May 1983

OS-9 GMX™ III Manual Addenda
Version 1.1, 05/10/83

This document describes features specific to OS-9 GMX™ III, Version 1.1 (the GIMIX version of OS-9 Level II, for the GMX™ 6809 CPU III board), that are not documented in the standard OS-9 USERS and/or SYSTEM PROGRAMMER'S manuals. It also describes some important differences between this version of OS-9 and the earlier versions.

CONTENTS

* Denotes IMPORTANT variations from earlier versions or from the information in the standard OS-9 manuals. Please read before using this version.

***** RBF Changes *****

Due to changes in RBF affecting the bitmap and directory handling, disks created under GMX™ III, V1.1 (or Level I, V1.2/Level II V1.1) CANNOT be written on by previous versions of RBF. V1.1 disks can always be READ by older versions of RBF. V1.1 can read and write disks created by old RBFs; however, once written on by V1.1 RBF, the disks should not be written on by the old RBF. Note: This change affects all current implementations of OS-9 Levels I and II and is not specfic to the GMX™ III version.

OS9defs Files

The "OS9defs" files have been revised. They have been split into several different files to allow selective inclusion in assembler programs, depending on the intended application. A special file (li.equates) may be included in existing programs, that use the old short-form names, to equate the old names to the new, longer names used in the new "OS9defs" files.

***** Interrupts *****

Since OS-9 is an interrupt driven system, all I/O devices (including the time-of-day clock, disk controller(s), and serial/parallel I/O boards for terminals and printers) must have their IRQ interrupt outputs enabled.

Most devices do not actually generate an interrupt on the bus until initialized by OS-9; and a system reset (or turning off the power) clears any interrupt output that may be pending: once the device has been initialized. However, the 6850 ACIAs (used on GIMIX 1-port, 2-port and 8-port serial interfaces) have no direct connection to the system reset line. They can only be reset by the software, or by turning the power off. Because of this, it is possible to reset a running system and leave unserviced interrupts (from the 6850s) on the bus. These "left-over" interrupts can interfere with proper operation of the system when it is rebooted after a reset. Since OS-9 does not automatically initialize all of the serial ports when it is booted (normally only "TERM" is initialized), the "left-over" interrupts, if any, must be cleared by some other means. Turning the system off before booting will insure that the interrupts are clear, but this is not always practical. The INIZ command should be included in the startup file to clear any 6850 interrupts that may be left after a reset.

INIZ Command

The "INIZ" command is used to clear any pending interrupts from 6850 ACIAs when the system is rebooted after a reset. If these interrupts are present and not cleared when the system is booted, a variety of problems, including a marked reduction in system speed (throughput) or the inability to boot the system at all, may result.

SYNTAX: INIZ (devname) [(devname) (devname) ...]

INIZ clears any pending interrupts on the device(s) listed by opening and then closing a path to each of them. The list of devices should include ALL of the 6850 type serial devices (normally T1, T2, ... Tn, and P1) in the system, with the exception of "TERM" which is automatically reinitialized by the system. Only those devices that are included when the system is booted (devices in the OS9boot file) should be listed. If no devices are listed, INIZ attempts to read a device list from the STD input path.

EXAMPLE: INIZ /T1 /T2 /T3 /P1

If INIZ cannot open one of the devices, an error is returned and iniz terminates without further processing of the device list.

COPY Command

In addition to the changes noted in the "User's Manual", the copy command has been modified to preserve the "creation" and "last-modified" dates of the copied file. The new "COPY" will also preserve the file's owner ID if you are user number zero. To work properly, this version of the copy command (edition 7 or later) must be used with edition 11 or later of RBF.

***** Memory Addressing *****

OS-9 GMX™ III and the GMX™ CPU III allow a full 64K of RAM to be installed on each of the first 15 memory banks ($0-$E). The 16th bank ($F) can have up to 56K of RAM installed; the upper 8K are reserved for I/O and the operating system PROMs. RAM should be installed in contiguous 64K banks beginning at address $0000 on bank $0. All I/O devices including the motherboard and disk controllers must have extended address decoding enabled and set to decode bank address $F (see appendix a).

Since the GMX™ CPU III automatically switches to the system state when servicing interrupts, the interrupt vectors need not appear in each of the 16 memory banks as in previous systems. This means that up to 64K of RAM may be allocated to a user or task with no system overhead; the ENTIRE 64K is available to the user/task.

