SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

05/025

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA

(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina USA 27709

SPLA CONTROL NUMBER: 232

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the SHARE Reference Manual, Section 6.

(1) Program Number (to be filled by SPLA) 360D-05.1.025

(2) Title of Program . . . . . HASP Performance Improvement Modifications, SEAS-542-002

(3) System Type(s) (Machine) S/360

(4) Search Key(s) . . . . .

(5) Programming Systems/Languages . . . . ASSEMBLER

(6) Primary Subject Code . . . . .

(7) Minimum System Requirements OS/MVT, HASP 3.1

(8) New (N) or Revision (R) (if revision, show prior Program Number in Item 1) N

(9) Date of Submittal 12/78

(10) Documentation (number of original pages submitted) . . . . . 30 PAGES

(11) Author's Name and Address . . . . . A. E. STORMER, SEAS
Rutherford Laboratory
Chilton, Didcot, Oxon OX11 0QX
ENGLAND

(12) Direct Technical Inquiries to Name & Address (if different than Author) SAME

(13) Submitter's Installation Membership Code SEAS-542

(14) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ATTACHED

DISTRIBUTION TAPE:
1 FILE DCB=(RECFM=FB, LRECL=80, BLKSIZE=32720)
NO LABEL

DISCLAIMER

Triangle Universities Computation Center (TUCC) serves solely as the distribution agent for contributed programs and does not test or maintain them. They are distributed essentially in the original form submitted by the author. Neither TUCC nor SHARE, INC., makes any warranty, expressed or implied, as to the documentation, function, or performance of the contributed programs.

(Please attach additional pages if necessary) . . . . . Total pages attached _____

An "Acknowledgement of Assistance" statement must be attached to this Submittal Form.

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program"

(15) Signature of Submitter and Date _____

(15) Signature of Installation Addressee _____

SEAS SHARE European Association

SUBMITTAL FORM FOR SEAS ITEMS TO EUROCOPI

Catalogue number (to be filled by EUROCOPI): SEAS-S42-002

Owner: (Name) A.E. STORMER ..... SEAS code: S42 .....

(Address) Rutherford Laboratory, Chilton, Didcot, Oxon OX11 0QX England .....

Name and address of individual to whom technical enquiries about the submitted item should be sent:

(Name) A.E. Stormer .....

(Address) Rutherford Laboratory, Chilton, Didcot, Oxon OX11 0QX England

Title (max. 2 lines in English):

HASP Performance Improvement Modifications

Brief abstract:

Several small areas of code consume the bulk of the cpu time used by Hasp. These areas have been located and are found to be concerned with acquisition of remote device control blocks and the dispatching of inactive remote processors. Several modifications have been installed which reduce the cpu time consumed by Hasp by some 78% of its original value on the Rutherford Laboratory IBM 360/195.

Keywords (according to the standard classification scheme):

Index.

Appendices.

1. The Rutherford Laboratory Environment.

The Rutherford Laboratory runs an IBM 360/195 under the control of Hasp version 3.1 and OS/MVT release 21.8.

Hasp at RL has been extensively modified to provide such facilities as workload control (1), setup volume pre-mounting (2), conditional job control (3), multiple remote console support (4) and auto-answer dial-in capability (5).

Hasp controls 2 local card readers, 4 local printers and currently 32 remote lines supporting 36 remote workstations.

The remote lines at RL are all, in Hasp terms, "dial-in", though most are on physical leased lines. This allows for greater control of remote workstations in the event of line failures of one form or another.

As most lines are in fact leased lines to particular remote workstations, each workstation normally signs onto a given line. However, in the event of a line fault it may be possible for that workstation to either sign onto a backup line or onto a genuine dial-in line.

Most remote workstations are multileaving cpu workstations, in general GEC2050s with RL written control programs, though a few are 2780s or cpus running 2780 emulators.

2. Observed Inefficiencies.

Running analysers on Hasp at RL show several small areas in which most of Hasp's cpu time is consumed. Further, it can be detected in user jobs that Hasp often holds the cpu for several tens of milliseconds. This obviously does no good to a terminal multi-access system running at a lower priority than Hasp.

The general areas of inefficiencies are found to be the dispatcher and the service routine to aquire a unit ($GETUNIT).

It is observed that a large number of remote processors wait on UNIT which when dispatched following a post UNIT fall back into the same wait on UNIT, possibly having called $GETUNIT to scan for a device on the way. These processors therefore cause a high and unnecessary load on the Hasp dispatcher and $GETUNIT.

The following is also noted:

(a) when a reader becomes active a post UNIT is issued, as readers wait on UNIT;

(b) every processor that manipulates a job will issue at least one post UNIT;

(c) every change of job status issues a post JOB;

(d) each job goes through at least four processors;

(e) when a job finishes printing/punching the printer/punch is freed causing a post UNIT;

(f) most write/read sequences on a remote line supporting a multileaving workstation are followed by a post UNIT if a post JOB has been issued since the last transaction on that line and there is an inactive printer or punch on it.

There are therefore about 6 post UNITs and 5 post JOBs caused by each job.

Further, if there is a high enough transaction rate on the

remote lines, nearly every post JOB issued will be followed by as many posts of UNIT as there are remote workstations with one or more inactive printers or punches.

It can be seen that inefficient code in the dispatcher and in the $GETUNIT service routine will impose a high cpu penalty on Hasp and can lock out user jobs for considerable lengths of time.

These inefficiencies have all been tackled.

3. Hasp Dispatcher.

Each Hasp processor is controlled by a processor control element (pce) which contains a two byte event wait field (ewf). The first byte of this field is concerned with global resources such as buffers and maps onto the single byte global Hasp event completion field ($HASPECF).

