SUPER-X version 1.22 documentation

Unofficial SUPER-X version 1.22 Made By: NYYRIKKI (2011)

Forewords:

SUPER-X is the best monitor & debugger in MSX history. Tremendous power is shown in developing, analysing and debugging of a machine language program. This fan version is expanded from the official version. I've been trying to expand the debugging possibilities while still keeping the original look & feeling. Some clear bugs have been removed and some commands have been changed. Unfortunately I have not been able to contact ***Romi*** who is the original author of this program.

I want to thank ***JP Grobler*** for making the English translation from original Japanese documentation. This up to date documentation would not exists without his previous efforts. The original Japanese documentation does not anymore ship with Super-X as I have no way to keep it updated as I totally lack Japanese skills.

Development of Super-X started 20 years ago, so this version can now be seen as kind of birthday version!

NYYRIKKI – April 2011

List of files

At least these seven files should be delivered with the program package:

"SUPER-X.BAT "	Batch file for DOS.
"SUPER-X.BAS "	Basic file that bloads "SUPER-X.LDR"
"SUPER-X.LDR"	RAW Loader – binary file
"SUPER-X.BDY"	SUPER-X itself.
"SUPER-X.TNK"	Note File (description later). If it is on the disk, it is loaded to the VRAM
"SUPER-X.FNT"	Japanese character font file. (Optional)
"SUPER-X.DOC"	This file

Loading SUPER-X

From BASIC type: RUN "SUPER-X.BAS"
From DOS type: SUPER-X

You can skip TNK and FNT file loading by holding GRAPH down.

Added commands to BASIC:

- **CALL @**
  Start Super-X

- **CALL S**
  Can now be used instead of CALL SYSTEM

- **CALL CHCPU (N):**
  Change CPU (MSX tR only)

  N = 0 Z80 (ROM)
  N = 1 R800 (ROM)
  N = 2 R800 (DRAM)
  N = 3 Z80 (DRAM)

Basic knowledge

In this documentation we use the following formatting:
< … > required parameter
[…] optional parameter
{ … } keyboard button

Before we start this is the basic knowledge you should have:

- When SUPER-X is started a summary of the machine's info will be displayed and a command prompt appears ('A:' or 'B:' etc).
- Commands are entered after the prompt.
- The prompt indicates the current active drive (like DOS)
- All commands are case sensitive
- Pressing **{GRAPH}** + **{SPACE}** opens the help window.
- During command execution if you push **{SPACE}** or **{ENTER}** execution halts.
  If pressed for a second time execution continues
- Pressing **{ESC}** or **{BackSpace}** (or **{CTRL}** + **{STOP}** during input) ends execution of a command.
- Command parameters can be separated by a blank space ' ' or a comma ','
  A space between the command and the first parameter is also allowed, for easy reading / input:

  Example: A>D 0000 1000 or A>D0000,1000

- Command functions can differ depending upon the number of parameters:
  One Address display block for editing
  Two Addresses give a non stop list output (used mostly for printer output)

Command line, search and assembly input modes use full screen editor that works like built in MSX-BASIC editor. You can move around with cursors and use same keyboard shortcuts including function keys. Here is list of some of the most important BASIC & Super-X common keyboard shortcuts in these modes:

Interactive mode keyboard shortcuts:

{HOME}	Move cursor to upper left corner
{SHIFT} + {HOME}	Clear screen and move cursor to upper left corner
{CTRL} + {N}	Move cursor to end of line
{CTRL} + {E}	Clear end of the line starting from cursor
{CTRL} + {U}	Clear whole line and move cursor to beginning of line
{CTRL} + {F}	Move cursor to start of next word
{CTRL} + {B}	Move cursor to start of previous word

Other basic commands have their own keyboard shortcuts. The details / exceptions have been explained under each command, but generally the logic goes like described in table below.

