fmsx

COLLABORATORS

	TITLE : fmsx	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 14, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

fmsx

1.1 "

fMSX Amiga 2.2

General information

1.2 "

Introduction to fMSX

fMSX stands for Fast MSX, and that's what it is: a reasonably speedy emulation of the
MSX
computer system.

fMSX Amiga is based on the original work of Marat Fayzullin (fms@komkon.org). However, by now every single line of code has changed, and there is absolutely no connection between fMSX UNIX and fMSX Amiga anymore.

fMSX Amiga has come a long way from its commandline roots. From the early days, where you could see each sprite being painstakingly plotted, to the current version with its full-featured GUI and support for every piece of hardware imaginable, some 30,000 lines of code were written, and then rewritten, sometimes many times. Nowadays fMSX Amiga is built from a mix of assembly, C, and C++.

In order to test the various fMSX versions I have selected a short BASIC program as a benchmark. Don't take this too seriously: the timing results obtained from running this problem are very different depending on the settings of fMSX.

```
10 FORT=1TO10000
20 NEXTT
```

Machine	Time
a4000/30	94 seconds (fMSX 0.3, original version)
Sun Sparc 2	86 seconds (fMSX 0.4, UNIX version)
Sun Tatung	43 seconds (fMSX 0.4, UNIX version)
a4000/30	22 seconds (fMSX 0.3, modified version)
a4000/60	22 seconds (fMSX Amiga 2.0, with speed control)
a4000/60	22 seconds (fMSX Amiga 2.1, with speed control)
VG-8235 MSX2	21 seconds
a4000/40	15 seconds (fMSX Amiga 0.2, standard settings)
a4000/40 (warp 4040)	9 seconds (fMSX 0.3, modified)
a4000/60	7 seconds (fMSX Amiga 1.4)
MSX Turbo-R/RAM mode	6 seconds
a4000/60	5 seconds (fMSX Amiga 2.0, without speed control)
a4000/60	5 seconds (fMSX Amiga 2.1, without speed control)

The ideal speed here is the one tested on the VG-8235, which is a real MSX machine. Since version 2.0, fMSX is available in two versions: one that always runs at precisely MSX speed and one that always runs as fast as possible.

The MSX Turbo-R was the last MSX machine that ever came out. It was using an R800 CPU, which is a Z80 variant that executes almost every instruction in a single cycle. Due to this, and some added instructions, it reaches speeds comparable to a Z80 running at 28MHz.

1.3 "

Disclaimer

THIS SOFTWARE IS PROVIDED AS-IS. THE
AUTHOR
WILL NOT BE HELD RESPONSIBLE
FOR ANYTHING IT DOES, WHETHER RIGHT OR WRONG, GOOD OR EVIL, LAWFUL OR
CHAOTIC. RUN AT YOUR OWN RISK!

1.4 "

The MSX system

The MSX originally came out in 1983, with the intention to create a
low-cost low-performance all-compatible computer system (a gap that
is now filled by the PC, ironic when you consider what MSX stands for):

MicroSoft eXtended

which means that it is an enhanced version of earlier Microsoft
products, notably their BASIC and DOS (that's right, MS-DOS!).

The MSX1 standard

Among other things, the MSX standard specifies that machines should have
at least a Z80/3.5Mhz, 16KB RAM (although most machines had 64KB RAM),
16KB video RAM, and the MSX ROM's. Graphics, produced by the v9918,
are functional but not spectacular:

screenmode	resolution	colors
0	40*24 characters	2
1	32*24 characters	16 (with restrictions)
2	256*192 pixels	16 (with restrictions)
3	64*48 pixels	16

In addition, screens 1-3 can have single-colored sprites, either 8*8
or 16*16, either at normal size or enlarged.

Sound is delivered by the AY-3-8912, a three-channel sound chip also
used in the Spectrum 128K and the Atari ST.

One thing that sets MSX machines apart from other comparable systems
is the ability to run cartridge software - most machines can, but for
MSX the system was very popular. It would definitely be possible to
build an MSX console; in fact the Colecovision is so close to MSX that it
can almost be called that.

Another unique feature is the popularity in both Japan and western
Europe. This resulted in a software catalogue filled with the best
of both worlds, a feature that always attracted me to the system.

Finally, it must be mentioned that MSX is not the product of a single
manufacturer. Instead, MSX is a minimum standard decided upon by ASCII
corporation, with individual manufacturers licensing and adding to the
system, much like 3D0 today.

There have been over 40 MSX manufacturers, and MSX machines were
manufactured by Daewoo in Korea after Commodore went broke in 1994!

The MSX2 standard

In 1985 the MSX2 came out, which sported several large improvements. The most remarkable is the graphics, provided by the v9938; it offers all the MSX1 modes, and several others:

screenmode	resolution	colors
0+	80*26.5 characters	4
4	256*192 pixels	16 (with restrictions)
5	256*212 pixels	16
6	512*212 pixels	4
7	512*212 pixels	16
8	256*212 pixels	256

Other improvements to the graphics include an enhanced sprite system, which allowed for more sprites with more colors, a color palette of 512 colors, support for genlocking and digitizing, smooth vertical scroll, and a built-in blitter which can do almost anything the Amiga blitter can, and more.

The rest of the system had been updated as well. MSX2 machines are equipped with at least 64KB RAM (128KB and 256KB are more common configurations), 128KB video RAM, and usually a 3.5" diskdrive.

The MSX system uses exactly the same disk layout as MS-DOS. In fact, CrossDos can be used to read/write MSX disks, even single sided ones (my compliments to the authors for that), while modern PC's have lost that ability!

MSX2+, MSX Turbo-R

These are newer versions of the MSX system. New features include better graphics (the MSX2+ VDP is capable of displaying 19268 colors at once on screen), and a faster processor (the Turbo-R has an R800 CPU, comparable to a Z80/28MHz).

This document used to say they were not on the list for emulation. Today I'm not so sure - if it is remotely possible to run Illusion City on fMSX I will do it.

1.5 "

COPYRIGHT

pay for, neither author refuses gifts, money or postcards. I’d appreciate
it very much if you were to send me email!

1.6 "

Installation

No specific installation procedure is required, just drag the directory
containing fMSX to some place of your liking. Note that the files "fmsx"
and "fmsx.data" absolutely have to be in the same directory!

Note: do not copy the MSXFonts/ directory to FONTS:! The fonts in that
directory are in MSX format, not in Amiga format.

To run cartridge images from the Workbench

Many games come with pre-made icons. These icons require that fMSX:
is assigned to the directory where you installed fMSX.

1.7 "

System requirements

Required are:

- Amiga OS 3.0
- 68020 processor
- 3MB RAM

Recommended:

- AHI (required for some soundmodes)
- 68060 processor
- A Concierto soundcard

If you want to use MSX disks, you will need to have CrossDos or similar
installed. CrossDos has been part of the AmigaOS since v2.1. Shareware
alternatives are available on Aminet.

Since fMSX 2.0, fMSX allocates RAM dynamically (earlier versions used mainly
static allocations). It also uses a lot more RAM. However, fMSX can run on
native CyberGfx screens now, in which case it uses almost no CHIP RAM.

This program was developed using:

- An a4000/060/PPC with 64MB RAM, 2.2GB HD, Picasso IV + Concierto.
- SAS/C 6.57
- GenAm 3.02
- TurboText 2.0

1.8 "

Running the emulator

From the shell

Type fMSX to start the program. It is possible to load a cartridge file or disk by typing fMSX <ROM-NAME>, which will load the specified cartridge. A full pathname may be used when specifying the cartridge.

From workbench

Double click on the fMSX icon to start the emulator without loading any cartridges.

