Marinchip M9900 Network Operating System

Multi-user version (NOS/MT) System Generation Guide

for Release 2.6

by John Walker

Revised October 1981

1. Introduction

The Marinchip Multi-user Network Operating System (NOS/MT) is a complex operating system. It offers a large number of optional features, supports many different hardware devices, and allows great flexibility to the user in configuring systems. Consequently, the process of system generation must allow the specification of all of these variables. As a result, system generation of NOS/MT is much more complicated and time consuming than generation of earlier Marinchip operating systems. There is no room for error, as many system generation parameters will cause the system to fail in strange and unpredictable ways if they are incorrectly specified.

This manual takes you through the process of system generation step-by-step, explaining at each step what your choices are and how to set each parameter. Since the system software configuration interacts closely with the hardware configuration, definition of the hardware configuration is also given a thorough discussion.

There is no way any manual can explain all of the possible combinations of configuration parameters available in NOS/MT, so this manual will try to explain each parameter in a way that makes obvious its interaction (if any) with the others. If the configuration you want to put together doesn't seem to be covered in this manual, or you do not understand something, please call us at Marinchip Systems and ask for an explanation. DO NOT TRY TO GUESS AN ANSWER! We will update this manual based on the questions and comments we receive from you.

2. Defining the hardware configuration

This section explains the hardware configurations supported by NOS/MT and tells you how to set the hardware options on each piece of hardware supported.

2.1. Memory

We will assume in this manual that the system contains a Marinchip PROM/RAM board which contains the debug monitor and disc boot PROMs and supports the system console terminal. Configurations without the PROM/RAM board are possible when using certain disc controllers, but will not be discussed here. Contact Marinchip Systems for assistance in installing the system on such a configuration.

The PROM/RAM board occupies memory addresses F400 to FFFF. The M9900 I/O page occupies F000 to F3FF, so user RAM may be installed from 0000 to EFFF on most systems. If additional ROMs, disc controller buffers, memory-mapped video buffers, etc., are to be placed in the system memory map, they should be placed immediately below F000 with as little wasted space as possible. Since such boards are rarely bank switched, space assigned to such boards is lost to all user spaces in the system, so the space they occupy should be minimised.

2.1.1. System memory map

Let us now consider the allocation of memory in NOS/MT. Memory is fundamentally divided into COMMON SPACE, SYSTEM SPACE, and USER SPACE. Common space is an area of memory at the high end of RAM space which is always present in the memory map: it is not bank switched, and in fact the memory which holds it need not even support bank switching. Common space is used to hold critical interrupt code and system tables which must be accessable at all times, even when a user is in control of the machine, and also contains the code and data space used to move data back and forth from the system to the user. Common space is not directly accessed by the user, and is not a part of the user's memory area. System space contains the bulk of the operating system. This space is below common space in memory, and is switchable in and out of the address space by bank selection. Each configured user (or background batch space) requires a user space. User spaces start at address zero and may extend up to the start of common space in the address map. User spaces normally overlap the addresses of system space.

If you did not immediately understand the preceding paragraph, do not be concerned: it takes a while to grasp the way NOS/MT manages memory. Let us consider an example which should clarify things a bit. Assume we are configuring a system with 196K bytes of memory (three M9900 64K RAM boards) which is to support two users. One 64K board will be given to the system, and each user will have a 64K board for his user space. Such a system will require a common space of less than 4K bytes, and since 4K bytes is the minimum memory page size in the M9900 64K board, we will assign 4K to common. (Determining the size of common space, and allocating it to the proper addresses in the link of the system will be explained later in this manual. For the moment, just assume we know how large it should be.) As always, the upper 4K

of RAM is occupied by the I/O vector and the PROM/RAM board. We assume this system has no peripherals that occupy memory space and no ROM other than that on the PROM/RAM board. Then the system memory map would look like the following:

Address	Common	System	User 1	User 2
0000-DFFF		64K £1	64K £2	64K £3
E000-EFFF	64K £1			
F000-F3FF	CPU I/O			
F400-FFFF	PROM/RAM			

In the above table "64K £1" indicates the first 64K board, "64K £2" the second 64K board, and so on. "PROM/RAM" indicates that the address range is occupied by the PROM/RAM board, and "CPU I/O" identifies the space used by the I/O ports. Hence we see that the space from 0000 to DFFF (56K bytes) is switched among the three boards, while addresses E000 to FFFF are always present in memory regardless of whether the system or any user is in control.

Normally the operating system will not fill the 56K bytes available to it under the above memory configuration. In a typical two user system there may be 24K to 32K unused bytes in the memory board used by the operating system. This space need not go to waste, as an additional user space may be configured there. It is even possible to configure a system for a single user in which the system and the user share a single 64K board. When doing this, however, the size of user programs is restricted to the 32K or less available to the user in such a configuration. We will cover how to configure such a system when we discuss the software memory configuration in a later chapter of this manual.

Memory banks may be configured as file storage devices instead of user spaces. All of the addressing considerations described above apply regardless of whether the bank is used for a normal user, a background batch space, or for file storage. The only difference is that the unused space below the operating system cannot be used for file storage.

2.1.2. Switch settings for the M9900 64K RAM

Each M9900 64K RAM board used in the system must have a unique Bank Select Port address. Marinchip Systems normally sets the first board (which will contain the system) to address 80 hex, the second board to 81 hex, the third to 82 hex, etc. If these I/O addresses conflict with other addresses in your system, they need not be used. However, the addresses used must agree with the code inserted in the file COMDATA (which will be explained later) to enable and disable the memory banks.

The Initial Bank Select switches on the first 64K board should be set to enable all the RAM space to be used for the system and common areas. Using our sample memory configuration given above, the switches on the first board would be set to enable pages 0 through E and disable page F. The Initial Bank Select switches on all other 64K RAM boards must be set to disable all memory (all switches on/closed). This configuration will cause the system memory board to be present in

the memory map when the system is loaded, with all user memory boards disabled. Once the system is loaded, it will assume software control over the memory map and select the various boards as required by the operation of the system.

2.1.3. Configuration for other memory boards

When using bank-switched memory boards other than the M9900 64K RAM board an analogous setup procedure is used. Most other bank-switched memory boards permit memory enable and disable only in increments of 16K bytes, so less freedom is available in setting up the memory map (since the common area must always be present in the address map and since a full 16K must be enabled to hold it, user space is restricted to 48K bytes). The board(s) which will hold the system should be initially enabled and boards assigned to user space should be initially disabled. Board(s) assigned to each distinct space (system, users) should be assigned distinct bank select port addresses. Since the operation of the bank select hardware differs so much from board to board, no more specific information can be given here.

2.2. Serial I/O boards

This section discusses the hardware setup for serial I/O boards used to attach terminals and printers to the system.

2.2.1. Marinchip PROM/RAM/SIO/RTC board

The serial I/O port on the PROM/RAM board is normally used to interface the terminal for the first user configured in the system. The CRU address used for the port on the board is forced to be 000 since the PROM monitor assumes that address for communication with the terminal during initial startup of the system. For use with NOS/MT, an interrupt jumper must be installed in option socket S10 on the board. We normally configure all serial I/O boards to interrupt with priority 2, and hence the jumper is installed between pin 3 and pin 14 on this socket. The TMS9902 chip on the PROM/RAM board is also normally used to provide the system's real time clock. Since a common interrupt line is used for both the terminal and the clock, the interrupt priority assigned to the terminal will also be used for the clock.

2.2.2. Marinchip Quad SIO

The Marinchip Quad SIO board provides four independent serial I/O ports implemented with the TMS9902 chip (the TMS9903 can also be used to provide synchronous communication, but NOS/MT does not provide software support for this chip). From a programming and configuration standpoint, these ports are identical to the port on the PROM/RAM board. Since the Quad SIO board provides a block of four ports, it occupies 080 hexadecimal CRU addresses. Thus, the board is configured at a base address which must be a multiple of 080 (e.g. 080, 0100, 0180, etc.). The four ports on the board are addressed by their port bias (0 for port 0, 020 for port 1, 040 for port 2, 060 for port 3) plus the base address set on the board. Refer to the Quad SIO manual for instructions on setting the switches on the board to configure the base address, and also how to configure the board to accomodate various terminals and printers.

Note that when the Quad SIO board and the PROM/RAM board are used in the same system, the Quad SIO board must be set to a base address of 080 hex or above. This is because the port on the PROM/RAM board is addressed at 0, and if the Quad SIO board were also addressed at 0, its port 0 and the PROM/RAM board port would overlap, causing both ports to work improperly.

Each port on the Quad SIO board has a separate interrupt lead. These leads are connected to the system's interrupt request lines by an option socket on the board. All ports which are being used to connect terminals are normally connected to interrupt priority 2. Ports being used to connect printers are normally connected to interrupt priority 4. Important: do not connect interrupt requests from ports on the Quad SIO board which are not configured in the NOS/MT configuration. If a port is not configured, NOS/MT does not send it the initialisation sequence which programs the interrupt generator. If this is not done, the port will generate random interrupts which will cause the system to "hang" as soon as it is loaded from disc. You must always configure a port in the software before enabling its interrupt on the Quad SIO board.

Any of the ports on the Quad SIO board may also be used for the system's real time clock. See the section on the real time clock table below for further information.

2.2.3. Marinchip SIO board

The Marinchip SIO board is a partly populated version of the PROM/RAM board which contains only the components necessary to provide one serial I/O port. This board has been superseded by the Quad SIO board described in the preceding paragraph and is no longer manufactured. This section describes its configuration so that users of this board can properly generate NOS/MT systems to support it. The Serial I/O port on the board is configured exactly like the port on the PROM/RAM board, so see the discussion of the PROM/RAM board configuration above for more information. When using the Marinchip SIO board to attach other terminals to the system, each board must be given a unique CRU address. Since the first terminal is configured at 000 and each board uses 020 bits of CRU space, the second terminal is normally configured at 020, the third at 040, the fourth at 060, and so on.

