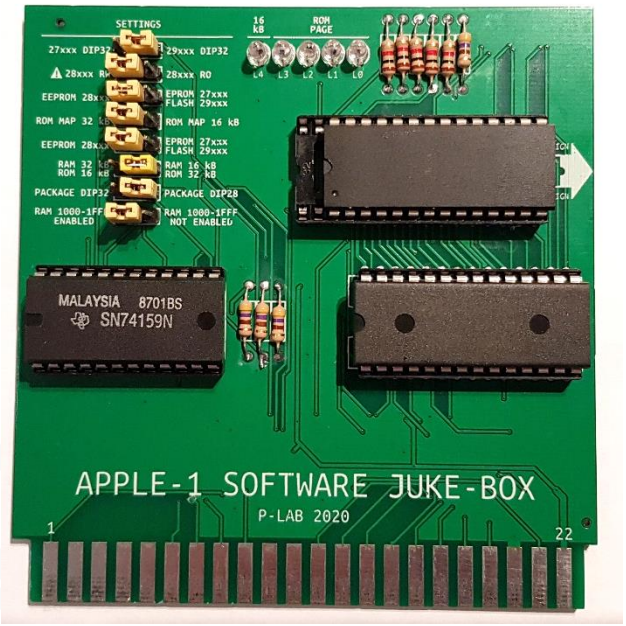
APPLE-1 JUKE-BOX EEPROM

v1.09

Have you decided to write your BASIC programs and your Machine Language routines by yourself?
Excellent! We are very pleased!



The *Apple-1 Juke-Box Software* will make your life easier, with a **dedicated memory device** on which you can save the fruits of your hard work.

This device is an **EEPROM (Electrically Erasable Programmable Read-Only Memory)**.
On it, you will find everything you need including:

- The Integer BASIC,
- The Program Manager as described in the main user manual,
- A special program that allows you to save your programs directly on the EEPROM.

1. REPLACEMENT OF THE STORAGE DEVICE

First, you need to replace your existing storage device (e.g. the supplied FLASH Memory 29c020) with the supplied **EEPROM 28c256**.
All these operations must be performed with the computer switched off.
Take all precautions against the electrostatic discharges described in the main manual.

Carefully pull out the FLASH and insert the EEPROM, taking care not to bend the pins during insertion into the socket.
Match the right side of the EEPROM with the edge of the board and keep the notch on the left, in the same direction of the other Integrated Circuits, according to the picture above and to the serigraphy on the Circuit Board.

Check at least three times that there are no bent pins, from all sides.
Now, **reconfigure the jumpers** as shown in the picture on the right.
Take care of the second jumper from the top: 28xxx RW/RO.
If it will be placed **on the RW (Read/Write) side, the writing on the device will be enabled** and so you will be able to save your programs. Vice versa, the device will be in *Read Only* mode.

2. EEPROM MAPPING

The EEPROM 28c256 can store 32 kBytes. It has been divided in 4-kByte BLOCKS and will be mapped on the computer according to the following addressing scheme:

START ADDRESS		END ADDRESS
$4000	4 kB reserved for BASIC	$4FFF
$5000	Block 6 4 kB	$5FFF
$6000	Block 5 4 kB	$6FFF
$7000	Block 4 4 kB	$7FFF
$8000	Block 3 4 kB	$8FFF
$9000	Block 2 4 kB	$9FFF
$A000	Block 1 4 kB	$AFFF
$B000	Block 0 / 2 kB	$B7FF
$B800	RESERVED	$BFFF

Some parts have been used to store the Integer Basic and the various management programs, i.e. the traditional Program Manager and the saving utility that will be described shortly.
It is important to note that six 4-kByte Blocks and one 2-kByte Block have been provided.

3. OPERATIONS

The use of the EEPROM 28c256 and its programs is identical to what already seen in the main manual.
The Program Manager starts as usual with the command:

```
BD00R
```
followed by ENTER key (from now on shown with {ENTER})

The following will be displayed:

```
BD00: A5
&
```

Let us now request the list of available files using the familiar **D** (Directory) command:

```
&D  {ENTER}
  PAGE  2

  A  FREEBLK1  $0000-$0000
  B  FREEBLK2  $0000-$0000
  C  FREEBLK3  $0000-$0000
  D  FREEBLK4  $0000-$0000
  E  FREEBLK5  $0000-$0000
  F  FREEBLK6  $0000-$0000
  G  FREEBLK0  $0000-$0000
  H  BASIC     $E000-$F000
OK

&
```

NOTE: Ignore the page indication (PAGE), and the LEDs since the EEPROM capacity is 32 kBytes only.

