APPLE-1 microSD STORAGE CARD

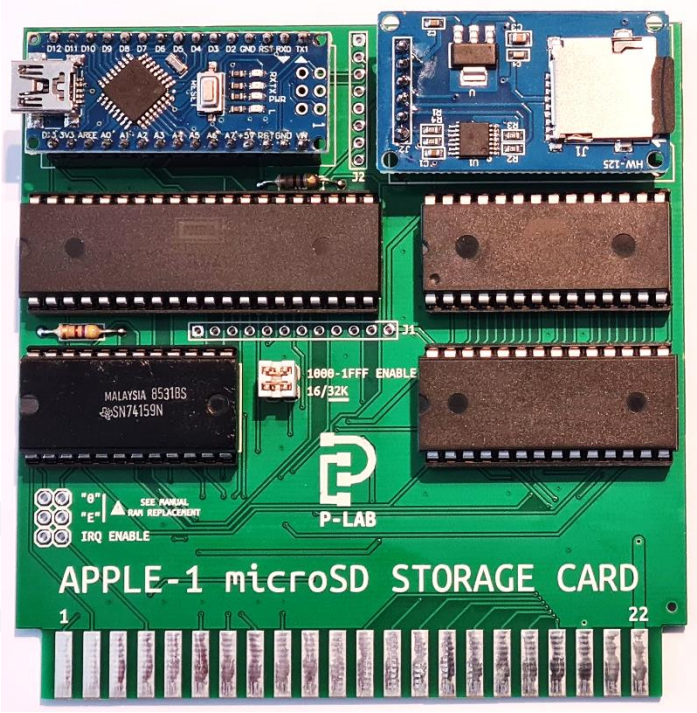
SD CARD OS 1.1 – PCB versions 1.13/1.16

Apple-1 microSD Storage Card is an electronic card that allows your Apple-1 computer, whether Original or Replica, to access and manipulate the files contained on the removable memory card.

It provides new experiences in using your Apple-1 computer by allowing you to save programs you wrote, add more programs easily, and many other interesting features.

This document includes:

1. Package contents
2. Warnings about the risk of damage due to electrostatic discharges
3. Product description and features
4. Instructions for setting up and installing
5. Operations
6. Jumpers set up
7. Final notes



Picture above is for illustration purpose only. Actual product may vary due to due to product enhancement or availability of components.

1. PACKAGE CONTENTS

- *Apple-1 microSD Storage* card, complete with microSD card.

2. DAMAGES FROM ELECTROSTATIC DISCHARGES

Apple-1 microSD Storage Box is sensitive to static electricity, *just like your Apple computer*, and may be damaged by it. Before any operation on your devices, you must discharge the static electricity accumulated by your body and prevent it from building up again. We do not accept any responsibility for damage, even serious or fatal, caused to people / things / intellectual property during the installation or use of this device.

3. PRODUCT DESCRIPTION AND FEATURES

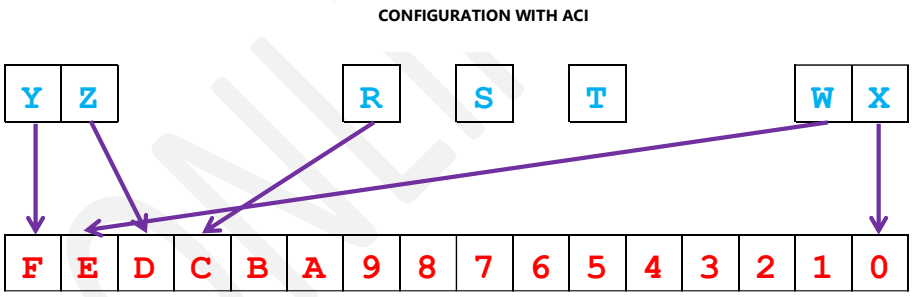
Apple-1 microSD Storage Card is based on the popular *Versatile Interface Adapter (VIA)* 65C22 chip, which in combination with the ATMEGA 329P microcontroller and a custom firmware, performs read/write operations on the memory card filesystem.
The user interface is text-based and is very similar to the **MS-DOS command prompt or the Linux CLI**. Even the commands available to the user are very similar, when not equal, to those of the operating systems mentioned.
Therefore, it will be possible to load / save / copy any kind of file. You will also be able to create and manage directories and subdirectories at your convenience.

