



# FLASHJACKS

## User manual

Aquijacks 2018 v1.1



## Index

Introduction.....	3
Technical specifications.....	5
Hardware features:.....	7
User's manual Configuration.....	9
Introduction.....	9
The proprietary operating system.....	11
The Slot expander.....	15
The FLASHROM unit.....	17
Load and save memory blocks.....	20
The RAM MAPPER unit.....	24
The MEGARAM unit.....	25
The SINFOX compatible unit.....	28
The SUNRISE unit compatible.....	30
The sound system of FLASHJACKS.....	33
The PSG module.....	36
The SCC module.....	38
The FM module.....	40
The audio bus.....	42
Other aspects general configuration.....	44
Loading ROMS from MSXDOS. "FROM.COM".....	47
A bit of MultiMente.....	51
Another bit of SDs and files.....	52
Epilogue.....	54

## Introduction

*I believe that since I have memories I have always been an enthusiast of electronics and computer science. Back in the eighties, I was mesmerized with computers and what they could do. For my life, the MSX world has crossed several times and I have always been fascinated by that fantastic computer that, curiously, not everyone was following.*

*I do not know if it was the music, the Japanese software companies or what I know but the MSX had something that hooked.*

*You may ask, why tell us a story if this is an instruction manual? Imagine that when reading the instructions of a microwave the designer began to tell you his life of how he got to that job.*

*Well, from the beginning I want to comment that this is not a microwave and I do not have a company, nor do I manufacture this device in a chain, nor do I have commercial purposes as if it were a game of today.*

*The hours of commitment and sacrifice that have been invested in this, if viewed by the commercial side, do not compensate by far.*

*It is a dream, a desire, a purpose. Something that with my knowledge and my impetus I have decided to build and I hope that your final result is to your liking as it has been for me.*

*Once commented on the concept of this dream and made it very clear that it is "the madness" of an MSX fan, let's discuss the reason for its existence.*



In this last encounter with the MSX, back in 2013, I observed that there was a lot of design electronics by amateurs that covered one or another lack of the system either due to obsolescence and non-existence of the original system or due to the need to adapt to new storage media.

And then I thought, Why not a cartridge that does everything? Why not also something that really sounds like it should have come out in its day?

And here you have the result. More than 2,000 hours of study and sacrifice, more than 23,000 LE blocks and a lot of code failures behind me have led to what you have in hand.

Something out personally benefit for the computer to be targeted and the century that we find where the 3D virtual world go hand in hand. Something that will go unnoticed for the vast majority because it is an object that is dedicated to paying homage to something obsolete and outdated. Something that no current company in their right mind would invest the technique and development that have been added to obtain a product that the "mass media" neither knows nor will know what it is for.

Without further delay, and because I do not want this to become a novel, we give way to its technical description and operation manual.

Thank you for sharing this moment together and I wish my dream could be enjoyed by all of you.

## Technical specifications.

### *Synthesized devices:*

- 1 SLOT Expander. With the possibility of canceling or activating its four subslots for the internal insertion of any device synthesized and shown in this manual. Any device can use any subslot that we say.
- 4 FlashROM units with 14 load mappers and several synthesized models of EPROMS that are self-selected by load predictive model (AMD, ATMEL, MICROCHIP, etc ...). The mappers have a predictive system that mixes the heuristics with a database. Each FlashROM unit has a loading capacity of up to 4 Megabytes each. All units have the possibility to use erasing and writing codes, etc .. allowing you to read and record infinitely many times until it is turned off. It does not deteriorate due to the number of recordings. Access speed lower than 40nS and even 2nS peaks.
- 4 MAPPER RAM units compatible even with the MSX1. Configurable from native 64kB (for some MSX primitives) up to 4 Megabytes each. Access speeds lower than 40nS and even 2nS peaks. Autorefresh cartridge without using the MSX itself.
- 4 units MEGARAM compatible. Configurable up to 2 Megabytes each. Access speeds lower than 40nS and even 2nS peaks. Autorefresh cartridge without using the MSX itself.



- 1 compatible SINFOX unit until you see 4.04 with access to SDCARD. Access to your EPROM of origin with your mapper and with the possibility of writing both in card and in EPROM.
- 1 SUNRISE compatible IDE unit. Autoconversion and on-the-fly operations to SDCARD card. The data is transformed IDE <-> SD transparently to the user. Compatible with NEXTOR SUNRISE IDE drivers See 2.1 Alpha2. Only compatible in LBA mode. Not compatible in CHS or other mode. Total read and write compatibility and SD tab against accidental writing. Access to SDCARD compatible SDSC and SDHC up to 4GB which is what the NEXTOR driver allows. Future possibility of increasing capacities with modification of the drivers. Compatible extraction and insertion SD card in hot. All of them managed with a driver adapted for the FlashJacks.
- 1 selectable PSG Stereo sound unit. Sampling rate 48khz 16bits per channel. No loss of slot.
- 2 selectables SCC Stereo sound unit. Sampling rate 48khz 16bits per channel. Selectable the Slot or occupied subplot. This device allows double slot occupancy.
- 1 selectable Stereo FM sound unit. Sampling rate 48khz 16bits per channel. No loss of slot for capturing audio. In addition, you can add a FMPAC unit with full synthesization where the internal soft is included with its original mapper and even an 8kbyte SRAM with predictive autosave.

## *Hardware features:*

- Several synthesized mappers (Auto, KONAMI4, KONAMI5, ASCII8K, ASCII16K, SUNRISE, SINFOX, ROM16K, ROM32K, ROM64K, RTYPE, ZEMINA6480, ZEMINA126, FMPAC, CROSSBLAIM, SLODERUN)
- 32 Mbytes of DDR RAM with self-management of refresh and capacity according to different configurations. 2nS access
- Multiple data buses at different frequencies. Walkways between them and interdependent. (300Mhz, 140Mhz, 50Mhz, 3.58Mhz, MSX Clock, 24Mhz among others).
- Own operating system compatible FAT12, 16 and 32 made entirely with synthesized logic systems (There is no synthesis of a CPU that manages the OS). This system is interdependent of the MSX so it allows the loading and management of all components, ROMS, etc ... before you even start the MSX. This OS is probably unique in the world since I have not found any synthesization of an OS with logic gates.
- No CPUs for sequential programming. Each sequential model follows its own Turing pattern independently so that the speed is maximum and simultaneous in all modules. There are no delays or intermodular involvement.
- Bidirectional sound system up to 96khz to 24 bits in Stereo for maximum experience.



