Separated Boot Process as of 11/2/83

Station		Master
1)	<----------	Boot Phase 1 (to user 254)
If a Z80 station, load this in and execute it. Otherwise, run a ROM program that does the same thing, i.e. get ready to react to the poll of the login-user. Set name/psw to the ascii representation of our serial number.		
2)	<----------	Poll loguser (user 253)
3) login req (name/psw, binary serial, product type)	---------->	
4)		Search for the serial # in the Machine Table. If found, the offset into the Machine Table (plus one) becomes the User Number. If not found, reject the login, i.e. go to step 5b.
prod-type matches type in machine table?		Find the product type in the Product Type Table. This tells us how to get the appropriate Boot Phase 2, "Login Please" program and "OS Menu" progra) If no entry, go to Step 5b.
		Look up name/psw in user table. Fill the Boot Phase 2 load list with: Load list from OS table entry, if name/psw found and a suitable OS is found in the table. Addr of "OS Menu" program, if name/psw found but no corresponding OS can be located. → 9c Addr of "Login Please" program, if name/psw not found. → 9b Insert default disk assignments & type-ahead buffer. Set flag to show whether this is the requested OS or a default. Set the IOBYTE according to the default IOBYTE in the Machine Table entry.
		Assign a user number and log the user in with name as sent, even if that name was not found in the user table.
		This is considered a successful login so go to step 5a.

5a) (if all essential table entries or defaults were found)

<---------- logack(ack,userno,time,binary seria')
(to user 253)

5b) (if a crucial table lookup failed so no Boot Phase 2 can be sent)

<---------- lognack(nack,userno,time,binary serial)
(to user 253)

6) If binary serial doesn't match the one we sent, or if the "reply" is really a poll, our request was missed. Return to Step 2.

If message if a logack, prepare to get Boot Phase 2 under the new user number.

If message is a lognack, we are in a heap of trouble and should probably dump serial # and product type to screen with message.

7) <---------- Boot Phase 2 (to new user number)

How control local-point?

The first byte of Boot Phase 2 is the number of 1024-byte chunks it contains. The station reads the first chunk, then looks at this length to determine whether there are more coming; if so, it reads them. The station executes Boot Phase 2 by jumping to the address following the length byte, i.e. the second byte of Boot Phase 2.

8) Boot Phase 2 is a loader for a series of programs and contains their addresses, lengths, etc. When run, it will load at least one program (by ordinary HiNet reads). It will transfer control to one of the programs it loads, as determined by the particular Boot Phase 2 code and by the data inserted into it when it was configured by the Master. The outcome of this loading process will be one of the following:

9a). If the program we loaded is a BIOS, it will run its cold-boot code. IF WE EVER GET HERE, WE'RE DONE. *****

9b). If we've actually loaded a "Login Please" program, it will respond to the next poll of our user number with a request to log us out; thereafter it will answer no polls of that user number. It will then ask the user for a name and password, which it uses to construct a new login request.
RETURN TO STEP 2. ****

9c). If we have an "OS menu" program, it will ask the user to choose among the possible OS's for this machine, and to choose default partition assignments compatible with the selected OS. The program will then reconfigure Boot Phase 2 to load the selected OS.
RETURN TO STEP 8. ****

System Directory Layout

Up to 128 entries.
Each entry is 24 bytes:

8 bytes	filename (ASCII)
5 bytes	disk address (v,p,t,s)
2 bytes	length (in 128-byte sects)
4 bytes	load address
2 bytes	offset to execution address
1 byte	prog/data flag
2 bytes	reserved

128 entries of 24 bytes each: 3k bytes total.

The System Directory describes storage allocation within a large contiguous area; i.e. a portion of Unit 0 is reserved for the exclusive use of files pointed to by this directory.

For simplicity the 8-character name should correspond to the name of the .COM file created under CP/M.
The disk address contains fields for volume and partition number and these should be filled in even though as of 9/13 we have decided to put all system files in Unit 0.
The length field allows programs to load files using this directory and allows utilities to determine disk usage.
The load address is the address in the host's memory where the program expects to be loaded, and the 2-byte offset is the distance from that point (always non-negative) where execution is to begin.
The prog/data flag has low bit 0 for tables or other data files or 1 for executable code.
A utility which alters or erases any directory entries is responsible for moving the remaining files so that there are no holes in the disk storage area except the one after the last file. This allows the use of a high-water mark entry whose name field is filled with nulls.
Each file has one entry in the System Directory and each entry refers to one file. In the future it may be permissable to have two or more directory entries that differ only in the load address and name fields, as in the case of relocatable 86 code.