Loading a cartridge is accomplished by double-clicking it. Note that the included game icon requires the directory fMSX: to be assigned to the directory where the emulator resides. You will need the shell to create this assign.

Restrictions

As an MSX1 or MSX2 emulator fMSX Amiga is fairly complete. MSX2+ or Turbo-R features are almost non-existant though.

The following sort of software will run:

- Standard ROMs (16KB or 32KB)
- MegaROMs (128KB or bigger)

These are not physical cartridges, but copies that are stored on the harddisk of your Amiga. They can be obtained from several
FTP
sites.

- Disk-based games
- Disk images
- Tape images

These are also available through
FTP
, from the same sites.

1.9 "

The control window

This is the main window of fMSX Amiga. Closing it closes the entire program. In it you can set the most important options.

Skipped frames

This is the number of frames skipped before a new frame is drawn. If this number has a low value the emulation runs smooth and slow. If the refresh

cycle has a high value it runs faster but also less smooth. Try to experiment with what works best for your system.

Interrupt rate

This controls the speed at which interrupts are generated. Normal MSX machines run at either PAL (50 interrupts per second) or NTSC (60 interrupts per second) speed, but fMSX Amiga also allows you to use a custom speed (from 10 to 100 interrupts per second), allowing you to run programs at exactly the speed you desire.

Cartridge

Here you can specify the name of a cartridge file. It will be loaded and executed when you reset the MSX.

Cartridge type

This button specifies the memory mapping method that will be used for the currently inserted cartridge. If this value is not set correctly the cartridge will certainly not run! Note: this is only required for cartridges larger than 32KB.

If you do not know the correct setting for a cartridge you have no option but to try them all. However, a list of known settings is available here.

Disk

In this area you can select which disks are used by the MSX. You can specify the number of disks used, the names of the disks, and whether the disks are currently in use by the MSX or the Amiga.

Drive A: and B:

In these gadgets you have to specify what is used as an MSX disk. You have three options:

Use a real device

You can use any device (such as a diskdrive or ramdisk) that conforms to certain parameters. These are:

- The device must have a blocksize of 512.
- The device must have a low-cyl of 0.
- The device must have a high-cyl of 39 or 79.
- The device must have 9 blocks per track.
- The device must be a real device, no assign or volume.
- The device must be mounted.

To use a device, just type its name in one of the drive gadgets.

Example: "PC0:".

Use a disk image

A disk image is a copy of a real disk, stored on your harddisk. Several types are supported, including raw, .IMG, and .DDI.

To use a disk image, type its name in one of the drive gadgets.

Example: "hd2:msxdisks/PassengersOnTheWind.DDI"

  Use a directory

fMSX Amiga can create a temporary disk image from a directory that contains MSX programs on the fly. Temporary disks created in this fashion are always write protected (this means that the MSX can never write to your harddisk directly). However, you can make a copy of the image after it was created and use that as a normal (writeable) diskimage. Only one temporary disk can be used at any one time (the second disk, if in use, must be a real device or disk image).

To use a directory as a temporary disk image, again just type its name in one of the drive gadgets.

Example: "hd2:msxdirs/A/Avenger/"

  Drives

With this gadget you can specify the number of drives connected to the MSX. Note that two drives need more MSX memory than one drive. Some games will not run if you have two drives connected due to memory shortage.

This setting only takes effect when you reset the MSX.

  Lock drives

If this button has been checked the emulation can access the drive, but the Amiga is locked out. Similarly, if it is not checked, the Amiga can access the drive but the MSX cannot.

  Running / Paused / Music

If you need every last cycle your machine can provide, but are unwilling to quit fMSX because you just reached level 48 of Tetris, you can pause and restart the emulation with these buttons. When it is paused it takes virtually no CPU time.

You can pause and restart fMSX from the MSX screen by pressing the HELP key.

Music mode allows you to listen to music produced by the MSX without the overhead of constant screen redraws. This mode makes fMSX a reasonably effective music player.

  Reset

This button resets the MSX. It is necessary to reset the MSX for some settings to take effect.

  MSX version

Selects what version of the MSX ROMs is used. MSX2 games require MSX2 ROMs to work. MSX1 games will run fine with MSX2 ROMs, but you may want to use MSX1 ROMs because of the shorter boottime. Finally, MSX2+ is an enhanced type of

MSX that is virtually unsupported in fMSX. Future versions will add more support, but for now there is almost no point in using this feature.

1.10 "

The cheat window

This window allows you to find cheats in games. This is done by finding the address that stores the number of lives, ammo, etc., and modifying it.

To find the number of lives, start playing the game you want to cheat in. Put the number of lives (or whatever you want to cheat) in the "value to search for" gadget. Press the "search" button. A list of all addresses containing the value you specified is put in the listview on the righthand side of the window.

Now play the game again, until the value you want to modify changes (ie. you loose a life or something). Put the new value in the "value to search for" gadget. Again, press "search". All addresses in the address list are checked, and are removed if they do not match the new value.

You can repeat this process as often as you want, until presumably you have only one address left. At this point you can open a
memory editor
and change
the address to whatever value you like.

If you want to start a new search, make sure the "start new search" radiobutton is pressed.

1.11 "

The compatibility window

This window allows you to select special compatibility features that will be needed only rarely, and that give a significant performance penalty when selected.

Currently only one option is available.

Interrupt handler

There are two possibilities here: "fast" and "accurate". In 99% of all cases, fast will be your best choice. However, a few games (those that have been hacked by SMA and start with their intro) rely on very specific interrupt behavior. This behavior can be selected by setting Interrupt handler to "accurate".

If an MSX2 game fails to start, try it with Interrupt handler set to "accurate". If it works it may be possible to switch back to "fast" later on. In all other cases you are better off in "fast" mode.

1.12 "

The memory editor

Although unfinished, the memory editor is already quite useful. In it you can look at all pages of memory currently seen by the MSX. However, you can only change those pages that represent RAM (ROM pages cannot be changed). The type of page is listed at the bottom of the window.

At the left of the window you will find the address column. This shows the address of the first byte shown on the same line. You can move the cursor into the address column, and type a new address. This will cause you to be teleported to that address.

The next set of columns show bytes of memory in hexadecimal representation, and the rightmost column shows the same bytes in ASCII format. You can use both columns to examine memory and change it.

You can open an unlimited number of memory editors.

1.13 "

The system preferences window

This window allows you to specify several system related settings for the emulation.

Instruction threshold

On slower computers, MSX interrupts may happen too fast. In some programs this can cause problems, because certain critical code that must be executed before a new interrupt occurs has not yet completed.

With this slider you can set the minimum number of branch-type instructions that MUST be executed before an interrupt is allowed. In normal code there are about 4.7 normal instructions per branch-type instruction.

In interrupt driven games, setting this to a low number speeds the game up, while high numbers slow it down. If the program you run does most of its work outside the interrupt, setting it to a low number may actually slow things down.

CAUTION: if this number is set too low, it may crash the MSX. Try again with a larger number. A value of 1500 should always be safe.

Maximum MSX speed

This slider is used to precisely determine the speed of the MSX. This is only available in the '060 version of fMSX.

Device used for MSX joystick 1

Allows you to use your mouse instead of a joystick for MSX joystick 1. Useful for MSX programs that support a mouse. Examples of these are productivity

packages (such as Ease or Hybrid), but also games (such as l'Affaire, Arkanoid, or Passengers on the Wind).

Usually the mouse detection is done at the start of the program, so be sure to set this gadget correctly before you start anything.

Memory

With this gadget you can select the amount of memory the emulated MSX uses. Note that the amount of free BASIC memory is constant (about 23KB), no matter how big a number you put here. The biggest you will ever need is the 512KB setting. The other settings are provided to let you boast about the amount of memory your MSX can use ;-) . And remember, this is for only one memory mapper! A real MSX can use 14 for a total of 56MB!