The interrupt jumper for all terminals is normally set to priority 2, which is implemented by a jumper from pin 3 to pin 14 on the option socket S1O. In systems with many terminals, it may be advantageous to configure some of the terminals on different interrupt priorities to reduce the time needed to determine which terminal caused an interrupt. If this is done, be sure to configure the board which is used for the real time clock function (normally the one at address 000) at the highest priority (which is the lowest priority number).

When the Marinchip SIO board is being used to interface a printer such as the T.I. 810, the interrupt priority is normally set to 4, which is done by installing the jumper from pin 5 to pin 12 of the interrupt option socket S1O. The lower priority is used on the printer interrupt to prevent a heavy interrupt load from a high speed printer causing loss of characters typed on terminals.

2.2.4. IMSAI SIO2-2 board

The Network Operating System also supports terminals and printers on the IMSAI SIO2-2 board which, even though it is no longer manufactured, is still very popular and is used in many M9900 configurations. Each SIO2-2 provides two independent serial I/O ports. In order to be used with NOS/MT, the interrupt option on the SIO board must be enabled. The two ports may be wired to the same interrupt priority, or to two different priorities. Adhering to the conventions discussed above, we normally wire the interrupt for a port used for a terminal to priority 2, and the interrupt for a printer port to priority 4. As always, other priorities may be used as long as the software configuration agrees. Refer to the SIO2-2 manual for information on how to connect the interrupt option jumpers.

There are no restrictions on the use of the SIO2-2 to connect terminals to the system. However, the way in which the interrupts on the SIO2-2 operate imposes restrictions on its use to interface printers. Basically, if an SIO2-2 is used to connect a printer, the other port on the SIO board cannot be used without the implementation of extremely tricky code which is far too complicated to explain here. Also, with the SIO2-2 the printer port is output only. This means that printers whose operation requires that they send characters to the computer (for example to tell it to stop sending data) cannot be used with the SIO2-2 board. There is no problem, however, in using the SIO2-2 to connect printers such as the T.I. 810, which use a hardware ready signal rather than sending characters to the computer.

2.3. Disc controller boards

This section discusses hardware configuration for the disc controller boards supported by NOS/MT.

2.3.1. Teletek FDC-II

The following instructions apply to the Teletek FDC-II revision 6 board. Revisions 5 and later are a complete redesign of the FDC-II, so the instructions given here are inapplicable to earlier models of the board. These earlier models can be used with NOS/MT; if you have one, contact us and we will tell you how to configure it. The software driver in NOS/MT supports both the old and new model FDC-II; no software changes are required.

If you bought your FDC-II from Marinchip Systems, it will have been properly configured for use with the M9900 and tested on an M9900 system. All FDC-II's are shipped with the interrupt jumper installed for priority 3, which is the standard for use with NOS/MT. No additional hardware configuration is required on FDC-II's set up by Marinchip Systems.

We assume in this manual that your FDC-II is already properly configured for use with the Disc Executive. Thus, the only possible change is installation of the interrupt jumper. If this jumper is not yet installed, install as explained below.

- Locate the interrupt jumper area. It is close to the edge

connector, on the left side of the board when viewed component side up. Install a jumper from the center hole to the hole which leads to pin 7 on the edge connector. This is the fourth hole from the left end of the group of holes.

These instructions configure the Teletek FDC-II for use with NOS/MT. The FDC-II may be run with its 1K RAM buffer at F400 or at EC00. Refer to the configuration table section for details on how this is specified when generating the system.

2.3.2. Tarbell floppy disc controller

The Tarbell floppy disc controller is supported in a non-interrupt mode by NOS/MT, and hence is configured exactly like it is for use with the Disc Executive, that is, all DIP switches off/open. NOS/MT allows computation to be done while the Tarbell controller is seeking by checking seek status at every interrupt of the real time clock. However, due to the timing constraints of the non-DMA, unbuffered Tarbell controller, the CPU must be dedicated to disc operation when I/O transfers are being done. Note that this means that the system will not respond to characters typed on terminal keyboards while a disc data transfer is in progress, and hence the Tarbell controller is poorly suited to multi-user configurations. We support it in NOS/MT because it is fine in a single user environment and because it is so popular.

2.3.3. Morrow Designs DISCUS hard disc controller

NOS/MT supports the Discus controller with any combination of 10-Megabyte and 26-Megabyte drives. The software is interrupt driven, and both of the interrupts on the controller must be strapped.

If you buy your hard disc system from Marinchip, the interrupts will be strapped correctly when you receive it. If you buy the disc from any other source than Marinchip, you will have to strap the interrupts before using the disc on NOS/MT. To do this, just connect the two marked holes at the lower left corner of the board (seen from the component side) to the hole marked "5" next to the edge connector.

Discs shipped by Morrow Designs have the wrong formatting for use under NOS/MT. We reformat all 110-volt systems that we sell, but we can not reformat 220-volt systems. Therefore, if yours runs on 220 or does not come from Marinchip, you must run our Morrow Format program before using the disc on NOS/MT. You can order this at the time you order your first Morrow disc.

2.3.4. Konan SMC-100 hard disc controller

NOS/MT provides full support of the Konan Storage Module controller. We offer full software support "off the shelf" for this controller when used with the Control Data Phoenix drive, and can configure systems for any drive this controller supports, upon request. The Konan controller is configured according to the standard Konan setup instructions, except that the open cable detect is set up to use CPU power rather than the power fail signal (A board jumper FF out, jumper GG in), the ROM is disabled (B board jumper P out, jumper N in), and the interrupt is wired to priority 5 (A board jumper G in). The I/O

address is normally left at Konan's standard value of 090 hex.

The Konan board is normally used in a system which also has a floppy disc drive. In such a system, the operating system is always loaded from the floppy, evem though the floppy may be seldom if ever used once the system has been brought up. Marinchip Systems can supply a PROM for the Konan board which loads NOS/MT directly from the cartridge disc. This PROM, which is installed in the Konan board with the PROM option enabled, performs the system load automatically when the system is RESET. Such a system need not have a PROM/RAM board. This PROM permits the configuration of systems with no floppy discs at all. Please contact Marinchip Systems if you wish to obtain the Konan direct load PROM.

2.3.5. Alpha Micro AM-200 controller

We do not recommend the Alpha Micro controller for new configurations. We include support code for the AM-200 with NOS/MT, and as of the last time we tested it it worked properly. However, we no longer have an AM-200 controller, so we cannot test the controller in new releases of the system nor offer assistance to users attempting to configure it. Assuming the AM-200 has been properly set up for use under the Disc Executive, the only additional configuration necessary is the installation of a jumper between the INT pad (at the bottom left side of the board) and the vectored interrupt hole that goes to pin 7 of the edge connector. This jumper assigns the AM-200 to interrupt priority 3. Since the AM-200 is a DMA controller, if it is to be allowed to transfer data while users are executing, all data transfers must be done to and from the common space area. In addition, all transfers must be to 8 bit memory, as the M9900 64K RAM board is not compatible with the AM-200. As a result, AM-200 configurations usually use an 8 bit RAM board as the common area, with system and user memory switched in 64K boards at lower addresses. Our AM-200 software driver handles all of the above mentioned problems so that the AM-200 can run under NOS/MT.

3. The NOS/MT generation process

This section presents an overview of the software generation process. Specific information on each phase of the process is given in the chapters which follow, each of which describes one phase of the generation.

3.1. Contents of release discs

NOS/MT is supplied on several diskettes. These diskettes are single density, single sided, IBM compatible, and are in Disc Executive format. The entire system generation of NOS/MT is performed under the Disc Executive, which avoids the "chicken and the egg problem" which would exist if NOS/MT were required to generate itself. The diskettes supplied are labeled as follows:

NOS/MT Configurator

> This diskette contains the System Generation Utility (SYSGEN), a BASIC program which is used to merge the configuration parameters for the system into the actual configuration files which will be assembled to generate the system's configuration tables. It is on this diskette that you will create the file which specifies the configuration of your system. This diskette also contains the executable programs (EDIT, BASIC, etc.) required to support the configuration process, as well as several sample configurations to serve as models for your system generations.

NOS/MT Rel & Obj

> This diskette contains all the assembled relocatables for the system, the source for the system configuration files, and the executable programs needed to assemble and link the system. The system boot file is eventually placed on this diskette.

NOS/MT Source A, B,...

> Two or more source diskettes will be included with your NOS/MT release. These diskettes include source code for all the disc, terminal, printer, and other I/O drivers currently supported by NOS/MT. If you are generating a system using the drivers supplied by Marinchip Systems, you will not need to use the source code on these diskettes. They are provided so that you can modify our drivers for your own special applications, or use them as a model for designing and implementing your own drivers for hardware we do not support.

NOS/MT Boot

> This diskette is the system disc for NOS/MT, and is, of course, in NOS/MT directory format. After you have generated your NOS/MT system, you will copy the newly generated boot file onto this disc, thus creating your production NOS/MT system disc.

3.2. Copying the release discs

As soon as you receive your NOS/MT release discs, you should make two copies of each disc. The original release discs should be stored in a safe place, and used only for "disaster backup". One of the copies you made should be labeled "Production", and the other should be labeled "Backup". In the rest of this manual we will assume that you are working with the "Production" copies you made in this step.

After you complete the system generation process, it is wise to make a new set of copies of your "Production" discs, which should be labeled "Production backup". These backups should be recopied whenever you are confident that the contents of the "Production" set are a correct new system. We encourage you to maintain as many backup copies as you need to feel secure. The Copyright on the release discs permits you to make as many copies as you wish FOR YOUR OWN USE. You may not, however, distribute these copies to any other person.