Product Type Table Layout

Up to 40 entries.

Each entry is 25 bytes:		
	1 byte	Product Type (0 for end)
	8 bytes	name of Boot Phase 2 program for this product type
	8 bytes	name of "Login Please" program
	8 bytes	name of "OS menu" program

40 entries of 25 bytes each: 1000 bytes (1k allocated.)

Search routines are expected to find a match for seven bits of the Product Number, i.e., they are to ignore the high bit which identifies a serial or parallel console. The programs which can be found through this table all use PROM console drivers and so are oblivious to the console type.

The names of programs in this table correspond to names in the System Directory.

Machine Table Layout

Up to 128 entries.

Each entry is 12 bytes:		
	4 bytes	Serial Number (binary form)
	1 byte	Product Number
	6 bytes	Option Map
	1 byte	IOBYTE

128 entries of 12 bytes each: 1.5 k.

Option map will have one bit assigned to each possible device driver, e.g. 5-inch HD with Xebec controller, Port 3 type-ahead, spooler, etc).

Machine table search routines will return the product number if the machine is found, or zero if the machine's serial number is not in the table.

New User Configuration Table Layout

Up to 128 entries.

Each entry is 64 bytes:		
	8 bytes	default A drive
	8 bytes	default B drive
	8 bytes	default C drive
	8 bytes	default D drive
	1 byte	length of typeahead
	31 bytes	typeahead

128 entries of 64 bytes each: 8k.

New User Name Table Layout

Up to 128 entries.

Each entry is 16 bytes:		
	8 bytes	User Name
	6 bytes	Password
	1 byte	OS number (upper nibble is the "generic" OS type, lower nibble is specific -- CP/M-80, 86 etc; lower nibble 0 means don't care)
	1 byte	OS size flag. lower bit 0 for full-service OS, 1 for smallest available OS.

128 entries of 16 bytes each: 2k.

The two bytes in each entry formerly listed as "reserved" are now used, but the layout is otherwise the same as before.

OS Table Layout

Up to 128 entries.

Each entry is 96 bytes:		
	1 byte	OS Number
	16 bytes	Product Map
	6 bytes	Option Map
	64 bytes	load list (up to 8 fields, each being an 8-character name; list is terminated by nulls)
	9 bytes	reserved

128 entries of 96 bytes each: 12k.

For an OS to be selected, the bit in the Product Map corresponding to the Product Number from the Machine Table must be "1".

To be selected, the upper nibble of the BIOS' OS number must match its counterpart in the User Option Table. The lower nibble of the BIOS' OS number must also match its counterpart in the User Option Table if the latter is non-zero. If the lower nibble in the User Option Table is zero, no test will be made on the lower nibble of the BIOS OS number.

(Thus, as a minimum, the BIOS must run the OS the user requested on the machine the user is using. By allowing a degree of defaulting on the OS type, we allow the automatic selection of the version of the specified OS which will run on the specified machine.)

If the User Table entry shows that a "full-service" BIOS is desired, the Option Map from the Machine Table will be matched against the Option Maps in the OS table. If an exact match is found this is th OS that will be sent. If no exact match is found, or if the user has requested a "high-TPA" BIOS, then the smallest BIOS matching the OS and machine requirements will be sent.

The "load list" contains the names of any system programs that must be loaded in booting this OS, in the order they are to be loaded, starting with the name of the OS itself. For example, a CP/M 2.2 BIOS would have the name of the BIOS first, then the name of the file containing the CCP and BDOS (to be loaded on each warm-boot.) There can be up to eight files specified in this way (i.e. the file to which this table entry specifically corresponds, plus up to seven others.)

(If there is no BIOS that matches the OS and machine conditions at all, an "OS menu" program for the appropriate machine will be sent instead.)

BOOT PHASE 2 program

Boot Phase 2 is the loader for the OS to follow.

Its first byte will be the number of 1024-byte blocks the Boot Phase 2 program occupies, enabling the receiving station to set up for another network transfer if necessary.
The next three bytes enable the host to jump to the beginning of the Boot Phase 2 code. In a Z80 this will simply be a jump instruction. In an 86 the first of these bytes will be null and the next two will be the offset from the beginning of Boot Phase 2 to the beginning of the executable code. dw ?s or bytes?
Next comes a data block, always in the following format:

Default partition assignments (four names of eight characters)		32 bytes
IOBYTE		1 byte
Default type-ahead buffer and pointer		32 bytes
Honor Flag (see description below)		1 byte
Load List (up to eight partial directory entries each 16 bytes, of the form:		
Disk Address (v,p,t,s)	5 bytes	
length (128-byte sects)	2 bytes	
RAM Load Address	4 bytes	
Relative Start Address	2 bytes	
Reserved	3 bytes)	
	total:	128 bytes
total length of data block		194 bytes + 4 = 198

This data block is initialized by the Master before Boot Phase 2 is sent out. The default partition assignments and default type-ahead information are obtained from the User Table entry. The IOBYTE is derived from the User Table entry possibly modifed by a bit in the Machine Table entry.
The default type-ahead buffer is initialized the same way as in the past, i.e. the entire 31 bytes are included together with a length byte showing how many of those 31 are actually used.

