fmsx

COLLABORATORS

	TITLE : fmsx	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

fmsx

1.1 "

```
                        fMSX Amiga ARexx commands

General information

                 fMSX and ARexx

                 Limitations
               Alphabetical list of commands


                 AnchorScreen

                 AutoSavePrefs

                 DeviceA

                 DeviceB

                 DirectColorLoad

                 DoubleBuffer

                 Drives

                 EnableExtFont

                 EnableExtROMs

                 EnableKanji

                 HideTitlebar

                 InterruptPeriod

                 JapaneseID

                 KanjiROM
```

LoadCartridge1

LockDrives

Memory

Mode

MSXFont

MSX1ROM

MSX2ROM

MSX2SubROM

MSXVersion

PlugAndPlay

Quit

RefreshCycle

RefreshWhenActive

Reset

RespectBlankBit

ROMType1

SCCDisable

SCCPlus

SoundMode

Version

1.2 "

fMSX and ARexx

Since version 1.4 fMSX has an ARexx port. This document lists all available commands and their options. It assumes that the reader already possesses some knowledge about ARexx, but don't worry if you don't: I know almost nothing about it either and I am supposed to write this stuff. Besides, documentation describing ARexx is widely available on Aminet and other places.

The ARexx portname is different for each running copy of fMSX. The first copy is called FMSX.0, the second copy FMSX.1, and so on. The example scripts assume you are running only one copy. If you wish to address

multiple copies you'll have to modify the scripts yourself.

Generally speaking all commands that set some attribute of fMSX also return the current setting of that attribute, and can be used without arguments just to obtain that setting. As an example, take a look at the mode command. It can be used to pause fMSX like this:

```
mode pause
```

However, if you want to toggle the mode between running and pause you would have to obtain the current mode and invert it, like this:

```
options results       /* We want to hear about results */
address FMSX.0        /* Talk to fMSX */
mode                  /* Find the current mode */
if result = run then
    mode pause        /* Pause if we were running */
else
    mode run          /* Run if we were pausing, or in music mode */
```

More examples are available in the ARexx directory of the fMSX distribution.

If you want to see more ARexx commands added, or if you have written your own scripts and want to share them with the world, please contact the author.

1.3 "

Limitations

In it's current form the fMSX ARexx port suffers from several problems. The most important of these is that it doesn't check very well for faulty input. In other words: it will recognize arguments that it is supposed to recognize, but it will not report errors when it finds arguments it cannot.

This situation will be remedied in the future. If you write your own ARexx scripts, be sure to specify valid options ONLY, or your script will mysteriously fail one day.

Currently only the first 1000 copies of fMSX can have an ARexx port (other copies will run but will not feature a port). If this limitation poses a problem please contact the author.

1.4 "

AnchorScreen

Purpose: to change screen anchoring.

Format: [ON | OFF]

Available: 1.4

Description: this command changes the screen anchoring used by
fMSX. When screen anchoring is turned on, an extra screen called the anchor
screen is opened. The MSX screen is then always opened relative to the
anchor screen, which means that it always stays at the same depth (and
doesn’t pop to the front every time a screen attribute changes).

On the downside, the anchor screen requires a small amount of CHIP memory.

Screen anchoring is only available in OS 3.0 and better. It is ignored for
earlier versions of the OS.

1.5 "

AutoSavePrefs

Purpose: to change the automatic saving of preferences.

Format: [ON | OFF]

Available: 1.4

Description: this command changes the automatic saving of preferences.
Normally fMSX saves it preferences to ENV: on exit, but when automatic
saving is activated preferences are also saved to ENVARC:.

Apart from fMSX settings the preferences file also contains the values
stored in the MSX battery-backed RAM. These include the initial screenmode,
color settings, whether function keys are on or off, prompt, password, etc.
A list of configuration options can be found here.

On a real MSX the battery backed RAM also contains time and date values,
but in fMSX these are always read from the Amiga’s clock. fMSX never tries
to change the Amiga clock - too many MSX programs reset the clock for some
reason.