Save preferences on quit

When turned on, fMSX preferences are saved when the program ends. When turned off it is necessary to save preferences by hand, using the menu option Preferences/Save as... or Preferences/Save default.

Use Japanese ROM ID

When turned on, the country ID bytes in the ROM files are set to Japanese. This causes some games to act differently. Examples:

- Nemesis 1 gets a new name (Gradius).
- Nemesis 2 uses Japanese text.
- Nemesis 3 gets an animated title screen.

Plug'n'Play

When turned on fMSX will reset automatically after you select a cartridge from the filerequester or when you drop a cartridge on the fMSX window. This way you get to play it even faster ;-)

Enable kanji ROM

When turned on, fMSX loads the kanji ROM into the system. This is a 256KB ROM that contains definitions for 8192 kanji characters. Not many programs require this ROM, and even then you need to be able to understand kanji before it is of any use.

If you see 16x16 blocks of solid color in places where you would expect characters, try enabling the kanji ROM.

Autofire

When turned on, autofire is emulated for the joystick buttons (only for the first button) and for the spacebar.

1.14 "

The audio window

In this window you can select the many sound options supported by fMSX Amiga.

Sound mode

Allows you to select which soundchip is currently in use by the MSX, and how it is emulated.

Off: In this mode fMSX does not produce any sound.

PSG: This is the Programmable Sound Generator, the standard MSX soundchip. The sound is emulated through direct hardware access. 85% of all software only uses PSG sound.

The PSG has three tone channels and one noise channel. PSG emulation is virtually perfect: it will even emulate sampled sound (usually produced through volume modulation of the noise channel).

SCC: This is the special soundchip used by Konami in some of their games. Instead of blockwaves, like the PSG, the SCC uses 32-byte samples which sounds much better. It has 5 channels, of which only 4 are emulated in this mode.

Games that support SCC sound can easily be recognized by the following features:

- The game is produced by Konami.
- It is a ROM of 128KB or bigger (only exception to this: SD Snatcher).
- PSG sound for this game consists of nothing more than beeps and clicks.

Examples of games that support SCC sound are:

MSX1 games	MSX2 games
Salamander	Space Manbow
Nemesis 2	Metal Gear 2
Nemesis 3	Quarth
King’s Valley 2	Gryzor
F1 Spirit	King’s Valley 2 MSX2
Parodius	The Snatcher (actually SCC+)
	SD Snatcher (actually SCC+)

In the box marked "SCC options", you can select to emulate the SCC as an SCC+ chip. The SCC+ is slightly different from the SCC (it allows five waveforms instead of four, and is addressed on different addresses). Only the two Snatcher games support SCC+ sound.

In the same box, you can also select to emulate the channel disable mode of the SCC. This enhances sound quality in Parodius, but tends to decrease it for other SCC games. The cause of this phenomenon is not known, but possibly Parodius has a different type of SCC chip as well.

PSG + SCC: In this mode fMSX uses AHI to play all PSG and SCC channels simultaneously. This is a bit slower than direct hardware access, but it sounds MUCH better, allowing you to hear the (often spectacular) SCC songs the way they were supposed to be heard. If you like Konami’s musical style (and who doesn’t?) you should really install AHI and listen to the songs in this mode.

PSG + FM-Pac: The FM-Pac is an OPL/L soundchip, which was sold in a separate cartridge. The FM-Pac supports 9 channels of FM sound, which is currently emulated at an extremely basic level. Quite a few games support FM-Pac. FM-Pac emulation requires AHI to be installed on your machine.

Concierto FM-Pac: In this mode fMSX uses the Concierto soundcard (which is an add-on to the Picasso IV graphics card) to emulate the FM-PAC. The emulation is spot-on; not surprising, since the OPL3 used in the Concierto is a descendant of the OPLL used in the FM-PAC. A Concierto soundcard is required to use this mode.

The FM-Pac will not work without its cartridge. Fortunately the cartridge is also emulated. Unfortunately there is not enough room in the memory map to do so freely - something else must go before the FM-Pac can work. In the "FM-Pac options" box you can choose where you want to put your virtual FM-Pac cartridge.

If you want to play a cartridge game with FM sound, choose the radiobutton marked FM-Pac in disk slot. After you reset the MSX, the diskdrives will no longer be available, but the FM-Pac will be.

On the other hand, if you want to play a disk-based game with FM sound, choose the radiobutton marked FM-Pac in cartridge slot. Again, resetting is required to activate this setting.

To get rid of the FM-Pac, select No FM-Pac and reset the MSX.

AHI mode

This button lets you select the AHI mode fMSX uses. AHI is used in the combined PSG/SCC soundmode, as well as the combined PSG/FM-Pac soundmode, and the AHI mode determines the quality of the sound in that mode.

Selecting higher frequencies increases the quality of the sound. Unfortunately higher frequencies require more CPU power as well, causing fMSX to run slower. Experiment to find a good setting.

You should not select a stereo mode, as fMSX only generates mono sound.

If you don’t have AHI installed this option is not available. AHI can be obtained from Aminet.

1.15 "

The video settings window

This window allows you to specify several video related settings for the emulation.

The screen selector

The listview at the top of the window shows the various screenmodes used by fMSX Amiga. The program will always attempt to use the screenmode with the smallest possible resolution and number of colors. Most programs will happily run with just the 256*212, 16 colors mode.

You can change the display mode used in any resolution and color depth by selecting that line in the listview, and pressing the "select screenmode" button. This will bring up an ASL requester which allows you to choose a screenmode of your liking.

Currently, the 24-bit modes are unused by fMSX Amiga. They will be required in a future version, when MSX2+ screens are also supported (MSX2+ can display up to 19268 colors on a screen).

It is possible to substitute an Extra Half Brite mode for any of the 256 color modes. fMSX Amiga will gracefully degrade to 64 colors.

Some of the modes presented in the listview are impossible on a real MSX, but can still occur when the software is using a screensplit of some sort (example: a 512*212 screen in 256 colors is impossible on a real MSX2, but the top half of the screen could be in 512*212 in 16 colors, while the bottom half is in 256*212 in 256 colors. fMSX Amiga doesn't deal well with this situation, but it will attempt to make things look good).

Go to current screenmode

With this gadget you can select the item in the listview that represents the current screenmode. Useful for when you want to make a quick adjustment, but do not know which screenmode is currently in use.

Select screenmode

Pressing this button brings up the ASL screenmode requester for the screenmode currently selected in the listview. You should choose a screenmode that fits the MSX screenmode you selected in the listview for optimal viewing pleasure.

Only refresh when active

When this option is turned on and the MSX window is deselected, the emulator will no longer redraw the MSX screen. This frees up some computing power when you work in another application while fMSX is running.

Hide titlebar

This turns the titlebar of the MSX screen on and off.

Run on public screen

fMSX can run on any public screen. You can specify a screenname, or you can select the default public screen by leaving the stringgadget blank.

This option is only available when CyberGfx or Picasso'96 is installed. It will not work on normal Amiga screens.

1.16 "

The tape window

In this window you can control how fMSX deals with tape images. Tape images

are stored by fMSX Amiga as a collection of files, similar to the way a real MSX stores files on tape. However, files on tape are ordered by nature, which is not quite true for files on disk.

To emulate the ordering of files on a tape fMSX uses an index file. This is a simple text file which lists all the files in the right order. The index file should be in the same directory as the files it indexes. For that reason it only needs to list filenames, without path information.

Index

The load button is used to load the index file of a tape. After selecting an index file the contents of the tape are shown in the listview. It is also possible to type in a pathname directly.