Dynamic Address Translation and Memory Attributes

The GMX™ CPU III includes an expanded Dynamic Address Translator (DAT) which, in addition to providing translation in 2K segments for more efficient memory usage, allows the assignment of "memory attributes" on a block by block (2K block) basis. These memory attributes are: Write Protect, Single Step, and Unallocated Memory. They may be selected separately or in combinations for each individual 2K memory segment in the address space. The purpose of these attributes is to provide protection for the system and to provide enhanced debugging capabilities. Currently only the Write Protect and Unallocated Memory attributes are implemented in OS-9 GMX™ III.

Write Protect Attribute

The WPT attribute is used to protect areas of memory, such as those containing sharable OS-9 modules or user programs, from unauthorized and/or unintentional modification. This greatly increases the security of the system by preventing, for example, one user from corrupting the copy of BASIC09 that is being shared by with users. OS-9 determines the intended status of a program (protected or unprotected) by testing a bit in the module header when the module is loaded into RAM. Once loaded, a protected module cannot be written to or modified, except by replacing it with another module.

Normally, all of the modules and programs supplied by GIMIX have their write-protect status set to "protected". Since the system prevents even intentional modification of protected modules, it is not possible to use the Debugger to make changes to a write-protected module. In order to permit modifications to certain modules, such as device descriptors, source files for these modules are included on the system disk(s). Any necessary modifications must be made to the source files, which can then be assembled and installed.

To facilitate debugging, modules can be assembled and run without write protection. However, to maintain maximum system security, they should be write protected once the debugging is completed. Note: It is advisable, when debugging a program, to test it first with the write protection enabled, to protect the system from crashes (especially if there are other users) and to help detect certain types of bugs.

The write protect status of a module is determined by bit 6 (previously undefined) in the module header's attributes/revision byte (module offset $7). (Bit 7 of this byte is the reentrant/sharable bit, see the System Programmer's Manual.) If this bit is cleared (0) the module will be write-protected when loaded by OS-9, if set (1) the module will be unprotected.

If the GMX™ CPU III detects an attempt to write to write-protected location in memory, a trap occurs and, as a result of the trap, OS-9 will close down the offending task (the task that initiated the write operation) and issue an error message (Error #198).

Unallocated Memory Attribute

The UAM attribute is used to prevent a user/task from making memory accesses (read or write) outside the memory area assigned to it by the operating system. Like the WP attribute, this increases the security of the system by preventing unauthorized access to selected areas of memory. The UAM attribute is automatically controlled by OS-9. When a task is initiated, all memory requested for the task is "allocated" to that task, all memory outside the requested area is "unallocated" and therefore inaccessible to that task.

If the GMX™ CPU III detects an attempt to access (read or write) unallocated memory, a trap occurs and, as a result of the trap, OS-9 will close down the offending task (the task that initiated the access) and issue an error message (Error #199).

Watchdog Counter

An additional function on the GMX™ CPU III board, the Watchdog Counter, is also implemented in OS-9 GMX™ III. The WDC is controlled by the operating system and, when enabled, limits the length of time that interrupts may remain masked and therefore unserviced. The WDC is enabled by OS-9 when the system is switched from the system state (operating system) to the user state (user program running). It begins counting CPU clock cycles when an interrupt occurs and, if the interrupt is not serviced by the CPU in a specified number of cycles (hardware selected: 128 standard,32 optional) a trap occurs. When a WDC trap occurs, OS-9 will close down the offending task and issue an error message (Error #197).

The WDC trap provides two types of protection. It protects against a user program that masks IRQ interrupts; necessary to the system for task switching and I/O processing. It also protects against the execution of certain illegal opcodes which can lock the 6809 in a state in which it will not execute further instructions or respond to any interrupts. To recover from this processor state, the WDC generates a special reset (only the 6809 is reset). The reset is processed through a special trap-vector instead of the normal reset vector, allowing OS-9 to resume processing of all but the task which caused the trap.

Three utility programs are provided with the OS-9 GMX™ III system to demonstrate and test the function of the WPT, UAM, and WDC traps. The utilities (test199uam, test198wpt, and test197wdc) are provided in both source and object form to help the user understand the function of the traps. Execution of one of the utilities will cause OS-9 to close down the task and issue the appropriate error message.