The Honor Flag is set to show the extent to which the login request was honored, i.e. what sort of program will be loaded next. Codes are:

0 - An OS exactly matching the request will be sent, i.e. the Option Map will be exactly the intersection of the map in the Machine Table and the map in the User Option Table.

1 - A default OS will be sent, of the general type as the requested one but of with different option map. Implies that there is no suitable system with the exact option configuration requested.

2 - A "login please" program will be sent. Implies that the name/psw were not found in the User Table.

3 - An "OS Menu" program will be sent. Implies that no OS of the requested type could be found that would stand a chance of running on the machine logging in. In the initial release this will probably be a tough-luck message rather than a real menu.

4 - Total failure, no program will be loaded.

(Boot Phase 2, continued)

The Honor Flag will be examined by Boot Phase 2 to decide whether to print a message (using PROM I/O facilities) notifying the user of this situation. Such a message would probably be required only in the case where a default OS is being sent.

The Load List consists of entries from the System Directory, with the eight-byte names removed, making each entry sixteen bytes long. Each entry describes a file to be read from the net, including information as to where it is to be loaded and where to begin executing it. If fewer than eight files are specified (actually two will be much more common), the length field in the entry after the last one used is set to zero by the master (in fact, the block is filled with zeros on the end, but these might be legal values in other fields.)

Thus there are at least 198 bytes in Boot Phase 2 before the real executable code.

"LOGIN PLEASE" program

This program will be loaded in lieu of an OS if the name/psw in the login request do not match any entry in the User Table. (This will often be true on auto-login by serial number.) It first action will be to request an "instant logout" of the current user number. The program will then ask the user for a name and password. After appending the binary serial number, product type and HiNet user number, the program will wait for a poll of the login pseudo-user (253), issue a login request, and prepare for the log-ack.

The Master's response will be handled the same by this program as it would be in Step 6 of the above boot procedure; i.e., the serial number in the response will be matched against our own, and the login request will be retried if the serial numbers don't match.
If they do match, and the response is an ack, the program notes the new HiNet user number and waits for a new Boot Phase 2 to be sent to that user number. It then executes Boot Phase 2.
If the serial number matches but the response is a nack, there is a major bug. A diagnostic message will be sent to the screen.

The login program will have access to the PROM I/O facilities, to the information passed in Boot Phase 2, and to the HiNet user number.

"OS MENU" program

This program loads in lieu of an OS. It uses PROM I/O facilities and has access to the information passed in the Boot Phase 2 code and t the HiNet user number.

The program reads the OS table from the master and presents to the user a menu of the OS's capable of running on the machine logging in. (Presenting the option map in a reasonable way may be difficult.) When the user selects one, the program loads it -- probably by re-initializing the still-resident Boot Phase 2 with the address of the selected OS, and re-running Boot Phase 2.

Since appearance of the "OS MENU" program implies that no system of the type specified in the User Option Table could be found for the machine, it is unlikely that the default partitions will be suitable for the OS eventually selected. Therefore the re-initialization of Boot Phase 2 must include new default partitions as specified by the user from the console. Program must check that the requested partitions can be used under the requested OS. Any TPA utility version of this program will also have to ask for the default partitions, for the same reason.

Note that this program is required to know the location of the OS table in the master.

Until completion of the OS menu program, the system would benefit from a skeletal program that tells the user why the login request could not be honored.

OS numbers
---

0 not to be used - can be returned if search fails, etc.

