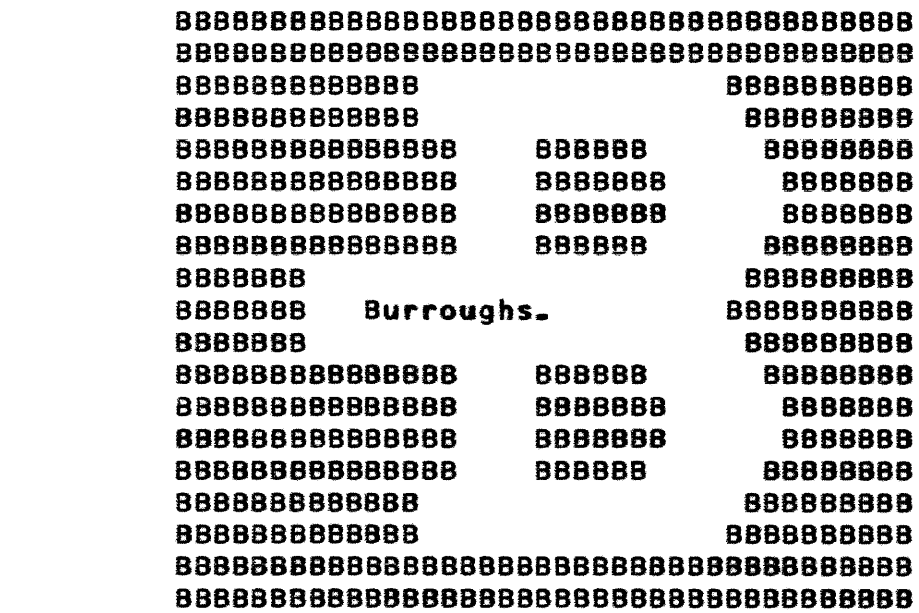
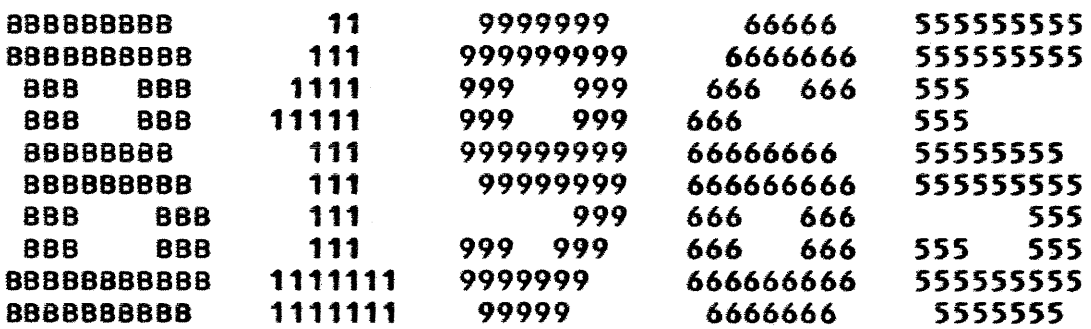
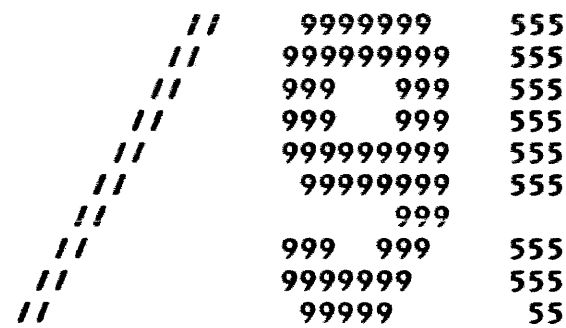

Burroughs.





################################ 04/22/85 ################################
##

```
MM        MM      AAAA       CCCCCCCC    IIIIIIIIIIII
MMM      MMM     AAAAAA     CCCCCCCCCC   IIIIIIIIIIII
MMMM    MMMM    AAA  AAA    CCC    CCC     III   III
MMMMM  MMMMM   AAA    AAA   CCC            III   III
MMMMMMMMMMMM   AAA    AAA   CCC            III   III
MMM MMMM MMM   AAAAAAAAAA   CCC            III   III
MMM  M   MMM   AAAAAAAAAA   CCC            III   III
MMM      MMM   AAA    AAA   CCC    CCC     III   III
MMM      MMM   AAA    AAA   CCCCCCCCCC   IIIIIIIIIIII
MMM      MMM   AAA    AAA    CCCCCCCC    IIIIIIIIIIII
```

##
##
###

MAINTENANCE ACCESS CARD MODEL II DESIGN SPECIFICATION

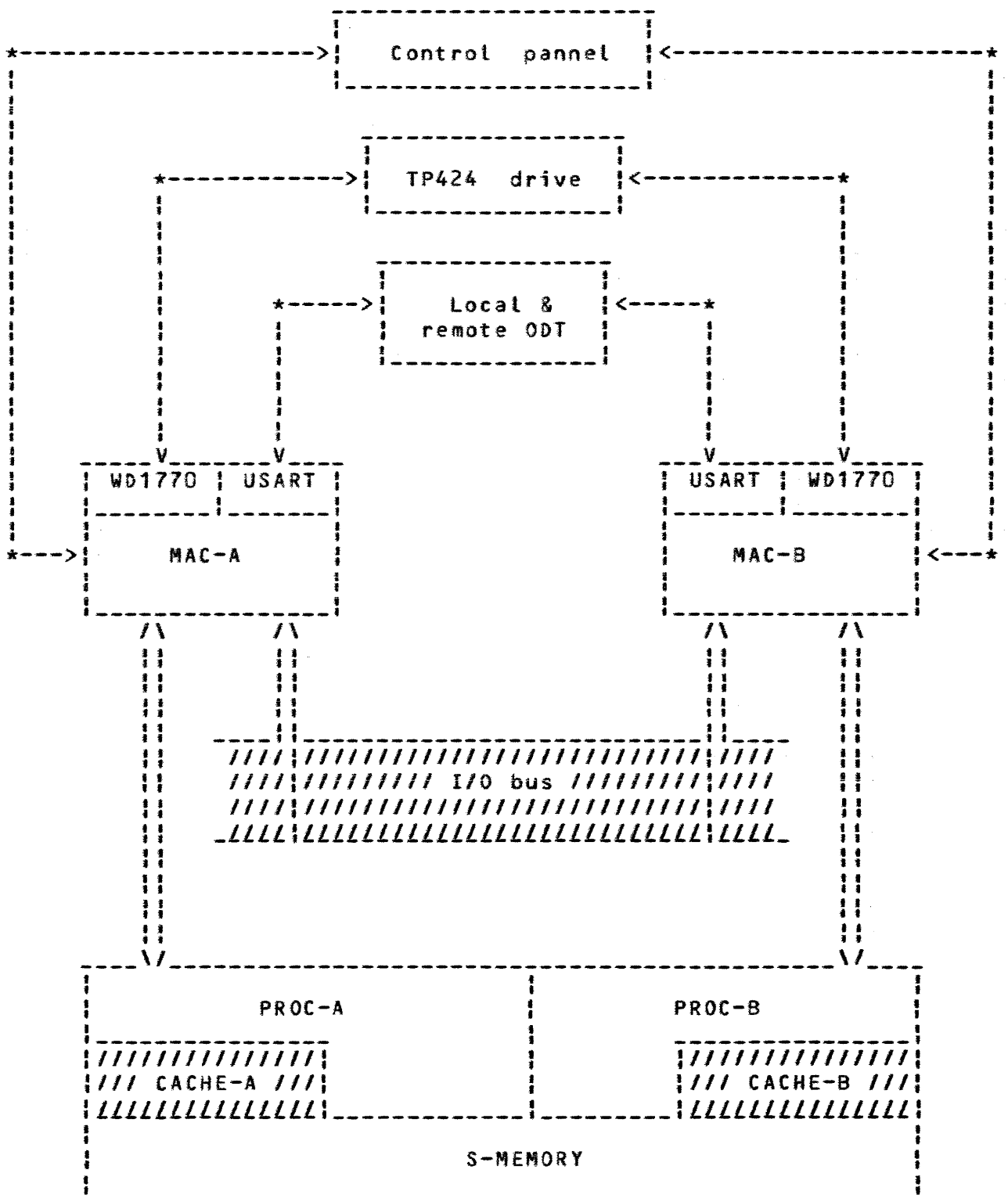
###
##
##

##
##

TABLE OF CONTENTS.

APPENDIX :

SYSTEM GENERAL STRUCTURE :



USE OF 5 1/4 INCH DISKS ON B1965/95 MAC.

The diskettes to be used on the new maintenance access card will be 800 Kbytes 5 1/4 inch TP400 disks (as formatted on an ET2000 processor), and will be driven by a WD1770 LSI.

The media characteristics will be as follows :

Sector size	512 bytes
Total size.	1600 sectors
Number of cylinders . .	80
Number of surfaces. . .	2
Sectors per track . . .	10

The software formatting may be summarized as follows :

Sector 0 : Reserved area.

Sector 1 : First copy of file allocation table (FAT), (341 entries of 12 bits).

Sector 2 : Second copy of file allocation table.

Sector 3 - 11 : 144 directory entries (32 bytes/entry), each entry contains (among other items) :
- file identifier (8 bytes),
- file id. extension (3 bytes),
- first cluster address (see below).

Sector 12 - 1599 : Data area, divided into several "CLUSTERS" - allocation units of 8 consecutive physical sectors.

Each cluster belonging to a file is linked to the next one (if any) via the file allocation table (FAT).

Any cluster address (12 bits) can be translated into a physical sector address (16 bits), using the following algorithm :

PHYSICAL ADDRESS := 12 + 8 * (CLUSTER ADDRESS - 2)

(continued next page)

There will be 3 types of files present on the diskette :

- xxxxxxxx.SYS files : containing MAC-II firmware.
- xxxxxxxx.MTR files : data files to be transferred to the host via the "U" register.
- xxxxxxxx.EXE files : programs executable by the MAC-II card (via the "EX" command).

For a more practical use the first file present on the diskette, starting from sector 12, will be the MAC-II firmware.

When powering the system up, the maintenance access card will automatically initiate its self test (resident in ROM), load its non resident firmware ("xxxxxxx.SYS" file), and give control to that piece of code.
One of the first steps will be to read the FAT & directory in RAM, which should be kept in memory for further use, and wait for an operator input.

GENERATION OF MTR FILES :

The input files will have to be located on an MCPII system and will be the same as the ones used to create MTR cassettes. A new program called "SSLOAD/MAKDISK", similar to the previous "SSLOAD/MAKCAS", will be designed to convert those input files (mainly output of the MIL compiler) to a 512/1 file (instead of 368/1) suitable for transfer to the ET2000, which will not produce a CRC byte between each data byte, and take into account the "CASSETTE STOP" (0022) micro to fill up the 512 bytes buffer (generate pseudo gap).
Once transfered to the ET2000, the file will be copied to the TP400 disk as a sequential data file, whose file name will be suffixed by a ".MTR" extension.
This procedure does definitively not allow the generation of disk code using a previously made system cassette.

DISKETTE DATA ACCESS :

Will be performed by a firmware module capable of handling the diskette functions, each one of those will executed from specific entry points named as follows :

Type 1 operations : (head motion)	Type 2 operations : (input / output)
- RESTOR	- READ
- SEEK	- WRITE
- STPIN	
- STPOUT	
- STEP	

(continued next page)

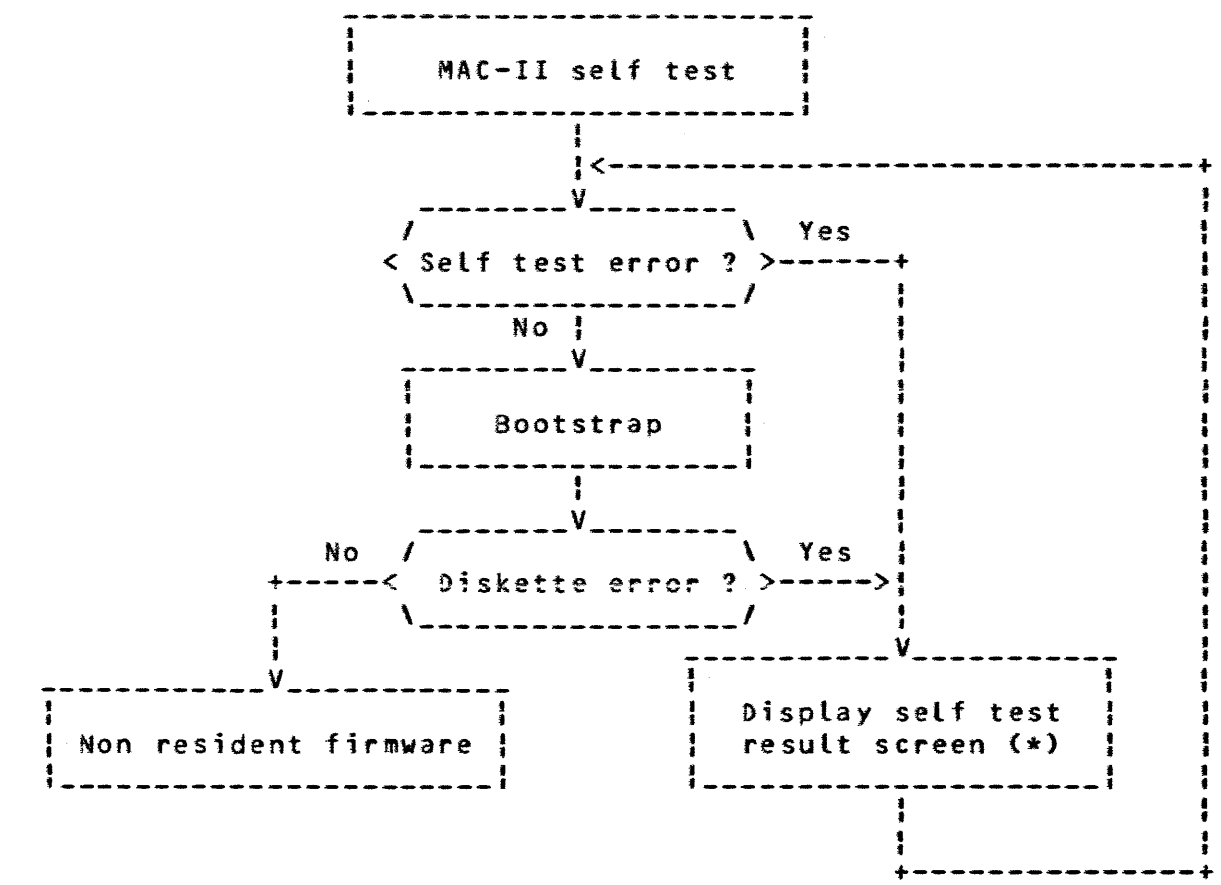
The required parameters to activate those operations are :

	PL/M 80	ASM 8080
First parameter	Disk address (absolute 0 relative)	Registers B & C
Second parameter	Memory buffer address	Registers D & E

MAC-II FIRMWARE STARTUP :

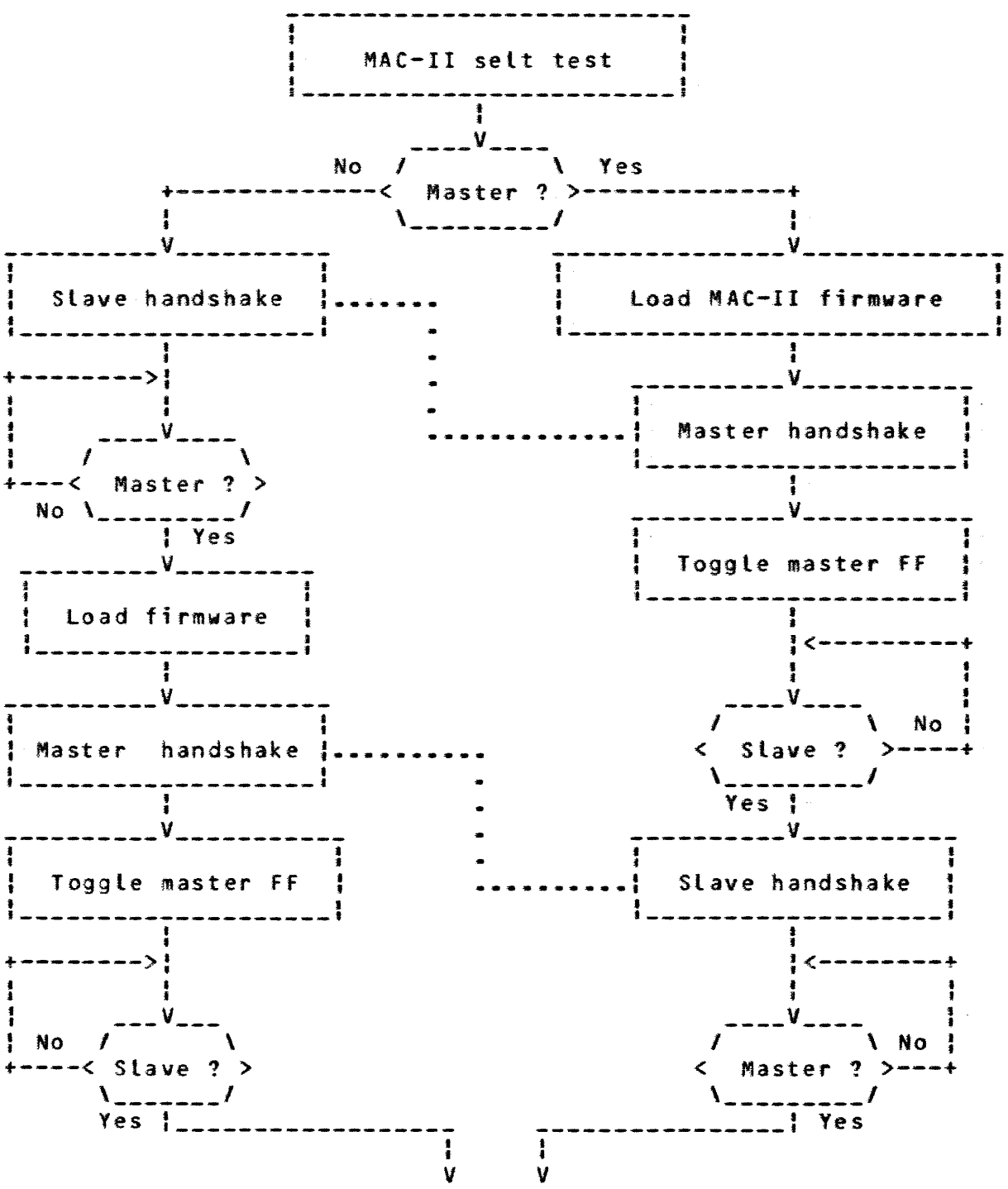
As mentionned above, the whole MAC firmware will no longer be resident in ROM, but will rather be bootstrap'ed from disk after completion of the self test, provided a suitable disk was inserted in the drive when the firmware is started up.
Since the ODT handling module is overlayed on disk, there needs to be an appropriate procedure to display any exeception condition that occurs while loading the RAM firmware.

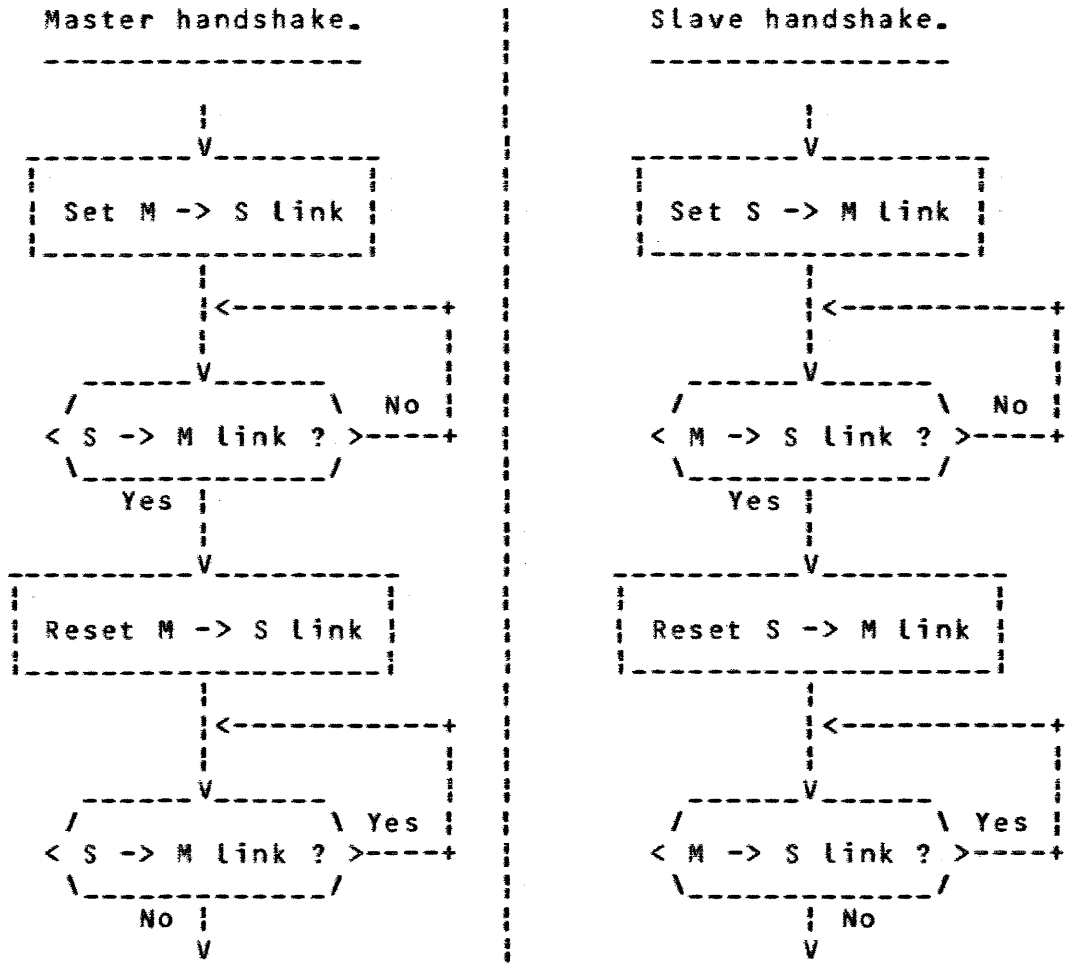
These functions will be performed as follows :



(*) : Contains information on the diskette availability.

Since only the master processor may access the disk driver chip, the above procedure has to be executed 2 times sequentially by each one of the A/B processor, according to the following flow :




Master handshake.
Set M -> S link
S -> M link ?
No
Yes
Reset M -> S link
S -> M link ?
Yes
No
Slave handshake.
Set S -> M link
M -> S link ?
No
Yes
Reset S -> M link
M -> S link ?
Yes
No

MAC-II I/O DEVICES :

The firmware will no longer access peripherals using an I/O memory map as on the previous H9 card, exercisable via moves to/from M micro instructions, but rather issue IN/OUT micros (@DBxx / @D3xx) on corresponding I/O ports (xx = address).

The following conversion table applies to this change :

Peripheral	Type	MAC-I Memory map addresses	MAC-II Port addresses
PPI1	8255	@7580 - @7583	@90 - @93
PPI2	8255	@7600 - @7603	@94 - @97
IPPI	8255	@7680 - @7683	@88 - @8B
HPPI	8255	@7700 - @7703	@84 - @87
MPPI	8255	@7780 - @7783	@80 - @83
USART	8251	@7500 - @7503	@B0 - @B3
PTMR	8253	@7480 - @7483	@A0 - @A3

In addition, the diskette replacing the old-fashion cassette drive will no longer be accessible via the USART, according to "MODE SELECT" in PPI1c, but via a specific disk controller, named WD1770, to which the I/O port addresses @C0 - @C3 will be assigned, while the diskette side select function will be execised by an OUT micro on port address @D0.

The new MAC-II error display LED's will be driven by the firmware using the I/O port address @E0.

See appendix for a more detailled description of all the main command formats to those LSI's.

CHANGES TO EXISTING ODT COMMANDS :

(1) TAPE ----> LOAD "< file name >"

(2) MTR ----> MTR "< file name >"

(3) REWIND ----> UNLOAD

Where < file name > is an 8 characters alpha string.

- (1) Will place the processor in MTR mode, search for "< file name >.MTR" in the directory of the diskette already loaded in the RAM of the MAC-II card. Any subsequent "GO" command will cause the first cluster of the file to be read (if it can be found), and the "U" register of the host to be filled with the data read from the RAM buffers.

- (2) In addition to known features of the MTR command, this command will cause FAT and directory to be reloaded from the diskette to RAM, "< file name >.MTR" to be searched. Afterwards, by means of a "GO" command, data from that file will be provided to the "U" register of the host.

Both commands (1) and (2) need an additional error message to handle unsuccessful search conditions.

If no < file name > is provided to commands (1) and (2), the file identifier stored in RAM will simply not be modified, and the previously existing one will be re-used for subsequent commands, after completion of the self test, the file identifier in RAM will be filled with the "CLRSTART" pattern.

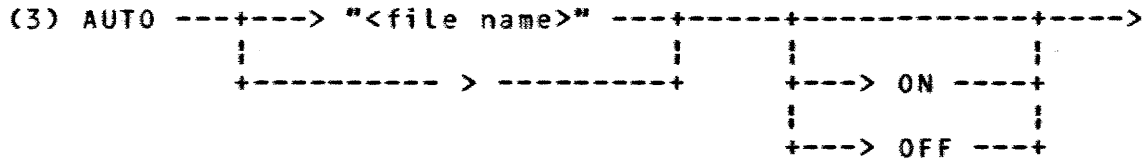
- (3) Will cause stepping the head mechanism of the diskette to track zero position in order to allow removing the floppy disk safely.