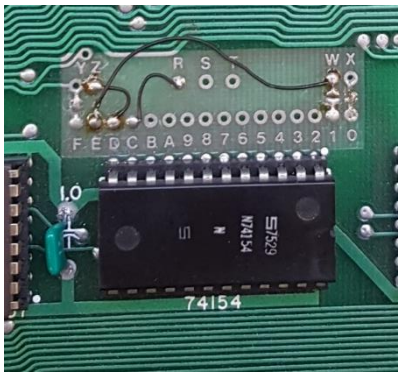
The board also includes a memory expansion that brings the computer to **32 kB contiguous RAM, plus the 4 kB dedicated to BASIC**, for a total of 36 kB. This configuration allows you to run basically any program for Apple-1.
In case of failures, or for testing purposes, the board is also able to replace the RAM on the motherboard, as it will be described in the **Appendix 2** - *Memory Replacement*.

4. INSTALLATION ON THE COMPUTER APPLE-1

4.1 INITIAL CHECKS

Apple-1 must be set up with the configuration "WITH ACI", with the following connections in the CHIP SELECT area:





The picture aside shows the necessary connections/wires:

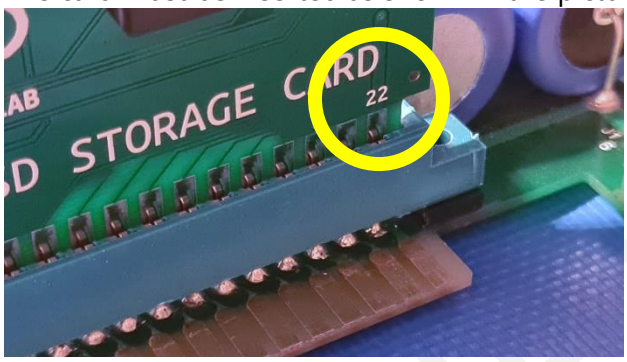
These connections are standard on most of the Apple-1

in circulation, both Originals and Replicas.

Therefore, no supplementary connections are necessary.

4.2 CARD INSTALLATION

The card must be inserted as shown in the picture:



The number "22" on the card and the number "22" on the motherboard must be on the same side.

The number "22" on the card must therefore face the outer side of the Apple-1 motherboard.

Using the card in combination with other cards and/or devices such as BUS EXTENDER could lead to malfunctions: do not use this kind of devices.

WARNING: Switching on the computer with the card incorrectly oriented **INSTANTLY DAMAGES** the computer and the card itself.

The microcontroller is equipped with a USB port that can be used for debugging.
To prevent any damage, the USB cable should be connected to external devices **only when the computer is turned on**. Under no circumstances should an external device supply power to the microcontroller when the Apple-1 computer is off.

5. OPERATIONS

After you switch on your Apple-1 computer, press the CLEAR SCREEN and RESET keys as usual.
The *entry point* of the user interface program is $8000, so it must be called with the command:

```
8000R
```
and press ENTER (from now on indicated with {ENTER})

The command prompt of the **SD CARD OS** environment will appear immediately:

```
8000: A9

*** SD CARD OS 1.1

/>
```

You can now issue the commands you like to browse folders, move between them, load files, etc. The current directory (/ is the root directory) will be displayed before the prompt >.
If necessary, two levels of help are available. The first level is:

```
?
```
{ENTER}

after that, the list of all available commands will appear immediately:

```
COMMANDS LIST:
READ, WRITE, DIR, TIME, LOAD, RUN, SAVE, TYPE, DUMP,
ASAVE, BAS, DEL, LS, CD, MKDIR, RMDIR, RM, MD, RD, PWD,
TEST, HELP, ?, MOUNT, EXIT.

USE HELP COMMAND FOR MORE DETAILS
```

The second level shows the syntax of each available command by typing the word `HELP` followed by the command for which you need help. Example:

```
HELP SAVE
```
{ENTER}

