ADDENDUM 3

VORTEX II Reference Manual

UP 8677

98A 9952 246

This addendum contains information relating to the G.O. release of VORTEX II.

PAGE ACTION

2-7 Add to end of first paragraph of section 2.1.10:

This macro is used to return from a reentrant routine called via ALOC.

2-10 Under logical addr section:

Change priority tasks to priority zero tasks.

2-10 In section 2.1.17 DEALPG add the note:

NOTE: This request should not be used in background tasks as it may leave "holes" in memory which can cause problems when checkpointing a background task.

2-12 Add the new section 2.1.20 RECOV (Error Recovery) Macro:

This macro allows the requestor to pass the address of a recovery routine to VORTEX. Control will be passed to the routine if VORTEX attempts to terminate the task abnormally. The recovery routine is executed after any VORTEX error recovery or reporting routine. Return from the user's recovery routine should be made via the EXIT macro. The macro has the general form:

```
LABEL     RECOV     ADDR
```

where:

ADDR is the address of the error recovery routine

The recovery address is kept in TBENTY or the user's TIDB. Repeated calls to RECOV are allowed but the last specified recovery address is always the address used. Note that if an a bend occurs and control is passed back to the user, registers are not preserved.

PAGE ACTION

3-10 Add after paragraph D:

The file extension number currently active is contained in word 2, bits 15-12. This field is updated each time a new extention is created or opened.

3-12 Insert Warning after Magnetic-Tape devices paragraph:

WARNING

V$IOC returns to the issuing program prior to rewind completion. When rewinding is complete, the issuing RQBLK is updated. Therefore, the issuing RQBLK must not be modified prior to rewind completion.

4-7 In section 4.2.19, change X option to N option.
Add to N option description:

If absent, an alphabetical SORT is printed.

4-9 Under 4.2.26 /CFILE, change the first paragraph to read:

This directive, which applies only to RMD's and MT's assigned to global logical units, causes the file currently attached to the global FCB file on a logical unit to closed with update.

4-9 In the form example for /CFILE change to read:

/CFILE,lun

where

lun is the name or number of the affected logical unit. The logical unit must be one of the global logical units.

Example: Close the file currently attached to the PO global FCB.

/CFILE,PO

4-9 Under 4.2.30 /RPG (RPG II Compiler) Directive change:

Parameter O to D.
Delete parameters M and N.

PAGE ACTION

6-2 Under Print Position description change:

the 'X' option to the 'N' option.

6-3 After last paragraph of 6.1 add:

V$PED cannot be used in an overlay segment of the overlayed module.

7-8 Add new section 7.4

7.4 INTERMAP DEBUG PROGRAM (V$DBUG)

The Intermap Debug Program (V$DBUG) is a catalogued foreground library program. It requires a VORTEX II system that was generated with the 228 word nucleus module V$FSD. Interaction between V$DBUG and the program being debugged is accomplished by an encoded halt violation. Data may be examined or changed in the task being debugged, in the nucleus (Map 0) area or in a VNO task. When a trap is set two words of memory in the task being debugged are replaced by an encoded halt (0525), V$DBUG is suspended and the task being debugged is activated. When the trap is reached the encoded halt is executed. This suspends the task, restores the two words of memory to their original content, sets the P counter in the tasks TIDB to the execution address contained in the trap Command and reactivates V$DBUG. All registers of the task being debugged are available for display. Input to V$DBUG is described below and is entered through the DI logical unit. Each V$DBUG command has from 0 to 72 characters and is terminated by a carriage return. All numeric inputs are treated as octal if they begin with a zero, otherwise, they are treated as decimal.

The program to be debugged may be scheduled prior to scheduling V$DBUG or it may be a foreground program scheduled by V$DBUG. V$DBUG should be scheduled with a higher priority than the task being debugged. The ;TSTAT command may be used to obtain the TIDB address of an already scheduled task. V$DBUG may be used as follows:

```
;SCHED,V$DBUG,20,FL,F          (Priority higher than task
                               to be debugged if the task
                               is already scheduled.)
```

PAGE ACTION

7-8 (continued)

Teletype dialog after V$DBUG is scheduled:

DA* ENTER TASK TIDB ADDRESS (There are three valid responses. An invalid response causes a DA01 error message and repeats the message.)

1) END (This will cause V$DBUG to exit.)

2) (S,Area [F-User Map, N-Map 0, V-VNO], Task Name)

3) (TIDB Address, Area) (Links V$DBUG to an already scheduled task.)

V$DBUG will then respond with:

DA* task name, map image addr (Task Name and Map Image address from TIDB of task to be debugged.)

(One of these three entries will be output. Task to be debugged must be re-scheduled if DA** is not output at this point.

DA02	Task Aborted
DA03	Task Exited
DA**	Ready for V$DBUG

There are three valid reponses to DA** at this point; any other response causes a DA04 error message and repeats the DA** query. The responses are:

1) END (This causes V$DBUG to exit. You should abort any task scheduled by V$DBUG.)

2) TC (This allows you to display the TIDB of the task being debugged. At this point if the task was scheduled by V$DBUG it is unallocated and unloaded. TIDB display commands are used following this entry.)

PAGE	ACTION
7-8	(continued)

3) OK (Allows use of regular, not TIDB display, commands.)

A 'TC' entry is followed by a DA** and TIDB display commands are allowed. An 'OK' is followed by:

MAP KEY task map key	(Map key of task being debugged.)
DA**	(Regular debug commands may now be entered and a current copy of the task to be debugged's TIDB is available for display.)

Regular commands (following an 'OK' or 'TEND'):

The first letter describes the action and the second letter is either part of the action code or an area code. The area codes (F-user map, N-nucleus or Map 0, V-VNO) are indicated by a lower case a in the examples. Parameters for regular commands consist of data and delimiters with no delimiter between the command and the first parameter. Entries are interpreted as follows:

DATA (First position indicates type and all data must be consistent with type)

O ---- Octal Value

1 - 9 - Decimal Value

@ ---- Base Symbol

Other - CL Directory Symbol

DELIMITERS

Space - End of parameters or command code

+ ---- Add to preceding value

- ---- Subtract from preceding value

' ---- Parameter Separator

---- Cancel the command

PAGE ACTION

7-8 (continued)

COMMANDS

ALTER

A a Start Loc, Data (up to 7 items separated by commas)

SET BASE SYMBOL (Set, Clear, Display)

B a @ One character, Value (there may be up to 10 symbols for each area. @T is automatically set to TIDB address of task being debugged in the appropriate table and does not count against the 10 symbols.)

BC@ a (Clears the symbol table for indicated area. An @ instead of an area code will clear all three tables.)

BD@ a (Displays the symbol table for indicated area. An @ instead of an area code will display all three tables.)

CHANGE/DISPLAY

C a Start Loc, Data (up to 7 items separated by commas.)

DISPLAY

D a Start Loc, Number (1-8 locations may be displayed.)

EXIT V$DBUG

END

INTIALIZE

I a Start Loc, Number (1-6), Data

PAGE ACTION

7-8 (continued)

LO LOGICAL UNIT DISPLAY

LP (Alternate entries start/stop printing on LO logical unit)

MEMORY DUMP

MD (Hook only - VORTEX ALOC to accomplish this will be incorporated when available.)

TRAP (Command is not allowed following a TX command)

T a Location, Address (The Address is required and changes TBRSP in the TIDB of the task being debugged so that when it is activated again it will go to that address. This action occurs after returning from a trap, not before trap execution.)

Upon completion of the specified trap, register contents are examined by using the TIDB display command (see below) and examining TBRSA-TBRSR7.

TIDB DISPLAY

TC (Allows execution of the TIDB Display commands after obtaining a copy of the task being debugged's TIDB.)

TRANSFER AREA (Required only for VNO debugging or display)
(Allows examination of areas outside the task being debugged. This commmand applies to all following commands which have an area code as the second letter until a 'TEND' command is encountered.)

PAGE ACTION

7-8 (continued)

TXO

(Zero indicated you wish to examine map 0 (N) data.)