NEW ODT COMMANDS :

(1) EX "<file name>" [for hypothetical future use]

(2) DIR

(3) AUTO

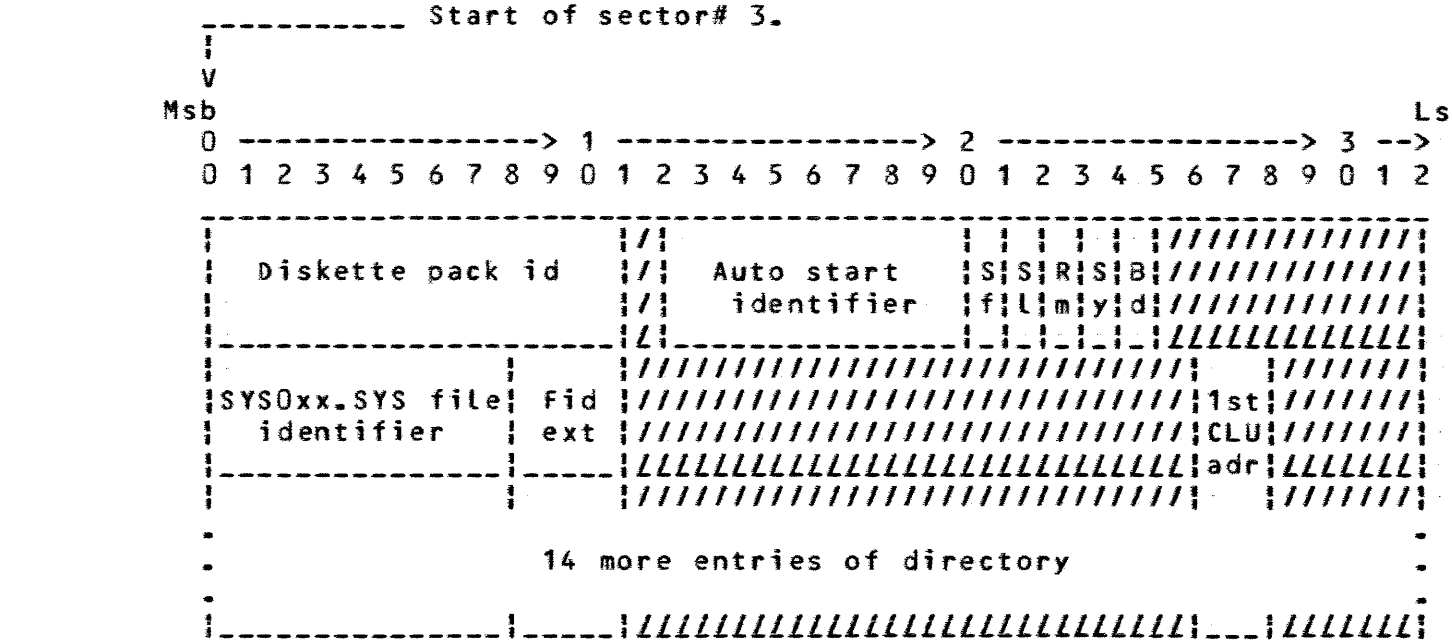


(1) will cause "< file name >.EXE" to be searched in the directory of the diskette, loaded into MACII memory, and executed as a firmware overlay.

(2) will cause the 128 first directory file identifiers of the diskette to be displayed on the ODT screen, free entries in the directory will be represented by a dot (".") pattern, all loadable files (.MTR files) will be highlighted.

(3) will cause the specified file name or the currently used file name (if not specified in the command) to be searched on the diskette, and, if found, this file identifier will be written at a specified location in sector number 3, followed by flags indicating the ON/OFF variant of the command (default is ON) and the status of the system when the "AUTO" command was entered on the ODT (i.e. slave processor ON/OFF status, remote odt status and transmission parameters), so that they can be retrieved later on.

The sector# 3 of the diskette will be updated by MAC-II firmware upon receipt of the "AUTO" command according to the following format :



Sf = auto start flag, equal to @FF for "ON" variant,
@00 for "OFF" variant.

Sl = auto slave flag, equal to @FF if slave off-line required at power-on/reset time, else equal to @00.

Rm = auto remote flag, equal to @01, if remote line to be initiated at power-on/reset time, else equal to @00.

Sy = auto remote synchronous flag, valid only if Rm = @01.

Bd = auto remote baud rate, valid only if Rm=@01 & Sy=@00 ;
allowed values : @00 = 1200 baud
@01 = 300 baud
@02 = 1800 baud

Sl & Rm flags will be taken into account only if the "ON" option of the AUTO command was selected.

Any subsequent power-on/reset of the card will retrieve this file name, make it the current file name, and, if the "ON" variant was specified, initiate its execution (with system updates described by the above paragraph).

A display of the file identifier being loaded/executed should be inserted in the menu section of the screen display, instead of the cassette status, while the "AUTO" information will be available in the "DIR" mode display.

FIRMWARE IMPLEMENTATION :

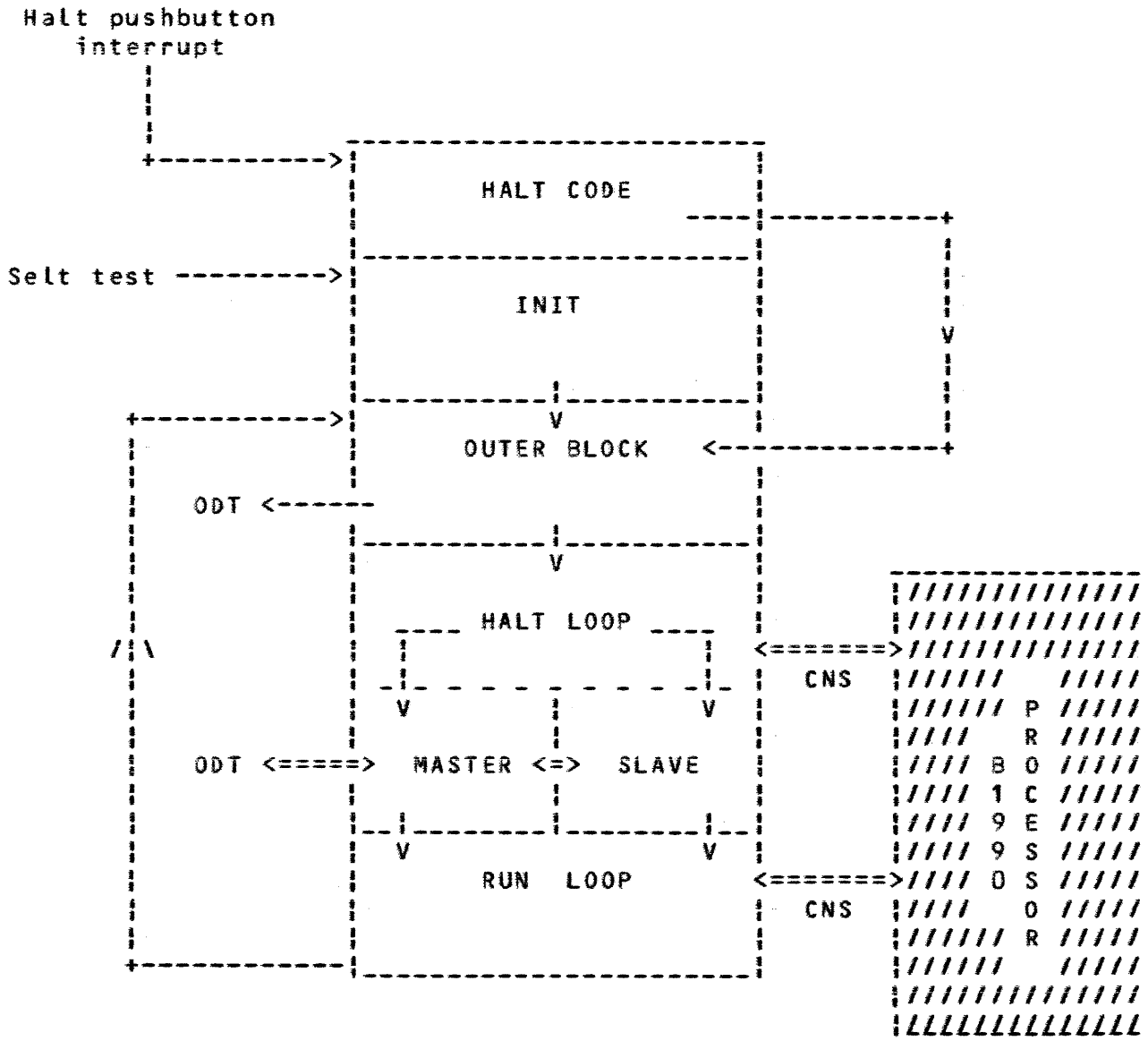
The following firmware modules are required to assemble the "SYSO.SYS" aggregate file :

Name :	PLM	ASM	Function :
DISK	//////	*	WD1770 disk driver
MAIN	*	//////	Firmware loop executive
SCANNER	*	//////	ODT input analyzer
SCREEN	*	//////	ODT message formatter
SPO	//////	*	Data comm driver for ODT's & slave <=> master
CPU	//////	*	Handles communications between MAC and B1990
TAPE	//////	*	TP424 disk file handler
SCAN	//////	*	Subroutines of scanner

The "DISK" module will be loaded at memory byte address @2000, and will contain links to & from self test routines.

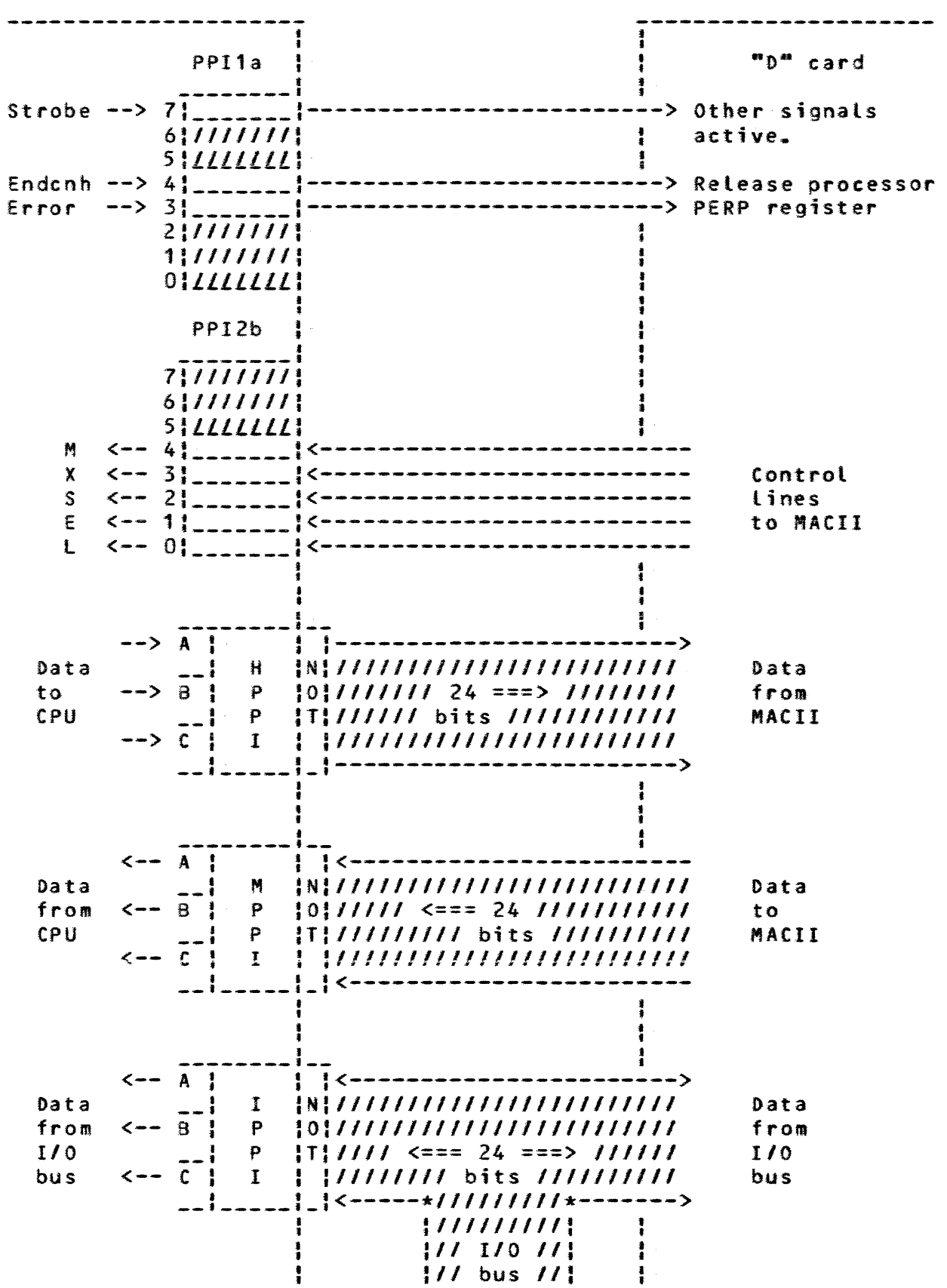
MAIN MODULE STRUCTURE :

The "MAIN" module has 2 entry points, one from the self test after power-on or reset ("BEGPLM" label), the other from the interrupt @38 mechanism ("HALTON" label). See following block diagram :



The "INIT" block will initialize PPIs and default system options, branch to "OUTER BLOCK" which displays the selected screen and enter the halt loop which handles the ODT input.
The "HALT LOOP" will be exited whenever either the "GO" command is entered, the "RUN" pushbutton depressed.
The "RUN LOOP" will poll the MEX select lines waiting for a CNS command, and will exit to "OUTER BLOCK" when PPI1b run bit is off.
The drawing on next page shows the interface (data + control lines) between MAC-II and B1990 cpu.

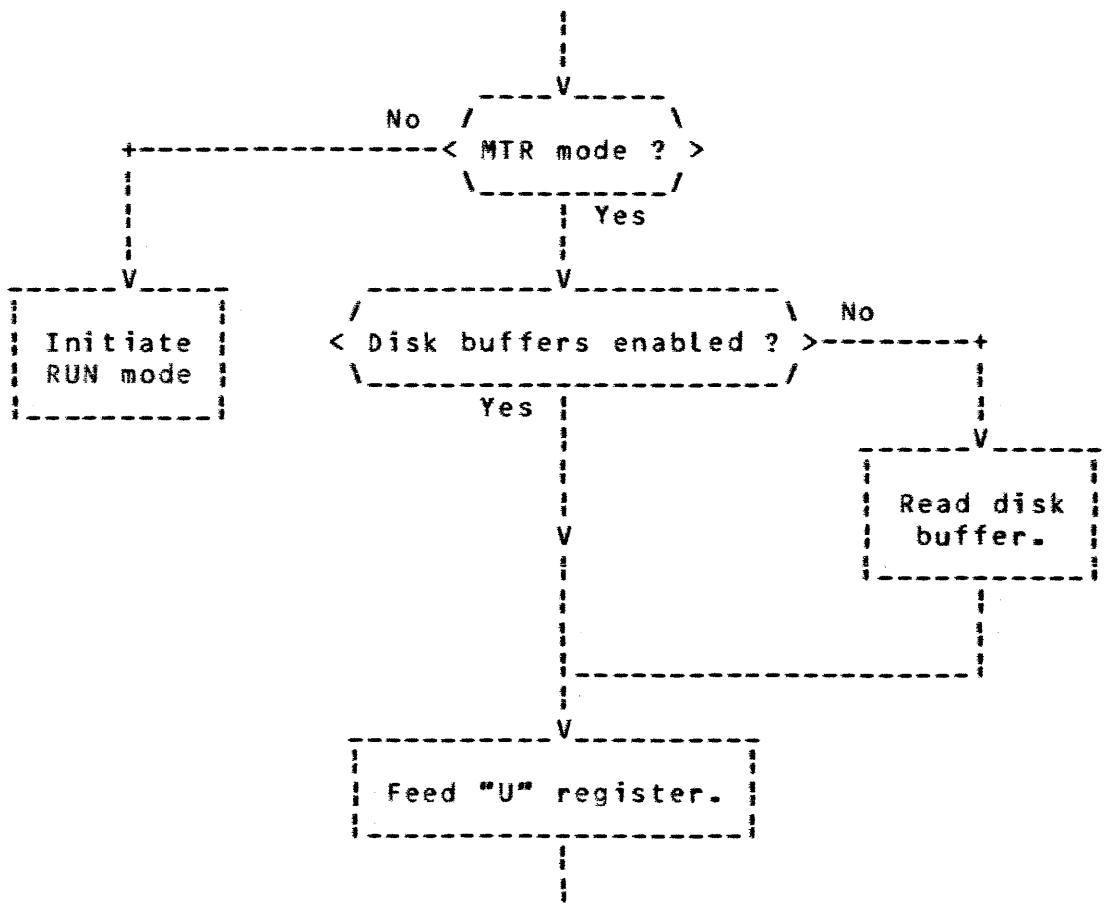
MAC-II TO B1990 CPU COMMUNICATION



SEMANTICS OF THE "GO" COMMAND :

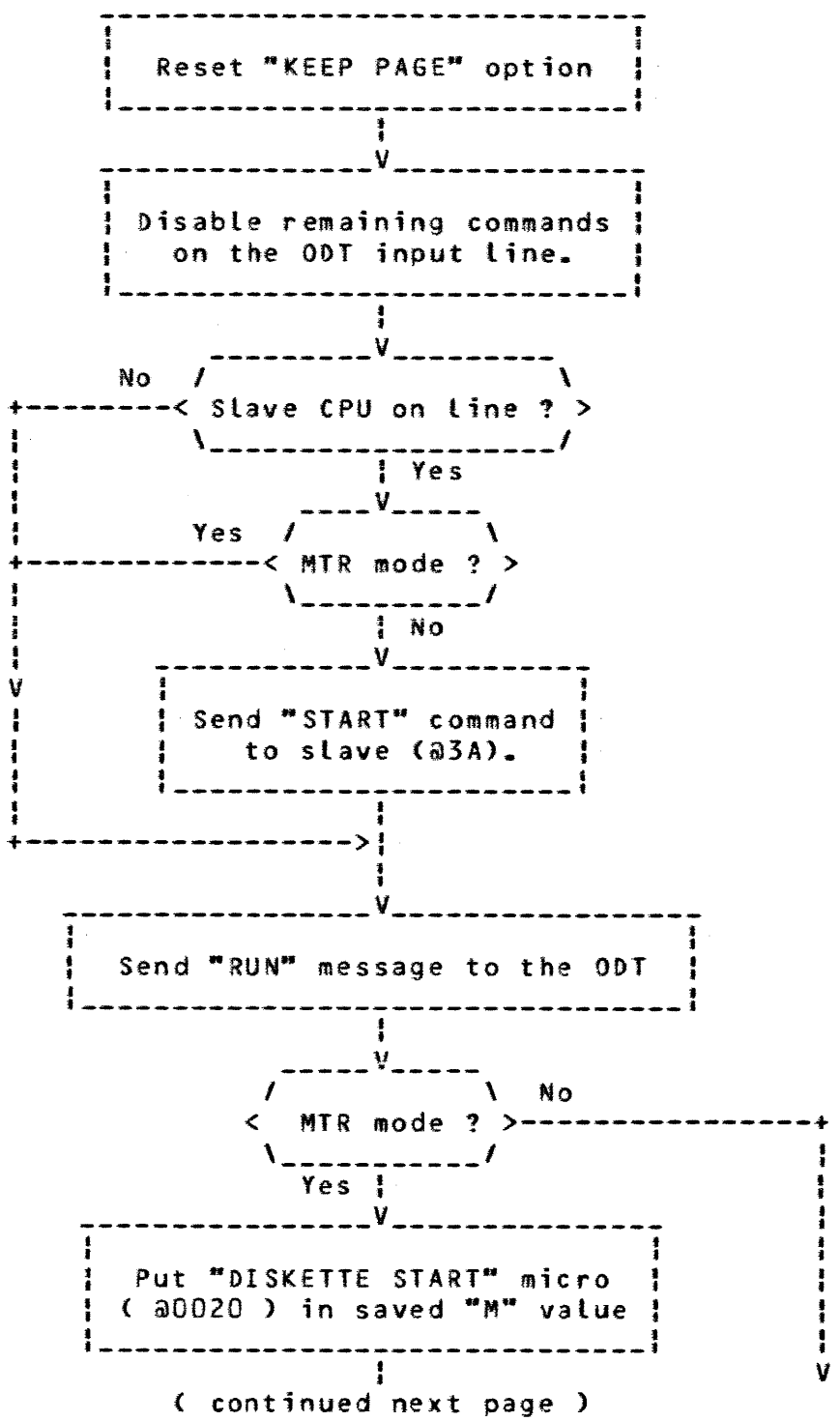
Any such command issued when the system is in MTR mode will cause the firmware to check whether the diskette read buffers have already been filled with valid data from a previously selected ".MTR" file, if not, data will be read from that file, starting from a sector address which logically follows the last used one (mainly the first one), afterwards the transfer to the "U" register will be enabled.

The following flow applies to the process described above :

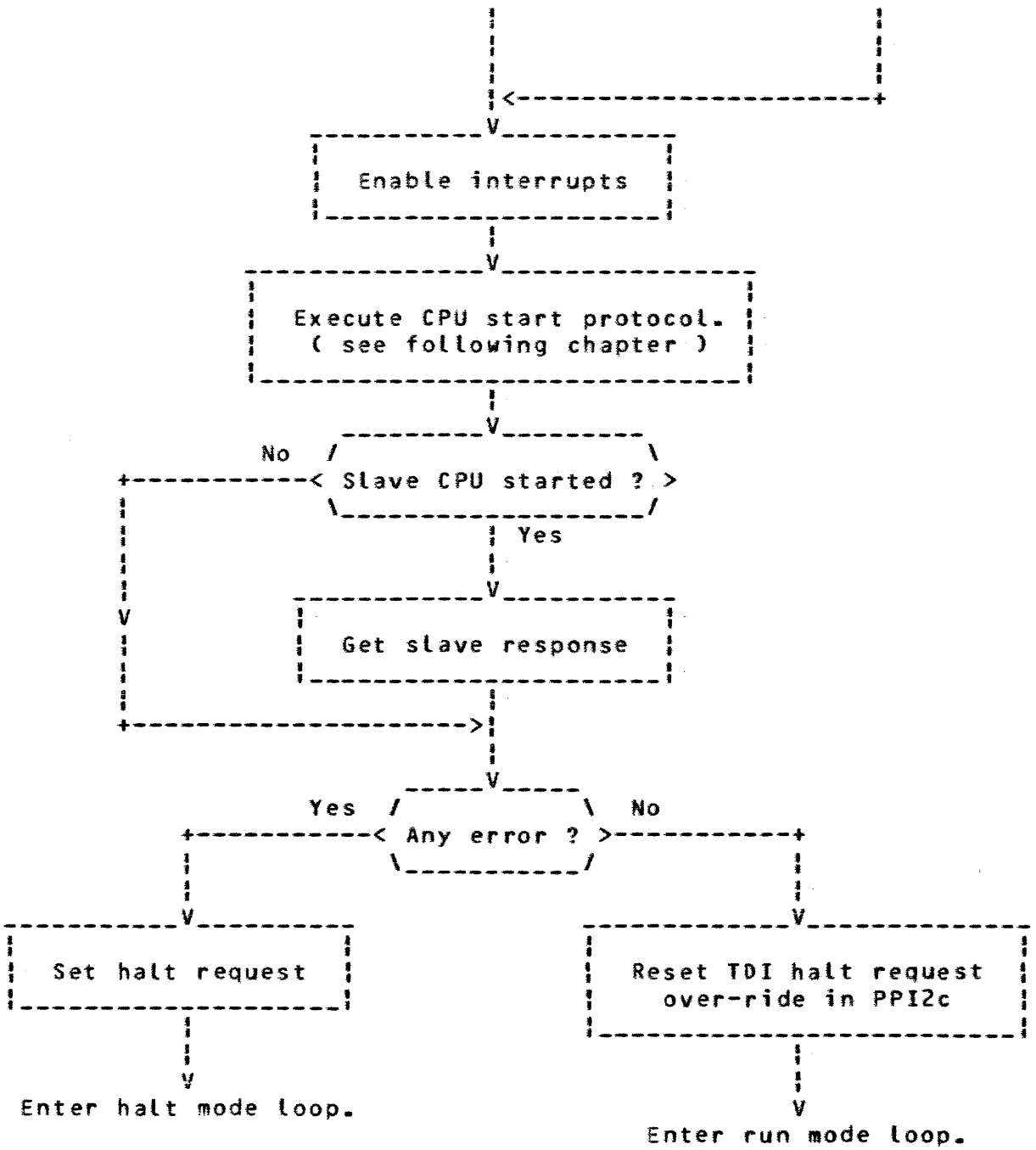


The following graph describes more precisely the sequence of operations required to put both master/slave CPU's in run mode, when either

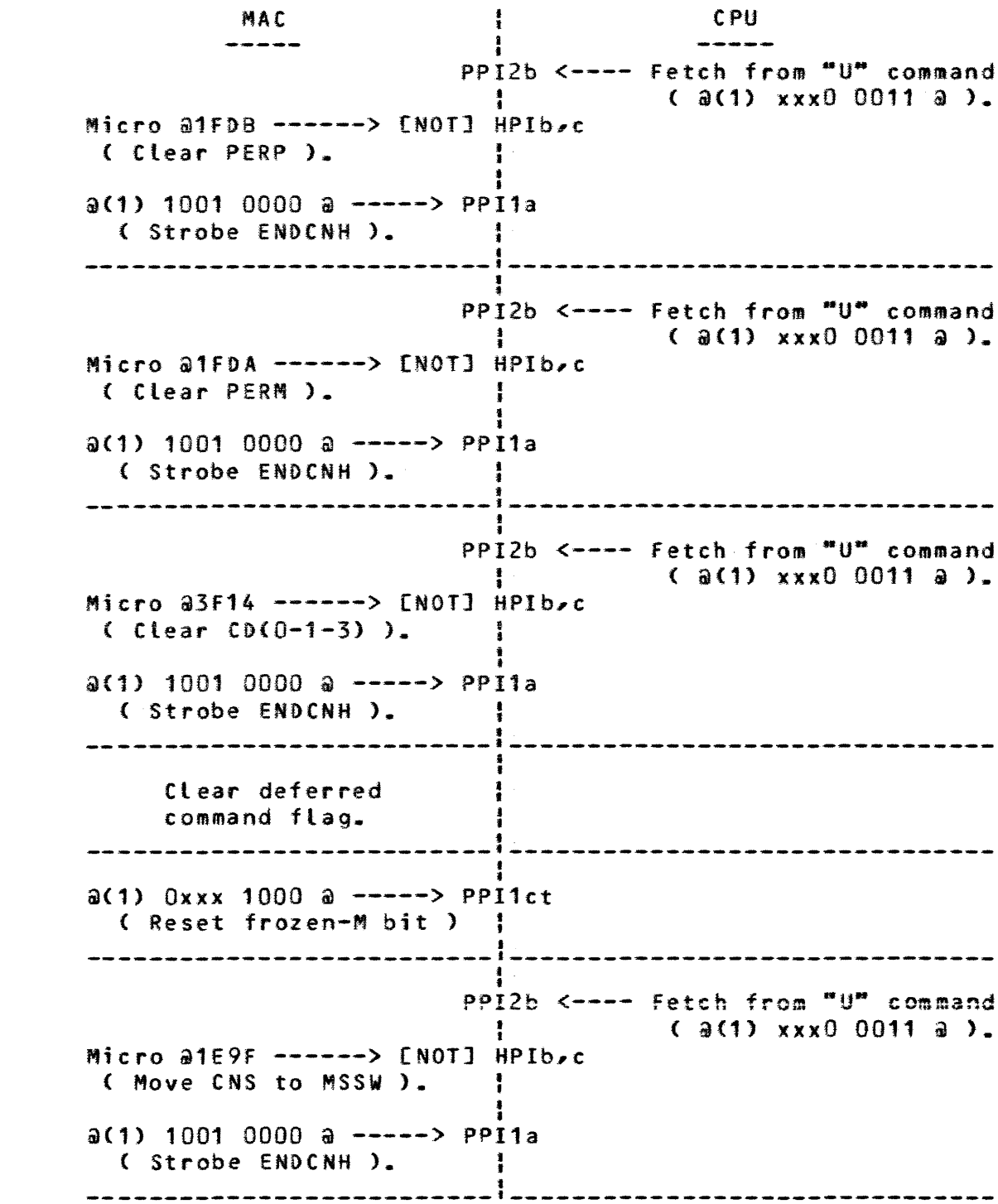
- a "GO" command is entered on the ODT,
- the "HALT/RUN" pushbutton is depressed in halted mode,
- a "HALT RESTART" request is made by the CPU.