- Stereo audio bus independent of the MSX, so the PSG+SCC+FM is mixed and treated internally. This results in a very low noise level and maximum sound purity. (The volume problem is eliminated and mixed with external cartridges).
- Filtering the audio by synthesizing and by physical hardware. Selectable
- Output and audio input by Jack with power capacity to handle headphones.
- Independence of the output bus by Jack and the internal mono audio bus of the MSX. Mono conversion for internal bus and independent hardware amplification for maximum compatibility with the MSX standard and minimal impact on the external Jack.
- Expansion connector for future hardware extensions.
- Recurrent and monitored anti-copying encryption system. The synthesized hardware is open to the use and experimentation for the creation of the desired software and the experimental configurations desired, but the copy of FLASHJACKS is protected by redundant cryptographic systems. (As I mentioned in the introduction, I'm not going to do a commercial exploitation of it, but I'm going to reserve the protection secret for now).



## User's manual Configuration

### Introduction.

Let's explain the inside philosophy of FLASHJACKS.

This cartridge is a whole and a nothing at a time. I explain. All part of an initial configuration that is where we select the hardware and from there and instantaneously the hardware is visible and present for our MSX until it is turned off. That's why it's a whole, and it's a nothing because if we do not configure anything, that option is also contemplated and nothing is synthesized so the MSX will not find anything in that SLOT.

Well, everything is very simple. We can leave the standard configuration without touching eternally and there will be no problems but we can also touch everything we want as if we had real hardware. To advance curiosities, we can load 16MB of RAM in the four available subslots or even load two NEMESIS in two independent FLASHROMS to play the extra screens without having to patch anything and as if we really had it (In case it was already compatible who have the habit of blocking where to put each game for the extra mode).

The configuration is loaded in the root directory of the SD using the text file CONFIG.CFG. The Cartridge on power up the first thing it does is block the MSX and read this file to know what has to be synthesized. As we have said we are independent of the Z80 and with an abysmal processing speed compared to the equipment to which we are connected. You will see that the whole process of synthesizing, initializing the SD card and loading ROMS is usually less than a couple of seconds.

Once the MSX is started, our system becomes transparent and does exactly what we said in the configuration until its next shutdown. With the reset it revives again to execute certain special operations that it could not otherwise do, such as writing ROMs to files or saving the SRAM. Yes, you read that right. There is the possibility of obtaining copies of the ROMs loaded, for example, to save a load that we have made from our MSX within their operating system.

And that's all. It seems complicated but you will see that it is very simple, both for the person who does not want to touch anything and who is curious to try out the most unlikely configurations. Please note that the MSX follow a standard internally but none is equal to another. That is why with these configurations you can skip failures that you did not know about their existence and that in other MSXs you are not able to reproduce them.

Let's go then to explain its use.



## *The proprietary operating system.*

FLASHJACKS has, as we have already said, a proprietary operating system. For us externally it communicates through some very simple instructions that it reads from the file `CONFIG.CFG` in the root directory.

This OS internally has the FAT system 12,16 and 32 integrated, reading the root directory, file management both readings and writes and a driver that manages the SD card natively both the SDSC and SDHC versions up to an unlimited capacity provided that comply with the SDHC high capacity standard.

That OS has a very high processing speed because no governs a CPU if not one or more pointers and exclusive dedicated to this task. In fact, I have managed 1-bit serial data transfers to the SD card higher than 4MBytes per second. Obviously as a whole, communicating with all the modules, we will not get those transfer rates but it will be usual to load and record ROMS at rates of 500kBytes per second, depending on the fragmentation of our card. Comment that security has been imposed at speed and that to give you an idea, each read byte is read 8 times and a statistical method is performed to avoid read errors due to bad contacts or similar (in fact, the statistical level is made to a state even lower than the interpretation byte).

Let's continue with the configuration and its commands.

We have a text file type CONFIG.CFG where the system will read our proposal. Comment that this system will try to read it ignoring additional spaces and other minor errors so in the worst case we ignore the command with the error and in the best case interpret it properly but never leave the system blocked.

This file is editable in the same style as an AUTOEXEC.BAT, so we can use the editor to which we are normally accustomed both in MSX and in a PC.

The file has a maximum capacity of 8192 bytes so we can not exceed that capacity in text. Neither is necessary so much.

The system internally interprets only capital letters but nothing happens that we write in lowercase errors. He post-processes it to uppercase internally.

Let's go to the commands.

There is always a command that precedes an equal symbol "=" and a series of configurations, ending with an enter that tells it is an end of operation.

As we have said spaces between commands, instructions, etc ... they try to ignore.

Add that, in the case of having edited from the MSX and having loaded the configuration, it may be the case of not having a PC on hand to restore it.

There is a way to get us started in a basic mode with access to the unit.

Pressing the HOME key for more than 5 seconds, in the BASIC or MSXDOS, until turning off the flashing red light, when resetting we obtain a safe charging mode. This will allow us to edit the CONFIG.CFG and fix it. When resetting again, the CONFIG.CFG will be loaded again.



In this secure load, we will load RESCUESU.ROM for a Sunrise mode or RESCUESI.ROM for a SINFOX mode. Everything depends on the ROM that you find will be changed to one mode or another.

If all this fails due to the complete deletion of the card, remember that you will always have two ZIPS with everything necessary to restore one system or another. Here a PC is already required to copy the files. In the rescue mode we can from BASIC perform a \_FDISK for SUNRISE or use the SINFOX applications to perform the corresponding formatting of the SD card.