3.3. Creating a system configuration file

The most complicated part of system generation is constructing the system configuration file. This is a source file which completely specifies the software configuration of the system being generated.

To begin this process, bring up your system under the Disc Executive as you normally do. Then remove the Disc Executive system disc from Drive 1 and replace it with the disc labeled "NOS/MT Configurator". Choose a name for the system you are about to generate of the form:

SP-<name>

and create a configuration file of that name with the command:

```
CREATE SP-<name>.150
```

For example, if you wished to call your system "MINE", you would use the command:

```
CREATE SP-MINE.150
```

The "SP" stands for Site Parameters, and protects against the name you choose duplicating any of the standard files on the Configurator disc.

Now you get to choose whether you want to build the configuration for your system from scratch, or modify one of the sample configurations we provide on the Configurator disc. To start with a blank configuration, enter the command:

```
BCOPY SP-<name>=GENPAR.MOD
```

To start with one of the sample configurations, use the command:

```
BCOPY SP-<name>=<sample>
```

where <sample> is the name of the sample configuration file you wish to use as a model for your system. The sample configurations we presently supply are as follows:

SC-TAR1B

This is a one memory bank (64K) system using the Tarbell floppy disc controller with dual Shugart-type discs, and a terminal connected via the PROM/RAM board.

SC-TAR2B

This is a two memory bank (128K) system using the Tarbell controller with dual Shugart discs, and a terminal connected via the PROM/RAM board.

SC-TTK1B

This is a one memory bank (64K) system using the Teletek floppy disc controller with dual Shugart discs, and a terminal connected via the PROM/RAM board.

SC-TTK2B

This is a two memory bank (128K) system using the Teletek floppy disc controller with dual Shugart discs, and a terminal connected via the PROM/RAM board.

SC-PHXTTK2B

This is a two memory bank (128K) system using the Control Data Phoenix hard disc drive configured on the Konan SMC-100 controller as disc units 1 and 2, and two Shugart floppy disc drives configured on the Teletek controller as disc units 3 and 4. The terminal is connected via the PROM/RAM board. This configuration is representative of a minimal hard disc system.

SC-MSEXP

This is the configuration of the production NOS/MT system at the Marinchip development center. This configuration varies constantly, but is a good illustration of how a "full blown" system is configured.

3.4. Editing the configuration file

Once the configuration file has been created and initialised with either a blank configuration or one of the sample configurations, you must edit it to insert and/or change parameters for your specific system. To edit the file, enter the command:

```
EDIT SP-<name>
```

and use the text editor as you would on any other file. The configuration file is divided into three sections, each delimited by a comment with percent signs (%) in column one. The next three chapters of this manual will discuss the contents of these three sections of the configuration file.

4. Configuration parameter section

The first section in the configuration file is the "Configuration parameter section". This section consists of assembly language EQU statements which specify various parameters in the system. This section is updated by editing the EQU statements and inserting the desired value in the operand field (starting in column 21) of each statement. Of course, if the value already specified is correct for your configuration, you may leave it unchanged.

The variables in the configuration parameter section are discussed in the paragraphs that follow.

4.1. NPRIOR

NPRIOR specifies the highest software process priority used in the system. Unless you are implementing extensions to the system itself which use the internal FORK$ primitive, leave this parameter as supplied by Marinchip Systems.

4.2. NUSERS

NUSERS is the total number of users to be supported by the system, including background batch spaces. For a single user system, set NUSERS to 1. For two users and one background batch space, set NUSERS to 3.

4.3. NINTLV

NINTLV is set to the highest vectored interrupt priority used, plus 1 (interrupt priorities are numbered from zero). If your system uses interrupt priorities 1, 3, and 5, you would set NINTLV to 6, because that is one higher than the largest priority number used (that is, 5). No harm is done if NINTLV is set higher than necessary.

4.4. NPROC

NPROC specifies the number of processes each user may create. Since this release of NOS/MT does not provide the FORK$ JSYS for user process creation, NPROC should be set to 1. Setting it greater than 1 only wastes memory space.

4.5. BSIZE

BSIZE specifies the size of the system's internal buffer pool. For most systems, the standard size of 01000 (4K bytes) is correct.

4.6. DTBUFL

DTBUFL specifies the size of the buffer in common space used to move data to and from user space. This buffer must be a multiple of 16 bytes. The larger this buffer is, up to the value of BLOCKBSZ (see below), the faster the system runs. Making the buffer too large may use up so much common space that user space is reduced. Generally, 256 is the optimum value for DTBUFL.

4.7. TICKSC

TICKSC specifies the number of interrupts generated per second by the real time clock configured on the system. When using the real time clock on the PROM/RAM board, TICKSC should be set to 42. When using another real time clock board, set TICKSC to the number of interrupts that board generates per second. If TICKSC is set incorrectly, the system clock will run too fast or too slow.

4.8. SLICE

SLICE specifies how often control of the processor is to be rotated among user processes competing for its use. Making SLICE smaller switches the processor more often and increases system responsiveness, but at the price of increased overhead. We have found that about three switches per second is about right, so set SLICE to the value of TICKSC divided by 3 (truncate any remainder). In systems with more than three users, you might set SLICE a bit smaller to provide quicker turnaround for small requests. Note that SLICE is relevant only to completely compute-bound processes, since when a process gives up control by a JSYS, other user processes get a chance to run.

4.9. PSYNCT

PSYNCT specifies how often, in seconds, the system will perform an automatic SYNC$ operation to write all changed data back to the discs they were originally read from. This "periodic sync" ensures validity of data stored on the disc, even though the system buffers data in memory to minimise physical I/O. In a normal multi user system, PSYNCT should be set between 15 and 60. In single user systems utilising non-buffered disc controllers such as the Tarbell, the periodic sync may be annoying because it generates I/O at random times, which causes the CPU to momentarily lock up and hence to stop responding to the keyboard. Thus, in systems using the Tarbell controller, PSYNCT may be set to zero, turning off the periodic sync. The system will then perform a SYNC$ whenever the system command prompt (:) appears, guaranteeing data validity so long as the user returns to command mode periodically.

4.10. TRMOBUF

TRMOBUF determines how far a program generating output for a terminal can get ahead of the terminal before it is deactivated to wait for the terminal to catch up. This is normally set to 3. Making it larger decreases overhead, but consumes more system buffer space. The number given specifies how many lines of output may be queued to the user terminal before the queuing program is deactivated.

4.11. TRMIBUF

TRMIBUF specifies how many lines of input a user may "type ahead" on a terminal without the lines having been read by the system or a program before the terminal will go into the input lock state (where carriage return is not accepted until the program catches up or Control C is used). This is normally set to 3. If greater typeahead is desired, this may be increased, but you may also have to make the system buffer pool larger to hold the additional queued lines.

4.12. MAXOF

MAXOF specifies the maximum number of files each user is permitted to have open at once. This count refers only to disc files, not device files. This is normally set to 10, and while it may be increased, it will consume more buffer space. If you make MAXOF larger, you will also probably have to increase NMEMI (see below) to accomodate the memory items for the additional open files.

4.13. NBLOCKS

NBLOCKS specifies the number of file paging buffers used. The system performs I/O to the disc via these paging buffers, and increasing the number of buffers will reduce the amount of physical I/O done, and hence increase system performance. Making NBLOCKS larger increases the size of the operating system, of course, but does not consume common space. The standard value of NBLOCKS is 5, which is the minimum practical setting of this parameter.

4.14. BLOCKBSZ

BLOCKBSZ specifies the size of the file paging buffers. This must be set to the largest device block configured on the system. For systems with only floppy discs, BLOCKBSZ is set to 256. For systems with hard disc, it must be set to the block size chosen for the hard disc, assuming it is larger than 256. For the Control Data Phoenix or Morrow Discus hard discs, BLOCKBSZ should be set to 512.

4.15. NMEMI

NMEMI specifies the number of buffers reserved for file index table items for open files. This parameter should be set to the maximum number of UNIQUE files expected to be concurrently open at any time. (If one file is opened by five users, it only uses one buffer.) The most conservative setting of NMEMI is the computed from the other parameters as follows:

NUSERS*(MAXOF+SCRMAX+1)

which will guarantee that the system never runs out of buffers. Such a setting uses a lot of memory, however, so if you're willing to bet on "averaging" between users, you can set it to about 5 to 7 per user and hardly ever run out. Running out simply makes users unable to open new files until somebody closes one, and doesn't crash the system.

4.16. MAXPAR

MAXPAR specifies the maximum number of program parameters the system will scan for the user when it builds the parameter table after loading a user program. This is specified only to determine the size of the parameter index table, and may be set to any reasonable value. The standard value of 10 is reasonable, as is 20. 20000 is not reasonable.

4.17. SCRMAX

SCRMAX specifies the maximum number of concurrently open SCRIPT files each user may have. When SCRIPT files are nested to this limit, further SCRIPT$ requests will not be accepted until an open SCRIPT is completed. Open SCRIPT files use NMEMI buffers and BSIZE space, so these parameters should be adjusted if you increase SCRMAX far beyond its standard value of 3.

4.18. CPUCLK

CPUCLK specifies the clock rate of the CPU in megahertz. Most M9900 CPUs operate at 2 Mhz, so the standard value of this parameter is 2. If your CPU has been upgraded to 3 Mhz, CPUCLK should be set to 3.

5. Configuration table section

The second section in the configuration file is the "Configuration table section", in which the actual configuration code for the system is defined. Entries in this section consist of both data tables and executable code. This section specifies many different tables, each of which will be discussed in a separate paragraph below. Each table will be identified in the configuration file by a comment of the form:

```
. * <table description> goes here
```

before the table, and the comment:

```
. * end
```

after it. The entries you make in the table, as described by the following paragraphs, are inserted between these two comments. The following section titles are identical to the comments in the configuration file which identify the start of the tables, and occur in the same order as the corresponding tables in the configuration file.