(continued next page)

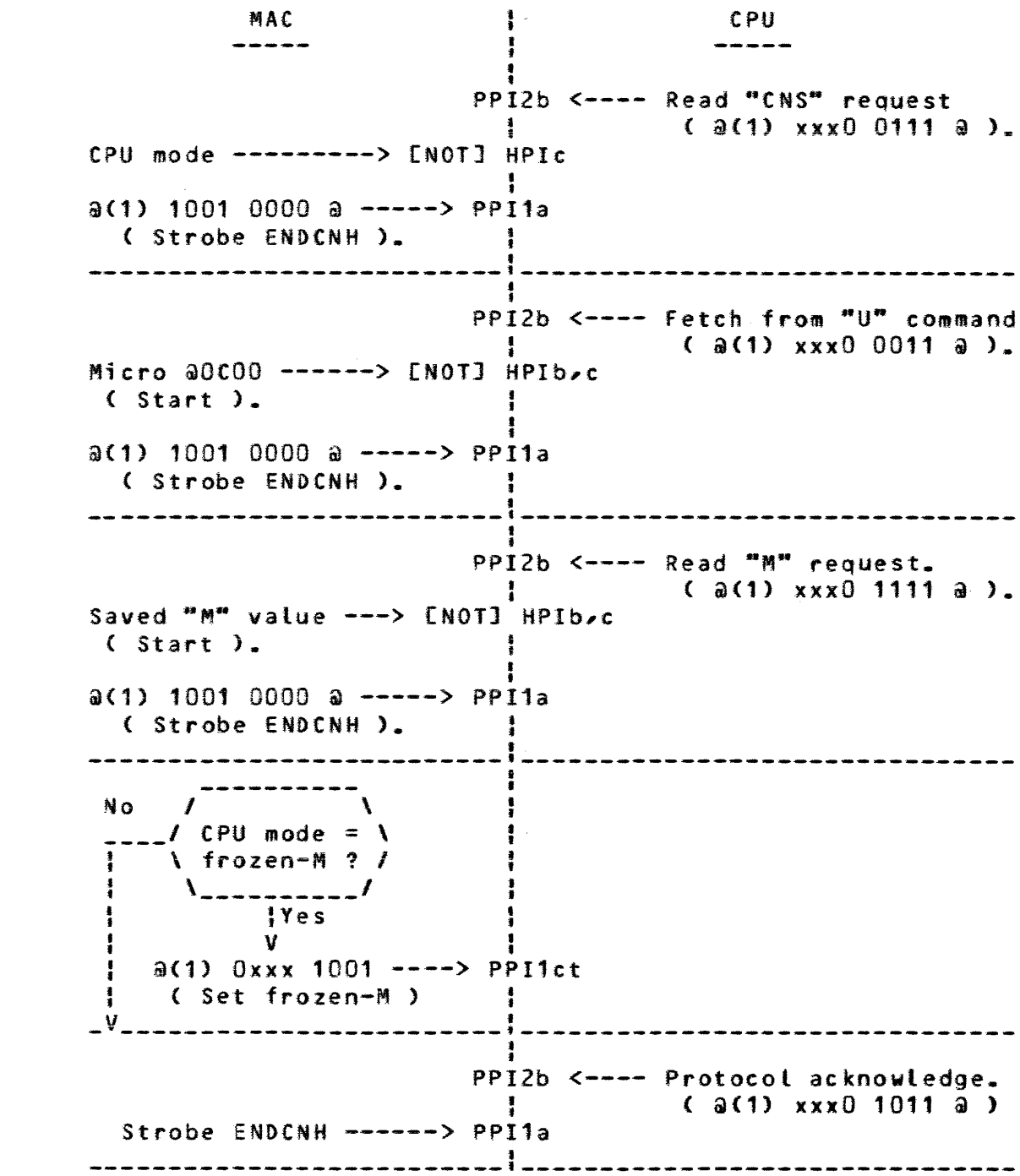

Enable interrupts
Execute CPU start protocol.
(see following chapter)
Slave CPU started ?
No
Yes
Get slave response
Any error ?
Yes
No
Set halt request
Enter halt mode loop.
Reset TDI halt request
over-ride in PPI2c
Enter run mode loop.

CPU START PROTOCOL :

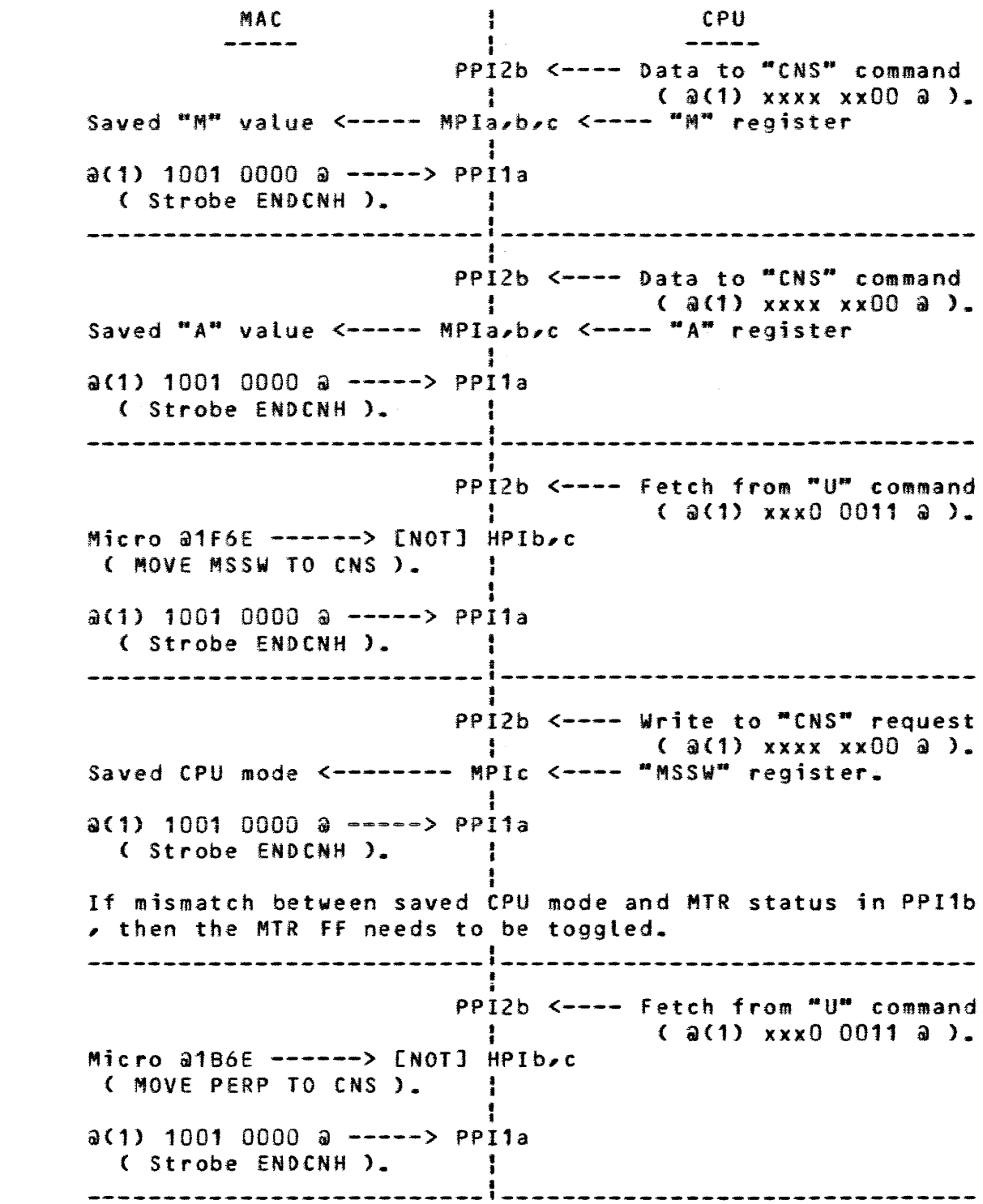


(continued next page)

CPU START PROTOCOL (continued) :

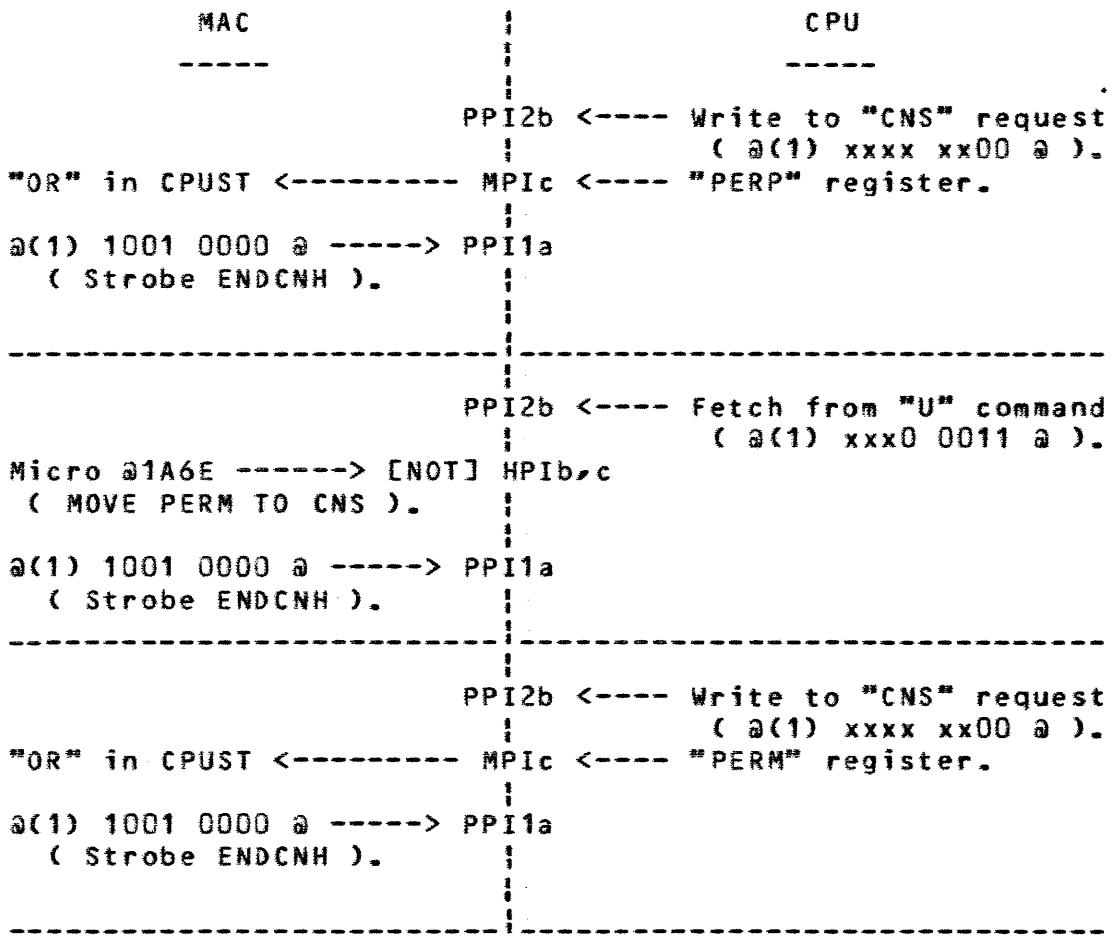


CPU HALT PROTOCOL :



(continued next page)

CPU HALT PROTOCOL (continued) :



The above procedure applies if either the CPU halts, or if the MAC-II is sending a "SYSTEM-CLEAR" command to the CPU in bit# 3 of PPI1c.

SEMANTICS OF THE "CLEAR" COMMAND :

The following sequence of operations will be performed by the MAC-II firmware of the master CPU upon receipt of a "CLEAR" command in halt mode :

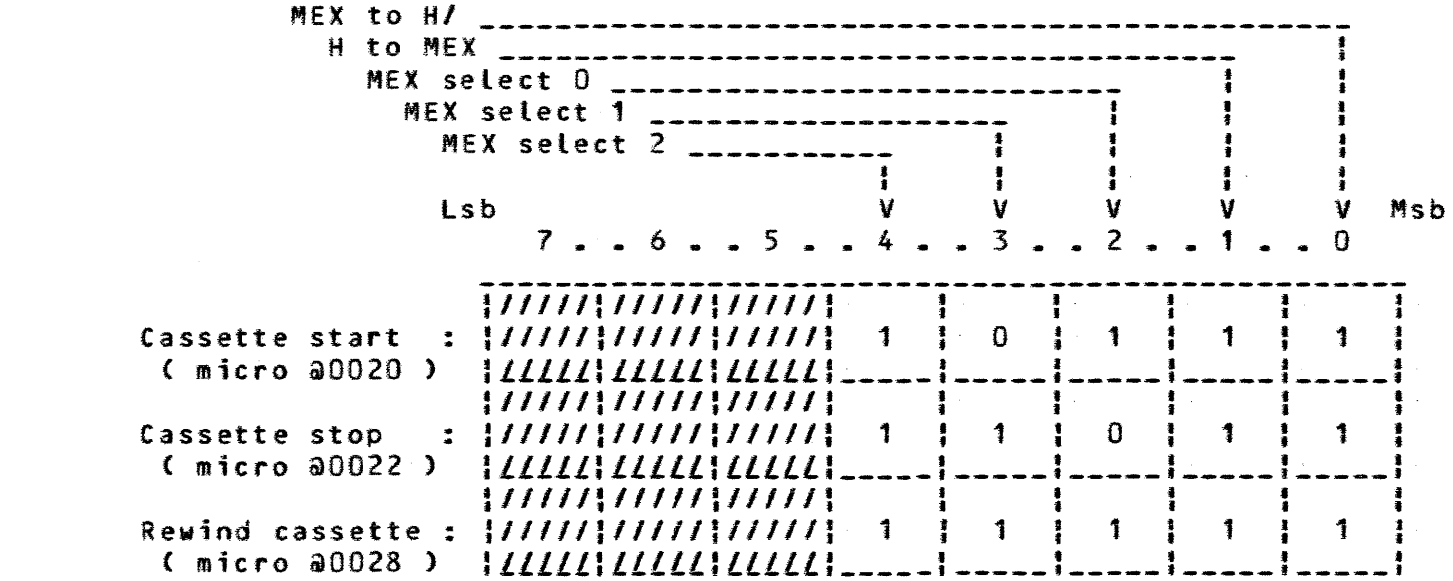
- Send a "SYSTEM-CLEAR" signal in PPI1CNTL (port @93) which will force the CPU to send registers "A" & "M" to the MAC card (see halt protocol).
- Send "MOVE NULL TO A" & "MOVE NULL TO BR" micros to the CPU (@1FE4 & @1FE6 respectively).
- Fill with zeroes the RAM area whose contents will be sent to the "M" register when executing the CPU start protocol.
- Send a "READ-CLEAR-ELOG" micro (@0B0B) micro the the CPU, read the results from the "Y" register (send "MOVE Y TO CNS" micro - @11AE) and store them in RAM.
- If not in cache-only source mode (CPUMD=2), send a clear cache micro (@0005) micro to the CPU.

Up to here, the "CLEAR" protocol is very similar to the one of the previous MAC, but in addition, the MAC-II will have to reload the FAT & directory from the system diskette, in order to allow the firmware to take into account a new diskette setting.

If the slave CPU is present and online, it will perform the same operations (except for the diskette handling), while if it is present but off-line, a RESET operation will just be executed.

SEMANTICS OF THE CASSETTE MANIPULATE MICROS :

These micros generate a request to the MAC-II processor on PPI2b (I/O address @95) with the following format :



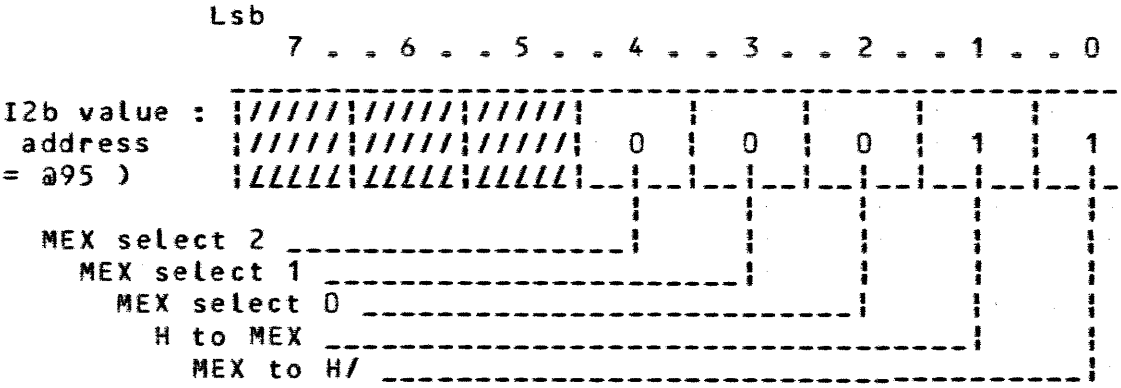
CASSETTE START : will enable data to be transferred to the "U" register from the diskette read buffers, if valid data is available in those buffers, if not, a read operation will take place, a sector pointer in the ".MTR" file will be incremented, and a byte pointer in the read buffers will be reset.

CASSETTE STOP : will invalidate the data in the read buffers of the diskette so that a subsequent CASSETTE START micro will cause next data to be read from the diskette, the sector pointer to be incremented, and the byte pointer to be reset.

CASSETTE REWIND : in addition of the known features of the cassette stop micro (invalidation of the buffers, buffer pointers reset), this instruction will cause the restore mechanism of the diskette to be initiated (the effect is then similar to the "UNLOAD" command of the ODT).

SEMANTICS OF THE "MOVE U TO register" MICRO :

Generates a request to the MAC-II processor on PPI2b according to the following format :



This request will cause 16 bits of data from the read buffers of the diskette to be moved to the "U" register of the CPU, provided those buffers were enabled by either a cassette start micro (run mode) or by a "GO" command from the ODT when the system is in MTR mode (see semantics of the "GO" command).

The transfer to "U" consists of 3 "OUT" micros to the HPPI port of the 2's complement of the data to be sent over, then strobe ENDCNH in PPI1a to release the host CPU.

After this transfer to the "U" register, the byte pointer in the disk read buffers will be incremented by 2, and when it reaches the end address of the buffers, a read operation of the next logical sector within the file will be initiated, the logical sector pointer will be incremented, and the byte pointer will be reset to the start address of the disk read buffers.

Crossings over TP400 cluster boundaries will be accomplished using the copy of the File Allocation Table in the RAM of the MAC-II card created by the execution of a previous "MTR" command from the ODT.

Disk read errors will cause the firmware to strobe the adequate error flag in PPI1a, in addition, the same procedure will be followed if the CPU tries to get data beyond the EOF cluster. (See appendix for End Of File indication on TP400 disks).

CNS REGISTER COMMANDS FORMAT :

All the commands that can be directed to the CNS register may be described by the following table :

	Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	0000 3210 Lsb
CNS value :	xxxx	xxxx	xxxx	xxxx	CDMR	ssss

CDMR = 1xxx : Control commands,
CDMR = 01xx : Deferred commands (master only),
CDMR = 001x : MAC-II typical commands,
CDMR = 0001 : Read/write MAC-II memory commands,
CDMR = 0000 : Immediate action commands,
ssss : Command sub-type.
All xxxx's must be reset for control, deferred, and immediate action commands, and are considered as data for MAC-II commands (will be described in the following paragraph).

Deferred, MAC-II, and immediate action commands are valid only if they were enable by the receipt of a specific control command (CNS = @000081).

All those commands are executed by the MAC processor when a request is made by the host to PPI2b (I/O address @95) with the following format :

MEX to H/
H to MEX
MEX select 0
MEX select 1
MEX select 2

Lsb ... Msb

	7	6	5	4	3	2	1	0
Data to CNS :	//////	//////	//////	//////	//////	//////	0	0
Register to CNS :	//////	//////	//////	1	0	0	1	0
Read CNS :	//////	//////	//////	0	0	1	1	1

CNS COMMAND FORMAT (CONTINUED) :

Whenever the MAC receives a command which implies an output to CNS from the host, the contents of that data will be available on MPIa,b,c (I/O address @80 - @83) and will be stored in the RAM of the MAC card.

If such a command has a "D" flag on (deferred command), the MAC firmware will memorize that the corresponding action has to be taken when the host processor halts, then, in halt mode the firmware main loop will check that there is a command pending, analyze the CNS, and execute the specified request. Note that any control command has to clear the flag which eventually indicates that a deferred command must be executed in halt mode, and any execution of the "START" procedure also clears that flag.

A "READ CNS" request from the host will cause the MAC firmware to read each byte of the CNS from the RAM, complement it, and send it sequentially over to HPIa (I/O address @84), HPIb (address @85), and HPIc (address @86) before releasing the host processor (send the "ENDCNH" strobe in PPI1a).

NEW CNS REGISTER COMMANDS :

A set of new CNS commands has to be introduced in order to allow the host processor :

- to select the name of the ".MTR" file he wants to read through the "U" register between diskette start/stop.
- to cause MAC-II firmware to read/write any particular 512 bytes disk sector to/from its RAM memory.
- to access (on input & output) the MAC-II RAM memory.

These commands may be summarized as follows :

Command name :	Value :
Reload FAT & directory.	00 00 21
Specify filename bytes 1 & 2.	xx xx 22
" " " 3 & 4.	xx xx 23
" " " 5 & 6.	xx xx 24
" " " 7 & 8.	xx xx 25
Read absolute sector.	xx xx 26
Write absolute sector	xx xx 27
Read RAM.	xx xx 11
Write RAM address	xx xx 12
Write RAM data.	xx xx 13
Write ODT mode.	xx xx 14

Where all xx's represent 8 bits command parameters which will be described later on.

Those commands will only be valid if enabled by the execution of the CNS @000081 command from the host, and will be nak'ed if the read buffers of the diskette are enabled (namely between a CASSETTE START and a CASSETTE STOP micro).

RELOAD DISK CNS COMMAND : [00 00 21]

The following command issued by the host to the CNS register will cause the firmware to read the FAT (file allocation table) sector and the 9 sectors of directory from the diskette currently inserted in the drive.
The RAM address of the buffer which contains this information will be available from the CNS response, so that it can be read later on using the "READ RAM" CNS command.

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
0000	0000	0000	0000	0010 CDMR	0001 ssss

Response format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
Lower byte of buffer address		Upper byte of buffer address		//////////	

Upper and lower bytes of the 16 bits directory buffer address are inverted in the CNS response - refer to 8080 microprocessor type of addressing.

SPECIFY FILE NAME BYTES 1 & 2 COMMAND : [xx xx 22]

The following command issued by the host to the CNS register will cause the firmware to update the first and the second bytes of the MTR filename to be searched later on with the first & second bytes of the command, the CNS response is meaningless.

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
1 st byte of MTR filename.		2 nd byte of MTR filename.		0010 CDMR	0010 ssss

SPECIFY FILE NAME BYTES 3 & 4 COMMAND : [xx xx 23]

The following command issued by the host to the CNS register will cause the firmware to update the third and the fourth bytes of the MTR filename to be searched later on with the first & second bytes of the command, the CNS response is meaningless.

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
3 rd byte of MTR filename.		4 th byte of MTR filename.		0010 CDMR	0011 ssss

SPECIFY FILE NAME BYTES 5 & 6 COMMAND : [xx xx 24]

The following command issued by the host to the CNS register will cause the firmware to update the fifth and the sixth bytes of the MTR filename to be searched later on with the first & second bytes of the command, the CNS response is meaningless.