You can select a name that doesn't yet exist. If you do so, a new index will be created.

Append

The biggest problem with writing to a real tape is positioning it. No matter how careful you are, the day will come when you accidentally overwrite something important.

The 'append' option prevents this. When turned on, new files are always appended to the end of a tape. When turned off, new files will overwrite the currently selected file.

The tape contents listview

The listview shows the contents of the currently selected tape. The tape is positioned just before the highlighted item. This means that the next item loaded by the MSX (or saved, if 'Append' is turned off) will be the selected item.

It is possible to change the name of an item. To do this, select it and type the new name in the stringgadget at the bottom of the window. It is strongly recommended that you do this for newly created files. fMSX searches for possible new filenames by simply going through a range of filenames and trying them all. The more filenames exist in this range, the longer the search takes.

Up

This button moves the currently selected file upwards in the index, ie. forwards on tape.

Down

This button moves the currently selected file downwards in the index, ie. backwards on tape.

Mark

This button marks the currently selected file for deletion. It will be deleted when you press the 'Remove marked' button.

  Clear

This button unmarks the currently selected file.

  Remove marked

The Remove button removes all marked files from tape. The files are removed
from both the index and your harddisk.

1.17 "

                    The menu bar

  Project/About...

Calls up the 'about' requester.

  Project/Quit

Quits the program.

  Cartridge/Save config

Saves several settings in the icon of the current cartridge. If the
cartridge does not have an icon, the fmsx_rom.info icon (in the program
directory) is used. If this icon does not exist either, no information is
saved.

You can also edit the
                icon
                 manually.

  Diskdrives/Write bootblock drive A:
  Diskdrives/Write bootblock drive B:

The MSX cannot boot from disks that have a PC bootblock. With these menu
options you can write an MSX bootblock to one of the MSX drives.

CAUTION: some games have their own bootblocks. Overwriting them will
kill the game.

  Diskdrives/Create 360KB diskimage
  Diskdrives/Create 720KB diskimage

With these options you can create a new diskimage. Select the size you want,
pick a location and name in the filerequester, and it's ready for use. There
is no need to write a bootblock to a diskimage created in this manner.

  Tools/Cheat finder...

Open the
                cheat finder
                 window.

  Tools/Memory editor...

Open a
memory editor
window.

Settings/Open...

Opens a previously saved fMSX settings file.

Settings/Save as...

Saves a settings file to a location of your choosing. You can reopen such a file later by selecting the Settings/open menu option.

Settings/Save default

Saves fMSX settings. Settings are saved to the file fmsx2.prefs in the ENV: and ENVARC: directories.

Settings/Audio...

Opens the
audio settings
window.

Settings/Paths...

Opens the
paths
window.

Settings/System...

Opens the
system settings
window.

Settings/Tape...

Opens the
tape settings
window.

Settings/Video...

Opens the
video settings
window.

Settings/Z80...

Opens the Z80 settings window.

1.18 "

Icon tooltypes

For both cartridges, diskimages, and directories with MSX files, the optimal settings can be stored in an icon. The tooltypes of the icon are interpreted as if they were ARexx commands - thus any command that can be executed from ARexx is also valid as a tooltype!

For a complete list of commands, look here

1.19 "

The paths window

This window lets you specify paths to several external resources.

Enable external ROMs

If turned on fMSX will load the MSX system ROM files from disk instead of using the internal ones. You will need to specify what ROM files get loaded using the appropriate string gadget. Note that internal ROMs are always used when the external ROMs are not available.

MSX1 rom, etc.

In this set of stringgadgets you can select which ROMs are used by fMSX in the various modes. The ROM appropriate to the selected mode is used; if this ROM cannot be found on disk the internal version will be used.

Enable external fonts

When activated an external MSX font is substituted for the internal ROM font. If you can read Japanese, selecting the Japanese font makes many games readable.

MSX font

Here you can select which font the MSX uses as its ROM font.

Many fonts are provided, including:

Default This is the same font that is present in the internal ROMs.
Italic This is the same as the default font, but has italicized characters.
Japanese This is a font containing Japanese characters.
Cyrillic This is a font containing Cyrillic (Russian) characters.

Note that the fonts are in MSX format, not in Amiga format. The two font formats are not compatible with each other.

Temporary disk

Since version 2.0, fMSX Amiga can create disks on the fly from directories. A

temporary disk image is created that contains all the files from the selected directory. With this gadget you can select where the temporary disk image must be created and what its name should be.

1.20 "

Using MSX diskdrives

The MSX disk system is an extension of the original MS-DOS disk system. Anything you know about MS-DOS probably applies to MSX as well. This section gives a short overview of features.

MSX filenames, like MS-DOS filenames, are very limited: they have a maximum length of eight characters, followed by a three-character extension. In addition, no lower-case characters or spaces are allowed.

The number of files on a disk is limited to 112. To make matters worse, subdirectories are not supported by MSX.

There are two wildcards: * and ?. These correspond to the Amiga wildcards #? and ?.

Examples:
. Everything
*.BAS All files with extension .BAS
GAME*.* All files starting with GAME
*.B?? All files that have a B as the first part of the extension.

fMSX Amiga supports up to two diskdrives. The MSX calls these drives A: and B:.

There are two ways of dealing with disks on the MSX: by using
MSX-DOS
and from
BASIC
.

1.21 "

About MSX-DOS

MSX-DOS is a control program for the MSX which is based on the very first version of MS-DOS. It is required if you want to use some MSX programs, and it can be useful if you want to perform certain floppy operations. It is close enough to CP/M that some CP/M programs run on it; many others only require minor modifications to run.

MSX programs that require MSX-DOS can easily be recognized by their filename. Programs that end in .BAT are MSX-DOS script files; programs that end in .COM are MSX-DOS executables. Both types can be started by typing the filename (without the extension) after the prompt.

Getting started

To run MSX-DOS you need three things: an MSX-formatted floppy, and the programs MSXDOS.SYS and COMMAND.COM. Obtaining an MSX-formatted floppy is easy: any 720KB PC floppy will do.

Step 1: Find, buy, or format a PC floppy. Assuming you have CrossDos installed this can easily be done from your workbench.

Step 2: Open the Preferences window by clicking on the Preferences button, and check that the Drives button is set to 1 or 2. If it isn't, change the setting and reset fMSX by clicking the Reset button in the control window.

Step 3: Convert the disk to MSX format by selecting the menu option Write bootblock (for the appropriate device).

Step 4: Copy MSXDOS.SYS and COMMAND.COM to this floppy.

Step 5: Reset fMSX by clicking the Reset button.

Step 6: Wait until MSX-DOS has booted. One of the first things you'll see is the prompt, which looks like this: A>

When MSX-DOS boots it looks for a file called AUTOEXEC.BAT. This is a script file that is executed automatically if found.

Complete overview of commands

Each command is followed by one or more examples. The lines between square brackets represent BASIC equivalents for the MSX-DOS examples.

BASIC
Quits MSX-DOS and returns you to the BASIC interpreter. From there you can type CALL SYSTEM or _SYSTEM to go back to MSX-DOS. Note: this doesn't work if you didn't originally boot from an MSX-DOS disk.

Examples:
A>BASIC (starts BASIC)
[CALL SYSTEM] or [_SYSTEM]
A>BASIC start.bas (starts BASIC and runs the BASIC program start.bas)
[no equivalent]

COPY
Copies one or more files.

Examples:
A>COPY A:*.* B: (copies all files from drive A: to drive B:)
[COPY "A:*.*" TO "B:"]
A>COPY MSX.TXT PRN (copies the file msg.txt to the printer)
[COPY "MSX.TXT" TO "PRN"]

DATE
Shows the date. Optionally you can also change it.