5.1. Terminal pointers go here

Following this comment, insert one line for each terminal to be configured on the system. Each line is of the form:

```
DATA      TRM<term number>
```

For example, in a two terminal system, you would insert:

```
DATA      TRM1
DATA      TRM2
```

You must make an entry in this table for every configured user, even if the user is a background batch space.

5.2. Storage unit pointers go here

Following this comment, insert one line for each configured storage unit. A storage unit is normally a disc drive, but may be a memory file storage unit or other kind of storage. Each line is of the form:

```
DATA      SU<drive number>
```

To configure two disc drives, you would insert:

```
DATA      SU1
DATA      SU2
```

5.3. Memory bank pointers go here

Following this comment, insert one line for each configured USER memory bank. THE SYSTEM MEMORY BANK DOES NOT APPEAR IN THIS TABLE. There must be as many entries in this table as there are memory banks configured (normally one bank for each user, plus any banks to be used

as memory file storage units). Each line is of the form:

```
        DATA      MB<bank number>
```

For example, in a 192K system (three 64K boards) with the first board assigned to the system, and the second two boards assigned to users (hence a two user system), you would insert:

```
        DATA      MB1
        DATA      MB2
```

5.4. Terminal control tables go here

Following this comment you must insert the terminal control tables. One terminal control table must be defined for each entry made in the "terminal pointer" table above. The terminal control table definition is different for each type of serial I/O board used to interface the terminal. We will give sample configurations for each board type below. In the examples, "x" will stand for the terminal number. Hence, for the first terminal, 1 would replace the "x" in the example.

5.4.1. Marinchip PROM/RAM and Quad SIO board

A terminal control table for a port on the PROM/RAM board or the Marinchip Quad SIO board is configured as follows:

```
TRMx      DATA      TRM902,TRMxD,TRMxIB,TRMxIBE,TRMxEB
          DATA      TRMxEBE,ILLOGIN
          BSS       CHBSS
TRMxD     DATA      <addr>,<modes>,<baud>
TRMxIB    BSS       80
TRMxIBE   .
TRMxEB    BSS       20
TRMxEBE   .
```

In the above table, <addr> should be replaced with the CRU address jumpered on the board multiplied by two. For the first terminal, supported by the PROM/RAM board, use 0. For ports on the Quad SIO board, double the sum of the board base address and the port offset, and use the value as the <addr> field. For example, if the Quad SIO were set for a base address of 080, and port 2 were being configured as a terminal, <addr> should be set to 2*(080+2*020), or 0180. The <modes> field should be set to 0AA00 for a 2 Mhz CPU and 0A200 for a 3 Mhz CPU. <baud> should be set to the terminal baud rate. To use the automatic baud rate sensing on the terminal connected to the PROM/RAM board, set <baud> to zero. Otherwise, and for all other terminals, set the desired baud rate (for example, 9600).

When using very fast baud rates (9600 and above) it is frequently more efficient to run the output to terminals in non-interrupt mode, because the overhead in processing an interrupt is greater than one character time on the terminal. A terminal may be configured to transmit output in non-interrupt mode by adding 1 to the value in the <modes> field. Hence, for non-interrupt output with a 2 Mhz CPU, the <modes> field would contain 0AA01. Input from a terminal always runs in interrupt mode, so the interrupt jumper on the board must still be connected and the interrupt configured in the system even if this mode

is selected. Selecting this mode for terminals running below 9600 baud will degrade the efficiency of the system by reducing the time available to run user tasks. The operation of the terminal is unaffected by setting non-interrupt mode; only the system's method of driving the terminal is changed.

Some terminals provide a hardware ready signal to cause the computer to stop sending data when the terminal cannot accept it. This facility is needed by terminals with features such as smooth scroll or lengthy command sequences which cannot be executed as fast as the computer can send data. NOS/MT supports such terminals ONLY in the non-interrupt output mode described above. To select non-interrupt output with a ready signal, add 3 to the value in the <modes> field. For example, a terminal with non-interrupt output and a ready line would specify 0AA03 in the <modes> field (for a 2 Mhz CPU). The terminal's ready signal should be wired to the Data Terminal Ready input of the port to which the terminal is connected. The system assumes that the ready signal is high true.

5.4.2. IMSAI SIO2-2

For each terminal configured on an IMSAI SIO2-2, use the following table:

```
 TRMx      DATA      SIO2,TRMxD,TRMxIB,TRMxIBE,TRMxEB
           DATA      TRMxEBE,ILLOGIN
           BSS       CHBSS
 TRMxD     DATA      <addr>,<intr>
 TRMxIB    BSS       80
 TRMxIBE   .
 TRMxEB    BSS       20
 TRMxEBE   .
```

Where <addr> should be set to the M9900 I/O address of the desired port on the SIO2-2. This is computed by adding 0F000 to two times the port base address. For example, if the address of the SIO2 was jumpered to 8080 I/O address 0, the <addr> setting for Port A on the board would be 0F004 and the setting for Port B would be 0F008. The <intr> parameter is set to enable interrupts on ports being used. If just Port A is used, set <intr> to 1. If just Port B is used, set <intr> to 010. Use 011 if both Port A and Port B are used. Note that if both Port A and Port B are being used, a separate terminal control table is supplied for each port, but the base port addresses given in the field <addr> differ. The <intr> field must be the same for both ports, though.

5.4.3. Background batch pseudo-terminals

For each background batch space configured, a corresponding pseudo-terminal must be configured. This terminal is listed in the terminal pointer list, and its control table is configured as follows:

```
 TRMx      DATA      0,0,0,80,0,0,0
           BSS       CHBSS
```

5.5. Real time clock table goes here

Following this comment you insert the control table for the real time clock. If you are using the real time clock on the PROM/RAM board, and that board also is being used to drive a terminal, use the following:

```
RTCT*     DATA      RTC902,RTCTD
RTCTD     DATA      0
          BYTE      186,1
```

5.6. Simulated interrupt code goes here

Following this comment you insert the simulated interrupt code which will be executed on every real time clock interrupt. Such code need be inserted only if you are using the Tarbell controller. If a Tarbell controller is configured, insert the following code:

```
          LI        R7,DSK1T
          LI        R8,DTRBL
          MOV       BVINT(R8),R8
          BL        *R8
```

If you are using one of the other disc controllers, no code is inserted, as they use hardware interrupts and do not require interrupt simulation.

5.7. Clock/calendar table goes here

Following this comment you insert the control table for the clock/calendar board if one is to be used with the system. If no clock/calendar board is present, nothing should be inserted here. The CLK-24 board, available from Marinchip Systems, maintains the time and date even when the power fails or the system is turned off, and allows NOS/MT to know the current date and time without having to enter them whenever the system is loaded. If the CLK-24 is to be used with the system, use the following:

```
          DATA      YRCL1
YRCL1     DATA      <addr>,CLK24,<model>
          BSS       12
```

Where <addr> should be set to the M9900 I/O address which corresponds to the port address set in the DIP switches on the CLK-24 board. This is computed by adding 0F000 to two times the port base address. If the CLK-24 board is set to the standard address of 0F0, <addr> should be set to 0F1E0. The field <model> should be set to 1 if the CLK-24 board is revision B or later and to 0 if the CLK-24 board is revision A or earlier.

5.8. Storage unit tables go here

Following this comment you insert storage unit tables which correspond to the entries you made in the storage unit pointer table at the beginning of CONFIG. The exact format of these tables differs for each controller type, so insert the proper set of tables according to the type of controller you are using. This sample configuration is for two drives. For other configurations, contact us for assistance.

5.8.1. Tarbell disc controller

Insert the following for a two drive system:

```
 SU1        DATA       256,1,1,2,12,DTRBL,DSK1T,1
            DATA       +(26*77)/2,256,+(26*77)/2
            DATA       0,0,0
            BSS        10
            BSS        DINL
 SU2        DATA       256,1,1,2,12,DTRBL,DSK1T,2
            DATA       +(26*77)/2,256,+(26*77)/2
            DATA       0,0,0
            BSS        10
            BSS        DINL
 DSK1T      DATA       0F1F0,<step>,D1UT,2,0F420
            BSS        56
 D1UT       DATA       D1UT1,D1UT2
 D1UT1      DATA       0,0
 D1UT2      DATA       1,0
```

In the above table, <step> should be set to 0 for a 6 millisecond step rate on the drives, to 2 for an 8 millisecond rate, and to 3 for a 10 millisecond step rate. Most modern disc drives will operate correctly with the 6 millisecond step rate. When using a PerSci disc drive, <step> must be set to -1. This sets the step rate to 6 milliseconds and informs the software driver that the two disc heads are driven by a common positioner.

5.8.2. Teletek FDC-II

For a two drive system, insert the following:

```
 SU1        DATA       256,1,1,2,12,TTKFDC,DSK1T,1
            DATA       +(26*77)/2,256,+(26*77)/2
            DATA       0,0,0
            BSS        10
            BSS        DINL
 SU2        DATA       256,1,1,2,12,TTKFDC,DSK1T,2
            DATA       +(26*77)/2,256,+(26*77)/2
            DATA       0,0,0
            BSS        10
            BSS        DINL
 DSK1T      DATA       0F0C0,<step>,<buffer>,D1UT,2
            BSS        66
 D1UT       DATA       D1UT1,D1UT2
 D1UT1      DATA       <twoheads>+0,0
 D1UT2      DATA       <twoheads>+1,0
```

The parameter <step> should be set to the step rate in increments of one millisecond (16 milliseconds maximum), or to -1 if the PerSci drive is being used. The parameter <buffer> specifies the address of the FDC-II's RAM buffer. This may be either 0F400 (when the RAM buffer on the PROM/RAM board is disabled) or 0EC00 if the buffer is placed at that location to avoid modifying the PROM/RAM board. The parameter <twoheads> is 0 if the disc drive being used is single sided and 0100 if the disc drive is double sided (such as the Qume DT-8 or the Shugart 851). If <twoheads> is set to 0100, the software driver

will sense whether a single or double sided diskette is installed and adjust automatically to the amount of storage available. When single sided drives are being used, <twoheads> should be set to 0 because many single sided drives do not properly generate the status line which tells the controller whether a single or double sided diskette is mounted and erratic operation may result.