Now let's learn two instructions that are useful for loading and saving configurations:

`LOADCONFIGFILE = FILE.CFG`

As we have put it, our system would interpret that we want to load a configuration file. This is to not use the CONFIG.CFG file as a configuration, but as a pointer or indicator of where our configuration is located. Let's imagine that we have eight different configurations and we want to keep them all. To not be renamed files all the time, we can call them as we please, to then enter the CONFIG.CFG and call our file directly. By the way, the file name format must comply with the MSDOS 8.3 standard, that is, 8 characters for the name, one point and three characters for the extension. For our configurations, they do not need to be .CFG extensions, but recommended.

The rest of the commands will be ignored after this one since it makes sequential execution of commands so when executing this, our next command to execute will be in the new read file.

`SAVECONFIGFILE = FILE.CFG`

This does not have much use at the moment. It saves the current file in a new or existing file with the present configuration. Here the only thing we will notice is that the destination file will have some corrected errors, such as lowercase characters that will appear in uppercase.

As you can see, it's easy. Our configuration will be written in a sequence of instructions.

The structure is simple. Remember the return that indicates the end of the instruction. Any omission will take a default value and if you can not execute it, it simply will not execute it.

One last thing. At the moment, all file management can only be done from the root directory. For now does not manage directories but as we will see is not very necessary because once the system starts, passes the control to the synthesized module and they already load the MSXDOS where it allows management of subdirectories and others.

From here we will explain module to module and its corresponding set of instructions.



## *The Slot expander.*

The Slot Expander behaves like a real one. We can deactivate it if we do not want it and plug one of our devices into the main slot. And we can activate it and insert everything we want in its subslots.

Remember that no longer hardware and if the software does not accept subslots, this will not work correctly. Many times you can fool the software by placing the hardware that does not work in the subslot zero and other devices in the other slots.

The expander follows the following type of configuration scheme in the CONFIG.CFG file:

```
RANURA_MASTERSLOT = Ranura_Vacia  
RANURA_SUBSLOTO = Ranura_Vacia  
RANURA_SUBSLOT1 = Ranura_Vacia  
RANURA_SUBSLOT2 = Ranura_Vacia  
RANURA_SUBSLOT3 = Ranura_Vacia
```

You can do it however you want but I declare it all at once, at the beginning of the configuration, as you see it. It is advisable to leave the structure of the initial CONFIG.CFG.

By default, the system has all its states in an empty slot.

If in RANURA\_MASTERSLOT we load something, this will cancel the Slot expander since it understands that we want that unit loaded in the main slot, deactivating even the slot expander itself and ignoring the rest of the configuration.

To use the subslots, therefore, we must leave the Masterslot empty slot and place our devices in the rest of the subslots.

And little more of the expander. Later we will explain each of the devices that we can plug into these slots.

Of course it supports four subslots as you see in the default configuration. It is not possible anymore because the MSX BIOS is programmed for this and no more, since by hardware there would have been no problem adding more subslots.

Comment that we can leave the initial structure of `CONFIG.CFG` and ignore commands by putting `"//"` double bar. This serves to ignore what comes next. It also serves to add comments, but remember the capacity cap of this file.



## *The FLASHROM unit.*

As we have detailed in the specifications, the FLASHROM unit behaves like an EPROM where inside we will load and save the ROMs to be executed with the mappers that we want.

The load of ROMs can be done at boot time, through instructions in the configuration or with a loading software from the MSXDOS OS (FROM.COM).

For loading from the configuration below we will explain how to load and save RAM blocks to files.

This unit synthesizes several EPROMS brands. It depends on the software that we use, by means of identification of access, the system will throw an ID of the EPROM or another one.

This is automatic and transparent for us so we should not worry about it.

To make a call to this unit from the Slot Expander it would be:

```
RANURA_SUBSLOTO = Cartucho_FlashROM
```

Idem for the rest of subslots or main slot.

Remember that we can load up to a total of four units, one for each subslot.

It will automatically give us a total of 4MBytes EPROM available for the selected subslot.

In this module we must also tell you the type of MAPPER to assign to that EPROM.

The instruction would be:

```
MAPPER_SELECT0 = KONAMI5
```

Where the MAPPER\_SELECT number can be 0,1,2 and 3. The 0 is the first subslot or main slot depends on what we have configured in the expander. If we do not tell him which mapper, by default he takes KONAMI5.

After the "=" we will place the type of mapper that we want. These are the available ones:

```
Auto, KONAMI4, KONAMI5, ASCII8K, ASCII16K, SUNRISE, SINFOX,  
ROM16K, ROM32K, ROM64K, RTYPE, ZEMINA6480, ZEMINA126, FMPAC,  
CROSSBLAIM, SLODERUN
```

The SUNRISE mode does not need to be indicated as we will see a posteriori. It is selected automatically if we select that module as well as the SINFOX.

The Auto mode will detect the type of mapper to load according to the loaded file, so it is essential to load the file first and then precede it with this instruction so that it can evaluate what we have loaded.



*Except for the above, the instructions can be given with the order we want although I advise grouping by type. All the slots together, then all the mappers together, etc ...*

*There are no more commands for this device. Now we are going to learn how to load and save memory blocks.*

## *Load and save memory blocks.*

To complete the previous device we can request a loading or saving of a data block.

To get an idea, FLASHJACKS has reserved 4 blocks of 4Mbytes of memory each. These are called:

BLOQUE\_RAM0,BLOQUE\_RAM1,BLOQUE\_RAM2 y BLOQUE\_RAM3.

Each block belongs to the corresponding subslot.

It also has a load or branded SRAM unit for the FMPAC (BLOCK\_SRAM0,1,2 and 3), these fixed at 8kbytes.

To make a load of a ROM we should place a type instruction:

LOADFILE=SUNRISE.ROM T512KB BLOQUE\_RAM0

This would load the SUNRISE.ROM file with a size of 512KB in RAM block No. 0.

If we had configured in that slot a Flashrom Cartridge what we are doing is to load the file in the Flashrom for its execution at the start of the boot.

The available sizes are:

T64KB,T128KB,T256KB,T512KB,T1MB,T2MB,T4MB.



By default, the pre-selected size is 512Kbytes.

For the load I advise you to place the 4Mbytes since the system will stop loading as soon as the loading file is finalized. That is, an 8Kbytes ROM will read 8K bytes and not 4Mbytes since there is nothing more to read in the file.