1.6 "

DeviceA, DeviceB

Purpose: to insert a new disk in an MSX drive.

Format: [FILENAME | DEVICENAME]

Available: 1.4

Description: These commands insert a new disk in MSX drive A: or B:,
respectively. A ’disk’ can be either a device like PC0: or a hardfile
like fmsx:disks/columns. Either way, it must be something fMSX can recognize
as a disk.

For a hardfile to be recognized as a disk, it must be either (precisely)

360KB or 720KB long. A device must have 40 or 80 tracks, and 9 blocks per track.

As with a real MSX, you shouldn't insert disks while the MSX is using that drive. If you do so anyway you risk corrupting the data on the disk.

1.7 "

DirectColorLoad

Purpose: to change direct color loading.

Format: [ON | OFF]

Available: 1.4

Description: this command turns direct color loading on or off.
When turned on, palette changes are immediately send to the screen. This allows certain graphical effects that would not otherwise be possible, but causes some visual confusion on other screens. While ugly, this is harmless.

When turned off fMSX is more OS-compliant. Unfortunately this means it can only change the palette of the MSX screen once per frame.

1.8 "

DoubleBuffer

Purpose: to change the way the display is buffered.

Format: [OFF | OS2 | OS3]

Available: 1.4

Description: This command changes the amount of screen buffers fMSX uses, and the method used for swapping them. There are three options:

When double buffering is turned off, the screen will flicker wildly in some games. However, turning double buffering off saves one screen worth of memory.

OS 2-style double buffering is available on all Amiga's. It works by swapping two views back and forth. While this lessens the flicker it does not completely eliminate it.

OS 3-style double buffering is absolutely flicker free. Obviously it is only available on machines equipped with (at least) OS 3.0. Somewhat less obviously, it doesn't work with CyberGraphics. Users of this system should choose OS-2 style doublebuffering or none at all.

OS 3-style double buffering is automatically changed to OS 2-style double buffering for machines running OS 2.

1.9 "

```
        Drives

    Purpose: to change the number of MSX drives used by fMSX.

     Format: [ 0 | 1 | 2 ]

  Available: 1.4

Description: this command changes the number of drives used by the MSX.
An MSX can use up to two drives, known as A: and B:. In most cases you will
want to use one drive (using a second drive uses up more MSX RAM, which will
cause some programs to fail).

Note that this setting takes effect when you reset the MSX.
```

1.10 "

```
                        EnableExtFont

    Purpose: to change external font loading.

     Format: [ ON | OFF ]

  Available: 1.4

Description: this command changes whether fMSX will use the internal
ROM font of the currently loaded MSX ROMs or an external font of your
choosing. See
                MSXFont
                 for more information.

Note that this setting takes effect when you reset the MSX.
```

1.11 "

```
                        EnableExtROMs

    Purpose: to change external ROM loading.

     Format: [ ON | OFF ]

  Available: 1.4

Description: this command changes whether fMSX will use externally
loaded ROMs or the internal ones. See
                MSXROM
                 for more information.

Note that this setting takes effect when you reset the MSX.
```

1.12 "

EnableKanji

Purpose: to change whether Kanji ROMs are loaded.

Format: [ON | OFF]

Available: 1.4

Description: this command changes whether fMSX will use the Kanji ROM. See
KanjiROM
for more information.

Note that this setting takes effect when you reset the MSX.

1.13 "

HideTitlebar

Purpose: to hide or show the titlebar of the MSX screen.

Format: [ON | OFF]

Available: 1.4

Description: This command hides or reveals the titlebar of the MSX screen. When it is turned on you can easily drag the screen down. However, turning it off makes for a completer MSX experience.

1.14 "

InterruptPeriod

Purpose: to change the interrupt period.

Format: [VALUE]

Available: 1.4

Description: this command changes the interrupt period. The interrupt period is the minimum number of branches taken by the Z80 between two interrupts. Although this is not very precise, it is an adequate method to pace interrupts.

Some programs need a fairly high interrupt period. These programs do a lot of processing during the interrupt, and while a normal MSX can do all this processing in a single frame fMSX may not cope. If that happens it is necessary to set a high interrupt period, so that interrupts do not occur while processing from the previous interrupt is still going on.

