OPX Sound Cartridge

NEOTRON™

User Manual

preliminary release
version 0.1 alpha

relevant to
NEOTRON

Presented by Jun Soft & SuperSoniqs

2018-2021

Table of Contents

Index of Tables

Illustration Index

1 Introduction

Thank you for purchasing and using Jun Soft's H/W. NEOTRON™ is a sound cartridge for MSX computer systems. It generates mix-up sound from SSG, FM or/and PCM sources depending on the installed sound chip. The major purpose of NEOTRON™ is to play back dedicated music files or to generate BGM/sound effects.

2 System Requirement

2.1 Minimum requirement

- MSX computer with a disk drive.
- MSX-DOS.

2.2 Recommended requirement

- MSX2 computer with 128MB system memory.
- A mass storage device (HDD/MMC/SD etc.)
- MSX-DOS2.
- audio amplifier or headphone.

2.3 Features (1N0)

- OPNB sound chip (YM2610).
- Sample ROM(flash memory) up to 8MBx2.
- Sample ROM(flash memory) host interface.
- Audio DAC and stereo head-phone driver.
- Flash memory 128KB minimum.
- Audio routed into MSX SNDIN with volume control (optional).

3 Quick Installation

3.1 Setup hardware

- Power off your MSX computer, if it's powered up.
- Put a NEOTRON™ cartridge into your MSX computer's empty slot.
- Connect an audio cable to NEOTRON™ and audio equipment (audio amplifier or headphone).
- Power on the MSX computer and boot with MSX-DOS(2).

3.2 Install software

NEOTRON™ cartridge has a program memory (Flash memory) to install dedicate BIOS. You can upload BIOS by a FlashPack™ memory tool from Jun Soft. You can download the BIOS and the tool from Jun Soft's blog. If you don't want to play with the BIOS, you can skip this step.

- Boot with MSX-DOS(2).
- Check if the BIOS image file exists in the current directory. You must also have "FPFM.COM" from Jun Soft in executable path.
- For example, if your NEOTRON™ is inserted in main slot 1, type "fpmt w <image_file> /tf/s1"
- Reboot your system (hardware reset is recommended).

3.3 ADPCM data management

NEOTRON™ cartridge has two flash memories for ADPCM data. You can write data into the memory or read out the content of the memory by the FlashPack™ memory tool. Generally, this is done by application software. But if you have any reason to write ADPCM data directly into the memory, try these.

- Boot with MSX-DOS(2).
- Check if the ADPCM data file exists in the current directory. You must also have "FPFM.COM" from Jun Soft.
- For example, if your NEOTRON™ is inserted in main slot 1, type "fpmt load <data_file> /tf/s1".
- if you want to load data into another ADPCM memory, type "fpmt load <data_file> /tf/s1/o80". This is for ADPCM-B of OPNB.

3.4 Connection diagram

There are one audio output connector(phone jack) in NEOTRON™ cartridge. You can connect a headphone or an external amplifier to the output connector. Typical configuration is shown in Figure 1. If you don't put any plug into the jack, sound is routed into a MSX slot and mixed up with MSX's internal sound.