TXTIDB ADDR

(A TIDB indicates you wish to examine data in a VNO (V) task area.)

TIDB DISPLAY COMMANDS

A,B,X,R3-R7,P,O --
These commands display the corresponding word of the TIDB of task being debugged.

ALL --
Displays all of the above.

1 - 39 --
Displays corresponding word of the TIDB

TEND --
Gets a new copy of the TIDB and allows regular commands.

PAGE ACTION

7-8 (continued)

V$DBUG ERROR MESSAGES

DA01	Incorrect response to initial query	Enter correct response
DA02	Requested task has aborted	Reschedule task
DA03	Requested task has exited	Reschedule task
DA04	Illegal response to query or number too large for TIDB	Enter correct response
DA05	Bad data in directive	Enter correct directive
DA06	Read error on DI logical unit	Repeat previous directive
DA07	Illegal command	Enter correct command
DA08	Add/Subtract to undefined Base Symbol	Repeat after defining base symbol
DA09	Base/CL Tag has more than 6 characters	Repeat with correct Base/CL Tag
DA10	Name not found on CL directory	Repeat with correct name
DA11	No room is Base Symbol table	Delete unwanted symbols and try again
DA12	Wrong area for command type	Command cannot be used for the area indicated
DA13	Illegal P address for trap	Enter correct address and retry

PAGE ACTION

9-1 Replace first two paragraphs with following:

9.1.2 File Name Directory

The File Name Directory is two sectors in length (240 words.) There are two physical directories, comprising a single logical file name directory. The second of the two physical directories is a shadow directory that contains attribute information for each file entry.

The directory for each partition has a variable number of entries arranged in n sectors, 19 entries per sector.

A file has the same corresponding entry in each of the two physical directories, i.e., if a file is the 3rd entry in the first directory, it is also the 3rd entry in the shadow directory.

Each RMD partition contains a file-name directory. The directory for each partition begins in the first sector of the partition. Sectors containing directory information are chained by pointers in the last word of each sector. Thus, directory sectors do not have to reside contiguously. The first of the two physical diectories has the following format:

Bit	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Word 0	File name
Word 1	File name
Word 2	File name
Word 3	Current position of file
Word 4	Beginning file address
Word 5	Ending file address

The file name comprises six ASCII characters packed two characters per word, left justified, with blank fill. Word 3 which contains the current address at which the

PAGE ACTION

9-1 (continued)

file is positioned, is initially set to the ending file address, and is manipulated by I/O control macros (section 5). The extent of the file is defined by the address set in words 4 and 5 when the file is created, and remains constant.

9-2 The shadow directory has the following format:

<table>
<tr><th>Bit</th><th colspan="3">15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th></tr>
<tr><td>Word 0</td><td colspan="3">Not Used</td></tr>
<tr><td>Word 1</td><td colspan="3">Not Used</td></tr>
<tr><td>Word 2</td><td colspan="3">Not Used</td></tr>
<tr><td>Word 3</td><td>Not Used</td><td>Extension Number</td><td>File Type</td></tr>
<tr><td>Word 4</td><td colspan="3">Date File Accessed</td></tr>
<tr><td>Word 5</td><td colspan="3">Date File Created</td></tr>
</table>

Words 0, 1 and 2 are reserved for future use. Word 3 has the following variable format. The lower 9 bits contain the File Type field as described below:

Bit 0	set indicates	System Binary
Bit 1	set indicates	Load Module
Bit 2	set indicates	Data file
Bit 3	set indicates	ICS file
Bit 4	set indicates	Compressed 8 bit ASCII
Bit 5	set indicates	Compressed 7 bit ASCII

Bit 6, 7, and 8 are reserved for future use.

Bits 9 through 12 of word 3 contain the Extension number field. The Extension number specifies the extension number of the specified file.

Word 4 of the shadow directory contains the date a file was last accessed. This field is updated on an OPEN request to a file (except when made by SAL.)

Word 5 of the shadow directory contains the date the file was created. The date, in Words 4 and 5, is in the following format:

PAGE ACTION

9-2 (continued)

Bits 0-3 contain the "MONTH"
Bits 4-8 contain the "DAY"
Bits 9-15 contain the "YEAR"

9-3 After the first paragraph of 9.2.1 CREATE Directive, add:

The CREATE directive sets up the date in Word 5 of the corresponding shadow directory. It also sets up the "File Type" variable in Word 3 of the shadow directory to "Data File" (bit 2 is set.)

9-3 After the first paragraph of 9.2.2. DELETE Directive, add:

This directive also deletes any extensions under this file name.

9-4 After the first paragraph of 9.2.4 ENTER Directive, add:

This directive will also set up the creation date and extension numbers date in the corresponding shadow directory. It also sets the "file type" variable to "Data File".

9-4 Replace 9.2.5 LIST Directive:

9.2.5 LIST Directive

This directive outputs to the LO logical unit the file-name and shadow directories of the specified logical unit. The output comprises the file name, file extent, current end-of-file position, logical unit name or number, extent of unassigned space in the partition, file type, file extension number, and the dates the file was created and last accessed. All numbers are in octal except for dates and extensions. The directive has the general form:

```
LIST,lun,key
```

where

lun is the number or name of the logical unit whose contents are to be listed.

key is the protection code, if any, required to access lun

PAGE ACTION

9-4 (continued)

The output format has a two line heading:

```
FILE DIRECTORY FOR LUN XXX
FILE NAME START END CURRENT F-TYPE EX CREATED ACCESSED
```

where

XXX is the number or name of the logical unit whose contents are being listed. The header is followed by a blank line and a listing of all file name from the directory. See section 9.1.2 for a description of header items. After the last file name, if there is any unassigned space in the partition, there is an entry describing the unassigned space in the partition, where the FILE NAME column contains *UNAS*, the START column contains the next available address, and both the CURRENT and END columns contain the last address +1. All numberical values are octal sectors.

Example: List the file name directory of logical unit 114 which has no protection code.

```
LIST,114
```

9-5 Replace title of first paragraph of 9.3 VORTEX FOREGROUND FILE MAINTENANCE with the following:

9-3 VORTEX FILE MAINTENANCE DRIVER (VZFMA)

The VORTEX File Maintenance driver provides a user programmable subset of the VORTEX FMAIN services. VZFMA operates as a system driver assigned to logical unit 115. All requests to VZFMA must be made through the OM library resident interface routine, V$FILE. Direct calls to VZFMA are not allowed. This is because conflicts arise in calling sequences if VZFMA services should be augmented.

PAGE ACTION

9-6 Add after chart of VZFMA control block:

The file name (words 2-4) consists of two characters per word, left justified and blank filled.

For CREATE, word 6 must be zero, and word 5 is right justified and zero filled. Words 0 and 1 are right justified and zero filled.

5 = find

Delete "-1 busy" from the list of completion codes
Add to list of compeltion codes:
Note: completion code 5 can also indicate invlaid LUN, invalid "CREATE" secor count, or invalid protect key.

10-2 Add after the last full paragraph of 10.2.1 COPYF Directive:

When specifying random length on input, the input directive should not be a RMD device. Also, when specifying blocking/deblocking, the input and output record lengths should be multiples of each other.

10-5 Add the warning after the last full paragraph of 10.2.9 PFILE Directive:

WARNING

IOUTIL uses the "recl" parameter to create the proper file extent. This parameter should match the record length parameter of the COPY directive if the record length is greater than 120 words, otherwise data beyond the specified file may be overwritten.