An example of a game that needs a high interrupt period (at least, on my machine) is Maze of Galious. When the interrupt period drops below 1500 it stops redrawing the screen, leaving large black gaps.

Why some games seem to ignore the interrupt period: the Z80 has an instruction to wait for the next interrupt. When it is executed the emulation knows that no further processing will be done before the next interrupt occurs, and waits only for the next vertical blank (and not for the interrupt period counter to reach zero).

Note that only multiples of 100 are accepted for the interrupt period. Other values will be rounded to the next lower multiple of 100.

Acceptable values for the interrupt period lie between 0 and 5000.

1.15 "

JapaneseID

Purpose: to change the country ID in the ROMs.

Format: [ON | OFF]

Available: 1.4

Description: this command changes the way the country ID bytes in the MSX ROMs are patched. This can be useful because some programs work differently when they find Japanese ID codes (for example: Penguin Adventure and the Nemesis series).

Note that this setting takes effect when you reset the MSX.

1.16 "

KanjiROM

Purpose: to change the path to the Kanji ROM.

Format: [FILENAME]

Available: 1.4

Description: this command sets the path to the Kanji ROM, which is a 128KB ROM that contains bitmaps for 4096 common Kanji characters. Because the Kanji ROM is rather big and of limited interested to many people it is not part of fMSX, but has to be loaded externally.

You need to enable the Kanji ROM when you see solid, square blocks where you would expect characters.

Obviously, enabling the Kanji ROM takes 128KB of RAM.

1.17 "

```
        LoadCartridge1

    Purpose: to load a cartridge into fMSX.

     Format: [ FILENAME ]

  Available: 1.4
```

Description: this command specifies a cartridge name for use with fMSX. The cartridge will be loaded when fMSX is reset.

If a cartridge is loaded the return value is the name of that cartridge, including the path (if any). If no cartridge is loaded the return value is the path to the cartridge directory as specified in the control window.

Note that certain operations on cartridges (such as saving cartridge-specific information in the icon) use the filename you specify here. If you change this name and then save the icon you will save the icon to the wrong location.

Also note that after a reset fMSX will set several controls based on information found in the icon of the cartridge, if any, overriding any changes you may have made before. Currently the following information can be found in the cartridge icons: refresh cycle, interrupt period, ROM type, MSX version, and soundmode. This may change in the future.

The ’1’ at the end of the commandname means that the cartridge will be loaded into MSX-slot 1. In the future support may (once again) be added for using two cartridges at the same time. A second cartridge will have to be loaded into slot 2.

1.18 "

```
        LockDrives

    Purpose: to lock or unlock the drives used by fMSX.

     Format: [ ON | OFF ]

  Available: 1.4
```

Description: this command locks or unlocks the drives used by fMSX. When locked the drives can be used by the emulated MSX, but not by the Amiga. When unlocked the situation is reversed: the MSX cannot access the drives but the Amiga can.

From the MSX point of view unlocking is equivalent to removing the disk from the drive. As might be expected, you should not do this while the disk is being accessed.

1.19 "

```
        Memory

    Purpose: to change the amount of memory used by the MSX.

     Format: [ 64KB | 128KB | 256KB | 512KB | 1MB | 2MB | 4MB ]

  Available: 1.4
```

Description: this command changes the amount of memory used by the MSX. An MSX can use up to 4MB per slot, and fMSX supports RAM in one slot (instead of the theoretical maximum of 14 slots!).

Most programs will feel perfectly fine when using 256KB of RAM. Only a handful need 512KB, and I cannot name any that need more than that (apart from RAM disks and the like).

Note that this setting takes effect when you reset the MSX.

1.20 "

```
        MSX1ROM, MSX2ROM, MSX2SubROM

    Purpose: to change the path to the one of the MSX ROMs.

     Format: [ FILENAME ]

  Available: 1.4
```

Description: this command sets the path to one of the MSX ROMs. You may want to use your own ROMs with fMSX instead of the ones that are built in. If you want to do that you will need to specify the correct paths.