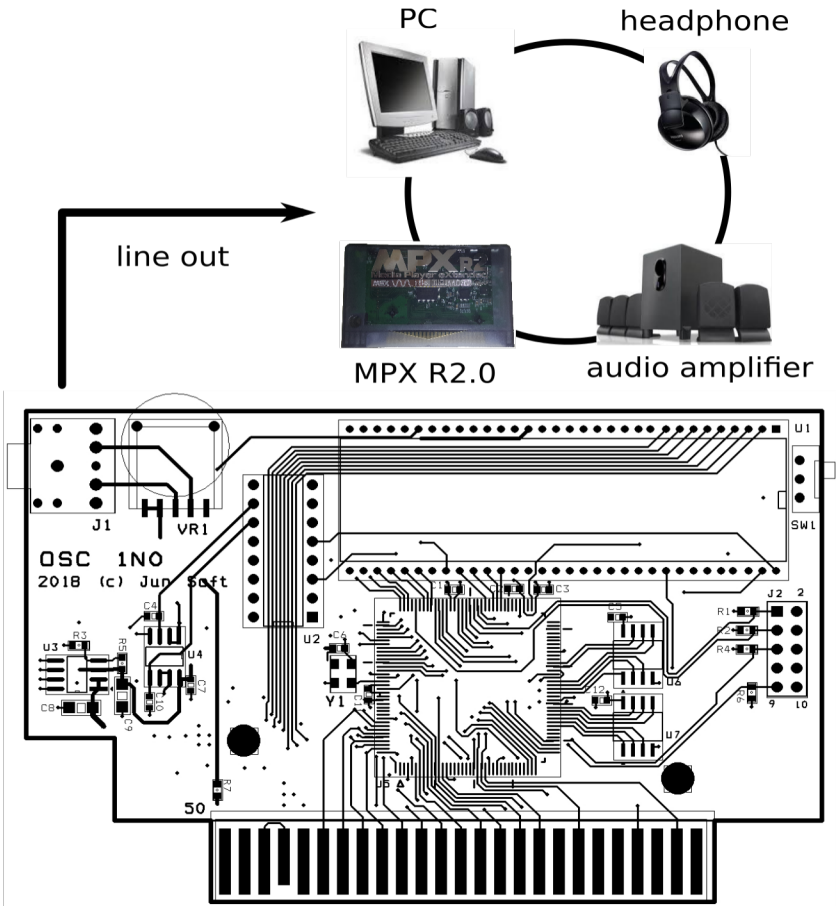

Figure 1: connection to audio devices

3.5 Checking hardware

You need a proper software to test NEOTRON™. Refer to section 6 Tools.

4 NEOTRON™ Internal

4.1 Overview

NEOTRON™ is a sound cartridge based on the sound chip from Yamaha which supports 3 SSG, 4 FM and 7 PCM voices simultaneously if YM2610 is installed, for example. The output digital audio data from YM2610 is converted to analog stereo signal by an audio DAC. By the sake of the headphone amplifier, you can hear the music from NEOTRON™ through a headphone without any external amplifier. Of course, you can connect NEOTRON™ to your beloved sound system.

NEOTRON™ has a flash memory for internal usage or/and user program interface. The flash memory is divided into several areas called "segment", the size of the segment is 16KB. Each segment can be dynamically mapped to any of two banks, a logical address space. More details are found in following sections. Also some memory address spaces are assigned to control the sound chip and other features.

4.2 Board layout

Figure 2 shows the board layout of NEOTRON™. There are one switch and one audio output connector(phone jack). You can connect another audio equipment supporting line level input using the connector. Or you can directly connect a headphone.

The switch is installed to disable flash memory access. This is used to prevent wrong software running.