The list of programs displays all Free Blocks (FREEBLK), numbered as in the previous map.
In this list you will find the programs that you will save in BASIC / Machine Language, in addition to BASIC itself.

3.1 SAVING BASIC PROGRAMS

Let us suppose that you have loaded the BASIC interpreter and already written your program.
To save your program you first have to return to WOZ Monitor by pressing the RESET key.
At the "\" prompt, launch the **Save Program** (*entry-point* is 0xB800) with the command:

```
B800R  {ENTER}
```

The following will be displayed:

```
B800: 20
#
```

The "#" symbol is the prompt generated by the Save Program. It indicates that it is ready to accept commands.
Unlike the Program Manager, the commands to be given are composed by one single letter.
Pressing the letter corresponding to a valid command causes the chosen command to be executed immediately.
Pressing a not allowed letter will cause a minimal Help to be printed on the screen:

```
#W/S/L/X?
```

The meaning of the commands is as the following:

- W (Write) writes onto EEPROM a segment of 4 kBytes starting from an arbitrary RAM address,
- S (Save) writes onto EEPROM a BASIC program,
- L (Loader) launch the Program Manager (or Loader, if you like),
- X (eXit) exits to WOZ Monitor.

Since we want to save a BASIC program, we will choose the command **S**.

Right after pressing the “S” key, this message will appear:

```
SAVE BASIC TO BLOCK:
```

Now you need to specify on which Block you want to save the BASIC program, considering that Block 0 cannot be used. Let us choose for example **Block 1**. After pressing the “1” key, another message will appear:

```
WITH NAME:
```

For example, write **TEST 1** and press ENTER to confirm.
Note that the program name can have a maximum of eight characters.
ONce the eighth character (or ENTER key) is pressed, the name will be considered complete **and the saving process will start immediately**.
The progress of the saving process is represented by a sequence of dots that gets longer:

```
. . . . . .
```

Each dot represents the writing of 256 bytes on EEPROM. At the end of the process 16 points will appear.
The writing is not immediate: it will take about 25 seconds to save 4 kBytes.
This time is absolutely normal and compliant with the operating specifications of the EEPROM in use.
At the end of the writing on EEPROM the Saving Program prompt will appear:

```
#
```

Now, if we want to verify that our program has been saved correctly we have the possibility to launch the Program Manager directly by pressing the **L** key.
The “&” prompt will immediately appear and you will be able to retrieve the program list, as usual:

```
#     press L → Note: L will not be shown.
&D    {ENTER}
  PAGE 2

  A  TEST 1     $0280-$1000 BAS
  B  FREEBLK2   $0000-$0000
  C  FREEBLK3   $0000-$0000
  D  FREEBLK4   $0000-$0000
  E  FREEBLK5   $0000-$0000
  F  FREEBLK6   $0000-$0000
  G  FREEBLK0   $0000-$0000
  H  BASIC      $E000-$F000
OK

&
```

We would remind you that the first address is the starting point of the program and the second is the first free address. After saving, your BASIC program becomes a permanent part of the programs stored in the EEPROM.
To load/reload it, simply give the command **LA** from the management program.
The same saving method applies for any other Block.
Note that there is no warning in case of overwriting a Block already written.
By default, the saving of BASIC programs has 0x0FFF as top address.
We will see in the following sections how it is possible to save programs that exceed this limit.