Command format :

Msb 15 14 13 12	11 10 9 8	7 6 5 4	Lsb 3 2 1 0
5 th byte of MTR filename.	6 th byte of MTR filename.	0010 CDMR	0100 ssss

SPECIFY FILE NAME BYTES 7-8 & SEARCH COMMAND : [xx xx 25]

The following command issued by the host to the CNS register will cause the firmware to update the seventh and eighth bytes of the MTR filename and to search for that file name pattern within the directory buffer in RAM, which should have been read from disk by a previous command ; the address of the directory buffer will be returned in the CNS response.

Command format :

Msb 15 14 13 12	11 10 9 8	7 6 5 4	Lsb 3 2 1 0
7 th byte of MTR filename.	8 th byte of MTR filename.	0010 CDMR	0101 ssss

Response format :

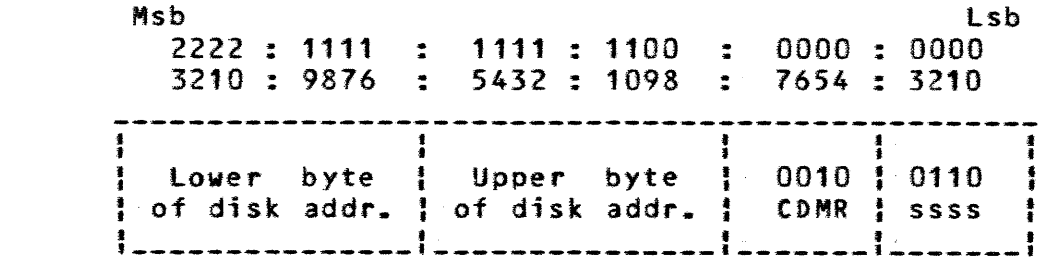
Msb 15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	Lsb
Lower byte of buffer address	Upper byte of buffer address	Search status	

@(1) 0000 0110 @ (ACK) : successful search <--
@(1) 0001 0101 @ (NAK) : file not found <-----

READ DISKETTE CNS COMMAND : [xx xx 26]

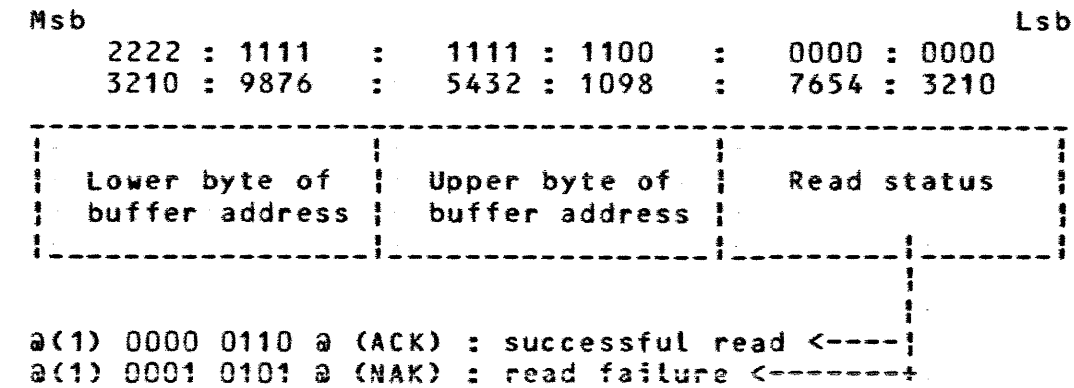
The following command issued by the host to the CNS register will cause the firmware to initiate a read disk operation on the disk sector number specified by the command; if successful, the data read will be available from the RAM locations specified in the CNS response (see "READ RAM" CNS command).

Command format :



Upper and lower bytes of the 16 bits disk address need to be inverted in the CNS command - refer to 8080 microprocessor type of addressing.

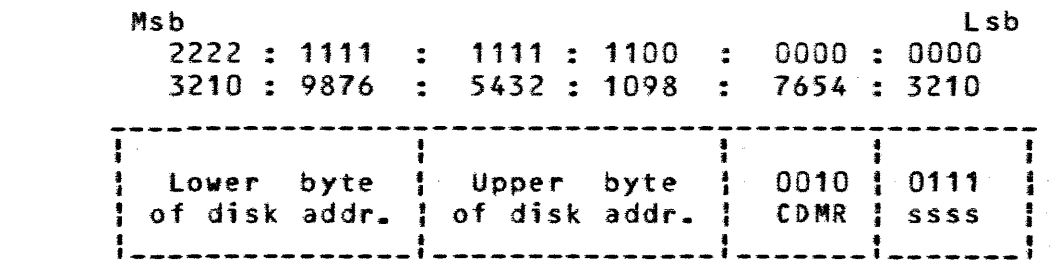
Response format :



WRITE DISKETTE CNS COMMAND : [xx xx 27]

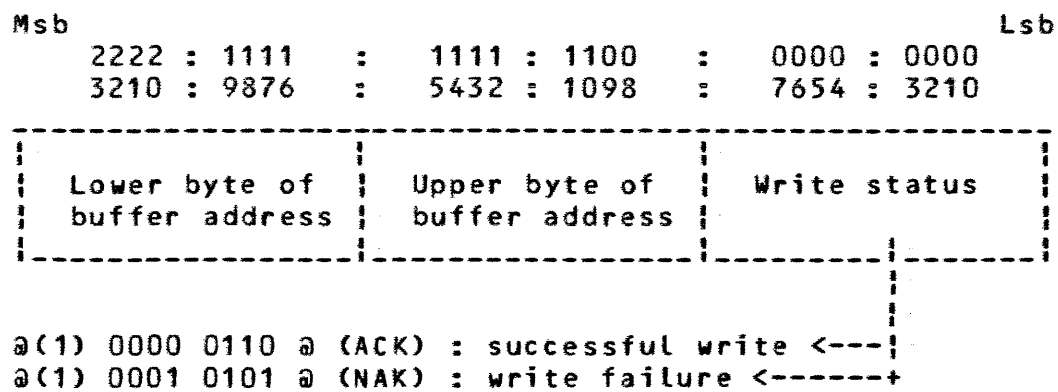
The following command issued by the host to the CNS register will cause the firmware to initiate a write disk operation on the disk sector number specified by the command.
The 512 bytes buffer used to perform this operation will have the same address as the one used to perform the read operation (see command above) and may eventually have been filled from the host using the "WRITE RAM ADDRESS" and "WRITE RAM DATA" CNS commands (see following paragraphs).

Command format :



Upper and lower bytes of the 16 bits disk address need to be inverted in the CNS command - refer to 8080 microprocessor type of addressing.

Response format :



READ MACII RAM CNS COMMAND : [xx xx 11]

The following command issued by the host to the CNS register will cause the firmware to read 24 bits of RAM at the address specified by the command, which will be presented to the MEX lines, and available to the host by a "MOVE CNS TO register".

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
Lower byte of RAM addr		Upper byte of RAM addr		0001 CDMR	0001 ssss

Upper and lower bytes of the 16 bits RAM address need to be inverted in the CNS command - refer to 8080 microprocessor type of addressing.

Response format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
RAM contents at [address]		RAM contents at [address]+1		RAM contents at [address]+2	

This command might be useful for programs who need to access data which are resident in the MAC memory, such as diskette directory, disk data, pack identifier, or current MTR filename.

WRITE MACII RAM ADDRESS COMMAND : **[xx xx 12]**

The following command issued by the host to the CNS register will cause the firmware to memorize the first 2 bytes of the MEX lines as the address where any subsequent "WRITE RAM DATA" command will update then RAM memory.

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
Lower byte of RAM addr		Upper byte of RAM addr		0001 CDMR	0010 ssss

Upper and lower bytes of the 16 bits RAM address need to be inverted in the CNS command - refer to 8080 microprocessor type of addressing.

The response format is meaningless.

WRITE MACII RAM DATA CNS COMMAND : [xx xx 13]

The following command issued by the host to the CNS register will cause the firmware to write 16 bits of data at the RAM address specified by a previous "WRITE RAM ADDRESS" command.

Command format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
Lower byte of data		Upper byte of data		0001 CDMR	0011 ssss

Upper and lower bytes of the 16 bits RAM address need to be inverted in the CNS command - refer to 8080 microprocessor type of addressing.

After completion of this update, the RAM address where the following such command should take place will automaticly be incremented by 2 (bytes), so that only one "WRITE ADDRESS" command is required to fill a buffer of contiguous memory locations.

Response format :

Msb 2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	Lsb 0000 3210
Lower byte of next address.		Upper byte of next address.			

WRITE ODT MODE CNS COMMAND : [xx xx 14]

The following command issued by the host to the CNS register will cause the firmware store the medium byte of the command in the memory location used by the screen formatter to specify the kind of menu that will be displayed on the screen in halt mode.

Command format :

Msb 2222 : 1111 3210 : 9876	1111 : 1100 5432 : 1098	0000 : 0000 7654 : 3210 Lsb
//////////////	Code for ODT screen mode	0001 CDMR / 0100 ssss

OPR page	0000 : 0000
FE page	0000 : 0001
MAC page	0000 : 0010
REG page	0000 : 0011
STACK page	0000 : 0100
CKEY page	0000 : 0101
S-MEM 16 page	0000 : 0110
S-MEM 24 page	0000 : 0111
S-MEM 39 page	0000 : 1000
DIR page	0000 : 1001
TEXT page	0000 : 1010

The response code is meaningless.

CHANGES TO EXISTING CNS COMMANDS :

The response to the "GET STATUS VECTOR" command should be modified in order to reflect the presence of a 5 1/4 disk as a bootstrap device instead of a cassette, so that any program wishing to use the "U" register will be able to know whether or not he has to use the file name selection commands (namely "SYSTEM/LOAD.CAS").

The response to a "GET STATUS VECTOR" CNS command (value = @000002) will be formatted as follows :

Msb Lsb

2222 : 1111	1111 : 1100	0000 : 0000
3210 : 9876	5432 : 1098	7654 : 3210

0000	00SR	Firmware level number	Contents of PPI1b

R = 1 : remote switch on,
S = 1 : synchronous link on (if R=1),

MAC card firmware level numbers are :
- @(1) 0001 0000 @ for MAC-I (H9),
- @(1) 0010 0000 @ for MAC-II (H10),

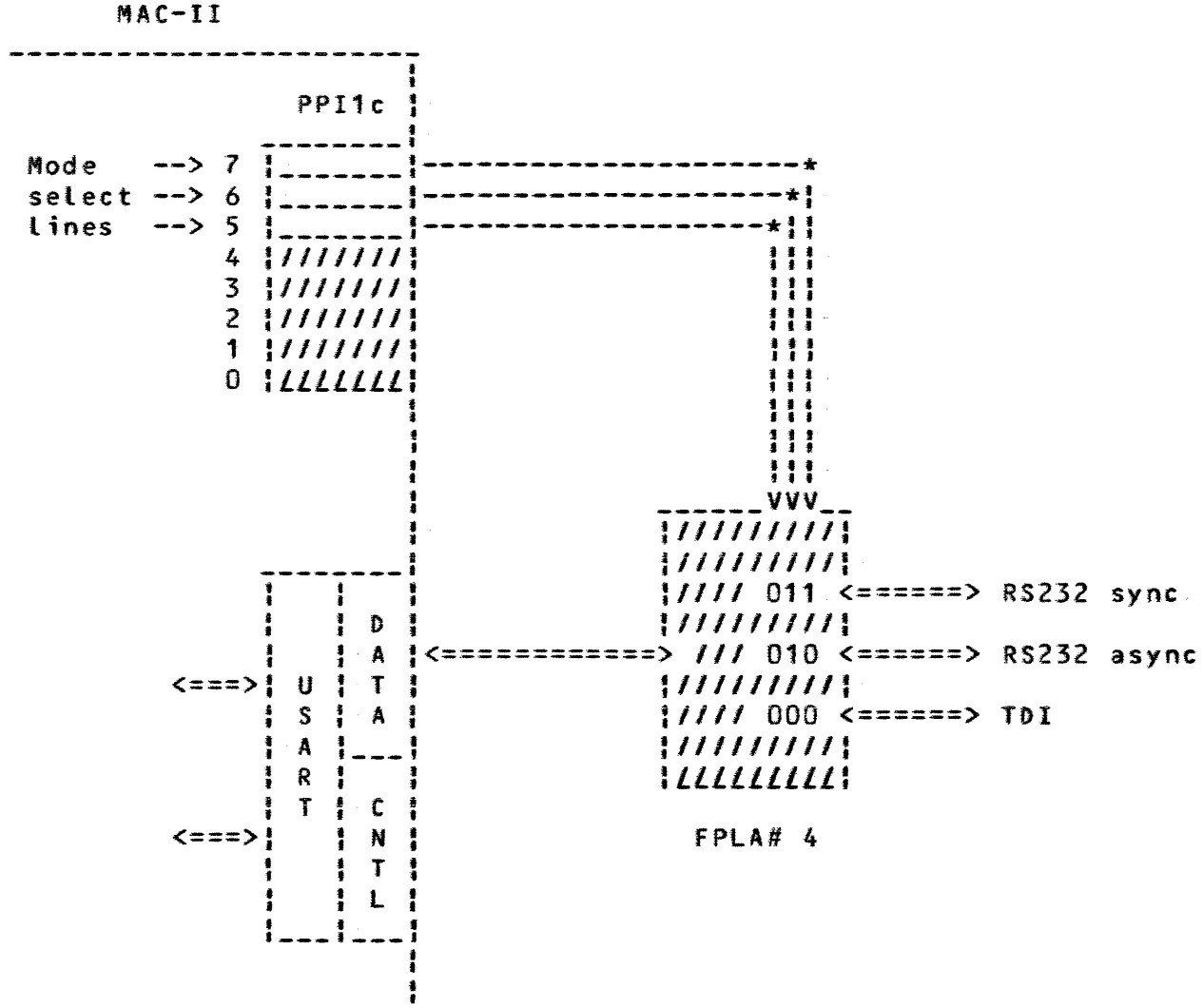
See appendix for a complete description of the bit mapping of PPI1b, found in the least significant byte of the CNS response.

In addition, the users of the "MTR restart" command, i.e. move @000041 to CNS, must be aware that, before they issue such a command, the file identifier field in MAC RAM must have been filled with a valid pattern, otherwise a NAK response will be provided by the firmware.

In case of parity detection in the RAM of the MAC card, a Non Maskable Interrupt will force the firmware to enter a loop which will give a NAK response to all the requests issued by the host to the CNS register.

MAC-II ODT COMMUNICATION

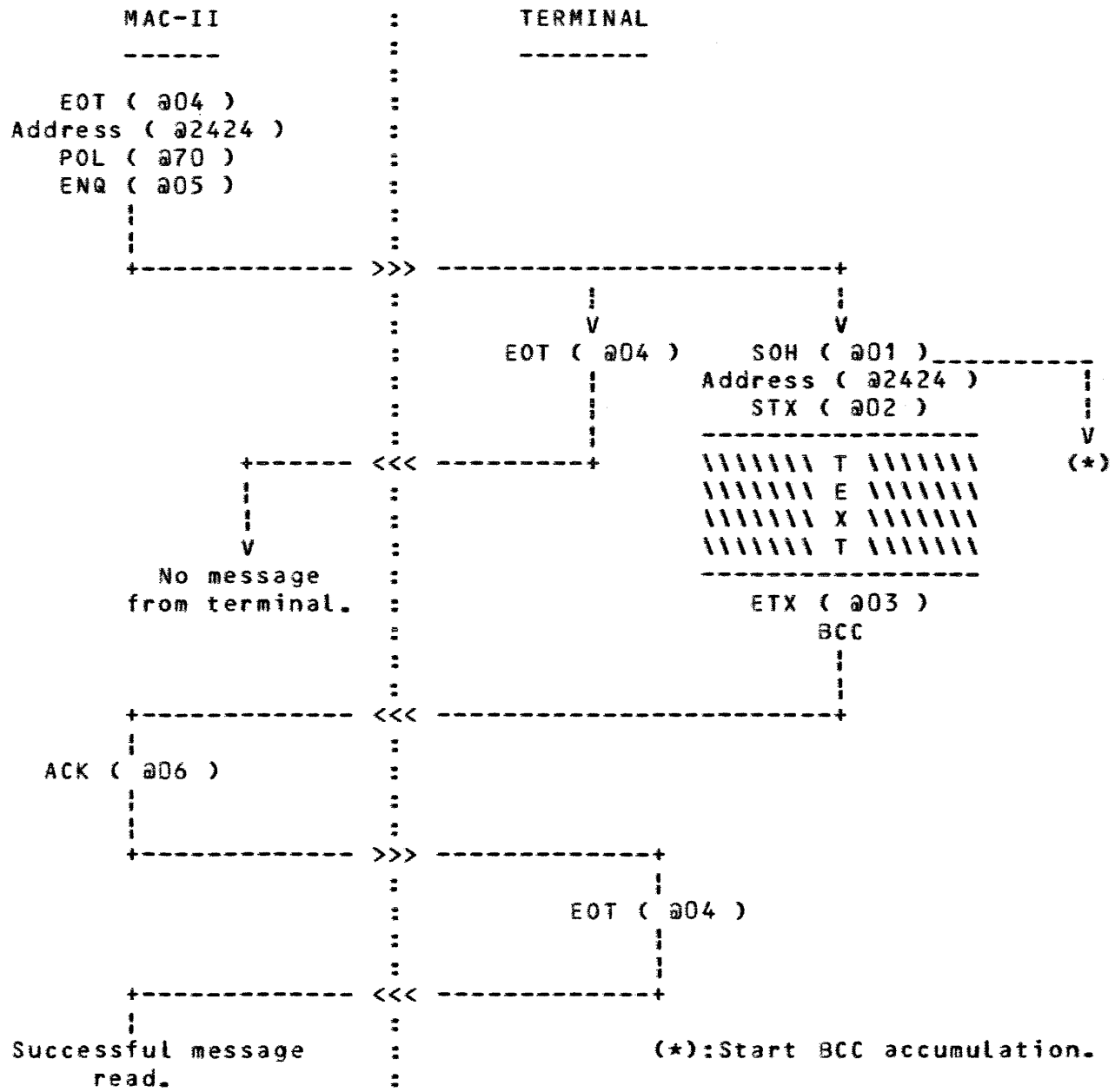
The messages of the MAC card can be directed through the USART to either a local ODT (using a TDI line) or a remote ODT (using an RS-232 sync or a-sync line) by an appropriate setting of the mode select lines of PPI1c - see drawing below :



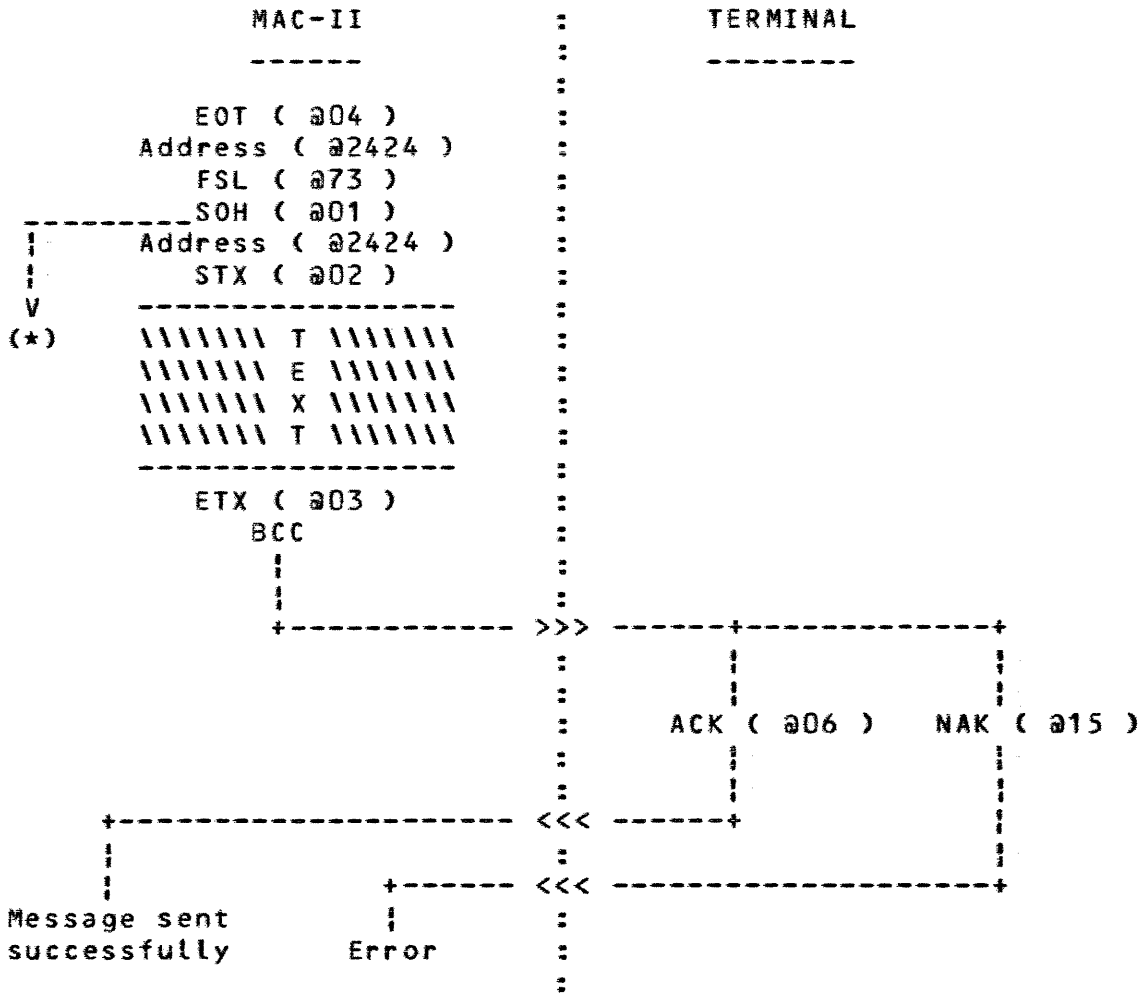
See appendix for a more complete description of mode select lines and 8251 USART programming.

In halt mode, the MAC-II firmware communicates to a local ODT using a Two-wire Direct Interface line at 9600 baud, with a terminal whose physical address is "$$" (@2424).
The protocols used are "FAST SELECT" (write operations) and "POLL" (read operations), without transmission number.
The following pages summarize the data interchange made to perform this :

"POLL" procedure :



"FAST SELECT" PROCEDURE :



(*) : Start BCC accumulation.

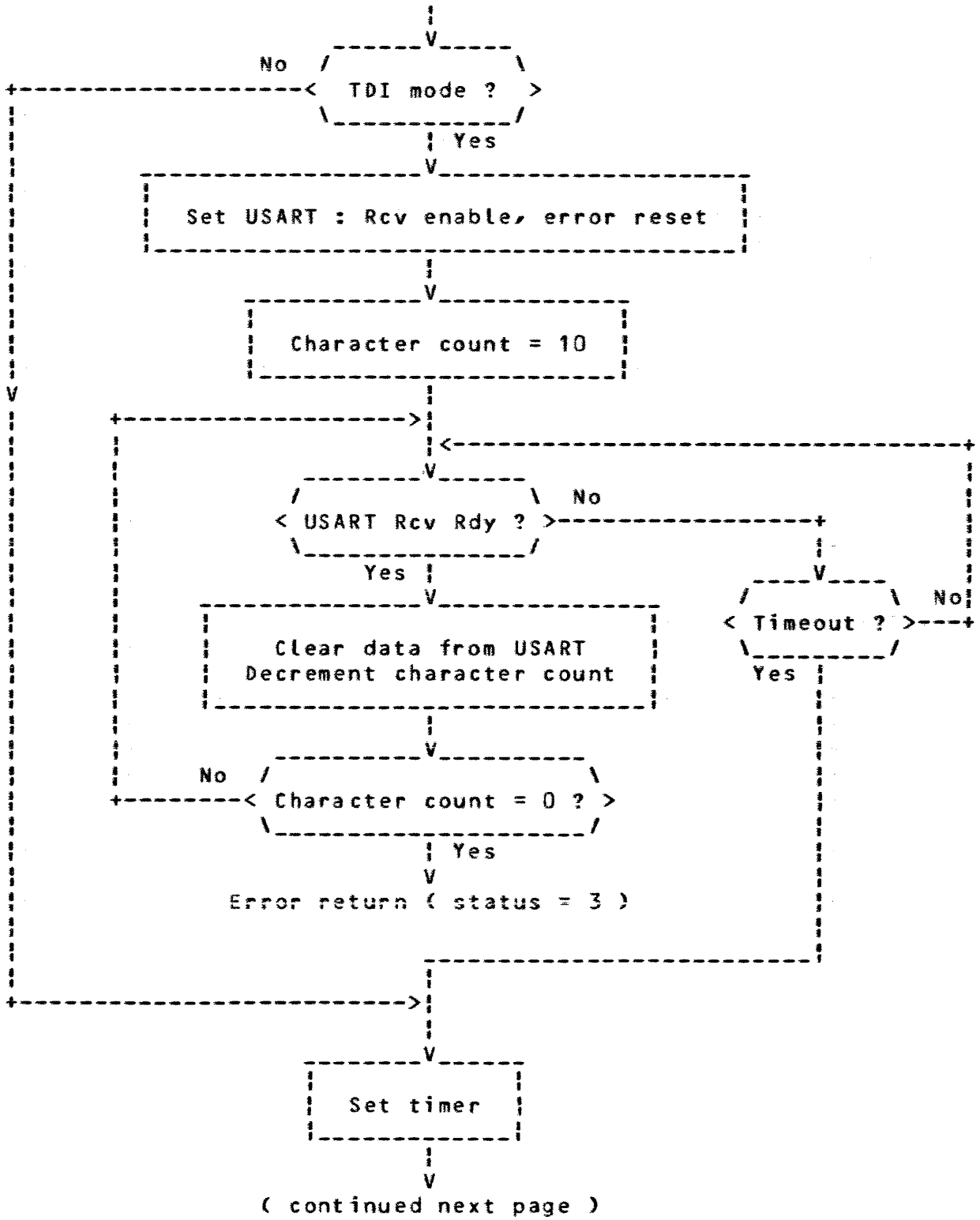
The Block Check Code (BCC) will in this case be a simple LRC (Longitudinal Redundancy Check - exclusive OR of all characters following SOH, up to and including ETX).

