JUKE-BOX ROM CREATOR SCRIPT PACK

By P-LAB 2020

This pack contains a series of useful scripts to make “custom-ROM” with your own favourite programs or games. As specified in the Apple-1 Juke-box User's Manual, the term ROM in this document generically refers to programmable devices such as EPROM/EEPROM/FLASH/OTP etc.

The file(s) produced by these scripts will be ready to be used by your favorite device programmer.

The use of these scripts requires some familiarity with the Linux terminal interface.
It is suggested, but not mandatory, the use of utilities like *Midnight Commander* in order to avoid the typing of long commands and the consequent risk of errors.

Scripts have been written in BASH for Linux and can be freely modified.

1. Operating requirements

To work properly, the scripts need the following programs:
- bc
- xxd
- ascii2binary

If they are not present in the system, they must be installed manually with the command:
`sudo apt-get install bc xxd ascii2binary`

Scripts and all service directories must be copied in a directory called `JUKEBOX_ROM_CREATOR`, within your *home directory*.

2. Description of operations

These scripts use the format and nomenclature of the files used by the CIDERPRESS and CFFA1 card:

`Name#TypeStartaddress`

Where:
- `Name` is the name of the file
- `#` is a separator,
- `Type` is the file type: `06` for binary files and `F1` for BASIC files,
- `Startaddress` is the first memory address (hexadecimal) where the program must be copied.

Example:
`BASIC#06E000` --> the program is called BASIC, it is binary (`06`) and must be copied to memory from address 0xE000.

STARTEK#F10300 --> the program's name is STARTREK, it is a BASIC program (F1) and must be copied to memory starting from address 0x300.

The files/programs/games we want to burn on the custom-ROM must first be processed by the script called 1-stripper.sh.
It removes the information not strictly necessary so that the efficiency of the storage on the ROM is maximized, and creates a support file that will be used later.

Once this is done the second script, called 2-packer.sh will create the binary file (or more binary files) that you will write on your custom-ROM.

3. Directory Structure

Directories are structured as follows:

1-programs	Manually copy here the programs you want to be processed. They can be copied from directory 9-archive or from other sources. They have to be compliant with the data structure already described.
2-stripped	Files and their accessory files are written here, after execution of the script 1-stripper.sh. Usually there is one .bin file and one .pat file for each program.
3-topack	Manually copy here all files from directory 2-stripped to be processed and packed into the ROM binary file.
4-packed	After processing, binary ROM files (16kB or 32kB) will be saved here, along with a text file with the content of the ROM itself.
9-archive	this directory contains an archive of Apple-1 software.

4. Example of creation of a custom-ROM

Suppose you want to create a custom-ROM containing the STARTREK program.
It is a BASIC program, so the BASIC interpreter must necessarily be included in the custom-ROM.
In the following example, the commands to give are written **in bold**.
Open a terminal and go to the working folder:

```
cd ~/JUKEBOX_ROM_CREATOR
```

Now copy the two desired programs from the archive to the source directory:

```
cp ./9-archive/BASIC/BASIC#06e000 ./1-programs
cp ./9-archive/BASIC/STARTREK#f10300 ./1-programs
```

Now run the file preparation program:

```
./1-stripper.sh
```

Service information will appear on screen, proving that the script is running:

```
found:
BASIC#06e000
BASIC---  BAS=0  LEN=4096  LEN_HEX=1000  LEN_HI=10  LEN_LO=00  START_HEX=E000
START_HI=E0 START_LO=00
found:
STARTREK#f10300
STARTREK  BAS=1  LEN=3328  LEN_HEX=0D00  LEN_HI=0D  LEN_LO=00  START_HEX=0300
START_HI=03 START_LO=00
Slicing: REAL_BASIC_START=0430 BYTES TO SLICE=305
         NEW_LEN=3024 NEW_LEN_HEX=0BD0 NEW_LEN_HI=0B NEW_LEN_LO=D0
Strip completed.
```

At the end of the run, you can look (if you like) at the destination folder to verify that everything worked: for each source file there must be two files: one with the extension .bin and one with the extension .pat:

`ls -la ./2-stripped/`

```
total 24
drwxrwxrwx 2 root root 4096 ago  3 11:26 .
drwxrwxrwx 7 root root 4096 ago  3 10:59 ..
-rwxr-xr-x 1 pi   pi   4096 ago  3 11:26 BASIC---.bin
-rw-r--r-- 1 pi   pi     15 ago  3 11:26 BASIC---.pat
-rw-r--r-- 1 pi   pi   3206 ago  3 11:26 STARTREK.bin
-rw-r--r-- 1 pi   pi     15 ago  3 11:26 STARTREK.pat
```

In the example above, all the files needed for the next steps have been created.