TEST199UAM

Test199uam tests the function of the Unallocated Memory trap by reading from a memory location beyond the end of its normally allocated data area. Note: if test199uam is run with a memory-size modifier in the command line (i.e. test199uam #6K) a trap will not occur and the program will loop until "KILL"ed.

TEST198WPT

Test198wpt tests the function of the Write Protect trap by writing to itself. Since bit 6 of the program's attributes field is clear (0), the module is write-protected and a trap will occur.

TEST197WDC

Test197wdc tests the function of the Watchdog Counter by setting the IRQ mask and entering a wait loop. Once the watchdog count expires (normally 128 CPU cycles) the trap will occur. Note: in order for this test to function, an IRQ must occur after the program is executed; to start the watchdog count. If the system clock has been started (by running "setime") the clock interrupts will cause the trap to occur. If the system clock is not generating interrupts, an interrupt can be caused by hitting a key on the terminal.

In addition to providing a considerable degree of protection for the system, the attributes and watchdog traps also assist in software debugging. In many cases they will detect common problems (caused by uninitialized pointers, incorrect addressing modes, etc.) that might otherwise go unnoticed or be difficult to identify.

Device Descriptors

The device descriptors provided with OS-9 GMX™ III follow the conventions established in earlier versions of OS-9 from GIMIX. The header on the catalog printout, included with the system disk, lists the configuration of the disk, terminal, and printer descriptors provided. Since the descriptors are write-protected by the GMX™ III system when they are in memory, the only way to modify them for user-specific requirements is to modify the source code and reassemble them. The GMX™ III system does not permit "hot patching" with debug as in previous versions. The source code to all of the standard device descriptors is provided in the "SOURCE" directory on the system disk(s).

***** Device Descriptor Changes *****

Due to changes in the Device Descriptors (Device Controller Address field), device descriptors from earlier versions of OS-9 (Level II V0.5 or earlier and OS-9 level I prior to version 1.2) CAN NOT BE USED, without modification, with OS-9 GMX™ III. In order to use existing descriptors, the extended address byte (at module offset $E) must be set to $0F. Without this modification, the system will appear to function, but performance will be seriously degraded!

***** X-ON/X-OFF *****

In addition to the address change noted above, the descriptors for ACIA devices (TERM, T1, T2, ... , and P1) have additional bytes added to control the X-ON/X-OFF functions (module offset $2A = X-ON, offset $2B = X-OFF). These bytes can be set to the desired ASCII codes for these functions if they are required. Setting these bytes to $00 disables X-ON/X-OFF (terminal descriptors shipped with GMX™ III have these bytes set to $00). X-ON/X-OFF can also be enabled using TMODE. When X-ON is set to $11 (control-Q), the QUIT character, normally control-Q, (module offset $23) must be set to some other control character. The recommended substitute character is $05 (control-E). Note: When terminals that generate X-ON/X-OFF sequences are used, X-ON/X-OFF must be enabled in the system.

G68 DMA Disk Controller Drivers

OS-9 GMX™ III uses an enhanced version of the G68 device driver. This G68 supports 3 ms. stepping rates for 5.25" disk drives, allowing faster access times with drives that are capable of stepping at this rate. (All 5.25", 80 track (96TPI) drives currently supplied by GIMIX are capable of stepping at 3 ms.) This option is controlled by a separate bit in the floppy disk device descriptors as described below.

The new G68 prevents the drive motors from timing-out if a drive with no disk (and no "Drive Ready" line) is accessed. On earlier versions of G68, if the drive motors timed-out the system would hang and usually require re-booting. This feature will only function if the system interrupt clock has been started by using the "SETIME" command. We recommend that "SETIME" be included in the "STARTUP" file ("setime 00" will start the clock running, without prompting for new values). Note: To prevent the system from hanging if a non-existant drive is accidentally accessed, a system disk should be created that does not have device descriptors for the non-existant drive(s).

G68 Device Descriptors (D0,D1, etc.)

The device descriptors for the G68 driver follow the definitions given in the OS-9 manuals with the following exception.