The desired item will appear immediately:

```
SYNTAX:
SAVE FILENAME [START] [END]

SAVES A FILE TO THE SD CARD. IF START AND END ARE
SPECIFIED, A BINARY FILE WITH TAG #06 WILL BE CREATED
WITH THE MEMORY CONTENT FROM THE ADDRESS RANGE START-END
(INCLUDED). IF START AND END ARE NOT SPECIFIED, THE
BASIC PROGRAM CURRENTLY LOADED IN MEMORY WILL BE CREATED
WITH THE CORRESPONDING #F1 TAG.
```

The DIR command displays all files in the current directory, along with other information summarized here:

```
/BASIC>DIR    {ENTER}
FACTORIALS     32767 BAS $0200
WORDSEARCH     16128 BAS $0300
BASIC           4096 BIN $E000
CONCENTRATION 16128 BAS 40300
PRIMEFINDER     2560 BAS $0800
...
/BASIC>
```

File size in bytes *File type* *Start Address*

The directory listing can be **temporarily paused** by pressing any key (except ESC).
To resume viewing the remaining files, press the **ENTER** key.
To interrupt the list before all files have been displayed, press the **ESC** key.

BACKSPACE functionality has also been implemented: pressing the undescore "_" key during any typing will produce a new line without the last character typed.
This allows you to correct any typing errors **without retyping the entire line**.
Example:

```
BASIC>LOAD CONCENTARTI    press underscore _ key
>LOAD CONCENTART          press underscore _ key
>LOAD CONCENTAR           press underscore _ key
>LOAD CONCENTA            press underscore _ key _
>LOAD CONCENT             type normally
>LOAD CONCENTRATION
```

Please Note: These features may not be available for some keyboards.

Other commands have been designed to **speed up and facilitate the loading and execution of programs**. One of them is the **RUN** command that loads and runs BASIC or ML programs without having to go back into the BASIC environment or exit to the WOZ monitor.

Another important feature of **SD CARD OS** is that it searches for the file with the closest name to the one typed in if the exact match is not found.
For example, if you want to run the BASIC program "Star Trek" directly, simply type in the commands in red:

```
/> CD BASIC {ENTER}          ← change working directory to BASIC
/BASIC> LOAD BASIC {ENTER}   ← load BASIC
FOUND BASIC#06E000           ← SD CARD OS found a file with an exact or similar name
LOADING                      ← the file is being loaded
```

```
BASIC#06E000                          ← loading is completed
E000.EFFF (4096 BYTES)                ← the RAM area where the program has been loaded is shown
OK                                    ← end of operations
/BASIC> RUN STARTR    {ENTER}         ← loads & run the program beginning with STARTR
FOUND STARTREK#F10300                 ← SD CARD OS shows the best match and loads the file
LOADING                               ← the file is being loaded

STARTREK#F10300                       ← loading is completed
(LOMEM=$0300 HIMEM=$1000)             ← the RAM area where the program has been loaded is shown
OK                                    ← end of operations
TYPE A NUMBER ?                       ← STAR TREK is now running
```

If you want to save your programs **written in INTEGER BASIC**, you must exit the WOZ monitor using the RESET button.
After that you need to launch the management program with 8000R and use the SAVE command:

```
8000R     {ENTER}

8000: A9

*** SD CARD OS 1.1

/>SAVE TEST  {ENTER}
SAVING

TEST#F10800
(LOMEM=$0800 HIMEM=$1000)
OK
/>
```

Your program will be **saved in the current directory** (the root directory / in the example). HIMEM and LOMEM settings of the BASIC will also be saved. In the example the default values are shown.
If you want to check that your program has been saved, you can issue the LS or DIR commands to get also an idea of the different nomenclature displayed:

```
/>LS      {ENTER}
2560 TEST#F10800

/>DIR     {ENTER}
TEST          2560 BAS $0800
```

The real name of the file is displayed by the LS command, the DIR command interprets it in a more readable way. In **Appendix 1** you will find a detailed description of this naming convention.

Note: the delete RM and DEL commands require the real file name.

To continue working on your program: exit to the WOZ monitor by pressing the RESET key (or with the EXIT command) and re-enter the BASIC environment with the usual E2B3R command.