3.2 SAVING PROGRAMS IN MACHINE LANGUAGE

Programs/Routines in Machine Language can be stored anywhere in RAM memory.
They can be saved using the **W** command.
In addition to the name, (let us suppose it is TESTPROG) the hexadecimal start address will be requested.
Starting from that address all the following 4 kBytes will be copied to the chosen block of the EEPROM.
Let us suppose that the routine we want to save starts from address 0x0300.
Once we have launched the Save Program with **B800R** we will press the **W** key. The following message will appear:

```
SAVE MEMORY FROM: $
```

At this point, we will enter the starting address, in hexadecimal format (four digits) by pressing the **0300** keys.
When the fourth key is pressed, the starting address will be consolidated. You will then be asked for the file name:

```
WITH NAME:
```

Similarly to the previous example, you need to give a name to the file you want to save. Once you press the ENTER key or the eighth character of the name (**TESTPROG** in this example) the name will be consolidated.
Lastly, the request for the Block on which to save the program will appear:

```
TO BLOCK:
```

Choose a Block and type the corresponding number (e.g. **3**). **The saving will start immediately**.
A 2-kByte Mini Block, defined as Block 0, is also available if needed.
As in the previous example, the progress of the saving process will be represented by a sequence of dots lining up, followed by the "#" character at the end of the operations.

```
. . . . . .
#
```

Now it will be possible to verify the successful saving, which will appear in the list as item "C", as expected:

```
#       press L → Note: L will not be shown.
&D {ENTER}
PAGE 2

 A TEST 1    $0280-$1000 BAS
 B FREEBLK2  $0000-$0000
 C TESTPROG  $0300-$1300
 D FREEBLK4  $0000-$0000
 E FREEBLK5  $0000-$0000
 F FREEBLK6  $0000-$0000
 G FREEBLK0  $0000-$0000
 H BASIC     $E000-$F000
OK

&
```

The loading of the portion of memory containing your routine is done in the usual way.
In this example by using the **LC** command. Note the missing suffix "BAS" for TESTPROG.

3.3 HIMEM AND LOMEM

The BASIC commands LOMEM and HIMEM are used to change the amount of RAM available for BASIC programs. LOMEM defines the *lower boundary* and HIMEM the *upper boundary* of the memory area dedicated to the programs. The default values of these settings are: LOMEM = **0x0800**, HIMEM = **0x1000**.
This limits the RAM available to the user to:
0x1000-0x0800=0x0800. In decimal, this value corresponds to 2048 bytes, 2 kBytes.

The Save Program, however, **takes as default value for LOMEM the address 0x0280**, and therefore saves the contents of all "unused" memory from BASIC with the default settings.
The reasons are listed below:

- In the Apple-1 architecture the address 0x0280 (640 in decimal) is the first address of free RAM that is not used by the computer itself. Therefore, it represents the absolute minimum value allowed for the LOMEM command.
- There are 1408 bytes available from address 0x0280 to address 0x0800, which can be used by both BASIC and other programs.
- It is not uncommon to find BASIC programs where the first command is a LOMEM. This saving system ensures the compatibility of such programs.
- It is also quite common to allocate the Machine Language routines in the range 0x0280..0x07FF so that they can be called from BASIC by the appropriate CALL.
- This system guarantees in this case also the automatic and transparent saving of such routines, together with the actual BASIC program.
- Last but not least: the saving of the entire memory range from 0x0280 to 0x0FFF (3456 bytes in total, plus other pointers related to the BASIC itself) reduces the amount of unused space on the EEPROM.

The reasons given above are also the reason why BLOCK 0 cannot be used to save programs in BASIC: it is simply not wide enough.

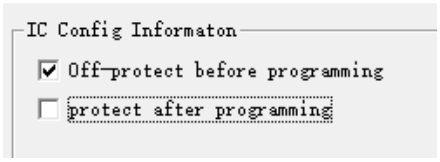
The HIMEM command works in a similar way, moving up the limit of the memory that can be used by the BASIC program.
If you want to write a very long BASIC program (let us assume a total of 6 kBytes), you should first give the command HIMEM=8192 (0x2000 in hexadecimal).
Since LOMEM has not been touched, the RAM available for the program would be 0x2000-0x0800=0x1800, equivalent to 6 kBytes, and this would comply with your needs.

However, how can you save our 6 kBytes BASIC program if the size of the Blocks dedicated to BASIC is only 4 kBytes?
The solution is **to use a 4-kByte Block to save the first part** of the program and **a second 4-kByte Block for the rest of it**.
The first part of the program (from 0x0280 to 0x0FFF) will be saved as a traditional BASIC program (obviously incomplete) with the **S command**.
The second part of the program (from 0x1000 to 0x1FFF) will be saved with the **W command** (specifying as starting address the location $1000). This will save the remaining 4 kBytes.
To reload the program correctly, it will be necessary to load both parts.

