```
      2 MODULE jsm$swapped_job_manager;
      3
      4 {
      5 {     The purpose of this module is to provide the interfaces for managing swapped jobs
      6 {  that are still in memory.
      7 {
      8
o  1300
o  1301   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
o  1302     VAR ijle_p: ^jmt$initiated_job_list_entry);
o  1303
   1311
   1312   PROCEDURE [XREF] jsp$free_swap_resident_job
   1313     (     swap_resident_ijlo: jmt$ijl_ordinal;
   1314           swap_resident_ijle_p: ^jmt$initiated_job_list_entry);
   1315   PROCEDURE [XREF] jsp$monitor_advance_swap (ijl_ordinal: jmt$ijl_ordinal);
   1316
o  1319
o  1320   PROCEDURE [XREF] mmp$free_memory_in_job_queues (VAR job_page_queue_list: mmt$job_page_queue_list;
o  1321         increment_now: boolean;
o  1322         decrement_soon: boolean;
o  1323         job_termination: boolean);
o  1324
   1327
   1328 PROCEDURE [INLINE] tmp$clear_lock (VAR lock: tmt$ptl_lock);
   1329
   1330     IF osv$cpus_logically_on > 1 THEN
   1331       IF lock.id <> #READ_REGISTER (osc$pr_base_constant) THEN
   1332         i#program_error; {Interlock failure - no message passed for performance reasons}
   1333       IFEND;
   1334       IF lock.count > 0 THEN
   1335         lock.count := lock.count - 1;
   1336       ELSE
   1337         lock.clear := 0;
   1338       IFEND;
   1339     IFEND;
   1340
   1341 PROCEND   tmp$clear_lock;
   1342
o  1424
o  1425   PROCEDURE [INLINE] tmp$set_lock (VAR lock: tmt$ptl_lock);
o  1426
o  1427     VAR
o  1428       b: boolean,
o  1429       bc: integer;
o  1430
o  1431     IF osv$cpus_logically_on > 1 THEN
o  1432       bc := #read_register (osc$pr_base_constant);
o  1433       IF lock.id <> bc  THEN
o  1434         REPEAT
o  1435           #TEST_SET (lock.locked, b);
o  1436         UNTIL NOT b;
o  1437         lock.id := bc;
o  1438       ELSE
o  1439         lock.count := lock.count + 1;
o  1440       IFEND;
o  1441     IFEND;
```



```
o  1442
o  1443 PROCEND tmp$set_lock;
o  1444
   1447
   1448 {  Global variables referenced by this module.
   1449
   1450
   1451   VAR
   1452     jmv$null_ijl_ordinal: [XREF] jmt$ijl_ordinal;
   1453
o  1456
o  1457 VAR
o  1458     jsv$ijl_serial_lock: [XREF] tmt$ptl_lock;
o  1459
   1462  VAR
   1463     jsv$ijl_swap_queue_list: [XREF] jst$ijl_swap_queue_list;
   1464
o  1482 {The following variable contains a count of the number of page frames that can be reassigned to be
o  1483 {used for another purpose. The count represents the number of pages that are in the free + available
o  1484 {queues. The count is broken into two parts - pages with no IO active, and pages with IO active.
o  1485
o  1486   VAR
o  1487     mmv$reassignable_page_frames: [XREF] mmt$reassignable_page_frames;
o  1497
o  1498 {  PURPOSE:      procedure mtp$error_stop
o  1499 {    Prefixes 'ERR=VEOS1000-' to the string and calls mtp$step_unstep_system to write string and step system}
o  1500
o  1501   PROCEDURE [XREF] mtp$error_stop (text: string(*<=63) );
o  1502
o  1503
o  1504 {  Global variables defined by this module.
o  1505
o  1506   VAR
o  1507
o  1508 {  Maximum time a job will stay swapped without initiating swapout IO after it has
o  1509 {  exceeded its think time.
o  1510
o  1511     jsv$max_time_swap_io_not_init: [XDCL, #GATE] integer := 100000000,
o  1512
o  1513 {  Maximum time a job will stay swapped with swap IO completed and memory resident
o  1514 {  after it exceeds its think time.
o  1515
o  1516     jsv$max_time_swap_io_complete: [XDCL, #GATE] integer := 200000000,
o  1517
o  1518 {  After exceeding think time by this much, a job becomes a good candidate to be
o  1519 {  removed from memory if memory is low.
o  1520
o  1521     jsv$think_expiration_time: [XDCL, #GATE] integer := 15000000;
o  1522
```

SELECT_BEST_CANDIDATE

```
 0  1524
 0  1525   PROCEDURE select_best_candidate
 A  1526     (    first_ijl_queue_link: jmt$ijl_ordinal;
 A  1527      VAR selected_jobs_ijl_ordinal: jmt$ijl_ordinal;
 A  1528      VAR selected_jobs_ijle_p: ^jmt$initiated_job_list_entry);
 A  1529
 A  1530 {
 A  1531 {     The purpose of this procedure is to select a job from the swap queue initiated by the
 A  1532 {   specified ijl queue link.  The job is selected that most exceeds think time or has the
 A  1533 {   longest estimated ready time.
 A  1534 {
 A  1535
 A  1536     VAR
 A  1537       ijle_p: ^jmt$initiated_job_list_entry,
 A  1538       ijl_ordinal: jmt$ijl_ordinal,
 A  1539       max_ijle_p: ^jmt$initiated_job_list_entry,
 A  1540       max_ijlo: jmt$ijl_ordinal,
 A  1541       min_ijle_p: ^jmt$initiated_job_list_entry,
 A  1542       min_ijlo: jmt$ijl_ordinal,
 A  1543       max_estimated_ready_time: integer,
 A  1544       min_estimated_ready_time: integer;
 A  1545
 A  1546     ijl_ordinal := first_ijl_queue_link;
 A  1547     max_estimated_ready_time := 0;
 A  1548     min_estimated_ready_time := 0fffffffffff(16);
 A  1549
 A  1550     REPEAT
12  1551       jmp$get_ijle_p (ijl_ordinal, ijle_p);
12  1552       IF ijle_p^.estimated_ready_time > max_estimated_ready_time THEN
3E  1553         max_estimated_ready_time := ijle_p^.estimated_ready_time;
3E  1554         max_ijlo := ijl_ordinal;
3E  1555         max_ijle_p := ijle_p;
4C  1556       IFEND;
4C  1557       IF ijle_p^.estimated_ready_time < min_estimated_ready_time THEN
58  1558         min_estimated_ready_time := ijle_p^.estimated_ready_time;
58  1559         min_ijlo := ijl_ordinal;
58  1560         min_ijle_p := ijle_p;
66  1561       IFEND;
66  1562
66  1563       ijl_ordinal := ijle_p^.swap_queue_link.forward_link;
66  1564     UNTIL ijl_ordinal = jmv$null_ijl_ordinal;
7E  1565
7E  1566     IF (#FREE_RUNNING_CLOCK (0) - min_estimated_ready_time) > jsv$think_expiration_time THEN
92  1567       selected_jobs_ijl_ordinal := min_ijlo;
92  1568       selected_jobs_ijle_p := min_ijle_p;
A2  1569     ELSE
A2  1570       selected_jobs_ijl_ordinal := max_ijlo;
A2  1571       selected_jobs_ijle_p := max_ijle_p;
AE  1572     IFEND;
AE  1573
AE  1574   PROCEND select_best_candidate;
```


JSP$ADV_EXPIRED_SWAPPED_JOBS

```
 0  1576
 0  1577   PROCEDURE [XDCL] jsp$adv_expired_swapped_jobs
 0  1578     (    swap_queue_id: jst$swapped_but_still_in_memory);
 0  1579
 0  1580 {
 0  1581 {     The purpose of this procedure is to advance jobs that are swapped but still in memory
 0  1582 {   that have exceeded the maximum time that can be spent in the respective swap queue.
 0  1583 {
 0  1584
 0  1585     VAR
 0  1586       current_time: integer,
 0  1587       ijl_ordinal: jmt$ijl_ordinal,
 0  1588       ijle_p: ^jmt$initiated_job_list_entry,
 0  1589       next_ijl_ordinal: jmt$ijl_ordinal,
 0  1590       time_limit: integer;
 0  1591
 0  1592     current_time := #FREE_RUNNING_CLOCK (0);
 C  1593     IF swap_queue_id = jsc$isqi_swapped_io_not_init THEN
16  1594       time_limit := jsv$max_time_swap_io_not_init;
22  1595     ELSEIF swap_queue_id = jsc$isqi_swapped_io_completed THEN
28  1596       time_limit := jsv$max_time_swap_io_complete;
36  1597     ELSE
36  1598
36  1599 {  Unexpected swap queue identifier.
36  1600
36  1601       RETURN;
38  1602     IFEND;
38  1603
38  1604 {  There is the possibility of a timing problem here.  If task switch decides
38  1605 {  to swap in the job that is saved as next_ijl_ordinal, we would cross over
38  1606 {  to the swapping queue.  Therefore a check is made to see if we are still
38  1607 {  in the right queue.  If not, just exit and try again next time (called by
38  1608 {  periodic).
38  1609
38  1610      ijl_ordinal := jsv$ijl_swap_queue_list [swap_queue_id].forward_link;
38  1611
38  1612    /loop/
38  1613     WHILE ijl_ordinal <> jmv$null_ijl_ordinal DO
5C  1614       jmp$get_ijle_p (ijl_ordinal, ijle_p);
5C  1615       IF ijle_p^.swap_queue_link.queue_id = swap_queue_id THEN
8C  1616         next_ijl_ordinal := ijle_p^.swap_queue_link.forward_link;
8C  1617         IF (ijle_p^.estimated_ready_time + time_limit) < current_time THEN
9A  1618           jsp$monitor_advance_swap (ijl_ordinal);
AA  1619         IFEND;
AE  1620       ELSE
AE  1621         EXIT /loop/;
B0  1622       IFEND;
B0  1623       ijl_ordinal := next_ijl_ordinal;
B0  1624     WHILEND /loop/;
C8  1625
C8  1626   PROCEND jsp$adv_expired_swapped_jobs;
```

JMP$INITIATE_SWAPOUT_IO

```
        0  1628
        0  1629    PROCEDURE [XDCL] jsp$initiate_swapout_io
        0  1630      (     pages_needed: mmt$page_frame_index);
        0  1631
        0  1632 (                 .
        0  1633 (       The purpose of this procedure is to initiate the swapout IO on swapped jobs
        0  1634 (   that have not had IO initiated to make memory available.  Jobs are advanced
        0  1635 (   until IO is initiated on enough pager to bring mmv$reassignable_page_frames.now + .soon
        0  1636 (   up to the number of pages requested or until all jobs have had swapout IO initiated.
        0  1637 (
        0  1638
        0  1639      VAR
        0  1640        ijle_p: ^jmt$initiated_job_list_entry,
        0  1641        ijl_ordinal: jmt$ijl_ordinal,
        0  1642        pages_io_initiated_on: integer,
        0  1643        pages_to_initiate_io_on: integer,
        0  1644        selected_jobs_ijle_p: ^jmt$initiated_job_list_entry,
        0  1645        selected_jobs_ijl_ordinal: jmt$ijl_ordinal;
        0  1646
        0  1647      #keypoint (osk$entry, 0, jsk$initiate_swapout_io);
        8  1648
        8  1649      pages_io_initiated_on := 0;
        8  1650      pages_to_initiate_io_on := pages_needed - (mmv$reassignable_page_frames.now +
        8  1651            mmv$reassignable_page_frames.soon);
        8  1652
        8  1653      WHILE pages_to_initiate_io_on > pages_io_initiated_on DO
       26  1654        tmp$set_lock (jsv$ijl_serial_lock);
       5E  1655        ijl_ordinal := jsv$ijl_swap_queue_list [jsc$isqi_swapped_io_not_init].forward_link;
       5E  1656
       5E  1657        IF ijl_ordinal = jmv$null_ijl_ordinal THEN
       76  1658          tmp$clear_lock (jsv$ijl_serial_lock);
       AE  1659          #keypoint (osk$exit, 0, jsk$initiate_swapout_io);
       B2  1660          RETURN;
       B4  1661        IFEND;
       B4  1662
       B4  1663        select_best_candidate (ijl_ordinal, selected_jobs_ijl_ordinal, selected_jobs_ijle_p);
       CA  1664        tmp$clear_lock (jsv$ijl_serial_lock);
      102  1665
      102  1666        jsp$monitor_advance_swap (selected_jobs_ijl_ordinal);
      116  1667        pages_io_initiated_on := pages_io_initiated_on + selected_jobs_ijle_p^.swap_data.swapped_job_page_count;
      116  1668      WHILEND;
      124  1669
      124  1670      #keypoint (osk$exit, 0, jsk$initiate_swapout_io);
      128  1671
      128  1672    PROCEND jsp$initiate_swapout_io;
```


JSP$FREE_SWAP_RESIDENT_JOB

```
        0  1674
        0  1675    PROCEDURE [XDCL] jsp$free_swapped_jobs_memory (
        0  1676          ijl_ordinal: jmt$ijl_ordinal);
        0  1677
        0  1678 (   The main purpose of this procedure is to select a swap-resident job (swapout IO has completed)
        0  1679 (   and free its memory.  The job selected is the one that most exceeds think time or has
        0  1680 (   the longest estimated ready time.  A null ijl ordinal passed into this procedure indicates
        0  1681 (   that memory manager needs memory from ANY swap resident job.  A job is selected from the
        0  1682 (   swapped_io_completed queue; if no jobs can be found in that queue, it is because all S2
        0  1683 (   jobs have been readied by dispatcher on another processor.  The S2 job has been relinked to
        0  1684 (   the swapping queue to swap in when swapper next executes.  The memory is still available to
        0  1685 (   be freed; the swapping queue must be searched to find the S2 job.
        0  1686 (   A second use of the procedure is to free the memory of a specific swap resident job so the the
        0  1687 (   memory used by the job can be given to a task needing contiguous memory.  Memory manager selects
        0  1688 (   the job to be freed and passes in the job's ijl ordinal.
        0  1689
        0  1690      VAR
        0  1691        swap_resident_ijle_p: ^jmt$initiated_job_list_entry,
        0  1692        swap_resident_ijlo: jmt$ijl_ordinal,
        0  1693        swap_resident_q_head: jmt$ijl_ordinal;
        0  1694
        0  1695      #keypoint (osk$entry, 0, jsk$free_swapped_jobs_memory);
        8  1696
        8  1697      IF ijl_ordinal = jmv$null_ijl_ordinal THEN
       18  1698        tmp$set_lock (jsv$ijl_serial_lock);
       52  1699
       52  1700        swap_resident_q_head := jsv$ijl_swap_queue_list [jsc$isqi_swapped_io_completed].forward_link;
       52  1701        IF swap_resident_q_head <> jmv$null_ijl_ordinal THEN
       6A  1702          select_best_candidate (swap_resident_q_head, swap_resident_ijlo, swap_resident_ijle_p);
       82  1703        ELSE
       82  1704          swap_resident_ijlo := jsv$ijl_swap_queue_list [jsc$isqi_swapping].forward_link;
       82  1705        /find_swap_resident_job/
       82  1706          WHILE swap_resident_ijlo <> jmv$null_ijl_ordinal DO
       9A  1707            jmp$get_ijle_p (swap_resident_ijlo, swap_resident_ijle_p);
       9A  1708            IF swap_resident_ijle_p^.swap_status = jmc$iss_swapped_io_complete THEN
       CC  1709              EXIT /find_swap_resident_job/;
       D0  1710            IFEND;
       D0  1711            swap_resident_ijlo := swap_resident_ijle_p^.swap_queue_link.forward_link;
       D0  1712          WHILEND /find_swap_resident_job/;
       E4  1713        IFEND;
       E4  1714
       E4  1715        tmp$clear_lock (jsv$ijl_serial_lock);
      11E  1716
      11E  1717        IF swap_resident_ijlo <> jmv$null_ijl_ordinal THEN
      12E  1718          jsp$free_swap_resident_job (swap_resident_ijlo, swap_resident_ijle_p);
      14A  1719        ELSE
      14A  1720          mtp$error_stop ('JS-could not find swap resident job to free');
      16A  1721        IFEND;
      16E  1722
      16E  1723      ELSE
      16E  1724        jsp$monitor_advance_swap (ijl_ordinal);
      17E  1725      IFEND;
      17E  1726
      17E  1727      #keypoint (osk$exit, 0, jsk$free_swapped_jobs_memory);
      182  1728
      182  1729    PROCEND jsp$free_swapped_jobs_memory;
```

JSP$FREE_SWAP_RESIDENT_JOB

```
      o  1730 MODEND jsm$swapped_job_manager;
```

**** I=$05578173AS0102D19890821T183254 L=ZZXXLIST B=LGO DA=NONE LO=R RC=NONE OPT=SCHED EL=F LF=CS612 PAD=0

**** NO DIAGNOSTICS

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
b	1428	1435	1436						
b	1629	1654	1654						
b	1675	1698	1698						
bc	1429	1432/M	1433	1437					
bc	1629	1654/M	1654	1654					
bc	1675	1698/M	1698	1698					
block_index	23	1305/S	1551/S	1614/S	1707/S				
block_number	22	1305/S	1551/S	1614/S	1707/S				
block_p	41	1305	1551	1614	1707				
clear	1368	1337/M	1658/M	1664/M	1715/M				
count	1365	1334	1335/M	1335	1439/M	1439	1654/M	1654	1658
		1658/M	1658	1664	1664/M	1664	1698/M	1698	1715
		1715/M	1715						
current_time	1586	1592/M	1617						
dfc$command_record_bytes	175	183							
dfc$division_overwrite_words	162	190							
dfc$esm_command_record_size	183	191							
dfc$esm_header_record_size	184	191							
dfc$esm_maintenance_buf_size	163	194							
dfc$esm_memory_base_shift	169	191	192	192					
dfc$header_record_bytes	174	184							
dfc$max_esm_memory_size	164	193							
dfc$max_number_of_mainframes	171	156							
dfc$min_data_record_bytes	179	190							
dfc$min_esm_division_size	189	193							
dft$mainframe_set	156	106	107	283	284				
dmt$system_file_id	205	138							
estimated_ready_time	95	1552	1553	1557	1558	1617			
find_swap_resident_job	1705	1705	1709	1712					
first_ijl_queue_link	1526	1546							
forward_link	953	1563	1616	1711					
forward_link	1473	1610	1655	1700	1704				
gft$file_descriptor_index	220	210							
gft$system_file_identifier	209	205	1022						
gft$table_residence	223	211							
i#program_error	1345	1332	1658	1664	1715				
id	1366	1331	1433	1437/M	1654	1654/M	1658	1664	1698
		1698/M	1715						
ijl_ordinal	1301	1305/S	1305/S						
ijl_ordinal	1525	1551/S	1551/S						
ijl_ordinal	1538	1546/M	1551/P	1554	1559	1563/M	1564		
ijl_ordinal	1577	1614/S	1614/S						
ijl_ordinal	1587	1610/M	1613	1613	1614/P	1618/P	1623/M		
ijl_ordinal	1641	1655/M	1657	1663/P					
ijl_ordinal	1675	1707/S	1707/S						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	1676	1697	1724/P						
ijle_p	1302	1305/M							
ijle_p	1525	1551/M							
ijle_p	1537	1551/P	1552	1553	1555	1557	1558	1560	1563
ijle_p	1577	1614/M							
ijle_p	1588	1614/P	1615	1616	1617				
ijle_p	1675	1707/M							
index_p	54	1305	1551	1614	1707				
iot$io_error	1108	139	1062						
jmc$highest_service_accumulator	680	681							
jmc$ies_job_swapped	460	469							
jmc$ies_swapin_in_progress	459	468							
jmc$iss_idle_tasks_initiated	475	502							
jmc$iss_swapin_io_complete	500	503							
jmc$iss_swapin_requested	496	503							
jmc$iss_swapout_complete	495	502							
jmc$iss_swapped_io_cannot_init	486	513							
jmc$iss_swapped_io_complete	491	1708							
jmc$iss_swapped_no_io	477	512							
jmc$kjl_maximum_entries	244	237	238	632					
jmc$kol_maximum_entries	254	239							
jmc$max_ajl_ord	236	229	235						
jmc$max_dispatching_control	410	414							
jmc$max_dispatching_priority	332	292	295	296					
jmc$max_ijl_entries	30	1474							
jmc$max_ijl_index_count	31	52							
jmc$maximum_job_classes	610	613							
jmc$maximum_job_count	251	244							
jmc$maximum_output_count	261	254							
jmc$maximum_service_classes	713	716							
jmc$min_dispatching_control	409	413							
jmc$null_service_class	706	707							
jmc$priority_p1	346	293							
jmc$priority_p10	355	294							
jmc$priority_p14	359	294							
jmc$priority_p8	353	293							
jmc$reserved_ajls	240	235							
jmc$service_accumulator_maximum	672	669							
jmc$system_default_offset	696	697							
jmc$system_supplied_name_size	937	934							
jmc$unlimited_offset	693	682							
jmp$get_ijle_p	1301	1307	1551	1614	1707				
jmt$ajl_ordinal	229	75							
jmt$delayed_swapin_work	276	105	280						
jmt$dispatching_control_index	413	370	380						
jmt$dispatching_controls	383	381							
jmt$dispatching_priority	292	87	371	372	373	385			
jmt$ijl_block_index	27	23	54						
jmt$ijl_block_number	26	22	42	43					
jmt$ijl_dispatching_control	369	88							
jmt$ijl_entry_status	455	74							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$ijl_ordinal	21	94	122	952	953	1015	1054	1301	1313
		1315	1452	1472	1473	1526	1527	1538	1540
		1542	1587	1589	1641	1645	1676	1692	1693
jmt$ijl_p	40	35							
jmt$ijl_page_fault_count	529	524	525	526					
jmt$ijl_page_stats	523	519							
jmt$ijl_service_class_stats	517	109							
jmt$ijl_statistics	562	108							
jmt$ijl_swap_count	538	534	535						
jmt$ijl_swap_counts	533	128	520						
jmt$ijl_swap_status	473	77	78	79					
jmt$initiated_job_list_block	51	57							
jmt$initiated_job_list_entry	71	54	979	1302	1314	1528	1537	1539	1541
		1588	1640	1644	1691				
jmt$initiated_job_list_p	57	41							
jmt$input_file_location	652	647							
jmt$job_abort_disposition	661	645							
jmt$job_class	613	133							
jmt$job_mode	616	90							
jmt$job_priority	621	130	131						
jmt$job_recovery_disposition	664	646							
jmt$kjl_index	632	76							
jmt$queue_file_ijl_information	644	115							
jmt$scheduling_data	121	99							
jmt$service_accumulator	669	123	124	125					
jmt$service_class_index	716	134							
jmt$swap_data	137	101							
jmt$swapout_reasons	719	129							
jmt$swapped_job_entry	734	146	980						
jmt$system_supplied_name	934	72							
jmt$task_time_slice	423	403	404						
jmt$time_slice_values	402	387							
jmv$ijl_p	35	1305	1551	1614	1707				
jmv$null_ijl_ordinal	1452	1564	1613	1613	1657	1697	1701	1706	1706
		1717							
jsc$isqi_swapped_io_completed	957	959	1595	1700/S					
jsc$isqi_swapped_io_not_init	956	959	1593	1655/S					
jsc$isqi_swapping	956	1704/S							
jsk$base	1230	1157	1161	1165	1169	1173	1177	1181	1185
		1189							
jsk$free_swapped_jobs_memory	1185	1695	1727						
jsk$initiate_swapout_io	1181	1647	1659	1670					
jsp$adv_expired_swapped_jobs	1577	1626							
jsp$free_swap_resident_job	1312	1718							
jsp$free_swapped_jobs_memory	1675	1729							
jsp$initiate_swapout_io	1629	1672							
jsp$monitor_advance_swap	1315	1618	1666	1724					
jst$changed_asid_entry	1002	993							
jst$ijl_swap_queue_id	956	951	1479						
jst$ijl_swap_queue_link	950	83							
jst$ijl_swap_queue_list	1479	1463							
jst$ijl_swap_queue_list_entry	1471	1479							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jst$io_control_information	964	102							
jst$swap_file_descriptor	978	103							
jst$swapped_but_still_in_memory	959	1578							
jst$swapped_page_descriptor	987	985							
jst$swapped_page_descriptors	984	981							
jsv$ijl_serial_lock	1458	1654/P	1658/P	1664/P	1698/P	1715/P			
jsv$ijl_swap_queue_list	1463	1610	1655	1700	1704				
jsv$max_time_swap_io_complete	1516	1596							
jsv$max_time_swap_io_not_init	1511	1594							
jsv$think_expiration_time	1521	1566							
lock	1328	1331	1334	1335/M	1335	1337/M			
lock	1425	1433	1435	1437/M	1439/M	1439			
lock	1629	1654	1654	1654/M	1654/M	1654			
lock	1629	1658	1658	1658/M	1658	1658/M	1664	1664	1664/M
		1664	1664/M						
lock	1675	1698	1698	1698/M	1698/M	1698			
lock	1675	1715	1715	1715/M	1715	1715/M			
locked	1364	1435	1654	1698					
loop	1612	1612	1621	1624					
max_estimated_ready_time	1543	1547/M	1552	1553/M					
max_ijle_p	1539	1555/M	1571						
max_ijlo	1540	1554/M	1570						
min_estimated_ready_time	1544	1548/M	1557	1558/M	1566				
min_ijle_p	1541	1560/M	1568						
min_ijlo	1542	1559/M	1567						
mmc$pq_avail	751	797							
mmc$pq_free	750	809							
mmc$pq_job_fixed	791	798	810						
mmc$pq_job_working_set	793	810	811						
mmc$pq_shared_first_site	801	805							
mmc$pq_shared_num_sites	802	805							
mmc$pq_shared_other	760	800							
mmc$pq_shared_site_01	762	801							
mmc$pq_shared_site_25	786	806							
mmc$pq_shared_task_service	755	799							
mmc$pq_swapped_io_error	789	809							
mmc$pq_wired	753	796							
mmt$active_segment_table_entry	1012	990	1028	1061					
mmt$ast_index	1044	145	1005						
mmt$global_page_queue_index	809	1146							
mmt$global_page_queue_list_ent	1136	1146							
mmt$job_page_queue_index	810	736	1147						
mmt$job_page_queue_list	1147	100	1320						
mmt$link	1035	1013	1051	1052	1133				
mmt$locked_page	1073	1057							
mmt$memory_reserve_request	1114	93							
mmt$page_age	1080	1060	1084	1084					
mmt$page_frame_index	973	965	967	968	969	1037	1037	1116	1117
		1630							
mmt$page_frame_queue_id	811	966	1021	1055					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$page_frame_table_entry	1050	988	1066						
mmt$page_queue_list_entry	1132	1137	1147						
mmt$reassignable_page_frames	1490	1487							
mmv$reassignable_page_frames	1487	1650	1651						
mtp$error_stop	1501	1720							
next_ijl_ordinal	1589	1616/M	1623						
now	1491	1650							
osc$free_running_clock_maximum	445	442							
osc$invalid_ring	866	906							
osc$max_number_of_processors	1353	1348							
osc$max_page_frames	816	140	141	735	737	973	1014	1134	1140
		1491	1492	1493	1494				
osc$max_page_table_entries	817	820							
osc$max_ring	865	906	907						
osc$max_segment_length	889	912							
osc$max_tasks	1106	1103							
osc$maximum_offset	888	889	909	909	910				
osc$maximum_processors	1357	1353							
osc$maximum_segment	887	908							
osc$min_ring	864	907							
osc$pr_base_constant	1387	1331	1432	1654	1658	1664	1698	1715	
osc$task_time_slice_maximum	434	437							
osk$entry	1254	1647	1695						
osk$exit	1255	1659	1670	1727					
osk$system_class	1268	1252	1253	1254	1255	1256	1257	1258	
ost$asid	852	84	848	992	1003	1004	1019		
ost$cp_time	550	518	563						
ost$cp_time_value	548	126	551	552					
ost$free_running_clock	442	95	96	97	98	132	142	143	144
		374	386	1018					
ost$global_task_id	1097	89	118						
ost$key_lock_value	901	898							
ost$page_id	822	832							
ost$page_table_entry	827	836	989						
ost$page_table_index	820	836	1058						
ost$paging_statistics	586	564							
ost$ring	906	918							
ost$segment	908	919							
ost$segment_offset	909	849	920						
ost$system_virtual_address	847	1063							
ost$task_index	1103	1091	1092	1098					
ost$task_time_slice	437	423							
osv$cpus_logically_on	1348	1330	1431	1654	1658	1664	1698	1715	
pages_io_initiated_on	1642	1649/M	1653	1653	1667/M	1667			
pages_needed	1630	1650							
pages_to_initiate_io_on	1643	1650/M	1653	1653					
queue_id	951	1615							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

JSP$FREE_SWAP_RESIDENT_JOB

IDENTIFIER	DEFINED ON LINE	REFERENCES							
select_best_candidate	1525	1574	1663	1702					
selected_jobs_ijl_ordinal	1527	1567/M	1570/M						
selected_jobs_ijl_ordinal	1645	1663/P	1666/P						
selected_jobs_ijle_p	1528	1568/M	1571/M						
selected_jobs_ijle_p	1644	1663/P	1667						
sft$counter	596	565	566						
soon	1492	1651							
swap_data	101	1667							
swap_queue_id	1578	1593	1595	1610/S	1615				
swap_queue_link	83	1563	1615	1616	1711				
swap_resident_ijle_p	1691	1702/P	1707/P	1708	1711	1718/P			
swap_resident_ijlo	1692	1702/P	1704/M	1706	1706	1707/P	1711/M	1717	1718/P
swap_resident_q_head	1693	1700/M	1701	1702/P					
swap_status	77	1708							
swapped_job_page_count	140	1667							
time_limit	1590	1594/M	1596/M	1617					
tmp$clear_lock	1328	1341	1658	1664	1715				
tmp$set_lock	1425	1443	1654	1698					
tmt$pt1_lock	1361	1328	1425	1458					
tmt$task_queue_link	1090	1059							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Memory Management: Manage Page Table and ASIDs

```
  3 MODULE mmm$asid_page_table_manager;
  4
  5 { PURPOSE
  6 {   This module contains the memory modules for managing the page table and ASIDs.
  7 {   There are three functional groups of procedures in this module. The modules are grouped
  8 {   here since they are related:
  9 {
 10 {        ASID MANAGEMENT          - mmp$assign_asid
 11 {                                   mmp$assign_specific_asid
 12 {                                   mmp$free_asid
 13 {                                   mmp$change_asid
 14 {                                   mmp$reclaim_ast_entries
 15 {
 16 {        PAGE TABLE MANAGEMENT    - mmp$make_pt_entry
 17 {                                   mmp$delete_pt_entry
 18 {                                   clear_continue_bits
 19 {                                   free_pt_entry_in_avail_queue
 20 {
 21 {        PAGE TABLE FULL MANAGER - mmp$process_page_table_full
 22 {                                   build_asid_list
 23 {                                   reassign_asid
 24 {
 25 {
```

Global Declarations Referenced by This Module

```
o   841
o   842   PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
o   843         ijle_p: ^jmt$initiated_job_list_entry;
o   844     VAR fde_p: gft$locked_file_desc_entry_p);
o   845
o   846
o  2061   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
o  2062     VAR ijle_p: ^jmt$initiated_job_list_entry);
o  2063
   2071
   2072   FUNCTION [INLINE] jmp$ijl_block_valid
   2073     (     ijl_ordinal: jmt$ijl_ordinal): boolean;
   2074
   2075     jmp$ijl_block_valid := NOT (jmv$ijl_p.block_p^ [ijl_ordinal.block_number].index_p = NIL);
   2076
   2077   FUNCEND jmp$ijl_block_valid;
   2078
   2079   PROCEDURE [INLINE] jmp$unlock_ajl
   2080     (     ijle_p: ^jmt$initiated_job_list_entry);
   2081
   2082     VAR
   2083       ajlo: jmt$ajl_ordinal;
   2084
   2085     tmp$set_lock (tmv$ptl_lock);
   2086     ajlo := ijle_p^.ajl_ordinal;
   2087     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
   2088       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
   2089     ELSE
   2090       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
   2091     IFEND;
   2092     tmp$clear_lock (tmv$ptl_lock);
   2093
   2094   PROCEND jmp$unlock_ajl;
   2095
o  2208
o  2209   PROCEDURE [XREF] jsp$recalculate_swapped_pages
o  2210     (    ijle_p: ^jmt$initiated_job_list_entry;
o  2211          pages_removed: mmt$page_frame_index);
o  2212
   2215
   2216   PROCEDURE [XREF] mmp$asid (asti: mmt$ast_index;
   2217     VAR asid: ost$asid);
   2218
o  2221   PROCEDURE [XREF] mmp$asti (asid: ost$asid;
o  2222     VAR asti: mmt$ast_index);
o  2223
   2226
   2227   PROCEDURE [XREF] mmp$aste_pointer (asid: ost$asid;
   2228     VAR aste_p: ^mmt$active_segment_table_entry);
   2229
o  2232
o  2233   PROCEDURE [INLINE] mmp$delete_last_pfti_from_array;
   2239
   2240   PROCEDURE [INLINE] mmp$find_next_pfti
   2241     (VAR xpfti: mmt$page_frame_index);
   2242
```


Global Declarations Referenced by This Module

```
o  2260
o  2261
o  2262   PROCEDURE [XREF] mmp$free_image_pages_mtr;
o *2263
o  2264
o  2265    PROCEDURE [INLINE] mmp$get_inhibit_io_status
o  2266      (     ijl_ordinal: jmt$ijl_ordinal;
o  2267            lock: boolean;
o  2268       VAR inhibit_io: boolean;
o  2269       VAR ijle_p: ^jmt$initiated_job_list_entry);
o  2270
o  2271     VAR
o  2272       ajlo: jmt$ajl_ordinal;
o  2273
o  2274      jmp$get_ijle_p (ijl_ordinal, ijle_p);
o  2275      inhibit_io := (ijle_p^.swap_status > jmc$inhibit_memory_manager_io);
o  2276      IF NOT inhibit_io THEN
o  2277        IF lock THEN
o  2278          tmp$set_lock (tmv$ptl_lock);
o  2279          jmp$lock_ajl_with_lock (ijle_p, ijl_ordinal, ajlo);
o  2280          tmp$clear_lock (tmv$ptl_lock);
o  2281        IFEND;
o  2282      IFEND;
o  2283
o  2284    PROCEND mmp$get_inhibit_io_status;
o  2285
   2326
   2327    PROCEDURE [INLINE] mmp$get_max_sdt_pointer
   2328      (     xcb_p: ^ost$execution_control_block;
   2329       VAR sdt_p: mmt$max_sdt_p);
   2330
o  3461
o  3462    FUNCTION [INLINE] mmp$get_sdt_entry_p
o  3463      (     xcb_p: ^ost$execution_control_block;
o  3464            segnum: ost$segment): ^mmt$segment_descriptor;
o  3465
o  3466      mmp$get_sdt_entry_p := #address (1, #segment (xcb_p),
o  3467        8 * segnum + xcb_p^.sdt_offset);
o  3468
o  3469    FUNCEND;
o  3470
   3473   PROCEDURE [XREF] mmp$initialize_find_next_pfti (xsva: ost$system_virtual_address;
   3474         length: ost$segment_length;
   3475         end_point_option: (include_partial_pages, exclude_partial_pages);
   3476         page_selection_criteria: mmt$page_selection_criteria;
   3477         aste_p: ^mmt$active_segment_table_entry;
   3478    VAR xpfti: mmt$page_frame_index);
   3479
o  3487
o  3488    PROCEDURE [INLINE] mmp$nudge_periodic_call;
o  3489
o  3490      mmv$time_to_call_mem_mgr := 0;
o  3491      osv$time_to_check_asyn := 0;
o  3492
o  3493    PROCEND mmp$nudge_periodic_call;
o  3494
```

Global Declarations Referenced by This Module

```
3504
3505    PROCEDURE [XREF] mmp$relink_page_frame (pfti: mmt$page_frame_index;
3506          queue_id: mmt$page_frame_queue_id);
3507
o  3510
o  3511    PROCEDURE [XREF] mmp$remove_page_from_jws (pfti: mmt$page_frame_index;
o  3512          ijle_p: ^jmt$initiated_job_list_entry;
o  3513      VAR mcount: integer;
o  3514      VAR rcount: integer);
o  3515
3518
3519    PROCEDURE [INLINE] mmp$reset_find_next_pfti
3520      (VAR xpfti: mmt$page_frame_index);
3521
o  3544
o  3545    PROCEDURE [XREF] tmp$find_next_xcb (search: tmt$fnx_search_type;
o  3546          ijle_p: ^jmt$initiated_job_list_entry;
o  3547          ijl_ordinal: jmt$ijl_ordinal;
o  3548      VAR state: tmt$find_next_xcb_state;
o  3549      VAR xcb_p: ^ost$execution_control_block);
o  3550
3566
3567
3568
3569    VAR
3570      dsv$ssr_sdte: [XREF] mmt$segment_descriptor;
3571
o  3574 {Monitor segment table.}
o  3575
o  3576    VAR
o  3577      mtv$monitor_segment_table: [XREF] record
o  3578        st: ALIGNED [0 MOD 8] array [0 .. 4095] of mmt$segment_descriptor,
o  3579      recend;
o  3580


3584 {NOS segment table.
3585
3586    VAR
3587      mtv$nos_segment_table_p: [XREF] ^ RECORD
3588        st: ALIGNED [0 MOD 8] array [0 .. *] OF mmt$segment_descriptor,
3589      RECEND;


o  3593
o  3594 VAR
o  3595    jmv$max_ajl_ordinal_in_use: [XREF] jmt$ajl_ordinal;
o  3596
o  3597
3600 {Define pointer to Initiated Job List (IJL).
3601
3602    VAR
3603      jmv$ijl_p: [XREF]  jmt$ijl_p;
o  3631
o  3632    VAR
o  3633      jmv$null_ijl_ordinal: [XREF] jmt$ijl_ordinal;
```


Global Declarations Referenced by This Module

```
o  3634


3638 {Define async worklist for processing periodic activities.
3639
3640    VAR
3641      mmv$async_work: [XREF] mmt$async_work_list;
3642


o  3655 {Pointer to the Active Segment Table - (AST).}
o  3656
o  3657    VAR
o  3658      mmv$ast_p: [XREF] ^mmt$active_segment_table;
o  3659


3663
3664    VAR
3665      mmv$image_file: [XREF] mmt$image_file;
3666


o  3670 {The following variable indicates if the configuration consists of multiple
o  3671 {caches that are not hardware connected for unified cache purging - ie, if a cache
o  3672 {purge is required each processor must purge its own cache.
o  3673
o  3674    VAR
o  3675      mmv$multiple_caches: [XREF] boolean;
o  3676


o  3678 {The following variable indicates if the configuration consists of multiple
o  3679 {page MAPS that are not hardware connected for unified map purging - ie,
o  3680 {if a page map purge is required each processor must purge its own map.
o  3681
o  3682    VAR
o  3683      mmv$multiple_page_maps: [XREF] boolean;
o  3684


o  3686 {Define pointer to array for holding PFTI lists. This array is used in monitor for holding lists
o  3687 {PFTIs of pages belonging to a segment.
o  3688
o  3689    VAR
o  3690      mmv$pfti_array_p: [XREF] ^mmt$pfti_array;
o  3691


o  3695 {Pointer to the 'PAGE FRAME TABLE' (PFT)
o  3696
o  3697    VAR
o  3698      mmv$pft_p: [XREF] ^mmt$page_frame_table;
o  3699
```

Global Declarations Referenced by This Module

```
3703 {Define page table length in words.
3704
3705   VAR
3706     mmv$pt_length: [XREF] integer;
3707


3709 {Pointer to the system PAGE TABLE (PT).
3710
3711   VAR
3712     mmv$pt_p: [XREF] ^ost$page_table;
3713


o 3717
o 3718   VAR
o 3719     mmv$test_reassign_asid: [XREF] boolean;
o 3720


o 3722 {System page size.}
o 3723
o 3724   VAR
o 3725     osv$page_size: [XREF] ost$page_size;
o 3726
```


Global Statistics Kept By Module

```
3730 { the following is used for debugging trace information:
3731
3732   PROCEDURE [INLINE] trace
3733     (     id: integer;
3734           inc: integer);
3735
3736     mmv$aptm_trace [id] := mmv$aptm_trace [id] + inc;
3737   PROCEND trace;
3738
3739   CONST
3740     mmc$ap_low_asids = 1,
3741     mmc$ap_no_asids = 2,
3742     mmc$ap_ast_reset = 3,
3743     mmc$ap_free_aste = 4,
3744     mmc$ap_assign = 5,
3745     mmc$ap_assign_specific = 6,
3746     mmc$ap_casid_swapped_job = 7,
3747     mmc$ap_casid_monitor = 8,
3748     mmc$ap_casid_template = 10,
3749     mmc$ap_casid_global = 11,
3750     mmc$ap_casid_job = 12,
3751     mmc$ap_reclaim_asids = 13,
3752     mmc$ap_mpte_full = 14,
3753     mmc$ap_mpte_recovered = 15,
3754     mmc$ap_mpte_rec1 = 16,
3755     mmc$ap_mpte_rec2 = 17,
3756     mmc$ap_ptf_called = 18,
3757     mmc$ap_ptf_tried = 19,
3758     mmc$ap_ptf_failed = 20,
3759     mmc$ap_ptf_remove = 21,
3760     mmc$ap_rea_called = 22,
3761     mmc$ap_rea_in_free = 23,
3762     mmc$ap_rea_mpte_fail = 24,
3763     mmc$ap_rea_ok = 25,
3764     mmc$ap_rea_ok1 = 26,
3765     mmc$ap_rea_ok2 = 27,
3766     mmc$ap_rea_ok3 = 28,
3767     mmc$ap_rea_ok4 = 29,
3768     mmc$ap_rea_fail = 30,
3769     mmc$ap_rea_fail1 = 31,
3770     mmc$ap_rea_fail2 = 32,
3771     mmc$ap_rea_quit = 33,
3772     mmc$ap_unused_34 = 34,
3773     mmc$ap_unused_35 = 35,
3774     mmc$ap_rea_make_pt_entry = 36,
3775     mmc$ap_ba_freed_terj = 37;
3776
3777 { The following table keeps statistics on page table full procesing.
3778
3779   TYPE
3780     mmt$pt_full_trace_info = record
3781       timestamp: integer,
3782       changed_asid: 0 .. 0ffffffff(16),
3783       failed: 0 .. 0ffffffff(16),
3784       pass: array [reassign_pass] of 0 .. 0ffffffff(16),
3785       last_sva: ost$system_virtual_address,
```

Global Statistics Kept By Module

```
3786        index: 0 .. 65535,
3787        asid: array [0 .. 127] of record
3788          old: ost$asid,
3789          new: ost$asid,
3790        recend,
3791      recend;
3792
3793    VAR
3794      mmv$pt_full_trace: [XDCL] mmt$pt_full_trace_info,
3795      mmv$aptm_trace: [XDCL] array [0 .. 50] of 0 .. 0ffffffff[16];
3796
```


Global Declarations Declared by This Module

```
3798
3799    VAR
3800      changing_asid: ost$asid := 0,
3801      changing_aste_p: ^mmt$active_segment_table_entry := NIL,
3802      mmv$continue_bit_count_p: [XDCL, #GATE] ^mmt$continue_bit_count,
3803      mmv$max_template_segment_number: [XDCL, #GATE] integer := mmc$first_loader_predefined_seg,
3804      mmv$mf_wired_asid: [XDCL, #GATE] mmt$mainframe_wired_asid := [0, 0],
3805      mmv$number_free_astes: [XDCL, #GATE] integer,
3806      mmv$pages_to_dump_p: [XDCL, #GATE] ^packed array [0 .. * ] of boolean := NIL,
3807      mmv$pt_search: integer [4 .. 32] := 32,
3808      mmv$time_changed_global_asid: [XDCL] ost$free_running_clock := 0ffffffffffff[16],
3809      mmv$time_changed_template_asid: [XDCL] ost$free_running_clock := 0ffffffffffff[16];
3810
```

Inline Procedures From Common Decks

```
          3813
          3814   PROCEDURE [INLINE] mmp$purge_all_cache_map;
          3815
          3816     VAR
          3817       null_sva: 0 .. 0ffffffffffff(16);
          3818
          3819     IF mmv$multiple_caches OR mmv$multiple_page_maps THEN
          3820       mmp$purge_all_cache_map_proc;
          3821     ELSE
          3822       #purge_buffer (osc$purge_all_cache, null_sva);
          3823       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
          3824     IFEND;
          3825
          3826   PROCEND;

        o 3832   PROCEDURE [INLINE] mmp$purge_all_page_seg_map;
        o 3833
        o 3834     VAR
        o 3835       null_sva: 0 .. 0ffffffffffff(16);
        o 3836
        o 3837     IF osv$cpus_logically_on > 1 THEN
        o 3838       mmp$purge_all_map_proc;
        o 3839     ELSE
        o 3840       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
        o 3841     IFEND;
        o 3842
        o 3843   PROCEND;
```


ASID MANAGEMENT
[XDCL] mmp$assign_asid

```
        o 3850 { PURPOSE:
        o 3851 {   This procedure is called to find an available entry in the AST table, assign the entry and
        o 3852 {   return an ASID for the entry. If no AST entry is available, the routine HALTS. This should never
        o 3853 {   happen since there are more ASIDs than page frames, and ASIDs not currently being used by any
        o 3854 {   page frames can be reassigned.
        o 3855 {
        o 3856 { DESIGN
        o 3857 {   This procedure searches the AST for an entry that is free and has been free since before
        o 3858 {   the last time CACHE and MAP were purged. If necessary, this routine will reclaim AST entries
        o 3859 {   assigned to segments/files if no pages are currently assigned to the segment
        o 3860
        o 3861   VAR
        o 3862     next_asti: [STATIC] mmt$ast_index := 1;
        o 3863
        o 3864
        o 3865   PROCEDURE [XDCL] mmp$assign_asid
        4 3866     (VAR asid: ost$asid;
        4 3867      VAR asti: mmt$ast_index;
        4 3868      VAR aste_p: ^mmt$active_segment_table_entry);
        4 3869
        4 3870     VAR
        4 3871       zaste_p: ^mmt$active_segment_table_entry,
        4 3872       last_purge_time: [STATIC] integer := 1;
        4 3873
        4 3874
        4 3875 { If there are NO free entries, call reclaim unused entries right now since an entry MUST be
        4 3876 { assigned. If the number of free entries is too small, force a call to MM periodic to do
        4 3877 { the reclaiming as soon as possible.
        4 3878
        4 3879     IF mmv$number_free_astes < 30 THEN {30 is arbitrary number}
       14 3880       IF mmv$number_free_astes = 0 THEN
       18 3881         trace (mmc$ap_no_asids, 1);
       18 3882         mmp$reclaim_ast_entries (0);
       34 3883         IF mmv$number_free_astes = 0 THEN
       3C 3884           mtp$error_stop ('MM26 - AST full');
       5A 3885         IFEND;
       5E 3886       ELSE
       5E 3887         trace (mmc$ap_low_asids, 1);
       5E 3888         mmv$async_work.reclaim_astes := TRUE;
       5E 3889         mmp$nudge_periodic_call;
       86 3890       IFEND;
       86 3891     IFEND;
       86 3892
       86 3893     mmv$number_free_astes := mmv$number_free_astes - 1;
       86 3894
       86 3895 { Find and assign a free AST entry. CACHE and MAP must be purged when the AST assignment algorithm wraps
       86 3896 { around and reassigns entries that may still be in cache or map. Note that free entries cannot be reused
       86 3897 { until a purge occurs.
       86 3898
       86 3899     REPEAT
       90 3900       next_asti := next_asti - 1;
       90 3901       IF next_asti = 0 THEN
       9E 3902         next_asti := UPPERBOUND (mmv$ast_p^);
       9E 3903         last_purge_time := #FREE_RUNNING_CLOCK (0);
       BA 3904         mmp$purge_all_cache_map;
```

ASID MANAGEMENT
[XDCL] mmp$assign_asid

```
      EE  3905          trace (mmc$ap_ast_reset, 1);
      F8  3906        IFEND;
      F8  3907      UNTIL NOT mmv$ast_p^ [next_asti].in_use AND (mmv$ast_p^ [next_asti].time_freed < last_purge_time);
     130  3908
     130  3909      asti := next_asti;
     130  3910      aste_p := ^mmv$ast_p^ [next_asti];
     130  3911      asid := aste_p^.asid;
     130  3912      aste_p^.in_use := TRUE;
     130  3913      trace (mmc$ap_assign, 1);
     130  3914
     130  3915      IF mmc$debug THEN
     130  3916        mmp$aste_pointer (asid, zaste_p);
     188  3917        IF zaste_p <> aste_p THEN
     198  3918          mtp$error_stop ('MM - bad ASID in assign  ASID');
     1B8  3919        IFEND;
     1B8  3920      IFEND;
     1B8  3921
     1B8  3922
     1B8  3923    PROCEND mmp$assign_asid;
```


ASID MANAGEMENT
[XDCL] mmp$assign_specific_asid

```
       0  3925 { PURPOSE:
       0  3926 {   This procedure is used by the Job Swapper to reclaim a specific ASID for swapin. No cache/map
       0  3927 {   purging is required since the ASID will be reclaimed ONLY if it is currently free AND has not
       0  3928 {   been used by another job while the reclaiming job was swapped out. It is important
       0  3929 {   that the ASID not have been used by another job because cache/map is not purged
       0  3930 {   as part of swapin.
       0  3931 {
       0  3932 { DESIGN:
       0  3933 {   To reclaim an ASID, simply reset the in_use field. Counts of AST entries in
       0  3934 {   use are adjusted as required.
       0  3935 {
       0  3936 { NOTE:
       0  3937 {   The caller is responsible for verifying that the ASID can correctly be reclaimed.
       0  3938
       0  3939    PROCEDURE [XDCL] mmp$assign_specific_asid
       0  3940      (     aste_p: ^mmt$active_segment_table_entry);
       0  3941
       0  3942      VAR
       0  3943        asti: mmt$ast_index,
       0  3944        zaste_p: ^mmt$active_segment_table_entry;
       0  3945
       0  3946      mmv$number_free_astes := mmv$number_free_astes - 1;
       4  3947      trace (mmc$ap_assign_specific, 1);
       4  3948
       4  3949      IF mmc$debug THEN
       4  3950        mmp$asti (aste_p^.asid, asti);
      3C  3951        zaste_p := ^mmv$ast_p^ [asti];
      3C  3952        IF zaste_p <> aste_p THEN
      5C  3953          mtp$error_stop ('MM - bad ASID in assign specific ASID');
      7C  3954        IFEND;
      7C  3955        IF aste_p^.in_use THEN
      84  3956          mtp$error_stop ('MM - assign specific of already in use');
      A4  3957        IFEND;
      A4  3958      IFEND;
      A4  3959
      A4  3960      aste_p^.in_use := TRUE;
      A4  3961
      A4  3962    PROCEND mmp$assign_specific_asid;
```

ASID MANAGEMENT
[XDCL] mmp$free_asid

```
        o  3964 { PURPOSE:
        o  3965 {   This procedure is called to free an ASID. The AST entry corresponding to the ASID is marked as free.
        o  3966 {
        o  3967 { DESIGN:
        o  3968 {   To free an ASID, the AST entry is marked as not in use and the ASID is saved. The
        o  3969 {   time the entry was freed is saved in the AST. The timestamp is used by mmp$assign to
        o  3970 {   correctly manage cache and map purges. An ASID cannot be assigned to a different segment
        o  3971 {   until cache and map are purged. Failing to purge the cache or map will cause failures since
        o  3972 {   stale data may be used.
        o  3973
        o  3974
        o  3975   PROCEDURE [XDCL] mmp$free_asid
        o  3976     (     asid: ost$asid;
        o  3977           aste_p: ^mmt$active_segment_table_entry);
        o  3978
        o  3979     VAR
        o  3980       zaste_p: ^mmt$active_segment_table_entry;
        o  3981
        o  3982     IF NOT aste_p^.in_use OR (aste_p^.pages_in_memory <> 0) THEN
       18  3983       mtp$error_stop ['MM42 - error in free ast entry'];
       38  3984     IFEND;
       38  3985
       38  3986     IF mmc$debug THEN
       38  3987       mmp$aste_pointer (asid, zaste_p);
       54  3988       IF zaste_p <> aste_p THEN
       64  3989         mtp$error_stop ('MM - bad ASID in free ASID');
       84  3990       IFEND;
       84  3991     IFEND;
       84  3992
       84  3993     mmv$number_free_astes := mmv$number_free_astes + 1;
       84  3994
       84  3995     aste_p^.in_use := FALSE;
       84  3996     aste_p^.asid := asid;
       84  3997     aste_p^.time_freed := #FREE_RUNNING_CLOCK (0);
       A6  3998
       A6  3999     IF mmc$debug AND mmv$test_reassign_asid THEN
       B6  4000       aste_p^.sfid.file_hash := 255;
       BE  4001     IFEND;
       BE  4002
       BE  4003     trace (mmc$ap_free_aste, 1);
       BE  4004
       BE  4005   PROCEND mmp$free_asid;
```


ASID MANAGEMENT
[XDCL] mmp$change_asid

```
        o  4007 { PURPOSE:
        o  4008 {   This procedure will change an ASID from an old value to a new value.
        o  4009 {   All affected Segment tables, FDEs, and system tables are updated as required.
        o  4010 {
        o  4011 { DESIGN:
        o  4012 {   This procedure searches all segment tables of all tasks that could be using the ASID.
        o  4013 {   The number of tasks and jobs to search depends on the attributes of the segment/file that is using
        o  4014 {   the ASID. If a job that should be searched is swapped out, a flag is set in the IJL
        o  4015 {   to notify the swapper that ASIDs must be fixed on swap-in.
        o  4016 {
        o  4017 {   The following table shows all the types of ASIDs that can be in SDTs and the way these
        o  4018 {   ASIDs are located when they change (ie., what is searched). Note that the table is
        o  4019 {   ordered from easiest-to-change to hardest-to-change - the number at the beginning of the line
        o  4020 {   indicates preferred order for reassignment.
        o  4021 {
        o  4022 {       ASID                        NOT SWAPPED                    SWAPPED
        o  4023 { 1   local file/transient   search all XCBs in AST.IJL0   dsw_job_asid_changed
        o  4024 {
        o  4025 { 2   perm file in JWS       search all XCBs in AST.IJL0   dsw_job_asid_changed
        o  4026 {
        o  4027 { 3   perm file in shared Q  search all XCBs               mmv$time_changed_global_asid > timestamp
        o  4028 {
        o  4029 { 3   template asid          search all XCBs               mmv$time_changed_template_asid  > timestamp
        o  4030 {
        o  4031
        o  4032
        o  4033   PROCEDURE [XDCL] mmp$change_asid
        o  4034     (     aste_p: ^mmt$active_segment_table_entry;
        o  4035           old_asid: ost$asid;
        o  4036           new_asid: ost$asid;
        o  4037           new_asti: mmt$ast_index);
        o  4038
        o  4039     VAR
        o  4040       cell_p: ^cell,
        o  4041       fde_p: gft$file_desc_entry_p,
        o  4042       ijle_p: ^jmt$initiated_job_list_entry,
        o  4043       max_segnum: ost$segment,
        o  4044       segnum: integer, {allow negative numbers}
        o  4045       sdt_p: mmt$max_sdt_p,
        o  4046       ste_p: ^mmt$segment_descriptor,
        o  4047       xcb_p: ^ost$execution_control_block,
        o  4048       xcb_state: tmt$find_next_xcb_state;
        o  4049
        o  4050
        o  4051
        o  4052 { Get a pointer to the XCB of the first task whose segment table is to be scanned. If the job that
        o  4053 { owns the file is swapped, this pointer will be NIL. If the job is partially swapped, this
        o  4054 { request will assign an AJL ordinal. Note that the actual XCBs scanned will be either all XCBs in
        o  4055 { the system (if global queue_id) or just XCBs in a specific job (job queue_id).
        o  4056
        o  4057     jmp$get_ijle_p (aste_p^.ijl_ordinal, ijle_p);
        4  4058     IF aste_p^.queue_id < mmc$pq_job_base THEN
       3C  4059       tmp$find_next_xcb (tmc$fnx_system, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
       76  4060     ELSE
       76  4061       tmp$find_next_xcb (tmc$fnx_job, ijle_p, aste_p^.ijl_ordinal, xcb_state, xcb_p);
```

ASID MANAGEMENT
[XDCL] mmp$change_asid

```
A6  4062      IFEND;
A6  4063
A6  4064
A6  4065  { If the FDE is accessible (job not swapped OR tables in mainframe wired), get a pointer
A6  4066  { to the FDE and update the ASTI. (If the job is swapped, the FDE will be updated
A6  4067  { on the next swap-in). If the ASID belongs to a job fixed segment, fix the ASID in the
A6  4068  { monitor segment table.
A6  4069
A6  4070      IF (xcb_p <> NIL) OR (aste_p^.sfid.residence = gfc$tr_system) THEN
BA  4071        gfp$mtr_get_locked_fde_p (aste_p^.sfid, ijle_p, fde_p);
13C 4072        fde_p^.asti := new_asti;
14A 4073      IFEND;
14A 4074
14A 4075      IF old_asid = ijle_p^.job_fixed_asid THEN
15A 4076        ijle_p^.job_fixed_asid := new_asid;
15A 4077        IF ijle_p^.ajl_ordinal <> jmc$null_ajl_ordinal THEN
16E 4078          mtv$monitor_segment_table.st [ijle_p^.ajl_ordinal + mtc$job_fixed_segment].ste.asid := new_asid;
17A 4079        IFEND;
17A 4080      IFEND;
17A 4081
17A 4082  { If the job is swapped, then set the delayed swapin flag and fix the ASIDs when the
17A 4083  { job next swaps in.
17A 4084
17A 4085      IF xcb_p = NIL THEN
184 4086        trace (mmc$ap_casid_swapped_job, 1);
184 4087        ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work +
1A8 4088              $jmt$delayed_swapin_work [jmc$dsw_job_asid_changed];
1A8 4089
1A8 4090
1A8 4091  { If the segment exists in monitor's address space ONLY, nothing needs to be done here. Code
1A8 4092  { further down will fix monitor's segment table.
1A8 4093
1A8 4094      ELSEIF fde_p^.file_kind = gfc$fk_monitor_only_unnamed THEN
1B6 4095        trace (mmc$ap_casid_monitor, 1);
1C8 4096
1C8 4097
1C8 4098  { The segment may be in segment tables in one or more jobs in memory. Fix the ASID in all jobs
1C8 4099  { that could be referencing the file.
1C8 4100  {!! Once async access to FDE.OPEN_COUNT is understood, maybe it can be used below.
1C8 4101
1C8 4102      ELSE
1C8 4103        IF fde_p^.flags.global_template_file THEN
1D2 4104          mmv$time_changed_template_asid := #FREE_RUNNING_CLOCK (0);
1DA 4105          trace (mmc$ap_casid_template, 1);
1F0 4106
1F0 4107        ELSEIF aste_p^.sfid.residence = gfc$tr_system THEN
1F8 4108          IF aste_p^.queue_id >= mmc$pq_job_base THEN
204 4109            trace (mmc$ap_casid_job, 1); {! delete dsw_job_shared_asid_changed}
216 4110          ELSE
216 4111            mmv$time_changed_global_asid := #FREE_RUNNING_CLOCK (0);
21E 4112            trace (mmc$ap_casid_global, 1);
230 4113          IFEND;
232 4114        IFEND;
232 4115
232 4116
```


ASID MANAGEMENT
[XDCL] mmp$change_asid

```
232 4117        IF fde_p^.flags.global_template_file THEN
240 4118          max_segnum := mmv$max_template_segment_number;
24C 4119        ELSE
24C 4120          max_segnum := 4095;
250 4121        IFEND;
250 4122
250 4123      /fix_ste_loop/
250 4124        WHILE xcb_p <> NIL DO
25E 4125          mmp$get_max_sdt_pointer (xcb_p, sdt_p);
25E 4126          segnum := xcb_p^.xp.segment_table_length;
25E 4127          IF segnum > max_segnum THEN
28C 4128            segnum := max_segnum;
290 4129          IFEND;
290 4130          WHILE segnum >= 0 DO
294 4131            IF sdt_p^.st [segnum].ste.asid = old_asid THEN
2A8 4132              sdt_p^.st [segnum].ste.asid := new_asid;
2A8 4133              sdt_p^.st [segnum].asti := new_asti;
2B8 4134            IFEND;
2B8 4135            segnum := segnum - 1;
2B8 4136          WHILEND;
2BE 4137          tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
2EA 4138        WHILEND /fix_ste_loop/;
2FE 4139
2FE 4140      IFEND;
2FE 4141
2FE 4142
2FE 4143  { Change the ASID in monitor's segment table and NOS segment table if segment is not pageable.
2FE 4144
2FE 4145      IF (aste_p^.queue_id = mmc$pq_wired) THEN
30C 4146        cell_p := ^mtv$monitor_segment_table;
30C 4147        #PURGE_BUFFER (osc$pva_purge_segment_cache, cell_p);
316 4148        FOR segnum := 0 TO mtc$job_fixed_segment - 1 DO
322 4149          IF mtv$monitor_segment_table.st [segnum].ste.asid = old_asid THEN
332 4150            mtv$monitor_segment_table.st [segnum].ste.asid := new_asid;
33A 4151          IFEND;
33A 4152        FOREND;
33E 4153        FOR segnum := 0 TO UPPERBOUND (mtv$nos_segment_table_p^.st) DO
350 4154          IF mtv$nos_segment_table_p^.st [segnum].ste.asid = old_asid THEN
364 4155            mtv$nos_segment_table_p^.st [segnum].ste.asid := new_asid;
36C 4156          IFEND;
36C 4157        FOREND;
370 4158      IFEND;
370 4159
370 4160
370 4161  { Purge the segment file and page file to remove any entries that have the old ASID.
370 4162
370 4163      mmp$purge_all_page_seg_map;
38C 4164
38C 4165
38C 4166    PROCEND mmp$change_asid;
```

ASID MANAGEMENT
[XDCL] mmp$reclaim_ast_entries

```
        o  4168 { PURPOSE
        o  4169 {   This routine is called to search through the AST and free entries that
        o  4170 {   have no pages in memory. The routine is normally called by mmp$periodic_call
        o  4171 {   with accounting being charged to the system.
        o  4172 {   If an AST full condition is imminent, this routine is called by mmp$assign_asid - a user will incur
        o  4173 {   the overhead necessary to try to keep the system from hanging.
        o  4174 {
        o  4175 { DESIGN:
        o  4176 {   An ASID can be reclaimed if there are no pages in memory using the ASID.  Since there
        o  4177 {   are more ASID than page frames, it should always be possible to reclaim enough ASIDs.
        o  4178 {   When an ASID is reclaimed, the ASID in all segment tables that contain the ASID is
        o  4179 {   zeroed out. Device manager is also notified to update its value of the ASID. Note
        o  4180 {   that ASIDs belonging to swapped out jobs may be reclaimed while the job is swapped out.
        o  4181 {   In this case, the job swapper is responsible for fixing the segment tables
        o  4182 {   at swap-in time.
        o  4183 {
        o  4184
        o  4185   PROCEDURE [XDCL] mmp$reclaim_ast_entries
        o  4186     (     asti_that_cannot_be_freed: mmt$ast_index);
        o  4187
        o  4188     VAR
        o  4189       asid: ost$asid,
        o  4190       end_of_table_seen: boolean,
        o  4191       next_aste_index: mmt$ast_index,
        o  4192       number_astes_reclaimed: integer;
        o  4193
        o  4194     next_aste_index := next_asti;
        4  4195     end_of_table_seen := FALSE;
        4  4196     number_astes_reclaimed := 0;
        4  4197     trace (mmc$ap_reclaim_asids, 1);
        4  4198
        4  4199   /reclaim_loop/
        4  4200
        4  4201     WHILE number_astes_reclaimed < 30 DO {30 is arbitrary number}
       24  4202       next_aste_index := next_aste_index - 1;
       24  4203       IF next_aste_index = 0 THEN
       2A  4204         IF end_of_table_seen THEN
       2E  4205           RETURN;
       30  4206         IFEND;
       30  4207         end_of_table_seen := TRUE;
       30  4208         next_aste_index := UPPERBOUND (mmv$ast_p^);
       4A  4209       IFEND;
       4A  4210       IF mmv$ast_p^ [next_aste_index].in_use AND (mmv$ast_p^ [next_aste_index].pages_in_memory = 0) AND
       70  4211            (asti_that_cannot_be_freed <> next_aste_index) THEN
       70  4212         mmp$asid (next_aste_index, asid);
       88  4213         mmp$change_asid (^mmv$ast_p^ [next_aste_index], asid, 0, 0);
       AE  4214         mmp$free_asid (asid, ^mmv$ast_p^ [next_aste_index]);
       CC  4215         number_astes_reclaimed := number_astes_reclaimed + 1;
       CE  4216       IFEND;
       CE  4217
       CE  4218     WHILEND /reclaim_loop/;
       D6  4219
       D6  4220   PROCEND mmp$reclaim_ast_entries;
```


PAGE TABLE MANAGEMENT
[XDCL] mmp$make_pt_entry

```
        o  4224 { Purpose:
        o  4225 {   This routine makes an entry in the system page table after
        o  4226 {   checking to make sure a page table entry for the page does
        o  4227 {   not already exist.
        o  4228 { Input:
        o  4229 {   sva - SVA of any byte in the page
        o  4230 {   pfti - index of page frame to assign to the page
        o  4231 {   aste_p - pointer to AST entry for page
        o  4232 {   pfte_p - pointer to PFT entry for page. ONLY THE PTI ENTRY IS USED. IMPORTANT
        o  4233 {            because REASSIGN_ASID passes a dummy pfte_p.
        o  4234 { Output:
        o  4235 {   pti - page_table_index of PT entry assigned to the page is stored into the PFT
        o  4236 {         entry located via the PFTE_P input parameter.
        o  4237 { Error Codes:
        o  4238 {   status - The following errors may be detected by this proc
        o  4239 {      page table full
        o  4240 {      page table entry exists
        o  4241
        o  4242   VAR
        o  4243     mmv$page_table_miss_count: [XDCL] array [1 .. 34] of integer;
        o  4244
        o  4245
        o  4246   PROCEDURE [XDCL] mmp$make_pt_entry
        o  4247     (     sva: ost$system_virtual_address;
        o  4248           pfti: mmt$page_frame_index;
        o  4249           aste_p: ^mmt$active_segment_table_entry;
        o  4250           pfte_p: ^mmt$page_frame_table_entry;
        o  4251      VAR mpt_status: mmt$make_pt_entry_status);
        o  4252
        o  4253     VAR
        o  4254       cbc_p: ^mmt$continue_bit_count,
        o  4255       count: 0 .. 33,
        o  4256       fcount: 0 .. 31,
        o  4257       found: boolean,
        o  4258       hcount: 1 .. 32,
        o  4259       pt_p: ^ost$page_table,
        o  4260       pte: ost$page_table_entry,
        o  4261       pti: integer,
        o  4262       save_pti: integer,
        o  4263       starting_pti: integer;
        o  4264
        o  4265
        o  4266
        o  4267 { Calculate the hash index for the page table entry and determine if the page already exists. Return an error
        o  4268 { code if an entry already exists.
        o  4269
        o  4270     #HASH_SVA (sva, starting_pti, hcount, found);
        A  4271     IF found THEN
       26  4272       mpt_status := mmc$mpt_page_already_exists;
       26  4273       RETURN;
       32  4274     IFEND;
       32  4275     starting_pti := starting_pti - hcount + 1;
       32  4276     IF starting_pti < 0 THEN
       3E  4277       starting_pti := starting_pti + mmv$pt_length;
       48  4278     IFEND;
```

PAGE TABLE MANAGEMENT
[XDCL] mmp$make_pt_entry

```
48  4279
48  4280
48  4281
48  4282 { Find an available slot for the new page table entry. Set 'continue' bits as required.  Return error if no
48  4283 { space is found within 32 entries. Note that early in deadstart, the continue bit array is not allocated.
48  4284 { During this time the page table is completely filled with entries with 'C' set.
48  4285
48  4286     count := 1;
48  4287     pt_p := mmv$pt_p;
48  4288     cbc_p := mmv$continue_bit_count_p;
62  4289     pti := starting_pti;
62  4290     WHILE (pt_p^ [pti].pageid.asid <> 0) AND (count < mmv$pt_search + 1) DO
84  4291       cbc_p^ [pti] := cbc_p^ [pti] + 1;
84  4292       IF cbc_p^ [pti] = 1 THEN
9A  4293         pt_p^ [pti].c := TRUE;
A8  4294       IFEND;
A8  4295       count := count + 1;
A8  4296       pti := pti + 1;
A8  4297       IF pti = mmv$pt_length THEN
BC  4298         pti := 0;
C2  4299       IFEND;
C2  4300     WHILEND;
E0  4301     mmv$page_table_miss_count [count] := mmv$page_table_miss_count [count] + 1;
E0  4302
E0  4303
E0  4304 { If no entry was found within the required 32 entries,try to free some entries in the page table
E0  4305 { in the area searched. Clear unnecessary continue bits that were set in the above loop.
E0  4306 { Exit if not possible to make entry in the page table.
E0  4307
E0  4308     IF count = (mmv$pt_search + 1) THEN
F8  4309       trace (mmc$ap_mpte_full, 1);
F8  4310       save_pti := pti;
F8  4311       free_pt_entry_in_avail_queue (starting_pti, pfti, pti, fcount, found);
124 4312       clear_continue_bits (save_pti, (mmv$pt_search + 1) - fcount);
174 4313       IF NOT found THEN
17C 4314         mpt_status := mmc$mpt_page_table_full;
17C 4315         RETURN;
186 4316       IFEND;
186 4317       mmv$page_table_miss_count [34] := mmv$page_table_miss_count [34] + 1;
186 4318       trace (mmc$ap_mpte_recovered, 1);
19A 4319     IFEND;
19A 4320
19A 4321
19A 4322 { Make the new page table entry, preserving the 'continue' bit in the old page table entry.
19A 4323
19A 4324     pte.v := FALSE;
19A 4325     pte.c := pt_p^ [pti].c;
19A 4326     pte.u := TRUE;
19A 4327     pte.m := FALSE;
19A 4328     pte.pageid.asid := sva.asid;
19A 4329     pte.pageid.pagenum :=
19A 4330     #SHIFT (sva.offset, -9);
19A 4331     pte.rma :=
19A 4332     #SHIFT (pfti * osv$page_size, -9);
19A 4333     pt_p^ [pti] := pte;
```


PAGE TABLE MANAGEMENT
[XDCL] mmp$make_pt_entry

```
19A 4334     pfte_p^.pti := pti;
19A 4335     IF NOT aste_p^.in_use THEN
208 4336       mtp$error_stop ('MM--MAKE_PT_ENTRY--AST NOT IN USE');
228 4337     IFEND;
228 4338
228 4339 { A non-zero changing_asid indicates this procedure was called by page table full processing.  The page is a
228 4340 { new page and should be linked to the segment only if the call to this procedure was NOT made for page table
228 4341 { full processing.
228 4342
228 4343     IF changing_asid = 0 THEN
230 4344       mmp$link_page_to_segment (pfti, pfte_p, aste_p);
2E6 4345     IFEND;
2E6 4346     aste_p^.pages_in_memory := aste_p^.pages_in_memory + 1;
2E6 4347     mmv$pages_to_dump_p^ [pfti] := aste_p^.include_pages_in_dump;
2E6 4348     mpt_status := mmc$mpt_done;
2E6 4349
2E6 4350   PROCEND mmp$make_pt_entry;
```

PAGE TABLE MANAGEMENT
[XDCL] mmp$delete_pt_entry

```
      o  4352 { Purpose:
      o  4353 {   This routine deletes a page table entry for a page.
      o  4354 {
      o  4355 { Input:
      o  4356 {   pfti - page frame table index of frame assigned to the page
      o  4357 {
      o  4358 { Output:
      o  4359 {   none
      o  4360 {
      o  4361 { NOTE !!! Caller must either clear the 'v' bit & purge map in the page table or otherwise ensure
      o  4362 {   that the entry being deleted is NOT being referenced by another CPU in a multi-CPU
      o  4363 {   configuration.
      o  4364 {   The unlink_page_from_segment parameter is used to indicate whether the page should be unlinked
      o  4365 {   from the segment.  This parameter should be FALSE only from a few calls in page table full
      o  4366 {   processing.  Because page table full processing creates two page table entries for one page frame
      o  4367 {   for a short time, the duplicate page table entry must be deleted without unlinking the page frame
      o  4368 {   from the segment.
      o  4369
      o  4370
      o  4371   PROCEDURE [XDCL] mmp$delete_pt_entry
      o  4372     (     pfti: mmt$page_frame_index;
      o  4373           unlink_page_from_segment: boolean);
      o  4374
      o  4375     VAR
      o  4376       aste_p: ^mmt$active_segment_table_entry,
      o  4377       count: integer,
      o  4378       found: boolean,
      o  4379       pfte_p: ^mmt$page_frame_table_entry,
      o  4380       pte_p: ^ost$page_table_entry,
      o  4381       pti: integer;
      o  4382
      o  4383
      o  4384 { Entry has been found.  Delete the entry by clearing the valid bit and setting ASID to zero.
      o  4385
      o  4386     pfte_p := ^mmv$pft_p^ [pfti];
      4  4387     #HASH_SVA (pfte_p^.sva, pti, count, found);
     2A  4388     pti := pfte_p^.pti;
     2A  4389     pte_p := ^mmv$pt_p^ [pti];
     2A  4390
     2A  4391     IF mmc$debug THEN
     2A  4392       IF (pti <> pfte_p^.pti) OR NOT found THEN
     56  4393         mtp$error_stop ('MM - bad PFT.pti on delete pte');
     76  4394       IFEND;
     76  4395       IF (pte_p^.pageid.asid <> pfte_p^.sva.asid) OR (pte_p^.pageid.pagenum * 512 <> pfte_p^.sva.offset) OR
     BE  4396           (pte_p^.rma * 512 <> (pfti * osv$page_size)) THEN
     BE  4397         mtp$error_stop ('MM - illegal delete pte');
     DE  4398       IFEND;
     DE  4399     IFEND;
     DE  4400
     DE  4401     pte_p^.v := FALSE;
     DE  4402     pte_p^.pageid.asid := 0;
     DE  4403
     DE  4404 { Clear continue bits if necessary.
     DE  4405
     DE  4406     IF (count > 1) THEN
```


PAGE TABLE MANAGEMENT
[XDCL] mmp$delete_pt_entry

```
     F8  4407       clear_continue_bits (pti, count);
    140  4408     IFEND;
    140  4409
    140  4410
    140  4411 { Decrement the 'pages in memory' field of the AST.
    140  4412
    140  4413     aste_p := pfte_p^.aste_p;
    140  4414     IF aste_p^.pages_in_memory = 0 THEN
    14C  4415       mtp$error_stop ('MM - delete pte, no pages in memory');
    16E  4416     IFEND;
    16E  4417     aste_p^.pages_in_memory := aste_p^.pages_in_memory - 1;
    16E  4418
    16E  4419 { Unlink page from segment is true from all callers except a few specific calls from process page table full.
    16E  4420
    16E  4421     IF unlink_page_from_segment THEN
    180  4422       mmp$unlink_page_from_segment (pfte_p, aste_p);
    230  4423     IFEND;
    230  4424
    230  4425   PROCEND mmp$delete_pt_entry;
      o  4426
```

PAGE TABLE MANAGEMENT
clear_continue_bits

```
o  4428
o  4429    PROCEDURE [INLINE] clear_continue_bits
o  4430      (     xpti: ost$page_table_index;
o  4431            count: integer);
o  4432
o  4433      VAR
o  4434        i: integer,
o  4435        pti: integer;
o  4436
o  4437
o  4438  { Clear 'continue' bits as required.  Decrement the count of the number of times the
o  4439  { bit is 'set'. When the count goes to zero, clear the continue bit in the page table.
o  4440
o  4441      pti := xpti;
o  4442      FOR i := 2 TO count DO
o  4443        pti := pti - 1;
o  4444        IF pti < 0 THEN
o  4445          pti := mmv$pt_length - 1;
o  4446        IFEND;
o  4447        mmv$continue_bit_count_p^ [pti] := mmv$continue_bit_count_p^ [pti] - 1;
o  4448        IF mmv$continue_bit_count_p^ [pti] = 0 THEN
o  4449          mmv$pt_p^ [pti].c := FALSE;
o  4450        IFEND;
o  4451      FOREND;
o  4452
o  4453    PROCEND clear_continue_bits;
o  4454
```


PAGE TABLE MANAGEMENT
mmp$link_page_to_segment

```
o  4456  { Purpose:
o  4457  {   This procedure is called from mmp$make_pt_entry to insert the page frame into the
o  4458  {   thread which links all pages of a segment that are in memory.  There must be NO OTHER CALLERS
o  4459  {   of this procedure, or the integrity of the links will be destroyed.
o  4460
o  4461    PROCEDURE [XDCL, INLINE] mmp$link_page_to_segment
4  4462      (     pfti: mmt$page_frame_index;
4  4463            pfte_p: ^mmt$page_frame_table_entry;
4  4464            aste_p: ^mmt$active_segment_table_entry);
4  4465
4  4466  { Debug code
4  4467
4  4468      IF (pfte_p^.segment_link.fwd <> 0) OR (pfte_p^.segment_link.bkw <> 0) THEN
18 4469        mtp$error_stop ('LINK PAGE TO SEGMENT ERROR');
38 4470      IFEND;
38 4471
38 4472      IF (aste_p^.pages_in_memory = 0) AND ((aste_p^.pft_link.bkw <> 0) OR (aste_p^.pft_link.fwd <> 0)) THEN
54 4473        mtp$error_stop ('LINK PAGE TO SEGMENT ERROR--AST');
74 4474      IFEND;
74 4475
74 4476  { End debug code
74 4477
74 4478      IF aste_p^.pft_link.fwd = 0 THEN
7C 4479        aste_p^.pft_link.fwd := pfti;
7C 4480        aste_p^.pft_link.bkw := pfti;
8A 4481      ELSE
8A 4482        mmv$pft_p^ [aste_p^.pft_link.bkw].segment_link.fwd := pfti;
8A 4483        pfte_p^.segment_link.bkw := aste_p^.pft_link.bkw;
8A 4484        aste_p^.pft_link.bkw := pfti;
BC 4485      IFEND;
BC 4486
BC 4487    PROCEND mmp$link_page_to_segment;
 0 4488
```

PAGE TABLE MANAGEMENT
mmp$unlink_page_from_segment

```
 0  4490 { Purpose:
 0  4491 {   This procedure is called from mmp$delete_pt_entry to remove the page frame from the
 0  4492 {   thread which links all pages of a segment that are in memory.  There must be NO OTHER CALLERS
 0  4493 {   of this procedure, or the integrity of the links will be destroyed.
 0  4494
 0  4495   PROCEDURE [XDCL, INLINE] mmp$unlink_page_from_segment
 4  4496     (     pfte_p: ^mmt$page_frame_table_entry;
 4  4497           aste_p: ^mmt$active_segment_table_entry);
 4  4498
 4  4499
 4  4500     IF pfte_p^.segment_link.fwd = 0 THEN
10  4501       aste_p^.pft_link.bkw := pfte_p^.segment_link.bkw;
20  4502     ELSE
20  4503       mmv$pft_p^ [pfte_p^.segment_link.fwd].segment_link.bkw := pfte_p^.segment_link.bkw;
42  4504     IFEND;
42  4505
42  4506     IF pfte_p^.segment_link.bkw = 0 THEN
4A  4507       aste_p^.pft_link.fwd := pfte_p^.segment_link.fwd;
5A  4508     ELSE
5A  4509       mmv$pft_p^ [pfte_p^.segment_link.bkw].segment_link.fwd := pfte_p^.segment_link.fwd;
7C  4510     IFEND;
7C  4511
7C  4512     pfte_p^.segment_link.fwd := 0;
7C  4513     pfte_p^.segment_link.bkw := 0;
7C  4514
7C  4515 { Debug code
7C  4516
7C  4517     IF (changing_asid = 0) AND (aste_p^.pages_in_memory = 0) AND
AE  4518         ((aste_p^.pft_link.bkw <> 0) OR (aste_p^.pft_link.fwd <> 0)) THEN
AE  4519       mtp$error_stop ('LINK PAGE TO SEGMENT ERROR--AST');
CE  4520     IFEND;
CE  4521
CE  4522 { End debug
CE  4523
CE  4524   PROCEND mmp$unlink_page_from_segment;
 0  4525
```


PAGE TABLE MANAGEMENT
free_pt_entry_in_avail_queue

```
 0  4527 { PURPOSE
 0  4528 {   This procedure is used in page table full processing. It scans the 32 page table entries starting
 0  4529 {   at the specified hash index.If an entry in the AVAIL queue is found, it is freed.
 0  4530 {
 0  4531 { INPUT:
 0  4532 {    initial_pti: starting hash index
 0  4533 {    initial_pfti: PFT index of page frame that is being entered into page table
 0  4534 {             A PT entry belonging to this page will NOT be deleted. (required for page table
 0  4535 {             full processing - see REASSIGN_ASID.
 0  4536 { OUTPUT:
 0  4537 {    pti : index to freed entry (undefined if no entry freed)
 0  4538 {    count: number of entries searched (- 1) before finding entry to free (0 = none found)
 0  4539 {    freed: boolean to indicate if entry freed
 0  4540
 0  4541   PROCEDURE free_pt_entry_in_avail_queue
 0  4542     (     initial_pti: integer;
 0  4543           initial_pfti: mmt$page_frame_index;
 0  4544      VAR xpti: integer;
 0  4545      VAR xcount: 0 .. 31;
 0  4546      VAR freed: boolean);
 0  4547
 0  4548     VAR
 0  4549       count: 0 .. 31,
 0  4550       pfte: mmt$page_frame_table_entry,
 0  4551       pfti: mmt$page_frame_index,
 0  4552       pte_p: ^ost$page_table_entry,
 0  4553       pti: integer;
 0  4554
 0  4555 { Scan the 32 entries and free the first entry found that is in the AVAIL queue. NOTE:
 0  4556 { link the page frame to the FREE queue only if the PFT.PTI field is correct. If its not
 0  4557 { correct, then the entry must belong to segment that is having its ASID reassigned as
 0  4558 { a result of a PAGE_TABLE_FULL condition. Also, an entry using the SAME page frame as the one
 0  4559 { for which a new page table entry is being made cannot be deleted. This case arises during page
 0  4560 { table full processing - both entries must exist at the same time in order for the
 0  4561 { page table full algorithms to work.
 0  4562
 0  4563     pti := initial_pti;
 4  4564     FOR count := 0 TO mmv$pt_search - 1 DO
16  4565       pte_p := ^mmv$pt_p^ [pti];
16  4566       IF NOT pte_p^.m THEN
2E  4567         pfti := (pte_p^.rma * 512) DIV osv$page_size;
2E  4568         IF (pfti >= LOWERBOUND (mmv$pft_p^)) AND (pfti <= UPPERBOUND (mmv$pft_p^)) AND
8C  4569               (mmv$pft_p^ [pfti].queue_id = mmc$pq_avail) AND (pfti <> initial_pfti) THEN
8C  4570           trace (mmc$ap_mpte_rec1, 1);
8C  4571           IF mmv$pft_p^ [pfti].pti = pti THEN
9E  4572             trace (mmc$ap_mpte_rec2, 1);
9E  4573             mmp$delete_pt_entry (pfti, TRUE);
BC  4574             mmp$relink_page_frame (pfti, mmc$pq_free);
D4  4575           ELSEIF (pte_p^.pageid.asid = changing_asid) THEN
E4  4576             pfte := mmv$pft_p^ [pfti];
F2  4577             mmv$pft_p^ [pfti].sva.asid := pte_p^.pageid.asid;
F2  4578             mmv$pft_p^ [pfti].pti := pti;
F2  4579             mmv$pft_p^ [pfti].aste_p := changing_aste_p;
F2  4580             mmp$delete_pt_entry (pfti, FALSE);
116 4581             mmv$pft_p^ [pfti] := pfte;
```