The USART mode word for the transmission from/to the local ODT will be equal to @7A, meaning 1 stop bit, even parity, 7 bits per character, Baud factor = 16 x (see appendix for further details).

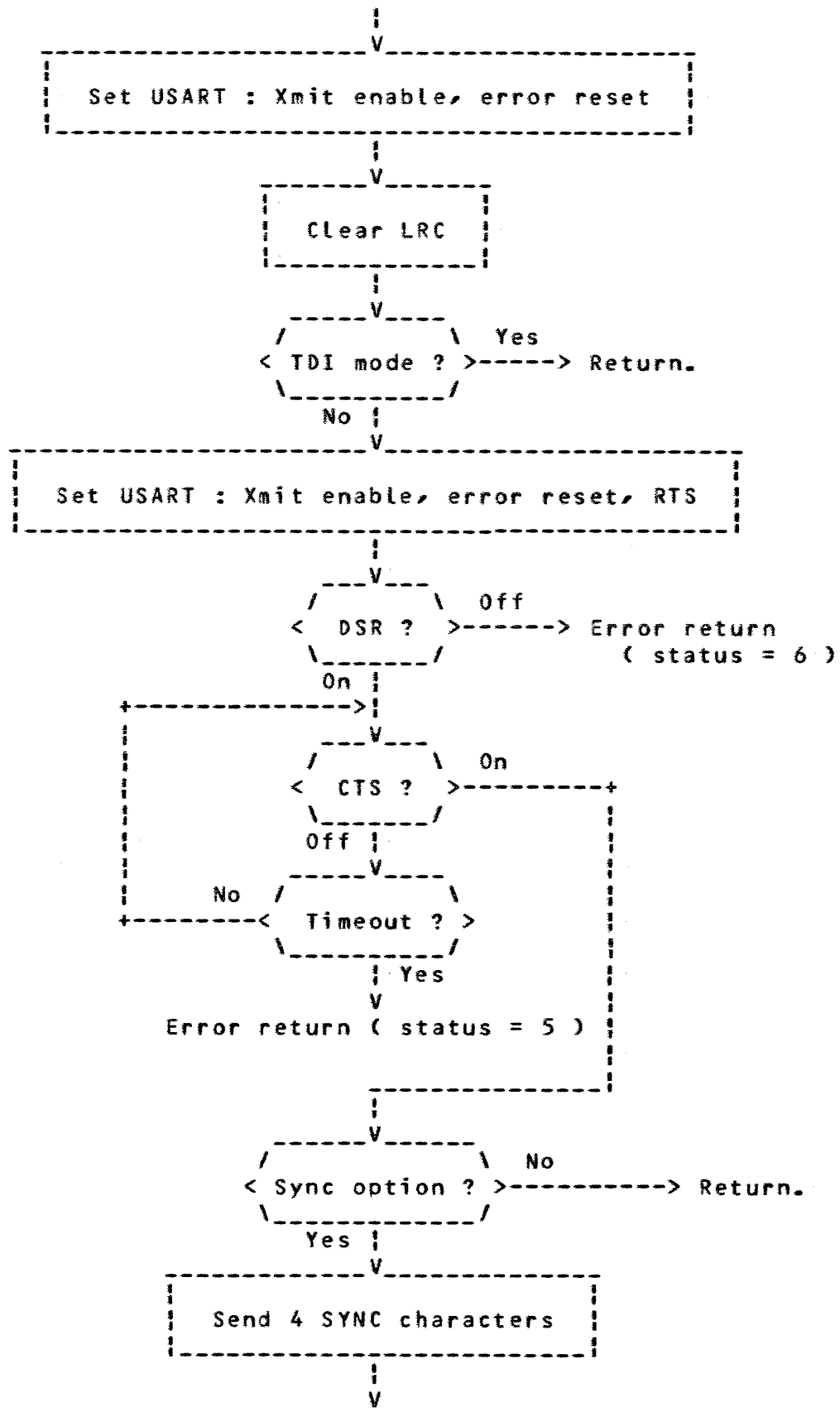
The same procedures also apply to a remote ODT, through a RS-232 line accessible via the USART, provided an appropriate command had turned the remote option on. The remote link, as selected by the "MODE" bits in PPI1c, may be either SYNC or ASYNC (the latter with 3 possible values of the Baud rate 300, 1200 or 1800 Bps).

USART TRANSMIT SEQUENCE :

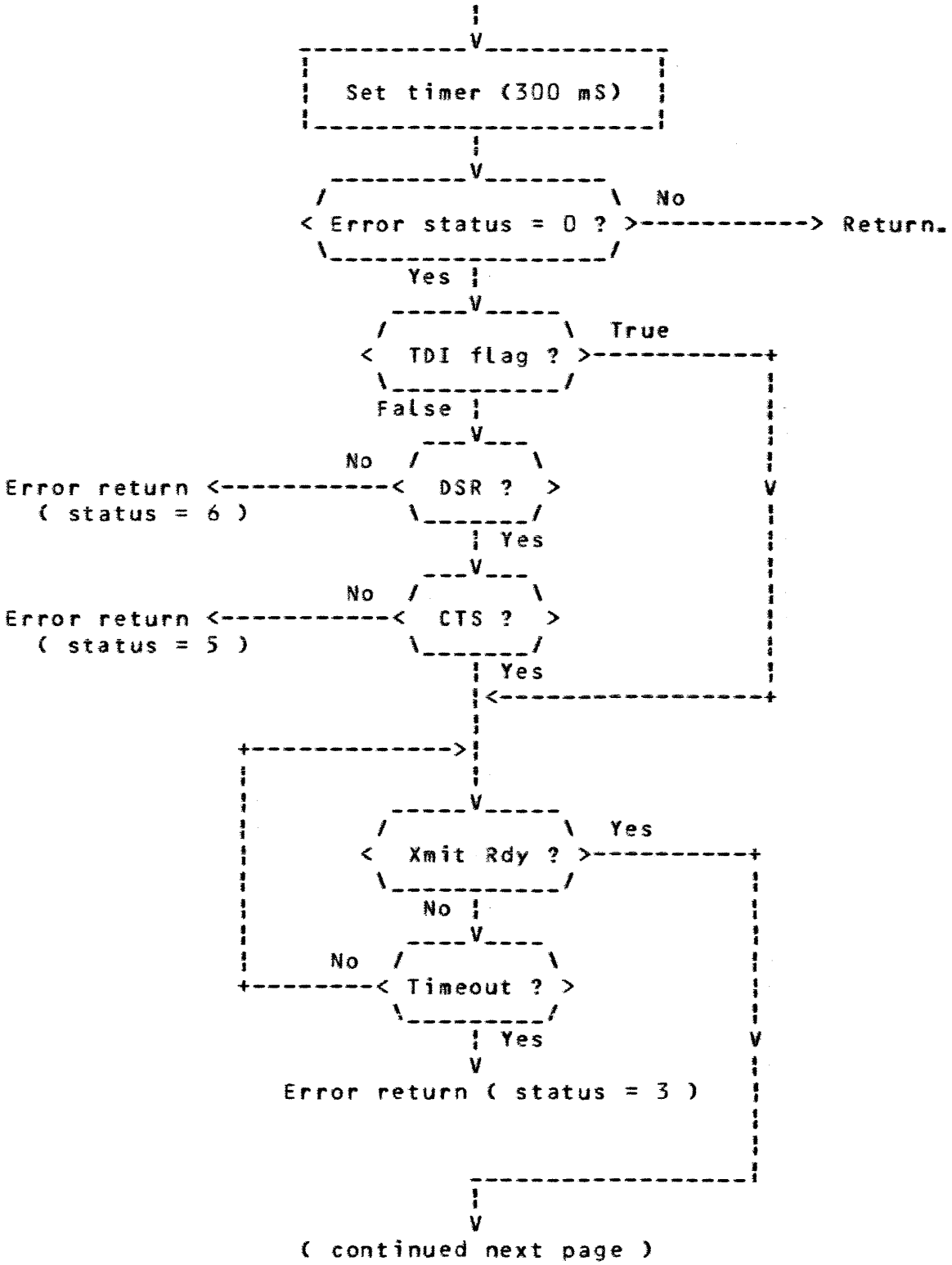
The following flow describes the procedure followed by the MAC-II firmware to initiate a transmit sequence through the USART :



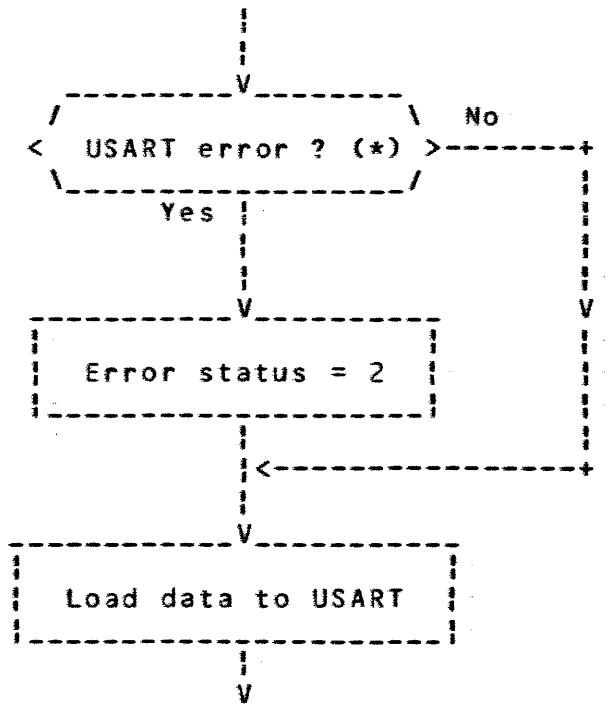
(continued next page)


Set USART : Xmit enable, error reset
Clear LRC
TDI mode ?
Yes
Return.
No
Set USART : Xmit enable, error reset, RTS
DSR ?
Off
Error return
(status = 6)
On
CTS ?
On
Off
Timeout ?
No
Yes
Error return (status = 5)
Sync option ?
No
Return.
Yes
Send 4 SYNC characters

The following sequence is used to send a character through the USART, once the transmit procedure has been initiated by the previous graph.

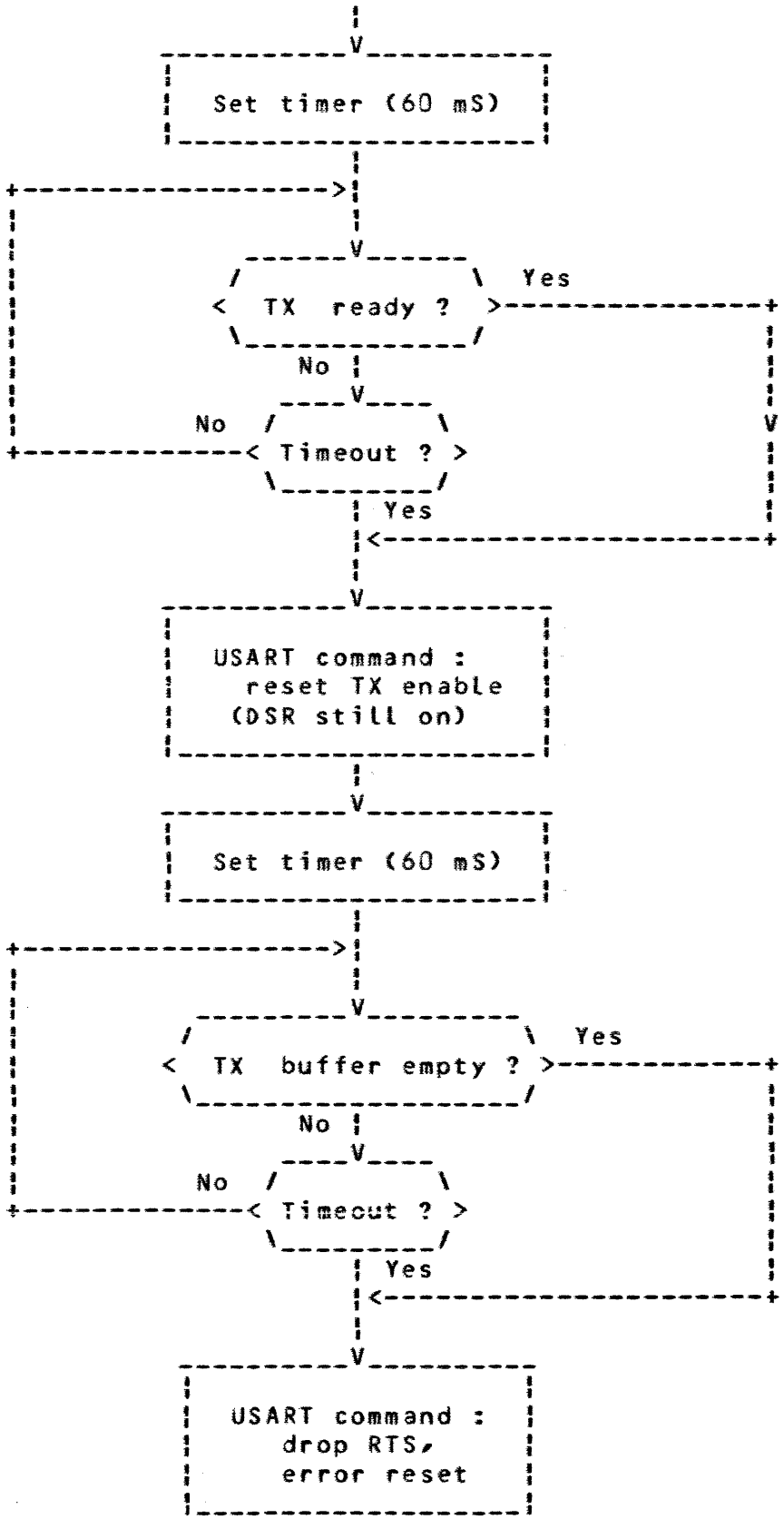


(continued next page)



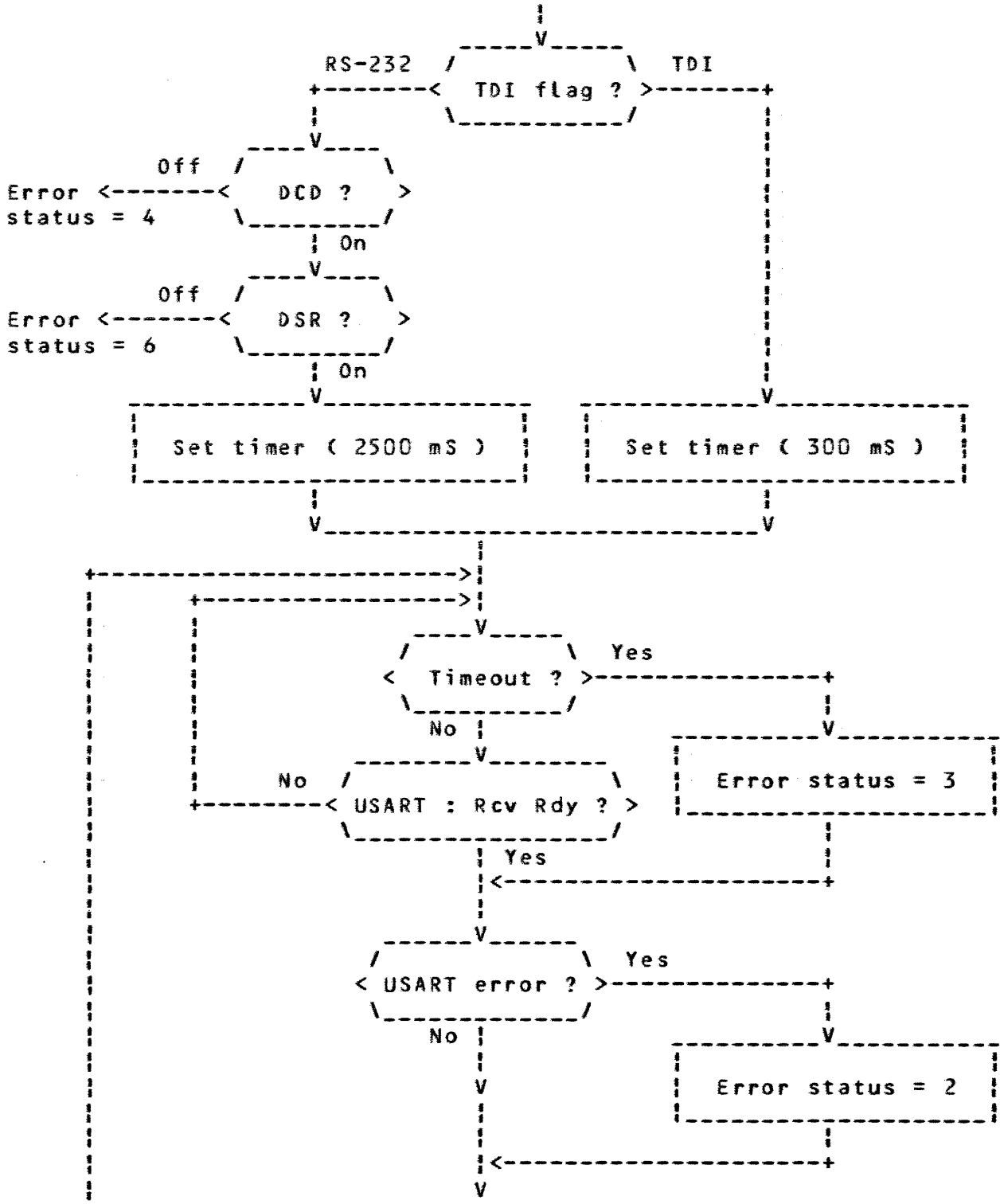
(*) USART errors - frame error,
- over-run,
- parity error.

All data transmit sequences through the USART end up according to the procedure described by the graph on following page.
Note that transmit enable (bit# 0 of USART command word) is reset as soon as the datacomm inteface chip is ready to accept one more character after the end of the message, while the Request To Send signal (RTS-bit# 5 of USART command word) is lowered whenever all characters have left the transmit buffer.

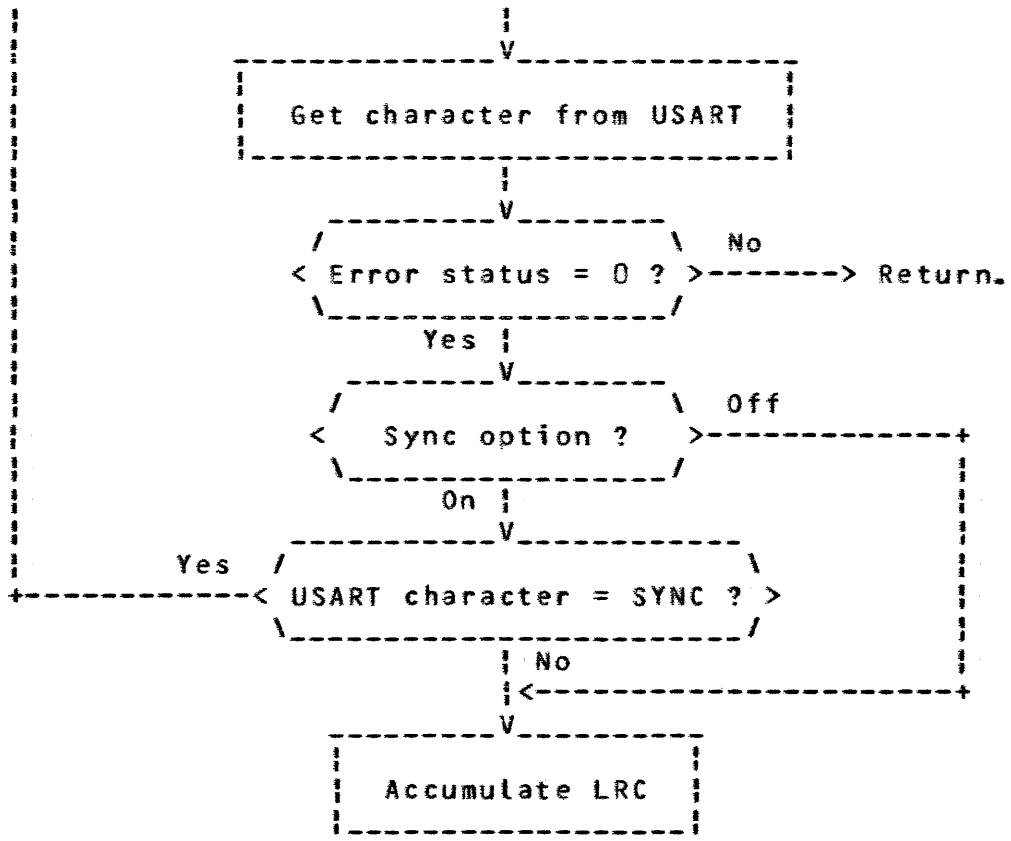

Set timer (60 mS)
TX ready ?
Yes
No
Timeout ?
Yes
USART command :
reset TX enable
(DSR still on)
Set timer (60 mS)
TX buffer empty ?
Yes
No
Timeout ?
Yes
USART command :
drop RTS,
error reset

USART RECEIVE SEQUENCE :

The following flow applies when the MAC-II firmware has to fetch a character from the USART (control port @B1, data port @B0)



(continued next page)



NOTES : USART errors include parity error, overrun & frame error

- A special handling has to be done in SYNC mode in order not to disregard SYN (@16) characters when receiving the LRC of a poll sequence.

- Prior to enter this flow, the Rcv enable, and eventually SYN detect flags must have been set in the USART command as well as mode select in PPI2c & control byte in PTMRC.

- The async ODT transactions will use PTMR1 in mode 3 (square wave rate generator) according to the terminal speed i.e. :

  TDI ..... @0D00 (9600 Baud)
  Rs 232 async ..... @6800 (1200 Baud)
  " " ..... @A001 (300 Baud)
  " " ..... @4500 (1800 Baud)

  while PTMR2 will be used in mode 0 (timeouts will be reported in PPI2a, bit# 7).

According to the flows above, a one byte result status will be returned by the ODT handler on each local (SPOST) or remote (RMPST) terminal transaction whose signification can be interpreted as follows :

SPOST/RMPST value	POLL sequence	SELECT sequence	
0	Valid message	Message ACK'ed	
1	Message NAK'ed	Transmission error	(*)
2	Parity / frame / overrun error		
3	<- Timeout error ->		
4	<- DCD error ->		
5	<- CTS error ->		
6	<- DSR error ->		
255	EOT received	//////////////////	

(*) : Ten retries will be performed before reporting select transmission errors.

MASTER -> SLAVE MAC-II COMMUNICATION :

Performed by a 1 byte communication over a TDI line, with the following signification :

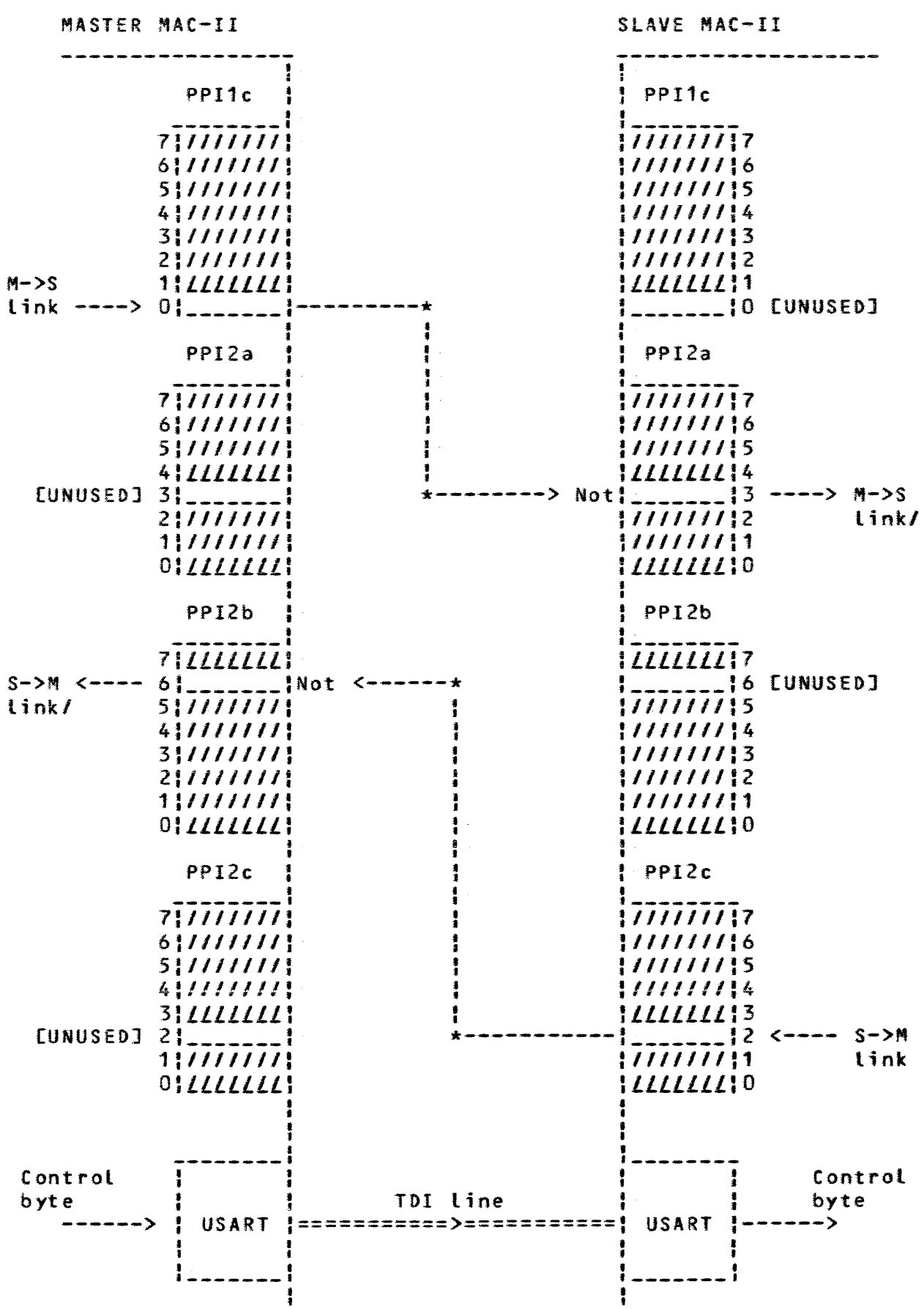
Request :	Value :
Stop / reset request	@30
Clear request	@31
Clear cache request	@32
Slave on-line request	@33
Slave off-line request	@34
Single step on request	@35
Single step off request	@36
Interrupt on request	@37
Interrupt off request	@38
Step request	@39
Start request	@3A
Remote on request	@3B
Remote off request	@3C
Baud rate select request	@3D - (1200)
" " " "	@3E - (300)
" " " "	@3F - (1800)
Synchronous remote request	@40
Start self test request	@41

Prior to establish this communication, the master will check if the SLAVE-TO-MASTER link (PPI2b) was already on, if not, the USART will be initialized for TDI transmit (see flow above), the MASTER-TO-SLAVE link (PPI1c) will be raised, and the master MAC will wait for a 500 mS timeout period for the slave (which is supposed to poll constantly the master link in PPI2a) to set the SLAVE-TO-MASTER link (PPI2c). The specified character may then be transmitted through the USART using the above flows.