5.8.3. Konan SMC-100

To configure the Control Data Phoenix hard disc on the Konan SMC-100 controller, use the following:

```
SU1       DATA      512,1,7,8,500,KONSMC,HDSK1T,1
          DATA      25600,512,25600
          DATA      0,0,0
          BSS       10
          BSS       DINL
SU2       DATA      512,1,7,8,500,KONSMC,HDSK1T,2
          DATA      25600,512,25600
          DATA      0,0,0
          BSS       10
          BSS       DINL
HDSK1T    DATA      0F120,HD1UT,2,256,4
          BSS       48
HD1UT     DATA      HD1UT1,HD1UT2
HD1UT1    DATA      0,010,64,1,800
HD1UT2    DATA      0,0,64,1,800
```

The above table configures the fixed disc as unit 1 (SU1) and the removable cartridge as unit 2 (SU2). To reverse this assignment, interchange the labels HD1UT1 and HD1UT2.

Most systems using a hard disc will also have floppy drives configured. If this is the case, insert the appropriate floppy configuration storage unit table following the hard disc table, changing the labels SU1 and SU2 in the floppy table to SU3 and SU4. Also remember to insert DATA statements for SU3 and SU4 in the storage unit pointer table. In systems using the Phoenix drive, BLOCKBSZ in the configuration parameter section must be set to 512.

5.8.4. Morrow Designs Discus M10 and M26

To configure the Morrow Designs 26-Megabyte disc, use the following:

```
SU1       DATA      512,1,13,14,600,DISCUS,MDSC1T,1  26-Mbyte Discus
          DATA      0CA00,512,0CA00
          DATA      0,0,0
          BSS       10+DINL
MDSC1T    DATA      0F0A0,MDSC1UT,1
          BSS       48
MDSC1UT   DATA      MDSC1UT1
MDSC1UT1  DATA      0,32,8,202,0
```

To configure the 10-Megabyte disc, use the following:

```
SU1       DATA      512,1,6,7,225,DISCUS,MDSC1T,1  10-Mbyte Discus
          DATA      20496,512,20496
```

```
           DATA      0,0,0
           BSS       10+DINL
MDSC1T     DATA      0F0A0,MDSC1UT,1
           BSS       48
MDSC1UT    DATA      MDSC1UT1
MDSC1UT1   DATA      0,21,4,244,0
```

For multiple discs on one controller, there must be additional SU and MDSC1UT entries. For example, a 26-Megabyte disc on unit 1 and a 10-Megabyte on unit 2:

```
SU1        DATA      512,1,13,14,600,DISCUS,MDSC1T,1  Unit 1: 26-Mbyte
           DATA      0CA00,512,0CA00
           DATA      0,0,0
           BSS       10+DINL
SU2        DATA      512,1,6,7,225,DISCUS,MDSC1T,2    Unit 2: 10-Mbyte
           DATA      20496,512,20496
           DATA      0,0,0
           BSS       10+DINL
MDSC1T     DATA      0F0A0,MDSC1UT,2      2 units in table
           BSS       48
MDSC1UT    DATA      MDSC1UT1,MDSC1UT2
MDSC1UT1   DATA      0,32,8,202,0         port 0 on controller
MDSC1UT2   DATA      1,21,4,244,0         port 1 on controller
```

BLOCKBSZ in the configuration parameter table must be set to 512 (or a higher power of 2).

Most systems using a hard disc will also have floppy drives configured. If this is the case, insert the appropriate floppy configuration storage unit table following the hard disc table, changing the labels SU1 and SU2 in the floppy table to SU2 and SU3 - or to the lowest numbers available after all hard disc units have been configured. Also remember to insert DATA statements in the storage unit pointer table for all storage units SU1 to SU3 (or whatever the last one is).

The NOS/MT software for the Discus assumes that all sectors on the disc have a storage key of 0. Discs bought from Marinchip Systems will always be formatted in this way. If you buy the discs directly from Morrow Designs, be sure to use the Marinchip formatting routine before attempting to use the discs under NOS/MT, since Morrow's formatting puts a key of hex 80 on all of cylinder 0.

5.8.5. Memory files

NOS/MT allows one or more memory banks to be configured as a storage unit. This storage unit works exactly like a fixed volume disc drive (except, of course, that data stored on it is lost when power is removed from the system), but runs at memory speed rather than disc speed. Hence, memory files are ideal when extremely fast access to moderately large amounts of data is required.

To configure a memory file storage unit, the memory banks which make up the unit should be configured exactly like memory banks used as user spaces. Hence, the entries made in the memory bank pointer table, user memory bank table, and memory bank enable and disable

routine section are as described for user memory banks. The memory banks used for memory files must be placed after all user memory banks in the memory bank pointer table.

Assuming the memory banks have been configured, the memory file storage unit is configured by the following table:

```
 SUx        DATA       0,0,0,0,<files>,MEMFIL,MEMF1T,1
            DATA       0,0,0,0,0,0
            BSS        10
            BSS        DINL
 MEMF1T     DATA       M1UT1,<banks>
 M1UT1      DATA       <bank1>,<bank2>,...
```

The parameter <files> should be set to the maximum number of files (including directories) you wish to be able to create on the memory file storage unit. <banks> specifies the number of memory banks making up the storage unit (1 or more), and <bank1>, etc., are the labels from the memory bank tables (see below) identifying the memory banks which make up the storage unit.

A memory file storage unit is exactly like any other storage unit; it must be PREPed before first being used, and all file security rules which apply to disc units apply to memory files as well. Any number of memory file storage units may be configured in the system; one need only specify multiple storage unit tables and divide the memory banks available for memory files among them. It is normally more efficient, however, to simply configure one storage unit composed of all memory available for such use, as allocation of space can be done with less waste when all space is allocated from one common pool.

5.8.6. Alpha Micro AM-200

Please contact Marinchip Systems for configuration information. The AM-200 configuration requires very careful planning of the overall system memory map, and we'd prefer to give you a custom configuration after looking at your system as a whole. If you write, please list all memory boards you intend to use in the system, both their type and address range, and your intended final memory map. Also please indicate the type of drives you're now using on the AM-200 (PerSci or Wangco), and how your existing Disc Executive is configured.

5.9. User memory bank tables go here

Following this comment you insert tables which describe the memory banks configured for user and background batch spaces or for memory files. These are the tables referenced in the "memory bank pointer" table which was described previously. The format of each table is as follows:

```
 MBx        DATA       <size>,ENMBx,DSMBx
```

where "x" is the bank number (1, 2, etc.), and <size> is the size of the user space in bytes. For the system memory map given as an example in the hardware configuration section where there were two user spaces of 56K bytes each (hexadecimal 0E000), the memory banks would be entered in this table as follows:

```
MB1         DATA        0E000,ENMB1,DSMB1
MB2         DATA        0E000,ENMB2,DSMB2
```

The system memory bank is not described in this table. Memory banks used for memory files should be placed after the last memory bank used as a user or background batch space.

The following describes configuration of a user memory space which occupies the same physical memory bank as the system. This is not the normal case, and to do this you must first go through the entire system generation, determine how much free space there is in the system memory bank, then go back and set the size of the space accordingly. Let us assume that there are 24K bytes of free space in the system's memory bank. When the system is LINKed, it will be placed as high as possible in memory so that there will be 06000 hexadecimal free bytes from 0000 to 05FFF. This space will be allocated to a user memory bank. This would be accomplished by the following table entry:

```
MBx         DATA        06000,SYSMAP,SYSDMAP
```

Note that programs executed in this space may not exceed 24K. It is not required that the free space in the system bank be configured as a user space. We recommend that you first bring up the system without such space, then configure it later after you're more confident doing system generations.

5.10. Printer control tables go here

If any printers are to be configured, table entries must be inserted following this comment. One entry must be made for each printer. The format of the table entry differs depending upon the type of I/O board used to interface the printer, but most of the fields in the printer table are the same. The following paragraphs describe the constant fields in the table, and the subsequent sections explain how to actually enter the printer table for each interface type.

Printer Configuration Fields

<ffsim> — Set to 0 if the printer responds to form feed characters. Set to 1 if the system must simulate form feeds by sending line feeds.

<autolf> — Set to 1 if carriage return does a line feed on the printer. Set to 0 if line feeds must be sent.

<crdly> — If delay characters must be sent following a carriage return, set <crdly> to the number of delay characters required.

<lfdly> — If delay characters must be sent following a line feed, set <lfdly> to the number of delay characters required.

<ffdly> — If delay characters must be sent following a form feed, set <ffdly> to the number of delay characters required.

<minl> If there is a minimum output line length for this printer (e.g., Memorex 1240) set <minl> to the line length required (41 for Memorex, 0 for real printers).

<autops> Set to 1 if the printer automatically skips the page perforation. Set to 0 if the system should count lines and do this.

<topm> Set to the size of the desired top margin in lines. Ignored if <autops> is nonzero.

<body> Set to the number of lines of printable page body desired. Ignored if <autops> is nonzero.