MODULE OFFSET	NAME	DESCRIPTION
$14	IT.STP	Bits 0 (LSB) and 1 determine the basic stepping rate as shown in the 179x table in the manual. Bit 7 (MSB) controls "fast stepping" for 5.25" floppy drives

The fast stepping bit (Bit 7) when set (1) in a descriptor for 5.25" drives causes the stepping rate (determined by bits 0 and 1) to be doubled. The 5.25" drive will be stepped at the rate shown in the 179X, 8" drive column in the table. When Bit 7 is clear (0), the drives will be stepped at the normal 5" rate shown in the 179X, 5" column. The fast stepping bit has no effect on 8" drives and should be clear (0) in 8" descriptors.

Caution: Be sure the drives are capable of stepping at the selected stepping rate. In general the only 5.25" drives capable of stepping at 3 ms. (IT.STP = $83) are the newer 80 track (96TPI) drives. Various combinations of the step rate and fast stepping bits can be used to match the stepping rate to the drive(s) used. See the disk drive manufacturer's literature to determine the stepping capabilities of the drive(s). Attempting to step a drive faster than its specified maximum rate will cause excessive disk errors and/or prevent the drive from working at all.

Hard Disk Driver and Descriptors

Like floppy disk drivers, the hard disk drivers consist of two parts; the device driver (XBC) that interfaces the hardware to the Random File manager (RBFman) and the device descriptors (H0 and H1) that describe the characteristics of the hard disk drives. The format of the hard disk device descriptors is essentially the same as the floppy disk descriptors with the following exceptions:

MODULE OFFSET	NAME	DESCRIPTION
$14	IT.STP	defines stepping method used by the hard disk controller set to $02 for CMI drives
$15	IT.TYP	Bits 6 and 7 same as floppy Bit 5 = Logical Unit Number (LUN) † Bits 0 through 4 select one of five possible controllers on the DMA interface. *
$16	IT.DNS	set to $00

† This bit is passed to the HD controller to select one of the two possible drives. Normally it must be cleared (0) if IT.DRV is $00 (drive 0) and set (1) if IT.DRV is $01 (drive 1).

* The current configuration supports only 1 HD controller. Bit 0 (LSB) must be set (1) and Bits 1 through 4 must be clear (0).

Formatting Hard Disks

The same FORMAT program is used to FORMAT both floppy and hard disks. There are several differences in the options available when formatting hard disks, as described below. Hard disks are normally formatted only once; when the drive is first used. Once the hard disk is formatted it does not require reformatting unless the file structure is damaged by a hardware or software fault, or if it becomes desirable to erase ALL of the files on the disk at once.

CAUTION !!

A physical format (see below) destroys all data recorded previously on the disk. A logical format, while it does not completely destroy previously recorded data, makes the data very difficult if not impossible to recover!

Hard disk systems are normally shipped with the disk(s) already formatted by GIMIX. This formatting is done as part of the standard testing performed on the system. The disk may also have copies of the files provided on the system (floppy) disk. Before formatting the hard disk(s) use the directory (DIR) command to check the contents of the hard disk(s). If the disk(s) are already formatted they can be used as-is or reformatted as desired. If reformatting is desired, using the logical-only format option (see below) will save time and accomplish the same results as a complete physical and logical reformatting. Note: the logical-only format option can be used to reformat disks that were previously formatted for a different operating system (such as FLEX).

When FORMAT is called with /H0 or /H1 as the device to be formatted, it first prints a list of the parameters that will be used to determine the format of the hard disk. Unlike the floppy disk format, these parameters are fixed by the requirements of the drives and cannot be changed from the FORMAT utility. Entering either "N" (NO) or "Q" (QUIT) at this time will abort the program and return to the calling program (normally shell). If "Y" is entered, FORMAT will ask :

Both Physical and Logical Format ?

Answering "Y" (YES) will cause FORMAT to perform both types of format. If "N" (NO) is entered only the logical format will be performed.

PHYSICAL FORMAT

The physical format is normally only required when formatting a new hard disk that has not been formatted previously. The physical format records the information required by the controller to divide the disk into sectors and to locate a particular track and sector on the disk. The operating system does not modify this information once it is written.

LOGICAL FORMAT

The logical format records the information that OS-9 requires to store and keep track of the data files that will be written to and read from the disk. Some of this information is modified by the operating system as files are written to and deleted from the disk. Performing a logical format has the same effect as deleting ALL files on the disk. The logical format takes considerably less time than the physical format.