Examples:
A>DATE (shows the current date)

```
[GETDATE A$ : PRINT A$] (requires MSX2 BASIC)

DEL
Removes a file from disk.

Examples:
A>DEL *.*   (removes all files)
[KILL "*.*"]

DIR
Shows the contents of a disk.

Examples:
A>DIR *.BAS   (shows all files with extension .BAS)
[FILES "*.BAS"]
A>DIR /P    (shows all files, pausing after every page of output)
[no equivalent]
A>DIR /W    (shows all files in shortened format)
[FILES]

ERASE
Same as DEL.

FORMAT
Allows you to format a disk. Not supported by fMSX Amiga, you'll have
to format a disk from the workbench instead. Don't forget to write
an MSX bootblock to it.

MODE
Change the width of the screen.

Examples:
A>MODE 40   (turns on 40-column mode)
[WIDTH 40]

PAUSE
Puts the text Strike any key when ready... on the screen and waits until
any key has been struck. It can also print an extra message.

Examples:
A>PAUSE     (waits until a key has been struck)
[no equivalent]
A>PAUSE Message   (prints Message on screen and waits)
[no equivalent]

REM
Does nothing at all. If any text follows the REM statement it is printed
on screen.

Examples:
A>REM Message   (prints Message)
[no equivalent]

REN
Renames one or more files.

Examples:
```

```
A>REN *.LDR *.BAS (changes all .LDR extensions to .BAS extensions)
[NAME "*.LDR" AS "*.BAS"]

RENAME
Same as REN.

TIME
Shows the time. Optionally you can also change it.

Examples:
A>TIME      (shows the current time)
[GETTIME A$ : PRINT A$] (requires MSX2 BASIC)

TYPE
Prints the contents of a file on the screen.

Examples:
A>TYPE MSG.TXT    (shows the contents of MSG.TXT)
[COPY "MSG.TXT" TO "CON"]

VERIFY
Turns on verification mode. This command is actually ignored by MSX.

Examples:
A>VERIFY ON   (turns verification on)
[no equivalent]
A>VERIFY OFF    (turns verification off)
[no equivalent]
```

1.22 "

```
                    Using the drives from BASIC

Unlike MSX-DOS, BASIC enforces no filenaming conventions for executables
or other kinds of files. However, some guidelines exist:

Files ending in .BAS are BASIC files. There are several ways to load them:

LOAD "FILENAME.BAS"   (loads the file)
LOAD "FILENAME.BAS",R   (loads the file and executes it)
RUN "FILENAME.BAS"    (loads the file and executes it)

If you want to save a BASIC file you can use this:

SAVE "FILENAME.BAS"

Files ending in .BIN are executable files. There is only one way to load
them:

BLOAD "FILENAME.BIN",R    (loads the file and executes it)

Files ending in .BAT or .COM are files that can only be used from
                MSX-DOS
                .
```

If the file you want to load has another kind of extension, try
experimenting with it. The MSX will tell you if you did something wrong.

There is one file that has special significance to MSX-BASIC. This
file is called AUTOEXEC.BAS and is executed automatically during the
boot sequence.

The section about
                MSX-DOS
                 contains BASIC equivalents for some MSX-DOS
statements.

1.23 "

    Quitting the emulator

There are three ways to quit the emulator: select Quit from the menu,
close the control window, or send an ARexx quit command to the ARexx port.

If the option 'Save preferences' is activated, quitting the emulator also
causes it to write many current settings to disk. These values are
automatically reloaded the next time you run the program.

1.24 configure

    Configuring your MSX

A real MSX2 has a small amount of battery-backed RAM, which is used to
store (among other things) information about your preferred working
environment.

The following commands are used to customize the environment:

  SCREEN n

Sets the desired screenmode. n can be 0 or 1.

  WIDTH n

Selects the amount of columns of text displayed on screen. Valid values for
n are 80 or less for screen 0, and 32 or less for screen 1.

  COLOR fore,back,border

Selects foreground, background, and border colors. Screen 0 does not support
border colors - one of the many weird features of the MSX2 VDP. Colors range
from 0 to 15.

  KEY [ON|OFF]

Turns the function key list underneath the screen on or off.

```
  SET PROMPT "prompt"
```

Changes the prompt to "prompt". The maximum length of the prompt is six characters. Tip: if you often use the "Files" command (like me), it may be useful to set it as the prompt.

```
  SET SCREEN
```

Stores the current settings in battery backed RAM. fMSX actually keeps a copy of these values on disk, in the file fmsx.prefs.

1.25 "

```
                    Using the Amiga hardware

  Keyboard
```

Keys are directly mapped to their MSX equivalents, which means that the MSX ROMs determine what character results from a key press. The ROMs that come with this version of fMSX Amiga give you an American keymap.

Some keys that may not be obvious:

```
Amiga key   MSX key
F6      select
F7      stop
F8      clear/home
F9      insert
F10/delete    delete
left alt    graphics
right alt   code
```

and on the numeric keypad:

```
)     ,
(     steal Amiga mouseport for use by the MSX as a second joystick
```

The following Amiga keys do nothing at all:

- Both Amiga keys.
- The extra international keys found on some keyboards.

```
  Joystick
```

The emulator can use two Amiga joysticks. The second joystick is activated by pressing '(' on the numeric keypad while in the emulation screen. Press '(' again to get the mouse back.

Joysticks with two buttons are very common with MSX systems, and many games take advantage of the second button. Often there is a keyboard equivalent for the second button; Konami games use 'M' and 'N' for this.

```
  Video
```

The emulator should run fine on any Amiga screen, including PAL, NTSC,

CyberGfx, and Picasso'96.

  Audio

Depending on which soundmode is selected, fMSX will either use the Amiga sound hardware directly or use AHI. AHI is a system for playing up to 128 channels of sound, and is available on Aminet.

  The Concierto soundcard

If a Concierto soundcard is installed, fMSX can use it to emulate the FM-Pac. FM-Pac emulation is spot-on using this card. No wonder - the OPL3 chip in the Concierto is a descendant of the OPLL chip in the FM-Pac.

  Clock

MSX2 machines have a built-in battery-backed clock. The emulator substitutes the Amiga clock for this clock.

  Diskdrives

MSX disk drives are emulated as Amiga devices. Read the section about

                diskdrives
                 for more information.

  Mouse

Some MSX2 programs can use a mouse. Activate the mouse in the system preferences window, and it will be available for use in those programs.

Normally it is turned off because an active mouse tends to confuse software that is expecting a joystick.

1.26 "

    Future plans

- Complete printer support
- Support for MSX2+ and MSX Turbo-R
- Support for horizontal panning
- The return of the second cartridge
- Snapshots
- Subslots
- Optional checks for sprite overrun or collision
- A 'database' for cartridge and / or disk settings (so no icons are necessary to play programs directly from CD)
- Emulation of MSX networking, including a version that works over the internet ;-)
- Serial port emulation
- Improved FM-Pac support for AHI
- A revolutionary GUI that will be completely user-definable (it is currently in the design stage, I don't know whether it will ever appear)
- Volume controls
- Better comparable volume (FM-Pac is relatively loud now)

```
- Gadgethelp
- HTML documentation
- Better memory viewer
- Direct support for Colecovision

and of course as always
- Better compatibility
- Higher speed
```

1.27 "

```
    Where do I find MSX software?

MSX software is available through FTP from the following sites:

  Site:       Directory:
  ftp.funet.fi      /pub/msx/
  komkon.com        /pub/MSX/
  jazz.snu.ac.kr      /pub/msx

MSX software is also available via WWW:

  Site:
  http://surf.to/fmsx
  http://www.casema.net/~tfh/
  http://www.cs.umd.edu/users/fms/
  http://www.upcnet.upc.es/~jsp4/msx.html
  http://dynamo.geol.msu.ru/msx/
  http://home.earthlink.net/~tbelmont/msx/index.html

  ... and many others!
```

1.28 "

```
    Why are the icons so ugly?

There are two possible reasons.

The first is that your Workbench palette is different from mine. If
you want to see what the icons should look like, set your Workbench
to 8 colors, and use the following palette:

color #0-3: standard Commodore colors
color #4: full red
color #5: - (unused)
color #6: full blue
color #7: - (unused)

The second reason is that I am not an artist. Feel free to draw nicer
icons, if you like. And *SEND ME A COPY*!

A new set of icons was donated by Johan Forsberg (d92-jfo@nada.kth.se).
```

These icons use the Magic Workbench colors and are much better than my own
poor attempt at art.