14-25 Insert new paragraph at end of 14.4.2:

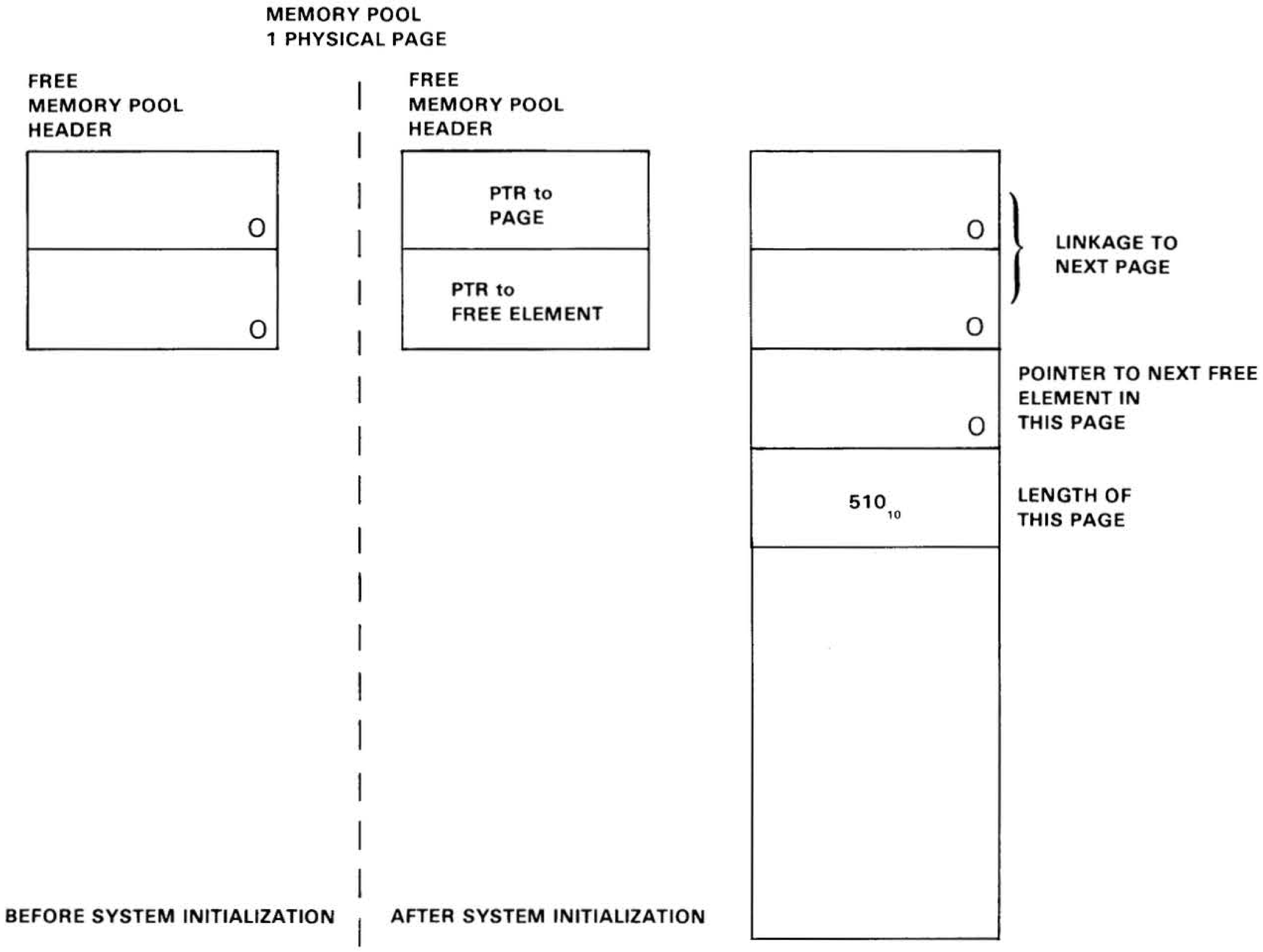
Use of RTE services: Certain RTE services (SCHED, OVLAY and DELAY 1) use TBRSTS. V$IOC requires TBRSTS of the drivers TIDB to contain the controller table address. Therefore, drivers using the RTE sevices must save and restore TBRSTS.

PAGE	ACTION
14-34	Insert new section 14.5.3:

14.5.3 ITE Intertask Communication Module

The ITE Module performs the equivalent functions of the ITC module, but has several unique conventions that are required by the 'D' revision of PRONTO.

ITE is a reentrant subroutine and resides in the nucleus. ITE consists of three areas: System Initialization, Mailbox Initialization and data transfers. ITE's memory pool initialization will occur at the first entry made by a task requesting a mailbox key. ITE will then request and link the physical page or pages to be used for ITE's internal message storage. Once physical memory has been allocated for ITE, it is ready for processing.

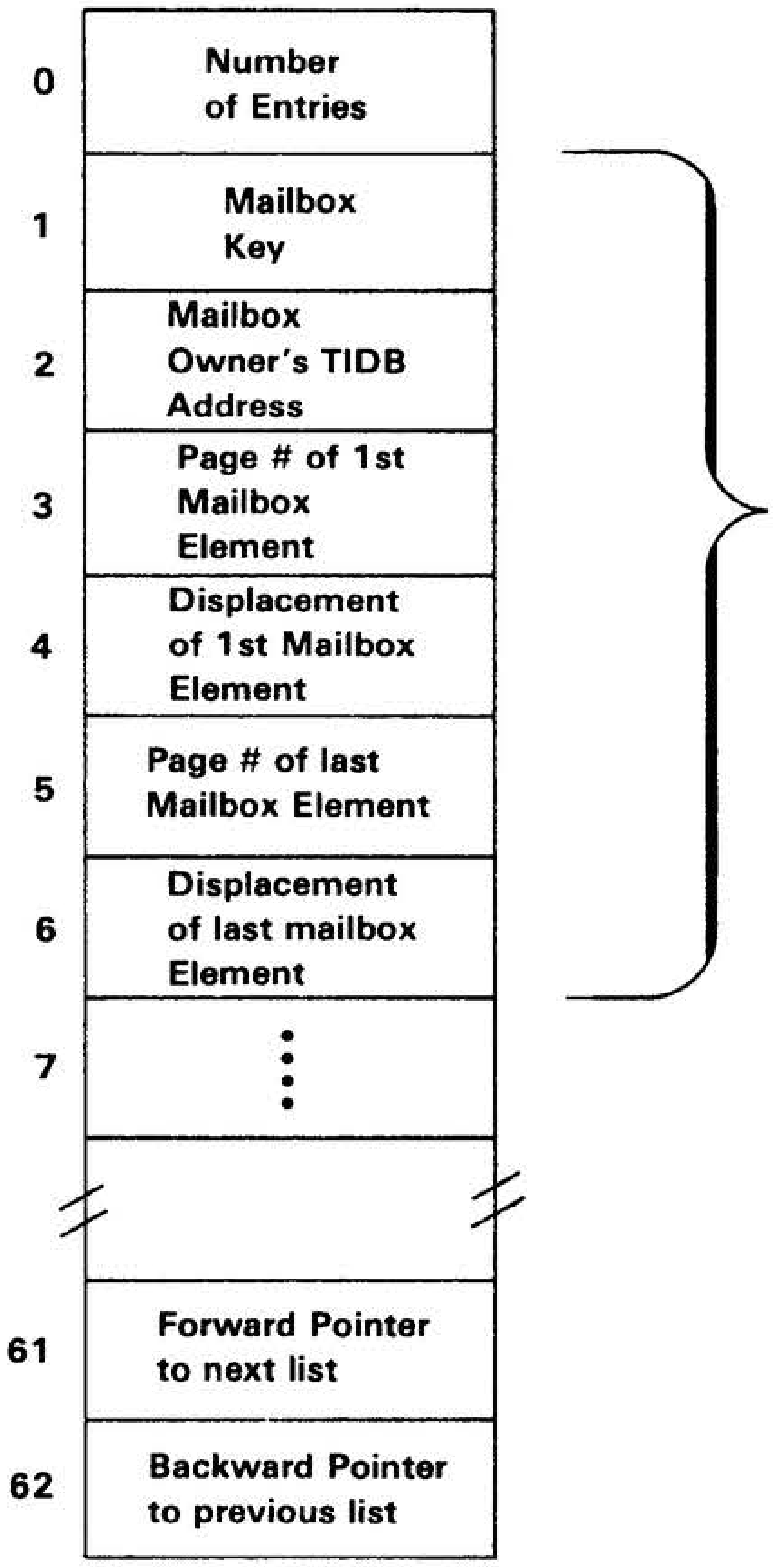


PAGE ACTION

14-34 (continued)

Mailbox initialization consists of calling ITE through the appropriate entry point to establish ownership of a mailbox. ITE will assign the mailbox key which will be associated with a mailbox. The mailbox will actually be an entry in the mailbox list.