After the type of format desired is entered, FORMAT will prompt for a disk name, which must be entered following the same syntax as the name for a floppy disk. The next prompt is:

Physical Verify Desired ?

Answering "Y" will cause a physical verify to be performed. Each sector is read and the information written there is verified. As the verify is performed, FORMAT prints the number of each completed track on the standard output device. Note: Before performing a physical verify, use TMODE to turn the pause function off or FORMAT will stop at the end of each page of track numbers. Answering "N" causes the physical verify to be skipped.

Booting From The Hard Disk

It is not possible, with this version of OS-9, to boot the system directly from the hard disk; the "OS9Boot" file must reside on a floppy disk. It is possible to create a special version of "OS9Boot", on a floppy disk, that will transfer control to the hard disk after the boot file is loaded from the floppy. The initial directories will be /H0 and /H0/CMDS, the system will attempt to execute the "STARTUP" file from /H0, and "Login" will expect to find the "PASSWORD" and "MOTD" files in the directory /H0/SYS.

To facilitate creation of the required OS9Boot file, the files OS9Boot.core and Init.hd are included on the system disk. OS9Boot.core consists of all of the modules normally included in OS9Boot except for the device descriptors (The "H0" and "Pipe" descriptors are included) and the init module. (Use the Ident command, -s option, to list the modules contained in OS9Boot.core.) Init.hd is a special version of Init for the hard disk.

Before attempting to create a floppy disk that will boot to the hard disk, use one of the procedures outlined in the OS-9 manuals to create backups of the original system disk and store the original in a safe place.

To create the special OS9boot required, you must use the "OS9gen" command to combine OS9Boot.core, Init.hd, and at least two other device descriptors, D0 and TERM. Additional device descriptors and modules can be included as required, but these four MUST be included or the disk will not boot. The "SAVE" command must first be used to create files containing copies of D0, TERM, and any additional device descriptors that are required. Note: if changes to the standard device descriptors are required (such as changing drive stepping speeds or enabling X-ON/X-OFF for terminals), the appropriate source files must be modified and assembled to generate the necessary files.

The following example outlines the procedure for a system that includes one floppy disk (D0), one hard disk (H0), two terminals (TERM and T1), and a parallel printer (P). It assumes that the hard disk has been formatted, that the necessary system files have been copied to the hard disk, and the current working and execution directories have been changed to the hard disk. The files OS9Boot.core and Init.hd should exist in the working directory. A freshly formatted floppy disk should be installed in D0. Note: Since the floppy for booting to the hard disk will only contain the OS9Boot file, it does not need to be formatted for its full capacity. A single-sided single-density disk is all that is required.

OS9:SAVE DO	Create a file containing the descriptor "DO"
OS9:SAVE TERM	Create a file containing the descriptor "TERM"
OS9:SAVE T1	Create a file containing the descriptor "T1"
OS9:SAVE P	Create a file containing the descriptor "P"
OS9:BUILD HARDBOOT	Create a text file containing a list of the files to be included in the new OS9Boot file.

```
? OS9Boot.core
? Init.hd
? TERM
? DO
? T1
? P
? [return]
```

Additional descriptors or modules can be included in the list as necessary.

```
OS9:OS9gen /DO  /HO/HARDBOOT

OS9:
```

The disk created by this procedure can now be used to boot the system to the hard disk. The root directory of the hard disk (HO) must contain a CMDS directory, which becomes the execution directory, and the "STARTUP" file if one is to be used. Note: The hard disk must be up to speed and "READY" for the boot to work properly. Be sure to allow the disk to come up to speed before booting with this disk.

OS-9 ERROR CODES

The following error codes should be added to the error code list in the OS-9 manuals. These errors are generated by hardware "traps" on the GMX™ III 6809 CPU board.

HEX	DEC	
$C5	197	WATCHDOG COUNTER TRAP - The CPU was unable to respond to an interrupt before the Watchdog count expired. The interrupts were masked for too long in a user task or the CPU executed an illegal instruction and was unable to respond.
$C6	198	WRITE PROTECT TRAP - An attempt has been made to write to a module that has its write-protect attribute enabled.
$C7	199	UNALLOCATED MEMORY TRAP - An attempt was made to access (read or write) memory not allocated to the program.