1 CP/M
- 11 = CP/M 2
- 12 = CP/M 86
- 13 = HiDOS
- 14 through 1F = other CP/M compatible OS's

2 MS-DOS
- 21 = MS-DOS vers. 2.0
- 22 = MS-DOS vers. 2.x
- 23 = MS-DOS vers. 3.x
- 24 through 2F = other MS-DOS compatible OS's

3 Unix 31 ? 3F ?
- ~~21~~ through ~~2F~~ = Unix-compatible OS's

4 through F not yet assigned

Product Numbers

The highest bit of the Product Number specifies the console type (0 for serial, 1 for parallel). This leaves 128 possible product numbers which specify the type of CPU board as follows:

0 Not to be used

1 ZSBC-3 CPU's:
- DMS-3 "DSC-3"
- DMS-3/A25 "Smart ADDS"
- DMS-3/4004
- DMS-3/4008
- DMS-3/101
- DMS-3/102
- DMS-3/103
- DMS-3/B
- DMS-3/F "Fox"
- DMS-15
- DMS-5080

2 DMS-4 "DSC-4"

3 DMS-1280

4 DMS-3/C "Killer Bee"

5 DMS-5086
- and HNS-86

6 DMS-5016

7 DMS-816 "Rosebud"

(Product numbers 8 through 127 not yet assigned)

Optional Device Drivers

This is a bit-map with the exception of the block describing the number of logical drives supported. This is permissable because as of 9/13 we have done away with the logical ANDing of two bit maps, requiring instead an exact match with the Machine Table version. Since we have agreed to give England an arbitrary number of logical drives (up to 16), the only bit-mapped way of doing it would use 16 bits or one-third of the bit map.

N.B. We are going to have to change the way we offer "custom" printer drivers since there is no way of knowing, a priori, whether a driver named "custom printer" will run on a given machine.

bit	device driver
0	8-inch floppies (SD and DMS DD) — how distinguish?
1	5-inch Fox-floppies
2	5-inch IBM-format floppies
3	8-inch HD with DMS controller
4	5-inch HD with Xebec controller
5	5-inch HD with Adaptec controller
6	Port 0 type-ahead (console)
7	Port 0 polled (")
8	Port 2 polled (printer)
9	Port 3 type-ahead (aux comm.)
10	Port 3 polled (" ")
11	Parallel Port 1 (console)
12	Parallel Port 1 (printer)
13	Parallel Port 2 (Fox printer)
14	Console/Printer Mux (ADDS, 1280, 5000)
15	Spooler
16	Net Buffer (1k)
17	Real-Time Clock
18	Front-Panel Interrupt
19	Number of logical drives (bit 0)
20	Number of logical drives (bit 1)
21	Number of logical drives (bit 2)
22	Number of logical drives (bit 3)

---------- Proposed New HiNet Hard Disk Layout 9/16/83 ----------

Contents of Partition 0 of a Hard Disk

Logical Address	Contents
track 0, sectors 01-1F	Controller Program (unchanged)
track 0, sectors 20-28	reserved for expansion of controller program
track 0, sectors 29-38	HiNet User Name Table
read ø (øø 29 5ø)	Up to 128 16-byte entries: 8 bytes: user name or stn serial # 6 bytes: password 1 byte: OS code 1 byte: flags (incl big/small request)
track 0, sectors 39-78	HiNet User Configuration Table
	Up to 128 64-byte entries: 8 bytes: default A drive 8 bytes: default B drive 8 bytes: default C drive 8 bytes: default D drive 1 byte: length of typeahead 31 bytes: typeahead buffer
read ø (øø 79 ø8)	
track 0, sectors 79-80	Disk Allocation Table (unchanged)
track 1, sectors 01-08	Bad Sector Table (unchanged)
track 1, sectors 09-14	Machine Table
read ø (ø1 ø9 øC)	Up to 128 12-byte entries: 4 bytes: Serial Number 1 byte: Product Number 6 bytes: Option Map 1 byte: IOBYTE
track 1, sectors 15-18:	Mike & Tracy's First-Write Table (What's This?)
track 1, sectors 19-20:	Product Type Table
read ø (ø1 19 ø8) E5 fill	Up to 40 25-byte entries: 1 byte: Product Type 8 bytes: Boot Phase 2 program name 8 bytes: Login Please program name 8 bytes: OS Menu program name
track 1, sectors 21-80:	OS Table
øø fill clobber.tab fills out this track & next with øø (boot loader must be re-written)	Up to 128 96-byte entries: 1 byte: OS number 16 bytes: Product Map 6 bytes: Option Map 64 bytes: Load List (8 names of 8 bytes 9 bytes: --reserved--

(continued next page)

track 2, sectors 01-02	Cold Boot Loader (unchanged)
track 2, sectors 03-08	reserved for use of Cold Boot Loader
track 2, sectors 09-20	System Directory
	Up to 128 24-byte entries: 8 bytes: File Name 5 bytes: Disk Address 2 bytes: Length (128-byte records) 4 bytes: Load Address 2 bytes: Execution Address Offset 1 byte: Program/Data flag 2 bytes: --reserved--
track 2, sectors 21-80	--reserved--

remainder of disk allocated according to contents of the System Directory

E5 fill