1.29 "

I don't understand how I'm supposed to load program <xxx>...

Diskbased software

An MSX filename consists of eight characters, followed by a dot and another
three characters. The last three characters are called the extension.
The extension generally tells you how to load a file. So, if the filename
is...

xxx.BAT or xxx.COM
Go to
MSX-DOS
and type xxx.

xxx.BAS
Try the following: in BASIC, type RUN "xxx.BAS". If it responds with an
error message it isn't a BASIC file, try loading it as an executable file.

xxx.BIN
Try the following: in BASIC, type BLOAD "xxx.BIN",R. If it responds with an
error message it isn't an executable file, try loading it as a BASIC file.

xxx.PMA
This is an MSX archive file, similar to a .LHA file on the Amiga. You
need PMEXT.COM (available from several
FTP sites
) to unpack it.
PMEXT.COM requires
MSX-DOS
.

xxx.DSK
This is a disk image. See below for more information.

any other
Try loading it as either a BASIC file or an executable file.

If none of these work, it is a file that is part of another program,
in which case it can only be loaded by that program.

Tapebased software

Tapebased software consists of a group files and an index file (see the
description
here
.

There are three ways to load files from tape:

BLOAD "CAS:",R

This command loads the first binary file it can find from tape and executes it.

```
LOAD "CAS:",R
CLOAD
```

These commands load the first BASIC file they can find from tape. The MSX has two ways to store BASIC files on tape - and they are not compatible. You must use the right command for each type of file.

After loading a file with CLOAD you must type RUN to activate it.

ROM files and disk images must be loaded from the Amiga side

Only MSX files can be loaded using MSX commands. ROM files and disk images are not really MSX files (a real MSX cannot load them) and can only be loaded from the Amiga side.

To load and activate ROM files you must use the "Load" and "Reset" buttons on the control window.

Disk images can be loaded from the system preferences window. Just type the name of the disk image in one of the "Device A:" or "Device B:" gadgets.

1.30 "

Program <xxx> does not work, what can I do?

Note: many tricks described here require you to reset fMSX before they take effect.

There is an ARexx script in the ARexx drawer that sets fMSX to its most compatible settings, as advised here. The script is called MostCompatible.fmsx.

Try a higher instruction threshold

Some games perform lots of processing between every two interrupts, and the emulated Z80 may not be able to keep up (this is especially true for graphically intensive operations). If interrupts happen too early this may cause the MSX to crash or behave erratically.

Setting the instruction threshold to a higher value causes the emulated Z80 to perform at least a certain number of instructions between two interrupts. This means that the game will run slower, but there is also a better chance of it working at all.

A good example of a game that needs a higher interrupt period is Maze of Galious.

The amount of instructions executed cannot easily be determined, but is normally about five times higher than the instruction threshold.

Although it is possible to enter values as high as 5000 it is not normally necessary to go above 1500. Note that you do not need to reboot for this setting to take effect.

Use less drives

Every drive costs a bit of BASIC memory. Many games will not run when more than one drive is present, because the BASIC loader can run out of memory. Turning off the second drive (using the appropriate gadget in the system preference window) may help. Allocating more memory to the MSX will not help, this does not increase BASIC memory at all. Such are the wonders of the MSX memory architecture...

Use more memory

Some games require an MSX device called a memory mapper to be installed. 98% of all programs will feel at home on a 256Kb mapper, and the rest will run fine with 512Kb. The amount of memory used by the memory mapper can be selected in the system preferences window.

Use MSX1 ROMs

Some MSX1 games do not properly turn off MSX2 features, causing garbage to appear on the screen (shades of ECS vs. AGA here?). Disable MSX2 in the system preferences window and see if the problem disappears.

Try hacking with BASIC

If all this doesn’t help there is still a whole bag of BASIC tricks left to try. Some of the following might do it - then again, it might not:

POKE &hFD9F,201

This turns off disk interrupt processing. On a real MSX this means the drive engine never stops (this is regulated through this interrupt) but fMSX controls the drive engine on its own.

CLEAR 0

This frees up extra BASIC memory.

MAXFILES = 0

This frees up extra BASIC memory.

Poking address -1 doesn’t help

The mythical address -1, aka 65535, aka &hFFFF, is used by a real MSX for secondary slot selection. fMSX Amiga supports nor needs secondary slots, therefore pokes to this address are ignored. This is a real feature, ask almost any MSX2 owner ;-)

Get the next release

fMSX Amiga is still under development. I try to have better compatibility with each new release. If something doesn't work and isn't listed in the bugs section send me a report and I'll see what I can do.

1.31 "

Will you ever implement snapshots?

Since many other features have now been realized, I have been thinking about this again lately. I may give it a try in a later version.

However, it won't be easy. The MSX does not have a flat memory space, but consists of many different blocks of RAM, ROM, videoRAM and countless customchip settings. Saving all of these would require a format that is highly future proof, and preferably portable to other platforms. Designing such a format is a difficult and time consuming job.

The real problem is not in saving though: when fMSX is running it constantly builds up internal state information. Most information is kept twice: once in MSX format, and once in pre-translated Amiga format. Reloading a snapshot would require a complete translation of all loaded MSX data into Amiga data, something which is very hard to do correctly.

1.32 "

Will you re-implement the second cartridge?

Quite possibly, yes!

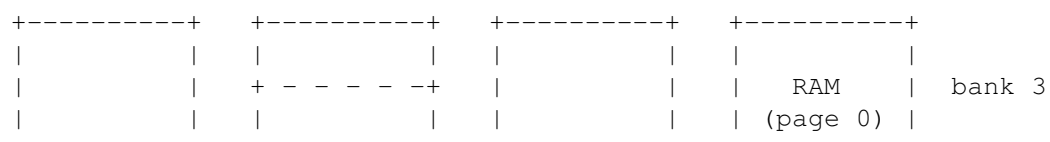
In the future fMSX may have the option to use subslots. Subslots are an MSX2 feature that allows bigger memory maps. Because subslots cause some serious compatibility problems, use of subslots will be software selectable.