Error #210 During FORMAT

The "FORMAT" program requires a large block (8K) of physically contiguous RAM for a track buffer. If the system memory is fragmented when "FORMAT" is run, and the first available memory blocks are too small, an error 210 will be returned and the format attempt will be aborted. This problem may occur after the system has been in use for some time or, depending on memory configuration and/or the contents of the "STARTUP" file, it may occur immediately after the system is booted. In order to format disks when the memory is fragmented it may be necessary to eliminate the small memory blocks so that a large enough block of physically contiguous memory is available for "FORMAT". In many cases re-booting the system will eliminate the fragmentation. If this is not practical, or if the memory is fragmented when the system is booted, it will be necessary to start an additional process, either on another user terminal or in "background" to use up the smaller memory fragments temporarily. The easiest way to accomplish this is to run the "SLEEP" command in background, assigning it enough memory to use up the small fragments. For example:

```
OS9: SLEEP 1000 #10K&          (sleep for 1000 "ticks" using 10K
                                 of memory)

OS9: FORMAT /D1
```

The sleep command will use up the smaller memory fragments (up to 10K) and allow "FORMAT" to load with sufficient contiguous RAM for its buffer. Once the format command has been loaded, the sleep can time-out and release its memory for other uses. The number of "ticks" can be increased or decreased to allow sufficient time for "FORMAT" to load, without tying up the memory for an excessive length of time. Any program that uses up the smaller blocks of memory and can run concurrently with "FORMAT" will eliminate the problem.

We would appreciate information from users on any other problems encountered while using this version of OS-9 GMX™ III. To be most useful, the information should be submitted in writing, and should include a complete description of the problem and the conditions under which it occurs, including the Edition number(s) of any programs involved (use the IDENT command). A brief description of the hardware configuration should also be included so that we can attempt to duplicate the problem if necessary. Information regarding problems with the software should be addressed to:

GIMIX Inc.
1337 W 37th Place
Chicago Il 60609

Attn: Mike Magnus

SWITCH CONFIGURATION DRAWINGS FOR OS-9 GMX II & III

The following drawings show the standard DIP-switch configurations for the GIMIX 64K RAM board(s), #68 DMA Disk Controller, Hard Disk Interface (SASI), and Mother Board; when used with OS-9 GMX II & III.

The disk controller(s) must be set to both drive and decode extended addressing (both "ENA" switches ON). The motherboard must be set to decode extended addressing. The boards are addressed so they appear only on bank $F (A16, 17, 18, 19 = ON), at the appropriate base address.

The memory boards are addressed for either 64K banks (OS-9 GMX III systems or OS-9 GMX II systems with modified #05 CPU boards) or 56K banks (unmodified OS-9 GMX II systems). The configuration of switch S3 will normally be the same on all boards in the system, with section-7 ON for 64K banks or OFF for 56K. Both extended address enable switches ("XON"-sections 1 and 6) must be "ON" on all boards. The remaining eight sections of S2 determine the bank address of the board. The boards are divided into two halves, with sections 2,3,4, and 5 used to set the bank address for one half and sections 7,8,9, and 10 the other. In this application both halves are set to the same bank address. The switches are set in a binary pattern with section-2(7) being the least significant bit and section-5(10) the most significant. See the drawing for examples. The boards should be addressed on consecutive banks with the first board on bank $0, the second on bank $1, etc.

***** CAUTION *****

In addition to the boards shown in the drawings, any other memory-mapped boards installed on the 50 pin bus (GIMIX 8-port Serial Interfaces, Intelligent Parallel Interfaces, PROM/ROM boards, etc.) must be capable of extended addressing. Normally, I/O-type boards must be addressed on bank $F, with a base address in the $E000-$EFFF range. Memory-type boards (PROM/ROM) can be addressed as appropriate to the application, as long as extended addressing is enabled. Note: Standard versions of OS-9 only search the lower 56K of bank $F for PROM/ROM memory modules. In order to be located by OS-9, boards containing OS-9 modules in PROM/ROM must be addressed on bank $F.

***** NOTE *****

The switch configuration shown for the #68 DMA board assumes that the system will be booted from a 5.25" drive (D0). If D0 is an 8" drive, S2 section-9 must be OFF.