Note on filenames: If the length of the original filename is longer than eight characters, the name will not be truncated to the first eight characters, but will be composed by the first seven characters of the original name, plus the last one of the original name.
This operation is done to limit the ambiguity that would be created with programs with names longer than eight characters but with the same root (e.g. STARTREK and STARTREK2003 → STARTREK and STARTRE3 instead of STARTREK and STARTREK).

Now manually copy these files to the `3-topack`:

`cp ./2-stripped/* ./3-topack`

The next step is to launch the actual binary file creation program, which will also contain the management program and everything needed for its operation. Run it then:

`./2-packer.sh`

The program will ask what kind of prefix you want to give the binary file name(s). In the example, "MYROM_" has been chosen.

Filename prefix for output files?
`MYROM_`

Now the program asks how the file that will be created should be physically mapped: 16kB (16384 bytes) or 32 kB (32768 bytes).

```
ROM file size in bytes [16384/32768]?
32768
```

Now the script automatically searches and chooses the best allocation for all the selected files and process them to create the ROM binary file:

```
Starting loop 0
analyzing: BASIC---.bin

PRG#1 BASIC---.bin FITS
analyzing: STARTREK.bin

PRG#2 STARTREK.bin FITS
24442+0 records in
24442+0 records out
24442 bytes (24 kB, 24 KiB) copied, 0,0494592 s, 494 kB/s
226+0 records in
226+0 records out
226 bytes copied, 0,000362394 s, 624 kB/s
32 kB EPROM detected: 32768 bytes
HEADER=F1 -- BASIC  PROGRAM DETECTED  -- PROGRAM LENGTH=0BD0 -- NEXT START
ADDRESS=4C86 -- Next PAT=31759
HEADER=FE -- BINARY PROGRAM DETECTED  -- PROGRAM LENGTH=1000 -- NEXT START
ADDRESS=5C86 -- Next PAT=31774
HEADER=FF

32kB file detected. Flipping banks...
PAT Completed.
Loop 0 completed.
All completed.
```

If you have selected many files, many cycles will be performed. At the end of the execution, the files will be stored in the folder `4-packed`:

`ls -la ./4-packed/`

```
total 44
drwxrwxrwx 2 root root  4096 ago  3 11:27 .
drwxrwxrwx 7 root root  4096 ago  3 10:59 ..
-rw-r--r-- 1 pi   pi   32768 ago  3 11:27 MYROM_0.BIN
-rw-r--r-- 1 pi   pi      19 ago  3 11:27 MYROM_0.txt
```

The `MYROM_0.BIN` file will be used to program the ROM. The file MYROM_0.txt contains the list of programs contained in the binary file. To see its contents, just use the command `cat`:

`cat ./4-packed/MYROM_0.txt`

```
1 BASIC
2 STARTREK
```

If your selection of files exceeds the amount of data than can be stored in the chosen physical size, (16 kB or 32 kB) all the necessary files will be created, adding an increasing numeric suffix.
Example:

`ls -la ./4-packed/`

```
total 44
drwxrwxrwx 2 root root  4096 ago  3 11:27 .
drwxrwxrwx 7 root root  4096 ago  3 10:59 ..
-rw-r--r-- 1 pi   pi   32768 ago  3 11:27 MYROM_0.BIN
-rw-r--r-- 1 pi   pi      52 ago  3 11:27 MYROM_0.txt
-rw-r--r-- 1 pi   pi   32768 ago  3 11:27 MYROM_1.BIN
-rw-r--r-- 1 pi   pi      59 ago  3 11:27 MYROM_1.txt
-rw-r--r-- 1 pi   pi   32768 ago  3 11:27 MYROM_2.BIN
-rw-r--r-- 1 pi   pi      81 ago  3 11:27 MYROM_2.txt
-rw-r--r-- 1 pi   pi   32768 ago  3 11:27 MYROM_3.BIN
-rw-r--r-- 1 pi   pi     136 ago  3 11:27 MYROM_3.txt
```

The .BIN files can be copied individually into four 32kB ROMs (27c256, for example), or they can be concatenated manually with the cat command in order to have a 128 kB (131072 bytes) file to write to an appropriately sized ROM.

`cat MYROM_0.BIN MYROM_1.BIN MYROM_2.BIN MYROM_3.BIN > MYROM_TOTAL.BIN`

The file `MYROM_TOTAL.BIN` will be 131072 bytes long.

If the produced files have a smaller overall size than the available ROM space, (e.g. my favourite games occupy only 32 kBytes but in my drawer I have only 64 kBytes ROMs) it is preferable to copy twice the 32 kBytes on to the ROM.
Empty/unallocated parts of the ROM can hang the Apple-1.

5. Compatibility

- All Apple-1 binary files are supported, as long as they do not exceed physical RAM/ROM Boundaries, as described in the *Apple-1 Juke-Box* manual.
- Only BASIC program written with APPLE-1 INTEGER BASIC are supported.
- Supported ROM size is up to four Mbit (512 kBytes), equivalent to a 27/29c040.