With this instruction every time you start the MSX either by the power button or the Reset button, it will load the file in the selected block.

The order of the instruction must be that indicated in the example and not another, separated by spaces.

To save a block we will place a type instruction:

```
SAVEFILE=SUNRISE.ROM T512KB BLOQUE_RAMO
```

This instruction has multiple utilities.

What our MSX is going to do is that at the first start it will interpret it as a LOADFILE. We will load the selected file in our memory block.

If this file does not exist, it will not load anything. (The same would happen in the previous LOADFILE case) .

Once started the MSX and after having done what we wanted on our computer, if we press the Reset button what the cartridge will do is save the selected block with the selected capacity in the name of the selected file. If this file does not exist, create it. All this always in the root directory, as we have commented in other sections.

This will happen many times you press the Reset button.

Beware of pressing the Reset several times before the end of the process of saving since we will interrupt this process and may leave corrupt data on our card.

Here, if the selected capacity matters because it will record that capacity, nothing more or less.

And what is the use of this? Well many.

Imagine that we have an empty FLASHROM and that by software, from the MSXDOS, we load a ROM. Resetting the MSX will do two things to us. First we will save the RAM block with the modifications made in the FLASHROM and the second one will load the ROM in the boot.

When we turn off the MSX, the volatile RAM will disappear but when turning on the computer, having this instruction will dump the RAM saved from the file to the RAM, recovering the data again and making the illusion of having a real non-volatile EPROM with the data intact at the start of our MSX.

Remember that FLASHJACKS uses a volatile RAM and that when you turn off the computer everything is lost. With this instruction and with the rapid charge that it is capable of performing, the illusion that there seems to be an EPROM will seem real. If we want to erase this EPROM in the next boot we just have to delete this configuration line or the saved file.

This instruction can also serve to make a dump of the RAM\_MAPPER of our MSX to a physical file as we will see later.



*Comment also that for the SRAM behaves in the same way so we can rename the file to save games from different games without crushing the scarce 8 blocks of bran that allows those 8Kbytes.*

*We have an instruction in which it avoids the double load of ROM. This serves to avoid being loading the same ROM again and again after a Reset. Remember that the RAM keeps the data until we turn off the MSX.*

*This is the instruction:*

**RELOADROM = OFF**

### *The RAM MAPPER unit.*

This device will be responsible for that, to provide us with extra RAM by following the MSX standards so that for the user it will be transparent without the need to install any additional driver.

For its invocation it would be an instruction of the type:

`RANURA_SUBSLOTO = Cartucho_RAM_Mapper`

By default we would load the maximum of the block in the subslot, which would be 4Mbytes of RAM.

If we want something else for whatever reason, we would also load the following:

`Capacidad_4MBRAMO = T4MB`

Where the number of 4MBRAM indicates the subslot and T is the size of the RAM. There are the following selectable sizes:

`T64KB,T128KB,T256KB,T512KB,T1MB,T2MB,T4MB`

As you can see everything follows the same configuration philosophy. We can indicate everything according to the order that we want but I advise grouping them by types. All the capacities together, all the slots together, etc ... In this way we will have everything more computer and easy to identify and modify. We have a total of four devices like this so once we have set one in each subslot, we could have a total of 16 Mbytes of RAM for our MSX. Something not useful because what it consumes are subslots and we would be left without them in order to add devices.



### *The MEGARAM unit.*

This device synthesizes a MEGARAM for all its effects.

For its invocation it would be an instruction of the type:

$RANURA\_SUBSLOTO = Cartucho\_MEGARAM$

By default we would load the maximum allowed by a MEGARAM device, which would be 2Mbytes of RAM.

If we want something else for whatever reason, we would also load the following:

$Capacidad\_4MBRAMO = T2MB$

Where the number of 4MBRAM indicates the subslot and T is the size of the RAM.

There are the following selectable sizes:

$T64KB, T128KB, T256KB, T512KB, T1MB, T2MB$

If we select 4MB, 2Mbytes will appear because the MEGARAM system allows a maximum of that.

We have a total of four devices like this so once we have set one in each subslot, we could have a total of 4 x 2Mbytes of MEGARAM for our MSX. Here, the software needed to access the MEGARAM must be compatible with the subslots and with multiple devices.

Now we are going to explain the utility of SAVEFILE in RAM and MEGARAM devices. If we make a:

`SAVEFILE=BACKUP.RAM T4MB BLOQUE_RAMO`

We are going to do an initial load of the BACKUP.RAM file if it existed in the subslot 0 or main slot. Pressing Reset will make us dump all the RAM\_Mapper or MEGARAM that we have from that subslot to the BACKUP.RAM file

This is useful, in the case of a RAM\_Mapper to capture the RAM for later analysis and to know what a particular program has done, a game, etc ... It basically serves to debug a process executed in our MSX.

In the case of MEGARAM it would serve to keep the information stored in it intact, always having that information available and saved on our card.

As we explained in the EPROMS, adjust the size of the load to our needs because he will do what we say, even if it is illogical. We can load 64Kbytes in a 4Mbytes RAM but we will be aware that we are losing data.

If we only wanted to have the file upload function but not to save, we would do a LOADFILE.



Finally comment that the Write Protect tab of the SD card is functional at all times so if we have it activated, any write function would be ignored. In MSXDOS the system would already indicate that we can not record but in the FLASHJACKS system there is no indication.

Comment that we have the following instruction to ignore the writing tab:

`IGNORA_WP = ON`

## *The SINFOX compatible unit*

This device synthesizes the standard of the old SINFOX units.

Comment that we only have one SDCARD device so we can only use one type of unit synthesized simultaneously despite the fact that FLASHJACKS has other types. It will always give preference to the SUNRISE unit since once activated this module, it eliminates the access of another module.

This unit would be invoked as follows:

`RANURA_SUBSLOTO = CARTUCHO_SINFOX`

Automatically the mapper of the EPROM of that subslot is configured to accept the ROM of the SINFOX.

Only we would have to do a LOADFILE or SAVEFILE of the ROM in that selected subslot.

Of course we could do a load of the ROM from the MSXDOS. All options are allowed.