SD CARD OS provides many other commands that will allow you to save on SD card not only your programs written in INTEGER BASIC, but also the binary programs.

In addition to this, compatibility with **AppleSoft Basic Lite** has been maintained, allowing you to write and execute more complex programs than those possible with INTEGER BASIC.
Full details are described in **Appendix 3** of this document.

The firmwares, both on-board EEPROM and the microcontroller, have been made available by their creator, **Antonino Porcino**, on the following GitHub repository:

https://github.com/nippur72/apple1-sdcard

For your convenience, the list of commands and their syntax are also listed in **Appendix 1** of this document.

The SD memory card can be safely removed at any time if there are no read/write operations in progress. After replacing the card, **press the RESET button of the microcontroller** (not the RESET of the computer) so that the card is read correctly by **SD CARD OS**.
Alternatively, you can use the command MOUNT.

These operations have no effect on the content of the RAM of the computer nor on its operations.

In case of error messages (e.g. I/O ERROR) during the operations: exit from the management program with the computer RESET button and re-enter with 8000R.

6. JUMPERS CONFIGURATION / EXPANSION CONNECTORS / IRQ

The few configurations of the board are made by some jumpers/pads whose operations are described below.
Also, there are two auxiliary connectors that allow you to use the unused I/O lines of the 65C22 chip and the microcontroller: you can use them for your projects, while continuing to use all the services offered by the board itself.

6.1 EXPANDED RAM ALLOCATION

Normally these two jumpers are inserted in their position and they enable the contiguity of RAM from address $0000 to address $7FFF (32 kB). By removing these jumpers, which are completely independent from each other, it is possible to create breaks in the mapping as specified below:

1000-1FFF ENABLE JUMPER — If you remove this jumper the board will no longer address the RAM between $1000 and $1FFF (4 kB total), which will be unallocated.

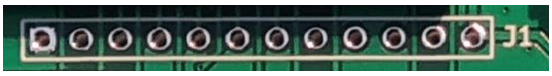
16/32K JUMPER — Removing this jumper will limit the RAM expansion to the first 16 kB, up to address $3FFF. Addresses $4000 to $7FFF (16 kB) will be unallocated and therefore available to other devices/peripherals.

2000-3FFF ENABLE JUMPER — **(PCB version 1.16 only)** If you remove this jumper the board will no longer address the RAM between $2000 and $3FFF (8 kB total), which will be unallocated.

6.2 EXPANSION CONNECTORS

The **J1** connector maps the unused 65C22's I/O ports:

PIN	1	2	3	4	5	6	7	8	9	10	11	12
PORT	GND	N/C	PB2	PB3	PB4	PB5	PB6	CA1	CA2	CB1	CB2	+5V



(PIN 1 has a square shape, and is located on the left)

The **J2** connector shows the microcontroller's unused I/O lines:

PIN	PORT
1	GND
2	A7
3	A6
4	A5
5	A4
6	A3
7	A2*
8	+5V

(PIN 1 is the square-shaped one, located at the top)

* Line A2 is also connected to line PB1 of the 65C22. This connection is not currently used by the firmware but is wired for future use. Use with caution to avoid any damage to the 65C22 or the microcontroller.

Please Note: Mapping of 65C22 <-> microcontroller lines is defined in the software/firmware source code. If you need these lines for your own 65C22-based projects, remove the microcontroller and the SD card reader and use their connectors for your wiring.

6.3 IRQ

A dedicated line has been provided for the IRQ line from the 65C22 to the 6502 of the Apple-1. Connect the pads shown in the picture if you want to use 65C22's interrupt-generating feature (e.g. timer) for your projects.

If you wish to write programs that use the 65C22, the base address of the device is $A000.

If you need to use the lines currently assigned to the microcontroller or to the SD card reader, these devices are removable.
Together with the J1 and J2 connectors described above you will have the possibility to use all the I/O lines of the 65C22.

If you need EEPROM memory for your code, the address range is 8 kB wide, from $8000 to $9FFF.

7. FINAL NOTES

The SD memory card must be formatted to **FAT32** standard.
If your operating system does not provide this option, you can use any external program capable of doing so: the standard is well documented, and many third-party utilities can do that.

For moving/copying/renaming files no special tools are needed, any file browser will work without problems.