Mailbox List Description


0
Number of Entries
1
Mailbox Key
2
Mailbox Owner's TIDB Address
3
Page # of 1st Mailbox Element
4
Displacement of 1st Mailbox Element
5
Page # of last Mailbox Element
6
Displacement of last mailbox Element
7
61
Forward Pointer to next list
62
Backward Pointer to previous list

Number of active mailboxes in list (maximum = 10)

Mailbox (6 words/entry)

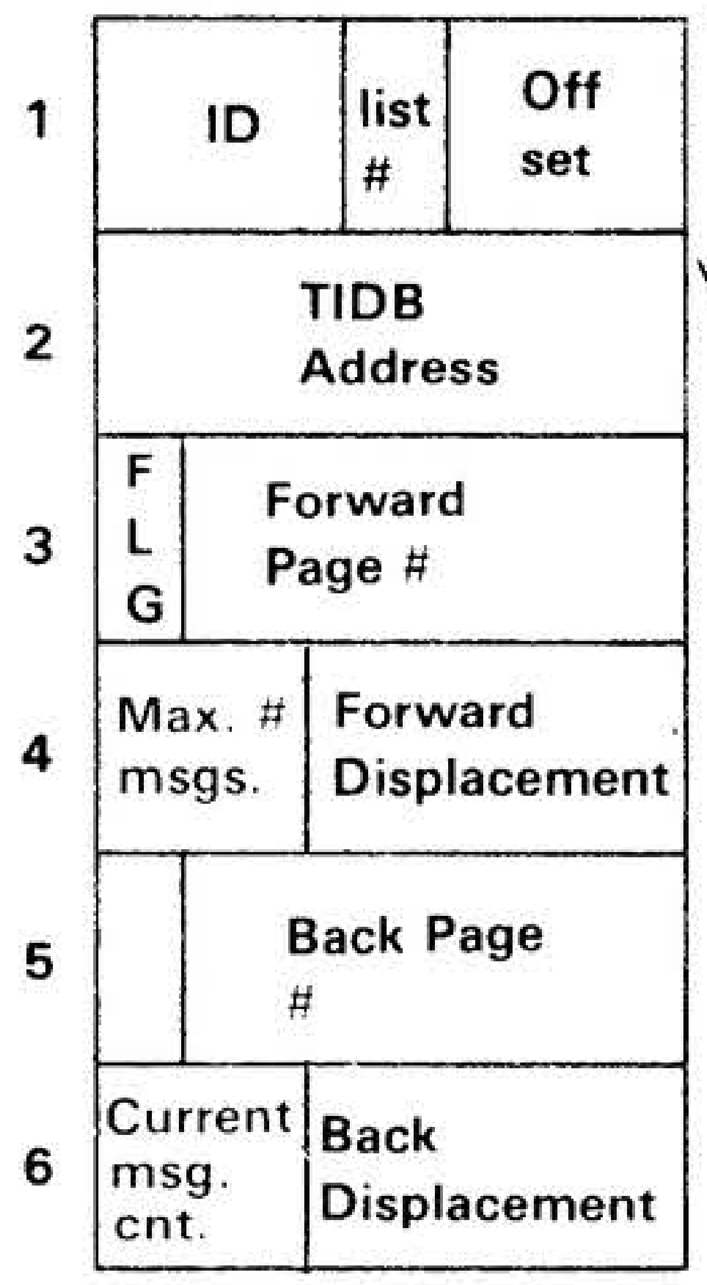
Pointer to next mailbox list (set to zero if none)

Pointer to previous mailbox (set to zero if none)

PAGE ACTION

14-34 (continued)

Mailbox Entry Description



Word 1
Mailbox Key

Bits 0 - 5: An offset value from the start of the table pointing to the start of this entry.

Bits 6 - 8: List segment sequence number. This field identifies in which segment the mailbox entry corresponding to this key resides.

Bits 9 - 15: Identification number used to ensure uniqueness between successive assignments of the key. The value is generated by taking the current value, incrementing by one, then taking the MOD 128 of it each time the key is reassigned.

Word 2
TIDB address of mailbox owner

Word 3

Bits 0 - 13: Physical map image of the first element in this mailbox queue

Bit 15: If set, message copied and page list pending

Word 4

Bits 0 - 9: Displacement of the first element

Bits 10 - 15: Maximum number of messages that may be queued to this mailbox (currently set at 10_{10}

PAGE ACTION

14-34 (continued)

Word 5

Bits 0 - 13: Physical map image of the last element in this mailbox queue

Word 6

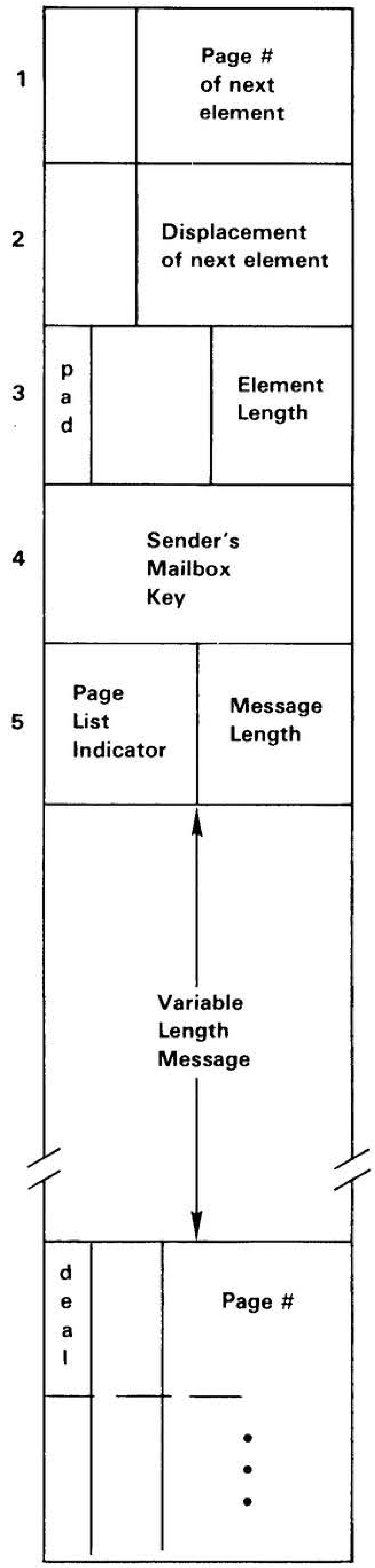
Bits 0 - 9: Displacement of the last element

Bits 10 - 15: Current count of message queued to this mailbox

PAGE ACTION

14-34 (continued)

ITE's Internal Mailbox Element Description



Word 1

Bits 0 - 14: Physical map image of next element in the queue

Word 2

Bits 0 - 9: Displacement of next element

Word 3

Bits 0 - 5: Length of mailbox element in words

Bit 15: Pad Flag: 0 means no pad word added
1 means pad word added to prevent a free element with a length of 1

Word 4

Sender's mailbox key (the value placed in this word is not validated by ITE)

Word 5

Bits 0 - 5: Message length in words

Bits 8 - 13: Page list indicator 0 means no page list present, otherwise, the field contains the length of the page list.

The sum of message length and page list length must not exceed

Word 6 +: Message

Word 7 + Message Length: Page list if present

Bits 0 - 9: Physical Page number

PAGE ACTION

14-34 (continued)

Bit 15: Deallocation flag
1 means the page has been unlinked from sender's map
0 means page is mapped to sender

ITE places the owners TIDB address in the entry point and initialized the pointers of the mailbox queue and increments the entry count of the mailbox list. The first list will reside within ITE. When that list is filled, memory from the dynamic pool will be used to form another list. These lists are linked forward and backward to allow deallocation of memory should a list at the end of the chain become empty.