I will not explain what we can do with the soft for this unit since I am not the owner of its software but I do know that it has its limitations and that we must consider them, especially that we have to format the card with its format. Caution because we will lose the CONFIG.CFG. Once formatted, I suggest to turn over the CONFIG.CFG card to continue with the copy of files, disks, etc ... or better yet, to dump the ZIP file for that purpose. As we said FLASHJACKS is compatible in the boot with FAT12 so he will not be disgusted with the new format (in fact it is compatible with FAT32 even though no MSX unit supports it).



In the case of SINFOX, remember that we will be limited to that 512MB card in SDSC and is incompatible with SDHC.

Commented that these types of units, such as SUNRISE I'm not the owner and this is simply a compatibility with the standard they created. They may use their software at the discretion of the owners but decline any future incompatibility or prohibition of use. The compatibility you have, but the use you make is your responsibility.

The same for the rest of the software and operating systems. At the end of the manual I will expand a little more at this point.

## The SUNRISE unit compatible

This device synthesizes the standard of the SUNRISE IDE units in LBA mode up to the maximum allowed by the driver of NEXTOR SUNRISE IDE See 2.1 Alpha2, which would be 4GB and compatible with SDSC and SDHC.

Inclined as in the previous section, that I am not the owner of the software and in this case everything is limited to the impositions of their owners. This unit is internally a unit with transparent double conversion for us and with access to the flight. This means that we have a bidirectional communication of the following modules:

SUNRISE ↔ IDE ↔ SDCARD ↔ NEXTOR

Each module is independent and has its own limitations of the original design.

All this is done on the fly and in the most efficient way possible but it has its drawbacks since this system has been designed to use a COMPACT FLASH and unfortunately there is no feedback that everything went well. They give a constant access time and you can not get out of those parameters which thing is sometimes impossible. It is designed for a Flash memory, with a parallel bus and constant times, and we are accessing a serial device, with another protocol and with variable wait times.

That is why a driver for FlashJacks has been created where all these issues are solved.

The driver is the FLASHIDE.ROM.

Basically it is the NEXTOR driver for Sunrise but adapted to our unit.



A part of the above takes some vitamins like the following:

`FREQVDP = 60HZ`

If we put this in the `CONFIG.CFG`, when Nextor executes, it will immediately go to forced 60HZ. Idem for 50HZ and for OFF.

Very useful for those games that were designed for 60HZ.

And now we discuss the configuration of the SUNRISE or FLASHIDE cartridge:  
Well, let's go to his invocation:

`RANURA_SUBSLOTO = CARTUCHO_SUNRISE`

Same as in the `SINFOX`. The mapper of the EPROM enters SUNRISE mode and allows us to load the ROM in the same subplot selected by the parameters that we already know. With `LOADFILE` we would load the corresponding NEXTOR ROM for SUNRISE IDE units.

Say that the SUNRISE compatibility is not total if the specific `FLASHIDE.ROM` driver is not loaded. The write access of the ROM has not been clear to me and although some original software can work to write in the EPROM, I can not guarantee it. Anyway, the purpose of this unit is access to the data of the SD card, using the mentioned driver that is covered.

One last tip is that if you do not load an album it can be for several things. Test to do a shutdown of the MSX instead of a reset. It seems silly but sometimes there are resident programs that do not allow a load of the game or program. Pressing CTRL at startup is also useful.

Certainly. The internal driver of the SD card allows hot insertion and extraction. This means that with the MSX turned on we can extract the card, modify, add files externally, etc ... and then insert it again. The reader will restart it without problems but you have to be careful with the MSXDOS because it does not know that we have done this. Do not change to different cards since the MSXDOS can give serious failures.



## *The sound system of FLASHJACKS*

Before going into the MSX sound modules, I will comment a bit on the audio philosophy of the FLASHJACKS system.

We have always seen that the sound in our MSX has not been very careful and the noise issue, compatibility with sound levels in certain MSX models, etc ... have been very varied.

In this section I wanted to overcome the original systems, redesigning the whole concept to a higher level. To begin with, the mixing of audio systems is done internally in the cartridge, so that all systems will have an adequate and mixable sound level between them. In addition, the internal audio of the MSX is separated with the external one, avoiding noise, inadequate levels, etc ...

In the second instance, I added an audio processor that delights any listener. I have seen many projects, type ONECHIP, ZEMMIX NEO or similar but the digital - analog conversion does not seem adequate, in fact neither the treatment of the signal, nor other things where I will not enter into controversy with them since obviously you have to recognize the prowess of such projects.

Our processor would be able to deliver stereo sound at 96Khz, with samples of 24bits per channel, with a SNR noise ratio of up to 100dB. A part includes as standard several systems of postprocessing, filters, mixing channels, etc .. and everything in stereo and bidirectional (Yes, you can also capture audio). Come on, crazy.

It also has the ability to handle a headphone line output of 16 or 32 Ohms with a power of 30 or 50mW respectively. I recommend some SONY MDR ZX300. They are cheap and they sound good.

To connect our audio system to the MSX we can use a 3.5mm jack and use the audio output as headphones or line output.

At the same time, inside the cartridge has a dedicated amplifier to the audio input of the Slot so it guarantees maximum compatibility with the internal audio of the MSX, if we wanted to extract the audio from there. Say that the stereo is obviously lost since the MSX hardware only supports Mono but despite this, the conversion to Mono is adequate and balanced.

The internal FLASHJACKS system uses more moderate specifications. The audio data bus treats signals from 48Khz to 16bits per channel in PCM. That is, PSG, SCC and FM extract their audio in this format for later mixing and processing. There is no 1 bit SCC or 4 bit PSG. Everything is synthesized in 16 bits. Comment that the waveforms and bit density of the instruments are the originals since we did not want to alter the original sound, but the mix of instruments, wavy, etc ... if that is 16 bits.

But for me what gives a touch superior to the original system is the Stereo. No PseudoStereo or adapted bi-channel. It is real Stereo and the best thing of all is that it extracts it from the original audio. We can listen to all our usual games on Stereo, being designed for a Mono system and the truth, it does not do anything wrong.