General keyboard shortcuts:

{cursor keys}	Cursor movement
{BackSpace}	Cursor one character back
{SHIFT} + {SPACE}	Information of the address at the cursor
{HOME}	Moves cursor to top left
{SHIFT} + {HOME}	Changes the display address to the current cursor position
{INS}	One screen down
{DEL}	One screen up
{\} or {TAB}	Input displayed address
{ENTER}	Command menu mode.
{CTRL} + {S}	Search for data. The command waits for the search string. *See **FD** command*
{CTRL} + {F}	Repeat search forward (The string entered above is searched)
{CTRL} + {D}	Repeat search backward
{CTRL} + {Z}	Reset global offset to zero (Pressing again will undo the reset)
{ESC}	Exits editing

General information about SUPER-X

- **Address with slot assignment**
  Format: [address] # [primary slot number] - [extended slot number]
  Example: Address 1234h in primary slot 3: 1234#3
  Example: Address 1234h in slot 3-1: 1234#3-1
  Example: Address 1234h in slot 1-3: 1234#1-3

  The normal (boot up) slot state can be chosen with #S or #5
  The VRAM slot can be chosen with #V or #4

- **The current slot**
  If the slot number is left out of an address the current slot is assumed.
  The current slot is the slot last selected via a basic command (Dump, ASCII, dIsassembler and asseMbly input)
  The basic commands change the active slot if specified in the address. You can see currently active slot by looking information area.

- **The VRAM slot**
  SUPER-X can access the VRAM same way as normal ram. Select the VRAM with slot #V or #4.
  Because the Japanese font is in the VRAM it might be displayed wrong if the VRAM is accessed.
  Also the Note File is located in the VRAM so proceed with caution if you need these features.

- **Numerical values**
  SUPER-X input address and such are generally hexadecimal numbers, but other values are input as "numerical values".

  SUPER-X can accept binary, decimal and hexadecimal numbers (See “assembly input” in “basic commands”), negative numbers, formulas (calculation range 2 bytes) and can handle overflow.
  ASCII codes can be used as numbers by surrounding max 2 letters with “ ′ “

  Please note: There is no precedence in operators. They are calculated from left to right!
  Available math operations are:

+	add
-	subtract
*	multiply
/	divide
%	modulus (like MOD in BASIC)
|	logical OR
&	logical AND
^	logical XOR

  Examples:
  CL 10-1|80H
  CL ‘A’&15

- **The information area**
  At the bottom of the screen there is the information area. It holds following information:

  [<current command> : <address of cursor> : <debug stack count>] SLOT: <current slot> 0=<global offset>

- **Screen print / Screen shot**
  {CTRL} + {P} contents of the screen is printed
  {CTRL} + {V} contents of the screen is copied to VRAM

  To view captured screenshot from VRAM, press **{SELECT}**

- **Commas in inputs can be replaced with spaces**
  Example:. SD0,1000,FILENAME or
  SD0 1000 FILENAME

- **Printer output of a command**
  Place a "?" in front of the command
  Example: ?D0 100 This dumps address 0 to 100h to the screen and printer

- **Debugger variables**
  7 Addresses can be saved in variables (address + slot number)
  An address in a command, can be substituted by these variables **@[variable number]**
  Normal variables are numbered from 0 to 3. Commands store base address in variable 0
  There are also 3 special variables that are set automatically by disk load functions:
  @B = Begin address
  @E = End address
  @S = Start address / Size

  Do display content of variables, you can type "@" in command prompt
  To set variable, type:
  @<variable name/number>=<address>[#<slot>]

Example:	@0=1000#3-1	sets @0 to 1000H in slot 3-1
	D@0	will dump 1000H in slot 3-1
	@1=FF2	sets @1 to 0FF2H
	CL @1+1	will display 0FF3H in various number formats

- **Loaded filename**
  When a file is loaded **LD** of **L#** commands, file name will be copied to Function key 1

- **Repeat command function**
  If **{ENTER}** is pressed on a blank command line, the previous command is echoed and can be executed or changed.