$HASPECF is used to post all pces. When the dispatcher is entered, $HASPECF is tested. If any bit other than the general POST bit is off a global resource has been posted.

In standard HASP the complete pce chain is then scanned, the first byte of each ewf being masked by the bits of $HASPECF thus clearing processor waits for the appropriate resources.

Having modified the ewfs, the pce chain is again scanned but this time the full two byte ewf of each pce is tested. If the ewf in a pce is zero, the processor is then dispatched.

Return is made from the processor to $WAIT in the dispatcher. Here $HASPECF is tested to see if any resource has been posted during the execution of the processor. If it has then the dispatcher is reentered. Otherwise the next pce is located and its ewf tested as above.

Modification RHEL47.

A simple modification which will save some time, though not a lot, is to remove the initial masking scan and to combine the masking with the ewf test. This test now becomes an NI on the first ewf byte which, if the byte is then clear, is followed by a CLI on the second ewf byte.

Return from a processor tests $HASPECF as before but if it finds that the processor has posted a resource the ewfs of the remaining pces in the chain are then masked with the previous $HASPECF before reentry to the dispatcher.

If, however, the previous $HASPECF had no global resources

posted, thus producing a null scan, the dispatcher is reentered immediately.

4. $GETUNIT - Get a Device.

All devices used by Hasp have a device control table (dct). These dcts are all chained together in a single chain.

Remote devices are chained during initialization such that all devices of a given remote workstation are chained consecutively, whereas all other devices are grouped according to device type. If other than default devices are generated they are recommended to be generated adjacent to other devices of the same type.

Every time there is a post UNIT in the system several processors are going to be dispatched as UNIT is a global resource. Some such as remote reader processors are going to call $GETUNIT to test for the availability of work. If there is no unit available this call causes the complete dct chain to be scanned.

Modification RHEL43.

It is assumed that all devices of a given type have their dcts generated consecutively. It is further assumed that all dcts of a given type are of the same length.

As part of the assembly of HASPINIT, two tables are created with one entry in each for each dct type. One table contains the relative addresses of the first dct of each type and the second table the length of each type of dct. In order to eliminate the test in $GETUNIT for the last dct in the chain a dummy dct with the type X'FF' is generated following the last genuine dct.

$GETUNIT in HASPNUC is also modified such that:

(a) scanning of dcts starts at the first of the specified type;

(b) scanning of dcts is by incrementing the dct address by the appropriate length not via the

chain field;

(c) scanning of dcts terminates on detection of a different type of dct;

(d) the test for the last dct is eliminated as a dummy dct follows the last genuine dct.

5. Remote Workstation Processing.

Each remote line has a line dct associated with it. During initialization OS UCBs and Hasp line numbers are connected either as specified to the Hasp generation process or as found.

Lines can be, in Hasp terms, "leased" or "dial-in".

A "leased" line is always associated with a given remote workstation.

A "dial-in" line and a remote workstation are connected via a sign-on sequence on that line.

During Hasp initialization all the dcts of a given remote workstation are chained together and their status set to HOLD. In the case of a "leased" line the line characteristics are transfered to the remote device dcts as well. Line charcteristics are transferred to "dial-in" remote device dcts during the sign-on sequence.

Following initialization all the remote processors are dispatched.

Each remote processor will then call $GETUNIT which, in the case of "dial-in" lines, will be unsuccessful followed by the processor waiting on UNIT. This will be repeated every time a post UNIT is issued.

When a remote workstation has signed on one or more post UNITs are issued causing the remote processors waiting on UNIT to be dispatched.

For every printer and punch on the remote workstation just signed-on, a remote processor will now successfully get a dct and then attempt to get a job via the service routine $QGET. If no job is available for that remote device the pce will again wait on UNIT having set the dct in a HOLD state. If the remote workstation is a multileaving cpu, a post UNIT will be issued by the line manager and the dct reset from the HOLD state after

completion of the next transaction on that line if a post JOB has been issued since the last transaction. Thus all inactive remote printer and punch processors will be dispatched whenever a post JOB is issued and a transaction, including idle handshaking, completes on any remote line.

Inactive remote reader processors will be waiting on UNIT, dcts only becoming available when a reader actually starts reading cards. Inactive readers whether or not the workstation is signed-on are therefore dispatched for every post UNIT that is issued.

Modification RHEL49.

To reduce the load imposed on the dispatcher and $GETUNIT by signed-off remotes the appropriate pces need to be removed from the pce chain. This is achieved as follows:

(a) remote pces are assembled in HASPNUC out of the pce chain in subsidiary pools;

(b) sign-on of "dial-in" lines and initialization of "leased" lines cause pces of the appropriate type to be set in the pce chain for each dct on the remote workstation;

(c) sign-off of "dial-in" lines causes pces of the appropriate type to be removed from the pce chain as and when they become quiesced.

A new processor, the pce manager, is assembled into the pce chain for each remote pce type at the point where the remote pces are to be inserted. This processor removes pces from the chain when it is posted. It is normally waiting on WORK.

The addition of pces to the chain is accomplished by a nucleus service routine ($GETPCE). This routine acquires a pce from the appropriate pool and sets it in a held state in the pce chain prior to the appropriate pce manager. Normally $GETPCE returns the pce address. If, however, a pce is not available a

counter is decremented to indicate overbooking and an error return is set up. In HASPINIT, setting up pces for "leased" lines, this is ignored. However the line manager in HASPRTAM setting up pces during sign-on will issue a warning message to the operator that contention may occur between remote devices of a particular type. If contention is to be avoided care must be taken to ensure that a sufficient number of pces are generated to satisfy all possible coincident requests. The error code returned from $GETPCE is the negative of the number of pces of that type currently overbooked.

A nucleus service routine ($FREEPCE) is called by the line manager in HASPRTAM as part of the line disconnect sequence for each device on the remote workstation during sign-off. If the line is a "leased" line the call to $FREEPCE is bypassed.

As the processors on that workstation may currently be active they cannot be removed easily from the pce chain by the line manager. Instead a counter is incremented to indicate the number of pces of a given type that need to be removed and the appropriate pce manager is posted.

As the pce manager is of lower priority than the remote processors, any outstanding activity on a remote processor will have the opportunity to clear itself before the pce manager is dispatched.

When it is dispatched the pce chain is scanned in reverse from the pce manager looking for remote pces that are quiescent and can therefore be removed. To determine whether or not a pce is quiescent the dct attached to that pce is tested. If the dct is flagged as INUSE and it is still attached to the pce then the processor is still active and cannot be removed. Otherwise it is quiescent. As many quiescent pces as are specified by the counter set by $FREEPCE are located and removed from the chain after which the pce manager again waits on WORK. If there are not enough quiescent pces to satisfy the counter, control is returned to the dispatcher but the pce manager is left dispatchable. It

will therefore be restarted each time the pce chain is scanned until all requests to remove pces have been satisfied.

If the service routine $FREEPCE finds that overbooking has occured the appropriate overbook counter is incremented but as no pce needs to be freed the pce manager is not posted.

The pce manager in conjunction with the service routines $GETPCE and $FREEPCE therefore maintains as many pces as are necessary in the chain to satisfy all the remote devices on "leased" lines and signed-on "dial-in" lines if at all possible.

Modification RHEL48.

Each "leased" line or signed-on "dial-in" line processor that is inactive is dispatched every time a post UNIT (for readers) or JOB (for printers and punches) is issued and calls $GETUNIT to scan the dcts. A large proportion of these dispatches will result in the processors returning to the same state, the dispatch and call of $GETUNIT having been wasted.

Even though the overhead in $GETUNIT can be dramatically reduced by modification RHEL43 previuosly described, several dcts will still need to be scanned on each call.

Modification RHEL49 is enhanced to reduce this overhead as follows:

(a) when a pce is set into the pce chain via $GETPCE it and the dct of the remote device concerned are cross connected;

(b) HASPRDR and HASPPRPU are modified so that the dct address is aquired from the pce as set up above and not via $GETUNIT;

(c) $FREEPCE sets a flag in the pce associated with the specified dct to indicate that that pce is one to be freed;

(d) this flag in the pce is tested by the dispatcher

before each pce is dispatched. If it has been set on as above and the pce is quiescent then the dispatch is bypassed and the next pce serviced;

(e) the remote pce manager tests for the flag set in the pce above before testing to see if the pce is quiescent;

(f) if on entry to $GETPCE there are one or more pces waiting to be freed, the pce connected to the dct is located. If the flag has been set in the pce by $FREEPCE as above and the pce points to the dct, the flag is cleared, the free count decremented and that pce used instead of one from the pool;

(g) enough pces should be generated to satisfy all coincident requests as each pce will only operate on that remote device to which it is attached at sign-on time or initialization. When a pce of the appropriate type is not available to satisfy a request to $GETPCE overbook is not permitted, unlike RHEL49 where it is. A code of zero will be returned and the line manager will abort the sign-on, removing any pces already aquired, and inform the operator.

Modification RHEL52.

During channel end processing following a transaction with a remote workstation, the line manager checks for a normal response. If it is normal and JOB has been posted since the last transaction on that line, every inactive printer or punch on that remote workstation is reset from the HOLD state and a post UNIT issued.

The number of posts of UNIT in the channel end processing are reduced as follows, based on modification RHEL48:

(a) when an available printer or punch dct is found on the line, the attached pce is posted on UNIT, thus making dispatchable only the appropriate pces.

It can be noted that, as the pces posted by the channel end processing in the line manager are of lower priority than the line manager itself, the general POST bit in $HASPECF does not need to be posted.

6. $FREUNIT DA - Free a Direct Access Device.

Every processor that manipulates a job requires the use of a direct access dct.

Enough direct access dcts can be generated to satisfy all possible requests. Thus no processor need wait for a direct access dct.

Unless the dct is in a HOLD or DRAIN state when $FREUNIT is called, a post UNIT is issued. As no processor sets the direct access dct in either of these states before calling $FREUNIT a post UNIT is issued every time.

Modification RHEL50.

It is ensured that enough direct access dcts are generated to satisfy all possible coincident requests.

$FREUNIT is modified to bypass the post UNIT when the dct is a direct access dct.

The first printer/punch processor to be dispatched following a warm start updates control blocks from the checkpoint area. Any printer/punch processors dispatched during this updating are set to wait on UNIT. As the post UNIT is removed from $FREUNIT for direct access devices, the call to free the device used by the updating routine must be followed by a post UNIT.

7. Implementation.

All the described modifications are independant with the exception of RHEL48 which enhances RHEL49, and RHEL52 which depends on RHEL48.

Modification RHEL47.

The dispatcher modification involves the following change:

(a) HASPNUC - the dispatcher is modified.

Modification RHEL43.

The $GETUNIT modification involves the following changes:

(a) HCT - the Hasp Communication Table is extended by one word to contain the address of the tables set up in HASPINIT to be used by $GETUNIT in HASPNUC;

(b) DCT - the Device Control Table macro is modified to define a maximum size to be used for the DCT tables in HASPINIT and HASPNUC;

(c) HASPINIT - the $GENDCT in-line macro is modified to create entries in the DCT tables as part of the assembly;

- the DCT tables are defined immediately preceding the DCTs;

- a dummy DCT is defined immediately following the DCTs;

- the HCT initialization is extended to set the DCT tables address in the HCT;

(d) HASPNUC - the $GETUNIT service routine is changed to make use of the DCT tables via the HCT.

Modification RHEL49.

The basic remote device pce modification involves the following changes:

(a) $GETPCE - a macro to call the $GETPCE service routine is added;

(b) $FREEPCE - a macro to call the $FREEPCE service routine is added;

(c) $PCEGET - a macro to define the $GETPCE service routine is added;

(d) $PCEFREE - a macro to define the $FREEPCE service routine is added;

(e) $PCEMAN - a macro to define the pce manager is added;

(f) $PCERTAM - a macro to define the action to be taken by the line manager when a pce is not available during sign-on is added;

(g) HCT - the Hasp Communication Table is extended by two words which contain two branch instructions into the service routine interfaces;

(h) HASPINIT - remote dct initialization is modified to include a call to $GETPCE which, in the case of a "leased" line remote, will set a pce of the appropriate type in the pce chain;

(i) HASPNUC - the $GENPCE macro is modified so that remote pces are assembled in subsidiary chains attached to the pce manager;

- the numbers of remote pces generated are defined as global assembler variables;

- interfaces to the service routines are

defined within the range of the Hasp base register;

- the service routines and the pce manager are assembled at the end of the HASPNUC code prior to the pce chain definitions;

- a pce for the pce manager of each type is assembled in the main pce chain following the remote pce chain definitions;

- the remote pce chains are terminated using the global assembler variables defined above;

(j) HASPRTAM - a call to $GETPCE is inserted in sign-on processing followed by a post of the aquired pce;

- the action to be taken when a pce is not available is defined;

- a call to $FREEPCE is inserted in "dial-in" remote disconnect processing.

Modification RHEL48.

The enhancement to the remote device pce modification involves the following changes:

(a) $PCEGET - the service routine $GETPCE in HASPNUC is modified to disallow overbooking and to use the connected pce if it is waiting to be freed;

- $GETPCE is further modified to set the DCT address in the aquired pce at location PDCT (if PDCT in the print/punch pce and RDRDCT in the reader pce are not the same an assembly error will occur) and the pce

address in the DCT;

(b) $PCEFREE - the service routine $FREEPCE in HASPNUC is modified to flag the pce being freed;

(c) $PCEMAN - the pce manager in HASPNUC is modified to only free pces if they have been flagged by $FREEPCE;

(d) $PCERTAN - the action to be taken when a pce is not available during sign-on in HASPRTAM is replaced;

(e) HASPNUC - a test is inserted in the dispatcher before a pce is dispatched in order that the dispatch can be bypassed if the pce is about to be freed;

(f) HASPRDR - the call to $GETUNIT is replaced to aquire the remote reader dct from location RDRDCT in the pce;

(g) HASPPRPU - the calls to $GETUNIT are replaced to aquire the remote printer or punch dct from location PDCT in the pce.

Modification RHEL52.

The channel end processing modification involves the following changes:

(a) HASPRTAM - the reader and printer/punch pce workareas are defined;

- the post UNIT is replaced by an NI on the ewf of the pce attached to the selected dct. This NI is the first instruction of the normal $POST macro.

Modification RHEL50.

The $FREUNIT modification involves the following changes:

(a) HASPNUC - a test for a direct access dct is inserted immediately preceding the $POST UNIT;

(b) HASPPRPU - a $POST UNIT is inserted following the call to $FREUNIT to release the direct access dct used in the warm start initialization.

8. Observed Improvements.

At RL some 12000 jobs are run in a week with about 50% of the jobs being submitted via Hasp RJE.

Normally about half of the workstations are signed on giving a load of about 20 remote readers, 22 remote printers and 18 remote punches.

The total workstation transaction rate is about 4 million transactions a week. During the day this corresponds to about 60000 transactions an hour.

Before any of the described performance modifications were implemented the Hasp cpu time was running at up to 20 hours a week. The overhead for each job was therefore some 6 seconds of cpu time.

When the $GETUNIT modification was installed the Hasp cpu time fell to about 7 hours a week giving an overhead per job of 2.5 seconds of cpu time.

The other modifications with the exception of the channel end processing modification were installed together and caused a further fall in Hasp cpu time to about 5 hours a week, the overhead dropping to some 1.8 seconds of cpu time per job.

The installation of the channel end processing modification brought the Hasp cpu time down to 4 hours a week giving a final figure of 1.4 seconds of cpu time per job.

An overall performance improvement of some 78% has therefore been achieved by the described modifications on the RL IBM 360/195. This improvement should be attainable at other installations if they have a large Hasp RJE network and a large number of small jobs as at RL.

References.

(1) COPPER - Workload Control on the Rutherford Laboratory Central Computer
M.H.Curtis RL/75/102

(2) SETUP - Preallocation of Mountable Devices on the Rutherford Laboratory Central Computer
A.R.Mayhook RL/../...

(3) A Multi-Job Facility A.R.Mayhook RL/../...

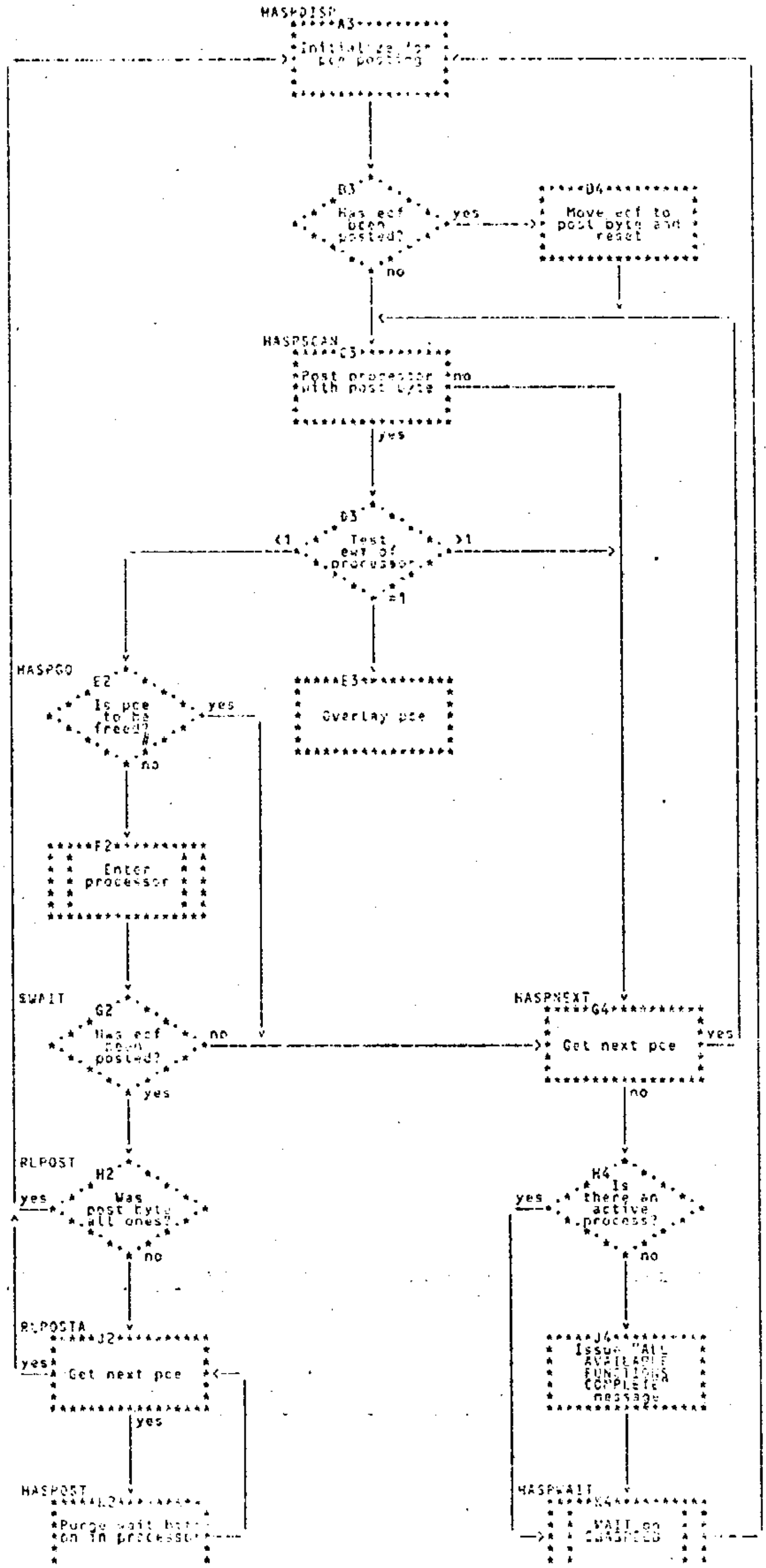
(4) HASP Remote Station Handling and Online Console Support P.M.Girard RL/73/074

(5) HASP Remote Line Auto-Answer Capability
P.M.Girard RL/../...

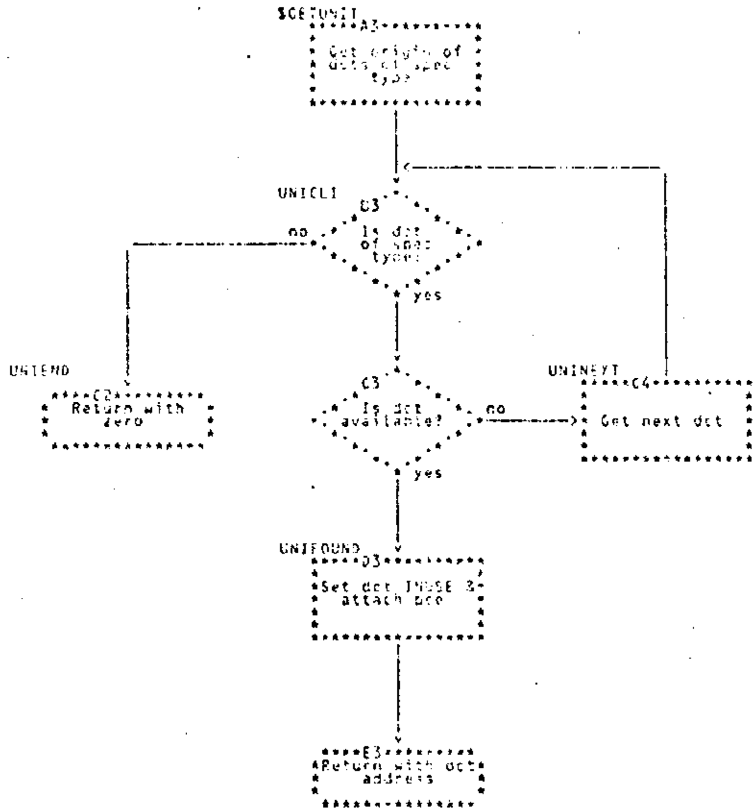
Acknowledgements.

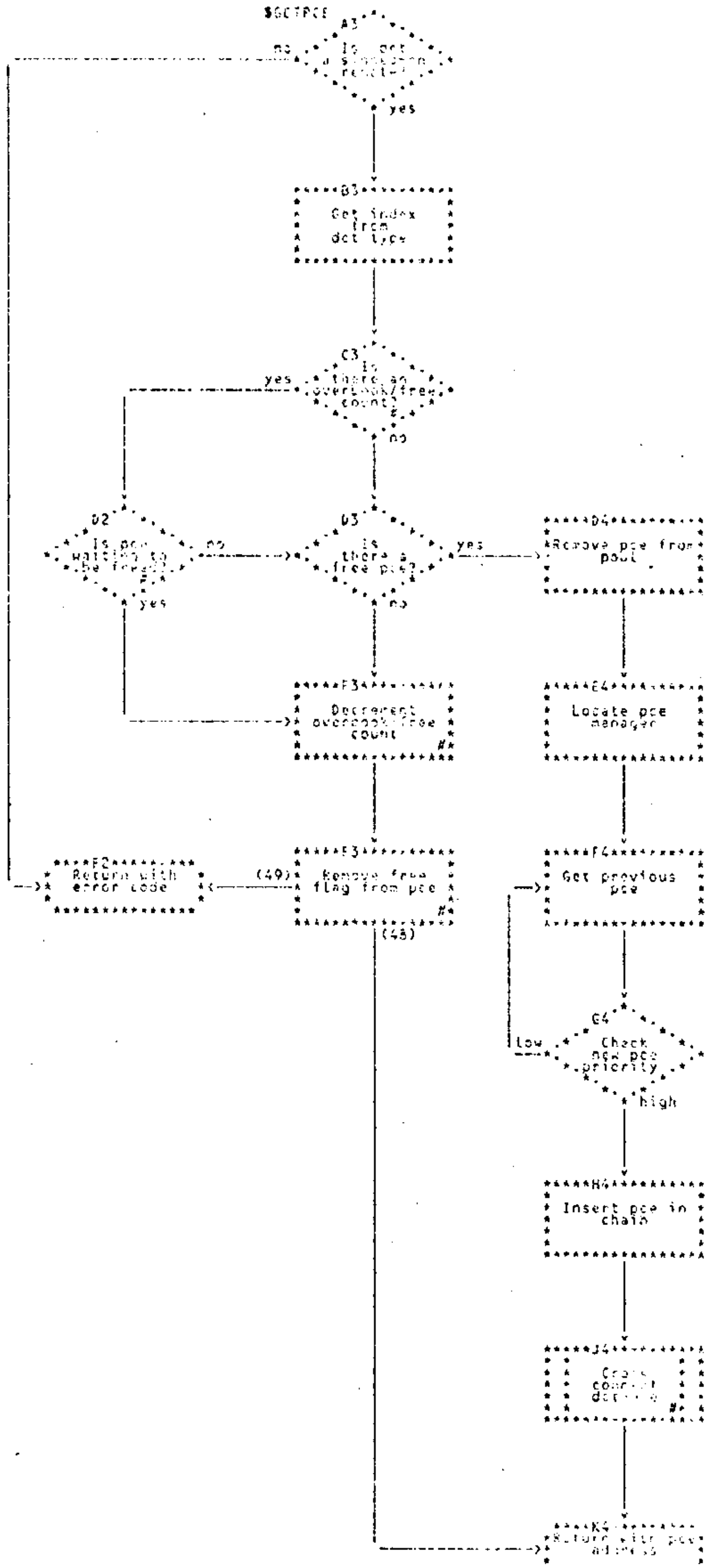
My thanks are due to my colleagues especially G.H.Adamson and P.M.Girard without whose intimate knowledge of Hasp none of the modifications in this report would have been achieved.

Appendices.



E2, [illegible] modification


$GETUNIT
A3
Get origin of dcts of spec type
UNICLI
B3
Is dct of spec type?
no
yes
UNIEND
C2
Return with zero
C3
Is dct available?
no
yes
UNINEXT
C4
Get next dct
UNIFOUND
D3
Set dct INUSE & attach pcb
E3
Return with dct address

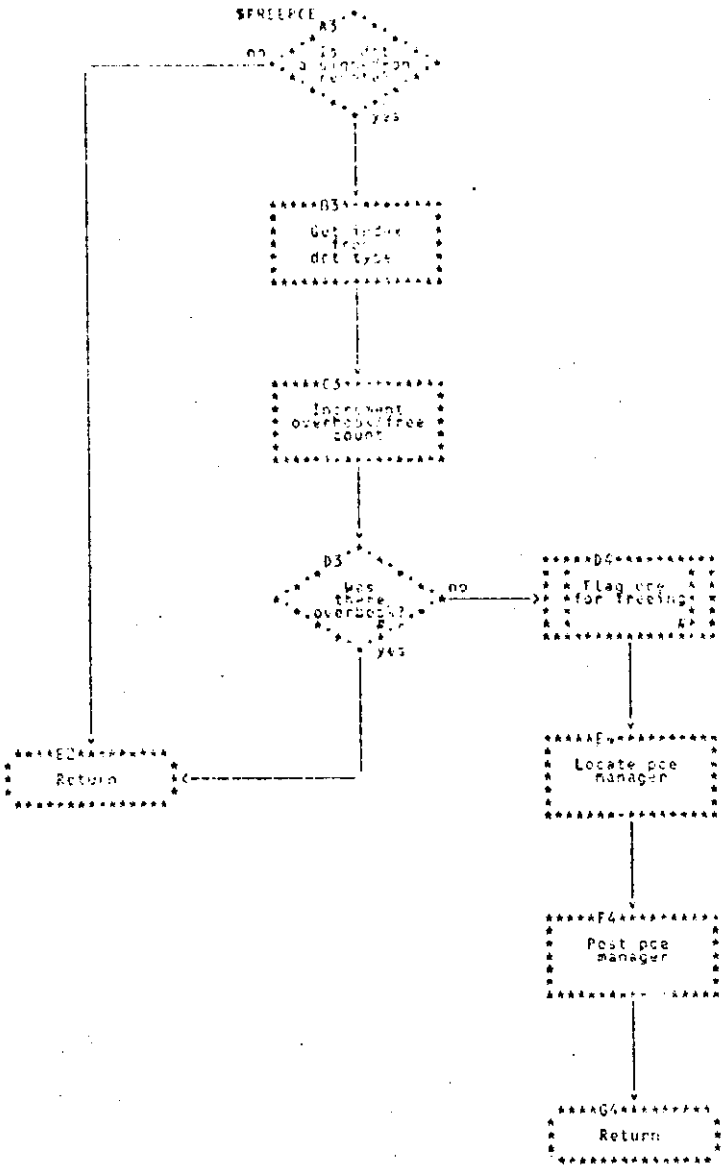


$GETPCE
A3
no
yes
B3
Get index from dct type
C3
Is there an overlook/free count?
yes
no
D2
Is pce waiting to be freed?
no
yes
D3
Is there a free pce?
yes
no
D4
Remove pce from pool
E3
Decrement overlook/free count
E4
Locate pce manager
F2
Return with error code
(49)
F3
Remove free flag from pce
(48)
F4
Get previous pce
G4
Check new pce priority
low
high
H4
Insert pce in chain
J4
K4
Return with pce address

D2.

C3.

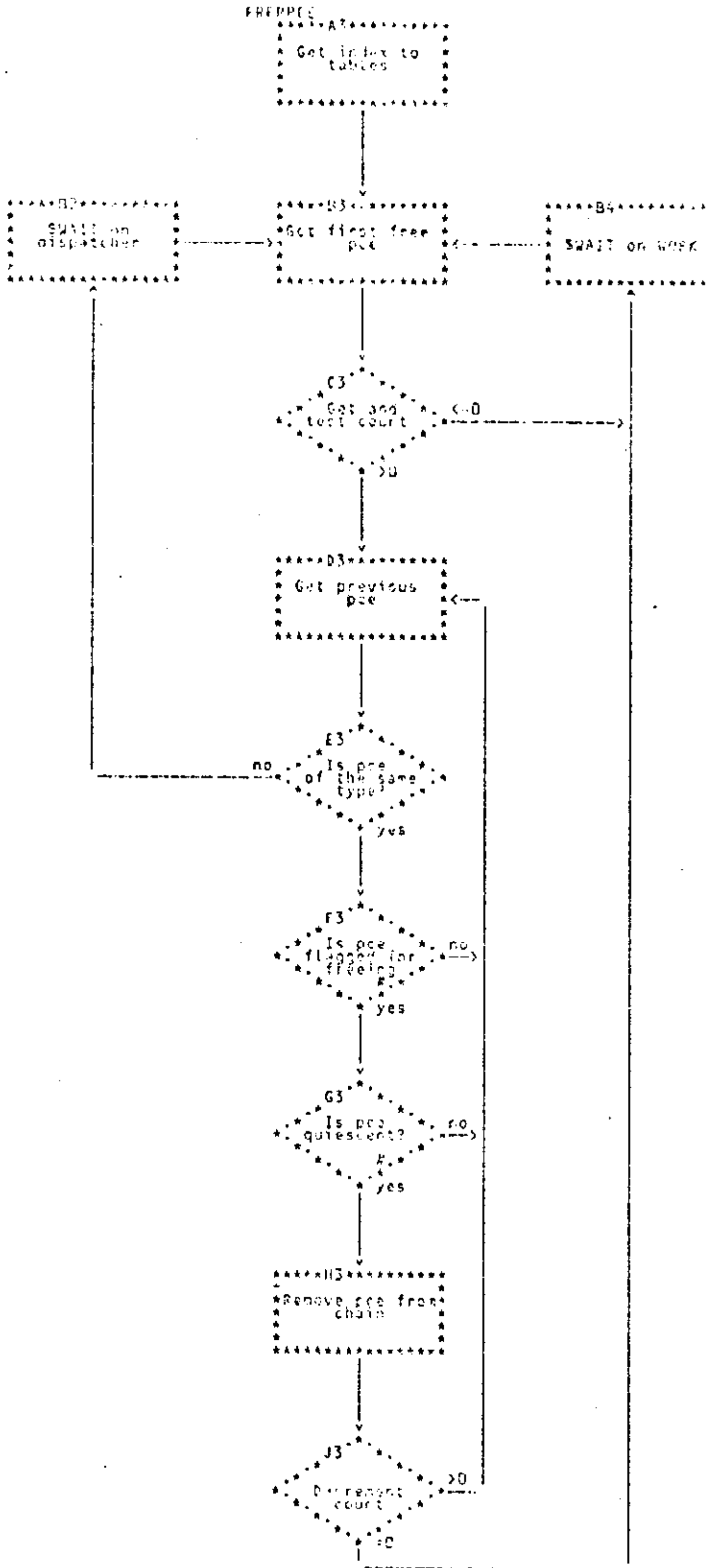
J4.