PAGE TABLE MANAGEMENT
free_pt_entry_in_avail_queue

```
13E  4582            ELSE
13E  4583              mtp$error_stop ('MM - PT/PFT mismatch');
15E  4584            IFEND;
15E  4585            xcount := count;
15E  4586            xpti := pti;
15E  4587            freed := TRUE;
15E  4588            RETURN;
16C  4589          IFEND;
16C  4590        IFEND;
16C  4591        pti := pti + 1;
16C  4592        IF pti = mmv$pt_length THEN
17A  4593          pti := 0;
17E  4594        IFEND;
17E  4595      FOREND;
182  4596
182  4597      xcount := 0;
182  4598      freed := FALSE;
182  4599
182  4600    PROCEND free_pt_entry_in_avail_queue;
```


PAGE TABLE FULL MANAGER
[XDCL] mmp$process_pt_full

```
 0  4604 { PURPOSE:
 0  4605 {   This procedure is called to try to recover from a 'page table full' condition.
 0  4606 {   The procedure does the following:
 0  4607 {     . try to reassign an ASID in the portion of the PT that is full.
 0  4608 {     . try to free or write to disk all pages in the part of the PT that
 0  4609 {       is full. (this step is attempted only if the previous step failed).
 0  4610 {
 0  4611 { INPUT:
 0  4612 {     sva: SVA that caused PT full.
 0  4613 {
 0  4614 { OUTPUT:
 0  4615 {     new_asid: new asid assigned
 0  4616 {     new_asti: asti of reassigned ASID
 0  4617 {     new_aste_p: aste_p of reassigned ASID
 0  4618 {     pt_full_status: indicates status of reassignment
 0  4619
 0  4620
 0  4621   PROCEDURE [XDCL] mmp$process_page_table_full
 0  4622     (     sva: ost$system_virtual_address;
 0  4623      VAR new_asid: ost$asid;
 0  4624      VAR new_asti: mmt$ast_index;
 0  4625      VAR new_aste_p: ^mmt$active_segment_table_entry;
 0  4626      VAR pt_full_status: mmt$pt_full_status);
 0  4627
 0  4628     VAR
 0  4629       asidl: asid_list_index_type,
 0  4630       asidlmax: asid_list_index_type,
 0  4631       asidt: [XDCL, STATIC] asid_list, {xdcled for debug only}
 0  4632       asti_that_cannot_be_freed: mmt$ast_index,
 0  4633       count: 1 .. 32,
 0  4634       found: boolean,
 0  4635       ijle_p: ^jmt$initiated_job_list_entry,
 0  4636       inhibit_io: boolean,
 0  4637       mcount: integer,
 0  4638       pass: reassign_pass,
 0  4639       pfte_p: ^mmt$page_frame_table_entry,
 0  4640       pfti: mmt$page_frame_index,
 0  4641       pte_p: ^ost$page_table_entry,
 0  4642       pti: integer,
 0  4643       pti_offset: 0 .. 32,
 0  4644       rcount: integer,
 0  4645       sort_index: asid_list_index_type;
 0  4646
 0  4647
 0  4648 { Reclaim unused ast entries.
 0  4649
 0  4650     IF mmv$async_work.reclaim_astes THEN
10  4651       mmv$async_work.reclaim_astes := FALSE;
10  4652       mmp$asti (sva.asid, asti_that_cannot_be_freed);
32  4653       mmp$reclaim_ast_entries (asti_that_cannot_be_freed);
46  4654     IFEND;
46  4655
46  4656 { Find the page table index that caused the page table full condition.
46  4657
46  4658     #HASH_SVA (sva, pti, count, found);
```

PAGE TABLE FULL MANAGER
[XDCL] mmp$process_pt_full

```
4C  4659      pti := pti - count + 1;
4C  4660      IF pti < 0 THEN
68  4661        pti := pti + mmv$pt_length;
72  4662      IFEND;
72  4663
72  4664      mmv$pt_full_trace.last_sva := sva;
72  4665      mmv$pt_full_trace.timestamp := #FREE_RUNNING_CLOCK (0);
86  4666
86  4667
86  4668 { Generate the list of ASIDs that can be changed to eliminate the page table full
86  4669 { condition. The list is sorted in order of 'easiest to change'.
86  4670
86  4671      build_asid_list (sva.asid, pti, ^asidt, asidlmax);
A4  4672
A4  4673
A4  4674 { Try to reassign an ASID until sucessful or reached end of list.
A4  4675
A4  4676      trace (mmc$ap_ptf_called, 1);
A4  4677      FOR sort_index := 1 TO asidlmax DO
BC  4678        trace (mmc$ap_ptf_tried, 1);
BC  4679        asidl := asidt [sort_index].index;
BC  4680        reassign_asid (asidt [asidl].asid, asidt [asidl].aste_p, new_asid, new_asti, new_aste_p,
10C 4681              pt_full_status);
10C 4682        IF pt_full_status = mmc$pfs_asid_reassigned THEN
118 4683          IF asidt [asidl].asid = sva.asid THEN
124 4684            pt_full_status := mmc$pfs_input_asid_reassigned;
12A 4685          IFEND;
12A 4686          mmv$pt_full_trace.index := mmv$pt_full_trace.index + 1;
12A 4687          IF mmv$pt_full_trace.index > 127 THEN
13C 4688            mmv$pt_full_trace.index := 0;
140 4689          IFEND;
140 4690          mmv$pt_full_trace.asid [mmv$pt_full_trace.index].old := asidt [asidl].asid;
140 4691          mmv$pt_full_trace.asid [mmv$pt_full_trace.index].new := new_asid;
140 4692          mmv$pt_full_trace.changed_asid := mmv$pt_full_trace.changed_asid + 1;
140 4693          pass := asidt [asidl].sort_key DIV osc$max_page_frames;
140 4694          mmv$pt_full_trace.pass [pass] := mmv$pt_full_trace.pass [pass] + 1;
140 4695          RETURN; {<------}
18E 4696        IFEND;
18E 4697      FOREND;
194 4698
194 4699
194 4700 { Reassigning an ASID failed. Try to free pages in the part of the PT that has
194 4701 { the PT full condition by removing pages from job working sets of jobs that
194 4702 { are in the part of the page table that is full. This may write the page to disk.
194 4703
194 4704      trace (mmc$ap_ptf_failed, 1);
194 4705      FOR pti_offset := 0 TO mmv$pt_search - 1 DO
1AE 4706        pte_p := ^mmv$pt_p^ [(pti)];
1AE 4707        pfti := (pte_p^.rma * 512) DIV osv$page_size;
1AE 4708        IF (pte_p^.pageid.asid <> 0) AND (pfti >= LOWERBOUND (mmv$pft_p^)) AND (pfti <= UPPERBOUND (mmv$pft_p^))
204 4709              THEN
204 4710          pfte_p := ^mmv$pft_p^ [pfti];
204 4711          IF ((pfte_p^.queue_id >= mmc$pq_shared_first) AND (pfte_p^.queue_id <= mmc$pq_shared_last)) OR
238 4712                (pfte_p^.queue_id = mmc$pq_job_working_set) THEN
238 4713            mmp$get_inhibit_io_status (pfte_p^.ijl_ordinal, TRUE [lock ajl] , inhibit_io, ijle_p);
```


PAGE TABLE FULL MANAGER
[XDCL] mmp$process_pt_full

```
356 4714            IF NOT inhibit_io THEN
35E 4715              trace (mmc$ap_ptf_remove, 1);
35E 4716              mmp$remove_page_from_jws (pfti, ijle_p, mcount, rcount);
390 4717              jmp$unlock_ajl (ijle_p);
446 4718            IFEND;
446 4719          IFEND;
446 4720        IFEND;
446 4721        pti := pti + 1;
446 4722        IF pti = mmv$pt_length THEN
450 4723          pti := 0;
454 4724        IFEND;
454 4725      FOREND;
458 4726
458 4727      pt_full_status := mmc$pfs_failed;
458 4728      mmv$pt_full_trace.failed := mmv$pt_full_trace.failed + 1;
458 4729
458 4730
458 4731    PROCEND mmp$process_page_table_full;
```

PAGE TABLE FULL MANAGER
build_asid_list

```
o  4733 { PURPOSE:
o  4734 {   The purpose of this procedure is to examine the 32 entries in the page
o  4735 {   table where the page table full occured.  A list of the unique ASIDs is
o  4736 {   generated and sorted in order of easiest to reassign to hardest to reassign.
o  4737 {
o  4738 { INPUT:
o  4739 {   pt_full_asid: This parameter specifies the ASID that
o  4740 {         hashed into the page table full area.
o  4741 {   pt_full_index: This parameter is the index of the first
o  4742 {         entry in the page table of the full area.
o  4743 {   asid_list_p: This parameter points to the array for the
o  4744 {         ASID list in the page table full area.
o  4745 {
o  4746 { OUTPUT:
o  4747 {   asid_list: (built in array pointed to by asid_list_p) List of ASID for potential
o  4748 {         reassignment. List is sorted in order of least overhead of reassignment.
o  4749 {   max_asid_list_index: index of the last entry in the ASID list.
o  4750
o  4751
o  4752 {  Define type definition for the list of ASIDs returned by BUILD_ASID_LIST.
o  4753
o  4754    TYPE
o  4755      asid_list_entry = record
o  4756        index: asid_list_index_type,
o  4757        sort_key: asid_list_key,
o  4758        asid: ost$asid,
o  4759        aste_p: ^mmt$active_segment_table_entry,
o  4760      recend,
o  4761      asid_list_index_type = 0 .. 33,
o  4762      asid_list_key = 0 .. osc$max_page_frames * 16,
o  4763      asid_list = array [1 .. 33] of asid_list_entry,
o  4764      reassign_pass = 1 .. 3;
o  4765
o  4766
o  4767    PROCEDURE build_asid_list
o  4768      (     pt_full_asid: ost$asid;
o  4769            pt_full_index: ost$page_table_index;
o  4770            asid_list_p: ^asid_list;
o  4771       VAR max_asid_list_index: asid_list_index_type);
o  4772
o  4773      VAR
o  4774        asid: ost$asid,
o  4775        asid_list_index: asid_list_index_type,
o  4776        asid_list_index_max: asid_list_index_type,
o  4777        aste_p: ^mmt$active_segment_table_entry,
o  4778        done: boolean,
o  4779        ijle_p: ^jmt$initiated_job_list_entry,
o  4780        inhibit_reassign: boolean,
o  4781        pass: reassign_pass,
o  4782        pfti: mmt$page_frame_index,
o  4783        pti: integer,
o  4784        pti_offset: 0 .. 31,
o  4785        save_index: 0 .. 32;
o  4786
o  4787
```


PAGE TABLE FULL MANAGER
build_asid_list

```
o  4788 { If the image file is still being processed, discard all image pages. Pages that are still needed
o  4789 { will be faulted for again and new page table entries will be made.
o  4790
o  4791      IF mmv$image_file.active THEN
10 4792        mmp$free_image_pages_mtr;
18 4793      IFEND;
18 4794
18 4795
18 4796 { Build an array of the ASIDs in the part of the PT that are involved in the
18 4797 { PT full condition. Ignore special ASIDs used by NOS or SSR. Skip free entries in the page table. If free
18 4798 { entries are encountered then either 1) PT full condition has cleared, or 2) more than one entry
18 4799 { is required in the page table. There is (currently) no way to distinguish between the cases. Since the
18 4800 { PT full condition may have been caused by a request that requires MULTIPLE entries to be made in the
18 4801 { page table, this routine cannot quit when a free entry is found.
18 4802 {!NOTE: current algorithm does NOT force out pages in the AVAIL MODIFIED queue. This is ok until we start
18 4803 { to keep large numbers of pages in this queue. Then we should write pages unless IO in inhibited.
18 4804 { Note that FFFF cannot be reassigned (special significance to hardware) and the ASID of the page table
18 4805 { cannot be reassigned (requires mods to preset_memory routine.)
18 4806
18 4807      asid_list_index_max := 0;
18 4808      pti := pt_full_index;
18 4809
18 4810      FOR pti_offset := 0 TO mmv$pt_search - 1 DO
2C 4811
2C 4812        IF pti_offset <> (mmv$pt_search - 1) THEN
36 4813          asid := mmv$pt_p^ [pti].pageid.asid;
36 4814          pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
36 4815          IF (asid <> 0) AND (pfti >= LOWERBOUND (mmv$pft_p^)) AND (pfti <= UPPERBOUND (mmv$pft_p^)) THEN
8A 4816            aste_p := mmv$pft_p^ [pfti].aste_p;
AA 4817          ELSE
AA 4818            asid := 0;
AE 4819          IFEND;
B2 4820        ELSE
B2 4821          asid := pt_full_asid;
B2 4822          mmp$aste_pointer (asid, aste_p);
CC 4823          pfti := 0;
D0 4824        IFEND;
D0 4825
D0 4826        IF (asid = 0) OR (asid = 0ffff(16)) OR (asid = dsv$ssr_sdte.ste.asid) OR
106 4827             (asid = mtv$monitor_segment_table.st [0].ste.asid) OR (aste_p = NIL) THEN
106 4828          {Do nothing - cant change asid}
106 4829        ELSEIF (NOT jmp$ijl_block_valid (aste_p^.ijl_ordinal)) OR
15C 4830             (jmv$ijl_p.block_p^ [aste_p^.ijl_ordinal.block_number].index_p^ [aste_p^.ijl_ordinal.block_index].
15C 4831             entry_status = jmc$ies_entry_free) THEN
15C 4832          IF pfti <> 0 THEN
160 4833            trace (mmc$ap_ba_freed_terj, 1);
160 4834            mmp$delete_pt_entry (pfti, TRUE);
17E 4835            mmp$relink_page_frame (pfti, mmc$pq_free);
192 4836          IFEND;
196 4837        ELSE
196 4838          IF (aste_p^.sfid.residence = gfc$tr_system) AND (aste_p^.queue_id < mmc$pq_job_base) OR
1B4 4839               (pti_offset = (mmv$pt_search - 1)) THEN
1B4 4840            inhibit_reassign := FALSE;
1BC 4841          ELSE
1BC 4842 [!] mmp$get_inhibit_io_status (aste_p^.ijl_ordinal, FALSE {lock ajl} , inhibit_reassign, ijle_p);
```

PAGE TABLE FULL MANAGER
build_asid_list

```
1FE  4843          IFEND;
1FE  4844
1FE  4845          IF NOT inhibit_reassign THEN
206  4846            asid_list_index := 1;
206  4847            asid_list_p^ [asid_list_index_max + 1].asid := asid;
206  4848            WHILE (asid <> asid_list_p^ [asid_list_index].asid) DO
22A  4849              asid_list_index := asid_list_index + 1;
22A  4850            WHILEND;
240  4851
240  4852            IF asid_list_index > asid_list_index_max THEN
244  4853
244  4854              IF aste_p^.sfid.residence = gfc$tr_job THEN
252  4855                pass := 1;
258  4856              ELSEIF aste_p^.queue_id >= mmc$pq_job_base THEN
264  4857                pass := 2;
26A  4858              ELSE
26A  4859                pass := 3;
26C  4860              IFEND;
26C  4861
26C  4862
26C  4863              asid_list_index_max := asid_list_index_max + 1;
26C  4864              asid_list_p^ [asid_list_index_max].sort_key := aste_p^.pages_in_memory + pass *
26C  4865                    osc$max_page_frames;
26C  4866              asid_list_p^ [asid_list_index_max].aste_p := aste_p;
26C  4867              asid_list_p^ [asid_list_index_max].index := asid_list_index_max;
2A4  4868            IFEND;
2A6  4869          IFEND;
2A6  4870        IFEND;
2A6  4871
2A6  4872        pti := pti + 1;
2A6  4873        IF pti = mmv$pt_length THEN
2B4  4874          pti := 0;
2B8  4875        IFEND;
2B8  4876
2B8  4877      FOREND;
2BC  4878
2BC  4879
2BC  4880  { Sort the ASID list. List is sorted in the order of 'easiest to reassign'. See procedure
2BC  4881  { mmp$change_asid for more details on 'easy to reassign'.
2BC  4882
2BC  4883      done := asid_list_index_max <= 1;
2BC  4884      WHILE NOT done DO
2CA  4885        done := TRUE;
2CA  4886        FOR asid_list_index := 1 TO asid_list_index_max - 1 DO
2D6  4887          IF asid_list_p^ [asid_list_p^ [asid_list_index].index].
310  4888                sort_key > asid_list_p^ [asid_list_p^ [asid_list_index + 1].index].sort_key THEN
310  4889            save_index := asid_list_p^ [asid_list_index].index;
310  4890            asid_list_p^ [asid_list_index].index := asid_list_p^ [asid_list_index + 1].index;
310  4891            asid_list_p^ [asid_list_index + 1].index := save_index;
310  4892            done := FALSE;
31C  4893          IFEND;
31C  4894        FOREND;
320  4895      WHILEND;
326  4896
326  4897      max_asid_list_index := asid_list_index_max;
```


PAGE TABLE FULL MANAGER
build_asid_list

```
326  4898
326  4899    PROCEND build_asid_list;
```

```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

        0  4901 { PURPOSE:
        0  4902 {   This procedure is called by the PAGE_TABLE_FULL_HANDLER to reassign an ASID
        0  4903 {   That appears in the part of the page table that is full.
        0  4904 {
        0  4905 { INPUT:
        0  4906 {    old_asid:    ASID to be reassigned
        0  4907 {    old_aste_p:  pointer to the AST table entry for the segment
        0  4908 {
        0  4909 { OUTPUT:
        0  4910 {    new_asid:    newly assigned ASID
        0  4911 {    new_asti:    newly assigned AST index
        0  4912 {    new_aste_p:  newly assigned AST pointer
        0  4913 {    pt_full_status: status of reassign request
        0  4914 {       mmc$pfs_asid_reassigned
        0  4915 {       mmc$pfs_failed
        0  4916 {
        0  4917
        0  4918
        0  4919   PROCEDURE reassign_asid
        0  4920     (    old_asid: ost$asid;
        0  4921          old_aste_p: ^mmt$active_segment_table_entry;
        0  4922      VAR new_asid: ost$asid;
        0  4923      VAR new_asti: mmt$ast_index;
        0  4924      VAR new_aste_p: ^mmt$active_segment_table_entry;
        0  4925      VAR pt_full_status: mmt$pt_full_status);
        0  4926
        0  4927     VAR
        0  4928       count: 1 .. 32,
        0  4929       found: boolean,
        0  4930       pti: integer,
        0  4931       mpt_status: mmt$make_pt_entry_status,
        0  4932       new_pte_p: ^ost$page_table_entry,
        0  4933       new_sva: ost$system_virtual_address,
        0  4934       temp_sva: ost$system_virtual_address,
        0  4935       old_pte_p: ^ost$page_table_entry,
        0  4936       old_sva: ost$system_virtual_address,
        0  4937       pfte: mmt$page_frame_table_entry,
        0  4938       pfte_p: ^mmt$page_frame_table_entry,
        0  4939       pfti: mmt$page_frame_index,
        0  4940       stop_pfti: mmt$page_frame_index,
        0  4941       try_count: 0 .. 4;
        0  4942
        0  4943
        0  4944 { Build the list of PFTIs for the segment being changed.
        0  4945
        0  4946     old_sva.asid := old_asid;
        4  4947     old_sva.offset := 0;
        4  4948     mmp$initialize_find_next_pfti (old_sva, 7ffffff0(16), include_partial_pages, psc_all, old_aste_p, pfti);
       40  4949
       40  4950
       40  4951 { Try several times to assign a new ASID for the segment.
       40  4952
       40  4953     FOR try_count := 1 TO UPPERVALUE (try_count) DO
       4A  4954       mmp$assign_asid (new_asid, new_asti, new_aste_p);
       62  4955       new_sva.asid := new_asid;
```



```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

       62  4956       new_aste_p^ := old_aste_p^;
       7C  4957       new_aste_p^.pages_in_memory := 0;
       7C  4958       mmp$reset_find_next_pfti (pfti);
       DA  4959
       DA  4960
       DA  4961 { Make page table entries using the new ASID for each page of the segment
       DA  4962 { that is currently in memory. Note that for a short time both the old and new PT entries
       DA  4963 { will exist. CAUTIONS:
       DA  4964 {     - If PT full conditions occur during mmp$make_pt_entry, PT entries that are in the AVAIL
       DA  4965 {       queue may be freed.
       DA  4966 {     - Since mmp$make_pt_entry may freed PT entries, some of the PFTI in the pfti array
       DA  4967 {       may already have been freed before this routine trys to change the ASID.
       DA  4968 { If a page table entry cannot be made and the page is in the AVAIL queue, the entry can be skipped.
       DA  4969 { Later in this procedure, the page will be deleted and linked to the FREE queue.
       DA  4970 { Note that the ^PFT entry passed to mmp$make_pt_entry is a dummy entry.
       DA  4971 {
       DA  4972 { When this loop exits (either normally or abnormally, each page frame in the PFTI list  will
       DA  4973 { have 0, 1 or 2 page table entries for the page frame:
       DA  4974 {    - both old and new (this is the state for ALL page frames except those in the AVAIL queue)
       DA  4975 {    - old only. New was not made because PT full occurred.
       DA  4976 {    - old only. New was made but was subsequently deleted by mmp$make_pt_entry due to PT full.
       DA  4977 {    - new only. Old was deleted by PT full processing in mmp$make_pt_entry after new entry was made.
       DA  4978 {    - no entries. Combination of previous 2 entries.
       DA  4979
       DA  4980       trace (mmc$ap_rea_called, 1);
       DA  4981
       DA  4982     /reassign_loop/
       DA  4983       BEGIN
       DA  4984         changing_asid := new_asid;
       DA  4985         changing_aste_p := new_aste_p;
       DA  4986         WHILE pfti <> 0 DO
      104  4987           pfte_p := ^mmv$pft_p^ [pfti];
      104  4988           IF pfte_p^.queue_id = mmc$pq_free THEN
      12E  4989             trace (mmc$ap_rea_in_free, 1);
      12E  4990             mmp$delete_last_pfti_from_array;
      14C  4991           ELSE
      14C  4992             trace (mmc$ap_rea_make_pt_entry, 1);
      14C  4993             new_sva.offset := pfte_p^.sva.offset;
      14C  4994             mmp$make_pt_entry (new_sva, pfti, new_aste_p, ^pfte, mpt_status);
      190  4995             IF (mpt_status <> mmc$mpt_done) THEN
      198  4996               IF (pfte_p^.queue_id <> mmc$pq_avail) THEN
      1A0  4997                 EXIT /reassign_loop/;
      1A4  4998               IFEND;
      1A4  4999               trace (mmc$ap_rea_mpte_fail, 1);
      1B2  5000             ELSE
      1B2  5001               old_pte_p := ^mmv$pt_p^ [pfte_p^.pti];
      1B2  5002               new_pte_p := ^mmv$pt_p^ [pfte.pti];
      1B2  5003               new_pte_p^.u := FALSE;
      1B2  5004               new_pte_p^.v := old_pte_p^.v;
      1E0  5005             IFEND;
      1E0  5006           IFEND;
      1E0  5007           mmp$find_next_pfti (pfti);
      22C  5008         WHILEND;
      23C  5009         changing_asid := 0;
      23C  5010
```

```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

     23C  5011
     23C  5012
     23C  5013
     23C  5014 { Page table entries have been made for all pages of the segment. Now locate
     23C  5015 { all segment table entries that have the ASID and change it to the new ASID.
     23C  5016
     23C  5017         mmp$change_asid (old_aste_p, old_asid, new_asid, new_asti);
     264  5018
     264  5019
     264  5020 { If the ASID being changed belongs to mainframe wired, save the new ASID. This is required in case
     264  5021 { the system crashes while the next couple of blocks of CYBIL statements are being executed. System
     264  5022 { recovery
     264  5023 { must be able to locate PT entries that belong to mainframe wired.
     264  5024
     264  5025         IF old_asid = mmv$mf_wired_asid.current THEN
     26C  5026           mmv$mf_wired_asid.new := new_asid;
     274  5027         IFEND;
     274  5028
     274  5029
     274  5030 { Now delete the PT entries that have the old ASID and update the PFT with the
     274  5031 { new segment info. The correct value of the 'used' and 'modified' bits are captured
     274  5032 { here from the old entries and copied to the new PT entries.
     274  5033
     274  5034         trace (mmc$ap_rea_ok, 1);
     274  5035         mmp$reset_find_next_pfti (pfti);
     2D4  5036         WHILE pfti <> 0 DO
     2E4  5037           pfte_p := ^mmv$pft_p^ [pfti];
     2E4  5038           IF pfte_p^.queue_id <> mmc$pq_free THEN
     30A  5039             temp_sva.offset := pfte_p^.sva.offset;
     30A  5040             temp_sva.asid := new_asid;
     30A  5041             #HASH_SVA (temp_sva, pti, count, found);
     326  5042             IF NOT found THEN
     33A  5043
     33A  5044 { Set changing_asid to a non-zero value to indicate that page table full processing is occurring.
     33A  5045 { This page frame is in the available queue and a new page table entry could not be made, so
     33A  5046 { delete_pt_entry will unlink the page from the segment.  Because page frames that had a new page
     33A  5047 { table entry made for them will not be unlinked from the segment, the debug code in unlink page
     33A  5048 { from segment must not be executed.  (If this is the last page in the segment, the ast.pages_in_memory
     33A  5049 { will be zero, but the ast.pft_link will still have the links to be copied to the new ast entry later.)
     33A  5050
     33A  5051               changing_asid := 1;
     33A  5052               mmp$delete_pt_entry (pfti, TRUE);
     352  5053               changing_asid := 0;
     352  5054               pfte_p^.sva.asid := new_asid;
     352  5055               trace (mmc$ap_rea_ok1, 1);
     352  5056               mmp$relink_page_frame (pfti, mmc$pq_free);
     384  5057             ELSE
     384  5058               mmp$delete_pt_entry (pfti, FALSE);
     398  5059               pfte_p^.sva.asid := new_asid;
     398  5060               trace (mmc$ap_rea_ok2, 1);
*WARN*    5061               IF mmv$pt_p^ [pfte_p^.pti].m THEN
     3CC  5062                 mmv$pt_p^ [pti].m := TRUE;
     3D6  5063               IFEND;
     3D6  5064               IF mmv$pt_p^ [pfte_p^.pti].u THEN
     3F0  5065                 mmv$pt_p^ [pti].u := TRUE;
```



```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

     3FA  5066               IFEND;
     3FA  5067               pfte_p^.pti := pti;
     3FA  5068               pfte_p^.aste_p := new_aste_p;
     406  5069             IFEND;
     40A  5070           ELSE
     40A  5071             trace (mmc$ap_rea_ok3, 1);
     40A  5072             pfte := pfte_p^;
     41E  5073             pfte_p^.sva.asid := new_asid;
     41E  5074             #HASH_SVA (pfte_p^.sva, pti, count, found);
     42C  5075             IF found THEN
     440  5076               trace (mmc$ap_rea_ok4, 1);
     440  5077               pfte_p^.pti := pti;
     440  5078               pfte_p^.aste_p := new_aste_p;
     440  5079               mmp$delete_pt_entry (pfti, FALSE);
     46A  5080             IFEND;
     46A  5081             pfte_p^ := pfte;
     478  5082           IFEND;
     478  5083           mmp$find_next_pfti (pfti);
     4D2  5084         WHILEND;
     4F2  5085
     4F2  5086 { Copy the ast.pft_link information again; delete_pt_entry may have changed the links.
     4F2  5087
     4F2  5088         new_aste_p^.pft_link := old_aste_p^.pft_link;
     4F2  5089         old_aste_p^.pft_link.fwd := 0;
     4F2  5090         old_aste_p^.pft_link.bkw := 0;
     4F2  5091
     4F2  5092         mmp$free_asid (old_asid, old_aste_p);
     51A  5093
     51A  5094
     51A  5095 { If the ASID that was changed belonged to mainframe wired, update the
     51A  5096 { mainframe-wired-asid record.
     51A  5097
     51A  5098         IF old_asid = mmv$mf_wired_asid.current THEN
     522  5099           mmv$mf_wired_asid.current := mmv$mf_wired_asid.new;
     522  5100           mmv$mf_wired_asid.new := 0;
     52E  5101         IFEND;
     52E  5102
     52E  5103         pt_full_status := mmc$pfs_asid_reassigned;
     52E  5104         RETURN; { <-------}
     534  5105
     534  5106       END /reassign_loop/;
     534  5107       changing_asid := 0;
     534  5108
     534  5109
     534  5110 { Control gets here only if a page table entry could not be made for the new ASID.
     534  5111 { Delete all PT entries made with the new ASID and try again. NOTE: If an entry cannot be found,
     534  5112 { then it must have been in the AVAIL queue and was deleted by mmp$make_pt_entry as a
     534  5113 { result of page table full processing.
     534  5114
     534  5115       stop_pfti := pfti;
     534  5116       mmp$reset_find_next_pfti (pfti);
     592  5117       trace (mmc$ap_rea_fail, 1);
     592  5118       WHILE pfti <> stop_pfti DO
     5A8  5119         trace (mmc$ap_rea_fail, 1);
     5A8  5120         pfte_p := ^mmv$pft_p^ [pfti];
```

```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

     5A8  5121          new_sva.offset := pfte_p^.sva.offset;
     5A8  5122          #HASH_SVA (new_sva, pti, count, found);
     5E2  5123          IF found THEN
     5F6  5124            trace (mmc$ap_rea_fail2, 1);
     5F6  5125            pfte := pfte_p^;
     60A  5126            pfte_p^.pti := pti;
     60A  5127            pfte_p^.sva.asid := new_asid;
     60A  5128            pfte_p^.aste_p := new_aste_p;
     60A  5129            mmp$delete_pt_entry (pfti, FALSE);
     632  5130            pfte_p^ := pfte;
     63C  5131          IFEND;
     63C  5132          mmp$find_next_pfti (pfti);
     68E  5133        WHILEND;
     6A4  5134        new_aste_p^.pft_link.fwd := 0;
     6A4  5135        new_aste_p^.pft_link.bkw := 0;
     6A4  5136        mmp$free_asid (new_asid, new_aste_p);
     6C8  5137      FOREND;
     6D0  5138
     6D0  5139
     6D0  5140 { Control gets here only if all attempts to reassign the ASID fail. Return
     6D0  5141 { bad status and exit.
     6D0  5142
     6D0  5143      trace (mmc$ap_rea_quit, 1);
     6D0  5144      pt_full_status := mmc$pfs_failed;
     6D0  5145
     6D0  5146    PROCEND reassign_asid;
       0  5147
       0  5148 MODEND mmm$asid_page_table_manager

**** I=$05578173AS0102D19890821T183254  L=ZZXXLIST  B=LGO  DA=NONE  LO=R  RC=NONE  OPT=SCHED  EL=F  LF=CS612  PAD=0
```



```
PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

          ERROR     LINE  TEXT
WARNING   CY 821    5061   Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY
   ****   1    warning diagnostic
```

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
active	516	4791							
ajl_ordinal	1280	849	883	2086	2279	2300	4071	4077	4078/S
		4713	4717	4842					
ajl_ordinal	2265	2279/M	2279	2279/S	2279/S	2279			
ajl_ordinal	2297	2300/M	2301	2305/S	2305/S	2306			
ajl_ordinal	4621	4713/M	4713	4713/S	4713/S	4713			
ajl_ordinal	4767	4842/M	4842	4842/S	4842/S	4842			
ajlo	2083	2086/M	2087/S	2090/S	2090/S				
ajlo	2265	2279/P	2279/M						
ajlo	2272	2279/P							
ajlo	2294	2303/P	2306/M						
ajlo	4621	4713/P							
ajlo	4621	4713/P	4713/M						
ajlo	4621	4717/M	4717/S	4717/S	4717/S				
ajlo	4767	4842/P							
ajlo	4767	4842/P	4842/M						
amc$file_byte_limit	968	971	973						
amt$file_byte_address	971	932							
amt$file_limit	973	936							
asid	340	3911	3950/P	3996/M					
asid	413	4290	4290	4328/M	4395	4402/M	4575	4577	4708
		4813							
asid	548	4328	4395	4577/M	4652/P	4671/P	4683	4946/M	4955/M
		5040/M	5054/M	5059/M	5073/M	5127/M			
asid	2386	4078/M	4131	4132/M	4148	4150/M	4154	4155/M	4826
		4827							
asid	3787	4690/M	4691/M						
asid	3866	3911/M	3916/P						
asid	3976	3987/P	3996						
asid	4189	4212/P	4213/P	4214/P					
asid	4758	4680/P	4683	4690	4847/M	4848	4848		
asid	4774	4813/M	4815	4818/M	4821/M	4822/P	4826	4826	4826
		4827	4847	4848	4848				
asid_list	4763	4631	4770						
asid_list_entry	4755	4763							
asid_list_index	4775	4846/M	4848/S	4848/S	4849/M	4849	4852	4886	4887/S
		4888/S	4889/S	4890/S	4890/S	4891/S			
asid_list_index_max	4776	4807/M	4847/S	4852	4863/M	4863	4864/S	4866/S	4867/S
		4867	4883	4886	4897				
asid_list_index_type	4761	4629	4630	4645	4756	4771	4775	4776	
asid_list_key	4762	4757							
asid_list_p	4770	4847/M	4848	4848	4864/M	4866/M	4867/M	4887	4887/S
		4888	4888/S	4889	4890/M	4890	4891/M		
asidl	4629	4679/M	4680/S	4680/S	4683/S	4690/S	4693/S		
asidlmax	4630	4671/P	4677						
asidt	4631	4671/P	4679	4680/P	4680/P	4683	4690	4693	
aste_p	1973	4413	4579/M	4816	5068/M	5078/M	5128/M		
aste_p	3868	3910/M	3911	3912/M	3917				
aste_p	3940	3950/P	3952	3955	3960/M				
aste_p	3977	3982	3982	3988	3995/M	3996/M	3997/M	4000/M	
aste_p	4034	4057/P	4058	4061/P	4070	4071/P	4107	4108	4145

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
aste_p	4246	4344	4344	4344	4344	4344/M	4344/M	4344/S	4344
		4344/M							
aste_p	4249	4335	4344/P	4346/M	4346	4347			
aste_p	4371	4422/M	4422/M	4422	4422	4422			
aste_p	4376	4413/M	4414	4417/M	4417	4422/P			
aste_p	4464	4472	4472	4472	4478	4479/M	4480/M	4482/S	4483
		4484/M							
aste_p	4497	4501/M	4507/M	4517	4518	4518			
aste_p	4759	4680/P	4866/M						
aste_p	4777	4816/M	4822/P	4827	4829/P	4830/S	4830/S	4838	4838
		4842/P	4854	4856	4864	4866			
asti	931	4072/M							
asti	2355	4133/M							
asti	3867	3909/M							
asti	3943	3950/P	3951/S						
asti_that_cannot_be_freed	4186	4211							
asti_that_cannot_be_freed	4632	4652/P	4653/P						
b	842	851	851						
b	2040	2050	2051						
b	2079	2085	2085						
b	2156	2163	2164						
b	2265	2278	2278						
b	4033	4071	4071						
b	4621	4713	4713	4717	4717				
b	4767	4842	4842						
bc	2079	2085/M	2085	2085					
bc	2157	2160/M	2161	2165					
bc	2265	2278/M	2278	2278					
bc	4621	4713/M	4713	4713	4717/M	4717	4717		
bc	4767	4842/M	4842	4842					
bkw	397	4344	4344	4344/M	4344/S	4344/M	4344	4344/M	4422/M
		4422	4422/M	4422	4422	4422/S	4422/M	4422	4468
		4472	4480/M	4482/S	4483/M	4483	4484/M	4501/M	4501
		4503/M	4503	4506	4509/S	4513/M	4518	5090/M	5135/M
block_index	382	2065/S	2274/S	4057/S	4713/S	4830/S	4842/S		
block_number	381	2065/S	2075/S	2274/S	4057/S	4713/S	4829/S	4830/S	4842/S
block_p	3609	2065	2075	2274	4057	4713	4829	4830	4842
build_asid_list	4767	4671	4899						
c	419	4293/M	4312/M	4325/M	4325	4407/M	4449/M		
cbc_p	4254	4288/M	4291/M	4291	4292				
cell_p	4040	4146/M	4147						
changed_asid	3782	4692/M	4692						
changing_asid	3800	4343	4422	4517	4575	4984/M	5009/M	5051/M	5053/M
		5107/M							
changing_aste_p	3801	4579	4985/M						
clear	2148	2092/M	2118/M	2280/M	4713/M	4717/M	4842/M		
clear_continue_bits	4429	4312	4407	4453					
count	2145	2085/M	2085	2092	2092/M	2092	2116	2117/M	2117
		2167/M	2167	2278/M	2278	2280	2280/M	2280	4713/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES						
		4713	4713	4713/M	4713	4717/M	4717	4717	4717/M
		4717	4842/M	4842	4842	4842/M	4842		
count	4246	4312							
count	4255	4286/M	4290	4290	4295/M	4295	4301/S	4301/S	4308
count	4371	4407							
count	4377	4387	4406	4407/P					
count	4431	4442							
count	4549	4564	4585						
count	4633	4658	4659						
count	4928	5041	5074	5122					
current	532	5025	5098	5099/M					
delayed_swapin_work	1310	4087/M	4087						
dfc$command_record_bytes	1380	1388							
dfc$division_overwrite_words	1367	1395							
dfc$esm_command_record_size	1388	1396							
dfc$esm_header_record_size	1389	1396							
dfc$esm_maintenance_buf_size	1368	1399							
dfc$esm_memory_base_shift	1374	1396	1397	1397					
dfc$header_record_bytes	1379	1389							
dfc$max_esm_memory_size	1369	1398							
dfc$max_number_of_mainframes	1376	1361							
dfc$min_data_record_bytes	1384	1395							
dfc$min_esm_division_size	1394	1398							
dft$mainframe_set	1361	1311	1312	1432	1433				
dmt$system_file_id	527	517	1343						
done	4778	4883/M	4884	4884	4885/M	4892/M			
dsv$ssr_sdte	3570	4826							
end_of_table_seen	4190	4195/M	4204	4207/M					
entry_status	1279	4831							
failed	3783	4728/M	4728						
fcount	4256	4311/P	4312/P						
fde_p	842	849/M	849						
fde_p	844	849/P	851/P						
fde_p	867	889/M	890						
fde_p	4033	4071/P	4071/P						
fde_p	4033	4071/M	4071						
fde_p	4041	4071/P	4072/M	4094	4103	4117			
file_entry_index	354	849	879	4071					
file_hash	356	849	849	878	890	4000/M	4071	4071	
file_hash	929	849	890	4071					
file_kind	928	4094							
fix_ste_loop	4123	4123	4138						
flags	922	4103	4117						
found	4257	4270	4271	4311/P	4313				
found	4378	4387	4392						
found	4634	4658							
found	4929	5041	5042	5074	5075	5122	5123		
free_pt_entry_in_avail_queue	4541	4311	4600						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
freed	4546	4587/M	4598/M						
fwd	398	4344	4344	4344	4344/M	4344/M	4422	4422/S	4422/M
		4422	4422/M	4422	4422/M	4422	4468	4472	4478
		4479/M	4482/M	4500	4503/S	4507/M	4507	4509/M	4509
		4512/M	4518	5089/M	5134/M				
gfc$fde_size	906	849	879	4071					
gfc$fde_table_base	904	849	879	905	4071				
gfc$fk_catalog	1005	1017							
gfc$fk_job_local_file	1007	1016							
gfc$fk_monitor_only_unnamed	1010	4094							
gfc$fm_mass_storage_file	1020	945							
gfc$fm_served_file	1021	948							
gfc$monitor_interlocks	863	850	4071						
gfc$tr_job	367	849	882	4071	4854				
gfc$tr_system	367	849	881	4070	4071	4107	4838		
gfp$mtr_get_fde_p	865	849	894	4071					
gfp$mtr_get_locked_fde_p	842	854	4071						
gft$allocation_unit_size	979	934							
gft$attach_count	984	925	926						
gft$fde_flags	954	922							
gft$file_desc_entry_p	911	867	4041						
gft$file_descriptor_entry	919	911	924	2016					
gft$file_descriptor_index	364	354							
gft$file_kind	1001	928	1013						
gft$file_media	1020	944							
gft$locked_file_desc_entry_p	2016	844							
gft$open_count	1050	927	1066						
gft$queue_status	1061	937							
gft$segment_lock_info	1065	930							
gft$signature_lock	1026	920							
gft$system_file_identifier	353	343	527	842	865	2464	2581	2783	
gft$table_residence	367	355	875						
gft$transfer_unit_size	990	935							
global_template_file	958	4103	4117						
hash	842	849/M	849/M						
hash	872	878/M	885/M						
hash	4033	4071/M	4071/M						
hcount	4258	4270	4275						
i	4246	4312							
i	4371	4407							
i	4434	4442							
i#program_error	1264	849	891	2092	2114	2280	4071	4713	4717
		4842							
id	2146	2085	2085/M	2092	2113	2161	2165/M	2278	2278/M
		2280	4713	4713/M	4713	4717	4717/M	4717	4842
		4842/M	4842						
id	3733	3736/S	3736/S						
id	3865	3881/S	3881/S	3887/S	3887/S	3905/S	3905/S	3913/S	3913/S

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
id	3939	3947/S	3947/S						
id	3975	4003/S	4003/S						
id	4033	4086/S	4086/S	4095/S	4095/S	4105/S	4105/S	4109/S	4109/S
		4112/S	4112/S						
id	4185	4197/S	4197/S						
id	4246	4309/S	4309/S	4318/S	4318/S				
id	4541	4570/S	4570/S	4572/S	4572/S				
id	4621	4676/S	4676/S	4678/S	4678/S	4704/S	4704/S	4715/S	4715/S
id	4767	4833/S	4833/S						
id	4919	4980/S	4980/S	4989/S	4989/S	4992/S	4992/S	4999/S	4999/S
		5034/S	5034/S	5055/S	5055/S	5060/S	5060/S	5071/S	5071/S
		5076/S	5076/S	5117/S	5117/S	5119/S	5119/S	5124/S	5124/S
		5143/S	5143/S						
ijl_ordinal	336	4057/P	4061/P	4829/P	4830/S	4830/S	4842/P		
ijl_ordinal	1966	4713/P							
ijl_ordinal	2061	2065/S	2065/S						
ijl_ordinal	2073	2075/S							
ijl_ordinal	2265	2274/S	2274/S						
ijl_ordinal	2266	2274/P	2279/P						
ijl_ordinal	4033	4057/S	4057/S						
ijl_ordinal	4621	4713/P	4713/P						
ijl_ordinal	4621	4713/S	4713/S						
ijl_ordinal	4767	4829/S							
ijl_ordinal	4767	4842/P	4842/P						
ijl_ordinal	4767	4842/S	4842/S						
ijle_p	842	849							
ijle_p	843	849/P							
ijle_p	866	883							
ijle_p	2062	2065/M							
ijle_p	2080	2086	2088/P						
ijle_p	2265	2274/M							
ijle_p	2265	2279	2279/P						
ijle_p	2269	2274/P	2275	2279/P					
ijle_p	2292	2300	2302/P						
ijle_p	4033	4057/M							
ijle_p	4033	4071/P							
ijle_p	4033	4071							
ijle_p	4042	4057/P	4061/P	4071/P	4075	4076/M	4077	4078/S	4087/M
		4087							
ijle_p	4621	4713/P	4713	4713/P					
ijle_p	4621	4713/M							
ijle_p	4621	4713	4713/P						
ijle_p	4621	4717	4717/P						
ijle_p	4635	4713/P	4716/P	4717/P					
ijle_p	4767	4842/P	4842	4842/P					
ijle_p	4767	4842/M							
ijle_p	4767	4842	4842/P						
ijle_p	4779	4842/P							
ijlo	2265	2279/P							
ijlo	2293	2302/P							
ijlo	4621	4713/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijlo	4767	4842/P							
in_use	337	3907	3912/M	3955	3960/M	3982	3995/M	4210	4335
in_use	2193	2087	2090/M	2090	2279/M	2279	2305/M	2305	4713/M
		4713	4717	4717/M	4717	4842/M	4842		
inc	3734	3736							
inc	3865	3881	3887	3905	3913				
inc	3939	3947							
inc	3975	4003							
inc	4033	4086	4095	4105	4109	4112			
inc	4185	4197							
inc	4246	4309	4318						
inc	4541	4570	4572						
inc	4621	4676	4678	4704	4715				
inc	4767	4833							
inc	4919	4980	4989	4992	4999	5034	5055	5060	5071
		5076	5117	5119	5124	5143			
include_pages_in_dump	344	4347							
include_partial_pages	3475	4948/P							
index	3786	4686/M	4686	4687	4688/M	4690/S	4691/S		
index	4756	4679	4867/M	4887/S	4888/S	4889	4890/M	4890	4891/M
index_p	3622	2065	2075	2274	4057	4713	4829	4830	4842
inhibit_io	2268	2275/M	2276						
inhibit_io	4621	4713/M	4713						
inhibit_io	4636	4713/P	4714						
inhibit_io	4767	4842/M	4842						
inhibit_reassign	4780	4840/M	4842/P	4845					
initial_pfti	4543	4569							
initial_pti	4542	4563							
iot$io_error	1999	1344	1974						
iot$transfer_count	2871	2859							
jmc$dsw_job_asid_changed	1422	4088							
jmc$highest_prio_age_interval	2532	2523	2533						
jmc$highest_service_accumulator	1820	1821							
jmc$highest_service_factor_valu	2566	2559							
jmc$ies_entry_free	1595	4831							
jmc$ies_job_swapped	1600	1609							
jmc$ies_swapin_in_progress	1599	1608							
jmc$inhibit_memory_manager_io	1652	2275	4713	4842					
jmc$iss_idle_tasks_initiated	1615	1642							
jmc$iss_swapin_io_complete	1640	1643							
jmc$iss_swapin_requested	1636	1643							
jmc$iss_swapout_complete	1635	1642							
jmc$iss_swapped_io_cannot_init	1626	1653							
jmc$iss_swapped_no_io	1617	1652							
jmc$keyword_offset_maximum	1837	2524							
jmc$kjl_maximum_entries	43	36	37	1772					
jmc$kol_maximum_entries	53	38							
jmc$lock_ajl	2100	2087	2088/P	2090	2279/P	2279	2302/P	2305	4713/P
		4713	4717	4717/P	4717	4842/P	4842		
jmc$max_active_jobs	34	2505	2513	2514					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$max_ajl_ord	35	30	34	1408					
jmc$max_dispatching_control	1559	1563							
jmc$max_dispatching_priority	1481	1441	1444	1445					
jmc$max_ijl_index_count	390	3620							
jmc$maximum_job_classes	1750	1753							
jmc$maximum_job_count	50	43							
jmc$maximum_output_count	60	53							
jmc$maximum_service_classes	1853	1856							
jmc$min_dispatching_control	1558	1562							
jmc$null_ajl_ordinal	30	2279	2301	4077	4713	4842			
jmc$null_service_class	1846	1847							
jmc$priority_aging_interval_max	2523	2520							
jmc$priority_p1	1495	1442							
jmc$priority_p10	1504	1443							
jmc$priority_p14	1508	1443							
jmc$priority_p8	1502	1442							
jmc$reserved_ajls	39	34							
jmc$service_accumulator_maximum	1812	1809							
jmc$service_factor_value_max	2559	2556							
jmc$system_default_offset	1836	1837							
jmc$system_supplied_name_size	1888	1885							
jmc$unlimited_offset	1833	1822	2534						
jmp$assign_ajl_with_lock	2314	2279	2302	4713	4842				
jmp$free_ajl_with_lock	2103	2088	4717						
jmp$get_ijle_p	2061	2067	2274	4057	4713	4842			
jmp$ijl_block_valid	2072	2077	4829						
jmp$ijl_block_valid	2073	2075/M	4829/M						
jmp$lock_ajl_with_lock	2291	2279	2309	4713	4842				
jmp$unlock_ajl	2079	2094	4717						
jmt$active_job_list	2200	2178							
jmt$active_job_list_entry	2192	2200							
jmt$ajl_ordinal	1408	1280	2083	2272	2294	2297	2319	3558	3595
jmt$delayed_swapin_work	1425	1310	1429	4088					
jmt$dispatching_control	1529	2488							
jmt$dispatching_control_index	1562	1519	1529						
jmt$dispatching_controls	1532	1530							
jmt$dispatching_priority	1441	1292	1520	1521	1522	1534	2436	2438	
jmt$ijl_block_index	386	382	3622						
jmt$ijl_block_number	385	381	3610	3611					
jmt$ijl_dispatching_control	1518	1293							
jmt$ijl_entry_status	1595	1279							
jmt$ijl_ordinal	380	336	1299	1327	1903	1904	1966	2061	2073
		2194	2266	2293	2316	3547	3559	3633	
jmt$ijl_p	3608	3603							
jmt$ijl_page_fault_count	1669	1664	1665	1666					
jmt$ijl_page_stats	1663	1659							
jmt$ijl_service_class_stats	1657	1314							
jmt$ijl_statistics	1702	1313							
jmt$ijl_swap_count	1678	1674	1675						
jmt$ijl_swap_counts	1673	1333	1660						
jmt$ijl_swap_status	1613	1282	1283	1284					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$initiated_job_list_block	3619	3625							
jmt$initiated_job_list_entry	1276	843	866	1928	2062	2080	2104	2195	2210
		2269	2292	3512	3546	3560	3622	4042	4635
		4779							
jmt$initiated_job_list_p	3625	3609							
jmt$input_file_location	1792	1787							
jmt$job_abort_disposition	1801	1785							
jmt$job_class	1753	1338							
jmt$job_mode	1756	1295							
jmt$job_priority	1761	1335	1336	2497	2498	2499	2500		
jmt$job_recovery_disposition	1804	1786							
jmt$kjl_index	1772	1281							
jmt$maximum_active_jobs	2505	2482							
jmt$priority_aging_interval	2520	2490							
jmt$queue_file_ijl_information	1784	1320							
jmt$scheduling_data	1326	1304							
jmt$scheduling_priority	2496	2489							
jmt$service_accumulator	1809	1328	1329	1330	2480	2481			
jmt$service_class_index	1856	1339	2473	2483					
jmt$service_class_name	2538	2475	2476						
jmt$service_factor_value	2556	2484							
jmt$service_factors	2552	2484							
jmt$swap_data	1342	1306							
jmt$swapout_reasons	1859	1334							
jmt$swapped_job_entry	1874	1351	1929						
jmt$system_supplied_name	1885	1277							
jmt$task_time_slice	1572	1552	1553						
jmt$time_slice_values	1551	1536	2449						
jmv$ajl_p	2178	2087	2090/M	2090	2279/M	2279	2305/M	2305	4713/M
		4713	4717	4717/M	4717	4842/M	4842		
jmv$ijl_p	3603	2065	2075	2274	4057	4713	4829	4830	4842
jmv$null_ijl_ordinal	3633	4059/P	4137/P						
job_fixed_asid	1289	2279/P	2302/P	4075	4076/M	4713/P	4842/P		
jsc$isqi_swapped_io_completed	1908	1910							
jsc$isqi_swapped_io_not_init	1907	1910							
jst$changed_asid_entry	1951	1942							
jst$ijl_swap_queue_id	1907	1902							
jst$ijl_swap_queue_link	1901	1288							
jst$io_control_information	1915	1307							
jst$swap_file_descriptor	1927	1308							
jst$swapped_page_descriptor	1936	1934							
jst$swapped_page_descriptors	1933	1930							
last_pfti_index	669	2248	2249	2253	3530	3531	3536	4958	4958
		4958	5007	5007	5007	5035	5035	5035	5083
		5083	5083	5116	5116	5116	5132	5132	5132
last_purge_time	3872	3903/M	3907						
last_sva	3785	4664/M							
lock	842	851							
lock	2037	2050							
lock	2079	2085	2085	2085/M	2085/M	2085			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	2079	2092	2092	2092/M	2092	2092/M			
lock	2110	2113	2116	2117/M	2117	2119/M			
lock	2153	2161	2163	2165/M	2167/M	2167			
lock	2265	2278	2278	2278/M	2278/M	2278			
lock	2265	2280	2280	2280/M	2280	2280/M			
lock	2267	2277							
lock	4033	4071							
lock	4621	4713							
lock	4621	4713	4713	4713/M	4713/M	4713	4717	4717	4717/M
		4717/M	4717						
lock	4621	4713	4713	4713/M	4713	4713/M	4717	4717	4717/M
		4717	4717/M						
lock	4767	4842							
lock	4767	4842	4842	4842/M	4842/M	4842			
lock	4767	4842	4842	4842/M	4842	4842/M			
locked	1104	851	2050	4071					
locked	2144	2085	2163	2278	4713	4717	4842		
m	421	4327/M	4566	5061	5062/M				
max_asid_list_index	4771	4897/M							
max_segnum	4043	4118/M	4120/M	4127	4128				
mcount	4637	4716/P							
mmc$ap_assign	3744	3913/P							
mmc$ap_assign_specific	3745	3947/P							
mmc$ap_ast_reset	3742	3905/P							
mmc$ap_ba_freed_terj	3775	4833/P							
mmc$ap_casid_global	3749	4112/P							
mmc$ap_casid_job	3750	4109/P							
mmc$ap_casid_monitor	3747	4095/P							
mmc$ap_casid_swapped_job	3746	4086/P							
mmc$ap_casid_template	3748	4105/P							
mmc$ap_free_aste	3743	4003/P							
mmc$ap_low_asids	3740	3887/P							
mmc$ap_mpte_full	3752	4309/P							
mmc$ap_mpte_rec1	3754	4570/P							
mmc$ap_mpte_rec2	3755	4572/P							
mmc$ap_mpte_recovered	3753	4318/P							
mmc$ap_no_asids	3741	3881/P							
mmc$ap_ptf_called	3756	4676/P							
mmc$ap_ptf_failed	3758	4704/P							
mmc$ap_ptf_remove	3759	4715/P							
mmc$ap_ptf_tried	3757	4678/P							
mmc$ap_rea_called	3760	4980/P							
mmc$ap_rea_fail	3768	5117/P							
mmc$ap_rea_fail1	3769	5119/P							
mmc$ap_rea_fail2	3770	5124/P							
mmc$ap_rea_in_free	3761	4989/P							
mmc$ap_rea_make_pt_entry	3774	4992/P							
mmc$ap_rea_mpte_fail	3762	4999/P							
mmc$ap_rea_ok	3763	5034/P							
mmc$ap_rea_ok1	3764	5055/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$ap_rea_ok2	3765	5060/P							
mmc$ap_rea_ok3	3766	5071/P							
mmc$ap_rea_ok4	3767	5076/P							
mmc$ap_rea_quit	3771	5143/P							
mmc$ap_reclaim_asids	3751	4197/P							
mmc$assign_active_null	2621	2622							
mmc$cell_pointer	2720	2725							
mmc$debug	67	3915	3949	3986	3999	4391			
mmc$first_loader_predefined_seg	83	85	3803						
mmc$heap_pointer	2721	2729							
mmc$kw_asid	2646	2682							
mmc$kw_clear_space	2644	2669							
mmc$kw_current_segment_length	2643	2663							
mmc$kw_error_exit_procedure	2645	2673							
mmc$kw_gl_key	2645	2667							
mmc$kw_hardware_attributes	2647	2676							
mmc$kw_inheritance	2647	2684							
mmc$kw_max_segment_length	2644	2665							
mmc$kw_preset_value	2646	2671							
mmc$kw_ps_transfer_size	2648	2692							
mmc$kw_ring_numbers	2642	2658							
mmc$kw_segment_access_control	2646	2680							
mmc$kw_segment_number	2643	2661							
mmc$kw_shadow_segment	2648	2686							
mmc$kw_software_attributes	2645	2678							
mmc$kw_wired_segment	2648	2689							
mmc$mpt_done	634	4348	4995						
mmc$mpt_page_already_exists	635	4272							
mmc$mpt_page_table_full	634	4314							
mmc$num_loader_predefined_segs	84	85							
mmc$pfs_asid_reassigned	676	4682	5103						
mmc$pfs_failed	675	4727	5144						
mmc$pfs_input_asid_reassigned	676	4684							
mmc$pq_avail	437	483	4569	4996					
mmc$pq_free	436	495	4574/P	4835/P	4988	5038	5056/P		
mmc$pq_job_base	484	4058	4108	4838	4856				
mmc$pq_job_fixed	477	484	496						
mmc$pq_job_working_set	479	496	497	4712					
mmc$pq_shared_first	485	4711							
mmc$pq_shared_first_site	487	491							
mmc$pq_shared_last	492	4711							
mmc$pq_shared_num_sites	488	491							
mmc$pq_shared_other	446	486							
mmc$pq_shared_site_01	448	487							
mmc$pq_shared_site_25	472	492							
mmc$pq_shared_task_service	441	485							
mmc$pq_swapped_io_error	475	495							
mmc$pq_wired	439	482	4145						
mmc$segment_fault_processor_id	3176	3230							
mmc$sequence_pointer	2720	2727							
mmc$ssk_none	2815	2787							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmc$ssk_segment_number	2816	2785
mmk$monitor_base	251	91 98 105 130 151 165 168 171
		174 177 180 183 186 189 192 196
		199 202 209 213 216 222
mmp$asid	2216	4212
mmp$assign_asid	3865	3923 4954
mmp$assign_specific_asid	3939	3962
mmp$aste_pointer	2227	3916 3987 4822
mmp$asti	2221	3950 4652
mmp$change_asid	4033	4166 4213 5017
mmp$delete_last_pfti_from_array	2233	2237 4990
mmp$delete_pt_entry	4371	4425 4573 4580 4834 5052 5058 5079 5129
mmp$find_next_pfti	2240	2258 5007 5083 5132
mmp$free_asid	3975	4005 4214 5092 5136
mmp$free_image_pages_mtr	2262	4792
mmp$get_inhibit_io_status	2265	2284 4713 4842
mmp$get_max_sdt_pointer	2327	2335 4125
mmp$get_sdt_entry_p	3464	3466/M
mmp$initialize_find_next_pfti	3473	4948
mmp$link_page_to_segment	4461	4344 4487
mmp$make_pt_entry	4246	4350 4994
mmp$nudge_periodic_call	3488	3493 3889
mmp$process_page_table_full	4621	4731
mmp$purge_all_cache_map	3814	3904
mmp$purge_all_cache_map_proc	3829	3820 3904
mmp$purge_all_map_proc	3845	3838 4163
mmp$purge_all_page_seg_map	3832	4163
mmp$reclaim_ast_entries	4185	3882 4220 4653
mmp$relink_page_frame	3505	4574 4835 5056
mmp$remove_page_from_jws	3511	4716
mmp$reset_find_next_pfti	3519	3542 4958 5035 5116
mmp$unlink_page_from_segment	4495	4422 4524
mmt$active_segment_table	348	3658
mmt$active_segment_table_entry	333	349 1939 1973 2228 3477 3651 3801 3868
		3871 3940 3944 3977 3980 4034 4249 4376
		4464 4497 4625 4759 4777 4921 4924
mmt$ast_index	1083	931 1350 1954 2216 2222 2355 3862 3862
		3867 3943 4037 4186 4191 4624 4632 4923
mmt$async_work_list	3647	3641
mmt$attribute_keyword	2642	2657
mmt$continue_bit_count	510	3802 4254
mmt$eoi_state	1092	933
mmt$global_page_queue_index	495	660
mmt$global_page_queue_list_ent	650	660
mmt$hardware_attribute_set	2711	2677
mmt$hardware_attributes	2699	2711
mmt$image_file	514	3665
mmt$job_page_queue_index	496	661 1876
mmt$job_page_queue_list	661	1305
mmt$link	396	334 647 1963 1964
mmt$lock_segment_status	2795	2586

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmt$locked_page	1985	1969
mmt$mainframe_wired_asid	531	3804 3804
mmt$make_pt_entry_status	634	4251 4931
mmt$max_sdt	2365	2369
mmt$max_sdt_p	2369	2329 4045
mmt$max_sdtx	2610	2614
mmt$memory_reserve_request	2005	1298
mmt$page_age	1992	1972 1996 1996
mmt$page_frame_index	402	398 398 670 1916 1918 1919 1920 2007
		2008 2211 2241 2245 3478 3505 3511 3520
		3524 4248 4372 4462 4543 4551 4640 4782
		4939 4940
mmt$page_frame_queue_id	497	342 1917 1967 3506
mmt$page_frame_table	1978	3698
mmt$page_frame_table_entry	1962	1937 1978 4250 4379 4463 4496 4550 4639
		4937 4938
mmt$page_queue_list_entry	646	651 661
mmt$page_selection_criteria	3484	3476
mmt$pfti_array	666	3690
mmt$pt_full_status	675	4626 4925
mmt$pt_full_trace_info	3780	3794
mmt$sdtx_stream_data	2593	2589
mmt$segment_access_condition	3203	3231
mmt$segment_access_rights	2759	2585
mmt$segment_access_state	2765	2580
mmt$segment_descriptor	2352	2362 2366 3464 3570 3578 3588 4046
mmt$segment_descriptor_extended	2578	2607 2611
mmt$segment_inheritance	2628	2582 2685
mmt$segment_pointer_kind	2720	2724
mmt$segment_reservation_state	2805	2583
mmt$shadow_info	2780	2587
mmt$shadow_reference_info	2828	2462
mmt$shadow_segment_kind	2815	2784
mmt$software_attribute_set	2713	2584 2679
mmt$software_attributes	2707	2713
mmt$xcb_page_wait_info	2839	2448
mmv$aptm_trace	3795	3736/M 3736 3881/M 3881 3887/M 3887 3905/M 3905
		3913/M 3913 3947/M 3947 4003/M 4003 4086/M 4086
		4095/M 4095 4105/M 4105 4109/M 4109 4112/M 4112
		4197/M 4197 4309/M 4309 4318/M 4318 4570/M 4570
		4572/M 4572 4676/M 4676 4678/M 4678 4704/M 4704
		4715/M 4715 4833/M 4833 4980/M 4980 4989/M 4989
		4992/M 4992 4999/M 4999 5034/M 5034 5055/M 5055
		5060/M 5060 5071/M 5071 5076/M 5076 5117/M 5117
		5119/M 5119 5124/M 5124 5143/M 5143
mmv$ast_p	3658	3902 3907 3907 3910 3951 4208 4210 4210
		4213/P 4214/P
mmv$async_work	3641	3888/M 4650 4651/M
mmv$continue_bit_count_p	3802	4288 4312/M 4312 4312 4407/M 4407 4407 4447/M
		4447 4448
mmv$image_file	3665	4791

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmv$max_template_segment_number	3803	4118							
mmv$mf_wired_asid	3804	5025	5026/M	5098	5099/M	5099	5100/M		
mmv$multiple_caches	3675	3819	3904						
mmv$multiple_page_maps	3683	3819	3904						
mmv$number_free_astes	3805	3879	3880	3883	3893/M	3893	3946/M	3946	3993/M
		3993							
mmv$page_table_miss_count	4243	4301/M	4301	4317/M	4317				
mmv$pages_to_dump_p	3806	4347/M							
mmv$pft_p	3698	4344/M	4386	4422/M	4422/M	4482/M	4503/M	4509/M	4568
		4568	4569	4571	4576	4577/M	4578/M	4579/M	4581/M
		4708	4708	4710	4815	4815	4816	4987	5037
		5120							
mmv$pfti_array_p	3690	2235/S	2235/M	2248	2248	2249	2249	2250/M	2250
		2251	2251/S	2253	2253	3527/M	3527	3528	3528/S
		3530	3530	3531	3531	3532/M	3532	3533	3533/S
		3536	3536	4958/M	4958	4958	4958/S	4958	4958
		4958/M	4958	4958	4958/S	4958	4958	4958	4958
		4990/S	4990/M	5007	5007	5007/M	5007	5007	5007/S
		5007	5007	5007	5007	5035/M	5035	5035	5035/S
		5035	5035	5035/M	5035	5035	5035/S	5035	5035
		5035	5035	5083	5083	5083/M	5083	5083	5083/S
		5083	5083	5083	5083	5116/M	5116	5116	5116/S
		5116	5116	5116/M	5116	5116	5116/S	5116	5116
		5116	5116	5132	5132	5132/M	5132	5132	5132/S
		5132	5132	5132	5132				
mmv$pt_full_trace	3794	4664/M	4665/M	4686/M	4686	4687	4688/M	4690/S	4690/M
		4691/S	4691/M	4692/M	4692	4694/M	4694	4728/M	4728
mmv$pt_length	3706	4277	4297	4312	4407	4445	4592	4661	4722
		4873							
mmv$pt_p	3712	4287	4312/M	4389	4407/M	4449/M	4565	4706	4813
		4814	5001	5002	5061	5062/M	5064	5065/M	
mmv$pt_search	3807	4290	4290	4308	4312/P	4564	4705	4810	4812
		4839							
mmv$test_reassign_asid	3719	3999							
mmv$time_changed_global_asid	3808	4111/M							
mmv$time_changed_template_asid	3809	4104/M							
mmv$time_to_call_mem_mgr	3499	3490/M	3889/M						
monitor_lock	921	851/P	4071/P						
mpt_status	4251	4272/M	4314/M	4348/M					
mpt_status	4931	4994/P	4995						
mtc$job_fixed_segment	687	849	883	4071	4078/S	4148			
mtp$error_stop	2033	3884	3918	3953	3956	3983	3989	4336	4344
		4344	4393	4397	4415	4422	4469	4473	4519
		4583							
mtp$set_interlock	2037	851	2058	4071					
mtt$monitor_interlock	1099	921	2037						
mtv$monitor_segment_table	3577	4078/M	4146	4149	4150/M	4827			
mtv$nos_segment_table_p	3587	4153	4154	4155/M					
nat$received_message_descriptor	2855	2848	2857						
nat$received_message_list	2847	2430							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
new	533	5026/M	5099	5100/M					
new	3789	4691/M							
new_asid	4036	4076	4078	4132	4150	4155			
new_asid	4623	4680/P	4691						
new_asid	4922	4954/P	4955	4984	5017/P	5026	5040	5054	5059
		5073	5127	5136/P					
new_aste_p	4625	4680/P							
new_aste_p	4924	4954/P	4956/M	4957/M	4985	4994/P	5068	5078	5088/M
		5128	5134/M	5135/M	5136/P				
new_asti	4037	4072	4133						
new_asti	4624	4680/P							
new_asti	4923	4954/P	5017/P						
new_pte_p	4932	5002/M	5003/M	5004/M					
new_sva	4933	4955/M	4993/M	4994/P	5121/M	5122			
next_aste_index	4191	4194/M	4202/M	4202	4203	4208/M	4210/S	4210/S	4211
		4212/P	4213/S	4214/S					
next_asti	3862	3900/M	3900	3901	3902/M	3907/S	3907/S	3909	3910/S
		4194							
nlc$cc_connect_confirm	2887	2878							
nlc$cc_connect_request	2886	2876							
nlc$cc_expedited_data	2892	2878							
nlc$cc_max_pdu_kind	2894	2897							
nlc$channel_connection_pdu	2910	2862							
nlc$channelnet_pdu	2910	2864							
nlt$cc_pdu_kind	2897	2875							
nlt$cc_seq#_or_connect_time	2874	2863							
nlt$cc_sequence_number	2900	2879							
nlt$device_identifier	2907	2858							
nlt$pdu_type	2910	2861							
null_sva	3817	3822	3823						
null_sva	3835	3840							
null_sva	3865	3904	3904						
null_sva	4033	4163							
number_astes_reclaimed	4192	4196/M	4201	4201	4215/M	4215			
offset	549	4330	4395	4947/M	4993/M	4993	5039/M	5039	5121/M
		5121							
offset	842	849/M	849/M	849					
offset	874	879/M	886/M	889					
offset	4033	4071/M	4071/M	4071					
old	3788	4690/M							
old_asid	4035	4075	4131	4149	4154				
old_asid	4920	4946	5017/P	5025	5092/P	5098			
old_aste_p	4921	4948/P	4956	5017/P	5088	5089/M	5090/M	5092/P	
old_pte_p	4935	5001/M	5004						
old_sva	4936	4946/M	4947/M	4948/P					
osc$call_instruction	3090	3098							
osc$data_read	3089	3098							
osc$free_running_clock_maximum	506	503							
osc$invalid_ring	566	606							
osc$max_fault_contents	3243	3237							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$max_name_size	2542	2546	2549						
osc$max_number_of_processors	2133	2128							
osc$max_page_frames	406	335	402	648	654	667	668	669	1345
		1346	1875	1877	4693	4762	4865		
osc$max_page_size	1205	1201							
osc$max_page_table_entries	407	410							
osc$max_ring	565	606	607						
osc$max_segment_length	589	612	2590	2621					
osc$max_status_condition_code	810	806	822						
osc$max_string_size	826	829	832	837					
osc$max_tasks	1045	1042							
osc$maximum_offset	588	589	609	609	610				
osc$maximum_processor_id	3115	3111							
osc$maximum_processors	2137	2133							
osc$maximum_segment	587	608							
osc$min_page_size	1204	1201							
osc$min_ring	564	607							
osc$pr_base_constant	704	2085	2092	2113	2160	2278	2280	4713	4713
		4717	4717	4842	4842				
osc$purge_all_cache	745	3822	3904						
osc$purge_all_page_seg_map	754	3823	3840	3904	4163				
osc$pva_purge_segment_cache	747	4147							
osc$task_time_slice_maximum	1583	1586							
osk$system_class	303	287	288	289	290	291	292	293	
ost$asid	552	340	532	533	548	1289	1941	1952	1953
		2217	2221	2227	2315	2386	2683	3788	3789
		3800	3800	3866	3976	4035	4036	4189	4623
		4758	4768	4774	4920	4922			
ost$binary_unique_name	1113	923							
ost$cp_time	1690	1658	1703	2447					
ost$cp_time_value	1688	1331	1691	1692	2460				
ost$cs_lock	1215	2428							
ost$debug_code	3089	3077							
ost$debug_list	3085	2989							
ost$debug_list_entry	3076	3085							
ost$debug_mask	3095	2988							
ost$exchange_package	2938	2415							
ost$execute_privilege	2399	2381	2394						
ost$execution_control_block	2414	2328	2440	3463	3549	4047			
ost$flags	2995	2945							
ost$frame_descriptor	3053	3068							
ost$free_running_clock	503	339	939	1300	1301	1302	1303	1337	1347
		1348	1349	1523	1535	2197	2446	3808	3808
		3809	3809						
ost$global_task_id	1036	941	1029	1294	1323	2425	2426	3127	3330
ost$key_lock	595	2387	2668						
ost$key_lock_value	601	598	3012	3014					
ost$keypoint_class	3027	2958	3029						
ost$keypoint_mask	3029	2961							
ost$minimum_save_area	3063	2950	3038	3224					
ost$monitor_condition	2914	2921							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$monitor_conditions	2921	2951	2955	3043	3299	3313			
ost$monitor_fault	3220	3169							
ost$monitor_fault_contents	3237	3233							
ost$name	2549	2474	2538	3262					
ost$p_register	3010	2939	3064	3291	3297				
ost$page_id	412	422							
ost$page_size	1201	1182	3725						
ost$page_table	426	3712	4259						
ost$page_table_entry	417	426	1938	4260	4380	4552	4641	4932	4935
ost$page_table_index	410	426	1970	4430	4769				
ost$paging_statistics	1726	1704	2455						
ost$processor_id	3111	2418	3105						
ost$processor_id_set	3105	2417							
ost$processor_model_number	1131	1115							
ost$processor_serial_number	1209	1114							
ost$pva	617	2983	3001	3015	3221	3314			
ost$read_privilege	2402	2382	2395						
ost$register_number	3006	2980	3049	3057	3058	3059			
ost$ring	606	618	2384	2385	2579	2659	2660	3000	
ost$ring1_termination_reason	3123	2451							
ost$segment	608	619	940	2662	2786	2978	3079	3464	4043
ost$segment_access_control	2392	2681							
ost$segment_descriptor	2379	2353							
ost$segment_length	612	2664	2666	2688	2690	2693	3474		
ost$segment_offset	609	549	620	2594	3080	3082			
ost$stack_frame_save_area	3037	3071	3257						
ost$status	794	2675	3292						
ost$status_condition	818	789							
ost$status_condition_code	822	797	818						
ost$string	835	798							
ost$string_size	829	836							
ost$system_flag	3398	3394							
ost$system_virtual_address	547	1975	3473	3650	3785	4247	4622	4933	4934
		4936							
ost$task_index	1042	1037	1076	1077	3562				
ost$task_time_slice	1586	1572							
ost$top_of_stack_pointer	2998	2990							
ost$trap_enable	3032	2947	3288						
ost$user_condition	2924	2931							
ost$user_conditions	2931	2948	2953	3041	3070	3260	3300		
ost$valid_relative_pointer	615	946	949	2444	2445				
ost$valid_ring	607	2990							
ost$virtual_machine_identifier	3020	2941	2943	3065					
ost$write_privilege	2405	2383	2396						
ost$x_register	3007	2980	3049						
osv$cpus_logically_on	2128	2085	2092	2112	2159	2278	2280	3837	4163
		4713	4713	4717	4717	4842	4842		
osv$page_size	3725	4332	4396	4567	4707	4814			
osv$time_to_check_asyn	3501	3491/M	3889/M						
pageid	422	4290	4290	4328/M	4329/M	4395	4395	4402/M	4575

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		4577	4708	4813					
pagenum	414	4329/M	4395						
pages_in_memory	335	3982	4210	4344	4346/M	4346	4414	4417/M	4417
		4422	4472	4517	4864	4957/M			
pass	3784	4694/M	4694						
pass	4638	4693/M	4694/S	4694/S					
pass	4781	4855/M	4857/M	4859/M	4864				
pft_link	334	4344	4344	4344	4344/M	4344/M	4344/S	4344	4344/M
		4422/M	4422/M	4422	4422	4472	4472	4478	4479/M
		4480/M	4482/S	4483	4484/M	4501/M	4507/M	4518	4518
		5088/M	5088	5089/M	5090/M	5134/M	5135/M		
pfte	4550	4576/M	4581						
pfte	4937	4994/P	5002/S	5072/M	5081	5125/M	5130		
pfte_p	4246	4344	4344	4344/M					
pfte_p	4250	4334/M	4344/P						
pfte_p	4371	4422	4422	4422/S	4422	4422	4422	4422/S	4422
		4422/M	4422/M						
pfte_p	4379	4386/M	4387	4388	4392	4395	4395	4413	4422/P
pfte_p	4463	4468	4468	4483/M					
pfte_p	4496	4500	4501	4503/S	4503	4506	4507	4509/S	4509
		4512/M	4513/M						
pfte_p	4639	4710/M	4711	4711	4712	4713/P			
pfte_p	4938	4987/M	4988	4993	4996	5001/S	5037/M	5038	5039
		5054/M	5059/M	5061/S	5064/S	5067/M	5068/M	5072	5073/M
		5074	5077/M	5078/M	5081/M	5120/M	5121	5125	5126/M
		5127/M	5128/M	5130/M					
pfti	2245	2247/M	2248	2248	2251/M	2254/M	2256		
pfti	3524	3528/M	3530	3530	3533/M	3537/M	3539		
pfti	4246	4344	4344	4344	4344				
pfti	4248	4311/P	4332	4344/P	4347/S				
pfti	4372	4386/S	4396						
pfti	4462	4479	4480	4482	4484				
pfti	4551	4567/M	4568	4568	4569/S	4569	4571/S	4573/P	4574/P
		4576/S	4577/S	4578/S	4579/S	4580/P	4581/S		
pfti	4640	4707/M	4708	4708	4710/S	4716/P			
pfti	4782	4814/M	4815	4815	4816/S	4823/M	4832	4834/P	4835/P
pfti	4919	4958/M	4958	4958/M	4958	4958/M	4958	5035/M	5035
		5035/M	5035	5035/M	5035	5116/M	5116	5116/M	5116
		5116/M	5116						
pfti	4919	5007/M	5007	5007/M	5007	5007/M	5007	5083/M	5083
		5083/M	5083	5083/M	5083	5132/M	5132	5132/M	5132
		5132/M	5132						
pfti	4939	4948/P	4958/P	4986	4986	4987/S	4994/P	5007/P	5035/P
		5036	5036	5037/S	5052/P	5056/P	5058/P	5079/P	5083/P
		5115	5116/P	5118	5118	5120/S	5129/P	5132/P	
pfti_first	667	3527	3528/S	4958	4958/S	5035	5035/S	5116	5116/S
pfti_index	668	2235/S	2248	2249	2250/M	2250	2251/S	2253	3527/M
		3530	3531	3532/M	3532	3533/S	3536	4958/M	4958
		4958/M	4958	4958/S	4958	4958	4990/S	5007	5007/M
		5007	5007/S	5007	5007	5035/M	5035	5035/M	5035
		5035/S	5035	5035	5083	5083/M	5083	5083/S	5083

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		5083	5116/M	5116	5116/M	5116	5116/S	5116	5116
		5132	5132/M	5132	5132/S	5132	5132		
pftis	670	2235/M	2251	3528	3533	4958	4958	4990/M	5007
		5035	5035	5083	5116	5116	5132		
pmc$kill_task_flag	3398	3414							
pmc$max_signal_contents	3381	3375							
pmc$max_task_id	3136	3133							
pmt$condition_identifier	3210	3204							
pmt$cpu_model_number	1191	1180	1187						
pmt$cpu_serial_number	1194	1181	1186						
pmt$initialization_value	1259	938	2672						
pmt$signal	3337	3331							
pmt$signal_contents	3375	3339							
pmt$signal_id	3342	3338							
pmt$task_id	3133	2442	3128						
psc_all	3484	4948/P							
pt_full_asid	4768	4821							
pt_full_index	4769	4808							
pt_full_status	4626	4681/P	4682	4684/M	4727/M				
pt_full_status	4925	5103/M	5144/M						
pt_p	4259	4287/M	4290	4290	4293/M	4325	4333/M		
pte	4260	4324/M	4325/M	4326/M	4327/M	4328/M	4329/M	4331/M	4333
pte_p	4380	4389/M	4395	4395	4396	4401/M	4402/M		
pte_p	4552	4565/M	4566	4567	4575	4577			
pte_p	4641	4706/M	4707	4708					
pti	1970	4334/M	4388	4392	4571	4578/M	5001/S	5002/S	5061/S
		5064/S	5067/M	5077/M	5126/M				
pti	4246	4312/M	4312/M	4312	4312	4312/M	4312/S	4312/S	4312/S
		4312/S							
pti	4261	4289/M	4290/S	4290/S	4291/S	4291/S	4292/S	4293/S	4296/M
		4296	4297	4298/M	4310	4311/P	4325/S	4333/S	4334
pti	4371	4407/M	4407/M	4407	4407	4407/M	4407/S	4407/S	4407/S
		4407/S							
pti	4381	4387	4388/M	4389/S	4392	4407/P			
pti	4435	4441/M	4443/M	4443	4444	4445/M	4447/S	4447/S	4448/S
		4449/S							
pti	4553	4563/M	4565/S	4571	4578	4586	4591/M	4591	4592
		4593/M							
pti	4642	4658	4659/M	4659	4660	4661/M	4661	4671/P	4706/S
		4721/M	4721	4722	4723/M				
pti	4783	4808/M	4813/S	4814/S	4872/M	4872	4873	4874/M	
pti	4930	5041	5062/S	5065/S	5067	5074	5077	5122	5126
pti_offset	4643	4705							
pti_offset	4784	4810	4812	4839					
queue_id	342	4058	4108	4145	4838	4856			
queue_id	1967	4569	4711	4711	4712	4988	4996	5038	
rcount	4644	4716/P							
reassign_asid	4919	4680	5146						
reassign_loop	4982	4982	4997	5106					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES						
reassign_pass	4764	3784	4638	4781				
reclaim_astes	3648	3888/M	4650	4651/M				
reclaim_loop	4199	4199	4218					
residence	355	849	877	4070	4071	4107	4838	4854
residence	842	849/M	849	849				
residence	875	877/M	881	882				
residence	4033	4071/M	4071	4071				
rma	423	4331/M	4396	4567	4707	4814		
save_index	4785	4889/M	4891					
save_pti	4262	4310/M	4312/P					
sdt_offset	2444	2333	3467	4125				
sdt_p	2329	2333/M						
sdt_p	4033	4125/M						
sdt_p	4045	4125/P	4131	4132/M	4133/M			
seg	842	849/M	849/M	849				
seg	873	880/M	883/M	889				
seg	4033	4071/M	4071/M	4071				
segment_link	1964	4344	4344	4344/M	4344/M	4422	4422	4422/S
		4422	4422	4422	4422/S	4422/M	4422	4422/M
		4468	4468	4482/M	4483/M	4500	4501	4503/S
		4503	4506	4507	4509/S	4509/M	4509	4512/M
segment_table_length	2978	4126						
segnum	3464	3467						
segnum	4044	4126/M	4127	4128/M	4130	4130	4131/S	4132/S
		4135/M	4135	4148	4149/S	4150/S	4153	4154/S
sfid	343	4000/M	4070	4071/P	4107	4838	4854	
sfid	842	849/P						
sfid	842	849	849	849	849			
sfid	865	877	878	879	890			
sfid	4033	4071/P						
sfid	4033	4071	4071	4071	4071			
sft$counter	1736	1705	1706					
sft$file_space_limit_kind	2824	2588						
sort_index	4645	4677	4679/S					
sort_key	4757	4693	4864/M	4888	4888			
st	2366	4131	4132/M	4133/M				
st	3578	4078/M	4149	4150/M	4827			
st	3588	4153	4154	4155/M				
starting_pti	4263	4270	4275/M	4275	4276	4277/M	4277	4289
status	2265	2279/P						
status	2298	2303/P						
status	4621	4713/P						
status	4767	4842/P						
ste	2353	4078/M	4131	4132/M	4149	4150/M	4154	4155/M
		4827						
stop_pfti	4940	5115/M	5118	5118				
sva	1975	4387	4395	4395	4577/M	4993	5039	5054/M
		5073/M	5074	5121	5127/M			
sva	4247	4270	4328	4330				
sva	4622	4652/P	4658	4664	4671/P	4683		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES						
swap_status	1282	2275	4713	4842				
syc$ucr_condition	3248	3259						
syc$user_defined_condition	3249	3261						
syt$monitor_flag	3155	3140						
syt$monitor_flags	3140	2416						
syt$monitor_status	787	2298	2320					
temp_sva	4934	5039/M	5040/M	5041				
time_freed	339	3907	3997/M					
timestamp	3781	4665/M						
tmc$broken_task_fault_id	3176	3226						
tmc$btc_invalid_a0	3274	3295						
tmc$btc_invalid_p	3274	3295						
tmc$btc_mcr_traps_disabled	3275	3296						
tmc$btc_mf_traps_disabled	3274	3294						
tmc$btc_mntr_fault_buffer_full	3273	3294						
tmc$btc_system_error	3276	3290						
tmc$btc_ucr_traps_disabled	3275	3296						
tmc$dummy_fault	3177	3232						
tmc$flag_available_31	3411	3415						
tmc$fnx_continue	3554	4137/P						
tmc$fnx_job	3554	4061/P						
tmc$fnx_system	3554	4059/P						
tmc$maximum_monitor_faults	3181	3172						
tmc$maximum_signals	3391	3388						
tmc$maximum_system_task_id	3424	3427						
tmc$mcr_fault	3176	3228						
tmc$signal_available_63	3373	3384						
tmc$stid_null_task	3430	3427						
tmp$clear_lock	2110	2092	2123	2280	4713	4717	4842	
tmp$find_next_xcb	3545	4059	4061	4137				
tmp$set_lock	2153	2085	2171	2278	4713	4717	4842	
tmt$broken_task_condition	3273	3289						
tmt$broken_task_monitor_fault	3287	3227						
tmt$find_next_xcb_state	3556	3548	4048					
tmt$fnx_search_type	3554	3545	3557					
tmt$mcr_faults	3312	3229						
tmt$monitor_fault_buffer	3166	2453						
tmt$monitor_fault_buffers	3172	3167	3168	3169				
tmt$monitor_fault_identifiers	3175	3225	3301					
tmt$ptl_lock	2141	2110	2153	2204				
tmt$signal	3329	3324						
tmt$signal_buffer	3321	2454						
tmt$signal_buffers	3388	3322	3323	3324				
tmt$system_flags	3394	2429						
tmt$system_task_id	3427	2420						
tmt$task_queue_link	1075	1068	1971					
tmv$ptl_lock	2204	2085/P	2092/P	2278/P	2280/P	4713/P	4713/P	4717/P
		4842/P	4842/P					
trace	3732	3737	3881	3887	3905	3913	3947	4003
		4095	4105	4109	4112	4197	4309	4318