If any error occurs during this process, the master MAC will drop the MASTER-TO-SLAVE link in PPI1c.

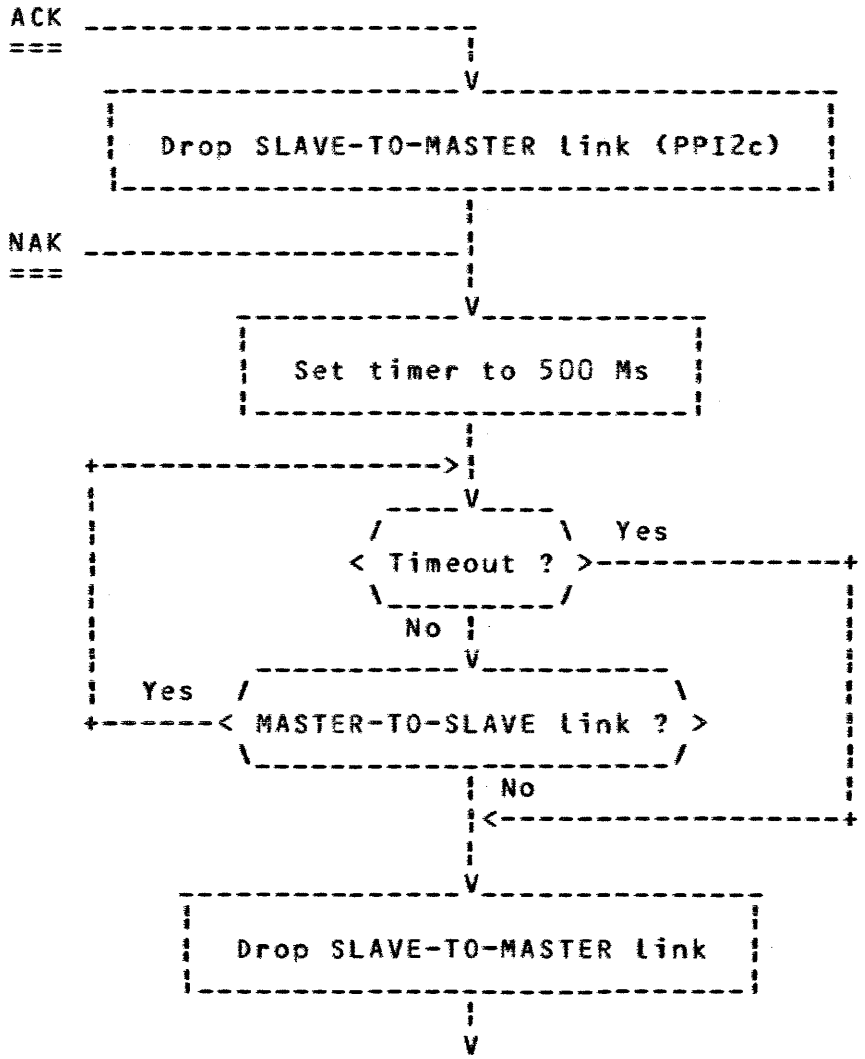
If either the command character could not be received correctly, or the command character was decoded as invalid, or any error occurred during the execution of the command character, the slave has to send a nak response to the master.

The drawing on the following page describes the communication lines between master and slave Maintenance Access Cards :


MASTER MAC-II
SLAVE MAC-II
PPI1c
PPI2a
PPI2b
PPI2c
M->S link ----> 0
[UNUSED] 3
S->M <---- 6 link/
[UNUSED] 2
Not
[UNUSED]
---> M->S link/
S->M link
<---- S->M link
Control byte ------>
TDI line
USART
USART
Control byte ------>

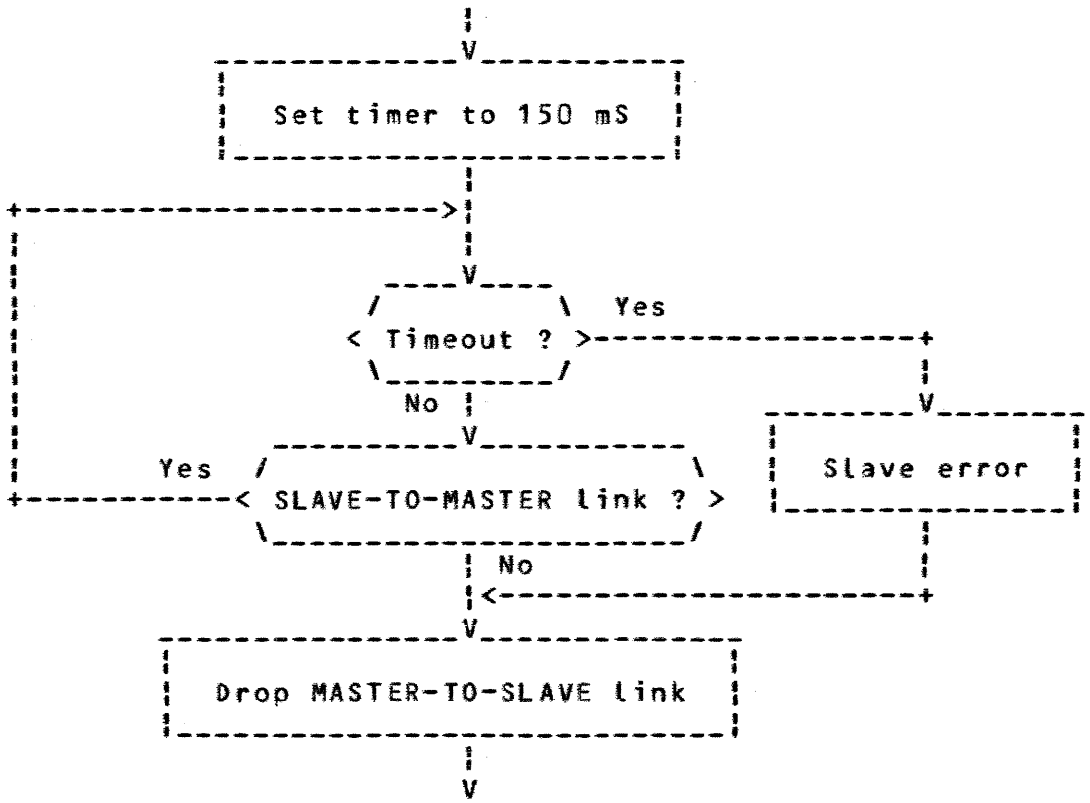
SLAVE RESPONSE TO A MASTER COMMAND :

The following flows apply to the handshake protocol between master and slave to evaluate the execution of a master command :



The ACK protocol will lower the SLAVE-TO-MASTER link immediately, while a NAK response will do it only when either a timer period of 500 mS has elapsed, or when the master has dropped his link with the slave.

EVALUATION OF THE SLAVE RESPONSE :



If the SLAVE-TO-MASTER link was still on after a time period of 150 mS, a slave error is assumed, otherwise the master command may be considered as successful, in both cases the MASTER-TO-SLAVE link needs to be lowered.

ODT MESSAGES HANDLING :

Consists of 2 main procedures : "GETTOKEN" and "SCANNER".

"GETTOKEN" is used to isolate the next keyword from the ODT input line and needs as an output the character pointer and the length of the previous token (respectively CHARP & TOKLEN).
The outputs will be the CHARP and TOKLEN of the new token.
This routine will disregard leading blanks, and convert any lower case alpha character of the new token to upper case.

GETTOKEN recognises 3 types of keywords (indicated by TOKENTYPE at the output of the procedure) i.e. :

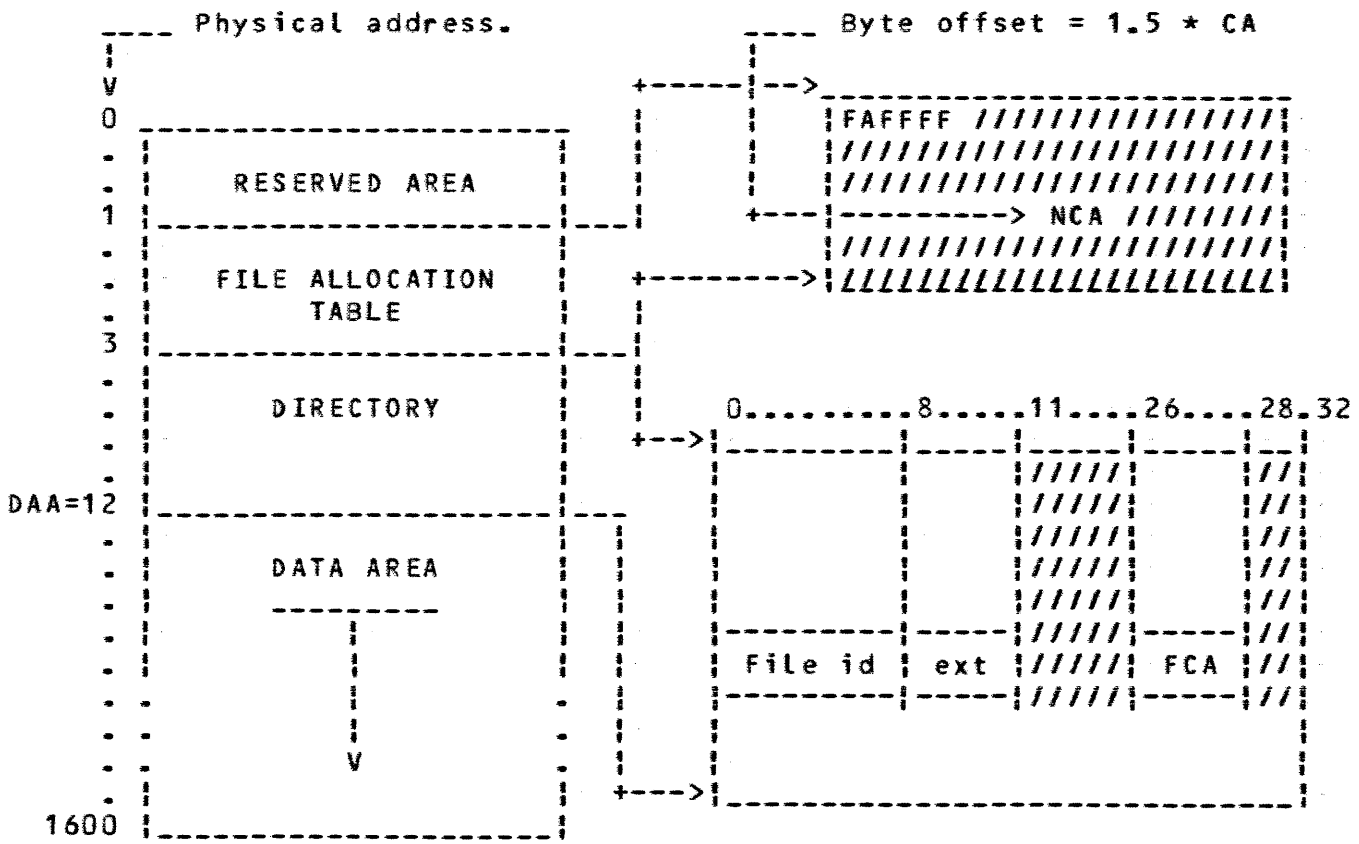
- End of line pseudo token (value 1) whenever the scan of the first character reaches the size of the ODT input line (indicated by OPMSLT).
- Alphanumeric token (value 2), character range : A->Z, 0->9
- Special token (value 3), TOKLEN always equal to 1.

"SCANNER" compares the symbol isolated by the GETTOKEN procedure with lists of allowed keywords and gives an associated symbol type and a symbol parameter as output.

The symbol type may be interpreted as follows :
- Value 0 : invalid symbol.
- Values 1 - 51 : primary command word.
- Values 101 - 105 : modifier symbol.

The "INTERACT" routine called by the halt mode main loop will perform a case statement on the symbol type in order to branch to the appropriate piece of code which will make use of the symbol parameter to execute the selected ODT function.

APPENDIX : TP400 DISK STRUCTURE.



UA = unit of allocation (8).
DAA = data area address (12).
FCA = first cluster address.
CA = cluster address.
NCA = next cluster address.

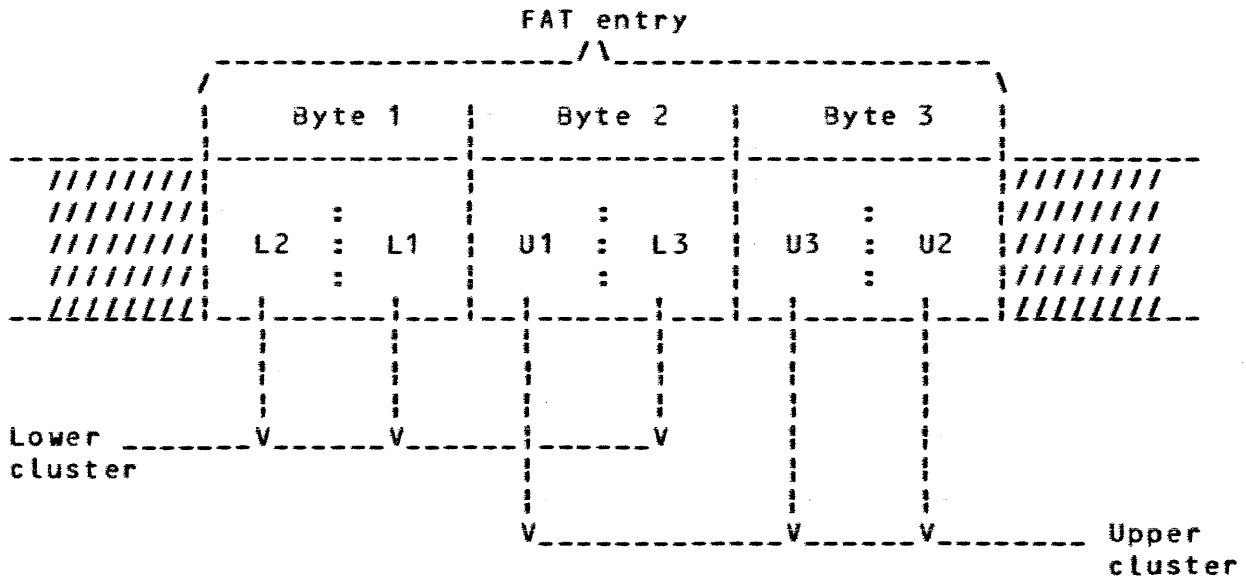
CA = @000 : Cluster available for file allocation.
CA = @FF7 : Hardware error within cluster.
CA = @FF8 - FFF : End of file cluster.
Else : Physical sector address := (CA - 2) * UA + DAA

(continued next page)

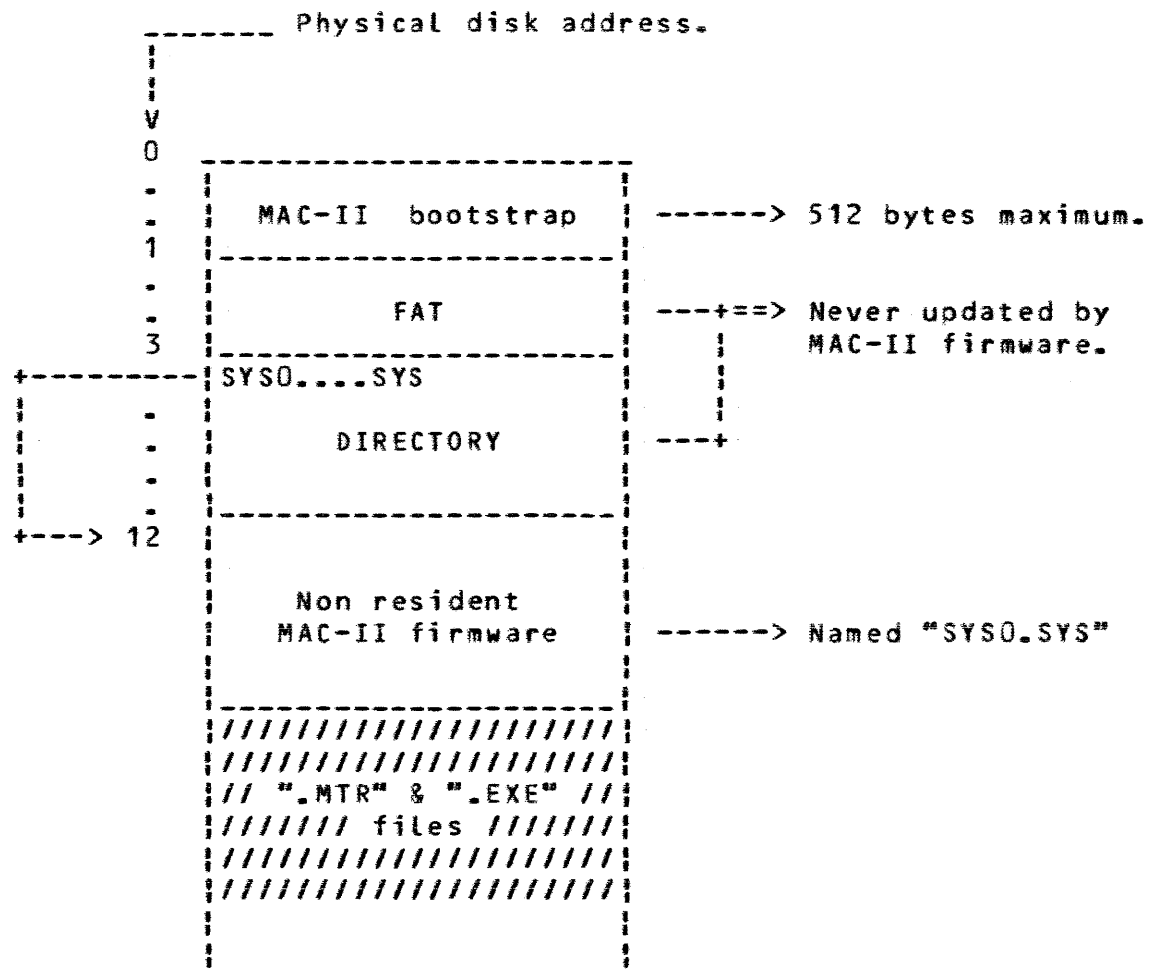
TP400 FAT FORMAT :

The addresses of the clusters pertaining to any particular file are linked in the disk FAT (file allocation table).
Each 24 bits FAT entry contains 2 x 12 bits cluster addresses.
Let L1-L2-L3 & U1-U2-U3 be 4 bit digits, and let the two 16 bit cluster addresses of a 24 bit FAT entry be @0L1L2L3 (lower) & @0U1U2U3 (upper).

Those addresses will be represented in the 24 bits FAT entry according to the following format :



APPENDIX : CONTENTS OF MAC-II DISK.



GENERATION OF MAC-II BOOTSTRAP DISKS :

The creation of MAC-II bootstrap code on the first sector of a TP400 disk can be done by means of the "BOOTSTRP" program on the ET2000.

The purpose of this program is to copy the first 512 bytes of a specified standard ET2000 file to the physical sector number zero of another specified disk.

(continued next page)

The program will first ask the input parameters as follows :

1) ENTER INPUT UNIT :

The answer to this question is a 1 character input which specifies the unit where the input file is located; the input format will be A,B,C,D or 0,1,2,3.

2) ENTER FILE IDENTIFIER :

The answer is an ET2000 file name (without extension) ended either when the 8th character is entered, or when the return key (@0D) is depressed. The blank characters will be disregarded, and the rub-out key may be used to correct typing errors.

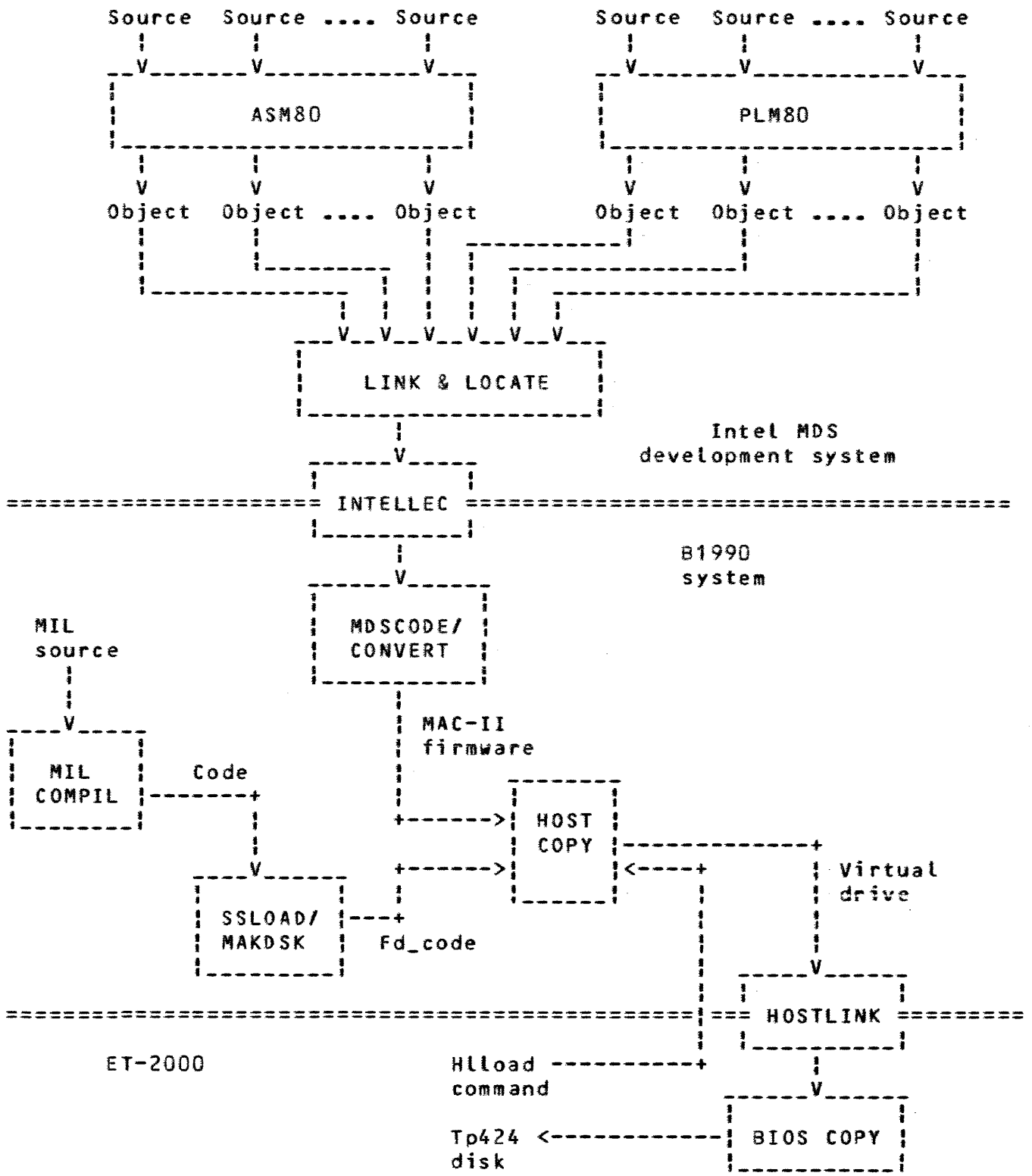
The program will then search for the specified file with a file identifier extension of ".BOT", if the file cannot be located, an appropriate message will be sent, and the whole process will be restarted, if the file is present, its first sector will be read and displayed on the ODT screen line by line (skip to next line by depressing any key).

The operator will then be asked to specify the output disk drive where the bootstrap code will be written, using the following message : "ENTER OUTPUT UNIT : ".
The format of the answer to this question will be similar to the one used to specify the input drive.