$FREEPCE
A3
Is not a [illegible] request
no
yes
B3
Get index for dst type
C3
Increment overboard free count
D3
Was there overboard?
no
yes
D4
Flag pce for freeing
E4
Locate pce manager
F4
Post pce manager
G4
Return
E2
Return

A3. [illegible]

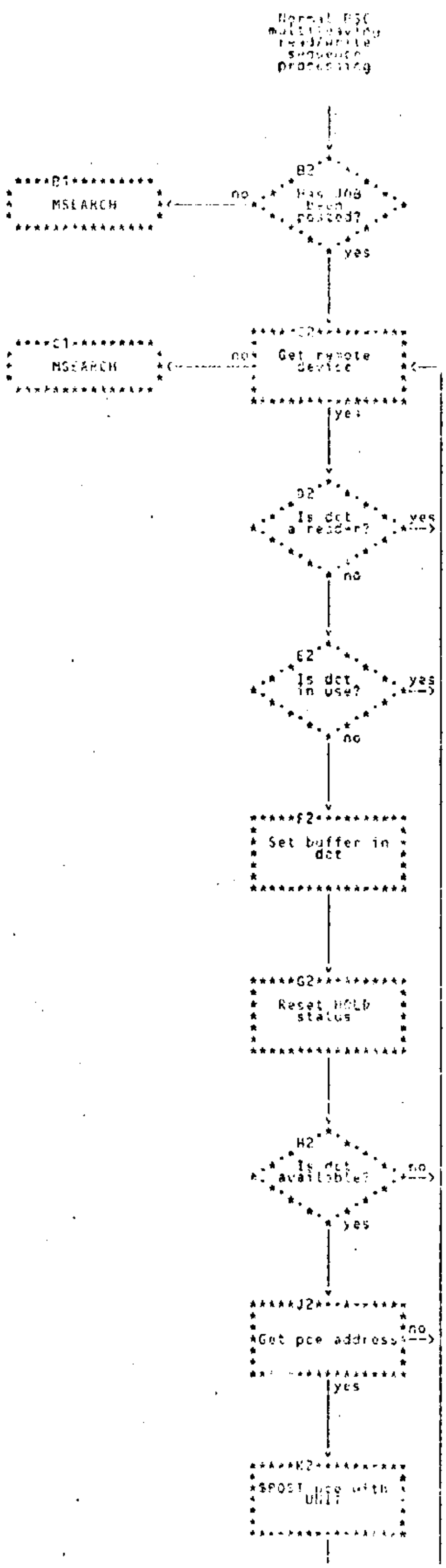
B3. [illegible]

D4. [illegible]

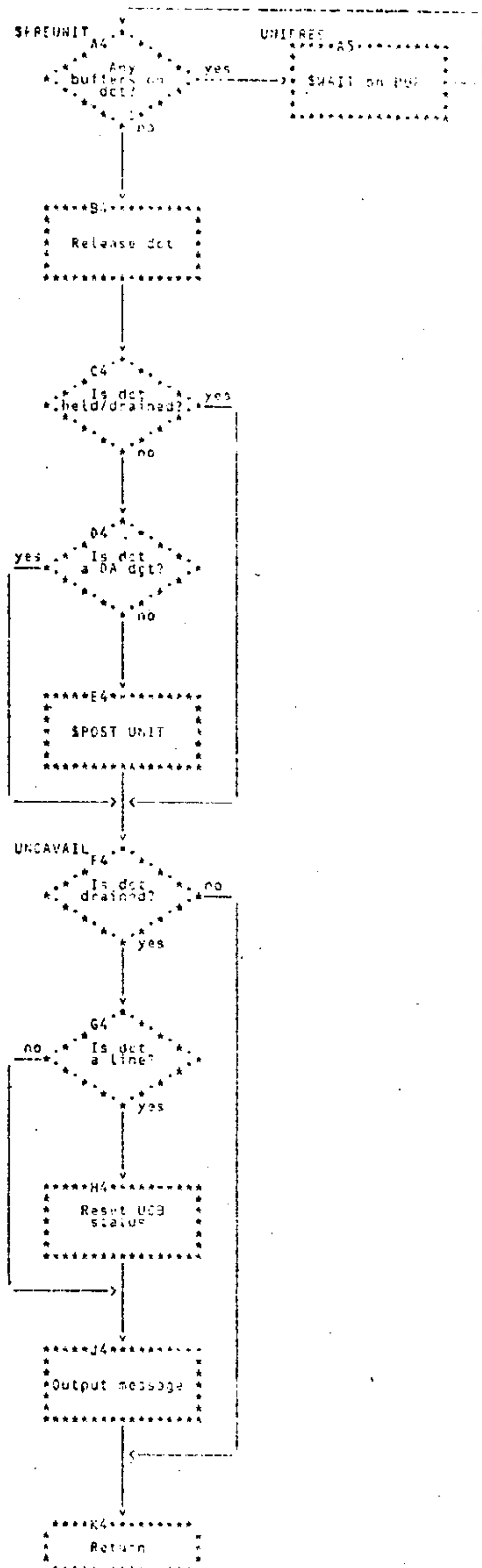

FREPPCE
A3
Get index to tables
B2
SWAIT on dispatcher
B3
Get first free pce
B4
SWAIT on WORK
C3
Get and test count
<=0
>0
D3
Get previous pce
E3
Is pce of the same type?
no
yes
F3
Is pce flagged for freeing
no
yes
G3
Is pce quiescent?
no
yes
H3
Remove pce from chain
J3
Decrement count
>0
=0

F3. [illegible]

G3. [illegible]



Normal BSC multileaving read/write sequence processing
B1
MSEARCH
B2
Has JOB been posted?
no
yes
C1
MSEARCH
C2
Get remote device
no
yes
D2
Is dct a reader?
yes
no
E2
Is dct in use?
yes
no
F2
Set buffer in dct
G2
Reset HOLD status
H2
Is dct available?
no
yes
J2
Get pce address
no
yes
K2
$POST pce with UNIT



$FREUNIT
A4
Any buffers on dct?
yes
no
UNIFREE
A5
$WAIT on BUF
B4
Release dct
C4
Is dct held/drained?
yes
no
D4
Is dct a DA dct?
yes
no
E4
$POST UNIT
UNCAVAIL
F4
Is dct drained?
no
yes
G4
Is dct a line?
no
yes
H4
Reset UCB status
J4
Output message
K4
Return