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

PAGE TABLE FULL MANAGER
reassign_asid - Used with page_table_full_handler

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		4572	4676	4678	4704	4715	4833	4980	4989
		4992	4999	5034	5055	5060	5071	5076	5117
		5119	5124	5143					
try_count	4941	4953	4953						
u	420	4326/M	5003/M	5064	5065/M				
unlink_page_from_segment	4373	4421							
v	418	4324/M	4401/M	5004/M	5004				
xcb_p	2328	2333	2333						
xcb_p	3463	3466	3467						
xcb_p	4033	4125	4125						
xcb_p	4047	4059/P	4061/P	4070	4085	4124	4124	4125/P	4126
		4137/P							
xcb_state	4048	4059/P	4061/P	4137/P					
xcount	4545	4585/M	4597/M						
xp	2415	4126							
xpfti	2241	2256/M							
xpfti	3520	3539/M							
xpfti	4919	4958/M	5035/M	5116/M					
xpfti	4919	5007/M	5083/M	5132/M					
xpti	4246	4312							
xpti	4371	4407							
xpti	4430	4441							
xpti	4544	4586/M							
zaste_p	3871	3916/P	3917						
zaste_p	3944	3951/M	3952						
zaste_p	3980	3987/P	3988						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter


```
            2 MODULE mmm$deadstart_initialization;
            3
            5 {
            6 {  PURPOSE:
            7 {     This module contains procedures used during deadstart to initialize
            8 {     memory manager tables.
            9 {
           10

      o  3459 {External procedures used by this module.
      o  3460
      o  3461
      o  3462   PROCEDURE [XREF] osp$reset_heap (hp: ^ost$heap;
      o  3463     heap_length: integer;
      o  3464     lock_option: boolean;
      o  3465     algorithm: 0 .. 255);
      o  3466
         3469
         3470
         3471   PROCEDURE [XREF] dsp$allocate_continuous_memory
         3472     (     heap_p: ^ost$heap;
         3473           size: integer;
         3474      VAR return_seq_p: ^SEQ ( * ));
         3475
      o  3478
      o  3479   PROCEDURE [INLINE] dsp$convert_seq_p_to_r_pointer
      o  3480     (     seq_p: ^SEQ ( * );
      o  3481      VAR r_pointer: dst$r_pointer);
      o  3482
         3512
         3513   PROCEDURE [XREF] dsp$fetch_boot_data
         3514     (     data_kind: dst$boot_data_kinds;
         3515      VAR data_p: ^SEQ ( * ));
         3516
      o  3534
      o  3535   PROCEDURE [XREF] dsp$store_data_in_ssr
      o  3536     (     ssr_entry_name: dst$ssr_entry_name;
      o  3537           ssr_value_p: ^SEQ ( * ));
      o  3538
         4212
         4213
         4214      PROCEDURE [XREF] gfp$assign_fde
         4215        (     residence: gft$table_residence;
         4216              segment_number: ost$segment;
         4217         VAR sfid: gft$system_file_identifier;
         4218         VAR fde_p: gft$file_desc_entry_p);
         4219
      o  4417
      o  4418   PROCEDURE [INLINE] gfp$get_fde_p (sfid: gft$system_file_identifier;
      o  4419     VAR fde_p: gft$file_desc_entry_p);
      o  4420
      o  4421
```

```
   4455   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
   4456     VAR ijle_p: ^jmt$initiated_job_list_entry);
   4457
o  4465
o  4466    PROCEDURE [XREF] mmp$assign_mass_storage
o  4467      (     segment_number: ost$segment;
o  4468            sfid: gft$system_file_identifier;
o  4469            min_allocation_length: ost$segment_length;
o  4470       VAR status: ost$status);
o  4471
   4474
   4475 { Convert page streaming transfer size from bytes number of pages expressed as a power of 2
   4476
   4477   PROCEDURE [INLINE] mmp$convert_ps_transfer_size (ps_transfer_size: integer;
   4478         VAR ps_transfer_size_power: 0..15);
o  4504
o  4505   PROCEDURE [XREF] mmp$free_pages (pva: ^cell;
o  4506         length: ost$byte_count;
o  4507         waitopt: ost$wait;
o  4508     VAR status: ost$status);
o  4509
   4520
   4521   PROCEDURE [INLINE] mmp$get_max_sdt_pointer
   4522     (     xcb_p: ^ost$execution_control_block;
   4523      VAR sdt_p: mmt$max_sdt_p);
   4524
o  4532
o  4533   PROCEDURE [INLINE] mmp$get_max_sdt_sdtx_pointer
o  4534     (     xcb_p: ^ost$execution_control_block;
o  4535      VAR sdt_p: mmt$max_sdt_p;
o  4536      VAR sdtx_p: mmt$max_sdtx_p);
o  4537
   4546
   4547    FUNCTION [INLINE] mmp$get_sdt_entry_p
   4548      (     xcb_p: ^ost$execution_control_block;
   4549            segnum: ost$segment): ^mmt$segment_descriptor;
   4550
   4551      mmp$get_sdt_entry_p := #address (1, #segment (xcb_p),
   4552        8 * segnum + xcb_p^.sdt_offset);
   4553
   4554    FUNCEND;
   4555
o  4558
o  4559    FUNCTION [INLINE] mmp$get_sdtx_entry_p
o  4560      (     xcb_p: ^ost$execution_control_block;
o  4561            segnum: ost$segment): ^mmt$segment_descriptor_extended;
o  4562
o  4563      mmp$get_sdtx_entry_p := #address (1, #segment (xcb_p),
o  4564        #SIZE (mmt$segment_descriptor_extended) * segnum + xcb_p^.sdtx_offset);
o  4565
o  4566    FUNCEND;
o  4567
   4570
   4571 PROCEDURE [XREF] mmp$issue_ring1_segment_request (
   4572     VAR rb: mmt$rb_ring1_segment_request];
o  4630
o  4631   PROCEDURE [INLINE] mmp$set_include_pages_in_dump
```



```
o  4632      (     segment_number: ost$segment;
o  4633            fde_p: gft$locked_file_desc_entry_p;
o  4634            sdt_p: ^mmt$segment_descriptor;
o  4635       VAR include_pages_in_dump: boolean);
o  4636
o  4637      IF (sdt_p^.ste.wp <> osc$non_writable) THEN
o  4638        IF (segment_number < mmc$first_loader_predefined_seg) OR
o  4639              (fde_p^.stack_for_ring <> 0) OR
o  4640              (fde_p^.flags.global_template_file) OR
o  4641              (sdt_p^.ste.r1 <= 2) THEN
o  4642          include_pages_in_dump := TRUE;
o  4643        ELSE
o  4644          include_pages_in_dump := FALSE;
o  4645        IFEND;
o  4646      ELSE
o  4647        include_pages_in_dump := FALSE;
o  4648      IFEND;
o  4649
o  4650    PROCEND mmp$set_include_pages_in_dump;
o  4651
   4658    PROCEDURE [XREF] mmp$set_segment_access_rights (sd: mmt$segment_descriptor;
   4659      VAR stxe: mmt$segment_descriptor_extended);
   4660
o  4663
o  4664    PROCEDURE [XREF] osp$fatal_system_error (error_message: string ( * );
o  4665          status: ^ost$status);
o  4666
o  4667    PROCEDURE [XREF] osp$system_error (error_message: string ( * );
o  4668          status: ^ost$status);
o  4669
   4672
   4673    PROCEDURE [INLINE] pmp$find_executing_task_xcb (VAR xcb:
   4674      ^ost$execution_control_block);
o  4690    PROCEDURE [XREF] pmp$zero_out_table
o  4691      (     p: ^cell;
o  4692            len: ost$byte_count);
o  4693
   4696



   4698
   4699    VAR
   4700      dsv$ssr_size: [XREF] 0 .. 0fffff(16);
   4701
   4702
   4703 {  Define variable that is the address (PVA) of the segment descriptor
   4704 {  table (SDT).  The pointer is valid in job mode during deadstart.
   4705
   4706    VAR
   4707      jmv$sdt: [XREF] mmt$max_sdt;
   4708
o  4711
o  4712
o  4713 {  Define variable that is the address (PVA) of the segment descriptor table
o  4714 {  extended (SDTX).  The pointer is valid in job mode during deadstart.
o  4715
```

```
o  4716   VAR
o  4717     jmv$sdtx: [XREF] ^mmt$segment_descriptor_table_ex;
o  4718
   4721
   4722   VAR
   4723     jmv$jmtr_xcb: [XREF] ost$execution_control_block;
   4724
o  4727
o  4728
o  4729   VAR
o  4730     jsv$swapped_page_entry_size: [XREF] 0..0ff(16);
o  4731
o  4732 {This deck contains XREFs to variables used by Memory Manager in
o  4733 {monitor mode. The variables are initialized by a job mode routine
o  4734 {during system deadstart.
o  4735
o  4736   VAR
o  4737     mmv$a_divisor: [XREF] 0 .. 10000(16),
o  4738     mmv$a_mult: [XREF] 0 .. 10000(16),
o  4739     mmv$number_free_astes: [XREF] integer;
o  4740
o  4741 {Define variable used to  indicate if memory manager tables have been initialized.
o  4742
o  4743   VAR
o  4744     mmv$tables_initialized: [XREF] boolean;
o  4745
o  4746
o  4747   VAR
o  4748     mmv$total_page_frames: [XREF] mmt$page_frame_index;
o  4749
o  4750 {Define page table length in words.
o  4751
o  4752   VAR
o  4753     mmv$pt_length: [XREF] integer;
o  4754
o  4755 {Time for next periodic call to Memory Manager from CP Monitor.
o  4756
o  4757   VAR
o  4758     mmv$time_to_call_mem_mgr: [XREF] integer;
o  4759 {Define pointer to array for holding PFTI lists. This array is used in monitor for holding lists
o  4760 {PFTIs of pages belonging to a segment.
o  4761
o  4762   VAR
o  4763     mmv$pfti_array_p: [XREF] ^mmt$pfti_array;
o  4764
o  4766
o  4767
o  4768   TYPE
o  4769     mmt$pfti_array = RECORD
o  4770       pfti_first: 0 .. osc$max_page_frames,
o  4771       pfti_index: 0 .. osc$max_page_frames,
o  4772       last_pfti_index: 0 .. osc$max_page_frames,
o  4773       pftis: ARRAY [0 .. *] of mmt$page_frame_index,
o  4774     RECEND;
o  4776 {The following variable contains a count of the number of page frames that can be reassigned to be
o  4777 {used for another purpose. The count represents the number of pages that are in the free + available
o  4778 {queues. The count is broken into two parts - pages with no IO active, and pages with IO active.
```



```
o  4779
o  4780   VAR
o  4781     mmv$reassignable_page_frames: [XREF] mmt$reassignable_page_frames;
   4791
   4792   VAR
   4793     mmv$image_file: [XREF] mmt$image_file;
   4794
o  4807 {Pointer to the Active Segment Table - (AST).}
o  4808
o  4809   VAR
o  4810     mmv$ast_p: [XREF] ^mmt$active_segment_table;
o  4811
   4814 {
   4815 {CONTINUE BIT COUNT - This array is used to manage the continue bits in the page table.
   4816 {     Each entry in the array contains a count of the number of 'times' the continue bit in
   4817 {     the corresponding page table entry is 'set'. If the count is non-zero, the continue bit
   4818 {     is set.
   4819
   4820   VAR
   4821     mmv$continue_bit_count_p: [XREF] ^mmt$continue_bit_count;
   4822
o  4827
o  4828 { Global Page Queue List array.
o  4829
o  4830   VAR
o  4831     mmv$gpql: [XREF] mmt$global_page_queue_list;
   4834
   4835
   4836 { The data consists of pointers to the various mmv$ variables managed by MMU and also the default value
   4837 { for each of the variables and the default value of the Global Page Queue List.  The default values
   4838 { are saved by mmm$deadstart_initialization.
   4839
   4840   VAR
   4841     mmv$manage_memory_utility: [XREF] mmt$manage_memory_utility;
o  4844
o  4845
o  4846   VAR
o  4847     mmv$pages_per_new_page_fault: [XREF] 1 .. 8;
o  4848 {This variable the time interval between calls from CP Monitor to
o  4849 {memory manager MMP$PERIODIC_CALL procedure.
o  4850
o  4851   VAR
o  4852     mmv$periodic_call_interval: [XREF] integer;
o  4853 {This variable the rate at which Memory Manager ages out all pages of all
o  4854 {working sets that have not been referenced since the last time the
o  4855 {aging was done.
o  4856
o  4857   VAR
o  4858     mmv$shared_queue_age_interval: [XREF] integer;
o  4859 {Pointer to the 'PAGE FRAME TABLE' (PFT)
o  4860
o  4861   VAR
o  4862     mmv$pft_p: [XREF] ^mmt$page_frame_table;
o  4863
   4866 {Pointer to the system PAGE TABLE (PT).
   4867
   4868   VAR
```

```
4869      mmv$pt_p: [XREF] ^ost$page_table;
4870
4873
4874
4875  {  Define XREF variable for default SDTX entry.
4876
4877    VAR
4878      mmv$default_sdtx_entry: [XREF, READ, oss$mainframe_paged_literal]
4879        mmt$segment_descriptor_extended;
4880
4886  {Maximum number of segments in the system (number of ASIDs).}
4887
4888    VAR
4889      mtv$mx_segments: [XREF] 0 .. 0ffffffff(16);
4890
4891    VAR
4892      osv$cpus_logically_on: [XREF] 0 .. osc$max_number_of_processors;
4893
4903  {This variable specifies the lower and upper RMA addresses available to NOSVE.
4904
4905    VAR
4906      osv$180_memory_limits: [XREF] record
4907        lower: 0 .. 0ffffffff(16),
4908        deadstart_upper: 0 .. 0ffffffff(16),  {  Upper limit of memory during deadstart.
4909        upper: 0 .. 0ffffffff(16),
4910      recend;
4911
4912  {System page size.}
4913
4914    VAR
4915      osv$page_size: [XREF] ost$page_size;
4916
4919
4920    VAR
4921      dfv$server_wired_heap: [XREF, dfs$server_wired] ^ost$heap;
4922
4929
4930    VAR
4931      nav$network_wired_heap: [XREF, oss$network_wired] ^ost$heap;
4932
4944
4945    VAR
4946      nav$network_paged_heap: [XREF, oss$network_paged] ^ost$heap;
4947
4954
4955  {  Define variable that will contain the SDT index for the first transient segment.
4956
4957    VAR
4958      mmv$first_transient_seg_index: [XDCL] ost$segment := mmc$first_transient_segment;
4959
4960    VAR
4961      mlv$shared_segment: [XREF] mlt$shared_segment;
4962
```


ASID CONVERSION FUNCTIONS

```
4965
4966  {-------------------------------------------------------------------------------------------------------------
4967  {Name:
4968  {     mmp$ast_index
4969  {Purpose:
4970  {These functions convert AST indexes into an ASID and vise-versa.
4971  {Input:
4972  {     AST_index or ASID
4973  {Output:
4974  {     asid or ast_index
4975  {-------------------------------------------------------------------------------------------------------------
4976
4977    VAR
4978      bits: array [0 .. 15] of 0 .. 255 := [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15];
4979
4980    PROCEDURE [INLINE] mmp$asti (xasid: ost$asid;
4981      VAR xasti: mmt$ast_index);
4982
4983      VAR
4984        asid: ost$asid,
4985        i: integer,
4986        asti: integer;
4987
4988
4989      asid := xasid DIV mmv$a_mult + (xasid MOD mmv$a_mult) * mmv$a_divisor;
4990      asti := 0;
4991      FOR i := 1 TO 4 DO
4992        asti := asti * 16 + bits [asid MOD 16];
4993        asid := asid DIV 16;
4994      FOREND;
4995      xasti := asti;
4996    PROCEND mmp$asti;
4997
4998
4999    PROCEDURE [INLINE] mmp$asid (xasti: mmt$ast_index;
5000      VAR asid: ost$asid);
5001
5002      VAR
5003        asti: mmt$ast_index;
5004
5005      asti := xasti;
5006      asid := (bits [asti MOD 16] * 4096) + (bits [(asti DIV 16) MOD 16] * 256) + (bits [(asid DIV 256) MOD 16]
5007          * 16) + bits [(asti DIV 4096) MOD 16];
5008      asid := asid DIV mmv$a_divisor + (asid MOD mmv$a_divisor) * mmv$a_mult;
5009
5010    PROCEND mmp$asid;
```

MMP$ADD_GLOBAL_TEMPLATE_SEGMENT

```
0  5012 {
0  5013 { This procedure is used during deadstart to add job template segments to the address space
0  5014 { of the system job monitor.
0  5015 {
0  5016
0  5017
0  5018   PROCEDURE [XDCL] mmp$add_global_template_segment
4  5019     (     sdt_entry: mmt$segment_descriptor;
4  5020           sdtx_entry: mmt$segment_descriptor_extended;
4  5021           segnum: ost$segment;
4  5022      VAR status: ost$status);
4  5023
4  5024     VAR
4  5025       fde_p: gft$file_desc_entry_p,
4  5026       page_streaming_transfer_size: 0 .. 15,
4  5027       ste_p: ^mmt$segment_descriptor,
4  5028       stxe_p: ^mmt$segment_descriptor_extended,
4  5029       xcb_p: ^ost$execution_control_block;
4  5030
4  5031     status.normal := TRUE;
4  5032
4  5033     pmp$find_executing_task_xcb (xcb_p);
14 5034     IF xcb_p <> ^jmv$jmtr_xcb THEN
34 5035       osp$system_error ('MM - incorrect call to mmp$add_global_template_segment', NIL);
5C 5036     IFEND;
5C 5037
5C 5038     mmp$convert_ps_transfer_size (16384, page_streaming_transfer_size); {convert TS to pages in a power of 2
96 5039     ste_p := mmp$get_sdt_entry_p (xcb_p, segnum);
96 5040     stxe_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
96 5041
96 5042     ste_p^ := sdt_entry;
96 5043     stxe_p^ := sdtx_entry;
FC 5044     stxe_p^.stream.transfer_size := page_streaming_transfer_size;
FC 5045
FC 5046     IF stxe_p^.sfid.residence = gfc$tr_null_residence THEN
114 5047      assign_fde (gfc$tr_job, 0, segnum, stxe_p^.sfid, fde_p);
12E 5048    IFEND;
12E 5049
12E 5050    gfp$get_fde_p (stxe_p^.sfid, fde_p);
184 5051    fde_p^.flags.global_template_file := TRUE;
184 5052
184 5053  PROCEND mmp$add_global_template_segment;
```


MMP$WRITE_ALL_SEGMENTS_TO_DISK

```
0  5055
0  5056   PROCEDURE [XDCL, #GATE] mmp$write_all_segments_to_disk
0  5057     (VAR status: ost$status);
0  5058
0  5059
0  5060     VAR
0  5061       xcb_p: ^ost$execution_control_block,
0  5062       rb: mmt$rb_ring1_segment_request,
0  5063       segnum: ost$segment,
0  5064       st_p: mmt$max_sdt_p;
0  5065
0  5066     rb.reqcode := syc$rc_ring1_segment_request;
4  5067     rb.request := mmc$sr1_flush_seg_segnum;
4  5068     rb.wait_for_io_complete := FALSE;
4  5069     pmp$find_executing_task_xcb (xcb_p);
1E 5070     mmp$get_max_sdt_pointer (xcb_p, st_p);
1E 5071     FOR segnum := 1 TO #READ_REGISTER (osc$pr_segment_table_length) DO
5E 5072       IF (st_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) THEN
72 5073         mmp$assign_mass_storage (segnum, gfv$null_sfid, 0, status);
9A 5074         IF status.normal AND (segnum <> xcb_p^.xp.tos_registers [1].pva.seg) THEN
B2 5075           rb.segnum := segnum;
B2 5076           mmp$issue_ring1_segment_request (rb);
CA 5077         IFEND;
CA 5078       IFEND;
CA 5079     FOREND;
D0 5080     status.normal := TRUE;
D0 5081
D0 5082   PROCEND mmp$write_all_segments_to_disk;
```

MMP$INITIALIZE

```
        0  5084
        0  5085 {----------------------------------------------------------------------------------------------------
        0  5086 {Name:
        0  5087 {  mmp$initialize
        0  5088 {Purpose:
        0  5089 {     This routine is the first procedure in the module mmm$deadstart_initialization to be called during
        0  5090 {     deadstart.  Later deadstart will make separate calls to mmp$assign_device_shared_segs,
        0  5091 {     mmp$pft_initialize, and to mmp$write_all_segments_to_disk.
        0  5092 {Input:
        0  5093 {     none
        0  5094 {Output:
        0  5095 {     The static data constants are initialized and the SDTX will be initialized via
        0  5096 {     a call to the procedure mmp$sdtx_initialization.
        0  5097 {Error Codes:
        0  5098 {     none
        0  5099 {Notes:
        0  5100 {     - The system heap must be initialized before calling this routine
        0  5101 {----------------------------------------------------------------------------------------------------
        0  5102
        0  5103   PROCEDURE [XDCL] mmp$initialize;
        0  5104
        0  5105     VAR
        0  5106       pti: integer,
        0  5107       i: integer,
        0  5108       index_ma: mmt$mmu_memory_attributes;
        0  5109
        0  5110 {Set up static data constants.
        0  5111
        0  5112     i := #READ_REGISTER (osc$pr_page_table_length);
        A  5113     mmv$pt_length := (i + 1) * 512;
        A  5114     i := i MOD 100(16);
        A  5115     mmv$a_divisor := 256 DIV (i + 1);
        A  5116     mmv$a_mult := 10000(16) DIV mmv$a_divisor;
        A  5117     mmv$pt_p := #ADDRESS (1, 0, 0);
        A  5118     osv$page_size := 512 * (128 - #READ_REGISTER (osc$pr_page_size_mask));
       62  5119
       62  5120     osv$180_memory_limits.lower := ((osv$180_memory_limits.lower + osv$page_size - 1) DIV osv$page_size) *
       62  5121           osv$page_size;
       62  5122
       62  5123
       62  5124 { Set all continue bits in the page table. Continue bits are not managed by the page
       62  5125 { fault processor used before AST/PFT initialization is complete. The continue bits will
       62  5126 { be cleaned up during PFT initialization.
       62  5127
       62  5128     FOR pti := 0 TO mmv$pt_length - 1 DO
       96  5129       mmv$pt_p^ [pti].c := TRUE;
       96  5130     FOREND;
       AA  5131
       AA  5132
       AA  5133 {  Copy the current values of mmv$gpq1 and other mmv$ variables managed by the Manage Memory Utility into
       AA  5134 {  the default copies so that the MMU will have available the original values when it is requested to
       AA  5135 {  reset values to their default.
       AA  5136
       AA  5137     FOR index_ma := LOWERBOUND (mmv$manage_memory_utility.ma) TO UPPERBOUND (mmv$manage_memory_utility.ma) DO
       B4  5138       CASE mmv$manage_memory_utility.ma [index_ma].value_type OF
       C6  5139       = mmu_integer =
```


MMP$INITIALIZE

```
       C6  5140         mmv$manage_memory_utility.ma [index_ma].default := mmv$manage_memory_utility.ma [index_ma].integer_p^;
       DC  5141       = mmu_byte =
       DC  5142         mmv$manage_memory_utility.ma [index_ma].default := mmv$manage_memory_utility.ma [index_ma].byte_p^;
       E8  5143       CASEND;
       E8  5144     FOREND;
       EC  5145
       EC  5146     mmv$manage_memory_utility.gpq1 := mmv$gpq1;
      108  5147
      108  5148
      108  5149 { Initialize fields in the SDTX.
      108  5150
      108  5151     mmp$sdtx_initialization;
      110  5152
      110  5153   PROCEND mmp$initialize;
```

```
        o  5156
        o  5157 {-------------------------------------------------------------------------------------------------------
        o  5158 {Name:
        o  5159 {  mmp$pft_initialize
        o  5160 {Purpose:
        o  5161 {  By the time this procedure is called the other procedures in mmm$deadstart_initialization have
        o  5162 {  already been executed.  This routine initializes the PFT, PQL and the defaults for Manage_Memory
        o  5163 {Input:
        o  5164 {Output:
        o  5165 {  none
        o  5166 {Error Code:
        o  5167 {  none
        o  5168 {-------------------------------------------------------------------------------------------------------
        o  5169
        o  5170   PROCEDURE [XDCL] mmp$pft_initialize;
        o  5171
        o  5172     TYPE
        o  5173       asidq_type = record
        o  5174         asid: ost$asid,
        o  5175         queue_id: mmt$page_frame_queue_id,
        o  5176         qcb_p: ^mmt$page_queue_list_entry,
        o  5177         aste_p: ^mmt$active_segment_table_entry,
        o  5178         fde_p: gft$file_desc_entry_p,
        o  5179       recend;
        o  5180
        o  5181     VAR
        o  5182       asid: ost$asid,
        o  5183       asid1: ost$asid,
        o  5184       asid2: ost$asid,
        o  5185       asid3: ost$asid,
        o  5186       asid_seq_p: ^SEQ ( * ),
        o  5187       asid_size: integer,
        o  5188       asidq: array [0 .. 40] of asidq_type,
        o  5189       asidq_p: ^asidq_type,
        o  5190       aste_p: ^mmt$active_segment_table_entry,
        o  5191       asti: mmt$ast_index,
        o  5192       boot_asids: dst$boot_asids,
        o  5193       boot_asids_seq_p: ^SEQ ( * ),
        o  5194       cbc_seq_p: ^SEQ ( * ),
        o  5195       count: 1 .. 32,
        o  5196       fde_p: gft$file_desc_entry_p,
        o  5197       first_image_pfti: 0 .. 0ffffffff(16),
        o  5198       found: boolean,
        o  5199       fwd_link: integer,
        o  5200       i: integer,
        o  5201       ijle_p: ^jmt$initiated_job_list_entry,
        o  5202       ipti: integer,
        o  5203       j: integer,
        o  5204       last_asid: ost$asid,
        o  5205       next_asidq_index: integer,
        o  5206       last_link_p: ^mmt$link,
        o  5207       mf_wired_asid_p: ^mmt$mainframe_wired_asid,
        o  5208       pages_to_dump_r_pointer: dst$r_pointer,
        o  5209       pft_p: ^mmt$page_frame_table_entry,
        o  5210       pft_seq_p: ^SEQ ( * ),
        o  5211       pft_size: integer,
```



```
        o  5212       pfti: mmt$page_frame_index,
        o  5213       pftimax: integer,
        o  5214       pftimin: integer,
        o  5215       pti: ost$page_table_index,
        o  5216       pt_p: ^ost$page_table,
        o  5217       pte_p: ^ost$page_table_entry,
        o  5218       qcb_p: ^mmt$page_queue_list_entry,
        o  5219       queue_id: mmt$page_frame_queue_id,
        o  5220       rb: mmt$rb_ring1_segment_request,
        o  5221       residence: gft$table_residence,
        o  5222       ssr_size: integer,
        o  5223       sdt_p: mmt$max_sdt_p,
        o  5224       sdtx_p: mmt$max_sdtx_p,
        o  5225       sva: ost$system_virtual_address,
        o  5226       sdt_entry: mmt$segment_descriptor,
        o  5227       sdte: [STATIC] mmt$segment_descriptor := [[osc$v1_cache_bypass, osc$non_executable,
        o  5228             osc$read_uncontrolled, osc$write_uncontrolled, 1, 1, 0, * ], 0, 0],
        o  5229       sdtx_entry: mmt$segment_descriptor_extended;
        o  5230
        o  5231
        o  5232 { Set maximum number of ASIDs based on memory size and allocate the AST.
        o  5233
        o  5234     IF mtv$mx_segments = 0 THEN
       18  5235       mtv$mx_segments := ((osv$180_memory_limits.upper - osv$180_memory_limits.lower) DIV osv$page_size) +
       18  5236             300;
       18  5237       IF mtv$mx_segments > OFFFE(16) THEN
       40  5238         mtv$mx_segments := OFFFE(16);
       48  5239       IFEND;
       48  5240     IFEND;
       48  5241     mmv$number_free_astes := mtv$mx_segments;
       48  5242     ALLOCATE mmv$ast_p: [0 .. (mtv$mx_segments)] IN osv$mainframe_wired_heap^;
       94  5243     i := osv$180_memory_limits.lower DIV osv$page_size;
       94  5244     IF i = 0 THEN
       AA  5245       i := 1;
       AE  5246     IFEND;
       AE  5247
       AE  5248 { Allocate the PFT.
       AE  5249
       AE  5250     pft_size := (osv$180_memory_limits.upper DIV osv$page_size - i) * (#SIZE (mmt$page_frame_table_entry));
       AE  5251     dsp$allocate_continuous_memory (osv$mainframe_wired_heap, pft_size, pft_seq_p);
       E4  5252     RESET pft_seq_p;
       E4  5253     NEXT mmv$pft_p: [i .. (osv$180_memory_limits.upper DIV osv$page_size) - 1] IN pft_seq_p;
      13C  5254
      13C  5255 { Allocate the critical dump page bit table.
      13C  5256
      13C  5257     ssr_size := ((dsv$ssr_size + osv$page_size - 1) DIV osv$page_size) * osv$page_size;
      13C  5258     pft_size := (((osv$180_memory_limits.upper + ssr_size) DIV osv$page_size) + 64) DIV 8;
      13C  5259     dsp$allocate_continuous_memory (osv$mainframe_wired_heap, pft_size, pft_seq_p);
      182  5260     RESET pft_seq_p;
      182  5261     NEXT mmv$pages_to_dump_p: [0 .. (pft_size * 8) - 1] IN pft_seq_p;
      1C6  5262
 *WARN*    5263     dsp$allocate_continuous_memory (osv$mainframe_wired_heap, mmv$pt_length, cbc_seq_p);
      1F2  5264     RESET cbc_seq_p;
      1F2  5265     NEXT mmv$continue_bit_count_p: [0 .. mmv$pt_length - 1] IN cbc_seq_p;
 *WARN*    5266     ALLOCATE mmv$pfti_array_p: [0 .. (osv$180_memory_limits.upper DIV osv$page_size) - 1] IN
      280  5267           osv$mainframe_wired_heap^;
```

```
      280  5268
      280  5269 {Zero out the tables allocated.}
      280  5270
      280  5271     pmp$zero_out_table (#LOC (mmv$ast_p^), #SIZE (mmv$ast_p^));
      29C  5272     pmp$zero_out_table (#LOC (mmv$pft_p^), #SIZE (mmv$pft_p^));
*WARN*     5273     pmp$zero_out_table (#LOC (mmv$pages_to_dump_p^), #SIZE (mmv$pages_to_dump_p^));
      2DA  5274     pmp$zero_out_table (#LOC (mmv$continue_bit_count_p^), #SIZE (mmv$continue_bit_count_p^));
      2FA  5275     pmp$zero_out_table (#LOC (mmv$pfti_array_p^), #SIZE (mmv$pfti_array_p^));
      31A  5276
      31A  5277 {  Set cache bypass if multiprocessing enabled.
      31A  5278
      31A  5279     mmp$get_max_sdt_sdtx_pointer (^jmv$jmtr_xcb, sdt_p, sdtx_p);
      31A  5280
      31A  5281     IF osv$cpus_logically_on > 1 THEN
      35E  5282       sdt_p^.st [osc$segnum_mainframe_paged].ste.v1 := osc$v1_cache_bypass;
      35E  5283       sdt_p^.st [#SEGMENT (#LOC (nav$network_paged_heap^))].ste.v1 := osc$v1_cache_bypass;
      35E  5284       sdt_p^.st [#SEGMENT (^mlv$shared_segment)].ste.v1 := osc$v1_cache_bypass;
      3AA  5285     ELSE
      3AA  5286       {Make mainframe wired a cache segment
      3AA  5287       rb.reqcode := syc$rc_ring1_segment_request;
      3AA  5288       rb.request := mmc$sr1_make_mfw_cache;
      3AA  5289       rb.wait_for_io_complete := FALSE;
      3AA  5290       sdt_p^.st [#SEGMENT (osv$mainframe_wired_heap)].ste.v1 := osc$v1_regular_segment;
*WARN*     5291       mmp$issue_ring1_segment_request (rb);
      3F2  5292     IFEND;
      3F2  5293
      3F2  5294
      3F2  5295 { Set up the table used to locate mainframe wired if the system crashes while ASID
      3F2  5296 { reassignment on mainframe wired is active. If the system crashes while this is
      3F2  5297 { happening, both old and new ASIDs must be located in the page table.
      3F2  5298
      3F2  5299     asid_size := #SIZE (mmt$mainframe_wired_asid);
      3F2  5300     dsp$allocate_continuous_memory (osv$mainframe_wired_heap, asid_size, asid_seq_p);
      416  5301     RESET asid_seq_p;
      416  5302     NEXT mf_wired_asid_p IN asid_seq_p;
      440  5303     mmv$mf_wired_asid := mf_wired_asid_p^;
      440  5304     mmv$mf_wired_asid.current := sdt_p^.st [osc$segnum_mainframe_wired].ste.asid;
      440  5305     mmv$mf_wired_asid.new := 0;
      440  5306
      440  5307     pftimin := LOWERBOUND (mmv$pft_p^);
      440  5308     pftimax := UPPERBOUND (mmv$pft_p^);
      440  5309     mmv$total_page_frames := pftimax - pftimin;
      440  5310
      440  5311     jmp$get_ijle_p (jmv$system_ijl_ordinal, ijle_p);
      440  5312     ijle_p^.job_fixed_asid := sdt_p^.st [osc$segnum_job_fixed_heap].ste.asid;
      440  5313
      440  5314
      440  5315 { Free the pages used by the boot. The easiest way to do this at this point in deadstart
      440  5316 { is to delete the page table entries used by the boot.
      440  5317
      440  5318     boot_asids_seq_p := #SEQ (boot_asids);
*WARN*     5319     dsp$fetch_boot_data (dsc$boot_asids, boot_asids_seq_p);
      4EC  5320     sdt_p^.st [osc$segnum_job_pageable_heap].ste.v1 := osc$v1_invalid_entry;
      4EC  5321
      4EC  5322     pt_p := mmv$pt_p;
      4EC  5323     asid1 := boot_asids.code_data;
```



```
      4EC  5324     asid2 := boot_asids.job_stack;
      4EC  5325     asid3 := boot_asids.mtr_stack;
      4EC  5326     FOR pti := 0 TO mmv$pt_length - 1 DO
      520  5327       asid := pt_p^ [pti].pageid.asid;
      520  5328       IF (asid = asid1) OR (asid = asid2) OR (asid = asid3) THEN
      538  5329         pt_p^ [pti].v := FALSE;
      538  5330         pt_p^ [pti].pageid.asid := 0;
      554  5331       IFEND;
      554  5332     FOREND;
      558  5333
      558  5334
      558  5335 { Search the segment table of the job monitor. For each valid entry, create an AST
      558  5336 { and FDE entry that describes the segment.
      558  5337
      558  5338     next_asidq_index := 0;
      558  5339
      558  5340   /scan_sdt/
      558  5341     FOR i := 0 TO jmv$jmtr_xcb.xp.segment_table_length DO
      56E  5342       IF sdt_p^.st [i].ste.v1 = osc$v1_invalid_entry THEN
      582  5343         CYCLE /scan_sdt/
      586  5344       IFEND;
      586  5345
      586  5346       asidq_p := ^asidq [next_asidq_index];
      586  5347
      586  5348       IF mmc$sa_wired IN sdtx_p^.sdtx_table [i].software_attribute_set THEN
      5AC  5349         queue_id := mmc$pq_wired;
      5B2  5350       ELSEIF mmc$sa_fixed IN sdtx_p^.sdtx_table [i].software_attribute_set THEN
      5BA  5351         queue_id := mmc$pq_job_fixed;
      5C4  5352       ELSEIF sdtx_p^.sdtx_table [i].open_validating_ring_number = 0 THEN
      5CC  5353         queue_id := mmc$pq_shared_task_service;
      5D4  5354       ELSE
      5D4  5355         queue_id := mmc$pq_job_working_set;
      5DA  5356       IFEND;
      5DA  5357
      5DA  5358       asidq_p^.asid := sdt_p^.st [i].ste.asid;
      5DA  5359       asidq_p^.queue_id := queue_id;
      5DA  5360       mmp$asti (asidq_p^.asid, asti);
      636  5361       sdt_p^.st [i].asti := asti;
      636  5362
      636  5363       IF asti <= UPPERBOUND (mmv$ast_p^) THEN
      662  5364         asidq_p^.aste_p := ^mmv$ast_p^ [asti];
      662  5365         IF NOT asidq_p^.aste_p^.in_use THEN
      676  5366           asidq_p^.aste_p^.in_use := TRUE;
      676  5367           IF queue_id < mmc$pq_job_base THEN
      682  5368             residence := gfc$tr_system;
      688  5369           ELSE
      688  5370             residence := gfc$tr_job;
      68A  5371           IFEND;
      68A  5372           assign_fde (residence, asti, i, asidq_p^.aste_p^.sfid, asidq_p^.fde_p);
      6B8  5373           sdtx_p^.sdtx_table [i].sfid := asidq_p^.aste_p^.sfid;
      6B8  5374           IF (queue_id = mmc$pq_job_working_set) OR (queue_id = mmc$pq_shared_task_service) THEN
      6E0  5375             sdtx_p^.sdtx_table [i].assign_active := mmc$assign_active_escaped;
      6FA  5376           IFEND;
      6FA  5377           asidq_p^.aste_p^.queue_id := asidq_p^.queue_id;
      6FA  5378           asidq_p^.aste_p^.ijl_ordinal := jmv$system_ijl_ordinal;
      6FA  5379           IF mmc$sa_stack IN sdtx_p^.sdtx_table [i].software_attribute_set THEN
```

```
72C  5380              asidq_p^.fde_p^.stack_for_ring := sdt_p^.st [i].ste.r1;
744  5381            IFEND;
744  5382
744  5383            mmp$set_include_pages_in_dump (i, asidq_p^.fde_p, ^sdt_p^.st [i],
744  5384                  asidq_p^.aste_p^.include_pages_in_dump);
7A8  5385            mmv$number_free_astes := mmv$number_free_astes - 1;
7B2  5386
7B2  5387          IFEND;
7B6  5388
7B6  5389        ELSE
7B6  5390          asidq_p^.aste_p := NIL;
7B6  5391          asidq_p^.fde_p := NIL;
7BE  5392        IFEND;
7BE  5393
7BE  5394        IF queue_id < mmc$pq_job_base THEN
7C6  5395          asidq_p^.qcb_p := ^mmv$gpql [queue_id].pqle;
7DA  5396        ELSE
7DA  5397          asidq_p^.qcb_p := ^ijle_p^.job_page_queue_list [queue_id];
7F0  5398        IFEND;
7F0  5399        next_asidq_index := next_asidq_index + 1;
7F0  5400
7F0  5401      FOREND /scan_sdt/;
806  5402
806  5403      last_asid := 0;
806  5404
806  5405
806  5406  { Search thru the page table. Initialize the PFT entry for each page found and
806  5407  { link the page to the correct page queue.
806  5408
806  5409    /scan_page_table/
806  5410      FOR pti := mmv$pt_length - 1 DOWNTO 0 DO
81C  5411        asid := mmv$pt_p^ [pti].pageid.asid;
81C  5412        IF asid = 0 THEN
830  5413          CYCLE /scan_page_table/
834  5414        IFEND;
834  5415        pte_p := ^mmv$pt_p^ [pti];
834  5416
834  5417  { Find the ASIDQ table entry for the segment. Usually page table entries for the same segment
834  5418  { are clustered together - skip the ASIDQ search if new entry is the same as the previous
834  5419  { entry. If the entry is not found, the ASID must belong to a segment that is accessible in
834  5420  { monitor mode ONLY.
834  5421
834  5422        IF asid <> last_asid THEN
83A  5423          j := 0;
83A  5424
83A  5425          WHILE (j < next_asidq_index) AND (asidq [j].asid <> asid) DO
84C  5426            j := j + 1;
84C  5427          WHILEND;
860  5428
860  5429          IF j = next_asidq_index THEN
864  5430            asidq_p := ^asidq [j];
864  5431            mmp$asti (asid, asti);
8C2  5432            IF asti <= UPPERBOUND (mmv$ast_p^) THEN
8DE  5433              asidq_p^.aste_p := ^mmv$ast_p^ [asti];
8DE  5434              IF NOT asidq_p^.aste_p^.in_use THEN
*WARN*  5435                assign_fde (gfc$tr_system, asti, 0, asidq_p^.aste_p^.sfid, asidq_p^.fde_p);
```



```
924  5436                asidq_p^.fde_p^.file_kind := gfc$fk_monitor_only_unnamed;
924  5437                asidq_p^.aste_p^.in_use := TRUE;
924  5438                asidq_p^.aste_p^.queue_id := mmc$pq_wired;
924  5439                asidq_p^.aste_p^.ijl_ordinal := jmv$system_ijl_ordinal;
924  5440                IF asid <> 0ffff(16) THEN
950  5441                  asidq_p^.aste_p^.include_pages_in_dump := TRUE;
954  5442                IFEND;
954  5443                mmv$number_free_astes := mmv$number_free_astes - 1;
962  5444              IFEND;
966  5445            ELSE
966  5446              asidq_p^.aste_p := NIL;
966  5447              asidq_p^.fde_p := NIL;
972  5448            IFEND;
972  5449            asidq_p^.asid := asid;
972  5450            asidq_p^.qcb_p := ^mmv$gpql [mmc$pq_wired].pqle;
972  5451            asidq_p^.queue_id := mmc$pq_wired;
972  5452            next_asidq_index := next_asidq_index + 1;
992  5453          IFEND;
992  5454
992  5455          asidq_p := ^asidq [j];
992  5456          aste_p := asidq_p^.aste_p;
992  5457          queue_id := asidq_p^.queue_id;
992  5458          qcb_p := asidq_p^.qcb_p;
992  5459          last_asid := asid;
992  5460          fde_p := asidq_p^.fde_p;
9BA  5461        IFEND;
9BA  5462
9BA  5463
9BA  5464        pfti := (pte_p^.rma * 512) DIV osv$page_size;
9BA  5465
9BA  5466        sva.asid := asid;
9BA  5467        sva.offset := pte_p^.pageid.pagenum * 512;
9BA  5468        #HASH_SVA (sva, ipti, count, found);
9EA  5469        FOR i := 2 TO count DO
A08  5470          ipti := ipti - 1;
A08  5471          IF ipti < 0 THEN
A0E  5472            ipti := mmv$pt_length - 1;
A20  5473          IFEND;
A20  5474          mmv$continue_bit_count_p^ [ipti] := mmv$continue_bit_count_p^ [ipti] + 1;
A20  5475        FOREND;
A34  5476
A34  5477        IF aste_p <> NIL THEN
A3A  5478          aste_p^.pages_in_memory := aste_p^.pages_in_memory + 1;
A44  5479        IFEND;
A44  5480
A44  5481        IF (pfti < pftimin) OR (pfti > pftimax) OR (pfti = pftimin) AND (asid = 0FFFF(16)) THEN
A58  5482          IF (pfti >= pftimin) AND (pfti <= UPPERBOUND (mmv$pages_to_dump_p^)) THEN
A6E  5483            IF asid <> 0FFFF(16) THEN
A76  5484              mmv$pages_to_dump_p^ [pfti] := TRUE;
A82  5485            IFEND;
A82  5486          IFEND;
A82  5487          CYCLE /scan_page_table/
A90  5488        IFEND;
A90  5489
A90  5490        pft_p := ^mmv$pft_p^ [pfti];
A90  5491        pft_p^.pti := pti;
```

MMP$PFT_INITIALIZE

```
      A90  5492        pft_p^.sva := sva;
      A90  5493        pft_p^.queue_id := queue_id;
      A90  5494        pft_p^.aste_p := aste_p;
      A90  5495        pft_p^.age := 1;
      A90  5496        pft_p^.ijl_ordinal := jmv$system_ijl_ordinal;
      A90  5497        pft_p^.locked_page := mmc$lp_not_locked;
      A90  5498        pft_p^.link.fwd := qcb_p^.link.fwd;
*WARN*     5499        IF (fde_p <> NIL) AND (fde_p^.eoi_byte_address < sva.offset + osv$page_size) THEN
      B10  5500          fde_p^.eoi_byte_address := sva.offset + osv$page_size;
      B18  5501        IFEND;
      B18  5502        IF pft_p^.aste_p <> NIL THEN
*WARN*     5503          link_page_to_segment_ds (pfti, pft_p, aste_p);
      BA6  5504          mmv$pages_to_dump_p^ [pfti] := aste_p^.include_pages_in_dump
      BBC  5505        ELSE
      BBC  5506          mmv$pages_to_dump_p^ [pfti] := TRUE;
      BCC  5507        IFEND;
      BCC  5508        fwd_link := qcb_p^.link.fwd;
      BCC  5509        IF fwd_link <> 0 THEN
      BD4  5510          mmv$pft_p^ [fwd_link].link.bkw := pfti;
      BF4  5511        IFEND;
      BF4  5512        qcb_p^.link.fwd := pfti;
      BF4  5513        qcb_p^.count := qcb_p^.count + 1;
      BF4  5514        IF qcb_p^.link.bkw = 0 THEN
      C0A  5515          qcb_p^.link.bkw := pfti;
      C0E  5516        IFEND;
      C0E  5517
      C0E  5518      FOREND /scan_page_table/;
      C36  5519
      C36  5520      first_image_pfti := osv$180_memory_limits.deadstart_upper DIV osv$page_size;
      C36  5521      qcb_p := ^mmv$gpql [mmc$pq_free].pqle;
      C36  5522      last_link_p := ^qcb_p^.link;
      C36  5523
      C36  5524    /scan_pft/
      C36  5525      FOR pfti := pftimin TO pftimax DO
      C56  5526        IF pfti >= first_image_pfti THEN
      C5A  5527          IF mmv$pft_p^ [pfti].age <> 0 THEN
      C7E  5528            osp$fatal_system_error (' Not enough memory to deadstart- PFT INIT', NIL);
      CA0  5529          IFEND;
      CA0  5530          mmv$pft_p^ [pfti].link.fwd := 0;
      CA0  5531          mmv$pft_p^ [pfti].link.bkw := 0;
      CA0  5532          mmv$pft_p^ [pfti].queue_id := mmc$pq_free;
      CA0  5533          mmv$pft_p^ [pfti].aste_p := NIL;
      CCC  5534        ELSE
      CCC  5535          IF mmv$pft_p^ [pfti].age <> 0 THEN
      CF0  5536            CYCLE /scan_pft/
      CF4  5537          IFEND;
      CF4  5538          last_link_p^.bkw := pfti;
      CF4  5539          last_link_p := ^mmv$pft_p^ [pfti].link;
      CF4  5540          last_link_p^.fwd := qcb_p^.link.fwd;
      CF4  5541          qcb_p^.link.fwd := pfti;
      CF4  5542          qcb_p^.count := qcb_p^.count + 1;
      CF4  5543          mmv$pft_p^ [pfti].queue_id := mmc$pq_free;
      CF4  5544          mmv$pft_p^ [pfti].aste_p := NIL;
      D1E  5545        IFEND;
      D1E  5546        mmv$pft_p^ [pfti].segment_link.fwd := 0;
      D1E  5547        mmv$pft_p^ [pfti].segment_link.bkw := 0;
```


MMP$PFT_INITIALIZE

```
      D1E  5548      FOREND /scan_pft/;
      D44  5549
      D44  5550
      D44  5551 {Store ASID in all unused AST entries.
      D44  5552
      D44  5553      FOR i := 1 TO mtv$mx_segments DO
      D56  5554        IF NOT mmv$ast_p^ [i].in_use THEN
      D68  5555          mmp$asid (i, mmv$ast_p^ [i].asid);
      DE8  5556        IFEND;
      DE8  5557      FOREND;
      DEC  5558
      DEC  5559
      DEC  5560 {Set the count of the number of reassignable page frames.
      DEC  5561
      DEC  5562      mmv$reassignable_page_frames.now := qcb_p^.count;
      DEC  5563
      DEC  5564
      DEC  5565      FOR pti := 0 TO mmv$pt_length - 1 DO
      E12  5566        mmv$pt_p^ [pti].c := mmv$continue_bit_count_p^ [pti] > 0;
      E12  5567      FOREND;
      E36  5568
      E36  5569      mmv$time_to_call_mem_mgr := #FREE_RUNNING_CLOCK (0) + 5000000;
      E3C  5570
      E3C  5571      jsv$swapped_page_entry_size := #SIZE (jst$swapped_page_descriptor);
      E3C  5572
      E3C  5573 { Set up pointer to flag SCI that a critical page dump is now available.
      E3C  5574
*WARN*     5575      dsp$convert_seq_p_to_r_pointer (#SEQ (mmv$pages_to_dump_p^), pages_to_dump_r_pointer);
      E92  5576      dsp$store_data_in_ssr (dsc$ssr_pages_to_dump, #SEQ (pages_to_dump_r_pointer));
      F16  5577      mmv$tables_initialized := TRUE;
      F16  5578
      F16  5579 { The number 16384 is arbitrary and only must be less than or equal to the
      F16  5580 { minimum allocation unit size.
      F16  5581
      F16  5582      mmv$pages_per_new_page_fault := 16384 DIV osv$page_size;
      F16  5583      IF mmv$pages_per_new_page_fault > 4 THEN
      F3A  5584        mmv$pages_per_new_page_fault := 4;
      F3E  5585      IFEND;
      F3E  5586
      F3E  5587
      F3E  5588 { The following code is to support the hyperchannel project.
      F3E  5589
      F3E  5590      IF osv$enable_hyperchannel THEN
      F4A  5591        sdt_entry := sdte;
      F4A  5592        sdt_entry.ste.r1 := 6;
      F4A  5593        sdt_entry.ste.r2 := 6;
      F4A  5594        sdt_entry.ste.asid := 0;
      F4A  5595        sdt_entry.ste.v1 := osc$v1_cache_bypass;
      F4A  5596        sdtx_entry := mmv$default_sdtx_entry;
      F74  5597        sdtx_entry.software_attribute_set := sdtx_entry.software_attribute_set +
      F74  5598             $mmt$software_attribute_set [mmc$sa_wired];
      F74  5599        sdtx_entry.open_validating_ring_number := 0;
      F74  5600        sdtx_entry.inheritance := mmc$si_share_segment;
      F74  5601        assign_fde (gfc$tr_system, 0, osc$segment_for_hyperchannel, sdtx_entry.sfid, fde_p);
      FAA  5602        fde_p^.file_limit := 989680(16);
      FAA  5603        fde_p^.last_segment_number := osc$segment_for_hyperchannel;
```

MMP$PFT_INITIALIZE

```
FAA  5604        sdt_p^.st [osc$segment_for_hyperchannel] := sdt_entry;
FAA  5605        sdtx_p^.sdtx_table [osc$segment_for_hyperchannel] := sdtx_entry;
FDO  5606      IFEND;
FDO  5607
FDO  5608    PROCEND mmp$pft_initialize;
  0  5609
```


assign_fde

```
  0  5611 { Purpose:
  0  5612 {   This procedure is called from mmp$pft_initialize to assign and initialize an FDE
  0  5613 {   entry for a segment.
  0  5614
  0  5615    PROCEDURE assign_fde
  4  5616      (     residence: gft$table_residence;
  4  5617            asti: mmt$ast_index;
  4  5618            segnum: ost$segment;
  4  5619       VAR sfid: gft$system_file_identifier;
  4  5620       VAR fde_p: gft$file_desc_entry_p);
  4  5621
  4  5622      gfp$assign_fde (residence, 0, sfid, fde_p);
 22  5623      IF sfid.residence <> gfc$tr_system THEN
 2C  5624        fde_p^.file_kind := gfc$fk_unnamed_file;
 2C  5625        fde_p^.open_count := 1;
 2C  5626        fde_p^.attach_count := 1;
 42  5627      ELSE
 42  5628        fde_p^.flags.global_template_file := TRUE;
 42  5629        fde_p^.queue_status := gfc$qs_global_shared;
 42  5630        fde_p^.attach_count := 0fff(16);
 42  5631        fde_p^.open_count := 0fff(16);
 5C  5632      IFEND;
 5C  5633      fde_p^.asti := asti;
 5C  5634      fde_p^.last_segment_number := segnum;
 5C  5635      fde_p^.global_task_id := jmv$jmtr_xcb.global_task_id;
 5C  5636      fde_p^.global_task_id := jmv$jmtr_xcb.global_task_id;
 5C  5637      fde_p^.file_hash := segnum;
 5C  5638      sfid.file_hash := segnum;
 5C  5639
 5C  5640    PROCEND assign_fde;
```

mmp$create_ssr_sdtx

```
       0  5642 { Purpose:
       0  5643 {   This procedure is called during deadstart to create a SDTX entry for the SSR.
       0  5644 {
       0  5645
       0  5646   PROCEDURE [XDCL] mmp$create_ssr_sdtx
       0  5647     (VAR sdt_entry: mmt$segment_descriptor;
       0  5648      VAR sdtx_entry: mmt$segment_descriptor_extended);
       0  5649
       0  5650     VAR
       0  5651       fde_p: gft$file_desc_entry_p,
       0  5652       sfid: gft$system_file_identifier;
       0  5653
       0  5654     sdtx_entry := mmv$default_sdtx_entry;
      16  5655     sdtx_entry.software_attribute_set := $mmt$software_attribute_set [mmc$sa_wired];
      16  5656     sdtx_entry.inheritance := mmc$si_none;
      16  5657
      16  5658     gfp$assign_fde (gfc$tr_system, 0, sfid, fde_p);
      4A  5659     fde_p^.file_kind := gfc$fk_monitor_only_unnamed;
      4A  5660     fde_p^.queue_status := gfc$qs_global_shared;
      4A  5661     fde_p^.attach_count := 0fff(16);
      4A  5662     fde_p^.open_count := 0fff(16);
      4A  5663
      4A  5664     mmp$asti (sdt_entry.ste.asid, fde_p^.asti);
      B4  5665     sdt_entry.asti := fde_p^.asti;
      B4  5666     fde_p^.global_task_id := jmv$jmtr_xcb.global_task_id;
      B4  5667     fde_p^.global_task_id := jmv$jmtr_xcb.global_task_id;
      B4  5668     fde_p^.file_hash := 0;
      B4  5669
      B4  5670   PROCEND mmp$create_ssr_sdtx;
       0  5671
```


link_page_to_segment_ds

```
       0  5673 { Purpose:
       0  5674 {   This procedure is called from mmp$pft_initialize to insert a page frame into the
       0  5675 {   thread which links all pages of a segment that are in memory.  There must be NO OTHER CALLERS
       0  5676 {   of this procedure, or the integrity of the links will be destroyed.
       0  5677
       0  5678   PROCEDURE [INLINE] link_page_to_segment_ds
       0  5679     (     pfti: mmt$page_frame_index;
       0  5680           pfte_p: ^mmt$page_frame_table_entry;
       0  5681           aste_p: ^mmt$active_segment_table_entry);
       0  5682
       0  5683 { Debug code
       0  5684
       0  5685     IF (pfte_p^.segment_link.fwd <> 0) AND (pfte_p^.segment_link.bkw <> 0) THEN
       0  5686       osp$system_error ('LINK PAGE TO SEGMENT ERROR.', NIL);
       0  5687     IFEND;
       0  5688
       0  5689 { End debug code
       0  5690
       0  5691     IF aste_p^.pft_link.fwd = 0 THEN
       0  5692       aste_p^.pft_link.fwd := pfti;
       0  5693       aste_p^.pft_link.bkw := pfti;
       0  5694     ELSE
       0  5695       mmv$pft_p^ [aste_p^.pft_link.bkw].segment_link.fwd := pfti;
       0  5696       pfte_p^.segment_link.bkw := aste_p^.pft_link.bkw;
       0  5697       aste_p^.pft_link.bkw := pfti;
       0  5698     IFEND;
       0  5699
       0  5700   PROCEND link_page_to_segment_ds;
       0  5701
```

mmp$sdtx_initialization

```
        0  5703
        0  5704    PROCEDURE [XDCL] mmp$sdtx_initialization;
        0  5705
        0  5706 {
        0  5707 {   The purpose of this request is to initialize the SDTX at system
        0  5708 { initialization time.
        0  5709 {
        0  5710 {       MMP$SDTX_INITIALIZATION (status)
        0  5711 {
        0  5712 { STATUS (output) This parameter specifies the request status.
        0  5713 {
        0  5714
        0  5715      VAR
        0  5716        sdt_p: mmt$max_sdt_p,
        0  5717        sdtx_entry: mmt$segment_descriptor_extended,
        0  5718        sdtx_p: mmt$max_sdtx_p,
        0  5719        xcb_p: ^ost$execution_control_block,
        0  5720        segnum: ost$segment,
        0  5721        tos_array_index: ost$ring,
        0  5722        rb: mmt$rb_ringi_segment_request;
        0  5723
        0  5724
        0  5725 {  Set pointer to SDTX in XCB.
        0  5726
        0  5727      xcb_p := ^jmv$jmtr_xcb;
        4  5728      xcb_p^.sdt_offset := #OFFSET (^jmv$sdt);
        4  5729      xcb_p^.sdtx_offset := #OFFSET (^jmv$sdtx);
        4  5730
        4  5731      mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
        4  5732
        4  5733
        4  5734 { Initialize the SDTX entry for each valid SDT entry. By default each segment is assumed
        4  5735 { to be pageable shared system template segments. Exceptions to this rule must be accounted for
        4  5736 { by specifically changing attributes later in this proc.
        4  5737
        4  5738      sdtx_entry := mmv$default_sdtx_entry;
       64  5739      sdtx_entry.inheritance := mmc$si_share_segment;
       64  5740      sdtx_entry.open_validating_ring_number := 0;
       64  5741
       64  5742      FOR segnum := xcb_p^.xp.segment_table_length DOWNTO 0 DO
       7E  5743        IF sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry THEN
       92  5744          sdtx_p^.sdtx_table [segnum] := sdtx_entry;
       AC  5745          mmp$set_segment_access_rights (sdt_p^.st [segnum], sdtx_p^.sdtx_table [segnum]);
       C4  5746        IFEND;
       C4  5747      FOREND;
       CC  5748
       CC  5749 { Set the page table to be read only.
       CC  5750
       CC  5751      sdtx_p^.sdtx_table [osc$segnum_page_table].access_rights := mmc$sar_read;
       CC  5752
       CC  5753
       CC  5754 { Set software attributes for special segments.
       CC  5755
       CC  5756      sdtx_p^.sdtx_table [osc$segnum_page_table].software_attribute_set :=
       CC  5757            $mmt$software_attribute_set [mmc$sa_wired];
       CC  5758      sdtx_p^.sdtx_table [#SEGMENT (#LOC (osv$mainframe_wired_heap^))].software_attribute_set :=
```


mmp$sdtx_initialization

```
       CC  5759            $mmt$software_attribute_set [mmc$sa_wired];
       CC  5760      sdtx_p^.sdtx_table [#SEGMENT (#LOC (osv$mainframe_wired_cb_heap^))].
       CC  5761            software_attribute_set := $mmt$software_attribute_set [mmc$sa_wired];
       CC  5762      sdtx_p^.sdtx_table [#SEGMENT (#LOC (nav$network_wired_heap^))].software_attribute_set :=
       CC  5763            $mmt$software_attribute_set [mmc$sa_wired];
       CC  5764      sdtx_p^.sdtx_table [osc$segnum_job_fixed_heap].software_attribute_set :=
       CC  5765            $mmt$software_attribute_set [mmc$sa_fixed];
       CC  5766
       CC  5767      sdtx_p^.sdtx_table [osc$segnum_job_fixed_heap].open_validating_ring_number := 1;
       CC  5768
       CC  5769
       CC  5770 { The file server wired heap is shared with the network wired heap.
       CC  5771 { The file server is the ending portion of this heap.
       CC  5772
       CC  5773      dfv$server_wired_heap := #ADDRESS (#RING (nav$network_wired_heap), #SEGMENT (nav$network_wired_heap),
       CC  5774            nac$network_heap_size);
       CC  5775      osp$reset_heap (dfv$server_wired_heap, 3fffffff(16) - nac$network_heap_size, TRUE, 2);
      180  5776      osp$reset_heap (nav$network_wired_heap, nac$network_heap_size, TRUE, nac$heap_algorithm);
      1A4  5777      osp$reset_heap (nav$network_paged_heap, nac$network_heap_size, TRUE, nac$heap_algorithm);
      1CC  5778
      1CC  5779
      1CC  5780 {  For ring 1, 2, and 3 stack segments, set software attributes.
      1CC  5781
      1CC  5782      FOR tos_array_index := 1 TO 3 DO
      1D4  5783        segnum := xcb_p^.xp.tos_registers [tos_array_index].pva.seg;
      1D4  5784        IF sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry THEN
      1F4  5785          sdtx_p^.sdtx_table [segnum].software_attribute_set := $mmt$software_attribute_set [mmc$sa_stack];
      1F4  5786          sdtx_p^.sdtx_table [segnum].open_validating_ring_number := 1;
      1F4  5787          sdtx_p^.sdtx_table [segnum].inheritance := mmc$si_new_segment;
      214  5788        IFEND;
      214  5789      FOREND;
      218  5790
      218  5791    PROCEND mmp$sdtx_initialization;
        0  5792
```

MMP$COMMIT_MEMORY

```
 0  5795
 0  5796   PROCEDURE [XDCL] mmp$commit_memory;
 0  5797
 0  5798     VAR
 0  5799       rb: mmt$rb_ring1_segment_request;
 0  5800
 0  5801 {
 0  5802 {   This procedure is called immediately after closing
 0  5803 {and detaching the image file.  It initializes the request
 0  5804 {block and makes the call to monitor.  In monitor, the page table
 0  5805 {is scanned from the deadstart upper bound to the upperbound, and
 0  5806 {page frame table entries are linked onto the free page queue.
 0  5807 {
 0  5808
 0  5809     rb.reqcode := syc$rc_ring1_segment_request;
 4  5810     rb.request := mmc$sr1_commit_memory;
 4  5811
 4  5812     mmp$issue_ring1_segment_request (rb);
26  5813
26  5814   PROCEND mmp$commit_memory;
```


MMP$FREE_IMAGE_PAGES

```
 0  5817
 0  5818   PROCEDURE [XDCL] mmp$free_image_pages;
 0  5819
 0  5820     VAR
 0  5821       rb: mmt$rb_ring1_segment_request;
 0  5822
 0  5823 {
 0  5824 {    This procedure is called prior to closing and detaching the
 0  5825 {image file.  It initializes the request block and makes a call to
 0  5826 {monitor.  From monitor, a procedure is called which scans the
 0  5827 {entire page table.  Valid page table entries greater than the
 0  5828 {deadstart upper bound are deleted.
 0  5829 {
 0  5830
 0  5831     rb.reqcode := syc$rc_ring1_segment_request;
 4  5832     rb.request := mmc$sr1_free_image_pages;
 4  5833
 4  5834     mmp$issue_ring1_segment_request (rb);
26  5835
26  5836   PROCEND mmp$free_image_pages;
```

MMP$DEFINE_IMAGE_FILE

```
       o  5839
       o  5840    PROCEDURE [XDCL] mmp$define_image_file
       4  5841      (     sfid: dmt$system_file_id;
       4  5842            length: 0 .. 0ffffffff(16));
       4  5843
       4  5844 {
       4  5845 {    This procedure is called during deadstart when the image file is
       4  5846 {attached and opened.  It initializes the fields in mmv$image_file.
       4  5847 {
       4  5848 {Parameters:
       4  5849 {  SFID: (dmt$system_file_id) This is the system file id of the image file.
       4  5850 {
       4  5851 {  LENGTH: This is the length of the information immediately
       4  5852 {           preceding the actual memory image on the image
       4  5853 {           file segment.
       4  5854 {
       4  5855
       4  5856      mmv$image_file.active := TRUE;
       4  5857      mmv$image_file.sfid := sfid;
       4  5858      mmv$image_file.file_offset := length;
       4  5859
       4  5860    PROCEND mmp$define_image_file;
       o  5861 MODEND mmm$deadstart_initialization;
```

**** I=$05578173AS0102D19890821T183254 L=ZZXXLIST B=LGO DA=NONE LD=R RC=NONE OPT=SCHED EL=F LF=CS612 PAD=0

MMP$DEFINE_IMAGE_FILE

```
          ERROR    LINE  TEXT
WARNING   CY 821    5263   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    5266   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 793    5273   #SIZE returning rounded bit size.
WARNING   CY 821    5291   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    5319   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    5435   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    5499   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    5503   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 793    5575   #SIZE returning rounded bit size.

LEVEL SUMMARY
   ****   9    warning diagnostics
```

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
access_rights	2490	5751/M							
active	4801	5856/M							
age	1094	5495/M	5527	5535					
amc$file_byte_limit	4280	4283	4285						
amt$file_byte_address	4283	4244	4625						
amt$file_limit	4285	4248							
asid	838	5327	5330/M	5411					
asid	863	5466/M							
asid	1067	5555/P							
asid	2468	5304	5312	5358	5594/M	5664/P			
asid	4984	4989/M	4992/S	4993/M	4993				
asid	5000	5006/M	5008/M	5008	5008				
asid	5170	5360/M	5360/S	5360/M	5360	5431/M	5431/S	5431/M	5431
asid	5170	5555/M	5555/M	5555	5555				
asid	5174	5358/M	5360/P	5425	5425	5449/M			
asid	5182	5327/M	5328	5328	5328	5411/M	5412	5422	5425
		5425	5431/P	5440	5449	5459	5466	5481	5483
asid	5646	5664/M	5664/S	5664/M	5664				
asid1	5183	5323/M	5328						
asid2	5184	5324/M	5328						
asid3	5185	5325/M	5328						
asid_seq_p	5186	5300/P	5301	5302					
asid_size	5187	5299/M	5300/P						
asidq	5188	5346	5425	5425	5430	5455			
asidq_p	5189	5346/M	5358/M	5359/M	5360/P	5364/M	5365	5366/M	5372/P
		5372/P	5373	5377/M	5377	5378/M	5380/M	5383/P	5384/P
		5390/M	5391/M	5395/M	5397/M	5430/M	5433/M	5434	5435/P
		5435/P	5436/M	5437/M	5438/M	5439/M	5441/M	5446/M	5447/M
		5449/M	5450/M	5451/M	5455/M	5456	5457	5458	5460
asidq_type	5173	5188	5189						
assign_active	2495	5375/M							
assign_fde	5615	5047	5372	5435	5601	5640			
aste_p	1095	5494/M	5502	5533/M	5544/M				
aste_p	5170	5503	5503/M	5503/M	5503/S	5503	5503/M		
aste_p	5177	5364/M	5365	5366/M	5372/P	5373	5377/M	5378/M	5384/P
		5390/M	5433/M	5434	5435/P	5437/M	5438/M	5439/M	5441/M
		5446/M	5456						
aste_p	5190	5456/M	5477	5478/M	5478	5494	5503/P	5504	
aste_p	5681	5691	5692/M	5693/M	5695/S	5696	5697/M		
asti	2437	5361/M	5665/M						
asti	4243	5633/M	5664/P	5665					
asti	4986	4990/M	4992/M	4992	4995				
asti	5003	5005/M	5006/S	5006/S	5006/S	5007/S			
asti	5170	5360/M	5360/M	5360	5360	5431/M	5431/M	5431	5431
asti	5170	5555/M	5555/S	5555/S	5555/S	5555/S			
asti	5191	5360/P	5361	5363	5364/S	5372/P	5431/P	5432	5433/S
		5435/P							
asti	5617	5633							
asti	5646	5664/M	5664/M	5664	5664				
attach_count	4238	5626/M	5630/M	5661/M					
bits	4978	4992	5006	5006	5006	5007	5360	5431	5555

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		5555	5555	5555	5664				
bkw	1036	5503	5503/M	5503/S	5503/M	5503	5503/M	5510/M	5514
		5515/M	5531/M	5538/M	5547/M	5685	5693/M	5695/S	5696/M
		5696	5697/M						
block_index	521	4459/S	5311/S						
block_number	520	4459/S	5311/S						
block_p	1825	4459	5311						
boot_asids	5192	5318	5323	5324	5325				
boot_asids_seq_p	5193	5318/M	5319/P						
byte_p	2252	5142							
c	844	5129/M	5566/M						
c170	1619	1611							
cbc_seq_p	5194	5263/P	5264	5265					
cmt$element_name	3687	3630							
code_data	3528	5323							
count	1015	5513/M	5513	5542/M	5542	5562			
count	5195	5468	5469						
current	2225	5304/M							
deadstart_upper	4908	5520							
default	2247	5140/M	5142/M						
dfc$command_record_bytes	186	194							
dfc$division_overwrite_words	173	201							
dfc$esm_command_record_size	194	202							
dfc$esm_header_record_size	195	202							
dfc$esm_maintenance_buf_size	174	205							
dfc$esm_memory_base_shift	180	202	203	203					
dfc$header_record_bytes	185	195							
dfc$max_esm_memory_size	175	204							
dfc$max_number_of_mainframes	182	167							
dfc$min_data_record_bytes	190	201							
dfc$min_esm_division_size	200	204							
dft$mainframe_set	167	117	118	276	277				
dfv$server_wired_heap	4921	5773/M	5775/P						
dmt$system_file_id	216	149	4607	4611	4612	4620	4624	4802	5841
dpc$console_row_size	3693	3612							
dsc$boot_asids	3522	5319/P							
dsc$dftb_mr_number_of_registers	3903	4109	4111						
dsc$ssr_pages_to_dump	3859	5576/P							
dsc$ssr_sds_number_of_disk_errs	3552	3623	3624	3625					
dsc$ssr_sds_number_of_mf_errors	3553	3651	3652						
dsc$ssr_sds_number_of_nos_id	3554	3671							
dsp$allocate_continuous_memory	3471	5251	5259	5263	5300				
dsp$convert_seq_p_to_r_pointer	3479	3498	5575						
dsp$fetch_boot_data	3513	5319							
dsp$store_data_in_ssr	3535	5576							
dst$boot_asids	3527	5192							
dst$boot_data_kinds	3520	3514							
dst$dftb_control_word	3991	3588							
dst$dftb_cw_dft_pp_number	3964	3996							
dst$dftb_cw_pointer_words	3965	3993							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dst$dftb_cw_revision_level	3966	3995	4169	4178					
dst$dftb_date_and_time	4034	3661							
dst$dftb_date_and_time_data	4039	4035	4173						
dst$dftb_date_and_time_field	4041	3677							
dst$dftb_dft_analysis_code	3967	4022	4180						
dst$dftb_element_size	3968	4030	4031	4126	4130	4141	4144	4153	4161
		4162	4163	4164	4165	4166	4167	4168	4176
		4177	4185						
dst$dftb_error_buffer_flags	4059	4024							
dst$dftb_fault_symptom_code	4070	3664	4077						
dst$dftb_mdb_data_header_id	3969	4133							
dst$dftb_mdb_mrb_word_index	3970	4122							
dst$dftb_mdb_pfs_error_id	3971	4132							
dst$dftb_mec_error_element	3973	4089	4090	4091					
dst$dftb_mec_threshold	3972	4087	4088						
dst$dftb_mr_header_entry	4104	4110							
dst$dftb_mr_register_header	4109	4114							
dst$dftb_mr_register_list	4111	4115							
dst$dftb_mrb_offset	3974	4025							
dst$dftb_mrt_element_index	3975	3662	4019						
dst$dftb_os_action_code	3976	4020							
dst$dftb_priority	3977	4021	4123						
dst$dftb_secded_address	3979	4098							
dst$dftb_secded_count	3978	4097							
dst$dftb_secded_syndrome	3980	4099							
dst$dftb_sequence_number	3981	3994	4023	4124	4181				
dst$dftb_ssb_mdb_ordinal	3982	4143							
dst$dftb_ssb_unlogged	3983	4142							
dst$dftb_stat_buffer_type	3984	4187							
dst$dftb_structure_length	3985	3997	3998	4139	4140	4151	4152		
dst$r_pointer	3502	3481	4080	5208					
dst$ssr_bmb_r_register	3574	3580	3581						
dst$ssr_entry_name	3605	3536	3592						
dst$ssr_sds_disk_error_entry	3628	3625							
dst$ssr_sds_disk_errors	3622	3615							
dst$ssr_sds_ds_performed_code	3634	3641							
dst$ssr_sds_general_info	3640	3614							
dst$ssr_sds_mainframe_errors	3650	3616							
dst$ssr_sds_mf_error_data	3655	3652							
dst$ssr_sds_mf_error_entry	3660	3657							
dst$ssr_sds_nos_nbe_words	3667	3617							
dst$ssr_sds_timestamp	3674	3629	3647						
dsv$ssr_size	4700	5257							
eoi_byte_address	4244	5499	5500/M						
fde_p	4419	4434/M	4435	4436/M					
fde_p	4633	4639	4640						
fde_p	5018	5050/M	5050	5050/M					
fde_p	5025	5047/P	5050/P	5051/M					
fde_p	5170	5384	5384						
fde_p	5178	5372/P	5380/M	5383/P	5391/M	5435/P	5436/M	5447/M	5460