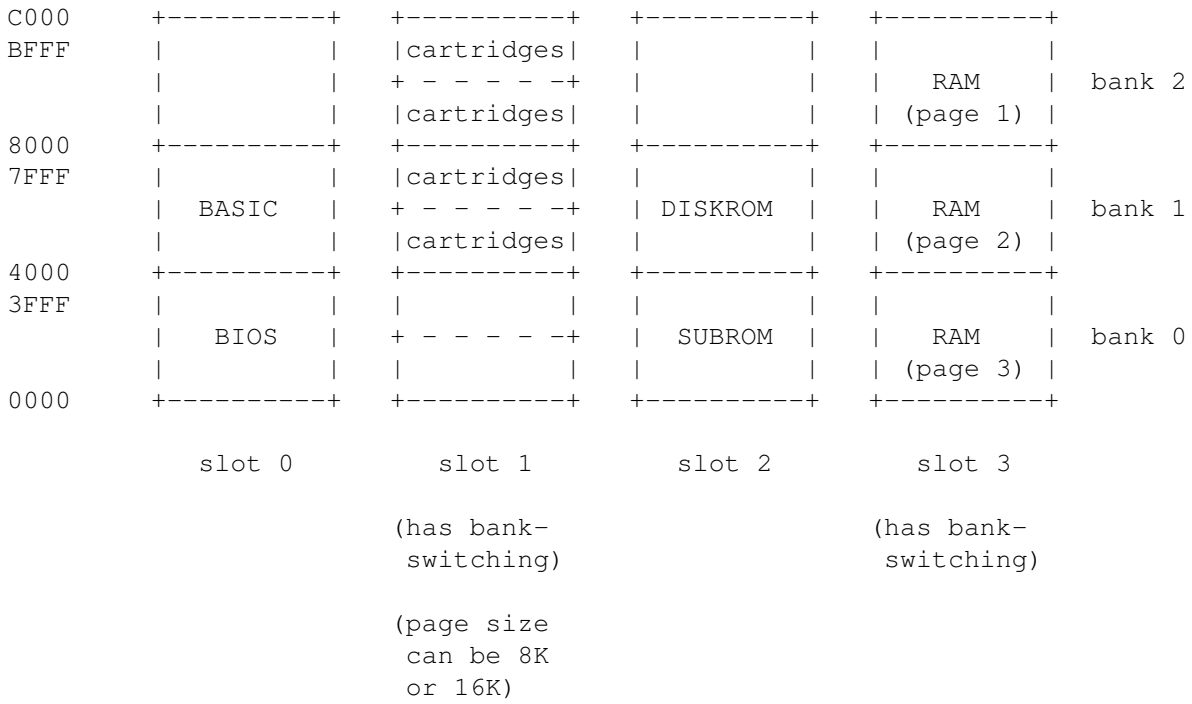
When subslots are implemented, it will be possible to select the contents of both the first and second slot (the second slot is currently reserved for the MSX2 SUBROM and disk ROM or FM-Pac ROM).

In case you are wondering how MSX memory works, here comes the hard to understand, detailed, technical explanation. Don't worry if you don't understand it.

The MSX memory model as implemented by fMSX

This is the twisted mess that is the MSX memory model, as used by fMSX Amiga. The MSX standard does not quite specify what goes where, but this meets all criteria and is common enough that most software runs on it.





Slot 0 always contains the BIOS and BASIC ROMs. Without these the MSX cannot even boot.

Slot 1 may or may not contain a cartridge. A cartridge can have 8K or 16K pages (unlike the rest of the system that works with 16K pages, only). Cartridges can have up to 256 pages. On a real MSX a cartridge can be all sorts of things: an extra soundchip, a serial port, a harddisk, more RAM, etc. In fMSX only two types of cartridge are supported: pure ROM, and ROM + SCC soundchip.

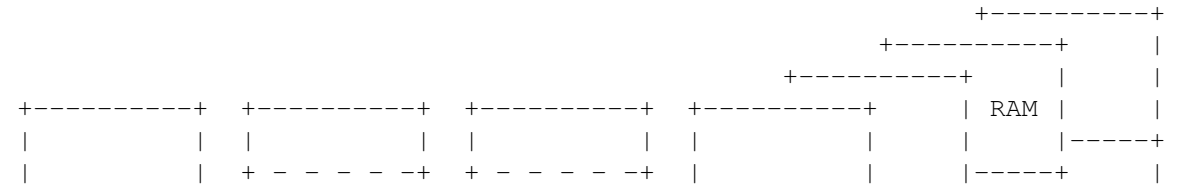
Slot 2 contains the MSX2 SUBROM if MSX2 is turned on, and the DISKROM if disks are turned on.

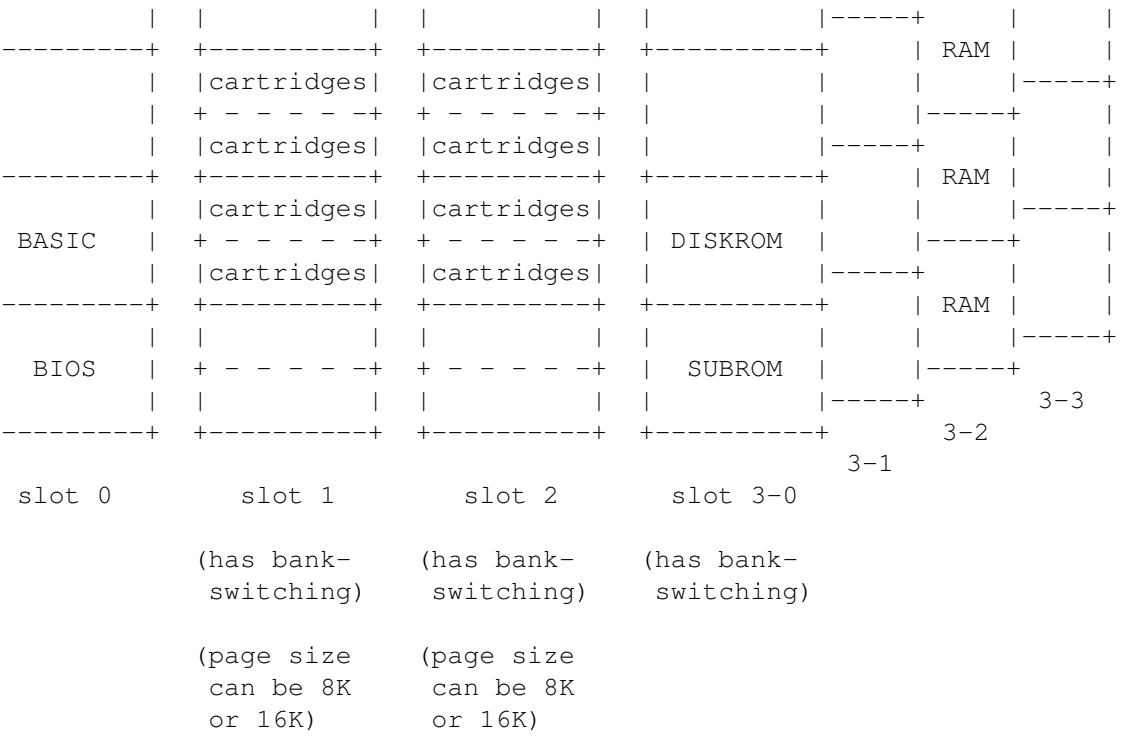
Slot 3 contains the RAM. Notice how the pages are numbered different from the banks. This is probably not _specifically_ meant to confuse people... There can be up to 256 RAM pages.

The only slot that can contain a second cartridge is slot 2. Unfortunately, slot 2 is already taken by the diskROM, which means that you would have a choice of either using the second cartridge or the diskdrive.

So how does a real MSX cope?

A real MSX has another gimmick: expanded (or sub-)slots. fMSX Amiga does not have subslots, which I consider a feature. When subslots are present the picture changes, like in this example:





Slots 0 and 1 remain the same. Slot 2 can now contain a second cartridge. Slot 3 has been expanded and contains both the DISKROM, SUBROM, and all RAM.

Tradionally subslots have been a source of incompatibility. Most software cannot find the RAM in slot 3-2, banks 0 and 1: it is 'covered' by the DISKROM and SUBROM, unlike the RAM in banks 2 and 3.

Each slot can be expanded, and each subslot can contain RAM (except the subslots that hold the BIOS / BASIC and DISKROM / SUBROM). Since there can be up to 256 pages of RAM in a subslot this comes down to 14 * 256 * 16K of RAM, or 56M!