<botm> Set to number of lines in the bottom margin on the page. Ignored if <autops> nonzero. The sum of <topm>, <body>, and <botm> must equal the number of lines on the paper used in the printer.

5.10.1. Printer on Marinchip Quad SIO

When configuring a printer on the Marinchip Quad SIO, use the following table:

```
 PRx      DATA     PR9902,PRxT,<ffsim>,0,<autolf>,<crdly>
          DATA     <lfdly>,<ffdly>,<minl>,<autops>
          DATA     <topm>,<body>,<botm>
          BSS      PFBSS
 PRxT     DATA     <addr>,0A200+0800*(1-(CPUCLK=3))
          DATA     <baud>,<busy>
```

In this table "x" is replaced by the printer number (1, 2, etc.). The parameters discussed above are set as their descriptions indicate. <addr> should be set to the CRU address of the printer port multiplied by two. For example, for port 1 of a Quad SIO board with CRU base address of 080, <addr> would be set to 2*(080+1*020), or 0140. The parameter <baud> should be set to the baud rate at which the printer is configured to accept data, for example, 9600. <busy> is set nonzero if the printer has a busy signal returned on the Data Terminal Ready line by which it causes the CPU to stop sending data. Such printers, like the Texas Instruments 810, require that <busy> be set to 1. For printers that do not have a busy line, set <busy> to zero. The above configuration may also be used for printers connected via the Marinchip Printer Interface, which was manufactured prior to the introduction of the Quad SIO board.

5.10.2. Printer on IMSAI SIO2-2 board

When configuring a printer on the IMSAI SIO2 board, use the following table:

```
 PRx      DATA     PRSIO2,PRxT,<ffsim>,0,<autolf>,<crdly>
          DATA     <lfdly>,<ffdly>,<minl>,<autops>
          DATA     <topm>,<body>,<botm>
          BSS      PFBSS
 PRxT     DATA     <addr>,<intr>
```

In the table, "x" is replaced by the printer number (1, 2, etc.). The parameters discussed above that are common to all printers are set as required by the printer hardware. <addr> should be set to 0F000 plus two times the 8080 I/O address of the data port for the port on the SIO2 the printer is connected to. For example, assume the SIO2 base address is jumpered to 010 hex and that the printer is connected to Port A on the SIO board. The data port offset for Port A is 2 (from the IMSAI manual), so the expression for <addr> becomes:

0F000+2*(010+2)

or 0F024. The parameter <intr> should be set to 1 if the printer is connected to Port A on the SIO2 and 010 if the printer is connected to Port B. The SIO2 printer driver supplied with NOS/MT requires that the printer be the only port used on the SIO2 board that connects the printer. Because of the way interrupts work on the SIO2, removing this restriction would enormously complicate the configuration process since it would introduce an interaction between the terminal and printer drivers over interrupt modes on the board.

No special software configuration is required when using the SIO2 to connect printers like the Texas Instruments 810 because the busy signal from the printer has the correct polarity to be wired directly to the Clear To Send input on the SIO2. Hence, when the printer is busy, the computer is hardware inhibited from sending characters to it.

5.10.3. Printer on non-interrupt board

It is highly desirable that the printer in an NOS/MT system be run on an interrupt driven interface like those described above. There are, however, many existing boards which do not properly support interrupt operation, and NOS/MT contains a driver which allows them to be used to drive a printer in non-interrupt mode. Because the printer is not driven in interrupt mode, user programs will not be able to use the processor while the printer is running; this will degrade overall system performance. If this degradation is excessive, the only alternative is to replace the printer board with an interrupt-driven board.

To configure a non-interrupt printer on the general driver for such boards, use the following table:

```
PRx     DATA    PRNONI,PRxT,<ffsim>,0,<autolf>,<crdly>
        DATA    <lfdly>,<ffdly>,<minl>,<autops>
        DATA    <topm>,<body>,<botm>
        BSS     PFBSS
PRxT    DATA    <addr>,<statport>,<dataport>
        DATA    <ordymask>,<ordyvalue>
```

In this table, "x" is replaced with the printer number (1, 2, etc.). The parameters discussed above that are common to all printers are set as required by the printer hardware. <addr> should be set to 0F000 plus two times the 8080 I/O address of the base address of the board. <statport> should be set to two times the offset of the status port from the board base address. <dataport> should be set to two times the offset of the data port from the board base address. <ordymask>

should have the the bit set which the board uses to indicate that the transmitter can accept another character of output. If the transmitter ready bit is "high true" (that is, it is one when output may be sent) <ordyvalue> should be set equal to <ordymask>. If the transmitter ready bit is "low true" (zero when output may be sent), <ordyvalue> should be set to zero.

5.11. Printer offline access tables go here

Following this comment you insert one control table for each printer for which an offline access (spool) file is desired. The format of the table is as follows:

```
OFFx    DATA    PRy,<maxfiles>,<buflen>
        BSS     30
```

where "x" is the number of the offline file (1, 2, etc.), "y" is the number of the printer this offline file prints on (and hence the label, "PRy", is the label on the printer control table for that printer), <maxfiles> is the maximum number of files which may be queued to the offline file before it rejects new requests until the printer catches up, and <buflen> is the size of the I/O buffer used to read the file being printed. We recommend that <maxfiles> be set to 15, and <buflen> set to 128. If either <maxfiles> or <buflen> is increased, BSIZE in the configuration parameter section may have to be increased to provide the additional space required. <buflen> should only be made larger if the printer runs too slowly with the recommended value. A typical offline access table is given below.

```
OFF1    DATA    PR1,15,128
        BSS     30
```

5.12. Background batch control tables go here

Following this comment you must insert one control table for each background batch space to be present in the system. If no background batch spaces are configured, nothing should be inserted here. In the earlier section titled "background batch pseudo-terminals" it was explained how to configure the pseudo-terminals used to pass data to the background batch spaces. For each pseudo-terminal configured, one batch control table should be placed here. The format of each batch control table is as follows:

```
BATx    DATA    TRMy
        BSS     BBFBSS
```

where "x" stands for the number of the batch space (1, 2, etc.), and "y" stands for the number of the corresponding pseudo-terminal previously configured. For example, if TRM3 was the label on the pseudo-terminal control table, and we were configuring one background batch space using it, we would insert:

```
BAT1    DATA    TRM3
        BSS     BBFBSS
```

5.13. Special file name tables go here

Following this label you insert entries that describe the names used to access the printers, background batch spaces, printer offline access (spool) files, interterminal message facilities, and the system status features configured in the system. If none of these features are present in the configuration, nothing need be inserted here. To construct the entries for this table, first prepare a list of all the special files you wish to configure, and assign names to them. Following the recommended names given in the user guide, we might come up the the following list:

```
PRINT.DEV
PRINT2.DEV
PRINT.OFF
BATCH.DEV
TERM1.DEV
SYSTAT.DEV
```

This would be for a system with two printers, one of which can be accessed through the offline access mechanism, and one background batch space. The name TERM1.DEV is to be configured to allow other users to send messages to the first terminal configured in the system, and the standard name SYSTAT.DEV is to be used to access the system status facility. Any desired names may be chosen, up to a maximum of 14 characters. Next, assign descending hexadecimal values to each name in the table, starting with OFE for the first. At this point, the table would be:

```
PRINT.DEV     OFE
PRINT2.DEV    OFD
PRINT.OFF     OFC
BATCH.DEV     OFB
TERM1.DEV     OFA
SYSTAT.DEV    OF9
```

Now associate the names with the labels on the printer, printer offline access, and background batch control tables you've previously entered. For interterminal messages, use the label on the terminal control table for the terminal to which messages are to be directed. For the system status device, use 0. Enter these labels in the table. Now you might have:

```
PRINT.DEV     OFE    PR1
PRINT2.DEV    OFD    PR2
PRINT.OFF     OFC    OFF1
BATCH.DEV     OFB    BAT1
TERM1.DEV     OFA    TRM1
SYSTAT.DEV    OF9    0
```

Now write the "driver type" after each item. This is:

```
SPFPRT        If the device is a printer
SPFAPQ        If the device is a printer offline access file
SPFBAT        If the device is a background batch space
SPFITM        If the device is interterminal message facility
SPFSTA        If the device is the system status facility
```

After this step, we have in the table:

```
PRINT.DEV     OFE    PR1     SPFPRT
PRINT2.DEV    OFD    PR2     SPFPRT
PRINT.OFF     OFC    OFF1    SPFAPQ
BATCH.DEV     OFB    BAT1    SPFBAT
TERM1.DEV     OFA    TRM1    SPFITM
SYSTAT.DEV    OF9    0       SPFSTA
```

Now make the actual entries in the table, one for each line in the list you've just made, according to the following format:

```
DATA      <length>,XSNSFx,<code>,<table>,<driver>
```

where <length> is the length of the name, in characters, <code> is the hexadecimal code, <table> is the corresponding control table name, and <driver> is the driver type. Given the sample configuration we've been working with, we would insert the following lines:

```
DATA      9,XSNSF1,OFE,PR1,SPFPRT
DATA      10,XSNSF2,OFD,PR2,SPFPRT
DATA      9,XSNSF3,OFC,OFF1,SPFAPQ
DATA      9,XSNSF4,OFB,BAT1,SPFBAT
DATA      9,XSNSF5,OFA,TRM1,SPFITM
DATA      10,XSNSF6,OF9,0,SPFSTA
```

Note that the "x" field in the table format is replaced with the number of each line in the table. The resulting name is used to associate the table entry with the actual name, inserted in the next step.

If you wish to assign multiple names (aliases) to special files, you may do so by adding additional entries in the special file name table. The entries for aliases MUST FOLLOW ALL NORMAL ENTRIES IN THE TABLE. The alias entries should duplicate all fields of the primary name except for the <length> and XSNSFx pointer, which should refer to the name of the alias.