The synthesized model has been thought of in the synthesis of an acoustic space of 180 degrees. That is, from our left ear to our right ear opens a range of 180 degrees and in that location the instruments are ordered according to their origin for what they were conceived.



*That is, the percussion systems, batteries, etc ... are located at 90° in the central channel while channels for accompanying instruments such as trumpets, violins, etc ... are located at 45° and 135° giving a unique surround feeling. It makes a whole truly feel the instruments fanned before us although were made for an audition in mono.*

*I hope that this system created makes a precedent for future composers and in this way to use a standard and conceive scores that better exploit these audio channels.*

*In the following modules we will detail which channels are diverted and which side to take as future references.*

## *The PSG module*

This module basically when activated enters slave mode. That is, it captures the MSX bus data and reproduces the PSG system in FLASHJACKS audio. You do not need to configure slot or subslot since you do not need it. It is invoked in the following way:

`SOUNDPSG = ON`

By default it is in the off state. It is not necessary to tell him.

If we have the Stereo mode activated, the instruments are distributed in the environment 180° as follows:

- Channel B, only tone on the left side at 45°.
- Channel C, only tone on the right side at 135°
- Other channels and noise on the central side at 90°.

I have extracted this configuration from somewhere where I have seen this distinction for PSG stereos. In the PSG and the rest of the system I tried that the separation of channels is not arbitrary and that if someone already separated the stereo, respect as much as possible what exists.

The PSG of serie emits to 112Khz but has been cut to 20Khz since the high band is very annoying.

In addition, if you want an extra filter, you can add a global filter, as we will see later like the rest of the parameters.



One last thing. The PSG is already issued by our MSX, if we use ours and listen to it through the MSX, this will be reinforced so you may notice an enhancement of the PSG to the rest of the audio systems. If you notice this and you find it annoying, when you play the audio on the MSX you may want to turn off the PSG and leave the rest of the modules activated so that the mix within the MSX is more linear. Mention that MSX PSG does not go through our system and you will not be able to hear it through headphones if you deactivate this module.

## *The SCC module*

This module activates the SCC sound. This system does require that we tell you where you will hear the instructions. Here your call:

`SOUNDSCC = MASTERSLOT`

`DUALSCC = OFF`

Where in MASTERSLOT we can put the following: (OFF, MASTERSLOT, SUBSLOTO, SUBSLOT1, SUBSLOT2 or SUBSLOT3).

It does not occupy a slot so it allows duplication of resources but it does come into conflict for the software to detect it correctly. The logical thing would be to put it in a subslot occupied by a FlashROM cartridge. This is your natural configuration. If we leave it loose, the game may not detect the SCC system because there is no FLASH unit.

It can also be the case that the software only looks in the main slot and we need to have things in the subslots. We can put the SCC in MASTERSLOT with the active Slot expander but it is probable that we have acoustic interference with instructions that are not directed to the SCC since they go to modules located in the subslots. This is a compromise solution that usually works but with the exception that sometimes you hear noises which are the communication to the rest of the components. If we do this, place a FLASHROM in the SUBSLOTO so that the program that does not accept the slot expander has it as easy as possible to locate.

The dualSCC is off by default. This serves to place two SCCs in two different subslots and that a program can access a double SCC simultaneously.



If we have the Stereo mode activated, the instruments are distributed in the environment  $180^\circ$  as follows:

- Channel A, C and E on the left side at  $45^\circ$ .
- Channel B and D, on the right side at  $135^\circ$

It's the most natural way I've seen and it adapts very well to existing music.

## *The FM module*

This module activates the FM audio. It does not occupy a slot but it needs to be loaded by the FMPAC.ROM driver. This is already up to each one. We can create a FLASHROM in a subslot and do a load at the start of the FMPAC.ROM or we can load it afterwards in any other way. Some games do not need ROM. This is already the decision of each one. It is best to load a FlashROM with FMPAC mapper and load Panasoft's FMPAC. With this we will have a faithful reproduction of that cartridge, including its CALL FMPAC or its internal SRAM, as we have spoken before.

To call the module for activation:

`SOUNDFM = ON`

By default it will be deactivated.

If we have the Stereo mode activated, the instruments are distributed in the environment 180° as follows:

- Channel 1, 3 and 5 on the left side at 45°.
- Channel 0, 2 and 4, on the right side at 135°
- Rest of channels and percussion through the central channel at 90°

By alternating odd and even channels what we do is that normally the scores are written on consecutive channels and if any instrument is to be reinforced, they usually duplicate it in the channel that follows it. With this system we obtain that both instruments are separated producing a unique spatial sensation.



*Personally, it is the configuration that best suits the largest available repertoire. Obviously there will be songs that this configuration will not be the most appropriate.*

*For new scores, taking into account this scheme, you can compose a fancy stereo acoustic deviation.*

## *The audio bus*

Here we will explain the other parameters that affect the audio generic form.

Let's start with the Stereo:

*SONIDO = STEREO*

You have two options, *STEREO* or *MONO*. In mono we will listen to it in its original composition. By default it will be in *STEREO*.

Now we will tell you the headset volume or line output:

*VOLUMEN = 5*

The available options are 0,1,2,3,4 and 5. Zero is without sound and 5 maximum amplitude. Intermediate values give attenuations from highest to lowest. This does not affect the internal bus volume of the MSX. The internal amplification of the MSX is calibrated to a specific level for greater compatibility with the standard. If we choose a zero, it will continue listening through the internal MSX. By default the value is 5.

The hardware filter:

*AUDIOFILTRO = ON*

By default it will be off. If we select it, we will attenuate the "glitches" and other noises but we will also lose frequency response, listening to the most serious tones.



If we want a configuration that respects the original system as much as possible, the SCC, the FM, the sound in MONO and the activated AUDIOFILTER should be activated.

If you want depth and purity since you already know, in STEREO and without filter.

Next, the sound mixer:

Internal audio bus	Audio output jack
PSGVOLIN = 0	PSGVOLOUT = 3
SCCVOLIN = 2	SCCVOLOUT = 3
FMVOLIN = 2	FMVOLOUT = 3
EXTVOLIN = 2	EXTVOLOUT = 2

The options are from 0 to 3, with 0 being silence or low level and 3 maximum.