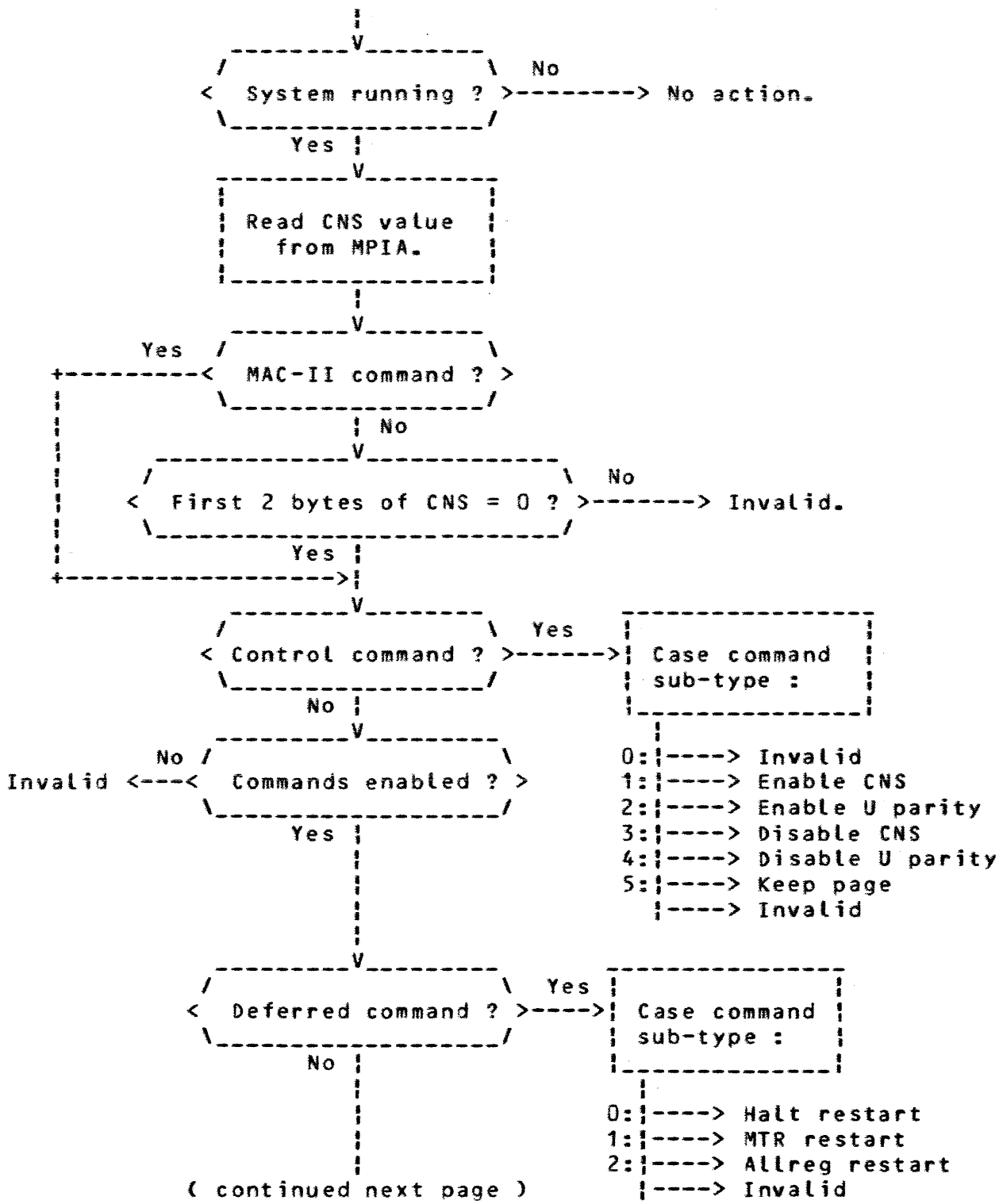
After a successful output to the requested drive, the program will be restarted from the beginning.

The whole process can be resumed/aborted by means of the CTRL-C mechanism.

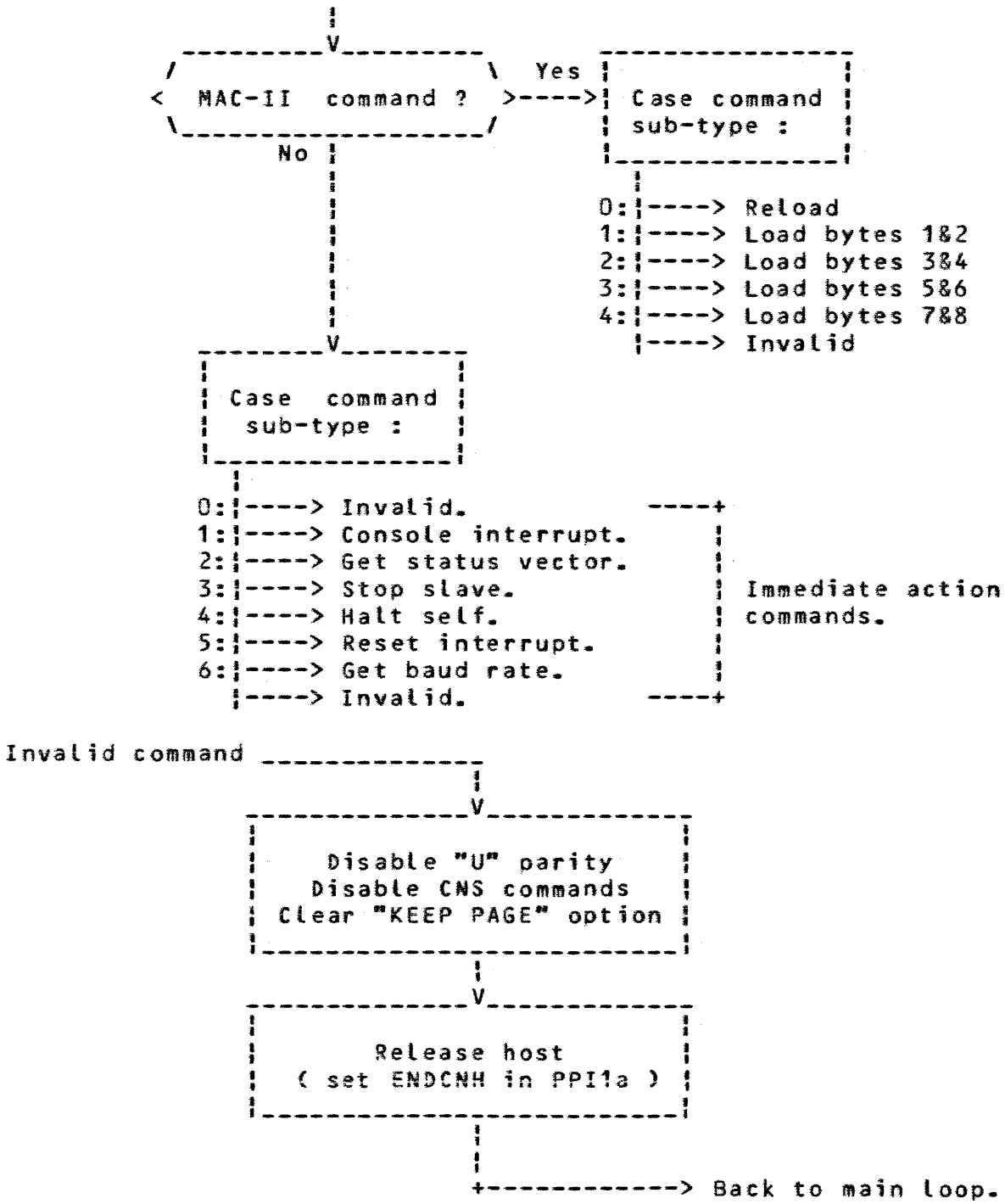
DISKETTE GENERATION PROCEDURE



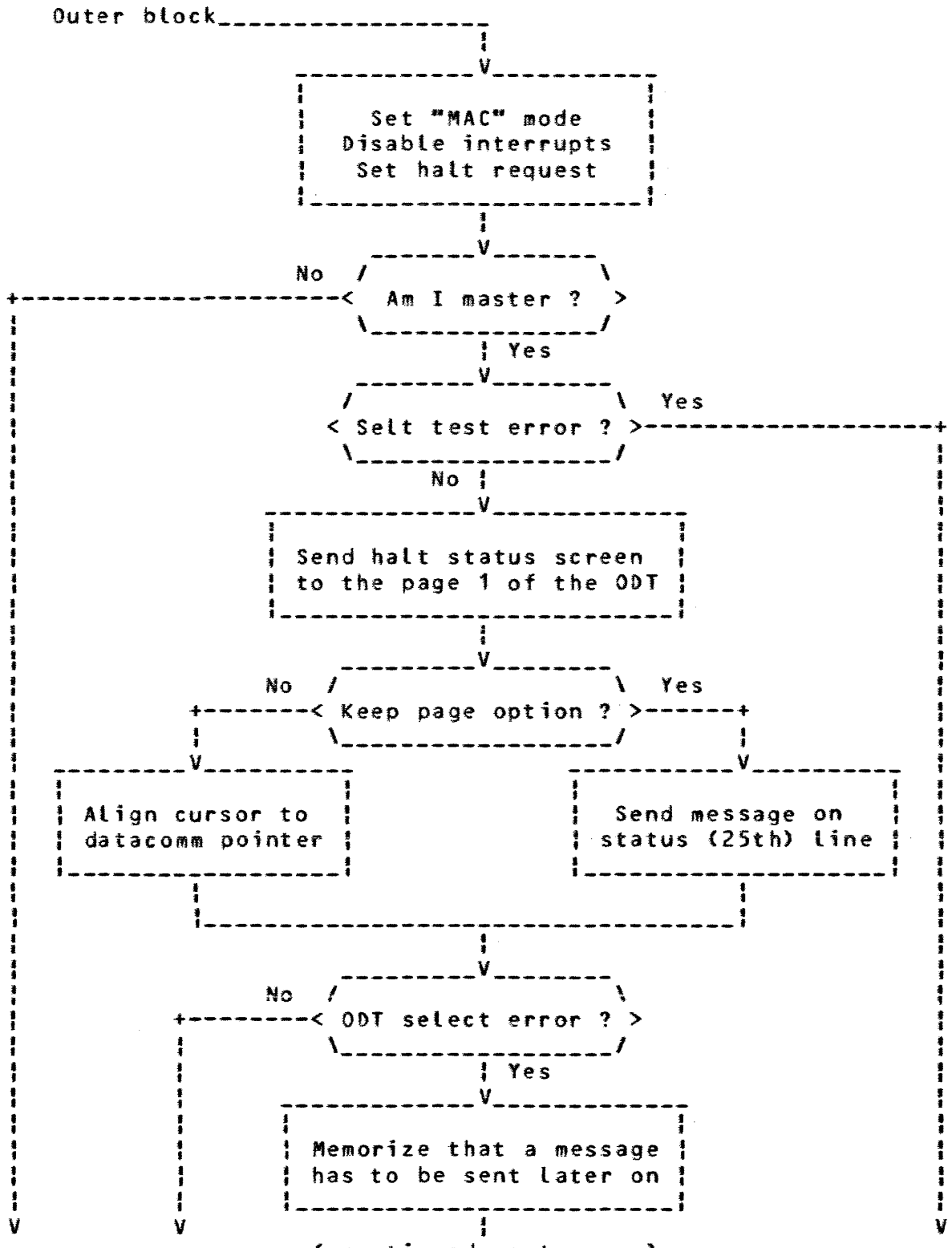
APPENDIX : CNS COMMANDS DECODE FLOW.

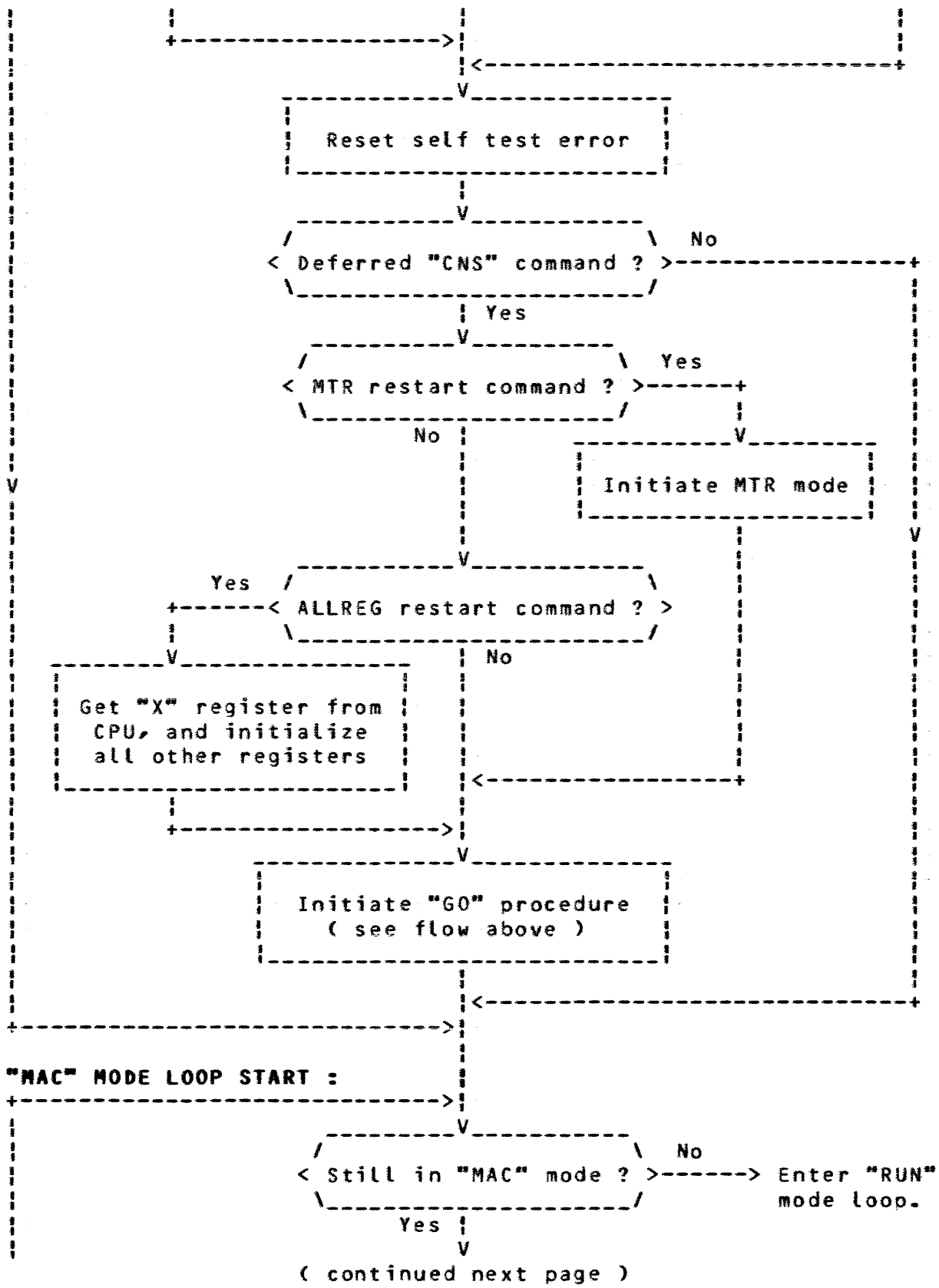


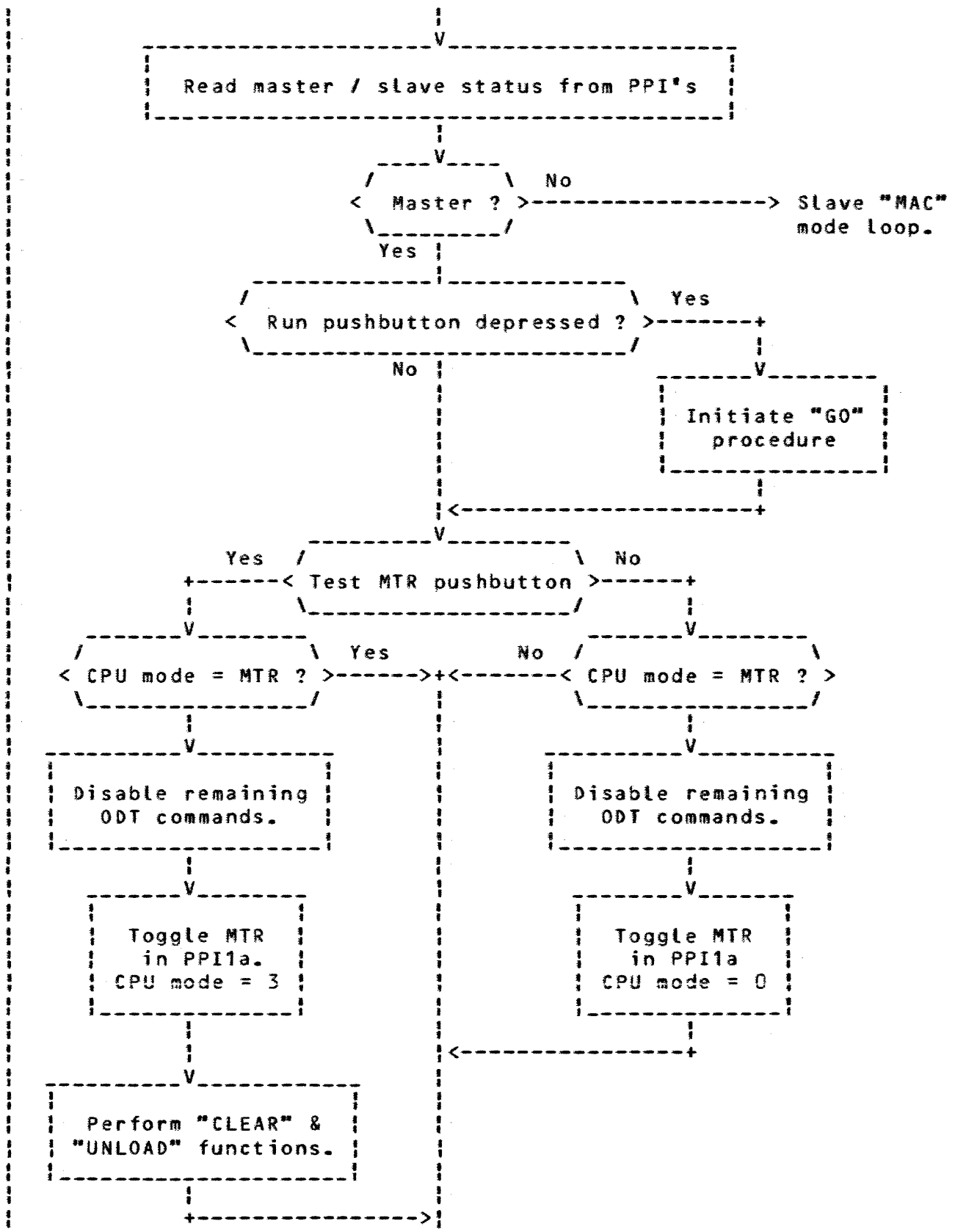
CNS COMMANDS DECODE FLOW (continued) :



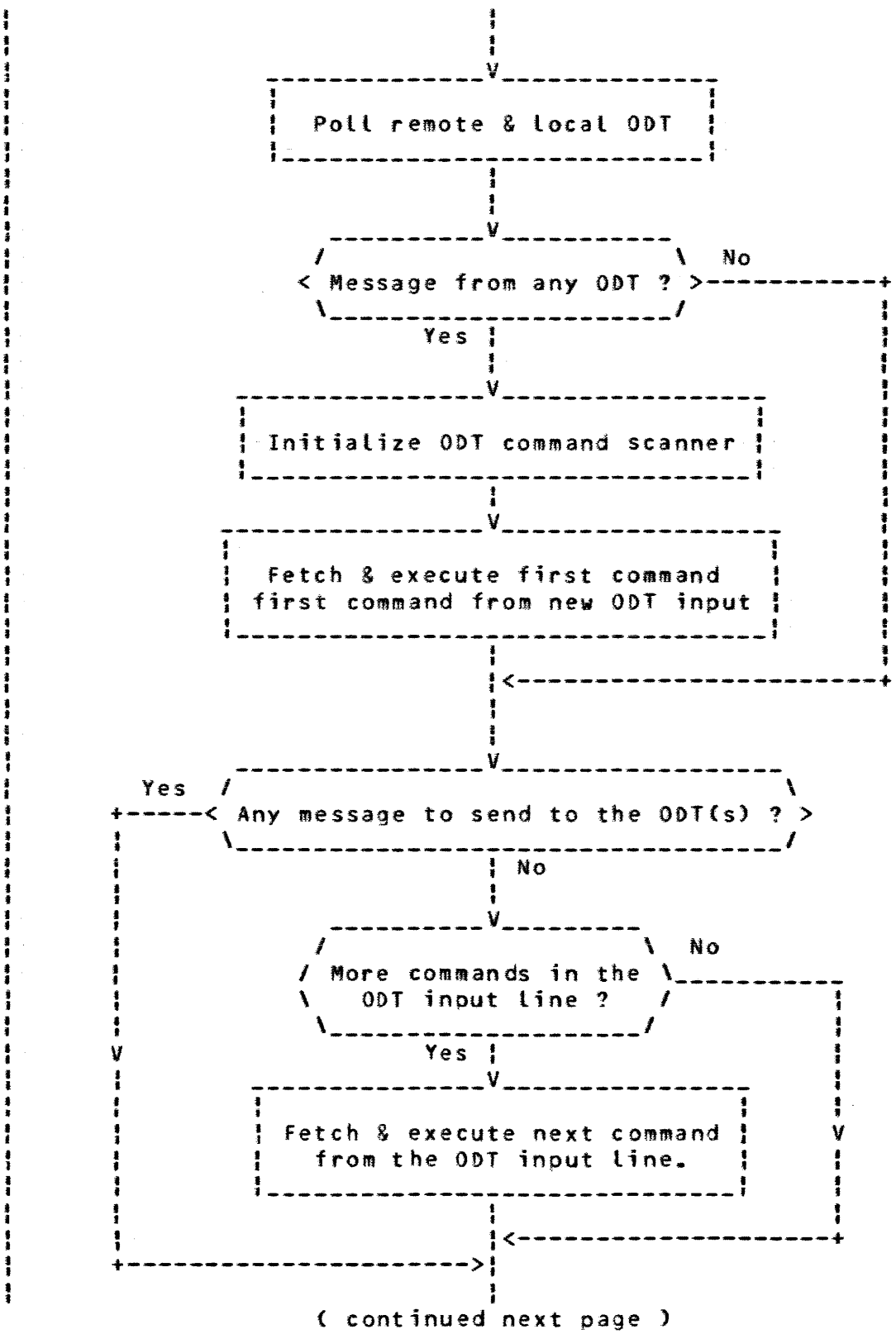
APPENDIX : HALT MODE MAIN LOOP FLOW.



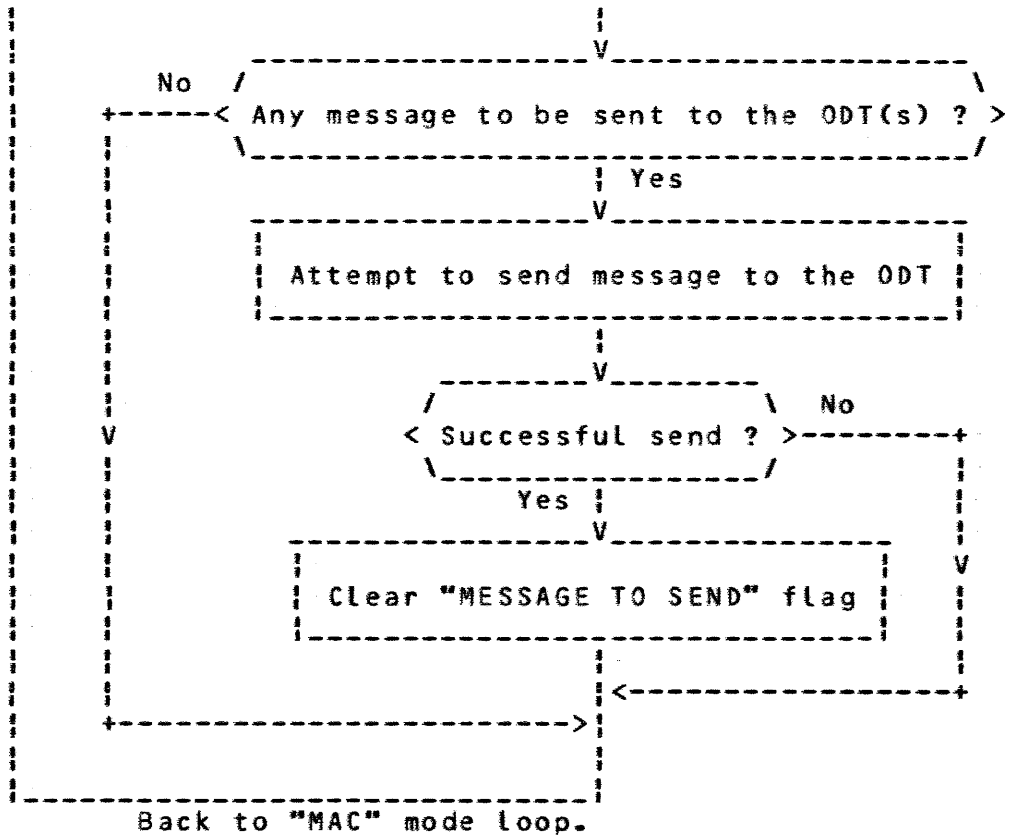

Reset self test error
Deferred "CNS" command ?
No
Yes
MTR restart command ?
Yes
No
Initiate MTR mode
ALLREG restart command ?
Yes
No
Get "X" register from CPU, and initialize all other registers
Initiate "GO" procedure (see flow above)
"MAC" MODE LOOP START :
Still in "MAC" mode ?
No
Enter "RUN" mode loop.
Yes
(continued next page)