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
fde_p	5196	5460/M	5499	5499	5500/M	5601/P	5602/M	5603/M	
fde_p	5620	5622/P	5624/M	5625/M	5626/M	5628/M	5629/M	5630/M	5631/M
		5633/M	5634/M	5635/M	5636/M	5637/M			
fde_p	5651	5658/P	5659/M	5660/M	5661/M	5662/M	5664/P	5665	5666/M
		5667/M	5668/M						
file_entry_index	19	4434	5050						
file_hash	21	4435	5050	5638/M					
file_hash	4241	4435	5050	5637/M	5668/M				
file_kind	4240	5436/M	5624/M	5659/M					
file_limit	4248	5602/M							
file_offset	4803	5858/M							
first_image_pfti	5197	5520/M	5526						
flags	4234	4640	5051/M	5384	5628/M				
found	5198	5468							
fwd	1037	5498/M	5498	5503	5503	5503/M	5503/M	5508	5512/M
		5530/M	5540/M	5540	5541/M	5546/M	5685	5691	5692/M
		5695/M							
fwd_link	5199	5508/M	5509	5510/S					
gfc$fde_size	4451	4434	5050						
gfc$fde_table_base	4449	4434	4450	5050					
gfc$fk_catalog	4317	4329							
gfc$fk_job_local_file	4319	4328							
gfc$fk_monitor_only_unnamed	4322	5436	5659						
gfc$fk_unnamed_file	4320	5624							
gfc$fm_mass_storage_file	4332	4257							
gfc$fm_served_file	4333	4260							
gfc$qs_global_shared	4361	5629	5660						
gfc$tr_job	32	4428	5047/P	5050	5370				
gfc$tr_null_residence	32	5046							
gfc$tr_system	32	4427	5050	5368	5435/P	5601/P	5623	5658/P	
gfp$assign_fde	4214	5622	5658						
gfp$get_fde_p	4418	4439	5050						
gft$allocation_unit_size	4291	4246							
gft$attach_count	4296	4237	4238						
gft$fde_flags	4266	4234							
gft$file_desc_entry_p	4223	4218	4419	5025	5178	5196	5620	5651	
gft$file_descriptor_entry	4231	4223	4236	4655					
gft$file_descriptor_index	29	19							
gft$file_kind	4313	4240	4325						
gft$file_media	4332	4256							
gft$locked_file_desc_entry_p	4655	4633							
gft$open_count	4350	4239	4366						
gft$queue_status	4361	4249							
gft$segment_lock_info	4365	4242							
gft$signature_lock	4338	4232							
gft$system_file_identifier	18	14	216	1070	2486	2578	2737	4217	4418
		4468	5619	5652					
gft$table_residence	32	20	4215	5221	5616				
gft$transfer_unit_size	4302	4247							
gfv$null_sfid	14	5073/P							
global_task_id	2698	5635	5636	5666	5667				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
global_task_id	4253	5635/M	5636/M	5666/M	5667/M				
global_template_file	4270	4640	5051/M	5384	5628/M				
gpql	2261	5146/M							
i	4985	4991							
i	5107	5112/M	5113	5114/M	5114	5115			
i	5170	5360	5431						
i	5200	5243/M	5244	5245/M	5250	5253	5266	5341	5342/S
		5348/S	5350/S	5352/S	5358/S	5361/S	5372/P	5373/S	5375/S
		5379/S	5380/S	5383/P	5383/S	5469	5553	5554/S	5555/P
		5555/S							
i	5646	5664							
i#real_memory_address	3509	3492	5575						
ijl_ordinal	1063	5378/M	5439/M						
ijl_ordinal	1088	5496/M							
ijl_ordinal	4455	4459/S	4459/S						
ijl_ordinal	5170	5311/S	5311/S						
ijle_p	4456	4459/M							
ijle_p	5170	5311/M							
ijle_p	5201	5311/P	5312/M	5397					
in_use	1064	5365	5366/M	5434	5437/M	5554			
include_pages_in_dump	1071	5384/P	5441/M	5504					
include_pages_in_dump	4635	4642/M	4644/M	4647/M					
include_pages_in_dump	5170	5384/M	5384/M	5384/M					
index_ma	5108	5137	5138/S	5140/S	5140/S	5142/S	5142/S		
index_p	1838	4459	5311						
inheritance	2487	5600/M	5656/M	5739/M	5787/M				
integer_p	2250	5140							
iot$io_error	1053	150	1096						
iot$transfer_count	2875	2863							
ipti	5202	5468	5470/M	5470	5471	5472/M	5474/S	5474/S	
j	5203	5423/M	5425	5425/S	5425	5425/S	5426/M	5426	5429
		5430/S	5455/S						
jmc$detached_job_wait_time_max	1916	1913							
jmc$highest_det_job_wait_time	1926	1916	1927						
jmc$highest_prio_age_interval	2805	2796	2806						
jmc$highest_service_accumulator	695	696							
jmc$highest_service_factor_valu	2829	2822							
jmc$highest_working_set_size	1962	1953	1963	1965	1967	1969			
jmc$ies_job_swapped	453	462							
jmc$ies_swapin_in_progress	452	461							
jmc$iss_idle_tasks_initiated	468	495							
jmc$iss_swapin_io_complete	493	496							
jmc$iss_swapin_requested	489	496							
jmc$iss_swapout_complete	488	495							
jmc$iss_swapped_io_cannot_init	479	506							
jmc$iss_swapped_no_io	470	505							
jmc$keyword_offset_maximum	712	1954	2797						
jmc$kjl_maximum_entries	237	230	231	647					
jmc$kol_maximum_entries	247	232							
jmc$max_active_jobs	228	2778	2786	2787					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$max_ajl_ord	229	222	228						
jmc$max_dispatching_control	403	407							
jmc$max_dispatching_priority	325	285	288	289					
jmc$max_ijl_index_count	529	1836							
jmc$maximum_job_classes	625	628							
jmc$maximum_job_count	244	237							
jmc$maximum_output_count	254	247							
jmc$maximum_service_classes	728	731							
jmc$min_dispatching_control	402	406							
jmc$null_service_class	721	722							
jmc$priority_aging_interval_max	2796	2793							
jmc$priority_p1	339	286							
jmc$priority_p10	348	287							
jmc$priority_p14	352	287							
jmc$priority_p8	346	286							
jmc$required_offset	710	1968							
jmc$reserved_ajls	233	228							
jmc$service_accumulator_maximum	687	684							
jmc$service_factor_value_max	2822	2819							
jmc$system_default_offset	711	712	1970						
jmc$system_supplied_name_size	952	949							
jmc$unlimited_offset	708	697	1917	1928	1964	2807			
jmc$unspecified_offset	709	1966							
jmc$working_set_size_maximum	1953	1950							
jmp$get_ijle_p	4455	4461	5311						
jmt$ajl_ordinal	222	86							
jmt$delayed_swapin_work	269	116	273						
jmt$detached_job_wait_time	1913	1898							
jmt$dispatching_control	373	2761							
jmt$dispatching_control_index	406	363	373						
jmt$dispatching_controls	376	374							
jmt$dispatching_priority	285	98	364	365	366	378	2709	2711	
jmt$ijl_block_index	525	521	1838						
jmt$ijl_block_number	524	520	1826	1827					
jmt$ijl_dispatching_control	362	99							
jmt$ijl_entry_status	448	85							
jmt$ijl_ordinal	519	105	133	967	968	1063	1088	1848	1888
		4455							
jmt$ijl_p	1824	1819							
jmt$ijl_page_fault_count	544	539	540	541					
jmt$ijl_page_stats	538	534							
jmt$ijl_service_class_stats	532	120							
jmt$ijl_statistics	577	119							
jmt$ijl_swap_count	553	549	550						
jmt$ijl_swap_counts	548	139	535						
jmt$ijl_swap_status	466	88	89	90					
jmt$initiated_job_list_block	1835	1841							
jmt$initiated_job_list_entry	82	42	1838	1887	4456	5201			
jmt$initiated_job_list_p	1841	1825							
jmt$input_file_location	667	662							
jmt$job_abort_disposition	676	660							
jmt$job_class	828	144							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$job_control_block	1869	1855							
jmt$job_mode	631	101							
jmt$job_priority	636	141	142	2770	2771	2772	2773		
jmt$job_recovery_disposition	679	661							
jmt$job_system_id	1932	1884							
jmt$kjl_index	647	87	1932						
jmt$maximum_active_jobs	2778	2755							
jmt$priority_aging_interval	2793	2763							
jmt$queue_file_ijl_information	659	126							
jmt$scheduling_data	132	110							
jmt$scheduling_priority	2769	2762							
jmt$service_accumulator	684	134	135	136	2753	2754			
jmt$service_class_index	731	145	2746	2756					
jmt$service_class_name	2811	2748	2749						
jmt$service_factor_value	2819	2757							
jmt$service_factors	2815	2757							
jmt$swap_data	148	112							
jmt$swapout_reasons	734	140							
jmt$swapped_job_entry	749	43	157	1907					
jmt$system_supplied_name	949	83	1628	1882					
jmt$task_time_slice	416	396	397						
jmt$time_slice_values	395	380	2722						
jmt$user_supplied_name	1936	1883							
jmt$working_set_size	1950	1894	1895						
jmv$ijl_p	1819	4459	5311						
jmv$jmtr_xcb	4723	5034	5279/P	5341	5635	5636	5666	5667	5727
jmv$sdt	4707	5728							
jmv$sdtx	4717	5729							
jmv$system_ijl_ordinal	1848	5311/P	5378	5439	5496				
job_fixed_asid	95	5312/M							
job_page_queue_list	111	5397							
job_stack	3529	5324							
jsc$isqi_swapped_io_completed	972	974							
jsc$isqi_swapped_io_not_init	971	974							
jst$changed_asid_entry	65	56							
jst$ijl_swap_queue_id	971	966							
jst$ijl_swap_queue_link	965	94							
jst$io_control_information	979	113							
jst$swap_file_descriptor	41	114							
jst$swapped_page_descriptor	50	48	5571						
jst$swapped_page_descriptors	47	44							
jsv$swapped_page_entry_size	4730	5571/M							
last_asid	5204	5403/M	5422	5459/M					
last_link_p	5206	5522/M	5538/M	5539/M	5540/M				
last_segment_number	4252	5603/M	5634/M						
length	3506	3496/M	5575/M						
length	5842	5858							
link	1014	5498	5508	5512/M	5514	5515/M	5522	5540	5541/M
link	1085	5498/M	5510/M	5530/M	5531/M	5539			
link_page_to_segment_ds	5678	5503	5700						
locked_page	1091	5497/M							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lower	4907	5120/M	5120	5235	5243				
ma	2262	5137	5137	5138	5140/M	5140	5142/M	5142	
mf_wired_asid_p	5207	5302	5303						
mlc$c180	1619	1613							
mlc$error	1202	1204	1211	1219	1230	1238	1246	1253	1261
		1268	1277	1285	1294	1302	1310	1318	1324
		1333	1341	1349	1358	1366	1373	1383	1393
		1401	1411	1418	1426	1435	1439		
mlc$fetch_link_partner_info_req	1505	1509							
mlc$max_ant_entries	1574	1620	1650						
mlc$max_message_length	1190	1463							
mlc$max_permits	1192	1617	1643						
mlc$max_queued_messages	1196	1462	1464	1465	1466	1641			
mlc$max_signons_per_job	1573	1595							
mlc$max_signons_per_system_name	1195	1630							
mlc$max_sn_entry	1571	1621	1648						
mlc$nosve_not_up	1310	1493							
mlc$ok	1318	1493							
mlc$sign_on_req	1497	1509							
mlc$unique_name	1193	1194							
mlt$ant_entry	1577	1650							
mlt$ant_index	1620	1581	1582	1583	1584	1648	1649		
mlt$application_name	1460	1486	1489	1580	1623	1633	1634	1639	
mlt$arbitrary_info	1461	1624							
mlt$c170_c180_flag	1619	1610							
mlt$direction	1479	1483	1635						
mlt$handler	1489	1592							
mlt$int_receive_list	1641	1586							
mlt$int_receive_list_entry	1622	1627	1642						
mlt$max_messages	1462	1585							
mlt$message_length	1463	1626							
mlt$operation	1508	1593							
mlt$permit_list	1643	1587							
mlt$permit_list_entry	1638	1643							
mlt$receive_entry	1468	1467							
mlt$shared_segment	1644	4961							
mlt$signal	1477	1491							
mlt$signal_record	1480	1477							
mlt$signaler_application_info	1485	1490							
mlt$system_name	1609	1588							
mlv$shared_segment	4961	5284/S							
mmc$assign_active_escaped	2527	5375							
mmc$assign_active_null	2526	2527							
mmc$cell_pointer	2174	2179							
mmc$first_loader_predefined_seg	2268	2270	4638	5384					
mmc$first_transient_segment	2270	4958							
mmc$heap_pointer	2175	2183							
mmc$kw_asid	2100	2136							
mmc$kw_clear_space	2098	2123							
mmc$kw_current_segment_length	2097	2117							
mmc$kw_error_exit_procedure	2099	2127							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES					
mmc$kw_gl_key	2099	2121					
mmc$kw_hardware_attributes	2101	2130					
mmc$kw_inheritance	2101	2138					
mmc$kw_max_segment_length	2098	2119					
mmc$kw_preset_value	2100	2125					
mmc$kw_ps_transfer_size	2102	2146					
mmc$kw_ring_numbers	2096	2112					
mmc$kw_segment_access_control	2100	2134					
mmc$kw_segment_number	2097	2115					
mmc$kw_shadow_segment	2102	2140					
mmc$kw_software_attributes	2099	2132					
mmc$kw_wired_segment	2102	2143					
mmc$lp_not_locked	1107	5497					
mmc$num_loader_predefined_segs	2269	2270					
mmc$pq_avail	766	812					
mmc$pq_free	765	824	5521/S	5532	5543		
mmc$pq_job_base	813	5367	5394				
mmc$pq_job_fixed	806	813	825	5351			
mmc$pq_job_working_set	808	825	826	5355	5374		
mmc$pq_shared_first_site	816	820					
mmc$pq_shared_num_sites	817	820					
mmc$pq_shared_other	775	815					
mmc$pq_shared_site_01	777	816					
mmc$pq_shared_site_25	801	821					
mmc$pq_shared_task_service	770	814	5353	5374			
mmc$pq_swapped_io_error	804	824					
mmc$pq_wired	768	811	5349	5438	5450/S	5451	
mmc$sa_fixed	2161	5350	5765				
mmc$sa_stack	2162	5379	5785				
mmc$sa_wired	2161	5348	5598	5655	5757	5759	5761 5763
mmc$sar_read	2554	5751					
mmc$segment_fault_processor_id	3173	3227					
mmc$sequence_pointer	2174	2181					
mmc$si_new_segment	2193	5787					
mmc$si_none	2192	5656					
mmc$si_share_segment	2192	5600	5739				
mmc$sri_change_swap_file_queue	4597	4604					
mmc$sri_commit_memory	4589	5810					
mmc$sri_delete_job_seg_by_sfid	4599	4605					
mmc$sri_delete_seg_segnum	4586	4608					
mmc$sri_delete_seg_sfid	4587	4603					
mmc$sri_detach_file	4590	4603					
mmc$sri_end_job_recovery	4594	4614					
mmc$sri_flush_avail_modified	4601	4606					
mmc$sri_flush_delete_seg_sfid	4591	4603					
mmc$sri_flush_seg_segnum	4592	4604	5067				
mmc$sri_free_image_pages	4588	5832					
mmc$sri_get_highest_offset	4598	4623					
mmc$sri_make_mfw_cache	4595	4617	5288				
mmc$sri_remove_detached_pages	4600	4605					
mmc$sri_remove_job_shared_pages	4596	4619					
mmc$sri_replace_sfid	4593	4610					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmc$sr_assign_file_to_disk	2276	2287						
mmc$sr_complete_seg_sft_entry	2278	2287						
mmc$sr_fetch_cyclic_aging_int	2281	2295						
mmc$sr_fetch_max_ws_size	2277	2289						
mmc$sr_fetch_min_ws_size	2279	2291						
mmc$sr_fetch_page_aging_int	2280	2293						
mmc$sr_store_cyclic_aging_int	2281	2295						
mmc$sr_store_max_ws_size	2277	2289						
mmc$sr_store_min_ws_size	2279	2291						
mmc$sr_store_page_aging_int	2280	2293						
mmc$ssk_none	2610	2582						
mmc$ssk_segment_number	2611	2580						
mmp$add_global_template_segment	5018	5053						
mmp$asid	4999	5010	5555					
mmp$assign_mass_storage	4466	5073						
mmp$asti	4980	4996	5360	5431	5664			
mmp$commit_memory	5796	5814						
mmp$convert_ps_transfer_size	4477	4502	5038					
mmp$create_ssr_sdtx	5646	5670						
mmp$define_image_file	5840	5860						
mmp$free_image_pages	5818	5836						
mmp$get_max_sdt_pointer	4521	4529	5070					
mmp$get_max_sdt_sdtx_pointer	4533	4543	5279	5731				
mmp$get_sdt_entry_p	4547	5039						
mmp$get_sdt_entry_p	4549	4551/M	5039/M					
mmp$get_sdtx_entry_p	4559	5040						
mmp$get_sdtx_entry_p	4561	4563/M	5040/M					
mmp$initialize	5103	5153						
mmp$issue_ring1_segment_request	4571	5076	5291	5812	5834			
mmp$pft_initialize	5170	5608						
mmp$sdtx_initialization	5704	5151	5791					
mmp$set_include_pages_in_dump	4631	4650	5383					
mmp$set_segment_access_rights	4658	5745						
mmp$write_all_segments_to_disk	5056	5082						
mmt$active_segment_table	1075	4810						
mmt$active_segment_table_entry	1060	53	1076	1095	5177	5190	5681	
mmt$ast_index	992	68	156	2437	4243	4613	4981	4999 5003
		5191	5617					
mmt$attribute_keyword	2096	2111						
mmt$continue_bit_count	4825	4821						
mmt$eoi_state	4379	4245						
mmt$global_page_queue_index	824	1027						
mmt$global_page_queue_list	1027	2244	4831					
mmt$global_page_queue_list_ent	1017	1027						
mmt$hardware_attribute_set	2165	2131						
mmt$hardware_attributes	2153	2165						
mmt$image_file	4799	4793						
mmt$job_page_queue_index	825	751	1028					
mmt$job_page_queue_list	1028	111						
mmt$link	1035	1014	1061	1085	1086	5206		
mmt$lock_segment_status	2590	2491						
mmt$locked_page	1107	1091						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$mainframe_wired_asid	2224	2220	5207	5299					
mmt$manage_memory_utility	2260	4841							
mmt$max_sdt	2447	2451	4707						
mmt$max_sdt_p	2451	4523	4535	5064	5223	5716			
mmt$max_sdtx	2515	2519							
mmt$max_sdtx_p	2519	4536	5224	5718					
mmt$memory_reserve_request	995	104							
mmt$mmu_gpql_default	2244	2261							
mmt$mmu_ma_data	2258	2262							
mmt$mmu_ma_info	2246	2258							
mmt$mmu_memory_attributes	2239	2242	2243	2258	5108				
mmt$mmu_value_types	2241	2248							
mmt$monitor_segment_request	2276	2286							
mmt$page_age	1114	1094	1118	1118					
mmt$page_frame_index	988	980	982	983	984	997	998	1037	1037
		4748	4773	5212	5679				
mmt$page_frame_queue_id	826	981	1069	1089	5175	5219			
mmt$page_frame_table	1100	4862							
mmt$page_frame_table_entry	1084	51	1100	5209	5250	5680			
mmt$page_queue_list_entry	1013	1018	1028	5176	5218				
mmt$pfti_array	4769	4763							
mmt$rb_ring1_segment_request	4580	4572	5062	5220	5722	5799	5821		
mmt$reassignable_page_frames	4784	4781							
mmt$sdtx_stream_data	2498	2494							
mmt$segment_access_condition	3200	3228							
mmt$segment_access_rights	2554	2490							
mmt$segment_access_state	2560	2485							
mmt$segment_descriptor	2434	2444	2448	4549	4634	4658	5019	5027	5226
		5227	5227	5647					
mmt$segment_descriptor_extended	2483	2512	2516	4561	4564	4659	4879	5020	5028
		5040	5229	5648	5717				
mmt$segment_descriptor_table_ex	2511	4717							
mmt$segment_inheritance	2192	2139	2487						
mmt$segment_pointer_kind	2174	2178							
mmt$segment_reservation_state	2600	2488							
mmt$shadow_info	2575	2492							
mmt$shadow_reference_info	2832	2735							
mmt$shadow_segment_kind	2610	2579							
mmt$software_attribute_set	2167	2133	2489	5598	5655	5757	5759	5761	5763
		5765	5785						
mmt$software_attributes	2161	2167							
mmt$xcb_page_wait_info	2843	2721							
mmu_boolean	2241	2253							
mmu_byte	2241	2251	5141						
mmu_integer	2241	2249	5139						
mmv$a_divisor	4737	4989	5008	5008	5115/M	5116	5360	5431	5555
		5555	5664						
mmv$a_mult	4738	4989	4989	5008	5116/M	5360	5360	5431	5431
		5555	5664	5664					
mmv$ast_p	4810	5242	5271/P	5271/P	5363	5364	5432	5433	5554
		5555/P							
mmv$continue_bit_count_p	4821	5265	5274/P	5274/P	5474/M	5474	5566		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmv$default_sdtx_entry	4878	5596	5654	5738					
mmv$gpql	4831	5146	5395	5450	5521				
mmv$image_file	4793	5856/M	5857/M	5858/M					
mmv$manage_memory_utility	4841	5137	5137	5138	5140/M	5140	5142/M	5142	5146/M
mmv$mf_wired_asid	2220	5303/M	5304/M	5305/M					
mmv$number_free_astes	4739	5241/M	5385/M	5385	5443/M	5443			
mmv$pages_per_new_page_fault	4847	5582/M	5583	5584/M					
mmv$pages_to_dump_p	2233	5261	5273/P	5273/P	5482	5484/M	5504/M	5506/M	5575/P
mmv$pft_p	4862	5253	5272/P	5272/P	5307	5308	5490	5503/M	5510/M
		5527	5530/M	5531/M	5532/M	5533/M	5535	5539	5543/M
		5544/M	5546/M	5547/M	5695/M				
mmv$pfti_array_p	4763	5266	5275/P	5275/P					
mmv$pt_length	4753	5113/M	5128	5263/P	5265	5326	5410	5472	5565
mmv$pt_p	4869	5117/M	5129/M	5322	5411	5415	5566/M		
mmv$reassignable_page_frames	4781	5562/M							
mmv$tables_initialized	4744	5577/M							
mmv$time_to_call_mem_mgr	4758	5569/M							
mmv$total_page_frames	4748	5309/M							
mtr_stack	3530	5325							
mtt$monitor_interlock	4386	4233							
mtv$mx_segments	4889	5234	5235/M	5237	5238/M	5241	5242	5553	
nac$heap_algorithm	4937	5776/P	5777/P						
nac$network_heap_size	4938	5774	5775/P	5776/P	5777/P				
nat$received_message_descriptor	2859	2852	2861						
nat$received_message_list	2851	2703							
nav$network_paged_heap	4946	5283/S	5777/P						
nav$network_wired_heap	4931	5762/S	5773	5773	5776/P				
new	2226	5305/M							
next_asidq_index	5205	5338/M	5346/S	5399/M	5399	5425	5425	5429	5452/M
		5452							
nlc$cc_connect_confirm	2891	2882							
nlc$cc_connect_request	2890	2880							
nlc$cc_expedited_data	2896	2882							
nlc$cc_max_pdu_kind	2898	2901							
nlc$channel_connection_pdu	2914	2866							
nlc$channelnet_pdu	2914	2868							
nlt$cc_pdu_kind	2901	2879							
nlt$cc_seq#_or_connect_time	2878	2867							
nlt$cc_sequence_number	2904	2883							
nlt$device_identifier	2911	2862							
nlt$pdu_type	2914	2865							
normal	1139	5031/M	5074	5080/M					
now	4785	5562/M							
offset	864	5467/M	5499	5500					
offset	3503	3493/M	5575/M						
open_count	4239	5625/M	5631/M	5682/M					
open_validating_ring_number	2484	5352	5599/M	5740/M	5767/M	5786/M			
osc$aging_interval_maximum	1974	1977							
osc$call_instruction	3087	3095							
osc$data_read	3086	3095							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$free_running_clock_maximum	438	435							
osc$invalid_ring	881	921							
osc$max_fault_contents	3240	3234							
osc$max_name_size	1940	1944	1947						
osc$max_number_of_processors	4897	4892							
osc$max_page_frames	831	151	152	750	752	988	1015	1021	1062
		4770	4771	4772	4785	4786	4787	4788	
osc$max_page_size	2073	2069							
osc$max_page_table_entries	832	835							
osc$max_ring	880	921	922						
osc$max_segment_length	904	927	2495	2526					
osc$max_status_condition_code	1154	1150	1166						
osc$max_string_size	1170	1173	1176	1181					
osc$max_tasks	1051	1048							
osc$maximum_offset	903	904	924	924	925				
osc$maximum_processor_id	3112	3108							
osc$maximum_processors	4901	4897							
osc$maximum_segment	902	923							
osc$min_page_size	2072	2069							
osc$min_ring	879	922							
osc$non_executable	2204	5227							
osc$non_writable	2210	4637	5384						
osc$pr_base_constant	2644	4687	5033	5069					
osc$pr_page_size_mask	2647	5118							
osc$pr_page_table_length	2646	5112							
osc$pr_segment_table_length	2642	5071							
osc$read_uncontrolled	2208	5228							
osc$segment_for_hyperchannel	2623	5601/P	5603	5604/S	5605/S				
osc$segnum_job_fixed_heap	1779	4686	5033	5069	5312/S	5764/S	5767/S		
osc$segnum_job_pageable_heap	1780	5320/S							
osc$segnum_mainframe_paged	1778	5282/S							
osc$segnum_mainframe_wired	1776	5304/S							
osc$segnum_page_table	1775	5751/S	5756/S						
osc$task_time_slice_maximum	427	430							
osc$vl_cache_bypass	2462	5227	5282	5283	5284	5595			
osc$vl_invalid_entry	2462	5072	5320	5342	5743	5784			
osc$vl_regular_segment	2462	5290							
osc$write_uncontrolled	2211	5228							
osp$fatal_system_error	4664	5528							
osp$reset_heap	3462	5775	5776	5777					
osp$system_error	4667	5035	5503	5686					
ost$aging_interval	1977	1896	1897						
ost$asid	867	55	66	67	95	863	1067	2137	2225
		2226	2468	4980	4984	5000	5174	5182	5183
		5184	5185	5204					
ost$binary_unique_name	4400	4235							
ost$byte_count	857	4506	4692						
ost$cp_time	565	533	578	2720					
ost$cp_time_value	563	137	566	567	1891	1892	2733		
ost$cs_lock	1517	1579	1646	1647	2701				
ost$date_time	4201	3679							
ost$debug_code	3086	3074							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$debug_list	3082	2993							
ost$debug_list_entry	3073	3082							
ost$debug_mask	3092	2992							
ost$exchange_package	2942	2688							
ost$execute_privilege	2204	2199	2463						
ost$execution_control_block	2687	2713	4522	4534	4548	4560	4674	4723	5029
		5061	5719						
ost$external_code_base_pointer	1748	1671	1704						
ost$family_name	1987	1982							
ost$flags	2999	2949							
ost$frame_descriptor	3050	3065							
ost$free_running_clock	435	106	107	108	109	143	153	154	155
		367	379	1066	1890	1899	2719	4251	
ost$global_task_id	1042	100	129	1487	1614	1651	1886	2698	2699
		3124	3327	4253	4341				
ost$heap	1796	1768	1803	1809	3462	3472	4921	4931	4946
ost$key_lock	910	2122	2469						
ost$key_lock_value	916	913	3016	3018					
ost$keypoint_class	3024	2962	3026						
ost$keypoint_mask	3026	2965							
ost$minimum_save_area	3060	2954	3035	3221					
ost$monitor_condition	2918	2925							
ost$monitor_conditions	2925	2955	2959	3040	3296	3310			
ost$monitor_fault	3217	3166							
ost$monitor_fault_contents	3234	3230							
ost$name	1947	1936	1985	1987	2747	2811	3259	3687	
ost$p_register	3014	2943	3061	3288	3294				
ost$page_id	837	847							
ost$page_size	2069	2050	4915						
ost$page_table	851	4869	5216						
ost$page_table_entry	842	52	851	5217					
ost$page_table_index	835	851	1092	5215					
ost$paging_statistics	601	579	2728						
ost$processor_id	3108	2691	3102						
ost$processor_id_set	3102	2690							
ost$processor_model_number	1999	1993	4402						
ost$processor_serial_number	2077	1994	4401						
ost$pva	932	2987	3005	3019	3218	3311			
ost$read_privilege	2207	2200	2464						
ost$register_number	3010	2984	3046	3054	3055	3056			
ost$ring	921	933	2113	2114	2466	2467	2484	3004	5721
ost$ring1_termination_reason	3120	2724							
ost$segment	923	934	2116	2288	2581	2982	3076	4216	4252
		4467	4549	4561	4609	4621	4632	4958	4958
		5021	5063	5618	5720				
ost$segment_access_control	2197	2135							
ost$segment_descriptor	2461	2435							
ost$segment_length	927	2118	2120	2142	2144	2147	4469		
ost$segment_offset	924	864	935	2499	3077	3079			
ost$stack_frame_save_area	3034	3068	3254						
ost$status	1138	1492	2129	3289	4470	4508	4665	4668	5022
		5057							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES						
ost$status_condition	1162	1510	2306					
ost$status_condition_code	1166	1141	1162					
ost$string	1179	1142						
ost$string_size	1173	1180						
ost$system_flag	3395	3391						
ost$system_virtual_address	862	1097	5225					
ost$task_index	1048	1043	1125	1126				
ost$task_time_slice	430	416						
ost$top_of_stack_pointer	3002	2994						
ost$trap_enable	3029	2951	3285					
ost$user_condition	2928	2935						
ost$user_conditions	2935	2953	2957	3038	3067	3257	3297	
ost$user_identification	1980	1885						
ost$user_name	1985	1981						
ost$valid_relative_pointer	930	2717	2718	4258	4261			
ost$valid_ring	922	2994						
ost$virtual_machine_identifier	1761	1741	1750	2945	2947	3062		
ost$wait	4518	4507						
ost$write_privilege	2210	2201	2465					
ost$x_register	3011	2984	3046					
osv$180_memory_limits	4906	5120/M	5120	5235	5235	5243	5250	5253	5258
		5266	5520					
osv$cpus_logically_on	4892	5281						
osv$enable_hyperchannel	2626	5590						
osv$mainframe_wired_cb_heap	1803	5760/S						
osv$mainframe_wired_heap	1768	5242	5251/P	5259/P	5263/P	5267	5290/S	5300/P	5758/S
osv$page_size	4915	4490	4490	5038	5038	5118/M	5120	5120	5121
		5235	5243	5250	5253	5257	5257	5257	5258
		5266	5464	5499	5500	5520	5582		
page_streaming_transfer_size	5026	5038/P	5044						
pageid	847	5327	5330/M	5411	5467				
pagenum	838	5467							
pages_in_memory	1062	5478/M	5478						
pages_to_dump_r_pointer	5208	5575/P	5576/P						
pft_link	1061	5503	5503/M	5503/M	5503/S	5503	5503/M	5691	5692/M
		5693/M	5695/S	5696	5697/M				
pft_p	5209	5490/M	5491/M	5492/M	5493/M	5494/M	5495/M	5496/M	5497/M
		5498/M	5502	5503/P					
pft_seq_p	5210	5251/P	5252	5253	5259/P	5260	5261		
pft_size	5211	5250/M	5251/P	5258/M	5259/P	5261			
pfte_p	5170	5503	5503	5503/M					
pfte_p	5680	5685	5685	5696/M					
pfti	5170	5503	5503	5503	5503				
pfti	5212	5464/M	5481	5481	5481	5482	5482	5484/S	5490/S
		5503/P	5504/S	5506/S	5510	5512	5515	5525	5526
		5527/S	5530/S	5531/S	5532/S	5533/S	5535/S	5538	5539/S
		5541	5543/S	5544/S	5546/S	5547/S			
pfti	5679	5692	5693	5695	5697				
pftimax	5213	5308/M	5309	5481	5525				
pftimin	5214	5307/M	5309	5481	5481	5482	5525		
pmc$kill_task_flag	3395	3411							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES						
pmc$max_signal_contents	3378	3372						
pmc$max_task_id	3133	3130						
pmp$find_executing_task_xcb	4673	4688	5033	5069				
pmp$zero_out_table	4690	5271	5272	5273	5274	5275		
pmt$binary_mainframe_id	1992	1889						
pmt$condition_identifier	3207	3201						
pmt$cpu_model_number	2059	2048	2055					
pmt$cpu_serial_number	2062	2049	2054					
pmt$initialization_value	2214	2126	4250					
pmt$sense_switches	2082	1900						
pmt$signal	3334	3328						
pmt$signal_contents	3372	3336						
pmt$signal_id	3339	3335						
pmt$task_id	3130	2715	3125					
power	4482	4491/M	4493/M	4493	4496	4499		
power	5018	5038/M	5038/M	5038	5038	5038		
pqle	1018	5395	5450	5521				
ps_transfer_size	4477	4490						
ps_transfer_size	5018	5038						
ps_transfer_size_power	4478	4496	4497/M	4497	4499/M			
ps_transfer_size_power	5018	5038	5038/M	5038	5038/M			
pt_p	5216	5322/M	5327	5329/M	5330/M			
pte_p	5217	5415/M	5464	5467				
pti	1092	5491/M						
pti	5106	5128	5129/S					
pti	5215	5326	5327/S	5329/S	5330/S	5410	5411/S	5415/S	5491
		5565	5566/S	5566/S					
pva	3005	5074	5783						
qcb_p	5176	5395/M	5397/M	5450/M	5458				
qcb_p	5218	5458/M	5498	5508	5512/M	5513/M	5513	5514	5515/M
		5521/M	5522	5540	5541/M	5542/M	5542	5562	
queue_id	1068	5377/M	5438/M						
queue_id	1089	5493/M	5532/M	5543/M					
queue_id	5175	5359/M	5377	5451/M	5457				
queue_id	5219	5349/M	5351/M	5353/M	5355/M	5359	5367	5374	5374
		5394	5395/S	5397/S	5457/M	5493			
queue_status	4249	5629/M	5660/M						
r1	2466	4641	5380	5384	5592/M				
r2	2467	5593/M							
r_pointer	3481	3493/M	3494/M	3495/M	3496/M				
r_pointer	5170	5575/M	5575/M	5575/M	5575/M				
rb	5062	5066/M	5067/M	5068/M	5075/M	5076/P			
rb	5220	5287/M	5288/M	5289/M	5291/P				
rb	5799	5809/M	5810/M	5812/P					
rb	5821	5831/M	5832/M	5834/P					
reqcode	4581	5066/M	5287/M	5809/M	5831/M				
request	4586	5067/M	5288/M	5810/M	5832/M				
residence	20	4427	4428	5046	5050	5050	5623		
residence	5221	5368/M	5370/M	5372/P					
residence	5616	5622/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
rlower	3505	3495/M	5575/M						
rma	848	5464							
rupper	3504	3494/M	5575/M						
scan_page_table	5409	5409	5413	5487	5518				
scan_pft	5524	5524	5536	5548					
scan_sdt	5340	5340	5343	5401					
sdt_entry	5019	5042							
sdt_entry	5226	5591/M	5592/M	5593/M	5594/M	5595/M	5604		
sdt_entry	5647	5664/P	5665/M						
sdt_offset	2717	4527	4540	4552	5039	5070	5279	5728/M	5731
sdt_p	4523	4527/M							
sdt_p	4535	4540/M							
sdt_p	4634	4637	4641						
sdt_p	5056	5070/M							
sdt_p	5170	5279/M							
sdt_p	5170	5384	5384						
sdt_p	5223	5279/P	5282/M	5283/M	5284/M	5290/M	5304	5312	5320/M
		5342	5358	5361/M	5380	5383/P	5604/M		
sdt_p	5704	5731/M							
sdt_p	5716	5731/P	5743	5745/P	5784				
sdte	5227	5591							
sdtx_entry	5020	5043							
sdtx_entry	5229	5596/M	5597/M	5597	5599/M	5600/M	5601/P	5605	
sdtx_entry	5648	5654/M	5655/M	5656/M					
sdtx_entry	5717	5738/M	5739/M	5740/M	5744				
sdtx_offset	2718	4541	4564	5040	5279	5729/M	5731		
sdtx_p	4536	4541/M							
sdtx_p	5170	5279/M							
sdtx_p	5224	5279/P	5348	5350	5352	5373/M	5375/M	5379	5605/M
sdtx_p	5704	5731/M							
sdtx_p	5718	5731/P	5744/M	5745/P	5751/M	5756/M	5758/M	5760/M	5762/M
		5764/M	5767/M	5785/M	5786/M	5787/M			
sdtx_table	2516	5348	5350	5352	5373/M	5375/M	5379	5605/M	5744/M
		5745/P	5751/M	5756/M	5758/M	5760/M	5762/M	5764/M	5767/M
		5785/M	5786/M	5787/M					
seg	934	5074	5783						
seg	4424	4426/M	4429/M	4431/M	4434	4435			
seg	5018	5050/M	5050/M	5050/M	5050	5050			
segment_link	1086	5503	5503	5503/M	5503/M	5546/M	5547/M	5685	5685
		5695/M	5696/M						
segment_number	4632	4638							
segment_number	5170	5384							
segment_table_length	2982	5341	5742						
segnum	4549	4552							
segnum	4561	4564							
segnum	4609	5075/M							
segnum	5018	5039							
segnum	5018	5040							
segnum	5021	5039/P	5040/P	5047/P					
segnum	5063	5071	5072/S	5073/P	5074	5075			
segnum	5618	5634	5637	5638					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES							
segnum	5720	5742	5743/S	5744/S	5745/S	5745/S	5783/M	5784/S	5785/S
		5786/S	5787/S						
seq_p	3480	3492/P	3496						
seq_p	5170	5575/P	5575						
seq_p_rma	3490	3492/P	3493	3494	3495				
seq_p_rma	5170	5575/P	5575	5575	5575				
sfid	1070	5372/P	5373	5435/P					
sfid	2486	5046	5047/P	5050/P	5373/M	5601/P			
sfid	4418	4427	4428	4434	4435				
sfid	4802	5857/M							
sfid	5018	5050	5050	5050	5050				
sfid	5619	5622/P	5623	5638/M					
sfid	5652	5658/P							
sfid	5841	5857							
sft$counter	611	580	581	1901	1903	1904	1906		
sft$file_space_limit_kind	2619	2493							
software_attribute_set	2489	5348	5350	5379	5597/M	5597	5655/M	5756/M	5758/M
		5761/M	5762/M	5764/M	5785/M				
ssr_size	5222	5257/M	5258						
st	2448	5072	5282/M	5283/M	5284/M	5290/M	5304	5312	5320/M
		5342	5358	5361/M	5380	5383/P	5604/M	5743	5745/P
		5784							
st_p	5064	5070/P	5072						
stack_for_ring	4254	4639	5380/M	5384					
status	5022	5031/M							
status	5057	5073/P	5074	5080/M					
ste	2435	4637	4641	5072	5282/M	5283/M	5284/M	5290/M	5304
		5312	5320/M	5342	5358	5380	5384	5384	5592/M
		5593/M	5594/M	5595/M	5664/P	5743	5784		
ste_p	5027	5039/M	5042/M						
stream	2484	5044/M							
stxe_p	5028	5040/M	5043/M	5044/M	5046	5047/P	5050/P		
sva	1097	5492/M							
sva	5225	5466/M	5467/M	5468	5492	5499	5500		
syc$rc_ring1_segment_request	2365	5066	5287	5809	5831				
syc$ucr_condition	3245	3256							
syc$user_defined_condition	3246	3258							
syt$monitor_flag	3152	3137							
syt$monitor_flags	3137	2689							
syt$monitor_request_code	2316	2284	4581						
syt$monitor_status	2304	2285	4582						
tmc$broken_task_fault_id	3173	3223							
tmc$btc_invalid_a0	3271	3292							
tmc$btc_invalid_p	3271	3292							
tmc$btc_mcr_traps_disabled	3272	3293							
tmc$btc_mf_traps_disabled	3271	3291							
tmc$btc_mntr_fault_buffer_full	3270	3291							
tmc$btc_system_error	3273	3287							
tmc$btc_ucr_traps_disabled	3272	3293							
tmc$dummy_fault	3174	3229							
tmc$flag_available_31	3408	3412							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES						
tmc$maximum_monitor_faults	3178	3169						
tmc$maximum_signals	3388	3385						
tmc$maximum_system_task_id	3421	3424						
tmc$mcr_fault	3173	3225						
tmc$signal_available_63	3370	3381						
tmc$stid_null_task	3427	3424						
tmt$broken_task_condition	3270	3286						
tmt$broken_task_monitor_fault	3284	3224						
tmt$mcr_faults	3309	3226						
tmt$monitor_fault_buffer	3163	2726						
tmt$monitor_fault_buffers	3169	3164	3165	3166				
tmt$monitor_fault_identifiers	3172	3222	3298					
tmt$signal	3326	3321						
tmt$signal_buffer	3318	2727						
tmt$signal_buffers	3385	3319	3320	3321				
tmt$system_flags	3391	2702						
tmt$system_task_id	3424	2693						
tmt$task_queue_link	1124	1093	4368					
tos_array_index	5721	5782	5783/S					
tos_registers	2994	5074	5783					
transfer_size	2501	5044/M						
ts	4483	4490/M	4492	4492	4494/M	4494		
ts	5018	5038/M	5038	5038/M	5038	5038		
upper	4909	5235	5250	5253	5258	5266		
v	843	5329/M						
value_type	2248	5138						
v1	2462	5072	5282/M	5283/M	5284/M	5290/M	5320/M	5342 5595/M
		5743	5784					
wait_for_io_complete	4583	5068/M	5289/M					
wp	2465	4637	5384					
xasid	4980	4989	4989					
xasid	5170	5360	5360	5431	5431			
xasid	5646	5664	5664					
xasti	4981	4995/M						
xasti	4999	5005						
xasti	5170	5360/M	5431/M					
xasti	5170	5555						
xasti	5646	5664/M						
xcb	4673	4686/M						
xcb	5018	5033/M						
xcb	5056	5069/M						
xcb_p	4522	4527	4527					
xcb_p	4534	4540	4540	4541	4541			
xcb_p	4548	4551	4552					
xcb_p	4560	4563	4564					
xcb_p	5018	5039	5039					
xcb_p	5018	5040	5040					
xcb_p	5029	5033/P	5034	5039/P	5040/P			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

MMP$DEFINE_IMAGE_FILE

IDENTIFIER	DEFINED ON LINE	REFERENCES					
xcb_p	5056	5070	5070				
xcb_p	5061	5069/P	5070/P	5074			
xcb_p	5170	5279	5279	5279	5279		
xcb_p	5704	5731	5731	5731	5731		
xcb_p	5719	5727/M	5728/M	5729/M	5731/P	5742	5783
xp	2688	5074	5341	5742	5783		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

```
        2 MODULE mmm$monitor_request_processor;
        3
        4 (*
        5 (*  PURPOSE: Memory_Manager
        6 (*     This module contains the monitor routines that are used to
        7 (*     manage physical memory and the page table.
        8 (*
O    9188 ( This decks defines compile-time constants to control conditional compilation
O    9189 ( of debug code in memory manager modules in monitor mode. All constants should
O    9190 ( be set to FALSE for the transmitted version of this deck.
O    9191
O    9192   CONST
O    9193     mmc$debug = TRUE;
O    9194
O    9195
O    9196   ?VAR
O    9197      mmc$debug_check_queues: boolean := TRUE, (*Check POL linkage*)
O    9198      mmc$debug_relink_swapping_job: boolean := TRUE , (*Stop if relink page of swapping job*)
O    9199      mmc$debug_pt: boolean := TRUE, (*check PT - AST linkage*)
O    9200      mmc$debug_free_asid: boolean := TRUE, (*Verify no attempt is made to free an AST
O    9201                                              { entry with pages not in AVAIL queue.
O    9202      mmc$debug_aste_p_from_pfti: boolean := TRUE, (*Verify aste_p in pfti is correct*)
O    9203      mmc$debug_rma_list: boolean := TRUE, (*Verify RMA list on lock/unlock rma list*)
O    9204      mmc$debug_esc_alloc: boolean := TRUE, (*check for escaped allocation*)
O    9205      mmc$debug_ast_pft: boolean := TRUE?; (*check pft.aste_p on reference to AST*)
O    9206


O    9208
O    9209 (*  PURPOSE: This deck (mmc$manage_memory_utility) primarily contains default values for the mmv$ variables
O    9210 (*    and the Global Page Queue List all of which are managed by the Manage Memory Utility which
O    9211 (*    is in the module mmm$manage_memory.
O    9212
O    9213     CONST
O    9214       mmc$mmu_age_interval_ceiling    =       10,   { pages }
O    9215       mmc$mmu_age_interval_floor      =        3,   { pages }
O    9216       mmc$mmu_aggressive_aging_one    =       10,   { pages }
O    9217       mmc$mmu_aggressive_aging_two    =       18,   { pages }
O    9218       mmc$mmu_aging_algorithm         =        4,   { flag to select algorithm}
O    9219       mmc$mmu_jws_age_interval        = 8000000,   { microseconds }
O    9220       mmc$mmu_min_avail_pages         =      400,   { pages }
O    9221       mmc$mmu_ps_prestream            =        4,   { page faults}
O    9222       mmc$mmu_ps_transfer_size        =        0,   { bytes}{when non-zero, overrides transfer size from DM
O    9223       mmc$mmu_ps_threshold            =    65536,   { bytes}
O    9224       mmc$mmu_ps_reads                =        3,   { transfer units
O    9225       mmc$mmu_ps_random_limit         =        3,   { random page faults
O    9226       mmc$mmu_periodic_call_interval = 1000000,   { microseconds }
O    9227       mmc$mmu_shared_age_interval     = 8000000,   { microseconds }
O    9228       mmc$mmu_swapping_aic            =        1,   { limit on number of times an unused page is swapped out}
O    9229       mmc$mmu_tick_time               =   100000,   { microseconds,  Real default is determined in deadstart.
O    9230       mmc$mmu_queue_age_task_service  =        3,   { number of shared_age_intervals}
O    9231       mmc$mmu_queue_age_pf_execute    =        1,   { number of shared_age_intervals}
O    9232       mmc$mmu_queue_age_pf_non_exec   =        1,   { number of shared_age_intervals}
O    9233       mmc$mmu_queue_age_device_file   =        1,   { number of shared_age_intervals}
O    9234       mmc$mmu_queue_age_file_server   =        1,   { number of shared_age_intervals}
O    9235       mmc$mmu_queue_age_other         =        1,   { number of shared_age_intervals}
O    9236       mmc$mmu_queue_age_site_queues   =        1,   { number of shared_age_intervals}
```



```
O    9237       mmc$mmu_queue_maximum           =       osc$max_page_frames,  {default maximum pages for all queues}
O    9238       mmc$mmu_queue_minimum           =       0;      {default minimum pages for all queues}
O    9239
O    9240
O    9241


O    9243 {External procedures used by this module.
O    9244 { Try to keep them in alphabetical order.
O    9245
O    9263   PROCEDURE [INLINE] display_monitor
O    9264     (    display_line: string ( * <= 255));
O    9265
O    9266     dpp$display_error (display_line);
O    9267   PROCEND display_monitor;




O    9269
O    9270   PROCEDURE display_integer_monitor
4    9271     (    desc: string ( * <= 80);
4    9272          int: integer);
4    9273
4    9274     VAR
4    9275       hex_digits: [READ] array [0 .. 15] of char := ['0', '1', '2', '3', '4',
4    9276            '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'];
4    9277
4    9278     TYPE
4    9279       convert = record
4    9280         case (integ, strng) of
4    9281         = integ =
4    9282           int: integer,
4    9283         = strng =
4    9284           strn: string (8),
4    9285         casend,
4    9286       recend;
4    9287
4    9288     VAR
4    9289       conv: convert,
4    9290       data: string (8),
4    9291       data_index: integer,
4    9292       line: string (80),
4    9293       line_index: integer;
4    9294
4    9295     line := desc;
22   9296     line_index := #SIZE (desc) + 2;
22   9297     conv.int := int;
22   9298     data := conv.strn;
22   9299     FOR data_index := 1 TO 8 DO
3E   9300       line (line_index) := hex_digits [$INTEGER (data (data_index)) DIV 16];
3E   9301       line (line_index + 1) := hex_digits [$INTEGER (data (data_index)) MOD 16];
3E   9302       line_index := line_index + 2;
3E   9303     FOREND;
6E   9304     display_monitor (line (1, (line_index - 1)));
B4   9305   PROCEND display_integer_monitor;
```

```
  9306
  9307
  9308   VAR
  9309     dfv$file_server_debug_enabled: [XREF] boolean;
  9310
  9311
  9312   PROCEDURE [XREF] dmp$deallocate_file_space (
  9313     p_fde: gft$locked_file_desc_entry_p;
  9314     initial_release_byte_address: amt$file_byte_address;
  9315     initial_bytes_to_release: integer);
  9316
o 9319
o 9320   PROCEDURE [INLINE] gfp$mtr_get_fde_p (sfid: gft$system_file_identifier;
o 9321         ijle_p: ^jmt$initiated_job_list_entry;
o 9322     VAR fde_p: gft$file_desc_entry_p);
o 9323
o 9324
o 9367
o 9368   PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
o 9369         ijle_p: ^jmt$initiated_job_list_entry;
o 9370     VAR fde_p: gft$locked_file_desc_entry_p);
o 9371
o 9372
o 9433
o 9434   PROCEDURE [XREF] i#real_memory_address (p: ^cell;
o 9435     VAR rma: integer);
o 9436
o 9437   PROCEDURE [INLINE] jmp$check_scheduler_memory_wait;
o 9438
  9471   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
  9472     VAR ijle_p: ^jmt$initiated_job_list_entry);
  9473
o 9481
o 9482   FUNCTION [INLINE] jmp$ijl_block_valid
o 9483     (    ijl_ordinal: jmt$ijl_ordinal): boolean;
o 9484
o 9485     jmp$ijl_block_valid := NOT (jmv$ijl_p.block_p^ [ijl_ordinal.block_number].index_p = NIL);
o 9486
o 9487   FUNCEND jmp$ijl_block_valid;
o 9488
o 9489 { This procedure is called when we need access to the job fixed of a job that
o 9490 { may or may not have an ajl assigned.
o 9491
o 9492   PROCEDURE [INLINE] jmp$lock_ajl
o 9493     (    ijle_p: ^jmt$initiated_job_list_entry;
o 9494          ijlo: jmt$ijl_ordinal;
o 9495      VAR ajlo: jmt$ajl_ordinal);
o 9496
o 9497     VAR
o 9498       ajl_ordinal: jmt$ajl_ordinal,
o 9499       status: syt$monitor_status;
o 9500
o 9501     tmp$set_lock (tmv$ptl_lock);
o 9502     ajl_ordinal := ijle_p^.ajl_ordinal;
o 9503     IF ajl_ordinal = jmc$null_ajl_ordinal THEN
o 9504       jmp$assign_ajl_with_lock (ijle_p^.job_fixed_asid, ijlo, jmc$lock_ajl, TRUE (MUST_ASSIGN),
o 9505             ajlo, status);
```



```
o 9506     ELSE
o 9507       jmv$ajl_p^ [ajl_ordinal].in_use := jmv$ajl_p^ [ajl_ordinal].in_use + jmc$lock_ajl;
o 9508       ajlo := ajl_ordinal;
o 9509     IFEND;
o 9510     tmp$clear_lock (tmv$ptl_lock);
o 9511
o 9512   PROCEND jmp$lock_ajl;
o 9513
  9627
  9628
  9629   PROCEDURE [INLINE] jmp$unlock_ajl
  9630     (    ijle_p: ^jmt$initiated_job_list_entry);
  9631
  9632     VAR
  9633       ajlo: jmt$ajl_ordinal;
  9634
  9635     tmp$set_lock (tmv$ptl_lock);
  9636     ajlo := ijle_p^.ajl_ordinal;
  9637     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
  9638       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
  9639     ELSE
  9640       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
  9641     IFEND;
  9642     tmp$clear_lock (tmv$ptl_lock);
  9643
  9644   PROCEND jmp$unlock_ajl;
  9645
o 9657
o 9658   PROCEDURE [XREF] jmp$set_scheduler_event (event: jmt$job_scheduler_events);
o 9659
  9662
  9663    PROCEDURE [XREF] jsp$adv_expired_swapped_jobs (swap_queue_id: jst$swapped_but_still_in_memory);
  9664
o 9667
o 9668
o 9669   PROCEDURE [XREF] jsp$initiate_swapout_io (pages_needed: mmt$page_frame_index);
o 9670
  9673   PROCEDURE [XREF] jsp$io_complete (ijle_p: ^jmt$initiated_job_list_entry);
  9674
o 9677
o 9678   PROCEDURE [XREF] jsp$recalculate_swapped_pages
o 9679     (    ijle_p: ^jmt$initiated_job_list_entry;
o 9680          pages_removed: mmt$page_frame_index);
o 9681
  9684
  9685   PROCEDURE [XREF] mmp$age_job_working_set (ijle_p: ^jmt$initiated_job_list_entry;
  9686         jcb_p: ^jmt$job_control_block);
  9687
o 9690
o 9691   PROCEDURE [XREF] mmp$asid (asti: mmt$ast_index;
o 9692     VAR asid: ost$asid);
o 9693
  9696
  9697   PROCEDURE [XREF] mmp$aste_pointer (asid: ost$asid;
  9698     VAR aste_p: ^mmt$active_segment_table_entry);
  9699
o 9702   PROCEDURE [XREF] mmp$asti (asid: ost$asid;
```

```
o  9703     VAR asti: mmt$ast_index);
o  9704
   9707
   9708   PROCEDURE [XREF] mmp$change_asid (aste_p: ^mmt$active_segment_table_entry;
   9709         old_asid: ost$asid;
   9710         new_asid: ost$asid;
   9711         new_asti: mmt$ast_index);
   9712
o  9715
o  9716   PROCEDURE [INLINE] mmp$check_queues;
o  9717
o  9718     ?IF mmc$debug_check_queues THEN
o  9719       IF mmv$check_queues > 0 THEN
o  9720         mmp$xcheck_queues;
o  9721       IFEND;
o  9722     ?IFEND;
o  9723
o  9724   PROCEND;
o  9725
o  9726
o  9727   PROCEDURE [XREF] mmp$convert_pva (p: ^cell;
o  9728         cst_p: ^ost$cpu_state_table;
o  9729     VAR sva: ost$system_virtual_address;
o  9730     VAR fde_p: gft$locked_file_desc_entry_p;
o  9731     VAR aste_p: ^mmt$active_segment_table_entry;
o  9732     VAR ste_p: ^mmt$segment_descriptor;
o  9733     VAR stxe_p: ^mmt$segment_descriptor_extended);
o  9734
   9737
   9738   PROCEDURE [INLINE] mmp$delete_last_pfti_from_array;
o  9744
o  9745   PROCEDURE [XREF] mmp$delete_pt_entry
o  9746     (     pfti: mmt$page_frame_index;
o  9747           unlink_page_from_segment: boolean);
o  9748
   9751
   9752
   9753   FUNCTION [XREF] mmp$determine_shared_queue_id
   9754     (     fde_p: gft$locked_file_desc_entry_p;
   9755           ste_p: ^mmt$segment_descriptor): mmt$page_frame_queue_id;
   9756
o  9759
o  9760   PROCEDURE [INLINE] mmp$fetch_pfti_array_size
o  9761     (VAR pfti_size: integer);
o  9762
   9768
   9769   PROCEDURE [INLINE] mmp$find_next_pfti
   9770     (VAR xpfti: mmt$page_frame_index);
   9771
o  9789
o  9790   PROCEDURE [XREF] mmp$free_asid (asid: ost$asid;
o  9791         aste_p: ^mmt$active_segment_table_entry);
o  9792
   9795
   9796   PROCEDURE [XREF] mmp$free_memory_in_job_queues (VAR job_page_queue_list: mmt$job_page_queue_list;
   9797         increment_now: boolean;
   9798         decrement_soon: boolean;
```



```
   9799         job_termination: boolean);
   9800
o  9803
o  9804
o  9805    PROCEDURE [INLINE] mmp$get_inhibit_io_status
o  9806      (     ijl_ordinal: jmt$ijl_ordinal;
o  9807            lock: boolean;
o  9808       VAR inhibit_io: boolean;
o  9809       VAR ijle_p: ^jmt$initiated_job_list_entry);
o  9810
o  9811     VAR
o  9812       ajlo: jmt$ajl_ordinal;
o  9813
o  9814      jmp$get_ijle_p (ijl_ordinal, ijle_p);
o  9815      inhibit_io := (ijle_p^.swap_status > jmc$inhibit_memory_manager_io);
o  9816      IF NOT inhibit_io THEN
o  9817        IF lock THEN
o  9818          tmp$set_lock (tmv$ptl_lock);
o  9819          jmp$lock_ajl_with_lock (ijle_p, ijl_ordinal, ajlo);
o  9820          tmp$clear_lock (tmv$ptl_lock);
o  9821        IFEND;
o  9822      IFEND;
o  9823
o  9824    PROCEND mmp$get_inhibit_io_status;
o  9825
   9855
   9856   PROCEDURE [INLINE] mmp$get_max_sdt_pointer
   9857     (     xcb_p: ^ost$execution_control_block;
   9858       VAR sdt_p: mmt$max_sdt_p);
   9859
o  9867
o  9868   PROCEDURE [INLINE] mmp$get_max_sdt_sdtx_pointer
o  9869     (     xcb_p: ^ost$execution_control_block;
o  9870       VAR sdt_p: mmt$max_sdt_p;
o  9871       VAR sdtx_p: mmt$max_sdtx_p);
o  9872
   9881  PROCEDURE [XREF] mmp$initialize_find_next_pfti (xsva: ost$system_virtual_address;
   9882        length: ost$segment_length;
   9883        end_point_option: (include_partial_pages, exclude_partial_pages);
   9884        page_selection_criteria: mmt$page_selection_criteria;
   9885        aste_p: ^mmt$active_segment_table_entry;
   9886    VAR xpfti: mmt$page_frame_index);
   9887
o  9890
o  9891  PROCEDURE [INLINE] mmp$maintain_memory_thresholds;
o  9892
o  9893  {
o  9894  {     The purpose of this procedure is to check the memory thresholds
o  9895  {  concerning the status of jobs being swapped.  If memory is needed
o  9896  {  swapped jobs that have not had swapout IO initiated are advanced.
o  9897  {
o  9898
o  9899     VAR
o  9900       available_memory: 0 .. osc$max_page_frames - 1;
o  9901
o  9902
o  9903     available_memory := mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.
```

```
o  9904            soon;
o  9905
o  9906      IF (available_memory <= jmv$long_wait_swap_threshold) AND
o  9907            (mmv$reassignable_page_frames.swapout_io_not_initiated <> 0) THEN
o  9908        jsp$initiate_swapout_io (jmv$long_wait_swap_threshold -
o  9909              available_memory + 1);
o  9910 {! Should do something here if not enough memory freed?
o  9911      ELSE
o  9912 {! What should be done here.
o  9913      IFEND;
o  9914
o  9915    PROCEND mmp$maintain_memory_thresholds;
o  9916
   9922
   9923    PROCEDURE [XREF] mmp$make_pt_entry (sva: ost$system_virtual_address;
   9924          pfti: mmt$page_frame_index;
   9925          aste_p: ^mmt$active_segment_table_entry;
   9926          pfte_p: ^mmt$page_frame_table_entry;
   9927      VAR mpt_status: mmt$make_pt_entry_status);
   9928
o  9931
o  9932     FUNCTION [INLINE] mmp$get_sdtx_entry_p
o  9933       (     xcb_p: ^ost$execution_control_block;
o  9934             segnum: ost$segment): ^mmt$segment_descriptor_extended;
o  9935
o  9936      mmp$get_sdtx_entry_p := #address (1, #segment (xcb_p),
o  9937        #SIZE (mmt$segment_descriptor_extended) * segnum + xcb_p^.sdtx_offset);
o  9938
o  9939     FUNCEND;
o  9940
   9943
   9944
   9945     FUNCTION [INLINE] mmp$get_sdt_entry_p
   9946       (     xcb_p: ^ost$execution_control_block;
   9947             segnum: ost$segment): ^mmt$segment_descriptor;
   9948
   9949      mmp$get_sdt_entry_p := #address (1, #segment (xcb_p),
   9950        8 * segnum + xcb_p^.sdt_offset);
   9951
   9952     FUNCEND;
   9953
o  9956
o  9957    PROCEDURE [XREF] mmp$mtr_set_get_segment_length (VAR request_block:
o  9958      mmt$rb_set_get_segment_length;
o  9959          cst_p: ^ost$cpu_state_table);
o  9960
   9980    PROCEDURE [XREF] mmp$process_page_table_full (sva: ost$system_virtual_address;
   9981      VAR new_asid: ost$asid;
   9982      VAR new_asti: mmt$ast_index;
   9983      VAR new_aste_p: ^mmt$active_segment_table_entry;
   9984      VAR pt_full_status: mmt$pt_full_status);
   9985
o  9988    PROCEDURE [XREF] mmp$process_volume_unavailable (xcb_p: ^ost$execution_control_block;
o  9989      reset_p_register: boolean);
   9992
   9993    PROCEDURE [XREF] mmp$reclaim_ast_entries
   9994      (     asti_that_cannot_be_freed: mmt$ast_index);
```



```
   9995
o  9998
o  9999
o 10000    PROCEDURE [XREF] mmp$relink_page_frame (pfti: mmt$page_frame_index;
o 10001          queue_id: mmt$page_frame_queue_id);
o 10002
  10005
  10006    PROCEDURE [XREF] mmp$remove_page_from_jws (pfti: mmt$page_frame_index;
  10007          ijle_p: ^jmt$initiated_job_list_entry;
  10008      VAR mcount: integer;
  10009      VAR rcount: integer);
  10010
o 10013
o 10014    PROCEDURE [XREF] mmp$remove_pages_from_jws
o 10015      (     modified_queue_id: mmt$page_frame_queue_id;
o 10016            ijle_p: ^jmt$initiated_job_list_entry;
o 10017       VAR modified_page_count: integer;
o 10018       VAR total_page_count: integer);
o 10019
  10022
  10023    PROCEDURE [XREF] mmp$remove_page_from_job (pfti: mmt$page_frame_index);
  10024
o 10027    PROCEDURE [XREF] mmp$remove_stale_pages (VAR pqle: mmt$page_queue_list_entry;
o 10028          age_limit: integer;
o 10029          jcb_p: ^jmt$job_control_block;
o 10030          ijle_p: ^jmt$initiated_job_list_entry;
o 10031          queue_id: mmt$page_frame_queue_id;
o 10032          minimum_working_set: 0 .. 0ffff(16);
o 10033      VAR modified_pages_removed: integer;
o 10034      VAR total_pages_removed: integer);
o 10035
  10038
  10039    PROCEDURE [INLINE] mmp$reset_find_next_pfti
  10040      (VAR xpfti: mmt$page_frame_index);
  10041
o 10064
o 10065    PROCEDURE [INLINE] mmp$reset_store_pfti;
o 10066
  10073
  10074    PROCEDURE [INLINE] mmp$set_include_pages_in_dump
  10075      (     segment_number: ost$segment;
  10076            fde_p: gft$locked_file_desc_entry_p;
  10077            sdt_p: ^mmt$segment_descriptor;
  10078       VAR include_pages_in_dump: boolean);
  10079
  10080      IF (sdt_p^.ste.wp <> osc$non_writable) THEN
  10081        IF (segment_number < mmc$first_loader_predefined_seg) OR
  10082              (fde_p^.stack_for_ring <> 0) OR
  10083              (fde_p^.flags.global_template_file) OR
  10084              (sdt_p^.ste.r1 <= 2) THEN
  10085          include_pages_in_dump := TRUE;
  10086        ELSE
  10087          include_pages_in_dump := FALSE;
  10088        IFEND;
  10089      ELSE
  10090        include_pages_in_dump := FALSE;
  10091      IFEND;
```

```
  10092
  10093   PROCEND mmp$set_include_pages_in_dump;
  10094
o 10103
o 10104   PROCEDURE [INLINE] mmp$store_pfti (pfti: mmt$page_frame_index);
o 10105
  10112
  10113 { This procedure verifies that the asti stored in the file descriptor entry is still being used by
  10114 { the same job for the same file.  If the asti is ok, it is returned; otherwise 0 is returned.
  10115
  10116   PROCEDURE [INLINE] mmp$get_verify_asti_in_fde
  10117     (     fde_p: gft$locked_file_desc_entry_p;
  10118           sfid: gft$system_file_identifier;
  10119           ijlo: jmt$ijl_ordinal;
  10120      VAR asti: mmt$ast_index);
  10121
o 10133
o 10134
o 10135   PROCEDURE [XREF] mmp$update_eoi
o 10136     (     fde_p: gft$locked_file_desc_entry_p;
o 10137           offset: ost$segment_offset;
o 10138           reason: mmt$update_eoi_reason);
o 10139
  10142   PROCEDURE [XREF] mmp$verify_pva (pointer_to_pva: ^cell;
  10143         access: mmt$segment_access_type;
  10144     VAR status: syt$monitor_status);
  10145
o 10152   PROCEDURE [XREF] mmp$write_page_to_disk
o 10153     (    fde_p: gft$locked_file_desc_entry_p;
o 10154          pfti: mmt$page_frame_index;
o 10155          iotype: iot$io_function;
o 10156          io_id: mmt$io_identifier;
o 10157          multiple_page_req: boolean;
o 10158     VAR write_status: mmt$write_page_to_disk_status);
o 10159
  10166
  10167   PROCEDURE [XREF] mmp$xtask_pva_to_sva (pva: ^cell;
  10168     VAR sva: ost$system_virtual_address;
  10169     VAR status: syt$monitor_status);
  10170
o 10173   PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
  10184
  10185   PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
  10186         condition: osc$max_status_condition_number + 1 .. 0fffffffff(16);
  10187     VAR status: syt$monitor_status);
  10188
o 10198  PROCEDURE [XREF] osp$process_keypoint_io_error;
o 10199  PROCEDURE [XREF] osp$process_keypoint_periodic;
o 10200   PROCEDURE [XREF] tmp$cause_task_switch;
o 10201
o 10202
o 10203   PROCEDURE [XREF] tmp$check_for_swapout_candidate (ajlo: jmt$ajl_ordinal);
o 10204
  10207
  10208   PROCEDURE [XREF] tmp$check_timed_wait_not_queued
  10209     (     time_next_scan_wait_not_queued: integer);
  10210
```



```
  10211
  10212   PROCEDURE [XREF] tmp$dequeue_task (VAR queue_link: tmt$task_queue_link;
  10213     VAR taskid: ost$global_task_id);
  10214
o 10217
o 10218   PROCEDURE [XREF] tmp$find_next_queued_task
o 10219     (VAR taskid: ost$global_task_id);
o 10220
  10223
  10224   PROCEDURE [INLINE] tmp$get_taskid_from_task_queue
  10225     (     task_queue: tmt$task_queue_link;
  10226      VAR taskid: ost$global_task_id);
  10227
o 10233   PROCEDURE [XREF] tmp$get_xcb_p (task_id: ost$global_task_id;
o 10234     VAR xcb_p: ^ost$execution_control_block;
o 10235     VAR ijle_p: ^jmt$initiated_job_list_entry);
o 10236
  10239
  10240   PROCEDURE [XREF] tmp$idle_non_dispatchable_job
  10241     (     ajl_ordinal: jmt$ajl_ordinal);
  10242
o 10245
o 10246
o 10247    PROCEDURE [XREF] tmp$obtain_ijl_ordinal_from_ptl
o 10248      (     global_task_id: ost$global_task_id;
o 10249       VAR ijl_ordinal: jmt$ijl_ordinal);
o 10250
  10253
  10254   PROCEDURE [XREF] tmp$queue_task (taskid: ost$global_task_id;
  10255         task_status: tmt$task_status;
  10256     VAR queue_link: tmt$task_queue_link);
  10257
o 10260
o 10261   PROCEDURE [XREF] tmp$reissue_monitor_request;
o 10262
o 10263
o 10264   PROCEDURE [XREF] tmp$send_monitor_fault (task_id (input) : ost$global_task_id;
o 10265         monitor_fault_p (input) : ^ost$monitor_fault;
o 10266         check_traps_enabled (input) : BOOLEAN);
o 10267
  10270
  10271   PROCEDURE [XREF] tmp$set_monitor_flag (task_id: ost$global_task_id;
  10272         flag_id: syt$monitor_flag;
  10273     VAR status: syt$monitor_status);
o 10276
o 10277   PROCEDURE [XREF] tmp$set_task_ready (task_id: ost$global_task_id;
o 10278         readying_task_priority: jmt$dispatching_priority;
o 10279         ready_condition: tmt$ready_condition);
o 10280
```

```
10284 {-------------------------------------------------------------------------------------------------------------
10285
10286   TYPE
10287     ptr_type = record
10288       case b: 0 .. 3 of
10289       = 0 =
10290         st_p: ^mmt$segment_descriptor_table,
10291       = 1 =
10292         pva: ost$pva,
10293       = 2 =
10294         p: ^cell,
10295       = 3 =
10296         sdtx_p: ^mmt$segment_descriptor_table_ex,
10297       casend,
10298     recend;
10299
10300
10301   VAR
10302     dmv$null_sfid: [XREF, READ, oss$mainframe_wired_literal] dmt$system_file_id;
10303


o 10310
o 10311   VAR
o 10312     jmv$idle_dispatching_controls: [XREF] jmt$idle_dispatching_controls;
o 10313


10338
10339   VAR
10340     jmv$job_scheduler_table: [XREF] jmt$job_scheduler_table;
10341


o 10504
o 10505   VAR
o 10506     jmv$scan_idle_dispatch_interval: [XREF] integer;

o 10508 {Monitor segment table.}
o 10509
o 10510   VAR
o 10511     mtv$monitor_segment_table: [XREF] record
o 10512       st: ALIGNED [0 MOD 8] array [0 .. 4095] of mmt$segment_descriptor,
o 10513     recend;
o 10514


10518 {NOS segment table.
10519
10520   VAR
10521     mtv$nos_segment_table_p: [XREF] ^ RECORD
10522       st: ALIGNED [0 MOD 8] array [0 .. *] OF mmt$segment_descriptor,
10523     RECEND;
```



```
o 10527
o 10528 VAR
o 10529   jmv$max_ajl_ordinal_in_use: [XREF] jmt$ajl_ordinal;
o 10530
o 10531
10534 {Define pointer to Initiated Job List (IJL).
10535
10536   VAR
10537     jmv$ijl_p: [XREF]  jmt$ijl_p;
o 10565
o 10566   VAR
o 10567     jmv$max_class_working_set: [XREF] jmt$working_set_size;
o 10568


10572
10573   VAR
10574     jsv$pages_needed_for_sfd: [XREF] integer;
10575
10576
10577   VAR
10578     jsv$swapped_page_entry_size: [XREF] 0..0ff(16);
10579


10581
10582   VAR
10583     tmv$null_global_task_id: [READ] ost$global_task_id := [0, 0];
10584


10586
10587 {Define minimum number of pages that must be kept in the free + available page
10588 {queues. If the actual number drops below this value, memory manager begins
10589 {an aggressive aging policy. If the number of page frames drops below mmv$aggressive_aging_level_2
10590 {then only critical system tasks are assigned memory. User tasks are put into a memory wait queue.
10591
10592   VAR
10593     mmv$aggressive_aging_level: [XREF] integer,
10594     mmv$aggressive_aging_level_2: [XREF] integer;
10595


10597
10598   VAR
10599     mmv$aging_statistics: [XREF] mmt$aging_statistics;
10600


o 10629 {Pointer to the Active Segment Table - (AST).}
o 10630
o 10631   VAR
o 10632     mmv$ast_p: [XREF] ^mmt$active_segment_table;
o 10633
```

```
10637 {Define async worklist for processing periodic activities.
10638
10639   VAR
10640     mmv$async_work: [XREF] mmt$async_work_list;
10641


10654
10655   VAR
10656     mmv$max_working_set_size: [XREF] integer;


10658
10659 { The following variable contains the maximum segment number of a global template segment.
10660
10661   VAR
10662     mmv$max_template_segment_number: [XREF] integer;


10664 {The following variable indicates if the configuration consists of multiple
10665 {caches that are not hardware connected for unified cache purging - ie, if a cache
10666 {purge is required each processor must purge its own cache.
10667
10668   VAR
10669     mmv$multiple_caches: [XREF] boolean;
10670


10672 {The following variable indicates if the configuration consists of multiple
10673 {page MAPS that are not hardware connected for unified map purging - ie,
10674 {if a page map purge is required each processor must purge its own map.
10675
10676   VAR
10677     mmv$multiple_page_maps: [XREF] boolean;
10678


10680 {Define variable used to activate/deactivate multi-page write. Multi-page write activated
10681 {means that memory manager attempts to write all modified pages in a transfer unit whenever
10682 {any page in the transfer unit is written.
10683
10684   VAR
10685     mmv$multi_page_write: [XREF] boolean;


10687
10688 {Define variable used as the head of the linked list of tasks waiting for
10689 {memory.
10690
10691   VAR
10692     mmv$memory_wait_queue: [XREF] tmt$task_queue_link;
10693


10697
10698 {Define option used to disable buffering of aged out pages in memory, i.e. when a page
```



```
10699 {is aged out, it is written to disk (if necessary) and placed in the free queue.
10700
10701   VAR
10702     mmv$no_memory_buffering: [XREF] boolean;
10703


10705 {Pointer to the 'PAGE FRAME TABLE' (PFT)
10706
10707   VAR
10708     mmv$pft_p: [XREF] ^mmt$page_frame_table;
10709


10713 {Define pointer to array for holding PFTI lists. This array is used in monitor for holding lists
10714 {PFTIs of pages belonging to a segment.
10715
10716   VAR
10717     mmv$pfti_array_p: [XREF] ^mmt$pfti_array;
10718


10722
10723 { Global Page Queue List array.
10724
10725   VAR
10726     mmv$gpql: [XREF] mmt$global_page_queue_list;
10729


10731 {Define page table length in words.
10732
10733   VAR
10734     mmv$pt_length: [XREF] integer;
10735


10737 {Pointer to the system PAGE TABLE (PT).
10738
10739   VAR
10740     mmv$pt_p: [XREF] ^ost$page_table;
10741


10745 {The following variable contains a count of the number of page frames that can be reassigned to be
10746 {used for another purpose. The count represents the number of pages that are in the free + available
10747 {queues. The count is broken into two parts - pages with no IO active, and pages with IO active.
10748
10749   VAR
10750     mmv$reassignable_page_frames: [XREF] mmt$reassignable_page_frames;


10761
10762 {  Define the number of free and available pages that job scheduler tries to
10763 {  keep avilable for all active jobs.
10764
```

Global Variable Declarations - XREF and XDCL

```
10765    VAR
10766      mmv$resident_job_target: [XREF] integer;


10768  {Define variable used to  indicate if memory manager tables have been initialized.
10769
10770    VAR
10771      mmv$tables_initialized: [XREF] boolean;
10772


10774
10775    VAR
10776      mmv$test_reassign_asid: [XREF] boolean;
10777


10779
10780  {Define option used to force aged out pages to be written to disk
10781  {immediately when it is aged out of a job working set.
10782  {Page are written if SOON + NOW < mmv$write_aged_out_pages.
10783
10784    VAR
10785      mmv$write_aged_out_pages: [XREF] integer;
10786


10788  {Time for next periodic call to Memory Manager from CP Monitor.
10789
10790    VAR
10791      mmv$time_to_call_mem_mgr: [XREF] integer;


10793   VAR
10794      osv$keypoint_control: [XREF] ost$keypoint_control;
10795


o 11001  {System page size.}
o 11002
o 11003    VAR
o 11004      osv$page_size: [XREF] ost$page_size;
o 11005


11009
11010      VAR
11011        mmv$shared_pages_in_jws: [XREF] boolean;
11012


11014    VAR
11015      jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
11016
```


Global Variable Declarations - XREF and XDCL

```
o 11020
o 11021    VAR
o 11022      osv$cpus_physically_configured: [XREF] 1 .. osc$max_number_of_processors;
o 11023


11027  {This variable specifies the lower and upper RMA addresses available to NOSVE.
11028
11029    VAR
11030      osv$180_memory_limits: [XREF] record
11031        lower: 0 .. 0ffffffff(16),
11032        deadstart_upper: 0 .. 0ffffffff(16),  {  Upper limit of memory during deadstart.
11033        upper: 0 .. 0ffffffff(16),
11034      recend;
11035


11037    VAR
11038      osv$time_to_check_asyn: [XREF] integer;
11039


11041
11042    VAR
11043      tmv$cpu_execution_statistics: [XREF] tmt$cpu_execution_statistics;
11044


o 11053
o 11054    VAR
o 11055      tmv$dispatching_controls: [XREF] tmt$dispatching_controls;
o 11056


11084
11085    VAR
11086      tmv$dispatching_control_sets: [XREF] tmt$dispatching_control_sets;
11087


o 11106
o 11107  {If a job goes into WAIT with a long time requested but expects a short time,
o 11108  {the scheduler does not swap it out. If it doesnt go ready with the amount of ti
o 11109  {specified by this constant, the scheduler will swap it out anyway.
o 11110
o 11111    VAR
o 11112      tmv$long_wait_force_swap_time: [XREF] integer;
o 11113


o 11115
o 11116    VAR
o 11117      tmv$ptl_p: [XREF] ^tmt$primary_task_list;
o 11118
```

Global Variable Declarations - XREF and XDCL

```
o 11169
o 11170   VAR
o 11171     tmv$timed_wait_not_queued: [XREF] integer;
o 11172


o 11174   VAR
o 11175     mmv$ring1_request_trace: [XDCL, #GATE] ARRAY [0 .. 20] of integer,
o 11176     mmv$free_file_server_pages: [XDCL] boolean := FALSE,
o 11177     mmv$jws_queue_age_interval: [XDCL, #GATE] integer := mmc$mmu_jws_age_interval,
o 11178     mmv$reduce_jws_for_thrashing: [XDCL] boolean := FALSE,
o 11179     mmv$shared_queue_age_interval: [XDCL, #GATE] integer := mmc$mmu_shared_age_interval,
o 11180     mmv$last_active_shared_queue: [XDCL, #GATE] mmt$global_page_queue_index := mmc$pq_shared_last_sys,
o 11181     mmv$periodic_call_interval: [XDCL, #GATE] integer := mmc$mmu_periodic_call_interval,
o 11182     mmv$searched_entire_pft: integer := 0,
o 11183     mmv$test_pt_full: [XDCL] integer := 0,
o 11184     mmv$total_page_frames: [XDCL, #GATE] mmt$page_frame_index := 1500, {deadstart init resets exactly}
o 11185     mmv$trap_m: [XDCL] 0..255 := 0,
o 11186     mmv$image_file: [XDCL, #GATE] mmt$image_file := [FALSE, * , * ],
o 11187     mmv$aging_algorithm: [XDCL, #GATE] integer := mmc$mmu_aging_algorithm,
o 11188     syv$user_templates: [XDCL, #GATE] boolean := FALSE;
o 11189
```


INLINE PROCEDURES FROM COMMON DECKS

```
o 11191   PROCEDURE [INLINE] mmp$aste_pointer_from_pfti (pfti: mmt$page_frame_index;
o 11192     VAR aste_p: ^mmt$active_segment_table_entry);
o 11193
o 11194     ? IF mmc$debug_aste_p_from_pfti THEN
o 11195         mmp$aste_pointer (mmv$pft_p^ [pfti].sva.asid, aste_p);
o 11196         IF aste_p <> mmv$pft_p^ [pfti].aste_p THEN
o 11197           mtp$error_stop ('MM - ERROR IN ASTE_POINTER_FROM_PFTI');
o 11198         IFEND;
o 11199     ? ELSE
  11200         aste_p := mmv$pft_p^ [pfti].aste_p;
o 11201     ? IFEND;
  11202
  11203   PROCEND mmp$aste_pointer_from_pfti;


  11205   PROCEDURE [INLINE] mmp$purge_all_cache;
  11206
  11207     VAR
  11208       null_sva: 0 .. 0ffffffffffff(16);
  11209
  11210     IF mmv$multiple_caches THEN
  11211       mmp$purge_all_cache_proc;
  11212     ELSE
  11213       #purge_buffer (osc$purge_all_cache, null_sva);
  11214     IFEND;
  11215
  11216   PROCEND;


o 11222
o 11223   PROCEDURE [INLINE] mmp$purge_all_cache_map;
o 11224
o 11225     VAR
o 11226       null_sva: 0 .. 0ffffffffffff(16);
o 11227
o 11228     IF mmv$multiple_caches OR mmv$multiple_page_maps THEN
o 11229       mmp$purge_all_cache_map_proc;
o 11230     ELSE
o 11231       #purge_buffer (osc$purge_all_cache, null_sva);
o 11232       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
o 11233     IFEND;
o 11234
o 11235   PROCEND;


  11241
  11242   PROCEDURE [INLINE] mmp$purge_all_page_map;
  11243
  11244     VAR
  11245       null_sva: 0 .. 0ffffffffffff(16);
  11246
  11247     IF mmv$multiple_page_maps THEN
  11248       mmp$purge_all_map_proc;
  11249     ELSE
  11250       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
  11251     IFEND;
```

INLINE PROCEDURES FROM COMMON DECKS

```
  11252
  11253   PROCEND;


o 11259   PROCEDURE [INLINE] mmp$sva_purge_all_page_map (sva: ost$system_virtual_address);
o 11260
o 11261     IF mmv$multiple_page_maps THEN
o 11262       mmp$purge_all_map_proc;
o 11263     ELSE
o 11264       #purge_buffer (osc$sva_purge_all_page_map, sva);
o 11265     IFEND;
o 11266
o 11267   PROCEND;


  11271   PROCEDURE [INLINE] mmp$sva_purge_one_page_map (sva: ost$system_virtual_address);
  11272
  11273     IF mmv$multiple_page_maps THEN
  11274       mmp$purge_all_map_proc;
  11275     ELSE
  11276       #purge_buffer (osc$sva_purge_one_page_map, sva);
  11277     IFEND;
  11278
  11279   PROCEND;
```


INLINE PROCEDURES FROM COMMON DECKS

```
o 11283
o 11284   PROCEDURE [INLINE] mmp$link_page_frame_to_queue (pfti: mmt$page_frame_index;
o 11285         pfte_p: ^mmt$page_frame_table_entry);
o 11286
o 11287     pfte_p^.link.fwd := mmv$gpql [pfte_p^.queue_id].pqle.link.fwd;
o 11288     IF mmv$gpql [pfte_p^.queue_id].pqle.link.fwd = 0 THEN
o 11289       mmv$gpql [pfte_p^.queue_id].pqle.link.bkw := pfti;
o 11290     ELSE
o 11291       mmv$pft_p^[mmv$gpql [pfte_p^.queue_id].pqle.link.fwd].link.bkw := pfti;
o 11292     IFEND;
o 11293     mmv$gpql [pfte_p^.queue_id].pqle.link.fwd := pfti;
o 11294     mmv$gpql [pfte_p^.queue_id].pqle.count := mmv$gpql [pfte_p^.queue_id].pqle.count + 1;
o 11295     mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
o 11296     mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now + 1;
o 11297     jmp$check_scheduler_memory_wait;
o 11298
o 11299   PROCEND mmp$link_page_frame_to_queue;
o 11300
```

GET_SYSTEM_JOBS_WORKING_SET

```
o 11303   PROCEDURE [INLINE] get_system_jobs_working_set
o 11304     (VAR working_set: mmt$page_frame_index);
o 11305
o 11306     VAR
o 11307       ijle_p: ^jmt$initiated_job_list_entry;
o 11308
o 11309     jmp$get_ijle_p (jmv$system_ijl_ordinal, ijle_p);
o 11310     working_set := ijle_p^.job_page_queue_list [mmc$pq_job_fixed].count + ijle_p^.job_page_queue_list
o 11311           [mmc$pq_job_io_error].count + ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count;
o 11312
o 11313   PROCEND get_system_jobs_working_set;
o 11314
```


MMP$UNLOCK_RMA_LIST - Unlock pages defined by rma list

```
o 11317 {-------------------------------------------------------------------------------------------------------------
o 11318 { This request is used to unlock page frames which were previously
o 11319 { locked via a MMP$BUILD_LOCK_RMA_LIST request. The 'used' and 'modified' bits
o 11320 { in the page table entry for each page frame are updated according to the
o 11321 { type of IO done into the page frames. All tasks queued waiting for the
o 11322 { page frames because of page faults are made ready when the page frames are
o 11323 { unlocked.
o 11324 { The 'active_io_page_count' of the job owning the pages is decremented.  The
o 11325 { 'inhibit_swap_count' of the job is also decremented unless the IO was a
o 11326 { write to a local file.  If the 'inhibit_swap_count' goes to zero for the
o 11327 { job and the job is being swapped out, the job swapping task is made ready.
o 11328 { If the job is swapped and a write error occurs on a local file, the page is put
o 11329 { into the io error while swapped queue.  A call is made to ready the task so the
o 11330 { job can swap in and reclaim the page.
o 11331 {
o 11332 { NOTE:  THE LIST THAT IS UNLOCKED MUST BE EXACTLY THE SAME LIST THAT WAS
o 11333 {        LOCKED BY MMP$BUILD_LOCK_RMA_LIST
o 11334 {
o 11335 {
o 11336 {     MMP$UNLOCK_RMA_LIST (IO_TYPE, LIST_P, LIST_LENGTH, IO_ID, MF_JOB_FILE,
o 11337 {          IO_ERROR, STATUS);
o 11338 {
o 11339 {  IO_TYPE: (input) This parameter specifies the type of IO that has
o 11340 {          taken place into the page frames. The value of this
o 11341 {          parameter should agree with the <io_type> passed on the
o 11342 {          MMP$BUILD_LOCK_RMA_LIST request.
o 11343 {  LIST_P: (input) This parameter points to an array which
o 11344 {          defines the page frames to be unlocked. This MUST be the same
o 11345 {          list returned on the MMP$BUILD_LOCK_RMA_LIST request.
o 11346 {          Several locked lists CANNOT be grouped together to be unlocked
o 11347 {          by mmp$unlock_rma_list.
o 11348 {  LIST_LENGTH: (input) This parameter specifies the number of entries
o 11349 {          in the RMA list. An RMA list entry that contains a 'length' of zero
o 11350 {          will terminate the list even though more entries remain.
o 11351 {  IO_ID: (input) This parameter is used to specify an io identifier that MM
o 11352 {          uses during request processing.
o 11353 {  MF_JOB_FILE: (input) This specifies whether or not the file is a local
o 11354 {          file.
o 11355 {  IO_ERROR: (input,output) This specifies how the IO to/from the pages
o 11356 {          terminated.  Page manager uses this parameter
o 11357 {          together with <io_type> to determine how to update the
o 11358 {          'used' and 'modified' bits in the page table.
o 11359 {          For local file writes, ioc$no_error is returned if the error is
o 11360 {          processed.
o 11361 {  STATUS: (output) This parameter specifies the request status.
o 11362 {          No error codes are returned.  All error will result in a call to
o 11363 {          mtp$error_stop.
o 11364 {
o 11365
o 11366 {-------------------------------------------------------------------------------------------------------------
o 11367
o 11368   PROCEDURE [XDCL] mmp$unlock_rma_list
o 11369     (     iotype: iot$io_function;
o 11370           list_p: ^mmt$rma_list;
o 11371           list_length: mmt$rma_list_length;
o 11372           io_identifier: mmt$io_identifier;
```

```
        0 11373           mf_job_file: boolean;
        0 11374         VAR io_error: iot$io_error;
        0 11375         VAR status: syt$monitor_status);
        0 11376
        0 11377
        0 11378       PROCEDURE process_write_failure;
        8 11379
        8 11380         pte_p^.v := TRUE;
        8 11381         pfte_p^.io_error := init_io_error;
        8 11382         IF pfte_p^.aste_p^.queue_id = mmc$pq_job_working_set THEN
       32 11383           IF pfte_p^.aste_p^.sfid.residence = gfc$tr_system THEN
       3A 11384             {Note link to shared error q - required by job exit to
       3A 11385             {leave pages in memory correctly.
       3A 11386             mmp$relink_page_frame (pfti, mmc$pq_shared_io_error);
       5A 11387           ELSE
       5A 11388             ijl_ordinal := pfte_p^.ijl_ordinal;
       5A 11389             jmp$get_ijle_p (ijl_ordinal, ijle_p);
       5A 11390
       5A 11391 {  If this page is already in the swapped error queue then it has already been
       5A 11392 {  put there as part of a transfer unit on an initial write.  If the job is
       5A 11393 {  swapped pass the wait io complete state (JW) and the page is part of the job
       5A 11394 {  working set leave it there and just set the modified bit.  We are OK unless
       5A 11395 {  memory gets freed.  But since there is a page in the avail modified queue
       5A 11396 {  that belongs to the same transfer unit in this write request that will be
       5A 11397 {  put into the swapped error queue, when we swap in and reclaim it we can
       5A 11398 {  reset the modified bit for all pages in this transfer unit.
       5A 11399
       5A 11400             IF (pfte_p^.queue_id <> mmc$pq_swapped_io_error) THEN
       92 11401               IF ijle_p^.swap_status <= jmc$iss_wait_job_io_complete THEN
       9C 11402                 mmp$relink_page_frame (pfti, mmc$pq_job_io_error);
       BC 11403               ELSEIF (pfte_p^.queue_id = mmc$pq_avail_modified) THEN
       C2 11404                 mmp$relink_page_frame (pfti, mmc$pq_swapped_io_error);
       DA 11405                 ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work +
       EE 11406                       $jmt$delayed_swapin_work [jmc$dsw_io_error_while_swapped];
       EE 11407               IFEND;
       F0 11408             IFEND;
       F0 11409
       F0 11410 {  If this is an initial write that failed we must remove any other pages in the
       F0 11411 {  transfer unit that might have been queued for IO after this request was made.
       F0 11412
       F0 11413             IF(io_error = ioc$error_on_init) OR (io_error = ioc$unit_down_on_init) THEN
      108 11414               remove_pages_in_tu (ijle_p, pfte_p);
      11E 11415             IFEND;
      11E 11416             io_error := ioc$no_error;
      12A 11417
      12A 11418           IFEND;
      12E 11419         ELSE
      12E 11420           mmp$relink_page_frame (pfti, mmc$pq_shared_io_error);
      14A 11421         IFEND;
      14A 11422         pte_p^.m := TRUE; {Must be after RELINK}
      14A 11423
      14A 11424       PROCEND process_write_failure;
        0 11425
        0 11426       PROCEDURE process_rewrite_success;
        8 11427
        8 11428         IF pfte_p^.active_io_count = 0 THEN
```



```
       14 11429           WHILE pfte_p^.task_queue.head <> 0 DO
       1C 11430             tmp$dequeue_task (pfte_p^.task_queue, taskid);
       3C 11431           WHILEND;
       44 11432           {Page that was in error queue has been written correctly !!
       44 11433           pfte_p^.io_error := ioc$no_error;
       44 11434           mmv$successful_error_retry := mmv$successful_error_retry + 1;
       44 11435
       44 11436 { If page is in available modified, move it to the correct queue.  If the page
       44 11437 { is part of the working set and the job is not swapped pass the wait_io_complete
       44 11438 { state, move it to the correct queue.  Else leave it were it is.  Otherwise
       44 11439 { mmv$reassignable_page_frames.soon will not be decremented correctly.
       44 11440
       44 11441           IF NOT mmv$pt_p^ [pfte_p^.pti].m THEN
       78 11442             ijl_ordinal := pfte_p^.ijl_ordinal;
       78 11443             jmp$get_ijle_p (ijl_ordinal, ijle_p);
       78 11444             IF (ijle_p^.swap_status <= jmc$iss_wait_job_io_complete) OR
       B8 11445                   (pfte_p^.queue_id = mmc$pq_avail_modified) THEN
       B8 11446               pte_p^.v := FALSE;
       B8 11447               mmp$sva_purge_one_page_map (pfte_p^.sva); {Essential for dual CPU}
       E4 11448               IF mmv$no_memory_buffering THEN
       F0 11449                 mmp$delete_pt_entry (pfti, TRUE);
      10A 11450                 mmp$relink_page_frame (pfti, mmc$pq_free);
      120 11451               ELSE
      120 11452                 mmp$relink_page_frame (pfti, mmc$pq_avail);
      13A 11453               IFEND;
      13A 11454             IFEND;
      13A 11455           IFEND;
      13A 11456         IFEND;
      13A 11457
      13A 11458       PROCEND process_rewrite_success;
        0 11459
        0 11460       PROCEDURE remove_pages_in_tu
        0 11461         (     ijle_p: ^jmt$initiated_job_list_entry;
        0 11462               init_pfte_p: ^mmt$page_frame_table_entry);
        0 11463
        0 11464         CONST
        0 11465           allow_allocation = TRUE;
        0 11466         VAR
        0 11467           ajlo: jmt$ajl_ordinal,
        0 11468           boffset: integer,
        0 11469           eoffset: integer,
        0 11470           fde_p: gft$file_desc_entry_p,
        0 11471           info: dmt$chapter_info,
        0 11472           pfte_p: ^mmt$page_frame_table_entry,
        0 11473           pfti: mmt$page_frame_index,
        0 11474           pte_p: ^ost$page_table_entry;
        0 11475
        0 11476         IF ijle_p^.swap_status >= jmc$iss_free_swapped_memory THEN
       14 11477           IF (init_pfte_p^.sva.offset - dmc$max_transfer_size) < 0 THEN
       26 11478             boffset := 0;
       2C 11479           ELSE
       2C 11480             boffset := init_pfte_p^.sva.offset - dmc$max_transfer_size;
       36 11481           IFEND;
       36 11482           IF ((init_pfte_p^.sva.offset - dmc$max_transfer_size) <= osc$max_segment_length) THEN
       4C 11483             eoffset := init_pfte_p^.sva.offset + dmc$max_transfer_size;
       56 11484           ELSE
```

```
      56 11485            eoffset := osc$max_segment_length;
      5C 11486          IFEND;
      60 11487        ELSE
      60 11488          jmp$lock_ajl (ijle_p, init_pfte_p^.aste_p^.ijl_ordinal, ajlo);
     144 11489
     144 11490          gfp$mtr_get_fde_p (init_pfte_p^.aste_p^.sfid, ijle_p, fde_p);
     1AA 11491          boffset := init_pfte_p^.sva.offset DIV fde_p^.allocation_unit_size *fde_p^.allocation_unit_size;
     1AA 11492          eoffset := boffset + fde_p^.allocation_unit_size;
     1AA 11493
     1AA 11494          jmp$unlock_ajl (ijle_p);
     278 11495        IFEND;
     278 11496
     278 11497        pfti := init_pfte_p^.aste_p^.pft_link.fwd;
     278 11498
     278 11499        WHILE pfti <> 0 DO
     284 11500          pfte_p := ^mmv$pft_p^ [pfti];
     284 11501          IF (pfte_p^.sva.offset >= boffset) AND (pfte_p^.sva.offset < eoffset) AND
     2BC 11502                (pfte_p^.active_io_count <> 0) THEN
     2BC 11503            IF pfte_p^.locked_page = mmc$lp_page_in_lock THEN
     2C6 11504              mmp$delete_pt_entry (pfti, TRUE);
     2DC 11505              mmp$relink_page_frame (pfti, mmc$pq_free);
     2F6 11506            ELSE
     2F6 11507
     2F6 11508 {    Assume write - other iotypes are not used with local files.
     2F6 11509 {    If this page is already in the swapped error queue then it was put there as part
     2F6 11510 {    of a transfer unit on another initial write.  The job is swapped and memory has
     2F6 11511 {    been freed so the maximum transfer size was used.
     2F6 11512 {    If the job is swapped pass the wait job io complete state and the page is part of
     2F6 11513 {    the job working set leave the page where it is and just set the modified bit.
     2F6 11514 {    If memory gets freed, all pages in the transfer unit will get their modified bit
     2F6 11515 {    reset when we swap in and reclaim io error pages.
     2F6 11516
     2F6 11517              IF (pfte_p^.queue_id <> mmc$pq_swapped_io_error) THEN
     302 11518                pte_p := ^mmv$pt_p^ [pfte_p^.pti];
     302 11519                pte_p^.v := TRUE;
     302 11520                IF ijle_p^.swap_status < jmc$iss_job_io_complete THEN
     328 11521                  mmp$relink_page_frame (pfti, mmc$pq_job_io_error);
     344 11522                ELSEIF (pfte_p^.queue_id = mmc$pq_avail_modified) THEN
     34A 11523                  mmp$relink_page_frame (pfti, mmc$pq_swapped_io_error);
     360 11524                IFEND;
     360 11525                pte_p^.m := TRUE;
     366 11526              IFEND;
     36A 11527            IFEND;
     36A 11528          IFEND;
     36A 11529          pfti := mmv$pft_p^ [pfti].segment_link.fwd;
     36A 11530        WHILEND;
     38A 11531
     38A 11532      PROCEND remove_pages_in_tu;
       0 11533
```



```
       0 11535
       0 11536      VAR
       0 11537        count: integer,
       0 11538        decrement_inhibit_swap: 0 .. mmc$max_rma_list_length,
       0 11539        ijl_ordinal: jmt$ijl_ordinal,
       0 11540        ijle_p: ^jmt$initiated_job_list_entry,
       0 11541        initial_reassignable_now: integer,
       0 11542        list_i: mmt$rma_list_index,
       0 11543        mmv$successful_error_retry: [XDCL] integer := 0,
       0 11544        pfte_p: ^mmt$page_frame_table_entry,
       0 11545        pte_p: ^ost$page_table_entry,
       0 11546        pfti: mmt$page_frame_index,
       0 11547        taskid: ost$global_task_id,
       0 11548        init_io_error: iot$io_error;
       0 11549
       0 11550      status.normal := TRUE;
       8 11551
       8 11552      init_io_error := io_error;
       8 11553      initial_reassignable_now := mmv$reassignable_page_frames.now;
       8 11554
       8 11555      decrement_inhibit_swap := list_length;
       8 11556      IF (iotype = ioc$swap_out) OR (iotype = ioc$swap_in) THEN
      3C 11557        ijl_ordinal := io_identifier.ijl_ordinal;
      3C 11558        jmp$get_ijle_p (ijl_ordinal, ijle_p);
      3C 11559        IF io_error <> ioc$no_error THEN
      72 11560          ijle_p^.swap_data.swapping_io_error := io_error;
      76 11561        IFEND;
      7A 11562
      7A 11563      ELSE { Not swap io }
      7A 11564
      7A 11565      /unlock_pages/
      7A 11566        FOR list_i := 1 TO list_length DO
      80 11567          IF list_p^ [list_i].length = 0 THEN
      94 11568            EXIT /unlock_pages/;
      98 11569          IFEND;
      98 11570          pfti := list_p^ [list_i].rma DIV osv$page_size;
      98 11571          #KEYPOINT (osk$debug, pfti * osk$m, mmk$unlock_rma1);
      B8 11572
      B8 11573          pfte_p := ^mmv$pft_p^ [pfti];
      B8 11574          IF pfte_p^.active_io_count = 0 THEN
      E0 11575            mtp$error_stop ('mm - unlock rma list error');
     100 11576          IFEND;
     100 11577          pfte_p^.active_io_count := pfte_p^.active_io_count - 1;
     100 11578
     100 11579          IF pfte_p^.queue_id = mmc$pq_free THEN
     116 11580            IF pfte_p^.active_io_count = 0 THEN
     11A 11581              IF (io_error <> ioc$no_error) AND (iotype = ioc$keypoint_io) THEN
     12C 11582                osp$process_keypoint_io_error;
     134 11583              IFEND;
     134 11584              mmp$link_page_frame_to_queue (pfti, pfte_p);
     1D0 11585            IFEND;
     1D4 11586          ELSE
     1D4 11587            pte_p := ^mmv$pt_p^ [pfte_p^.pti];
     1D4 11588
     1D4 11589 {!!!        ?IF mmc$debug_rma_list THEN
     1D4 11590              IF (iotype = ioc$write_locked_page) AND (pfte_p^.queue_id < mmc$pq_first_valid_in_pt) THEN
```