Each audio chip carries its own gain.

The system detects if we connect to the output jack and swap from the "IN" to the "OUT" setting. In this way we can listen with a configuration without PSG and suitable levels for a reproduction from the audio bus of our own MSX and its internal PSG, with the particularity that if we connect a jack to the jack, the system automatically switches to the "OUT" configuration ", adjusting to the new hearing system.

The EXTVOL is designed for us to provide the external audio system we want for mixing and leveling completely digital, without loss of quality and incorporated into the rest of the audio chips.

This is intended to add a Moonsound, Moonblaster or similar.

You can even plug in your phone and put your favorite MP3s. In some games designed for SCC, if you remove the volume from it, you can play with your favorite music and the system mixes the shots and effects of the PSG.

## *Other aspects general configuration*

And finally a command to turn off the RAM transfer status of the RED LED:

```
LEDROJO = ON;
```

It only supports two states, on (ON) and off (OFF). By default it will be on.

When it is on you will see a flicker of the LED while it works. This tells us two things.

That the system is working correctly and the availability of RAM for other uses.

Apart from all the management of the modules, FLASHJACKS RAM has the privilege of having multiple access and simultaneous shared access. This means that apart from attending the needs of the MSX we are also accessing simultaneously for other tasks.

Internally we have a small read and write loop in RAM. He is reading and writing incremental positions and compares the result.

A single wrong byte and the led would be frozen. Each full flicker means that you have done a complete check of 256Kbytes of RAM. The speed of the blinking indicates the availability of access to this RAM since the MSX always has priority. If we see it go slower it means that the MSX is occupying it more and vice versa. You can observe the amount of blinks that you do to imagine the transfer rate that our system has since these accesses are only a part, the MSX does a lot of recurring access.



Now we will indicate an example configuration for our CONFIG.CFG file:

```
RANURA_MASTERSLOT = Ranura_Vacia
RANURA_SUBSLOTO = CARTUCHO_SUNRISE
RANURA_SUBSLOT1 = Cartucho_RAM_Mapper
RANURA_SUBSLOT2 = Cartucho_FlashROM
RANURA_SUBSLOT3 = Cartucho_FlashROM
RELOADROM = OFF
LOADFILE = FLASHIDE.ROM T512KB BLOQUE_RAM0
LOADFILE = FMPAC.ROM T512KB BLOQUE_RAM3
MAPPER_SELECT2 = KONAMI5
MAPPER_SELECT3 = FMPAC
IGNORA_WP = ON
SAVEFILE = SAVESRAM.RAM BLOQUE_SRAM3
Capacidad_4MBRAM1 = T4MB
FREQVDP = 60HZ
SOUNDSCC = SUBSLOT2
SOUNDPSG = ON
SOUNDFM = ON
SONIDO = STEREO
VOLUMEN = 5
AUDIOFILTRO = OFF
LEDROJO = ON
PSGVOLIN = 0
SCCVOLIN = 2
FMVOLIN = 2
EXTVOLIN = 2
PSQVOLOUT = 3
SCCVOLOUT = 3
FMVOLOUT = 3
EXTVOLOUT = 2
```

This would be an example where we have a very complete configuration.

We would have a SUNRISE FLASHIDE cartridge, a 4Mb RAM memory, two FLASHROMs cartridges where the SCC would be connected to the first available FlashROM, which is the most compatible. In the second FLASHROM we would have it to load the FMPAC.ROM and in the SUNRISE also its corresponding ROM. The first FLASHROM would be free for uses. All in Mapper KONAMI5, which is the mapper defect in most applications.

We would also have all the audio devices activated, in stereo and without filters. And finally the system of checking the activated RAM also.



## **Loading ROMS from MSXDOS. "FROM.COM"**

Next we will describe an alternative to loading through the FLASHJACKS configuration system.

The idea of the system is based on two modes. The manual mode where you configure at will and the automatic mode that with a standard configuration can access most of the catalog of programs and games that were made for MSX.

With FROM.COM is intended to address this automatic press and ready. The standard configuration is optimized to have a bit of everything. In sublot 0 we have the disk drive, in the subslot 1 RAM, in the 2 a free FlashROM and in the 3 a complete FMPAC which we can throw it so that a second FlashROM coexists.

Having said that we comment your options:

FROM NOMBRE1.ROM NOMBRE2.ROM

In this mode, if we do not place options, it will accept the loading of two simultaneous ROMS. These will place them in the first two FlashROMs you find. In our case they are located in the sublot2 and 3. The disk drive is forbidden to become FlashROM.

If we just put a ROM (the usual), it will load us in sublot 2 leaving us the FMPAC in sublot 3.

When there are no parameters, it will load the mapper in AUTO mode. The AUTO mode chooses the mapper that best suits that ROM for you. It even autoconfigures to primary Slot if it observes in the code of the ROM some incompatibility with the subslot.

We have not yet talked about the AUTO system of the mappers.

When this mode is chosen, internally the ROMs are analyzed with a double method. On the one hand there is a heuristic type method, where you look for marks in the jump type ROM to find out how the most accurate mapper behaves and adapt to it. This system makes a multiple crossing of these brands and the winner is the one selected by the system. In addition, they cross paths with a small database where they fill in those gaps where they have doubts.

It really behaves similar to what an antivirus does.

The process is performed in the Flashjacks hardware at vertigo speed while loading the ROM without it being affected in speed.

Besides the previous search, the system tries to look for incompatibility marks before subslots and in the case of finding them, pulls the configuration of the Slot expander, leaving as primary the first load ROM.

Of course the system is not infallible and that is why FROM.COM has a manual system. Let's go for this second loading mode:

FROM /Sxx /My NOMBRE.ROM

Here it only allows us to load a ROM with parameters.

In /Sxx we can force Slot and Subslot, as long as there is a FlashROM of FlashJacks in that slot.

If we place only one digit instead of two, we are telling the system that we want to force the ROM into the main slot so it will cancel the Slot expander.