The procedures for sending messages consists of constructing a Message Control Block as shown below.

1	Receiver's Mailbox Key
2	Sender's Mailbox Key
3	MSG WD 1
4	MGS WD 2
5	Page List Indicator

Word 1: Destination mailbox key

Word 2: Sender's mailbox key(the value in this word is not validated by ITE)

Word 3: If positive, this word contains the length in words, of the message to be posted. If negative, this and word 4 contain the message to be posted.

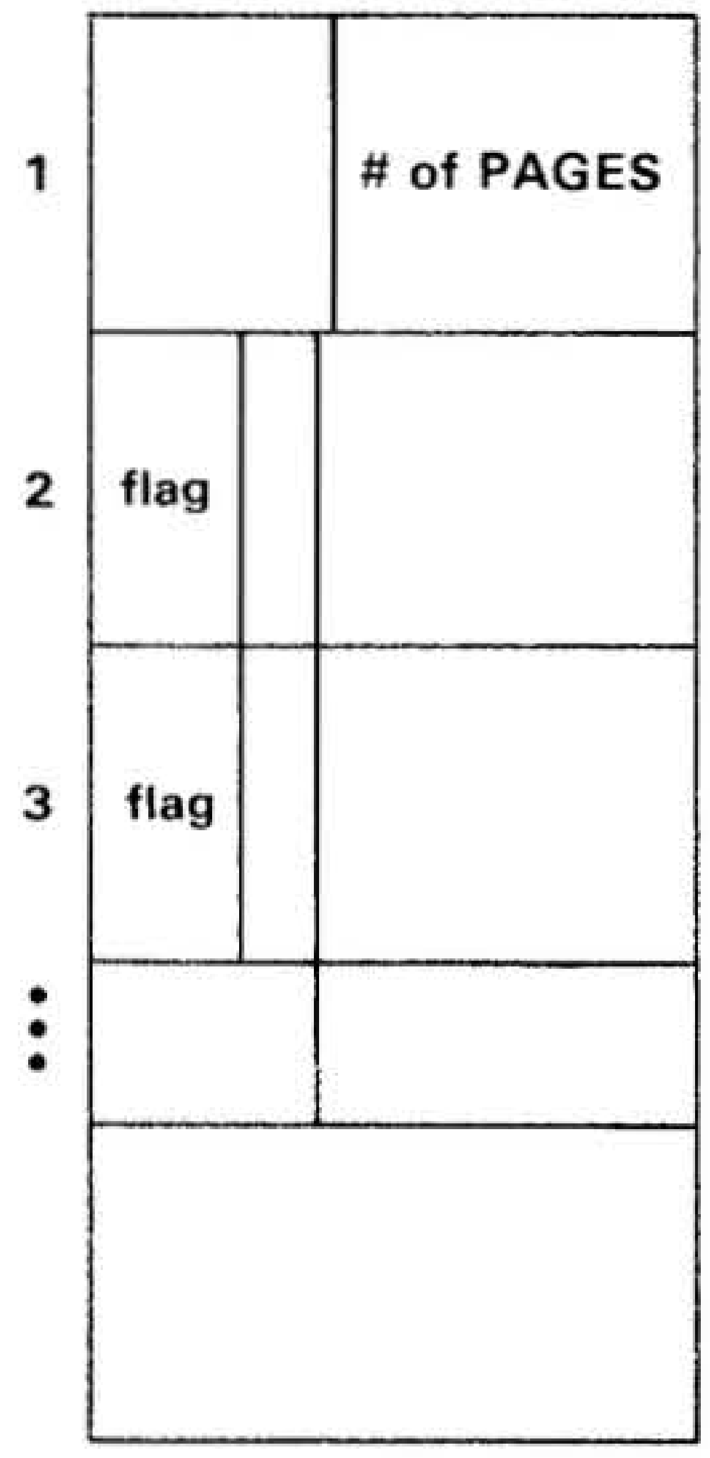
Word 4: If word 3 is positive, this word contains the address to the message. If word 3 is negative, this word contains data (i.e., message is embedded in MCB.)

Word 5: If positive, this word is the address to the page list, containing logical page addressess.
If negative, this word is the l's complement of the address to a page list containing physical page numbers. If zero, no page list is to be posted.

PAGE ACTION

14-34 (continued)

The MCB will contain the destination mailbox key and optionally, the senders key. If the sending task does not have a mailbox, the senders mailbox key should be set to zero by the sending task. If the sending task has pages to transmit, the page list indicator is set to the address of a page list containing the page numbers. The following illustration shows the format of the physical page list.



Word 1

Bits 0 - 5: Number of pages to be transferred

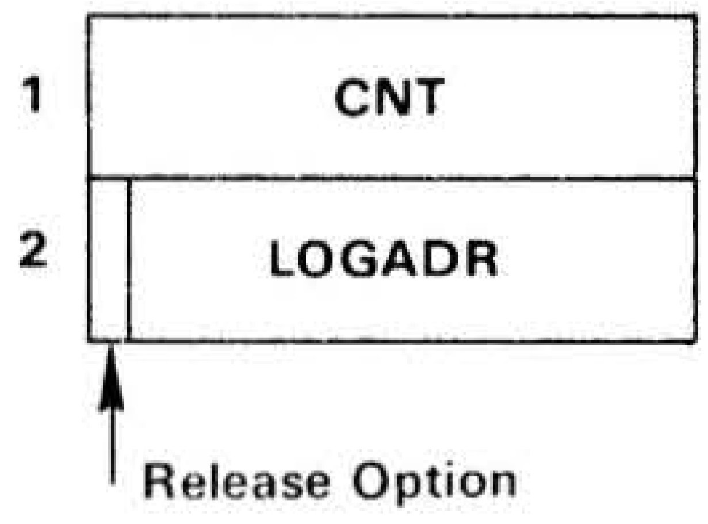
Word 2 - n

Bits 6 - 9: Physical page number, 1 page #/entry

Bit 15: Receiver mapping option flag. If zero: map the page (i.e., set bit 14 of the receivers map image word.)

If set to 1, reallocate the page to the receivers map (i.e., Bit 15 of the map image word will be set.)

A logical page list has the form:



Word 1 The number of pages to be transferred.

Word 2

Bits 0 - 14: The starting logical address in the senders map modules 1000_8

(i.e., on a page boundary) from which the page or pages will be fetched.