- **Change the current drive**
  Same as DOS: Type A:or B: (up to H:)

- **Note function**
  Notes can be added to addresses. The Note File is "SUPER-X.TNK"
  The **Note File** is located from 8000H - FFFFH in the VRAM
  The number of possible notes is 512.
  About 470 notes is already in the file supplied with SUPER-X. It contains the main BIOS, the main work area, the main hook and the main I / the O port et cetera (Currently only in Japanese)
  When contents of the VRAM are changed, the **Note File** is destroyed, please reload it with the **iL** command.

  The format of the Note File:

	total number of notes left	first 2 bytes
+ 512 *		
	address	2 bytes
	slot data	1 byte
	type data	1 byte
	explanatory data	60 bytes
= 32770 bytes (32kB)		

- **Debug stack (Jump / call stack)**
  When in disassembly mode a call / jump can be followed by pressing {CURSOR RIGHT}.
  {CURSOR LEFT} returns to the origin of the jump / call.
  Up to 16 jumps / calls can be stored (the slot, address and cursor position are stored)

  For example:
  1. During disassembling editing, say there was a line, "CALL 0134h".
  2. Then when the cursor is on the line and **{CURSOR RIGHT}** is pressed
     "P>> [0134 # _-_]" will appear and the base address of disassembling editing changes to 134h when {ENTER} is pushed
  3. You can keep on analysing as normal.
  4. When **{CURSOR LEFT}** is pressed SUPER-X returns to the address which called 0134h in the first place

Basic commands

There exists 5 different basic editor modes that you can interact with.
When you are in one of these editor modes, you can quickly change beween them:

Press {ENTER} to open the basic command window, then use the {CURSORS} or {SPACE} to choose the command.

```
      ------            {CURSOR UP} switch to Dump mode
------|Dump|------      {CURSOR LEFT} switch to Ascii mode
|Ascii Char Multi|      {SPACE} switch to Character (font / sprite / binary )mode
-----|Disasm|-----      {CURSOR RIGHT} switch to asseMbly input) mode
     --------           {CURSOR DOWN} switch to dIsassembly mode
```

From command line these modes can be accessed with following commands:

- **HexDump editing / Listing**
  D <begin address>[#<slot>][,<end address>[,<file name>]]

  Gives a hex dump of the memory
  All general keyboard shortcuts are available. Also following keys have special meaning on this mode:

{0} – {F}	Hexadecimal data entry
{"}	ASCII entry
{SHIFT} + {cursor UP / DOWN}	Scrolls the page up / down
{@}	Repeat previous byte

- **ASCII editing / Listing**
  A <begin address>[#<slot>][,<end address>[,<file name>]]
  Gives an ASCII listing of the memory

  General keyboard shortcuts are available. Most of the keys function as data input on this mode.

  Example:. To Edit #4000 in slot 3-2
  A4000#3-2

- **Disassembly editing / Listing**
  I <begin address>[#<slot>][,<end address>[,<file name>]]
  Gives a disassembles output of the memory

  General keyboard shortcuts are available. Also following keys have special meaning on this mode:

{DEL}	32 bytes back
{SHIFT} + {SPACE}	Information of the command address parameter pointed by cursor
{UP}	Move cursor up (move to previous command or byte)
{DOWN}	Move cursor down (move to next command)
{RIGHT}	Follows a jump / call (push to debug stack)
{LEFT}	Returns to previous call / jump address (pop from debug stack)
{CTRL} + {O}	Set global offset (relative to current offset)

  Example:. To disassemble address #4000 in slot 3-2
  I4000#3-2

Character editing / Listing
H <begin address> [#<slot>][,<end address>[,<file name>]]
Displays the memory as compilation of sprite and the character font. (Binary editing)
The 2 bytes after the cursors is the Kanji character code
A graphic window of a 16x16 Font is displayed
General keyboard shortcuts are available. Also following keys have special meaning on this mode:

{SPACE}	Toggles bit on or off
{SHIFT} + {SPACE}	Information on the address at the cursor
{U}	The data at the cursor and above is exchanged.
{D}	The data at the cursor and below is exchanged
{R}	The data at the cursor is moved to the right. If the X position of the cursor is 7 or 8 the whole line is moved!
{L}	The data at the cursor is moved to the left. If the X position of the cursor is 7 or 8 the whole line is moved!
{V}	The data at the cursor is reversed
{O}	All dots at the cursor are turned on
{F}	All dots at the cursor are turned off
{C}	Clears the whole data range

- **Assembly input**
M <begin address>[#<slot>]
Enters interactive mode, displays address and waits for input which is written to memory. Address counter is automatically increased after successful write to memory. By default the input is assembler (R800 commands are also accepted).