In /My allows us to form the mapper where "and" can be:

0: AUTO	1:KONAMI5	2:ASCII8K
3: KONAMI4	4:ASCII16K	5:SUNRISE
6: SINFOX	7:ROM16K	8:ROM32K
9: ROM64K	A:RTYPE	B:ZEMINA6480
C: ZEMINA126	D:FMPAC	E:CROSSBLAIM
F: SLODERUN		

Each name intuitis the type of mapper that will load.

All these mappers, are expanded to the maximum that can address that format, surpassing the address and capacity for which they were thought. The mapper that has the capacity to address the 4Mb (or 32Mbits) of Flashjacks is the ASCII16K type.

Once we press ENTER, the magic happens. Although apparently it gives you a result and the MSX continues to work as if nothing, the device already has those instructions in the breech for the next reset.

FROM.COM does not work like other EPROM loaders where it dumps through the MSX CPU.

FROM.COM is a program that sends instructions to FLASHJACKS, where after a reset, the beast of the internal hardware will process at a speed infinitely superior to what the Z80 could do.

This mode of operation will continue like this until we do not perform a Power OFF.

Indicate that other EPROM loaders can be used since our FlashROM will behave as if it were an original EPROM.

The system FlashJacks internally has an adaptive system that makes give the appropriate reference of EPROM depending on what the software asks.

This process is slow since it is the Z80 that is responsible for transferring and even patching our ROM to work in the EPROM (in our case a RAM simulates this EPROM).

By the way, our system is pure hardware and behaves as such. We will never patch a ROM to load emulating a mapper, which is what traditional systems do. FlashJacks contributes that mapper as if it were real (well, in fact it is).

And little more than commenting from FROM.COM. Remember to do Reset to execute the requested and Power to start over.

Remember that in the self-saving settings of ROMs or SRAM, that saving is done at the next reset. If we want to save an SRAM file, we must do a reset after having played and saved.



## **A bit of MultiMente**

Although this program is not part of FlashJacks, I want to comment because it is widely used, so it has been added to the files on the SD card.

Comment that when we load the SD, it will come with a mix of various files configured for direct use from MSXDOS, or through MultiMente.

The boot is automatic, unless we are in an MSX1, where in the final load will launch an invalid command. In this case, you have no choice but to use the keyboard and type MSXDOS commands as usual.

Comment that even if the MultiMente does not work on an MSX1, we can access all the FlashJacks features without major problem.

Well. Within Multimente, everything is easy. Press ENTER in the ROM that you want and it will load it.

Comment that the DSK will be loaded through Nextor through the EMUFILE. For this case I dithe above, ENTER and enjoy.

If we select several files and ENTER, it will load multiple files.

In the case of ROMs, remember that there are a maximum of two files.

And a little more. Of the rest surely you will already be accustomed.

## Another bit of SDs and files

I comment a little what you will find in the SD that is included.

The card is already formatted in FAT16 from the BASIC of an MSX using the CALL FDISK command.

The instructions for this can be found in the leeme.txt of the card itself.

Inside comes a structure configured for automatic operation.

In the root directory appear the ROMs and configurations for loading FLASHJACKS.

There are also directories to place the ROMS, DSK, etc.

In addition, there is a Backup directory where there are two ZIPs.

One of them is to restore the contents of the card for a SUNRISE operation.

And the other ZIP is to restore in a SINFOX operating mode.

In the leeme.txt of the SD card, it explains in detail how to carry out such formatting to these systems.

Comment that SINFOX is a closed system and that it has not been altered in any way so it will go exactly like an original one. This means that certain load parameters such as the 60Hz change will not work since there is no FLASHJACKS driver to tell you.

And the NEXTOR SUNRISE IDE system will provide all the provisions of the Hardware since the original driver has been adapted to this device (FLASHIDE).



*One last thing. Remember to leave the RESCUESU.ROM and RESCUESI.ROM. These are used when you press the HOME key (panic key) for more than five seconds, on the next load, self-select a rescue configuration and, among other things, load the mentioned ROMs. This quality is useful for playing an altered CONFIG.CFG and we do not want to use a PC to fix it.*

## Epilogue

And we come to the end. You can observe the flexibility of the cartridge where each one can adapt it to their needs occupying only one SLOT. As a curiosity, you can connect it in a SLOTS expander but then you can only configure the MASTERSLOT slot for obvious reasons.

Thank and apologize at once, to the entire MSX community. Thank you for all the information available on the network and allowed me to carry out this work. And apologize to anyone who may have been offended to see that this device is part of what he created. It has not been my intention to plagiarize or anything like that. I have only taken from the network the standard of operation of electronic devices, which if you observe are not in production. I would obviously have liked to add certain devices that are currently on sale, and that, in addition, certain MSX emulators already have emulates, but out of respect for their owners I have not made them.

All the software, names, etc ... that are mentioned here belong to their respective owners and their compatibility with this device is based on the protocol of hardware operation and not patches of their programs. Your work remains intact and its use and distribution depends on the end user of this device.

I also want to comment on the continuity and support of this device. Of course I would like to give support and continue expanding the possibilities of it but unfortunately I can not guarantee it. This is done entirely in my spare time and leisure, altruistically. I do not know what uses and availability I will have in the future, and as I said, I am not a company, I do not have a trade nor is this a business.



The units that will circulate will surely be very limited due to several restrictions. The production cost is high both economically and in time. There are pieces that are not available all year round due to their huge price fluctuation.

It is a device of an amateur and whoever buys it must know what that entails. If you want security, support, continuity, etc ... better buy other equipment on the market that will give you that stability you are looking for. This is rather aimed at restless people, who like to chat, experiment, etc ... and want a plus in their systems but knowing their limitations.

I will try to add more synthesizations and updates of bugs that are found but I can not guarantee continued support. The updates have to be done in person since I need to connect to the device physically. I will try to be in the meetings of users of MSX but it is something that I can not guarantee either.

As you can see, I have been very strict in the responsibility of it. I put the truth in front of everything and I do not want anyone to have disappointments or wrong ideas. Surely it will surpass your expectations but I prefer to be strict and comment all the butts of this system.

And a little more to say. Enjoy it and take advantage.

Regards.

AQUIJACKS 2018.