MMP$UNLOCK_RMA_LIST - Unlock pages defined by rma list

```
1FA 11591                mtp$error_stop ('MM - unlock rma1, bad queue');
21A 11592              IFEND;
21A 11593 {!!!       ?IFEND;
21A 11594
21A 11595            CASE iotype OF
28C 11596            = ioc$no_io =
2C2 11597            = ioc$explicit_write, ioc$write_mass_storage, ioc$initialize_sectors
2C8 11598              , ioc$write_to_client  =
2C8 11599            = ioc$explicit_read, ioc$read_uft, ioc$read_mass_storage, ioc$explicit_read_no_purge =
2C8 11600              IF (list_i = 1) AND (iotype <> ioc$explicit_read_no_purge) THEN
2D6 11601                mmp$purge_all_cache;
2F8 11602              IFEND;
2F8 11603              IF (pfte_p^.queue_id = mmc$pq_avail_modified) AND NOT pte_p^.m THEN
312 11604                mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
31C 11605              IFEND;
31C 11606              pte_p^.m := TRUE;
32A 11607            = ioc$read_from_client =
32A 11608 {!           We will decide whether or not to do this when we support active readers on the server with
32A 11609 {!           writers on (multiple) clients.
32A 11610 {!           IF list_i = 1 THEN
32A 11611 {!             mmp$purge_all_cache;
32A 11612 {!           IFEND;
32A 11613              pfte_p^.locked_page := mmc$lp_not_locked;
32A 11614              IF io_error = ioc$no_error THEN
33C 11615                IF pfte_p^.queue_id >= mmc$pq_first_valid_in_pt THEN
348 11616                  pte_p^.v := TRUE;
356 11617                ELSEIF NOT pte_p^.m THEN
362 11618                  mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
36E 11619                IFEND;
36E 11620                pte_p^.u := TRUE;
36E 11621                pte_p^.m := TRUE;
386 11622              ELSE
386 11623                mmp$unlock_rma_list_error (pfti, pfte_p, io_error);
398 11624              IFEND;
39C 11625            = ioc$read_page, ioc$read_for_server, ioc$read_ahead_on_server =
39C 11626              pfte_p^.locked_page := mmc$lp_not_locked;
39C 11627              IF io_error = ioc$no_error THEN
3AE 11628                IF pfte_p^.queue_id >= mmc$pq_first_valid_in_pt THEN
3B8 11629                  pte_p^.v := TRUE;
3B8 11630                  pte_p^.u := TRUE;
3CA 11631                IFEND;
3CE 11632              ELSE
3CE 11633                mmp$unlock_rma_list_error (pfti, pfte_p, io_error);
3DE 11634              IFEND;
3E2 11635            = ioc$allocate =
3E2 11636              pfte_p^.locked_page := mmc$lp_not_locked;
3E2 11637              IF io_error = ioc$no_error THEN
3F4 11638                IF pfte_p^.queue_id >= mmc$pq_first_valid_in_pt THEN
3FE 11639                  pte_p^.v := TRUE;
408 11640                IFEND;
40C 11641              ELSE
40C 11642                mmp$unlock_rma_list_error (pfti, pfte_p, io_error);
41C 11643              IFEND;
420 11644            = ioc$write_page, ioc$write_locked_page, ioc$write_for_server =
420 11645              IF iotype = ioc$write_locked_page THEN
42A 11646                IF pfte_p^.queue_id >= mmc$pq_first_valid_in_pt THEN
```


MMP$UNLOCK_RMA_LIST - Unlock pages defined by rma list

```
438 11647                  pte_p^.v := TRUE;
444 11648                IFEND;
444 11649                pfte_p^.locked_page := mmc$lp_not_locked;
44E 11650              IFEND;
44E 11651              IF init_io_error <> ioc$no_error THEN
456 11652                process_write_failure;
464 11653              ELSEIF pfte_p^.io_error <> ioc$no_error THEN
470 11654                process_rewrite_success;
47C 11655              IFEND;
482 11656            ELSE
482 11657              mtp$error_stop ('MM - bad IO type unlock_rma_list');
4A2 11658            CASEND;
4A2 11659          IFEND;
4A2 11660
4A2 11661 { If the page is no longer locked (all IO complete), there is some more processing to be done:
4A2 11662 { Dequeue all tasks waiting for the IO to complete.
4A2 11663
4A2 11664          IF pfte_p^.active_io_count = 0 THEN
4AE 11665            WHILE pfte_p^.task_queue.head <> 0 DO
4B6 11666              tmp$dequeue_task (pfte_p^.task_queue, taskid);
4D6 11667            WHILEND;
4E2 11668
4E2 11669 {  If the page is in the available modified queue and has been sucessfully written to disk,
4E2 11670 {  the page should be moved to the available queue. (Debugging option allows for available
4E2 11671 {  queue to be disabled).
4E2 11672
4E2 11673            IF (pfte_p^.queue_id = mmc$pq_avail_modified) THEN
4F0 11674              IF NOT mmv$pt_p^ [pfte_p^.pti].m THEN
50C 11675                IF mmv$no_memory_buffering THEN
518 11676                  mmp$delete_pt_entry (pfti, TRUE);
530 11677                  mmp$relink_page_frame (pfti, mmc$pq_free);
54A 11678                ELSE
54A 11679                  mmp$relink_page_frame (pfti, mmc$pq_avail);
564 11680                IFEND;
564 11681              IFEND;
568 11682
568 11683  {  If the page was being used by file server on the SERVER and and has been sucessfully
568 11684  {  written to disk/client, remove the page from the shared queue if possible. NOTE that
568 11685  { the page cannot be removed if it is modified. Also note that since the page is VALID
568 11686  { the MODIFIED BIT in the page table cannot be examined until the VALID BIT
568 11687  { is cleared and the page maps purged.
568 11688
568 11689            ELSEIF (pfte_p^.queue_id >= mmc$pq_shared_first) AND (pfte_p^.queue_id <= mmc$pq_shared_last) AND
58A 11690                ((iotype = ioc$write_to_client) OR (iotype = ioc$write_for_server)) THEN
58A 11691              pte_p^.v := FALSE;
58A 11692              mmp$sva_purge_one_page_map (pfte_p^.sva); {Essential for dual CPU}
5B8 11693              IF pte_p^.m THEN
5C6 11694                pte_p^.v := TRUE;
5D0 11695              ELSEIF mmv$free_file_server_pages THEN
5DC 11696                mmp$delete_pt_entry (pfti, TRUE);
5F4 11697                mmp$relink_page_frame (pfti, mmc$pq_free);
60E 11698              ELSE
60E 11699                mmp$relink_page_frame (pfti, mmc$pq_avail);
628 11700              IFEND;
628 11701            IFEND;
628 11702          IFEND;
```

```
628 11703
628 11704       FOREND /unlock_pages/;
62C 11705
62C 11706       ijl_ordinal := pfte_p^.ijl_ordinal;
62C 11707       jmp$get_ijle_p (ijl_ordinal, ijle_p);
62C 11708
62C 11709
62C 11710 {  If IO was a "write" of a local file, dont decrement inhibit swap.
62C 11711 {  If an IO error occurred on a swapped job, set the task ready so it can swap in and reclaim
62C 11712 {  its error pages.
62C 11713
62C 11714       IF (mf_job_file) AND ((iotype = ioc$write_page) OR (iotype = ioc$write_locked_page)) THEN
674 11715         decrement_inhibit_swap := 0;
674 11716         IF (init_io_error <> ioc$no_error) AND
696 11717               (ijle_p^.entry_status <> jmc$ies_entry_free) AND
696 11718               (ijle_p^.swap_status >= jmc$iss_job_io_complete) THEN
696 11719           tmp$set_task_ready (ijle_p^.job_monitor_taskid, 0 {readying_task_priority},
6B2 11720                 tmc$rc_ready_conditional_wi);
6B2 11721         IFEND;
6B4 11722       IFEND;
6B4 11723
6B4 11724
6B4 11725     IFEND; { Not swap io }
6B4 11726
6B4 11727 { The active_io_page_count must always be accurate but the number of active_io_requests can sometimes
6B4 11728 { be incorrect.  If pages are moved from a JWS to the shared working set, the active_io_page_count is
6B4 11729 { modified to reflect the move but the active_io_requests count can not be modified because it is not
6B4 11730 { known how many IO requests were made.  Therefore, the code here must ensure active_io_requests does
6B4 11731 { not become negative and if the active_io_active count becomes zero then zero active_io_requests.  If
6B4 11732 { active_io_count is non-zero then we know there must be at least one IO request outstanding therefore
6B4 11733 { do not decrement active_io_requests if it is not greater than one.
6B4 11734 { This will correct the active_io_requests count if it was incorrect.  The only use of the requests
6B4 11735 { count is to slowdown the task if it initiates too many IO requests while at MAXWS.
6B4 11736
6B4 11737     IF ijle_p^.active_io_page_count < list_length THEN
6C4 11738       mtp$error_stop ('MM-NEGATIVE IO COUNT IN UNLOCK RMA');
6E4 11739     IFEND;
6E4 11740
6E4 11741     ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count - decrement_inhibit_swap;
6E4 11742     ijle_p^.active_io_page_count := ijle_p^.active_io_page_count - list_length;
6E4 11743     IF ijle_p^.active_io_page_count = 0 THEN
704 11744       ijle_p^.active_io_requests := 0;
70E 11745     ELSEIF ijle_p^.active_io_requests > 1 THEN
716 11746       ijle_p^.active_io_requests := ijle_p^.active_io_requests - 1;
71C 11747     IFEND;
71C 11748     IF (ijle_p^.inhibit_swap_count = 0) AND (ijle_p^.notify_swapper_when_io_complete) THEN
730 11749       jsp$io_complete (ijle_p);
740 11750     IFEND;
740 11751
740 11752     count := mmv$reassignable_page_frames.now - initial_reassignable_now;
740 11753     WHILE (count > 0) AND (mmv$memory_wait_queue.head <> 0) DO
756 11754       tmp$dequeue_task (mmv$memory_wait_queue, taskid);
76E 11755       count := count - 1;
76E 11756     WHILEND;
77C 11757
77C 11758     mmp$check_queues;
```



```
790 11759
790 11760   PROCEND mmp$unlock_rma_list;


  0 11762
  0 11763 {This procedure is called to process an IO error on a page-in request.
  0 11764
  0 11765   PROCEDURE mmp$unlock_rma_list_error
  0 11766     (     pfti: mmt$page_frame_index;
  0 11767           pfte_p: ^mmt$page_frame_table_entry;
  0 11768           io_error: iot$io_error);
  0 11769
  0 11770     VAR
  0 11771       delete_pt_entry_ok: boolean,
  0 11772       ijle_p: ^jmt$initiated_job_list_entry,
  0 11773       monitor_fault: ost$monitor_fault,
  0 11774       sac_p: ^mmt$segment_access_condition,
  0 11775       sdte_segment_number: ost$segment,
  0 11776       sdtxe_p: ^mmt$segment_descriptor_extended,
  0 11777       status: syt$monitor_status,
  0 11778       taskid: ost$global_task_id,
  0 11779       xcb_p: ^ost$execution_control_block;
  0 11780
  0 11781     monitor_fault.identifier := mmc$segment_fault_processor_id;
  4 11782     sac_p := #LOC (monitor_fault.contents);
  E 11783     sac_p^.identifier := mmc$sac_io_read_error;
  E 11784     delete_pt_entry_ok := TRUE;
  E 11785
  E 11786     tmp$get_taskid_from_task_queue (pfte_p^.task_queue, taskid);
  E 11787     WHILE taskid <> tmv$null_global_task_id DO
 50 11788       tmp$get_xcb_p (taskid, xcb_p, ijle_p);
 74 11789       IF xcb_p <> NIL THEN
 7E 11790         IF io_error = ioc$unrecovered_error_unit_down THEN
 84 11791           mmp$process_volume_unavailable (xcb_p, FALSE);
 9E 11792         ELSEIF io_error = ioc$server_allocation_error THEN
 A4 11793           sdte_segment_number := #SEGMENT (xcb_p^.page_wait_info.pva);
 A4 11794           sdtxe_p := mmp$get_sdtx_entry_p (xcb_p, sdte_segment_number);
 A4 11795           sdtxe_p^.assign_active := 0;
 A4 11796           tmp$set_monitor_flag (taskid, mmc$mf_segment_mgr_flag, status);
 FE 11797           IF (xcb_p^.xp.trap_enable <> osc$traps_enabled) OR (xcb_p^.xp.p_register.pva.ring = 1) THEN
11C 11798             delete_pt_entry_ok := FALSE;
11C 11799             mmv$pt_p^ [pfte_p^.pti].v := TRUE;
138 11800           IFEND;
13E 11801         ELSEIF io_error = ioc$server_has_terminated THEN
144 11802           sac_p^.identifier := mmc$sac_file_server_terminated;
144 11803           IF xcb_p^.page_wait_info.pva <> NIL THEN
154 11804             sac_p^.segment := xcb_p^.page_wait_info.pva;
154 11805             tmp$send_monitor_fault (taskid, #LOC (monitor_fault), TRUE);
178 11806           IFEND;
17E 11807         ELSE {io_error <> ioc$unrecovered_error_unit_down, io_error <> ioc$server_allocation_error}
17E 11808           IF xcb_p^.page_wait_info.pva <> NIL THEN
188 11809             sac_p^.segment := xcb_p^.page_wait_info.pva;
188 11810             tmp$send_monitor_fault (taskid, #LOC (monitor_fault), TRUE);
1AC 11811           IFEND;
1AE 11812         IFEND;
```

MMP$UNLOCK_RMA_LIST - Unlock pages defined by rma list

```
      1AE 11813
      1AE 11814          jmp$unlock_ajl (ijle_p);
      264 11815        IFEND;
      264 11816
      264 11817        tmp$find_next_queued_task (taskid);
      278 11818      WHILEND;
      284 11819
      284 11820      IF NOT delete_pt_entry_ok THEN
      288 11821        RETURN;
      28A 11822      IFEND;
      28A 11823
      28A 11824      mmp$delete_pt_entry (pfti, TRUE);
      29E 11825      mmp$relink_page_frame (pfti, mmc$pq_free);
      2B4 11826
      2B4 11827    PROCEND mmp$unlock_rma_list_error;
        0 11828
```


MMP$BUILD_LOCK_RMA_LIST - Build and lock pages defined by rma list

```
        0 11831
        0 11832 {-------------------------------------------------------------------------------------------------------------
        0 11833 { This request is used by physical IO to lock one or more page frames
        0 11834 { before doing IO to the page frames. Page frames being accessed by
        0 11835 { the IOU must be locked to prevent memory manager from reassigning the
        0 11836 { page frame while IO is active. If page frames are not assigned to
        0 11837 { the entire range of addresses specified in the request, no frames
        0 11838 { will be locked and an error code will be returned.
        0 11839 { The 'active_io_page_count' for the job owning the page frames is incremented.
        0 11840 { The 'inhibit_swap_count' is incremented if the IO is not a write to a
        0 11841 { local file.  The counts are decremented when the page frames are unlocked.
        0 11842 {
        0 11843 {   NOTE:  ALL pages locked MUST belong to the same segment.
        0 11844 {          THE LIST THAT IS LOCKED MUST BE THE SAME LIST THAT IS UNLOCKED.
        0 11845 {          64K byte page size is not currently supported.
        0 11846 {
        0 11847 {     MMP$BUILD_LOCK_RMA_LIST (BUFFER_DESCRIPTOR, LENGTH, IO_TYPE, LIST_P,
        0 11848 {                LIST_LENGTH, STATUS);
        0 11849 {
        0 11850 {  BUFFER_DESCRIPTOR: (input) This parameter specifies the pages of memory
        0 11851 {          to be locked.  All pages must belong to the same segment.
        0 11852 {  LENGTH: (input) This parameter specifies the number of bytes to be
        0 11853 {          locked.
        0 11854 {  IO_TYPE: (input) This parameter specifies the type of IO that will
        0 11855 {          take place into the page frames.
        0 11856 {  LIST_P: (input) This parameter points to an array into which this
        0 11857 {          procedure returns a list of the real memory addresses (RMA) of
        0 11858 {          the pages locked. One entry is required in this list for
        0 11859 {          each page frame containing portions of the area being
        0 11860 {          locked.  The list that is locked by mmp$build_lock_rma_list
        0 11861 {          must be the same list that is unlocked by mmp$unlock_rma_list.
        0 11862 {          Several locked lists CANNOT be grouped together to be unlocked
        0 11863 {          by mmp$unlock_rma_list.
        0 11864 {  LIST_LENGTH: (input) This parameter specifies the number of entries
        0 11865 {          in the RMA list. If the length of the RMA list exceeds the
        0 11866 {          length required, the 'length' field in the unused RMA list
        0 11867 {          entries will be set to zero. If the rma list is not large enough, a
        0 11868 {          fatal monitor abort will occur.
        0 11869 {  STATUS: (output) This parameter specifies the request status.
        0 11870 {          The only error code returned is mme$page_frame_not_assigned.
        0 11871 {          All other errors will result in a call to mtp$error_stop.
        0 11872 {
        0 11873
        0 11874 {-------------------------------------------------------------------------------------------------------------
        0 11875
        0 11876   PROCEDURE [XDCL] mmp$build_lock_rma_list
        0 11877     (    buffer_descriptor: mmt$buffer_descriptor;
        0 11878          length: ost$byte_count;
        0 11879          iotype: iot$io_function;
        0 11880          list_p: ^mmt$rma_list;
        0 11881          list_length: mmt$rma_list_length;
        0 11882       VAR status: syt$monitor_status);
        0 11883
        0 11884     VAR
        0 11885       found: boolean,
        0 11886       hash_count: 1 .. 32,
```

MMP$BUILD_LOCK_RMA_LIST - Build and lock pages defined by rma list

```
        0 11887        ijle_p: ^jmt$initiated_job_list_entry,
        0 11888        ijl_ordinal: jmt$ijl_ordinal,
        0 11889        index: integer,
        0 11890        io_error: iot$io_error,
        0 11891        ioid: mmt$io_identifier,
        0 11892        jf_asid: ost$asid,
        0 11893        purge_map: boolean,
        0 11894        list_i: mmt$rma_list_index,
        0 11895        mf_job_file: boolean,
        0 11896        osv$keypoint_periodic_lpid: [XREF] integer,
        0 11897        page_count: integer,
        0 11898        page_offset: 0 .. 65535,
        0 11899        pfte_p: ^mmt$page_frame_table_entry,
        0 11900        pfti: mmt$page_frame_index,
        0 11901        pti: integer,
        0 11902        spde_p: ^jst$swapped_page_descriptor,
        0 11903        sva: ost$system_virtual_address;
        0 11904
        0 11905
        0 11906      status.normal := TRUE;
        8 11907      list_i := 1;
        8 11908
        8 11909
        8 11910 {    Lock the pages depending on format of the buffer descriptor.
        8 11911
        8 11912      CASE buffer_descriptor.buffer_descriptor_type OF
       2E 11913
       2E 11914      = mmc$bd_paging_io, mmc$bd_explicit_io =
       2E 11915        sva := buffer_descriptor.sva;
       2E 11916        page_offset := sva.offset MOD osv$page_size;
       2E 11917        page_count := ((page_offset + length - 1) DIV osv$page_size) + 1;
       2E 11918        IF (list_length < page_count) OR (list_length = 0) THEN
       64 11919          mtp$error_stop ('MM - lock rma1, list too small');
       84 11920        IFEND;
       84 11921        purge_map := FALSE;
       84 11922
       84 11923      /lp/
       84 11924        WHILE TRUE DO
       86 11925          IF iotype <> ioc$keypoint_io THEN
       90 11926            #HASH_SVA (sva, pti, hash_count, found);
       92 11927            IF NOT found THEN
       A6 11928              EXIT /lp/
       AA 11929            IFEND;
       AA 11930            pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
       CC 11931          ELSE
       CC 11932            pfti := osv$keypoint_control.cpus [osv$keypoint_periodic_lpid].io_pfti [list_i];
       EC 11933          IFEND;
       EC 11934          #KEYPOINT (osk$debug, pfti * osk$m, mmk$build_lock_rma1);
       F4 11935          pfte_p := ^mmv$pft_p^ [pfti];
       F4 11936          pfte_p^.active_io_count := pfte_p^.active_io_count + 1;
       F4 11937          ?IF mmc$debug_rma_list THEN
       F4 11938            IF list_i = 1 THEN
      120 11939              ijl_ordinal := pfte_p^.ijl_ordinal;
      128 11940            ELSEIF ijl_ordinal <> pfte_p^.ijl_ordinal THEN
      13C 11941              mtp$error_stop ('MM - lock rma1, mixed ijl ordinal in lock');
      15C 11942            IFEND;
```


MMP$BUILD_LOCK_RMA_LIST - Build and lock pages defined by rma list

```
      15C 11943            IF ((iotype = ioc$write_page) OR (iotype = ioc$write_locked_page)) AND NOT mmv$pt_p^ [pti].m THEN
      182 11944              mtp$error_stop ('MM - lock rma1, write page error');
      1A2 11945            IFEND;
      1A2 11946            IF (iotype = ioc$read_page) AND ((pfte_p^.queue_id < mmc$pq_first_valid_in_pt) OR mmv$pt_p^ [pti].
                                v) THEN
      1CE 11947              mtp$error_stop ('MM - lock rma1, read page error');
      1CE 11948            IFEND;
      1F2 11949          ?IFEND;
      1F2 11950
      1F2 11951
      1F2 11952          CASE iotype OF
      2A6 11953          = ioc$explicit_read, ioc$read_uft, ioc$read_mass_storage, ioc$explicit_write, ioc$write_mass_storage,
      2A6 11954                ioc$initialize_sectors, ioc$explicit_read_no_purge =
      2A6 11955            IF pfte_p^.queue_id = mmc$pq_avail THEN
      2AE 11956              mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
      2CA 11957              mmv$pt_p^ [pti].v := TRUE;
      2E0 11958            IFEND;
      2E4 11959          = ioc$write_to_client =
      2E4 11960            IF pfte_p^.locked_page = mmc$lp_page_in_lock THEN
      2EE 11961              pfte_p^.active_io_count := pfte_p^.active_io_count - 1;
      2EE 11962              EXIT /lp/;
      2FE 11963            IFEND;
      2FE 11964            IF pfte_p^.queue_id = mmc$pq_avail THEN
      306 11965              mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
      322 11966              mmv$pt_p^ [pti].v := TRUE;
      338 11967            IFEND;
      33E 11968          = ioc$read_from_client =
      33E 11969            IF NOT mmv$pt_p^ [pti].v THEN
      358 11970              pfte_p^.locked_page := mmc$lp_page_in_lock;
      35E 11971            IFEND;
      364 11972          = ioc$no_io =
      36A 11973          = ioc$read_page, ioc$read_for_server, ioc$read_ahead_on_server =
      36A 11974            pfte_p^.locked_page := mmc$lp_page_in_lock;
      376 11975          = ioc$allocate =
      376 11976            pfte_p^.locked_page := mmc$lp_server_allocate_lock;
      382 11977          = ioc$write_page, ioc$write_for_server =
      382 11978            mmv$pt_p^ [pti].m := FALSE;
      382 11979            IF pfte_p^.queue_id = mmc$pq_avail_modified THEN
      3A2 11980              mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + 1;
      3B4 11981            ELSEIF mmv$pt_p^ [pti].v THEN
      3BE 11982              purge_map := TRUE;
      3C2 11983            IFEND;
      3C6 11984          = ioc$write_locked_page =
      3C6 11985            pfte_p^.locked_page := mmc$lp_write_protected_lock;
      3C6 11986            IF pfte_p^.queue_id = mmc$pq_avail_modified THEN
      3D4 11987              mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
      3F4 11988            ELSEIF mmv$pt_p^ [pti].v THEN
      40E 11989              mmv$pt_p^ [pti].v := FALSE;
      40E 11990              purge_map := TRUE;
      416 11991            IFEND;
      416 11992            mmv$pt_p^ [pti].m := FALSE;
      430 11993          = ioc$keypoint_io =
      430 11994            mmp$relink_page_frame (pfti, mmc$pq_free);
      44E 11995          ELSE
      44E 11996            mtp$error_stop ('MM - bad IO type lock_rma_list');
      46E 11997          CASEND;
      46E 11998 {!!!     ?IF mmc$debug_rma_list THEN
```

```
      46E 11999            IF (pfte_p^.queue_id <= mmc$pq_last_reassignable) AND (iotype <> ioc$keypoint_io) THEN
      480 12000              mtp$error_stop ('MM - Tried to ioc$write_page in AVAIL');
      4A0 12001            IFEND;
      4A0 12002 {!!!     ?IFEND;
      4A0 12003
      4A0 12004          list_p^ [list_i].rma := pfti * osv$page_size + page_offset;
      4A0 12005          page_count := page_count - 1;
      4A0 12006          IF page_count <= 0 THEN
      4C0 12007            list_p^ [list_i].length := ((buffer_descriptor.sva.offset + length - 1) MOD osv$page_size) -
      4C0 12008                 page_offset + 1;
      4C0 12009            jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
      4C0 12010            ijle_p^.active_io_page_count := ijle_p^.active_io_page_count + list_length;
      4C0 12011            ijle_p^.active_io_requests := ijle_p^.active_io_requests + 1;
*WARN*    12012            IF (pfte_p^.aste_p^.sfid.residence <> gfc$tr_job) OR
      542 12013                 ((iotype <> ioc$write_page) AND (iotype <> ioc$write_locked_page)) THEN
      542 12014              ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count + list_length;
      554 12015            IFEND;
      554 12016            IF list_i < list_length THEN
      55C 12017              list_p^ [list_i + 1].length := 0;
      566 12018            IFEND;
      566 12019            IF purge_map THEN
      56A 12020              mmp$purge_all_page_map;
      586 12021            IFEND;
      586 12022            RETURN; {<-----}
      588 12023          IFEND;
      588 12024          list_p^ [list_i].length := osv$page_size - page_offset;
      588 12025          sva.offset := sva.offset + osv$page_size;
      588 12026          page_offset := 0;
      588 12027          list_i := list_i + 1;
      588 12028        WHILEND /lp/;
      5C0 12029
      5C0 12030
      5C0 12031 {Control gets here only if a page frame is not assigned to a page that is being locked.  Unlock the pages (if
      5C0 12032 {any) that have already been locked.
      5C0 12033
      5C0 12034        IF list_i > 1 THEN
      5C4 12035          jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
      5C4 12036          ijle_p^.active_io_page_count := ijle_p^.active_io_page_count + list_i - 1;
      5C4 12037          ijle_p^.active_io_requests := ijle_p^.active_io_requests + 1;
      5C4 12038          IF (pfte_p^.aste_p^.sfid.residence <> gfc$tr_job) OR
      61E 12039               ((iotype <> ioc$write_page) AND (iotype <> ioc$write_locked_page)) THEN
      61E 12040            ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count + list_i - 1;
      61E 12041            mf_job_file := FALSE;
      636 12042          ELSE
      636 12043            mf_job_file := TRUE;
      63A 12044          IFEND;
      63A 12045          io_error := ioc$no_error;
      63A 12046          mmp$unlock_rma_list (ioc$no_io, list_p, list_i - 1, ioid, mf_job_file, io_error, status);
      67C 12047        IFEND;
      67C 12048        mtp$set_status_abnormal ('MM', mme$page_frame_not_assigned, status);
      696 12049
      696 12050
      696 12051      = mmc$bd_job_swapping_io =
      696 12052
      696 12053
      696 12054 {  Lock pages for job swapping io.
```



```
      696 12055
      696 12056        ijl_ordinal := buffer_descriptor.ijl_ordinal;
      696 12057        jmp$get_ijle_p (ijl_ordinal, ijle_p);
*     696 12058        page_count := list_length;
      696 12059        ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count + page_count;
      696 12060        ijle_p^.active_io_page_count := ijle_p^.active_io_page_count + page_count;
      696 12061        ijle_p^.active_io_requests := ijle_p^.active_io_requests + 1;
      696 12062
      696 12063        IF ijle_p^.swap_io_control.spd_index = 0 THEN
      6E8 12064          ijle_p^.swap_io_control.next_queue_id := SUCC (mmc$pq_job_fixed);
      6E8 12065          ijle_p^.swap_io_control.next_pfti := ijle_p^.job_page_queue_list [mmc$pq_job_fixed].link.bkw;
      6E8 12066          ijle_p^.swap_io_control.stop_pfti := ijle_p^.swap_io_control.swap_file_descriptor_pfti;
      6E8 12067          ijle_p^.swap_data.swapping_io_error := ioc$no_error;
      706 12068        IFEND;
      706 12069
      706 12070        jf_asid := ijle_p^.job_fixed_asid;
      706 12071
      706 12072        WHILE TRUE DO
      70E 12073          IF ijle_p^.swap_io_control.next_pfti = ijle_p^.swap_io_control.stop_pfti THEN
      71E 12074            IF ijle_p^.swap_io_control.next_queue_id = mmc$pq_job_fixed THEN
      72A 12075              ijle_p^.swap_io_control.next_pfti := ijle_p^.swap_io_control.swap_file_descriptor_pfti;
      736 12076            ELSEIF (ijle_p^.swap_io_control.next_queue_id = SUCC (mmc$pq_job_fixed)) AND
      742 12077                 (ijle_p^.swap_io_control.stop_pfti = 0) THEN
      742 12078              mtp$error_stop ('MM - error in locking swap file pages');
      762 12079            ELSE
      762 12080              ijle_p^.swap_io_control.stop_pfti := 0;
      762 12081              ijle_p^.swap_io_control.next_pfti := ijle_p^.job_page_queue_list
      784 12082                   [ijle_p^.swap_io_control.next_queue_id].link.bkw;
      784 12083            IFEND;
      784 12084            IF ijle_p^.swap_io_control.next_queue_id = UPPERVALUE (mmt$job_page_queue_index) THEN
      794 12085              ijle_p^.swap_io_control.next_queue_id := mmc$pq_job_fixed;
      7A0 12086            ELSE
      7A0 12087              ijle_p^.swap_io_control.next_queue_id := SUCC (ijle_p^.swap_io_control.next_queue_id);
      7A6 12088            IFEND;
      7A6 12089          IFEND;
      7A6 12090
      7A6 12091          WHILE ijle_p^.swap_io_control.next_pfti <> ijle_p^.swap_io_control.stop_pfti DO
      7C2 12092            pfte_p := ^mmv$pft_p^ [ijle_p^.swap_io_control.next_pfti];
      7C2 12093 {!!!       ?IF mmc$debug_rma_list THEN
      7C2 12094              IF ijl_ordinal <> pfte_p^.ijl_ordinal THEN
      7F2 12095                mtp$error_stop ('MM - mixed ijl ordinal in swap');
      812 12096              IFEND;
      812 12097            { The active_io_count check is being disabled until job shared pages CANNOT be in a working set
      812 12098            { at all after a second attach of a job shared file.
      812 12099            {   IF pfte_p^.active_io_count <> 0 THEN
      812 12100            {     mtp$error_stop ('MM - active IO on swapout');
      812 12101            {   IFEND;
      812 12102 {!!!       ?IFEND;
      812 12103            IF iotype = ioc$swap_out THEN
      81C 12104              spde_p := ^ijle_p^.sfd_p^.swapped_page_descriptors [ijle_p^.swap_io_control.spd_index];
      81C 12105              spde_p^.pft_entry := pfte_p^;
      83C 12106              spde_p^.page_table_entry := mmv$pt_p^ [pfte_p^.pti];
      83C 12107              spde_p^.ast_entry := pfte_p^.aste_p^;
      862 12108
      862 12109 { The entry_updated field in the swapped page descriptor is set to TRUE for job fixed pages, FALSE for
      862 12110 { all other pages.  Setting the field helps reset_swapped_job_mm_tables in swapper to differentiate
```

MMP$BUILD_LOCK_RMA_LIST - Build and lock pages defined by rma list

```
  862 12111 { between job fixed pages and other fixed pages when checking/changing ASIDs.
  862 12112
  862 12113             spde_p^.entry_updated := (spde_p^.pft_entry.sva.asid = jf_asid);
  862 12114
  862 12115             IF syv$perf_keypoints_enabled.memory_keypoints THEN
  87E 12116               #KEYPOINT (osk$performance, osk$m * ijle_p^.swap_io_control.next_pfti,
  88E 12117                     ptk$pfti_for_swapout);
  88E 12118             IFEND;
  88E 12119           IFEND;
  88E 12120
  88E 12121           list_p^ [list_i].rma := ijle_p^.swap_io_control.next_pfti * osv$page_size;
  88E 12122           list_p^ [list_i].length := osv$page_size;
  88E 12123
  88E 12124           ijle_p^.swap_io_control.spd_index := ijle_p^.swap_io_control.spd_index + 1;
  88E 12125           ijle_p^.swap_io_control.next_pfti := pfte_p^.link.bkw;
  88E 12126
  88E 12127           page_count := page_count - 1;
  88E 12128           IF page_count = 0 THEN
  8C6 12129             RETURN;
  8C8 12130           IFEND;
  8C8 12131           list_i := list_i + 1;
  8C8 12132         WHILEND;
  8E2 12133
  8E2 12134       WHILEND;
  8EA 12135     ELSE
  8EA 12136       mtp$error_stop ('MM - bad bufr desc in lock_rma_list');
  90A 12137     CASEND;
  90A 12138
  90A 12139     mmp$check_queues;
  91E 12140
  91E 12141   PROCEND mmp$build_lock_rma_list;
    0 12142
```


MMP$BUILD_LOCK_RMA_LIST_TAPE - Build and lock pages defined by tape request

```
    0 12145
    0 12146 {-------------------------------------------------------------------------------------------------------
    0 12147 { Purpose:
    0 12148 {   This procedure is a build_lock_rma_list routine that is customized for NOS/VE tape I/O.
    0 12149 {   This allows all the pages associated with a tape request to be locked in one call from
    0 12150 {   iom$tape_queue_manager_mtr.  This replaces the use of mmp$build_lock_rma_list, which
    0 12151 {   required up to 30 calls for write requests and 60 calls for read requests.
    0 12152 {
    0 12153 { Notes:
    0 12154 {   Although buffer_descriptor is not a parameter to this routine, the buffer_descriptor_type
    0 12155 {   is assumed to be mmc$bd_explicit_io.  Tape I/O does not use anything else.  If any changes
    0 12156 {   are made to mmp$build_lock_rma_list relating to mmc$bd_explicit_io, those changes may
    0 12157 {   also be needed here.
    0 12158 {
    0 12159 {   The only io_type values ever used for locking pages are ioc$explicit_read and
    0 12160 {   ioc$explicit_write.  When the pages are unlocked, io_type can be ioc$explicit_read_no_purge
    0 12161 {   if cache purge is not required.
    0 12162 {
    0 12163 {   The parameter tape_request_p points to the wired tape request for which pages need to
    0 12164 {   be locked and rma lists built.  The request is both an input and output parameter.
    0 12165 {   For writes and reads, the data buffers for the request are locked.  For reads the
    0 12166 {   first store transfer count buffer is also locked.  Since all transfer count buffers must
    0 12167 {   be on the same memory page, it is not necessary to lock more than one.
    0 12168 {
    0 12169 {   If page frames are not assigned to the entire range of addresses specified in the
    0 12170 {   request, no frames are locked and an error code will be returned.
    0 12171 {-------------------------------------------------------------------------------------------------------
    0 12172
    0 12173   PROCEDURE [XDCL] mmp$build_lock_rma_list_tape (
    0 12174         tape_request_p: ^iot$wired_tape_request;    {input/output
    0 12175       VAR status: syt$monitor_status];
    0 12176
    0 12177     VAR
    0 12178       command_index: iot$tape_command_index,
    0 12179       found: boolean,
    0 12180       hash_count: 1 .. 32,
    0 12181       ignore_status: syt$monitor_status,
    0 12182       ijle_p: ^jmt$initiated_job_list_entry,
    0 12183       ijl_ordinal: jmt$ijl_ordinal,
    0 12184       io_error: iot$io_error,
    0 12185       ioid: mmt$io_identifier,
    0 12186       length: ost$byte_count,
    0 12187       list_i: mmt$rma_list_index,
    0 12188       list_p: ^mmt$rma_list,
    0 12189       loop_count: 1 .. 2,
    0 12190       loop_count_index: 1 .. 2,
    0 12191       page_count: integer,
    0 12192       page_offset: 0 .. 65535,
    0 12193       pfte_p: ^mmt$page_frame_table_entry,
    0 12194       pfti: mmt$page_frame_index,
    0 12195       pti: integer,
    0 12196       pva: ^cell,
    0 12197       rma: integer,
    0 12198       sva: ost$system_virtual_address,
    0 12199       total_list_entries: mmt$rma_list_index;
    0 12200
```

MMP$BUILD_LOCK_RMA_LIST_TAPE - Build and lock pages defined by tape request

```
       0 12201      status.normal := TRUE;
       8 12202      total_list_entries := 1;
       8 12203      list_p := #LOC (tape_request_p^.wired_command_heap_p^.rma_list [1]);
      1C 12204      loop_count := 1;
      1C 12205      IF tape_request_p^.io_type = ioc$explicit_read THEN
      2C 12206        loop_count := 2;  { must lock data buffer(s) and store transfer count buffer
      30 12207      IFEND;
      30 12208
      30 12209      FOR loop_count_index := 1 TO loop_count DO
      36 12210
      36 12211      /lock_loop/
      36 12212        FOR command_index := 1 TO tape_request_p^.no_of_data_commands DO
      46 12213          list_i := 1;
      46 12214          IF tape_request_p^.io_type = ioc$explicit_read THEN
      54 12215            IF loop_count_index = 1 THEN {data buffer
      58 12216              length := tape_request_p^.max_input_count;
      58 12217              pva := tape_request_p^.wired_read_description_p^ [command_index].buffer_area;
      6C 12218            ELSE {store transfer count buffer
      6C 12219              length := 8;
      6C 12220              pva := tape_request_p^.wired_read_description_p^ [command_index].block_transfer_length;
      7A 12221            IFEND;
      7E 12222          ELSE {ioc$explicit_write
      7E 12223            length := tape_request_p^.wired_write_description_p^ [command_index].transfer_length;
      7E 12224            pva := tape_request_p^.wired_write_description_p^ [command_index].buffer_area;
      92 12225          IFEND;
      92 12226          mmp$xtask_pva_to_sva (pva, sva, status);
      AE 12227          IF NOT status.normal THEN
      B6 12228            IF total_list_entries > 1 THEN
      BA 12229              io_error := ioc$no_error;
      BA 12230              mmp$unlock_rma_list (ioc$no_io, list_p, total_list_entries - 1, ioid,
      F8 12231                    {MF_JOB_FILE} FALSE, io_error, ignore_status);
      F8 12232            IFEND;
      F8 12233            RETURN;
      FA 12234          IFEND;
      FA 12235
      FA 12236          page_offset := sva.offset MOD osv$page_size;
      FA 12237          page_count := ((page_offset + length - 1) DIV osv$page_size) + 1;
      FA 12238          IF page_count + total_list_entries - 1 > tape_request_p^.allocated_address_pair_count THEN
     130 12239            mtp$error_stop ('MM - tape lock rma list, list too small');
     14C 12240          IFEND;
     14C 12241
     14C 12242          REPEAT
     14C 12243            #HASH_SVA (sva, pti, hash_count, found);
     154 12244            IF NOT found THEN
     16A 12245
     16A 12246 { A page frame is not assigned to a page that is being locked or the page frame is not valid.
     16A 12247 { Unlock the pages (if any) that have already been locked and return the error
     16A 12248 { mme$page_frame_not_assigned to iom$tape_queue_manager_mtr, who will in turn return to job mode
     16A 12249 { where all the pages will be touched and the monitor request reissued.
     16A 12250
     16A 12251              IF total_list_entries > 1 THEN
     170 12252                IF list_i > 1 THEN
     174 12253                  jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
     174 12254                  ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count + list_i - 1;
     174 12255                  ijle_p^.active_io_page_count := ijle_p^.active_io_page_count + list_i - 1;
     174 12256                  ijle_p^.active_io_requests := ijle_p^.active_io_requests + 1;
```


MMP$BUILD_LOCK_RMA_LIST_TAPE - Build and lock pages defined by tape request

```
     1BE 12257                IFEND;
     1BE 12258                io_error := ioc$no_error;
     1BE 12259                mmp$unlock_rma_list (ioc$no_io, list_p, total_list_entries - 1, ioid,
     1FA 12260                      {MF_JOB_FILE} FALSE, io_error, status);
     1FA 12261                IF NOT status.normal THEN
     204 12262                  mtp$error_stop ('MM - lock tape rma list, unlock error');
     220 12263                IFEND;
     222 12264              IFEND;
     222 12265              mtp$set_status_abnormal ('MM', mme$page_frame_not_assigned, status);
     222 12266              RETURN; {<-----}
     230 12267            IFEND;
     230 12268
     230 12269            pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
     230 12270            pfte_p := ^mmv$pft_p^ [pfti];
     230 12271            pfte_p^.active_io_count := pfte_p^.active_io_count + 1;
     230 12272
     230 12273            ?IF mmc$debug_rma_list THEN
     230 12274              IF loop_count_index = 1 THEN
     27C 12275                IF total_list_entries = 1 THEN
     280 12276                  ijl_ordinal := pfte_p^.ijl_ordinal;
     288 12277                ELSEIF ijl_ordinal <> pfte_p^.ijl_ordinal THEN
     29C 12278                  mtp$error_stop ('MM - tape lock rma list, mixed ijl ordinal in lock');
     2BC 12279                IFEND;
     2C0 12280              IFEND;
     2C0 12281            ?IFEND;
     2C0 12282
     2C0 12283            list_p^ [total_list_entries].rma := pfti * osv$page_size + page_offset;
     2C0 12284            page_count := page_count - 1;
*WARN*    12285            IF page_count > 0 THEN
     2E0 12286              ?IF mmc$debug_rma_list THEN
     2E0 12287                IF loop_count_index = 2 THEN
     2E6 12288                  mtp$error_stop ('MM - tape lock rma list, transfer count buffer more than 1 page');
     304 12289                IFEND;
     304 12290              ?IFEND;
     304 12291              list_p^ [total_list_entries].length := osv$page_size - page_offset;
     304 12292              sva.offset := sva.offset + osv$page_size;
     304 12293              page_offset := 0;
*WARN*    12294              list_i := list_i + 1;
     338 12295            ELSE
     338 12296              list_p^ [total_list_entries].length := ((sva.offset + length - 1) MOD osv$page_size) -
     338 12297                    page_offset + 1;
     338 12298              jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
     338 12299              ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count + list_i;
*WARN*    12300              ijle_p^.active_io_page_count := ijle_p^.active_io_page_count + list_i;
     39E 12301            IFEND;
     39E 12302            total_list_entries := total_list_entries + 1;
     39E 12303          UNTIL page_count <= 0;
     3B0 12304
     3B0 12305          i#real_memory_address (^list_p^ [total_list_entries - list_i], rma);
     3D6 12306          IF tape_request_p^.io_type = ioc$explicit_read THEN
     3E2 12307            IF loop_count_index = 1 THEN
     3E6 12308              tape_request_p^.request.tape_command [command_index * 2].address := rma;
     3E6 12309              tape_request_p^.request.tape_command [command_index * 2].length := list_i * 8;
     402 12310            ELSE
     402 12311              tape_request_p^.request.tape_command [command_index * 2 + 1].address :=
     402 12312                    list_p^ [total_list_entries - 1].rma;
```

MMP$BUILD_LOCK_RMA_LIST_TAPE - Build and lock pages defined by tape request

```
402 12313              EXIT /lock_loop/;
41C 12314            IFEND;
420 12315          ELSE { ioc$explicit_write
420 12316            tape_request_p^.request.tape_command [command_index * 2 + 1].address := rma;
420 12317            tape_request_p^.request.tape_command [command_index * 2 + 1].length := list_i * 8;
438 12318          IFEND;
438 12319        FOREND /lock_loop/;
43C 12320
43C 12321 { Increment active_io_requests in the ijle entry that was used in the preceding trip thru
43C 12322 { the above FOR loop.  Since all pages must be in the same segment, the ijle pointer is
43C 12323 { the correct one.
43C 12324
43C 12325        ijle_p^.active_io_requests := ijle_p^.active_io_requests + 1;
43C 12326
43C 12327      FOREND;
44E 12328
44E 12329 { The following check is necessary in the case of error recovery.  In that situation,
44E 12330 { the number of blocks being retried may be less than the original and therefore, the
44E 12331 { total_list_entries will be less than the amount allocated.  For performance reasons,
44E 12332 { a new allocated_address_pair_count is not re-calculated before retrying the IO.
44E 12333
44E 12334      IF total_list_entries - 1 < tape_request_p^.allocated_address_pair_count THEN
45E 12335        list_p^ [total_list_entries].length := 0;
46A 12336      IFEND;
46A 12337
46A 12338      tape_request_p^.list_p := list_p;
46A 12339      tape_request_p^.address_pair_count := total_list_entries - 1;
46A 12340
46A 12341    PROCEND mmp$build_lock_rma_list_tape;
  0 12342
```


MMP$REMOVE_JWS_TO_SHARED_PAGES

```
  0 12345 {-----------------------------------------------------------------------------
  0 12346 { If pages of a segment were being kept in a jws, but are now going to be kept
  0 12347 { in the global queue, it is necessary to remove all pages of the segment from
  0 12348 { the jws.  This procedure is called to remove the pages when a job shared file
  0 12349 { segment is opened and more than one user has the file attached.
  0 12350 { The swap state of the job the pages are being removed from (which is not the
  0 12351 { job that issued the monitor request) must be considered.
  0 12352 { If the job is executing or in a "safe" swap status, page will be removed.  If
  0 12353 { swapout I/O is active, the monitor request will be reissued and a task switch
  0 12354 { forced (the request will have to wait until the I/O has completed).  If the
  0 12355 { job is swapped completely out, a delayed swapin bit will be set so that the
  0 12356 { pages will be removed when the job swaps back in.
  0 12357 { NOTE:  This request is usually issued because a second user has attached a
  0 12358 { read only file and the file is now being shared.  The request is also used
  0 12359 { to remove pages from the JWS when the flush_pages request for a detach_file
  0 12360 { fails.  When that happens the FDE for the file is turned into a global_shared
  0 12361 { file.  The AST entry and all pages must reflect that the file is shared.
  0 12362 { ALSO NOTE:  In the flush-detach fail case, the job which has the pages in
  0 12363 { its working set cannot be swapped out, because it is issuing the monitor
  0 12364 { request.
  0 12365 {-----------------------------------------------------------------------------
  0 12366
  0 12367   PROCEDURE mmp$remove_jws_to_shared_pages
  0 12368     (     fde_p: gft$locked_file_desc_entry_p;
  0 12369           cst_p: ^ost$cpu_state_table;
  0 12370      VAR rb: mmt$rb_ring1_segment_request);
  0 12371
  0 12372     VAR
  0 12373       asid: ost$asid,
  0 12374       aste_p: ^mmt$active_segment_table_entry,
  0 12375       count_removed: integer,
  0 12376       dsw_job_shared_asid_changed: [STATIC] jmt$delayed_swapin_work := [jmc$dsw_job_shared_asid_changed],
  0 12377       ijle_p: ^jmt$initiated_job_list_entry,
  0 12378       i: integer,
  0 12379       inhibit_io: boolean,
  0 12380       jws_ijl_ordinal: jmt$ijl_ordinal,
  0 12381       next_pfti: mmt$page_frame_index,
  0 12382       pfti: mmt$page_frame_index,
  0 12383       pfte_p: ^mmt$page_frame_table_entry,
  0 12384       queue_id: mmt$page_frame_queue_id,
  0 12385       sdte_p: ^mmt$segment_descriptor,
  0 12386       system_ijle_p: ^jmt$initiated_job_list_entry;
  0 12387
  0 12388     rb.status.normal := TRUE;
  4 12389     aste_p := ^mmv$ast_p^ [fde_p^.asti];
  4 12390
  4 12391 { The file may already be shared.  In that case nothing needs to be done.
  4 12392
  4 12393     IF (aste_p^.queue_id > mmc$pq_shared_last)  THEN
 2A 12394       jws_ijl_ordinal := aste_p^.ijl_ordinal;
 2A 12395       IF NOT (jmp$ijl_block_valid (jws_ijl_ordinal)) THEN
 58 12396         RETURN;
 5A 12397       IFEND;
 5A 12398
 5A 12399       jmp$get_ijle_p (jws_ijl_ordinal, ijle_p);
 5A 12400
```

```
      5A 12401 { If swapout I/O is active, cause the task to cycle until the swapout I/O has completed.
      5A 12402 { Do not change any AST fields now if the request has to be reissued.
      5A 12403
      5A 12404         IF (ijle_p^.swap_status >= jmc$iss_initiate_swapout_io) AND (ijle_p^.swap_status <=
      8A 12405               jmc$iss_swapout_io_complete) THEN
      8A 12406
      8A 12407           tmp$reissue_monitor_request;
      92 12408           tmp$cause_task_switch;
      9C 12409
      9C 12410         ELSE
      9C 12411
      9C 12412 { The pages of this segment were being kept in a JWS queue but are now going to be kept in
      9C 12413 { one of the shared queues, all pages must be removed from the JWS.  The procedure called
      9C 12414 { will remove the pages immediately if the job is addressable; if the job is swapped out,
      9C 12415 { a delayed swapin bit will be set in the job's ijl entry and the pages will be removed
      9C 12416 { when the job swaps in.  The segment must be changed to reflect that it is now being
      9C 12417 { shared before the call to the remove procedure; the ijlo passed into the remove procedure
      9C 12418 { must be the ijlo of the job that we need to remove the pages from.
      9C 12419 { First, determine which Shared Queue will be used.  NOTE:  A segment number of zero indicates
      9C 12420 { that this request was issued because a flush pages on detach_file failed.  In that case, do
      9C 12421 { not store the asid/asti in the segment table; use shared_other queue.
      9C 12422
      9C 12423           IF rb.server_file THEN
      A4 12424             aste_p^.queue_id := mmc$pq_shared_file_server;
      AE 12425           ELSEIF rb.segment_number <> 0 THEN
      B6 12426             sdte_p := mmp$get_sdt_entry_p (cst_p^.xcb_p, rb.segment_number);
      B6 12427             sdte_p^.asti := fde_p^.asti;
      B6 12428             mmp$asid (fde_p^.asti, asid);
      FC 12429             sdte_p^.ste.aSid := asid;
      FC 12430             aste_p^.queue_id := mmp$determine_shared_queue_id (fde_p, sdte_p);
     12O 12431           ELSE
     12O 12432             aste_p^.queue_id := mmc$pq_shared_other;
     126 12433           IFEND;
     126 12434
     126 12435           aste_p^.ijl_ordinal := jmv$system_ijl_ordinal;
     126 12436           jmp$get_ijle_p (jmv$system_ijl_ordinal, system_ijle_p);
     126 12437
     126 12438 { Scan the pages in memory belonging to the segment.  Store pfti's of working set pages so they
     126 12439 { can be removed.  Adjust I/O counts for pages in the available modified queue.
     126 12440
     126 12441           count_removed := 0;
     126 12442           mmp$reset_store_pfti;
     126 12443           pfti := aste_p^.pft_link.fwd;
     126 12444           WHILE pfti <> 0 DO
     17E 12445             pfte_p := ^mmv$pft_p^ [pfti];
     17E 12446             IF pfte_p^.queue_id = mmc$pq_job_working_set THEN
     1AC 12447               mmp$store_pfti (pfti);
     1CA 12448             ELSEIF pfte_p^.queue_id = mmc$pq_avail_modified THEN
     1DO 12449               ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count - pfte_p^.active_io_count;
     1DO 12450               ijle_p^.active_io_page_count := ijle_p^.active_io_page_count - pfte_p^.active_io_count;
     1DO 12451               system_ijle_p^.inhibit_swap_count := system_ijle_p^.inhibit_swap_count + pfte_p^.
     1DO 12452                     active_io_count;
     1DO 12453               system_ijle_p^.active_io_page_count := system_ijle_p^.active_io_page_count + pfte_p^.
     1DO 12454                     active_io_count;
     1DO 12455               pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
     20C 12456             IFEND;
```



```
     20C 12457             pfti := mmv$pft_p^ [pfti].segment_link.fwd;
     20C 12458           WHILEND;
     236 12459
     236 12460           IF (ijle_p^.swap_status <= jmc$iss_swapped_io_cannot_init) OR (ijle_p^.swap_status =
     24C 12461                 jmc$iss_swapped_io_complete) THEN
     24C 12462
     24C 12463             mmp$fetch_pfti_array_size (count_removed);
     24C 12464             IF count_removed > 0 THEN
     264 12465               mmp$reset_find_next_pfti (pfti);
     2AE 12466               WHILE pfti <> 0 DO
     2BE 12467                 pfte_p := ^mmv$pft_p^ [pfti];
     2BE 12468                 mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
     2F8 12469                 ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count - pfte_p^.active_io_count;
     2F8 12470                 ijle_p^.active_io_page_count := ijle_p^.active_io_page_count - pfte_p^.active_io_count;
     2F8 12471                 system_ijle_p^.inhibit_swap_count := system_ijle_p^.inhibit_swap_count + pfte_p^.
     2F8 12472                       active_io_count;
     2F8 12473                 system_ijle_p^.active_io_page_count := system_ijle_p^.active_io_page_count + pfte_p^.
     2F8 12474                       active_io_count;
     2F8 12475                 pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
     2F8 12476                 mmp$find_next_pfti (pfti);
     380 12477               WHILEND;
     38C 12478
     38C 12479               IF ((ijle_p^.swap_status >= jmc$iss_job_idle_tasks_complete) AND (ijle_p^.swap_status <=
     3BO 12480                     jmc$iss_swapped_io_cannot_init)) OR (ijle_p^.swap_status = jmc$iss_swapped_io_complete) THEN
     3BO 12481                 jsp$recalculate_swapped_pages (ijle_p, count_removed);
     3CC 12482               IFEND;
     3CC 12483             IFEND;
     3CE 12484           ELSE
     3CE 12485             ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work + dsw_job_shared_asid_changed;
     3EO 12486           IFEND;
     3EO 12487         IFEND;
     3EO 12488       IFEND;
     3EO 12489
     3EO 12490     PROCEND mmp$remove_jws_to_shared_pages;
       0 12491
       0 12492
```

[XDCL] mmp$remove_swapped_shared_pages

```
        0 12494 {-------------------------------------------------------------------------------
        0 12495 { This procedure is called on swapin to remove job shared pages from the working
        0 12496 { set of a job that is swapping in.  The job was swapped out when the when a job
        0 12497 { shared file was attached by a second job, causing a working-set-to-shared
        0 12498 { transition.  At the time of the transition a delayed_swapin_work indicator
        0 12499 { was set for the job.  The job's entire working set must be scanned for pages
        0 12500 { that should no be in the shared queue.
        0 12501 {-------------------------------------------------------------------------------
        0 12502
        0 12503   PROCEDURE [XDCL] mmp$remove_swapped_shared_pages
        0 12504     (     ijle_p: ^jmt$initiated_job_list_entry);
        0 12505
        0 12506     VAR
        0 12507       next_pfti: mmt$page_frame_index,
        0 12508       pfti: mmt$page_frame_index,
        0 12509       pfte_p: ^mmt$page_frame_table_entry,
        0 12510       system_ijle_p: ^jmt$initiated_job_list_entry;
        0 12511
        0 12512       { Scan the job_working_set queue; any pages for which the ast.queue_id is the global
        0 12513       { queue need to be removed and put in the available queue.
        0 12514
        0 12515     jmp$get_ijle_p (jmv$system_ijl_ordinal, system_ijle_p);
        4 12516
        4 12517     pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.bkw;
        4 12518     WHILE pfti <> 0 DO
       3A 12519       pfte_p := ^mmv$pft_p^ [pfti];
       3A 12520       next_pfti := pfte_p^.link.bkw;
       3A 12521       IF (pfte_p^.aste_p^.queue_id >= mmc$pq_shared_first) AND
       72 12522            (pfte_p^.aste_p^.queue_id <= mmc$pq_shared_last)  THEN
       72 12523         mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
       86 12524         ijle_p^.inhibit_swap_count := ijle_p^.inhibit_swap_count - pfte_p^.active_io_count;
       86 12525         ijle_p^.active_io_page_count := ijle_p^.active_io_page_count - pfte_p^.active_io_count;
       86 12526         system_ijle_p^.inhibit_swap_count := system_ijle_p^.inhibit_swap_count + pfte_p^.
       86 12527               active_io_count;
       86 12528         system_ijle_p^.active_io_page_count := system_ijle_p^.active_io_page_count + pfte_p^.
       86 12529               active_io_count;
       86 12530         pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
       BE 12531       IFEND;
       BE 12532       pfti := next_pfti;
       BE 12533     WHILEND;
       C6 12534
       C6 12535   PROCEND mmp$remove_swapped_shared_pages;
        0 12536
```


MMP$REMOVE_DETACHED_PAGES

```
        0 12538 {-------------------------------------------------------------------------
        0 12539 { This procedure removes job working set pages of a file being detached
        0 12540 { from the working set of the job doing the detach.
        0 12541 {-------------------------------------------------------------------------
        0 12542
        0 12543   PROCEDURE mmp$remove_detached_pages
        0 12544     (     sva: ost$system_virtual_address;
        0 12545           aste_p: ^mmt$active_segment_table_entry;
        0 12546           ijl_ordinal: jmt$ijl_ordinal);
        0 12547
        0 12548     VAR
        0 12549       pfti: mmt$page_frame_index;
        0 12550
        0 12551     mmp$initialize_find_next_pfti (sva, 7ffffff0(16), include_partial_pages, psc_nominal_queue, aste_p,
       32 12552           pfti);
       32 12553     WHILE pfti <> 0 DO
       3A 12554       IF (mmv$pft_p^ [pfti].queue_id > mmc$pq_first_valid_in_pt) THEN
       64 12555         IF mmv$pft_p^ [pfti].locked_page = mmc$lp_page_in_lock THEN
       6C 12556           mmp$delete_pt_entry (pfti, TRUE);
       82 12557           mmp$relink_page_frame (pfti, mmc$pq_free);
       9E 12558         ELSE
       9E 12559           mmp$remove_page_from_job (pfti);
       B0 12560         IFEND;
       B0 12561         mmv$pft_p^ [pfti].locked_page := mmc$lp_not_locked;
       CE 12562       IFEND;
       CE 12563       mmp$find_next_pfti (pfti);
      122 12564     WHILEND;
      12E 12565
      12E 12566   PROCEND mmp$remove_detached_pages;
        0 12567
```

MMP$REMOVE_PAGES_WORKING_SET

```
        0 12570 {------------------------------------------------------------------------
        0 12571 {This procedure removes pages from a job working set or shared working set
        0 12572 {All pages totally or partially contained between SVA to SVA + LENGTH are
        0 12573 {moved to the AVAILABLE or AVAILABLE-MODIFIED queue.
        0 12574 {------------------------------------------------------------------------
        0 12575
        0 12576   PROCEDURE [XDCL] mmp$remove_pages_working_set
        0 12577     (     sva: ost$system_virtual_address;
        0 12578           length: ost$segment_length;
        0 12579           aste_p: ^mmt$active_segment_table_entry;
        0 12580      VAR rcount: integer);
        0 12581
        0 12582     VAR
        0 12583       cst_p: ^ost$cpu_state_table,
        0 12584       mcount: integer,
        0 12585       pfti: mmt$page_frame_index;
        0 12586
        0 12587 { #### CHECK THIS ALGORITHM CAREFULLY.  Consider restricted attach/detach, job shared files, restricted
        0 12588 { #### attach/detach of global shared files.
        0 12589
        0 12590     mtp$cst_p (cst_p);
       14 12591
       14 12592 { Do not remove pages from the shared queue, nor from another job's working set.
       14 12593
       14 12594     IF (aste_p^.queue_id <> mmc$pq_job_working_set) OR (aste_p^.ijl_ordinal <> cst_p^.ijl_ordinal) THEN
       48 12595       RETURN;
       4A 12596     IFEND;
       4A 12597
       4A 12598     mmp$initialize_find_next_pfti (sva, length, include_partial_pages, psc_nominal_queue, aste_p, pfti);
       80 12599     IF pfti = 0 THEN
       88 12600       rcount := 0;
       88 12601       RETURN
       92 12602     IFEND;
       92 12603
       92 12604 { Delete locked pages from the array of pages to be removed.
       92 12605 { ### There should be a better way of doing this than scanning the whole array.
       92 12606
       92 12607     WHILE pfti <> 0 DO
       9A 12608
       9A 12609       IF mmv$pft_p^ [pfti].locked_page <> mmc$lp_not_locked THEN
       C0 12610           mmp$delete_last_pfti_from_array;
       D4 12611       IFEND;
       D4 12612       mmp$find_next_pfti (pfti);
      128 12613     WHILEND;
      134 12614
      134 12615     mmp$remove_pages_from_jws (mmc$pq_avail_modified, cst_p^.ijle_p, mcount, rcount);
      162 12616
      162 12617   PROCEND mmp$remove_pages_working_set;
        0 12618
```


MMP$MM_FREE_PAGES

```
        0 12621 {------------------------------------------------------------------------
        0 12622 {This procedure frees (moves the page frames to the free page queue) all
        0 12623 {pages TOTALLY contained in the range SVA to SVA + LENGTH.
        0 12624 {NOTE that modified pages are NOT written to disk.
        0 12625 {If ASID is to be freed, map is not purged prior to deleting page frames.
        0 12626 {------------------------------------------------------------------------
        0 12627
        0 12628   PROCEDURE [XDCL] mmp$mm_free_pages
        0 12629     (     sva: ost$system_virtual_address;
        0 12630           length: ost$segment_length;
        0 12631           aste_p: ^mmt$active_segment_table_entry;
        0 12632           free_asid: boolean;
        0 12633      VAR count: integer);
        0 12634
        0 12635     VAR
        0 12636       contiguous_pages: integer,
        0 12637       first_pfti: mmt$page_frame_index,
        0 12638       ijl_p: ^jmt$initiated_job_list_entry,
        0 12639       pfti: mmt$page_frame_index;
        0 12640
        0 12641     count := 0;
        4 12642
        4 12643     mmp$initialize_find_next_pfti (sva, length, exclude_partial_pages, psc_all, aste_p, pfti);
       46 12644
       46 12645     IF NOT free_asid THEN
       4E 12646       WHILE pfti <> 0 DO
       56 12647         mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := FALSE;
       56 12648         mmp$find_next_pfti (pfti);
       E0 12649       WHILEND;
       EC 12650       mmp$purge_all_cache_map;
      118 12651       mmp$reset_find_next_pfti (pfti);
      176 12652     IFEND;
      176 12653
      176 12654     WHILE pfti <> 0 DO
      182 12655       mmp$delete_pt_entry (pfti, TRUE);
      19A 12656       mmp$relink_page_frame (pfti, mmc$pq_free);
      1B2 12657       count := count + 1;
      1B2 12658       mmp$find_next_pfti (pfti);
      208 12659     WHILEND;
      214 12660
      214 12661     IF aste_p^.queue_id = mmc$pq_job_fixed THEN
      224 12662       jmp$get_ijle_p (aste_p^.ijl_ordinal, ijl_p);
      224 12663       contiguous_pages := 0;
      224 12664       first_pfti := ijl_p^.job_page_queue_list [mmc$pq_job_fixed].link.bkw;
      224 12665       IF (mmv$pft_p^ [first_pfti].sva.offset <> 0) THEN
      278 12666         WHILE mmv$pft_p^ [first_pfti].sva.offset <> 0 DO
      278 12667           contiguous_pages := contiguous_pages + 1;
      278 12668           first_pfti := mmv$pft_p^ [first_pfti].link.bkw;
      278 12669         WHILEND;
      2B2 12670       IFEND;
      2B2 12671       ijl_p^.job_fixed_contiguous_pages := contiguous_pages;
      2B2 12672       mmv$total_contig_pages_assigned := mmv$total_contig_pages_assigned - contiguous_pages;
      2C8 12673     IFEND;
      2C8 12674     IF free_asid THEN
      2D0 12675       mmp$free_asid (sva.asid, aste_p);
      2EC 12676     IFEND;
```

MMP$MM_FREE_PAGES

```
      2EC 12677
      2EC 12678    PROCEND mmp$mm_free_pages;
        0 12679
```


MMP$MM_WRITE_MODIFIED_PAGES

```
        0 12682 {-----------------------------------------------------------------------
        0 12683 {Name:
        0 12684 {  mmp$mm_write_modified_pages
        0 12685 {Purpose:
        0 12686 {  This procedure writes all modified pages within a specified SVA range
        0 12687 {  to the backing file for the segment.  All pages totally or partially
        0 12688 {  contained in the range of SVA to SVA + LENGTH are written.
        0 12689 {
        0 12690 {  If init_new_io is FALSE then the monitor request has been reissued
        0 12691 {  and only status will be returned.   Job mode sets init_new_io to TRUE
        0 12692 {  in the RB and mmp$process_wmp_status or mmp$mtr_write will set it
        0 12693 {  to FALSE if the wait option is selected.
        0 12694 {-----------------------------------------------------------------------
        0 12695
        0 12696
        0 12697    PROCEDURE [XDCL] mmp$mm_write_modified_pages
        0 12698      (     sva: ost$system_virtual_address;
        0 12699            length: ost$segment_length;
        0 12700            fde_p: gft$locked_file_desc_entry_p;
        0 12701            aste_p: ^mmt$active_segment_table_entry;
        0 12702            iotype: iot$io_function;
        0 12703            init_new_io: boolean;
        0 12704            remove_page: boolean;
        0 12705            io_id: mmt$io_identifier;
        0 12706       VAR io_count: mmt$active_io_count;
        0 12707       VAR io_already_active: boolean;
        0 12708       VAR last_written_pfti: mmt$page_frame_index;
        0 12709       VAR wmp_status: mmt$write_modified_pages_status);
        0 12710
        0 12711      VAR
        0 12712        cst_p: ^ost$cpu_state_table,
        0 12713        write_status: mmt$write_page_to_disk_status,
        0 12714        pfti: mmt$page_frame_index;
        0 12715
        0 12716      io_count := 0;
        4 12717      io_already_active := FALSE;
        4 12718      wmp_status := mmc$wmp_io_complete;
        4 12719      last_written_pfti := 0;
        4 12720
        4 12721      mtp$cst_p (cst_p);
       38 12722      mmp$initialize_find_next_pfti (sva, length, include_partial_pages, psc_all_except_avail, aste_p, pfti);
       7E 12723
       7E 12724      IF init_new_io THEN
       86 12725      /write_loop/
       86 12726        WHILE pfti <> 0 DO
       8E 12727          IF mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m THEN
       CC 12728          /write_to_disk/
       CC 12729            BEGIN
       CC 12730            mmp$write_page_to_disk (fde_p, pfti, iotype, io_id, mmv$multi_page_write, write_status);
      108 12731            IF write_status = ws_ok THEN
      110 12732              last_written_pfti := pfti;
      110 12733
      110 12734              IF io_id.specified THEN
      120 12735                io_count := io_count + 1;
      12A 12736              IFEND;
      12E 12737
```

MMP$MM_WRITE_MODIFIED_PAGES

```
12E 12738            ELSEIF write_status = ws_disk_flaws THEN
132 12739              wmp_status := mmc$wmp_io_errors;
13C 12740            ELSEIF write_status = ws_volume_unavailable THEN
142 12741              wmp_status := mmc$wmp_volume_unavailable;
14A 12742            ELSEIF write_status = ws_server_terminated THEN
150 12743              wmp_status := mmc$wmp_server_terminated;
158 12744            ELSE
158 12745              wmp_status := mmc$wmp_io_initiation_reject;
158 12746              EXIT /write_loop/;
160 12747            IFEND;
160 12748
160 12749            END /write_to_disk/;
164 12750          ELSEIF (mmv$pft_p^ [pfti].active_io_count <> 0) AND (last_written_pfti = 0) THEN
174 12751            last_written_pfti := pfti;
174 12752            IF io_id.specified THEN
180 12753              io_already_active := TRUE;
184 12754            IFEND;
184 12755          IFEND;
184 12756
184 12757          { REMOVE_PAGE is true only if the request code is mmc$sr1_detach_file.
184 12758          { When a permanent file is detached, if the pages are being kept in a job's working set
184 12759          { all those pages must be removed.
184 12760 { ###!!    The ijl ordinal check is needed to prevent a "restricted" attach/detach (access mode = none)
184 12761          { from removing pages from the working set of the job that really has the file attached.
184 12762          { (There is a timing problem--if the real detach and a new shared attach occurrs before the
184 12763          { restricted detach, the queue status will incorrectly remain working set for a shared file.)
184 12764          { When perm files cleans up the restricted attach, the ijl ordinal check can probably be removed.
184 12765
184 12766          IF (remove_page) AND (mmv$pft_p^ [pfti].queue_id > mmc$pq_first_valid_in_pt) AND
1F6 12767                NOT (mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m) AND (mmv$shared_pages_in_jws) AND
1F6 12768                (aste_p^.queue_id = mmc$pq_job_working_set) AND (mmv$pft_p^ [pfti].ijl_ordinal =
1F6 12769                cst_p^.ijl_ordinal) THEN
1F6 12770            IF mmv$pft_p^ [pfti].locked_page = mmc$lp_page_in_lock THEN
1FE 12771              mmp$delete_pt_entry (pfti, TRUE);
212 12772              mmp$relink_page_frame (pfti, mmc$pq_free);
22E 12773            ELSE
22E 12774              mmp$remove_page_from_job (pfti);
23E 12775            IFEND;
23E 12776            mmv$pft_p^ [pfti].locked_page := mmc$lp_not_locked;
25C 12777          IFEND;
25C 12778          mmp$find_next_pfti (pfti);
2B0 12779        WHILEND /write_loop/;
2C0 12780      ELSE
2C0 12781      /status_loop/
2C0 12782        WHILE pfti <> 0 DO
2D4 12783          IF mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m THEN
308 12784            IF (mmv$pft_p^ [pfti].queue_id = mmc$pq_job_io_error) OR
31C 12785                (mmv$pft_p^ [pfti].queue_id = mmc$pq_shared_io_error) THEN
31C 12786              IF (mmv$pft_p^ [pfti].io_error <> ioc$unrecovered_error_unit_down) AND
346 12787                  (mmv$pft_p^ [pfti].io_error <> ioc$unit_down_on_init) THEN
346 12788                wmp_status := mmc$wmp_io_errors;
34E 12789              ELSE
34E 12790                wmp_status := mmc$wmp_volume_unavailable;
354 12791              IFEND;
354 12792            IFEND;
358 12793
```


MMP$MM_WRITE_MODIFIED_PAGES

```
358 12794          ELSEIF (mmv$pft_p^ [pfti].active_io_count <> 0)  THEN
360 12795            last_written_pfti := pfti;
360 12796            IF io_id.specified THEN
36C 12797              io_already_active := TRUE;
370 12798            IFEND;
370 12799          IFEND;
370 12800
370 12801          mmp$find_next_pfti (pfti);
3BC 12802        WHILEND /status_loop/;
3C8 12803      IFEND;
3C8 12804
3C8 12805      IF io_id.specified THEN
3D0 12806        IF (io_count > 0) AND (wmp_status = mmc$wmp_io_complete) THEN
3E0 12807          wmp_status := mmc$wmp_io_active;
3E6 12808        IFEND;
3E8 12809      ELSE
3E8 12810        IF (last_written_pfti <> 0) AND (wmp_status <> mmc$wmp_volume_unavailable) AND
3FE 12811            (wmp_status <> mmc$wmp_io_initiation_reject) THEN
3FE 12812          wmp_status := mmc$wmp_io_active;
404 12813        IFEND;
404 12814      IFEND;
404 12815
404 12816    PROCEND mmp$mm_write_modified_pages;
  0 12817
```

```
        0 12820
        0 12821 {-------------------------------------------------------------------------------------
        0 12822 {Name:
        0 12823 {  mmp$process_wmp_status
        0 12824 {Purpose:
        0 12825 {  This procedure processes the status returned by mmp$mm_write_modified_pages.  If
        0 12826 {  IO inititation was not completed the monitor request will be reissued.  If the
        0 12827 {  wait option was selected, init_new_io in the RB will be set to FALSE and the
        0 12828 {  monitor request reissued.
        0 12829 {
        0 12830 {  Rb_init_new_io and rb_status are input/output parameters and should not be initialized.
        0 12831 {
        0 12832 {-------------------------------------------------------------------------------------
        0 12833
        0 12834
        0 12835   PROCEDURE [XDCL] mmp$process_wmp_status (wmp_status: mmt$write_modified_pages_status;
        0 12836         last_written_pfti: mmt$page_frame_index;
        0 12837         rb_wait: ost$wait;
        0 12838         VAR rb_init_new_io: boolean;
        0 12839         VAR rb_status: syt$monitor_status);
        0 12840
        0 12841     VAR cst_p: ^ost$cpu_state_table;
        0 12842
        0 12843     mtp$cst_p (cst_p);
       14 12844
       14 12845     CASE wmp_status OF
       52 12846     = mmc$wmp_io_initiation_reject =
       52 12847       IF last_written_pfti <> 0 THEN
       5A 12848         IF mmv$pft_p^[last_written_pfti].active_io_count = 0 THEN
       7C 12849           mtp$error_stop('MM - WMP tried to queue and no IO');
       9C 12850         IFEND;
       9C 12851         tmp$queue_task(cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^[last_written_pfti].task_queue);
       DE 12852         tmp$reissue_monitor_request;
       EA 12853       ELSE
       EA 12854         tmp$reissue_monitor_request;
       F2 12855         tmp$cause_task_switch;
       FA 12856       IFEND;
       FA 12857
       FA 12858     = mmc$wmp_io_complete =
       FC 12859
       FC 12860     = mmc$wmp_io_active =
       FC 12861       IF rb_wait = osc$wait THEN
      104 12862         IF mmv$pft_p^[last_written_pfti].active_io_count = 0 THEN
      12A 12863           mtp$error_stop('MM - WMP tried to queue and no IO');
      14A 12864         IFEND;
      14A 12865         tmp$queue_task(cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^[last_written_pfti].task_queue);
      18C 12866         rb_init_new_io := FALSE;
      18C 12867         tmp$reissue_monitor_request;
      19E 12868       IFEND;
      1A0 12869
      1A0 12870     = mmc$wmp_volume_unavailable =
      1A0 12871       mtp$set_status_abnormal ('MM', mme$volume_unavailable, rb_status);
      1B8 12872
      1B8 12873     = mmc$wmp_io_errors =
      1B8 12874       mtp$set_status_abnormal ('MM', mme$io_write_error, rb_status);
      1D0 12875
```



```
      1D0 12876     = mmc$wmp_server_terminated =
      1D0 12877       mtp$set_status_abnormal ('DF', dfe$server_has_terminated, rb_status);
      1E6 12878
      1E6 12879     CASEND;
      1E6 12880
      1E6 12881   PROCEND mmp$process_wmp_status;
        0 12882
        0 12883
```

```
      0 12886
      0 12887 { Purpose:
      0 12888 {   This procedure conditionally frees pages of a segment in the address range of SVA to
      0 12889 {   SVA + LENGTH - 1.  The modified bits are cleared to prevent writing the pages to disk.
      0 12890 {   The pages will remain in the job's working set though in case they are referenced again
      0 12891 {   soon.  Normal aging will move the pages to the free queue when they are no longer being
      0 12892 {   referenced.
      0 12893
      0 12894   PROCEDURE mmp$mm_conditional_free (sva: ost$system_virtual_address;
      0 12895         length: ost$segment_length;
      0 12896         aste_p: ^mmt$active_segment_table_entry);
      0 12897
      0 12898     VAR
      0 12899       pfte_p: ^mmt$page_frame_table_entry,
      0 12900       pfti: mmt$page_frame_index,
      0 12901       pte_p: ^ost$page_table_entry,
      0 12902       purge_all_page_maps: boolean;
      0 12903
      0 12904     purge_all_page_maps := FALSE;
      4 12905
      4 12906     mmp$initialize_find_next_pfti (sva, length, exclude_partial_pages, psc_all_except_avail, aste_p, pfti);
     32 12907     WHILE pfti <> 0 DO
     3C 12908       pfte_p := ^mmv$pft_p^ [pfti];
     3C 12909       pte_p := ^mmv$pt_p^ [pfte_p^.pti];
     3C 12910       IF (pfte_p^.queue_id = mmc$pq_avail_modified) AND (pfte_p^.active_io_count = 0) THEN
     80 12911         pte_p^.m := FALSE;
     80 12912
     80 12913 { Whenever the modified bit is cleared on a page in the available_modified queue that does not have
     80 12914 { I/O active, the SOON count must be incremented.  When relink_page_frame takes an UNMODIFIED page out
     80 12915 { of the available_modified queue, it assumes I/O has been done and will decrement the soon count.
     80 12916
     80 12917         mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + 1;
     80 12918         mmp$relink_page_frame (pfti, mmc$pq_avail);
     AC 12919       ELSEIF pte_p^.v THEN
     B6 12920
     B6 12921 { Pages must be removed from the io error queues before the modified bit is cleared.
     B6 12922
     B6 12923         IF (pfte_p^.queue_id = mmc$pq_shared_io_error) OR (pfte_p^.queue_id = mmc$pq_job_io_error) THEN
     CA 12924           mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id];
     EC 12925         IFEND;
     EC 12926         pte_p^.m := FALSE;
     EC 12927         IF pte_p^.u THEN
     FE 12928           pte_p^.u := FALSE;
     FE 12929           pfte_p^.age := 0;
     FE 12930           pfte_p^.cyclic_age := 0;
    10C 12931         IFEND;
    10C 12932         purge_all_page_maps := TRUE;
    112 12933       IFEND;
    112 12934       mmp$find_next_pfti (pfti);
    166 12935     WHILEND;
    172 12936
    172 12937     IF purge_all_page_maps THEN
    176 12938       mmp$sva_purge_all_page_map (sva);
    18E 12939     IFEND;
    18E 12940
    18E 12941   PROCEND mmp$mm_conditional_free;
```



```
      0 12942
```

```
        0 12945 {----------------------------------------------------------------------------------------------------------
        0 12946 {Name:
        0 12947 {  mmp$free_flush
        0 12948 {Purpose:
        0 12949 {  This routine processes the 'FREE' , 'WRITE MODIFIED PAGES', and
        0 12950 {  'CONDITIONAL FREE' requests
        0 12951 {Input:
        0 12952 {  rb - request block
        0 12953 {Output:
        0 12954 {  none
        0 12955 {Error Codes:
        0 12956 {  invalid PVA
        0 12957 {----------------------------------------------------------------------------------------------------------
        0 12958
        0 12959   PROCEDURE [XDCL] mmp$free_flush
        0 12960     (VAR rb: mmt$rb_free_flush;
        0 12961          cst_p: ^ost$cpu_state_table);
        0 12962
        0 12963     VAR
        0 12964       fde_p: gft$locked_file_desc_entry_p,
        0 12965       wmp_status: mmt$write_modified_pages_status,
        0 12966       last_written_pfti: mmt$page_frame_index,
        0 12967       ste_p: ^mmt$segment_descriptor,
        0 12968       stxe_p: ^mmt$segment_descriptor_extended,
        0 12969       aste_p: ^mmt$active_segment_table_entry,
        0 12970       page_count: integer,
        0 12971       io_id: mmt$io_identifier,
        0 12972       io_count: mmt$active_io_count,
        0 12973       io_already_active: boolean,
        0 12974       sva: ost$system_virtual_address;
        0 12975
        0 12976
        0 12977     io_id.specified := FALSE;
        4 12978
        4 12979     IF NOT mmv$tables_initialized THEN
       1A 12980       RETURN
       1C 12981     IFEND;
       1C 12982     #KEYPOINT (osk$entry, rb.reqcode * osk$m, mmk$free_flush);
       2C 12983     rb.status.normal := TRUE;
       2C 12984
       2C 12985
       2C 12986 {Free pages.
       2C 12987     IF cst_p^.xcb_p^.xp.p_register.pva.ring > 3 THEN
       48 12988       mmp$verify_pva (^rb.pva, mmc$sat_write, rb.status);
       6E 12989     ELSE
       6E 12990       mmp$verify_pva (^rb.pva, mmc$sat_read_or_write, rb.status);
       8E 12991     IFEND;
       8E 12992     IF rb.status.normal THEN
       96 12993       mmp$convert_pva (rb.pva, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
       DA 12994       CASE rb.reqcode OF
       F6 12995       = syc$rc_write_modified_pages =
       F6 12996         IF (aste_p^.queue_id <> mmc$pq_job_working_set) AND ((aste_p^.queue_id < mmc$pq_shared_first) OR
      114 12997               (aste_p^.queue_id > mmc$pq_shared_last)) THEN     {Queue_id is neither JWS nor a shared queue}
      114 12998           mtp$set_status_abnormal ('MM', mme$segment_not_pageable, rb.status);
      12C 12999         ELSEIF fde_p^.media = gfc$fm_transient_segment THEN
      138 13000           mtp$set_status_abnormal ('MM', mme$segment_not_assigned_device, rb.status);
```



```
      150 13001         ELSE
      150 13002           mmp$mm_write_modified_pages (sva, rb.length, fde_p, aste_p, ioc$write_page, rb.init_new_io, FALSE,
      1AC 13003                 io_id, io_count, io_already_active, last_written_pfti, wmp_status);
      1AC 13004           mmp$process_wmp_status (wmp_status, last_written_pfti, rb.waitopt, rb.init_new_io, rb.status);
      1E0 13005         IFEND;
      1E4 13006       = syc$rc_free_pages =
      1E4 13007         mmp$mm_free_pages (sva, rb.length, aste_p, FALSE, page_count);
      218 13008       = syc$rc_conditional_free =
      218 13009         mmp$mm_conditional_free (sva, rb.length, aste_p);
      234 13010       ELSE
      234 13011         mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
      248 13012       CASEND;
      248 13013     IFEND;
      248 13014
      248 13015
      248 13016     #KEYPOINT (osk$exit, 0, mmk$free_flush);
      24C 13017   PROCEND mmp$free_flush;
```