1.33 "

What is the difference between the normal and '060 versions?

The normal version runs at the maximum possible speed. On the 68060 this is usually much too fast, which makes some games unplayable.

The '060 version has special delay-code built in, which causes it to run at exactly the same speed as a real Z80. This code in itself slows fMSX down quite a lot!

THE 68060 VERSION IS A LOT SLOWER THAN THE 'NORMAL' VERSION!

1.34 "

Will you implement a PPC version?

I don't know yet. If I do, it will be a lot of work. Certainly all of the Z80 would have to be rewritten, as well as all the graphics routines.

Even then it may be slower than the current version. This depends heavily on how much IO is really going on, ie. how often the program needs to switch back to 680x0 mode.

For now, don't hold your breath.

1.35 "

How to obtain new versions

New versions of fMSX Amiga will be distributed on Aminet, and I also mail a copy to everybody on the fMSX Amiga distribution list. If you want to be placed on this list (or removed from it) simply send mail to

hguijt@inter.nl.net

stating what you want. If you have mailed me in the past with questions about fMSX you have automatically been placed on the list; I apologize if you did not intend to receive new versions of fMSX. If you asked to be on the list but aren't, try mailing me again. I am not a mail-demon, and I occasionally misplace mail messages.

You do not need to fear being swamped by mail; there is usually a few months between each successive version of the emulator.

1.36 "

About comp.sys.msx

For public questions about MSX or fMSX, your best bet would be the newsgroup comp.sys.msx.

I'm no longer on comp.emulations.misc. There is just too much noise in that group.

Of course, questions can be aimed at the
author
as well.

1.37 "

About the author

fMSX Amiga is being written by Hans Guijt. I studied Computer Science at the

University of Leiden from 1989 to 1995. After that I found a job with Palm Automatisering in Aalsmeer, where I programmed administrative software for the flower trade. In 1997 I changed jobs again, this time to Terma Elektronik A/S, a Danish company who have hired me out to the European Space Agency. I still work for ESA at their ESTEC site, where I am responsible for designing and building the STAMP system.

Past and future projects include:
- Photobase: a large UNIX based system for image normalization and recognition, written for the University of Leiden.
- HuKra: a big administrative package for the flower trade, written in PowerBuilder.
- OZ: a data analysis package for the flower trade, also written in PowerBuilder.
- TempDAS: a small system for thermal measurement and presentation written in C++.
- STAMP: a huge system for thermal measurement and presentation, also written in C++.

I also did some three months of debugging on the Envisat EGSE software. This system is some 600,000 lines of code (20 megabytes, in case you are wondering). All of it in K&R C (programmers now run in horror!). I did a good job on it, but barely managed to keep onto my sanity. One day I will write a horror novel describing my experiences with this piece of source.

My (snailmail) address:

  Hans Guijt
  Kagersingel 30
  2172 XG Sassenheim
  The Netherlands

  Telephone: (Holland) 2522-17251
  Email: hguijt@inter.nl.net

1.38 "

    About RAMSX:

Included in this package is a mountlist for a device called RAMSX:. This device (built from ramdrive.device and crossdosfilesystem) can be used with fMSX to play disk-based games directly from RAM (it’s not as good as using a harddrive, but it’s much better than using disks!).

If you want to use RAMSX: you will need about 720Kb of free RAM and the crossdosfilesystem. You must also enter the device name RAMSX: into the appropriate drive gadget in the preferences window.

It can be rather hard to recover RAMSX after a reboot, but it can be done. Open a shell and type the following commands:

```
  assign RAMSX: dismount
  mount RAMSX:
```

and RAMSX: is back online. Thanks to Nikos Drogosis (ndrog@acropolis.net)

for reporting this.

1.39 "

About FMSX0: and FMSX1:

Included in this package are mountlists for two devices called FMSX0:
and FMSX1:. These devices require fmsdisk.device to be installed.
fmsdisk.device is available from Aminet (directory misc/disk).

FMSX0: and FMSX1: allow you to store MSX disks on the Amiga harddisk.
If you wish to easily access floppy-based MSX disks but cannot be
bothered with real floppies these devices are an excellent alternative.

FMSX0: and FMSX1: were contributed by Andrew G. Robson
(a.g.robson@northumbria.ac.uk). Complaints and compliments are
rightly his.

1.40 "

A big thank-you to:

Marat Fayzullin (marat@giganda.komkon.org)
Marat is the author of fMSX UNIX, on which fMSX Amiga was originally based.
Since fMSX Amiga 2.2, no code of his remains in fMSX Amiga, but without his
original work fMSX Amiga would not have existed.

Peter McGavin (peterm@maths.grace.cri.nz)
Peter (who wrote an excellent Spectrum emulation) contributed many
ideas for the Z80 emulation. It was his Spectrum 1.7 which convinced me
that fast Z80 emulation is possible. In addition the c2p necessary
for the MSX2 screenmodes were inspired by his Flick 1.5.

Jeroen Vermeulen (jtv@xs4all.nl)
Jeroen kindly offered to proof-read the accompanying documentation, and
found heaps of spelling errors. He is also a willing betatester (victim?)
who has found many enforcer hits and general nastiness.

Alex Wulms (awulms@inter.nl.net)
Alex owns an MSX Turbo-R, and is the author of Zone Terra, an excellent
shoot'em-up for that system. He is also an MSX-wizard, who explained
many of the MSX's internal workings to me.

Sergi Martinez and Juan Gomez (pixador@adam.es)
Juan and Sergi are the authors of
AmiMSX
, a program similar to fMSX. Their
input so far includes a trick to speed up fMSX by 30%, information about
the MSX mouse interface (to be added in a future version), and of course
the constant inspiration provided by new AmiMSX features ;-)

Martin Blom (lcs@lysator.liu.se)

Martin is the author of AHI, a system for playing up to 128 channels of
sound at the same time on a fast Amiga. AHI made the shared PSG/SCC mode
possible, and is so much better than my own attempts at channel mixing
that I can safely say it would never have happened without it.

Kenny (kenny@bgnett.no)
The otherwise unnamed Kenny posted a short ARexx tutorial on
comp.sys.amiga.programmer. Thanks to his writings fMSX now has an ARexx
port.

Børge Nøst (borgen@icenet.no)
For finding, fixing and reporting a major bug in high-speed mode.

Andrew G. Robson (a.g.robson@northumbria.ac.uk)
For contributing
                FMSX0: and FMSX1:
                .

Nikos Drogosis (ndrog@acropolis.net)
For explaining how
                RAMSX:
                 can be remounted after a reboot.

Johan Forsberg (d92-jfo@nada.kth.se)
For donating the Magic Workbench iconset for MSX.

Mark Knibbs (markk@netcomuk.co.uk)
For reporting lots of bugs and misfeatures.

Roderick Mouthaan (roderick@xs4all.nl)
For setting up an fMSX homepage (http://surf.to/fmsx). He also sent me a huge
number of bug reports, many of which are still waiting to be fixed.

...and all the betatesters and supporters, and everyone who kept nagging me
throughout the empty years between fMSX 1.4 and fMSX 2.0!

1.41 "

    About AmiMSX

fMSX is not the only MSX emulation available for the Amiga. A similar program
called AmiMSX is also available through Aminet. The two programs have pretty
similar feature sets, but fMSX is aimed more at high-end machines while AmiMSX
runs better on low-end machines.

Since fMSX is aiming for high-end machines it can and does support high-end
hardware: graphics cards, soundcards, etc. Because of this it is also able to
support more MSX hardware: full SCC, FM-Pac, mouse, on-the-fly disks, etc.

Finally, fMSX is still under development.

If you like MSX emulation you will probably want to check out AmiMSX as well.