The MSX1 system only needs one ROM, which contains both BIOS and BASIC. However, the MSX2 system needs a second ROM called the sub ROM, which contains the extra BASIC commando's supported by the MSX2. The BIOS/BASIC ROM is 32KB for both MSX1 and MSX2, and the sub ROM is 16KB.

Note that this setting takes effect when you reset the MSX.

1.21 "

```
        MSXFont

    Purpose: to change the path to the externally loaded MSX font.

     Format: [ FILENAME ]

  Available: 1.4
```

Description: this command tells fMSX what external MSX font you

want to use, if any. This can be used to change to a Japanese or Cyrillic font (which can be useful), or simply to a 'cool' font such as the italic font.

Externally loaded MSX fonts are not compatible with Amiga bitmap fonts. If you try to load an external Amiga bitmap font your MSX screen will be unreadable.

Several fonts can be found in the MSXFonts directory in the fMSX distribution.

Note that this setting takes effect when you reset the MSX.

1.22 "

MSXVersion

Purpose: to change the MSX ROMs used by fMSX.

Format: [MSX1 | MSX2]

Available: 1.4

Description: this command changes the ROMs used by fMSX. Although all MSX1 programs work on MSX2 as well you may prefer to use MSX1 ROMs when possible because the MSX1 bootsequence is much faster than the one for MSX2.

This command only affects the ROMs used by fMSX. Hardware-wise fMSX always emulates an MSX2 machine.

Note that this setting takes effect when you reset the MSX.

1.23 "

Mode

Purpose: to run or pause fMSX, or put it in music mode.

Format: [RUN | PAUSE | MUSIC]

Available: 1.4

Description: this command sets the mode of fMSX. fMSX has three modes of operation:

Mode RUN

fMSX is executing an MSX program. It will take 100% CPU time, split between the MSX emulation and redrawing the screen. Because the MSX emulation is running at priority -1 it has no impact on other programs running on the Amiga.

```
    Mode PAUSE

fMSX is suspended, and takes no CPU time whatsoever.

    Mode MUSIC

fMSX is running but does not redraw the screen. This frees up more CPU
cycles for the actual MSX emulation which runs faster as a result. Although
you cannot see anything you can hear whatever sounds it may generate, which
is great for running MSX music demo's.
```

1.24 "

```
                        PlugAndPlay

    Purpose: to change the way fMSX reacts to cartridge insertions.

     Format: [ ON | OFF ]

  Available: 1.4

Description: this command changes the way fMSX reacts to the
insertion of cartridges. When plug'n'play is turned on fMSX will reboot
after a cartridge is inserted (when it is turned off nothing will happen).

Note that only fMSX only reboots after interactive cartridge insertions (ie.
cartridges insert through GUI-commands). If you want to reboot after
inserting a cartridge through ARexx you will have to call
                Reset
                .
```

1.25 "

```
        Quit

    Purpose: to quit fMSX.

     Format: -

  Available: 1.4

Description: this command quits fMSX.
```

1.26 "

```
        RefreshCycle

    Purpose: to change the refresh cycle.

     Format: [ VALUE ]
```

```
  Available: 1.4

Description: this command sets the refresh cycle used by fMSX. The
refresh cycle is the number of frames skipped by fMSX before a new frame is
drawn. Redrawing a frame takes comparatively long, so skipping a few frames
means more CPU cycles will be available for the actual emulation. On the
other hand, when few frames are skipped the action seems more fluid.

Acceptable values for the interrupt period lie between 1 and 10.
```

1.27 "

```
        RefreshWhenActive

    Purpose: to change screenrefreshing when it is inactive.

     Format: [ ON | OFF ]

  Available: 1.4

Description: With this command uou can instruct fMSX not to refresh
the MSX screen when it is inactive. This frees up processor power for other
applications or the main emulation while you are working in another window.
```

1.28 "

```
        Reset

    Purpose: to reset the emulated MSX.

     Format: -

  Available: 1.4

Description: this command resets the MSX machine emulated by fMSX.
You need to reset when you want to leave the current game, or if you want
to (de-)activate certain features.
```

1.29 "

```
        RespectBlankBit

    Purpose: to change the way fMSX deals with screen blanking.

     Format: [ ON | OFF ]

  Available: 1.4

Description: Due to the design of fMSX it may be possible that in
```

some cases MSX screen blanking and fMSX screen redrawing are badly
synchronized. If this happens you may want to turn MSX screen blanking off.