MMP$MOVE_MODIFIED_SERVER_PAGE

```
o 13019 {-------------------------------------------------------------------------------------------------------------
o 13020 { NAME:
o 13021 {   MMP$MOVE_MODIFIED_SERVER_PAGE
o 13022 { PURPOSE:
o 13023 {   This monitor request will move a page frame from a source file , specified by a system_file_id, to a
o 13024 { destination file.  It is used to update the server image file on the client in the case of a server crash.
o 13025 { When the request completes, the page in the range from <rb.byte_offset> to <rb.byte_offset>+ <osv$page_size>
o 13026 { will have been moved to the range of addresses specified by <rb.destination_pva> to <rb.destination_pva>+
o 13027 { <osv$page_size>.  This procedure assumes that <rb.destination_pva> is on a page boundary.  The procedure
o 13028 { executes in monitor mode on behalf of a system task which writes the server image file with pages from all
o 13029 { of the currently attached server files.
o 13030 {
o 13031 { CAUTION:  Be sure to fully understand how the 'move' is accomplished before changing this
o 13032 {           procedure.  Because mmp$delete_pt_entry USES information and mmp$make_pt_entry
o 13033 {           CHANGES information in the page frame table entry, it is necessary to 'move' the
o 13034 {           page in the following order:
o 13035 {             1.  Delete the source page table entry.
o 13036 {             2.  Change the page frame table entry to reflect destination page information.
o 13037 {             3.  Make the page table entry for the destination page.
o 13038 {             4.  If necessary, relink the page frame to the queue for the destination segment.
o 13039 {             5.  Set the valid bit on the destination page.
o 13040 {
o 13041 { PROCEDURE [XDCL] mmp$move_modified_server_page
o 13042 {   (VAR rb: mmt$rb_ring1_server_seg_request;
o 13043 {         cst_p: ^ost$cpu_state_table];
o 13044 {
o 13045 {   RB: (INPUT/OUTPUT) Specifies the request block containing the information
o 13046 {       which will be used by the procedure to move (if possible) a single page
o 13047 {       to the destination (server image) file.
o 13048 {         RB.SFID: Specifies the System File ID of the file whose pages must be
o 13049 {            moved to the destination file
o 13050 {         RB.GLOBAL_FILE_NAME: Specifies the global file name of the file whose
o 13051 {            pages must be moved to the destination file
o 13052 {         RB.DESTINATION_PVA: Specifies the location within the server image
o 13053 {            file to which a located modified page will be written
o 13054 {         RB.BYTE_OFFSET: Specifies the beginning offset of the located page
o 13055 {            which has been moved.  This is an output value.
o 13056 {         RB.STATUS: Specifies the completion status of the request.  Conditions
o 13057 {            which can be returned are:
o 13058 {              mme$io_active_on_move_page
o 13059 {              mme$no_pages_found_for_move
o 13060 {              mme$page_table_full
o 13061 {   CST_P: (INPUT) Specifies the pointer to the CPU state table which is
o 13062 {       executing this process.
o 13063 {-------------------------------------------------------------------------------------------------------------
o 13064
o 13065   PROCEDURE [XDCL] mmp$move_modified_server_page
o 13066     (VAR rb: mmt$rb_ring1_server_seg_request;
o 13067           cst_p: ^ost$cpu_state_table];
o 13068
```


MMP$MOVE_MODIFIED_SERVER_PAGE
  DETERMINE_MOVE_PAGE_STATUS

```
o 13070
o 13071       PROCEDURE [INLINE] determine_move_page_status
o 13072         (     pfti: mmt$page_frame_index;
o 13073               fde_p: gft$locked_file_desc_entry_p;
o 13074          VAR move_page: boolean;
o 13075          VAR lock_encountered: boolean);
o 13076
o 13077         VAR
o 13078           pft_entry_p: ^mmt$page_frame_table_entry;
4 13079
4 13080         move_page := FALSE;
4 13081         pft_entry_p := ^mmv$pft_p^ [pfti];
4 13082
4 13083         IF NOT mmv$pt_p^ [pft_entry_p^.pti].m THEN
4 13084           RETURN;
4 13085         IFEND;
4 13086
4 13087 { The page is not valid for the move to the image if the page lock indicates the page doesn't contain data
4 13088 { (page is being read from disk or server allocation is occurring) or the user has locked the page to prevent
4 13089 { IO.
4 13090
4 13091         IF (pft_entry_p^.locked_page = mmc$lp_aging_lock) OR
4 13092               (pft_entry_p^.locked_page = mmc$lp_server_allocate_lock) OR
4 13093               (pft_entry_p^.locked_page = mmc$lp_page_in_lock) THEN
4 13094           lock_encountered := TRUE;
4 13095           RETURN;
4 13096         IFEND;
4 13097
4 13098 { The page is not valid for the move to the image if the segment is locked (MMP$LOCK_SEGMENT) UNLESS the page
4 13099 { is still being written from a previous MMP$UNLOCK_SEGMENT with write_protection.
4 13100
4 13101         IF fde_p^.segment_lock.locked_for_write AND
4 13102               (pft_entry_p^.locked_page <> mmc$lp_write_protected_lock) THEN
4 13103           lock_encountered := TRUE;
4 13104           RETURN;
4 13105         IFEND;
4 13106
4 13107         move_page := TRUE;
4 13108
4 13109       PROCEND determine_move_page_status;
```

```
        4 13111
        4 13112      VAR
        4 13113        asid: ost$asid,
        4 13114        aste_p: ^mmt$active_segment_table_entry,
        4 13115        asti: mmt$ast_index,
        4 13116        count: 1 .. 32,
        4 13117        destination_aste_p: ^mmt$active_segment_table_entry,
        4 13118        destination_pfte_p: ^mmt$page_frame_table_entry,
        4 13119        destination_pfti: mmt$page_frame_index,
        4 13120        destination_pti: integer,
        4 13121        destination_ste_p: ^mmt$segment_descriptor,
        4 13122        destination_stxe_p: ^mmt$segment_descriptor_extended,
        4 13123        destination_sva: ost$system_virtual_address,
        4 13124        fde_p: gft$locked_file_desc_entry_p,
        4 13125        found: boolean,
        4 13126        lock_encountered: boolean,
        4 13127        move_page: boolean,
        4 13128        mpt_status: mmt$make_pt_entry_status,
        4 13129        pfti: mmt$page_frame_index,
        4 13130        save_valid: boolean,
        4 13131        source_aste_p: ^mmt$active_segment_table_entry,
        4 13132        source_sva: ost$system_virtual_address;
        4 13133
        4 13134
        4 13135      rb.status.normal := TRUE;
        4 13136
        4 13137    { Set up variables for the search through the ASTE for pages associated with the RB.SFID.
        4 13138
        4 13139      gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
       94 13140      mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
       F4 13141      IF asti = 0 THEN
       FC 13142        mtp$set_status_abnormal ('MM', mme$no_pages_found_for_move, rb.status);
       FC 13143        RETURN;
      114 13144      IFEND;
      114 13145
      114 13146      mmp$asid (asti, asid);
      130 13147      source_sva.asid := asid;
      130 13148
      130 13149      mmp$verify_pva (^rb.destination_pva, mmc$sat_write, rb.status);
      15C 13150      IF NOT rb.status.normal THEN
      164 13151        RETURN;
      166 13152      IFEND;
      166 13153
      166 13154      mmp$convert_pva (rb.destination_pva, cst_p, destination_sva, fde_p, destination_aste_p, destination_ste_p,
      1AA 13155            destination_stxe_p);
      1AA 13156
      1AA 13157      source_aste_p := ^mmv$ast_p^ [asti];
      1AA 13158
      1AA 13159    { Start looking through the aste_p^.pft_link for modified pages.  Free those pages which are not modified
      1AA 13160    { and have no IO active on them.
      1AA 13161
      1AA 13162      pfti := source_aste_p^.pft_link.fwd;
      1AA 13163      lock_encountered := FALSE;
      1AA 13164
      1AA 13165      /locate_a_modified_page/
      1AA 13166      WHILE (pfti <> 0) AND (mmv$pft_p^ [pfti].active_io_count = 0) DO
```



```
      1EC 13167        save_valid := mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v;
      1EC 13168        mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := FALSE;
      1EC 13169        mmp$purge_all_map_proc;
      228 13170        determine_move_page_status (pfti, fde_p, move_page, lock_encountered);
      29C 13171        IF NOT move_page THEN
      2A4 13172          IF dfv$file_server_debug_enabled AND lock_encountered THEN
      2B8 13173            display_integer_monitor (' Lock encountered ', pfti);
      2DC 13174          IFEND;
      2DC 13175          mmp$delete_pt_entry (pfti, TRUE);
      2F0 13176          mmp$relink_page_frame (pfti, mmc$pq_free);
      304 13177          pfti := source_aste_p^.pft_link.fwd;
      30C 13178        ELSE
      30C 13179          mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := save_valid;
      30C 13180          EXIT /locate_a_modified_page/;
      340 13181        IFEND;
      340 13182      WHILEND /locate_a_modified_page/;
      368 13183
      368 13184      IF pfti = 0 THEN
      36C 13185        mtp$set_status_abnormal ('MM', mme$no_pages_found_for_move, rb.status);
      36C 13186        mmp$purge_all_map_proc;
      38C 13187        IF lock_encountered THEN
      394 13188          mmp$purge_all_cache_proc;
      39C 13189        IFEND;
      39C 13190        RETURN;
      3A2 13191      ELSEIF (mmv$pft_p^ [pfti].active_io_count > 0) THEN
      3C6 13192        mtp$set_status_abnormal ('MM', mme$io_active_on_move_page, rb.status);
      3C6 13193        mmp$purge_all_map_proc;
      3E6 13194        IF lock_encountered THEN
      3EE 13195          mmp$purge_all_cache_proc;
      3F6 13196        IFEND;
      3F6 13197        RETURN;
      3F8 13198      IFEND;
      3F8 13199
      3F8 13200    { We have located a modified page of the file which now can be moved.
      3F8 13201    { Delete the source page, but save the valid bit in case we need to restore the page table.
      3F8 13202    { (The valid bit was saved as part of the operation to locate a page above.)
      3F8 13203
      3F8 13204      source_sva.offset := mmv$pft_p^ [pfti].sva.offset;
      3F8 13205      mmp$delete_pt_entry (pfti, TRUE);
      436 13206      mmp$purge_all_cache_map_proc;
      43E 13207
      43E 13208    { The destination page should not be in the page table; if it is, delete it.
      43E 13209
      43E 13210      #hash_sva (destination_sva, destination_pti, count, found);
      444 13211      IF found THEN
      458 13212        destination_pfti := (mmv$pt_p^ [destination_pti].rma *512) DIV osv$page_size;
      458 13213        mmp$delete_pt_entry (destination_pfti, TRUE);
      48E 13214        mmp$relink_page_frame (destination_pfti, mmc$pq_free);
      4A2 13215      IFEND;
      4A2 13216
      4A2 13217    { Change the page frame table entry to the destination page information.
      4A2 13218
      4A2 13219      destination_pfte_p := ^mmv$pft_p^ [pfti];
      4A2 13220      destination_pfte_p^.aste_p := destination_aste_p;
      4A2 13221      destination_pfte_p^.sva := destination_sva;
      4A2 13222
```

MMP$MOVE_MODIFIED_SERVER_PAGE

```
4A2 13223 { Make the page table entry for the destination page.  If the page table is full, replace the source page to
4A2 13224 { the page table and return an abnormal status; job mode will reissue the request.
4A2 13225
4A2 13226     mmp$make_pt_entry (destination_sva, pfti, destination_aste_p, destination_pfte_p, mpt_status);
4F8 13227     IF mpt_status = mmc$mpt_page_table_full THEN
500 13228       mmv$async_work.pt_full_aste_p := destination_aste_p;
500 13229       mmv$async_work.pt_full_sva := destination_sva;
500 13230       mmv$async_work.pt_full := TRUE;
500 13231       mmv$time_to_call_mem_mgr := 0;
500 13232       osv$time_to_check_asyn := 0;
500 13233       destination_pfte_p^.aste_p := source_aste_p;
500 13234       destination_pfte_p^.sva := source_sva;
500 13235       mmp$make_pt_entry (source_sva, pfti, source_aste_p, destination_pfte_p, mpt_status);
550 13236       IF mpt_status <> mmc$mpt_done THEN
558 13237         mtp$error_stop ('MOVE_MODIFIED_SERVER_PAGE -- COULD NOT REMAKE PAGE TABLE ENTRY');
578 13238       IFEND;
578 13239       mmv$pt_p^ [destination_pfte_p^.pti].m := TRUE;
578 13240       mmv$pt_p^ [destination_pfte_p^.pti].v := save_valid;
578 13241       mtp$set_status_abnormal ('MM', mme$page_table_full, rb.status);
578 13242       IF lock_encountered THEN
5BA 13243         mmp$purge_all_cache_proc;
5C2 13244       IFEND;
5C2 13245       RETURN;
5C4 13246     IFEND;
5C4 13247
5C4 13248 { Relink the page into the destination's queue.
5C4 13249
5C4 13250     mmp$relink_page_frame (pfti, mmc$pq_wired);
5DA 13251     mmv$pft_p^ [pfti].ijl_ordinal:= cst_p^.ijl_ordinal;
5DA 13252     mmp$relink_page_frame (pfti, destination_aste_p^.queue_id);
61A 13253
61A 13254     mmv$pt_p^ [destination_pfte_p^.pti].v := TRUE;
61A 13255     mmv$pt_p^ [destination_pfte_p^.pti].m := TRUE;
61A 13256
61A 13257     IF lock_encountered THEN
642 13258       mmp$purge_all_cache_proc;
64A 13259     IFEND;
64A 13260
64A 13261     rb.byte_offset := source_sva.offset;
64A 13262
64A 13263   PROCEND mmp$move_modified_server_page;
```


MMP$MTR_R1_SERVER_SEG_REQUEST

```
  0 13265 {-------------------------------------------------------------------------------------------------------------
  0 13266 { NAME:
  0 13267 {   MMP$MTR_R1_SERVER_SEG_REQUEST
  0 13268 { PURPOSE:
  0 13269 {   This procedure processes some ring 1 requests for server segments.
  0 13270 {-------------------------------------------------------------------------------------------------------------
  0 13271
  0 13272   PROCEDURE [XDCL] mmp$mtr_r1_server_seg_request
  0 13273     (VAR rb: mmt$rb_ring1_server_seg_request;
  0 13274          cst_p: ^ost$cpu_state_table);
  0 13275
  0 13276     VAR
  0 13277       able: boolean,
  0 13278       asid: ost$asid,
  0 13279       aste_p: ^mmt$active_segment_table_entry,
  0 13280       asti: mmt$ast_index,
  0 13281       fde_p: gft$locked_file_desc_entry_p,
  0 13282       ijle_p: ^jmt$initiated_job_list_entry,
  0 13283       inhibit_io: boolean,
  0 13284       mcount: integer,
  0 13285       pfti: mmt$page_frame_index,
  0 13286       rcount: integer,
  0 13287       save_v: boolean,
  0 13288       xsva: ost$system_virtual_address;
  0 13289
  0 13290
  0 13291     CASE rb.request OF
 20 13292     = mmc$ssr1_flush_delete_seg_sfid, mmc$ssr1_free_delete_seg_sfid =
 20 13293       rb.pages_not_deleted := 0;
 2A 13294 { Do nothing. This is normal.
 2A 13295       ;
 2A 13296     = mmc$ssr1_move_modified_df_page =
 2A 13297       mmp$move_modified_server_page (rb, cst_p);
 42 13298       RETURN;
 48 13299     ELSE
 48 13300       mtp$error_stop ('MM - Unknown request - mmp$mtr_r1_server_seg_request');
 6A 13301     CASEND;
 6A 13302
 6A 13303 { #### This will not work--if the sfid is not good, get fde_p will crash.
 6A 13304 { Entering monitor with the FDE locked in job mode will greatly increase chances so that the sfid WILL be good
 6A 13305 { and thus avoid a crash in gfp$mtr_get_locked_fde_p.  This procedure is called from DEACT_TERM_SERVER_FILE in
 6A 13306 { the module DFM$RECOVERY SERVICES.
 6A 13307
 6A 13308     gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
 F0 13309     mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
14E 13310     IF (asti = 0) OR (fde_p^.attach_count = 0 ) THEN
162 13311       RETURN;
164 13312     IFEND;
164 13313
164 13314     mmp$asid (asti, asid);
180 13315     xsva.asid := asid;
180 13316     xsva.offset := 0;
180 13317     aste_p := ^mmv$ast_p^ [asti];
180 13318     mmp$initialize_find_next_pfti (xsva, 7fffffff(16), include_partial_pages, psc_all, aste_p, pfti);
1CC 13319
1CC 13320     WHILE pfti <> 0 DO
```

MMP$MTR_R1_SERVER_SEG_REQUEST

```
1D8 13321        IF (mmv$pft_p^ [pfti].aste_p <> NIL) AND (mmv$pft_p^ [pfti].aste_p^.sfid = rb.sfid) THEN
20C 13322          mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, TRUE {lock ajl}, inhibit_io, ijle_p);
32A 13323          IF NOT inhibit_io THEN
332 13324            save_v := mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v;
332 13325            mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := FALSE;
332 13326            mmp$sva_purge_all_page_map (mmv$pft_p^ [pfti].sva);
386 13327            IF (NOT mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m) AND (mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked)
3CA 13328                 AND (mmv$pft_p^ [pfti].active_io_count = 0) THEN
3CA 13329              mmp$delete_pt_entry (pfti, TRUE);
3DE 13330              mmp$relink_page_frame (pfti, mmc$pq_free);
3FA 13331            ELSEIF mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m AND (mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked)
440 13332                 AND (mmv$pft_p^ [pfti].queue_id > mmc$pq_first_valid_in_pt) THEN
440 13333              IF rb.request = mmc$ssr1_flush_delete_seg_sfid THEN
448 13334
448 13335 {              MMP$REMOVE_PAGE_FROM_JWS does not neccessarily write the page to disk.  If it doesn't we may
448 13336 {              have a problem in the future when we really want the page to go out to disk.
448 13337
448 13338                mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := save_v;
448 13339                mmp$remove_page_from_jws (pfti, cst_p^.ijle_p, mcount, rcount);
476 13340                rb.pages_not_deleted := rb.pages_not_deleted + 1;
484 13341              ELSE {rb.request = mmc$ssr1_free_delete_seg_sfid }
484 13342                mmp$delete_pt_entry (pfti, TRUE);
498 13343                mmp$relink_page_frame (pfti, mmc$pq_free);
4B0 13344              IFEND;
4B4 13345            ELSEIF rb.request = mmc$ssr1_flush_delete_seg_sfid THEN
4BC 13346              mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := save_v;
4BC 13347              rb.pages_not_deleted := rb.pages_not_deleted + 1;
4F4 13348            IFEND;
4F4 13349            jmp$unlock_ajl (ijle_p);
5AE 13350          ELSE    { inhibit_io
5AE 13351            IF rb.request = mmc$ssr1_flush_delete_seg_sfid THEN
5B6 13352              rb.pages_not_deleted := rb.pages_not_deleted + 1;
5C0 13353            IFEND;
5C0 13354          IFEND;
5C0 13355        IFEND;
5C0 13356        mmp$find_next_pfti (pfti);
614 13357      WHILEND;
620 13358
620 13359      IF aste_p <> NIL THEN
62A 13360        IF aste_p^.pages_in_memory = 0 THEN
632 13361          mmp$change_asid (aste_p, asid, 0, 0);
652 13362          mmp$free_asid (asid, aste_p);
66A 13363        IFEND;
66A 13364      IFEND;
66A 13365
66A 13366    PROCEND mmp$mtr_r1_server_seg_request;
```


MMP$MTR_RING1_SEGMENT_REQUEST

```
 0 13369
 0 13370 {-------------------------------------------------------------------------------------------------------------
 0 13371 {name:
 0 13372 {  mmp$mtr_ring1_segment_request
 0 13373 {purpose:
 0 13374 { This procedure some ring 1 requests for segments.
 0 13375 {-------------------------------------------------------------------------------------------------------------
 0 13376
 0 13377   PROCEDURE [XDCL] mmp$mtr_ring1_segment_request
 0 13378     (VAR rb: mmt$rb_ring1_segment_request;
 0 13379          cst_p: ^ost$cpu_state_table);
 0 13380
 0 13381
 0 13382     VAR
 0 13383       asid: ost$asid,
 0 13384       asid_can_be_deleted: boolean,
 0 13385       aste_p: ^mmt$active_segment_table_entry,
 0 13386       asti: mmt$ast_index,
 0 13387       fde_p: gft$locked_file_desc_entry_p,
 0 13388       first_image_pfti: 0 .. 0ffffffff(16),
 0 13389       i: integer,
 0 13390       ijle_p: ^jmt$initiated_job_list_entry,
 0 13391       io_already_active: boolean,
 0 13392       io_count: mmt$active_io_count,
 0 13393       io_id: mmt$io_identifier,
 0 13394       j: integer,
 0 13395       last_written_pfti: mmt$page_frame_index,
 0 13396       nowait_wait: [STATIC] array [boolean] of ost$wait := [osc$nowait, osc$wait],
 0 13397       old_sfid: gft$system_file_identifier,
 0 13398       page_count: integer,
 0 13399       pfti: mmt$page_frame_index,
 0 13400       sdte_p: ^mmt$segment_descriptor,
 0 13401       sdtxe_p: ^mmt$segment_descriptor_extended,
 0 13402       sva: ost$system_virtual_address,
 0 13403       wmp_status: mmt$write_modified_pages_status,
 0 13404       unrecovered_files: [STATIC] integer := 0,
 0 13405       unrecovered_pages: [STATIC] integer := 0;
 0 13406
 0 13407
 0 13408     #KEYPOINT (osk$entry, osk$m * $INTEGER (rb.request), mmk$ring1_segment_request);
14 13409
14 13410     io_id.specified := FALSE;
14 13411
14 13412     mmv$ring1_request_trace [$INTEGER (rb.request)] := mmv$ring1_request_trace [$INTEGER (rb.request)] + 1;
14 13413
14 13414     rb.status.normal := TRUE;
14 13415     sva.asid := 0;
14 13416
14 13417   /request/
14 13418     BEGIN
14 13419     CASE rb.request OF
B6 13420     = mmc$sr1_detach_file =
B6 13421       gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
13E 13422      mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
19C 13423      IF asti <> 0 THEN
1A4 13424        aste_p := ^mmv$ast_p^ [asti];
```

```
1A4 13425          IF (fde_p^.queue_status = gfc$qs_job_shared) AND (aste_p^.queue_id <= mmc$pq_shared_last) THEN
1C8 13426
1C8 13427 { The file is job-shared so there can be no modified pages; the pages are already in the shared queue so
1C8 13428 { they do not need to be removed from a job working set.  Therefore, nothing needs to be done.
1C8 13429
1C8 13430            EXIT /request/;
1CC 13431          IFEND;
1CC 13432          mmp$asid (asti, asid);
1E8 13433          sva.asid := asid;
1F4 13434        ELSE
1F4 13435          EXIT /request/;
1F8 13436        IFEND;
1FC 13437
1FC 13438      = mmc$sr1_delete_seg_sfid, mmc$sr1_flush_delete_seg_sfid =
1FC 13439        gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
284 13440        mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
2E2 13441        IF asti <> 0 THEN
2EA 13442          mmp$asid (asti, asid);
302 13443          sva.asid := asid;
302 13444          aste_p := ^mmv$ast_p^ [asti];
31E 13445        ELSE
31E 13446          EXIT /request/;
322 13447        IFEND;
326 13448
326 13449      = mmc$sr1_delete_job_seg_by_sfid =
326 13450        gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
3AE 13451        mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
40C 13452        IF (asti = 0) OR (mmv$ast_p^ [asti].ijl_ordinal <> cst_p^.ijl_ordinal) THEN
42E 13453          EXIT /request/;
432 13454        IFEND;
432 13455        mmp$asid (asti, asid);
44E 13456        sva.asid := asid;
44E 13457        aste_p := ^mmv$ast_p^ [asti];
46A 13458
46A 13459      = mmc$sr1_flush_avail_modified =
46A 13460        gfp$mtr_get_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
4D2 13461        mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
52E 13462        IF asti <> 0 THEN
536 13463          mmp$asid (asti, asid);
54E 13464          mmp$replenish_free_queues (asid);
562 13465        IFEND;
562 13466        EXIT /request/;
56A 13467
56A 13468      = mmc$sr1_get_highest_offset =
56A 13469        rb.highest_offset := 0;
56A 13470        gfp$mtr_get_locked_fde_p (rb.file_sfid, cst_p^.ijle_p, fde_p);
5F6 13471        mmp$get_verify_asti_in_fde (fde_p, rb.file_sfid, cst_p^.ijl_ordinal, asti);
652 13472        IF asti = 0 THEN
65A 13473          EXIT /request/;
65E 13474        IFEND;
65E 13475        aste_p := ^mmv$ast_p^ [asti];
65E 13476        pfti := aste_p^.pft_link.fwd;
65E 13477
65E 13478       /locate_highest_offset/
65E 13479        WHILE pfti <> 0 DO
672 13480          IF mmv$pft_p^ [pfti].sva.offset > rb.highest_offset THEN
```



```
6A0 13481            rb.highest_offset := mmv$pft_p^ [pfti].sva.offset;
6A4 13482          IFEND;
6A4 13483          pfti := mmv$pft_p^ [pfti].segment_link.fwd;
6A4 13484        WHILEND /locate_highest_offset/;
6C4 13485
6C4 13486        EXIT /request/;
6CC 13487
6CC 13488      = mmc$sr1_delete_seg_segnum, mmc$sr1_flush_seg_segnum =
6CC 13489        sdtxe_p := mmp$get_sdtx_entry_p (cst_p^.xcb_p, rb.segnum);
6CC 13490        gfp$mtr_get_locked_fde_p (sdtxe_p^.sfid, cst_p^.ijle_p, fde_p);
780 13491        mmp$get_verify_asti_in_fde (fde_p, sdtxe_p^.sfid, cst_p^.ijl_ordinal, asti);
7DE 13492        IF asti <> 0 THEN
7E6 13493          mmp$asid (asti, asid);
7FE 13494          sva.asid := asid;
7FE 13495          aste_p := ^mmv$ast_p^ [asti];
81A 13496        ELSE
81A 13497          EXIT /request/;
81E 13498        IFEND;
82A 13499
82A 13500      = mmc$sr1_commit_memory =
82A 13501        first_image_pfti := osv$180_memory_limits.deadstart_upper DIV osv$page_size;
82A 13502        FOR pfti := first_image_pfti TO UPPERBOUND (mmv$pft_p^) DO
85C 13503          IF mmv$pft_p^ [pfti].queue_id = mmc$pq_free THEN
87C 13504            mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + 1;
87C 13505            mmp$link_page_frame_to_queue (pfti, ^mmv$pft_p^ [pfti]);
91A 13506          IFEND;
91A 13507        FOREND;
920 13508        mmv$image_file.active := FALSE;
920 13509        EXIT /request/;
930 13510
930 13511      = mmc$sr1_free_image_pages =
930 13512        mmp$free_image_pages_mtr;
938 13513        EXIT /request/;
940 13514
940 13515      = mmc$sr1_replace_sfid =
940 13516
940 13517 { The SFID of a permanent file which is awaiting recovery is updated in the ast entry with the new SFID.
940 13518 { (The new SFID residence is also gft$tr_system_wait_recovery.)
940 13519 { If the ast index passed in is not correct, the AST must be searched.
940 13520 { #### I believe IJL ordinals do NOT have to be checked because these can only be perm files. CORRECT??
940 13521
940 13522        aste_p := NIL;
940 13523        IF (rb.asti <> 0) AND (rb.asti <= UPPERBOUND (mmv$ast_p^)) THEN
966 13524          IF mmv$ast_p^ [rb.asti].in_use THEN
974 13525            IF mmv$ast_p^ [rb.asti].sfid = rb.new_sfid THEN [sfid already replaced
980 13526              EXIT /request/;
988 13527            ELSEIF mmv$ast_p^ [rb.asti].sfid = rb.old_sfid THEN
990 13528              aste_p := ^mmv$ast_p^ [rb.asti];
99A 13529            IFEND;
9A6 13530          IFEND;
9A6 13531        IFEND;
9A6 13532
9A6 13533        /ast_search/
9A6 13534        BEGIN
9A6 13535          IF aste_p = NIL THEN
9AA 13536            old_sfid := rb.old_sfid;
```

```
9AA 13537            FOR asti := LOWERBOUND (mmv$ast_p^) TO UPPERBOUND (mmv$ast_p^) DO
9C8 13538              IF mmv$ast_p^ [asti].in_use AND (mmv$ast_p^ [asti].sfid = old_sfid) THEN
9EE 13539                rb.asti := asti;
9EE 13540                aste_p := ^mmv$ast_p^ [asti];
9EE 13541                EXIT /ast_search/;
9F8 13542              IFEND;
9F8 13543            FOREND;
9FC 13544          IFEND;
9FC 13545        END /ast_search/;
9FE 13546
9FE 13547        IF aste_p = NIL THEN
A04 13548          EXIT /request/;
A08 13549        IFEND;
A08 13550
A08 13551        aste_p^.sfid := rb.new_sfid;
A08 13552        gfp$mtr_get_locked_fde_p (rb.new_sfid, cst_p^.ijle_p, fde_p);
A9C 13553        fde_p^.asti := rb.asti;
A9C 13554        EXIT /request/;
AB0 13555
AB0 13556      = mmc$sr1_end_job_recovery =
AB0 13557
AB0 13558        FOR pfti := LOWERBOUND (mmv$pft_p^) TO UPPERBOUND (mmv$pft_p^) DO
AD4 13559          IF (mmv$pft_p^ [pfti].aste_p <> NIL) AND
B08 13560                (mmv$pft_p^ [pfti].aste_p^.sfid.residence = gfc$tr_system_wait_recovery) AND
B08 13561                (mmv$pft_p^ [pfti].queue_id <> mmc$pq_free) THEN
B08 13562            asid := mmv$pft_p^ [pfti].sva.asid;
B08 13563            aste_p := mmv$pft_p^ [pfti].aste_p;
B08 13564            jmp$get_ijle_p (mmv$pft_p^ [pfti].ijl_ordinal, ijle_p);
B08 13565            IF (ijle_p^.swap_status >= jmc$iss_initiate_swapout_io) AND (ijle_p^.swap_status <=
B48 13566              jmc$iss_swapout_io_complete) THEN
B48 13567              tmp$reissue_monitor_request;
B50 13568              tmp$cause_task_switch;
B58 13569            IFEND;
B58 13570            mmp$delete_pt_entry (pfti, TRUE);
B6C 13571            mmp$relink_page_frame (pfti, mmc$pq_free);
B80 13572            IF aste_p^.pages_in_memory = 0 THEN
B88 13573              mmp$free_asid (asid, aste_p);
BA0 13574              unrecovered_files := unrecovered_files + 1;
BAA 13575            IFEND;
BAA 13576            unrecovered_pages := unrecovered_pages + 1;
BAA 13577            IF ((ijle_p^.swap_status >= jmc$iss_job_idle_tasks_complete) AND
BD8 13578                (ijle_p^.swap_status <= jmc$iss_swapped_io_cannot_init)) OR
BD8 13579                (ijle_p^.swap_status = jmc$iss_swapped_io_complete) THEN
BD8 13580              jsp$recalculate_swapped_pages (ijle_p, 1);
BF0 13581            IFEND;
BF0 13582          IFEND;
BF0 13583        FOREND;
BF4 13584
BF4 13585        rb.unrecovered_files := unrecovered_files;
BF4 13586        rb.unrecovered_pages := unrecovered_pages;
BF4 13587        EXIT /request/;
C10 13588
C10 13589      = mmc$sr1_make_mfw_cache =
C10 13590        mtv$monitor_segment_table.st [#segment (osv$mainframe_wired_heap)].ste.v1 :=
C10 13591          osc$v1_regular_segment;
C10 13592        mtv$nos_segment_table_p^.st [#segment (osv$mainframe_wired_heap)].ste.v1 :=
```



```
C10 13593            osc$v1_regular_segment;
C10 13594        #PURGE_BUFFER (osc$purge_all_page_seg_map, sva);
C5C 13595        EXIT /request/;
C64 13596
C64 13597      = mmc$sr1_remove_job_shared_pages =
C64 13598        gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
CEC 13599        IF fde_p^.asti <> 0 THEN
CF8 13600          mmp$remove_jws_to_shared_pages (fde_p, cst_p, rb);
D0A 13601        IFEND;
D0A 13602        EXIT /request/;
D12 13603
D12 13604      = mmc$sr1_change_swap_file_queue =
D12 13605        gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
D9A 13606        mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
DF6 13607        IF asti <> 0 THEN
DFE 13608          mmv$ast_p^ [asti].ijl_ordinal := jmv$system_ijl_ordinal;
DFE 13609          mmv$ast_p^ [asti].queue_id := mmc$pq_shared_pf_non_execute;
E1A 13610        IFEND;
E1A 13611        fde_p^.queue_status := gfc$qs_global_shared;
E1A 13612        EXIT /request/;
E2A 13613
E2A 13614      ELSE
E2A 13615        mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
E2A 13616        EXIT /request/;
E42 13617      CASEND;
E42 13618
E42 13619        sva.offset := 0;
E42 13620        asid_can_be_deleted := NOT ((rb.request = mmc$sr1_detach_file) OR
E5C 13621              (rb.request = mmc$sr1_flush_seg_segnum));
E5C 13622        IF asid_can_be_deleted AND fde_p^.flags.global_template_file THEN
E6E 13623
E6E 13624 [!!!!!!This check is required in order to run multiple job templates. It may be worth
E6E 13625 [ some time to look into the file kinds of multiple job template segments. There appear to
E6E 13626 [ some strange uses of file kind.
E6E 13627
E6E 13628          IF NOT syv$user_templates THEN
E76 13629            mtp$error_stop ('MM - tried to delete template segment');
E96 13630          IFEND;
E96 13631        IFEND;
E96 13632        IF (rb.request = mmc$sr1_detach_file) OR (rb.request = mmc$sr1_flush_delete_seg_sfid) OR
EAC 13633            (rb.request = mmc$sr1_flush_seg_segnum) THEN
EAC 13634          mmp$mm_write_modified_pages (sva, 7ffffff0(16), fde_p, aste_p, ioc$write_page, rb.init_new_io,
F12 13635                (rb.request = mmc$sr1_detach_file),
F12 13636                io_id, io_count, io_already_active, last_written_pfti, wmp_status);
F12 13637          mmp$process_wmp_status (wmp_status, last_written_pfti, nowait_wait [rb.wait_for_io_complete],
F4A 13638                rb.init_new_io, rb.status);
F4A 13639          IF ((wmp_status <> mmc$wmp_io_complete) AND (wmp_status <> mmc$wmp_io_active)) OR
F6A 13640             ((wmp_status = mmc$wmp_io_active) AND rb.wait_for_io_complete) THEN
F6A 13641            asid_can_be_deleted := FALSE;
F6E 13642          IFEND;
F6E 13643        IFEND;
F6E 13644
F6E 13645        IF asid_can_be_deleted THEN
F72 13646          fde_p^.asti := 0;
F72 13647          mmp$mm_free_pages (sva, 7fffffff(16), aste_p, TRUE, page_count);
FA4 13648        IFEND;
```

MMP$MTR_RING1_SEGMENT_REQUEST

```
FA4 13649
FA4 13650      END /request/;
FA4 13651
FA4 13652      #KEYPOINT (osk$exit, osk$m * $INTEGER (rb.request), mmk$ring1_segment_request);
FB4 13653
FB4 13654    PROCEND mmp$mtr_ring1_segment_request;
  0 13655
```


MMP$FETCH_STACK_SEGMENT_INFO

```
  0 13658 {-------------------------------------------------------------
  0 13659 { This request returns the segment number and maximum segment length of the
  0 13660 { stack segment for the ring number specified by the caller.  It sets the
  0 13661 { stack segment length to zero if requested.  This request is available in
  0 13662 { Monitor Mode only.
  0 13663 {
  0 13664 {     MMP$FETCH_STACK_SEGMENT_INFO (XCB_P, RING, SET_LENGTH, SEGNUM,
  0 13665 {             LENGTH, FOUND);
  0 13666 {
  0 13667 { XCB_P: (input) This parameter specifies a pointer to the execution control
  0 13668 {       block of the task.  The task must be swapped in.
  0 13669 {
  0 13670 { RING: (input) This parameter specifies the ring number of the stack.
  0 13671 {
  0 13672 { SET_LENGTH: (input) This parameter specifies whether or not to set the
  0 13673 {       length of the stack to zero.
  0 13674 {
  0 13675 { SEGNUM: (output) This parameter returns the segment number for the stack
  0 13676 {       segment of the specified ring.
  0 13677 {
  0 13678 { LENGTH: (output) This parameter returns the maximum size allowed for the
  0 13679 {       stack.
  0 13680 {
  0 13681 { FOUND: (output) This parameter returns whether or not the stack was
  0 13682 {       found.
  0 13683 {
  0 13684
  0 13685 {-------------------------------------------------------------
  0 13686
  0 13687
  0 13688   PROCEDURE [XDCL] mmp$fetch_stack_segment_info
  0 13689     (     xcb_p: ^ost$execution_control_block;
  0 13690           ring: ost$valid_ring;
  0 13691           set_length_to_zero: boolean;
  0 13692      VAR stack_segment_number: ost$segment;
  0 13693      VAR maximum_segment_length: ost$segment_length;
  0 13694      VAR found: boolean);
  0 13695
  0 13696     VAR
  0 13697       fde_p: gft$locked_file_desc_entry_p,
  0 13698       ijl_ordinal: jmt$ijl_ordinal,
  0 13699       ijle_p: ^jmt$initiated_job_list_entry,
  0 13700       limit: amt$file_limit,
  0 13701       sdt_p: mmt$max_sdt_p,
  0 13702       sdtx_p: mmt$max_sdtx_p,
  0 13703       segnum: ost$segment,
  0 13704       status: syt$monitor_status;
  0 13705
  0 13706     tmp$obtain_ijl_ordinal_from_pt1 (xcb_p^.global_task_id, ijl_ordinal);
 24 13707     jmp$get_ijle_p (ijl_ordinal, ijle_p);
 24 13708     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
 24 13709
 24 13710     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
 92 13711       IF (mmc$sa_stack IN sdtx_p^.sdtx_table [segnum].software_attribute_set) AND
 C4 13712             (sdt_p^.st [segnum].ste.r1 = ring) THEN
 C4 13713         found := TRUE;
```

MMP$FETCH_STACK_SEGMENT_INFO

```
      C4 13714          stack_segment_number := segnum;
      C4 13715          gfp$mtr_get_locked_fde_p (sdtx_p^.sdtx_table [segnum].sfid, ijle_p, fde_p);
     158 13716          maximum_segment_length := fde_p^.file_limit;
     158 13717          IF set_length_to_zero THEN
     170 13718            mmp$update_eoi (fde_p, 0, mmc$uer_set_exact_eoi);
     188 13719          IFEND;
     188 13720          RETURN;
     18A 13721        IFEND;
     18A 13722      FOREND;
     18E 13723
     18E 13724      found := FALSE;
     18E 13725
     18E 13726    PROCEND mmp$fetch_stack_segment_info;
```


MMP$PERIODIC_CALL

```
       0 13729 {-------------------------------------------------------------------
       0 13730 {This procedure is called periodically to age the Shared Page Queues.
       0 13731 {
       0 13732 {   MMP$PERIODIC_CALL
       0 13733 {
       0 13734 {-------------------------------------------------------------------
       0 13735
       0 13736
       0 13737   PROCEDURE [XDCL] mmp$periodic_call;
       0 13738
       0 13739
       0 13740     VAR
       0 13741       fde_p: gft$locked_file_desc_entry_p,
       0 13742       aste_p: ^mmt$active_segment_table_entry,
       0 13743       jcb_p: ^jmt$job_control_block,
       0 13744       ajlo: jmt$ajl_ordinal,
       0 13745       inhibit_io: boolean,
       0 13746       clock: ost$free_running_clock,
       0 13747       cptime: integer,
       0 13748       new_asid: ost$asid,
       0 13749       new_asti: mmt$ast_index,
       0 13750       new_aste_p: ^mmt$active_segment_table_entry,
       0 13751       pfti: mmt$page_frame_index,
       0 13752       ijle_p: ^jmt$initiated_job_list_entry,
       0 13753       pt_full_status: mmt$pt_full_status,
       0 13754       taskid: ost$global_task_id,
       0 13755       mcount: integer,
       0 13756       rcount: integer,
       0 13757       aggressive_aging: boolean,
       0 13758       maxws_left_for_user_jobs: mmt$page_frame_index,
       0 13759       dp: jmt$dispatching_priority,
       0 13760       i: integer,
       0 13761       idle_candidates: jmt$dispatching_priority_set,
       0 13762       total_idle: ost$free_running_clock,
       0 13763       user_dp_set: [STATIC] jmt$dispatching_priority_set := $jmt$dispatching_priority_set
       0 13764             [7,8,9,10,11,12,13,14],
       0 13765       queue: mmt$page_frame_queue_id,
       0 13766       temp_max_working_set_size: mmt$page_frame_index,
       0 13767       temp_target: mmt$page_frame_index,
       0 13768       time_last_idle_dispatching_scan: [STATIC] integer := 5000000,
       0 13769       time_last_shared_queue_scan: [STATIC] integer := 5000000,
       0 13770       time_last_full_jws_scan: [STATIC] integer := 5000000,
       0 13771       time_next_free_astes: [STATIC] integer := 0,
       0 13772       time_next_scan_wait_not_queued: [STATIC] integer := 600000000,
       0 13773       pass: integer,
       0 13774       pti: ost$page_table_index,
       0 13775       asid: ost$asid,
       0 13776       asti: mmt$ast_index,
       0 13777       system_jws: mmt$page_frame_index;
       0 13778
       0 13779
       0 13780
       0 13781
       0 13782     #KEYPOINT (osk$entry, 0, mmk$periodic_call);
       8 13783
       8 13784 {Age the shared and job working set queues if necessary:
```

```
      8 13785 {     shared - aged every few seconds
      8 13786 {     job (algorithm 0) - aged every few seconds and all pages not referenced since last time
      8 13787 {                         are removed
      8 13788 {     job (algorithm 1) - every few seconds a scan is made of each job. If job has used a TICKTIME of
      8 13789 {                         cp time, the working set is aged same as though a page fault occurred.
      8 13790 {***NOTE - if system is real low on free/avail pages, aging is forced even if not necessary.
      8 13791
      8 13792 { Set the global maximum working set to the number of pages available to user jobs.
      8 13793 { The size of a job's working set will be constrained to the lesser of the global maximum working set
      8 13794 { and the maximum working set size value in the job control block, which is determined by the class attribute.
      8 13795 { The value for the global maximum working set is the largest working set the system can accommodate.  Since
      8 13796 { the System Job is also limited by the global maximum working set (mmv$max_working_set_size) the System Job
      8 13797 { is effectively limited to 50% of the available space.  If the System Job is the only job, the maximum is
      8 13798 { calculated as the maximum space available to the system job (although it is unlikely to grow that large).
      8 13799
      8 13800     IF jmv$max_class_working_set = 0  THEN  { =0 when system job is the only job.}
     14 13801       system_jws := 0;
     1E 13802     ELSE
     1E 13803       get_system_jobs_working_set (system_jws);
     5E 13804     IFEND;
     5E 13805     maxws_left_for_user_jobs := mmv$total_page_frames - mmv$gpql [mmc$pq_wired].pqle.count -
     5E 13806           mmv$gpql [mmc$pq_shared_io_error].pqle.count - system_jws;
     5E 13807     FOR queue := mmc$pq_shared_first TO mmv$last_active_shared_queue  DO
     88 13808       maxws_left_for_user_jobs := maxws_left_for_user_jobs - mmv$gpql [queue].pqle.count;
     88 13809     FOREND;
     96 13810     IF jmv$max_class_working_set < maxws_left_for_user_jobs THEN
     9E 13811       temp_max_working_set_size := jmv$max_class_working_set;
     A2 13812     ELSE
     A2 13813       temp_max_working_set_size := maxws_left_for_user_jobs;
     A4 13814     IFEND;
     A4 13815     temp_target := (temp_max_working_set_size * jsv$swapped_page_entry_size DIV
     A4 13816           osv$page_size + 1) + mmv$aggressive_aging_level;
     A4 13817     IF temp_target > jmv$job_scheduler_table.scheduling_memory_levels.target THEN
     D0 13818       mmv$resident_job_target := temp_target;
     DC 13819     ELSE
     DC 13820       mmv$resident_job_target := jmv$job_scheduler_table.scheduling_memory_levels.target;
     E4 13821     IFEND;
     E4 13822     mmv$max_working_set_size := maxws_left_for_user_jobs - mmv$resident_job_target;
     E4 13823     IF mmv$max_working_set_size < 10 THEN
     F8 13824       mmv$max_working_set_size := 10;
     FC 13825     IFEND;
     FC 13826
     FC 13827 { Mark any non-dispatchable dispatching priorities as blocked in the idle dispatching controls.
     FC 13828 { Clear the blocked field and swapin jobs for priorities which were blocked, but can now be
     FC 13829 { dispatched.  NOTE:  A dispatching priority in a SET is converted so that the highest dispatching
     FC 13830 { priority in the SET corresponds to the leftmost bit in the SET. (See jmt$dispatching_priority.)
     FC 13831
     FC 13832     clock := #free_running_clock (0);
    102 13833     IF (clock - time_last_idle_dispatching_scan) > jmv$scan_idle_dispatch_interval THEN
    116 13834       FOR i := 0 TO (osv$cpus_physically_configured - 1) DO
    126 13835         IF mtv$cst0 [i].processor_state = cmc$on THEN
    13C 13836           total_idle := mtv$cst0 [i].cpu_idle_statistics.idle_no_io_active +
    14A 13837                         mtv$cst0 [i].cpu_idle_statistics.idle_io_active;
    14A 13838         IFEND;
    14A 13839       FOREND;
    14E 13840       IF (total_idle > jmv$idle_dispatching_controls.controls [0].last_cp_time) OR
```



```
    176 13841             ((jmv$idle_dispatching_controls.unblocked_priorities * tmv$dispatching_control_sets.ready_tasks) -
    176 13842             jmv$idle_dispatching_controls.maximums_exceeded = $jmt$dispatching_priority_set []) THEN
    176 13843
    176 13844 { Unblock all idled dispatching priorities.
    176 13845
    176 13846         jmv$idle_dispatching_controls.unblocked_priorities := user_dp_set;
    176 13847         FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
    188 13848           jmv$idle_dispatching_controls.controls [dp].blocked := FALSE;
    188 13849           jmv$idle_dispatching_controls.controls [dp].timestamp := clock;
    188 13850           jmv$idle_dispatching_controls.controls [dp].last_cp_time := tmv$cpu_execution_statistics [dp].
    188 13851                 time_spent_in_job_mode + tmv$cpu_execution_statistics [dp].time_spent_in_mtr_mode;
    188 13852         FOREND;
    1BC 13853         jmp$set_scheduler_event (jmc$examine_swapin_queue);
    1CE 13854         jmp$set_scheduler_event (jmc$examine_input_queue);
    1E4 13855       ELSE
    1E4 13856
    1E4 13857 { Checked for blocked priorities.
    1E4 13858
    1E4 13859         idle_candidates := [user_dp_set - tmv$dispatching_controls.minimums_to_satisfy) *
    1E4 13860               tmv$dispatching_control_sets.ready_tasks;
    1E4 13861
    1E4 13862         FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
    200 13863           IF NOT ((jmc$dp_conversion - dp) IN idle_candidates) OR (jmv$idle_dispatching_controls.controls [dp]
    250 13864                 .last_cp_time <> (tmv$cpu_execution_statistics [dp].time_spent_in_job_mode +
    250 13865                 tmv$cpu_execution_statistics [dp].time_spent_in_mtr_mode]) THEN
    250 13866
    250 13867             jmv$idle_dispatching_controls.controls [dp].last_cp_time := tmv$cpu_execution_statistics
    250 13868                 [dp].time_spent_in_job_mode + tmv$cpu_execution_statistics [dp].
    250 13869                 time_spent_in_mtr_mode;
    250 13870             jmv$idle_dispatching_controls.controls [dp].timestamp := clock;
    280 13871
    280 13872           ELSE
    280 13873             IF ((jmv$idle_dispatching_controls.controls [dp].timestamp + jmv$job_scheduler_table.
    28E 13874                   idle_dispatching_queue_time) < clock) THEN
    28E 13875               jmv$idle_dispatching_controls.controls [dp].blocked := TRUE;
    28E 13876               jmv$idle_dispatching_controls.unblocked_priorities := jmv$idle_dispatching_controls.
    2AC 13877                     unblocked_priorities - $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
    2AC 13878             IFEND;
    2AC 13879           IFEND;
    2AC 13880         FOREND;
    2B2 13881       IFEND;
    2B2 13882       jmv$idle_dispatching_controls.maximums_exceeded := $jmt$dispatching_priority_set [];
    2B2 13883       jmv$idle_dispatching_controls.controls [0].last_cp_time := total_idle;
    2B2 13884       jmv$idle_dispatching_controls.controls [0].timestamp := clock;
    2B2 13885       time_last_idle_dispatching_scan := clock;
    2C2 13886     IFEND; {time to scan idle dispatching}
    2C2 13887
    2C2 13888 { Insert timed_wait_not_queued tasks into the timed wait queue if it is nearly time for them to be readied.
    2C2 13889
    2C2 13890     clock := #free_running_clock (0);
    2C8 13891     IF clock >= time_next_scan_wait_not_queued THEN
    2D0 13892       time_next_scan_wait_not_queued := clock + tmv$timed_wait_not_queued;
    2D0 13893       tmp$check_timed_wait_not_queued (time_next_scan_wait_not_queued);
    2EE 13894     IFEND;
    2EE 13895
    2EE 13896
```

```
2EE 13897      mmp$maintain_memory_thresholds;
328 13898
328 13899      aggressive_aging := mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon <=
328 13900            mmv$aggressive_aging_level + jsv$pages_needed_for_sfd;
328 13901      IF aggressive_aging THEN
34C 13902        mmv$aging_statistics.aggressive_age_shared_queue := mmv$aging_statistics.aggressive_age_shared_queue +1;
35A 13903      IFEND;
35A 13904
35A 13905      IF aggressive_aging OR ((#FREE_RUNNING_CLOCK (0) - time_last_shared_queue_scan) >
372 13906            mmv$shared_queue_age_interval) THEN
372 13907        jsp$adv_expired_swapped_jobs (jsc$isqi_swapped_io_not_init);
384 13908        jsp$adv_expired_swapped_jobs (jsc$isqi_swapped_io_completed);
396 13909        FOR queue := mmc$pq_shared_first TO mmv$last_active_shared_queue  DO
3A4 13910          IF aggressive_aging THEN
3A8 13911            mmp$remove_stale_pages (mmv$gpql [queue].pqle, mmv$gpql [queue].age_interval, NIL, NIL,
3F0 13912                mmc$pq_avail_modified, 0, mcount, rcount);
3F0 13913          ELSE
3F0 13914            mmp$remove_stale_pages (mmv$gpql [queue].pqle, mmv$gpql [queue].age_interval, NIL, NIL,
438 13915                mmc$pq_avail_modified, mmv$gpql [queue].minimum, mcount, rcount);
438 13916          IFEND;
438 13917          mmv$aging_statistics.age_unused_page_in_shared_queue := mmv$aging_statistics.
438 13918                age_unused_page_in_shared_queue + rcount;
438 13919          IF queue <= mmc$pq_shared_last_sys THEN
44C 13920            mmv$aging_statistics.age_sys_shared_queue [queue]:=
45A 13921                  mmv$aging_statistics.age_sys_shared_queue[queue] + rcount;
45A 13922          IFEND;
45A 13923        FOREND;
45E 13924        mmp$remove_stale_pages (mmv$gpql [mmc$pq_shared_io_error].pqle, mmv$gpql [mmc$pq_shared_io_error].
49E 13925              age_interval, NIL, NIL, mmc$pq_avail_modified, 0, mcount, rcount);  {enforces "0" as minimum size
49E 13926        time_last_shared_queue_scan := #FREE_RUNNING_CLOCK (0);
4AC 13927      IFEND;
4AC 13928
4AC 13929      aggressive_aging := mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon <=
4AC 13930            mmv$aggressive_aging_level + jsv$pages_needed_for_sfd;
4AC 13931      IF aggressive_aging THEN
4D8 13932        mmv$aging_statistics.aggressive_age_job_queues := mmv$aging_statistics.aggressive_age_job_queues + 1;
4E6 13933      IFEND;
4E6 13934      IF aggressive_aging OR mmv$reduce_jws_for_thrashing OR
50A 13935            ((#FREE_RUNNING_CLOCK (0) - time_last_full_jws_scan) > mmv$jws_queue_age_interval) THEN
50A 13936
50A 13937        FOR ajlo := LOWERBOUND (jmv$ajl_p^) TO jmv$max_ajl_ordinal_in_use DO
51E 13938          tmp$set_lock (tmv$ptl_lock);
556 13939          IF (jmv$ajl_p^ [ajlo].in_use <> 0) AND (jmv$ajl_p^ [ajlo].ijle_p^.swap_status =
578 13940                jmc$iss_executing) THEN
578 13941            jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use + jmc$lock_ajl;
578 13942            tmp$clear_lock (tmv$ptl_lock);
5B8 13943            ijle_p := jmv$ajl_p^ [ajlo].ijle_p;
5B8 13944            IF ijle_p^.maxws_aio_slowdown_display > 0 THEN
5D6 13945              ijle_p^.maxws_aio_slowdown_display := ijle_p^.maxws_aio_slowdown_display - 1;
5DC 13946            IFEND;
5DC 13947            IF mmv$aging_algorithm >= 4 THEN
5EA 13948              cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
5F6 13949            ELSE
5F6 13950              cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode + ijle_p^.statistics.cp_time.
606 13951                    time_spent_in_mtr_mode;
606 13952            IFEND;
```



```
606 13953            jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajlo, 0);
606 13954            IF jcb_p^.next_cyclic_aging_time < #FREE_RUNNING_CLOCK (0) THEN
626 13955              mmp$remove_stale_pages (ijle_p^.job_page_queue_list [mmc$pq_job_working_set], 1, jcb_p, ijle_p,
668 13956                    mmc$pq_avail_modified, jcb_p^.min_working_set_size, mcount, rcount);
668 13957              mmp$remove_stale_pages (ijle_p^.job_page_queue_list [mmc$pq_job_io_error], 30, jcb_p, ijle_p,
6A0 13958                    mmc$pq_avail_modified, 0, mcount, rcount);
6A0 13959            IFEND;
6A0 13960            IF (ijle_p^.swap_status = jmc$iss_executing) AND (ijle_p^.entry_status = jmc$ies_job_in_memory) AND
6CC 13961                  (NOT jmv$ajl_p^ [ajlo].job_is_good_swap_candidate) THEN
6CC 13962              IF ((#FREE_RUNNING_CLOCK (0) - jcb_p^.last_execution_time) > tmv$long_wait_force_swap_time) AND
6EC 13963                    (ijle_p^.statistics.ready_task_count = 0) THEN
6EC 13964                tmp$check_for_swapout_candidate (ajlo);
700 13965              ELSEIF jmv$idle_dispatching_controls.controls [ijle_p^.scheduling_dispatching_priority].blocked
71C 13966                    THEN
71C 13967                tmp$idle_non_dispatchable_job (ajlo);
72C 13968              IFEND;
72C 13969            IFEND;
72C 13970            IF aggressive_aging THEN
730 13971              mmp$age_job_working_set (ijle_p, jcb_p);
74C 13972            ELSEIF cptime > (jcb_p^.cptime_next_age_working_set + 2 * jcb_p^.page_aging_interval) THEN
75E 13973              mmv$aging_statistics.age_cp_bound_job := mmv$aging_statistics.age_cp_bound_job + 1;
75E 13974              mmp$age_job_working_set (ijle_p, jcb_p);
784 13975            IFEND;
784 13976            jmp$unlock_ajl (ijle_p);
836 13977          ELSE
836 13978            tmp$clear_lock (tmv$ptl_lock);
86E 13979          IFEND;
86E 13980        FOREND;
872 13981        time_last_full_jws_scan := #FREE_RUNNING_CLOCK (0);
880 13982      IFEND;
880 13983
880 13984
880 13985 {Call replenish free queue.
880 13986
880 13987      mmp$replenish_free_queues (0);
890 13988
890 13989 {Reclaim unused ast entries
890 13990
890 13991      IF mmv$async_work.reclaim_astes THEN
89C 13992        mmv$async_work.reclaim_astes := FALSE;
89C 13993        mmp$reclaim_ast_entries (0);
8B0 13994      IFEND;
8B0 13995
8B0 13996 {Process outstanding page table full conditions.
8B0 13997
8B0 13998      IF mmv$async_work.pt_full THEN
8B8 13999        IF mmv$async_work.pt_full_aste_p^.in_use THEN
8C4 14000
8C4 14001          IF jmp$ijl_block_valid (mmv$async_work.pt_full_aste_p^.ijl_ordinal) THEN
8F2 14002            mmp$get_inhibit_io_status (mmv$async_work.pt_full_aste_p^.ijl_ordinal, FALSE {lock ajl},
930 14003                  inhibit_io, ijle_p);
930 14004          ELSE
930 14005            inhibit_io := FALSE;
934 14006          IFEND;
934 14007
934 14008          IF NOT inhibit_io THEN
```

```
93C 14009            mmp$process_page_table_full (mmv$async_work.pt_full_sva, new_asid, new_asti,
970 14010                  new_aste_p, pt_full_status);
970 14011          IFEND;
970 14012        IFEND;
970 14013        mmv$async_work.pt_full := FALSE;
974 14014      IFEND;
974 14015
974 14016
974 14017 {If tasks are in the memory wait queue, ready one task. This mechanism is NOT the normal mechanism
974 14018 {for waking tasks in memory wait. Normally this is done as soon as the memory becomes available. This
974 14019 {mechanism is a FAIL-SAFE mechanism in case 1) the task waiting for memory doesnt request it again
974 14020 {when it is readied from the memory-wait queue, or 2) give critical tasks memory when sever thrashing occurs.
974 14021
974 14022      IF (mmv$memory_wait_queue.head <> 0) AND (mmv$reassignable_page_frames.now > 0) THEN
98C 14023        tmp$dequeue_task (mmv$memory_wait_queue, taskid);
9A4 14024      IFEND;
9A4 14025
9A4 14026      IF osv$keypoint_control.periodic_requested THEN
9B0 14027        osp$process_keypoint_periodic;
9B8 14028      IFEND;
9B8 14029
9B8 14030 {Update statistics.
9B8 14031
9B8 14032      IF mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon <= mmv$aggressive_aging_level THEN
9D2 14033        mmv$aging_statistics.aggressive_aging_failed := mmv$aging_statistics.aggressive_aging_failed + 1;
9E0 14034      IFEND;
9E0 14035
9E0 14036
9E0 14037 {** Debug code - allow for testing of ASID REASSIGNMENT.
9E0 14038
9E0 14039      IF mmv$test_reassign_asid AND (#FREE_RUNNING_CLOCK (0) > time_next_free_astes) THEN
9FE 14040        FOR asti := 1 TO UPPERBOUND (mmv$ast_p^) DO
A16 14041          aste_p := ^mmv$ast_p^ [asti];
A16 14042          IF (aste_p^.pages_in_memory = 0) AND aste_p^.in_use THEN
A32 14043            mmp$asid (asti, asid);
A4A 14044            mmp$change_asid (^mmv$ast_p^ [asti], asid, 0, 0);
A70 14045            mmp$free_asid (asid, ^mmv$ast_p^ [asti]);
A8E 14046          IFEND;
A8E 14047          IF aste_p^.in_use THEN
A96 14048            IF aste_p^.sfid.residence = gfc$tr_system THEN
A9E 14049              gfp$mtr_get_locked_fde_p (aste_p^.sfid, NIL, fde_p);
B24 14050              IF fde_p^.asti <> asti THEN
B30 14051                mtp$error_stop ('MM - dangling AST entry found');
B50 14052              IFEND;
B50 14053            IFEND;
B50 14054          IFEND;
B50 14055        FOREND;
B54 14056        time_next_free_astes := #FREE_RUNNING_CLOCK (0) + 1000000;
B6C 14057      IFEND;
B6C 14058
B6C 14059
B6C 14060 {** DEBUG code - allow for testing PAGE TABLE full.
B6C 14061
B6C 14062      FOR pass := 1 TO mmv$test_pt_full DO
B8A 14063        pti := #FREE_RUNNING_CLOCK (0) MOD mmv$pt_length;
B90 14064        pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
```



```
B90 14065        WHILE (mmv$pt_p^ [pti].pageid.asid = 0) OR (pfti < LOWERBOUND (mmv$pft_p^)) OR
C1A 14066              (pfti > UPPERBOUND (mmv$pft_p^)) OR (mmv$pft_p^ [pfti].aste_p = NIL) OR
C1A 14067              (mmv$pft_p^ [pfti].aste_p^.in_use = FALSE) DO
C1A 14068          pti := pti + 1;
C1A 14069          IF pti = mmv$pt_length THEN
C24 14070            pti := 0;
C28 14071          IFEND;
C28 14072          pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
C28 14073        WHILEND;
CC8 14074        IF jmp$ijl_block_valid (mmv$pft_p^ [pfti].ijl_ordinal) THEN
D02 14075          mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, FALSE {lock ajl}, inhibit_io, ijle_p);
D54 14076        ELSE
D54 14077          inhibit_io := FALSE;
D58 14078        IFEND;
D58 14079        IF NOT inhibit_io THEN
D60 14080          mmp$process_page_table_full (mmv$pft_p^ [pfti].sva, new_asid, new_asti, new_aste_p,
DAC 14081                pt_full_status);
DAC 14082        IFEND;
DAC 14083      FOREND;
DBO 14084
DBO 14085
DBO 14086 {Reset the time that CP Monitor should next call this procedure.
DBO 14087
DBO 14088      mmv$time_to_call_mem_mgr := mmv$periodic_call_interval + #FREE_RUNNING_CLOCK (0);
DB6 14089
DB6 14090      #KEYPOINT (osk$exit, 0, mmk$periodic_call);
DCC 14091
DCC 14092
DCC 14093    PROCEND mmp$periodic_call;
```

```
      0 14096
      0 14097 {----------------------------------------------------------------------------------------------------------
      0 14098 {Name:
      0 14099 {  replenish_free_queues
      0 14100 {Purpose:
      0 14101 {  This routine is called to determine if the number of FREE + AVAILABLE
      0 14102 {  is getting too low.
      0 14103 {
      0 14104 {    asid: (input) If only pages belonging to a specific ASID should be written then this parameter
      0 14105 {        specifies the ASID. If ALL ASIDs should be written then a 0 (zero) is passed.
      0 14106 {
      0 14107 {
      0 14108 {----------------------------------------------------------------------------------------------------------
      0 14109
      0 14110
      0 14111   PROCEDURE [XDCL] mmp$replenish_free_queues
      0 14112     (    asid: ost$asid);
      0 14113
      0 14114     CONST
      0 14115       max_dm_rejects = 32;
      0 14116
      0 14117     VAR
      0 14118       fde_p: gft$locked_file_desc_entry_p,
      0 14119       write_status: mmt$write_page_to_disk_status,
      0 14120       pfti: mmt$page_frame_index,
      0 14121       next_pfti: mmt$page_frame_index,
      0 14122       ijle_p: ^jmt$initiated_job_list_entry,
      0 14123       io_id: mmt$io_identifier,
      0 14124       dm_reject_table: array [1 .. max_dm_rejects] of ost$asid,
      0 14125       i,
      0 14126       j: integer,
      0 14127       write_ok: boolean,
      0 14128       inhibit_io: boolean;
      0 14129
      0 14130     i := 0;
      4 14131     io_id.specified := FALSE;
      4 14132
      4 14133 {Move pages from the AVAIL_MODIFIED queue to the AVAIL queue until free pages exceeds the threshold.
      4 14134
      4 14135     pfti := mmv$gpql [mmc$pq_avail_modified].pqle.link.bkw;
      4 14136
      4 14137     WHILE (pfti <> 0) AND ((mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon) <
     36 14138           mmv$write_aged_out_pages) DO
     36 14139       IF mmv$pft_p^ [pfti].aste_p^.in_use = FALSE THEN
     5E 14140         mtp$error_stop ('MM - replenish found page in AM q with AST free');
     7E 14141       IFEND;
     7E 14142       next_pfti := mmv$pft_p^ [pfti].link.bkw;
     7E 14143       IF (mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m) AND ((asid = 0) OR (mmv$pft_p^ [pfti].sva.asid = asid)) THEN
     C6 14144         mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, TRUE {lock ajl}, inhibit_io, ijle_p);
    1FE 14145         IF NOT inhibit_io THEN
    206 14146           write_ok := TRUE;
    206 14147
    206 14148         /asid_check/
    206 14149           FOR j := 1 TO i DO
    212 14150             IF mmv$pft_p^ [pfti].sva.asid = dm_reject_table [j] THEN
    23A 14151               write_ok := FALSE;
```



```
    23A 14152               jmp$unlock_ajl (ijle_p);
    2F2 14153               EXIT /asid_check/;
    2F8 14154             IFEND;
    2F8 14155           FOREND /asid_check/;
    2FC 14156           IF write_ok THEN
    300 14157             gfp$mtr_get_locked_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijle_p, fde_p);
    3AE 14158             mmp$write_page_to_disk (fde_p, pfti, ioc$write_page, io_id, mmv$multi_page_write, write_status);
    3E4 14159             jmp$unlock_ajl (ijle_p);
    4B0 14160             IF write_status = ws_physical_io_reject THEN
    4B8 14161               RETURN;
    4BE 14162             ELSEIF write_status <> ws_ok THEN
    4C2 14163               i := i + 1;
    4C2 14164               dm_reject_table [i] := mmv$pft_p^ [pfti].sva.asid;
    4C2 14165               IF i >= max_dm_rejects THEN
    4F2 14166                 RETURN;
    4F4 14167               IFEND;
    4F6 14168             IFEND;
    4FE 14169           IFEND;
    4FE 14170         IFEND;
    4FE 14171       IFEND;
    4FE 14172       pfti := next_pfti;
    4FE 14173     WHILEND;
    518 14174
    518 14175   PROCEND mmp$replenish_free_queues;
```

FREE_IMAGE_PAGES

```
 0 14178
 0 14179   PROCEDURE [XDCL] mmp$free_image_pages_mtr;
 0 14180
 0 14181     VAR
 0 14182       i: integer,
 0 14183       pte_p: ^ost$page_table_entry,
 0 14184       pfti: mmt$page_frame_index;
 0 14185
 0 14186     FOR i := 0 TO (mmv$pt_length - 1) DO
16 14187       pte_p := ^mmv$pt_p^ [i];
16 14188       IF (pte_p^.v) AND ((pte_p^.rma * 512) >= osv$180_memory_limits.deadstart_upper) AND
4E 14189            ((pte_p^.rma * 512) < osv$180_memory_limits.upper) THEN
4E 14190         pfti := (pte_p^.rma * 512) DIV osv$page_size;
4E 14191         mmp$delete_pt_entry (pfti, TRUE);
6E 14192         mmv$pft_p^ [pfti].queue_id := mmc$pq_free;
6E 14193         mmv$pft_p^ [pfti].sva.asid := 0;
94 14194       IFEND;
94 14195     FOREND;
98 14196
98 14197   PROCEND mmp$free_image_pages_mtr;
```


MMP$CREATE_TASK

```
 0 14200 {-------------------------------------------------------------------------------------------------------
 0 14201 {This procedure is called by the Dispatcher when a new task is created to initialize the SDT of the
 0 14202 {new task. Initialization consists if copying the ASID's of shared segments from the SDT of the parent
 0 14203 {task into the SDT of the new task.  Update the real memory address of the task's SDT in the task's
 0 14204 {exchange package.
 0 14205 {
 0 14206 {    MMP$CREATE_TASK (PARENT_XCB_P, XCB_P)
 0 14207 {
 0 14208 {-------------------------------------------------------------------------------------------------------
 0 14209
 0 14210
 0 14211   PROCEDURE [XDCL] mmp$create_task
 0 14212     (    parent_xcb_p: ^ost$execution_control_block;
 0 14213          xcb_p: ^ost$execution_control_block;
 0 14214          ijle_p: ^jmt$initiated_job_list_entry);
 0 14215
 0 14216     VAR
 0 14217       fde_p: gft$file_desc_entry_p,
 0 14218       max_segnum: ost$segment,
 0 14219       parent_sdt_p: mmt$max_sdt_p,
 0 14220       parent_sdtx_p: mmt$max_sdtx_p,
 0 14221       sdte: mmt$segment_descriptor,
 0 14222       sdt_p: mmt$max_sdt_p,
 0 14223       st_rma: integer,
 0 14224       sdtx_p: mmt$max_sdtx_p,
 0 14225       segnum: ost$segment,
 0 14226       taskid: ost$global_task_id;
 0 14227
 0 14228     #KEYPOINT (osk$debug, 0, mmk$create_task);
 8 14229
 8 14230     mmp$get_max_sdt_sdtx_pointer (parent_xcb_p, parent_sdt_p, parent_sdtx_p);
 8 14231     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
 8 14232
 8 14233 {  Update the RMA of the task's segment table in the task's exchange package.
 8 14234
 8 14235     i#real_memory_address (sdt_p, st_rma);
8E 14236     xcb_p^.xp.segment_table_address_1 := st_rma DIV 10000(16);
8E 14237     xcb_p^.xp.segment_table_address_2 := st_rma MOD 10000(16);
8E 14238
8E 14239     IF parent_xcb_p^.xp.segment_table_length > xcb_p^.xp.segment_table_length THEN
CC 14240       max_segnum := xcb_p^.xp.segment_table_length;
D0 14241     ELSE
D0 14242       max_segnum := parent_xcb_p^.xp.segment_table_length;
D2 14243     IFEND;
D2 14244
D2 14245 { For performance, try to propagate the ASID/ASTI from the segment table entry of the parent (copy the
D2 14246 { entire st entry) if the parent and child are both using corresponding segments for the same file
D2 14247 { (compare sfids).  If the segments are not being used for the same file, the ASID/ASTI in the child's
D2 14248 { segment table entry will remain zero.  When the child task first page faults for a page of the segment,
D2 14249 { an ASID will be assigned.
D2 14250
D2 14251     taskid := xcb_p^.global_task_id;
D2 14252     FOR segnum := 0 TO max_segnum DO
DC 14253       IF sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry THEN
F0 14254         IF (parent_sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) AND
11E 14255            (sdtx_p^.sdtx_table [segnum].sfid = parent_sdtx_p^.sdtx_table [segnum].sfid) THEN
```

MMP$CREATE_TASK

```
      11E 14256            sdt_p^.st [segnum] := parent_sdt_p^.st [segnum];
      12A 14257          ELSEIF sdtx_p^.sdtx_table [segnum].inheritance = mmc$si_new_segment THEN
      142 14258            gfp$mtr_get_fde_p (sdtx_p^.sdtx_table [segnum].sfid, ijle_p, fde_p);   {No need to lock}
      1A6 14259            fde_p^.global_task_id := taskid;
      1AE 14260          IFEND;
      1AE 14261        IFEND;
      1AE 14262      FOREND;
      1B2 14263
      1B2 14264    PROCEND mmp$create_task;
        0 14265
```


MMP$EXIT_TASK

```
        0 14268 {-------------------------------------------------------------------------------------------------------
        0 14269 {This procedure is called by the dispatcher when a task exits to  free pages and ASIDs assigned
        0 14270 {to task template segments.
        0 14271 {
        0 14272 {     MMP$EXIT_TASK (PARENT_XCB_P, XCB_P)
        0 14273 {
        0 14274 {-------------------------------------------------------------------------------------------------------
        0 14275
        0 14276
        0 14277   PROCEDURE [XDCL] mmp$exit_task
        0 14278     (     xcb_p: ^ost$execution_control_block);
        0 14279
        0 14280     VAR
        0 14281       fde_p: gft$locked_file_desc_entry_p,
        0 14282       sdt_p: mmt$max_sdt_p,
        0 14283       cst_p: ^ost$cpu_state_table,
        0 14284       sdtx_p: mmt$max_sdtx_p,
        0 14285       page_count: integer,
        0 14286       aste_p: ^mmt$active_segment_table_entry,
        0 14287       sva: ost$system_virtual_address,
        0 14288       segnum: ost$segment;
        0 14289
        0 14290
        0 14291     #KEYPOINT (osk$debug, 0, mmk$exit_task);
        8 14292
        8 14293     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
        8 14294     mtp$cst_p (cst_p);
       50 14295
       50 14296     sva.offset := 0;
       50 14297     FOR segnum := 0 TO xcb_p^.xp.segment_table_length DO
       7E 14298       IF sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry THEN
       92 14299         IF sdtx_p^.sdtx_table [segnum].inheritance = mmc$si_new_segment THEN
       AA 14300           sva.asid := sdt_p^.st [segnum].ste.asid;
       AA 14301           IF sva.asid <> 0 THEN
       B6 14302             sdt_p^.st [segnum].ste.asid := 0;
       B6 14303             aste_p := ^mmv$ast_p^ [sdt_p^.st [segnum].asti];
       B6 14304             IF NOT aste_p^.in_use THEN
       D6 14305               mtp$error_stop ('MM - ast not in use');
       F6 14306             IFEND;
       F6 14307             gfp$mtr_get_locked_fde_p (aste_p^.sfid, cst_p^.ijle_p, fde_p);
      17E 14308             fde_p^.asti := 0;
      17E 14309             mmp$mm_free_pages (sva, 7fffffff(16), aste_p, TRUE, page_count);
      1AE 14310           IFEND;
      1AE 14311         IFEND;
      1AE 14312       IFEND;
      1AE 14313     FOREND;
      1B2 14314
      1B2 14315   PROCEND mmp$exit_task;
        0 14316
```

```
        0 14319 {-------------------------------------------------------------------------------------------------------
        0 14320 {This procedure is called by the Dispatcher when a new job is created to initialize the SDT of the
        0 14321 {new job.
        0 14322 {
        0 14323 {    MMP$CREATE_JOB (PARENT_XCB, XCB_P)
        0 14324 {
        0 14325 {! * * * MUST CHANGE WHEN MULTIPLE JOB TEMPLATES ARE SUPPORTED.
        0 14326 {-------------------------------------------------------------------------------------------------------
        0 14327
        0 14328
        0 14329   PROCEDURE [XDCL] mmp$create_job
        0 14330     (     new_job_ajl_ordinal: jmt$ajl_ordinal;
        0 14331           xcb_segnum_relative_jobs_as: ost$segment;
        0 14332           parent_xcb_p: ^ost$execution_control_block;
        0 14333           xcb_p: ^ost$execution_control_block);
        0 14334
        0 14335     VAR
        0 14336       aste_p: ^mmt$active_segment_table_entry,
        0 14337       cst_p: ^ost$cpu_state_table,
        0 14338       fde_p: gft$file_desc_entry_p,
        0 14339       ijle_p: ^jmt$initiated_job_list_entry,
        0 14340       jcb_p: ^jmt$job_control_block,
        0 14341       jf_fde_p: gft$locked_file_desc_entry_p,
        0 14342       new_job_ijl_ordinal: jmt$ijl_ordinal,
        0 14343       parent_fde_p: gft$locked_file_desc_entry_p,
        0 14344       parent_sdt_p: mmt$max_sdt_p,
        0 14345       parent_sdtx_p: mmt$max_sdtx_p,
        0 14346       pfti: mmt$page_frame_index,
        0 14347       sdt_p: mmt$max_sdt_p,
        0 14348       sdtx_p: mmt$max_sdtx_p,
        0 14349       segnum: ost$segment,
        0 14350       sva: ost$system_virtual_address,
        0 14351       taskid: ost$global_task_id;
        0 14352
        0 14353
        0 14354     #KEYPOINT (osk$debug, 0, mmk$create_job);
        8 14355
        8 14356     mtp$cst_p (cst_p);
       18 14357     mmp$get_max_sdt_sdtx_pointer (parent_xcb_p, parent_sdt_p, parent_sdtx_p);
       18 14358     mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
       18 14359
       18 14360 { Copy the segment table entry of the segment used for the new job fixed in the parent to the new job's
       18 14361 { job fixed segment table entry.  Fix the ring and cache bypass values; they are not correct in the
       18 14362 { parent's ste.
       18 14363 { Invalidate the segment used by the parent.
       18 14364
       18 14365     sdt_p^.st [osc$segnum_job_fixed_heap] := parent_sdt_p^.st [xcb_segnum_relative_jobs_as];
       18 14366     sdt_p^.st [osc$segnum_job_fixed_heap].ste.v1 := osc$v1_cache_bypass;
       18 14367     sdt_p^.st [osc$segnum_job_fixed_heap].ste.r2 := 3;
       18 14368     parent_sdt_p^.st [xcb_segnum_relative_jobs_as].ste.v1 := osc$v1_invalid_entry;
       18 14369     parent_sdt_p^.st [xcb_segnum_relative_jobs_as].ste.asid := 0;
       18 14370     gfp$mtr_get_locked_fde_p (parent_sdtx_p^.sdtx_table [xcb_segnum_relative_jobs_as].sfid,
       DC 14371           cst_p^.ijle_p, parent_fde_p);
      162 14372     parent_fde_p^.asti := 0;
      162 14373     sva.asid := sdt_p^.st [osc$segnum_job_fixed_heap].ste.asid;
      162 14374     sva.offset := 0;
```



```
      162 14375     #PURGE_BUFFER (osc$purge_all_page_seg_map, sva);
      180 14376     new_job_ijl_ordinal := jmv$ajl_p^ [new_job_ajl_ordinal].ijl_ordinal;
      180 14377     jmp$get_ijle_p (new_job_ijl_ordinal, ijle_p);
      180 14378
      180 14379 { Copy template segments from the parent (system job task) to the new job's job monitor task.
      180 14380
      180 14381     taskid := xcb_p^.global_task_id;
      180 14382     FOR segnum := 0 TO mmv$max_template_segment_number DO
      1D4 14383       IF (sdt_p^.st [segnum].ste.v1 <> osc$v1_invalid_entry) THEN
      1E8 14384
      1E8 14385 { The second clause of the following IF statement is to prevent the copying of
      1E8 14386 { "shared" segments when executing within a multiple job template.
      1E8 14387
      1E8 14388         IF (sdtx_p^.sdtx_table [segnum].open_validating_ring_number = 0) AND
      20E 14389           (sdtx_p^.sdtx_table [segnum].sfid = parent_sdtx_p^.sdtx_table [segnum].sfid) THEN
      20E 14390           sdt_p^.st [segnum] := parent_sdt_p^.st [segnum];
      21E 14391         ELSE
      21E 14392           gfp$mtr_get_fde_p (sdtx_p^.sdtx_table [segnum].sfid, ijle_p, fde_p);   {No need to lock}
      294 14393           fde_p^.global_task_id := taskid;
      29C 14394         IFEND;
      29C 14395       IFEND;
      29C 14396     FOREND;
      2A2 14397
      2A2 14398 {  Move the job fixed segment of the new job to the job queue of that job.
      2A2 14399
      2A2 14400     aste_p := ^mmv$ast_p^ [sdt_p^.st [osc$segnum_job_fixed_heap].asti];
      2A2 14401     aste_p^.sfid := sdtx_p^.sdtx_table [osc$segnum_job_fixed_heap].sfid;
      2A2 14402     aste_p^.queue_id := mmc$pq_job_fixed;
      2A2 14403     aste_p^.ijl_ordinal := new_job_ijl_ordinal;
      2A2 14404     gfp$mtr_get_locked_fde_p (aste_p^.sfid, ijle_p, jf_fde_p);
      358 14405     jf_fde_p^.last_segment_number := osc$segnum_job_fixed_heap;
      358 14406     jf_fde_p^.global_task_id := xcb_p^.global_task_id;
      358 14407     jf_fde_p^.asti := sdt_p^.st [osc$segnum_job_fixed_heap].asti;
      358 14408     jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, 0);
      358 14409     jcb_p^.next_cyclic_aging_time := #FREE_RUNNING_CLOCK (0) + jcb_p^.cyclic_aging_interval;
      398 14410     ijle_p^.job_fixed_asid := sva.asid;
      398 14411
      398 14412     mmp$initialize_find_next_pfti (sva, 7ffffff0(16), include_partial_pages, psc_all, aste_p, pfti);
      3DA 14413
      3DA 14414   /relink_job_fixed_in_new_queue/
      3DA 14415     WHILE pfti <> 0 DO
      3E2 14416       mmp$relink_page_frame (pfti, mmc$pq_wired);
      3FC 14417       mmv$pft_p^ [pfti].ijl_ordinal := new_job_ijl_ordinal;
      3FC 14418       mmp$relink_page_frame (pfti, mmc$pq_job_fixed);
      438 14419       mmp$find_next_pfti (pfti);
      48C 14420     WHILEND /relink_job_fixed_in_new_queue/;
      498 14421
      498 14422   PROCEND mmp$create_job;
```