When you are in interactive mode you can also type: **I** [<number of lines>]
This lists the following <number of lines> instructions – like basic list command
With cursor keys you can now go up and change commands or addresses. If <number of lines> is empty then 20 lines is displayed. All default editing keys are available in this mode.

You can also use this command mode to edit memory directly like this:
<address>[#<slot>] [<type sign>[<data 1>, <data 2>, ...,<data n>]]
Type sign is as follows:

.	1 byte hexadecimal	example: 0000 .CD, 4d, 00
:	2 byte hexadecimal	example: 0000 :1234, ABCD
;	numerical value (range 2 byte)	example: 0000 ; 'A' * 100 , 18200 | 11b
[	numerical value (range 1 byte)	example: 0000 [100,' # '
"	character string data	example: 0000 " the LAUGHTER MEDITATION"

It is also ok to use debugger variables when 16bit input is required.
Example:

```
M@E
D000 LD HL,@B
D003 LD DE,@2
D006 LD BC,@E-@B
D009 LDIR
D00B RST 30H
D00C .1
D00D ;@S
D00F RET
```

Please note: This command has special exit. To exit back to command line mode, you need to press {CTRL}+{STOP}

Other commands:

- **BASIC list**
  BL <line number>
  LIST same as BASIC
  When this command is executed and the memory area where BASIC should be stored contains other than BASIC program data the routine may be unstable.

- **Hexadecimal keypad setting**
  TK <number>
  Used if you have a numeric keypad on your MSX. This selects the order of ABCDEF on the keypad for hex input: ABCDEF will be on the '+' '-' '/' '*' ',' '.' keys.
  Number can be from 1 to 4 (The following table comes from NMS8250):

KEY / Number	-	*	/	+	.	,
Type 1 Arrangement 1	**1**	D	F	E	C	A	B
Type 1 Arrangement 2	**2**	C	A	B	D	F	E
Type 2 Arrangement 1	**3**	F	E	D	C	B	A
Type 2 Arrangement 2	**4**	A	B	C	D	E	F

- **Block transfer**
  BT <source begin address>[#<slot >],<source end address>, <destination begin address>[#<slot>]
  Transfers a block of data

- **Clear screen**
  CLS
  Clears the screen. You can also use {SHIFT} + {HOME} to do same thing.

- **Change directory**
  CD <name of directory>
  This command requires MSX-DOS 2.

- **Relocate**
  RT <source start address>[#<slot >],<source end address>, <destination start address>[#<slot>]
  This command transfers machine language program to another address. All the memory pointers that point inside transfer area will be updated and changes will be listed to screen.

- **Filling memory**
  FL <begin address>[#<slot >],<end address>,<value>
  Fills memory with a single value.

- **Memory comparing**
  CM <begin address>[#<slot>],<end address>,<compare begin address>[#<slot>][,S]
  By default differences are listed.
  'S' switch lists similar values

- **Data searching**
  FD <begin address>[#<slot>], <end address>
  This command lists all memory addresses where searched can be found.
  Command asks for search string input which can be same as in **M** command although relative jump (JR and DJNZ) can not be searched. Search can be terminated with the {BS} or {ESC} key if needed.

- **Checksum type**
  CS
  Changes checksum type (toggle)
  'NORMAL' adds all the bytes together
  '+ ADR' adds start address + all the bytes together

- **Calculate checksum**
  TS <begin address>[#<slot>], <end address>
  The checksum of the memory segment calculated.

