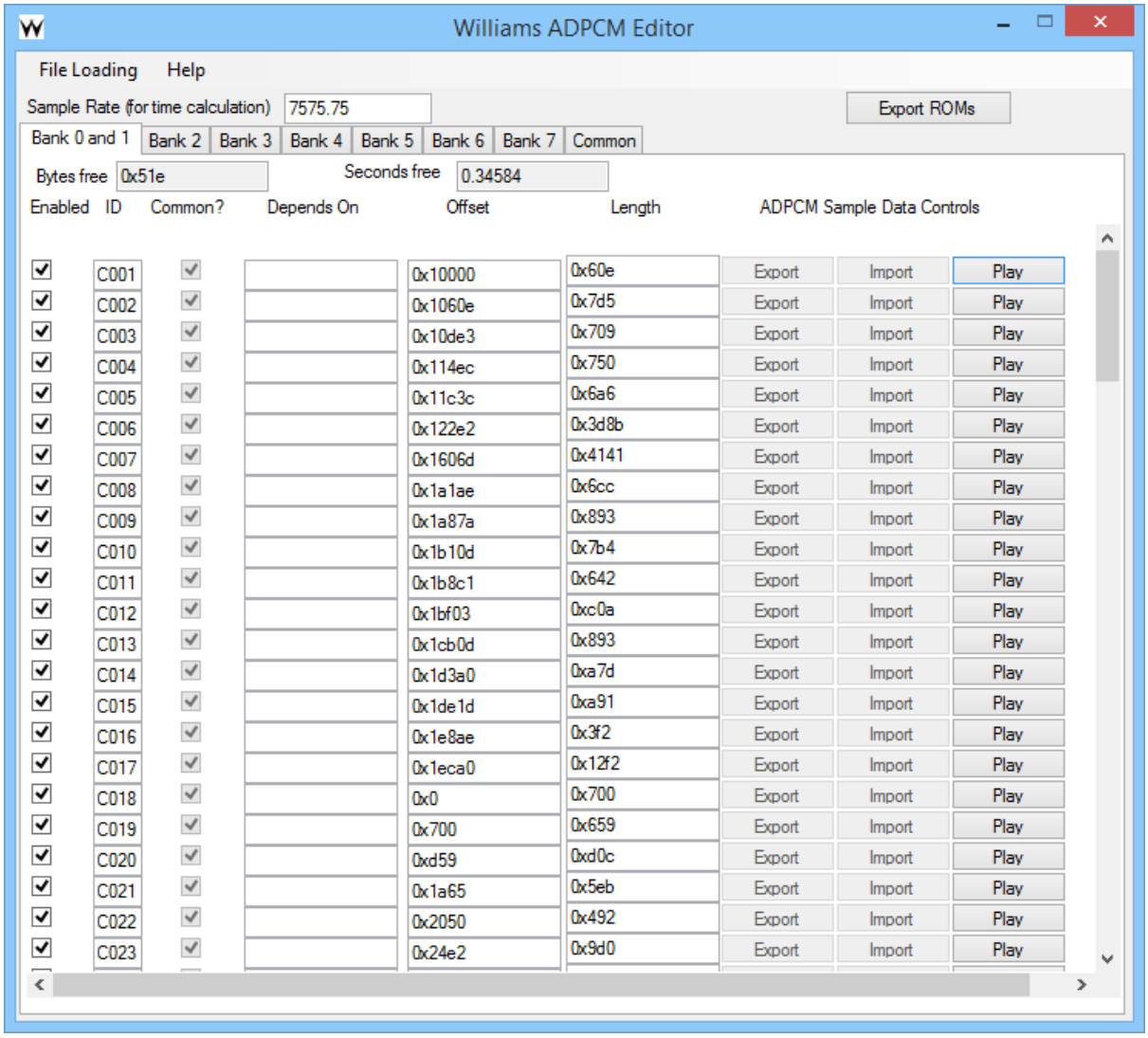
To load a ROMset, Press File, then Load. This will pop up a window with the U12 and U13 loaders.



For those who are unaware, the Williams ADPCM board reads its data from two locations, U12 and U13 on the sound board. These can be a number of sizes, for example MK uses half size ROMs, and NBA Jam uses the full size. The hardware automatically mirrors the half size ROMs to fill the space, effectively creating duplicate banks. Press 'Add' on each control in turn to open up a dialog to open the ROM (or ROMs) needed. Just make sure you have them in the correct order. Once done, press OK, and the code will load the U12 and U13 ROMs as on hardware, and separate out the banks automatically. When done, you can navigate between the banks and edit accordingly. Bank 0 and 1 are identical in all configurations, and so are shared here.

NOTE: For the half size ROMs, Banks 2 and 3 are disabled for a half sized U12 and Banks 4 and 5 disabled for a half sized U13 ROMs as for consistency you are only given access to one instance of each duplicated bank for mirrored ROMs).

When done, the Main Window will fill with the data, and will look something look this:



The Status strip immediately below contains the functions needed for final operation, namely the sample rate, and the button for exporting the finished ROM images. The sample rate is preconfigured for Williams at 7575.75 Hz, but you can change it, all calculations for timing and playback are based on this figure. When done with editing, the ‘Export ROMs’ button will open a dialog box to open to save new U12 and U13 ROMs. These will export one file for each of U12 and U13, so if you were using split files for these before they will be merged.

The rest of the window is taken up with controls that handle the actual editing.

Bank Controls

At the top of each bank, the code identifies how many bytes of addressable space remain, and therefore how many seconds of audio you have time for.

Note that if this is negative, the bank will be invalid and the exported ROM is likely to have problems.

Enabled determines if a sample should be considered or not (because of the way each sample is assigned a space in the ROM table, if this isn't the last sample, it saves no space as we have to leave a placeholder there, like the first non-common sample in Bank 0/1 of NBA Jam TE, which is dummied out).

Common determines if a sample is actually pointing to the common space (you'll need to change the id to the relevant 'C' value)

ID indicates the sample id in the bank (Common samples that point to common space start with 'C', and use the position in the Common Bank as the number)

Depends On implies this sample is an offset into another, existing sample in the bank, so doesn't need its own ROM space. Specify which number it is (common IDs can’t be accepted here), and then you can specify a hex offset into the existing sample, and a new length. You'll notice some samples do this for looping etc., and the Common samples show the offset into the Common Bank space for the benefit of external Hex Editing.

Offset indicates the offset in hex (0x notation) into the sample (only relevant for those where ‘Depends On’ values are assigned)

Length indicates in hex how much of the stored sample to use.

Import will load in a .VOX file (must be at the right sample rate, and already VOX ADPCM encoded, or a .WAV file. WAV files are automatically resampled to the correct rate and set to Mono to be encoded.

Export will read out the stored sample to VOX ADPCM if the chosen filename is .VOX, or as PCM for a .WAV.

Play uses NAudio to play the sample as it would be called by the hardware (useful for testing).

KNOWN ISSUES:

We don't have a mechanism to save a project in progress, except for outputting the ROMs themselves. The plan is a JSON format, but this will have to be done after initial release.

The program only supports the Williams system for now, but should be sufficiently extensible to do MSM6295 and similar over time.

Testing has been a one man job so far, there are likely to be bugs. Nothing that should blow up on real hardware, but be warned.

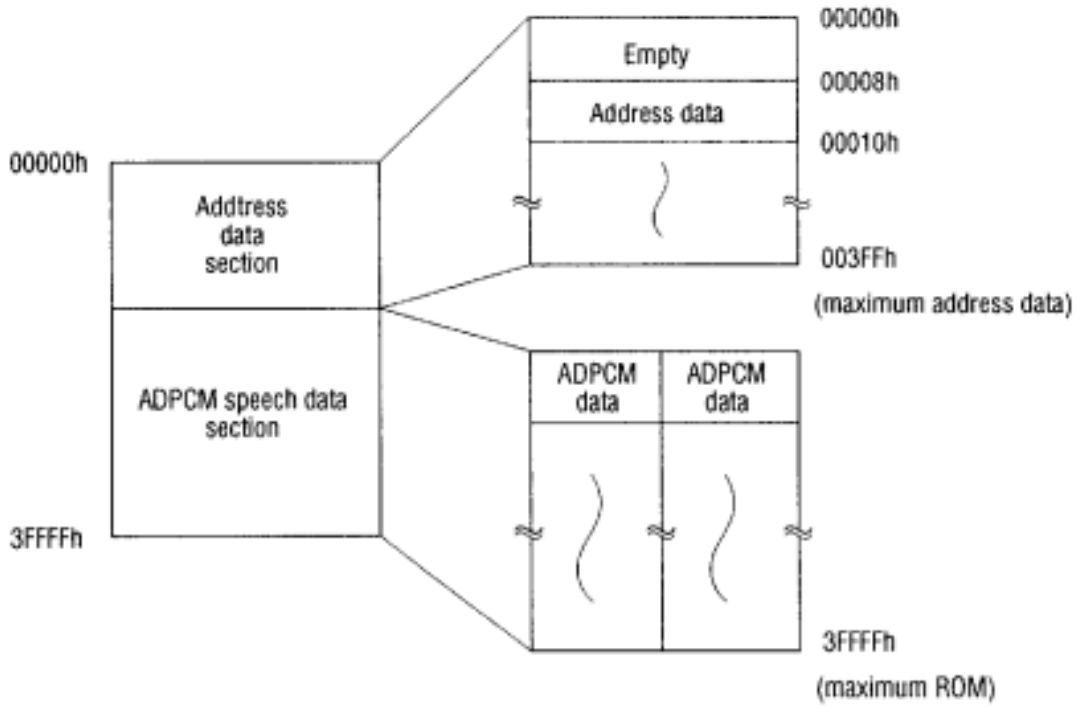
Williams ADPCM Theory of Operation

The use of ADPCM hardware in Williams products seems to be as a direct replacement for the Harris CVSD hardware that was previously used for sample speech, an efficient if noisy technique that required the chip to be connected to a preamp. Using the MSM6295 removed the need for a preamp and a separate clock circuit but took a fixed rate 4 bits per sample, compared to the theoretical minimum of 1 bit per sample for the CVSD.

The MSM6295 uses the VOX/Dialogic ADPCM, addressing an external ROM. 127 samples can be addressed per ROM, with a total addressable space of 0x40000 bytes, of which at least 0x3C000 is audio. This can be increased if fewer words are addressed, as the sample player can be pointed to inside the lookup table.

DATA ROM STRUCTURE

The following chart shows the memory map of the source data ROM.



Address ROM

Each sample pointer consists of 8 bytes (the first 8 bytes of the ROM are blank, as the true sample 0 is invalid). Of these 8, three contain the start address in big endian, 3 contain the end address in the same encoding, and the other two are unused padding bytes set to 0.

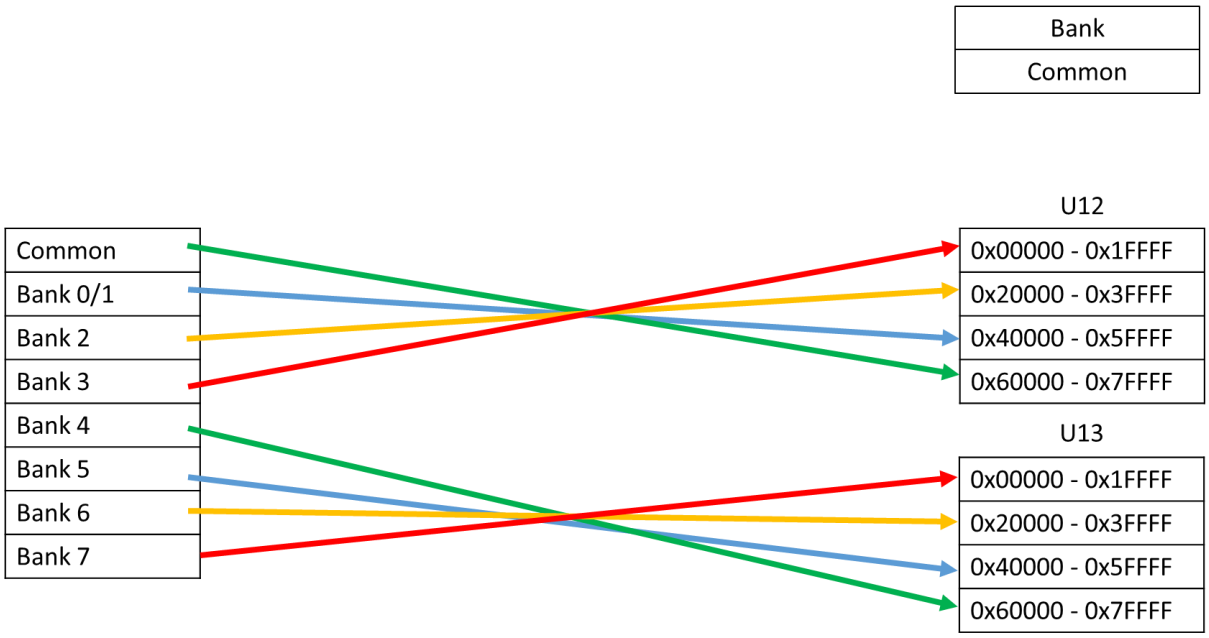
e.g. An address set of 03 00 00 03 06 0e 00 00 will correspond to a sample playing from 0x030000 to 0x03060e

When a sample is called by the system, the chip will read the ROM at an area at the value of (ID * 0x08) and reads from there.

Bankswitching

Instead of limiting to 0x40000, the Williams hardware performs bankswitching to allow a much larger space (1 Meg) to be observed. In order to match the standard OKI chip header, the space that is changed is the first 0x20000 (128k), with a ‘common’ 128k always in place. Bank 0 and 1 map to the same bank, as regular Bank 0 value is the permanent ‘common’ space, used as the last 128K.

Layout



As each bank is 128K, and positioned first in the addressed space, the banked space must contain the headers used to load the samples (a lot of duplication).

As a consequence, samples designed to be common to any bank configuration should have their data placed in the common space, and have addresses pointing at 0x30000 – 0x4ffff