5.14. Special file names go here

After this comment you insert the text for the names of the special files configured above. If there were no entries made in the special file table, nothing need be inserted here. Each entry here is of the form:

```
XSNSFx    TEXT      "<name>"
```

where <name> is the name of the special file from the list prepared for the previous step, and "x" is the line number in the table. Using the above example, we would insert:

```
XSNSF1    TEXT      "PRINT.DEV"
XSNSF2    TEXT      "PRINT2.DEV"
XSNSF3    TEXT      "PRINT.OFF"
XSNSF4    TEXT      "BATCH.DEV"
XSNSF5    TEXT      "TERM1.DEV"
XSNSF6    TEXT      "SYSTAT.DEV"
```

The name text in these statements MUST BE IN UPPER CASE!

5.15. Interrupt routines for level x go here

There are 7 lines with the above form, with "x" replaced with the interrupt priority from 1 to 7. Following the comment, you should insert the interrupt service routines for those devices whose interrupt lines are wired to that priority. The recommended interrupt priority assignments are as follows:

Priority	Assignment
2	Terminals, real time clock
3	Floppy disc controller
4	Printers
5	Hard disc controller

These priorities may be changed at will. The actual code which is inserted at each level depends on the type of controller. Each of the following subsections gives the code for one controller type. Be sure that where you insert the code (which interrupt level) agrees with the hardware interrupt level the controller has been wired to generate.

5.15.1. Terminals on PROM/RAM and/or Marinchip Quad SIO board

For each terminal supported by the SIO port on the PROM/RAM board or by a Marinchip Quad SIO board, use the following interrupt code:

```
LI      R2,TRM902
MOV     CVINT(R2),R1
LI      R7,TRMx
BL      *R1
```

where "x" is the terminal number, so the label corresponds to the label on the terminal control table previously configured. Note that this code is replicated for each terminal configured.

5.15.2. Terminals on IMSAI SIO2-2

For each terminal supported by an IMSAI SIO2 board, insert the following code:

```
LI      R2,SIO2
MOV     CVINT(R2),R1
LI      R7,TRMx
BL      *R1
```

where "x" is the terminal number, so the label corresponds with the label on the terminal control table previously configured. Note that this code is replicated for each SIO2-supported terminal configured.

5.15.3. Real time clock on PROM/RAM board or Quad SIO board

Insert the following interrupt code at the level for which the interrupt jumper on the PROM/RAM board is set (normally 2) when using the real time clock on the PROM/RAM board:

```
LI          R2,RTC902
MOV         RTVINT(R2),R1
LI          R7,RTCT
BL          *R1
```

5.15.4. Teletek FDC-II disc controller

Insert the following code after the comment for interrupt level 3 when using the Teletek FDC-II disc controller:

```
LI          R2,TTKFDC
MOV         BVINT(R2),R1
LI          R7,DSK1T
BL          *R1
```

Note that this code is inserted only once per controller, not once for each disc drive configured.

5.15.5. Tarbell floppy disc controller

The Tarbell controller does not generate interrupts, so no interrupt code is inserted for it.

5.15.6. Konan SMC-100 hard disc controller

Insert the following code after the comment for interrupt level 5 when using the Konan SMC-100 hard disc controller:

```
LI          R2,KONSMC
MOV         BVINT(R2),R1
LI          R7,HDSK1T
BL          *R1
```

Note that this code is inserted only once per SMC-100 board installed in a system (normally only one), rather than once for each disc drive configured.

5.15.7. Morrow Designs Discus M10-M26 Controller

Insert the following code after the comment for interrupt level 5 when using the Discus hard disc controller:

```
LI          R2,DISCUS
MOV         BVINT(R2),R1
LI          R7,MDSC1T
BL          *R1
```

Note that this code is inserted only once per controller board installed in a system (normally only one), rather than once for each disc drive configured. In the strange event that you have both a Discus and a Konan hard disc controller, both this code and the interrupt code for the Konan controller should be here.

When the controller is bought through Marinchip Systems, both interrupts are strapped to level 5. If you buy from another source, you must make sure that the interrupts are properly strapped, as explained in the hardware configuration section of this document.

5.15.8. Printers

For each printer configured, whether supported by a Marinchip board or by the IMSAI SIO2, insert the following code:

```
        LI      R7,PRx
        MOV     PFDRVV(R7),R2
        MOV     CVINT(R2),R1
        BL      *R1
```

where "x" is the printer number, corresponding to the label on the printer control table previously configured. Note that this code is replicated for each configured printer.

6. Memory bank control section

The third and final section of the configuration file is the "Memory bank control section". This section contains executable subroutines which enable and disable the memory banks configured in the system. Subroutines are provided for the memory bank containing the system and for each user memory bank.

The following will describe how to prepare the memory bank control section when using the Marinchip 64K RAM board. When using other memory boards, you must write subroutines to enable and disable them based on the description of their bank selection mechanism provided by their manufacturers.

When using the Marinchip 64K RAM, in which memory allocation is controlled entirely by software, three basic parameters must be determined based on the memory map desired for the system. These parameters, the "system memory mask", the "common memory mask", and the "user memory mask", are bit maps that identify the 4K memory pages to be enabled for the system, for common space, and for users, respectively. Each bit map is a single 16 bit word, in which each bit controls one 4K byte page. The low-order bit (1) controls the addresses 0000 to 0FFF, the next bit (2) controls 1000 to 1FFF, and so on up to the most significant bit (08000), which controls F000 to FFFF. A one bit in the mask word enables the corresponding page, and a zero bit disables it.

The system memory mask will be configured to enable all memory in the system bank, including common space. Hence, the system memory mask will have one bits for all pages not occupied by ROM or other special memories. In systems using the Tarbell controller or the Teletek controller with the data buffer at the standard address of 0F400, the system memory mask is normally 07FFF hexadecimal, which enables 60K bytes of memory (addresses 0000 to EFFF), disabling only the page occupied by the PROM/RAM board. In systems with the Teletek controller which configure the I/O buffer at 0EC00 to avoid modifications to the PROM/RAM board, the system memory mask is normally 03FFF, which enables the 56K bytes of memory from 0000 to DFFF, leaving the memory for the disc buffer and PROM/RAM board free.

The common memory mask has only the bits set to enable the page or pages containing common space. In a normal system with 4K or less of common space, this mask will have only a single bit set, which will be the most significant bit set in the system memory mask. Hence, in the Tarbell or Teletek system with the buffer at 0F400, the common memory mask would be 04000. In a Teletek-based system with the buffer at 0EC00 the common memory mask would be 02000.

The user memory mask is set to enable all memory with addresses below that of common space. Hence, its mask value has all bits set that were set in the system memory mask, except for the highest bit, which is left off to prevent a conflict with common space. For example, in the Tarbell system or Teletek system with the buffer at 0F400, the user memory mask would be 03FFF, which enables a 56K user space from 0000 to 0DFFF. In a Teletek system with the buffer at 0EC00 the user memory mask would be 01FFF, which enables the 52K user space from 0000

to OCFFF. Note that the memory area enabled by the user memory mask must agree with the user space size configured in the "User memory bank control table" which was defined back in the configuration table section.

Now having defined the three memory enable masks, make a table of the I/O addresses of the memory boards to be used in the system. The I/O address for a memory is computed from the formula:

OF200+2*(<badr>)

where <badr> is the bank select port address set in the DIP switches on the memory board. Assuming that the RAM boards are assigned the recommended bank select port addresses of 080, 081, 082, etc., the I/O addresses will be OF300, OF302, OF304, and so on.

With this information at hand, we are now ready to actually construct the memory bank control section.

6.1. Memory bank enable and disable routines go here

Following this comment, first insert statements defining the system, common, and user memory masks as follows:

```
KSYSM    DATA     <system mask>
KCOMM    DATA     <common mask>
KUSRM    DATA     <user mask>
```

where the masks are specified in hexadecimal as defined above.

Next, you must insert the subroutines which enable and disable the system memory bank. These subroutines are coded as follows:

```
SYSMAP*  MOV      KSYSM,<ioadr>
         RT
SYSDMAP* MOV      KCOMM,<ioadr>
         RT
```

where <ioadr> is the I/O address of the system memory bank control port (normal value OF300).

Following the system memory bank enable and disable routines, you must insert enable and disable routines for each separate user memory bank. The labels on these routines correspond to the labels referenced in the "User memory bank control tables" defined in the configuration table section. For each user memory bank, insert code of the form:

```
ENMBx*   MOV      KUSRM,<ioadr>
         RT
DSMBx*   CLR      <ioadr>
         RT
```

where "x" is the memory bank number (1, 2, etc.), and <ioadr> is the I/O address of the bank select port on that memory board (normally OF302 for the first, OF304 for the second, and so on).

The process of configuring these memory bank enable and disable routines may seem extremely confusing at first, but if you examine the sample configuration files provided on the Configurator disc, and also review the 64K RAM theory of operation manual, it should become more clear. Memory banks used for user spaces, background batch spaces, and memory files are configured in exactly the same way. Their use is controlled by other sections of the configuration.

7. Performing the configuration merge

At this point, you should have a complete configuration file describing your system, constructed according to the instructions given in the preceding sections. Before going any farther with the system generation, it is recommended that you list the configuration file and check it carefully. It is often useful to compare it with the most similar sample configuration file provided. If you discover a discrepancy, or are unsure about the setting of a parameter, either try to figure it out by rereading the appropriate section of this manual, or get in touch with us for an answer before trying to generate the system.