4. LIMITS AND WARNINGS

Saving on EEPROM should be considered experimental, but it allows you to save your work relatively safely and, most importantly, to reload it instantly.
We do not assume any responsibility, of course, for damage, even serious or fatal, caused to people / things / intellectual property during the installation or use of this device.
Please note that:

- **All Blocks are rewritable** tens of thousands of times, but there is no UNDELETE function.
  Once a Block (or a single byte) has been rewritten, the previous content is lost forever.
- Within the Save Program, there is no key or "Undo Operation" function.
  If you make mistakes while entering names/blocks/addresses, and writing to EEPROM has not yet started, the only way to quit the Save Program is to use the RESET key and start over.
- Placing the jumper **in the RW position makes ALL the EEPROM rewritable**, not just the Blocks listed above. This means that when you give to WOZ Monitor a write command in the address range 0x4000 ... 0xBFFF the contents of the EEPROM will be changed.
- If improperly altered data are related to BASIC, or Management Programs, or even worse to your programs, the computer may malfunction and you may lose valuable data.
- For this reason always use extreme caution with the WOZ Monitor commands when the write mode RW is enabled.
- The manufacturer of the EEPROM states that the content you write on the EEPROM will not degrade for at least ten years. For a computer that is more than forty years old, that is not too much... so: **MAKE A BACKUP!** Backups will also be your insurance for accidental writes.
- If you should decide to program a 28c256 yourself, be sure to leave the write protection DISABLED at the end of the writing.
  In the TL-866 programmer, for example, the choice is made by unchecking the *protect after programming* checkbox.
  Always refer to your EEPROM programmer's manual.



- If the EEPROM is write-protected by the feature described above (or because you have not moved the jumper to the RW position) the Write Program will freeze. The sequence of dots that normally indicates that writing is in progress will not appear and the only possible operation will be a RESET.

We realize that the functions described in this manual are very simple, but we have chosen to give priority to functionality over appearance, with particular focus on maximizing the memory available to you.

5. API

The copy routine from RAM to EEPROM (and vice versa) is allocated by the Write Program from 0x023A to 0x027F (at the end of the keyboard buffer, to optimize RAM usage).

You can use it outside of the management programs to copy arbitrary memory segments.
The routine is available after launching the Write Program at least once (e.g. **B800R** {ENTER} followed by **X** or RESET).

Six bytes in Zero Page define the start and destination addresses and the amount of bytes to be copied:
$40-**$41**: source address (i.e. 0x0280)
$42-**$43**: destination address (i.e. 0xA000)
$44-**$45**: bytes to copy (i.e 512 bytes, 0x0200 in hexadecimal)

Using WOZ Monitor enter the desired addresses (in Little Endian format, according to the examples above):

```
40: 80 02 00 A0 00 02
```
{ENTER}

Now execute the copy routine:

```
23AR
```
{ENTER}

The usual sequence of dots will appear until the copy is finished:

```
023A: A0..
```

The copy routine was not designed to be launched by the WOZ monitor, but by a program.
It is therefore possible that at the end of the process the routine will remain "hanging" without releasing the terminal. In this case, just press the RESET key.
There are many reasons for this behaviour and they will not be covered in this document, however they are not an indication of malfunction and usually do not affect the success of the copy.

Moreover, since the routine is allocated at the end of the keyboard buffer, it is also possible that extra characters are printed: ignore them.
The copy routine works in both directions: RAM→EEPROM and EEPROM→RAM, just set the Zero Page pointers properly.

All the programs saved with this procedure will be "invisible" to the Management Program.
If you want to use this saving methodology, we suggest you to dedicate an entire Block to it according to the map described in section 2.
We recommend you to annotate all the information relevant to the retrieval of your programs.

Happy programming with *Apple-1 Juke-Box!*

APPLE-1 JUKE-BOX

INFO | ORDERS | SUPPORT: p-l4b @ protonmail.com