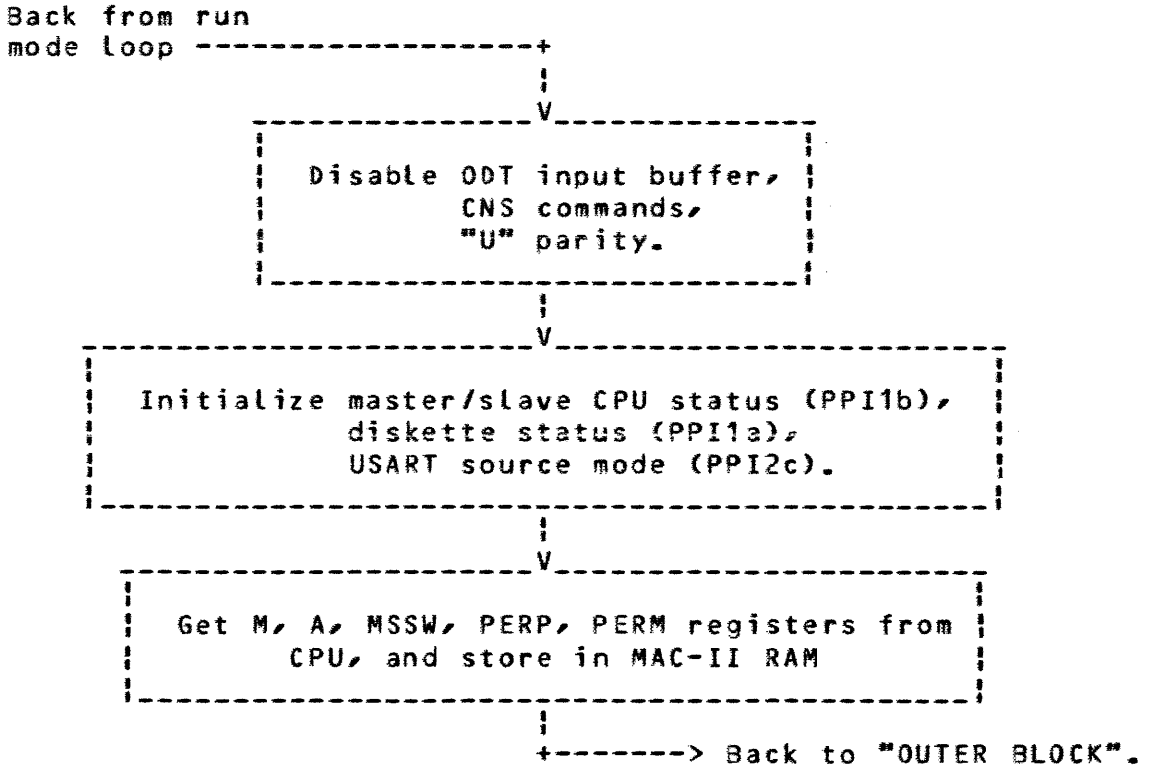


(continued next page)

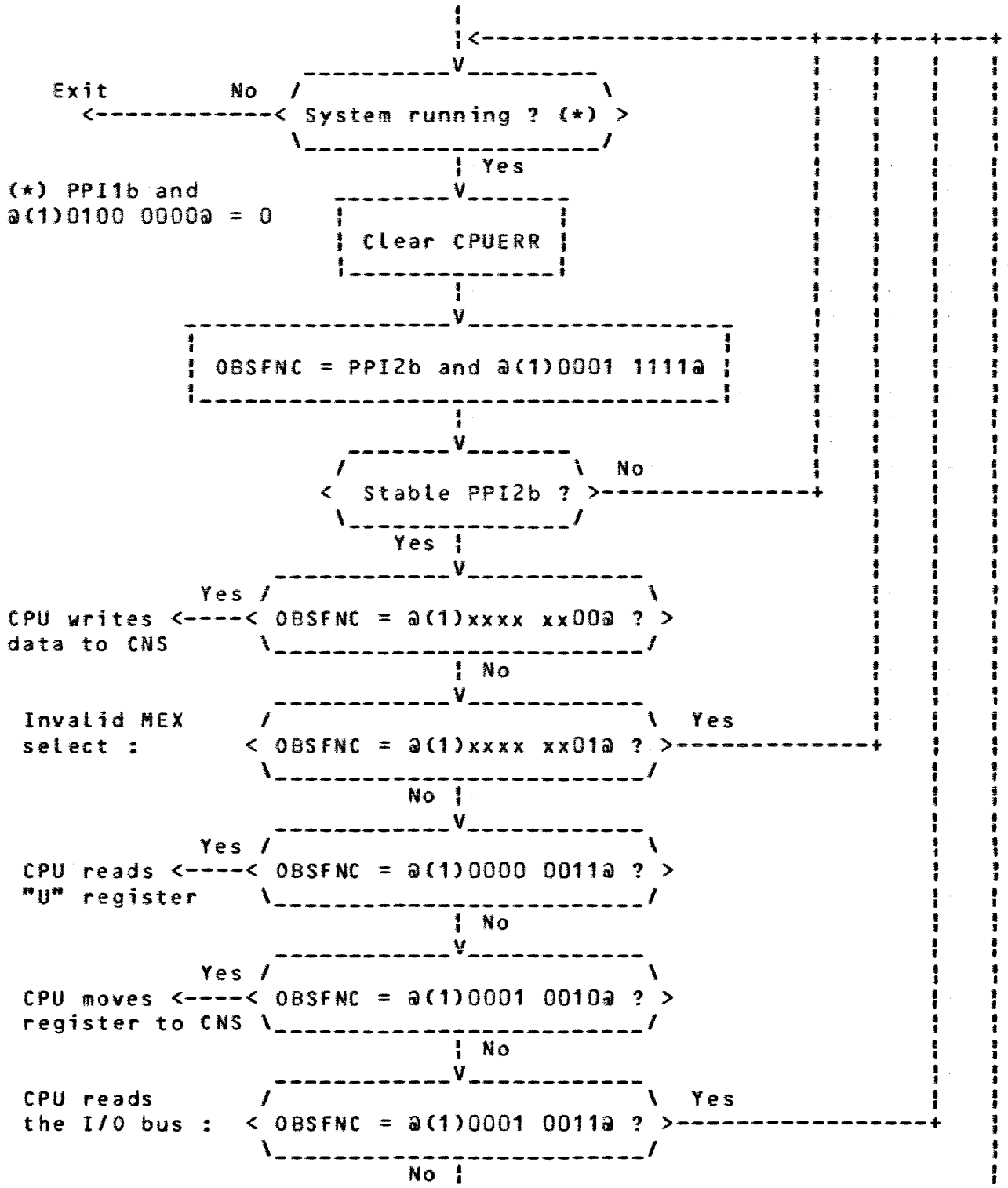


(continued next page)


No
Any message to be sent to the ODT(s) ?
Yes
Attempt to send message to the ODT
Successful send ?
No
Yes
Clear "MESSAGE TO SEND" flag
Back to "MAC" mode loop.


Back from run mode loop
Disable ODT input buffer, CNS commands, "U" parity.
Initialize master/slave CPU status (PPI1b), diskette status (PPI1a), USART source mode (PPI2c).
Get M, A, MSSW, PERP, PERM registers from CPU, and store in MAC-II RAM
Back to "OUTER BLOCK".

APPENDIX : RUN MODE MAIN LOOP FLOW.

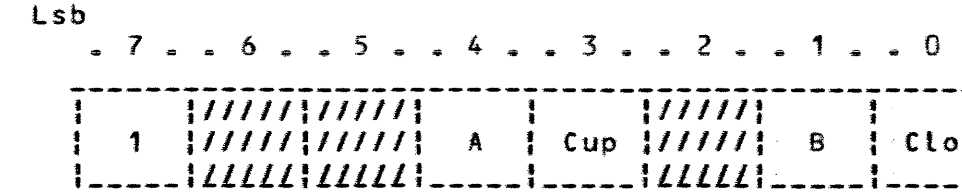


(continued next page)

RUN MODE MAIN LOOP FLOW (continued) :

MAC-II PPI'S PORT ASSIGNMENT.

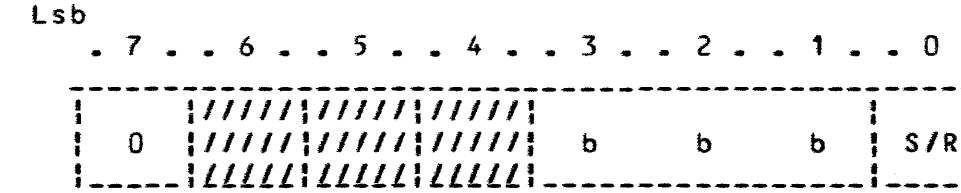
In mode 0, the ports of the 8255 PPI's are assigned by the following commands to the corresponding PPICNTL's :



A = mode of port A (0 = output , 1 = input)
B = mode of port B (" , ")
Cup = mode of port C upper part (0 = output , 1 = input)
Clo = mode of port C lower part (" , ")

PPI name :	I/O address :	Port A	Port B	Port C	Value :
PPI1	@90 - 93	Out	In	Out	@82
PPI2	@94 - 97	In	In	Out	@92
IPPI	@88 - 8B	In	In	In	@9B
HPPI	@84 - 87	Out	Out	Out	@80
MPPI	@80 - 83	In	In	In	@9B

Selected port C bits may be set/reset by the following commands issued to the corresponding PPICNTL :



bbb = bit number in PPI port C (0 to 7).
S/R = set/reset bit (1 = set, 0 = reset).

MAC-II PPI'S BIT ASSIGNMENT.

	PPI1a	PPI1b	PPI1c	PPI2a	PPI2b	PPI2c
I/O address	@90	@91	@92	@94	@95	@96
Mode	Out	In	Out	In	In	Out
Bit# 7 (MSB)	Strobe	Slave off/	Mode sel.3	Event timer	MUX'ed test result	Intrpt
Bit# 6	Remote halt/run toggle	Run/	Mode sel.2	Rate timer	Slave-Master link/	Enable clock 2
Bit# 5	Echo	A Cpu/	Mode sel.1	Real timer	\\\\\\\\ \\\\\\\\ \\\\\\\\	32 uS clock source
Bit# 4	Endcnh	Halt request	Frozen M	\\\\\\\\ \\\\\\\\ \\\\\\\\	MEX sel.2	Front plane LED
Bit# 3	Disk parity error	Other CPU run/	System clear	Master -Slave link/	MEX sel.1	Hltreq over- -ride
Bit# 2	Local halt/run toggle	MTR/	DTR/	\\\\\\\\ \\\\\\\\ \\\\\\\\	MEX sel.0	Slave_ Master link
Bit# 1	Toggle sel.2	A Cpu master	Slave off	\\\\\\\\ \\\\\\\\ \\\\\\\\	H->Mx	Enable clock 1
Bit# 0 (LSB)	Toggle sel.1	B Cpu in/	Master -Slave link	Clock : 0->3 MHz 1->4 MHz	H<-Mx/	Enable clock 0

MAC-II PPI'S BIT ASSIGNMENT (CONTINUED) :

Significance of toggle select :

Toggle sel.1	Toggle sel.2	Strobe	Meaning :
x	x	0	No action
0	0	1	No action
0	1	1	Toggle master FF
1	0	1	Toggle MTR FF
1	1	1	Force memory bad parity

Significance of mode select :
Select the source of data & clocks into the USART.

Mode sel.3	Mode sel.2	Mode sel.1	Meaning :
0	0	0	TDI
0	0	1	Clear memory parity error
0	1	0	Rs232 asynchronous
0	1	1	Rs232 synchronous
1	0	0	Loop test, card in system
1	0	1	Loop test, card not in system
1	1	0	Send 0 (MAC test summary)
1	1	1	Send 1 (MAC test summary)

Significance of MEX select function :

MXs2	MXs1	MXs0	H->MX	H<-MX/	Meaning :
x	x	x	0	0	Write to CNS
x	x	x	0	1	No action
0	0	0	1	1	Read U register
1	0	0	1	0	Register to CNS
1	0	0	1	1	Read I/O bus
0	0	1	1	1	Read CNS
1	0	1	1	1	Diskette start
1	1	0	1	1	Diskette stop
1	1	1	1	1	Unload diskette
0	1	1	1	1	Read M register
0	1	0	1	1	Acknowledge

MAC-II PPI'S BIT ASSIGNMENT (CONTINUED) :

MULTIPLEXED TEST RESULTS.

a) Clear-to-send test.

	Msb 7	6	5	4	3	2	1	Lsb 0
PPI1a : (output)	0	0	0	0	0	0	1	1
PPI2b : (input)	CTS/							

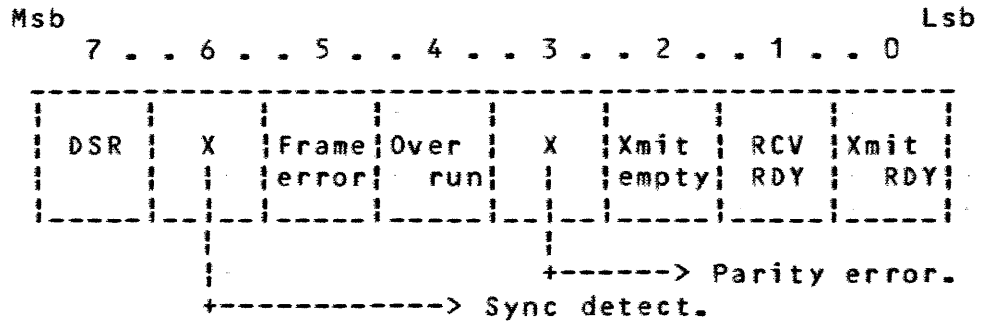
b) Data-carrier-detect test.

	Msb 7	6	5	4	3	2	1	Lsb 0
PPI1a : (output)	0	0	0	0	0	1	1	1
PPI2b : (input)	DCD/							

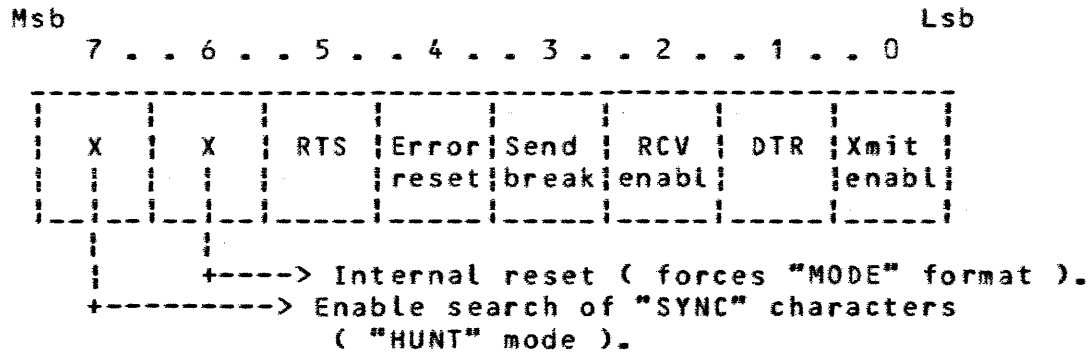
APPENDIX : 8251 USART FORMAT.

I/O port addresses = @B0 - data port (HSCID)
= @B1 - command / status port (HSCIC)

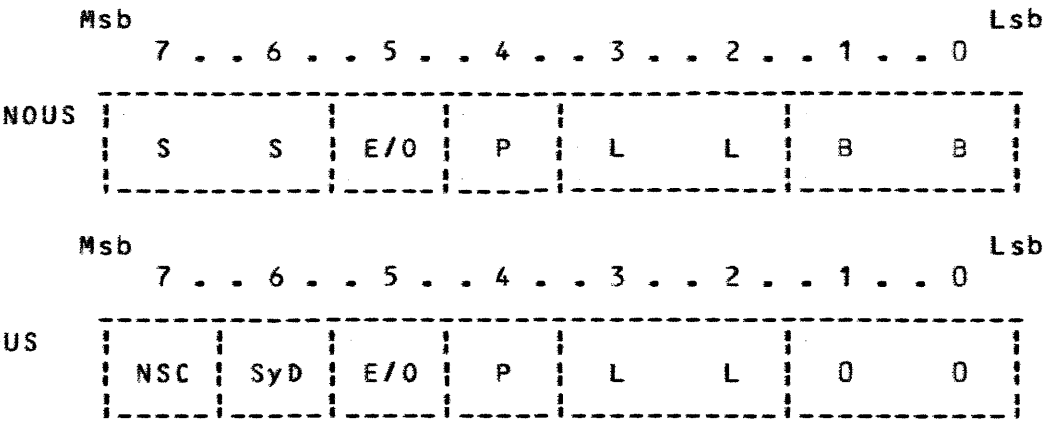
STATUS WORD FORMAT :



COMMAND WORD FORMAT :



MODE WORD FORMAT :



(continued next page)

8251 USART FORMAT (continued) :

Significance of mode format flags :

- SS = Number of start/stop bits,
  SS = 00 : invalid,
  SS = 01 : 1 bit,
  SS = 10 : 1 1/2 bit,
  SS = 11 : 2 bits.

- E/O = Even / odd parity (1 = even, 0 = odd).

- P = Enable parity.

- LL = Character length,
  LL = 00 : 5 bits,
  LL = 01 : 6 bits,
  LL = 10 : 7 bits,
  LL = 11 : 8 bits.

- BB = Baud rate factor,
  BB = 00 : synchronous mode only,
  BB = 01 : 1 x XMT / RCV clock,
  BB = 10 : 16 x XMT / RCV clock,
  BB = 11 : 64 x XMT / RCV clock.

- NSC = Number of sync. characters,
  NSC = 1 : single "SYNC" character,
  NSC = 0 : double "SYNC" character.

- SyD = Sync. detect (0 = output, 1 = input).

APPENDIX : WD1770 DISK CONTROLLER FORMAT.

I/O port address	Input	Output
@C0	Status register	Command register
@C1	Track register	
@C2	Sector register	
@C3	Data register	

COMMAND REGISTER FORMAT :

+---- Type I (head motion) commands.

	Msb 7	6	5	4	3	2	1	Lsb 0
RESTORE :	0	0	0	0	M	V	Step rate	
SEEK :	0	0	0	1	M	V	Step rate	
STEP :	0	0	1	U	M	V	Step rate	
STEP IN :	0	1	0	U	M	V	Step rate	
STEP OUT :	0	1	1	U	M	V	Step rate	

U : update LSI's internal track register (U=0, no update)

M : disable "MOTOR ON" flag (M=0, enabled).

V : verify on destination track (V=0, no verify).

Step rate : 00 = 6 milli sec.
01 = 12 milli sec.
10 = 20 milli sec.
11 = 30 milli sec.

WD1770 FORMAT (continued) :

+---- Type II (input / output) commands.

	Msb 7	6	5	4	3	2	1	Lsb 0
READ :	1	0	0	S/M	M	D	0	0
WRITE :	1	0	1	S/M	M	D	P	W/D

S/M : single/multiple sector (0 = single).

M : same as type I commands.

D : add 30 milli sec. delay (0 = no delay).

P : disable write precomp flag (0 = enable).

W/D : write/delete flag (1 = delete)

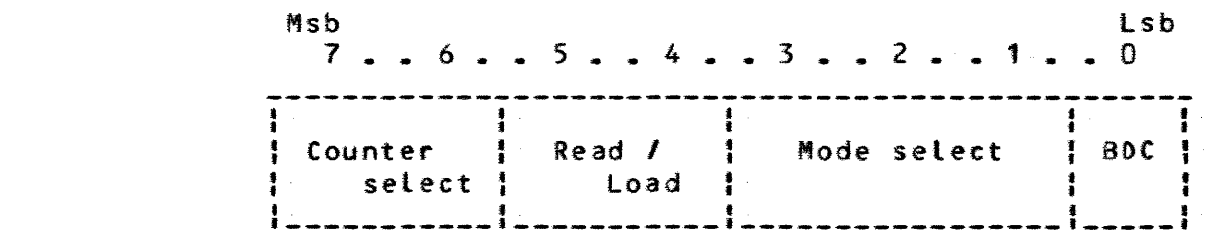
STATUS REGISTER FORMAT :

Bit#	Meaning :	Type I	Type II
Msb 7	Motor on	X	X
6	Write protect.	-	X
5	Spin up.	X	-
	Record type.	-	X
4	Record not found	-	X
3	CRC error.	X	X
2	Lost data.	-	X
	Track 00 pin	X	-
1	Index pin.	X	-
	Data request	-	X
Lsb 0	Busy	X	X

APPENDIX : 8253 PROGRAMMABLE TIMER.

I/O port addresses : PTMR0 = @A0
PTMR1 = @A1
PTMR2 = @A2
PTMRC = @A3 (control byte)

Control byte format :



Counter select : 00 = PTMR0,
01 = PTMR1,
10 = PTMR2,
11 = invalid.

Read / load parameters : 00 = counter latch operation,
01 = LSB only,
10 = MSB only,
11 = LSB first, then MSB.

Mode select : 000 = interrupt on terminal count,
001 = programmable one-shot,
010 = rate generator,
011 = square wave rate generator,
100 = soft triggerable strobe,
101 = hard triggerable strobe.

BCD = 0 : binary counter,
1 : binary coded decimal counter.

On the MAC-II card, PTMR's will be assigned as follows :

PTMR0 = real time clock,
PTMR1 = ODT baud rate generator,
PTMR2 = event timer (see PPI2a, bit# 7).

For safety reasons, whenever those PTMR's are programmed (mode programming or loading time values) the corresponding clock enable signal in PPI2c should be lowered, and raised afterwards.

The time value loaded in PTMR0 to generate a 0.1 real time clock signal for the B1990 processor is @30D4.

For PTMR1, the clock dividers will be computed according to the following algorithm :

$$\text{CLOCK DIVIDER} = \frac{4{,}000{,}000 \text{ (or } 3{,}000{,}000 \text{ - see PPI2a bit\# 0)}}{2 * 16 * \text{BAUD RATE}}$$

USART baud rate factor <--------**

the results of which can be summarized by the following table :

		3 MHz	4 MHz
RS232	300	@0138	@01A0
	1200	@004E	@0068
	1800	@0034	@0045
TDI	9600	@000A	@000D

The count values loaded in PTMR2 are representing the ammount of 2 millisecond time intervals that will elap between clock 2 enable setting (bit# 6 of PPI2c) and the setting of the event timer bit (bit# 7 of PPI2a).

MEX SELECT SUMMARY.

The following table may be useful whenever the MEX select lines (5 lsb's of PPI2b) are monitored to determine what functions are requested to the MAC.

MEX select 2.	MEX select 1.	MEX select 0.	H-to-MEX	MEX-to-H/				
///	///	///	0	0	. .	@00	. . .	Move data to CNS
///	///	///	0	1	. .	@01	. . .	Invalid
0	0	0	1	0	. .	@02	. . .	Nop
0	0	0	1	1	. .	@03	. . .	Move from "U"
///	///	///	0	0	. .	@04	. . .	Move data to CNS
///	///	///	0	1	. .	@05	. . .	Invalid
0	0	1	1	0	. .	@06	. . .	Nop
0	0	1	1	1	. .	@07	. . .	Read CNS
///	///	///	0	0	. .	@08	. . .	Move data to CNS
///	///	///	0	1	. .	@09	. . .	Invalid
0	1	0	1	0	. .	@0A	. . .	Nop
0	1	0	1	1	. .	@0B	. . .	Start acknowledge
///	///	///	0	0	. .	@0C	. . .	Move data to CNS
///	///	///	0	1	. .	@0D	. . .	Invalid
0	1	1	1	0	. .	@0E	. . .	Nop
0	1	1	1	1	. .	@0F	. . .	Nop
///	///	///	0	0	. .	@10	. . .	Move data to CNS
///	///	///	0	1	. .	@11	. . .	Invalid
1	0	0	1	0	. .	@12	. . .	Register to CNS
1	0	0	1	1	. .	@13	. . .	Nop
///	///	///	0	0	. .	@14	. . .	Move data to CNS
///	///	///	0	1	. .	@15	. . .	Invalid
1	0	1	1	0	. .	@16	. . .	Nop
1	0	1	1	1	. .	@17	. . .	Diskette start
///	///	///	0	0	. .	@18	. . .	Move data to CNS
///	///	///	0	1	. .	@19	. . .	Invalid
1	1	0	1	0	. .	@1A	. . .	Nop
1	1	0	1	1	. .	@1B	. . .	Diskette stop
///	///	///	0	0	. .	@1C	. . .	Move data to CNS
///	///	///	0	1	. .	@1D	. . .	Invalid
1	1	1	1	0	. .	@1E	. . .	Nop
1	1	1	1	1	. .	@1F	. . .	Unload diskette

No-op functions will cause the firmware to strobe the "ENDCNH" signal to the host CPU , while invalid commands will not.

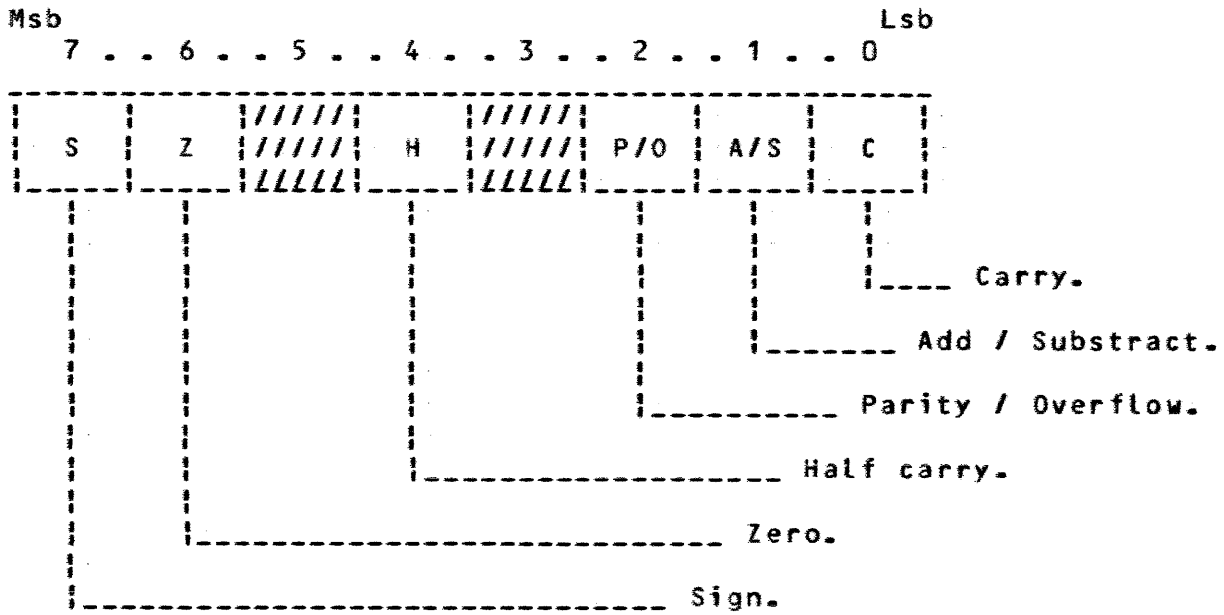
NOTES

NOTES

NOTES

NOTES

INTEL 8080 FLAGS.



Parity : 0 = odd,
1 = even.

Sign : 0 = positive,
1 = negative.