If you update the files on the SD Card with an external computer remember to manually remove any "service" directory automatically created by your Operating System, such as: *System Informations / Deleted Items* etc. They may affect the display of the content on the screen.

The *tagged file naming*, better described in **Appendix 1** of this document, is compliant with the PRODOS/CFFA1 standard. Therefore, it is possible to transfer files between the two architectures without having to manipulate or convert the files.

Please note that SD memory cards, in general, **are not permanent storage devices**.

It is therefore essential to always make a backup copy of the contents of the SD card to avoid accidental data loss.

The product is constantly evolving: visit the reference page often:

https://p-l4b.github.io/sdcard

to be always up to date about news and updates!

Have fun with Apple-1 microSD Storage Card !!

APPLE-1 microSD
STORAGE CARD

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

P-L4B @ PROTONMAIL.COM

APPENDIX 1 - https://github.com/nippur72/apple1-sdcard

SD CARD

- all numbers must be provided in hexadecimal format unless specified
- arguments in [ ] brackets means they are optional
- nested paths are allowed with the / character, e.g. /, /folder1/foo
- no path given implies current working directory

TAGGED FILE NAMES

File names containing a tag **#** character have a special meaning: the part after the **#** indicates the file type (two characters) and the hex loading address (4 characters).
#06 for plain binary files
#F1 for INTEGER BASIC programs
#F8 for AppleSoft BASIC programs
E.g.:
BASIC#06E000 is binary file named **BASIC** that loads at address **$E000**.
STARTREK#F10300 is a BASIC program named **STARTREK** that loads at address **$0300**.
Tagged file names are used by the LOAD, RUN, SAVE and DIR commands to simplify working with files.
For example to execute the above files, it's enough to type:

```
LOAD BASIC
RUN STARTREK
```

COMMANDS

READ filename startaddress
Reads a binary file from the SD card and loads it in memory at the specified address.

WRITE filename startaddress endaddress
Writes the memory range from startaddress to endaddress (inclusive) in a file on the SD card.

TYPE filename
Reads the specified ASCII file from the SD card and prints it on the screen. Press any key to stop the printing and return to the command prompt.

DUMP filename [start] [end]
Reads the specified binary file from the SD card and prints it on the screen in hexadecimal format. start and end are optional and are used to print a smaller portion of the file. Press any key to stop the printing and return to the command prompt.

LOAD filename
Loads a file from the SD card. `filename` refers to a "tagged file name" described above. For convenience, `filename` can be partially given, the first matching file will be loaded.

SAVE filename [start] [end]
Saves a file to the SD card.
If `start` and end are specified, a binary file with tag `#06` will be created with the memory content from the address range `start`-end (included).
If `start` and end are not specified, the BASIC program currently loaded in memory will be created with the corresponding `#F1` tag.

ASAVE filename
Saves an *AppleSoft BASIC lite* file to the SD card. The program currently loaded in memory will be created with the corresponding #F8 tag.

RUN filename
Same as `LOAD` but runs the file after loading it. Binary files are executed at the starting address specified in the file name tag; BASIC files are `RUN` from the BASIC interpreter.

DEL filename - RM filename
Deletes a file from the SD card.

DIR [path] - LS [path]
Lists the files from the specified directory, or from the current directory if no path is given. `LS` has a shorter but quicker output format. Press any key to stop the file listing and return to the command prompt.

CD path
Changes the current working directory to the specified path. The current directory is also shown in the command prompt.

MD path - MKDIR path
Creates the specified directory.

RD path - RMDIR path
Removes the specified directory. The directory to remove must be empty (no files or directories within).

PWD
Prints on the screen the current working directory.

BAS
Prints `LOMEM` and `HIMEM` pointers from the BASIC program currently loaded in memory.

MOUNT
Force the remount of the filesystem of the memory card.

address R
Runs the program loaded at the specified memory address. Useful addresses: 6000R AppleSoft BASIC *cold start* (needed at least once) 6003R AppleSoft BASIC *warm start* (it does not destroy the BASIC program in RAM) E000R Integer BASIC *cold start* EFECR Integer BASIC "RUN" command (can be used as a *warm entry point*) 8000R SD card OS command prompt.