As long as everything looks O.K., you can proceed with the next step, which takes the configuration file and uses it to construct the assembly programs which will be used to configure the system. Insert the "NOS Configurator" disc in Drive 1, and the "NOS Rel & Obj" disc in Drive 2, then enter the commands:

```
BASIC
COMPILE:SYSGEN
RUN
```

The SYSGEN program will sign on and ask for a command. Enter:

```
1
```

which will select the "Merge" function. The program will then prompt with:

```
Generation parameter file name:
```

and you should enter the name of the configuration file you have just prepared (e.g., "SP-<name>"). There will be a lot of disc I/O, and finally the message:

```
Merge complete.
```

should appear, and the "Select function" prompt will be re-issued. Enter the command:

```
0
```

which will terminate the SYSGEN program, and then the command:

```
BYE
```

which will exit BASIC and return to the Disc Executive command level. If an error message appears during the execution of SYSGEN, this normally indicates an error in the configuration file. Examine the configuration file versus one of the sample files and correct the error.

8. Assembling the configuration files

Once the configuration merge is complete, remove the "NOS/MT Configurator" disc from Drive 1 and return it to its storage box. Remove the "NOS/MT Rel & Obj" disc from Drive 2 and install it in Drive 1. Then execute the following two commands:

```
ASM CONFIG.REL=CONFIG
ASM COMDATA.REL=COMDATA
```

which will assemble the two system configuration files. If you would like printed listings of these two files, append ",PRINT.DEV" to these commands. These assemblies should generate no error messages on the console, nor should there be any flags other than "U" in the printed listing of the assemblies. If there are errors, the most likely cause is an incorrect statement in one of the configuration tables. Check for the error line in your configuration file, correct it, and go back and redo the configuration merge and assembly steps.

9. Linking the system

At this point, all that remains is to link the system modules into the executable boot file. The link process is complicated by the fact that the system must be linked so that data intended to go into common space are actually placed there. In order to accomplish this, it is necessary to adjust the system base address so that all common data will be linked into common space.

This is best accomplished by performing two links of the system. First, we will do a trial link to establish how big the system is going to be. To do this, put the "NOS/MT Rel & Obj" disc in Drive 1 and edit the system map with the command:

```
EDIT L
```

List the first line of the map, which will be a BASE command, and change the number on the BASE command to 01000, then leave the editor with the command:

```
END
```

Now insert the "NOS/MT Source B" disc in Drive 2 and perform the following steps.

Link the system by entering the following commands:

```
LINK
OUT 2/BOOT$.SAV
IN @L
```

The linker should eventually respond with its normal command prompt of "£". If it prompts with a minus sign, enter a REF command to list the undefined symbols, make a note of them, and go back and correct the configuration file to eliminate them. You must then, of course, repeat the configuration merge and assembly phases before you can attempt the link again.

Assuming that the linker encountered no undefined symbols, enter the command:

```
MAP
```

to list the memory map. Write down the address at which the file "COMMON" starts. Taking the start of COMMON with the BASE at 01000, compute what BASE is required so that COMMON will start after the first address of the common memory page. A BASE specification must be a multiple of 256 (0100 hex), so the BASE should be set so that COMMON starts in the first 256 bytes of the common memory page.

For example, assume that in the trial link with BASE of 01000, the MAP listed:

```
COMMON   7154-731B
```

and we are configuring a system which will have its common space from

E000 to EFFF. Thus, we want a BASE such that COMMON will start between E000 and E0FF, and we can calculate that base as follows:

1. Round the start address of COMMON down to the next lowest multiple of 0100 hex (in this case, 07154 gets rounded down to 07100).

2. Subtract 01000 to subtract the BASE of the trial link (in this case, we get 06100 after subtracting 01000).

3. Subtract the result of step 2 from the first address of common space. This is the BASE for the actual link. (In the example, we subtract 06100 from 0E000, which gives 07F00 for the new BASE).

Having now determined the BASE for the actual link of the system, edit the "L" file again and replace the trial BASE with the base calculated above. Now perform the actual link of the system with the following sequence of commands:

```
LINK
OUT 2/BOOT$.SAV
IN @L
MAP
END
```

Examine the memory map and confirm that the file COMMON begins at the intended location, within 0100 hex bytes of the start of the common memory page, and also make sure that the ending address of the last file listed in the map is less that the last address of the common memory page. If the end of the system overflows the common memory area you have configured, you must go back and reallocate memory to add space to the common area. This involves making the user areas smaller and adjusting the system, common, and user memory masks to enable the larger common area.

10. Initial system testing

You are now ready to load the newly generated system and test its operation. Remove all discs and place the "NOS/MT Source B" disc in Drive 1. Reset the computer, and follow the normal bootstrap procedure. If the system has been properly generated, the messages:

```
Running on MT:  Marinchip NOS/MT ver 2.6
Error 21.
:
```

should be displayed on every terminal configured in the system. The error message is displayed because the system expects to have been loaded from an NOS format disc, but in this case, having been loaded from a Disc Executive disc, the automatic login process has errored. Remove the "NOS/MT Source B" disc from Drive 1 and insert the "NOS/MT Boot" disc. Enter the commands:

```
MOUNT 1:
LOGIN
```

The LOGIN program should ask for a user name, to which you should respond:

```
SUPERMAN
```

and then will ask for a password. Enter:

```
DAILYPLANET
```

which will not be echoed on the terminal. LOGIN should then confirm that you are logged on as the privileged user, group number 0, user number 0.

The next steps will copy the newly generated boot file onto the system disc. Insert the "NOS/MT Source B" disc into Drive 2, and enter the commands:

```
MOUNT 2:*
CONVERT 1:BIN=2/BOOT$.SAV
DISMOUNT 2:
```

then remove the disc from Drive 2.

At the completion of these commands, your system disc is ready for use. You should now be able to boot that disc directly, and the system will automatically mount it and initiate the LOGIN process on all terminals.

11. Setting up your system for users

If you have successfully completed all the steps so far, you now have a working NOS/MT system. This chapter provides an overview of how to set up this system for its users. After bringing up the system, log in as SUPERMAN (see the "Initial system testing" section), and list the system user name file with the command:

```
TCOPY CONS.DEV=1:UTIL/USERDATABASE
```

In this file you will see all the user names, passwords, and group and user numbers configured on the supplied system disc. Make a note of the secret user name and password of the user with group 0 and user 666, which is used to PREP discs for use with NOS/MT. Note that the file includes several user names on group 112. These are users here at Marinchip Systems, and are included as samples of how to set up a normal user with an assumed directory.

To configure new users, update 1:UTIL/USERDATABASE with EDIT to add the new users. Give each user an assumed directory (normally in 1:UFILES). Several users may share an assumed directory (there are no restrictions). For information on the format of the user database file, refer to the description of the LOGIN command in the User Guide manual. It is also wise to change the user name and password associated with the privileged user and the user permitted to PREP discs.

The next step is to create the working directory files for the new users. This is done with the MAKEDIR command. For example, you might use:

```
MAKEDIR 1:UFILES/BONEHEAD
```

to create the directory for Melvin Bonehead. Since you have created his directory from group 0 user 0, however, he will not be allowed to access it unless you use the OWNER command to move it to his group and user number. For example, if Melvin Bonehead had been assigned group 904, user 218, you would use:

```
OWNER 1:UFILES/BONEHEAD(904,218)
```

to change the directory ownership. If you don't like the access modes the system assumes for the directories it creates, change them with the ACCESS command.

Note that since the privileged user, SUPERMAN, does not have an assumed directory, you must specify a completely explicit file name, including unit or volume, for every file named. If you don't like this, you can do an ASSUME to any directory you like.

12. Using double density and double sided diskettes

When using the Teletek FDC-II controller, NOS/MT may be used with double density, and assuming the drive hardware supports it, double sided diskettes. This section explains how the system handles various kinds of diskettes, and restrictions on their use.

While single density diskettes may normally be used "right out of the box", double density diskettes must be reformatted for use with the M9900. This formatting is done by the Marinchip utility program FDCIIFORMAT, which must be run under the Disc Executive. Because the formatting process requires direct control over the disc hardware, it would disrupt NOS/MT, and hence formatting must be done under the Disc Executive. We recommend that you format all discs in a box of double density diskettes as soon as they are received. Then you will always have a supply of preformatted diskettes ready for use under NOS/MT, and will not have to go back to the Disc Executive to format a disc whenever you need one.

It is physically possible to reformat a single density diskette as double density. However, the actual recording surface of a single density diskette is not certified for the much more demanding bit density of double density operation, so this is likely to result in unreliable operation, especially on the inner tracks of the diskette.

Double sided diskettes are physically different from single sided diskettes (they have the index hole in a different location). This permits the disc drive to sense which type of disc has been installed. When FDCIIFORMAT is run, you must tell it whether the disc being formatted is single or double sided. If the disc is physically double sided (offset index hole), it MUST be formatted as double sided if it is to be used with a system supporting double sided discs.

Once a disc is formatted, no special considerations are required for use with NOS/MT. When the directory is created with the PREP command, NOS/MT will sense the amount of storage available on the disc and construct the directory properly. Any of the following possibilities will work:

Single sided and single density
Single sided and double density
Double sided and single density
Double sided and double density

The system increases the number of files permitted on a volume proportional to the amount of space available on it.

Because of the internal addressing structure of the NOS/MT file system and the allocation block size used on diskette volumes, the largest non-contiguous file which can be created on a floppy disc is 262144 bytes, even though more storage may be available on the volume. Thus, four maximum size files would be required to fill a double sided double density diskette. This restriction does not apply to contiguous files, so applications which require larger files should use contiguous files. Since a file larger than 262144 bytes is a very substantial portion of an entire diskette, contiguous allocation is

desirable for such a large file because it reduces the overhead in accessing the data in the file. The file size restriction discussed in this paragraph applies only to floppy disc volumes. Much larger files are permitted on hard disc volumes.