- **Execution of program in memory**
  GO <address>[#<slot>]
  Executes machine language program from defined memory address. For break point information, see RG-command.

- **Registers**

RG	Displays all the resisters
RG *	All areas other than stack are cleared
RG +	The stack is reset to the start address.
RG <register name>,<value>	Register is changed to the new value.

  Supported register names: A, B, C, D, E, F, H, L, A', B', C', D', E', F', H', L', AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, IXH, IXL, IYH, IYL, SP, BP and PC

  Please note: When the SP is changed (the stack pointer), the stack of the system changes! Stack pointer can not be set below 80FFH.

  The BP (break point) is not a CPU register, but can be used for debugging.
  If a GO command executed in the current slot and the BP address is reached, the program execution stops! Registers are saved, PC (the program counter) changes to the BP address and the stack stays the same. BP address can only be in the RAM.

- **Trace**
  TR <address>
  Executes one instruction at a time and displays the registers afterwards.
  [ESC] or [BS] quits
  When printer output is specified (A>?TR) the trace results are printed - used for tracing programs which alter the screen.

- **Machine check**
  CK
  Returns information on the machine. This information is printed to screen also when Super-X is started.

SLOT	Slot which is currently active
RAM	System RAM slot
X	Location of Super-X in memory
MAIN-ROM	Location of main ROM in memory
SUB-ROM	Location of SUB-ROM (or EXT-ROM) in memory
FDC-ROM	Slots of floppy disk conrollers
EXP	Presence of expansion of each basic slot (o = it is expanded, / * = it is not)
DISK	Drive names that are available
STACK	Stack pointer.
MAPPER SUPPORT	If a mapper support routine (DOS2 etc) is not found mapper details are not printed.
MAPPER RAM	Slots and available RAM segments (mapper pages) listed
MAPPER:	Selected segments for each RAM page

- **Setting of the function key**
  SF [<key number (numerical value)>, <character string>]
  Defines a function key
  When a numerical value is surrounded with "[" and "]", it is seen as a character code. For example [65] = A
  Command only without parameters initialises all the function keys.

- **Calculate**
  CL <expression>
  The binary, hexadecimal and decimal of a expression is returned. (See: Numerical values)

  Searching string in BASIC listing
  BF
  Search for data. The search string is asked.
  "?"can be used as a single character wildcard
  When this command is executed and the memory area where BASIC should be stored contains other than BASIC program data the routine may be unstable.

- **Mapper page select**
  PP [<page>,]
  The mapper segment which is assigned to the page which is appointed, is set.
  255 Segments are possible.

  This routine can only execute if a mapper support routine is present. If mapper support is not present you can use PO command instead. The command without parameters initialise the mapper.

- **Super Disassembler / data export**
  SD <file name>,<begin address>[#<slot>],<end address>[,<data export switch (B|D|X)>]
  By default this command disassembles and saves a text file to disk
  Data can be exported to other formats by adding data export switch. Available switches are:

  B DEFB type for assemblers.
  Example: **DEFB 3EH,80H,0CDH,80H,01H,0C9H**
  X X-BASIC type of inline ML
  Example: **10001 ´#I &H3E,&H80,&HCD,&H80,&H01,&HC9**
  D DATA type for BASIC.
  Example: **10001 DATA 3E80CD8001C9**
  Example of reading in BASIC:
  READ A$: FOR I = 0 TO LEN(A$)\2-1:POKE AD + I, VAL ("&H " + MID$ (A$, I * 2 +1, 2)): NEXT I

  Example:
  SD MYPROG.ASM,9000,9014

  Output:

```
;9000H-9014H
        LD      HL,X900E
        LD      DE,0C000H
        LD      BC,0123H
        LDIR
        JP      0C000H
X900E:  LD      H,40H
        LD      A,01H
        PUSH    AF
        PUSH    HL
        CALL    0024H
```

- **File summary (Directory listing)**
  FS <drive name>
  Lists files on disk. If DOS2 is available, you can also define directory or filemask after command.