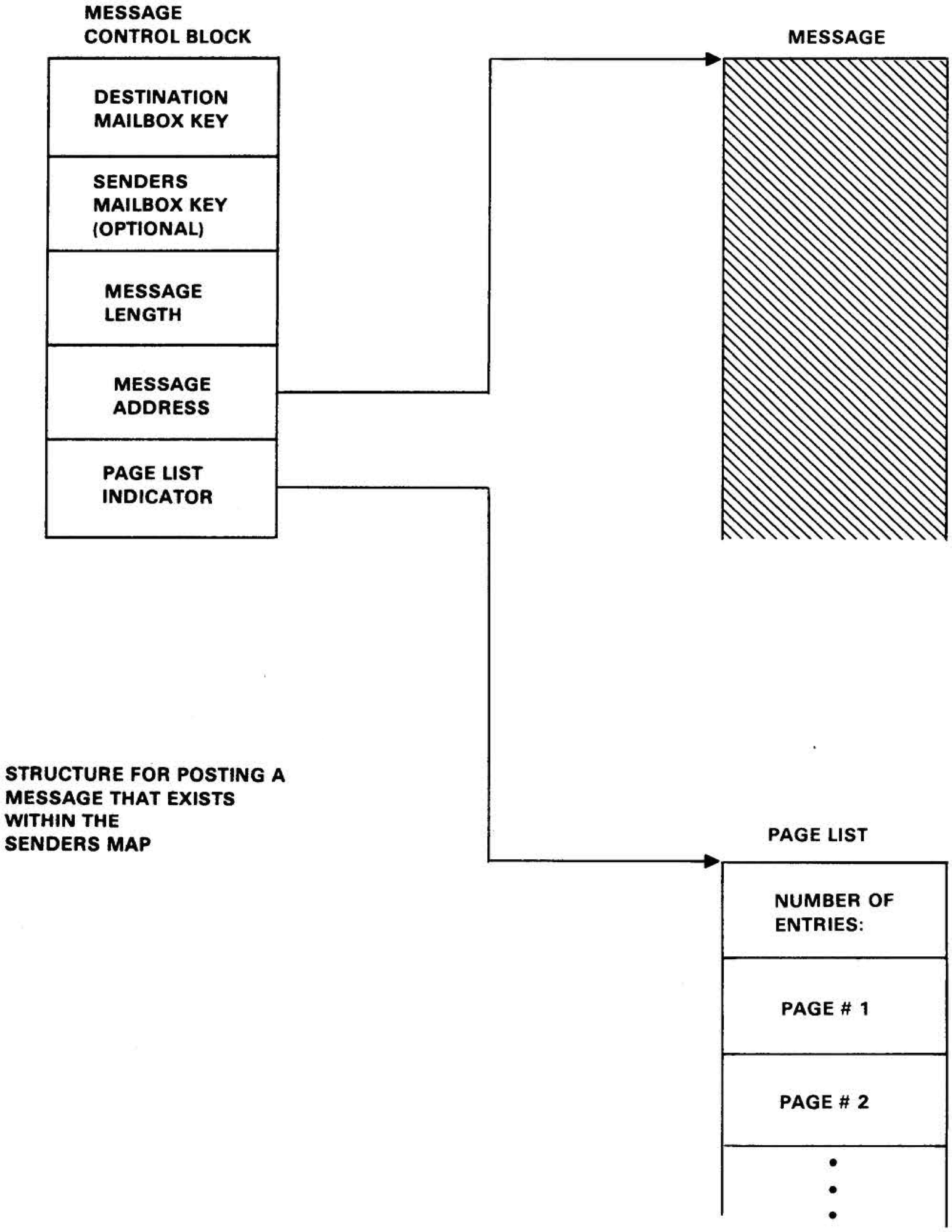
Bit 15: release option 0 = Release Pages (i.e., unmap from senders map)

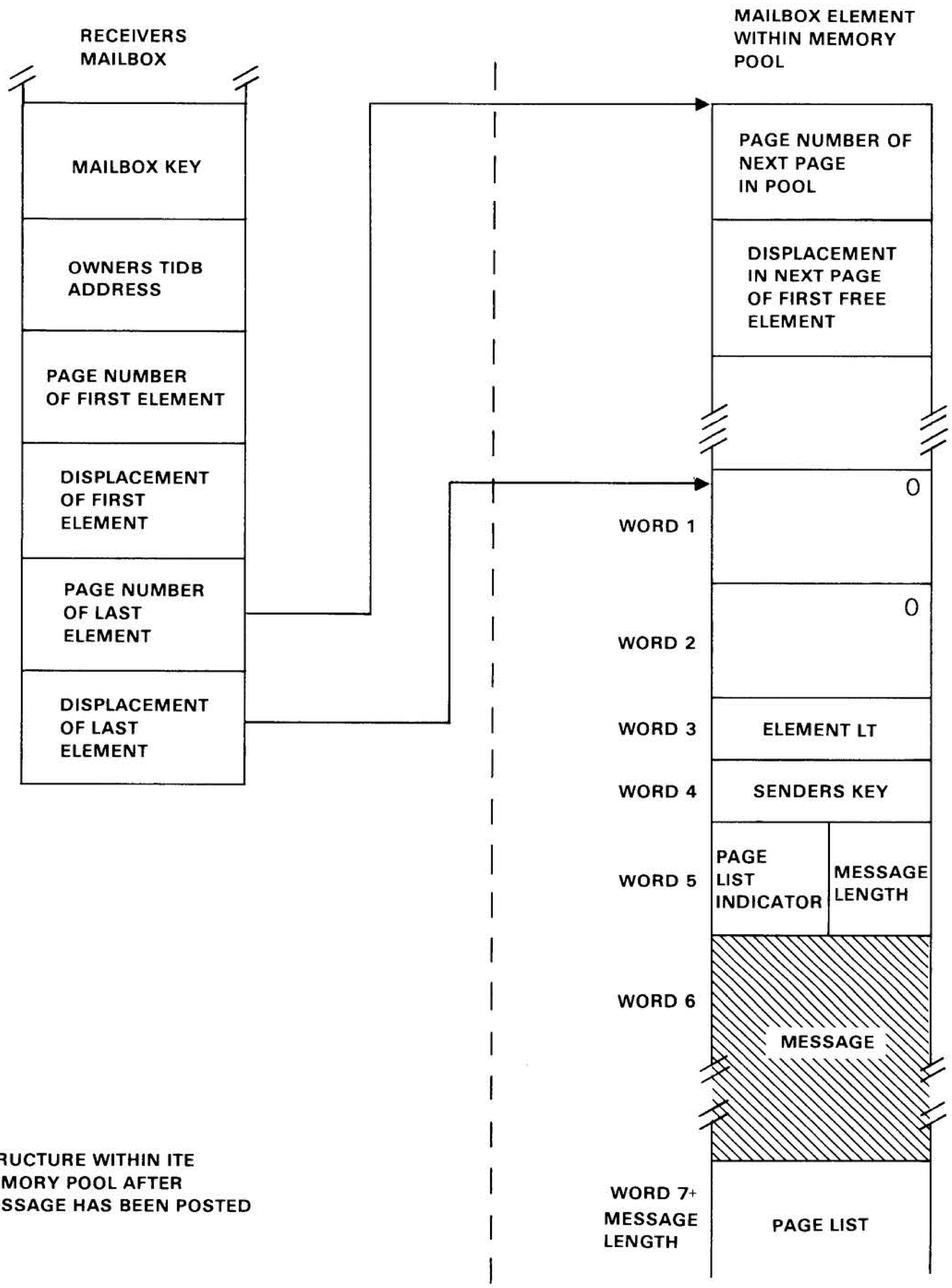
1 = do not unmap pages.

PAGE ACTION

14-34 (continued)

ITE copies the MCB, message and page list, if one has been provided, into its memory pool to form a Mailbox Element. The Mailbox Element is queued to the mailbox indicated by the mailbox key. ITE will then set Bit 0 of the receiving tasks TBEVNT word and exit. If the page list is provided and the pages to be transferred are also to be unmapped from the senders map, ITE will perform the unmapping as part of the message posting procedure. The following is a data flow that occurs with the posting of a message.