64K RAM BOARD
SWITCH CONFIGURATION
FOR OS-9 GMX II & OS-9 GMX III

BANK ADDRESS (S2)

	$Ø	$1	$2	$3	$F	
1	ON ←				→ ON	XON
2	OFF	ON	OFF	ON	ON	A16
3	OFF	OFF	ON	ON	ON	A17
4	OFF	OFF	OFF	OFF	ON	A18
5	OFF	OFF	OFF	OFF	ON	A19
6	ON ←				→ ON	XON
7	OFF	ON	OFF	ON	ON	A16
8	OFF	OFF	ON	ON	ON	A17
9	OFF	OFF	OFF	OFF	ON	A18
1Ø	OFF	OFF	OFF	OFF	ON	A19

BOARDS SHOULD BE ADDRESSED ON SUCCESSIVE BANKS - (1ST BOARD, BANK Ø; 2ND BOARD, BANK 1; 3RD BOARD, BANK 2; ETC.)

S3

1	2	3	4	5	6	7	8	9	1Ø
ON	ON	ON	ON	ON	OFF	*	ON	ON	ON

$ØØØØ-$DFFF ($FFFF)

* ON=64K OFF=56K

USE 64K FOR ALL BOARDS (EXCEPT BANK $F) IN GMX III AND MODIFIED GMX II SYSTEMS.

USE 56K FOR UNMODIFIED GMX II SYSTEMS.

DMA FLOPPY DISK CONTROLLER
OS-9 GMX II & GMX III CONFIGURATION

ADDRESS= $FE3BØ BOOT DRIVE (0)= $5\frac{1}{4}$"

EXTENDED ADDRESS (S1 1–5) — REGISTER BASE ADDRESS (S1 6–10, S2 1–8)

S1

	1	2	3	4	5	6	7	8	9	10
	ENA	A16	A17	A18	A19	A3	A4	A5	A6	A7
	ON	ON	ON	ON	ON	OFF	ON	ON	OFF	ON

ON=1

OFF=Ø

S2

	1	2	3	4	5	6	7	8	9	10
	A8	A9	A10	A11	A12	A13	A14	A15	5"	ENA
	ON	ON	OFF	OFF	OFF	ON	ON	ON	ON	ON

S1 1: EXT ADDRESS DECODE

S2 9: $5\frac{1}{4}$" OR 8" BOOT SELECT

S2 10: DMA EXT. ADDRESS

HARD DISK INTERFACE
SWITCH CONFIGURATION
OS-9 GMX II & GMX III

ADDRESS= $FE3B8

REGISTER BASE ADDRESS (S1 A3 – S2 A15)

EXTENDED ADDRESS (S2 A16 – A19)

ON = 1

OFF = Ø

S1

	1	2	3	4	5	6	7	8	9	10
Signal	A3	A4	A5	A6	A7	A8	A9	A10	A11	RST
Setting	ON	ON	ON	OFF	ON	ON	ON	OFF	OFF	OFF

RESET DISABLE — switch 10

S2

	1	2	3	4	5	6	7	8	9	10
Signal	A12	A13	A14	A15	A16	A17	A18	A19	ENA	ENA
Setting	OFF	ON	ON	ON	ON	ON	ON	ON	ON	ON

DMA EXT. ADDRESS — switch 9

EXT. ADDRESS DECODE — switch 10

MOTHER BOARD SWITCH CONFIGURATION
OS-9 GMX II & GMX III

ADDRESS = $FE000

S2 — OFF (OPEN) Ø / ON (CLOSED) 1

S2		
OFF	4	I-O SLOT
OFF	8	
ON	16	
OFF	A5	
OFF	A6	
OFF	A7	LOW ORDER
OFF	A8	STARTING
OFF	A9	ADDRESS BITS
OFF	A10	
OFF	A11	

S1 — OFF (OPEN) Ø / ON (CLOSED) 1

S1		
OFF	DIS	I-O ENABLE/DISABLE
OFF	A12	
ON	A13	HIGH ORDER STARTING
ON	A14	ADDRESS BITS
ON	A15	
ON	EXON	EXTENDED ADDRESS ENABLE/DISABLE
ON	A16	
ON	A17	EXTENDED ADDRESS
ON	A18	
ON	A19	