If you turn blanking off you may see trash and redrawing going on that would
otherwise be hidden.

1.30 "

ROMType1

Purpose: to set the ROM type for the currently loaded cartridge.

Format: [SCC1 | SCC2 | KONAMI1 | KONAMI2 | ASCII1 | ASCII2]

Available: 1.4

Description: this command sets the ROM type for the currently
loaded cartridge. Setting the ROM type only affects megaROMs (cartridges
that are bigger than 32KB). 16KB and 32KB ROMs do not have a ROM type.

It is necessary to set the correct ROM type for a megaROM to function. If
you do not know the ROM type for a particular megaROM you have no choice but
to try them all. However, a list of ROM types is available here.

The ’1’ at the end of the commandname means that the rom type will be set
for the cartridge in MSX-slot 1. In the future support may (once again) be
added for using two cartridges at the same time. A command called ROMType2
will then be added as well.

1.31 "

SCCDisable

Purpose: to change SCC channel disabling.

Format: [ON | OFF]

Available: 1.4

Description: with this command you can select whether fMSX ignores
or respects the SCC disable settings. In most games, ignoring those settings
gives the best soundquality, but in some games (such as Parodius) the
soundquality is much better when this feature is turned on.

You’ll have to experiment to find the best value.

This command only affects the SCC and SCC/PSG soundmodes.

1.32 "

Version

Purpose: to change SCC+ emulation.

Format: [ON | OFF]

Available: 1.4

Description: this command enables or disables SCC+ emulation. SCC+ is a soundchip used in some Konami games, and is an improved version of the SCC soundchip. When SCC+ emulation is enabled, (normal) SCC emulation is disabled and vice versa.

Enabling SCC+ only makes sense when you are using the SCC or PSG/SCC soundmodes.

As far as I know, SCC+ is only used in The Snatcher and S.D. Snatcher.

1.33 "

SoundMode

Purpose: to change the way fMSX emulates sound.

Format: [OFF | PSG | SCC | PSGSCC]

Available: 1.4

Description: this command changes the way fMSX emulates sound. Currently four methods are provided:

Off

fMSX is not playing any sound, either because that is what the user wanted or because it could not allocate the sound hardware.

PSG

fMSX is emulating the native MSX soundchip, called the Programmable Sound Generator or PSG. Although it allocates the sound hardware before use it hits the hardware for producing sound. This mode is in many ways the best fMSX has to offer: it is fast, sounds exactly like a real MSX, and has support for digitized samples (through volume modulation of the noise channel).

SCC

fMSX is emulating a soundchip found in some Konami games. It produces much better sound than the PSG, but offers some features that cannot easily be emulated on the Amiga (such as five-channel sound).

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
                              S.D. Snatcher

    PSGSCC
```

fMSX is emulating both the PSG and SCC soundchips. This is done with the help of AHI, a system for playing up to 128 channels of sound using nothing more than the standard Amiga audio hardware and a fast processor.

In this mode fMSX runs a lot slower because AHI is mixing samples in the background. The soundquality is not quite as good as with the normal PSG or SCC modes, but at least you can listen to the full musical scores which tend to be rather spectacular for SCC games.

1.34 "

```
        Version

    Purpose: to obtain the version number of fMSX.

     Format: [ MINOR | MAJOR ]

  Available: 1.4
```

Description: this command returns the version number of fMSX.
You can use the version number to decide whether fMSX will support certain operations.

When used with the MINOR (or MAJOR) argument, only the minor (or major) version number is returned.

When used without arguments, both major and minor version numbers will be returned, seperated by a dot.