TIME value
Set the internal timeout value used in the I/O operations with the SD cards.

TEST
Internal test.

?
Shows the list of available commands.

HELP command
Shows a detailed help for the given command.

EXIT
Exits to the WOZ monitor.

APPENDIX 2 – MEMORY REPLACEMENT

This feature can be useful in case of testing, failure, or unavailability of components.

The operations described below should be considered temporary and at your own risk.

It is mandatory to put in place all the safety procedures against the accumulation of electrostatic charges already described in this manual.

Apple-1 can have up to 8 kBytes of RAM on the mainboard. They are organized in two 4 kBytes "banks" with eight chips each, for a total of sixteen chips.

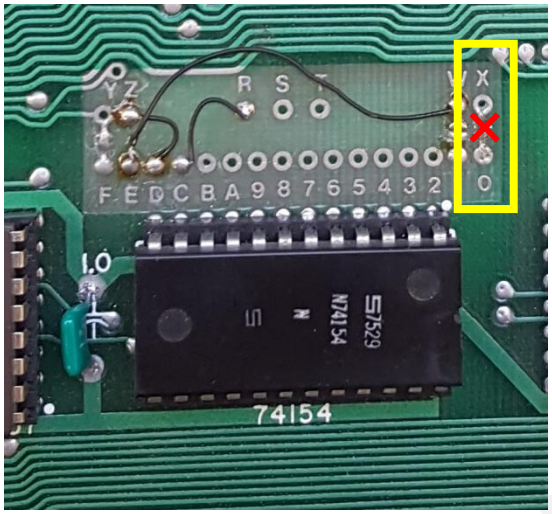
According to the "with ACI" configuration described in the Computer Operating Manual and mentioned in this manual, they are divided as follows:

- Bank "0", addresses from $0000 to $0FFF, 8 chips from position B11 to B18.
- Bank "E", addresses from $E000 to $EFFF, 8 chips from position A11 to A18.

Follow the instructions in section 7.1 2 (or both, if necessary), depending on the bank you wish to replace.

7.1. SUBSTITUTION OF MEMORY BANK "0" (Addresses from $0000 to $0FFF)

7.1.1 On Apple-1: Open/desolder the connection, shown in the picture, between the pad "**X"** and pad "**0"**:



There must be no electrical connection between the two pads.

7.1.2 Remove all eight memory chips from position B11 to position B18.

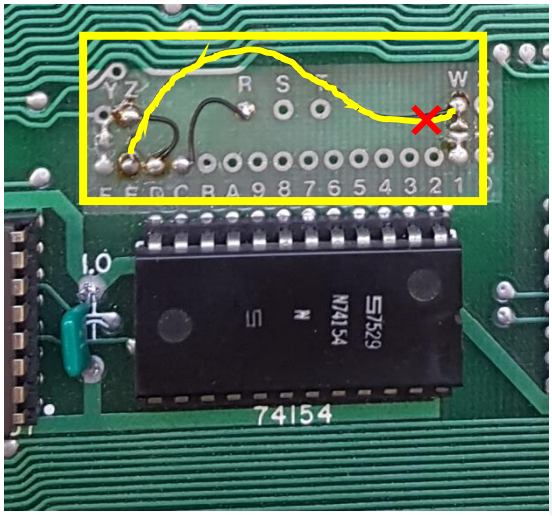
7.1.3 Use a jumper or a temporary wire to connect the pads shown in the yellow box below.

7.1.4 Switch on the computer and operate normally.

At the end of the test/troubleshoot/etc. restore the original connections, put the memory chips back in place and remove the temporary connection on the *Apple-1 microSD Storage Card*.

7.2. SUBSTITUTION OF MEMORY BANK "E" (Addresses from $E000 to $EFFF)

7.2.1 On Apple-1: Open/desolder the connection, shown in the picture, between pad "**W**" and pad "**E**":



Desolder the wire on the "**W**" side.

Insulate the freshly desoldered end of the wire to ensure that it does not make accidental contact with other components.