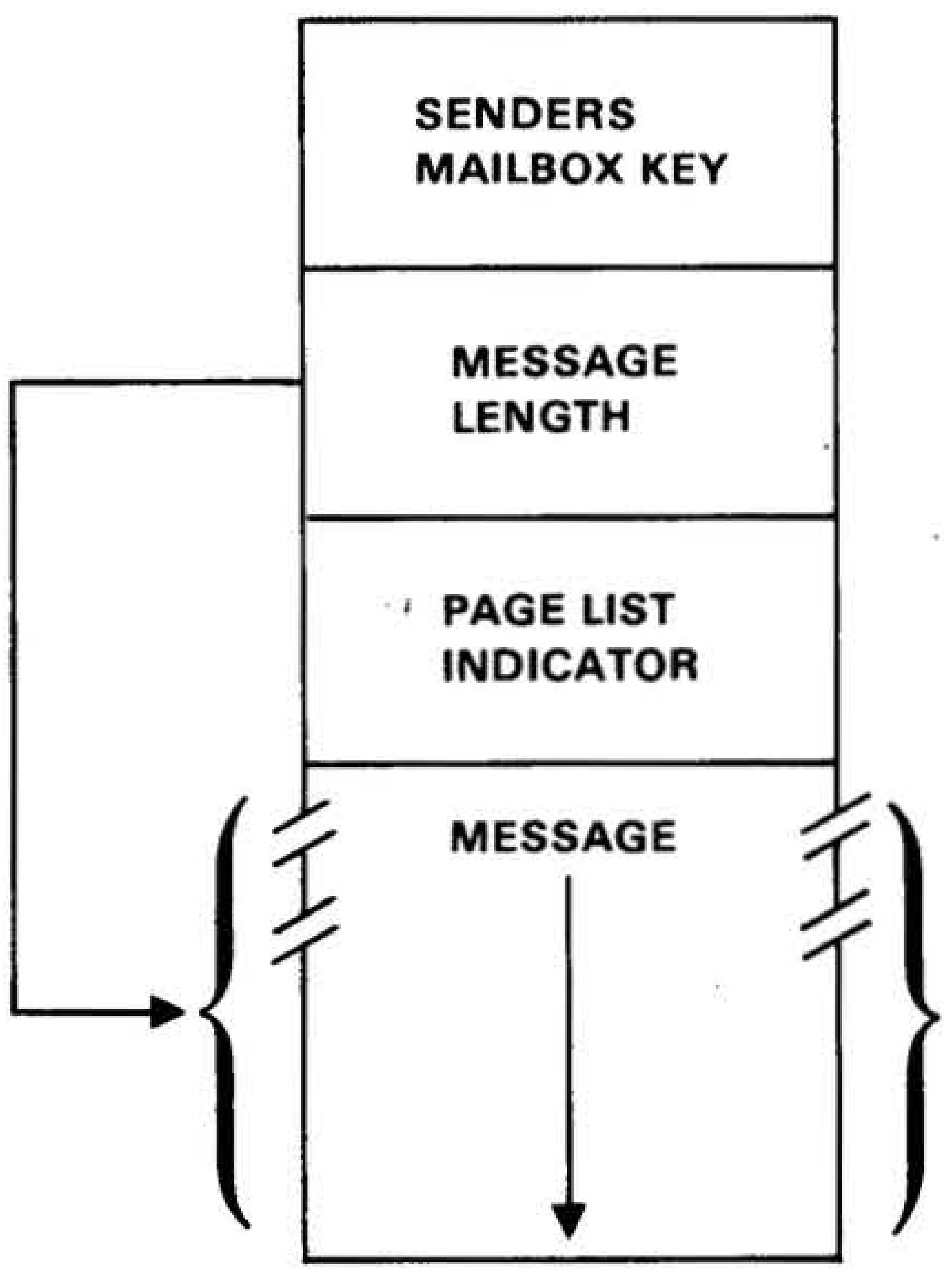



RECEIVERS MAILBOX
MAILBOX KEY
OWNERS TIDB ADDRESS
PAGE NUMBER OF FIRST ELEMENT
DISPLACEMENT OF FIRST ELEMENT
PAGE NUMBER OF LAST ELEMENT
DISPLACEMENT OF LAST ELEMENT
MAILBOX ELEMENT WITHIN MEMORY POOL
PAGE NUMBER OF NEXT PAGE IN POOL
DISPLACEMENT IN NEXT PAGE OF FIRST FREE ELEMENT
0
WORD 1
0
WORD 2
WORD 3
ELEMENT LT
WORD 4
SENDERS KEY
WORD 5
PAGE LIST INDICATOR
MESSAGE LENGTH
WORD 6
MESSAGE
WORD 7+ MESSAGE LENGTH
PAGE LIST

STRUCTURE WITHIN ITE MEMORY POOL AFTER MESSAGE HAS BEEN POSTED

PAGE	ACTION
14-34	(continued)

To receive a message or to determine whether or not the queue is empty, a task will issue a request to ITE to copy a message. ITE will validate that the requesting task is the proper receiver by comparing the tasks TIDB address with that contained in the mailbox. ITE will then use the forward pointer words of the mailbox to locate and dequeue the next element. If the forward pointers are zero, ITE will return a completion status indicating that the queue is currently empty. If the queue is not empty, the length of the next element is compared with the length of the buffer specified by the receiving task. If the length of the buffer is too small, ITE will return a completion status to inform the task, and message will not be dequeued.



BUFFER ADDRESS SPECIFIED BY RECEIVING TASK

NOT PRESENT IF MESSAGE LENGTH SET TO ZERO

PAGE ACTION

14-34 (continued)

The above illustration shows the format the message has when copied into the receiving tasks buffer. The space occupied by the mailbox element is returned to ITE's free memory pool. If the newly released element is next to another free element, the two are concatenated, forming one large free element space. This is done to limit the fragmentation within the memory pool.

Free Memory Pool Header Description

1	PAGE NUMBER	Physical page map image of first page in memory pool
2	DISPLACEMENT	Displacement of the first available element within this page (Set to zero if the page has no elements available.) This header is resident in the ITC Module. A similar header is located in the first two words of each page in the pool. The last page in the pool has these words set to zero.

Free Element Description

1	FORWARD POINTER	Pointer to the next available length within this page Set to zero if this element is the last in the page.
2	ELEMENT LENGTH	Length of this element (in words)

If the page list indicator is non-zero, the receiving task must get its pages before it can proceed to the next mailbox element. To check for a pages pending status, the sign bit of the forward page number word in the mailbox, for this task, will be set. This bit is not cleared until all pages have been processed or the receiving task requests that the remaining be dropped. Once the page list is entirely processed or dropped, the mailbox element is released back to ITE's memory pool.

PAGE ACTION

14-34 (continued)

Request Parameter Block to Get a Message

RECEIVER'S MAILBOX KEY	Receiving task's Mailbox Key
BUFFER ADDRESS	Logical address of receiver's message buffer
BUFFER LENGTH	If positive, message buffer length in words If negative, (1's complement of buffer length in words) indicates the page list is to be included in the message.

The receiving task will issue a request to ITE to retrieve a page or pages. If more than a single page is requested, the pages are linked to the receivers map contiguously beginning from the specified logical address. If the page(s) have been previously allocated to the sending task and the sender requested that the pages be unlinked, ITE allocates the pages to the receiving task's map (i.e., Bit 15 in the map image word(s) used, is set). Otherwise, the pages will only be mapped in (i.e., Bit 14 of the map image word(s) used will be set.)

In the event that the receiving task does not need to get the pages once it has copied the message, it can issue a request to drop those pages. This request will then dispose of the pages as required and release the mailbox element to the ITE memory pool.

The entry point VI$ITC is used as a housekeeping agent for the memory pool within which a task has exited or aborted. This entails checking the tasks queue for mailbox elements. If none are present, the mailbox is cleared. If the mailbox queue is not empty, ITE will release and consolidate the task's mailbox elements before clearing the mailbox. All other entry points which delete mailbox elements, first check to verify the presence of any mailbox elements. A completion status will be returned if there are elements left in the mailbox.

ITE utilizes several macros. In addition to the existing calls for ITC. ITE uses modified versions of VI$PST and VI$CPY. ITE also uses four unique calls of its own; VI$GTK (get a key), VI$GTP (get a page), VI$DRP (drop a page) and VI$RMB (Release a mailbox). ITE also uses the entry point VI$TRT; this will contain the logical starting address for mapping operations.

PAGE ACTION

14-34 (continued)

VI$PST

The calling sequence for the ITE VI$PST is:

R0 = address of the message control block
R1 = 0100000_8

Return:

R0 = Completion status
-1 = ITC is busy -- Go to sleep and try later
0 = Successful completion
1 = ITC memory pool has not been initialized
2 = Invalid or outdated mailbox key
3 = Map loading error
4 = Not enough or no free space available
5 = Receiving task's queue is full
6 = Message length error (too long or sum of PG list & MSG = 0)
7 = Page list error (page unassigned or not legal) or starting address together with number of pages exceeds map
8 = Page list specified with zero page count

If R0 = 7:
R1 = Starting logical address if page count exceeds map or logical

LABEL ALOC VI$PST

VI$CPY

The calling sequence for the ITE version is:

R0 = Pointer to 3 word block containing the calling task's mailbox key, the address of the buffer, and the length of the buffer.