MMP$EXIT_JOB

```
 0 14425 {-------------------------------------------------------------------------------------------------------
 0 14426 {This procedure is called by the dispatcher when a job exits to  free pages and ASIDs assigned
 0 14427 {to non-inherited segments.
 0 14428 {
 0 14429 {         MMP$EXIT_JOB (XCB_P)
 0 14430 {
 0 14431 { XCB_P: (input) This parameter is a pointer to the execution control block
 0 14432 {         of the job exiting.
 0 14433 {
 0 14434 {-------------------------------------------------------------------------------------------------------
 0 14435
 0 14436
 0 14437   PROCEDURE [XDCL] mmp$exit_job
 0 14438     (     xcb_p: ^ost$execution_control_block);
 0 14439
 0 14440     VAR
 0 14441       cst_p: ^ost$cpu_state_table;
 0 14442
 0 14443
 0 14444     mtp$cst_p (cst_p);
14 14445     #KEYPOINT (osk$debug, cst_p^.ajlo * osk$m, mmk$exit_job);
34 14446     mmp$free_memory_in_job_queues (cst_p^.ijle_p^.job_page_queue_list, TRUE, FALSE, TRUE);
5E 14447
5E 14448   PROCEND mmp$exit_job;
 0 14449
```


MMP$MTR_LOCK_RING_1_STACK

```
 0 14452
 0 14453 { PURPOSE:
 0 14454 {   This procedure is the monitor part of the process necessary to free a job's ring one stack at termination.
 0 14455 { DESIGN:
 0 14456 {   This procedure changes the ring one stack to a transient file, and returns the disk file descriptor offset
 0 14457 {   (if there is one) to job mode.  Job mode will then free the disk space associated with the ring one stack.
 0 14458
 0 14459
 0 14460   PROCEDURE [XDCL] mmp$mtr_lock_ring_1_stack
 0 14461     (VAR request_block: mmt$rb_lock_ring_1_stack;
 0 14462         cst_p: ^ost$cpu_state_table);
 0 14463
 0 14464     VAR
 0 14465       aste_p: ^mmt$active_segment_table_entry,
 0 14466       count: 1..32,
 0 14467       fde_p: gft$locked_file_desc_entry_p,
 0 14468       found: boolean,
 0 14469       ipti: integer,
 0 14470       pointer {CYBIL trick} : ptr_type,
 0 14471       ste_p: ^mmt$segment_descriptor,
 0 14472       stxe_p: ^mmt$segment_descriptor_extended,
 0 14473       sva: ost$system_virtual_address;
 0 14474
 0 14475     request_block.status.normal := TRUE;
 4 14476     #KEYPOINT (osk$debug, cst_p^.ajlo * osk$m, mmk$mtr_lock_ring_1_stack);
1E 14477
1E 14478 { Verify that the ring 1 stack page (an assumption is made that the stack is on one page) is valid in memory.
1E 14479 { The only way it is not in memory is that it was freed just after the monitor request was issued from ring 1.
1E 14480 { That is unlikely to happen, so if it has simply reissue the request.  This will cause the job to return to
1E 14481 { ring 1, reference and get back its ring 1 stack page, and call monitor again.
1E 14482 { The ring 1 stack must be valid in memory when the file is changed to transient.  If the stack has been
1E 14483 { written to disk and freed, the job will not be able to page fault and get the page back from disk when it
1E 14484 { returns.  Instead a new (zeroed out) page would be assigned, which the job cannot return to.
1E 14485
1E 14486     pointer.pva := cst_p^.xcb_p^.xp.tos_registers [1].pva;
1E 14487
1E 14488     mmp$convert_pva (pointer.p, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
6C 14489     #hash_sva (sva, ipti, count, found);
72 14490     IF NOT found OR NOT mmv$pt_p^ [ipti].v THEN
A0 14491       request_block.status.normal := FALSE;
A0 14492       RETURN;
A8 14493     IFEND;
A8 14494
A8 14495     IF fde_p^.media = gfc$fm_mass_storage_file THEN
B4 14496       request_block.disk_file_descriptor_offset := fde_p^.disk_file_descriptor_p;
B4 14497       dmp$deallocate_file_space (fde_p, 0, amc$file_byte_limit);
D8 14498       fde_p^.media := gfc$fm_transient_segment;
E2 14499     ELSE
E2 14500       request_block.disk_file_descriptor_offset := 0;
E6 14501     IFEND;
E6 14502
E6 14503   PROCEND mmp$mtr_lock_ring_1_stack;
 0 14504
```

MMP$DETERMINE_ERROR_STATE

```
       0 14507   PROCEDURE [XDCL] mmp$determine_error_state (
       0 14508           list_p: ^mmt$rma_list;
       0 14509           list_length: mmt$rma_list_length;
       0 14510     VAR io_error: boolean);
       0 14511
       0 14512     VAR
       0 14513       pfte_p: ^mmt$page_frame_table_entry,
       0 14514       pfti: mmt$page_frame_index,
       0 14515       list_i: mmt$rma_list_index;
       0 14516
       0 14517     io_error := FALSE;
       4 14518
       4 14519     /check_pages/
       4 14520       FOR list_i := 1 TO list_length DO
      1C 14521         IF list_p^ [list_i].length = 0 THEN
      30 14522           EXIT /check_pages/;
      32 14523         IFEND;
      32 14524         pfti := list_p^ [list_i].rma DIV osv$page_size;
      32 14525         pfte_p := ^mmv$pft_p^ [pfti];
      32 14526         IF (pfte_p^.io_error = ioc$media_error) OR
      74 14527               (pfte_p^.io_error = ioc$unrecovered_error) OR
      74 14528               (pfte_p^.io_error = ioc$error_on_init) THEN
      74 14529           io_error := TRUE;
      74 14530           RETURN;
      7A 14531         IFEND;
      7A 14532       FOREND /check_pages/;
      7E 14533
      7E 14534   PROCEND mmp$determine_error_state;
       0 14535
       0 14537 MODEND mmm$monitor_request_processor;
```

**** I=$05578173AS0102D19890821T183254 L=ZZXXLIST B=LGO DA=NONE LO=R RC=NONE OPT=SCHED EL=F LF=CS612 PAD=0

	ERROR	LINE	TEXT
WARNING	CY 821	12012	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	12285	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	12294	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	12300	Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY
   **** 4 warning diagnostics

IDENTIFIER	DEFINED ON LINE	REFERENCES							
active	9179	13508/M							
active_io_count	4510	11428	11502	11574	11577/M	11577	11580	11664	11936/M
		11936	11961/M	11961	12271/M	12271	12449	12450	12452
		12454	12469	12470	12472	12474	12524	12525	12527
		12529	12750	12794	12848	12862	12910	13166	13166
		13191	13328						
active_io_page_count	3911	11737	11742/M	11742	11743	12010/M	12010	12036/M	12036
		12060/M	12060	12255/M	12255	12300/M	12300	12450/M	12450
		12453/M	12453	12470/M	12470	12473/M	12473	12525/M	12525
		12528/M	12528						
active_io_requests	3912	11744/M	11745	11746/M	11746	12011/M	12011	12037/M	12037
		12061/M	12061	12256/M	12256	12325/M	12325		
address	1396	12308/M	12311/M	12316/M					
address_pair_count	1100	12339/M							
age	4514	12929/M							
age_cp_bound_job	10610	13973/M	13973						
age_interval	4567	13911/P	13914/P	13925/P					
age_sys_shared_queue	10619	13920/M	13921						
age_unused_page_in_shared_queue	10618	13917/M	13918						
aggressive_age_job_queues	10608	13932/M	13932						
aggressive_age_shared_queue	10607	13902/M	13902						
aggressive_aging	13757	13899/M	13901	13905	13910	13929/M	13931	13934	13970
aggressive_aging_failed	10609	14033/M	14033						
ajl_ordinal	3905	9338	9375	9502	9636	9819	9840	11488	11490
		11494	11814	13139	13308	13322	13349	13421	13439
		13450	13460	13470	13490	13552	13598	13605	13715
		13976	14003	14049	14075	14144	14152	14157	14159
		14258	14307	14371	14392	14404	14408		
ajl_ordinal	9498	9502/M	9503	9507/S	9507/S	9508			
ajl_ordinal	9805	9819/M	9819	9819/S	9819/S	9819			
ajl_ordinal	9837	9840/M	9841	9845/S	9845/S	9846			
ajl_ordinal	11460	11488/M	11488	11488/S	11488/S	11488			
ajl_ordinal	13272	13322/M	13322	13322/S	13322/S	13322			
ajl_ordinal	13737	14003/M	14003	14003/S	14003/S	14003	14075/M	14075	14075/S
		14075/S	14075						
ajl_ordinal	14111	14144/M	14144	14144/S	14144/S	14144			
ajlo	3813	14445	14476						
ajlo	9495	9505/P	9508/M						
ajlo	9633	9636/M	9637/S	9640/S	9640/S				
ajlo	9805	9819/P	9819/M						
ajlo	9812	9819/P							
ajlo	9834	9843/P	9846/M						
ajlo	11460	11488/P	11488/M						
ajlo	11460	11494/M	11494/S	11494/S	11494/S				
ajlo	11467	11488/P							
ajlo	11765	11814/M	11814/S	11814/S	11814/S				
ajlo	13272	13322/P							
ajlo	13272	13322/P	13322/M						
ajlo	13272	13349/M	13349/S	13349/S	13349/S				
ajlo	13737	13976/M	13976/S	13976/S	13976/S				
ajlo	13737	14003/P	14075/P						
ajlo	13737	14003/P	14003/M	14075/P	14075/M				
ajlo	13744	13937	13939/S	13939/S	13941/S	13941/S	13943/S	13953	13961/S

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		13964/P	13967/P						
ajlo	14111	14144/P							
ajlo	14111	14144/P	14144/M						
ajlo	14111	14152/M	14152/S	14152/S	14152/S	14159/M	14159/S	14159/S	14159/S
allocated_address_pair_count	1101	12238	12334						
allocation_unit_size	413	11491	11491	11492					
amc$access_mode	3089	2614							
amc$average_record_length	3091	2689							
amc$block_type	3092	2616							
amc$character_conversion	3093	2618							
amc$clear_space	3094	2620							
amc$collate_table_name	3096	2691							
amc$compression_procedure_name	3149	2693							
amc$data_padding	3097	2696							
amc$dynamic_home_block_space	3150	2698							
amc$embedded_key	3098	2700							
amc$error_exit_name	3099	2622							
amc$error_limit	3101	2702							
amc$error_options	3102	2624							
amc$estimated_record_count	3103	2704							
amc$file_access_procedure	3104	2626							
amc$file_byte_limit	447	450	452	2895	2933	2969	3203	3265	14497/P
amc$file_contents	3105	2628							
amc$file_limit	3107	2630							
amc$file_organization	3108	2632							
amc$file_processor	3109	2634							
amc$file_structure	3110	2636							
amc$forced_write	3111	2638							
amc$hashing_procedure_name	3151	2706							
amc$index_levels	3117	2708							
amc$index_padding	3118	2710							
amc$initial_home_block_count	3152	2712							
amc$internal_code	3119	2640							
amc$key_length	3120	2714							
amc$key_position	3121	2716							
amc$key_type	3122	2718							
amc$label_exit_name	3123	2642							
amc$label_options	3125	2644							
amc$label_type	3126	2646							
amc$line_number	3127	2648							
amc$loading_factor	3153	2720							
amc$lock_expiration_time	3154	2722							
amc$log_residence	3156	2726							
amc$logging_options	3155	2724							
amc$max_attribute	3197	3201							
amc$max_block_length	3128	2650							
amc$max_block_number	2758	2761							
amc$max_error_count	3081	3084							
amc$max_file_id_ordinal	3066	3073							
amc$max_home_blocks	2933	2936							
amc$max_index_level	2928	2931							
amc$max_key_length	3222	3226							
amc$max_key_position	3231	3228							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES			
amc$max_line_number	2946	2949			
amc$max_lines_per_inch	3289	3286			
amc$max_page_width	2898	2901			
amc$max_path_name_size	2912	2915			
amc$max_record_length	3129	2652			
amc$max_records_per_block	3273	3277			
amc$max_statement_id_length	2996	2999			
amc$max_user_info	3284	3280			
amc$maximum_block	2226	2219	2763	3256	
amc$maximum_keyed_record	3234	3231			
amc$maximum_record	2969	2972	3047	3260	
amc$message_control	3130	2728			
amc$min_block_length	3131	2654			
amc$min_record_length	3132	2656			
amc$null_attribute	3133	2658			
amc$open_position	3134	2660			
amc$padding_character	3135	2662			
amc$page_format	3136	2664			
amc$page_length	3137	2666			
amc$page_width	3138	2668			
amc$preset_value	3140	2670			
amc$record_limit	3141	2730			
amc$record_type	3142	2672			
amc$records_per_block	3143	2732			
amc$return_option	3144	2674			
amc$ring_attributes	3145	2676			
amc$statement_identifier	3146	2678			
amc$user_info	3147	2680			
amc$vertical_print_density	3148	2682			
amt$access_selection	2603	2601			
amt$average_record_length	3047	2690			
amt$block_header_type	2739	2742	2748		
amt$block_number	2761	2744	2751		
amt$block_status	2740	2753			
amt$block_type	3050	2617			
amt$collation_value	3055	3052			
amt$compression_procedure_name	2903	2695			
amt$data_padding	3058	2697			
amt$dynamic_home_block_space	2922	2699			
amt$entry_point_reference	2906	2903	2924		
amt$error_limit	3077	2703			
amt$estimated_record_count	3086	2705			
amt$file_attribute_keys	3201	2608			
amt$file_byte_address	450	411	7718	8462	9314
amt$file_contents	2781	2629			
amt$file_id_ordinal	3073	3070			
amt$file_id_sequence	3074	3071			
amt$file_identifier	3069	3061	3240	10849	
amt$file_item	2607	2603	2606		
amt$file_limit	452	415	2631	13700	
amt$file_organization	3207	2633			
amt$file_position	3210	2594			
amt$file_processor	2842	2635			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
amt$file_structure	2888	2637						
amt$forced_write	3213	2639						
amt$hashing_procedure_name	2924	2707						
amt$index_levels	2931	2709						
amt$index_padding	3217	2711						
amt$initial_home_block_count	2936	2713						
amt$internal_code	3219	2641						
amt$key_length	3226	2715						
amt$key_position	3228	2717						
amt$key_type	3237	2719						
amt$label_options	2596	2645						
amt$label_type	3244	2647						
amt$line_number	2940	2649						
amt$line_number_length	2949	2941						
amt$line_number_location	2951	2942						
amt$loading_factor	2954	2721						
amt$lock_expiration_time	2956	2723						
amt$log_residence	2958	2727						
amt$logging_options	2961	2725						
amt$logging_possibilities	2964	2961						
amt$max_block_length	2763	2556	2651	2743	2749	2750		
amt$max_record_length	2972	2653						
amt$message_control	3254	2729						
amt$min_block_length	3256	2555	2655					
amt$min_record_length	3260	2657						
amt$open_position	2976	2661						
amt$padding_character	3263	2663						
amt$page_format	2891	2665						
amt$page_length	2895	2667						
amt$page_width	2901	2669	2951	3001				
amt$path_name	2915	2908	2958					
amt$preset_value	2980	2671	6081					
amt$record_limit	3265	2731						
amt$record_type	3269	2673						
amt$records_per_block	3277	2733						
amt$return_option	2597	2675						
amt$ring_attributes	2983	2677						
amt$statement_id_length	2999	2991						
amt$statement_id_location	3001	2992						
amt$statement_identifier	2990	2679						
amt$tape_error_action	3011	3006						
amt$tape_error_options	3004	2625						
amt$unused_bit_count	2768	2745	2752					
amt$user_info	3280	2681						
amt$vertical_print_density	3286	2683						
asid	1010	11195/P	12113	12675/P	13147/M	13315/M	13415/M	13433/M 13443/M
		13456/M	13494/M	13562	14143	14150	14164	14193/M 14300/M
		14301	14373/M	14410				
asid	3389	14065	14065					
asid	4978	12429/M	14300	14302/M	14369/M	14373		
asid	12373	12428/P	12429					
asid	13113	13146/P	13147					
asid	13278	13314/P	13315	13361/P	13362/P			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
asid	13383	13432/P	13433	13442/P	13443	13455/P	13456	13463/P	13464/P
		13493/P	13494	13562/M	13573/P				
asid	13775	14043/P	14044/P	14045/P					
asid	14112	14143	14143						
asid_can_be_deleted	13384	13620/M	13622	13641/M	13645				
asid_check	14148	14148	14153	14155					
assign_active	5005	11795/M							
ast_entry	4481	12107/M							
ast_search	13533	13533	13541	13545					
aste_p	4515	11196	11382	11383	11488/P	11490/P	11497	11956/P	11965/P
		11987/P	12012	12038	12107	12468/P	12521	12522	12523/P
		12924/P	13220/M	13233/M	13321	13321	13559	13560	13563
		14065	14065	14066	14067	14139	14157/P		
aste_p	11192	11195/P	11196						
aste_p	12374	12389/M	12393	12394	12424/M	12430/M	12432/M	12435/M	12443
aste_p	12545	12551/P							
aste_p	12579	12594	12594	12598/P					
aste_p	12631	12643/P	12661	12662/P	12675/P				
aste_p	12701	12722/P	12768						
aste_p	12896	12906/P							
aste_p	12969	12993/P	12996	12996	12997	13002/P	13007/P	13009/P	
aste_p	13279	13317/M	13318/P	13359	13360	13361/P	13362/P		
aste_p	13385	13424/M	13425	13444/M	13457/M	13475/M	13476	13495/M	13522/M
		13528/M	13535	13540/M	13547	13551/M	13563/M	13572	13573/P
		13634/P	13647/P						
aste_p	13742	14041/M	14042	14042	14047	14048	14049/P		
aste_p	14286	14303/M	14304	14307/P	14309/P				
aste_p	14336	14400/M	14401/M	14402/M	14403/M	14404/P	14412/P		
aste_p	14465	14488/P							
asti	410	10123	10127/M	12389/S	12427	12428/P	13140	13140/M	13309
		13309/M	13422	13422/M	13440	13440/M	13451	13451/M	13461
		13461/M	13471	13471/M	13491	13491/M	13553/M	13599	13606
		13606/M	13646/M	14050	14308/M	14372/M	14407/M		
asti	4947	12427/M	14303/S	14400/S	14407				
asti	7706	13523	13523	13524/S	13525/S	13527/S	13528/S	13539/M	13553
asti	10120	10123/M	10124/S	10124/S	10125/S	10126/M			
asti	13065	13140/M	13140/S	13140/S	13140/S	13140/M			
asti	13115	13140/P	13141	13146/P	13157/S				
asti	13272	13309/M	13309/S	13309/S	13309/S	13309/M			
asti	13280	13309/P	13310	13314/P	13317/S				
asti	13377	13422/M	13422/S	13422/S	13422/S	13422/M	13440/M	13440/S	13440/S
		13440/S	13440/M	13451/M	13451/S	13451/S	13451/S	13451/M	13461/M
		13461/S	13461/S	13461/S	13461/M	13471/M	13471/S	13471/S	13471/S
		13471/M	13491/M	13491/S	13491/S	13491/S	13491/M	13606/M	13606/S
		13606/S	13606/S	13606/M					
asti	13386	13422/P	13423	13424/S	13432/P	13440/P	13441	13442/P	13444/S
		13451/P	13452	13452/S	13455/P	13457/S	13461/P	13462	13463/P
		13471/P	13472	13475/S	13491/P	13492	13493/P	13495/S	13537
		13538/S	13538/S	13539	13540/S	13606/P	13607	13608/S	13609/S
asti	13776	14040	14041/S	14043/P	14044/S	14045/S	14050		
attach_count	405	13310							
available_memory	9900	9903/M	9906	9909/P					
available_memory	13737	13897/M	13897	13897/P					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
b	9368	9377	9377						
b	9412	9422	9423						
b	9492	9501	9501						
b	9575	9582	9583						
b	9629	9635	9635						
b	9805	9818	9818						
b	11460	11488	11488	11494	11494				
b	11765	11814	11814						
b	13065	13139	13139						
b	13272	13308	13308						
b	13272	13322	13322	13349	13349				
b	13377	13421	13421	13439	13439	13450	13450	13470	13470
		13490	13490	13552	13552	13598	13598	13605	13605
b	13688	13715	13715						
b	13737	13938	13938	13976	13976	14003	14003	14075	14075
b	13737	14049	14049						
b	14111	14144	14144	14152	14152	14159	14159		
b	14111	14157	14157						
b	14277	14307	14307						
b	14329	14371	14371	14404	14404				
bc	9492	9501/M	9501	9501					
bc	9576	9579/M	9580	9584					
bc	9629	9635/M	9635	9635					
bc	9805	9818/M	9818	9818					
bc	11460	11488/M	11488	11488	11494/M	11494	11494		
bc	11765	11814/M	11814	11814					
bc	13272	13322/M	13322	13322	13349/M	13349	13349		
bc	13737	13938/M	13938	13938	13976/M	13976	13976	14003/M	14003
		14003	14075/M	14075	14075				
bc	14111	14144/M	14144	14144	14152/M	14152	14152	14159/M	14159
		14159							
bkw	3465	11289/M	11291/M	11584/M	11584/M	12065	12082	12125	12517
		12520	12664	12668	13505/M	13505/M	14135	14142	
block_index	943	9475/S	9814/S	11309/S	11389/S	11443/S	11558/S	11707/S	12009/S
		12035/S	12057/S	12253/S	12298/S	12399/S	12436/S	12515/S	12662/S
		13322/S	13564/S	13707/S	13803/S	14003/S	14075/S	14144/S	14377/S
block_number	942	9475/S	9485/S	9814/S	11309/S	11389/S	11443/S	11558/S	11707/S
		12009/S	12035/S	12057/S	12253/S	12298/S	12395/S	12399/S	12436/S
		12515/S	12662/S	13322/S	13564/S	13707/S	13803/S	14001/S	14003/S
		14074/S	14075/S	14144/S	14377/S				
block_p	10543	9475	9485	9814	11309	11389	11443	11558	11707
		12009	12035	12057	12253	12298	12395	12399	12436
		12515	12662	13322	13564	13707	13803	14001	14003
		14074	14075	14144	14377				
block_transfer_length	2234	12220							
blocked	10326	13848/M	13875/M	13965					
boffset	11468	11478/M	11480/M	11491/M	11492	11501			
buffer_area	2236	12217							
buffer_area	3346	12224							
buffer_descriptor	11877	11912	11915	12007	12056				
buffer_descriptor_type	972	11912							
byte_offset	7732	13261/M							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES
check_pages	14519	14519 14522 14532
clear	9567	9510/M 9545/M 9642/M 9820/M 11488/M 11494/M 11814/M 13322/M 13349/M 13942/M 13976/M 13978/M 14003/M 14075/M 14144/M 14152/M 14159/M
clock	13746	13832/M 13833 13849 13870 13874 13884 13885 13890/M 13891 13892
cmc$central_memory_element	8990	8866 9096
cmc$central_processor_element	8991	8855
cmc$channel_adapter_element	8992	8859 9101
cmc$choose_any_pp	9033	9004
cmc$choose_pp_by_barrel	9034	9006
cmc$choose_pp_by_channel	9033	9008
cmc$choose_specific_pp	9034	9011
cmc$communications_element	8995	8859 9101
cmc$controller_element	8991	8860 9102
cmc$data_channel_element	8992	8857 9098
cmc$external_processor_element	8995	8860
cmc$iou_element	8993	8866
cmc$mainframe_element	8993	8866
cmc$max_equipment_per_channel	7880	7883
cmc$max_esm_size	8057	9052 9057
cmc$max_low_speed_port_number	9060	9044
cmc$max_pp_per_iou	1667	1558
cmc$max_side_door_port_number	9061	8816 8837 8838 9045
cmc$max_units_per_controller	8982	8985
cmc$mt5680_xx	3295	1059
cmc$mt5698_xx	3296	1056
cmc$mt698_xx	3296	1052
cmc$mt7021_3x	3294	1052
cmc$mt7021_4x	3295	1052
cmc$mt7221_1	3296	1052
cmc$mt7221_2_so	3295	1052
cmc$on	3850	13835
cmc$pem_element	8993	8867
cmc$pp_element	8994	8863
cmc$storage_device_element	8994	8861
cmt$central_memory_port_number	9112	9097
cmt$channel_descriptor	8873	8858
cmt$channel_identification	8963	8956 9010
cmt$channel_ordinal	8887	8879 8964
cmt$controller_type	3293	1051 2554
cmt$element_name	7877	7854 8802 8803 8813 8827 8843 8844 8856 8874 8877 8946 8955 8965 9019 9040 9093 9099 9100
cmt$element_reservation	8853	8821 9121
cmt$element_state	3850	3809 3810 3840
cmt$element_type	8990	8854 9095
cmt$esm_maintenance_buffer_loc	9051	8819 8833 9047
cmt$esm_memory_size	9057	9046
cmt$hardware_address	8953	8948
cmt$model_number	9073	9068
cmt$peripheral_descriptor	8943	8862

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES
cmt$physical_address_parts	8977	8974
cmt$physical_address_specifier	8974	8954
cmt$physical_equipment_number	7883	7864 8957
cmt$physical_unit_number	8985	8958
cmt$pp_identification	9017	9002 9012
cmt$pp_ordinal	9023	9018
cmt$pp_reservation	8998	8864
cmt$pp_reservation_choices	9033	9003
cmt$product_identification	9065	9041
cmt$product_number	9071	9066
cmt$serial_number	9076	9042
cmt$upline_connection	9079	9044 9045
command_index	12178	12212 12217/S 12220/S 12223/S 12224/S 12308/S 12309/S 12311/S 12316/S 12317/S
condition	1786	10191/M 12048/M 12265/M 12871/M 12874/M 12877/M 12998/M 13000/M 13011/M 13142/M 13185/M 13192/M 13241/M 13615/M
condition	10186	10191
condition	11876	12048
condition	12173	12265
condition	12835	12871 12874 12877
condition	12959	12998 13000 13011
condition	13065	13142 13185 13192 13241
condition	13377	13615
contents	5625	11782
contiguous_pages	12636	12663/M 12667/M 12667 12671 12672
controls	10320	13840 13848/M 13849/M 13850/M 13863 13867/M 13870/M 13873 13875/M 13883/M 13884/M 13965
conv	9289	9297/M 9298
convert	9279	9289
count	4563	11294/M 11294 11310 11311 11311 11584/M 11584 13505/M 13505 13803 13803 13803 13805 13806 13808
count	9564	9501/M 9501 9510 9510/M 9510 9542 9543/M 9543 9586/M 9586 9635/M 9635 9642 9642/M 9642 9818/M 9818 9820 9820/M 9820 11488/M 11488 11488 11488/M 11488 11494/M 11494 11494 11494/M 11494 11814/M 11814 11814 11814/M 11814 13322/M 13322 13322 13322/M 13322 13349/M 13349 13349 13349/M 13349 13938/M 13938 13942 13942/M 13942 13976/M 13976 13976 13976/M 13976 13978 13978/M 13978 14003/M 14003 14003 14003/M 14003 14075/M 14075 14075 14075/M 14075 14144/M 14144 14144 14144/M 14144 14152/M 14152 14152 14152/M 14152 14159/M 14159 14159 14159/M 14159
count	11537	11752/M 11753 11753 11755/M 11755
count	12633	12641/M 12657/M 12657
count	13116	13210
count	14466	14489
count_removed	12375	12441/M 12463/P 12464 12481/P
cp_time	4245	13948 13950 13950
cptime	13747	13948/M 13950/M 13972
cptime_next_age_working_set	4612	13972
cpu_idle_statistics	3834	13836 13837
cpus	10853	11932
cst_p	10173	10175/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cst_p	12369	12426/P							
cst_p	12576	12590/M							
cst_p	12583	12590/P	12594	12615/P					
cst_p	12697	12721/M							
cst_p	12712	12721/P	12769						
cst_p	12835	12843/M							
cst_p	12841	12843/P	12851/P	12865/P					
cst_p	12961	12987	12993/P						
cst_p	13067	13139/P	13140/P	13154/P	13251				
cst_p	13274	13297/P	13308/P	13309/P	13339/P				
cst_p	13379	13421/P	13422/P	13439/P	13440/P	13450/P	13451/P	13452	13460/P
		13461/P	13470/P	13471/P	13489/P	13490/P	13491/P	13552/P	13598/P
		13600/P	13605/P	13606/P					
cst_p	14277	14294/M							
cst_p	14283	14294/P	14307/P						
cst_p	14329	14356/M							
cst_p	14337	14356/P	14371/P						
cst_p	14437	14444/M							
cst_p	14441	14444/P	14445	14446/P					
cst_p	14462	14476	14486	14488/P					
cyclic_age	4507	12930/M							
cyclic_aging_interval	4618	14409							
data	9290	9298/M	9300/S	9301/S					
data_index	9291	9299	9300/S	9301/S					
deadstart_upper	11032	13501	14188						
decrement_inhibit_swap	11538	11555/M	11715/M	11741					
delayed_swapin_work	3935	11405/M	11405	12485/M	12485				
delete_pt_entry_ok	11771	11784/M	11798/M	11820					
desc	9271	9295	9296						
destination_aste_p	13117	13154/P	13220	13226/P	13228	13252/P			
destination_pfte_p	13118	13219/M	13220/M	13221/M	13226/P	13233/M	13234/M	13235/P	13239/S
		13240/S	13254/S	13255/S					
destination_pfti	13119	13212/M	13213/P	13214/P					
destination_pti	13120	13210	13212/S						
destination_pva	7731	13149/P	13154/P						
destination_ste_p	13121	13154/P							
destination_stxe_p	13122	13155/P							
destination_sva	13123	13154/P	13210	13221	13226/P	13229			
determine_move_page_status	13071	13109	13170						
dfc$active	8312	8281							
dfc$awaiting_recovery	8313	8288							
dfc$command_record_bytes	4005	4013	8226						
dfc$deactivated	8312	8297							
dfc$division_overwrite_words	3992	4020							
dfc$esm_command_record_size	4013	4021							
dfc$esm_connection	8256	7852	8253						
dfc$esm_header_record_size	4014	4021							
dfc$esm_maintenance_buf_size	3993	4024							
dfc$esm_memory_base_shift	3999	4021	4022	4022					
dfc$header_record_bytes	4004	4014							
dfc$inactive	8312	8288							
dfc$max_data_record_bytes	4008	8849							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dfc$max_esm_divisions	4002	7881							
dfc$max_esm_memory_size	3994	4023	8814	8828					
dfc$max_number_of_mainframes	4001	3986	7890						
dfc$max_number_of_queues	8322	7938	7940	7942	8327				
dfc$max_queue_entries	8323	8070	8071	8112	8113	8328			
dfc$max_req_timeout_count_value	8207	8085	8127						
dfc$max_request_buffer_entries	7887	8774							
dfc$max_retransmit_count_value	8211	8086	8128						
dfc$maximum_lifetime	8387	8384							
dfc$maximum_queue_interfaces	9125	9128							
dfc$maximum_user_buffer_area	8334	8344	8345						
dfc$maximum_user_data_area	8338	8347	8348						
dfc$min_cdcnet_errors	308	314	316	319	322	325	328	331	334
		337	340	343	363				
dfc$min_data_record_bytes	4009	4020	8849						
dfc$min_driver_test_errors	303	351	354	357	360				
dfc$min_ecc	13	19	22	25	28	31	35	39	42
		45	48	51	54	57	60	63	66
		69	72	76	79	82	86	89	92
		95	98	101	104	107	110	113	116
		119	122	125	129	133	136	139	143
		146	149	152	155	158	161	164	167
		171	174	178	181	184	187	190	193
		196	200	204	208	211	214	217	221
		225	228	231	234	237	240	244	247
		251	254	257	260	263	266	270	273
		276	280	284	289	292	296	303	305
		308	310						
dfc$min_esm_division_size	4019	4023							
dfc$min_esm_memory_size	3995	8814	8828						
dfc$min_mm_recovery_errors	310	366	370	373					
dfc$mock_connection	8257	8253							
dfc$monitor	8317	8136							
dfc$monitor_allocate	8097	8087							
dfc$monitor_io	8097	8087							
dfc$queue_assignment_strng_size	8112	8073							
dfc$recovering	8313	8302							
dfc$task_services	8317	8146							
dfc$terminated	8313	8288							
dfc$unrecovered_disk_error	8513	8541							
dfe$server_has_terminated	161	12877/P							
dft$allocated_command_buffer	8221	8220							
dft$allocated_data_rma_list	8182	8181							
dft$allocated_monitor_buffer	8245	8244							
dft$channel_definition	8842	8816	8838						
dft$channel_specification	8801	7855	7856						
dft$connection_address	7987	7982	7983						
dft$connection_descriptor	7981	7968							
dft$connection_flags	7995	7988							
dft$connection_type	8256	7851	8074						
dft$cpu_queue	8061	7955							
dft$cpu_queue_entries	8066	8063							
dft$cpu_queue_entry	8117	8066							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dft$cpu_queue_header	8069	8062						
dft$cpu_queue_pva_entries	7942	7928						
dft$cpu_queue_pva_entry	7954	7943						
dft$data_descriptor	8042	8009	8010	8011				
dft$dma_adapter	7931	7918						
dft$driver_queue	7959	7951						
dft$driver_queue_entries	8000	7961						
dft$driver_queue_entry	8002	8000						
dft$driver_queue_header	7964	7960						
dft$driver_queue_header_flags	7971	7965						
dft$driver_queue_pva_entries	7940	7927						
dft$driver_queue_pva_entry	7950	7941						
dft$driver_queue_rma_entries	7938	7926						
dft$driver_queue_rma_entry	7945	7939						
dft$esm_base_addresses	7907	7900	8817	8831				
dft$esm_definition_table_entry	8812	8809	8822					
dft$esm_pp_information	7862	7857	7858					
dft$inquiry_message	8728	8719	8782					
dft$inquiry_tracer	8733	8729						
dft$interrupt	7976	7966						
dft$lifetime	8384	8380						
dft$mainframe_set	3986	3936	3937	4049	4050			
dft$maximum_data_bytes	8849	8818	8832					
dft$monitor_io_types	8097	8141						
dft$p_allocated_data_rma_list	8181	8093						
dft$p_command_buffer	8219	8130	8131					
dft$p_data_rma_list	8168	8133						
dft$p_queue_interface_table	7895	7848						
dft$p_send_data	8348	8148	8149					
dft$partner_status	8271	8078						
dft$pp_element_reservations	9121	7866						
dft$pp_status	7869	7863						
dft$q_interface_directory_entry	7846	7844						
dft$queue_directory	7917	7902						
dft$queue_directory_index	9128	7832						
dft$queue_entry_flags	8014	8003	8125					
dft$queue_entry_index	8328	7834						
dft$queue_entry_location	7831	7821						
dft$queue_entry_type	8317	8135						
dft$queue_index	8327	7833						
dft$queue_interface_directory	7843	7841						
dft$queue_interface_table	7897	7895						
dft$request_buffer	8769	8765						
dft$request_buffer_directory	8758	7899						
dft$request_buffer_entries	8774	8771						
dft$request_buffer_entry	8779	8775						
dft$request_buffer_entry_flags	8787	8780						
dft$response_flags	8706	8698						
dft$response_parameter	8716	8700						
dft$retransmission_digit	8739	8735						
dft$rpc_progress_record	8359	8151						
dft$send_data_size	8347	8150	8156	8158	8159	8362	8363	8368
dft$send_parameter_size	8344	8367						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dft$server_iocb_error_condition	8478	8464							
dft$server_lifetime	8380	8079							
dft$server_state	8312	8280	8315						
dft$side_door_ports	8836	8830							
dft$transaction_data	8100	8092							
dft$transaction_digit	8738	8734							
dft$transaction_state	8426	8126	8433	8730					
dfv$file_server_debug_enabled	9309	13172							
disk_file_descriptor_offset	3598	14496/M	14500/M						
disk_file_descriptor_p	425	14496							
display_integer_monitor	9270	9305	13173						
display_line	9264	9266/P							
display_line	9304	9304/P							
display_monitor	9263	9267	9304						
dm_reject_table	14124	14150	14164/M						
dmc$a2	6115	6119							
dmc$device_manager_error_code	6132	6133	6136	6139	6142	6145	6148	6151	6154
		6157	6160	6163	6166	6169	6172	6175	6178
		6181	6184	6187	6190	6193	6196	6199	6202
		6205	6208	6211	6214	6217	6220	6223	6226
		6229	6232	6235	6238	6241	6244	6247	6250
		6253	6256	6259	6262	6265	6268	6271	6274
		6277	6280	6283	6286	6289	6292	6295	6298
		6301	6304	6307	6310	6313	6316	6319	6322
		6325	6328	6331	6334	6337	6340	6343	6346
		6349	6352	6355	6358	6361	6364	6367	6370
		6373	6376	6379	6382	6385	6388	6391	6394
		6397	6400	6403	6406	6409	6412	6415	6418
		6421	6424	6427	6430	6433	6436	6439	6442
		6445	6448	6451	6454	6457	6460	6463	6466
		6469	6472	6475	6478	6481	6484	6487	6490
		6493	6496	6499	6502	6505	6508	6511	6514
		6517	6520	6523	6526	6529	6532	6535	6538
		6541	6544	6547	6550	6553	6556	6559	6562
		6565	6568	6571	6574	6577	6580	6583	6586
		6589	6592	6595	6598	6601	6604	6607	6610
		6613	6616	6619	6622	6625	6628	6631	6634
		6637	6640	6643	6646	6649	6652	6655	6658
		6661	6664	6667	6670	6673	6676	6679	6682
		6685	6688	6691	6694	6700	6703	6706	6709
		6712	6715	6718	6721	6724	6727	6730	6733
		6736	6739	6743	6746	6749	6754	6757	6760
		6763	6766	6769	6772	6775	6778	6781	
dmc$max_bytes_per_allocation	6111	6084	6104	6106	6107				
dmc$max_transfer_size	386	381	11477	11480	11482	11483			
dmc$min_bytes_per_allocation	6110	6105							
dmp$deallocate_file_space	9312	14497							
dmt$chapter_info	6079	11471							
dmt$chapter_number	6073	6064							
dmt$global_file_name	8545	8457							
dmt$page_status	6090	6080							
dmt$system_file_id	3588	3579	3968	6063	7700	7704	7705	7713	7717
		8140	8459	9180	10302				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dp	13759	13847	13848/S	13849/S	13850/S	13850/S	13851/S	13862	13863
		13863/S	13864/S	13865/S	13867/S	13868/S	13868/S	13870/S	13873/S
		13875/S	13877						
dpc$console_row_size	9260	9254							
dpc$top_line_message_size	9254	9249							
dpp$display_error	9248	9266	9304						
dsc$max_number_of_ious	1671	1558							
dsw_job_shared_asid_changed	12376	12485							
entry_status	3904	11717	13960						
entry_updated	4482	12113/M							
eoffset	11469	11483/M	11485/M	11492/M	11501				
exclude_partial_pages	9883	12643/P	12906/P						
fde_p	9322	9344/M	9345						
fde_p	9368	9375/M	9375						
fde_p	9370	9375/P	9377/P						
fde_p	10076	10082	10083						
fde_p	10117	10123	10127/M						
fde_p	11460	11490/M	11490						
fde_p	11470	11490/P	11491	11491	11492				
fde_p	12368	12389/S	12427	12428/P	12430/P				
fde_p	12700	12730/P							
fde_p	12964	12993/P	12999	13002/P					
fde_p	13065	13139/P	13139/P						
fde_p	13065	13139/M	13139						
fde_p	13065	13140	13140/M						
fde_p	13065	13170							
fde_p	13073	13101							
fde_p	13124	13139/P	13140/P	13154/P	13170/P				
fde_p	13272	13308/P	13308/P						
fde_p	13272	13308/M	13308						
fde_p	13272	13309	13309/M						
fde_p	13281	13308/P	13309/P	13310					
fde_p	13377	13421/P	13421/P	13439/P	13439/P	13450/P	13450/P	13470/P	13470/P
		13490/P	13490/P	13552/P	13552/P	13598/P	13598/P	13605/P	13605/P
fde_p	13377	13421/M	13421	13439/M	13439	13450/M	13450	13460/M	13460
		13470/M	13470	13490/M	13490	13552/M	13552	13598/M	13598
		13605/M	13605						
fde_p	13377	13422	13422/M	13440	13440/M	13451	13451/M	13461	13461/M
		13471	13471/M	13491	13491/M	13606	13606/M		
fde_p	13387	13421/P	13422/P	13425	13439/P	13440/P	13450/P	13451/P	13460/P
		13461/P	13470/P	13471/P	13490/P	13491/P	13552/P	13553/M	13598/P
		13599	13600/P	13605/P	13606/P	13611/M	13622	13634/P	13646/M
fde_p	13688	13715/P	13715/P						
fde_p	13688	13715/M	13715						
fde_p	13697	13715/P	13716	13718/P					
fde_p	13737	14049/P	14049/P						
fde_p	13737	14049/M	14049						
fde_p	13741	14049/P	14050						
fde_p	14111	14157/P	14157/P						
fde_p	14111	14157/M	14157						
fde_p	14118	14157/P	14158/P						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
fde_p	14211	14258/M	14258						
fde_p	14217	14258/P	14259/M						
fde_p	14277	14307/P	14307/P						
fde_p	14277	14307/M	14307						
fde_p	14281	14307/P	14308/M						
fde_p	14329	14371/P	14371/P	14404/P	14404/P				
fde_p	14329	14371/M	14371	14392/M	14392	14404/M	14404		
fde_p	14338	14392/P	14393/M						
fde_p	14467	14488/P	14495	14496	14497/P	14498/M			
file_entry_index	3447	9334	9375	11490	13139	13308	13421	13439	13450
		13460	13470	13490	13552	13598	13605	13715	14049
		14157	14258	14307	14371	14392	14404		
file_hash	408	9345	9375	11490	13139	13308	13421	13439	13450
		13460	13470	13490	13552	13598	13605	13715	14049
		14157	14258	14307	14371	14392	14404		
file_hash	3449	9333	9345	9375	9375	11490	11490	13139	13139
		13308	13308	13421	13421	13439	13439	13450	13450
		13460	13460	13470	13470	13490	13490	13552	13552
		13598	13598	13605	13605	13715	13715	14049	14049
		14157	14157	14258	14258	14307	14307	14371	14371
		14392	14392	14404	14404				
file_limit	415	13716							
file_sfid	7717	13470/P	13471/P						
first_image_pfti	13388	13501/M	13502						
first_pfti	12637	12664/M	12665/S	12666/S	12666/S	12668/M	12668/S		
flags	401	10083	13622						
found	11885	11926	11927						
found	12179	12243	12244						
found	13125	13210	13211						
found	13694	13713/M	13724/M						
found	14468	14489	14490						
free_asid	12632	12645	12674						
fwd	3466	11287/M	11287	11288	11291/S	11293/M	11497	11529	11584/M
		11584	11584	11584/S	11584/M	12443	12457	13162	13177
		13476	13483	13505/M	13505	13505	13505/S	13505/M	
get_system_jobs_working_set	11303	11313	13803						
gfc$fde_size	9361	9334	9375	11490	13139	13308	13421	13439	13450
		13460	13470	13490	13552	13598	13605	13715	14049
		14157	14258	14307	14371	14392	14404		
gfc$fde_table_base	9359	9334	9360	9375	11490	13139	13308	13421	13439
		13450	13460	13470	13490	13552	13598	13605	13715
		14049	14157	14258	14307	14371	14392	14404	
gfc$fk_catalog	487	499							
gfc$fk_job_local_file	489	498							
gfc$fm_mass_storage_file	502	424	14495						
gfc$fm_served_file	503	427							
gfc$fm_transient_segment	502	12999	14498						
gfc$monitor_interlocks	9389	9376	13139	13308	13421	13439	13450	13470	13490
		13552	13598	13605	13715	14049	14157	14307	14371
		14404							
gfc$qs_global_shared	543	13611							
gfc$qs_job_shared	543	13425							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gfc$tr_job	3457	9337	9375	10125	11490	12012	12038	13139	13140
		13308	13309	13421	13422	13439	13440	13450	13451
		13460	13461	13470	13471	13490	13491	13552	13598
		13605	13606	13715	14049	14157	14258	14307	14371
		14392	14404						
gfc$tr_system	3457	9336	9375	11383	11490	13139	13308	13421	13439
		13450	13460	13470	13490	13552	13598	13605	13715
		14048	14049	14157	14258	14307	14371	14392	14404
gfc$tr_system_wait_recovery	3457	13560							
gfp$mtr_get_fde_p	9320	9349	9375	11490	13139	13308	13421	13439	13450
		13460	13470	13490	13552	13598	13605	13715	14049
		14157	14258	14307	14371	14392	14404		
gfp$mtr_get_locked_fde_p	9368	9380	13139	13308	13421	13439	13450	13470	13490
		13552	13598	13605	13715	14049	14157	14307	14370
		14404							
gft$allocation_unit_size	458	413							
gft$attach_count	463	404	405						
gft$fde_flags	433	401							
gft$file_desc_entry_p	390	9322	9971	11470	14217	14338			
gft$file_descriptor_entry	398	390	403	824					
gft$file_descriptor_index	472	3447							
gft$file_kind	483	407	495						
gft$file_media	502	423							
gft$locked_file_desc_entry_p	824	9313	9370	9730	9754	10076	10117	10136	10153
		12368	12700	12984	13073	13124	13281	13387	13697
		13741	14118	14281	14341	14343	14467		
gft$open_count	532	406	548						
gft$queue_status	543	416							
gft$segment_lock_info	547	409							
gft$signature_lock	508	399							
gft$system_file_identifier	3446	3436	3588	4836	4996	5198	7726	9320	9368
		10118	10826	13397					
gft$table_residence	3457	3448	9330						
gft$transfer_unit_size	469	414							
global_task_id	420	14259/M	14393/M	14406/M					
global_task_id	4797	13706/P	14251	14381	14406				
global_template_file	437	10083	13622						
hash	9327	9333/M	9340/M						
hash	9368	9375/M	9375/M						
hash	11460	11490/M	11490/M						
hash	13065	13139/M	13139/M						
hash	13272	13308/M	13308/M						
hash	13377	13421/M	13421/M	13439/M	13439/M	13450/M	13450/M	13460/M	13460/M
		13470/M	13470/M	13490/M	13490/M	13552/M	13552/M	13598/M	13598/M
		13605/M	13605/M						
hash	13688	13715/M	13715/M						
hash	13737	14049/M	14049/M						
hash	14111	14157/M	14157/M						
hash	14211	14258/M	14258/M						
hash	14277	14307/M	14307/M						
hash	14329	14371/M	14371/M	14392/M	14392/M	14404/M	14404/M		
hash_count	11886	11926							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
hash_count	12180	12243							
head	558	10229	10230/S	11429	11429	11665	11665	11753	11753
		11786	11786/S	14022					
hex_digits	9275	9300	9301						
highest_offset	7718	13469/M	13480	13481/M					
i	13760	13834	13835/S	13836/S	13837/S				
i	14125	14130/M	14149	14163/M	14163	14164/S	14165		
i	14182	14186	14187/S						
i#program_error	9365	9346	9375	9510	9540	9642	9820	11488	11490
		11494	11814	13139	13308	13322	13349	13421	13439
		13450	13460	13470	13490	13552	13598	13605	13715
		13942	13976	13978	14003	14049	14075	14144	14152
		14157	14159	14258	14307	14371	14392	14404	
i#real_memory_address	9434	12305	14235						
id	9565	9501	9501/M	9510	9539	9580	9584/M	9635	9635/M
		9642	9818	9818/M	9820	11488	11488/M	11488	11494
		11494/M	11494	11814	11814/M	11814	13322	13322/M	13322
		13349	13349/M	13349	13938	13938/M	13942	13976	13976/M
		13976	13978	14003	14003/M	14003	14075	14075/M	14075
		14144	14144/M	14144	14152	14152/M	14152	14159	14159/M
		14159							
identifier	5596	11783/M	11802/M						
identifier	5617	11781/M							
idle_candidates	13761	13859/M	13863						
idle_dispatching_queue_time	10361	13874							
idle_io_active	4754	13837							
idle_no_io_active	4753	13836							
ignore_status	12181	12231/P							
ijl_ordinal	976	12056							
ijl_ordinal	3429	10125	11488/P	12394	12435/M	12594	12662/P	13140	13309
		13422	13440	13451	13452	13461	13471	13491	13606
		13608/M	14001/P	14002/P	14403/M				
ijl_ordinal	3832	12594	12768	13140/P	13251	13309/P	13422/P	13440/P	13451/P
		13452	13461/P	13471/P	13491/P	13606/P			
ijl_ordinal	4508	11388	11442	11706	11939	11940	12009/P	12035/P	12094
		12253/P	12276	12277	12298/P	12455/M	12475/M	12530/M	12768
		13251/M	13322/P	13564/P	14074/P	14075/P	14144/P	14417/M	
ijl_ordinal	7814	11557							
ijl_ordinal	9471	9475/S	9475/S						
ijl_ordinal	9483	9485/S							
ijl_ordinal	9613	14376							
ijl_ordinal	9805	9814/S	9814/S						
ijl_ordinal	9806	9814/P	9819/P						
ijl_ordinal	11303	11309/S	11309/S						
ijl_ordinal	11368	11558/S	11558/S	11707/S	11707/S				
ijl_ordinal	11378	11389/S	11389/S						
ijl_ordinal	11426	11443/S	11443/S						
ijl_ordinal	11539	11388/M	11389/P	11442/M	11443/P	11557/M	11558/P	11706/M	11707/P
ijl_ordinal	11876	12009/S	12009/S	12035/S	12035/S	12057/S	12057/S		
ijl_ordinal	11888	11939/M	11940	12056/M	12057/P	12094			
ijl_ordinal	12173	12253/S	12253/S	12298/S	12298/S				
ijl_ordinal	12183	12276/M	12277						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	12367	12395/S							
ijl_ordinal	12367	12399/S	12399/S	12436/S	12436/S				
ijl_ordinal	12503	12515/S	12515/S						
ijl_ordinal	12628	12662/S	12662/S						
ijl_ordinal	13272	13322/P	13322/P						
ijl_ordinal	13272	13322/S	13322/S						
ijl_ordinal	13377	13564/S	13564/S						
ijl_ordinal	13688	13707/S	13707/S						
ijl_ordinal	13698	13706/P	13707/P						
ijl_ordinal	13737	13803/S	13803/S	14003/S	14003/S	14075/S	14075/S		
ijl_ordinal	13737	14001/S	14074/S						
ijl_ordinal	13737	14003/P	14003/P	14075/P	14075/P				
ijl_ordinal	14111	14144/P	14144/P						
ijl_ordinal	14111	14144/S	14144/S						
ijl_ordinal	14329	14377/S	14377/S						
ijl_p	12638	12662/P	12664	12671/M					
ijle_p	3833	12615/P	13139/P	13308/P	13339/P	13421/P	13439/P	13450/P	13460/P
		13470/P	13490/P	13552/P	13598/P	13605/P	14307/P	14371/P	14446/P
ijle_p	9321	9338							
ijle_p	9368	9375							
ijle_p	9369	9375/P							
ijle_p	9472	9475/M							
ijle_p	9493	9502	9504/P						
ijle_p	9614	13939	13943						
ijle_p	9630	9636	9638/P						
ijle_p	9805	9814/M							
ijle_p	9805	9819	9819/P						
ijle_p	9809	9814/P	9815	9819/P					
ijle_p	9832	9840	9842/P						
ijle_p	11303	11309/M							
ijle_p	11307	11309/P	11310	11310	11311				
ijle_p	11368	11558/M	11707/M						
ijle_p	11378	11389/M							
ijle_p	11426	11443/M							
ijle_p	11460	11488	11488/P						
ijle_p	11460	11490							
ijle_p	11460	11494	11494/P						
ijle_p	11461	11476	11488/P	11490/P	11494/P	11520			
ijle_p	11540	11389/P	11401	11405/M	11405	11414/P	11443/P	11444	11558/P
		11560/M	11707/P	11717	11718	11719/P	11737	11741/M	11741
		11742/M	11742	11743	11744/M	11745	11746/M	11746	11748
		11748	11749/P						
ijle_p	11765	11814	11814/P						
ijle_p	11772	11788/P	11814/P						
ijle_p	11876	12009/M	12035/M	12057/M					
ijle_p	11887	12009/P	12010/M	12010	12011/M	12011	12014/M	12014	12035/P
		12036/M	12036	12037/M	12037	12040/M	12040	12057/P	12059/M
		12059	12060/M	12060	12061/M	12061	12063	12064/M	12065/M
		12065	12066/M	12066	12067/M	12070	12073	12073	12074
		12075/M	12075	12076	12077	12080/M	12081/M	12081	12082/S
		12084	12085/M	12087/M	12087	12091	12091	12091	12091
		12092/S	12104	12104/S	12116	12121	12124/M	12124	12125/M
ijle_p	12173	12253/M	12298/M						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijle_p	12182	12253/P	12254/M	12254	12255/M	12255	12256/M	12256	12298/P
		12299/M	12299	12300/M	12300	12325/M	12325		
ijle_p	12367	12399/M	12436/M						
ijle_p	12377	12399/P	12404	12404	12449/M	12449	12450/M	12450	12460
		12460	12469/M	12469	12470/M	12470	12479	12479	12480
		12481/P	12485/M	12485					
ijle_p	12503	12515/M							
ijle_p	12504	12517	12524/M	12524	12525/M	12525			
ijle_p	12628	12662/M							
ijle_p	13065	13139/P							
ijle_p	13065	13139							
ijle_p	13272	13308/P							
ijle_p	13272	13308							
ijle_p	13272	13322/P	13322	13322/P					
ijle_p	13272	13322/M							
ijle_p	13272	13322	13322/P						
ijle_p	13272	13349	13349/P						
ijle_p	13282	13322/P	13349/P						
ijle_p	13377	13421/P	13439/P	13450/P	13470/P	13490/P	13552/P	13598/P	13605/P
ijle_p	13377	13421	13439	13450	13460	13470	13490	13552	13598
		13605							
ijle_p	13377	13564/M							
ijle_p	13390	13564/P	13565	13565	13577	13578	13579	13580/P	
ijle_p	13688	13707/M							
ijle_p	13688	13715/P							
ijle_p	13688	13715							
ijle_p	13699	13707/P	13715/P						
ijle_p	13737	13803/P	13803	13803	13803				
ijle_p	13737	13803/M	14003/M	14075/M					
ijle_p	13737	13976	13976/P						
ijle_p	13737	14003/P	14003	14003/P	14075/P	14075	14075/P		
ijle_p	13737	14003	14003/P	14075	14075/P				
ijle_p	13737	14049/P							
ijle_p	13737	14049							
ijle_p	13752	13943/M	13944	13945/M	13945	13948	13950	13950	13955/P
		13955/P	13957/P	13957/P	13960	13960	13963	13965/S	13971/P
		13974/P	13976/P	14003/P	14075/P				
ijle_p	14111	14144/P	14144	14144/P					
ijle_p	14111	14144/M							
ijle_p	14111	14144	14144/P						
ijle_p	14111	14152	14152/P	14159	14159/P				
ijle_p	14111	14157/P							
ijle_p	14111	14157							
ijle_p	14122	14144/P	14152/P	14157/P	14159/P				
ijle_p	14211	14258							
ijle_p	14214	14258/P							
ijle_p	14277	14307/P							
ijle_p	14277	14307							
ijle_p	14329	14371/P	14404/P						
ijle_p	14329	14371	14392	14404					
ijle_p	14329	14377/M							
ijle_p	14339	14377/P	14392/P	14404/P	14408	14410/M			
ijlo	9494	9504/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijlo	9805	9819/P							
ijlo	9833	9842/P							
ijlo	10119	10125							
ijlo	11460	11488/P							
ijlo	13065	13140							
ijlo	13272	13309							
ijlo	13272	13322/P							
ijlo	13377	13422	13440	13451	13461	13471	13491	13606	
ijlo	13737	14003/P	14075/P						
ijlo	14111	14144/P							
in_use	3430	10124	13140	13309	13422	13440	13451	13461	13471
		13491	13524	13538	13606	13999	14042	14047	14065
		14067	14139	14304					
in_use	9612	9507/M	9507	9637	9640/M	9640	9819/M	9819	9845/M
		9845	11488/M	11488	11494	11494/M	11494	11814	11814/M
		11814	13322/M	13322	13349	13349/M	13349	13939	13941/M
		13941	13976	13976/M	13976	14003/M	14003	14075/M	14075
		14144/M	14144	14152	14152/M	14152	14159	14159/M	14159
include_pages_in_dump	10078	10085/M	10087/M	10090/M					
include_partial_pages	9883	12551/P	12598/P	12722/P	13318/P	14412/P			
index	519	10229/M	11786/M						
index_p	10556	9475	9485	9814	11309	11389	11443	11558	11707
		12009	12035	12057	12253	12298	12395	12399	12436
		12515	12662	13322	13564	13707	13803	14001	14003
		14074	14075	14144	14377				
inheritance	4997	14257	14299						
inhibit_io	9808	9815/M	9816						
inhibit_io	13272	13322/M	13322						
inhibit_io	13283	13322/P	13323						
inhibit_io	13737	14003/M	14003	14075/M	14075				
inhibit_io	13745	14003/P	14005/M	14008	14075/P	14077/M	14079		
inhibit_io	14111	14144/M	14144						
inhibit_io	14128	14144/P	14145						
inhibit_swap_count	3910	11741/M	11741	11748	12014/M	12014	12040/M	12040	12059/M
		12059	12254/M	12254	12299/M	12299	12449/M	12449	12451/M
		12451	12469/M	12469	12471/M	12471	12524/M	12524	12526/M
		12526							
init_io_error	11548	11381	11552/M	11651	11716				
init_new_io	7678	13634/P	13638/P						
init_new_io	7764	13002/P	13004/P						
init_new_io	12703	12724							
init_pfte_p	11462	11477	11480	11482	11483	11488/P	11490/P	11491	11497
initial_reassignable_now	11541	11553/M	11752						
int	9272	9297							
int	9282	9297/M							
integ	9280	9281							
io_already_active	12707	12717/M	12753/M	12797/M					
io_already_active	12973	13003/P							
io_already_active	13391	13636/P							
io_count	12706	12716/M	12735/M	12735	12806				
io_count	12972	13003/P							
io_count	13392	13636/P							
io_error	4516	11381/M	11433/M	11653	12786	12787	14526	14527	14528

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
io_error	11374	11413	11413	11416/M	11552	11559	11560	11581	11614
		11623/P	11627	11633/P	11637	11642/P			
io_error	11768	11790	11792	11801					
io_error	11890	12045/M	12046/P						
io_error	12184	12229/M	12231/P	12258/M	12260/P				
io_error	14510	14517/M	14529/M						
io_id	12705	12730/P	12734	12752	12796	12805			
io_id	12971	12977/M	13003/P						
io_id	13393	13410/M	13636/P						
io_id	14123	14131/M	14158/P						
io_identifier	11372	11557							
io_pfti	10822	11932							
io_type	1115	12205	12214	12306					
ioc$allocate	1034	7820	11635	11975					
ioc$bid_area_size	2130	1844							
ioc$bid_status_response_length	2125	2130	2134						
ioc$bid_window_length	1292	1301	1302						
ioc$ccc_cart_device_status_size	1158	1161							
ioc$ccc_cart_error_log_size	1160	1163							
ioc$ccc_cart_sense_bytes_size	1159	1162							
ioc$communication_buffer_length	1559	1622							
ioc$detailed_status_length	1448	1490							
ioc$detailed_status_length_b	1446	1448	1495						
ioc$device_status_length	2126	2131							
ioc$disk_detailed_status_length	1449	1450							
ioc$error_on_init	1038	11413	14528						
ioc$explicit_read	1028	11599	11953	12205	12214	12306			
ioc$explicit_read_no_purge	1032	11599	11600	11954					
ioc$explicit_write	1029	11597	11953						
ioc$extended_status_length	2127	2209							
ioc$initialize_sectors	1032	11597	11954						
ioc$ipi_max_status_size	1842	1859							
ioc$ipi_tape_status_size	1841	1844							
ioc$keypoint_io	1032	11581	11925	11993	11999				
ioc$major_status_size	1843	1859							
ioc$max_ccc_cart_error_id	1285	1244							
ioc$max_ipi_retry	3324	1088							
ioc$max_multiple_tape_requests	1414	1094	1120	2502	3305				
ioc$max_number_tape_units	2217	2213							
ioc$max_response_length	1445	1446							
ioc$max_tape_block_length	2219	2222	2254						
ioc$max_tape_blocks_to_process	2218	1089	1090	2222	2252	2442	2552		
ioc$max_tape_io_id	1413	1417							
ioc$max_unit_number	1537	1540							
ioc$media_error	1036	14526							
ioc$min_ccc_cart_resp_size	1161	1162							
ioc$min_response_length	1444	1161	1447	1844					
ioc$min_tape_block_length	2223	2253							
ioc$no_error	1036	11416	11433	11559	11581	11614	11627	11637	11651
		11653	11716	12045	12067	12229	12258		
ioc$no_io	1031	11596	11972	12046/P	12230/P	12259/P			
ioc$pp_count	1558	1049	1430						
ioc$read_ahead_on_server	1034	7822	11625	11973					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ioc$read_cmd_per_block	2407	2442							
ioc$read_for_server	1033	7818	11625	11973					
ioc$read_from_client	1033	7819	11607	11968					
ioc$read_mass_storage	1030	11599	11953						
ioc$read_page	1028	7815	11625	11946	11973				
ioc$read_uft	1030	11599	11953						
ioc$response_buffer_length	1561	1562							
ioc$response_buffer_length_b	1562	1630							
ioc$server_allocation_error	1037	11792							
ioc$server_has_terminated	1037	11801							
ioc$swap_in	1029	7813	11556						
ioc$swap_out	1029	7813	11556	12103					
ioc$tape_completion_code_max	2269	2308							
ioc$tape_max_busy_retry	3328	1085							
ioc$tape_max_chan_parity_retry	3331	3333							
ioc$tape_max_data_transfer	2222	1076							
ioc$tape_max_lateack_retry	3329	1086							
ioc$tape_max_lost_data_retry	3327	1084							
ioc$tape_max_misc_retry	3330	1087							
ioc$tape_max_parity_retry	3333	1083							
ioc$tape_max_tape_parity_retry	3332	3334							
ioc$tape_max_tcu_parity_retry	3326	1082							
ioc$tape_pkt_lng_read	2387	2442							
ioc$unit_commun_buffer_length	1693	1717							
ioc$unit_down_on_init	1038	11413	12787						
ioc$unrecovered_error	1036	14527							
ioc$unrecovered_error_unit_down	1037	11790	12786						
ioc$write_for_server	1034	7818	11644	11690	11977				
ioc$write_locked_page	1031	11590	11644	11645	11714	11943	11984	12013	12039
ioc$write_mass_storage	1031	11597	11953						
ioc$write_page	1028	7815	11644	11714	11943	11977	12013	12039	13002/P
		13634/P	14158/P						
ioc$write_to_client	1034	7819	11598	11690	11959				
ioid	11891	12046/P							
ioid	12185	12230/P	12259/P						
iot$abnormal_status	1509	1481							
iot$alert_conditions	1145	1136	1480	1485					
iot$bid_index	1301	2565	2567						
iot$bid_window	1302	2566	2568						
iot$cartridge_tape_bid	1305	1187	2563						
iot$ccc_cart_device_status	1166	1060							
iot$ccc_cart_error_log	1226	1062							
iot$ccc_cart_sense_bytes	1194	1061							
iot$command	1392	1138	1614	1615					
iot$command_code	1399	1393							
iot$command_length	1407	1395							
iot$communication_buffer	1610	1626							
iot$communication_buffer_length	1629	1579							
iot$detailed_status	1490	1429	1498	1503					
iot$flags	1401	1394							
iot$id24_byte1	2004	1995							
iot$id26_byte1	2015	1996							
iot$id26_byte2	2024	1997							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
iot$id26_byte4	2033	1998							
iot$id2a_byte3	2040	1999							
iot$interface_error_code	1523	1482							
iot$interrupt	1545	1134	1478	7978					
iot$io_error	1036	3969	4516	11374	11548	11768	11890	12184	
iot$io_function	1028	1092	1115	7812	8139	10155	11369	11879	12702
iot$io_id	1417	1073	1107	2495					
iot$io_request	1422	1128	1240	1435	1460	1472	1583	1708	
iot$ipi_major_status	1884	1858							
iot$ipi_major_status_header	1876	1857							
iot$ipi_tape_status	1847	1058							
iot$lock_owner	1680	1677							
iot$lockword	1674	1581	1705	1706					
iot$logical_unit	1540	1132	1476	1567	1568	1604	1697	8701	
iot$mtc_sense_bytes	2050	2000							
iot$no_of_tape_units	2213	1118	2500						
iot$pageable_tape_request_entry	3309	3306							
iot$pageable_tape_requests	3305	2501							
iot$physical_path	1640	1606							
iot$port_number	1550	1547							
iot$pp_number	1652	1566	1688	7919	7920				
iot$pp_response	1470	1050	1428						
iot$priority	1549	1135	1479						
iot$read_block_descriptor	2233	2230							
iot$read_tape_description	2229	1068	1113						
iot$request_length	1141	1080	1131						
iot$request_recovery	1543	1133	1477						
iot$response_buffer	1630	1587	1637						
iot$response_buffer_offset	1635	1593	1595	1597					
iot$response_code	1525	1483							
iot$response_length	1507	1475							
iot$response_processor	1428	1423							
iot$short_response_flags	1463	1459							
iot$special_ipi_status	1883	1856							
iot$tape_bid_status_response	2134	1053	1057						
iot$tape_block_count	2252	2229	2322	3338					
iot$tape_collected_pp_response	1048	1093	1116	3312					
iot$tape_command_heap	2260	1114							
iot$tape_command_index	2439	1069	1070	1072	1104	1106	2449	12178	
iot$tape_completion_codes	2308	2329							
iot$tape_device_status	2137	1054							
iot$tape_extended_status	2209	1055							
iot$tape_format_parameters	2572	1137	2558						
iot$tape_hardware_command	2440	2451							
iot$tape_io_status	2311	1091							
iot$tape_job_unit_descriptor	2494	1077	1119						
iot$tape_pp_request	1126	1096	1123						
iot$tape_request	1066	1117	2502	3311					
iot$tape_request_length	2442	2450							
iot$tape_request_types	2460	1075	1111	2553					
iot$tape_transfer_count	2244	2234							
iot$transfer_count	1553	1071	1105	1486	3344	5274			
iot$unit_commun_buffer_length	1723	1703							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
iot$unit_communication_buffer	1717	1721							
iot$unit_descriptor_entry	1603	1601							
iot$unit_descriptors	1601	1598							
iot$unit_interface_table	1696	1605							
iot$unit_status	1713	1698							
iot$unit_type	1780	1699							
iot$unsolicited_response_code	1531	1484							
iot$wired_tape_request	1099	12174							
iot$write_block_descriptor	3342	3339							
iot$write_tape_description	3338	1067	1112						
iotype	11369	11556	11556	11581	11590	11595	11600	11645	11690
		11690	11714	11714					
iotype	11879	11925	11943	11943	11946	11952	11998	12013	12013
		12039	12039	12103					
iotype	12702	12730/P							
ipti	14469	14489	14490/S						
j	14126	14149	14150/S						
jcb_p	13743	13953/M	13954	13955/P	13956/P	13957/P	13962	13971/P	13972
		13972	13974/P						
jcb_p	14340	14408/M	14409/M	14409					
jf_asid	11892	12070/M	12113						
jf_fde_p	14341	14404/P	14405/M	14406/M	14407/M				
jmc$detached_job_wait_time_max	4637	4634							
jmc$dp_conversion	869	13863	13877						
jmc$dsw_io_error_while_swapped	4041	11406							
jmc$dsw_job_shared_asid_changed	4039	12376							
jmc$examine_input_queue	916	13854/P							
jmc$examine_swapin_queue	917	13853/P							
jmc$highest_det_job_wait_time	4647	4637	4648						
jmc$highest_prio_age_interval	4904	4895	4905						
jmc$highest_sched_memory_level	10484	10477							
jmc$highest_service_accumulator	4362	4363							
jmc$highest_service_factor_valu	4928	4921							
jmc$highest_service_interval	10500	10493							
jmc$highest_working_set_size	4673	4664	4674	4676	4678	4680			
jmc$ies_entry_free	4137	11717							
jmc$ies_job_in_memory	4140	13960							
jmc$ies_job_swapped	4142	4151							
jmc$ies_swapin_in_progress	4141	4150							
jmc$inhibit_memory_manager_io	4194	9815	13322	14003	14075	14144			
jmc$iss_executing	4156	13940	13960						
jmc$iss_free_swapped_memory	4174	11476							
jmc$iss_idle_tasks_initiated	4157	4184							
jmc$iss_initiate_swapout_io	4169	12404	13565						
jmc$iss_job_idle_tasks_complete	4158	12479	13577						
jmc$iss_job_io_complete	4165	11520	11718						
jmc$iss_swapin_io_complete	4182	4185							
jmc$iss_swapin_requested	4178	4185							
jmc$iss_swapout_complete	4177	4184							
jmc$iss_swapout_io_complete	4172	12405	13566						
jmc$iss_swapped_io_cannot_init	4168	4195	12460	12480	13578				
jmc$iss_swapped_io_complete	4173	12461	12480	13579					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$iss_swapped_no_io	4159	4194							
jmc$iss_wait_job_io_complete	4164	11401	11444						
jmc$keyword_offset_maximum	4379	4665	4896						
jmc$kjl_maximum_entries	3871	3864	3865	4314					
jmc$kol_maximum_entries	3881	3866							
jmc$lock_ajl	9518	9504/P	9507	9637	9638/P	9640	9819/P	9819	9842/P
		9845	11488/P	11488	11494	11494/P	11494	11814	11814/P
		11814	13322/P	13322	13349	13349/P	13349	13941	13976
		13976/P	13976	14003/P	14003	14075/P	14075	14144/P	14144
		14152	14152/P	14152	14159	14159/P	14159		
jmc$max_active_jobs	3862	4877	4885	4886					
jmc$max_ajl_ord	3863	3856	3862	9521					
jmc$max_dispatching_control	4101	4105							
jmc$max_dispatching_priority	871	831	834	835	11048				
jmc$max_ijl_index_count	951	10554							
jmc$maximum_job_categories	10441	10438	10442						
jmc$maximum_job_classes	4292	4295							
jmc$maximum_job_count	3878	3871							
jmc$maximum_output_count	3888	3881							
jmc$maximum_service_classes	4395	4398							
jmc$min_dispatching_control	4100	4104							
jmc$min_dispatching_priority	872	11048							
jmc$needed_memory_available	919	9462/S	9464/S	11297/S	11297/S	11584/S	11584/S	13505/S	13505/S
jmc$null_ajl_ordinal	9521	9503	9819	9841	11488	13322	14003	14075	14144
jmc$null_service_class	4388	4389							
jmc$priority_aging_interval_max	4895	4892							
jmc$priority_bias_maximum	10461	10457	10457						
jmc$priority_p1	885	832	5971	10400	13847	13862			
jmc$priority_p10	894	833	10400						
jmc$priority_p14	898	833	5971						
jmc$priority_p8	892	832	13847	13862					
jmc$required_offset	4377	4679							
jmc$reserved_ajls	3867	3862							
jmc$scheduling_memory_level_max	10477	10474							
jmc$service_accumulator_maximum	4354	4351							
jmc$service_factor_value_max	4921	4918							
jmc$service_interval_maximum	10493	10490							
jmc$system_default_offset	4378	4379	4681						
jmc$system_supplied_name_size	4430	4427							
jmc$unlimited_offset	4375	4364	4638	4649	4675	4906			
jmc$unspecified_offset	4376	4677							
jmc$working_set_size_maximum	4664	4661							
jmp$assign_ajl_with_lock	9525	9504	9819	9842	11488	13322	14003	14075	14144
jmp$check_scheduler_memory_wait	9437	9468	11297	11584	13505				
jmp$free_ajl_with_lock	9649	9638	11494	11814	13349	13976	14152	14159	
jmp$get_ijle_p	9471	9477	9814	11309	11389	11443	11558	11707	12009
		12035	12057	12253	12298	12399	12436	12515	12662
		13322	13564	13707	13803	14003	14075	14144	14377
jmp$ijl_block_valid	9482	9487	12395	14001	14074				
jmp$ijl_block_valid	9483	9485/M	12395/M	14001/M	14074/M				
jmp$lock_ajl	9492	9512	11488						
jmp$lock_ajl_with_lock	9831	9819	9849	13322	14003	14075	14144		
jmp$set_scheduler_event	9858	13853	13854						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmp$set_scheduler_memory_event	9458	9465						
jmp$set_scheduler_memory_event	11284	11297						
jmp$set_scheduler_memory_event	11368	11584						
jmp$set_scheduler_memory_event	13377	13505						
jmp$unlock_ajl	9629	9644	11494	11814	13349	13976	14152	14159
jmt$active_job_list	9619	9597						
jmt$active_job_list_entry	9611	9619						
jmt$ajl_ordinal	3856	3813	3905	8138	9495	9498	9530	9633
		9812	9834	9837	10203	10241	10529	11467
		13744	14330					
jmt$cpu_dispatching_allocation	10385	10357						
jmt$delayed_swapin_work	4042	3935	4046	11406	12376	12376		
jmt$detached_job_wait_time	4634	4619						
jmt$dispatching_allocation	10388	10386						
jmt$dispatching_control	4071	4860						
jmt$dispatching_control_index	4104	4061	4071					
jmt$dispatching_controls	4074	4072						
jmt$dispatching_interval	10396	10359						
jmt$dispatching_priority	831	3805	3917	4062	4063	4064	4076	4808
		4810	10278	10323	10334	11145	11146	13759
jmt$dispatching_priority_set	10334	10318	10319	11062	11063	11064	11097	11098
		11099	11100	13761	13763	13763	13763	13842
		13877	13882					
jmt$dual_state_priority	10408	10404						
jmt$dual_state_priority_control	10400	10360						
jmt$dual_state_priority_entry	10403	10401						
jmt$dual_state_subpriority	10409	10405						
jmt$idle_dispatch_controls	10323	10320						
jmt$idle_dispatching_controls	10317	10312						
jmt$idle_dispatching_entry	10325	10323						
jmt$idle_dispatching_queue_time	10415	10361						
jmt$ijl_block_index	947	943	10556					
jmt$ijl_block_number	946	942	10544	10545				
jmt$ijl_dispatching_control	4060	3918						
jmt$ijl_entry_status	4137	3904						
jmt$ijl_ordinal	941	927	976	3429	3832	3924	3952	4445
		4446	4508	4609	7814	9471	9483	9494
		9527	9613	9806	9833	10119	10249	10844
		11015	11137	11539	11888	12183	12380	12546
		13698	14342					
jmt$ijl_p	10542	10537						
jmt$ijl_page_fault_count	4211	4206	4207	4208				
jmt$ijl_page_stats	4205	4201						
jmt$ijl_service_class_stats	4199	3939						
jmt$ijl_statistics	4244	3938						
jmt$ijl_swap_count	4220	4216	4217					
jmt$ijl_swap_counts	4215	3958	4202					
jmt$ijl_swap_status	4155	3907	3908	3909				
jmt$initiated_job_list_block	10553	10559						
jmt$initiated_job_list_entry	3901	3833	4470	4608	9321	9369	9472	9493
		9614	9630	9650	9673	9679	9685	9809
		9832	10007	10016	10030	10235	10556	11307
		11461	11540	11772	11887	12182	12377	12386
		12504	12510	12638	13282	13390	13699	13752
		14122	14214	14339				
jmt$initiated_job_list_p	10559	10543						
jmt$input_file_location	4334	4329						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmt$jl_job_leveler_state	8442	8437						
jmt$jl_job_leveler_status	8436	8077						
jmt$job_abort_disposition	4343	4327						
jmt$job_category	10438	10434						
jmt$job_category_set	10434	10362	10363	10451	10452			
jmt$job_class	4295	3963						
jmt$job_control_block	4590	3815	9686	10029	13743	14340		
jmt$job_mode	4298	3920						
jmt$job_priority	4303	3960	3961	4869	4870	4871	4872	
jmt$job_recovery_disposition	4346	4328						
jmt$job_sched_event_selections	924	9442						
jmt$job_scheduler_event	922	9447						
jmt$job_scheduler_events	902	922	924	9658				
jmt$job_scheduler_table	10345	10340						
jmt$job_system_id	4653	4605						
jmt$kjl_index	4314	3906	4653					
jmt$mainframe_categories	10445	10368						
jmt$mainframe_entry	10448	10445						
jmt$maximum_active_jobs	4877	4854						
jmt$priority_aging_interval	4892	4862						
jmt$priority_bias	10457	10372						
jmt$queue_file_ijl_information	4326	3945						
jmt$scheduling_data	3951	3929						
jmt$scheduling_memory_level	10474	10379	10380					
jmt$scheduling_memory_levels	10378	10364						
jmt$scheduling_priority	4868	4861						
jmt$service_accumulator	4351	3953	3954	3955	4852	4853		
jmt$service_class_index	4398	3964	4845	4855				
jmt$service_class_name	4910	4847	4848					
jmt$service_factor_value	4918	4856						
jmt$service_factors	4914	4856						
jmt$service_interval	10490	10351	10353					
jmt$swap_data	3967	3931						
jmt$swapout_reasons	4401	3959						
jmt$swapped_job_entry	4416	3976	4471	4628				
jmt$system_supplied_name	4427	3902	4603	10843				
jmt$task_time_slice	4114	4094	4095					
jmt$time_slice_values	4093	4078	4821					
jmt$user_dispatching_priority	832	10385	11075					
jmt$user_supplied_name	4657	4604						
jmt$working_set_size	4661	4615	4616	10567				
jmv$ajl_p	9597	9507/M	9507	9637	9640/M	9640	9819/M	9819
		9845/M	9845	11488/M	11488	11494	11494/M	11494
		11814	11814/M	11814	13322/M	13322	13348	13349/M
		13348	13937	13939	13939	13941/M	13941	13943
		13961	13976	13976/M	13976	14003/M	14003	14075/M
		14075	14144/M	14144	14152	14152/M	14152	14159
		14159/M	14159	14376				
jmv$idle_dispatching_controls	10312	13840	13841	13842	13846/M	13848/M	13849/M	13850/M
		13863	13867/M	13870/M	13873	13875/M	13876/M	13876
		13882/M	13883/M	13884/M	13965			
jmv$ijl_p	10537	9475	9485	9814	11309	11389	11443	11558
		11707	12009	12035	12057	12253	12298	12395
		12399	12436	12515	12662	13322	13564	13707
		13803	14001	14003				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		14074	14075	14144	14377				
jmv$job_sched_events_selected	9442	9462							
jmv$job_sched_events_selected	11284	11297							
jmv$job_sched_events_selected	11368	11584							
jmv$job_sched_events_selected	13377	13505							
jmv$job_scheduler_event	9447	9464							
jmv$job_scheduler_event	11284	11297							
jmv$job_scheduler_event	11368	11584							
jmv$job_scheduler_event	13377	13505							
jmv$job_scheduler_table	10340	13817	13820	13873					
jmv$long_wait_swap_threshold	9920	9906	9908/P	13897	13897/P				
jmv$max_ajl_ordinal_in_use	10529	13937							
jmv$max_class_working_set	10567	13800	13810	13811					
jmv$memory_needed_by_scheduler	9453	9463							
jmv$memory_needed_by_scheduler	11284	11297							
jmv$memory_needed_by_scheduler	11368	11584							
jmv$memory_needed_by_scheduler	13377	13505							
jmv$scan_idle_dispatch_interval	10506	13833							
jmv$system_ijl_ordinal	11015	11309/P	12435	12436/P	12455	12475	12515/P	12530	13608
		13803/P							
job_fixed_asid	3914	9504/P	9819/P	9842/P	11488/P	12070	13322/P	14003/P	14075/P
		14144/P	14410/M						
job_fixed_contiguous_pages	3940	12671/M							
job_is_good_swap_candidate	9615	13961							
job_monitor_taskid	3919	11719/P							
job_page_queue_list	3930	11310	11310	11311	12065	12081	12517	12664	13803
		13803	13803	13955/P	13957/P	14446/P			
jsc$isqi_swapped_io_completed	4450	4452	13908/P						
jsc$isqi_swapped_io_not_init	4449	4452	13907/P						
jsp$adv_expired_swapped_jobs	9663	13907	13908						
jsp$initiate_swapout_io	9669	9908	13897						
jsp$io_complete	9673	11749							
jsp$recalculate_swapped_pages	9678	12481	13580						
jst$changed_asid_entry	4493	4484							
jst$ijl_swap_queue_id	4449	4444							
jst$ijl_swap_queue_link	4443	3913							
jst$io_control_information	4457	3932							
jst$swap_file_descriptor	4469	3933							
jst$swapped_but_still_in_memory	4452	9663							
jst$swapped_page_descriptor	4478	4476	11902						
jst$swapped_page_descriptors	4475	4472							
jsv$pages_needed_for_sfd	10574	13900	13930						
jsv$swapped_page_entry_size	10578	13815							
jws_ijl_ordinal	12380	12394/M	12395/P	12399/P					
last_cp_time	10329	13840	13850/M	13864	13867/M	13883/M			
last_execution_time	4611	13962							
last_pfti_index	3374	9764	9777	9778	9782	10050	10051	10056	10068/M
		10107/S	10108/M	10108	12442/M	12447/S	12447/M	12447	12463
		12465	12465	12465	12476	12476	12476	12563	12563
		12563	12612	12612	12612	12648	12648	12648	12651
		12651	12651	12658	12658	12658	12778	12778	12778
		12801	12801	12801	12934	12934	12934	13356	13356