7.2.2 Remove all eight memory chips from position A11 to position A18.

7.2.3 Use a jumper or a temporary wire to connect the pads shown in the yellow box below.

7.2.4 Switch on the computer and operate normally.

PLEASE NOTE:
In this configuration the memory segment from 0x6000 to 0x6FFF will not be usable. In "RAM 32kB" configuration, therefore, the maximum contiguous memory size will be limited to 24 kBytes (addresses from 0x0000 to 0x5FFF) instead of 32 kBytes. The "RAM 16kB" configuration, instead, will operate normally.

At the end of the test/troubleshoot/etc. restore the original connections, put the memory chips back in place and remove the temporary connection on the *Apple-1 microSD Storage Card*.

APPENDIX 3 – APPLESOFT BASIC LITE

AppleSoft BASIC is a variant of Microsoft BASIC (hence the name) that was developed in 1977 as an improvement for INTEGER BASIC for Apple II computers.
It has a larger number of commands and instructions but most importantly it can handle floating point numbers. Graphical functions are not available, of course.

For the "*Replica-1*" project, a modified version has been realized specifically, available here:
https://cowgod.org/replica1/applesoft/

To guarantee the operation also with *Apple-1 microSD Storage Card* a special version has been realized, capable of providing LOAD/SAVE/MENU commands as in the original version, although with some differences.

On to the SD card there is a special folder called ASOFT, with AppleSoft BASIC interpreter and an example program in it.

STARTUP

As with traditional INTEGER BASIC, AppleSoft BASIC **must be loaded and run first at least once**. In this example we assume that we are already inside the management program, the commands to be typed are in red:

```
*** SD CARD OS 1.1

/>CD ASOFT    {ENTER}
/ASOFT>RUN APPLESOFT    {ENTER}
FOUND APPLESOFT-SD#066000
LOADING

APPLESOFT-SD#066000
$6000-$7FFF (8192 BYTES)
OK

*** APPLESOFT BASIC LITE SD V1.1 ***

22525 BYTES FREE

]
```

Now you can write your programs in AppleSoft BASIC.

LINE EDITOR

A **line editor** has been implemented which allows you to **delete the last character typed** by pressing the BACKSPACE (Control H) key:

```
] 10 PRINT "THIS IS A TEST    → press BACKSPACE or Control H)
] 10 PRINT "THIS IS A TES     → press BACKSPACE or Control H)
] 10 PRINT "THIS IS A TE      → press BACKSPACE or Control H)
] 10 PRINT "THIS IS A T       etc.
```

Please Note: This feature may not be available on some keyboards.

LOAD A PROGRAM

To load a program from SD card, *overwriting the current one in memory*, simply issue from the AppleSoft BASIC prompt the LOAD command followed by the name of the desired file within double quotes, e.g.:

```
]LOAD "LEMO"          {ENTER}
FOUND LEMO#F8801
LOADING
LEMO#F8801
$0801-$23F8 (7160 BYTES)
OK
]RUN          {ENTER}
```

SAVE A PROGRAM

To save your programs you can use the SAVE command. The syntax is the same of the LOAD command:

```
]SAVE "MYPROG"    {ENTER}
SAVING

MYPROG#F80801:
$0801-$080C (12 BYTES)      → the final address and size will be related to your program
]
```

SD CARD OS INTEGRATION

You can enter the **SD CARD OS** main menu directly from AppleSoft BASIC, using the MENU command:

```
]MENU        {ENTER}
*** SD CARD OS 1.1
/ASOFT>
```

You can load and save your AppleSoft BASIC programs also from **SD CARD OS** environment, by using the dedicated **ASAVE - AppleSoft SAVE** - command and not the traditional SAVE command.
The SAVE command is dedicated exclusively to INTEGER BASIC programs. Do not use it, use ASAVE instead.

It is also necessary to re-enter the AppleSoft BASIC environment by the ***warm entry point* 6003**, otherwise the program currently in memory will be deleted:

```
/ASOFT>6003R        {ENTER}
]
```

As you can see in this example, *warm-starting* AppleSoft BASIC via address 6003 does not cause the welcome message to appear.

Programs saved with AppleSoft BASIC will have tag #F8 and extension "ASB" (AppleSoft BASIC).

We suggest practicing with the various menus and loading and saving commands to avoid data loss.