- **Usage of the disk**
  CI <drive name>
  Shows "<the number of used clusters> / <number of all clusters>".

- **Global offset**
  OF [<offset>]
  Sets offset address that is applied to all addresses inside Super-X including for example other offsets and disk routines. Great care must be taken to avoid problems with OS and hardware, but this can be used for example to debug routines that are located in other than the real execution address. Offset parameter is absolute offset subtracted from real hardware memory address 0. (Parameter will become new address 0) If command is executed without parameters global offset will be set to zero.

  Example:
  OF 8100
  L#COMMAND2.COM,100
  I100

  This will load and start disassembling COMMAND2.COM from it's normal load address (100H) while the program is loaded to real life address 08000H

- **Change CPU mode**
  CU [<number>]
  Change the CPU. (MSX tR only)

  If parameter is not given, this will display current CPU mode.
  0 = Z80
  1 = R800 (ROM)
  2 = R800 (DRAM)
  3 = Z80 (DRAM)

- **Colour of the screen**
  CO [<foreground>], [<background>], [<edge>]
  Change the colour of the screen.

- **Displays memory in text using Japanese font**
  KR <address>[#<slot>]

  Display memory using Japanese font.
  (Japanese characters start from ASCII 128 – 255, 0-127 is the normal character set)

- **Type japanese text file on screen**
  KT <file name>
  Displays text file on screen with the Japanese character set.

  (Japanese characters start from ASCII 128 – 255, 0-127 is the normal character set)
  {ENTER} or {SPACE} = Pause
  {ESC}=Quit

- **Type text file on screen**
  TP <file name>
  Displays text file on screen.
  {ENTER} or {SPACE} = Pause
  {ESC}=Quit

- **Loads the Japanese character font.**
  KL [<drive name>]
  (re)Loads the Japanese character font file "SUPER-X.FNT " into the VRAM slot.

- **Save (BSAVE)**
  SV <file name>,<begin address>[#<slot>],<end address>,[<start address>[,<offset>]]
  Similar to BSAVE of the BASIC, but you can also set offset for data loading. Using offset here will only affect the BASIC binary file header.

- **Load (BLOAD)**
  LD <file name>[,<offset>[#<slot>]]
  Similar to BLOAD of BASIC.

- **RAW Saving**
  S# <file name>,<begin address>[#<slot>],<end address>
  Memory contents are saved in the file.

  Example:
  . To save 4000 – 7FFF of slot 2 as file “DUMP.ROM”
  S#DUMP.ROM,4000#2,7FFF

- **RAW Load**
  L# <file name>,<address>[#<slot>]
  Loads the selected file to the selected memory (and slot) address

- **Sector saving**
  S% [<drive name>:]<begin sector (numerical value)>,[< end sector>],<address>[#<slot>]
  Writes contents of memory to sector(s)
  If no end sector is selected then only one is saved!

- **Sector load**
  L% [<drive name>:]<begin sector (numerical value)>,[< end sector>],<address>[#<slot>]
  Loads the content of sector(s) to memory
  If no end sector is selected then only one is loaded!

- **Add a note**
  iM <address>, <slot data>, <type data>
  A new note is saved in the Note File.
  {cursor keys} move
  {ENTER} saves the note.
  {ESC} cancel.
  The slot data and the type data add a supplemental explanation of the note:

Slot data	0: General	1: MAIN	2: SUB	3: FDC	4: RAM			
Type data	0: General	1: BIOS	2: WORK	3: DATA	4: PORT	5: MATH	6: KEY	7: HOOK

- **Read note on address**
  iC <address>
  Checks to see if there is an note for the address given

- **Load the Note File**
  iL <drive name>
  Loads the Note File.

- **Save the Note File**
  iS <drive name>
  Saves the Note File.

- **I/O port in**
  PI <port>
  Reads a byte from the I/O port.

- **I/O port out**
  PO <port>, <numerical value>
  Writes a numerical value to the I/O port.

- **Quit**
  QT
  Returns to BASIC from SUPER-X. (Exits)