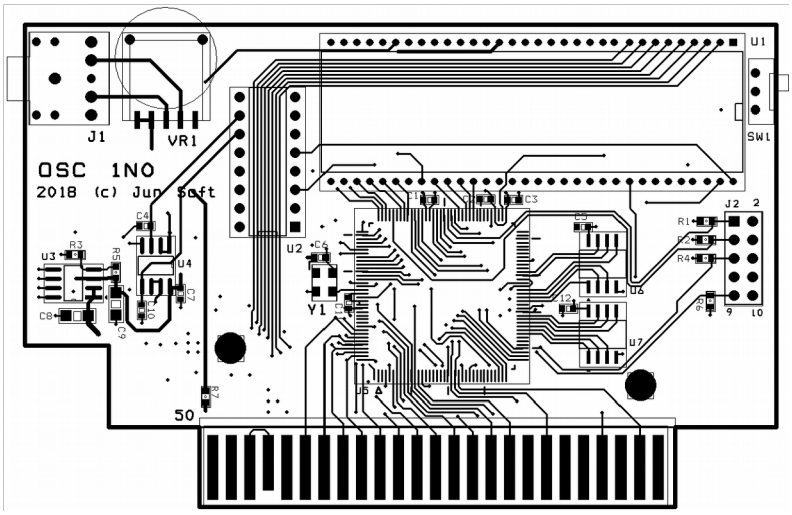

Figure 2: board layout

4.3 Memory map

NEOTRON™ uses memory mapped I/O to control the OPx sound chip. Some addresses are used to access FM registers, some are used for SSG and some are used for ADPCM registers. Refer to a OPx manual for detail operations. Table 1 shows all memory addresses used in NEOTRON™. Since NEOTRON™ uses memory mapped I/O, that means you can control NEOTRON™ through predefined memory addresses, some addresses are assigned to special purpose registers. Those registers are normally invisible. You must set bit 7 of REG_BANK1 for accessing special registers through bank 1, 7C00H~7FFFH address range. Also you can set bit 7 of REG_BANK2 for the same purpose but through bank 2.

address	name	type	description
4000H-7FFFH	Bank 1	r	Flash memory (bank 1).
8000H-BFFFH	Bank 2	r	Flash memory (bank 2).
7000H-77FFH	REG_BANK1	w	Bit 2..0: Memory mapper for bank 1, default 0. Bit 7: enable special registers over 7C00H~7FFFH
B000H-B7FFH	REG_BANK2	w	Bit 2..0: Memory mapper for bank 2, default 1. Bit 7: enable special registers over BC00H~BFFFH

Table 1: memory map

address	name	type	description
7C00H	REG_YM_P0	r/w	SSG/ADPCM-B/FM/FM1,2 (register index)
7C01H	REG_YM_P1	r/w	SSG/ADPCM-B/FM/FM1,2 (data)
7C02H	REG_YM_P2	r/w	ADPCM-A/FM3,4 (register index)
7C03H	REG_YM_P3	r/w	ADPCM-A/FM3,4 (data)
7DxxH	REG_MEMA	r/w	Flash memory access for sample data A
7ExxH	REG_MEMB	r/w	Flash memory access for sample data B
7FxxH	REG_CFG	w	Configuration register

Table 2: special registers (bank 1)

address	name	type	description
BC00H	REG_YM_P0	r/w	SSG/ADPCM-B/FM/FM1,2 (register index)
BC01H	REG_YM_P1	w	SSG/ADPCM-B/FM/FM1,2 (data)

BC02H	REG_YM_P2	r/w	ADPCM-A/FM3,4 (register index)
BC03H	REG_YM_P3	w	ADPCM-A/FM3,4 (data)
BDxxH	REG_MEMA	r/w	Flash memory access for sample data A
BExxH	REG_MEMB	r/w	Flash memory access for sample data B
BFxxH	REG_CFG	w	Configuration register

Table 3: special registers (bank 2)

4.3.1 Memory mapper

NEOTRON™ has two memory mappers for mapping physical flash memory segments to a bank, a logical space. Each segment's size is 16KB and maximum 8 segments are available (it depends on installed memory of your board). You can assign a segment number to a bank via a 3-bit memory mapper register (REG_BANK[1..2]). Note that some part of the bank 1 or 2 is occupied by special registers, those registers are visible when write '1' to bit 7 of REG_BANK[1..2] and used for accessing YM2610, sample memories or the configuration register. The sample memories are shared by YM2610 and a host (MSX) but not simultaneously. The connection to the memories is controlled by writing '1' into bit 4 of REG_CFG .

The memory banking registers are summarized in Table 4.

bit	name	description
7	EN_SR	'1' enables special registers
6..3	reserved	Must write 0.
2..0	Segment	Segment number (0~31)

Table 4: memory bank selection register (REG_BANK1/2)

4.3.2 Special registers

NEOTRON™ has special registers to control a sound chip, sample memories and system configurations.
Refer to Table 2 and 3 for overall memory layout. Refer to the sound chip manual for accessing special registers of OPNB. The system configuration register, REG_CFG is shown in Table 5. To

access the sample memories, set CFG_FWE. This disconnect memory link to YM2610. In general, don’t access special registers directly and use APIs of SIOS(5 NEOTRONTM BIOS) instead.

bit	name	description
7	reserved	Must write 0.
6	reserved	Must write 0.
5	reserved	Must write 0.
4	CFG_FWE	Flash memory for sample data write enable (enable host control)
3	reserved	Must write 0.
2	reserved	Must write 0.
1	CFG_MBE	Sample memory selection B
0	CFG_MAE	Sample memory selection A

Table 5: system configuration register (REG_CFG)

5 NEOTRON™ BIOS

You can write any software and put it into the flash memory of NEOTRON™. Or use "SIOS(Sound I/O System)" from Jun Soft if necessary. At this moment the SIOS is under development. Some useful entries are show in Table 6. Note that same entries exist in bank 2 with different addresses. You can find the entry point just by adding 4000H. Another addresses helpful for applications are listed in Table 7. Note that APIs and related entry points can be changed without notification.

5.1 API details

5.1.1 SIOS_INIT

in	
out	
reg	
description	Do not use this call

5.1.2 SIOS_RESET

in	none
out	none
reg	AF
description	Use this API to enter normal state. You must call this after memory management APIs. Otherwise, OPNB can's access sample memories.

5.1.3 SIOS_ENABLE_IO

in	Cy
out	none
reg	none
description	If Cy = 0, enable special registers. If Cy = 1, disable special registers. Use this API prior before any other APIs.

5.1.4 SIOS_INFO_SMEM

in	A = memory ID (1 or 2)

out	A = capacity in MB B = memory ID C = total number of memories
reg	AF, BC, IX
description	Get the amount of memory attached. Use ID=1 for ADPCM-A memory and ID=2 for ADPCM-B memory.

5.1.5 SIOS_ERASE_SMEM

in	A = ID, B = the number of blocks to erase
out	Cy = 1 when error occurs.
reg	AF, BC, DE, HL, IX
description	IF B = 0, erase all. IF B > 0, erase B * 64KB. In this case, the start address must be given by SIOS_SET_SMEM API. This takes up to several minutes, in the worst case.

5.1.6 SIOS_WRITE_SMEM

in	A = ID, BC = length (only B is effective), HL = source address
out	Cy = 1 when error occurs.
reg	AF, BC, DE, HL, IX
description	Write data into sample memory. Note that C is ignored. To prevent compatibility issue, set C as zero.

5.1.7 SIOS_READ_SMEM

in	A = ID, BC = length, HL = destination address
out	Cy = 1 when error occurs.
reg	AF, BC, DE, HL, IX
description	Read out data from the sample memory.

5.1.8 SIOS_SET_SMEM

in	A = ID, DE = address[23..8]
out	None (internal variables updated)
reg	AF, IX
description	Setup memory address to access. You need to call this before erase/read/write (from/to) sample memories. Note that the address is 256-byte base.

5.1.9 SIOS_WRITE_REG0

in	A = register index, D = register data

out	None
reg	A
description	This writes data to the OPNB register of bank 1

5.1.10 SIOS_WRITE_REG1

in	A = register index, D = register data
out	None
reg	A
description	This writes data to the OPNB register of bank 2.

5.1.11 SIOS_READ_REG0

in	A = register index
out	A = register data
reg	A
description	This reads data from the OPNB register of bank 1.

5.1.12 SIOS_READ_REG1

in	A = register index
out	A = register data
reg	A
description	This reads data from the OPNB register of bank 2. Note that this call has no effect. Do not use.

5.1.13 SIOS_READ_STAT0

in	None
out	A = status of port 0
reg	A
description	Read OPNB status register 0.

5.1.14 SIOS_READ_STAT1

in	A = ID, DE = address[23..8]
out	None (internal variables updated)
reg	AF, IX
description	Read OPNB status register 1.

5.2 Examples of API usage

5.2.1 Erase and write sample memory

```
or a                            ; reset carry flag
call SIOS_ENABLE_IO             ; enable register access
ld a, 1                         ; memory ID = 1
ld de, 0x1000                   ; address to access = 0x1000
call SIOS_SET_SMEM              ; set access address
ld a, 1                         ; memory ID = 1
ld b, 0x10                      ; number of blocks to erase = 16
call SIOS_ERASE_SMEM            ; erase memory
ld a, 1                         ; memory ID = 1
ld de, 0x1000                   ; address to access = 0x1000
call SIOS_SET_SMEM              ; set access address
ld a, 1                         ; memory ID = 1
ld bc, 0x1000                   ; number of bytes to write = 0x1000
ld hl, 0x8000                   ; source address = 0x8000
call SIOS_WRITE_SMEM            ; write data
call SIOS_RESET                 ; back to the normal state
```

5.2.2 Read sample memory

```
or a                            ; reset carry flag
call SIOS_ENABLE_IO             ; enable register access
ld a, 1                         ; memory ID = 1
ld de, 0x1000                   ; address to access = 0x1000
call SIOS_SET_SMEM              ; set access address
ld a, 1                         ; memory ID = 1
ld bc, 0x1000                   ; number of bytes to read = 0x1000
ld hl, 0x8000                   ; source address = 0x8000
call SIOS_READ_SMEM             ; read data
```

```
call SIOS_RESET                 ; back to the normal state
```

5.3 Summary of API list

5.3.1 API entry points

address	name	description
4010H	reserved	Internal usage. Do not use.
4013H	reserved	Internal usage. Do not use.
4016H	sios_init	Do not use.
4019H	sios_reset	Set up NEOTRON™ for normal operation
401CH	sios_enable_io	Make special registers visible or not
401FH	sios_info_smem	Get sample memory information
4022H	sios_erase_smem	Erase sample memory contents
4025H	sios_write_smem	Write data into sample memory
4028H	sios_read_smem	Read data from sample memory
402BH	sios_set_smem	Setup memory address for following access
40C0H	sios_write_reg0	Write register (port 0)
40C3H	sios_write_reg1	Write register (port 1)
40D0H	sios_read_reg0	Read register (port 0)
40D3H	sios_read_reg1	Read register (port 1)
40E0H	sios_read_stat0	Read status register (port 0)
40E3H	sios_read_stat1	Read status register (port 1)

Table 6: SIOS API list

5.3.2 Summary of Hardware Dependent Constants

address	name	Description (example)
4060H	ID_APP	Application ID (“SIOS”)
4064H	ID_VER	Version (“V1.0”)
4068H	ID_MAN	Manufacturer (“JSFX”)

406CH	ID_DEV	Device (“NEOTRON ”)
4070H	ID_CHIP_VEND	Chip vendor (“YM “)
4074H	ID_CHIP_TYPE	Chip type (“OPNB”)
4078H	ID_CHIP_NAME	Chip model name (“YM2610 “)
40F8H	port_opn	Base address for chip access. Not recommended to access the chip via this address. Use APIs.
40FAH	port_smema	Sample memory A control address. Do not use.
40FCH	port_smemb	Sample memory B control address. Do not use.
40FEH	port_ctrl	System configuration address. Do not use.

Table 7: SIOS constant variables

6 Tools

There are many programs working with NEOTRON™. A lot of music re-players, editors and games are widely released or will be released. Find them and enjoy the music. Another important tool is “fpmt.com”. It is necessary to write software into the flash memory. Download it from Jun Soft's blog and refer to output messages of the program with 'h' option. Some examples are following.

To update BIOS

Off the switch of NEOTRON™ if needed. When your NEOTRON™ is inserted in the slot 2-1:

A:> fpmt write sios.rom /s21/tf

To write ADPCM-A data

When your NEOTRON™ is inserted in the slot 2-1:

A:> fpmt load <file> /s21/tf

To write ADPCM-B data

When your NEOTRON™ is inserted in the slot 2-1:

A:> fpmt load sios.rom /s21/tf/o80

7 Acknowledgement

Thank to all MSX users in the world,
MSX is still alive and kicking!
Enjoy your MSX forever!