R1 = 0100000_8

PAGE ACTION

14-34 (continued)

Return:

R0 = Completion Status
-1 = ITE busy -- Go to sleep and try again later
0 = Successful completion
1 = ITE memory pool not initialized
2 = Invalid key
3 = Map loading error
4 = Buffer length too short
5 = Page list pending
6 = Mailbox queue is empty

R1 = Number of words required if R0 = 4

VI$GTK

The entry point VI$GTK is used by a task requesting a mailbox key. No entry parameters are used.

On Exit: R0 will contain the completion status

R0 = Completion status
0 = Successful completion
1 = Memory not available for extending list
2 = No more segments available for extending mailbox list
3 = Map loading errors
4 = No pages have been specified for the ITE memory pool
5 = No physical pages available for memory pool
6 = Dynamic memory required for initialization unavailable
7 = No space available in map 0 for the ITE memory pool

R1 = Mailbox key if R0 is zero

PAGE ACTION

14-34 (continued)

VI$GTP

VI$GTP is used by the receiving task to get the page or pages sent with a message. If the page(s) were deallocated from the sender's map, the pages will be allocated to the receiver (Bit 15 of the corresponding map image word will be set). If the pages were not deallocated by the sender, ITE will map the pages (Bit 14 of the map image word will be set.)

Calling sequence:

R0 = address of the request parameter block, consisting of three words:

Word 1 = requestor's mailbox key

Word 2 = logical address from which to begin linking the page or pages

Word 3 = the number of pages to link

R1 = The number of pages to process

Return:

R0 = Completion status, where

-1 = ITE is busy -- Go to sleep and try again later

0 = Successful completion

1 = ITE memory pool has not been initialized

2 = Invalid mailbox key

3 = Map loading error

4 = No page list is pending

5 = Logical address or number of pages invalid

R1 = Number of pages processed if R0 = 0, or
The logical address in error if R0 = 5.

VI$DRP

VI$DRP is used by the calling task to drop the page list of the current message. This function enables the user to proceed to the next message without first processing the pages associated with the prior message.

PAGE ACTION

14-34 (continued)

Calling sequence:

R0 = Mailbox Key

Return:

R0 = Completion status, where:

```
-1 = ITE is busy -- Go to sleep and try again later
 0 = Successful completion
 1 = ITE memory pool not initialized
 2 = Invalid mailbox key
 3 = Map error
 4 = No page list pending
```

VI$RMB

VI$RMB is used by the calling task to relinquish ownership of a mailbox.

Calling Sequences:

R0 = mailbox key of entry to be released or zero if all mailboxes belonging to the task are to be released.

Return:

R0 = Completion status, where:

```
-1 = ITE is busy -- Go to sleep and try again later
 0 = Successful completion
 1 = ITE memory pool has not been initialized
 2 = Invalid key
 3 = Map error
 4 = No mailboxes assigned to calling task
 5 = A mailbox with an unempty queue encountered
```

R1 = Key of the mailbox if R0 = 5

PAGE ACTION

14-34 (continued)

LIMITATIONS, RESTRICTIONS & CONSIDERATIONS

1. Any task establishing ownership of an ITE mailbox will have the responsibility of checking the mailbox for messages prior to exiting and going to sleep.

2. A task which issues an I/O without wait and then issues a delay type 3 will have to determine for itself what event caused to be activated.

3. If a task exits or aborts prior to emptying its mailbox, ITE will release the mailbox elements but no notification will be made to the senders that messages were thrown away.

4. Four possible cases can occur in the disposition of a physical page when a sender requests that the page be transferred by ITE.

   a. The sender allocated the page and requests that the page be unlinked from his map.

   b. The sender allocated the page and does not want it unlinked.

   c. The sender mapped in the page and wants it unmapped.

   d. The sender mapped in the page and does not want it unmapped.

In the first case, the ownership of the page would be transferred to the receiving task (i.e., the receiving task would be able to deallocate the page and release it to VORTEX). In all the other cases, the page would merely be mapped into the receiving task's logical memory. Therefore, in the event that an abort occurs in the receiving task before it was able to get its pages, ITE will release to VORTEX only those pages which would have been allocated to the receiver, (Case A.)

PAGE ACTION

14-34 (continued)

SYSTEM GENERATION REQUIREMENTS

The following DEF directives are required:

```
DEF,VI$MXQ,n
```

where n = number of elements in the main box queue. The value is placed in bits 9-15 and must be octal i.e., a value of 10 would be represented by DEF,VI$MXQ,012000.

```
DEF,VI$NPG,m
```

where m = number of physical pages to be used for ITE internal pool. This number of pages is made unavailable to VORTEX.

15-3a Figure 15-2

Change figure 15-2 to

```
     o
     o same as before
     o

CTL, PART0003
Library Processor
Firmware (V77-600)
System Library Routines

     o
     o same as before
     o
```

Add to last paragraph:

The library processor also creates an eight page firmware file named WCSIMG on the partition assigned to logical unit 116. This file is built only if the system contains WCS and is not a V77-400.

PAGE ACTION

15-13 Add to the Preset logical unit/RMD partition table:

Name	Number	Partition
optional	116	optional*

* must reside on the system RMD, be at least 160 sectors, only required for V70/V77-600 systems with WCS.

15-19a Add new section 15.5.20:

15.5.20 MOD directive

This directive performs the same function as the EQP (section 15.5.2) directive except that a controller table link is not generated for the module. It has the same parameters as the EQP directive. It is used to select a nucleus module under the same criteria as the EQP directive, but when the nucleus module does not have a corresponding controller table.

15-18 Replace first paragraph of section 15.5.15 with:

This directive, which must be the last SGEN directive, specifies several system parameters for the system generation. The directive form is

```
EDR,type,tidb,stack,part,list,kpun,map,analysis
```

where

type is S for a standard (full) system generation. In this mode, the entire system is regenerated and all partitions are initialized (existing files are lost.)

PAGE ACTION

15-18 (continued)

N for a nucleus only system regeneration. In this mode, only the nucleus image and SGEN generated load modules are recreated. Existing file directories are not destroyed. Note: this mode has the following restrictions:

1) The "memory" parameter on the MRY directive must be the same as the previous SGEN.

2) SGEN catologued load modules must be the same size (in sectors) as the previous SGEN versions and must reside in their original sectors.

3) All existing non SGEN created load modules must be "relinked" using RELINK (section 6.4.)

4) All RMDs must be partitioned the same as the previous SYSGEN.

This mode is not generally usable between versions of VORTEX. It is normally used to change a nucleus component or add a nucleus component on the same revision of VORTEX.

PAGE ACTION

15-19a Add to required directives section:

V77-600 (and V70) systems containing WCS require a DEF directive to select the appropriate WCS modules (firmware options), The format is:

```
DEF,V$$WCS,n
```

where n is

1. for FORTRAN accelerator (no FPP) and commercial firmware.
2. for FORTRAN accelerator (with FPP) and commercial firmware.
3. for commercial firmware only.

This directive is not required on V77-400 systems. Note: the unit parameter on the WCS EQP/MOD directive is set to the number of pages of WCS.

```
ASN,116=D00n
```

where n is a partition on the system RMD which is to contain the WCS image file. This partition must contain at least 160 sectors, have no protection key, must not be initialized, and cannot be the same partition as PO, SS, GO, SW, CU, BI, BO, FL, BL, CL, or OM (partitions on which SYSGEN creates files).

17-4 Change TSTAT as indicated below:

task tidb addr Plevel Sstatus TMmin TSmilli

where task as before

tidb addr TIDB address of the subject task all else as before

A-20 Add following error message:

SG47	Number of WCS pages specified on EQP directive does not accomodate selected option (DEF, V$$WCS).	SYSGEN terminates, waits for action.	D16,D12