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		13356	14419	14419	14419				
last_segment_number	419	14405/M							
last_written_pfti	12708	12719/M	12732/M	12750	12751/M	12795/M	12810		
last_written_pfti	12836	12847	12848/S	12851/S	12862/S	12865/S			
last_written_pfti	12966	13003/P	13004/P						
last_written_pfti	13395	13636/P	13637/P						
length	996	11567	12007/M	12017/M	12024/M	12122/M	12291/M	12296/M	12335/M
		14521							
length	1395	12309/M	12317/M						
length	7762	13002/P	13007/P	13009/P					
length	11878	11917	12007						
length	12186	12216/M	12219/M	12223/M	12237	12296			
length	12578	12598/P							
length	12630	12643/P							
length	12699	12722/P							
length	12895	12906/P							
line	9292	9295/M	9300/M	9301/M	9304/P				
line_index	9293	9296/M	9300/M	9301/M	9302/M	9302	9304/P		
link	4505	11287/M	11291/M	11584/M	11584/M	12125	12520	12668	13505/M
		13505/M	14142						
link	4562	11287	11288	11289/M	11291/S	11293/M	11584	11584	11584/M
		11584/S	11584/M	12065	12082	12517	12664	13505	13505
		13505/M	13505/S	13505/M	14135				
list_i	11542	11566	11567/S	11570/S	11600				
list_i	11894	11907/M	11932/S	11938	12004/S	12007/S	12016	12017/S	12024/S
		12027/M	12027	12034	12036	12040	12046/P	12121/S	12122/S
		12131/M	12131						
list_i	12187	12213/M	12252	12254	12255	12294/M	12294	12299	12300
		12305/S	12309	12317					
list_i	14515	14520	14521/S	14524/S					
list_length	11371	11555	11566	11737	11742				
list_length	11881	11918	11918	12010	12014	12016	12058		
list_length	14509	14520							
list_p	1103	12338/M							
list_p	11370	11567	11570						
list_p	11880	12004/M	12007/M	12017/M	12024/M	12046/P	12121/M	12122/M	
list_p	12188	12203/M	12230/P	12259/P	12283/M	12291/M	12296/M	12305/P	12312
		12335/M	12338						
list_p	14508	14521	14524						
locate_a_modified_page	13165	13165	13180	13182					
locate_highest_offset	13478	13478	13484						
lock	9368	9377							
lock	9409	9422							
lock	9492	9501	9501	9501/M	9501/M	9501			
lock	9492	9510	9510	9510/M	9510	9510/M			
lock	9536	9539	9542	9543/M	9543	9545/M			
lock	9572	9580	9582	9584/M	9586/M	9586			
lock	9629	9635	9635	9635/M	9635/M	9635			
lock	9629	9642	9642	9642/M	9642	9642/M			
lock	9805	9818	9818	9818/M	9818/M	9818			
lock	9805	9820	9820	9820/M	9820	9820/M			
lock	9807	9817							
lock	11460	11488	11488	11488/M	11488/M	11488	11494	11494	11494/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11494/M	11494						
lock	11460	11488	11488	11488/M	11488	11488/M	11494	11494	11494/M
		11494	11494/M						
lock	11765	11814	11814	11814/M	11814/M	11814			
lock	11765	11814	11814	11814/M	11814	11814/M			
lock	13065	13139							
lock	13272	13308							
lock	13272	13322							
lock	13272	13322	13322	13322/M	13322/M	13322	13349	13349	13349/M
		13349/M	13349						
lock	13272	13322	13322	13322/M	13322	13322/M	13349	13349	13349/M
		13349	13349/M						
lock	13377	13421	13439	13450	13470	13490	13552	13598	13605
lock	13688	13715							
lock	13737	13938	13938	13938/M	13938/M	13938	13976	13976	13976/M
		13976/M	13976	14003	14003	14003/M	14003/M	14003	14075
		14075	14075/M	14075/M	14075				
lock	13737	13942	13942	13942/M	13942	13942/M	13976	13976	13976/M
		13976	13976/M	13978	13978	13978/M	13978	13978/M	14003
		14003	14003/M	14003	14003/M	14075	14075	14075/M	14075
		14075/M							
lock	13737	14003	14075						
lock	13737	14049							
lock	14111	14144							
lock	14111	14144	14144	14144/M	14144/M	14144	14152	14152	14152/M
		14152/M	14152	14159	14159	14159/M	14159/M	14159	
lock	14111	14144	14144	14144/M	14144	14144/M	14152	14152	14152/M
		14152	14152/M	14159	14159	14159/M	14159	14159/M	
lock	14111	14157							
lock	14277	14307							
lock	14329	14371	14404						
lock_encountered	13065	13170/M	13170/M						
lock_encountered	13075	13094/M	13103/M						
lock_encountered	13126	13163/M	13170/P	13172	13187	13194	13242	13257	
lock_loop	12211	12211	12313	12319					
locked	586	9377	9422	13139	13308	13421	13439	13450	13470
		13490	13552	13598	13605	13715	14049	14157	14307
		14371	14404						
locked	9563	9501	9582	9635	9818	11488	11494	11814	13322
		13349	13938	13976	14003	14075	14144	14152	14159
locked_for_write	549	13101	13170						
locked_page	4511	11503	11613/M	11626/M	11636/M	11649/M	11960	11970/M	11974/M
		11976/M	11985/M	12555	12561/M	12609	12770	12776/M	13091
		13092	13093	13102	13170	13170	13170	13170	13327
		13331							
loop_count	12189	12204/M	12206/M	12209					
loop_count_index	12190	12209	12215	12274	12287	12307			
lp	11923	11923	11928	11962	12028				
m	3397	11422/M	11441	11525/M	11603	11606/M	11617	11621/M	11674
		11693	11943	11978/M	11992/M	12727	12767	12783	12911/M
		12926/M	13083	13170	13239/M	13255/M	13327	13331	14143
max_dm_rejects	14115	14124	14165						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
max_input_count	1105	12216							
max_segnum	14218	14240/M	14242/M	14252					
maximum_segment_length	13693	13716/M							
maximums_exceeded	10319	13842	13882/M						
maxws_aio_slowdown_display	3943	13944	13945/M	13945					
maxws_left_for_user_jobs	13758	13805/M	13808/M	13808	13810	13813	13822		
mcount	12584	12615/P							
mcount	13284	13339/P							
mcount	13755	13912/P	13915/P	13925/P	13956/P	13958/P			
media	423	12999	14495	14498/M					
memory_keypoints	7784	12115							
mf_job_file	11373	11714							
mf_job_file	11895	12041/M	12043/M	12046/P					
min_working_set_size	4616	13956/P							
minimum	4568	13915/P							
minimums_to_satisfy	11062	13859							
mmc$	7309	7315	7318	7321	7324	7327	7330	7333	7337
		7340	7343	7346	7349	7352	7355	7359	7362
		7365	7368	7371	7374	7378	7381	7384	7387
		7390	7393	7396	7399	7402	7405	7408	7411
		7414	7417	7420	7423	7426	7429	7432	7435
		7438	7442	7445	7448	7451	7454	7457	7460
		7463	7466	7469	7472	7475	7478	7481	7484
		7487	7490	7493	7496	7499	7502	7506	7510
		7513	7517	7520	7523	7526	7529	7532	7535
		7538	7541	7544	7547	7550	7553	7556	7559
		7562	7565	7568	7571	7574	7578	7582	7585
		7588	7591	7594	7597	7600	7603	7606	7609
		7612	7615	7618	7621	7624	7627	7630	7633
		7636	7640	7643	7646				
mmc$assign_active_null	5036	5037							
mmc$bd_explicit_io	981	973	11914						
mmc$bd_job_swapping_io	980	975	12051						
mmc$bd_paging_io	980	973	11914						
mmc$cell_pointer	5135	5140							
mmc$first_loader_predefined_seg	10098	10081	10100						
mmc$heap_pointer	5136	5144							
mmc$iocb_table_size	9145	9138							
mmc$iorc_await_io_completion	8588	8581							
mmc$iorc_write_pages	8588	8579							
mmc$irs_active	8558	8552							
mmc$irs_complete	8558	8553							
mmc$irs_none	8558	8552							
mmc$kw_asid	5061	5097							
mmc$kw_clear_space	5059	5084							
mmc$kw_current_segment_length	5058	5078							
mmc$kw_error_exit_procedure	5060	5088							
mmc$kw_gl_key	5060	5082							
mmc$kw_hardware_attributes	5062	5091							
mmc$kw_inheritance	5062	5099							
mmc$kw_max_segment_length	5059	5080							
mmc$kw_preset_value	5061	5086							
mmc$kw_ps_transfer_size	5063	5107							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$kw_ring_numbers	5057	5073							
mmc$kw_segment_access_control	5061	5095							
mmc$kw_segment_number	5058	5076							
mmc$kw_shadow_segment	5063	5101							
mmc$kw_software_attributes	5060	5093							
mmc$kw_wired_segment	5063	5104							
mmc$lp_aging_lock	4527	13091	13170						
mmc$lp_not_locked	4527	11613	11626	11636	11649	12561	12609	12776	13327
		13331							
mmc$lp_page_in_lock	4528	11503	11960	11970	11974	12555	12770	13093	13170
mmc$lp_server_allocate_lock	4529	11976	13092	13170					
mmc$lp_write_protected_lock	4528	11985	13102	13170					
mmc$max_rma_list_length	986	991	992	1081	1100	1101	11538		
mmc$mf_segment_mgr_flag	3762	11796/P							
mmc$mmu_aging_algorithm	9218	11187							
mmc$mmu_jws_age_interval	9219	11177							
mmc$mmu_periodic_call_interval	9226	11181							
mmc$mmu_shared_age_interval	9227	11179							
mmc$mpt_done	3770	13236							
mmc$mpt_page_table_full	3770	13227							
mmc$num_loader_predefined_segs	10099	10100							
mmc$pq_avail	3479	3525	11452/P	11679/P	11699/P	11955	11964	12918/P	
mmc$pq_avail_modified	3480	11403	11445	11522	11603	11673	11979	11986	12448
		12615/P	12910	13912/P	13915/P	13925/P	13956/P	13958/P	14135/S
mmc$pq_first_valid_in_pt	3524	11590	11615	11628	11638	11646	11946	12554	12766
		13332							
mmc$pq_free	3478	3537	11450/P	11505/P	11579	11677/P	11697/P	11825/P	11994/P
		12557/P	12656/P	12772/P	13176/P	13214/P	13330/P	13343/P	13503
		13561	13571/P	14192					
mmc$pq_job_fixed	3519	3526	3538	11310/S	12064	12065/S	12074	12076	12085
		12661	12664/S	13803/S	14402	14418/P			
mmc$pq_job_io_error	3520	11311/S	11402/P	11521/P	12784	12923	13803/S	13957/S	
mmc$pq_job_working_set	3521	3538	3539	11311/S	11382	12446	12517/S	12594	12768
		12996	13803/S	13955/S					
mmc$pq_last_reassignable	3525	11999							
mmc$pq_shared_file_server	3487	12424							
mmc$pq_shared_first	3527	10619	11689	12521	12996	13807	13909		
mmc$pq_shared_first_site	3529	3533							
mmc$pq_shared_io_error	3516	11386/P	11420/P	12785	12923	13806/S	13924/S	13924/S	
mmc$pq_shared_last	3534	11689	12393	12522	12997	13425			
mmc$pq_shared_last_sys	3528	10619	11180	13919					
mmc$pq_shared_num_sites	3530	3533							
mmc$pq_shared_other	3488	3528	12432						
mmc$pq_shared_pf_non_execute	3485	13609							
mmc$pq_shared_site_01	3490	3529							
mmc$pq_shared_site_25	3514	3534							
mmc$pq_shared_task_service	3483	3527							
mmc$pq_swapped_io_error	3517	3537	11400	11404/P	11517	11523/P			
mmc$pq_wired	3481	3524	13250/P	13805/S	14416/P				
mmc$sa_stack	5123	13711							
mmc$sac_file_server_terminated	5590	11802							
mmc$sac_io_read_error	5587	11783							
mmc$sat_read_or_write	10150	12990/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES				
mmc$sat_write	10150	12988/P	13149/P			
mmc$segment_fault_processor_id	5568	5622	11781			
mmc$sequence_pointer	5135	5142				
mmc$server_iocb_table_size	8449	8452				
mmc$si_new_segment	5044	14257	14299			
mmc$ssr1_change_swap_file_queue	7690	7697	13604			
mmc$ssr1_commit_memory	7682	13500				
mmc$ssr1_delete_job_seg_by_sfid	7692	7698	13449			
mmc$ssr1_delete_seg_segnum	7679	7701	13488			
mmc$ssr1_delete_seg_sfid	7680	7696	13438			
mmc$ssr1_detach_file	7683	7696	13420	13620	13632	13635/P
mmc$ssr1_end_job_recovery	7687	7707	13556			
mmc$ssr1_flush_avail_modified	7694	7699	13459			
mmc$ssr1_flush_delete_seg_sfid	7684	7696	13438	13632		
mmc$ssr1_flush_seg_segnum	7685	7697	13488	13621	13633	
mmc$ssr1_free_image_pages	7681	13511				
mmc$ssr1_get_highest_offset	7691	7716	13468			
mmc$ssr1_make_mfw_cache	7688	7710	13589			
mmc$ssr1_remove_detached_pages	7693	7698				
mmc$ssr1_remove_job_shared_pages	7689	7712	13597			
mmc$ssr1_replace_sfid	7686	7703	13515			
mmc$sr_assign_file_to_disk	3716	3727				
mmc$sr_complete_seg_sft_entry	3718	3727				
mmc$sr_fetch_cyclic_aging_int	3721	3735				
mmc$sr_fetch_max_ws_size	3717	3729				
mmc$sr_fetch_min_ws_size	3719	3731				
mmc$sr_fetch_page_aging_int	3720	3733				
mmc$sr_store_cyclic_aging_int	3721	3735				
mmc$sr_store_max_ws_size	3717	3729				
mmc$sr_store_min_ws_size	3719	3731				
mmc$sr_store_page_aging_int	3720	3733				
mmc$ssk_none	5230	5202				
mmc$ssk_segment_number	5231	5200				
mmc$ssr1_flush_delete_seg_sfid	7738	7728	13292	13333	13345	13351
mmc$ssr1_free_delete_seg_sfid	7738	7728	13292			
mmc$ssr1_move_modified_df_page	7739	7730	13296			
mmc$uer_set_exact_eoi	3745	13718/P				
mmc$wmp_io_active	3406	12807	12812	12860	13639	13640
mmc$wmp_io_complete	3406	12718	12806	12858	13639	
mmc$wmp_io_errors	3407	12739	12788	12873		
mmc$wmp_io_initiation_reject	3406	12745	12811	12846		
mmc$wmp_server_terminated	3407	12743	12876			
mmc$wmp_volume_unavailable	3407	12741	12790	12810	12870	
mme$invalid_request	7442	13011/P	13615/P			
mme$io_active_on_move_page	7606	13192/P				
mme$io_write_error	7445	12874/P				
mme$no_pages_found_for_move	7408	13142/P	13185/P			
mme$page_frame_not_assigned	7337	12048/P	12265/P			
mme$page_table_full	7318	13241/P				
mme$segment_not_assigned_device	7423	13000/P				
mme$segment_not_pageable	7448	12998/P				
mme$volume_unavailable	7578	12871/P				
mmk$build_lock_rma1	6945	11834				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmk$create_job	6920	14354							
mmk$create_task	6917	14228							
mmk$exit_job	6926	14445							
mmk$exit_task	6923	14291							
mmk$free_flush	6847	12982	13016						
mmk$monitor_base	7000	6840	6847	6854	6879	6900	6914	6917	6920
		6923	6926	6929	6932	6935	6938	6941	6945
		6948	6951	6958	6962	6965	6971		
mmk$mtr_lock_ring_1_stack	6929	14476							
mmk$periodic_call	6941	13782	14090						
mmk$ring1_segment_request	6854	13408	13652						
mmk$unlock_rmal	6948	11571							
mmp$age_job_working_set	9685	13971	13974						
mmp$asid	9691	12428	13146	13314	13432	13442	13455	13463	13493
		14043							
mmp$aste_pointer	9697	11195							
mmp$aste_pointer_from_pfti	11191	11203							
mmp$build_lock_rma_list	11876	12141							
mmp$build_lock_rma_list_tape	12173	12341							
mmp$change_asid	9708	13361	14044						
mmp$check_queues	9716	11758	12139						
mmp$convert_pva	9727	12993	13154	14488					
mmp$create_job	14329	14422							
mmp$create_task	14211	14264							
mmp$delete_last_pfti_from_array	9738	9742	12610						
mmp$delete_pt_entry	9745	11449	11504	11676	11696	11824	12556	12655	12771
		13175	13205	13213	13329	13342	13570	14191	
mmp$determine_error_state	14507	14534							
mmp$determine_shared_queue_id	9753	12430							
mmp$exit_job	14437	14448							
mmp$exit_task	14277	14315							
mmp$fetch_pfti_array_size	9760	9766	12463						
mmp$fetch_stack_segment_info	13688	13726							
mmp$find_next_pfti	9769	9787	12476	12563	12612	12648	12658	12778	12801
		12934	13356	14419					
mmp$free_asid	9790	12675	13362	13573	14045				
mmp$free_flush	12959	13017							
mmp$free_image_pages_mtr	14179	13512	14197						
mmp$free_memory_in_job_queues	9796	14446							
mmp$get_inhibit_io_status	9805	9824	13322	14002	14075	14144			
mmp$get_max_sdt_pointer	9856	9864							
mmp$get_max_sdt_sdtx_pointer	9868	9878	13708	14230	14231	14293	14357	14358	
mmp$get_sdt_entry_p	9945	12426							
mmp$get_sdt_entry_p	9947	9949/M	12426/M						
mmp$get_sdtx_entry_p	9932	11794	13489						
mmp$get_sdtx_entry_p	9934	9936/M	11794/M	13489/M					
mmp$get_verify_asti_in_fde	10116	10130	13140	13309	13422	13440	13451	13461	13471
		13491	13606						
mmp$initialize_find_next_pfti	9881	12551	12598	12643	12722	12906	13318	14412	
mmp$link_page_frame_to_queue	11284	11299	11584	13505					
mmp$maintain_memory_thresholds	9891	9915	13897						
mmp$make_pt_entry	9923	13226	13235						
mmp$mm_conditional_free	12894	12941	13009						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmp$mm_free_pages	12628	12678	13007	13647	14309				
mmp$mm_write_modified_pages	12697	12816	13002	13634					
mmp$move_modified_server_page	13065	13263	13297						
mmp$mtr_lock_ring_1_stack	14460	14503							
mmp$mtr_r1_server_seg_request	13272	13366							
mmp$mtr_ring1_segment_request	13377	13654							
mmp$periodic_call	13737	14093							
mmp$process_page_table_full	9980	14009	14080						
mmp$process_volume_unavailable	9988	11791							
mmp$process_wmp_status	12835	12881	13004	13637					
mmp$purge_all_cache	11205	11601							
mmp$purge_all_cache_map	11223	12650							
mmp$purge_all_cache_map_proc	11238	11229	12650	13206					
mmp$purge_all_cache_proc	11218	11211	11601	13188	13195	13243	13258		
mmp$purge_all_map_proc	11255	11248	11262	11274	11447	11692	12020	12938	13169
		13186	13193	13326					
mmp$purge_all_page_map	11242	12020							
mmp$reclaim_ast_entries	9993	13993							
mmp$relink_page_frame	10000	11386	11402	11404	11420	11450	11452	11505	11521
		11523	11677	11679	11697	11699	11825	11956	11965
		11987	11994	12468	12523	12557	12656	12772	12918
		12924	13176	13214	13250	13252	13330	13343	13571
		14416	14418						
mmp$remove_detached_pages	12543	12566							
mmp$remove_jws_to_shared_pages	12367	12490	13600						
mmp$remove_page_from_job	10023	12559	12774						
mmp$remove_page_from_jws	10006	13339							
mmp$remove_pages_from_jws	10014	12615							
mmp$remove_pages_working_set	12576	12617							
mmp$remove_stale_pages	10027	13911	13914	13924	13955	13957			
mmp$remove_swapped_shared_pages	12503	12535							
mmp$replenish_free_queues	14111	13464	13987	14175					
mmp$reset_find_next_pfti	10039	10062	12465	12651					
mmp$reset_store_pfti	10065	12442							
mmp$set_include_pages_in_dump	10074	10093							
mmp$store_pfti	10104	12447							
mmp$sva_purge_all_page_map	11259	12938	13326						
mmp$sva_purge_one_page_map	11271	11447	11692						
mmp$unlock_rma_list	11368	11760	12046	12230	12259				
mmp$unlock_rma_list_error	11765	11623	11633	11642	11827				
mmp$update_eoi	10135	13718							
mmp$verify_pva	10142	12988	12990	13149					
mmp$write_page_to_disk	10152	12730	14158						
mmp$xcheck_queues	954	9720	11758	12139					
mmp$xtask_pva_to_sva	10167	12226							
mmt$active_io_count	8587	8466	8576	12706	12972	13392			
mmt$active_segment_table	3441	10632							
mmt$active_segment_table_entry	3426	3415	3442	4481	4515	9698	9708	9731	9791
		9885	9925	9983	10650	11192	12374	12545	12579
		12631	12701	12896	12969	13114	13117	13131	13279
		13385	13742	13750	14286	14336	14465		
mmt$aging_statistics	10605	10599							
mmt$asid_list_ptf_index	3418	3413							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$ast_index	565	410	3975	4496	4947	7706	9691	9703	9711
		9982	9994	10120	13115	13280	13386	13749	13776
mmt$async_work_list	10646	10640							
mmt$attribute_keyword	5057	5072							
mmt$buffer_descriptor	970	11877							
mmt$buffer_descriptor_type	980	972							
mmt$eoi_state	574	412							
mmt$global_page_queue_index	3537	4575	11180	11180					
mmt$global_page_queue_list	4575	10726							
mmt$global_page_queue_list_ent	4565	4575							
mmt$hardware_attribute_set	5126	5092							
mmt$hardware_attributes	5114	5126							
mmt$image_file	9177	11186	11186						
mmt$io_identifier	7810	8137	10156	11372	11891	12185	12705	12971	13393
		14123							
mmt$io_request_status	8558	8551							
mmt$io_status	8550	8560	8573						
mmt$iocb_index	9138	7817	7823						
mmt$job_page_queue_index	3538	4418	4576	12084					
mmt$job_page_queue_list	4576	3930	9796						
mmt$link	3464	3427	4505	4506	4562				
mmt$lock_segment_status	5210	5001							
mmt$locked_page	4527	4511							
mmt$mainframe_wired_asid	9150	9162							
mmt$make_pt_entry_status	3770	9927	13128						
mmt$max_sdt	4957	4961							
mmt$max_sdt_p	4961	9858	9870	13701	14219	14222	14282	14344	14347
mmt$max_sdtx	5025	5029							
mmt$max_sdtx_p	5029	9871	13702	14220	14224	14284	14345	14348	
mmt$memory_reserve_request	4543	3923							
mmt$monitor_segment_request	3716	3726							
mmt$old_modified_bits	9171	9161							
mmt$page_age	4534	4514	4538	4538					
mmt$page_frame_index	3378	3375	3466	3466	4458	4460	4461	4462	4545
		4546	9453	9669	9680	9746	9770	9774	9886
		9924	10000	10006	10023	10040	10044	10104	10154
		10820	10822	10824	11184	11184	11191	11284	11304
		11473	11546	11766	11900	12194	12381	12382	12507
		12508	12549	12585	12637	12639	12708	12714	12836
		12900	12966	13072	13119	13129	13285	13395	13399
		13751	13758	13766	13767	13777	14120	14121	14184
		14346	14514						
mmt$page_frame_queue_id	3539	3435	4459	4509	9755	10001	10015	10031	12384
		13765							
mmt$page_frame_table	4520	9160	10708						
mmt$page_frame_table_entry	4504	4479	4520	9926	11285	11462	11472	11544	11767
		11899	12193	12383	12509	12899	13078	13118	14513
mmt$page_queue_list_entry	4561	4566	4576	10027					
mmt$page_selection_criteria	3366	9884							
mmt$pfti_array	3371	10717							
mmt$pt_full_status	963	9984	13753						
mmt$rb_free_flush	7758	12960							
mmt$rb_lock_ring_1_stack	3595	14461							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$rb_ring1_segment_request	7673	12370	13378						
mmt$rb_ring1_server_seg_request	7724	13066	13273						
mmt$rb_set_get_segment_length	9969	9958							
mmt$reassignable_page_frames	10753	10750							
mmt$rma_list	989	1103	11370	11880	12188	14508			
mmt$rma_list_entry	994	989	2262	8168	8183				
mmt$rma_list_index	991	989	11542	11894	12187	12199	14515		
mmt$rma_list_length	992	971	11371	11881	14509				
mmt$sdtx_stream_data	5008	5004							
mmt$segment_access_condition	5595	5623	11774						
mmt$segment_access_rights	5174	5000							
mmt$segment_access_state	5180	4995							
mmt$segment_access_type	10150	10143							
mmt$segment_descriptor	4944	4954	4958	9732	9755	9947	10077	10512	10522
		12385	12967	13121	13400	14221	14471		
mmt$segment_descriptor_extended	4993	5022	5026	9733	9934	9937	11776	11794	12968
		13122	13401	13489	14472				
mmt$segment_descriptor_table	4953	10290							
mmt$segment_descriptor_table_ex	5021	10296							
mmt$segment_inheritance	5043	4997	5100						
mmt$segment_pointer_kind	5135	5139							
mmt$segment_reservation_state	5220	4998							
mmt$server_iocb_entry	8456	8145	8453						
mmt$server_segment_request	7738	7727							
mmt$server_state	8594	8458							
mmt$set_get_subfunction_codes	9976	9973							
mmt$shadow_info	5195	5002							
mmt$shadow_reference_info	5243	4834							
mmt$shadow_segment_kind	5230	5199							
mmt$software_attribute_set	5128	4999	5094						
mmt$software_attributes	5122	5128							
mmt$sub_reqcodes	8588	8463	8578						
mmt$update_eoi_reason	3745	10138							
mmt$write_modified_pages_status	3406	12709	12835	12965	13403				
mmt$write_page_to_disk_status	10162	10158	12713	14119					
mmt$xcb_page_wait_info	5254	4820							
mmv$aggressive_aging_level	10593	13816	13900	13930	14032				
mmv$aging_algorithm	11187	13947							
mmv$aging_statistics	10599	13902/M	13902	13917/M	13917	13920/M	13921	13932/M	13932
		13973/M	13973	14033/M	14033				
mmv$ast_p	10632	10124	10124	10125	12389	13140	13140	13140	13157
		13309	13309	13309	13317	13422	13422	13422	13424
		13440	13440	13440	13444	13451	13451	13451	13452
		13457	13461	13461	13461	13471	13471	13471	13475
		13491	13491	13491	13495	13523	13524	13525	13527
		13528	13537	13537	13538	13538	13540	13606	13606
		13606	13608/M	13609/M	14040	14041	14044/P	14045/P	14303
		14400							
mmv$async_work	10640	13228/M	13229/M	13230/M	13991	13992/M	13998	13999	14001/P
		14002/P	14009/P	14013/M					
mmv$check_queues	956	9719	11758	12139					
mmv$free_file_server_pages	11176	11695							
mmv$gpql	10726	11287	11288	11289/M	11291/S	11293/M	11294/M	11294	11584

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11584	11584/M	11584/S	11584/M	11584/M	11584	13505	13505
		13505/M	13505/S	13505/M	13505/M	13505	13805	13806	13808
		13911/P	13911/P	13914/P	13914/P	13915/P	13924/P	13924/P	14135
mmv$image_file	11186	13508/M							
mmv$jws_queue_age_interval	11177	13935							
mmv$last_active_shared_queue	11180	13807	13909						
mmv$max_template_segment_number	10662	14382							
mmv$max_working_set_size	10656	13822/M	13823	13824/M					
mmv$memory_wait_queue	10692	11753	11753	11754/P	14022	14023/P			
mmv$multi_page_write	10685	12730/P	14158/P						
mmv$multiple_caches	10669	11210	11228	11601	12650				
mmv$multiple_page_maps	10677	11228	11247	11261	11273	11447	11692	12020	12650
		12938	13326						
mmv$no_memory_buffering	10702	11448	11675						
mmv$periodic_call_interval	11181	14088							
mmv$pft_p	10708	11195/P	11196	11291/M	11500	11529	11573	11584/M	11935
		12092	12270	12445	12457	12467	12519	12554	12555
		12561/M	12609	12647/S	12665	12666	12666	12668	12727/S
		12750	12766	12767/S	12768	12770	12776/M	12783/S	12784
		12785	12786	12787	12794	12848	12851/P	12862	12865/P
		12908	13081	13166	13166	13167/S	13168/S	13170	13179/S
		13191	13204	13219	13251/M	13321	13321	13322/P	13324/S
		13325/S	13326/P	13327/S	13327	13328	13331/S	13331	13332
		13338/S	13346/S	13480	13481	13483	13502	13503	13505/P
		13505/M	13558	13558	13559	13560	13561	13562	13563
		13564/P	14065	14065	14065	14065	14065	14066	14066
		14067	14074/P	14075/P	14080/P	14139	14142	14143/S	14143
		14144/P	14150	14157/P	14164	14192/M	14193/M	14417/M	14525
mmv$pfti_array_p	10717	9740/S	9740/M	9764	9764	9777	9777	9778	9778
		9779/M	9779	9780	9780/S	9782	9782	10047/M	10047
		10048	10048/S	10050	10050	10051	10051	10052/M	10052
		10053	10053/S	10056	10056	10068/M	10069/M	10070/M	10107/S
		10107/M	10108/M	10108	12442/M	12442/M	12442/M	12447/S	12447/M
		12447/M	12447	12463	12463	12465/M	12465	12465	12465/S
		12465	12465	12465/M	12465	12465	12465/S	12465	12465
		12465	12465	12476	12476	12476/M	12476	12476	12476/S
		12476	12476	12476	12476	12563	12563	12563/M	12563
		12563	12563/S	12563	12563	12563	12563	12610/S	12610/M
		12612	12612	12612/M	12612	12612	12612/S	12612	12612
		12612	12612	12648	12648	12648/M	12648	12648	12648/S
		12648	12648	12648	12648	12651/M	12651	12651	12651/S
		12651	12651	12651/M	12651	12651	12651/S	12651	12651
		12651	12651	12658	12658	12658/M	12658	12658	12658/S
		12658	12658	12658	12658	12778	12778	12778/M	12778
		12778	12778/S	12778	12778	12778	12778	12801	12801
		12801/M	12801	12801	12801/S	12801	12801	12801	12801
		12934	12934	12934/M	12934	12934	12934/S	12934	12934
		12934	12934	13356	13356	13356/M	13356	13356	13356/S
		13356	13356	13356	13356	14419	14419	14419/M	14419
		14419	14419/S	14419	14419	14419	14419		
mmv$pt_length	10734	14063	14069	14186					
mmv$pt_p	10740	11441	11518	11587	11674	11799/M	11930	11943	11946
		11957/M	11966/M	11969	11978/M	11981	11988	11989/M	11992/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		12106	12269	12647/M	12727	12767	12783	12909	13083
		13167	13168/M	13170	13179/M	13212	13239/M	13240/M	13254/M
		13255/M	13324	13325/M	13327	13331	13338/M	13346/M	14064
		14065	14065	14072	14143	14187	14490		
mmv$reassignable_page_frames	10750	9463	9903	9903	9907	11295/M	11295	11296/M	11296
		11297	11553	11584/M	11584	11584/M	11584	11584	11604/M
		11604	11618/M	11618	11752	11980/M	11980	12917/M	12917
		13504/M	13504	13505/M	13505	13505/M	13505	13505	13897
		13897	13897	13899	13899	13929	13929	14022	14032
		14032	14137	14137	14137	14137			
mmv$reduce_jws_for_thrashing	11178	13934							
mmv$resident_job_target	10766	13818/M	13820/M	13822					
mmv$ring1_request_trace	11175	13412/M	13412						
mmv$shared_pages_in_jws	11011	12767							
mmv$shared_queue_age_interval	11179	13906							
mmv$successful_error_retry	11543	11434/M	11434						
mmv$tables_initialized	10771	12979							
mmv$test_pt_full	11183	14062							
mmv$test_reassign_asid	10776	14039							
mmv$time_to_call_mem_mgr	10791	13231/M	14088/M						
mmv$total_contig_pages_assigned	9186	12672/M	12672						
mmv$total_page_frames	11184	13805							
mmv$write_aged_out_pages	10785	14137	14138						
monitor_fault	11773	11781/M	11782	11805/P	11810/P				
monitor_lock	400	9377/P	13139/P	13308/P	13421/P	13439/P	13450/P	13470/P	13490/P
		13552/P	13598/P	13605/P	13715/P	14049/P	14157/P	14307/P	14371/P
		14404/P							
move_page	13065	13170/M	13170/M						
move_page	13074	13080/M	13107/M						
move_page	13127	13170/P	13171						
mpt_status	13128	13226/P	13227	13235/P	13236				
mtc$job_fixed_segment	7306	9338	9375	11490	13139	13308	13421	13439	13450
		13460	13470	13490	13552	13598	13605	13715	13953
		14049	14157	14258	14307	14371	14392	14404	14408
mtp$cst_p	10173	12590	12721	12843	14294	14356	14444		
mtp$error_stop	9405	11197	11575	11591	11657	11738	11919	11941	11944
		11948	11996	12000	12078	12095	12136	12239	12262
		12278	12288	12849	12863	13237	13300	13629	14051
		14140	14305						
mtp$set_interlock	9409	9377	9430	13139	13308	13421	13439	13450	13470
		13490	13552	13598	13605	13715	14049	14157	14307
		14371	14404						
mtp$set_status_abnormal	10185	10192	12048	12265	12871	12874	12877	12998	13000
		13011	13142	13185	13192	13241	13615		
mtt$monitor_interlock	581	400	9409						
mtv$cst0	10179	10175	12590	12721	12843	13835	13836	13837	14294
		14356	14444						
mtv$monitor_segment_table	10511	13590/M							
mtv$nos_segment_table_p	10521	13592/M							
nat$received_message_descriptor	5270	5263	5272						
nat$received_message_list	5262	4802							
new_asid	13748	14009/P	14080/P						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
new_aste_p	13750	14010/P	14080/P						
new_asti	13749	14009/P	14080/P						
new_job_ajl_ordinal	14330	14376/S							
new_job_ijl_ordinal	14342	14376/M	14377/P	14403	14417				
new_sfid	7705	13525	13551	13552/P					
next_cyclic_aging_time	4620	13954	14409/M						
next_pfti	4460	12065/M	12073	12075/M	12081/M	12091	12091	12092/S	12116
		12121	12125/M						
next_pfti	12507	12520/M	12532						
next_pfti	14121	14142/M	14172						
next_queue_id	4459	12064/M	12074	12076	12082/S	12084	12085/M	12087/M	12087
nlc$cc_connect_confirm	5299	5290							
nlc$cc_connect_request	5298	5288							
nlc$cc_expedited_data	5304	5290							
nlc$cc_max_pdu_kind	5306	5309							
nlc$channel_connection_pdu	5322	5277							
nlc$channelnet_pdu	5322	5279							
nlt$cc_pdu_kind	5309	5287							
nlt$cc_seq#_or_connect_time	5286	5278							
nlt$cc_sequence_number	5312	5291							
nlt$device_identifier	5319	5273							
nlt$pdu_type	5322	5276							
no_of_data_commands	1104	12212							
normal	1785	10190/M	11550/M	11906/M	12048/M	12201/M	12227	12261	12265/M
		12388/M	12871/M	12874/M	12877/M	12983/M	12992	12998/M	13000/M
		13011/M	13135/M	13142/M	13150	13185/M	13192/M	13241/M	13414/M
		13615/M	14475/M	14491/M					
notify_swapper_when_io_complete	3916	11748							
now	10754	9463	9903	11296/M	11296	11297	11553	11584/M	11584
		11584	11752	13505/M	13505	13505	13897	13899	13929
		14022	14032	14137	14137				
nowait_wait	13396	13637/P							
null_sva	11208	11213							
null_sva	11226	11231	11232						
null_sva	11245	11250							
null_sva	11368	11601							
null_sva	11876	12020							
null_sva	12628	12650	12650						
offset	1011	11477	11480	11482	11483	11491	11501	11501	11916
		12007	12025/M	12025	12236	12292/M	12292	12296	12665
		12666	12666	13204/M	13204	13261	13316/M	13480	13481
		13619/M	14296/M	14374/M					
offset	9329	9334/M	9341/M	9344					
offset	9368	9375/M	9375/M	9375					
offset	11460	11490/M	11490/M	11490					
offset	13065	13139/M	13139/M	13139					
offset	13272	13308/M	13308/M	13308					
offset	13377	13421/M	13421/M	13421	13439/M	13439/M	13439	13450/M	13450/M
		13450	13460/M	13460/M	13460	13470/M	13470/M	13470	13490/M
		13490/M	13490	13552/M	13552/M	13552	13598/M	13598/M	13598
		13605/M	13605/M	13605					
offset	13688	13715/M	13715/M	13715					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
offset	13737	14049/M	14049/M	14049					
offset	14111	14157/M	14157/M	14157					
offset	14211	14258/M	14258/M	14258					
offset	14277	14307/M	14307/M	14307					
offset	14329	14371/M	14371/M	14371	14392/M	14392/M	14392	14404/M	14404/M
		14404							
old_sfid	7704	13527	13536						
old_sfid	13397	13536/M	13538						
open_validating_ring_number	4994	14388							
osc$aging_interval_maximum	4685	4688							
osc$base_exception	5983	5985	6132	10923					
osc$call_instruction	5502	5510							
osc$data_read	5501	5510							
osc$free_running_clock_maximum	767	764	10430						
osc$invalid_ring	602	642							
osc$keypoint_base	10923	10925	10929	10933	10936	10940	10944	10948	10951
		10955	10959	10963	10967	10970	10973	10977	10980
		10984	10987	10992					
osc$max_channel_number	8936	8939							
osc$max_fault_contents	5635	5629							
osc$max_idle_count	4749	4757							
osc$max_integer	8607	8612	8613						
osc$max_kpt_pages	10814	10820	10822	10824	10860				
osc$max_name_size	2811	2815	2818	3025					
osc$max_number_of_processors	4734	3800	9554	10849	10850	11022			
osc$max_page_frames	3382	3372	3373	3374	3378	3428	3970	3971	4417
		4419	4563	4569	9237	9900	10754	10755	10756
		10757							
osc$max_page_size	755	751							
osc$max_page_table_entries	3383	3386							
osc$max_ring	601	642	643						
osc$max_segment_length	625	648	2226	5005	5036	11482	11485		
osc$max_status_condition_code	1807	1803	1819						
osc$max_status_condition_number	10195	10186							
osc$max_string_size	1823	1826	1829	1834					
osc$max_tasks	527	524							
osc$maximum_offset	624	625	645	645	646				
osc$maximum_processor_id	5527	5523							
osc$maximum_processor_number	4726	4721							
osc$maximum_processors	4730	4726	4734						
osc$maximum_segment	623	644							
osc$min_ecc	5982	5983							
osc$min_integer	8606	8610	8611						
osc$min_page_size	754	751							
osc$min_ring	600	643							
osc$non_writable	3787	10080							
osc$nowait	7774	13396							
osc$pr_base_constant	6800	9501	9510	9539	9579	9635	9642	9818	9820
		11488	11488	11494	11494	11814	11814	13322	13322
		13349	13349	13938	13942	13976	13976	13978	14003
		14003	14075	14075	14144	14144	14152	14152	14159
		14159							
osc$purge_all_cache	7655	11213	11231	11601	12650				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$purge_all_page_seg_map	7664	11232	11250	12020	12650	13594	14375		
osc$segnum_job_fixed_heap	3551	14365/S	14366/S	14367/S	14373/S	14400/S	14401/S	14405	14407/S
osc$sva_purge_all_page_map	7660	11264	12938	13326					
osc$sva_purge_one_page_map	7659	11276	11447	11692					
osc$task_time_slice_maximum	4125	4128							
osc$traps_enabled	5445	11797							
osc$vl_cache_bypass	4972	14366							
osc$vl_invalid_entry	4972	14253	14254	14298	14368	14383			
osc$vl_regular_segment	4972	13591	13593						
osc$wait	7774	12861	13396						
osk$debug	7040	11571	11934	14228	14291	14354	14445	14476	
osk$entry	7038	12982	13408	13782					
osk$exit	7039	13016	13652	14090					
osk$m	7077	11571	11934	12116	12982	13408	13652	14445	14476
osk$performance	7042	12116							
osk$system_class	7052	7036	7037	7038	7039	7040	7041	7042	
osp$process_keypoint_io_error	10198	11582							
osp$process_keypoint_periodic	10199	14027							
ost$aging_interval	4688	4617	4618						
ost$asid	1014	1010	3414	3433	3914	4483	4494	4495	4978
		5098	9151	9152	9526	9692	9697	9702	9709
		9710	9790	9981	11892	12373	13113	13278	13383
		13748	13775	14112	14124				
ost$binary_unique_name	663	402	8545						
ost$byte	1017	1496	1631						
ost$byte_count	1004	11878	12186						
ost$class_15_keypoint	10899	10867							
ost$clear_file_space	3019	2621							
ost$cp_time	4232	4200	4245	4819	11049				
ost$cp_time_value	4230	3956	4233	4234	4612	4613	4832		
ost$cpu_element_id	4718	3831							
ost$cpu_idle_statistics	4752	3834							
ost$cpu_memory_port_mask	4720	3807							
ost$cpu_running_or_stepped	4770	4767	4767						
ost$cpu_state	4765	3816							
ost$cpu_state_reason	4776	3837							
ost$cpu_state_table	3803	3800	9728	9959	10173	12369	12583	12712	12841
		12961	13067	13274	13379	14283	14337	14441	14462
ost$cs_lock	774	4800							
ost$cst_trace_control	5870	3835							
ost$date_time	8617	8101	10901						
ost$debug_code	5501	5489							
ost$debug_list	5497	5401							
ost$debug_list_entry	5488	5497							
ost$debug_mask	5507	5400							
ost$exchange_package	5350	4787							
ost$execute_privilege	3781	3776	4973						
ost$execution_control_block	4786	3817	4812	9857	9869	9933	9946	9988	10234
		11779	13689	14212	14213	14278	14332	14333	14438
ost$external_interrupt_request	5858	3823							
ost$family_name	4698	4693							
ost$flags	5407	5357							
ost$frame_descriptor	5465	5480							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$free_running_clock	764	418	3432	3925	3926	3927	3928	3962	3972
		3973	3974	4065	4077	4611	4620	4818	9616
		10328	10329	10415	11071	11078	11079	13746	13762
ost$global_task_id	518	420	511	1122	3812	3919	3948	4607	4797
		4798	5539	5722	6062	7816	8129	10213	10219
		10226	10233	10248	10254	10264	10271	10277	10583
		10583	11547	11778	13754	14226	14351		
ost$halfword	1019	4738							
ost$heap	3568	7795	7803						
ost$idle_type	4761	4756							
ost$key_lock	631	4979	5083						
ost$key_lock_value	637	634	5424	5426					
ost$keypoint	10890	10900							
ost$keypoint_class	5439	5370	5441						
ost$keypoint_control	10831	10794							
ost$keypoint_environment	10881	10838							
ost$keypoint_mask	5441	5373	10837	10837	10837	10837			
ost$keypoint_multipro_option	10885	10839							
ost$logical_processor_id	4721	3808							
ost$minimum_save_area	5475	5362	5450	5616					
ost$monitor_condition	5326	5333							
ost$monitor_conditions	5333	5363	5367	5455	5691	5705			
ost$monitor_fault	5612	5561	10265	11773					
ost$monitor_fault_contents	5629	5625							
ost$name	2818	2781	2842	2888	2919	3247	4657	4696	4698
		4846	4910	5654	7847	7877	10352		
ost$non_negative_integers	8612	8084							
ost$p_register	5422	5351	5476	5683	5689				
ost$page_id	3388	3398							
ost$page_size	751	732	11004						
ost$page_table	3402	10740							
ost$page_table_entry	3393	3402	4480	11474	11545	12901	14183		
ost$page_table_index	3386	3402	4512	13774					
ost$paging_statistics	4268	4246	4827						
ost$parcel	1021	3829	3830						
ost$physical_channel_number	8939	8881							
ost$pre_processed_for_reconfig	5866	3838							
ost$processor_element_id	4737	4718							
ost$processor_element_number	4746	4739							
ost$processor_id	5523	4790	5517						
ost$processor_id_set	5517	4789	10851						
ost$processor_keypoint_control	10818	10853							
ost$processor_model_number	681	665	4704	4740					
ost$processor_serial_number	759	664	4705	4741					
ost$pva	653	5395	5413	5427	5613	5706	10292		
ost$read_privilege	3784	3777	4974						
ost$real_memory_address	1002	1130	1396	1437	1474	1487	1576	1577	1580
		1585	1591	1607	1613	1617	1704	1710	3828
		7847	8048	8763					
ost$register_number	5418	5392	5461	5469	5470	5471			
ost$ring	642	654	4976	4977	4994	5074	5075	5412	
ost$ring1_termination_reason	5535	4823							
ost$segment	644	419	655	3728	5077	5201	5390	5491	6080

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7702	7714	9934	9947	10075	11775	13692	13703
		14218	14225	14288	14331	14349			
ost$segment_access_control	3774	5096							
ost$segment_descriptor	4971	4945							
ost$segment_length	648	5079	5081	5103	5105	5108	6083	7747	7749
		7762	8461	8572	9882	9972	12578	12630	12699
		12895	13693						
ost$segment_offset	645	656	1011	2247	5009	5492	5494	6061	6082
		7732	8460	10137					
ost$signature_lock	775	7853							
ost$stack_frame_save_area	5449	5483	5649						
ost$state_tables	3800	10179							
ost$status	1791	3062	3241	5090	5684				
ost$status_condition	1815	1786	8554	8575					
ost$status_condition_code	1819	1794	1815						
ost$string	1832	1795							
ost$string_size	1826	1833							
ost$system_flag	5790	5786							
ost$system_virtual_address	1009	974	4517	9729	9881	9923	9980	10168	10649
		11259	11271	11903	12198	12544	12577	12629	12698
		12894	12974	13123	13132	13288	13402	14287	14350
		14473							
ost$task_index	524	519	558	559	3842	11134	11147		
ost$task_time_slice	4128	4114	10349						
ost$top_of_stack_pointer	5410	5402							
ost$trap_enable	5444	5359	5680						
ost$user_condition	5336	5343							
ost$user_conditions	5343	5361	5365	5453	5482	5652	5692		
ost$user_identification	4691	4606							
ost$user_name	4696	4692							
ost$valid_relative_pointer	651	425	428	3598	4816	4817			
ost$valid_ring	643	2984	2985	2986	5402	13690			
ost$virtual_machine_identifier	5432	5353	5355	5477					
ost$wait	7774	7763	8574	12837	13396				
ost$word	1023	1490	1623	1718	7984				
ost$write_privilege	3787	3778	4975						
ost$x_register	5419	5392	5461						
osv$i80_memory_limits	11030	13501	14188	14189					
osv$cpus_logically_on	9554	9501	9510	9538	9578	9635	9642	9818	9820
		11488	11488	11494	11494	11814	11814	13322	13322
		13349	13349	13938	13942	13976	13976	13978	14003
		14003	14075	14075	14144	14144	14152	14152	14159
		14159							
osv$cpus_physically_configured	11022	13834							
osv$keypoint_control	10794	11932	14026						
osv$keypoint_periodic_lpid	11896	11932/S							
osv$mainframe_wired_heap	7795	13590/S	13592/S						
osv$page_size	11004	11570	11916	11917	11930	12004	12007	12024	12025
		12121	12122	12236	12237	12269	12283	12291	12292
		12296	13212	13501	13816	14064	14072	14190	14524
osv$time_to_check_asyn	11038	13232/M							
p	10294	14488/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
p_register	5351	11797	12987						
page_aging_interval	4617	13972							
page_count	11897	11917/M	11918	12005/M	12005	12006	12058/M	12059	12060
		12127/M	12127	12128					
page_count	12191	12237/M	12238	12284/M	12284	12285	12303		
page_count	12970	13007/P							
page_count	13398	13647/P							
page_count	14285	14309/P							
page_offset	11898	11916/M	11917	12004	12008	12024	12026/M		
page_offset	12192	12236/M	12237	12283	12291	12293/M	12297		
page_table_entry	4480	12106/M							
page_wait_info	4820	11793	11803	11804	11808	11809			
pageid	3398	14065	14065						
pages_in_memory	3428	13360	13572	14042					
pages_not_deleted	7729	13293/M	13340/M	13340	13347/M	13347	13352/M	13352	
parent_fde_p	14343	14371/P	14372/M						
parent_sdt_p	14219	14230/P	14254	14256					
parent_sdt_p	14344	14357/P	14365	14368/M	14369/M	14390			
parent_sdtx_p	14220	14230/P	14255						
parent_sdtx_p	14345	14357/P	14370/P	14389					
parent_xcb_p	14212	14230/P	14239	14242					
parent_xcb_p	14332	14357/P							
pass	13773	14062							
periodic_requested	10832	14026							
pfc$execute	3028	3030	3033						
pfc$read	3027	3030	3033						
pft$share_options	3033	3034							
pft$usage_options	3030	3031							
pft$usage_selections	3031	2615							
pft_entry	4479	12105/M	12113						
pft_entry_p	13065	13170/M	13170/S	13170	13170	13170	13170		
pft_entry_p	13078	13081/M	13083/S	13091	13092	13093	13102		
pft_link	3427	11497	12443	13162	13177	13476			
pfte_p	11285	11287/M	11287/S	11288/S	11289/S	11291/S	11293/S	11294/S	11294/S
pfte_p	11368	11584/M	11584/S	11584/S	11584/S	11584/S	11584/S	11584/S	11584/S
pfte_p	11472	11500/M	11501	11501	11502	11503	11517	11518/S	11522
pfte_p	11544	11381/M	11382	11383	11388	11400	11403	11414/P	11428
		11429	11429	11430/P	11433/M	11441/S	11442	11445	11447/P
		11573/M	11574	11577/M	11577	11579	11580	11584/P	11587/S
		11590	11603	11613/M	11615	11623/P	11626/M	11628	11633/P
		11636/M	11638	11642/P	11646	11649/M	11653	11664	11665
		11665	11666/P	11673	11674/S	11689	11689	11692/P	11706
pfte_p	11767	11786/P	11799/S						
pfte_p	11899	11935/M	11936/M	11936	11939	11940	11946	11955	11956/P
		11960	11961/M	11961	11964	11965/P	11970/M	11974/M	11976/M
		11979	11985/M	11986	11987/P	11999	12009/P	12012	12035/P
		12038	12092/M	12094	12105	12106/S	12107	12125	
pfte_p	12193	12253/P	12270/M	12271/M	12271	12276	12277	12298/P	
pfte_p	12383	12445/M	12446	12448	12449	12450	12451	12453	12455/M
		12467/M	12468/P	12469	12470	12471	12473	12475/M	
pfte_p	12509	12519/M	12520	12521	12522	12523/P	12524	12525	12526
		12528	12530/M						
pfte_p	12899	12908/M	12909/S	12910	12910	12923	12923	12924/P	12929/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		12930/M							
pfte_p	13377	13505/M	13505/S	13505/S	13505/S	13505/S	13505/S	13505/S	13505/S
pfte_p	14513	14525/M	14526	14527	14528				
pfti	9774	9776/M	9777	9777	9780/M	9783/M	9785		
pfti	10044	10048/M	10050	10050	10053/M	10057/M	10059		
pfti	10104	10107							
pfti	11191	11195/S	11196/S						
pfti	11284	11289	11291	11293					
pfti	11368	11584	11584	11584					
pfti	11473	11497/M	11499	11499	11500/S	11504/P	11505/P	11521/P	11523/P
		11529/M	11529/S						
pfti	11546	11386/P	11402/P	11404/P	11420/P	11449/P	11450/P	11452/P	11570/M
		11571	11573/S	11584/P	11623/P	11633/P	11642/P	11676/P	11677/P
		11679/P	11696/P	11697/P	11699/P				
pfti	11766	11824/P	11825/P						
pfti	11900	11930/M	11932/M	11934	11935/S	11956/P	11965/P	11987/P	11994/P
		12004							
pfti	12194	12269/M	12270/S	12283					
pfti	12367	12447							
pfti	12367	12465/M	12465	12465/M	12465	12465/M	12465		
pfti	12367	12476/M	12476	12476/M	12476	12476/M	12476		
pfti	12382	12443/M	12444	12444	12445/S	12447/P	12457/M	12457/S	12465/P
		12466	12466	12467/S	12468/P	12476/P			
pfti	12508	12517/M	12518	12518	12519/S	12523/P	12532/M		
pfti	12543	12563/M	12563	12563/M	12563	12563/M	12563		
pfti	12549	12552/P	12553	12553	12554/S	12555/S	12556/P	12557/P	12559/P
		12561/S	12563/P						
pfti	12576	12612/M	12612	12612/M	12612	12612/M	12612		
pfti	12585	12598/P	12599	12607	12607	12609/S	12612/P		
pfti	12628	12648/M	12648	12648/M	12648	12648/M	12648	12658/M	12658
		12658/M	12658	12658/M	12658				
pfti	12628	12651/M	12651	12651/M	12651	12651/M	12651		
pfti	12639	12643/P	12646	12646	12647/S	12648/P	12651/P	12654	12654
		12655/P	12656/P	12658/P					
pfti	12697	12778/M	12778	12778/M	12778	12778/M	12778	12801/M	12801
		12801/M	12801	12801/M	12801				
pfti	12714	12722/P	12726	12726	12727/S	12730/P	12732	12750/S	12751
		12766/S	12767/S	12768/S	12770/S	12771/P	12772/P	12774/P	12776/S
		12778/P	12782	12782	12783/S	12784/S	12785/S	12786/S	12787/S
		12794/S	12795	12801/P					
pfti	12894	12934/M	12934	12934/M	12934	12934/M	12934		
pfti	12900	12906/P	12907	12907	12908/S	12918/P	12924/P	12934/P	
pfti	13065	13170/S							
pfti	13072	13081/S							
pfti	13129	13162/M	13166	13166/S	13166	13166/S	13167/S	13168/S	13170/P
		13173/P	13175/P	13176/P	13177/M	13179/S	13184	13191/S	13204/S
		13205/P	13219/S	13226/P	13235/P	13250/P	13251/S	13252/P	
pfti	13272	13356/M	13356	13356/M	13356	13356/M	13356		
pfti	13285	13318/P	13320	13320	13321/S	13321/S	13322/S	13324/S	13325/S
		13326/S	13327/S	13327/S	13328/S	13329/P	13330/P	13331/S	13331/S
		13332/S	13338/S	13339/P	13342/P	13343/P	13346/S	13356/P	
pfti	13377	13505	13505	13505					
pfti	13399	13476/M	13479	13479	13480/S	13481/S	13483/M	13483/S	13502

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		13503/S	13505/P	13505/S	13558	13559/S	13560/S	13561/S	13562/S
		13563/S	13564/S	13570/P	13571/P				
pfti	13751	14064/M	14065	14065	14065	14065/S	14065/S	14066	14066/S
		14067/S	14072/M	14074/S	14075/S	14080/S			
pfti	14120	14135/M	14137	14137	14139/S	14142/S	14143/S	14143/S	14144/S
		14150/S	14157/S	14158/P	14164/S	14172/M			
pfti	14184	14190/M	14191/P	14192/S	14193/S				
pfti	14329	14419/M	14419	14419/M	14419	14419/M	14419		
pfti	14346	14412/P	14415	14415	14416/P	14417/S	14418/P	14419/P	
pfti	14514	14524/M	14525/S						
pfti_first	3372	9764	10047	10048/S	10070/M	12442/M	12463	12465	12465/S
		12651	12651/S						
pfti_index	3373	9740/S	9777	9778	9779/M	9779	9780/S	9782	10047/M
		10050	10051	10052/M	10052	10053/S	10056	10069/M	12442/M
		12465/M	12465	12465/M	12465	12465/S	12465	12465	12476
		12476/M	12476	12476/S	12476	12476	12563	12563/M	12563
		12563/S	12563	12563	12610/S	12612	12612/M	12612	12612/S
		12612	12612	12648	12648/M	12648	12648/S	12648	12648
		12651/M	12651	12651/M	12651	12651/S	12651	12651	12658
		12658/M	12658	12658/S	12658	12658	12778	12778/M	12778
		12778/S	12778	12778	12801	12801/M	12801	12801/S	12801
		12801	12934	12934/M	12934	12934/S	12934	12934	13356
		13356/M	13356	13356/S	13356	13356	14419	14419/M	14419
		14419/S	14419	14419					
pfti_size	9761	9764/M							
pfti_size	12367	12463/M							
pftis	3375	9740/M	9780	10048	10053	10107/M	12447/M	12465	12465
		12476	12563	12610/M	12612	12648	12651	12651	12658
		12778	12801	12934	13356	14419			
pmc$kill_task_flag	5790	5806							
pmc$mainframe_id_size	8635	8632							
pmc$max_signal_contents	5773	5767							
pmc$max_task_id	5548	5545							
pmc$processor_model_number_size	8643	8635	8640						
pmc$processor_serial_num_size	8693	8636	8690						
pmt$binary_mainframe_id	4703	4610	8075	10450					
pmt$condition_identifier	5602	5596							
pmt$cpu_model_number	741	730	737						
pmt$cpu_serial_number	744	731	736						
pmt$initialization_value	818	417	5087						
pmt$mainframe_id	8632	8076	8845	10449					
pmt$program_name	2919	2623	2627	2643	2692	2907	9043		
pmt$sense_switches	4712	4621							
pmt$signal	5729	5723							
pmt$signal_contents	5767	5731							
pmt$signal_id	5734	5730	6059						
pmt$task_id	5545	4814	5540						
pointer	14470	14486/M	14488/P						
pqle	4566	11287	11288	11289/M	11291/S	11293/M	11294/M	11294	11584
		11584	11584/M	11584/S	11584/M	11584/M	11584	13505	13505
		13505/M	13505/S	13505/M	13505/M	13505	13805	13806	13808
		13911/P	13914/P	13924/P	14135				
process_rewrite_success	11426	11458	11654						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
process_write_failure	11378	11424	11652						
processor_state	3809	13835							
psc_all	3366	12643/P	13318/P	14412/P					
psc_all_except_avail	3366	12722/P	12906/P						
psc_nominal_queue	3366	12551/P	12598/P						
pt_full	10648	13230/M	13998	14013/M					
pt_full_aste_p	10650	13228/M	13999	14001/P	14002/P				
pt_full_status	13753	14010/P	14081/P						
pt_full_sva	10649	13229/M	14009/P						
pte_p	11474	11518/M	11519/M	11525/M					
pte_p	11545	11380/M	11422/M	11446/M	11587/M	11603	11606/M	11616/M	11617
		11620/M	11621/M	11629/M	11630/M	11639/M	11647/M	11691/M	11693
		11694/M							
pte_p	12901	12909/M	12911/M	12919	12926/M	12927	12928/M		
pte_p	14183	14187/M	14188	14188	14189	14190			
pti	4512	11441/S	11518/S	11587/S	11674/S	11799/S	12106/S	12647/S	12727/S
		12767/S	12783/S	12909/S	13083/S	13167/S	13168/S	13170/S	13179/S
		13239/S	13240/S	13254/S	13255/S	13324/S	13325/S	13327/S	13331/S
		13338/S	13346/S	14143/S					
pti	11901	11926	11930/S	11943/S	11946/S	11957/S	11966/S	11969/S	11978/S
		11981/S	11988/S	11989/S	11992/S				
pti	12195	12243	12269/S						
pti	13774	14063/M	14064/S	14065/S	14065/S	14068/M	14068	14069	14070/M
		14072/S							
ptk$performance_base	6982	7081	7084	7087	7090	7093	7096	7099	7102
		7105	7108	7111	7114	7117	7120	7123	7126
		7129	7132	7135	7138	7141	7144	7147	7150
		7153	7156	7159	7162	7165	7168	7171	7174
		7177	7180	7183	7186	7189	7192	7195	7198
		7201	7204	7207	7210	7213	7216	7219	7222
		7225	7228	7231	7234	7237	7240	7243	7246
		7249	7252	7255	7258	7261	7264	7267	7270
		7273	7276	7279	7282	7285	7288	7291	7294
		7297	7300						
ptk$pfti_for_swapout	7117	12117							
ptr_type	10287	14470							
purge_all_page_maps	12902	12904/M	12932/M	12937					
purge_map	11893	11921/M	11982/M	11990/M	12019				
pva	5255	11793	11803	11804	11808	11809			
pva	5413	14486							
pva	5427	11797	12987						
pva	7761	12988/P	12990/P	12993/P					
pva	10292	14486/M							
pva	12196	12217/M	12220/M	12224/M	12226/P				
queue	13765	13807	13808/S	13909	13911/S	13911/S	13914/S	13914/S	13915/S
		13919	13920/S	13921/S					
queue_id	3435	11382	11956/P	11965/P	11987/P	12393	12424/M	12430/M	12432/M
		12468/P	12521	12522	12523/P	12594	12661	12768	12924/P
		12996	12996	12997	13252/P	13425	13609/M	14402/M	
queue_id	4509	11287/S	11288/S	11289/S	11291/S	11293/S	11294/S	11294/S	11400
		11403	11445	11517	11522	11579	11584/S	11584/S	11584/S
		11584/S	11584/S	11584/S	11584/S	11590	11603	11615	11628

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11638	11646	11673	11689	11689	11946	11955	11964
		11979	11986	11999	12446	12448	12554	12766	12784
		12785	12910	12923	12923	13332	13503	13505/S	13505/S
		13505/S	13505/S	13505/S	13505/S	13505/S	13561	14192/M	
queue_status	416	13425	13611/M						
r1	4976	10084	13712						
r2	4977	14367/M							
rb	12370	12388/M	12423	12425	12426/P				
rb	12960	12982	12983/M	12988/P	12988/P	12990/P	12990/P	12992	12993/P
		12994	12998/P	13000/P	13002/P	13002/P	13004/P	13004/P	13004/P
		13007/P	13009/P	13011/P					
rb	13066	13135/M	13139/P	13140/P	13142/P	13149/P	13149/P	13150	13154/P
		13185/P	13192/P	13241/P	13261/M				
rb	13273	13291	13293/M	13297/P	13308/P	13309/P	13321	13333	13340/M
		13340	13345	13347/M	13347	13351	13352/M	13352	
rb	13378	13408	13412/S	13412/S	13414/M	13419	13421/P	13422/P	13439/P
		13440/P	13450/P	13451/P	13460/P	13461/P	13469/M	13470/P	13471/P
		13480	13481/M	13489/P	13523	13523	13524/S	13525/S	13525
		13527/S	13527	13528/S	13536	13539/M	13551	13552/P	13553
		13585/M	13586/M	13598/P	13600/P	13605/P	13606/P	13615/P	13620
		13621	13632	13632	13633	13634/P	13635/P	13637/S	13638/P
		13638/P	13640	13652					
rb_init_new_io	12838	12866/M							
rb_status	12839	12871/P	12874/P	12877/P					
rb_wait	12837	12861							
rcount	12580	12600/M	12615/P						
rcount	13286	13339/P							
rcount	13756	13912/P	13915/P	13918	13921	13925/P	13956/P	13958/P	
ready_task_count	4249	13963							
ready_tasks	11099	13841	13860						
reclaim_astes	10647	13991	13992/M						
relink_job_fixed_in_new_queue	14414	14414	14420						
remove_page	12704	12766							
remove_pages_in_tu	11460	11414	11532						
reqcode	7759	12982	12994						
request	1123	12308/M	12309/M	12311/M	12316/M	12317/M			
request	7679	13408	13412/S	13412/S	13419	13620	13621	13632	13632
		13633	13635/P	13652					
request	7727	13291	13333	13345	13351				
request	13417	13417	13430	13435	13446	13453	13466	13473	13486
		13497	13508	13513	13526	13548	13554	13587	13595
		13602	13612	13616	13650				
request_block	14461	14475/M	14491/M	14496/M	14500/M				
residence	3448	9332	9375	10125	11383	11490	12012	12038	13139
		13140	13308	13309	13421	13422	13439	13440	13450
		13451	13460	13461	13470	13471	13490	13491	13552
		13560	13598	13605	13606	13715	14048	14049	14157
		14258	14307	14371	14392	14404			
residence	9330	9332/M	9336	9337					
residence	9368	9375/M	9375	9375					
residence	11460	11490/M	11490	11490					
residence	13065	13139/M	13139	13139					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
residence	13272	13308/M	13308	13308					
residence	13377	13421/M	13421	13421	13439/M	13439	13439	13450/M	13450
		13450	13460/M	13460	13460	13470/M	13470	13470	13490/M
		13490	13490	13552/M	13552	13552	13598/M	13598	13598
		13605/M	13605	13605					
residence	13688	13715/M	13715	13715					
residence	13737	14049/M	14049	14049					
residence	14111	14157/M	14157	14157					
residence	14211	14258/M	14258	14258					
residence	14277	14307/M	14307	14307					
residence	14329	14371/M	14371	14371	14392/M	14392	14392	14404/M	14404
		14404							
ring	654	11797	12987						
ring	13690	13712							
rma	997	11570	12004/M	12121/M	12283/M	12312	14524		
rma	3399	11930	12269	13212	14064	14072	14188	14189	14190
rma	12197	12305/P	12308	12316					
rma_list	2261	12203							
rmt$density	2485	2320							
sac_p	11774	11782/M	11783/M	11802/M	11804/M	11809/M			
save_v	13287	13324/M	13338	13346					
save_valid	13130	13167/M	13179	13240					
scheduling_dispatching_priority	3917	13965/S							
scheduling_memory_levels	10364	13817	13820						
sdt_offset	4816	9862	9875	9950	12426	13708	14230	14231	14293
		14357	14358						
sdt_p	9858	9862/M							
sdt_p	9870	9875/M							
sdt_p	10077	10080	10084						
sdt_p	13688	13708/M							
sdt_p	13701	13708/P	13712						
sdt_p	14211	14230/M	14231/M						
sdt_p	14222	14231/P	14235/P	14253	14256/M				
sdt_p	14277	14293/M							
sdt_p	14282	14293/P	14298	14300	14302/M	14303/S			
sdt_p	14329	14357/M	14358/M						
sdt_p	14347	14358/P	14365/M	14366/M	14367/M	14373	14383	14390/M	14400/S
		14407							
sdte_p	12385	12426/M	12427/M	12429/M	12430/P				
sdte_segment_number	11775	11793/M	11794/P						
sdtx_offset	4817	9876	9937	11794	13489	13708	14230	14231	14293
		14357	14358						
sdtx_p	9871	9876/M							
sdtx_p	13688	13708/M							
sdtx_p	13702	13708/P	13711	13715/P					
sdtx_p	14211	14230/M	14231/M						
sdtx_p	14224	14231/P	14255	14257	14258/P				
sdtx_p	14277	14293/M							
sdtx_p	14284	14293/P	14299						
sdtx_p	14329	14357/M	14358/M						
sdtx_p	14348	14358/P	14388	14389	14392/P	14401			
sdtx_table	5026	13711	13715/P	14255	14255	14257	14258/P	14299	14370/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		14388	14389	14389	14392/P	14401			
sdtxe_p	11776	11794/M	11795/M						
sdtxe_p	13401	13489/M	13490/P	13491/P					
seg	9328	9335/M	9338/M	9344					
seg	9368	9375/M	9375/M	9375					
seg	11460	11490/M	11490/M	11490					
seg	13065	13139/M	13139/M	13139					
seg	13272	13308/M	13308/M	13308					
seg	13377	13421/M	13421/M	13421	13439/M	13439/M	13439	13450/M	13450/M
		13450	13460/M	13460/M	13460	13470/M	13470/M	13470	13490/M
		13490/M	13490	13552/M	13552/M	13552	13598/M	13598/M	13598
		13605/M	13605/M	13605					
seg	13688	13715/M	13715/M	13715					
seg	13737	14049/M	14049/M	14049					
seg	14111	14157/M	14157/M	14157					
seg	14211	14258/M	14258/M	14258					
seg	14277	14307/M	14307/M	14307					
seg	14329	14371/M	14371/M	14371	14392/M	14392/M	14392	14404/M	14404/M
		14404							
segment	5597	11804/M	11809/M						
segment_link	4506	11529	12457	13483					
segment_lock	409	13101	13170						
segment_number	7714	12425	12426/P						
segment_number	10075	10081							
segment_table_address_1	5394	14236/M							
segment_table_address_2	5396	14237/M							
segment_table_length	5390	13710	14239	14239	14240	14242	14287		
segnum	7702	13489/P							
segnum	9834	9837							
segnum	9947	9950							
segnum	11765	11794							
segnum	12367	12426							
segnum	13377	13489							
segnum	13703	13710	13711/S	13712/S	13714	13715/S			
segnum	14225	14252	14253/S	14254/S	14255/S	14255/S	14256/S	14256/S	14257/S
		14258/S							
segnum	14288	14297	14298/S	14299/S	14300/S	14302/S	14303/S		
segnum	14349	14382	14383/S	14388/S	14389/S	14389/S	14390/S	14390/S	14392/S
seqno	520	10230/M	11786/M						
sequence_number	11135	10230	11786						
server_file	7715	12423							
set_length_to_zero	13691	13717							
sfd_p	3933	12104							
sfid	3436	10124	11383	11490/P	12012	12038	13140	13309	13321
		13422	13440	13451	13461	13471	13491	13525	13527
		13538	13551/M	13560	13606	14048	14049/P	14157/P	14307/P
		14401/M	14404/P						
sfid	4996	13490/P	13491/P	13715/P	14255	14255	14258/P	14370/P	14389
		14389	14392/P	14401					
sfid	7700	13421/P	13422/P	13439/P	13440/P	13450/P	13451/P	13460/P	13461/P
		13598/P	13605/P	13606/P					
sfid	7726	13139/P	13140/P	13308/P	13309/P	13321			
sfid	9320	9332	9333	9334	9345				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
sfid	9368	9375/P							
sfid	9368	9375	9375	9375					
sfid	10118	10124	10125						
sfid	11460	11490	11490	11490					
sfid	13065	13139/P							
sfid	13065	13139	13139	13139					
sfid	13065	13140	13140						
sfid	13272	13308/P							
sfid	13272	13308	13308	13308					
sfid	13272	13309	13309						
sfid	13377	13421/P	13439/P	13450/P	13470/P	13490/P	13552/P	13598/P	13605/P
sfid	13377	13421	13421	13421	13421	13439	13439	13439	13439
		13450	13450	13450	13450	13460	13460	13460	13460
		13470	13470	13470	13470	13490	13490	13490	13490
		13552	13552	13552	13552	13598	13598	13598	13598
		13605	13605	13605	13605				
sfid	13377	13422	13422	13440	13440	13451	13451	13461	13461
		13471	13471	13491	13491	13606	13606		
sfid	13688	13715/P							
sfid	13688	13715	13715	13715	13715				
sfid	13737	14049/P							
sfid	13737	14049	14049	14049	14049				
sfid	14111	14157/P							
sfid	14111	14157	14157	14157	14157				
sfid	14211	14258	14258	14258	14258				
sfid	14277	14307/P							
sfid	14277	14307	14307	14307	14307				
sfid	14329	14371/P	14404/P						
sfid	14329	14371	14371	14371	14371	14392	14392	14392	14392
		14404	14404	14404	14404				
sft$counter	4278	4247	4248	4622	4624	4625	4627		
sft$file_space_limit_kind	5239	5003							
software_attribute_set	4999	13711							
soon	10755	9904	11295/M	11295	11584/M	11584	11604/M	11604	11618/M
		11618	11980/M	11980	12917/M	12917	13504/M	13504	13505/M
		13505	13897	13899	13929	14032	14137	14137	
source_aste_p	13131	13157/M	13162	13177	13233	13235/P			
source_sva	13132	13147/M	13204/M	13234	13235/P	13261			
spd_index	4458	12063	12104/S	12124/M	12124				
spde_p	11902	12104/M	12105/M	12106/M	12107/M	12113/M	12113		
specified	7811	12734	12752	12796	12805	12977/M	13410/M	14131/M	
st	4958	13712	14253	14254	14256/M	14256	14298	14300	14302/M
		14303/S	14365/M	14365	14366/M	14367/M	14368/M	14369/M	14373
		14383	14390/M	14390	14400/S	14407			
st	10512	13590/M							
st	10522	13592/M							
st_rma	14223	14235/P	14236	14237					
stack_for_ring	421	10082							
stack_segment_number	13692	13714/M							
statistics	3938	13948	13950	13950	13963				
status	3597	14475/M	14491/M						
status	7675	12388/M	13414/M	13615/P	13638/P				
status	7733	13135/M	13142/P	13149/P	13150	13185/P	13192/P	13241/P	

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
status	7760	12983/M	12988/P	12990/P	12992	12998/P	13000/P	13004/P	13011/P
status	9499	9505/P							
status	9805	9819/P							
status	9838	9843/P							
status	10187	10190/M	10191/M						
status	11375	11550/M							
status	11460	11488/P							
status	11777	11796/P							
status	11876	12048/M	12048/M						
status	11882	11906/M	12046/P	12048/P					
status	12173	12265/M	12265/M						
status	12175	12201/M	12226/P	12227	12260/P	12261	12265/P		
status	12835	12871/M	12871/M	12874/M	12874/M	12877/M	12877/M		
status	12959	12998/M	12998/M	13000/M	13000/M	13011/M	13011/M		
status	13065	13142/M	13142/M	13185/M	13185/M	13192/M	13192/M	13241/M	13241/M
status	13272	13322/P							
status	13377	13615/M	13615/M						
status	13737	14003/P	14075/P						
status	14111	14144/P							
status_loop	12781	12781	12802						
ste	4945	10080	10084	12429/M	13590/M	13592/M	13712	14253	14254
		14298	14300	14302/M	14366/M	14367/M	14368/M	14369/M	14373
		14383							
ste_p	12967	12993/P							
ste_p	14471	14488/P							
stop_pfti	4461	12066/M	12073	12077	12080/M	12091	12091		
strn	9284	9298							
strng	9280	9283							
stxe_p	12968	12993/P							
stxe_p	14472	14488/P							
sva	974	11915	12007						
sva	4517	11195/P	11447/P	11477	11480	11482	11483	11491	11501
		11501	11692/P	12113	12665	12666	12666	13204	13221/M
		13234/M	13326/P	13480	13481	13562	14080/P	14143	14150
		14164	14193/M						
sva	11259	11264							
sva	11271	11276							
sva	11368	11692							
sva	11426	11447							
sva	11903	11915/M	11916	11926	12025/M	12025			
sva	12198	12226/P	12236	12243	12292/M	12292	12296		
sva	12544	12551/P							
sva	12577	12598/P							
sva	12629	12643/P	12675/P						
sva	12698	12722/P							
sva	12894	12906/P	12938/P						
sva	12894	12938							
sva	12974	12993/P	13002/P	13007/P	13009/P				
sva	13272	13326							
sva	13402	13415/M	13433/M	13443/M	13456/M	13494/M	13594	13619/M	13634/P
		13647/P							
sva	14287	14296/M	14300/M	14301	14309/P				
sva	14350	14373/M	14374/M	14375	14410	14412/P			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
sva	14473	14488/P	14489					
swap_data	3931	11560/M	12067/M					
swap_file_descriptor_pfti	4462	12066	12075					
swap_io_control	3932	12063	12064/M	12065/M	12066/M	12066	12073	12073	12074
		12075/M	12075	12076	12077	12080/M	12081/M	12082/S	12084
		12085/M	12087/M	12087	12091	12091	12091	12091	12092/S
		12104/S	12116	12121	12124/M	12124	12125/M		
swap_status	3907	9815	11401	11444	11476	11520	11718	12404	12404
		12460	12460	12479	12479	12480	13322	13565	13565
		13577	13578	13579	13939	13960	14003	14075	14144
swapout_io_not_initiated	10756	9907	13897						
swapped_page_descriptors	4472	12104							
swapping_io_error	3969	11560/M	12067/M						
syc$rc_conditional_free	3694	13008							
syc$rc_free_pages	3636	13006							
syc$rc_write_modified_pages	3637	12995							
syc$ucr_condition	5640	5651							
syc$user_defined_condition	5641	5653							
system_ijle_p	12386	12436/P	12451/M	12451	12453/M	12453	12471/M	12471	12473/M
		12473							
system_ijle_p	12510	12515/P	12526/M	12526	12528/M	12528			
system_jws	13777	13801/M	13803/P	13806					
syt$180_idle_code	5878	3824							
syt$monitor_flag	3760	5552	10272						
syt$monitor_flags	5552	4788	11142						
syt$monitor_request_code	3608	3596	3724	7674	7725	7744	7759	8569	9970
		10863							
syt$monitor_status	1784	1431	3597	3725	7675	7733	7745	7760	8570
		9499	9531	9838	10144	10169	10187	10273	10864
		11375	11777	11882	12175	12181	12839	13704	
syt$perf_keypoints_enabled	7783	7779							
syv$perf_keypoints_enabled	7779	12115							
syv$user_templates	11188	13628							
tape_command	1138	12308/M	12309/M	12311/M	12316/M	12317/M			
tape_request_p	12174	12203	12205	12212	12214	12216	12217	12220	12223
		12224	12238	12306	12308/M	12309/M	12311/M	12316/M	12317/M
		12334	12338/M	12339/M					
target	10380	13817	13820						
task_queue	4513	11429	11429	11430/P	11665	11665	11666/P	11786/P	12851/P
		12865/P							
task_queue	10225	10229	10230/S						
task_queue	11765	11786	11786/S						
taskid	3812	12851/P	12865/P						
taskid	10226	10229/M	10230/M						
taskid	11547	11430/P	11666/P	11754/P					
taskid	11765	11786/M	11786/M						
taskid	11778	11786/P	11787	11787	11788/P	11796/P	11805/P	11810/P	11817/P
taskid	13754	14023/P							
taskid	14226	14251/M	14259						
taskid	14351	14381/M	14393						
temp_max_working_set_size	13766	13811/M	13813/M	13815					
temp_target	13767	13815/M	13817	13818					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
time_last_full_jws_scan	13770	13935	13981/M					
time_last_idle_dispatching_scan	13768	13833	13885/M					
time_last_shared_queue_scan	13769	13905	13926/M					
time_next_free_astes	13771	14039	14056/M					
time_next_scan_wait_not_queued	13772	13891	13892/M	13893/P				
time_spent_in_job_mode	4233	13851	13864	13868	13948	13950		
time_spent_in_mtr_mode	4234	13851	13865	13869	13951			
timestamp	10328	13849/M	13870/M	13873	13884/M			
tmc$	5985	5991	5994	5997	6000	6003	6006	6009	6012
		6015	6018	6021	6024	6027	6030	6033	6036
tmc$broken_task_fault_id	5568	5618							
tmc$btc_invalid_a0	5666	5687							
tmc$btc_invalid_p	5666	5687							
tmc$btc_mcr_traps_disabled	5667	5688							
tmc$btc_mf_traps_disabled	5666	5686							
tmc$btc_mntr_fault_buffer_full	5665	5686							
tmc$btc_system_error	5668	5682							
tmc$btc_ucr_traps_disabled	5667	5688							
tmc$dummy_fault	5569	5624							
tmc$flag_available_31	5803	5807							
tmc$maximum_monitor_faults	5573	5564							
tmc$maximum_signals	5783	5780							
tmc$maximum_system_task_id	5816	5819							
tmc$mcr_fault	5568	5620							
tmc$rc_ready_conditional_wi	5967	11720/P							
tmc$signal_available_63	5765	5776							
tmc$stid_null_task	5822	5819							
tmc$ts_io_wait_not_queued	5959	5941							
tmc$ts_page_wait	5961	5942	12851/P	12865/P					
tmc$ts_ready	5946	5937							
tmc$ts_timed_wait_not_queued	5953	5940							
tmc$ts_timeout_reqexp_longvlong	5951	5939							
tmc$ts_timeout_reqexp_shortshrt	5950	5938							
tmp$cause_task_switch	10200	12408	12855	13568					
tmp$check_for_swapout_candidate	10203	13964							
tmp$check_timed_wait_not_queued	10208	13893							
tmp$clear_lock	9536	9510	9549	9642	9820	11488	11494	11814	13322
		13349	13942	13976	13978	14003	14075	14144	14152
		14159							
tmp$dequeue_task	10212	11430	11666	11754	14023				
tmp$find_next_queued_task	10218	11817							
tmp$get_taskid_from_task_queue	10224	10231	11786						
tmp$get_xcb_p	10233	11788							
tmp$idle_non_dispatchable_job	10240	13967							
tmp$obtain_ijl_ordinal_from_ptl	10247	13706							
tmp$queue_task	10254	12851	12865						
tmp$reissue_monitor_request	10261	12407	12852	12854	12867	13567			
tmp$send_monitor_fault	10264	11805	11810						
tmp$set_lock	9572	9501	9590	9635	9818	11488	11494	11814	13322
		13349	13938	13976	14003	14075	14144	14152	14159
tmp$set_monitor_flag	10271	11796							
tmp$set_task_ready	10277	11719							
tmt$broken_task_condition	5665	5681							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
tmt$broken_task_monitor_fault	5679	5619						
tmt$cpu_execution_statistics	11048	11043						
tmt$dispatch_control	5908	3819						
tmt$dispatching_control_sets	11096	11086						
tmt$dispatching_controls	11060	11055						
tmt$dispatching_prio_controls	11070	11065						
tmt$dispatching_priority_time	11075	11072						
tmt$dual_state_priority_entry	5975	3806	5972					
tmt$idle_status	5964	11140						
tmt$mcr_faults	5704	5621						
tmt$monitor_fault_buffer	5558	4825						
tmt$monitor_fault_buffers	5564	5559	5560	5561				
tmt$monitor_fault_identifiers	5567	5617	5693					
tmt$primary_task_list	11151	11117						
tmt$primary_task_list_entry	11133	11151						
tmt$ptl_flags	11157	11144						
tmt$ptl_lock	9560	9536	9572	9623				
tmt$ready_condition	5967	10279						
tmt$signal	5721	5716						
tmt$signal_buffer	5713	4826						
tmt$signal_buffers	5780	5714	5715	5716				
tmt$system_flags	5786	4801	11143					
tmt$system_task_id	5819	4792						
tmt$task_queue_link	557	550	4513	10212	10225	10256	10692	11141
tmt$task_status	5946	5911	10255	11138	11139			
tmt$time_limits	11077	11075						
tmt$wait_inhibited	11164	11159						
tmt$xcb_offset_size	11155	11136						
tmv$cpu_execution_statistics	11043	13850	13851	13864	13865	13867	13868	
tmv$dispatching_control_sets	11086	13841	13860					
tmv$dispatching_controls	11055	13859						
tmv$long_wait_force_swap_time	11112	13962						
tmv$null_global_task_id	10583	11787	11787					
tmv$ptl_lock	9623	9501/P	9510/P	9635/P	9642/P	9818/P	9820/P	11488/P 11488/P
		11494/P	11494/P	11814/P	11814/P	13322/P	13322/P	13349/P 13349/P
		13938/P	13942/P	13976/P	13976/P	13978/P	14003/P	14003/P 14075/P
		14075/P	14144/P	14144/P	14152/P	14152/P	14159/P	14159/P
tmv$ptl_p	11117	10230	11786					
tmv$timed_wait_not_queued	11171	13892						
tos_registers	5402	14486						
total_idle	13762	13836/M	13840	13883				
total_list_entries	12199	12202/M	12228	12230/P	12238	12251	12259/P	12275 12283/S
		12291/S	12296/S	12302/M	12302	12305/S	12312/S	12334 12335/S
		12339						
transfer_length	3344	12223						
trap_enable	5359	11797						
u	3396	11620/M	11630/M	12927	12928/M			
unblocked_priorities	10318	13841	13846/M	13876/M	13877			
unlock_pages	11565	11565	11568	11704				
unrecovered_files	7709	13585/M						
unrecovered_files	13404	13574/M	13574	13585				
unrecovered_pages	7708	13586/M						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
unrecovered_pages	13405	13576/M	13576	13586					
upper	11033	14189							
user_dp_set	13763	13846	13859						
v	3394	11380/M	11446/M	11519/M	11616/M	11629/M	11639/M	11647/M	11691/M
		11694/M	11799/M	11947	11957/M	11966/M	11969	11981	11988
		11989/M	12647/M	12919	13167	13168/M	13179/M	13240/M	13254/M
		13324	13325/M	13338/M	13346/M	14188	14490		
v1	4972	13590/M	13592/M	14253	14254	14298	14366/M	14368/M	14383
wait_for_io_complete	7676	13637/S	13640						
waitopt	7763	13004/P							
wired_command_heap_p	1114	12203							
wired_read_description_p	1113	12217	12220						
wired_write_description_p	1112	12223	12224						
wmp_status	12709	12718/M	12739/M	12741/M	12743/M	12745/M	12788/M	12790/M	12806
		12807/M	12810	12811	12812/M				
wmp_status	12835	12845							
wmp_status	12965	13003/P	13004/P						
wmp_status	13403	13636/P	13637/P	13639	13639	13640			
working_set	11304	11310/M							
working_set	13737	13803/M							
wp	4975	10080							
write_loop	12725	12725	12746	12779					
write_ok	14127	14146/M	14151/M	14156					
write_status	12713	12730/P	12731	12738	12740	12742			
write_status	14119	14158/P	14160	14162					
write_to_disk	12728	12728	12749						
ws_disk_flaws	10162	12738							
ws_ok	10162	12731	14162						
ws_physical_io_reject	10162	14160							
ws_server_terminated	10163	12742							
ws_volume_unavailable	10163	12740							
xcb_p	3817	12426/P	12987	13489/P	14486				
xcb_p	9857	9862	9862						
xcb_p	9869	9875	9875	9876	9876				
xcb_p	9933	9936	9937						
xcb_p	9946	9949	9950						
xcb_p	11765	11794	11794						
xcb_p	11779	11788/P	11789	11791/P	11793	11794/P	11797	11797	11803
		11804	11808	11809					
xcb_p	12367	12426	12426						
xcb_p	13377	13488	13488						
xcb_p	13688	13708	13708	13708	13708				
xcb_p	13689	13706/P	13708/P	13710					
xcb_p	14211	14230	14230	14230	14230	14231	14231	14231	14231
xcb_p	14213	14231/P	14236/M	14237/M	14239	14240	14251		
xcb_p	14277	14293	14293	14293	14293				
xcb_p	14278	14293/P	14297						
xcb_p	14329	14357	14357	14357	14357	14358	14358	14358	14358
xcb_p	14333	14358/P	14381	14406					
xcb_segnum_relative_jobs_as	14331	14365/S	14368/S	14369/S	14370/S				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xp	4787	11797	11797	12987	13710	14236/M	14237/M	14239	14239
		14240	14242	14297	14486				
xpfti	9770	9785/M							
xpfti	10040	10059/M							
xpfti	12367	12465/M							
xpfti	12367	12476/M							
xpfti	12543	12563/M							
xpfti	12576	12612/M							
xpfti	12628	12648/M	12658/M						
xpfti	12628	12651/M							
xpfti	12697	12778/M	12801/M						
xpfti	12894	12934/M							
xpfti	13272	13356/M							
xpfti	14329	14419/M							
xsva	13288	13315/M	13316/M	13318/P					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter