```
                3 MODULE mmm$mtr_user_request_processor [MMMMUR] ;
                4
                5
                6 {
                7 {  PURPOSE:
                8 {     This module contains request processors that deal with interfacing
                9 {     job mode mem mgr requests to mem mgr in mtr mode, locking and unlocking
               10 {     pages, locking and unlocking segments, and setting segment lengths.
               11 {
               12
               13
        o   6677
        o   6678
        o   6679 {  External procedures referenced by this module.
        o   6680
        o   6681
        o   6682   PROCEDURE [XREF] dfp$fetch_page_status
        o   6683      (     fde_p: gft$locked_file_desc_entry_p;
        o   6684            offset: ost$segment_offset;
        o   6685       VAR allocate_status: gft$page_status);
        o   6686
            6707
            6708   PROCEDURE [XREF] dmp$fetch_page_status
            6709      (     fde_p: gft$locked_file_desc_entry_p;
            6710            offset: ost$segment_offset;
            6711            enforce_limits: sft$file_space_limit_kind;
            6712            allow_allocation: boolean;
            6713       VAR allocate_status: gft$page_status);
            6714
        o   6717
        o   6718   PROCEDURE [INLINE] dmp$get_disk_file_descriptor_p (fde_p: gft$file_desc_entry_p;
        o   6719     VAR dfd_p: ^dmt$disk_file_descriptor);
        o   6720
            6758 ?? FMT (FORMAT := ON, KEYW := UPPER, IDENT := LOWER) ??
        o   6774
        o   6775   PROCEDURE [INLINE] dmp$get_fau_entry (p_dfd: ^dmt$disk_file_descriptor;
        o   6776         byte_address: amt$file_byte_address;
        o   6777     VAR p_fau_entry: ^dmt$file_allocation_unit);
        o   6778
            6807
            6808   FUNCTION [INLINE] gfp$mtr_convert_job_mode_fde_p (xfde_p: gft$file_desc_entry_p;
            6809         cst_p: ^ost$cpu_state_table): gft$file_desc_entry_p;
        o   6822
        o   6823   PROCEDURE [INLINE] gfp$mtr_get_fde_p (sfid: gft$system_file_identifier;
        o   6824         ijle_p: ^jmt$initiated_job_list_entry;
        o   6825     VAR fde_p: gft$file_desc_entry_p);
        o   6826
        o   6827
        o   6868
        o   6869   PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
        o   6870         ijle_p: ^jmt$initiated_job_list_entry;
        o   6871     VAR fde_p: gft$locked_file_desc_entry_p);
        o   6872
        o   6873
        o   6986
        o   6987   PROCEDURE [INLINE] gfp$mtr_get_sfid_from_fde_p
```



```
        o   6988      (     fde_p: gft$locked_file_desc_entry_p;
        o   6989       VAR sfid: gft$system_file_identifier;
        o   6990       VAR ijl_ordinal: jmt$ijl_ordinal);
        o   7047
        o   7048
        o   7049   PROCEDURE [INLINE] gfp$mtr_unlock_fde_p (fde_p: gft$locked_file_desc_entry_p);
        o   7050
        o   7051
        o   7081
        o   7082   PROCEDURE [INLINE] i#move (source: ^cell;
        o   7083         dest: ^cell;
        o   7084         length: 0 .. 7fffffff(16));
        o   7085
        o   7086     VAR
        o   7087       str1: ^string (65535),
        o   7088       str2: ^string (65535);
        o   7089
        o   7091     IF length <> 0 THEN
        o   7092       str1 := source;
        o   7093       str2 := dest;
        o   7094       str2^ (1, length) := str1^ (1, length);
        o   70`5       #SPOIL (str2^);
        o   7096     IFEND;
        o   7098   PROCEND i#move;
        o   7099
        o   7100   PROCEDURE [XREF] i#real_memory_address (p: ^cell;
        o   7101     VAR rma: integer);
        o   7102
        o   7103   PROCEDURE [INLINE] jmp$unlock_ajl
        o   7104     (     ijle_p: ^jmt$initiated_job_list_entry);
        o   7105
        o   7106     VAR
        o   7107       ajlo: jmt$ajl_ordinal;
        o   7108
        o   7109     tmp$set_lock (tmv$ptl_lock);
        o   7110     ajlo := ijle_p^.ajl_ordinal;
        o   7111     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
        o   7112       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
        o   7113     ELSE
        o   7114       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
        o   7115     IFEND;
        o   7116     tmp$clear_lock (tmv$ptl_lock);
        o   7117
        o   7118   PROCEND jmp$unlock_ajl;
        o   7119
            7198
            7199   PROCEDURE [XREF] mmp$asid (asti: mmt$ast_index;
            7200     VAR asid: ost$asid);
            7201
        o   7204
        o   7205   PROCEDURE [XREF] mmp$aste_pointer (asid: ost$asid;
        o   7206     VAR aste_p: ^mmt$active_segment_table_entry);
        o   7207
            7210
            7211   PROCEDURE [XREF] mmp$convert_pva (p: ^cell;
            7212         cst_p: ^ost$cpu_state_table;
```

```
7213      VAR sva: ost$system_virtual_address;
7214      VAR fde_p: gft$locked_file_desc_entry_p;
7215      VAR aste_p: ^mmt$active_segment_table_entry;
7216      VAR ste_p: ^mmt$segment_descriptor;
7217      VAR stxe_p: ^mmt$segment_descriptor_extended);
7218
o 7221
o 7222    PROCEDURE [XREF] mmp$delete_pt_entry
o 7223      (     pfti: mmt$page_frame_index;
o 7224            unlink_page_from_segment: boolean);
o 7225
7228
7229    PROCEDURE [INLINE] mmp$fetch_pfti_array_size
7230      (VAR pfti_size: integer);
7231
o 7237
o 7238    PROCEDURE [INLINE] mmp$find_next_pfti
o 7239      (VAR xpfti: mmt$page_frame_index);
o 7240
7258
7259    PROCEDURE [INLINE] mmp$get_max_sdt_sdtx_pointer
7260      (     xcb_p: ^ost$execution_control_block;
7261       VAR sdt_p: mmt$max_sdt_p;
7262       VAR sdtx_p: mmt$max_sdtx_p);
7263
o 7272
o 7273  { This procedure verifies that the asti stored in the file descriptor entry is still being used by
o 7274  { the same job for the same file.  If the asti is ok, it is returned; otherwise 0 is returned.
o 7275
o 7276    PROCEDURE [INLINE] mmp$get_verify_asti_in_fde
o 7277      (     fde_p: gft$locked_file_desc_entry_p;
o 7278            sfid: gft$system_file_identifier;
o 7279            ijlo: jmt$ijl_ordinal;
o 7280       VAR asti: mmt$ast_index);
o 7281
7293
7294   PROCEDURE [XREF] mmp$initialize_find_next_pfti (xsva: ost$system_virtual_address;
7295         length: ost$segment_length;
7296         end_point_option: (include_partial_pages, exclude_partial_pages);
7297         page_selection_criteria: mmt$page_selection_criteria;
7298         aste_p: ^mmt$active_segment_table_entry;
7299    VAR xpfti: mmt$page_frame_index);
7300
o 7303    PROCEDURE [XREF] mmp$mm_free_pages (sva: ost$system_virtual_address;
o 7304          length: ost$segment_length;
o 7305          aste_p: ^mmt$active_segment_table_entry;
o 7306          free_asid: boolean;
o 7307      VAR count: integer);
o 7308
o 7309
7312    PROCEDURE [XREF] mmp$mm_write_modified_pages
7313      (     sva: ost$system_virtual_address;
7314            length: ost$segment_length;
7315            fde_p: gft$locked_file_desc_entry_p;
7316            aste_p: ^mmt$active_segment_table_entry;
7317            iotype: iot$io_function;
```



```
7318            init_new_io: boolean;
7319            remove_page: boolean;
7320            io_id: mmt$io_identifier;
7321       VAR io_count: mmt$active_io_count;
7322       VAR io_already_active: boolean;
7323       VAR last_written_pfti: mmt$page_frame_index;
7324       VAR wmp_status: mmt$write_modified_pages_status);
7325
o 7333
o 7334
o 7335    PROCEDURE [INLINE] mmp$purge_all_cache_map;
o 7336
o 7337      VAR
o 7338        null_sva: 0 .. 0ffffffffffff(16);
o 7339
o 7340      IF mmv$multiple_caches OR mmv$multiple_page_maps THEN
o 7341        mmp$purge_all_cache_map_proc;
o 7342      ELSE
o 7343        #purge_buffer (osc$purge_all_cache, null_sva);
o 7344        #purge_buffer (osc$purge_all_page_seg_map, null_sva);
o 7345      IFEND;
o 7346
o 7347    PROCEND;
7369
7370    PROCEDURE [XREF] mmp$process_wmp_status (wmp_status: mmt$write_modified_pages_status;
7371          last_written_pfti: mmt$page_frame_index;
7372          rb_wait: ost$wait;
7373          VAR rb_init_new_io: boolean;
7374          VAR rb_status: syt$monitor_status);
7375
o 7378
o 7379    PROCEDURE [XREF] mmp$relink_page_frame (pfti: mmt$page_frame_index;
o 7380          queue_id: mmt$page_frame_queue_id);
o 7381
7384    PROCEDURE [XREF] mmp$remove_pages_working_set (sva: ost$system_virtual_address;
7385          length: ost$segment_length;
7386          aste_p: ^mmt$active_segment_table_entry;
7387      VAR count: integer);
7388
o 7391
o 7392    PROCEDURE [XREF] mmp$xtask_pva_to_sva (pva: ^cell;
o 7393      VAR sva: ost$system_virtual_address;
o 7394      VAR status: syt$monitor_status);
o 7395
7398    PROCEDURE [XREF] mmp$verify_pva (pointer_to_pva: ^cell;
7399          access: mmt$segment_access_type;
7400      VAR status: syt$monitor_status);
7401
o 7408    PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
7419
7420    PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
7421          condition: osc$max_status_condition_number + 1 .. 0fffffffffff(16);
7422      VAR status: syt$monitor_status);
7423
o 7433
o 7434  {  External variables referenced by this module.
```

```
o  7435
o  7436 {Pointer to the Active Segment Table - (AST).}
o  7437
o  7438   VAR
o  7439     mmv$ast_p: [XREF] ^mmt$active_segment_table;
o  7440
   7443 {The following variable indicates if the configuration consists of multiple
   7444 {caches that are not hardware connected for unified cache purging - ie, if a cache
   7445 {purge is required each processor must purge its own cache.
   7446
   7447   VAR
   7448     mmv$multiple_caches: [XREF] boolean;
   7449
   7450 {The following variable indicates if the configuration consists of multiple
   7451 {page MAPS that are not hardware connected for unified map purging - ie,
   7452 {if a page map purge is required each processor must purge its own map.
   7453
   7454   VAR
   7455     mmv$multiple_page_maps: [XREF] boolean;
   7456
   7457 {Define pointer to array for holding PFTI lists. This array is used in monitor for holding lists
   7458 {PFTIs of pages belonging to a segment.
   7459
   7460   VAR
   7461     mmv$pfti_array_p: [XREF] ^mmt$pfti_array;
   7462
   7464
   7465
   7466   TYPE
   7467     mmt$pfti_array = RECORD
   7468       pfti_first: 0 .. osc$max_page_frames,
   7469       pfti_index: 0 .. osc$max_page_frames,
   7470       last_pfti_index: 0 .. osc$max_page_frames,
   7471       pftis: ARRAY [0 .. *] of mmt$page_frame_index,
   7472     RECEND;
   7474 {Pointer to the 'PAGE FRAME TABLE' (PFT)
   7475
   7476   VAR
   7477     mmv$pft_p: [XREF] ^mmt$page_frame_table;
   7478
o  7481
o  7482 { Global Page Queue List array.
o  7483
o  7484   VAR
o  7485     mmv$gpql: [XREF] mmt$global_page_queue_list;
   7488
   7489 {Pointer to the system PAGE TABLE (PT).
   7490
   7491   VAR
   7492     mmv$pt_p: [XREF] ^ost$page_table;
   7493
o  7496
o  7497   PROCEDURE [XREF] tmp$dequeue_task (VAR queue_link: tmt$task_queue_link;
o  7498     VAR taskid: ost$global_task_id);
o  7499
   7502
```



```
   7503   PROCEDURE [INLINE] tmp$get_taskid_from_task_queue
   7504     (     task_queue: tmt$task_queue_link;
   7505       VAR taskid: ost$global_task_id);
   7506
o  7512
o  7513   PROCEDURE [XREF] tmp$mtr_begin_lock_activity
o  7514     (     xcb_p: ^ost$execution_control_block;
o  7515           activity: 1 .. 256);
o  7516
   7519
   7520   PROCEDURE [XREF] tmp$mtr_end_lock_activity
   7521     (     cst_p: ^ost$cpu_state_table;
   7522           activity: 1 .. 256;
   7523       VAR xcb_p: ^ost$execution_control_block);
   7524
o  7527   PROCEDURE [XREF] tmp$get_xcb_p (task_id: ost$global_task_id;
o  7528     VAR xcb_p: ^ost$execution_control_block;
o  7529     VAR ijle_p: ^jmt$initiated_job_list_entry);
o  7530
   7533
   7534   PROCEDURE [XREF] tmp$queue_task (taskid: ost$global_task_id;
   7535         task_status: tmt$task_status;
   7536     VAR queue_link: tmt$task_queue_link);
   7537
o  7540
o  7541   VAR
o  7542     tmv$ptl_p: [XREF] ^tmt$primary_task_list;
o  7543
   7593 {System page size.}
   7594
   7595   VAR
   7596     osv$page_size: [XREF] ost$page_size;
   7597
o  7600
o  7601
o  7602
o  7603 {  Procedures local to this module.
o  7604
```

CONVERT_SVA_TO_PFTE_P

```
        0  7607
        0  7608    PROCEDURE convert_sva_to_pfte_p
        4  7609      (     sva: ost$system_virtual_address;
        4  7610       VAR pfte_p: ^mmt$page_frame_table_entry;
        4  7611       VAR status: syt$monitor_status);
        4  7612
        4  7613
        4  7614 {
        4  7615 {  This procedure returns a pointer to the page frame table entry for a specified pva.
        4  7616 {
        4  7617
        4  7618      VAR
        4  7619        count: 1 .. 32,
        4  7620        hash_sva_param2: integer, {Kludge for compiler bug
        4  7621        pfti: mmt$page_frame_index,
        4  7622        pti: ost$page_table_index;
        4  7623
        4  7624
        4  7625      #HASH_SVA (sva, hash_sva_param2, count, status.normal);
        6  7626      pti := hash_sva_param2; {Kludge for compiler bug
        6  7627      IF status.normal = FALSE THEN
       1E  7628        status.condition := mme$page_not_in_page_table;
       1E  7629        RETURN;
       2C  7630      IFEND;
       2C  7631
       2C  7632      pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
       2C  7633      status.normal := mmv$pt_p^ [pti].v;
       2C  7634      IF status.normal = FALSE THEN
       6O  7635        status.condition := mme$not_valid_in_page_table;
       6C  7636      IFEND;
       6C  7637
       6C  7638      pfte_p := ^mmv$pft_p^ [pfti];
       6C  7639
       6C  7640    PROCEND convert_sva_to_pfte_p;
```


LOCK_PAGES

```
        0  7643
        0  7644    PROCEDURE lock_pages
        0  7645      (     sva: ost$system_virtual_address;
        0  7646            length: ost$byte_count;
        0  7647       VAR status: syt$monitor_status);
        0  7648
        0  7649
        0  7650 {
        0  7651 {  This procedure processes the lock pages monitor function.
        0  7652 {
        0  7653
        0  7654      VAR
        0  7655        initial_lock_offset: ost$segment_offset,
        0  7656        lock_length: integer,
        0  7657        lock_sva: ost$system_virtual_address,
        0  7658        pfte_p: ^mmt$page_frame_table_entry,
        0  7659        unlock_status: syt$monitor_status;
        0  7660
        0  7661
        0  7662      status.normal := TRUE;
        4  7663      IF length = 0 THEN
        E  7664        RETURN;
       10  7665      IFEND;
       10  7666
       10  7667      IF ((length + sva.offset) > UPPERVALUE (ost$segment_offset)) THEN
       24  7668        mtp$set_status_abnormal ('MM', mme$lock_unlock_invalid_length, status);
       24  7669        RETURN;
       38  7670      IFEND;
       38  7671
       38  7672      lock_length := length + (sva.offset MOD osv$page_size);
       38  7673      lock_sva := sva;
       38  7674      lock_sva.offset := lock_sva.offset - (lock_sva.offset MOD osv$page_size);
       38  7675      initial_lock_offset := lock_sva.offset;
       38  7676
       38  7677
       38  7678 {  Lock page frame table entries until all specified entries locked or encounter a page frame
       38  7679 {  table entry that is already locked.  If this happens unlock all pages locked so far and
       38  7680 {  return error status.
       38  7681
       38  7682    /lock_pages_loop/
       38  7683      WHILE TRUE DO
       7A  7684        convert_sva_to_pfte_p (lock_sva, pfte_p, status);
       96  7685        IF status.normal = FALSE THEN
       AO  7686          unlock_pages (sva, lock_sva.offset - initial_lock_offset, unlock_status);
       BE  7687          RETURN;
       CO  7688        IFEND;
       CO  7689
       CO  7690        IF pfte_p^.locked_page <> mmc$lp_not_locked THEN
       CC  7691          unlock_pages (sva, lock_sva.offset - initial_lock_offset, status);
       E8  7692          mtp$set_status_abnormal ('MM', mme$page_already_locked, status);
       E8  7693          RETURN;
       FC  7694        IFEND;
       FC  7695
       FC  7696        pfte_p^.locked_page := mmc$lp_aging_lock;
       FC  7697        lock_length := lock_length - osv$page_size;
       FC  7698        IF lock_length <= 0 THEN
```

LOCK_PAGES

```
10E  7699          EXIT /lock_pages_loop/;
110  7700        IFEND;
110  7701
110  7702        lock_sva.offset := lock_sva.offset + osv$page_size;
110  7703      WHILEND /lock_pages_loop/;
128  7704
128  7705    PROCEND lock_pages;
```


UNLOCK_PAGES

```
 0  7708
 0  7709    PROCEDURE unlock_pages
 0  7710      (     sva: ost$system_virtual_address;
 0  7711            length: ost$byte_count;
 0  7712       VAR status: syt$monitor_status);
 0  7713
 0  7714
 0  7715 {
 0  7716 {  This procedure processes the unlock pages monitor function.
 0  7717 {
 0  7718
 0  7719      VAR
 0  7720        unlock_length: integer,
 0  7721        unlock_sva: ost$system_virtual_address,
 0  7722        pfte_p: ^mmt$page_frame_table_entry;
 0  7723
 0  7724
 0  7725      status.normal := TRUE;
 4  7726      IF length = 0 THEN
 E  7727        RETURN;
10  7728      IFEND;
10  7729
10  7730      IF ((sva.offset + length) > UPPERVALUE (ost$segment_offset)) THEN
24  7731        mtp$set_status_abnormal ('MM', mme$lock_unlock_invalid_length, status);
24  7732        RETURN;
38  7733      IFEND;
38  7734
38  7735      unlock_sva := sva;
38  7736      unlock_sva.offset := unlock_sva.offset - (unlock_sva.offset MOD osv$page_size);
38  7737      unlock_length := length + (sva.offset MOD osv$page_size);
38  7738
38  7739
38  7740 {  Unlock page frame table entries specified by pva and length.
38  7741
38  7742    /unlock_page_frames/
38  7743      WHILE TRUE DO
70  7744        convert_sva_to_pfte_p (unlock_sva, pfte_p, status);
88  7745        IF (status.normal = TRUE) AND (pfte_p^.locked_page = mmc$lp_aging_lock) THEN
9E  7746          pfte_p^.locked_page := mmc$lp_not_locked;
A4  7747        IFEND;
A4  7748
A4  7749        unlock_length := unlock_length - osv$page_size;
A4  7750        IF unlock_length <= 0 THEN
AE  7751          EXIT /unlock_page_frames/;
B2  7752        IFEND;
B2  7753
B2  7754        unlock_sva.offset := unlock_sva.offset + osv$page_size;
B2  7755      WHILEND /unlock_page_frames/;
C8  7756
C8  7757      status.normal := TRUE;
C8  7758
C8  7759    PROCEND unlock_pages;
```

MMP$MTR_CHANGE_SEGMENT_TABLE

```
        o  7762
        o  7763 {
        o  7764 {   The purpose of this procedure is to process the monitor request to
        o  7765 { change (move) the segment table.
        o  7766 {
        o  7767 {       MMP$MTR_CHANGE_SEGMENT_TABLE (REQUEST_BLOCK)
        o  7768 {
        o  7769 { REQUEST_BLOCK: (input, output) This parameter contains the change segment
        o  7770 {       table monitor request block.  The request status is returned to the
        o  7771 {       caller in this request block.  The new SDT and SDTX pointers are
        o  7772 {       passed to this procedure in this request block.
        o  7773 {
        o  7774
        o  7775   PROCEDURE [XDCL] mmp$mtr_change_segment_table
        o  7776     (VAR request_block: mmt$rb_change_segment_table;
        o  7777          cst_p: ^ost$cpu_state_table);
        o  7778
        o  7779     VAR
        o  7780       new_sdt_p: ^cell,
        o  7781       new_sdtx_p: ^cell,
        o  7782       old_sdt_p: mmt$max_sdt_p,
        o  7783       old_sdtx_p: mmt$max_sdtx_p,
        o  7784       sdt_rma: integer,
        o  7785       sdt_entries: integer,
        o  7786       xcb_p: ^ost$execution_control_block;
        o  7787
        o  7788     request_block.status.normal := TRUE;
        4  7789     xcb_p := cst_p^.xcb_p;
        4  7790
        4  7791 {  Convert SDT and SDTX pointer from pointers relative to job's address space to pointers
        4  7792 {  relative to monitor's address space.
        4  7793
        4  7794     new_sdt_p := #ADDRESS (1, #SEGMENT (xcb_p), request_block.new_sdt_offset);
        4  7795     mmp$get_max_sdt_sdtx_pointer (xcb_p, old_sdt_p, old_sdtx_p);
        4  7796     sdt_entries := xcb_p^.xp.segment_table_length +1;
        4  7797     i#move (old_sdt_p, new_sdt_p, sdt_entries * 8);
       AC  7798
       AC  7799 { The only time the new_sdtx_offset should equal zero is when this request
       AC  7800 { is called during job recovery.
       AC  7801
       AC  7802     IF request_block.new_sdtx_offset <> 0 THEN
       B4  7803       new_sdtx_p := #ADDRESS (1, #SEGMENT (xcb_p), request_block.new_sdtx_offset);
       B4  7804       i#move (old_sdtx_p, new_sdtx_p, sdt_entries * #SIZE (mmt$segment_descriptor_extended));
      10A  7805       xcb_p^.sdtx_offset := request_block.new_sdtx_offset;
      112  7806     IFEND;
      112  7807
      112  7808 {  Update the segment table address and length in the job's exchange package.
      112  7809
      112  7810     i#real_memory_address (new_sdt_p, sdt_rma);
      12C  7811     xcb_p^.xp.segment_table_address_1 := sdt_rma DIV 10000(16);
      12C  7812     xcb_p^.xp.segment_table_address_2 := sdt_rma MOD 10000(16);
      12C  7813     xcb_p^.sdt_offset := request_block.new_sdt_offset;
      12C  7814     xcb_p^.xp.segment_table_length := request_block.new_sdt_length;
      12C  7815
      12C  7816   PROCEND mmp$mtr_change_segment_table;
```


MMP$MTR_FETCH_OFFSET_MOD_PAGES

```
        o  7819
        o  7820   PROCEDURE [XDCL] mmp$mtr_fetch_offset_mod_pages
        o  7821     (VAR rb: mmt$rb_fetch_offset_mod_pages;
        o  7822          cst_p: ^ost$cpu_state_table);
        o  7823
        o  7824 {
        o  7825 {   The purpose of this request is to process the  monitor request
        o  7826 {  'syc$rc_fetch_offset_mod_pages'.  This monitor request returns to
        o  7827 {  job mode a list of segment offsets for all modified pages for a
        o  7828 {  given segment.
        o  7829 {   If the array is not large enough to hold all offsets for the modified
        o  7830 {  pages for the given segment, monitor will update the offsets_returned
        o  7831 {  field in the request block and the caller then should check and see
        o  7832 {  if he should re-allocate the array using the offsets_returned value.
        o  7833 {
        o  7834 {       MMP$MTR_FETCH_OFFSET_MOD_PAGES (RB)
        o  7835 {
        o  7836 { RB: (input, output): This parameter is a record which contains the
        o  7837 {       information specifying whats offsets to return.  The offsets
        o  7838 {       and control information are returned in this parameter.
        o  7839 {
        o  7840
        o  7841     TYPE
        o  7842       array_ptr_type = record
        o  7843         case b: 0 .. 1 of
        o  7844         = 0 =
        o  7845           array_p: ^array [1 .. * ] of ost$segment_offset,
        o  7846         = 1 =
        o  7847           array_pva: ost$pva,
        o  7848         casend,
        o  7849       recend;
        o  7850
        o  7851     VAR
        o  7852       asid: ost$asid,
        o  7853       aste_p: ^mmt$active_segment_table_entry,
        o  7854       asti: mmt$ast_index,
        o  7855       change_array_ptr: array_ptr_type,
        o  7856       converted_array_p: ^array [1 .. * ] of ost$segment_offset,
        o  7857       dfd_p: ^dmt$disk_file_descriptor,
        o  7858       offset_list_index: integer,
        o  7859       fde_p: gft$file_desc_entry_p,
        o  7860       p_fau: ^dmt$file_allocation_unit,
        o  7861       pfti_array_size: integer,
        o  7862       pft_index: mmt$page_frame_index,
        o  7863       sva: ost$system_virtual_address;
        o  7864
        o  7865 {  Initialize status.
        o  7866
        o  7867     rb.status.normal := TRUE;
        4  7868
        4  7869 {  Convert PVA to SVA and get AST pointer.
        4  7870
        4  7871     gfp$mtr_get_locked_fde_p (rb.sfid, cst_p^.ijle_p, fde_p);
       94  7872     mmp$get_verify_asti_in_fde (fde_p, rb.sfid, cst_p^.ijl_ordinal, asti);
       F2  7873     IF asti <> 0 THEN
       FA  7874       mmp$asid (asti, asid);
```

```
112  7875        sva.asid := asid;
112  7876        aste_p := ^mmv$ast_p^ [asti];
132  7877      ELSE
132  7878        rb.offsets_returned := 0;
132  7879        RETURN;
13A  7880      IFEND;
13A  7881
13A  7882    sva.offset := 0;
13A  7883 {  Locate all possibly MODIFIED pages for segment and get first pfti.
13A  7884
13A  7885      mmp$initialize_find_next_pfti (sva, 7ffffff0(16), include_partial_pages, psc_all_except_avail, aste_p,
16C  7886           pft_index);
16C  7887
16C  7888 {  Ensure that caller's array is large enough to hold all the offsets.
16C  7889
16C  7890      mmp$fetch_pfti_array_size (pfti_array_size);
16C  7891
16C  7892      IF pfti_array_size > rb.offsets_returned THEN
18A  7893        rb.offsets_returned := pfti_array_size;
18A  7894        RETURN;
190  7895      IFEND;
190  7896
190  7897      change_array_ptr.array_p := rb.offset_list;
19A  7898      change_array_ptr.array_pva.seg := #SEGMENT (^rb);
19A  7899      converted_array_p := change_array_ptr.array_p;
1B6  7900
1B6  7901 {  Move all offsets to array supplied by caller.
1B6  7902
1B6  7903      offset_list_index := 0;
1B6  7904
1B6  7905      IF fde_p^.media = gfc$fm_mass_storage_file THEN
1C4  7906        dmp$get_disk_file_descriptor_p (fde_p, dfd_p);
1E8  7907      IFEND;
1E8  7908
1E8  7909      WHILE (pft_index <> 0) AND (offset_list_index < rb.offsets_returned) DO
1F8  7910        IF mmv$pt_p^ [mmv$pft_p^ [pft_index].pti].m THEN
232  7911          offset_list_index := offset_list_index + 1;
232  7912          converted_array_p^ [offset_list_index] := mmv$pft_p^ [pft_index].sva.offset;
250  7913        ELSEIF (rb.return_unallocated_offsets) AND
264  7914          (fde_p^.media = gfc$fm_transient_segment) THEN
264  7915          offset_list_index := offset_list_index + 1;
264  7916          converted_array_p^ [offset_list_index] := mmv$pft_p^ [pft_index].sva.offset;
282  7917        ELSEIF (rb.return_unallocated_offsets) AND
296  7918          (fde_p^.media = gfc$fm_mass_storage_file) THEN
296  7919          dmp$get_fau_entry (dfd_p, mmv$pft_p^ [pft_index].sva.offset, p_fau);
352  7920          IF (p_fau = NIL) OR (p_fau^.state = dmc$fau_free) THEN
364  7921            offset_list_index := offset_list_index + 1;
364  7922            converted_array_p^ [offset_list_index] := mmv$pft_p^ [pft_index].sva.offset;
398  7923          IFEND;
398  7924        IFEND;
398  7925        mmp$find_next_pfti (pft_index);
3E4  7926      WHILEND;
3F8  7927
3F8  7928      rb.offsets_returned := offset_list_index;
3F8  7929
3F8  7930    PROCEND mmp$mtr_fetch_offset_mod_pages;
```



```
 0  7933
 0  7934    PROCEDURE [XDCL] mmp$mtr_fetch_pva_unwritten_pgs
 0  7935      (VAR rb: mmt$rb_fetch_pva_unwritten_pgs;
 0  7936           cst_p: ^ost$cpu_state_table);
 0  7937
 0  7938
 0  7939 {
 0  7940 {    The purpose of this request is to process the
 0  7941 { 'syc$rc_fetch_pva_unwritten_pgs' monitor function.  This monitor
 0  7942 { function returns to job mode the segment offsets of pages that could
 0  7943 { not be written to mass storage for a specified PVA.
 0  7944 {    The following assumptions are made when processing this request:
 0  7945 { . Any new pages added to the queue are added at the forward link.
 0  7946 { . The order of entries in the queue does not change.
 0  7947 {
 0  7948 {        MMP$MTR_FETCH_PVA_UNWRITTEN_PGS (RB)
 0  7949 {
 0  7950 { RB: (input, output): This parameter is a record which contains the
 0  7951 {        information specifying whats offsets to return.  The offsets
 0  7952 {        and control information are returned in this parameter.
 0  7953 {
 0  7954
 0  7955      VAR
 0  7956        offset_list_index: 0 .. 6,
 0  7957        pft_entry: mmt$page_frame_table_entry,
 0  7958        pft_index: mmt$page_frame_index,
 0  7959        sva: ost$system_virtual_address;
 0  7960
 0  7961
 0  7962      mmp$xtask_pva_to_sva (rb.pva, sva, rb.status);
2C  7963      IF rb.status.normal = FALSE THEN
34  7964        RETURN;
36  7965      IFEND;
36  7966
36  7967      pft_index := mmv$gpql [mmc$pq_wired].pqle.link.bkw;
36  7968      IF rb.subsequent_request_for_same_pva = TRUE THEN
48  7969
48  7970      /search_pft_for_offset/
48  7971        BEGIN
48  7972
48  7973        /find_starting_pft_entry/
48  7974          WHILE pft_index <> 0 DO
4C  7975            pft_entry := mmv$pft_p^ [pft_index];
74  7976            IF (sva.asid = pft_entry.sva.asid) AND (rb.next_offset_to_return = pft_entry.sva.offset) THEN
92  7977              EXIT /search_pft_for_offset/;
96  7978            IFEND;
96  7979
96  7980            pft_index := mmv$pft_p^ [pft_index].link.bkw;
96  7981          WHILEND /find_starting_pft_entry/;
B6  7982          mtp$set_status_abnormal ('MM', mme$no_matching_offset, rb.status);
B6  7983          RETURN;
CE  7984        END /search_pft_for_offset/;
CE  7985      IFEND;
CE  7986
CE  7987      rb.offset_list_overflow := FALSE;
CE  7988
```

MMP$MTR_FETCH_PVA_UNWRITTEN_PGS

```
      CE  7989      IF rb.starting_with_first_page = TRUE THEN
      DC  7990        sva.offset := 0;
      E4  7991      ELSE
      E4  7992        sva.offset := ((sva.offset DIV osv$page_size) + 1) * osv$page_size;
     100  7993      IFEND;
     100  7994
     100  7995      offset_list_index := 0;
     100  7996
     100  7997
     100  7998 {  Search wired queue of page frame table entries for entry with matching ASID and an
     100  7999 {  offset that is >=  starting offset.
     100  8000
     100  8001    /search_pft/
     100  8002      WHILE pft_index <> 0 DO
     108  8003        pft_entry := mmv$pft_p^ [pft_index];
     12C  8004        IF (sva.asid = pft_entry.sva.asid) AND (sva.offset <= pft_entry.sva.offset) THEN
     150  8005          IF offset_list_index >= 6 THEN
     156  8006            rb.offset_list_overflow := TRUE;
     156  8007            rb.next_offset_to_return := pft_entry.sva.offset;
     156  8008            rb.offsets_returned := 6;
     156  8009            RETURN;
     168  8010          ELSE
     168  8011            offset_list_index := offset_list_index + 1;
     168  8012            rb.offset_list [offset_list_index] := pft_entry.sva.offset;
     172  8013          IFEND;
     172  8014        IFEND;
     172  8015        pft_index := pft_entry.link.bkw;
     172  8016      WHILEND /search_pft/;
     17A  8017
     17A  8018      rb.offsets_returned := offset_list_index;
     17A  8019
     17A  8020    PROCEND mmp$mtr_fetch_pva_unwritten_pgs;
```


MMP$MTR_LOCK_UNLOCK_PAGES

```
       0  8023
       0  8024    PROCEDURE [XDCL] mmp$mtr_lock_unlock_pages
       0  8025      (VAR rb: mmt$rb_lock_unlock_pages;
       0  8026           cst_p: ^ost$cpu_state_table);
       0  8027
       0  8028
       0  8029 {
       0  8030 {   The purpose of this request is to process the 'syc$rc_lock_pages' and
       0  8031 { 'syc$rc_unlock_pages' monitor requests.  Locking pages prevents implicit
       0  8032 { IO being done on a page, unlocking a page removes this restriction.
       0  8033 {
       0  8034 {       MMP$MTR_LOCK_UNLOCK_PAGES (RB)
       0  8035 {
       0  8036 { RB: (input, output) This parameter is a record that specifies pages to
       0  8037 {       lock or unlock.  Request status is returned in this parameter.
       0  8038 {
       0  8039
       0  8040      VAR
       0  8041        sva: ost$system_virtual_address;
       0  8042
       0  8043
       0  8044      mmp$xtask_pva_to_sva (rb.pva, sva, rb.status);
      2C  8045      IF rb.status.normal = FALSE THEN
      34  8046        RETURN;
      36  8047      IFEND;
      36  8048
      36  8049      CASE rb.reqcode OF
      4E  8050      = syc$rc_lock_pages =
      4E  8051        CASE rb.lock_page_type OF
      58  8052        = mmc$lp_aging_lock =
      58  8053          lock_pages (sva, rb.length, rb.status);
      74  8054        ELSE
      74  8055          mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
      8A  8056        CASEND;
      8C  8057      = syc$rc_unlock_pages =
      8C  8058        CASE rb.lock_page_type OF
      96  8059        = mmc$lp_aging_lock =
      96  8060          unlock_pages (sva, rb.length, rb.status);
      B2  8061        ELSE
      B2  8062          mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
      C8  8063        CASEND;
      CA  8064      ELSE
      CA  8065        mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
      E0  8066      CASEND;
      E0  8067
      E0  8068    PROCEND mmp$mtr_lock_unlock_pages;
```

```
        0   8071
        0   8072    PROCEDURE [XDCL] mmp$mtr_set_get_segment_length
        0   8073      (VAR request_block: mmt$rb_set_get_segment_length;
        0   8074           cst_p: ^ost$cpu_state_table);
        0   8075
        0   8076  {
        0   8077  {   The purpose of this procedure is to process the monitor request to set
        0   8078  { or get current segment length.  If the segment is shortened, pages beyond
        0   8079  { new segment length are freed.
        0   8080  {
        0   8081  {         MMP$MTR_SET_GET_SEGMENT_LENGTH (REQUEST_BLOCK)
        0   8082  {
        0   8083  { REQUEST_BLOCK: (input,output) This parameter contains the monitor function
        0   8084  {         request block for the set or get segment length monitor function.
        0   8085  {
        0   8086
        0   8087      VAR
        0   8088        asid: ost$asid,
        0   8089        asti: mmt$ast_index,
        0   8090        aste_p: ^mmt$active_segment_table_entry,
        0   8091        fde_p: gft$locked_file_desc_entry_p,
        0   8092        ijl_ordinal: jmt$ijl_ordinal,
        0   8093        new_segment_length: integer,
        0   8094        old_eoi_state: mmt$eoi_state,
        0   8095        old_segment_length: integer,
        0   8096        page_count_freed: integer,
        0   8097        sfid: gft$system_file_identifier,
        0   8098        sva: ost$system_virtual_address;
        0   8099
        0   8100
        0   8101      fde_p := gfp$mtr_convert_job_mode_fde_p (request_block.fde_p, cst_p);
       3A   8102
       3A   8103      CASE request_block.subfunction_code OF
       4A   8104      = mmc$sf_get_segment_length_fde_p =
       4A   8105        IF fde_p^.stack_for_ring <> 0 THEN
       52   8106          IF cst_p^.xcb_p^.xp.p_register.pva.ring > fde_p^.stack_for_ring THEN
       68   8107            request_block.segment_length := 0;
       72   8108          ELSEIF cst_p^.xcb_p^.xp.p_register.pva.ring = fde_p^.stack_for_ring THEN
       76   8109            request_block.segment_length := #OFFSET (cst_p^.xcb_p^.xp.a0_dynamic_space_pointer);
       86   8110          ELSE
       86   8111            request_block.segment_length :=
       98   8112              cst_p^.xcb_p^.xp.tos_registers [fde_p^.stack_for_ring].pva.offset;
       98   8113          IFEND;
       9C   8114        ELSE
       9C   8115          IF fde_p^.eoi_state = mmc$eoi_uncertain THEN
*WARN*      8116            fixup_chapter_length (fde_p);
      29A   8117          IFEND;
      29A   8118          request_block.segment_length := fde_p^.eoi_byte_address;
      2A6   8119        IFEND;
      2A8   8120
      2A8   8121      = mmc$sf_set_segment_length_fde_p =
      2A8   8122        old_segment_length := fde_p^.eoi_byte_address;
      2A8   8123        old_eoi_state := fde_p^.eoi_state;
      2A8   8124        new_segment_length := request_block.segment_length;
      2A8   8125
      2A8   8126        fde_p^.eoi_byte_address := new_segment_length;
```



```
      2A8   8127        fde_p^.eoi_state := mmc$eoi_actual;
      2A8   8128        fde_p^.flags.eoi_modified := TRUE;
      2A8   8129
      2A8   8130        IF (old_eoi_state = mmc$eoi_uncertain) OR
      2DC   8131             ((old_segment_length DIV osv$page_size) > (new_segment_length DIV osv$page_size)) THEN
      2DC   8132          gfp$mtr_get_sfid_from_fde_p (fde_p, sfid, ijl_ordinal);
      33A   8133          mmp$get_verify_asti_in_fde (fde_p, sfid, ijl_ordinal, asti);
      39A   8134          IF asti <> 0 THEN
      3A2   8135            aste_p := ^mmv$ast_p^ [asti];
      3A2   8136            IF aste_p^.pages_in_memory > 0 THEN
      3B6   8137              mmp$asid (fde_p^.asti, asid);
      3D2   8138              sva.asid := asid;
      3D2   8139              sva.offset :=  new_segment_length;
      3D2   8140              mmp$mm_free_pages (sva, 7fffffff(16), aste_p, FALSE, page_count_freed);
      40A   8141            IFEND;
      40A   8142          IFEND;
      40A   8143        IFEND;
      40C   8144
      40C   8145      ELSE
      40C   8146        mtp$error_stop ('MM - Bad option on get_segment length');
      42C   8147      CASEND;
      42C   8148
      42C   8149    PROCEND mmp$mtr_set_get_segment_length;
        0   8150
```

fixup_chapter_length

```
o  8152
o  8153 { This procedure will find the unused pages assigned by mmp$page_pull when the
o  8154 { task page faulted for a "new page", release them, and set eoi to the end of the
o  8155 { last used page.
o  8156
o  8157   PROCEDURE [INLINE] fixup_chapter_length
o  8158     (     fde_p: gft$locked_file_desc_entry_p);
o  8159
o  8160     VAR
o  8161       asid: ost$asid,
o  8162       aste_p: ^mmt$active_segment_table_entry,
o  8163       asti: mmt$ast_index,
o  8164       count: 1 .. 32,
o  8165       eoi: ost$segment_length,
o  8166       found: boolean,
o  8167       i: 1 .. 10,
o  8168       ijl_ordinal: jmt$ijl_ordinal,
o  8169       max_eoi: ost$segment_length,
o  8170       offset: integer,
o  8171       page_count_freed: integer,
o  8172       pages_freed: integer,
o  8173       pfti_array: array [1 .. 10] OF mmt$page_frame_index,
o  8174       pti: integer,
o  8175       sfid: gft$system_file_identifier,
o  8176       sva: ost$system_virtual_address;
o  8177
o  8178
o  8179 { Check if the asid in the fde is still valid.  If not then there are no pages
o  8180 { of the file in memory and EOI will be assumed correct.
o  8181
o  8182     gfp$mtr_get_sfid_from_fde_p (fde_p, sfid, ijl_ordinal);
o  8183     mmp$get_verify_asti_in_fde (fde_p, sfid, ijl_ordinal, asti);
o  8184     IF asti = 0 THEN
o  8185       RETURN;
o  8186     IFEND;
o  8187     mmp$asid (asti, asid);
o  8188
o  8189 { Start searching for unused pages at the highest page assigned and work backwards,
o  8190 { stopping at the first modified page.  Eoi is currently set at the end of the page
o  8191 { that faulted.  The number 16384 is an arbitrary number that only must be less than
o  8192 { or equal to the minimum allocation unit size.  It was used to determine the number
o  8193 { of extra pages to assign.
o  8194
o  8195     eoi := fde_p^.eoi_byte_address;
o  8196     sva.asid := asid;
o  8197
o  8198     offset := eoi + 16384 - osv$page_size;
o  8199     max_eoi := offset;
o  8200     pages_freed := 0;
o  8201
o  8202   /find_eoi/
o  8203     WHILE offset > eoi DO
o  8204       offset := offset - osv$page_size;
o  8205       IF offset < osc$maximum_offset THEN
o  8206         sva.offset := offset;
o  8207         #HASH_SVA (sva, pti, count, found);
```


fixup_chapter_length

```
o  8208         IF found THEN
o  8209           IF mmv$pt_p^ [pti].m THEN
o  8210             offset := offset + osv$page_size;
o  8211             EXIT /find_eoi/;
o  8212           IFEND;
o  8213           pages_freed := pages_freed + 1;
o  8214           mmv$pt_p^ [pti].v := FALSE;
o  8215           pfti_array [pages_freed] := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
o  8216         IFEND;
o  8217       IFEND;
o  8218     WHILEND /find_eoi/;
o  8219
o  8220     fde_p^.eoi_byte_address := offset;
o  8221     fde_p^.eoi_state := mmc$eoi_rounded;
o  8222
o  8223     IF pages_freed > 0 THEN
o  8224       mmp$purge_all_cache_map;
o  8225       FOR i := 1 to pages_freed DO
o  8226         mmp$delete_pt_entry (pfti_array [i], TRUE);
o  8227         mmp$relink_page_frame (pfti_array [i], mmc$pq_free)
o  8228       FOREND;
o  8229     IFEND;
o  8230
o  8231   PROCEND fixup_chapter_length;
o  8232
```

```
  0  8234 {-------------------------------------------------------------------------------------------}
  0  8235 {This procedure processes the following requests:
  0  8236 {    mmp$lock_segment
  0  8237 {    mmp$unlock_segment
  0  8238 {-------------------------------------------------------------------------------------------}
  0  8239
  0  8240
  0  8241    PROCEDURE [XDCL] mmp$mtr_lock_unlock_segment
  0  8242      (VAR rb: mmt$rb_lock_unlock_segment;
  0  8243           cst_p: ^ost$cpu_state_table);
  0  8244
  0  8245      VAR
  0  8246        aste_p: ^mmt$active_segment_table_entry,
  0  8247        ste_p: ^mmt$segment_descriptor,
  0  8248        stxe_p: ^mmt$segment_descriptor_extended,
  0  8249        taskid: ost$global_task_id,
  0  8250        xcb_p: ^ost$execution_control_block,
  0  8251        page_status: gft$page_status,
  0  8252        qrb_p: ^mmt$rb_lock_unlock_segment,
  0  8253        fde_entry_p: gft$locked_file_desc_entry_p,
  0  8254        dequeue_tasks: boolean,
  0  8255        sdt_p: mmt$max_sdt_p,
  0  8256        sdtx_p: mmt$max_sdtx_p,
  0  8257        count: integer,
  0  8258        iotype: [READ, STATIC] array [mmc$lus_protected_write .. mmc$lus_write] of
  0  8259              iot$io_function := [ioc$write_locked_page, ioc$write_page],
  0  8260        sva: ost$system_virtual_address,
  0  8261        ijle_p: ^jmt$initiated_job_list_entry,
  0  8262        io_id: mmt$io_identifier,
  0  8263        io_count: mmt$active_io_count,
  0  8264        io_already_active: boolean,
  0  8265        last_written_pfti: mmt$page_frame_index,
  0  8266        wmp_status: mmt$write_modified_pages_status;
  0  8267
  0  8268
  0  8269      mmp$verify_pva (^rb.pva, mmc$sat_read_or_write, rb.status);
 2A  8270
 2A  8271      IF rb.status.normal THEN
 32  8272        mmp$convert_pva (rb.pva, cst_p, sva, fde_entry_p, aste_p, ste_p, stxe_p);
 76  8273        CASE rb.request OF
 88  8274        = mmc$lus_lock_segment =
 88  8275
 88  8276 { Determine the status/location of the page.
 88  8277
 88  8278          CASE fde_entry_p^.media OF
 A4  8279          = gfc$fm_transient_segment =
 A4  8280            page_status := gfc$ps_page_doesnt_exist;
 AE  8281          = gfc$fm_mass_storage_file =
 AE  8282            dmp$fetch_page_status (fde_entry_p, sva.offset, stxe_p^.file_limits_enforced,
 EC  8283                  FALSE {allocate_if_new}, page_status);
 EC  8284          = gfc$fm_served_file =
 EC  8285            dfp$fetch_page_status (fde_entry_p, sva.offset, page_status);
118  8286          ELSE
118  8287            mtp$error_stop ('MM - bad FDE.MEDIA');
13A  8288          CASEND;
13A  8289
```



```
13A  8290          IF page_status = gfc$ps_volume_unavailable THEN
144  8291            mtp$set_status_abnormal ('MM', mme$volume_unavailable, rb.status);
144  8292            RETURN;
15E  8293          ELSEIF page_status = gfc$ps_server_terminated THEN
164  8294            mtp$set_status_abnormal ('DF', dfe$server_has_terminated, rb.status);
164  8295            RETURN;
17A  8296          IFEND;
17A  8297          IF (stxe_p^.segment_lock <> mmc$lss_none) AND
192  8298            (stxe_p^.segment_lock <> mmc$lss_queued_for_lock_r3) AND
192  8299            (stxe_p^.segment_lock <> mmc$lss_queued_for_lock_user) THEN
192  8300            mtp$set_status_abnormal ('MM', mme$segment_locked_by_task, rb.status);
1AA  8301          ELSEIF (rb.access = mmc$lus_lock_for_read) AND NOT
1D4  8302            (fde_entry_p^.segment_lock.locked_for_write) AND (fde_entry_p^.segment_lock.task_queue.head = 0) AND
1D4  8303               (fde_entry_p^.segment_lock.locked_for_read <
1D4  8304                     UPPERVALUE (fde_entry_p^.segment_lock.locked_for_read)) THEN
1D4  8305            fde_entry_p^.segment_lock.locked_for_read := fde_entry_p^.segment_lock.locked_for_read + 1;
1D4  8306            IF cst_p^.xcb_p^.xp.p_register.pva.ring <= 3 THEN
1F0  8307              tmp$mtr_begin_lock_activity (cst_p^.xcb_p, osc$system_lock_activity);
208  8308              stxe_p^.segment_lock := mmc$lss_lock_for_read_r3;
216  8309            ELSE
216  8310              tmp$mtr_begin_lock_activity (cst_p^.xcb_p, osc$subsystem_lock_activity);
22A  8311              stxe_p^.segment_lock := mmc$lss_lock_for_read_user;
232  8312            IFEND;
236  8313          ELSEIF (rb.access = mmc$lus_lock_for_write) AND (fde_entry_p^.segment_lock.locked_for_read = 0) AND
254  8314                NOT fde_entry_p^.segment_lock.locked_for_write THEN
254  8315            fde_entry_p^.segment_lock.locked_for_write := TRUE;
254  8316            IF cst_p^.xcb_p^.xp.p_register.pva.ring <= 3 THEN
26E  8317              tmp$mtr_begin_lock_activity (cst_p^.xcb_p, osc$system_lock_activity);
286  8318              stxe_p^.segment_lock := mmc$lss_lock_for_write_r3;
294  8319            ELSE
294  8320              tmp$mtr_begin_lock_activity (cst_p^.xcb_p, osc$subsystem_lock_activity);
2A8  8321              stxe_p^.segment_lock := mmc$lss_lock_for_write_user;
2B2  8322            IFEND;
2B6  8323          ELSE
2B6  8324            mtp$set_status_abnormal ('MM', mme$segment_locked_another_task, rb.status);
2B6  8325            IF rb.wait = osc$wait THEN
2D2  8326              IF cst_p^.xcb_p^.xp.p_register.pva.ring <= 3 THEN
2E8  8327                stxe_p^.segment_lock := mmc$lss_queued_for_lock_r3;
2F4  8328              ELSE
2F4  8329                stxe_p^.segment_lock := mmc$lss_queued_for_lock_user;
2FC  8330              IFEND;
2FC  8331              tmp$queue_task (cst_p^.taskid, tmc$ts_segment_lock_wait, fde_entry_p^.segment_lock.task_queue);
322  8332            IFEND;
322  8333          IFEND;
324  8334
324  8335        = mmc$lus_unlock_segment =
324  8336          IF stxe_p^.segment_lock = mmc$lss_none THEN
330  8337            mtp$set_status_abnormal ('MM', mme$segment_not_locked, rb.status);
348  8338          ELSEIF fde_entry_p^.segment_lock.locked_for_write THEN
354  8339            sva.offset := 0;
354  8340            CASE rb.page_disposition OF
384  8341            = mmc$lus_write, mmc$lus_protected_write =
384  8342              io_id.specified := FALSE;
384  8343              io_id.io_function := iotype [rb.page_disposition];
384  8344              mmp$mm_write_modified_pages (sva, 7fffffff0(16), fde_entry_p, aste_p, iotype [rb.page_disposition],
3FC  8345                    rb.init_new_io, FALSE, io_id, io_count, io_already_active, last_written_pfti, wmp_status);
```

```
3FC  8346              mmp$process_wmp_status (wmp_status, last_written_pfti, rb.wait, rb.init_new_io, rb.status);
430  8347              IF ((wmp_status <> mmc$wmp_io_complete) AND (wmp_status <> mmc$wmp_io_active)) OR
44C  8348                 ((wmp_status = mmc$wmp_io_active) AND (rb.wait = osc$wait)) THEN
44C  8349                RETURN;
44E  8350              IFEND;
452  8351            = mmc$lus_remove_from_working_set =
452  8352              mmp$remove_pages_working_set (sva, 7ffffff0(16), aste_p, count);
482  8353            = mmc$lus_free =
482  8354              mmp$mm_free_pages (sva, 7ffffff0(16), aste_p, FALSE, count);
4B6  8355            = mmc$lus_none =
4BA  8356            ELSE
4BA  8357              mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
4BA  8358              RETURN;
4D0  8359            CASEND;
4D0  8360            IF stxe_p^.segment_lock = mmc$lss_lock_for_write_r3 THEN
4DE  8361              tmp$mtr_end_lock_activity (cst_p, osc$system_lock_activity, cst_p^.xcb_p);
50A  8362            ELSE
50A  8363              tmp$mtr_end_lock_activity (cst_p, osc$subsystem_lock_activity, cst_p^.xcb_p);
52E  8364            IFEND;
52E  8365            fde_entry_p^.segment_lock.locked_for_write := FALSE;
53A  8366          ELSE
53A  8367            IF stxe_p^.segment_lock = mmc$lss_lock_for_read_r3 THEN
540  8368              tmp$mtr_end_lock_activity (cst_p, osc$system_lock_activity, cst_p^.xcb_p);
568  8369            ELSE
568  8370              tmp$mtr_end_lock_activity (cst_p, osc$subsystem_lock_activity, cst_p^.xcb_p);
588  8371            IFEND;
588  8372            fde_entry_p^.segment_lock.locked_for_read := fde_entry_p^.segment_lock.locked_for_read - 1;
598  8373          IFEND;
598  8374          stxe_p^.segment_lock := mmc$lss_none;
598  8375
598  8376          dequeue_tasks := NOT fde_entry_p^.segment_lock.locked_for_write AND
5CC  8377            ((fde_entry_p^.segment_lock.locked_for_read = 0) OR
5CC  8378            (fde_entry_p^.segment_lock.locked_for_read =
5CC  8379              UPPERVALUE (fde_entry_p^.segment_lock.locked_for_read) - 1));
5CC  8380          WHILE dequeue_tasks AND (fde_entry_p^.segment_lock.task_queue.head <> 0) DO
5E0  8381            tmp$get_taskid_from_task_queue (fde_entry_p^.segment_lock.task_queue, taskid);
5E0  8382            tmp$get_xcb_p (taskid, xcb_p, ijle_p);
628  8383            IF xcb_p <> NIL THEN
632  8384              qrb_p := #LOC (xcb_p^.xp.x_registers [0]);
636  8385              mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
636  8386              stxe_p := ^sdtx_p^.sdtx_table [#SEGMENT (qrb_p^.pva)];
636  8387              IF qrb_p^.access = mmc$lus_lock_for_read THEN
68C  8388                fde_entry_p^.segment_lock.locked_for_read := fde_entry_p^.segment_lock.locked_for_read + 1;
68C  8389                IF stxe_p^.segment_lock = mmc$lss_queued_for_lock_r3 THEN
6A4  8390                  tmp$mtr_begin_lock_activity (xcb_p, osc$system_lock_activity);
6BC  8391                  stxe_p^.segment_lock := mmc$lss_lock_for_read_r3;
6CA  8392                ELSE
6CA  8393                  tmp$mtr_begin_lock_activity (xcb_p, osc$subsystem_lock_activity);
6DE  8394                  stxe_p^.segment_lock := mmc$lss_lock_for_read_user;
6E8  8395                IFEND;
6E8  8396                IF fde_entry_p^.segment_lock.locked_for_read =
6F8  8397                   UPPERVALUE (fde_entry_p^.segment_lock.locked_for_read) THEN
6F8  8398                  dequeue_tasks := FALSE;
6FC  8399                IFEND;
700  8400              ELSEIF fde_entry_p^.segment_lock.locked_for_read = 0 THEN
70C  8401                fde_entry_p^.segment_lock.locked_for_write := TRUE;
```



```
70C  8402              IF stxe_p^.segment_lock = mmc$lss_queued_for_lock_r3 THEN
71A  8403                tmp$mtr_begin_lock_activity (xcb_p, osc$system_lock_activity);
732  8404                stxe_p^.segment_lock := mmc$lss_lock_for_write_r3;
740  8405              ELSE
740  8406                tmp$mtr_begin_lock_activity (xcb_p, osc$subsystem_lock_activity);
754  8407                stxe_p^.segment_lock := mmc$lss_lock_for_write_user;
75E  8408              IFEND;
75E  8409              dequeue_tasks := FALSE;
766  8410            ELSE
766  8411              jmp$unlock_ajl (ijle_p);
81A  8412              RETURN;
81C  8413            IFEND;
81C  8414            qrb_p^.status.normal := TRUE;
81C  8415            jmp$unlock_ajl (ijle_p);
8D6  8416          IFEND;
8D6  8417          tmp$dequeue_task (fde_entry_p^.segment_lock.task_queue, taskid);
8F6  8418        WHILEND;
908  8419      ELSE
908  8420        mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
91A  8421    CASEND;
91A  8422  IFEND;
91A  8423
91A  8424  PROCEND mmp$mtr_lock_unlock_segment;
```

mmp$mtr_wait_io_completion

```
        0  8426 {
        0  8427 {  The purpose of this request is to suspend execution of the requesting
        0  8428 { task if IO is active for the specified page. If IO is not active
        0  8429 { or if the page is not in memory, this request completes immediately and a
        0  8430 { normal status is returned. If IO is active, execution of the task
        0  8431 { is suspended until all IO to the page is completed.
        0  8432 {
        0  8433 {    MMP$WAIT_IO_COMPLETION (P, STATUS)
        0  8434 {
        0  8435 {  P: (INPUT) This parameter specifies the address of the page.
        0  8436 {
        0  8437 {  STATUS: (OUTPUT) This parameter specifies request status.
        0  8438 {
        0  8439
        0  8440
        0  8441   PROCEDURE [XDCL] mmp$mtr_wait_io_completion
        0  8442     (VAR rb: mmt$rb_wait_io_completion;
        0  8443          cst_p: ^ost$cpu_state_table);
        0  8444
        0  8445     VAR
        0  8446       count: 1 .. 32,
        0  8447       found: boolean,
        0  8448       pti: integer,
        0  8449       pfti: mmt$page_frame_index,
        0  8450       sva: ost$system_virtual_address;
        0  8451
        0  8452     mmp$verify_pva (^rb.pva, mmc$sat_read_or_write, rb.status);
       2A  8453     IF rb.status.normal THEN
       32  8454       mmp$xtask_pva_to_sva (rb.pva, sva, rb.status);
       52  8455       IF rb.status.normal THEN
       5A  8456         #HASH_SVA (sva, pti, count, found);
       60  8457         IF found THEN
       74  8458           pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
       74  8459           IF mmv$pft_p^ [pfti].active_io_count <> 0 THEN
       BA  8460             cst_p^.xcb_p^.page_wait_info.pva := NIL;
       BA  8461             tmp$queue_task (cst_p^.taskid, tmc$ts_io_wait_queued, mmv$pft_p^ [pfti].task_queue);
      106  8462           IFEND;
      106  8463         IFEND;
      106  8464       IFEND;
      106  8465     IFEND;
      106  8466   PROCEND mmp$mtr_wait_io_completion;
```


MMP$MODIFY_PAGES

```
        0  8469
        0  8470   PROCEDURE [XDCL] mmp$modify_pages
        0  8471     (     fde_p: gft$locked_file_desc_entry_p;
        0  8472           offset: ost$segment_offset;
        0  8473           length: ost$byte_count;
        0  8474           set_modified_bit: boolean;
        0  8475       VAR status: syt$monitor_status);
        0  8476
        0  8477
        0  8478 {  This procedure verifies that all pages of a given sva range
        0  8479 {  are in memory and optionally sets the modified bits.
        0  8480
        0  8481 {  This request is used only by dmp$reallocate_file_space to verify
        0  8482 {  that all the pages of an allocation unit being reallocated are in memory
        0  8483 {  and to cause them to be modified and therefore written to the new allocation
        0  8484 {  unit.
        0  8485
        0  8486     VAR
        0  8487       asid: ost$asid,
        0  8488       asti: mmt$ast_index,
        0  8489       ijlo: jmt$ijl_ordinal,
        0  8490       lock_length: integer,
        0  8491       sfid: gft$system_file_identifier,
        0  8492       sva: ost$system_virtual_address,
        0  8493       pfte_p: ^mmt$page_frame_table_entry,
        0  8494       pfti: mmt$page_frame_index;
        0  8495
        0  8496
        0  8497     status.normal := TRUE;
        4  8498
        4  8499     gfp$mtr_get_sfid_from_fde_p (fde_p, sfid, ijlo);
       70  8500     mmp$get_verify_asti_in_fde (fde_p, sfid, ijlo, asti);
       D6  8501     IF asti = 0 THEN
       DE  8502       mtp$set_status_abnormal ('MM', mme$page_not_in_page_table, status);
       DE  8503       RETURN;
       EE  8504     IFEND;
       EE  8505     mmp$asid (asti, asid);
      10A  8506
      10A  8507     lock_length := length + offset;
      10A  8508     sva.asid := asid;
      10A  8509     sva.offset := offset;
      10A  8510
      10A  8511   /modify_pages_loop/
      10A  8512     WHILE TRUE DO
      126  8513       convert_sva_to_pfte_p (sva, pfte_p, status);
      13A  8514       IF NOT status.normal THEN
      142  8515         mtp$set_status_abnormal ('MM', mme$page_not_in_page_table, status);
      142  8516         RETURN;
      150  8517       IFEND;
      150  8518
      150  8519       IF NOT mmv$pt_p^ [pfte_p^.pti].v THEN
      170  8520         {Return if page is not modify-able
      170  8521         mtp$set_status_abnormal ('MM', mme$page_not_in_page_table, status);
      170  8522         RETURN;
      17E  8523       IFEND;
      17E  8524
```

MMP$MODIFY_PAGES

```
     17E  8525         IF set_modified_bit THEN
     186  8526           mmv$pt_p^ [pfte_p^.pti].m := TRUE;
     186  8527           {Allow retry of write operations
     186  8528           pfte_p^.io_error := ioc$no_error;
     186  8529           pfti := (mmv$pt_p^ [pfte_p^.pti].rma * 512) DIV osv$page_size;
     186  8530           mmp$relink_page_frame (pfti, pfte_p^.aste_p^.queue_id);
     1DA  8531         ELSEIF pfte_p^.active_io_count > 0 THEN
     1E6  8532           {Return if page is not idle
     1E6  8533           mtp$set_status_abnormal ('MM', mme$page_not_in_page_table, status);
     1E6  8534           RETURN;
     1F4  8535         IFEND;
     1F4  8536
     1F4  8537         lock_length := lock_length - osv$page_size;
     1F4  8538         IF lock_length <= 0 THEN
     202  8539           EXIT /modify_pages_loop/;
     204  8540         IFEND;
     204  8541
     204  8542         sva.offset := sva.offset + osv$page_size;
     204  8543       WHILEND /modify_pages_loop/;
     218  8544
     218  8545     PROCEND mmp$modify_pages;
       0  8546   MODEND mmm$mtr_user_request_processor;
```

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

MMP$MODIFY_PAGES

```
          ERROR     LINE  TEXT
WARNING   CY 821    8116   Code scheduling abandoned for this block due to register jamming.

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
a0_dynamic_space_pointer	4628	8109							
access	2384	8301	8313	8387					
active_io_count	4095	8459	8531						
ajl_ordinal	3415	6841	6876	7110	7871	8411	8415		
ajlo	7107	7110/M	7111/S	7114/S	7114/S				
ajlo	8241	8411/M	8411/S	8411/S	8411/S	8415/M	8415/S	8415/S	8415/S
amc$access_mode	1131	608							
amc$average_record_length	1133	683							
amc$block_type	1134	610							
amc$character_conversion	1135	612							
amc$clear_space	1136	614							
amc$collate_table_name	1138	685							
amc$compression_procedure_name	1191	687							
amc$data_padding	1139	690							
amc$dynamic_home_block_space	1192	692							
amc$embedded_key	1140	694							
amc$error_exit_name	1141	616							
amc$error_limit	1143	696							
amc$error_options	1144	618							
amc$estimated_record_count	1145	698							
amc$file_access_procedure	1146	620							
amc$file_byte_limit	430	433	435	892	930	966	1245	1307	
amc$file_contents	1147	622							
amc$file_limit	1149	624							
amc$file_organization	1150	626							
amc$file_processor	1151	628							
amc$file_structure	1152	630							
amc$forced_write	1153	632							
amc$hashing_procedure_name	1193	700							
amc$index_levels	1159	702							
amc$index_padding	1160	704							
amc$initial_home_block_count	1194	706							
amc$internal_code	1161	634							
amc$key_length	1162	708							
amc$key_position	1163	710							
amc$key_type	1164	712							
amc$label_exit_name	1165	636							
amc$label_options	1167	638							
amc$label_type	1168	640							
amc$line_number	1169	642							
amc$loading_factor	1195	714							
amc$lock_expiration_time	1196	716							
amc$log_residence	1198	720							
amc$logging_options	1197	718							
amc$max_attribute	1239	1243							
amc$max_block_length	1170	644							
amc$max_block_number	752	755							
amc$max_error_count	1123	1126							
amc$max_file_id_ordinal	1060	1067							
amc$max_home_blocks	930	933							
amc$max_index_level	925	928							
amc$max_key_length	1264	1268							

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
amc$max_key_position	1273	1270							
amc$max_line_number	943	946							
amc$max_lines_per_inch	1331	1328							
amc$max_page_width	895	898							
amc$max_path_name_size	909	912							
amc$max_record_length	1171	646							
amc$max_records_per_block	1315	1319							
amc$max_statement_id_length	990	993							
amc$max_user_info	1326	1322							
amc$maximum_block	760	757	1298						
amc$maximum_keyed_record	1276	1273							
amc$maximum_record	966	969	1041	1302					
amc$message_control	1172	722							
amc$min_block_length	1173	648							
amc$min_record_length	1174	650							
amc$null_attribute	1175	652							
amc$open_position	1176	654							
amc$padding_character	1177	656							
amc$page_format	1178	658							
amc$page_length	1179	660							
amc$page_width	1180	662							
amc$preset_value	1182	664							
amc$record_limit	1183	724							
amc$record_type	1184	666							
amc$records_per_block	1185	726							
amc$return_option	1186	668							
amc$ring_attributes	1187	670							
amc$statement_identifier	1188	672							
amc$user_info	1189	674							
amc$vertical_print_density	1190	676							
amt$access_selection	597	595							
amt$average_record_length	1041	684							
amt$block_header_type	733	736	742						
amt$block_number	755	738	745						
amt$block_status	734	747							
amt$block_type	1044	611							
amt$collation_value	1049	1046							
amt$compression_procedure_name	900	689							
amt$data_padding	1052	691							
amt$dynamic_home_block_space	919	693							
amt$entry_point_reference	903	900	921						
amt$error_limit	1119	697							
amt$estimated_record_count	1128	699							
amt$file_attribute_keys	1243	602							
amt$file_byte_address	433	400	401	448	452	464	466	482	1738
		1786	1787	1813	1837	2004	5981	6761	6776
amt$file_contents	778	623							
amt$file_id_ordinal	1067	1064							
amt$file_id_sequence	1068	1065							
amt$file_identifier	1063	1055	1282						
amt$file_item	601	597	600						
amt$file_limit	435	470	625	1660	2008				

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
amt$file_organization	1249	627							
amt$file_position	1252	588							
amt$file_processor	839	629							
amt$file_structure	885	631							
amt$forced_write	1255	633							
amt$hashing_procedure_name	921	701							
amt$index_levels	928	703							
amt$index_padding	1259	705							
amt$initial_home_block_count	933	707							
amt$internal_code	1261	635							
amt$key_length	1268	709							
amt$key_position	1270	711							
amt$key_type	1279	713							
amt$label_options	590	639							
amt$label_type	1286	641							
amt$line_number	937	643							
amt$line_number_length	946	938							
amt$line_number_location	948	939							
amt$loading_factor	951	715							
amt$lock_expiration_time	953	717							
amt$log_residence	955	721							
amt$logging_options	958	719							
amt$logging_possibilities	961	958							
amt$max_block_length	757	645	737	743	744				
amt$max_record_length	969	647							
amt$message_control	1296	723							
amt$min_block_length	1298	649							
amt$min_record_length	1302	651							
amt$open_position	973	655							
amt$padding_character	1305	657							
amt$page_format	888	659							
amt$page_length	892	661							
amt$page_width	898	663	948	995					
amt$path_name	912	905	955						
amt$preset_value	439	490	665	1658					
amt$record_limit	1307	725							
amt$record_type	1311	667							
amt$records_per_block	1319	727							
amt$return_option	591	669							
amt$ring_attributes	977	671							
amt$statement_id_length	993	985							
amt$statement_id_location	995	986							
amt$statement_identifier	984	673							
amt$tape_error_action	1005	1000							
amt$tape_error_options	998	619							
amt$unused_bit_count	765	739	746						
amt$user_info	1322	675							
amt$vertical_print_density	1328	677							
array_p	7845	7897/M	7899						
array_ptr_type	7842	7855							
array_pva	7847	7898/M							
asid	1705	7875/M	7976	7976	8004	8004	8116/M	8138/M	8196/M

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8508/M							
asid	7852	7874/P	7875						
asid	8072	8116/P	8116						
asid	8088	8137/P	8138						
asid	8161	8187/P	8196						
asid	8487	8505/P	8508						
aste_p	4100	8530/P							
aste_p	7853	7876/M	7885/P						
aste_p	8090	8135/M	8136	8140/P					
aste_p	8246	8272/P	8344/P	8352/P	8354/P				
asti	2003	7283	7287/M	7872	7872/M	8116	8116/M	8133	8133/M
		8137/P	8183	8183/M	8500	8500/M			
asti	7280	7283/M	7284/S	7284/S	7285/S	7286/M			
asti	7820	7872/M	7872/S	7872/S	7872/S	7872/M			
asti	7854	7872/P	7873	7874/P	7876/S				
asti	8072	8116/P	8116	8116/P					
asti	8072	8116/M	8116/S	8116/S	8116/S	8116/M	8133/M	8133/S	8133/S
		8133/S	8133/M						
asti	8089	8133/P	8134	8135/S					
asti	8157	8183/M	8183/S	8183/S	8183/S	8183/M			
asti	8163	8183/P	8184	8187/P					
asti	8470	8500/M	8500/S	8500/S	8500/S	8500/M			
asti	8488	8500/P	8501	8505/P					
b	6869	6878	6878						
b	6965	6975	6976						
b	7103	7109	7109						
b	7173	7180	7181						
b	7820	7871	7871						
b	8241	8411	8411	8415	8415				
bc	7103	7109/M	7109	7109					
bc	7174	7177/M	7178	7182					
bc	8241	8411/M	8411	8411	8415/M	8415	8415		
bkw	2244	7967	7980	8015					
byte	2126	7054/M	7076/M						
byte_address	6776	6794	6799						
byte_address	7820	7919	7919						
bytes_per_allocation	396	6799	7919						
bytes_per_level_2	401	6794	6799	7919	7919				
change_array_ptr	7855	7897/M	7898/M	7899					
clear	7165	7116/M	7143/M	8411/M	8415/M				
cmc$central_memory_element	6518	6394	6624						
cmc$central_processor_element	6519	6383							
cmc$channel_adapter_element	6520	6387	6629						
cmc$choose_any_pp	6561	6532							
cmc$choose_pp_by_barrel	6562	6534							
cmc$choose_pp_by_channel	6561	6536							
cmc$choose_specific_pp	6562	6539							
cmc$communications_element	6523	6387	6629						
cmc$controller_element	6519	6388	6630						
cmc$data_channel_element	6520	6385	6626						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
cmc$external_processor_element	6523	6388						
cmc$iou_element	6521	6394						
cmc$mainframe_element	6521	6394						
cmc$max_equipment_per_channel	5372	5375						
cmc$max_esm_size	5549	6580	6585					
cmc$max_low_speed_port_number	6588	6572						
cmc$max_side_door_port_number	6589	6344	6365	6366	6573			
cmc$max_units_per_controller	6510	6513						
cmc$pem_element	6521	6395						
cmc$pp_element	6522	6391						
cmc$storage_device_element	6522	6389						
cmt$central_memory_port_number	6640	6625						
cmt$channel_descriptor	6401	6386						
cmt$channel_identification	6491	6484	6538					
cmt$channel_ordinal	6415	6407	6492					
cmt$element_name	5369	5346	6330	6331	6341	6355	6371	6372	6384
		6402	6405	6474	6483	6493	6547	6568	6621
		6627	6628						
cmt$element_reservation	6381	6349	6649						
cmt$element_state	3360	3319	3320	3350					
cmt$element_type	6518	6382	6623						
cmt$esm_maintenance_buffer_loc	6579	6347	6361	6575					
cmt$esm_memory_size	6585	6574							
cmt$hardware_address	6481	6476							
cmt$model_number	6601	6596							
cmt$peripheral_descriptor	6471	6390							
cmt$physical_address_parts	6505	6502							
cmt$physical_address_specifier	6502	6482							
cmt$physical_equipment_number	5375	5356	6485						
cmt$physical_unit_number	6513	6486							
cmt$pp_identification	6545	6530	6540						
cmt$pp_ordinal	6551	6546							
cmt$pp_reservation	6526	6392							
cmt$pp_reservation_choices	6561	6531							
cmt$product_identification	6593	6569							
cmt$product_number	6599	6594							
cmt$serial_number	6604	6570							
cmt$upline_connection	6607	6572	6573						
condition	2363	7426/M	7628/M	7635/M	7668/M	7692/M	7731/M	7982/M	8055/M
		8062/M	8065/M	8291/M	8294/M	8300/M	8324/M	8337/M	8357/M
		8420/M	8502/M	8515/M	8521/M	8533/M			
condition	7421	7426							
condition	7644	7668	7692						
condition	7709	7731							
condition	7934	7982							
condition	8024	8055	8062	8065					
condition	8241	8291	8294	8300	8324	8337	8357	8420	
condition	8470	8502	8515	8521	8533				
convert_sva_to_pfte_p	7608	7640	7684	7744	8513				
converted_array_p	7856	7899/M	7912/M	7916/M	7922/M				
count	7162	7109/M	7109	7116	7116/M	7116	7140	7141/M	7141
		7184/M	7184	8411/M	8411	8411	8411/M	8411	8415/M

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8415	8415	8415/M	8415				
count	7619	7625							
count	8072	8116							
count	8164	8207							
count	8257	8352/P	8354/P						
count	8446	8456							
cst_p	6809	6816							
cst_p	7408	7410/M							
cst_p	7777	7789							
cst_p	7822	7871/P	7872/P						
cst_p	8072	8101							
cst_p	8074	8101/P	8106	8108	8109	8112			
cst_p	8243	8272/P	8306	8307/P	8310/P	8316	8317/P	8320/P	8326
		8331/P	8361/P	8361/P	8363/P	8363/P	8368/P	8368/P	8370/P
		8370/P							
cst_p	8443	8460/M	8461/P						
dequeue_tasks	8254	8376/M	8380	8380	8398/M	8409/M			
dest	7083	7083							
dest	7775	7797	7804						
dfc$active	5804	5773							
dfc$awaiting_recovery	5805	5780							
dfc$command_record_bytes	3515	3523	5718						
dfc$deactivated	5804	5789							
dfc$division_overwrite_words	3502	3530							
dfc$esm_command_record_size	3523	3531							
dfc$esm_connection	5748	5344	5745						
dfc$esm_header_record_size	3524	3531							
dfc$esm_maintenance_buf_size	3503	3534							
dfc$esm_memory_base_shift	3509	3531	3532	3532					
dfc$header_record_bytes	3514	3524							
dfc$inactive	5804	5780							
dfc$max_data_record_bytes	3518	6377							
dfc$max_esm_divisions	3512	5383							
dfc$max_esm_memory_size	3504	3533	6342	6356					
dfc$max_number_of_mainframes	3511	3496	5382						
dfc$max_number_of_queues	5814	5430	5432	5434	5819				
dfc$max_queue_entries	5815	5562	5563	5604	5605	5820			
dfc$max_req_timeout_count_value	5699	5577	5619						
dfc$max_request_buffer_entries	5378	6287							
dfc$max_retransmit_count_value	5703	5578	5620						
dfc$maximum_lifetime	5879	5876							
dfc$maximum_queue_interfaces	6653	6656							
dfc$maximum_user_buffer_area	5826	5836	5837						
dfc$maximum_user_data_area	5830	5839	5840						
dfc$min_cdcnet_errors	317	323	325	328	331	334	337	340	343
		346	349	352	372				
dfc$min_data_record_bytes	3519	3530	6377						
dfc$min_driver_test_errors	312	360	363	366	369				
dfc$min_ecc	22	28	31	34	37	40	44	48	51
		54	57	60	63	66	69	72	75
		78	81	85	88	91	95	98	101

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		104	107	110	113	116	119	122	125
		128	131	134	138	142	145	148	152
		155	158	161	164	167	170	173	176
		180	183	187	190	193	196	199	202
		205	209	213	217	220	223	226	230
		234	237	240	243	246	249	253	256
		260	263	266	269	272	275	279	282
		285	289	293	298	301	305	312	314
		317	319						
dfc$min_esm_division_size	3529	3533							
dfc$min_esm_memory_size	3505	6342	6356						
dfc$min_mm_recovery_errors	319	375	379	382					
dfc$mock_connection	5749	5745							
dfc$monitor	5809	5628							
dfc$monitor_allocate	5589	5579							
dfc$monitor_io	5589	5579							
dfc$queue_assignment_strng_size	5604	5565							
dfc$recovering	5805	5794							
dfc$task_services	5809	5638							
dfc$terminated	5805	5780							
dfc$unrecovered_disk_error	6032	6060							
dfd_p	6719	6733/M							
dfd_p	7820	7906/M							
dfd_p	7857	7906/P	7919/P						
dfe$server_has_terminated	170	8294/P							
dfp$fetch_page_status	6682	8285							
dft$allocated_command_buffer	5713	5712							
dft$allocated_data_rma_list	5674	5673							
dft$allocated_monitor_buffer	5737	5736							
dft$channel_definition	6370	6344	6366						
dft$channel_specification	6329	5347	5348						
dft$connection_address	5479	5474	5475						
dft$connection_descriptor	5473	5460							
dft$connection_flags	5487	5480							
dft$connection_type	5748	5343	5566						
dft$cpu_queue	5553	5447							
dft$cpu_queue_entries	5558	5555							
dft$cpu_queue_entry	5609	5558							
dft$cpu_queue_header	5561	5554							
dft$cpu_queue_pva_entries	5434	5420							
dft$cpu_queue_pva_entry	5446	5435							
dft$data_descriptor	5534	5501	5502	5503					
dft$dma_adapter	5423	5410							
dft$driver_queue	5451	5443							
dft$driver_queue_entries	5492	5453							
dft$driver_queue_entry	5494	5492							
dft$driver_queue_header	5456	5452							
dft$driver_queue_header_flags	5463	5457							
dft$driver_queue_pva_entries	5432	5419							
dft$driver_queue_pva_entry	5442	5433							
dft$driver_queue_rma_entries	5430	5418							
dft$driver_queue_rma_entry	5437	5431							

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dft$esm_base_addresses	5399	5392	6345	6359				
dft$esm_definition_table_entry	6340	6337	6350					
dft$esm_pp_information	5354	5349	5350					
dft$inquiry_message	6241	6232	6295					
dft$inquiry_tracer	6246	6242						
dft$interrupt	5468	5458						
dft$lifetime	5876	5872						
dft$mainframe_set	3496	3446	3447	3559	3560			
dft$maximum_data_bytes	6377	6346	6360					
dft$monitor_io_types	5589	5633						
dft$p_allocated_data_rma_list	5673	5585						
dft$p_command_buffer	5711	5622	5623					
dft$p_data_rma_list	5660	5625						
dft$p_queue_interface_table	5387	5340						
dft$p_send_data	5840	5640	5641					
dft$partner_status	5763	5570						
dft$pp_element_reservations	6649	5358						
dft$pp_status	5361	5355						
dft$q_interface_directory_entry	5338	5336						
dft$queue_directory	5409	5394						
dft$queue_directory_index	6656	5324						
dft$queue_entry_flags	5506	5495	5617					
dft$queue_entry_index	5820	5326						
dft$queue_entry_location	5323	5313						
dft$queue_entry_type	5809	5627						
dft$queue_index	5819	5325						
dft$queue_interface_directory	5335	5333						
dft$queue_interface_table	5389	5387						
dft$request_buffer	6282	6278						
dft$request_buffer_directory	6271	5391						
dft$request_buffer_entries	6287	6284						
dft$request_buffer_entry	6292	6288						
dft$request_buffer_entry_flags	6300	6293						
dft$response_flags	6219	6211						
dft$response_parameter	6229	6213						
dft$retransmission_digit	6252	6248						
dft$rpc_progress_record	5851	5643						
dft$send_data_size	5839	5642	5648	5650	5651	5854	5855	5860
dft$send_parameter_size	5836	5859						
dft$server_iocb_error_condition	5997	5983						
dft$server_lifetime	5872	5571						
dft$server_state	5804	5772	5807					
dft$side_door_ports	6364	6358						
dft$transaction_data	5592	5584						
dft$transaction_digit	6251	6247						
dft$transaction_state	5918	5618	5925	6243				
disk_file_descriptor_p	2018	6732	7906					
dmc$a2	1348	1352						
dmc$allocated_length	509	447	1836					
dmc$asid	509	449						
dmc$byte_address	510	451						
dmc$bytes_per_allocation	510	453						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dmc$bytes_per_level_2	1769	1777						
dmc$chapter_length	517	501						
dmc$class	510	455	1538					
dmc$class_ordinal	510	457	1540					
dmc$clear_space	511	459						
dmc$default_number_fau_entries	1981	1974						
dmc$device_file_list_index	511	461	1838					
dmc$eof_byte_address	511	463						
dmc$eoi_byte_address	512	465						
dmc$fau_free	1976	7920						
dmc$file_hash	512	467						
dmc$file_kind	513	473	1544					
dmc$file_limit	512	469						
dmc$file_status	512	471						
dmc$global_file_name	513	475						
dmc$internal_vsn	513	477	1840					
dmc$level_1_table_size	1765	1769	1772					
dmc$locked_file	513	479						
dmc$logical_length	514	481						
dmc$master_volume_required	514	483	1546					
dmc$max_bytes_per_allocation	1344	396	1337	1339	1340			
dmc$max_bytes_per_dau	1878	1870						
dmc$max_bytes_per_mau	1914	1900						
dmc$max_class_ordinal	1369	1366						
dmc$max_dau_address	1880	1874	1922					
dmc$max_daus_allocation	1882	1871						
dmc$max_daus_position	1884	1872	1924					
dmc$max_daus_transfer	1886	1873						
dmc$max_device_file_list_index	1387	1384						
dmc$max_fau_entries	1982	1975						
dmc$max_file_hash	1397	422	424	1394				
dmc$max_mau_address	1922	1910						
dmc$max_maus_per_allocation	1916	1902	1903					
dmc$max_maus_per_dau	1918	1904	1905	1922	1924			
dmc$max_maus_per_transfer	1920	1907	1908					
dmc$max_maus_position	1924	1909						
dmc$max_transfer_size	1686	1681						
dmc$min_bytes_per_allocation	1343	1338						
dmc$min_bytes_per_dau	1877	1870						
dmc$min_dau_address	1879	1874						
dmc$min_mau_address	1921	1910						
dmc$min_maus_per_allocation	1915	1901						
dmc$min_maus_per_dau	1917	1904						
dmc$min_maus_per_transfer	1919	1906						
dmc$overflow	514	485						
dmc$owner	515	487						
dmc$preset_value	515	489						
dmc$queue_status	517	505						
dmc$recorded_vsn	515	491	1542	1842				
dmc$requested_allocation_size	515	493	1548					
dmc$requested_transfer_size	516	495						
dmc$requested_volume	516	497						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dmc$setname	516	499	1550					
dmc$write_mode	517	503						
dmp$fetch_page_status	6708	8282						
dmp$get_disk_file_descriptor_p	6718	7906						
dmp$get_fau_entry	6775	6805	7919					
dmp$get_level_2_ptr	6761	6771	6797	7919				
dmt$access_kind	1518	1524						
dmt$active_volume_table_index	1855	1809						
dmt$allocation_size	1337	404	454	494	1549	1664		
dmt$allocation_styles	1348	1821						
dmt$bytes_per_mau	1900	1814						
dmt$class_member	1361	405	456	1360	1539			
dmt$class_ordinal	1366	406	458	1541				
dmt$dau_address	1874	1968						
dmt$daus_per_allocation	1871	1816						
dmt$daus_per_position	1872	1815						
dmt$delete_count	417	393						
dmt$delete_logging_count	1825	1811						
dmt$device_file_list_index	1384	462	1810	1839				
dmt$disk_file_descriptor	391	6719	6775	7857				
dmt$fau_states	1976	1969						
dmt$file_allocation_unit	1967	1797	1798	1965	6777	7860		
dmt$file_attribute_keywords	509	446	1537	1835				
dmt$file_hash	1394	468						
dmt$file_hash_thread	420	422						
dmt$file_medium_descriptor	1806	410	1820					
dmt$fmd_attribute	1834	1830						
dmt$fmd_index	1894	399	409	1829	1970			
dmt$global_file_name	1400	476	1663	5976				
dmt$internal_vsn	1511	478	1817	1841				
dmt$level_1_index	1772	398	1786	6782				
dmt$level_1_table	1786	397	6784					
dmt$level_2_index	1777	1797	6783					
dmt$level_2_table	1797	6762	6785					
dmt$locked_file	1521	480						
dmt$maus_per_dau	1904	1818						
dmt$maus_per_transfer	1906	1819						
dmt$queue_status	1674	1661						
dmt$requested_volume	1554	408	498					
dmt$system_file_id	1939	1808	3478	5632	5978			
dmt$transfer_size	1681	407	496	1665				
dmt$usage_count	1693	1662						
dmt$write_lock	1520	1526						
eoi	8072	8116/M	8116	8116	8116			
eoi	8165	8195/M	8198	8203	8203			
eoi_byte_address	2004	8116	8116/M	8118	8122	8126/M	8195	8220/M
eoi_modified	2027	8128/M						
eoi_state	2005	8115	8116/M	8123	8127/M	8221/M		
fat_upper_bound	398	6795	7919					
fde_entry_p	8253	8272/P	8278	8282/P	8285/P	8302	8302	8303 8304

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8305/M	8305	8313	8314	8315/M	8331/P	8338	8344/P
		8365/M	8372/M	8372	8376	8377	8378	8379	8380
		8380	8381/P	8388/M	8388	8396	8397	8400	8401/M
		8417/P							
fde_p	3286	8101/P							
fde_p	6718	6727	6730						
fde_p	6812	6814/M	6815	6816/M	6816	6818			
fde_p	6825	6847/M	6848						
fde_p	6869	6876/M	6876						
fde_p	6871	6876/P	6878/P						
fde_p	6988	6997	6998	6999					
fde_p	7049	7054/P							
fde_p	7277	7283	7287/M						
fde_p	7820	7871/P	7871/P						
fde_p	7820	7871/M	7871						
fde_p	7820	7872	7872/M						
fde_p	7820	7906	7906						
fde_p	7859	7871/P	7872/P	7905	7906/P	7914	7918		
fde_p	8072	8101/M	8101	8101/M	8101	8101			
fde_p	8072	8116/P	8116/P	8116	8116/M	8116/M			
fde_p	8072	8116	8116	8116	8132	8132	8132		
fde_p	8072	8116	8116/M	8133	8133/M				
fde_p	8091	8101/M	8105	8106	8108	8112/S	8115	8116/P	8118
		8122	8123	8126/M	8127/M	8128/M	8132/P	8133/P	8137/P
fde_p	8157	8182	8182	8182					
fde_p	8157	8183	8183/M						
fde_p	8158	8182/P	8183/P	8195	8220/M	8221/M			
fde_p	8470	8499	8499	8499					
fde_p	8470	8500	8500/M						
fde_p	8471	8499/P	8500/P						
file_allocation_table	397	6793	6796	7919	7919				
file_entry_index	1944	6837	6876	6998/M	7871	8116/M	8132/M	8182/M	8499/M
file_hash	1946	6836	6848	6876	6876	6997/M	7871	7871	8116/M
		8132/M	8182/M	8499/M					
file_hash	2001	6848	6876	6997	7871	8116	8132	8182	8499
file_limits_enforced	3063	8282/P							
find_eoi	8072	8116	8116	8116					
find_eoi	8202	8202	8211	8218					
find_starting_pft_entry	7973	7973	7981						
fixup_chapter_length	8157	8116	8231						
flags	1994	8128/M							
found	8072	8116	8116						
found	8166	8207	8208						
found	8447	8456	8457						
gfc$fde_size	6864	6837	6876	6998	7871	8116	8132	8182	8499
gfc$fde_table_base	6862	6837	6863	6876	6998	7871	8116	8132	8182
		8499							
gfc$fk_catalog	1573	1585							
gfc$fk_job_local_file	1575	1584							
gfc$fm_mass_storage_file	2055	2017	6727	7905	7906	7918	8281		
gfc$fm_served_file	2056	2020	8284						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gfc$fm_transient_segment	2055	7914	8279						
gfc$monitor_interlocks	6890	6877	7053	7871					
gfc$ps_page_doesnt_exist	6694	8280							
gfc$ps_server_terminated	6701	8293							
gfc$ps_volume_unavailable	6699	8290							
gfc$tr_job	1957	6840	6876	7004	7285	7871	7872	8116	8116
		8132	8133	8182	8183	8499	8500		
gfc$tr_system	1957	6839	6876	7001	7871	8116	8132	8182	8499
gfp$mtr_convert_job_mode_fde_p	6808	6820	8101						
gfp$mtr_convert_job_mode_fde_p	6809	6818/M	8101/M						
gfp$mtr_get_fde_p	6823	6852	6876	7871					
gfp$mtr_get_locked_fde_p	6869	6881	7871						
gfp$mtr_get_sfid_from_fde_p	6987	7011	8116	8132	8182	8499			
gfp$mtr_unlock_fde_p	7049	7057							
gft$allocation_unit_size	2041	2006							
gft$attach_count	2046	1997	1998						
gft$fde_flags	2026	1994							
gft$file_desc_entry_p	2210	3286	6718	6808	6809	6809	6812	6825	7859
gft$file_descriptor_entry	1991	420	1996	2210	2214	6724			
gft$file_descriptor_index	1954	1944							
gft$file_kind	1569	474	1545	1581	2000				
gft$file_media	2055	2016							
gft$locked_file_desc_entry_p	2214	6683	6709	6871	6988	7049	7214	7277	7315
		8091	8158	8253	8471				
gft$open_count	2085	1999	2090						
gft$page_status	6694	6685	6713	8251					
gft$queue_status	1750	506	2009						
gft$segment_lock_info	2089	2002							
gft$signature_lock	2061	1992							
gft$system_file_identifier	1943	1939	2230	2896	3056	3243	4417	6823	6869
		6989	6995	7278	8097	8175	8491		
gft$table_residence	1957	1945	6833						
gft$transfer_unit_size	2052	2007							
gft$trick_pointer	6740	6725							
hash	6830	6836/M	6843/M						
hash	6869	6876/M	6876/M						
hash	7820	7871/M	7871/M						
hash_sva_param2	7620	7625	7626						
head	2100	7508	7509/S	8302	8380	8380	8381	8381/S	
i	8072	8116	8116/S	8116/S					
i	8167	8225	8226/S	8227/S					
i#move	7082	7098	7797	7804					
i#program_error	6754	6728	6849	6876	7116	7138	7871	7906	8411
		8415							
i#real_memory_address	7100	7810							
id	7163	7109	7109/M	7116	7137	7178	7182/M	8411	8411/M
		8411	8415	8415/M	8415				
ijl_ordinal	2223	7285	7872	8116	8133	8183	8500		
ijl_ordinal	3342	7872/P							
ijl_ordinal	6990	7009/M							

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
ijl_ordinal	7037	7005	8116	8132	8182	8499		
ijl_ordinal	8072	8116/P	8116/P					
ijl_ordinal	8072	8116/M	8132/M					
ijl_ordinal	8092	8132/P	8133/P					
ijl_ordinal	8157	8182/M						
ijl_ordinal	8168	8182/P	8183/P					
ijl_ordinal	8470	8499/M						
ijle_p	3343	7871/P						
ijle_p	6824	6841						
ijle_p	6869	6876						
ijle_p	6870	6876/P						
ijle_p	7104	7110	7112/P					
ijle_p	7820	7871/P						
ijle_p	7820	7871						
ijle_p	8241	8411	8411/P	8415	8415/P			
ijle_p	8261	8382/P	8411/P	8415/P				
ijlo	7279	7285						
ijlo	7820	7872						
ijlo	8072	8116	8133					
ijlo	8157	8183						
ijlo	8470	8500						
ijlo	8489	8499/P	8500/P					
in_use	2224	7284	7872	8116	8133	8183	8500	
in_use	7036	7111	7114/M	7114	8411	8411/M	8411	8415	8415/M
		8415						
include_partial_pages	7296	7885/P						
index	2072	7508/M	8381/M					
init_new_io	2381	8345/P	8346/P					
initial_lock_offset	7655	7675/M	7686/P	7691/P				
io_already_active	8264	8345/P						
io_count	8263	8345/P						
io_error	4101	8528/M						
io_function	5304	8343/M						
io_id	8262	8342/M	8343/M	8345/P				
ioc$allocate	5936	5312						
ioc$max_unit_number	1861	1855	1864					
ioc$no_error	4118	8528						
ioc$read_ahead_on_server	5936	5314						
ioc$read_for_server	5935	5310						
ioc$read_from_client	5935	5311						
ioc$read_page	5930	5307						
ioc$swap_in	5931	5305						
ioc$swap_out	5931	5305						
ioc$write_for_server	5936	5310						
ioc$write_locked_page	5933	8259						
ioc$write_page	5930	5307	8259					
ioc$write_to_client	5936	5311						
iot$interrupt	6319	5470						
iot$io_error	4118	3479	4101					
iot$io_function	5930	5304	5631	7317	8259			
iot$logical_unit	1864	6214						
iot$port_number	6324	6321						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
iot$pp_number	6314	5411	5412					
iot$transfer_count	4555	4543						
iotype	8258	8343	8344/P					
jmc$detached_job_wait_time_max	4218	4215						
jmc$highest_det_job_wait_time	4228	4218	4229					
jmc$highest_prio_age_interval	4485	4476	4486					
jmc$highest_service_accumulator	3947	3948						
jmc$highest_service_factor_valu	4509	4502						
jmc$highest_working_set_size	4254	4245	4255	4257	4259	4261		
jmc$ies_job_swapped	3727	3736						
jmc$ies_swapin_in_progress	3726	3735						
jmc$iss_idle_tasks_initiated	3742	3769						
jmc$iss_swapin_io_complete	3767	3770						
jmc$iss_swapin_requested	3763	3770						
jmc$iss_swapout_complete	3762	3769						
jmc$iss_swapped_io_cannot_init	3753	3780						
jmc$iss_swapped_no_io	3744	3779						
jmc$keyword_offset_maximum	3964	4246	4477					
jmc$kjl_maximum_entries	3381	3374	3375	3899				
jmc$kol_maximum_entries	3391	3376						
jmc$lock_ajl	7124	7111	7112/P	7114	8411	8411/P	8411	8415	8415/P
		8415						
jmc$max_active_jobs	3372	4458	4466	4467				
jmc$max_ajl_ord	3373	3366	3372					
jmc$max_dispatching_control	3686	3690						
jmc$max_dispatching_priority	3608	3568	3571	3572				
jmc$maximum_job_classes	3877	3880						
jmc$maximum_job_count	3388	3381						
jmc$maximum_output_count	3398	3391						
jmc$maximum_service_classes	3980	3983						
jmc$min_dispatching_control	3685	3689						
jmc$null_service_class	3973	3974						
jmc$priority_aging_interval_max	4476	4473						
jmc$priority_p1	3622	3569	5263					
jmc$priority_p10	3631	3570						
jmc$priority_p14	3635	3570	5263					
jmc$priority_p8	3629	3569						
jmc$required_offset	3962	4260						
jmc$reserved_ajls	3377	3372						
jmc$service_accumulator_maximum	3939	3936						
jmc$service_factor_value_max	4502	4499						
jmc$system_default_offset	3963	3964	4262					
jmc$system_supplied_name_size	4015	4012						
jmc$unlimited_offset	3960	3949	4219	4230	4256	4487		
jmc$unspecified_offset	3961	4258						
jmc$working_set_size_maximum	4245	4242						
jmp$free_ajl_with_lock	7127	7112	8411	8415				
jmp$unlock_ajl	7103	7118	8411	8415				
jmt$active_job_list	7043	7021						
jmt$active_job_list_entry	7035	7043						
jmt$ajl_ordinal	3366	3323	3415	5630	7107			

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmt$delayed_swapin_work	3552	3445	3556					
jmt$detached_job_wait_time	4215	4200						
jmt$dispatching_control	3656	4441						
jmt$dispatching_control_index	3689	3646	3656					
jmt$dispatching_controls	3659	3657						
jmt$dispatching_priority	3568	3315	3427	3647	3648	3649	3661	4389	4391
		7570	7571					
jmt$ijl_block_index	2202	2198						
jmt$ijl_block_number	2201	2197						
jmt$ijl_dispatching_control	3645	3428						
jmt$ijl_entry_status	3722	3414						
jmt$ijl_ordinal	2196	2223	3342	3434	3462	4030	4031	4093	4190
		5306	6990	6994	7015	7037	7279	7562	8092
		8168	8489					
jmt$ijl_page_fault_count	3796	3791	3792	3793					
jmt$ijl_page_stats	3790	3786						
jmt$ijl_service_class_stats	3784	3449						
jmt$ijl_statistics	3829	3448						
jmt$ijl_swap_count	3805	3801	3802					
jmt$ijl_swap_counts	3800	3468	3787					
jmt$ijl_swap_status	3740	3417	3418	3419				
jmt$initiated_job_list_entry	3411	3343	4055	4189	6824	6870	7038	7104	7128
		7529	8261					
jmt$input_file_location	3919	3914						
jmt$jl_job_leveler_state	5945	5940						
jmt$jl_job_leveler_status	5939	5569						
jmt$job_abort_disposition	3928	3912						
jmt$job_class	3880	3473						
jmt$job_control_block	4171	3325						
jmt$job_mode	3883	3430						
jmt$job_priority	3888	3470	3471	4450	4451	4452	4453		
jmt$job_recovery_disposition	3931	3913						
jmt$job_system_id	4234	4186						
jmt$kjl_index	3899	3416	4234					
jmt$maximum_active_jobs	4458	4435						
jmt$priority_aging_interval	4473	4443						
jmt$queue_file_ijl_information	3911	3455						
jmt$scheduling_data	3461	3439						
jmt$scheduling_priority	4449	4442						
jmt$service_accumulator	3936	3463	3464	3465	4433	4434		
jmt$service_class_index	3983	3474	4426	4436				
jmt$service_class_name	4491	4428	4429					
jmt$service_factor_value	4499	4437						
jmt$service_factors	4495	4437						
jmt$swap_data	3477	3441						
jmt$swapout_reasons	3986	3469						
jmt$swapped_job_entry	4001	3486	4056	4209				
jmt$system_supplied_name	4012	3412	4184					
jmt$task_time_slice	3699	3679	3680					
jmt$time_slice_values	3678	3663	4402					
jmt$user_supplied_name	4238	4185						
jmt$working_set_size	4242	4196	4197					

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmv$ajl_p	7021	7005	7111	7114/M	7114	8116	8132	8182	8411
		8411/M	8411	8415	8415/M	8415	8499		
jmv$null_ijl_ordinal	7015	7002	8116	8132	8182	8499			
jsc$isqi_swapped_io_completed	4035	4037							
jsc$isqi_swapped_io_not_init	4034	4037							
jst$changed_asid_entry	4078	4069							
jst$ijl_swap_queue_id	4034	4029							
jst$ijl_swap_queue_link	4028	3423							
jst$io_control_information	4042	3442							
jst$swap_file_descriptor	4054	3443							
jst$swapped_page_descriptor	4063	4061							
jst$swapped_page_descriptors	4060	4057							
last_pfti_index	7470	7233	7246	7247	7251	7890	7925	7925	7925
last_written_pfti	8265	8345/P	8346/P						
length	2941	8053/P	8060/P						
length	7084	7091	7094/M	7094					
length	7646	7663	7667	7672					
length	7711	7726	7730	7737					
length	7775	7797	7797/M	7797	7804	7804/M	7804		
length	8473	8507							
level_1_index	6782	6794/M	6795	6797/S					
level_1_index	7820	7919/M	7919	7919/S					
level_2_index	6783	6799/M	6800/S						
level_2_index	7820	7919/M	7919/S						
link	4090	7980	8015						
link	4143	7967							
local_fde_p	6724	6730/M	6731	6732					
local_fde_p	7820	7906/M	7906	7906					
lock	6869	6878							
lock	6962	6975							
lock	7049	7054/M							
lock	7064	7076/M							
lock	7103	7109	7109	7109/M	7109/M	7109			
lock	7103	7116	7116	7116/M	7116	7116/M			
lock	7134	7137	7140	7141/M	7141	7143/M			
lock	7170	7178	7180	7182/M	7184/M	7184			
lock	7820	7871							
lock	8241	8411	8411	8411/M	8411/M	8411	8415	8415	8415/M
		8415/M	8415						
lock	8241	8411	8411	8411/M	8411	8411/M	8415	8415	8415/M
		8415	8415/M						
lock_length	7656	7672/M	7697/M	7697	7698				
lock_length	8490	8507/M	8537/M	8537	8538				
lock_page_type	2939	8051	8058						
lock_pages	7644	7705	8053						
lock_pages_loop	7682	7682	7699	7703					
lock_sva	7657	7673/M	7674/M	7674	7674	7675	7684/P	7686/P	7691/P
		7702/M	7702						
locked	2128	6878	6975	7871					
locked	7161	7109	7180	8411	8415				
locked_for_read	2090	8303	8304	8305/M	8305	8313	8372/M	8372	8377

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8378	8379	8388/M	8388	8396	8397	8400	
locked_for_write	2091	8302	8314	8315/M	8338	8365/M	8376	8401/M	
locked_page	4096	7690	7696/M	7745	7746/M				
m	2268	7910	8116	8209	8526/M				
max_eoi	8072	8116/M							
max_eoi	8169	8199/M							
media	2016	6727	7905	7906	7914	7918	8278		
mmc$	2538	2544	2547	2550	2553	2556	2559	2562	2566
		2569	2572	2575	2578	2581	2584	2588	2591
		2594	2597	2600	2603	2607	2610	2613	2616
		2619	2622	2625	2628	2631	2634	2637	2640
		2643	2646	2649	2652	2655	2658	2661	2664
		2667	2671	2674	2677	2680	2683	2686	2689
		2692	2695	2698	2701	2704	2707	2710	2713
		2716	2719	2722	2725	2728	2731	2735	2739
		2742	2746	2749	2752	2755	2758	2761	2764
		2767	2770	2773	2776	2779	2782	2785	2788
		2791	2794	2797	2800	2803	2807	2811	2814
		2817	2820	2823	2826	2829	2832	2835	2838
		2841	2844	2847	2850	2853	2856	2859	2862
		2865	2869	2872	2875				
mmc$assign_active_null	3096	3097							
mmc$cell_pointer	3195	3200							
mmc$eoi_actual	2116	8127							
mmc$eoi_rounded	2116	8116	8221						
mmc$eoi_uncertain	2116	8115	8130						
mmc$heap_pointer	3196	3204							
mmc$iocb_table_size	6673	6666							
mmc$iorc_await_io_completion	6101	6094							
mmc$iorc_write_pages	6101	6092							
mmc$irs_active	6071	6065							
mmc$irs_complete	6071	6066							
mmc$irs_none	6071	6065							
mmc$kw_asid	3121	3157							
mmc$kw_clear_space	3119	3144							
mmc$kw_current_segment_length	3118	3138							
mmc$kw_error_exit_procedure	3120	3148							
mmc$kw_gl_key	3120	3142							
mmc$kw_hardware_attributes	3122	3151							
mmc$kw_inheritance	3122	3159							
mmc$kw_max_segment_length	3119	3140							
mmc$kw_preset_value	3121	3146							
mmc$kw_ps_transfer_size	3123	3167							
mmc$kw_ring_numbers	3117	3133							
mmc$kw_segment_access_control	3121	3155							
mmc$kw_segment_number	3118	3136							
mmc$kw_shadow_segment	3123	3161							
mmc$kw_software_attributes	3120	3153							
mmc$kw_wired_segment	3123	3164							
mmc$lp_aging_lock	2949	7696	7745	8052	8059				
mmc$lp_not_locked	2949	7690	7746						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES				
mmc$lss_lock_for_read_r3	3256	8308	8367	8391		
mmc$lss_lock_for_read_user	3256	8311	8394			
mmc$lss_lock_for_write_r3	3257	8318	8360	8404		
mmc$lss_lock_for_write_user	3256	8321	8407			
mmc$lss_none	3255	8297	8336	8374		
mmc$lss_queued_for_lock_r3	3255	8298	8327	8389	8402	
mmc$lss_queued_for_lock_user	3255	8299	8329			
mmc$lus_free	2403	8353				
mmc$lus_lock_for_read	2400	8301	8387			
mmc$lus_lock_for_write	2401	8313				
mmc$lus_lock_segment	2382	2383	8274			
mmc$lus_none	2402	8355				
mmc$lus_protected_write	2402	8258	8341			
mmc$lus_remove_from_working_set	2403	8351				
mmc$lus_unlock_segment	2382	2385	8335			
mmc$lus_write	2403	8258	8341			
mmc$max_rma_list_length	5951	5956	5957			
mmc$pq_avail	2284	2330				
mmc$pq_free	2283	2342	8116/P	8227/P		
mmc$pq_job_fixed	2324	2331	2343			
mmc$pq_job_working_set	2326	2343	2344			
mmc$pq_shared_first_site	2334	2338				
mmc$pq_shared_num_sites	2335	2338				
mmc$pq_shared_other	2293	2333				
mmc$pq_shared_site_01	2295	2334				
mmc$pq_shared_site_25	2319	2339				
mmc$pq_shared_task_service	2288	2332				
mmc$pq_swapped_io_error	2322	2342				
mmc$pq_wired	2286	2329	7967/S			
mmc$sat_read_or_write	7406	8269/P	8452/P			
mmc$segment_fault_processor_id	4860	4914				
mmc$sequence_pointer	3195	3202				
mmc$server_iocb_table_size	5968	5971				
mmc$sf_get_segment_length_fde_p	3291	8104				
mmc$sf_set_segment_length_fde_p	3292	8121				
mmc$ssk_none	3275	3247				
mmc$ssk_segment_number	3276	3245				
mmc$wmp_io_active	7330	8347	8348			
mmc$wmp_io_complete	7330	8347				
mme$invalid_request	2671	8055/P	8062/P	8065/P	8357/P	8420/P
mme$lock_unlock_invalid_length	2686	7668/P	7731/P			
mme$no_matching_offset	2667	7982/P				
mme$not_valid_in_page_table	2658	7635				
mme$page_already_locked	2649	7692/P				
mme$page_not_in_page_table	2556	7628	8502/P	8515/P	8521/P	8533/P
mme$segment_locked_another_task	2713	8324/P				
mme$segment_locked_by_task	2710	8300/P				
mme$segment_not_locked	2716	8337/P				
mme$volume_unavailable	2807	8291/P				
mmp$asid	7199	7874	8116	8137	8187	8505
mmp$convert_pva	7211	8272				
mmp$delete_pt_entry	7222	8116	8226			

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmp$fetch_pfti_array_size	7229	7235	7890					
mmp$find_next_pfti	7238	7256	7925					
mmp$get_max_sdt_sdtx_pointer	7259	7269	7795	8385				
mmp$get_verify_asti_in_fde	7276	7290	7872	8116	8133	8183	8500	
mmp$initialize_find_next_pfti	7294	7885						
mmp$mm_free_pages	7303	8140	8354					
mmp$mm_write_modified_pages	7312	8344						
mmp$modify_pages	8470	8545						
mmp$mtr_change_segment_table	7775	7816						
mmp$mtr_fetch_offset_mod_pages	7820	7830						
mmp$mtr_fetch_pva_unwritten_pgs	7934	8020						
mmp$mtr_lock_unlock_pages	8024	8068						
mmp$mtr_lock_unlock_segment	8241	8424						
mmp$mtr_set_get_segment_length	8072	8149						
mmp$mtr_wait_io_completion	8441	8466						
mmp$process_wmp_status	7370	8346						
mmp$purge_all_cache_map	7335	8116	8224					
mmp$purge_all_cache_map_proc	7350	7341	8116	8224				
mmp$relink_page_frame	7379	8116	8227	8530				
mmp$remove_pages_working_set	7384	8352						
mmp$verify_pva	7398	8269	8452					
mmp$xtask_pva_to_sva	7392	7962	8044	8454				
mmt$active_io_count	6100	5985	6089	7321	8263			
mmt$active_segment_table	2235	7439						
mmt$active_segment_table_entry	2220	2236	4066	4100	7206	7215	7298	7305
		7316	7386	7853	8090	8162	8246	
mmt$ast_index	2107	2003	2990	3485	4081	7199	7280	7854
		8089	8163	8488				
mmt$attribute_keyword	3117	3132						
mmt$eoi_state	2116	2005	8094					
mmt$global_page_queue_index	2342	4156						
mmt$global_page_queue_list	4156	7485						
mmt$global_page_queue_list_ent	4146	4156						
mmt$hardware_attribute_set	3186	3152						
mmt$hardware_attributes	3174	3186						
mmt$io_identifier	5302	5629	7320	8262				
mmt$io_request_status	6071	6064						
mmt$io_status	6063	6073	6086					
mmt$iocb_index	6666	5309	5315					
mmt$job_page_queue_index	2343	4003	4157					
mmt$job_page_queue_list	4157	3440						
mmt$link	2243	2221	4090	4091	4143			
mmt$lock_segment_status	3255	3061						
mmt$locked_page	2949	2939	4096					
mmt$lus_lock_type	2400	2384						
mmt$lus_page_disposition	2402	2386						
mmt$max_sdt	3000	3004						
mmt$max_sdt_p	3004	7261	7782	8255				
mmt$max_sdtx	3085	3089						
mmt$max_sdtx_p	3089	7262	7783	8256				
mmt$memory_reserve_request	4124	3433						
mmt$page_age	4111	4099	4115	4115				

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmt$page_frame_index	2249	2245	2245	4043	4045	4046	4047	4126
		4127	7223	7239	7243	7299	7323	7371
		7379	7471	7621	7862	7958	8173	8265
		8449	8494					
mmt$page_frame_queue_id	2344	2229	4044	4094	7380			
mmt$page_frame_table	4105	7477						
mmt$page_frame_table_entry	4089	4064	4105	7610	7658	7722	7957	8493
mmt$page_queue_list_entry	4142	4147	4157					
mmt$page_selection_criteria	2955	7297						
mmt$pfti_array	7467	7461						
mmt$rb_change_segment_table	2528	7776						
mmt$rb_fetch_offset_mod_pages	2893	7821						
mmt$rb_fetch_pva_unwritten_pgs	2919	7935						
mmt$rb_lock_unlock_pages	2936	8025						
mmt$rb_lock_unlock_segment	2376	8242	8252					
mmt$rb_set_get_segment_length	3284	8073						
mmt$rb_wait_io_completion	2350	8442						
mmt$rma_list_entry	5959	5660	5675	5954				
mmt$rma_list_index	5956	5954						
mmt$sdtx_stream_data	3068	3064						
mmt$segment_access_condition	4887	4915						
mmt$segment_access_rights	3219	3060						
mmt$segment_access_state	3225	3055						
mmt$segment_access_type	7406	7399						
mmt$segment_descriptor	2987	2997	3001	7216	8247			
mmt$segment_descriptor_extended	3053	3082	3086	7217	7804/P	8248		
mmt$segment_inheritance	3103	3057	3160					
mmt$segment_pointer_kind	3195	3199						
mmt$segment_reservation_state	3265	3058						
mmt$server_iocb_entry	5975	5637	5972					
mmt$server_state	6107	5977						
mmt$set_get_subfunction_codes	3291	3288						
mmt$shadow_info	3240	3062						
mmt$shadow_reference_info	4512	4415						
mmt$shadow_segment_kind	3275	3244						
mmt$software_attribute_set	3188	3059	3154					
mmt$software_attributes	3182	3188						
mmt$sub_reqcodes	6101	5982	6091					
mmt$write_modified_pages_status	7330	7324	7370	8266				
mmt$xcb_page_wait_info	4523	4401						
mmv$ast_p	7439	7284	7284	7285	7872	7872	7872	7876
		8116	8116	8116	8133	8133	8133	8135
		8183	8183	8183	8500	8500		
mmv$gpql	7485	7967						
mmv$multiple_caches	7448	7340	8116	8224				
mmv$multiple_page_maps	7455	7340	8116	8224				
mmv$pft_p	7477	7638	7910/S	7912	7916	7919/P	7922	7975
		7980	8003	8459	8461/P			
mmv$pfti_array_p	7461	7233	7233	7246	7246	7247	7247	7248/M
		7248	7249	7249/S	7251	7251	7890	7890
		7925	7925	7925/M	7925	7925/S	7925	7925
		7925	7925					
mmv$pt_p	7492	7632	7633	7910	8116	8116/M	8116	8209
		8214/M	8215	8458	8519	8526/M	8529	

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
modify_pages_loop	8511	8511	8539	8543					
monitor_lock	1993	6878/P	7054/P	7871/P					
mtc$job_fixed_segment	3298	6841	6876	7005/S	7871	8116/S	8132/S	8182/S	8499/S
mtp$clear_interlock	7064	7054	7078						
mtp$error_stop	6906	8146	8287						
mtp$set_interlock	6962	6878	6983	7871					
mtp$set_status_abnormal	7420	7427	7668	7692	7731	7982	8055	8062	8065
		8291	8294	8300	8324	8337	8357	8420	8502
		8515	8521	8533					
mtt$monitor_interlock	2123	1993	6962	7064					
mtv$csto	7414	7410							
nat$received_message_descriptor	4539	4532	4541						
nat$received_message_list	4531	4383							
new_sdt_length	2533	7814							
new_sdt_offset	2531	7794	7813						
new_sdt_p	7780	7794/M	7797/P	7810/P					
new_sdtx_offset	2532	7802	7803	7805					
new_sdtx_p	7781	7803/M	7804/P						
new_segment_length	8093	8124/M	8126	8131	8139				
next_offset_to_return	2925	7976	8007/M						
nlc$cc_connect_confirm	4571	4562							
nlc$cc_connect_request	4570	4560							
nlc$cc_expedited_data	4576	4562							
nlc$cc_max_pdu_kind	4578	4581							
nlc$channel_connection_pdu	4594	4546							
nlc$channelnet_pdu	4594	4548							
nlt$cc_pdu_kind	4581	4559							
nlt$cc_seq#_or_connect_time	4558	4547							
nlt$cc_sequence_number	4584	4563							
nlt$device_identifier	4591	4542							
nlt$pdu_type	4594	4545							
normal	2362	7425/M	7625	7627	7633/M	7634	7662/M	7668/M	7685
		7692/M	7725/M	7731/M	7745	7757/M	7788/M	7867/M	7963
		7982/M	8045	8055/M	8062/M	8065/M	8271	8291/M	8294/M
		8300/M	8324/M	8337/M	8357/M	8414/M	8420/M	8453	8455
		8497/M	8502/M	8514	8515/M	8521/M	8533/M		
null_sva	7338	7343	7344						
null_sva	8072	8116	8116						
null_sva	8157	8224	8224						
offset	585	8112							
offset	1706	7667	7672	7674/M	7674	7674	7675	7686/P	7691/P
		7702/M	7702	7730	7736/M	7736	7736	7737	7754/M
		7754	7882/M	7912	7916	7919/P	7922	7976	7990/M
		7992/M	7992	8004	8004	8007	8012	8116/M	8139/M
		8206/M	8282/P	8285/P	8339/M	8509/M	8542/M	8542	
offset	6747	6732/M	7906/M						
offset	6832	6837/M	6844/M	6847					
offset	6869	6876/M	6876/M	6876					
offset	7820	7871/M	7871/M	7871					
offset	8072	8116/M	8116	8116	8116/M	8116	8116	8116	8116/M

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8116	8116	8116					
offset	8170	8198/M	8199	8203	8203	8204/M	8204	8205	8206
		8210/M	8210	8220					
offset	8472	8507	8509						
offset_list	2898	7897							
offset_list	2928	8012/M							
offset_list_index	7858	7903/M	7909	7909	7911/M	7911	7912/S	7915/M	7915
		7916/S	7921/M	7921	7922/S	7928			
offset_list_index	7956	7995/M	8005	8011/M	8011	8012/S	8018		
offset_list_overflow	2926	7987/M	8006/M						
offsets_returned	2897	7878/M	7892	7893/M	7909	7909	7928/M		
offsets_returned	2927	8008/M	8018/M						
old_eoi_state	8094	8123/M	8130						
old_sdt_p	7782	7795/P	7797/P						
old_sdtx_p	7783	7795/P	7804/P						
old_segment_length	8095	8122/M	8131						
osc$aging_interval_maximum	4266	4269							
osc$call_instruction	4774	4782							
osc$data_read	4773	4782							
osc$free_running_clock_maximum	1720	1717							
osc$invalid_ring	531	571							
osc$max_channel_number	6464	6467							
osc$max_fault_contents	4927	4921							
osc$max_idle_count	4330	4338							
osc$max_integer	6120	6125	6126						
osc$max_name_size	808	812	815	1019					
osc$max_number_of_processors	4315	3310	7152						
osc$max_page_frames	2253	2222	2249	3480	3481	4002	4004	4144	4150
		7468	7469	7470					
osc$max_page_size	1500	1496							
osc$max_page_table_entries	2254	2257							
osc$max_ring	530	571	572						
osc$max_segment_length	554	577	760	3065	3096				
osc$max_status_condition_code	1087	1083	1099						
osc$max_status_condition_number	7430	7421							
osc$max_string_size	1103	1106	1109	1114					
osc$max_tasks	2080	2077							
osc$maximum_offset	553	554	574	574	575	8116	8205		
osc$maximum_processor_id	4799	4795							
osc$maximum_processor_number	4307	4302							
osc$maximum_processors	4311	4307	4315						
osc$maximum_segment	552	573							
osc$min_integer	6119	6123	6124						
osc$min_page_size	1499	1496							
osc$min_ring	529	572							
osc$pr_base_constant	6924	7109	7116	7137	7177	8411	8411	8415	8415
osc$purge_all_cache	7356	7343	8116	8224					
osc$purge_all_page_seg_map	7365	7344	8116	8224					
osc$subsystem_lock_activity	17	8310/P	8320/P	8363/P	8370/P	8393/P	8406/P		
osc$system_lock_activity	18	8307/P	8317/P	8361/P	8368/P	8390/P	8403/P		
osc$task_time_slice_maximum	3710	3713							
osc$wait	2412	8325	8348						

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$aging_interval	4269	4198	4199						
ost$asid	1709	450	1657	1705	2227	3021	3158	3424	4068
		4079	4080	7200	7205	7852	8088	8161	8487
ost$binary_unique_name	1408	1400	1511	1622	1995				
ost$byte_count	1699	2925	2928	2941	7646	7711	8473		
ost$clear_file_space	1013	460	615						
ost$cp_time	3817	3785	3830	4400					
ost$cp_time_value	3815	3466	3818	3819	4193	4194	4413		
ost$cpu_element_id	4299	3341							
ost$cpu_idle_statistics	4333	3344							
ost$cpu_memory_port_mask	4301	3317							
ost$cpu_running_or_stepped	4351	4348	4348						
ost$cpu_state	4346	3326							
ost$cpu_state_reason	4357	3347							
ost$cpu_state_table	3313	3310	6809	7212	7408	7521	7777	7822	7936
		8026	8074	8243	8443				
ost$cs_lock	2137	4381							
ost$cst_trace_control	5162	3345							
ost$date_time	6130	5593							
ost$debug_code	4773	4761							
ost$debug_list	4769	4673							
ost$debug_list_entry	4760	4769							
ost$debug_mask	4779	4672							
ost$exchange_package	4622	4368							
ost$execute_privilege	3034	3016	3029						
ost$execution_control_block	4367	3327	4393	7260	7514	7523	7528	7786	8250
ost$external_interrupt_request	5150	3333							
ost$family_name	4279	4274							
ost$flags	4679	4629							
ost$frame_descriptor	4737	4752							
ost$free_running_clock	1717	2011	2226	3435	3436	3437	3438	3472	3482
		3483	3484	3650	3662	4192	4201	4399	7040
ost$global_task_id	2071	2013	2064	3322	3429	3458	4188	4378	4379
		4811	5014	5308	5621	7498	7505	7527	7534
		8249							
ost$halfword	1723	4319							
ost$idle_type	4342	4337							
ost$key_lock	560	3022	3143						
ost$key_lock_value	566	563	4696	4698					
ost$keypoint_class	4711	4642	4713						
ost$keypoint_mask	4713	4645							
ost$logical_processor_id	4302	3318							
ost$minimum_save_area	4747	4634	4722	4908					
ost$monitor_condition	4598	4605							
ost$monitor_conditions	4605	4635	4639	4727	4983	4997			
ost$monitor_fault	4904	4853							
ost$monitor_fault_contents	4921	4917							
ost$name	815	778	839	885	916	1289	1621	1630	4238
		4277	4279	4427	4491	4946	5339	5369	
ost$non_negative_integers	6125	5576							
ost$p_register	4694	4623	4748	4975	4981				
ost$page_id	2259	2269							

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$page_size	1496	1477	7596						
ost$page_table	2273	7492							
ost$page_table_entry	2264	2273	4065						
ost$page_table_index	2257	2273	4097	7622					
ost$paging_statistics	3853	3831	4408						
ost$parcel	1725	3339	3340						
ost$physical_channel_number	6467	6409							
ost$pre_processed_for_reconfig	5158	3348							
ost$processor_element_id	4318	4299							
ost$processor_element_number	4327	4320							
ost$processor_id	4795	4371	4789						
ost$processor_id_set	4789	4370							
ost$processor_model_number	1426	1410	4285	4321					
ost$processor_serial_number	1504	1409	4286	4322					
ost$pva	582	4667	4685	4699	4905	4998	7847		
ost$read_privilege	3037	3017	3030						
ost$real_memory_address	1697	3338	5439	5540	6276				
ost$register_number	4690	4664	4733	4741	4742	4743			
ost$ring	571	583	3019	3020	3054	3134	3135	4684	
ost$ring1_termination_reason	4807	4404							
ost$segment	573	584	2012	2533	3137	3246	4662	4763	6993
ost$segment_access_control	3027	3156							
ost$segment_descriptor	3014	2988							
ost$segment_length	577	502	2967	3139	3141	3163	3165	3168	3287
		5980	6085	7295	7304	7314	7385	8165	8169
ost$segment_offset	574	585	1706	2898	3069	4764	4766	5979	6684
		6710	7655	7667	7730	7845	7856	8472	
ost$signature_lock	2138	424	5345						
ost$stack_frame_save_area	4721	4755	4941						
ost$state_tables	3310	7414							
ost$status	1071	1056	1283	3150	4976				
ost$status_condition	1095	2363	6067	6088					
ost$status_condition_code	1099	1074	1095						
ost$string	1112	1075							
ost$string_size	1106	1113							
ost$system_flag	5082	5078							
ost$system_virtual_address	1704	4102	7213	7294	7303	7313	7384	7393	7609
		7645	7657	7710	7721	7863	7959	8041	8098
		8176	8260	8450	8492				
ost$task_index	2077	2072	2100	2101	3352	7559	7572		
ost$task_time_slice	3713	3699							
ost$top_of_stack_pointer	4682	4674							
ost$trap_enable	4716	4631	4972						
ost$user_condition	4608	4615							
ost$user_conditions	4615	4633	4637	4725	4754	4944	4984		
ost$user_identification	4272	4187							
ost$user_name	4277	4273							
ost$valid_relative_pointer	580	2018	2021	4397	4398				
ost$valid_ring	572	978	979	980	4674				
ost$virtual_machine_identifier	4704	4625	4627	4749					
ost$wait	2412	2380	2968	6087	7372				
ost$word	1727	5476							

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$write_privilege	3040	3018	3031						
ost$x_register	4691	4664	4733						
osv$cpus_logically_on	7152	7109	7116	7136	7176	8411	8411	8415	8415
osv$page_size	7596	7632	7672	7674	7697	7702	7736	7737	7749
		7754	7992	7992	8116	8116	8116	8116	8131
		8131	8198	8204	8210	8215	8458	8529	8537
		8542							
p	6743	6731/M	6733	7906/M	7906				
p_dfd	6775	6793	6794	6795	6796	6796	6799	6799	
p_dfd	7820	7919	7919	7919	7919	7919	7919	7919	
p_fau	7860	7919/P	7920	7920					
p_fau_entry	6777	6788/M	6800/M						
p_fau_entry	7820	7919/M	7919/M						
p_level1	6784	6796/M	6797/P						
p_level1	7820	7919/M	7919/P						
p_level2	6785	6797/P	6798	6800					
p_level2	7820	7919/P	7919	7919					
p_offset	6761	6765	6766	6766					
p_offset	6775	6797	6797	6797					
p_offset	7820	7919	7919	7919					
p_register	4623	8106	8108	8306	8316	8326			
page_count_freed	8096	8140/P							
page_disposition	2386	8340	8343/S	8344/S					
page_status	8251	8280/M	8283/P	8285/P	8290	8293			
page_wait_info	4401	8460/M							
pages_freed	8072	8116/M	8116/M	8116	8116/S	8116	8116		
pages_freed	8172	8200/M	8213/M	8213	8215/S	8223	8225		
pages_in_memory	2222	8136							
pfc$execute	1022	1024	1027						
pfc$read	1021	1024	1027						
pft$share_options	1027	1028							
pft$usage_options	1024	1025							
pft$usage_selections	1025	609							
pft_entry	7957	7975/M	7976	7976	8003/M	8004	8004	8007	8012
		8015							
pft_index	7862	7886/P	7909	7909	7910/S	7912/S	7916/S	7919/S	7922/S
		7925/P							
pft_index	7958	7967/M	7974	7974	7975/S	7980/M	7980/S	8002	8002
		8003/S	8015/M						
pfte_p	7610	7638/M							
pfte_p	7658	7684/P	7690	7696/M					
pfte_p	7722	7744/P	7745	7746/M					
pfte_p	8493	8513/P	8519/S	8526/S	8528/M	8529/S	8530/P	8531	
pfti	7243	7245/M	7246	7246	7249/M	7252/M	7254		
pfti	7621	7632/M	7638/S						
pfti	7820	7925/M	7925	7925/M	7925	7925/M	7925		
pfti	8449	8458/M	8459/S	8461/S					
pfti	8494	8529/M	8530/P						
pfti_array	8072	8116/M	8116/P	8116/P					
pfti_array	8173	8215/M	8226/P	8227/P					
pfti_array_size	7861	7890/P	7892	7893					

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pfti_first	7468	7233	7890						
pfti_index	7469	7246	7247	7248/M	7248	7249/S	7251	7925	7925/M
		7925	7925/S	7925	7925				
pfti_size	7230	7233/M							
pfti_size	7820	7890/M							
pftis	7471	7249	7925						
pmc$kill_task_flag	5082	5098							
pmc$mainframe_id_size	6148	6145							
pmc$max_signal_contents	5065	5059							
pmc$max_task_id	4820	4817							
pmc$processor_model_number_size	6156	6148	6153						
pmc$processor_serial_num_size	6206	6149	6203						
pmt$binary_mainframe_id	4284	4191	5567						
pmt$condition_identifier	4894	4888							
pmt$cpu_model_number	1486	1475	1482						
pmt$cpu_serial_number	1489	1476	1481						
pmt$initialization_value	2181	2010	3147						
pmt$mainframe_id	6145	5568	6373						
pmt$program_name	916	617	621	637	686	904	6571		
pmt$sense_switches	4293	4202							
pmt$signal	5021	5015							
pmt$signal_contents	5059	5023							
pmt$signal_id	5026	5022							
pmt$task_id	4817	4395	4812						
pqle	4147	7967							
psc_all_except_avail	2955	7885/P							
pti	4097	7910/S	8519/S	8526/S	8529/S				
pti	7622	7626/M	7632/S	7633/S					
pti	8072	8116	8116/S	8116/S	8116/S				
pti	8174	8207	8209/S	8214/S	8215/S				
pti	8448	8456	8458/S						
ptr_level_2	6762	6766/M	6768/M						
ptr_level_2	6775	6797/M	6797/M						
ptr_level_2	7820	7919/M	7919/M						
pva	2353	8452/P	8454/P						
pva	2379	8269/P	8272/P	8386/S					
pva	2922	7962/P							
pva	2940	8044/P							
pva	4524	8460/M							
pva	4685	8112							
pva	4699	8106	8108	8306	8316	8326			
pva	6725	6731/M	6732/M	6733					
pva	6745	6732/M	7906/M						
pva	7820	7906/M	7906/M	7906					
qrb_p	8252	8384/M	8386/S	8387	8414/M				
queue_id	2229	8530/P							
rb	7821	7867/M	7871/P	7872/P	7878/M	7892	7893/M	7897	7898
		7909	7909	7913	7917	7928/M			
rb	7935	7962/P	7962/P	7963	7968	7976	7982/P	7987/M	7989
		8006/M	8007/M	8008/M	8012/M	8018/M			

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
rb	8025	8044/P	8044/P	8045	8049	8051	8053/P	8053/P	8055/P
		8058	8060/P	8060/P	8062/P	8065/P			
rb	8242	8269/P	8269/P	8271	8272/P	8273	8291/P	8294/P	8300/P
		8301	8313	8324/P	8325	8337/P	8340	8343/S	8344/S
		8345/P	8346/P	8346/P	8346/P	8348	8357/P	8420/P	
rb	8442	8452/P	8452/P	8453	8454/P	8454/P	8455		
reqcode	2937	8049							
request	2382	8273							
request_block	7776	7788/M	7794	7802	7803	7805	7813	7814	
request_block	8073	8101/P	8103	8107/M	8109/M	8111/M	8118/M	8124	
residence	1945	6835	6876	7001/M	7004/M	7285	7871	7872	8116/M
		8116/M	8116	8132/M	8132/M	8133	8182/M	8182/M	8183
		8499/M	8499/M	8500					
residence	6833	6835/M	6838	6840					
residence	6869	6876/M	6876	6876					
residence	7820	7871/M	7871	7871					
return_unallocated_offsets	2899	7913	7917						
ring	583	8106	8108	8306	8316	8326			
rma	2270	7632	8116	8215	8458	8529			
rmc$external_vsn_size	1589	1595							
rmc$recorded_vsn_size	1592	1602							
rmc$unspecified_file_class	1377	1370							
rmt$external_vsn	1595	1609							
rmt$recorded_vsn	1602	492	1543	1555	1608	1843			
rmt$volume_descriptor	1607	1614							
sdt_entries	7785	7796/M	7797/P	7804/P					
sdt_offset	4397	7266	7795	7813/M	8385				
sdt_p	7261	7266/M							
sdt_p	7775	7795/M							
sdt_p	8241	8385/M							
sdt_p	8255	8385/P							
sdt_rma	7784	7810/P	7811	7812					
sdtx_offset	4398	7267	7795	7805/M	8385				
sdtx_p	7262	7267/M							
sdtx_p	7775	7795/M							
sdtx_p	8241	8385/M							
sdtx_p	8256	8385/P	8386						
sdtx_table	3086	8386							
search_pft	8001	8001	8016						
search_pft_for_offset	7970	7970	7977	7984					
seg	584	7898/M							
seg	6831	6838/M	6841/M	6847					
seg	6869	6876/M	6876/M	6876					
seg	7820	7871/M	7871/M	7871					
segment_length	3287	8107/M	8109/M	8111/M	8118/M	8124			
segment_lock	2002	8302	8302	8303	8304	8305/M	8305	8313	8314
		8315/M	8331/P	8338	8365/M	8372/M	8372	8376	8377
		8378	8379	8380	8380	8381/P	8388/M	8388	8396
		8397	8400	8401/M	8417/P				
segment_lock	3061	8297	8298	8299	8308/M	8311/M	8318/M	8321/M	8327/M
		8329/M	8336	8360	8367	8374/M	8389	8391/M	8394/M

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8402	8404/M	8407/M					
segment_table_address_1	4666	7811/M							
segment_table_address_2	4668	7812/M							
segment_table_length	4662	7796	7814/M						
segnum	6993	6999/M	7000	7005/S					
segnum	8072	8116/M	8116	8116/S	8132/M	8132	8132/S		
segnum	8157	8182/M	8182	8182/S					
segnum	8470	8499/M	8499	8499/S					
seqno	2073	7509/M	8381/M						
sequence_number	7560	7509	8381						
set_modified_bit	8474	8525							
sfid	2230	7284	7872	8116	8133	8183	8500		
sfid	2896	7871/P	7872/P						
sfid	6823	6835	6836	6837	6848				
sfid	6869	6876/P							
sfid	6869	6876	6876	6876	6876				
sfid	6989	7008/M							
sfid	7278	7284	7285						
sfid	7820	7871/P							
sfid	7820	7871	7871	7871	7871				
sfid	7820	7872	7872						
sfid	8072	8116/P	8116/P						
sfid	8072	8116/M	8132/M						
sfid	8072	8116	8116	8133	8133				
sfid	8097	8132/P	8133/P						
sfid	8157	8182/M							
sfid	8157	8183	8183						
sfid	8175	8182/P	8183/P						
sfid	8470	8499/M							
sfid	8470	8500	8500						
sfid	8491	8499/P	8500/P						
sft$counter	3863	3832	3833	4203	4205	4206	4208		
sft$file_space_limit_kind	1759	488	3063	6711					
source	7082	7092							
source	7775	7797	7804						
specified	5303	8342/M							
stack_for_ring	2014	8105	8106	8108	8112/S				
starting_with_first_page	2923	7989							
state	1969	7920							
status	2352	8452/P	8453	8454/P	8455				
status	2378	8269/P	8271	8291/P	8294/P	8300/P	8324/P	8337/P	8346/P
		8357/P	8414/M	8420/P					
status	2530	7788/M							
status	2895	7867/M							
status	2921	7962/P	7963	7982/P					
status	2938	8044/P	8045	8053/P	8055/P	8060/P	8062/P	8065/P	
status	7422	7425/M	7426/M						
status	7611	7625	7627	7628/M	7633/M	7634	7635/M		
status	7644	7668/M	7668/M	7692/M	7692/M				
status	7647	7662/M	7668/P	7684/P	7685	7691/P	7692/P		
status	7709	7731/M	7731/M						
status	7712	7725/M	7731/P	7744/P	7745	7757/M			

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
status	7934	7982/M	7982/M						
status	8024	8055/M	8055/M	8062/M	8062/M	8065/M	8065/M		
status	8241	8291/M	8291/M	8294/M	8294/M	8300/M	8300/M	8324/M	8324/M
		8337/M	8337/M	8357/M	8357/M	8420/M	8420/M		
status	8470	8502/M	8502/M	8515/M	8515/M	8521/M	8521/M	8533/M	8533/M
status	8475	8497/M	8502/P	8513/P	8514	8515/P	8521/P	8533/P	
ste_p	8247	8272/P							
str1	7087	7092/M	7094						
str1	7775	7797/M	7797	7804/M	7804				
str2	7088	7093/M	7094/M	7095					
str2	7775	7797/M	7797/M	7797	7804/M	7804/M	7804		
stt$set_name	1621	500	1551	1556					
stxe_p	8248	8272/P	8282/P	8297	8298	8299	8308/M	8311/M	8318/M
		8321/M	8327/M	8329/M	8336	8360	8367	8374/M	8386/M
		8389	8391/M	8394/M	8402	8404/M	8407/M		
subfunction_code	3288	8103							
subsequent_request_for_same_pva	2924	7968							
sva	4102	7912	7916	7919/P	7922	7976	7976	8004	8004
		8007	8012						
sva	7609	7625							
sva	7645	7667	7672	7673	7686/P	7691/P			
sva	7710	7730	7735	7737					
sva	7863	7875/M	7882/M	7885/P					
sva	7959	7962/P	7976	7990/M	7992/M	7992	8004	8004	
sva	8041	8044/P	8053/P	8060/P					
sva	8072	8116/M	8116/M	8116					
sva	8098	8138/M	8139/M	8140/P					
sva	8176	8196/M	8206/M	8207					
sva	8260	8272/P	8282/P	8285/P	8339/M	8344/P	8352/P	8354/P	
sva	8450	8454/P	8456						
sva	8492	8508/M	8509/M	8513/P	8542/M	8542			
syc$rc_lock_pages	2474	8050							
syc$rc_unlock_pages	2475	8057							
syc$ucr_condition	4932	4943							
syc$user_defined_condition	4933	4945							
syt$180_idle_code	5170	3334							
syt$monitor_flag	4839	4824							
syt$monitor_flags	4824	4369	7567						
syt$monitor_request_code	2420	2351	2377	2529	2894	2920	2937	2964	3285
		6082							
syt$monitor_status	2361	2352	2378	2530	2895	2921	2938	2965	6083
		7374	7394	7400	7422	7611	7647	7659	7712
		8475							
task_queue	2092	8302	8331/P	8380	8380	8381/P	8417/P		
task_queue	4098	8461/P							
task_queue	7504	7508	7509/S						
task_queue	8241	8381	8381/S						
taskid	3322	8331/P	8461/P						
taskid	7505	7508/M	7509/M						
taskid	8241	8381/M	8381/M						
taskid	8249	8381/P	8382/P	8417/P					

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$broken_task_fault_id	4860	4910							
tmc$btc_invalid_a0	4958	4979							
tmc$btc_invalid_p	4958	4979							
tmc$btc_mcr_traps_disabled	4959	4980							
tmc$btc_mf_traps_disabled	4958	4978							
tmc$btc_mntr_fault_buffer_full	4957	4978							
tmc$btc_system_error	4960	4974							
tmc$btc_ucr_traps_disabled	4959	4980							
tmc$dummy_fault	4861	4916							
tmc$flag_available_31	5095	5099							
tmc$maximum_monitor_faults	4865	4856							
tmc$maximum_signals	5075	5072							
tmc$maximum_system_task_id	5108	5111							
tmc$mcr_fault	4860	4912							
tmc$signal_available_63	5057	5068							
tmc$stid_null_task	5114	5111							
tmc$ts_io_wait_not_queued	5251	5233							
tmc$ts_io_wait_queued	5254	8461/P							
tmc$ts_page_wait	5253	5234							
tmc$ts_ready	5238	5229							
tmc$ts_segment_lock_wait	5253	8331/P							
tmc$ts_timed_wait_not_queued	5245	5232							
tmc$ts_timeout_reqexp_longvlong	5243	5231							
tmc$ts_timeout_reqexp_shortshrt	5242	5230							
tmp$clear_lock	7134	7116	7147	8411	8415				
tmp$dequeue_task	7497	8417							
tmp$get_taskid_from_task_queue	7503	7510	8381						
tmp$get_xcb_p	7527	8382							
tmp$mtr_begin_lock_activity	7513	8307	8310	8317	8320	8390	8393	8403	8406
tmp$mtr_end_lock_activity	7520	8361	8363	8368	8370				
tmp$queue_task	7534	8331	8461						
tmp$set_lock	7170	7109	7188	8411	8415				
tmt$broken_task_condition	4957	4973							
tmt$broken_task_monitor_fault	4971	4911							
tmt$dispatch_control	5200	3329							
tmt$dual_state_priority_entry	5267	3316	5264						
tmt$idle_status	5256	7565							
tmt$mcr_faults	4996	4913							
tmt$monitor_fault_buffer	4850	4406							
tmt$monitor_fault_buffers	4856	4851	4852	4853					
tmt$monitor_fault_identifiers	4859	4909	4985						
tmt$primary_task_list	7576	7542							
tmt$primary_task_list_entry	7558	7576							
tmt$ptl_flags	7582	7569							
tmt$ptl_lock	7158	7134	7170	7194					
tmt$signal	5013	5008							
tmt$signal_buffer	5005	4407							
tmt$signal_buffers	5072	5006	5007	5008					
tmt$system_flags	5078	4382	7568						
tmt$system_task_id	5111	4373							
tmt$task_queue_link	2099	2092	4098	7497	7504	7536	7566		
tmt$task_status	5238	5203	7535	7563	7564				

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

MMP$MODIFY_PAGES

IDENTIFIER	DEFINED ON LINE	REFERENCES						
tmt$wait_inhibited	7589	7584						
tmt$xcb_offset_size	7580	7561						
tmv$ptl_lock	7194	7109/P	7116/P	8411/P	8411/P	8415/P	8415/P	
tmv$ptl_p	7542	7509	8381					
tos_registers	4674	8112						
unlock_length	7720	7737/M	7749/M	7749	7750			
unlock_page_frames	7742	7742	7751	7755				
unlock_pages	7709	7686	7691	7759	8060			
unlock_status	7659	7686/P						
unlock_sva	7721	7735/M	7736/M	7736	7736	7744/P	7754/M	7754
v	2265	7633	8116/M	8214/M	8519			
wait	2380	8325	8346/P	8348				
wmp_status	8266	8345/P	8346/P	8347	8347	8348		
x_registers	4664	8384						
xcb_p	3327	6816	7789	8101	8106	8108	8109	8112
		8307/P	8310/P	8316	8317/P	8320/P	8326	8361/P
		8368/P	8370/P	8460/M				
xcb_p	7260	7266	7266	7267	7267			
xcb_p	7775	7795	7795	7795	7795			
xcb_p	7786	7789/M	7794	7795/P	7796	7803	7805/M	7811/M
		7813/M	7814/M					
xcb_p	8241	8385	8385	8385	8385			
xcb_p	8250	8382/P	8383	8384	8385/P	8390/P	8393/P	8403/P
xfde_p	6808	6814						
xfde_p	8072	8101						
xijl_ordinal	6994	7002/M	7005/M	7009				
xijl_ordinal	8072	8116/M	8116/M	8116	8132/M	8132/M	8132	
xijl_ordinal	8157	8182/M	8182/M	8182				
xijl_ordinal	8470	8499/M	8499/M	8499				
xp	4368	7796	7811/M	7812/M	7814/M	8106	8108	8109
		8306	8316	8326	8384			
xpfti	7239	7254/M						
xpfti	7820	7925/M						
xsfid	6995	6997/M	6998/M	7001/M	7004/M	7008		
xsfid	8072	8116/M	8116/M	8116/M	8116/M	8116	8132/M	8132/M
		8132/M	8132					
xsfid	8157	8182/M	8182/M	8182/M	8182/M	8182		
xsfid	8470	8499/M	8499/M	8499/M	8499/M	8499		

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


```
            2 MODULE mmm$page_fault_processor;
            3 {
            4 {  PURPOSE: Memory_Manager
            5 {      This module contains the monitor routines that are used to
            6 {      manage physical memory and the page table.
            7 {
            8 {
            9
      o  6892
      o  6893 { This decks defines compile-time constants to control conditional compilation
      o  6894 { of debug code in memory manager modules in monitor mode. All constants should
      o  6895 { be set to FALSE for the transmitted version of this deck.
      o  6896
      o  6897   CONST
      o  6898     mmc$debug = TRUE;
      o  6899
      o  6900
      o  6901   ?VAR
      o  6902      mmc$debug_check_queues: boolean := TRUE, {Check PQL linkage}
      o  6903      mmc$debug_relink_swapping_job: boolean := TRUE , {Stop if relink page of swapping job}
      o  6904      mmc$debug_pt: boolean := TRUE, {check PT - AST linkage}
      o  6905      mmc$debug_free_asid: boolean := TRUE, {Verify no attempt is made to free an AST
      o  6906                                             { entry with pages not in AVAIL queue.
      o  6907      mmc$debug_aste_p_from_pfti: boolean := TRUE, {Verify aste_p in pfti is correct}
      o  6908      mmc$debug_rma_list: boolean := TRUE, {Verify RMA list on lock/unlock rma list}
      o  6909      mmc$debug_esc_alloc: boolean := TRUE, {check for escaped allocation}
      o  6910      mmc$debug_ast_pft: boolean := TRUE?; {check pft.aste_p on reference to AST}
      o  6911


      o  6913
      o  6914 {  PURPOSE: This deck (mmc$manage_memory_utility) primarily contains default values for the mmv$ variables
      o  6915 {    and the Global Page Queue List all of which are managed by the Manage Memory Utility which
      o  6916 {    is in the module mmm$manage_memory.
      o  6917
      o  6918     CONST
      o  6919       mmc$mmu_age_interval_ceiling    =       10,   { pages }
      o  6920       mmc$mmu_age_interval_floor      =        3,   { pages }
      o  6921       mmc$mmu_aggressive_aging_one    =       10,   { pages }
      o  6922       mmc$mmu_aggressive_aging_two    =       18,   { pages }
      o  6923       mmc$mmu_aging_algorithm         =        4,   { flag to select algorithm}
      o  6924       mmc$mmu_jws_age_interval        = 8000000,   { microseconds }
      o  6925       mmc$mmu_min_avail_pages         =      400,   { pages }
      o  6926       mmc$mmu_ps_prestream            =        4,   { page faults}
      o  6927       mmc$mmu_ps_transfer_size        =        0,   { bytes}{when non-zero, overrides transfer size from DM
      o  6928       mmc$mmu_ps_threshold            =    65536,   { bytes}
      o  6929       mmc$mmu_ps_reads                =        3,   { transfer units
      o  6930       mmc$mmu_ps_random_limit         =        3,   { random page faults
      o  6931       mmc$mmu_periodic_call_interval = 1000000,   { microseconds }
      o  6932       mmc$mmu_shared_age_interval     = 8000000,   { microseconds }
      o  6933       mmc$mmu_swapping_aic            =        1,   { limit on number of times an unused page is swapped out}
      o  6934       mmc$mmu_tick_time               =   100000,   { microseconds,  Real default is determined in deadstart.
      o  6935       mmc$mmu_queue_age_task_service  =        3,   { number of shared_age_intervals}
      o  6936       mmc$mmu_queue_age_pf_execute    =        1,   { number of shared_age_intervals}
      o  6937       mmc$mmu_queue_age_pf_non_exec   =        1,   { number of shared_age_intervals}
      o  6938       mmc$mmu_queue_age_device_file   =        1,   { number of shared_age_intervals}
      o  6939       mmc$mmu_queue_age_file_server   =        1,   { number of shared_age_intervals}
```

```
 0  6940       mmc$mmu_queue_age_other        :       1,   { number of shared_age_intervals}
 0  6941       mmc$mmu_queue_age_site_queues  :       1,   { number of shared_age_intervals}
 0  6942       mmc$mmu_queue_maximum          :       osc$max_page_frames,  {default maximum pages for all queues}
 0  6943       mmc$mmu_queue_minimum          :       0;     {default minimum pages for all queues}
 0  6944
 0  6945
 0  6946


 0  6948
 0  6949 {External procedures used by this module.
 0  6950
 0  6951
 0  6969   PROCEDURE [INLINE] display_monitor
 0  6970     (    display_line: string ( * <= 255));
 0  6971
 0  6972     dpp$display_error (display_line);
 0  6973   PROCEND display_monitor;




 0  6975
 0  6976   PROCEDURE display_integer_monitor
 4  6977     (    desc: string ( * <= 60);
 4  6978          int: integer);
 4  6979
 4  6980     VAR
 4  6981       hex_digits: [READ] array [0 .. 15] of char := ['0', '1', '2', '3', '4',
 4  6982            '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'];
 4  6983
 4  6984     TYPE
 4  6985       convert = record
 4  6986         case (integ, strng) of
 4  6987         = integ =
 4  6988           int: integer,
 4  6989         = strng =
 4  6990           strn: string (8),
 4  6991         casend,
 4  6992       recend;
 4  6993
 4  6994     VAR
 4  6995       conv: convert,
 4  6996       data: string (8),
 4  6997       data_index: integer,
 4  6998       line: string (80),
 4  6999       line_index: integer;
 4  7000
 4  7001     line := desc;
22  7002     line_index := #SIZE (desc) + 2;
22  7003     conv.int := int;
22  7004     data := conv.strn;
22  7005     FOR data_index := 1 TO 8 DO
3E  7006       line (line_index) := hex_digits [$INTEGER (data (data_index)) DIV 16];
3E  7007       line (line_index + 1) := hex_digits [$INTEGER (data (data_index)) MOD 16];
3E  7008       line_index := line_index + 2;
```



```
3E  7009     FOREND;
6E  7010     display_monitor (line (1, (line_index - 1)));
B4  7011   PROCEND display_integer_monitor;
    7012
    7013
    7014
    7015   PROCEDURE [XREF] dfp$fetch_multi_page_status
    7016     (    fde_p: gft$locked_file_desc_entry_p;
    7017          offset: ost$segment_offset;
    7018          length: ost$segment_length;
    7019       VAR allocate_status: gft$page_status);
    7020
 0  7208
 0  7209   PROCEDURE [XREF] dfp$fetch_page_status
 0  7210     (    fde_p: gft$locked_file_desc_entry_p;
 0  7211          offset: ost$segment_offset;
 0  7212       VAR allocate_status: gft$page_status);
 0  7213
    7216
    7217   PROCEDURE [XREF] dfp$file_server_allocation
    7218     (    sfid: gft$system_file_identifier;
    7219          segment_offset: ost$segment_offset;
    7220          segment_length: ost$segment_length;
    7221          io_id: mmt$io_identifier;
    7222          buffer_descriptor: mmt$buffer_descriptor;
    7223          file_limit: sft$file_space_limit_kind;
    7224       VAR spio_status: syt$monitor_status);
    7225
 0  7228 { Xref deck for dfp$server_io.
 0  7229
 0  7230   PROCEDURE [XREF] dfp$server_io (
 0  7231     fde_p: gft$locked_file_desc_entry_p;
 0  7232     io_type: iot$io_function;
 0  7233     segment_offset: ost$segment_offset;
 0  7234     segment_length: ost$segment_length;
 0  7235     io_id: mmt$io_identifier;
 0  7236     buffer_descriptor: mmt$buffer_descriptor;
 0  7237     VAR status: syt$monitor_status);
 0  7238
    7241
    7242
    7243   PROCEDURE [XREF] dmp$fetch_multi_page_status
    7244     (    fde_p: gft$locked_file_desc_entry_p;
    7245          offset: ost$segment_offset;
    7246          length: ost$segment_length;
    7247          enforce_limits: sft$file_space_limit_kind;
    7248       VAR reject_offset: ost$segment_offset;
    7249       VAR allocate_status: gft$page_status);
    7250
 0  7253
 0  7254   PROCEDURE [XREF] dmp$fetch_page_status
 0  7255     (    fde_p: gft$locked_file_desc_entry_p;
 0  7256          offset: ost$segment_offset;
 0  7257          enforce_limits: sft$file_space_limit_kind;
 0  7258          allow_allocation: boolean;
 0  7259       VAR allocate_status: gft$page_status);
 0  7260
```

```
  7263
  7264   PROCEDURE [INLINE] gfp$mtr_get_sfid_from_fde_p
  7265     (     fde_p: gft$locked_file_desc_entry_p;
  7266       VAR sfid: gft$system_file_identifier;
  7267       VAR ijl_ordinal: jmt$ijl_ordinal);
o 7324
o 7325
o 7326   PROCEDURE [INLINE] gfp$mtr_get_fde_p (sfid: gft$system_file_identifier;
o 7327         ijle_p: ^jmt$initiated_job_list_entry;
o 7328     VAR fde_p: gft$file_desc_entry_p);
o 7329
o 7330
o 7377
o 7378   PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
o 7379         ijle_p: ^jmt$initiated_job_list_entry;
o 7380     VAR fde_p: gft$locked_file_desc_entry_p);
o 7381
o 7382
o 7443
o 7444   PROCEDURE [INLINE] gfp$mtr_unlock_fde_p (fde_p: gft$locked_file_desc_entry_p);
o 7445
o 7446
o 7476
o 7477   PROCEDURE [XREF] i#real_memory_address (p: ^cell;
o 7478     VAR rma: integer);
o 7479
o 7480
o 7481   PROCEDURE [XREF] iop$enable_all_disk_units
o 7482     (VAR status: syt$monitor_status);
o 7483
  7486
  7487   PROCEDURE [XREF] iop$pager_io (
  7488     fde_p: gft$locked_file_desc_entry_p;
  7489     chapter_offset: ost$segment_offset;
  7490     buffer_descriptor: mmt$buffer_descriptor;
  7491     length: ost$byte_count;
  7492     io_function: iot$io_function;
  7493     io_identifier: mmt$io_identifier;
  7494     VAR status: syt$monitor_status);
o 7497
o 7498   PROCEDURE [INLINE] jmp$check_scheduler_memory_wait;
o 7499
  7532   PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
  7533     VAR ijle_p: ^jmt$initiated_job_list_entry);
  7534
o 7542   PROCEDURE [XREF] jmp$recognize_thrashing;
o 7543
o 7544
o 7545   PROCEDURE [XREF] jmp$set_scheduler_event (event: jmt$job_scheduler_events);
o 7546
  7549
  7550   PROCEDURE [INLINE] jmp$unlock_ajl
  7551     (     ijle_p: ^jmt$initiated_job_list_entry);
  7552
  7553     VAR
  7554       ajlo: jmt$ajl_ordinal;
  7555
```



```
  7556     tmp$set_lock (tmv$ptl_lock);
  7557     ajlo := ijle_p^.ajl_ordinal;
  7558     IF (jmv$ajl_p^ [ajlo].in_use = jmc$lock_ajl) THEN
  7559       jmp$free_ajl_with_lock (ijle_p, jmc$lock_ajl);
  7560     ELSE
  7561       jmv$ajl_p^ [ajlo].in_use := jmv$ajl_p^ [ajlo].in_use - jmc$lock_ajl;
  7562     IFEND;
  7563     tmp$clear_lock (tmv$ptl_lock);
  7564
  7565   PROCEND jmp$unlock_ajl;
  7566
o 7645
o 7646    PROCEDURE [XREF] jsp$free_swapped_jobs_memory
o 7647      (    ijl_ordinal: jmt$ijl_ordinal);
o 7648
  7651
  7652   PROCEDURE [XREF] mmp$assign_asid (VAR asid: ost$asid;
  7653     VAR asti: mmt$ast_index;
  7654     VAR aste_p: ^mmt$active_segment_table_entry);
  7655
o 7658
o 7659   PROCEDURE [INLINE] mmp$check_queues;
o 7660
o 7661     ?IF mmc$debug_check_queues THEN
o 7662       IF mmv$check_queues > 0 THEN
o 7663         mmp$xcheck_queues;
o 7664       IFEND;
o 7665     ?IFEND;
o 7666
o 7667   PROCEND;
o 7668
o 7669
o 7670   PROCEDURE [XREF] mmp$delete_pt_entry
o 7671     (     pfti: mmt$page_frame_index;
o 7672           unlink_page_from_segment: boolean);
o 7673
  7676
  7677   PROCEDURE [INLINE] mmp$fetch_pfti_array_size
  7678     (VAR pfti_size: integer);
  7679
o 7685
o 7686   PROCEDURE [INLINE] mmp$find_next_pfti
o 7687     (VAR xpfti: mmt$page_frame_index);
o 7688
  7706
  7707   PROCEDURE [XREF] mmp$free_asid (asid: ost$asid;
  7708         aste_p: ^mmt$active_segment_table_entry);
  7709
o 7712
o 7713
o 7714    PROCEDURE [INLINE] mmp$get_inhibit_io_status
o 7715      (     ijl_ordinal: jmt$ijl_ordinal;
o 7716            lock: boolean;
o 7717        VAR inhibit_io: boolean;
o 7718        VAR ijle_p: ^jmt$initiated_job_list_entry);
o 7719
o 7720     VAR
```

```
o  7721       ajlo: jmt$ajl_ordinal;
o  7722
o  7723      jmp$get_ijle_p (ijl_ordinal, ijle_p);
o  7724      inhibit_io := (ijle_p^.swap_status > jmc$inhibit_memory_manager_io);
o  7725      IF NOT inhibit_io THEN
o  7726        IF lock THEN
o  7727          tmp$set_lock (tmv$ptl_lock);
o  7728          jmp$lock_ajl_with_lock (ijle_p, ijl_ordinal, ajlo);
o  7729          tmp$clear_lock (tmv$ptl_lock);
o  7730        IFEND;
o  7731      IFEND;
o  7732
o  7733    PROCEND mmp$get_inhibit_io_status;
o  7734
   7779  PROCEDURE [XREF] mmp$initialize_find_next_pfti (xsva: ost$system_virtual_address;
   7780        length: ost$segment_length;
   7781        end_point_option: (include_partial_pages, exclude_partial_pages);
   7782        page_selection_criteria: mmt$page_selection_criteria;
   7783        aste_p: ^mmt$active_segment_table_entry;
   7784    VAR xpfti: mmt$page_frame_index);
   7785
o  7793
o  7794   PROCEDURE [XREF] mmp$make_pt_entry (sva: ost$system_virtual_address;
o  7795         pfti: mmt$page_frame_index;
o  7796         aste_p: ^mmt$active_segment_table_entry;
o  7797         pfte_p: ^mmt$page_frame_table_entry;
o  7798     VAR mpt_status: mmt$make_pt_entry_status);
o  7799
   7802
   7803    FUNCTION [INLINE] mmp$get_sdt_entry_p
   7804      (     xcb_p: ^ost$execution_control_block;
   7805            segnum: ost$segment): ^mmt$segment_descriptor;
   7806
   7807      mmp$get_sdt_entry_p := #address (1, #segment (xcb_p),
   7808        8 * segnum + xcb_p^.sdt_offset);
   7809
   7810    FUNCEND;
   7811
o  7814
o  7815    FUNCTION [INLINE] mmp$get_sdtx_entry_p
o  7816      (     xcb_p: ^ost$execution_control_block;
o  7817            segnum: ost$segment): ^mmt$segment_descriptor_extended;
o  7818
o  7819      mmp$get_sdtx_entry_p := #address (1, #segment (xcb_p),
o  7820        #SIZE (mmt$segment_descriptor_extended) * segnum + xcb_p^.sdtx_offset);
o  7821
o  7822    FUNCEND;
o  7823
   7826
   7827
   7828 { This procedure verifies that the asti stored in the file descriptor entry is still being used by
   7829 { the same job for the same file.  If the asti is ok, it is returned; otherwise 0 is returned.
   7830
   7831   PROCEDURE [INLINE] mmp$get_verify_asti_in_fde
   7832     (     fde_p: gft$locked_file_desc_entry_p;
   7833           sfid: gft$system_file_identifier;
   7834           ijlo: jmt$ijl_ordinal;
```



```
   7835       VAR asti: mmt$ast_index);
   7836
o  7848
o  7849 {-------------------------------------------------------------------------------------------------------
o  7850 {  The procedure mmp$page_pull_hash_sva is called to hash for a page in the page table.  If the page is in
o  7851 {  one of the available queues, OR if it is locked for IO, OR if it is a valid page in the page table,
o  7852 {  page_count will be set to 1, pstatus will be set, and pfti will be set.  If it is an available page, the
o  7853 {  page will also be relinked into the appropriate queue.
o  7854 {  If the page is not found, page_count will be set to zero and pstatus will be set to ps_done.
o  7855 {-------------------------------------------------------------------------------------------------------
o  7856
o  7857   PROCEDURE [inline] mmp$page_pull_hash_sva
o  7858     (     sva: ost$system_virtual_address;
o  7859           aste_p: ^mmt$active_segment_table_entry;
o  7860       VAR page_count: mmt$page_frame_index;
o  7861       VAR pstatus: mmt$page_pull_status;
o  7862       VAR pfti: mmt$page_frame_index);
   7918
   7919   PROCEDURE [XREF] mmp$preset_real_memory (sva: ost$system_virtual_address;
   7920         preset_value: pmt$initialization_value);
   7921
o  7924
o  7925   PROCEDURE [INLINE] mmp$purge_all_cache_map;
o  7926
o  7927     VAR
o  7928       null_sva: 0 .. 0ffffffffffff(16);
o  7929
o  7930     IF mmv$multiple_caches OR mmv$multiple_page_maps THEN
o  7931       mmp$purge_all_cache_map_proc;
o  7932     ELSE
o  7933       #purge_buffer (osc$purge_all_cache, null_sva);
o  7934       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
o  7935     IFEND;
o  7936
o  7937   PROCEND;
   7942   PROCEDURE [XREF] mmp$purge_all_cache_proc;
   7943
   7944   PROCEDURE [XREF] mmp$purge_all_map_proc;
   7945
   7946   PROCEDURE [XREF] mmp$remove_pages_working_set (sva: ost$system_virtual_address;
   7947         length: ost$segment_length;
   7948         aste_p: ^mmt$active_segment_table_entry;
   7949     VAR count: integer);
   7950
o  7953
o  7954   PROCEDURE [INLINE] mmp$reset_find_next_pfti
o  7955     (VAR xpfti: mmt$page_frame_index);
o  7956
   7979 {      mmp$reset_store_next_pfti contains inline proc named mmp$reset_store_pfti
   7980
   7981   PROCEDURE [INLINE] mmp$reset_store_pfti;
   7982
o  7989
o  7990   PROCEDURE [INLINE] mmp$reset_store_pfti_reverse;
o  7991
   7998
   7999   PROCEDURE [INLINE] mmp$set_include_pages_in_dump
```

```
8000      (     segment_number: ost$segment;
8001            fde_p: gft$locked_file_desc_entry_p;
8002            sdt_p: ^mmt$segment_descriptor;
8003       VAR include_pages_in_dump: boolean);
8004
8005      IF (sdt_p^.ste.wp <> osc$non_writable) THEN
8006        IF (segment_number < mmc$first_loader_predefined_seg) OR
8007             (fde_p^.stack_for_ring <> 0) OR
8008             (fde_p^.flags.global_template_file) OR
8009             (sdt_p^.ste.r1 <= 2) THEN
8010          include_pages_in_dump := TRUE;
8011        ELSE
8012          include_pages_in_dump := FALSE;
8013        IFEND;
8014      ELSE
8015        include_pages_in_dump := FALSE;
8016      IFEND;
8017
8018    PROCEND mmp$set_include_pages_in_dump;
8019
o 8022 {      mmp$store_next_pfti contains  inline proc named mmp$store_pfti
o 8023
o 8024    PROCEDURE [INLINE] mmp$store_pfti (pfti: mmt$page_frame_index);
o 8025
8032
8033    PROCEDURE [INLINE] mmp$store_pfti_reverse (pfti: mmt$page_frame_index);
8034
o 8041    PROCEDURE [INLINE] mmp$sva_purge_all_cache (sva: ost$system_virtual_address);
o 8042
o 8043      IF mmv$multiple_caches THEN
o 8044        mmp$purge_all_cache_proc;
o 8045      ELSE
o 8046        #purge_buffer (osc$sva_purge_all_cache, sva);
o 8047      IFEND;
o 8048
o 8049    PROCEND;
8052    PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
o 8063
o 8064    PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
o 8065          condition: osc$max_status_condition_number + 1 .. 0ffffffffff(16);
o 8066      VAR status: syt$monitor_status);
o 8067
8077
8078 {  PURPOSE:     procedure mtp$step_unstep_system
8079 {     Writes a line of text to the Top Line of the console.  If the idle code specified indicates an error
8080 {     (i.e. not an operator command or a recovery), then 'ERR=' must be first 4 characters or it will be
8081 {     prefixed.  Then the system is Terminated, Aborted, Stepped, or Idled per the idle code specified.
8082 {     All messages sent to the Top Line must have a VEOSxxxx code in the text.  Guidelines for using
8083 {     VEOSxxxx codes and a list of all VEOSxxxx codes are documented in the comments found in the proc
8084 {     dpp$display_error which is in the deck dpm$system_console_monitor.
8085
8086    PROCEDURE [XREF] mtp$step_unstep_system
8087          (term_code: syt$180_idle_code; text: string(*<=76) );
8088
o 8091
o 8092    PROCEDURE [XREF] osp$process_keypoint_page_fault
o 8093      (     utp_offset: integer;
```



```
o 8094       VAR keypoint_page_fault_status: mmt$keypoint_page_fault_status);
o 8095
8098    PROCEDURE [XREF] tmp$cause_task_switch;
8099
8100
8101    PROCEDURE [XREF] tmp$dequeue_task (VAR queue_link: tmt$task_queue_link;
8102      VAR taskid: ost$global_task_id);
8103
o 8106
o 8107    PROCEDURE [XREF] tmp$find_next_xcb (search: tmt$fnx_search_type;
o 8108          ijle_p: ^jmt$initiated_job_list_entry;
o 8109          ijl_ordinal: jmt$ijl_ordinal;
o 8110      VAR state: tmt$find_next_xcb_state;
o 8111      VAR xcb_p: ^ost$execution_control_block);
o 8112
8115
8116    PROCEDURE [XREF] tmp$get_top_of_stack (taskid: ost$global_task_id;
8117          ring: ost$valid_ring;
8118      VAR top_of_stack: integer);
8119
o 8122    PROCEDURE [XREF] tmp$get_xcb_p (task_id: ost$global_task_id;
o 8123      VAR xcb_p: ^ost$execution_control_block;
o 8124      VAR ijle_p: ^jmt$initiated_job_list_entry);
o 8125
8128
8129    PROCEDURE [XREF] tmp$monitor_flag_job_tasks
8130      (     monitor_flag_id: syt$monitor_flag;
8131            ijle_p: ^jmt$initiated_job_list_entry);
8132
8133
o 8136
o 8137    PROCEDURE [XREF] tmp$queue_task (taskid: ost$global_task_id;
o 8138          task_status: tmt$task_status;
o 8139      VAR queue_link: tmt$task_queue_link);
o 8140
8143
8144    PROCEDURE [XREF] tmp$reissue_monitor_request;
8145
8146
8147    PROCEDURE [XREF] tmp$send_monitor_fault (task_id [input] : ost$global_task_id;
8148          monitor_fault_p [input] : ^ost$monitor_fault;
8149          check_traps_enabled [input] : BOOLEAN);
8150
o 8153
o 8154    PROCEDURE [XREF] tmp$set_monitor_flag (task_id: ost$global_task_id;
o 8155          flag_id: syt$monitor_flag;
o 8156      VAR status: syt$monitor_status);
8159
8160    PROCEDURE [XREF] tmp$test_get_xcb_p (task_id: ost$global_task_id;
8161      VAR xcb_p: ^ost$execution_control_block;
8162      VAR ijle_p: ^jmt$initiated_job_list_entry);
8163
```

Global Variable Declarations - XREF and XDCL

```
o  8168
o  8169 {-------------------------------------------------------------------------------------------------------
o  8170


o  8172   VAR
o  8173     dfv$file_server_debug_enabled: [XREF] boolean;
o  8174


o  8176
o  8177   VAR
o  8178     gfv$null_sfid: [XREF, READ, OSS$MAINFRAME_WIRED_LITERAL] gft$system_file_identifier;
o  8179
o  8180
o  8181   SECTION
o  8182     oss$mainframe_wired_literal: READ;



o  8185 [Define pointer to Initiated Job List (IJL).
o  8186
o  8187   VAR
o  8188     jmv$ijl_p: [XREF]  jmt$ijl_p;



   8218   VAR
   8219     jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
   8220


o  8224
o  8225   VAR
o  8226     jsv$free_working_set_on_swapout: [XREF] boolean;
o  8227


o  8229  VAR
o  8230     jsv$ijl_swap_queue_list: [XREF] jst$ijl_swap_queue_list;
o  8231


   8250
   8251   VAR
   8252     jsv$max_pages_first_swap_task: [XREF] integer;


   8254
   8255    VAR
   8256      jsv$maximum_pages_to_swap: [XREF] integer;

   8258 [The following variable defines the agining algorithm that is used by memory manager.
```


Global Variable Declarations - XREF and XDCL

```
   8259 {     0 - no swapping active
   8260 {     1 - swapping active
   8261 {   > 1 - to be defined
   8262   VAR
   8263     mmv$aging_algorithm: [XREF] integer;


   8265
   8266   VAR
   8267     mmv$image_file: [XREF] mmt$image_file;
   8268


o  8282 [This variable defines the rate at which memory manager scans all jobs and
o  8283 [ages the working sets of any CP bound job that is found.
o  8284
o  8285   VAR
o  8286     mmv$jws_queue_age_interval: [XREF] integer;
o  8287


o  8289
o  8290 [  Define a variable to contain the index of the last shared site queue that is actually being used.
o  8291
o  8292   VAR
o  8293     mmv$last_active_shared_queue: [XREF] mmt$global_page_queue_index;
   8296


   8298
   8299   VAR
   8300     mmv$pages_to_dump_p: [XREF] ^packed array [0 .. *] of boolean;

   8302 [Define pointer to array for holding PFTI lists. This array is used in monitor for holding lists
   8303 [PFTIs of pages belonging to a segment.
   8304
   8305   VAR
   8306     mmv$pfti_array_p: [XREF] ^mmt$pfti_array;
   8307
   8309
   8310
   8311   TYPE
   8312     mmt$pfti_array = RECORD
   8313       pfti_first: 0 .. osc$max_page_frames,
   8314       pfti_index: 0 .. osc$max_page_frames,
   8315       last_pfti_index: 0 .. osc$max_page_frames,
   8316       pftis: ARRAY [0 .. *] of mmt$page_frame_index,
   8317     RECEND;



   8321
   8322   VAR
   8323     mmv$preset_conversion_table: [XREF, READ] array [pmt$initialization_value]
```

Global Variable Declarations - XREF and XDCL

```
          8324          of integer;
          8325
          8327 {Time for next periodic call to Memory Manager from CP Monitor.
          8328
          8329    VAR
          8330      mmv$time_to_call_mem_mgr: [XREF] integer;
          8332 {Monitor segment table.}
          8333
          8334    VAR
          8335      mtv$monitor_segment_table: [XREF] record
          8336        st: ALIGNED [0 MOD 8] array [0 .. 4095] of mmt$segment_descriptor,
          8337      recend;
          8338
        o 8342 VAR
        o 8343     mtv$sys_core_init_complete: [XREF] boolean;
        o 8345 {This variable specifies the lower and upper RMA addresses available to NOSVE.
        o 8346
        o 8347    VAR
        o 8348      osv$180_memory_limits: [XREF] record
        o 8349        lower: 0 .. 0ffffffff(16),
        o 8350        deadstart_upper: 0 .. 0ffffffff(16),  {  Upper limit of memory during deadstart.
        o 8351        upper: 0 .. 0ffffffff(16),
        o 8352      recend;
        o 8353
        o 8355    VAR
        o 8356      osv$time_to_check_asyn: [XREF] integer;
        o 8357
        o 8359
        o 8360    VAR
        o 8361      tmv$null_global_task_id: [READ] ost$global_task_id := [0, 0];
        o 8362
        o 8364
        o 8365    VAR
        o 8366      tmv$ptl_p: [XREF] ^tmt$primary_task_list;
        o 8367
        o 8418
        o 8419 {Define AST template for job swapper to use to create a new entry for a Job Fixed segment.
        o 8420
        o 8421    VAR
        o 8422      mmv$initial_job_fixed_ast_entry: [XDCL, READ] mmt$active_segment_table_entry :=
```


Global Variable Declarations - XREF and XDCL

```
        o 8423              [[0, 0], 0, [0, 0], TRUE, mmc$pq_job_fixed, * , TRUE];
        o 8424
        o 8425 { Define array for keeping statistics on status values returned from MMP$WRITE_PAGE_TO_DISK.
        o 8426
        o 8427    VAR
        o 8428      mmv$write_page_statistics: [XDCL] array [mmt$write_page_to_disk_status] of integer :=
        o 8429            [0, 0, 0, 0, 0, 0, 0];
        o 8430
        o 8432
        o 8433 {  The following variables + the Global Page Queue List below are all managed by the Manage Memory Utility.
        o 8434 {       mmv$age_interval_ceiling
        o 8435 {       mmv$age_interval_floor
        o 8436 {       mmv$aggressive_aging_level
        o 8437 {       mmv$aggressive_aging_level_2
        o 8438 {       mmv$aging_algorithm
        o 8439 {       mmv$jws_queue_age_interval
        o 8440 {       mmv$min_avail_pages
        o 8441 {       mmv$page_streaming_prestream
        o 8442 {       mmv$page_streaming_threshold
        o 8443 {       mmv$page_streaming_reads
        o 8444 {       mmv$page_streaming_random_limit
        o 8445 {       mmv$periodic_call_interval
        o 8446 {       mmv$tick_time
        o 8447 {       mmv$shared_queue_age_interval
        o 8448 {       mmv$swapping_aic
        o 8449 {
        o 8450 {  Define array for the Global Page Queue List and initialize it to the default values as defined in the
        o 8451 {  common deck mmc$manage_memory_utility.
        o 8452 {
        o 8453
        o 8454
        o 8455 ?? FMT (FORMAT := OFF) ??
        o 8456    VAR
        o 8457      mmv$gpql: [XDCL, #GATE] mmt$global_page_queue_list :=
        o 8458            [ [[[0,0],0],0,0,0],                {free }
        o 8459              [[[0,0],0],0,0,0],                {available}
        o 8460              [[[0,0],0],0,0,0],                {available modified}
        o 8461              [[[0,0],0],0,0,0],                {wired}
        o 8462              [[[0,0],0],mmc$mmu_queue_age_task_service,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8463              [[[0,0],0],mmc$mmu_queue_age_pf_execute,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8464              [[[0,0],0],mmc$mmu_queue_age_pf_non_exec,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8465              [[[0,0],0],mmc$mmu_queue_age_device_file,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8466              [[[0,0],0],mmc$mmu_queue_age_file_server,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8467              [[[0,0],0],mmc$mmu_queue_age_other,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum],
        o 8468              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_01}
        o 8469              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_02}
        o 8470              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_03}
        o 8471              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_04}
        o 8472              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_05}
        o 8473              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_06}
        o 8474              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_07}
        o 8475              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_08}
        o 8476              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_09}
        o 8477              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_10}
```

Global Variable Declarations - XREF and XDCL

```
o  8478              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_11}
o  8479              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_12}
o  8480              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_13}
o  8481              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_14}
o  8482              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_15}
o  8483              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_16}
o  8484              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_17}
o  8485              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_18}
o  8486              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_19}
o  8487              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_20}
o  8488              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_21}
o  8489              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_22}
o  8490              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_23}
o  8491              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_24}
o  8492              [[[0,0],0],mmc$mmu_queue_age_site_queues,mmc$mmu_queue_minimum,mmc$mmu_queue_maximum], {site_25}
o  8493              [[[0,0],0],1,0,0],                      {shared_io_error}
o  8494              [[[0,0],0],1,0,0]   ];                  {swapped_io_error}
o  8495 ?? FMT (FORMAT := ON) ??
o  8496


o  8498
o  8499    VAR
o  8500      null_pva: 0 .. 0ffffffffffff(16),
o  8501      null_sva: [STATIC] ost$system_virtual_address := [0, 0],
o  8502      total_contig_pages_assigned: integer := 0,
o  8503
o  8504      dfv$max_bytes_to_write: [XDCL] integer := 16384 * 8, {!!! temp - move to DF}
o  8505      mmv$advise_in_aio_limit: [XDCL, #GATE] integer := 24,
o  8506      mmv$age_interval_ceiling: [XDCL, #GATE] 0 .. 255 := mmc$mmu_age_interval_ceiling,
o  8507      mmv$age_interval_floor: [XDCL, #GATE] 0 .. 255 := mmc$mmu_age_interval_floor,
o  8508      mmv$aggressive_aging_level: [XDCL, #GATE] integer := mmc$mmu_aggressive_aging_one,
o  8509      mmv$aggressive_aging_level_2: [XDCL, #GATE] integer := mmc$mmu_aggressive_aging_two,
o  8510      mmv$aging_statistics: [XDCL, #GATE] mmt$aging_statistics,
o  8511      mmv$aio_limit: [XDCL, #GATE] integer := 60000,
o  8512      mmv$aio_limit_count: [XDCL, #GATE] integer := 0,
o  8513      mmv$assign_contiguous_pass_cnt: mmt$assign_contig_passes := [0, 0, 0],
o  8514      mmv$assign_contig_reject: integer,
o  8515      mmv$assign_multiple_pages: [XDCL, #GATE] integer := 50,
o  8516      mmv$assign_pages_purge_count: integer := 0,
o  8517      mmv$ast_p: [XDCL, #GATE] ^mmt$active_segment_table := NIL,
o  8518      mmv$async_work: [XDCL] mmt$async_work_list := [FALSE, FALSE, [0, 0], NIL],
o  8519      mmv$contiguous_mem_length_max: [XDCL, #GATE] ost$segment_length := 65536,
o  8520      mmv$dm_flag_on_write: [XDCL] 0 .. 0ffffffff(16) := 1,
o  8521      mmv$file_allocation_interval: [XDCL, #GATE] integer := 20000,
o  8522      mmv$jmtr_escaped_allocate: [XDCL, #GATE] integer := 0,
o  8523      mmv$last_segment_accessed: [XDCL, #GATE] ost$segment,
o  8524      mmv$lost_escaped_allocate: [XDCL, #GATE] integer := 0,
o  8525      mmv$max_pages_no_file: [XDCL, #GATE] integer := 15, {This constant is forced negative during deadstart
o  8526      { to disable transient segments til Space Mgr runs.
o  8527      mmv$max_working_set_size: [XDCL, #GATE] integer := 1000,
o  8528      mmv$maxws_aio_count: [XDCL] integer := 0,
o  8529      mmv$maxws_aio_slowdown: [XDCL] integer := 60000000,
o  8530      mmv$maxws_aio_threshold: [XDCL, #GATE] integer := 10,
o  8531      mmv$memory_wait_queue: [XDCL] tmt$task_queue_link := [0, 0],
o  8532      mmv$min_avail_pages: [XDCL, #GATE] integer := mmc$mmu_min_avail_pages,
```


Global Variable Declarations - XREF and XDCL

```
o  8533      mmv$multiple_caches: [XDCL, #GATE] boolean := FALSE,
o  8534      mmv$multiple_page_maps: [XDCL, #GATE] boolean := FALSE,
o  8535      mmv$multi_page_write: [XDCL, #GATE] boolean := TRUE,
o  8536      mmv$no_memory_buffering: [XDCL, #GATE] boolean := FALSE,
o  8537      mmv$pages_per_new_page_fault: [XDCL, #GATE] 1 .. 8 := 1,
o  8538      mmv$page_streaming_prestream: [XDCL, #GATE] 0 .. 255 := mmc$mmu_ps_prestream,
o  8539      mmv$page_streaming_threshold: [XDCL, #GATE] integer := mmc$mmu_ps_threshold,
o  8540      mmv$page_streaming_transfer: [XDCL, #GATE] integer := mmc$mmu_ps_transfer_size,
o  8541      mmv$page_streaming_reads: [XDCL, #GATE] 0 .. 255 := mmc$mmu_ps_reads,
o  8542      mmv$page_streaming_random_limit: [XDCL, #GATE] 0 .. 255 := mmc$mmu_ps_random_limit,
o  8543      mmv$pages_for_overallocation: [XDCL, #GATE] integer := 16,
o  8544      mmv$paging_statistics: [XDCL, #GATE] mmt$paging_statistics,
o  8545      mmv$pf_statistics: [XDCL, #GATE] mmt$pf_statistics,
o  8546      mmv$pf_sva_array: [XDCL] record
o  8547        next_i: integer,
o  8548        pf_recs: array [0 .. num_pf_recs - 1] of packed record
o  8549          pstatus_time: 0 .. 0ffff(16),
o  8550          sva: ost$system_virtual_address,
o  8551        recend,
o  8552      recend,
o  8553      mmv$pft_p: [XDCL, #GATE] ^mmt$page_frame_table := NIL,
o  8554      mmv$pt_length: [XDCL, #GATE] integer,
o  8555      mmv$pt_p: [XDCL, #GATE] ^ost$page_table,
o  8556      mmv$reassignable_page_frames: [XDCL, #GATE] mmt$reassignable_page_frames,
o  8557      mmv$read_tu_execute: [XDCL, #GATE] 0 .. 0ffffffff(16) := 1,
o  8558      mmv$read_tu_read_write: [XDCL, #GATE] 0 .. 0ffffffff(16) := 1,
o  8559      mmv$refs_to_unrec_df_file_inhib: [XDCL, #GATE] integer := 0,
o  8560      mmv$refs_to_unrec_df_file_term: [XDCL, #GATE] integer := 0,
o  8561      mmv$reserved_page_count: [XDCL, #GATE] integer := 0,
o  8562
o  8563
o  8564 {  Define the number of free and available pages that job scheduler tries to keep
o  8565 {  available for all active jobs.
o  8566
o  8567      mmv$resident_job_target: [XDCL, #GATE] integer := 60,
o  8568
o  8569 {! Define the number of pages that memory manager and job swapper will try to
o  8570 {  keep available in 'now' + 'soon' reassignable memory.  If 'now' + 'soon'
o  8571 {  is less than or equal to this value IO is initiated on jobs in the long
o  8572 {  wait queue until this value is exceeded.
o  8573
o  8574      mmv$sdtx_entry_size: [XDCL, #GATE] integer := #SIZE (mmt$segment_descriptor_extended),
o  8575      mmv$shared_pages_in_jws: [XDCL, #GATE] boolean := TRUE,
o  8576      mmv$swapping_aic: [XDCL, #GATE] integer := mmc$mmu_swapping_aic,
o  8577      mmv$tables_initialized: [XDCL, #GATE] boolean := FALSE,
o  8578      mmv$test_reassign_asid: [XDCL] boolean := FALSE,
o  8579      mmv$tick_time: [XDCL, #GATE] integer := mmc$mmu_tick_time,
o  8580      mmv$volume_wait_queue: [XDCL] tmt$task_queue_link := [0, 0],
o  8581      mmv$write_aged_out_pages: [XDCL, #GATE] integer := 100000,
o  8582      osv$page_size: [XDCL, #GATE] ost$page_size,
o  8583      syv$perf_keypoints_enabled: [XDCL, #GATE] syt$perf_keypoints_enabled :=
o  8584          [FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE],
o  8585      syv$recovering_job_count: [XDCL, #GATE] integer := 0,
o  8586      syv$refs_to_unrecovered_seg: [XDCL] integer := 0;
o  8587
o  8588
```

Global Variable Declarations - XREF and XDCL

```
o  8589    CONST
o  8590      num_pf_recs = 256;
o  8591
```


MMP$PURGE_ALL_CACHE, MMP$PURGE_ALL_MAP - CACHE/MAP MANAGEMENT

```
o  8594    PROCEDURE [INLINE] mmp$aste_pointer_from_pfti (pfti: mmt$page_frame_index;
o  8595      VAR aste_p: ^mmt$active_segment_table_entry);
o  8596
o  8597      ? IF mmc$debug_aste_p_from_pfti THEN
o  8598          mmp$aste_pointer (mmv$pft_p^ [pfti].sva.asid, aste_p);
o  8599          IF aste_p <> mmv$pft_p^ [pfti].aste_p THEN
o  8600            mtp$error_stop ('MM - ERROR IN ASTE_POINTER_FROM_PFTI');
o  8601          IFEND;
o  8602      ? ELSE
   8603          aste_p := mmv$pft_p^ [pfti].aste_p;
o  8604      ? IFEND;
   8605
   8606    PROCEND mmp$aste_pointer_from_pfti;


   8608
   8609    PROCEDURE [INLINE] mmp$purge_all_page_map;
   8610
   8611      VAR
   8612        null_sva: 0 .. 0ffffffffffff(16);
   8613
   8614      IF mmv$multiple_page_maps THEN
   8615        mmp$purge_all_map_proc;
   8616      ELSE
   8617        #purge_buffer (osc$purge_all_page_seg_map, null_sva);
   8618      IFEND;
   8619
   8620    PROCEND;


o  8624    PROCEDURE [INLINE] mmp$sva_purge_one_page_map (sva: ost$system_virtual_address);
o  8625
o  8626      IF mmv$multiple_page_maps THEN
o  8627        mmp$purge_all_map_proc;
o  8628      ELSE
o  8629        #purge_buffer (osc$sva_purge_one_page_map, sva);
o  8630      IFEND;
o  8631
o  8632    PROCEND;


   8636    PROCEDURE [INLINE] mmp$sva_purge_all_page_map (sva: ost$system_virtual_address);
   8637
   8638      IF mmv$multiple_page_maps THEN
   8639        mmp$purge_all_map_proc;
   8640      ELSE
   8641        #purge_buffer (osc$sva_purge_all_page_map, sva);
   8642      IFEND;
   8643
   8644    PROCEND;
```

mmp$update_eoi

```
0  8648
0  8649 {
0  8650 { This procedure is called to update EOI (if necessary) after adding a page to a segment
0  8651 { or writing a page to disk.
0  8652 { If the beginning of the new page is beyond the current FDE EOI, then
0  8653 { the file EOI is set to the beginning of the next page.
0  8654 {
0  8655 {       OFFSET - must be the FIRST byte of the page assigned/written. If multiple pages
0  8656 {                are assigned, offset should the beginning of the page faulted for.
0  8657 {
0  8658
0  8659
0  8660   PROCEDURE [INLINE, XDCL] mmp$update_eoi(
4  8661     (     fde_p: gft$locked_file_desc_entry_p;
4  8662           offset: ost$segment_offset;
4  8663           reason: mmt$update_eoi_reason);
4  8664
4  8665     IF offset >= fde_p^.eoi_byte_address THEN
14 8666       fde_p^.eoi_byte_address := offset + osv$page_size;
14 8667       fde_p^.flags.eoi_modified := TRUE;
14 8668       IF reason = mmc$uer_multiple_pages_assigned THEN
36 8669         fde_p^.eoi_state := mmc$eoi_uncertain;
3E 8670       ELSE
3E 8671         fde_p^.eoi_state := mmc$eoi_rounded;
42 8672       IFEND;
44 8673     ELSEIF reason = mmc$uer_page_written THEN
4E 8674       IF (offset + osv$page_size) > fde_p^.eoi_byte_address THEN
5C 8675         IF fde_p^.eoi_state <> mmc$eoi_actual THEN
64 8676           fde_p^.flags.eoi_modified := TRUE;
64 8677           fde_p^.eoi_byte_address := offset + osv$page_size;
64 8678           fde_p^.eoi_state := mmc$eoi_rounded;
76 8679         IFEND;
76 8680       IFEND;
76 8681     IFEND;
76 8682
76 8683   PROCEND mmp$update_eoi;
```


MMP$ASID, MMP$AST_INDEX - Convert between ASID and AST INDEX

```
0  8686
0  8687 {-------------------------------------------------------------------------------------------------------------
0  8688 {Name:
0  8689 {     mmf$asid
0  8690 {     mmp$aste_pointer
0  8691 {Purpose:
0  8692 {  These functions convert AST indexes into an ASID and vise-versa.
0  8693 {Input:
0  8694 {     pfl_index or ASID
0  8695 {Output:
0  8696 {     asid or pfl_index
0  8697 {Error Codes:
0  8698 {     none
0  8699 {-------------------------------------------------------------------------------------------------------------
0  8700
0  8701   VAR
0  8702     mmv$a_mult: [XDCL, #GATE] 0 .. 10000(16),
0  8703     mmv$a_divisor: [XDCL, #GATE] 0 .. 10000(16),
0  8704     bits: array [0 .. 15] of 0 .. 15 := [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15];
0  8705
0  8706
0  8707   PROCEDURE [XDCL, INLINE] mmp$asid
4  8708     (     xasti: mmt$ast_index;
4  8709      VAR x_asid: ost$asid);
4  8710
4  8711     VAR
4  8712       asti: mmt$ast_index,
4  8713       asid: integer;
4  8714
4  8715
4  8716     asti := xasti;
4  8717     asid := (bits [asti MOD 16] * 4096) + (bits [(asti DIV 16) MOD 16] * 256) +
4  8718             (bits [(asti DIV 256) MOD 16] * 16) + bits [(asti DIV 4096) MOD 16];
4  8719     asid := asid DIV mmv$a_divisor + (asid MOD mmv$a_divisor) * mmv$a_mult;
4  8720     x_asid := asid;
4  8721
4  8722   PROCEND mmp$asid;
0  8723
0  8724
0  8725   PROCEDURE [XDCL, INLINE] mmp$aste_pointer
4  8726     (     xasid: ost$asid;
4  8727      VAR aste_p: ^mmt$active_segment_table_entry);
4  8728
4  8729
4  8730     VAR
4  8731       asid: ost$asid,
4  8732       asti: integer;
4  8733
4  8734
4  8735     asid := xasid;
4  8736     asid := asid DIV mmv$a_mult + (asid MOD mmv$a_mult) * mmv$a_divisor;
4  8737     asti := (bits [asid MOD 16] * 4096) + (bits [(asid DIV 16) MOD 16] * 256) +
4  8738             (bits [(asid DIV 256) MOD 16] * 16) + bits [(asid DIV 4096) MOD 16];
4  8739     aste_p := ^mmv$ast_p^ [asti];
4  8740     IF NOT aste_p^.in_use THEN
94 8741       aste_p := NIL;
```

MMP$ASID, MMP$AST_INDEX - Convert between ASID and AST INDEX

```
9C  8742      IFEND;
9C  8743
9C  8744    PROCEND mmp$aste_pointer;


 0  8746
 0  8747    PROCEDURE [XDCL, INLINE] mmp$asti
 4  8748      (     xasid: ost$asid;
 4  8749       VAR asti: mmt$ast_index);
 4  8750
 4  8751      VAR
 4  8752        asid: ost$asid;
 4  8753
 4  8754      asid := xasid;
 4  8755      asid := asid DIV mmv$a_mult + (asid MOD mmv$a_mult) * mmv$a_divisor;
 4  8756      asti := (bits [asid MOD 16] * 4096) + (bits [[asid DIV 16) MOD 16] * 256) +
 4  8757            (bits [(asid DIV 256) MOD 16] * 16) + bits [(asid DIV 4096) MOD 16];
 4  8758
 4  8759    PROCEND mmp$asti;
```


MMP$DETERMINE_SHARED_QUEUE_ID

```
 0  8761
 0  8762 {
 0  8763 { This function determines the shared queue id for a file whose pages are going
 0  8764 { to be kept in one of the shared queues.
 0  8765 {
 0  8766
 0  8767    FUNCTION [XDCL, INLINE] mmp$determine_shared_queue_id
 4  8768      (     fde_p: gft$locked_file_desc_entry_p;
 4  8769            ste_p: ^mmt$segment_descriptor): mmt$page_frame_queue_id;
 4  8770
 4  8771      IF fde_p^.file_kind = gfc$fk_job_permanent_file THEN
10  8772        IF (ste_p^.ste.xp = osc$non_executable) OR (ste_p^.ste.wp <> osc$non_writable) THEN
28  8773          mmp$determine_shared_queue_id := mmc$pq_shared_pf_non_execute;
2E  8774        ELSE
2E  8775          mmp$determine_shared_queue_id := mmc$pq_shared_pf_execute;
30  8776        IFEND;
34  8777      ELSEIF fde_p^.file_kind = gfc$fk_catalog THEN
3A  8778        mmp$determine_shared_queue_id := mmc$pq_shared_pf_non_execute;
42  8779      ELSEIF fde_p^.flags.global_template_file THEN
4C  8780        mmp$determine_shared_queue_id := mmc$pq_shared_task_service;
50  8781      ELSEIF fde_p^.file_kind = gfc$fk_device_file THEN
56  8782        mmp$determine_shared_queue_id := mmc$pq_shared_device_file;
5E  8783      ELSE
5E  8784        mmp$determine_shared_queue_id := mmc$pq_shared_other;
62  8785      IFEND;
62  8786
62  8787    FUNCEND mmp$determine_shared_queue_id;
```

NITIALIZE_NEW_AST_ENTRY

```
o  8789
o  8790 {-------------------------------------------------------------------------------------------
o  8791 {This routine is called to assign and initialize a new AST entry for a segment.
o  8792 {
o  8793 {-------------------------------------------------------------------------------------------
o  8794
o  8795   PROCEDURE [INLINE] initialize_new_ast_entry
o  8796     (     fde_p: gft$locked_file_desc_entry_p;
o  8797           segnum: ost$segment;
o  8798           ste_p: ^mmt$segment_descriptor;
o  8799           stxe_p: ^mmt$segment_descriptor_extended;
o  8800           cst_p: ^ost$cpu_state_table;
o  8801           force_to_global: boolean;
o  8802      VAR asid: ost$asid;
o  8803      VAR aste_p: ^mmt$active_segment_table_entry);
o  8804
o  8805     VAR
o  8806       asti: mmt$ast_index,
o  8807       queue_id: mmt$page_frame_queue_id;
o  8808
o  8809     mmp$assign_asid (asid, asti, aste_p);
o  8810
o  8811     fde_p^.asti := asti;
o  8812
o  8813     IF fde_p^.stack_for_ring = 0 THEN
o  8814       fde_p^.last_segment_number := segnum;
o  8815       fde_p^.global_task_id := cst_p^.taskid;
o  8816     IFEND;
o  8817
o  8818     aste_p^.ijl_ordinal := cst_p^.ijl_ordinal;
o  8819
o  8820     IF mmc$sa_wired IN stxe_p^.software_attribute_set THEN
o  8821       queue_id := mmc$pq_wired;
o  8822     ELSEIF mmc$sa_fixed IN stxe_p^.software_attribute_set THEN
o  8823       queue_id := mmc$pq_job_fixed;
o  8824     ELSEIF force_to_global THEN
o  8825       aste_p^.ijl_ordinal := jmv$system_ijl_ordinal;
o  8826       queue_id := mmp$determine_shared_queue_id (fde_p, ste_p);
o  8827     ELSE
o  8828       queue_id := mmc$pq_job_working_set;
o  8829     IFEND;
o  8830
o  8831     aste_p^.queue_id := queue_id;
o  8832     aste_p^.sfid := stxe_p^.sfid;
o  8833
o  8834     mmp$set_include_pages_in_dump (segnum, fde_p, ste_p, aste_p^.include_pages_in_dump);
o  8835
o  8836     ste_p^.ste.asid := asid;
o  8837     ste_p^.asti := asti;
o  8838
o  8839   PROCEND initialize_new_ast_entry;
```


SET_ASSIGN_ACTIVE

```
o  8841
o  8842 {-------------------------------------------------------------------------------------------
o  8843 {This routine is used to set the SDTX flags for a request that requires job mode work.
o  8844 {       SDTX.ASSIGN_ACTIVE  has the following values
o  8845 {             mmc$assign_active_null      - Null value
o  8846 {             mmc$assign_active_escaped   - Implies escaped allocation
o  8847 {              otherwise                  - Address that requires job mode work
o  8848 {-------------------------------------------------------------------------------------------
o  8849
o  8850   PROCEDURE [INLINE] set_assign_active
o  8851     (     stxe_p: ^mmt$segment_descriptor_extended;
o  8852           offset: ost$segment_offset);
o  8853
o  8854     IF stxe_p^.assign_active = mmc$assign_active_null THEN
o  8855       stxe_p^.assign_active := offset;
o  8856     ELSE
o  8857       stxe_p^.assign_active := mmc$assign_active_escaped;
o  8858     IFEND;
o  8859
o  8860   PROCEND set_assign_active;
```

MMP$CONVERT_PVA - Convert job mode PVA to SVA

```
0  8863
0  8864 {-------------------------------------------------------------------------------------------------------------
0  8865 {Name:
0  8866 {  mmp$convert_pva
0  8867 {Purpose:
0  8868 {  This routine converts a PVA relative to the CURRENT USER TASK
0  8869 {  to an SVA and returns pointers to the SDTX, AST entries for the segment.
0  8870 {-------------------------------------------------------------------------------------------------------------
0  8871
0  8872   PROCEDURE [XDCL] mmp$convert_pva
0  8873     (     p: ^cell;
0  8874           cst_p: ^ost$cpu_state_table;
0  8875       VAR xsva: ost$system_virtual_address;
0  8876       VAR fde_p: gft$locked_file_desc_entry_p;
0  8877       VAR aste_p: ^mmt$active_segment_table_entry;
0  8878       VAR ste_p: ^mmt$segment_descriptor;
0  8879       VAR stxe_p: ^mmt$segment_descriptor_extended);
0  8880
0  8881
0  8882     VAR
0  8883       asid: ost$asid,
0  8884       asti: mmt$ast_index,
0  8885       segnum: ost$segment,
0  8886       sva: ost$system_virtual_address, {use local var for performance
0  8887       force_to_global: boolean; {TRUE if determined page goes in global queue.}
0  8888
0  8889
0  8890     segnum := #SEGMENT (p);
4  8891
4  8892     ste_p := mmp$get_sdt_entry_p (cst_p^.xcb_p, segnum);
4  8893     stxe_p := mmp$get_sdtx_entry_p (cst_p^.xcb_p, segnum);
4  8894
4  8895     IF (segnum > cst_p^.xcb_p^.xp.segment_table_length) OR (ste_p^.ste.vl = osc$vl_invalid_entry) THEN
90 8896       mtp$error_stop ('MM - invalid PVA');
B0 8897     IFEND;
B0 8898
B0 8899
B0 8900 { The following trap code is used periodically to trap escaped allocation. Please do not delete this code.
B0 8901
B0 8902 {  IF mmc$debug THEN
B0 8903 {    IF (stxe_p^.assign_active <> mmc$assign_active_null) AND (cst_p^.xcb_p^.xp.p_register.pva.ring > 1) AND
B0 8904 {      (cst_p^.xcb_p^.xp.trap_enable = osc$traps_enabled) AND
B0 8905 {      NOT cst_p^.xcb_p^.stic_allocation AND
B0 8906 {      (osc$page_fault IN cst_p^.xcb_p^.xp.monitor_condition_register) AND
B0 8907 {      NOT (osc$trap_exception IN cst_p^.xcb_p^.xp.monitor_condition_register) THEN
B0 8908 {      mtp$error_stop ('PFP-CONVERT--Escaped Allocation.');
B0 8909 {    IFEND;
B0 8910 {  IFEND;
B0 8911
B0 8912 {! Does this really happen}
B0 8913     IF stxe_p^.sfid.residence = gfc$tr_system_wait_recovery THEN
BE 8914       syv$refs_to_unrecovered_seg := syv$refs_to_unrecovered_seg + 1;
BE 8915       aste_p := NIL;
BE 8916       RETURN;
DA 8917     IFEND;
DA 8918
```


MMP$CONVERT_PVA - Convert job mode PVA to SVA

```
DA 8919     sva.asid := ste_p^.ste.asid;
DA 8920     sva.offset := #OFFSET (p);
DA 8921
DA 8922     gfp$mtr_get_locked_fde_p (stxe_p^.sfid, cst_p^.ijle_p, fde_p);
180 8923
180 8924     IF sva.asid <> 0 THEN
188 8925       aste_p := ^mmv$ast_p^ [fde_p^.asti];
188 8926       IF mmc$debug AND ((fde_p^.asti <> ste_p^.asti) OR (aste_p^.sfid <> stxe_p^.sfid)) THEN
1D0 8927         mtp$error_stop ('MM - bad tables in CONVERT_PVA');
1F0 8928       IFEND;
1F4 8929     ELSE
1F4 8930       #PURGE_BUFFER (osc$purge_all_page_seg_map, null_pva); { only job mode segment map purge is required}
1FE 8931       force_to_global := (fde_p^.queue_status = gfc$qs_global_shared) OR (fde_p^.attach_count > 1);
21E 8932
21E 8933       mmp$get_verify_asti_in_fde (fde_p, stxe_p^.sfid, cst_p^.ijl_ordinal, asti);
27E 8934
27E 8935       IF (asti = 0) THEN
286 8936         initialize_new_ast_entry (fde_p, segnum, ste_p, stxe_p, cst_p, force_to_global, asid, aste_p);
3EC 8937       ELSE
3EC 8938         mmp$asid (asti, asid);
3EC 8939         ste_p^.ste.asid := asid;
3EC 8940         ste_p^.asti := asti;
3EC 8941
3EC 8942 { Determine which queue (JWS or Shared) this file should be in by looking at FORCE_TO_GLOBAL but
3EC 8943 { with one important exception.  If there are currently pages in memory and the current queue is not JWS,
3EC 8944 { then the file was recently global and there are still pages in memory--maybe in the global queue, maybe
3EC 8945 { in the available queue.  In this case we must leave the pages in the global queue, or we may end up with
3EC 8946 { modified pages in both the jws and global queues-this is a problem when it is time to write modified pages.
3EC 8947 { If the file belongs in a JWS queue but has pages in memory it's queue must not be changed.
3EC 8948 { Also, if it is currently in the JWS queue, the ijl_ordinal must be reset because it may point
3EC 8949 { to an old ijl.
3EC 8950
3EC 8951         aste_p := ^mmv$ast_p^ [asti];
3EC 8952         IF NOT force_to_global THEN
486 8953           IF (aste_p^.pages_in_memory = 0) OR (aste_p^.queue_id = mmc$pq_job_working_set) THEN
49A 8954             aste_p^.queue_id := mmc$pq_job_working_set;
49A 8955             aste_p^.ijl_ordinal := cst_p^.ijl_ordinal;
4AE 8956           IFEND;
4B2 8957         ELSEIF aste_p^.queue_id > mmc$pq_shared_last THEN
4BE 8958
4BE 8959 { The file belongs in the shared queue.  Make sure the AST entry is correct.
4BE 8960
4BE 8961           aste_p^.ijl_ordinal := jmv$system_ijl_ordinal;
4BE 8962           aste_p^.queue_id := mmp$determine_shared_queue_id (fde_p, ste_p);
526 8963         IFEND; {not force_to_global}
526 8964
526 8965       IFEND; {asti = 0}
526 8966       sva.asid := asid;
52E 8967     IFEND; {asid <> 0}
52E 8968
52E 8969     IF mmc$debug AND ((aste_p^.ijl_ordinal <> cst_p^.ijl_ordinal) AND
556 8970           (aste_p^.ijl_ordinal <> jmv$system_ijl_ordinal)) THEN
556 8971       mtp$error_stop ('MM - Bad IJLO in CONVERT_PVA');
576 8972     IFEND;
576 8973
576 8974     xsva := sva;
```

MMP$CONVERT_PVA - Convert job mode PVA to SVA

```
576  8975
576  8976    PROCEND mmp$convert_pva;
```


MMP$VERIFY_PVA - Test job mode PVA to see if its valid

```
 0  8979 {
 0  8980 { Purpose:
 0  8981 {    This routine verifies a PVA relative to the CURRENT USER TASK
 0  8982 {    and returns an error code if its not valid.
 0  8983 {
 0  8984
 0  8985   PROCEDURE [XDCL] mmp$verify_pva
 0  8986     (     p: ^cell;
 0  8987           segment_access: mmt$segment_access_type;
 0  8988      VAR status: syt$monitor_status);
 0  8989
 0  8990     VAR
 0  8991       cst_p: ^ost$cpu_state_table,
 0  8992       ring: 0 .. 15,
 0  8993       pva_p: ^ost$pva,
 0  8994       ste_p: ^mmt$segment_descriptor,
 0  8995       stxe_p: ^mmt$segment_descriptor_extended,
 0  8996       segnum: ost$segment;
 0  8997
 0  8998     status.normal := TRUE;
 4  8999     mtp$cst_p (cst_p);
1E  9000
1E  9001     pva_p := p;
1E  9002     ring := pva_p^.ring;
1E  9003
1E  9004     IF (ring = 0) OR (pva_p^.offset < 0) THEN
4E  9005       mtp$set_status_abnormal ('MM', mme$invalid_pva, status);
4E  9006       RETURN;
62  9007     IFEND;
62  9008
62  9009     IF ring < cst_p^.xcb_p^.xp.p_register.pva.ring THEN
76  9010       ring := cst_p^.xcb_p^.xp.p_register.pva.ring;
7A  9011     IFEND;
7A  9012
7A  9013     segnum := pva_p^.seg;
7A  9014     ste_p := mmp$get_sdt_entry_p (cst_p^.xcb_p, segnum);
7A  9015     stxe_p := mmp$get_sdtx_entry_p (cst_p^.xcb_p, segnum);
7A  9016
7A  9017
7A  9018     IF (segnum > cst_p^.xcb_p^.xp.segment_table_length) OR (ste_p^.ste.v1 = osc$v1_invalid_entry) THEN
EA  9019       mtp$set_status_abnormal ('MM', mme$invalid_pva, status);
FE  9020     ELSEIF mtv$sys_core_init_complete THEN
10A 9021       IF (segment_access = mmc$sat_read) AND ((ring > ste_p^.ste.r2) OR (ste_p^.ste.rp = osc$non_readable))
126 9022           THEN
126 9023         mtp$set_status_abnormal ('MM', mme$invalid_pva, status);
13A 9024       ELSEIF (segment_access = mmc$sat_write) AND ((ring > ste_p^.ste.r1) OR
15C 9025             (ste_p^.ste.wp = osc$non_writable)) THEN
15C 9026         mtp$set_status_abnormal ('MM', mme$invalid_pva, status);
170 9027       ELSEIF (segment_access = mmc$sat_read_or_write) AND (ring > ste_p^.ste.r2) THEN
186 9028         mtp$set_status_abnormal ('MM', mme$invalid_pva, status);
19A 9029       ELSEIF (stxe_p^.sfid.residence = gfc$tr_system_wait_recovery) THEN
1A6 9030         syv$refs_to_unrecovered_seg := syv$refs_to_unrecovered_seg + 1;
1A6 9031         mtp$set_status_abnormal ('MM', mme$ref_to_unrecovered_file, status);
1C6 9032       IFEND;
1C6 9033     IFEND;
1C6 9034
```

MMP$VERIFY_PVA - Test job mode PVA to see if its valid

```
     1C6  9035   PROCEND mmp$verify_pva;
```


MMP$XTASK_PVA_TO_SVA - Convert job mode PVA to SVA

```
       0  9038 {  This request is used in monitor to translate a PVA relative to the
       0  9039 {  CURRENTLY EXECUTING TASK to an SVA. No test is made to see if pages
       0  9040 {  are currently assigned to the PVA.
       0  9041 {
       0  9042 {       MMP$XTASK_PVA_TO_SVA (PVA, SVA, STATUS)
       0  9043 {
       0  9044 {    PVA: (input) This parameter specifies the PVA to be converted.
       0  9045 {    SVA: (output) This parameter specifies the SVA corresponding to
       0  9046 {          PVA.
       0  9047 {    STATUS: (output) This parameter specifies request status.
       0  9048 {          The possible error codes are:
       0  9049 {                mme$invalid_pva
       0  9050 {                mme$ref_to_unrecovered_file
       0  9051 {
       0  9052
       0  9053
       0  9054   PROCEDURE [XDCL] mmp$xtask_pva_to_sva
       0  9055     (     p: ^cell;
       0  9056      VAR sva: ost$system_virtual_address;
       0  9057      VAR status: syt$monitor_status);
       0  9058
       0  9059     VAR
       0  9060       cst_p: ^ost$cpu_state_table,
       0  9061       fde_p: gft$locked_file_desc_entry_p,
       0  9062       ste_p: ^mmt$segment_descriptor,
       0  9063       stxe_p: ^mmt$segment_descriptor_extended,
       0  9064       aste_p: ^mmt$active_segment_table_entry;
       0  9065
       0  9066     mmp$verify_pva (^p, mmc$sat_read_or_write, status);
      26  9067     IF status.normal THEN
      2E  9068       mtp$cst_p (cst_p);
      3E  9069       mmp$convert_pva (p, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
      92  9070     IFEND;
      92  9071
      92  9072   PROCEND mmp$xtask_pva_to_sva;
```

GET_AVAILABLE_PAGE_FRAME

```
        0  9075
        0  9076 {-------------------------------------------------------------------------------------------------------------
        0  9077 {Name:
        0  9078 {    mmp$get_avail_page_frame
        0  9079 {Purpose:
        0  9080 {   This routine is called to find a free or available page frame.
        0  9081 {   No assignment of the page frame is made by this procedure.
        0  9082 {Input:
        0  9083 {   none
        0  9084 {Output:
        0  9085 {   pfti - Page Frame Table index of an available page frame. A value
        0  9086 {         of zero indicates no page frame was available.
        0  9087 {Error Codes:
        0  9088 {  none
        0  9089 {-------------------------------------------------------------------------------------------------------------
        0  9090
        0  9091
        0  9092   PROCEDURE [XDCL, INLINE] mmp$get_avail_page_frame
        4  9093     (VAR pfti: mmt$page_frame_index);
        4  9094
        4  9095     pfti := mmv$gpql [mmc$pq_free].pqle.link.bkw;
        4  9096     IF pfti = 0 THEN
       18  9097       IF (mmv$gpql [mmc$pq_avail].pqle.count <= mmv$min_avail_pages) AND
       32  9098           (mmv$gpql [mmc$pq_free].pqle.count + mmv$gpql [mmc$pq_avail].pqle.count <
       32  9099           mmv$reassignable_page_frames.now) THEN
       32  9100         jsp$free_swapped_jobs_memory (jmv$null_ijl_ordinal);
       4A  9101         pfti := mmv$gpql [mmc$pq_free].pqle.link.bkw;
       54  9102       ELSE
       54  9103         pfti := mmv$gpql [mmc$pq_avail].pqle.link.bkw;
       54  9104         IF pfti <> 0 THEN
       60  9105           mmp$delete_pt_entry (pfti, TRUE);
       76  9106         IFEND;
       76  9107       IFEND;
       76  9108     IFEND;
       76  9109
       76  9110   PROCEND mmp$get_avail_page_frame;
```


RELINK_PAGE_FRAME

```
        0  9113
        0  9114 {-------------------------------------------------------------------------------------------------------------
        0  9115 {  This procedure moves a page frame from its current position
        0  9116 {  in a page queue to the head of a new queue identified by
        0  9117 {  the caller.
        0  9118 {-------------------------------------------------------------------------------------------------------------
        0  9119
        0  9120   PROCEDURE [XDCL] mmp$relink_page_frame
        0  9121     (    pfti: mmt$page_frame_index;
        0  9122          queue_id: mmt$page_frame_queue_id);
        0  9123
        0  9124     VAR
        0  9125       taskid: ost$global_task_id,
        0  9126       ijle_p: ^jmt$initiated_job_list_entry,
        0  9127       qcb_p: ^mmt$page_queue_list_entry,
        0  9128       pfte_p: ^mmt$page_frame_table_entry;
        0  9129
        0  9130     pfte_p := ^mmv$pft_p^ [pfti];
        4  9131
        4  9132     IF syv$perf_keypoints_enabled.memory_keypoints THEN
       2E  9133       #KEYPOINT (osk$performance, osk$m * pfti, ptk$page_assigned_pfti);
       36  9134       #KEYPOINT (osk$performance, osk$m * $INTEGER (queue_id), ptk$page_assigned_queue);
       42  9135       #KEYPOINT (osk$performance, osk$m * (pfte_p^.ijl_ordinal.block_number *
       60  9136             32 + pfte_p^.ijl_ordinal.block_index), ptk$page_assigned_ijl);
       60  9137     IFEND;
       60  9138
       60  9139     IF pfte_p^.queue_id < mmc$pq_job_base THEN
       6C  9140       IF pfte_p^.queue_id <= mmc$pq_last_reassignable THEN
       72  9141         mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now - 1;
       80  9142       ELSEIF (pfte_p^.queue_id = mmc$pq_avail_modified) AND NOT mmv$pt_p^ [pfte_p^.pti].m THEN
       9E  9143         mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
       A8  9144       IFEND;
       A8  9145       qcb_p := ^mmv$gpql [pfte_p^.queue_id].pqle;
       BC  9146     ELSE
       BC  9147       jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
       BC  9148       qcb_p := ^ijle_p^.job_page_queue_list [pfte_p^.queue_id];
       F6  9149     IFEND;
       F6  9150     IF pfte_p^.link.fwd = 0 THEN
       FE  9151       qcb_p^.link.bkw := pfte_p^.link.bkw;
      10A  9152     ELSE
      10A  9153       mmv$pft_p^ [pfte_p^.link.fwd].link.bkw := pfte_p^.link.bkw;
      128  9154     IFEND;
      128  9155     IF pfte_p^.link.bkw = 0 THEN
      130  9156       qcb_p^.link.fwd := pfte_p^.link.fwd;
      13C  9157     ELSE
      13C  9158       mmv$pft_p^ [pfte_p^.link.bkw].link.fwd := pfte_p^.link.fwd;
      15A  9159     IFEND;
      15A  9160     qcb_p^.count := qcb_p^.count - 1;
      15A  9161
      15A  9162     pfte_p^.link.bkw := 0;
      15A  9163     IF (queue_id <= mmc$pq_last_reassignable) AND (pfte_p^.active_io_count > 0) THEN
      17A  9164       IF mmc$debug AND (queue_id <> mmc$pq_free) THEN
      17E  9165         mtp$error_stop ('MM - relink 234');
      19C  9166       IFEND;
      19C  9167       mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + 1;
      19C  9168       pfte_p^.link.fwd := 0;
```

```
1AE  9169      ELSE
1AE  9170        IF queue_id < mmc$pq_job_base THEN
1BA  9171          IF (queue_id <= mmc$pq_last_reassignable) THEN
1BE  9172            mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now + 1;
1BE  9173            IF mmc$debug AND ((queue_id = mmc$pq_avail) AND ((mmv$pt_p^ [pfte_p^.pti].m) OR
1EE  9174                  (mmv$pt_p^ [pfte_p^.pti].v))) THEN
1EE  9175              mtp$error_stop ('MM - relink - trapped modified/valid page in avail');
210  9176            IFEND;
210  9177
210  9178 { Check if scheduler is waiting for memory and ready scheduler if necessary.
210  9179
210  9180            jmp$check_scheduler_memory_wait;
244  9181
244  9182          ELSEIF (queue_id = mmc$pq_avail_modified) AND NOT mmv$pt_p^ [pfte_p^.pti].m THEN
262  9183            IF mmc$debug AND (pfte_p^.active_io_count = 0) THEN
26A  9184              mtp$error_stop ('MM - no IO');
288  9185            IFEND;
288  9186            mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + 1;
292  9187          IFEND;
292  9188          qcb_p := ^mmv$gpql [queue_id].pqle;
2A6  9189        ELSE
2A6  9190          jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
2A6  9191          qcb_p := ^ijle_p^.job_page_queue_list [queue_id];
2DE  9192        IFEND;
2DE  9193        IF qcb_p^.link.fwd = 0 THEN
2E6  9194          qcb_p^.link.bkw := pfti;
2F2  9195        ELSE
2F2  9196          mmv$pft_p^ [qcb_p^.link.fwd].link.bkw := pfti;
310  9197        IFEND;
310  9198        pfte_p^.link.fwd := qcb_p^.link.fwd;
310  9199        qcb_p^.link.fwd := pfti;
310  9200        qcb_p^.count := qcb_p^.count + 1;
32A  9201      IFEND;
32A  9202
32A  9203      pfte_p^.queue_id := queue_id;
32A  9204      IF queue_id = mmc$pq_free THEN
336  9205        pfte_p^.sva.asid := 0; { DONT clear offset - required by mmp$change_asid}
33A  9206
33A  9207 {! delete until DM deletes active IO count - this code causes a timing problem swapping
33A  9208 {! out a job that has recently deleted a segment that had pages being written to disk.
33A  9209 { IF pfte_p^.active_io_count <> 0 THEN
33A  9210 { jmp$get_ijle_p (pfte_p^.ijl_ordinal, ijle_p);
33A  9211 { jmv$ajl_p^ [ijle_p^.ajl_ordinal].active_io_page_count := jmv$ajl_p^ [ijle_p^.ajl_ordinal].
33A  9212 { active_io_page_count - pfte_p^.active_io_count;
33A  9213 { IFEND;
33A  9214
33A  9215      IFEND;
33A  9216
33A  9217      IF (mmv$memory_wait_queue.head <> 0) THEN
342  9218        IF (queue_id <= mmc$pq_last_reassignable) AND (pfte_p^.active_io_count = 0) THEN
352  9219          tmp$dequeue_task (mmv$memory_wait_queue, taskid);
36E  9220        IFEND;
36E  9221      IFEND;
36E  9222
36E  9223      mmp$check_queues;
37E  9224
```



```
37E  9225    PROCEND mmp$relink_page_frame;


 0   9227
 0   9228    PROCEDURE [INLINE] mmp$link_page_frame_to_queue (pfti: mmt$page_frame_index;
 0   9229          pfte_p: ^mmt$page_frame_table_entry);
 0   9230
 0   9231      pfte_p^.link.fwd := mmv$gpql [pfte_p^.queue_id].pqle.link.fwd;
 0   9232      IF mmv$gpql [pfte_p^.queue_id].pqle.link.fwd = 0 THEN
 0   9233        mmv$gpql [pfte_p^.queue_id].pqle.link.bkw := pfti;
 0   9234      ELSE
 0   9235        mmv$pft_p^[mmv$gpql [pfte_p^.queue_id].pqle.link.fwd].link.bkw := pfti;
 0   9236      IFEND;
 0   9237      mmv$gpql [pfte_p^.queue_id].pqle.link.fwd := pfti;
 0   9238      mmv$gpql [pfte_p^.queue_id].pqle.count := mmv$gpql [pfte_p^.queue_id].pqle.count + 1;
 0   9239      mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
 0   9240      mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now + 1;
 0   9241      jmp$check_scheduler_memory_wait;
 0   9242
 0   9243    PROCEND mmp$link_page_frame_to_queue;
```

```
        0  9245
        0  9246 {-------------------------------------------------------------------------------------------------------
        0  9247 {  This procedure is used by the job swapper to claim a large number of pages on swapin.
        0  9248 {-------------------------------------------------------------------------------------------------------
        0  9249
        0  9250
        0  9251   PROCEDURE [XDCL] mmp$claim_pages_for_swapin
        0  9252     (     swapped_job_entry: jmt$swapped_job_entry;
        0  9253           aste_p: ^mmt$active_segment_table_entry;
        0  9254           ijl_ordinal: jmt$ijl_ordinal;
        0  9255      VAR job_page_queue_list: mmt$job_page_queue_list);
        0  9256
        0  9257     VAR
        0  9258       pfti: mmt$page_frame_index,
        0  9259       pfte_p: ^mmt$page_frame_table_entry,
        0  9260       ijl_p: ^jmt$initiated_job_list_entry,
        0  9261       source_queue_id: mmt$page_frame_queue_id,
        0  9262       queue_id: mmt$page_frame_queue_id,
        0  9263       count: integer,
        0  9264       queue_count: integer,
        0  9265       first_pfti: mmt$page_frame_index;
        0  9266
        0  9267     count := 0;
        4  9268     jmp$get_ijle_p (ijl_ordinal, ijl_p);
        4  9269     FOR queue_id := LOWERVALUE (mmt$job_page_queue_index) TO UPPERVALUE (mmt$job_page_queue_index) DO
       3C  9270       count := swapped_job_entry.job_page_queue_count [queue_id] + count;
       3C  9271       IF (queue_id = mmc$pq_job_fixed) AND NOT (jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work) THEN
       68  9272         count := count - ijl_p^.job_fixed_contiguous_pages;
       6E  9273       IFEND;
       6E  9274     FOREND;
       72  9275
       72  9276     WHILE (mmv$gpql [mmc$pq_free].pqle.count < count) AND
       98  9277           (mmv$gpql [mmc$pq_free].pqle.count + mmv$gpql [mmc$pq_avail].pqle.count - count <=
       98  9278           mmv$min_avail_pages) AND (mmv$gpql [mmc$pq_free].pqle.count + mmv$gpql [mmc$pq_avail].pqle.count <
       98  9279           mmv$reassignable_page_frames.now) DO
       98  9280       jsp$free_swapped_jobs_memory (jmv$null_ijl_ordinal);
       B0  9281     WHILEND;
       D4  9282
       D4  9283     source_queue_id := mmc$pq_free;
       D4  9284
       D4  9285   /claim_pages/
       D4  9286     FOR queue_id := LOWERVALUE (mmt$job_page_queue_index) TO UPPERVALUE (mmt$job_page_queue_index) DO
       E8  9287       count := swapped_job_entry.job_page_queue_count [queue_id];
       E8  9288       IF count = 0 THEN
       F8  9289         CYCLE /claim_pages/;
       FC  9290       IFEND;
       FC  9291       job_page_queue_list [queue_id].count := count;
       FC  9292       IF (queue_id = mmc$pq_job_fixed) AND NOT (jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work) THEN
      12A  9293         count := count - ijl_p^.job_fixed_contiguous_pages;
      130  9294       IFEND;
      130  9295       mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now - count;
      130  9296       REPEAT
      13C  9297         queue_count := count;
      13C  9298         pfti := mmv$gpql [source_queue_id].pqle.link.bkw;
      13C  9299         IF pfti = 0 THEN
      14A  9300           IF source_queue_id = mmc$pq_avail THEN
```



```
      14E  9301             mtp$error_stop ('MM - no memory for claim pages');
      16E  9302           IFEND;
      16E  9303           source_queue_id := mmc$pq_avail;
      16E  9304           pfti := mmv$gpql [mmc$pq_avail].pqle.link.bkw;
      176  9305         IFEND;
      176  9306         first_pfti := pfti;
      176  9307         WHILE (count > 0) AND (pfti <> 0) DO
      180  9308           IF syv$perf_keypoints_enabled.memory_keypoints THEN
      188  9309             #KEYPOINT (osk$performance, osk$m * pfti, ptk$pfti_for_swapin);
      190  9310           IFEND;
      190  9311           pfte_p := ^mmv$pft_p^ [pfti];
      190  9312           IF source_queue_id = mmc$pq_avail THEN
      1AE  9313             mmp$delete_pt_entry (pfti, TRUE);
      1C2  9314             pfte_p^.sva := null_sva;
      1CA  9315           IFEND;
      1CA  9316           pfte_p^.ijl_ordinal := ijl_ordinal;
      1CA  9317           pfte_p^.aste_p := aste_p;
      1CA  9318           pfte_p^.queue_id := queue_id;
      1CA  9319           count := count - 1;
      1CA  9320           pfti := pfte_p^.link.bkw;
      1CA  9321         WHILEND;
      1EC  9322
      1EC  9323         mmv$gpql [source_queue_id].pqle.count := mmv$gpql [source_queue_id].pqle.count - queue_count + count;
      1EC  9324         IF job_page_queue_list [queue_id].link.bkw = 0 THEN
      210  9325           job_page_queue_list [queue_id].link.bkw := first_pfti;
      218  9326         ELSE
      218  9327           mmv$pft_p^ [job_page_queue_list [queue_id].link.fwd].link.bkw := first_pfti;
      218  9328           mmv$pft_p^ [first_pfti].link.fwd := job_page_queue_list [queue_id].link.fwd;
      250  9329         IFEND;
      250  9330         IF pfti = 0 THEN
      254  9331           job_page_queue_list [queue_id].link.fwd := mmv$gpql [source_queue_id].pqle.link.fwd;
      254  9332           mmv$gpql [source_queue_id].pqle.link.bkw := 0;
      254  9333           mmv$gpql [source_queue_id].pqle.link.fwd := 0;
      278  9334         ELSE
      278  9335           job_page_queue_list [queue_id].link.fwd := mmv$pft_p^ [pfti].link.fwd;
      278  9336           mmv$pft_p^ [mmv$pft_p^ [pfti].link.fwd].link.bkw := 0;
      278  9337           mmv$gpql [source_queue_id].pqle.link.bkw := pfti;
      278  9338           mmv$pft_p^ [pfti].link.fwd := 0;
      2C4  9339         IFEND;
      2C4  9340       UNTIL count = 0;
      2C8  9341     FOREND /claim_pages/;
      2CC  9342
      2CC  9343     mmp$check_queues;
      2DC  9344
      2DC  9345   PROCEND mmp$claim_pages_for_swapin;
        0  9346
```

MMP$FREE_MEMORY_IN_JOB_QUEUES

```
        0  9349
        0  9350 {-------------------------------------------------------------------------------------------------------
        0  9351 {
        0  9352 { The purpose of this procedure is to efficiently relink ALL the pages in a jobs page queue list
        0  9353 { to the free queue.
        0  9354 {
        0  9355 {-------------------------------------------------------------------------------------------------------
        0  9356
        0  9357   PROCEDURE [XDCL] mmp$free_memory_in_job_queues
        0  9358     (VAR job_page_queue_list: mmt$job_page_queue_list;
        0  9359          increment_now: boolean;
        0  9360          decrement_soon: boolean;
        0  9361          job_termination: boolean);
        0  9362
        0  9363     VAR
        0  9364       asid: ost$asid,
        0  9365       aste_p: ^mmt$active_segment_table_entry,
        0  9366       count: integer,
        0  9367       found: boolean,
        0  9368       ijl_p: ^jmt$initiated_job_list_entry,
        0  9369       index: integer,
        0  9370       ipti: integer,
        0  9371       last_contiguous_pfti: mmt$page_frame_index,
        0  9372       hcount: 1 .. 32,
        0  9373       next_pfti: mmt$page_frame_index,
        0  9374       original_bkw_link: mmt$page_frame_index,
        0  9375       pfte_p: ^mmt$page_frame_table_entry,
        0  9376       pfti: mmt$page_frame_index,
        0  9377       queue_id: mmt$job_page_queue_index,
        0  9378       taskid: ost$global_task_id,
        0  9379       total_pages_freed: integer;
        0  9380
        0  9381     total_pages_freed := 0;
        4  9382
        4  9383     FOR queue_id := UPPERVALUE (mmt$job_page_queue_index) DOWNTO LOWERVALUE (mmt$job_page_queue_index) DO
       16  9384       pfti := job_page_queue_list [queue_id].link.bkw;
       16  9385       original_bkw_link := job_page_queue_list [queue_id].link.bkw;
       16  9386
       16  9387 { If we are freeing the job-fixed queue, we must verify that there are not any
       16  9388 { contiguous pages assigned. If there are contiguous pages assigned, it is
       16  9389 { necessary to determine where in the queue the non-contiguous pages begin.
       16  9390 { The contiguous pages are ALWAYS at the very beginning of the job-fixed
       16  9391 { page queue. Contiguous pages are not freed.
       16  9392 { NOTE: If pages have ASID = 0, then froames are being freed after aborted swapin.
       16  9393
       16  9394       IF pfti <> 0 THEN
       30  9395         IF (queue_id = mmc$pq_job_fixed) THEN
       38  9396           jmp$get_ijle_p (mmv$pft_p^ [pfti].ijl_ordinal, ijl_p);
       38  9397           IF (ijl_p^.job_fixed_contiguous_pages <> 0) AND NOT job_termination AND
       98  9398                 NOT (jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work) THEN
       98  9399             FOR index := 1 TO ijl_p^.job_fixed_contiguous_pages DO
       A0  9400               last_contiguous_pfti := pfti;
       A0  9401               #HASH_SVA (mmv$pft_p^ [pfti].sva, ipti, hcount, found);
       C0  9402               IF found THEN
       D4  9403                 mmp$delete_pt_entry (pfti, TRUE);
       E8  9404               IFEND;
```


MMP$FREE_MEMORY_IN_JOB_QUEUES

```
       E8  9405               pfti := mmv$pft_p^ [pfti].link.bkw;
       E8  9406             FOREND;
      10E  9407           IFEND;
      10E  9408         IFEND;
      10E  9409
      10E  9410         WHILE pfti <> 0 DO
      116  9411           pfte_p := ^mmv$pft_p^ [pfti];
      116  9412           asid := pfte_p^.sva.asid;
      116  9413           next_pfti := pfte_p^.link.bkw;
      116  9414           IF syv$perf_keypoints_enabled.memory_keypoints THEN
      140  9415             #KEYPOINT (osk$performance, osk$m * pfti, ptk$page_assigned_pfti);
      148  9416             #KEYPOINT (osk$performance, osk$m * $INTEGER (mmc$pq_free), ptk$page_assigned_queue);
      14C  9417           IFEND;
      14C  9418
      14C  9419           IF asid <> 0 THEN
      150  9420             IF pfte_p^.active_io_count = 0 THEN
      158  9421               mmp$delete_pt_entry (pfti, TRUE);
*WARN*     9422               mmp$aste_pointer_from_pfti (pfti, aste_p);
      26C  9423               IF aste_p^.pages_in_memory = 0 THEN
      278  9424                 IF aste_p^.sfid.residence = gfc$tr_job THEN
      282  9425                   mmp$free_asid (asid, aste_p);
      296  9426                 IFEND;
      296  9427               IFEND;
      296  9428               pfte_p^.sva := null_sva;
      296  9429               pfte_p^.queue_id := mmc$pq_free;
      2A8  9430             ELSE
      2A8  9431
      2A8  9432 {  IO is still active on a local file or a shared file.
      2A8  9433
      2A8  9434               mmv$pt_p^ [pfte_p^.pti].v := FALSE;
      2A8  9435               IF (pfte_p^.aste_p^.sfid.residence = gfc$tr_job) AND (job_termination) THEN
      2D2  9436                 mmp$delete_pt_entry (pfti, TRUE);
      2E8  9437                 mmp$relink_page_frame (pfti, mmc$pq_free);
      300  9438               ELSE
      300  9439                 mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
      316  9440               IFEND;
      316  9441               IF decrement_soon THEN
      31E  9442                 mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - 1;
      328  9443               IFEND;
      328  9444
      328  9445 { If NOW was already incremented, OC to S2, then decrement NOW since IO is active.
      328  9446 { NOW will be incremented when IO completes.  This applies to local files only.
      328  9447 { NOW count is updated for shared file when movement from JWS to shared queues takes place.
      328  9448
      328  9449               IF (NOT increment_now) AND (pfte_p^.aste_p^.sfid.residence = gfc$tr_job) THEN
      33C  9450                 mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now - 1;
      346  9451               IFEND;
      348  9452             IFEND;
      34C  9453           ELSE
      34C  9454             pfte_p^.queue_id := mmc$pq_free;
      352  9455           IFEND;
      352  9456           pfti := next_pfti;
      352  9457         WHILEND;
      35C  9458
      35C  9459 { The contiguous pages assigned to the job-fixed segment must not be counted
      35C  9460 { when freeing pages in the queues.
```

```
35C  9461
35C  9462          IF (queue_id = mmc$pq_job_fixed) AND (ijl_p^.job_fixed_contiguous_pages <> 0) AND
                         NOT (jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work) AND NOT job_termination THEN
386  9463            count := job_page_queue_list [queue_id].count - ijl_p^.job_fixed_contiguous_pages;
386  9464            job_page_queue_list [queue_id].link.bkw := mmv$pft_p^ [last_contiguous_pfti].link.bkw;
386  9465          ELSE
3C8  9466            count := job_page_queue_list [queue_id].count;
3C8  9467          IFEND;
3DE  9468          IF count > 0 THEN
3DE  9469            IF mmv$gpql [mmc$pq_free].pqle.link.bkw = 0 THEN
3E2  9470              mmv$gpql [mmc$pq_free].pqle.link.bkw := job_page_queue_list [queue_id].link.bkw;
3EE  9471            ELSE
406  9472              mmv$pft_p^ [mmv$gpql [mmc$pq_free].pqle.link.fwd].link.bkw := job_page_queue_list [queue_id].
406  9473                    link.bkw;
406  9474              mmv$pft_p^ [job_page_queue_list [queue_id].link.bkw].link.fwd := mmv$gpql [mmc$pq_free].
406  9475                    pqle.link.fwd;
406  9476            IFEND;
44C  9477            mmv$gpql [mmc$pq_free].pqle.link.fwd := job_page_queue_list [queue_id].link.fwd;
44C  9478            IF (queue_id = mmc$pq_job_fixed) AND (ijl_p^.job_fixed_contiguous_pages <> 0) AND
44C  9479                  NOT (jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work) AND NOT job_termination THEN
48A  9480              job_page_queue_list [queue_id].link.fwd := last_contiguous_pfti;
48A  9481              job_page_queue_list [queue_id].link.bkw := original_bkw_link;
48A  9482              job_page_queue_list [queue_id].count := ijl_p^.job_fixed_contiguous_pages;
48A  9483            ELSE
4A2  9484              job_page_queue_list [queue_id].link.fwd := 0;
4A2  9485              job_page_queue_list [queue_id].link.bkw := 0;
4A2  9486              job_page_queue_list [queue_id].count := 0;
4A2  9487            IFEND;
4C2  9488            mmv$gpql [mmc$pq_free].pqle.count := mmv$gpql [mmc$pq_free].pqle.count + count;
4C2  9489            IF increment_now THEN
4C2  9490              mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now + count;
4D4  9491            IFEND;
4DE  9492            IF decrement_soon THEN
4DE  9493              mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - count;
4E6  9494            IFEND;
4F0  9495            total_pages_freed := total_pages_freed + count;
4F0  9496          IFEND;
4F2  9497        IFEND;
4F6  9498      FOREND;
4F6  9499
4FC  9500      IF increment_now THEN
4FC  9501
504  9502
504  9503  { Check if scheduler is waiting for memory and ready scheduler if necessary; this check is
504  9504  { necessary anytime page_frames.now is incremented, but don't do it inside a loop.
504  9505
504  9506        jmp$check_scheduler_memory_wait;
538  9507
538  9508      IFEND;
538  9509
538  9510      WHILE (mmv$memory_wait_queue.head <> 0) AND (total_pages_freed > 0) DO
548  9511        total_pages_freed := total_pages_freed - 1;
548  9512        tmp$dequeue_task (mmv$memory_wait_queue, taskid);
566  9513      WHILEND;
572  9514
572  9515      mmp$check_queues;
582  9516
```



```
582  9517    PROCEND mmp$free_memory_in_job_queues;
```

```
0  9520
0  9521 {-------------------------------------------------------------------------------------------------------------
0  9522 {Name:
0  9523 {  pf_proc_tables_not_initialized
0  9524 {Purpose:
0  9525 {  This routine is called to process page faults which occur before
0  9526 {  the PQL, PFT, and AST have been initialized. The routine assigns
0  9527 {  an available page frame and makes the page table entry for the
0  9528 {  page.
0  9529 {-------------------------------------------------------------------------------------------------------------
0  9530
0  9531
0  9532   PROCEDURE pf_proc_tables_not_initialized
0  9533     (    xcb_p: ^ost$execution_control_block);
0  9534
0  9535     VAR
0  9536       sva: ost$system_virtual_address,
0  9537       ste_p: ^mmt$segment_descriptor,
0  9538       static_next_rma: [STATIC] integer := 7fffffff(16),
0  9539       static_stop_rma: [STATIC] integer := 7fffffff(16),
0  9540       next_rma: integer,
0  9541       stop_rma: integer,
0  9542       pte: ost$page_table_entry,
0  9543       count: 1 .. 32,
0  9544       found: boolean,
0  9545       full_scan_has_been_done: boolean,
0  9546       pt_length: integer,
0  9547       pt_p: ^ost$page_table,
0  9548       pti: integer;
0  9549
0  9550
0  9551     pt_p := mmv$pt_p;
4  9552     pt_length := mmv$pt_length;
4  9553     ste_p := mmp$get_sdt_entry_p (xcb_p, xcb_p^.xp.untranslatable_pointer.seg);
4  9554     sva.asid := ste_p^.ste.asid;
4  9555     sva.offset := xcb_p^.xp.untranslatable_pointer.offset;
4  9556     sva.offset := (sva.offset DIV osv$page_size) * osv$page_size;
4  9557
4  9558
4  9559 {The following loop is somewhat obscure but is structured for best performance during deadstart.
4  9560 {The loop sets <NEXT_RMA> .. <STOP_RMA> to point to the next block of free pages that can be assigned.
4  9561 {The block is determined by scanning the page table. The FIRST time thru the loop, it locates the
4  9562 {large block at the end of memory. (Deadstart loads most of the OS at the beginning of memory.)
4  9563 {Subsequent passes thru the loop locate the next block following the block that was just assigned.
4  9564 {    (The loop has been optimized by looking at the object code generated and adjusting
4  9565 {     the source to get good object code).
4  9566
4  9568     IF static_next_rma = static_stop_rma THEN
76  9569
76  9570 {  Locate the starting point of the next block.
76  9571
76  9572       next_rma := static_next_rma;
76  9573       stop_rma := static_stop_rma;
76  9574       IF next_rma = 7fffffff(16) THEN
7E  9575         next_rma := osv$180_memory_limits.lower DIV 512;
7E  9576         stop_rma := (osv$180_memory_limits.deadstart_upper DIV 512);
```



```
7E  9577         pti := pt_length;
7E  9578         REPEAT
94  9579           pti := pti - 1;
94  9580           pte := pt_p^ [pti];
94  9581           IF pte.v AND (pte.rma > next_rma) AND (pte.rma < stop_rma) THEN
B2  9582             next_rma := pte.rma;
B6  9583           IFEND;
B6  9584         UNTIL pti = 0;
BA  9585         next_rma := next_rma + osv$page_size DIV 512;
C8  9586       ELSE
C8  9587         full_scan_has_been_done := FALSE;
C8  9588         REPEAT
CE  9589           next_rma := next_rma + (osv$page_size DIV 512);
CE  9590           IF next_rma >= (osv$180_memory_limits.deadstart_upper DIV 512) THEN
E4  9591             IF full_scan_has_been_done THEN
E8  9592               mtp$error_stop ('MM - not enough mem to deadstart');
108  9593             IFEND;
108  9594             next_rma := osv$180_memory_limits.lower DIV 512;
108  9595             full_scan_has_been_done := TRUE;
114  9596           IFEND;
114  9597           pti := pt_length - 1;
114  9598           pte := pt_p^ [pti];
114  9599           WHILE (pti > 0) AND (NOT pte.v OR (pte.rma <> next_rma)) DO
134  9600             pti := pti - 1;
134  9601             pte := pt_p^ [pti];
134  9602           WHILEND;
152  9603         UNTIL NOT pte.v OR (pte.rma <> next_rma);
162  9604       IFEND;
162  9605
162  9606 {  Locate the end of the block just selected.
162  9607
162  9608       stop_rma := (osv$180_memory_limits.deadstart_upper DIV 512);
162  9609       pti := pt_length;
162  9610       REPEAT
16C  9611         pti := pti - 1;
16C  9612         pte := pt_p^ [pti];
16C  9613         IF pte.v AND (pte.rma > next_rma) AND (pte.rma < stop_rma) THEN
18A  9614           stop_rma := pte.rma;
18E  9615         IFEND;
18E  9616       UNTIL pti = 0;
192  9617       static_next_rma := next_rma;
192  9618       static_stop_rma := stop_rma;
19E  9619     IFEND;
19E  9620
19E  9622
19E  9623     #HASH_SVA (sva, pti, count, found);
1A0  9624     IF found THEN
1B4  9625       mtp$error_stop ('MM - PTE exists');
1D2  9626     IFEND;
1D2  9627     pti := pti - count + 1;
1D2  9628     IF pti < 0 THEN
1DE  9629       pti := pti + mmv$pt_length;
1E4  9630     IFEND;
1E4  9631     count := 1;
1E4  9632     WHILE (mmv$pt_p^ [pti].pageid.asid <> 0) AND (count < 33) DO
202  9633       count := count + 1;
```

PF_PROC_TABLES_NOT_INITIALIZED

```
202  9634        pti := pti + 1;
202  9635        IF pti = mmv$pt_length THEN
20E  9636          pti := 0;
212  9637        IFEND;
212  9638      WHILEND;
22E  9639
22E  9640      IF count = 33 THEN
236  9641        mtp$error_stop ('MM - PT full in deadstart');
256  9642      IFEND;
256  9643
256  9644      pte.v := FALSE;
256  9645      pte.c := TRUE;
256  9646      pte.u := TRUE;
256  9647      pte.m := FALSE;
256  9648      pte.pageid.asid := sva.asid;
256  9649      pte.pageid.pagenum := sva.offset DIV 512;
256  9650      pte.rma := static_next_rma;
256  9651      static_next_rma := static_next_rma + (osv$page_size DIV 512);
256  9652      mmv$pt_p^ [pti] := pte;
256  9653      mmp$preset_real_memory (sva, pmc$initialize_to_zero);
2BE  9654      mmv$pt_p^ [pti].v := TRUE;
2BE  9655
2BE  9656    PROCEND pf_proc_tables_not_initialized;
```


MMP$SEND_ESCAPED_ALLOC_FLAG

```
 0  9658
 0  9659 { Purpose:
 0  9660 {   This procedure is called when a WRITE_PAGE_TO_DISK request discovers escaped
 0  9661 {   allocation. This procedure sends a flag to a task to assign the backing storage.
 0  9662 {
 0  9663
 0  9664
 0  9665    PROCEDURE mmp$send_escaped_alloc_flag
 0  9666      (     fde_p: gft$locked_file_desc_entry_p;
 0  9667            pfte_p: ^mmt$page_frame_table_entry);
 0  9668
 0  9669      VAR
 0  9670        ijl_ordinal: jmt$ijl_ordinal,
 0  9671        ijle_p: ^jmt$initiated_job_list_entry,
 0  9672        old_assign_active: integer,
 0  9673        pfte_ijle_p: ^jmt$initiated_job_list_entry,
 0  9674        sfid: gft$system_file_identifier,
 0  9675        status: syt$monitor_status,
 0  9676        ste_p: ^mmt$segment_descriptor,
 0  9677        stxe_p: ^mmt$segment_descriptor_extended,
 0  9678        xcb_p: ^ost$execution_control_block;
 0  9679
 0  9680
 0  9681 { Try to get a pointer the the XCB of the last task using the segment. The GTID will
 0  9682 { be invalid if the task has terminated.
 0  9683
 0  9684      tmp$test_get_xcb_p (fde_p^.global_task_id, xcb_p, ijle_p);
28  9685      jmp$get_ijle_p (pfte_p^.ijl_ordinal, pfte_ijle_p);
28  9686      gfp$mtr_get_sfid_from_fde_p (fde_p, sfid, ijl_ordinal);
B0  9687
B0  9688
B0  9689 { If the GTID is still valid, notify the task to assign space. If the segment is still valid AND the
B0  9690 { same SFID, use the SDTX.ASSIGN_ACTIVE mechanism to notify the task to allocate space. If the segment
B0  9691 { is NOT the same SFID, use field in the XCB to pass the SFID.
B0  9692
B0  9693      IF (xcb_p <> NIL) AND (ijle_p = pfte_ijle_p) THEN
D6  9694        ste_p := mmp$get_sdt_entry_p (xcb_p, fde_p^.last_segment_number);
D6  9695        stxe_p := mmp$get_sdtx_entry_p (xcb_p, fde_p^.last_segment_number);
D6  9696        IF (ste_p^.ste.vl <> osc$vl_invalid_entry) AND (sfid = stxe_p^.sfid) THEN
138 9697          old_assign_active := stxe_p^.assign_active;
138 9698          set_assign_active (stxe_p, pfte_p^.sva.offset);
164 9699          IF old_assign_active = mmc$assign_active_null THEN
16C 9700            tmp$set_monitor_flag (fde_p^.global_task_id, mmc$mf_segment_mgr_flag, status);
18E 9701          IFEND;
192 9702        ELSEIF xcb_p^.assign_active_sfid = gfv$null_sfid THEN
1A6 9703          xcb_p^.assign_active_sfid := sfid;
1A6 9704          tmp$set_monitor_flag (fde_p^.global_task_id, mmc$mf_segment_mgr_flag, status);
1D0 9705        IFEND;
1D0 9706        jmp$unlock_ajl (ijle_p);
28A 9707
28A 9708
28A 9709 { If the GTID is no longer valid, let the job monitor of the job take care of allocation.
28A 9710
28A 9711      ELSE
28A 9712        tmp$get_xcb_p (pfte_ijle_p^.job_monitor_taskid, xcb_p, ijle_p);
2B2 9713        IF xcb_p <> NIL THEN
```

MMP$SEND_ESCAPED_ALLOC_FLAG

```
2BC  9714          IF xcb_p^.assign_active_sfid = gfv$null_sfid THEN
2CC  9715            xcb_p^.assign_active_sfid := sfid;
2CC  9716            tmp$set_monitor_flag (ijle_p^.job_monitor_taskid, mmc$mf_segment_mgr_flag, status);
2FA  9717          IFEND;
2FA  9718          mmv$jmtr_escaped_allocate := mmv$jmtr_escaped_allocate + 1;
2FA  9719          jmp$unlock_ajl (ijle_p);
3C2  9720        ELSE
3C2  9721          mtp$error_stop ('MM - lost segment owner');  {!! can we get here??
3DE  9722          mmv$lost_escaped_allocate := mmv$lost_escaped_allocate + 1;
3EC  9723        IFEND;
3EC  9724      IFEND;
3EC  9725
3EC  9726    PROCEND mmp$send_escaped_alloc_flag;
  o  9727
```


MMP$WRITE_PAGE_TO_DISK

```
  o  9730
  o  9731 {------------------------------------------------------------------------
  o  9732 { This procedure is used to write a page to disk. All pages in the transfer unit will be
  o  9733 { written unless they are locked.
  o  9734 {
  o  9735 {------------------------------------------------------------------------
  o  9736
  o  9737
  o  9738   TYPE
  o  9739     mmt$write_page_to_disk_status = (ws_ok, ws_physical_io_reject, ws_no_file_assigned, ws_disk_flaws,
  o  9740     ws_device_manager_reject, ws_volume_unavailable, ws_server_terminated);
  o  9741
  o  9742
  o  9743   PROCEDURE [XDCL] mmp$write_page_to_disk
  o  9744     (     fde_p: gft$locked_file_desc_entry_p;
  o  9745           pfti: mmt$page_frame_index;
  o  9746           iotype: iot$io_function;
  o  9747           io_id: mmt$io_identifier;
  o  9748           multiple_page_req: boolean;
  o  9749      VAR write_status: mmt$write_page_to_disk_status);
  o  9750
  o  9751     VAR
  o  9752       aste_p: ^mmt$active_segment_table_entry,
  o  9753       boffset: integer,
  o  9754       buffer_descriptor: mmt$buffer_descriptor,
  o  9755       count: 1 ..32,
  o  9756       eoffset: integer,
  o  9757       found: boolean,
  o  9758       ijle_p: ^jmt$initiated_job_list_entry,
  o  9759       ijl_ordinal: jmt$ijl_ordinal,
  o  9760       length: integer,
  o  9761       lsva: ost$system_virtual_address,
  o  9762       max_bytes_to_write: integer,
  o  9763       offset: integer, {dont make this a subrange}
  o  9764       pfte_p: ^mmt$page_frame_table_entry,
  o  9765       pte_p: ^ost$page_table_entry,
  o  9766       pti: integer,
  o  9767       served_file: boolean,
  o  9768       status: syt$monitor_status,
  o  9769       stxe_p: ^mmt$segment_descriptor_extended,
  o  9770       sva: ost$system_virtual_address,
  o  9771       tu_start: integer,
  o  9772       tu_end: integer,
  o  9773       write_multiple_pages: boolean,
  o  9774       xcb_p: ^ost$execution_control_block,
  o  9775       xpfti: mmt$page_frame_index;
  o  9776
  o  9777
  o  9778
  o  9779 { If the segment is not assigned to a file, reject the request and send a signal to the
  o  9780 { owner of the segment to assign a backing file.
  o  9781
  o  9782     pfte_p := ^mmv$pft_p^ [pfti];
  4  9783     IF fde_p^.media = gfc$fm_transient_segment THEN
 30  9784       mmp$send_escaped_alloc_flag (fde_p, pfte_p);
 42  9785       write_status := ws_no_file_assigned;
```

```
42  9786        mmv$write_page_statistics [write_status] := mmv$write_page_statistics [write_status] + 1;
42  9787        RETURN;
5C  9788      IFEND;
5C  9789
5C  9790
5C  9791  { Reject the write if the page belongs to a file that has not yet been recovered.
5C  9792  { The write must be delayed for a while.
5C  9793
5C  9794      aste_p := pfte_p^.aste_p;
5C  9795      IF aste_p^.sfid.residence = gfc$tr_system_wait_recovery THEN
6A  9796        write_status := ws_physical_io_reject;
6A  9797        mmv$write_page_statistics [write_status] := mmv$write_page_statistics [write_status] + 1;
6A  9798        RETURN;
84  9799      IFEND;
84  9800
84  9801
84  9802  { Determine the maximum number of bytes that can be written. For mass storage files, it is
84  9803  { an allocation unit. For served files, the size is the smaller of the files allocation unit size
84  9804  { and a constant that is dependent on the buffer size in STORENET.
84  9805  { Note also that served files do not allow multiple outstanding write requests on a
84  9806  { page because writes can be processed out of order.
84  9807
84  9808      max_bytes_to_write := fde_p^.allocation_unit_size;
84  9809
84  9810      served_file := (fde_p^.media = gfc$fm_served_file);
84  9811      IF served_file THEN
9A  9812        IF mmv$pft_p^ [pfti].active_io_count <> 0 THEN
BC  9813          write_status := ws_physical_io_reject;
BC  9814          mmv$write_page_statistics [write_status] := mmv$write_page_statistics [write_status] + 1;
BC  9815          RETURN;
D6  9816        IFEND;
D6  9817  {!??} mmv$pft_p^ [pfti].io_error := ioc$no_error;
D6  9818        IF max_bytes_to_write > dfv$max_bytes_to_write THEN
FE  9819          max_bytes_to_write := dfv$max_bytes_to_write;
102 9820        IFEND;
102 9821      IFEND;
102 9822
102 9823
102 9824  {Calculate the SVA and LENGTH of the data to write to disk. The algorithm is to
102 9825  {start with the page specified by <pfti> and search contiguous pages
102 9826  {in both directions in the segment until 1) the ends of the transfer unit are passed, 2) a locked page is
102 9827  {found (PFT.LOCKED_PAGE), 3) a page not in memory is found , 4) a non-modified page is found, OR
102 9828  {5) a page is found that already has active IO (server only).
102 9829  {The amount of data to write to disk is bounded by the outermost modified pages found by the search.
102 9830
102 9831  {Pages in the available modified queue will always be written.  Multiple pages not in the available
102 9832  {modified queue will not be written if the page belongs to a swapped job.
102 9833
102 9834      sva := pfte_p^.sva;
102 9835      tu_start := (sva.offset DIV max_bytes_to_write) * max_bytes_to_write;
102 9836      tu_end := tu_start + max_bytes_to_write;
102 9837      jmp$get_ijle_p (aste_p^.ijl_ordinal, ijle_p);
102 9838      write_multiple_pages := multiple_page_req AND (ijle_p^.swap_status = jmc$iss_executing);
152 9839      lsva := sva;
152 9840      offset := sva.offset;
152 9841      boffset := offset;
```



```
152 9842
152 9843    /find_starting_page/
152 9844      WHILE boffset > tu_start DO
164 9845        boffset := boffset - osv$page_size;
164 9846        lsva.offset := boffset;
164 9847        #HASH_SVA (lsva, pti, count, found);
170 9848        IF NOT found OR NOT mmv$pt_p^ [pti].m THEN
19A 9849          EXIT /find_starting_page/;
19E 9850        IFEND;
19E 9851        xpfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
19E 9852        IF (mmv$pft_p^ [xpfti].locked_page <> mmc$lp_not_locked) OR
1FC 9853              ((mmv$pft_p^ [xpfti].queue_id <> mmc$pq_avail_modified) AND
1FC 9854              (NOT write_multiple_pages)) OR (served_file AND (mmv$pft_p^ [xpfti].active_io_count <> 0)) THEN
1FC 9855          EXIT /find_starting_page/;
200 9856        IFEND;
200 9857        offset := boffset;
200 9858      WHILEND /find_starting_page/;
208 9859
208 9860      eoffset := sva.offset + osv$page_size;
208 9861      sva.offset := offset;
208 9862
208 9863    /find_ending_page/
208 9864      WHILE (eoffset < tu_end) DO
220 9865        lsva.offset := eoffset;
220 9866        #HASH_SVA (lsva, pti, count, found);
226 9867        IF NOT found OR NOT mmv$pt_p^ [pti].m THEN
250 9868          EXIT /find_ending_page/;
254 9869        IFEND;
254 9870        xpfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
254 9871        IF (mmv$pft_p^ [xpfti].locked_page <> mmc$lp_not_locked) OR
2B6 9872              ((mmv$pft_p^ [xpfti].queue_id <> mmc$pq_avail_modified) AND
2B6 9873              (NOT write_multiple_pages)) OR (served_file AND (mmv$pft_p^ [xpfti].active_io_count <> 0)) THEN
2B6 9874          EXIT /find_ending_page/;
2BA 9875        IFEND;
2BA 9876        eoffset := eoffset + osv$page_size;
2BA 9877      WHILEND /find_ending_page/;
2C4 9878      length := eoffset - sva.offset;
2C4 9879
2C4 9880
2C4 9881  {Issue the write request to device manager. NOTE that the process of locking the page frames
2C4 9882  {will clear the 'modified' bit in the page table.
2C4 9883
2C4 9884      buffer_descriptor.buffer_descriptor_type := mmc$bd_paging_io;
2C4 9885      buffer_descriptor.sva := sva;
2C4 9886      buffer_descriptor.page_count := length DIV osv$page_size;
2C4 9887
2C4 9888  { Issue the i/o.  Case includes pages on/not on server.
2C4 9889  { Note: EOI update must be done first since it is used by file server.
2C4 9890
2C4 9891      mmp$update_eoi (fde_p, eoffset - osv$page_size, mmc$uer_page_written);
330 9892      IF NOT served_file THEN
334 9893        iop$pager_io (fde_p, sva.offset, buffer_descriptor, length, iotype, io_id, status);
37E 9894      ELSE
37E 9895        dfp$server_io (fde_p, iotype, sva.offset, length, io_id, buffer_descriptor, status);
3C4 9896      IFEND;
3C4 9897
```

MMP$WRITE_PAGE_TO_DISK

```
3C4  9898      IF status.normal THEN
3CC  9899        fde_p^.time_last_modified := #free_running_clock (0);
3D2  9900        write_status := ws_ok;
3E2  9901      ELSEIF status.condition = dme$transient_error THEN
3F2  9902        write_status := ws_device_manager_reject;
400  9903      ELSEIF status.condition = ioe$requests_full THEN
408  9904        write_status := ws_physical_io_reject;
414  9905      ELSEIF (status.condition = ioe$unit_disabled) OR (status.condition = dme$volume_unavailable) THEN
424  9906        write_status := ws_volume_unavailable;
432  9907      ELSEIF status.condition = dfe$server_has_terminated THEN
43A  9908        write_status := ws_server_terminated;
448  9909      ELSEIF status.condition = dme$job_mode_allocate_required THEN
450  9910        mmp$send_escaped_alloc_flag (fde_p, pfte_p);
462  9911        write_status := ws_device_manager_reject;
470  9912      ELSE
470  9913        mtp$error_stop ('MM - unexpected phy io error');
48C  9914      IFEND;
48C  9915
48C  9916  {Update statistics.
48C  9917
48C  9918      mmv$write_page_statistics [write_status] := mmv$write_page_statistics [write_status] + 1;
48C  9919      IF length <> osv$page_size THEN  {!This stat should be moved to IF STATUS.NORMAL
4AA  9920        mmv$aging_statistics.multiple_pages_written_to_disk :=
4B6  9921              mmv$aging_statistics.multiple_pages_written_to_disk + 1;
4B6  9922      ELSE
4B6  9923        mmv$aging_statistics.page_written_to_disk := mmv$aging_statistics.page_written_to_disk + 1;
4C0  9924      IFEND;
4C0  9925
4C0  9926    PROCEND mmp$write_page_to_disk;
  0  9927
```


MMP$REMOVE_PAGES_FROM_JWS

```
  0  9929
  0  9930  {-------------------------------------------------------------------------------------------------------
  0  9931  {Name:
  0  9932  {  mmp$remove_pages_from_JWS
  0  9933  {Purpose:
  0  9934  {  This procedure is called to remove a page from the working set of a job.
  0  9935  {Notes:
  0  9936  {  - this routine will take care of page map purges if the page goes to the AVAIL_MODIFIED queue.
  0  9937  {     No purging is done if the page goes to the JWS queue. This queue is used for swapping only and no
  0  9938  {     purging is necessary.
  0  9939  {-------------------------------------------------------------------------------------------------------
  0  9940
  0  9941
  0  9942    PROCEDURE [XDCL] mmp$remove_pages_from_jws
  0  9943      (     modified_queue_id: mmt$page_frame_queue_id;
  0  9944            ijle_p: ^jmt$initiated_job_list_entry;
  0  9945       VAR xmcount: integer;
  0  9946       VAR xrcount: integer);
  0  9947
  0  9948      VAR
  0  9949        fde_p: gft$locked_file_desc_entry_p,
  0  9950        tos: integer,
  0  9951        pfte_p: ^mmt$page_frame_table_entry,
  0  9952        pfti: mmt$page_frame_index,
  0  9953        aste_p: ^mmt$active_segment_table_entry,
  0  9954        write_status: mmt$write_page_to_disk_status,
  0  9955        io_id: mmt$io_identifier,
  0  9956        mcount: integer,
  0  9957        rcount: integer,
  0  9958        pte_p: ^ost$page_table_entry;
  0  9959
  0  9960
  0  9961      mcount := 0;
  4  9962      rcount := 0;
  4  9963      io_id.specified := FALSE;
  4  9964
  4  9965  { Scan the PFTI array and eliminate any entries that cannot be removed. Clear the
  4  9966  { 'valid' and 'used' bits for the entries that may be removed. Note: this step is unnecessary if
  4  9967  { the pages are going to the JWS queue - this is done ONLY for job swapout. The map purge is not
  4  9968  { required until the job starts running again. The job swapper insures that the purge occurs.
  4  9969
  4  9970
  4  9971      IF modified_queue_id <> mmc$pq_job_working_set THEN
 1E  9972        mmp$reset_find_next_pfti (pfti);
 78  9973        WHILE pfti <> 0 DO
 8A  9974          mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := FALSE;
 8A  9975          mmv$pt_p^ [mmv$pft_p^ [pfti].pti].u := FALSE;
 8A  9976          mmp$find_next_pfti (pfti);
10A  9977        WHILEND;
116  9978
116  9979
116  9980  { Now that all used and valid bits have been cleared, purge that page map. It is important on a dual
116  9981  { CPU system to purge the page maps before deleting the page table entry. Also, it is not
116  9982  { possible to reliably determine the state of the 'modified' bit without clearing the 'valid'
116  9983  { bit and purging the page map.
116  9984
```

```
 116  9985        mmp$purge_all_page_map;
 134  9986      IFEND;
 134  9987
 134  9988      mmp$reset_find_next_pfti (pfti);
 18E  9989
 18E  9990      WHILE pfti <> 0 DO
 1A2  9991        pfte_p := ^mmv$pft_p^ [pfti];
 1A2  9992        aste_p := pfte_p^.aste_p;
 1A2  9993        pfte_p^.age := 0;
 1A2  9994        pfte_p^.cyclic_age := 0;
 1A2  9995        pte_p := ^mmv$pt_p^ [pfte_p^.pti];
 1A2  9996        IF modified_queue_id = mmc$pq_job_working_set THEN
 1E4  9997          pte_p^.v := FALSE;
 1E4  9998          pte_p^.u := FALSE;
 1F0  9999        IFEND;
 1F0 10000
 1F0 10001
 1F0 10002 { If the segment is locked and (potentially) modified, the pages cannot be removed.
 1F0 10003 { Reset PTE.V because it was cleared above.
 1F0 10004
 1F0 10005        gfp$mtr_get_locked_fde_p (aste_p^.sfid, ijle_p, fde_p);
 272 10006        IF fde_p^.segment_lock.locked_for_write AND (pte_p^.m OR (pfte_p^.active_io_count > 0)) THEN
 290 10007          pfti := 0;
 290 10008          pte_p^.v := TRUE;
 29E 10009
 29E 10010
 29E 10011 { If the page belongs to a device file that has the WIRE_EOI attribute, dont remove it if
 29E 10012 { it is the last page of the segment. Set the USED bit so it wont be aged out again
 29E 10013 { for a while. Reset PTE.V because it was cleared above.
 29E 10014
 29E 10015        ELSEIF fde_p^.flags.wire_eoi_page THEN
 2AC 10016          IF (fde_p^.eoi_byte_address - mmv$pft_p^ [pfti].sva.offset) <= osv$page_size THEN
 2DE 10017            pte_p^.v := TRUE;
 2DE 10018            pte_p^.u := TRUE;
 2DE 10019            pfti := 0;
 2F0 10020          IFEND;
 2F6 10021
 2F6 10022
 2F6 10023 { If the page belongs to a stack segment and is no longer needed, delete the page
 2F6 10024 { and relink the page frame to the free queue.
 2F6 10025
 2F6 10026        ELSEIF fde_p^.stack_for_ring <> 0 THEN
 2FE 10027          tmp$get_top_of_stack (fde_p^.global_task_id, fde_p^.stack_for_ring, tos);
 31E 10028          IF pfte_p^.sva.offset >= tos THEN
 330 10029            mmp$delete_pt_entry (pfti, TRUE);
 348 10030            mmp$relink_page_frame (pfti, mmc$pq_free);
 360 10031            fde_p^.eoi_byte_address := tos;
 360 10032            pfti := 0;
 360 10033            rcount := rcount + 1;
 372 10034          IFEND;
 376 10035        IFEND;
 376 10036
 376 10037 { Remove the page from the JWS and put it in the new queue.  New queue is determined by the state of the 'UM'
 376 10038 { bits in the page table entry. New queue may also be specified by caller - modified pages must
 376 10039 { be put in JWS queue if job is being swapped out.
 376 10040
```



```
 376 10041        IF pfti <> 0 THEN
 37E 10042          rcount := rcount + 1;
 37E 10043          IF NOT pte_p^.m THEN
 38A 10044            IF pfte_p^.active_io_count <> 0 THEN
 392 10045              mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
 3AA 10046            ELSEIF mmv$no_memory_buffering THEN
 3B2 10047              mmp$delete_pt_entry (pfti, TRUE);
 3C6 10048              mmp$relink_page_frame (pfti, mmc$pq_free);
 3E2 10049            ELSE
 3E2 10050              mmp$relink_page_frame (pfti, mmc$pq_avail);
 3F6 10051            IFEND;
 3FA 10052          ELSE
 3FA 10053            mmp$relink_page_frame (pfti, modified_queue_id);
 412 10054            IF ((mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon) <
 428 10055                 mmv$write_aged_out_pages) AND (modified_queue_id = mmc$pq_avail_modified) THEN
 428 10056              io_id.specified := FALSE;
 428 10057              mmp$write_page_to_disk (fde_p, pfti, ioc$write_page, io_id, mmv$multi_page_write, write_status);
 460 10058            IFEND;
 460 10059            mcount := mcount + 1;
 462 10060          IFEND;
 464 10061        IFEND;
 464 10062
 464 10063        mmp$find_next_pfti (pfti);
 4B0 10064      WHILEND;
 4BC 10065
 4BC 10066      mmv$aging_statistics.remove_unmodified_page_from_ws :=
 4BC 10067            mmv$aging_statistics.remove_unmodified_page_from_ws + rcount - mcount;
 4BC 10068      mmv$aging_statistics.remove_modified_page_from_ws := mmv$aging_statistics.remove_modified_page_from_ws +
 4BC 10069            mcount;
 4BC 10070      xrcount := rcount;
 4BC 10071      xmcount := mcount;
 4BC 10072
 4BC 10073    PROCEND mmp$remove_pages_from_jws;
   0 10074
```

```
  0 10076
  0 10077 {-------------------------------------------------------------------------------------------------------------
  0 10078 {Name:
  0 10079 {  mmp$remove_page_from_JWS
  0 10080 {Purpose:
  0 10081 {  This procedure is called to remove a page from the working set of a job.
  0 10082 {Notes:
  0 10083 {  - this routine will take care of page map purges.
  0 10084 {  - this routine does not neccessarily write the page to disk.
  0 10085 {-------------------------------------------------------------------------------------------------------------
  0 10086
  0 10087
  0 10088   PROCEDURE [XDCL] mmp$remove_page_from_jws
  0 10089     (     pfti: mmt$page_frame_index;
  0 10090           ijle_p: ^jmt$initiated_job_list_entry;
  0 10091      VAR mcount: integer;
  0 10092      VAR rcount: integer);
  0 10093
  0 10094     VAR
  0 10095       fde_p: gft$locked_file_desc_entry_p,
  0 10096       tos: integer,
  0 10097       pfte_p: ^mmt$page_frame_table_entry,
  0 10098       aste_p: ^mmt$active_segment_table_entry,
  0 10099       write_status: mmt$write_page_to_disk_status,
  0 10100       io_id: mmt$io_identifier,
  0 10101       pte_p: ^ost$page_table_entry;
  0 10102
  0 10103
  0 10104
  0 10105 {Reject the request if the page is locked.
  0 10106
  0 10107     mcount := 0;
  4 10108     pfte_p := ^mmv$pft_p^ [pfti];
  4 10109     IF (pfte_p^.locked_page <> mmc$lp_not_locked) THEN
 36 10110       rcount := 0;
 36 10111       RETURN; {<----}
 40 10112     IFEND;
 40 10113
 40 10114     aste_p := pfte_p^.aste_p;
 40 10115
 40 10116 {Clear the valid bit in the page table entry for the page.
 40 10117 {Valid bit MUST be cleared and map purged (in dual CPU) before examining modified bit.
 40 10118
 40 10119     pte_p := ^mmv$pt_p^ [pfte_p^.pti];
 40 10120     pte_p^.v := FALSE;
 40 10121     pte_p^.u := FALSE;
 40 10122     mmp$sva_purge_one_page_map (pfte_p^.sva);
 7A 10123
 7A 10124
 7A 10125 {If page belongs to a locked segment and is modified, leave it alone. NOTE: valid bit must
 7A 10126 {be set again because it was cleared in a previous step.
 7A 10127
 7A 10128     gfp$mtr_get_locked_fde_p (aste_p^.sfid, ijle_p, fde_p);
 FA 10129
 FA 10130     IF fde_p^.segment_lock.locked_for_write AND (pte_p^.m OR (pfte_p^.active_io_count > 0)) THEN
118 10131       pte_p^.v := TRUE;
```



```
118 10132       rcount := 0;
118 10133       RETURN; {<----}
128 10134     IFEND;
128 10135
128 10136 {Reset page ages.
128 10137
128 10138     pfte_p^.age := 0;
128 10139     pfte_p^.cyclic_age := 0;
128 10140     rcount := 1;
128 10141
128 10142
128 10143 {If the page belongs to a stack segment and is no longer needed, delete the page
128 10144 {and relink the page frame to the free queue.
128 10145
128 10146     IF fde_p^.stack_for_ring <> 0 THEN
144 10147       tmp$get_top_of_stack (fde_p^.global_task_id, fde_p^.stack_for_ring, tos);
164 10148       IF pfte_p^.sva.offset >= tos THEN
176 10149         mmp$delete_pt_entry (pfti, TRUE);
18E 10150         mmp$relink_page_frame (pfti, mmc$pq_free);
1A2 10151         fde_p^.eoj_byte_address := tos;
1A2 10152         RETURN; {<----}
1B0 10153       IFEND;
1B0 10154     IFEND;
1B0 10155
1B0 10156
1B0 10157 {Remove the page from the JWS and put it in the new queue.  New queue is determined by the state of the 'UM'
1B0 10158 {bits in the page table entry. New queue may also be specified by caller - modified pages must
1B0 10159 {be put in JWS queue if job is being swapped out.
1B0 10160
1B0 10161     IF NOT pte_p^.m THEN
1BA 10162       IF pfte_p^.active_io_count <> 0 THEN
1C2 10163         mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
1DE 10164       ELSEIF mmv$no_memory_buffering THEN
1E6 10165         mmp$delete_pt_entry (pfti, TRUE);
1FE 10166         mmp$relink_page_frame (pfti, mmc$pq_free);
216 10167       ELSE
216 10168         mmp$relink_page_frame (pfti, mmc$pq_avail);
22E 10169       IFEND;
22E 10170       mmv$aging_statistics.remove_unmodified_page_from_ws :=
23A 10171             mmv$aging_statistics.remove_unmodified_page_from_ws + 1;
23A 10172     ELSE
23A 10173       mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
252 10174       IF ((mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon) <
264 10175             mmv$write_aged_out_pages) THEN
264 10176         io_id.specified := FALSE;
264 10177         mmp$write_page_to_disk (fde_p, pfti, ioc$write_page, io_id, mmv$multi_page_write, write_status);
29C 10178       IFEND;
29C 10179       mmv$aging_statistics.remove_modified_page_from_ws :=
29C 10180             mmv$aging_statistics.remove_modified_page_from_ws + 1;
29C 10181       mcount := 1;
2AA 10182     IFEND;
2AA 10183
2AA 10184   PROCEND mmp$remove_page_from_jws;
```

MMP$REMOVE_PAGE_FROM_JOB

```
        0 10187
        0 10188 {-------------------------------------------------------------------------------
        0 10189 {
        0 10190 { This procedure is called to remove pages from a job's working set. It is called
        0 10191 { from mmp$mm_write_modified_pages (if the request is coming from detach file) to
        0 10192 { relink unmodified jws pages to the available or free queues.
        0 10193 {
        0 10194 {-------------------------------------------------------------------------------
        0 10195
        0 10196   PROCEDURE [XDCL] mmp$remove_page_from_job
        0 10197     (     pfti: mmt$page_frame_index);
        0 10198
        0 10199     VAR
        0 10200       pte_p: ^ost$page_table_entry;
        0 10201
        0 10202     pte_p := ^mmv$pt_p^ [mmv$pft_p^ [pfti].pti];
        4 10203
        4 10204 {Clear the valid bit in the page table entry for the page.
        4 10205
        4 10206     pte_p^.v := FALSE;
        4 10207     pte_p^.u := FALSE;
        4 10208     mmp$sva_purge_one_page_map (mmv$pft_p^ [pfti].sva);
       5A 10209     mmv$pft_p^ [pfti].age := 0;
       5A 10210     mmv$pft_p^ [pfti].cyclic_age := 0;
       5A 10211
       5A 10212 {Remove the page from the JWS and put it in the new queue.
       5A 10213
       5A 10214     IF NOT pte_p^.m THEN
       88 10215       IF mmv$pft_p^ [pfti].queue_id = mmc$pq_job_io_error THEN
       94 10216         mmp$relink_page_frame (pfti, mmc$pq_shared_io_error);
       B0 10217       ELSEIF mmv$pft_p^ [pfti].active_io_count <> 0 THEN
       B8 10218         mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
       D0 10219       ELSEIF mmv$no_memory_buffering THEN
       D8 10220         mmp$delete_pt_entry (pfti, TRUE);
       EE 10221         mmp$relink_page_frame (pfti, mmc$pq_free);
      106 10222       ELSE
      106 10223         mmp$relink_page_frame (pfti, mmc$pq_avail);
      11C 10224       IFEND;
      11E 10225     ELSE
      11E 10226       mtp$error_stop ('PAGE FOUND MODIFIED ON REMOVE');
      13E 10227     IFEND;
      13E 10228   PROCEND mmp$remove_page_from_job;
        0 10229
```


CHECK_FREE_QUEUES

```
        0 10232
        0 10233 {-------------------------------------------------------------------
        0 10234 {
        0 10235 {This procedure is called after processing a request to determine if
        0 10236 {the free queues need to be replenished. If so, a flag is set to cause
        0 10237 {CP Monitor to call Memory Manager.
        0 10238 {
        0 10239 {-------------------------------------------------------------------
        0 10240
        0 10241   PROCEDURE [INLINE] check_free_queues
        0 10242     (     cst_p: ^ost$cpu_state_table);
        0 10243
        0 10244     VAR
        0 10245       count: integer;
        0 10246
        0 10247     count := mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon;
        0 10248     IF count < mmv$aggressive_aging_level_2 THEN
        0 10249       jmp$recognize_thrashing;
        0 10250     IFEND;
        0 10251     IF count <= mmv$aggressive_aging_level THEN
        0 10252       IF count < mmv$aggressive_aging_level THEN
        0 10253         cst_p^.dispatch_control.asynchronous_interrupts_pending := TRUE;
        0 10254       IFEND;
        0 10255       mmv$time_to_call_mem_mgr := 0;
        0 10256       osv$time_to_check_asyn := 0;
        0 10257       mmv$aging_statistics.force_aggressive_aging := mmv$aging_statistics.force_aggressive_aging + 1;
        0 10258     IFEND;
        0 10259
        0 10260   PROCEND check_free_queues;
        0 10261
```

MMP$AGE_JOB_WORKING_SET

```
       0 10263
       0 10264 {-------------------------------------------------------------------------------------------------------------
       0 10265 {  This routine scan the page frames in the working set of a
       0 10266 {  job and updates the page ages, clears the page table 'USED' bits,
       0 10267 {  and removes unused pages from the working set of the job.
       0 10268 {-------------------------------------------------------------------------------------------------------------
       0 10269
       0 10270
       0 10271   PROCEDURE [XDCL] mmp$age_job_working_set
       0 10272     (     ijle_p: ^jmt$initiated_job_list_entry;
       0 10273           jcb_p: ^jmt$job_control_block);
       0 10274
       0 10275
       0 10276     VAR
       0 10277       fde_p: gft$file_desc_entry_p,
       0 10278       pqle_p: ^mmt$page_queue_list_entry,
       0 10279       i: integer,
       0 10280       cptime: integer,
       0 10281       pfti: mmt$page_frame_index,
       0 10282       perf,
       0 10283       link,
       0 10284       lu_link: mmt$link,
       0 10285       pfte_p,
       0 10286       lu_pfte_p: ^mmt$page_frame_table_entry,
       0 10287       mcount: integer,
       0 10288       rcount: integer,
       0 10289       aii,
       0 10290       aic,
       0 10291       aif: integer;
       0 10292
       0 10293     IF mmv$aging_algorithm >= 4 THEN
      18 10294       cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
      24 10295     ELSE
      24 10296       cptime := ijle_p^.statistics.cp_time.time_spent_in_mtr_mode +
      34 10297             ijle_p^.statistics.cp_time.time_spent_in_job_mode;
      34 10298     IFEND;
      34 10299     aii := (cptime - jcb_p^.cptime_next_age_working_set + jcb_p^.page_aging_interval) DIV
      34 10300           jcb_p^.page_aging_interval;
      34 10301     mmv$aging_statistics.calls_to_age_jws := mmv$aging_statistics.calls_to_age_jws + 1;
      34 10302     IF (aii < 1) THEN
      5C 10303       aii := 1;
      60 10304     IFEND;
      60 10305     jcb_p^.cptime_next_age_working_set := cptime + jcb_p^.page_aging_interval;
      60 10306     pqle_p := ^ijle_p^.job_page_queue_list [mmc$pq_job_working_set];
      60 10307
      60 10308
      60 10309 { Calculate the values of  AIC, and AIF to be used in processing this request.
      60 10310
      60 10311     aic := mmv$age_interval_ceiling;
      60 10312     aif := mmv$age_interval_floor;
      60 10313
      60 10314
      60 10315 { Age the job working set and relink the page frames into LRU order.
      60 10316
      60 10317     mmp$reset_store_pfti;
      60 10318     lu_link.bkw := 0;
```


MMP$AGE_JOB_WORKING_SET

```
      60 10319     link.bkw := pqle_p^.link.bkw;
      60 10320     rcount := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count - jcb_p^.min_working_set_size;
      60 10321
      60 10322     IF syv$perf_keypoints_enabled.aging_keypoints THEN
      B6 10323       perf := link;
      B6 10324       WHILE perf.bkw <> 0 DO
      C2 10325         pfte_p := ^mmv$pft_p^ [perf.bkw];
      C2 10326         perf := pfte_p^.link;
      C2 10327         gfp$mtr_get_fde_p (pfte_p^.aste_p^.sfid, ijle_p, fde_p);
     14C 10328         #KEYPOINT (osk$performance, osk$m * fde_p^.last_segment_number, ptk$aging_segment);
     15C 10329         #KEYPOINT (osk$performance, osk$m * (pfte_p^.sva.offset DIV osv$page_size), ptk$aging_page_number);
     174 10330       WHILEND;
     18A 10331       #KEYPOINT (osk$performance, osk$m * (ijle_p^.job_page_queue_list [mmc$pq_job_fixed].count),
     19A 10332             ptk$aging_job_fixed);
     19A 10333       #KEYPOINT (osk$performance, osk$m * (pfte_p^.ijl_ordinal.block_number *
     1B8 10334             32 + pfte_p^.ijl_ordinal.block_index), ptk$aging_ijl_ordinal);
     1B8 10335     IFEND;
     1B8 10336     IF syv$perf_keypoints_enabled.aging_stack_trace THEN
     1C0 10337       tmp$monitor_flag_job_tasks (syc$mf_for_keypoint_traceback, ijle_p);
     1DA 10338     IFEND;
     1DA 10339
     1DA 10340     WHILE (link.bkw <> 0) AND (rcount > 0) DO
     1EC 10341       pfte_p := ^mmv$pft_p^ [link.bkw];
     1EC 10342       pfti := link.bkw;
     1EC 10343       link := pfte_p^.link;
     1EC 10344       IF mmv$pt_p^ [pfte_p^.pti].u THEN
     228 10345         mmv$pt_p^ [pfte_p^.pti].u := FALSE;
     228 10346         pfte_p^.age := 0;
     228 10347         pfte_p^.cyclic_age := 0;
     228 10348         IF lu_link.bkw = 0 THEN
     23C 10349           lu_pfte_p := pfte_p;
     23C 10350           lu_link.fwd := link.fwd;
     23C 10351           lu_link.bkw := pfti;
     24C 10352         IFEND;
     250 10353       ELSEIF pfte_p^.locked_page <> mmc$lp_not_locked THEN
     25C 10354
     25C 10355 { Do nothing
     25C 10356
     25C 10357       ELSEIF ((pfte_p^.age + aii) > aic) THEN
     266 10358         mmp$store_pfti (pfti);
     266 10359         rcount := rcount - 1;
     294 10360       ELSEIF ((pfte_p^.age + aii) > aif) THEN
     298 10361         mmp$store_pfti (pfti);
     298 10362         rcount := rcount - 1;
     298 10363         mmv$aging_statistics.age_exceeds_aif := mmv$aging_statistics.age_exceeds_aif + 1;
     298 10364         aif := 65536; {Only remove one page for age > AIF}
     2D4 10365       ELSE
     2D4 10366         pfte_p^.age := pfte_p^.age + aii;
     2D4 10367         IF (lu_link.bkw <> 0) THEN
     2E0 10368           IF link.bkw = 0 THEN
     2E8 10369             pqle_p^.link.fwd := link.fwd;
     2F4 10370           ELSE
     2F4 10371             mmv$pft_p^ [link.bkw].link.fwd := link.fwd;
     308 10372           IFEND;
     308 10373           mmv$pft_p^ [link.fwd].link.bkw := link.bkw;
     308 10374           pfte_p^.link := lu_link;
```

MMP$AGE_JOB_WORKING_SET

```
      308 10375            IF lu_link.fwd = 0 THEN
      336 10376              pqle_p^.link.bkw := pfti;
      33E 10377            ELSE
      33E 10378              mmv$pft_p^ [lu_link.fwd].link.bkw := pfti;
      354 10379            IFEND;
      354 10380            lu_pfte_p^.link.fwd := pfti;
      354 10381            lu_link.fwd := pfti;
      35E 10382          IFEND;
      35E 10383        IFEND;
      35E 10384      WHILEND;
      370 10385
      370 10386
      370 10387 { If any pages have been selected for removal, remove the pages from the working set.
      370 10388
      370 10389      mmp$fetch_pfti_array_size (rcount);
      370 10390      mcount := 0;
      370 10391      IF rcount > 0 THEN
      38E 10392        mmp$remove_pages_from_jws (mmc$pq_avail_modified, ijle_p, mcount, rcount);
      3B8 10393        mmv$aging_statistics.age_exceeds_aic := mmv$aging_statistics.age_exceeds_aic + rcount;
      3CA 10394      ELSE
      3CA 10395        mmp$purge_all_page_map;
      3E4 10396      IFEND;
      3E4 10397      IF syv$perf_keypoints_enabled.aging_keypoints THEN
      3EC 10398        #KEYPOINT (osk$performance, osk$m * mcount, ptk$aging_modified_pages);
      3F8 10399        #KEYPOINT (osk$performance, osk$m * rcount, ptk$aging_pages_removed);
      404 10400      IFEND;
      404 10401
      404 10402    PROCEND mmp$age_job_working_set;
```


MMP$REMOVE_STALE_PAGES

```
        0 10404
        0 10405 {-------------------------------------------------------------------------------------------------------
        0 10406 {
        0 10407 { This procedure is called to remove stale pages from a page queue. A stale page is defined as a page
        0 10408 { that has a 'SWAP_COUNT' (field should be renamed) greater or equal to the value specified by the caller.
        0 10409 {
        0 10410 { This procedure does the following:
        0 10411 {     . Scan each page in the page queue
        0 10412 {     . If the 'u' bit in the page table is set
        0 10413 {         clear it if aging the shared queue (if aging because of swapping,dont clear it - this would defeat
        0 10414 {             the page aging algorithms.
        0 10415 {     . ELSE if the swap count < stale count, increment swap count
        0 10416 {     . ELSE remove the page from the page queue (queue_id is passed from the caller to indicate if page goes
        0 10417 {         to AM or JWS queue (used for swap aging))
        0 10418 {
        0 10419 { This procedure is intended to be used to:
        0 10420 {     . age the shared queue
        0 10421 {     . provide SWAPPING_AIC aging of job working sets prior to swap.
        0 10422 {
        0 10423 { This procedure does NOT keep the page queue in a LRU order.
        0 10424 {
        0 10425 {-------------------------------------------------------------------------------------------------------
        0 10426
        0 10427
        0 10428   PROCEDURE [XDCL] mmp$remove_stale_pages
        0 10429     (VAR pqle: mmt$page_queue_list_entry;
        0 10430          limit: integer;
        0 10431          jcb_p: ^jmt$job_control_block;
        0 10432          ijle_p: ^jmt$initiated_job_list_entry;
        0 10433          queue_id: mmt$page_frame_queue_id;
        0 10434          minimum_working_set: 0 .. 0ffff(16);
        0 10435      VAR modified_pages_removed: integer;
        0 10436      VAR total_pages_removed: integer);
        0 10437
        0 10438      VAR
        0 10439        cptime: integer,
        0 10440        eoi: ost$segment_length,
        0 10441        lstatus: syt$monitor_status,
        0 10442        rcount: integer,
        0 10443        pfti: mmt$page_frame_index,
        0 10444        end_pfti: mmt$page_frame_index,
        0 10445        next_pfti: mmt$page_frame_index,
        0 10446        mmv$age_not_pageable: [XDCL] integer := 0,
        0 10447        pfte_p: ^mmt$page_frame_table_entry;
        0 10448
        0 10449      IF jcb_p <> NIL THEN
       12 10450        jcb_p^.next_cyclic_aging_time := #FREE_RUNNING_CLOCK (0) + jcb_p^.cyclic_aging_interval;
       1A 10451        IF mmv$aging_algorithm >= 4 THEN
       36 10452          cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
       42 10453        ELSE
       42 10454          cptime := ijle_p^.statistics.cp_time.time_spent_in_mtr_mode +
       52 10455                ijle_p^.statistics.cp_time.time_spent_in_job_mode;
       52 10456        IFEND;
       52 10457        jcb_p^.cptime_next_age_working_set := cptime + jcb_p^.page_aging_interval;
       5C 10458      IFEND;
       5C 10459
```

MMP$REMOVE_STALE_PAGES

```
5C 10460      mmp$reset_store_pfti;
5C 10461      pfti := pqle.link.bkw;
5C 10462      end_pfti := pqle.link.fwd;
5C 10463      rcount := pqle.count - minimum_working_set;
5C 10464
5C 10465      WHILE (pfti <> end_pfti) AND (rcount > 0) DO
9A 10466        pfte_p := ^mmv$pft_p^ [pfti];
9A 10467        next_pfti := pfte_p^.link.bkw;
9A 10468        IF mmv$pt_p^ [pfte_p^.pti].u AND NOT jsv$free_working_set_on_swapout THEN
DC 10469          mmv$pt_p^ [pfte_p^.pti].u := FALSE;
DC 10470          pfte_p^.cyclic_age := 0;
DC 10471          pfte_p^.age := 0;
EC 10472        ELSEIF pfte_p^.cyclic_age < limit THEN
F8 10473          pfte_p^.cyclic_age := pfte_p^.cyclic_age + 1;
102 10474       ELSEIF pfte_p^.locked_page <> mmc$lp_not_locked THEN
10E 10475
10E 10476 { Do nothing
10E 10477
10E 10478       ELSE
10E 10479         mmp$store_pfti (pfti);
10E 10480         rcount := rcount - 1;
134 10481       IFEND;
134 10482
134 10483       pfti := next_pfti;
134 10484     WHILEND;
14E 10485
14E 10486
14E 10487 { If any pages have been selected for removal, remove the pages from the working set.
14E 10488
14E 10489     mmp$fetch_pfti_array_size (rcount);
14E 10490     IF rcount > 0 THEN
164 10491       mmp$remove_pages_from_jws (queue_id, ijle_p, modified_pages_removed, total_pages_removed);
192 10492     ELSE
192 10493       total_pages_removed := 0;
192 10494       modified_pages_removed := 0;
192 10495       IF queue_id = mmc$pq_avail_modified THEN
1AC 10496         mmp$purge_all_page_map;
1C8 10497       IFEND;
1C8 10498     IFEND;
1C8 10499
1C8 10500
1C8 10501   PROCEND mmp$remove_stale_pages;
```


MMP$TRIM_JOB_WORKING_SET

```
0 10503
0 10504 {-------------------------------------------------------------------------------------------------
0 10505 { This procedure is called to trim a job working set.
0 10506 { If the size of the working set exceeds the max allowed, pages are removed until the size is ok.
0 10507 {-------------------------------------------------------------------------------------------------
0 10508
0 10509   PROCEDURE [XDCL] mmp$trim_job_working_set
0 10510     (     ijle_p: ^jmt$initiated_job_list_entry;
0 10511           jcb_p: ^jmt$job_control_block;
0 10512           trim_to_swap_size: boolean);
0 10513
0 10514     VAR
0 10515       pfti: mmt$page_frame_index,
0 10516       last_pfti: mmt$page_frame_index,
0 10517       pte_p: ^ost$page_table_entry,
0 10518       maximum_pages_to_swap: integer,
0 10519       mcount: integer,
0 10520       rcount: integer,
0 10521       smallest_maximum_working_set: integer,
0 10522       pfte_p: ^mmt$page_frame_table_entry;
0 10523
0 10524     IF syv$recovering_job_count <> 0 THEN
10 10525       RETURN;
12 10526     IFEND;
12 10527
12 10528
12 10529     IF (jcb_p^.max_working_set_size < mmv$max_working_set_size) THEN
22 10530       smallest_maximum_working_set := jcb_p^.max_working_set_size;
26 10531     ELSE
26 10532       smallest_maximum_working_set := mmv$max_working_set_size;
28 10533     IFEND;
28 10534
28 10535     IF trim_to_swap_size THEN
30 10536       IF ijle_p^.task_created_after_last_swap THEN
3C 10537         maximum_pages_to_swap := jsv$max_pages_first_swap_task;
48 10538       ELSE
48 10539         maximum_pages_to_swap := jsv$maximum_pages_to_swap;
50 10540       IFEND;
50 10541
50 10542       IF smallest_maximum_working_set > maximum_pages_to_swap THEN
54 10543         smallest_maximum_working_set := maximum_pages_to_swap;
58 10544       IFEND;
58 10545     IFEND;
58 10546
58 10547     pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.bkw;
58 10548     WHILE (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count > smallest_maximum_working_set) AND
6C 10549           (pfti <> 0) DO
6C 10550       pfte_p := ^mmv$pft_p^ [pfti];
6C 10551       pte_p := ^mmv$pt_p^ [pfte_p^.pti];
6C 10552       last_pfti := pfti;
6C 10553       pfti := pfte_p^.link.bkw;
6C 10554       mmp$remove_page_from_jws (last_pfti, ijle_p, mcount, rcount);
C2 10555     WHILEND;
CE 10556
CE 10557   PROCEND mmp$trim_job_working_set;
```

```
      0 10560
      0 10561 {-------------------------------------------------------------------------------------------------------
      0 10562 { This procedure is called to take pages out of the shared queue.  Pages will be removed until
      0 10563 { mmv$reassignable_page_frames.now minus mmv$aggressive_aging_level_2 is greater than
      0 10564 { the number of pages requested.
      0 10565 {
      0 10566 { The removal of pages from the shared queues will be done in two passes.  On the first pass a number of
      0 10567 { pages will be removed from each shared queue as determined by the minimum size attribute of the queue.
      0 10568 { If the minimum is zero, then all of the pages in that queue will be removed during the first pass.
      0 10569 { If the first pass does not remove enough pages, then another pass will be made during which all pages
      0 10570 { can be removed if necessary.  The passes are terminated early whenever enough pages have been removed.
      0 10571 {-------------------------------------------------------------------------------------------------------
      0 10572
      0 10573   PROCEDURE [XDCL] mmp$dump_shared_queue
      0 10574     (    total_pages_needed: mmt$page_frame_index);
      0 10575
      0 10576     VAR
      0 10577       modified_pages_removed: integer,
      0 10578       next_pfti: mmt$page_frame_index,
      0 10579       pages_from_queue: integer,
      0 10580       pages_removed: integer,
      0 10581       pfti: mmt$page_frame_index,
      0 10582       reduce_queue_below_minimum: boolean,
      0 10583       queue_id: mmt$page_frame_queue_id;
      0 10584
      0 10585     reduce_queue_below_minimum := FALSE;
      4 10586
      4 10587   /two_passes/
      4 10588     WHILE 1 = 1 DO
      8 10589       FOR queue_id := mmv$last_active_shared_queue DOWNTO mmc$pq_shared_first DO
     1C 10590         pfti := mmv$gpq1 [queue_id].pq1e.link.bkw;
     1C 10591         IF reduce_queue_below_minimum THEN
     28 10592           pages_from_queue := mmv$gpq1 [queue_id].pq1e.count;
     30 10593         ELSE
     30 10594           pages_from_queue := mmv$gpq1 [queue_id].pq1e.count - mmv$gpq1 [queue_id].minimum;
     3A 10595         IFEND;
     3A 10596
     3A 10597 { If the count of pages in the queue is less than or equal to the minimum, one of the pages will be
     3A 10598 { removed on the first pass.  If there are no pages in the queue, pfti will be zero.
     3A 10599
     3A 10600       /dump_a_queue/
     3A 10601         WHILE pfti <> 0 DO
     42 10602           next_pfti := mmv$pft_p^ [pfti].link.bkw;
     42 10603           mmp$remove_page_from_jws (pfti, NIL, modified_pages_removed, pages_removed);
     82 10604           pfti := next_pfti;
     82 10605           pages_from_queue := pages_from_queue - 1;
     82 10606           IF pages_from_queue <= 0 THEN
     8A 10607             EXIT /dump_a_queue/
     8E 10608           IFEND;
     8E 10609           IF (mmv$reassignable_page_frames.now - mmv$aggressive_aging_level_2) >= total_pages_needed THEN
     A0 10610             EXIT /two_passes/ {Terminate both passes since the pages needed are available.
     A2 10611           IFEND;
     A2 10612         WHILEND /dump_a_queue/;
     A8 10613       FOREND;
     AE 10614
     AE 10615       IF reduce_queue_below_minimum THEN
```



```
     B2 10616         EXIT /two_passes/; { exit, All the shared queues have been dumped including the minimums.
     B4 10617       IFEND;
     B4 10618       reduce_queue_below_minimum := TRUE; {Allow queues to be reduced below the minimum size on pass 2.
     B4 10619     WHILEND /two_passes/;
     BC 10620
     BC 10621
     BC 10622   PROCEND mmp$dump_shared_queue;
      0 10623
```

```
      0 10626
      0 10627 {-------------------------------------------------------------------------------------------------------------
      0 10628 { This procedure is called to assign a new page frame to a segment.
      0 10629 { The routine performs the following steps:
      0 10630 {     . obtain a free page frame.
      0 10631 {     . delete the PT entry using the page frame (if necessary).
      0 10632 {     . make a new PT entry for the page.
      0 10633 {     . update the PFT entry for the page frame and the AST entry for the seg.
      0 10634 {
      0 10635 {       MMP$ASSIGN_PAGE_FRAME (SVA, ASTE_P, NUMBER_OF_PAGES_TO_ASSIGN, STARTING_PFTI,
      0 10636 {          ASSIGNED_PAGE_COUNT, PFTI, PSTATUS);
      0 10637 {
      0 10638 {  SVA: (INPUT) SVA that identifies page
      0 10639 {  ASTE_P: (INPUT) Pointer to AST table entry for the segment
      0 10640 {  NUMBER_OF_PAGES_TO_ASSIGN: (INPUT) This parameter specifies how many pages
      0 10641 {        the caller wants assigned.
      0 10642 {  STARTING_PFTI: (INPUT) This parameter specifies the pfti where page assignment is to begin.
      0 10643 {        This parameter will be non-zero only if the request for page assignment is coming
      0 10644 {        from the ASSIGN_CONTIGUOUS_MEMORY request.
      0 10645 {  ASSIGNED_PAGE_COUNT: (OUTPUT) Number of pages actually assigned. May be less than requested if a
      0 10646 {        page already exists in the specified range.
      0 10647 {  FIRST_PFTI: (OUTPUT) Page Frame Table index of first page frame assigned.  If more
      0 10648 {        than one page assigned the other pages are linked through the
      0 10649 {        backward link in the page frame table entry.
      0 10650 {  PSTATUS: (OUTPUT) Status
      0 10651 {            ps_done - if all pages were assigned
      0 10652 {            ps_no_memory - if insufficient memory is available to assign ALL requested
      0 10653 {            ps_pt_full - if page table full. Some pages may have been assigned before page table full
      0 10654 {               occurred; ASSIGNED_PAGE_COUNT will indicate how many pages were assigned.
      0 10655 {            ps_valid_in_pt - if a page is in PT. Some pages may have been assigned before valid in page
      0 10656 {               table occurred; ASSIGNED_PAGE_COUNT will inicate how many pages were assigned.
      0 10657 {-------------------------------------------------------------------------------------------------------------
      0 10658
      0 10659   PROCEDURE [XDCL] mmp$assign_page_frame
      0 10660     (     sva: ost$system_virtual_address;
      0 10661           aste_p: ^mmt$active_segment_table_entry;
      0 10662           number_of_pages_to_assign: mmt$page_frame_index;
      0 10663           starting_pfti: mmt$page_frame_index;
      0 10664       VAR assigned_page_count: mmt$page_frame_index;
      0 10665       VAR first_pfti: mmt$page_frame_index;
      0 10666       VAR pstatus: mmt$page_pull_status);
      0 10667
      0 10668       VAR
      0 10669         assign_page_loop_count: mmt$page_frame_index,
      0 10670         page_sva: ost$system_virtual_address,
      0 10671         pfti: mmt$page_frame_index,
      0 10672         pfte_p: ^mmt$page_frame_table_entry,
      0 10673         mpt_status: mmt$make_pt_entry_status;
      0 10674
      0 10675
      0 10676
      0 10677       assigned_page_count := 0;
      4 10678       first_pfti := starting_pfti;
      4 10679       IF first_pfti <> 0 THEN
     1E 10680         pfti := starting_pfti - 1;
     24 10681       IFEND;
```



```
     24 10682       IF number_of_pages_to_assign > mmv$reassignable_page_frames.now THEN
     34 10683         pstatus := ps_no_memory;
     34 10684         RETURN;
     40 10685       IFEND;
     40 10686
     40 10687       pstatus := ps_done;
     40 10688       page_sva := sva;
     40 10689       assign_page_loop_count := number_of_pages_to_assign;
     40 10690
     40 10691       WHILE assign_page_loop_count > 0 DO
     54 10692
     54 10693 {Get an available page frame to use for the new page.  Return an error code if no memory is available.
     54 10694
     54 10695         IF starting_pfti = 0 THEN
     5C 10696           mmp$get_avail_page_frame (pfti);
     D0 10697           IF pfti = 0 THEN
     D8 10698             mtp$error_stop ('MM - no mem for assign_page');
     F8 10699           IFEND;
     FC 10700         ELSE
     FC 10701
     FC 10702 {The non-zero starting_pfti indicates that the request to assign page frames is
     FC 10703 {coming from an ASSIGN_CONTIGUOUS_MEMORY request. That request has verified
     FC 10704 {that the page frames from (starting_pfti-->number_pages_to_assign) are available.
     FC 10705
     FC 10706           pfti := pfti + 1;
    106 10707         IFEND;
    106 10708         pfte_p := ^mmv$pft_p^ [pfti];
    106 10709
    106 10710
    106 10711 {Make a PT entry for the new page. If page table was full, link the page frame back to the free queue.
    106 10712
    106 10713         mmp$make_pt_entry (page_sva, pfti, aste_p, pfte_p, mpt_status);
    14C 10714         IF mpt_status <> mmc$mpt_done THEN
    154 10715           mmp$relink_page_frame (pfti, mmc$pq_free);
    16C 10716           IF mpt_status = mmc$mpt_page_table_full THEN
    176 10717             mmv$async_work.pt_full_aste_p := aste_p;
    176 10718             mmv$async_work.pt_full_sva := page_sva;
    176 10719             mmv$async_work.pt_full := TRUE;
    176 10720             mmv$time_to_call_mem_mgr := 0;
    176 10721             osv$time_to_check_asyn := 0;
    176 10722             pstatus := ps_pt_full;
    19A 10723           ELSE {must be valid in PT - make sure no other statuses}
    19A 10724             pstatus := ps_valid_in_pt;
    1A0 10725           IFEND;
    1A0 10726           RETURN;
    1A2 10727         IFEND;
    1A2 10728
    1A2 10729
    1A2 10730 {Update the page frame table entry for the new entry.
    1A2 10731
    1A2 10732         IF pfte_p^.task_queue.head <> 0 THEN
    1AA 10733           mtp$error_stop ('MM - reassigned PF with task queue');
    1CA 10734         IFEND;
    1CA 10735         pfte_p^.age := 0;
    1CA 10736         pfte_p^.cyclic_age := 0;
    1CA 10737         pfte_p^.io_error := ioc$no_error;
```

MMP$ASSIGN_PAGE_FRAME

```
1CA 10738         pfte_p^.sva := page_sva;
1CA 10739         pfte_p^.aste_p := aste_p;
1CA 10740         pfte_p^.locked_page := mmc$lp_not_locked;
1CA 10741         pfte_p^.ijl_ordinal := aste_p^.ijl_ordinal;
1CA 10742
1CA 10743
1CA 10744 [Link the page frame into the new queue.
1CA 10745
1CA 10746         mmp$relink_page_frame (pfti, aste_p^.queue_id);
20E 10747
20E 10748         IF first_pfti = 0 THEN
216 10749           first_pfti := pfti;
21E 10750         IFEND;
21E 10751
21E 10752         assigned_page_count := assigned_page_count + 1;
21E 10753         assign_page_loop_count := assign_page_loop_count - 1;
21E 10754         IF assign_page_loop_count > 0 THEN
22E 10755           page_sva.offset := page_sva.offset + osv$page_size;
242 10756         IFEND;
242 10757       WHILEND;
246 10758
246 10759
246 10760     PROCEND mmp$assign_page_frame;
```


MMP$PAGE_PULL

```
 0 10763
 0 10764 {-------------------------------------------------------------------------------------------------------------
 0 10765 {  This procedure is called to make a page of a segment accessible in
 0 10766 {  the address space of the current user task.
 0 10767 {  For the File Server project, the CPU state table pointer ( cst_p ) will
 0 10768 {  be passed as NIL when called (asynchronously) by the server procedure
 0 10769 {  PROCESS_READ_FOR_SERVER.  Also, the stxe_p is NIL in this case.
 0 10770 {-------------------------------------------------------------------------------------------------------------
 0 10771
 0 10772 {! make sure all possible status values are in the CASE statement in pr_pf.
 0 10773
 0 10774
 0 10775
 0 10776   PROCEDURE [XDCL] mmp$page_pull
 0 10777     (     xsva: ost$system_virtual_address;
 0 10778           fde_p: gft$locked_file_desc_entry_p;
 0 10779           cst_p: ^ost$cpu_state_table;
 0 10780           aste_p: ^mmt$active_segment_table_entry;
 0 10781           stxe_p: ^mmt$segment_descriptor_extended;
 0 10782           io_id: mmt$io_identifier;
 0 10783           pages_to_read: integer;
 0 10784           io_function: iot$io_function;
 0 10785           allocate_if_new: boolean;
 0 10786       VAR page_count: mmt$page_frame_index;
 0 10787       VAR pstatus: mmt$page_pull_status;
 0 10788       VAR pfti: mmt$page_frame_index);
 0 10789
 0 10790     VAR
 0 10791       active_au_offset: integer,
 0 10792       assigned_page_count: mmt$page_frame_index,
 0 10793       bytes_to_read: integer,
 0 10794       buffer_descriptor: mmt$buffer_descriptor,
 0 10795       sva: ost$system_virtual_address,
 0 10796       file_limits_enforced: sft$file_space_limit_kind,
 0 10797       file_kind: gft$file_kind,
 0 10798       low_on_page_frames: boolean,
 0 10799       next_pfti: mmt$page_frame_index,
 0 10800       shadow_au_offset: integer,
 0 10801       page_status: gft$page_status,
 0 10802       pages_to_allocate: integer,
 0 10803       passive_fde_p: gft$locked_file_desc_entry_p,
 0 10804       ijlo: jmt$ijl_ordinal,
 0 10805       status: syt$monitor_status,
 0 10806       update_eoi_reason: mmt$update_eoi_reason;
 0 10807
 0 10808     page_count := 0;
 4 10809     pfti := 0;
 4 10810     sva := xsva;
 4 10811     sva.offset := (sva.offset DIV osv$page_size) * osv$page_size;
 4 10812
 4 10813 { Previously  page_pull first called #HASH_SVA to determine if the page was already in the page table.
 4 10814 { If found the valid bit was checked.  If valid a status of ps_valid_in_pt was returned.  If not valid,
 4 10815 { the page was checked for IO activity and a status of ps_locked was returned if it was locked.  Finally,
 4 10816 { If not locked for IO, the page was moved to the JWS from one of the available queues.  Now the HASH is done
 4 10817 { via the inline proc mmp$page_pull_hash_sva which must be called before calling mmp$page_pull.
 4 10818
```

```
        4 10819
        4 10820 {A new page frame is required. If the system is running low on memory and the requesting task is not a
        4 10821 {system task, reject the request. This will cause the user to be put in a WAIT state.
        4 10822 {For a served file fault, just check low on memory.
        4 10823
        4 10824     low_on_page_frames := mmv$reassignable_page_frames.now < mmv$aggressive_aging_level_2;
        4 10825     IF low_on_page_frames AND ((cst_p = NIL) OR (cst_p^.xcb_p^.system_table_lock_count < 256) AND
       70 10826           NOT cst_p^.xcb_p^.critical_task) THEN
       70 10827       pstatus := ps_low_on_memory;
       70 10828       RETURN;
       7C 10829     IFEND;
       7C 10830
       7C 10831
       7C 10832 { Check for reference beyond EOI if user does not have EXTEND permission.
       7C 10833
       7C 10834     IF (sva.offset >= fde_p^.eoi_byte_address) AND (cst_p <> NIL) AND
       B6 10835           ((stxe_p^.access_rights <> mmc$sar_write_extend) OR NOT allocate_if_new) THEN
       B6 10836       IF stxe_p^.access_rights = mmc$sar_modify THEN
       C0 10837         pstatus := ps_no_extend_permission;
       CC 10838       ELSE
       CC 10839         pstatus := ps_read_beyond_eoi;
       D6 10840       IFEND;
       D6 10841       RETURN;
       D8 10842     IFEND;
       D8 10843
       D8 10844
       D8 10845 { Check for reference beyond file limit. Note: during deadstart, a reference beyond
       D8 10846 { EOI may be for the memory resident portion of the old image file.
       D8 10847
       D8 10848     IF sva.offset >= fde_p^.file_limit THEN
       EA 10849       IF (mmv$image_file.active) AND (aste_p^.sfid = mmv$image_file.sfid) THEN
*WARN*    10850         process_memory_image_pf (sva, aste_p, pfti, pstatus);
      128 10851         page_count := 1;
      132 10852       ELSE
      132 10853         pstatus := ps_beyond_file_limit;
      132 10854         RETURN;
      13E 10855       IFEND;
      140 10856     IFEND;
      140 10857
      140 10858
      140 10859 { Determine limits options. Served files always have a NIL cst_p and require no
      140 10860 { limits checking.
      140 10861
      140 10862     IF cst_p = NIL THEN
      14E 10863       ijlo := jmv$null_ijl_ordinal;
      14E 10864       file_limits_enforced := sfc$no_limit;
      15C 10865     ELSE
      15C 10866       ijlo := cst_p^.ijl_ordinal;
      15C 10867       file_limits_enforced := stxe_p^.file_limits_enforced;
      16C 10868     IFEND;
      16C 10869
      16C 10870
      16C 10871 { Determine the status/location of the page.
      16C 10872
      16C 10873     CASE fde_p^.media OF
      184 10874     = gfc$fm_transient_segment =
```

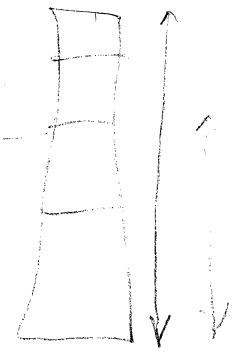


```
      184 10875       IF (aste_p^.pages_in_memory > mmv$max_pages_no_file) AND NOT
      1A6 10876           ((aste_p^.queue_id = mmc$pq_wired) OR (aste_p^.queue_id = mmc$pq_job_fixed)) THEN
      1A6 10877         set_assign_active (stxe_p, sva.offset);
      1D4 10878         tmp$set_monitor_flag (cst_p^.taskid, mmc$mf_segment_mgr_flag, status);
      1FA 10879       IFEND;
      1FA 10880       page_status := gfc$ps_page_doesnt_exist;
      202 10881     = gfc$fm_mass_storage_file =
      202 10882       IF cst_p <> NIL THEN
      20C 10883         mmv$last_segment_accessed := (#OFFSET (#LOC (stxe_p^)) - cst_p^.xcb_p^.sdtx_offset) DIV
      22C 10884               #SIZE (mmt$segment_descriptor_extended);
      22C 10885       IFEND;
      22C 10886       dmp$fetch_page_status (fde_p, sva.offset, file_limits_enforced, allocate_if_new, page_status);
      266 10887     = gfc$fm_served_file =
      266 10888       dfp$fetch_page_status (fde_p, sva.offset, page_status);
      294 10889     ELSE
      294 10890       mtp$error_stop ('MM - bad FDE.MEDIA');
      2B4 10891     CASEND;
      2B4 10892
      2B4 10893
      2B4 10894 { If job mode work is required but the task is in some state where it is not advisable
      2B4 10895 { to interrupt it, allow escaped allocation to occur. Otherwise reject the page fault
      2B4 10896 { and let the task fix the problem in job mode before assigning the page.
      2B4 10897
      2B4 10898     IF (page_status = gfc$ps_job_mode_work_required) AND (cst_p <> NIL) THEN
      2C8 10899       set_assign_active (stxe_p, sva.offset);
      2F6 10900       IF (cst_p^.xcb_p^.system_table_lock_count > 255) AND (cst_p^.xcb_p^.xp.p_register.pva.ring > 1) THEN
      318 10901         cst_p^.xcb_p^.stlc_allocation := TRUE;
      320 10902       ELSE
      320 10903         tmp$set_monitor_flag (cst_p^.taskid, mmc$mf_segment_mgr_flag, status);
      344 10904       IFEND;
      344 10905       IF (cst_p^.xcb_p^.xp.trap_enable <> osc$traps_enabled) OR (cst_p^.xcb_p^.xp.p_register.pva.ring = 1) OR
      37E 10906           (osc$trap_exception IN cst_p^.xcb_p^.xp.monitor_condition_register) OR
      37E 10907           (fde_p^.stack_for_ring <> 0) OR (cst_p^.xcb_p^.system_table_lock_count > 255) THEN
      37E 10908         page_status := gfc$ps_page_doesnt_exist;
      382 10909       IFEND;
      382 10910     IFEND;
      382 10911
      382 10912
      382 10913 {Process page fault depending on the location of the page.
      382 10914
      382 10915     CASE page_status OF
      3D4 10916     = gfc$ps_page_on_disk, gfc$ps_page_on_server =
      3D4 10917       IF NOT low_on_page_frames THEN
      3D8 10918         bytes_to_read := (fde_p^.allocation_unit_size - (sva.offset MOD fde_p^.allocation_unit_size));
      3D8 10919         IF sva.offset + bytes_to_read > fde_p^.eoi_byte_address THEN
      3FE 10920           bytes_to_read := fde_p^.eoi_byte_address - sva.offset + osv$page_size - 1;
      40C 10921         IFEND;
      40C 10922         page_count := bytes_to_read DIV osv$page_size;
      40C 10923         IF page_count > pages_to_read THEN
      420 10924           page_count := pages_to_read;
      424 10925         IFEND;
      428 10926       ELSE
      428 10927         page_count := 1;
      42C 10928       IFEND;
      42C 10929
      42C 10930       read_pages_from_disk_or_server (io_function, fde_p, sva, page_count, aste_p, sva.offset, io_id, FALSE,
```

```
46A 10931              pstatus, pfti);
46A 10932
46A 10933      = gfc$ps_job_mode_work_required =
46A 10934        mmp$update_eoi (fde_p, sva.offset, mmc$uer_page_assigned);  {Needed so job mode knows adr to allocate}
494 10935        pstatus := ps_job_work_required;
4A2 10936
4A2 10937      = gfc$ps_volume_unavailable =
4A2 10938        IF fde_p^.flags.wire_eoi_page AND (sva.offset >= fde_p^.eoi_byte_address) THEN
4BE 10939          mmp$assign_page_frame (sva, aste_p, 1, 0, page_count, pfti, pstatus);
4EE 10940          IF pstatus = ps_done THEN
4F6 10941            mmp$preset_real_memory (sva, fde_p^.preset_value);
50E 10942            mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := TRUE;
50E 10943            pstatus := ps_new_page_assigned;
50E 10944            mmp$update_eoi (fde_p, sva.offset, mmc$uer_page_assigned);
562 10945          IFEND;
566 10946        ELSE
566 10947          pstatus := ps_volume_unavailable;
570 10948        IFEND;
572 10949
572 10950      = gfc$ps_server_allocate_required =
572 10951
572 10952        mmp$assign_page_frame (sva, aste_p, 1, 0, page_count, pfti, pstatus);
5A4 10953        IF pstatus = ps_done THEN
5AC 10954          mmp$preset_real_memory (sva, fde_p^.preset_value);
5C4 10955          pstatus := ps_allocate_required_on_server;
5C4 10956          IF pages_to_read < mmv$pages_for_overallocation THEN
5D8 10957            pages_to_allocate := mmv$pages_for_overallocation;
5DC 10958          ELSE
5DC 10959            pages_to_allocate := pages_to_read;
5DE 10960          IFEND;
5DE 10961          buffer_descriptor.buffer_descriptor_type := mmc$bd_paging_io;
5DE 10962          buffer_descriptor.sva := sva;
5DE 10963          buffer_descriptor.page_count := page_count;
5DE 10964          dfp$file_server_allocation (aste_p^.sfid, sva.offset, (pages_to_allocate * osv$page_size) +
644 10965                sva.offset, io_id, buffer_descriptor, stxe_p^.file_limits_enforced, status);
644 10966          IF status.normal THEN
64C 10967            mmp$update_eoi (fde_p, sva.offset, mmc$uer_page_assigned);
66C 10968            RETURN; {<----}
672 10969          ELSEIF (status.condition = dme$transient_error) OR (status.condition = ioe$requests_full) THEN
68A 10970            pstatus := ps_io_temp_reject;
694 10971          ELSEIF status.condition = ioe$unit_disabled THEN
69C 10972            pstatus := ps_volume_unavailable;
6A6 10973          ELSEIF status.condition = dfe$server_has_terminated THEN
6AE 10974            pstatus := ps_server_terminated;
6BA 10975          ELSE
6BA 10976            mtp$error_stop ('MM - unexpected phy io error');
6D6 10977          IFEND;
6D6 10978          mmp$delete_pt_entry (pfti, TRUE);
6EE 10979          mmp$relink_page_frame (pfti, mmc$pq_free);
706 10980        IFEND;
708 10981
708 10982      = gfc$ps_temp_reject, gfc$ps_account_limit_exceeded =
708 10983        pstatus := ps_io_temp_reject;
714 10984
714 10985      = gfc$ps_server_terminated =
714 10986        pstatus := ps_server_terminated;
```



```
722 10987
722 10988      = gfc$ps_page_doesnt_exist =
722 10989
722 10990 { For page not on server, the file cannot be shadowed.  This applies where
722 10991 { the active file is on the server.
722 10992 { The pointer to the CPU_STATE_TABLE will always be non-nil on the client side of file_server.
722 10993
722 10994        IF (stxe_p = NIL) OR (stxe_p^.shadow_info.shadow_segment_kind = mmc$ssk_none) OR
750 10995              (sva.offset >= (stxe_p^.shadow_info.shadow_length_page_count * osv$page_size)) THEN
750 10996
750 10997 { Determine if multiple pages should be assigned for this "new page" page fault.
750 10998 { The numbers 32768 and 16384 are arbitrary.  Files less then 32768 are not as likely
750 10999 { to use the extra assigned pages.  Offset MOD 16384 is because most allocation units
750 11000 { are 16384.  This fault would then probably be for the first page in the AU.
750 11001
750 11002          pages_to_allocate := 1;
750 11003
750 11004          IF (sva.offset >= 32768) AND ((sva.offset MOD 16384) = 0) AND
7B6 11005                (fde_p^.media <> gfc$fm_transient_segment) AND
7B6 11006                (fde_p^.file_kind <> gfc$fk_device_file) AND
7B6 11007                (mmv$reassignable_page_frames.now > mmv$assign_multiple_pages) AND
7B6 11008                (mmv$pages_per_new_page_fault > 1) AND
7B6 11009
7B6 11010 { Check if assignment of extra pages will fit into allocated space.
7B6 11011
7B6 11012          (fde_p^.allocation_unit_size - (sva.offset MOD fde_p^.allocation_unit_size) >=
7B6 11013                mmv$pages_per_new_page_fault * osv$page_size) THEN
7B6 11014            pages_to_allocate := mmv$pages_per_new_page_fault;
7BA 11015          IFEND;
7BA 11016
7BA 11017          mmp$assign_page_frame (sva, aste_p, pages_to_allocate, 0, page_count, pfti, pstatus);
7EA 11018
7EA 11019          IF page_count > 0 THEN
7F2 11020            next_pfti := pfti;
7F2 11021            WHILE next_pfti <> 0 DO
7FA 11022              IF (fde_p^.stack_for_ring = 0) OR (fde_p^.stack_for_ring > 3) THEN
808 11023                mmp$preset_real_memory (sva, fde_p^.preset_value);
820 11024              IFEND;
820 11025              mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].v := TRUE;
820 11026              next_pfti := mmv$pft_p^ [next_pfti].link.bkw;
820 11027              IF next_pfti <> 0 THEN
854 11028                sva.offset := sva.offset + osv$page_size;
868 11029              IFEND;
868 11030            WHILEND;
86C 11031            pstatus := ps_new_page_assigned;
86C 11032            IF pages_to_allocate > 1 THEN
87A 11033              update_eoi_reason := mmc$uer_multiple_pages_assigned;
880 11034            ELSE
880 11035              update_eoi_reason := mmc$uer_page_assigned;
882 11036            IFEND;
882 11037            mmp$update_eoi (fde_p, sva.offset, update_eoi_reason);
8E4 11038          IFEND;
8E8 11039
8E8 11040        ELSE
8E8 11041
8E8 11042 { The page is shadowed by another file and the page resides on the shadow file.
```



```
8E8 11043 { If shadow is by segment number then assign pages for the transfer unit and set the 'm'
8E8 11044 { bit in the page table.  Put the source and destination pva into the xcb along with the
8E8 11045 { page count for the transfer.  Set the monitor flag, mmc$mf_shadow_file_reference, so that
8E8 11046 { mmp$mfh_shadow_file_reference will be called to copy the data.  Otherwise initiate I/O to
8E8 11047 { read the transfer unit containing the page from the shadow file and set the 'm' bit in the
8E8 11048 { page table entry for each page.
8E8 11049
8E8 11050         active_au_offset := (sva.offset DIV fde_p^.allocation_unit_size) * fde_p^.allocation_unit_size;
8E8 11051         shadow_au_offset := (stxe_p^.shadow_info.shadow_start_page_number * osv$page_size) + active_au_offset;
8E8 11052
8E8 11053         gfp$mtr_get_locked_fde_p (stxe_p^.shadow_info.shadow_sfid, cst_p^.ijle_p, passive_fde_p);
97E 11054         sva.offset := active_au_offset;
97E 11055
97E 11056         bytes_to_read := stxe_p^.shadow_info.shadow_length_page_count * osv$page_size - active_au_offset;
97E 11057         IF bytes_to_read > fde_p^.allocation_unit_size THEN
996 11058           bytes_to_read := fde_p^.allocation_unit_size;
99A 11059         IFEND;
99A 11060
99A 11061         page_count := bytes_to_read DIV osv$page_size;
99A 11062
99A 11063 { If the job is able to take a trap and copy the pages from the passive segment, send a
99A 11064 { flag to the job to do this.
99A 11065
99A 11066         IF (cst_p^.xcb_p^.xp.trap_enable = osc$traps_enabled) AND
9CE 11067               (cst_p^.xcb_p^.xp.p_register.pva.ring > 1) AND (stxe_p^.shadow_info.shadow_segment_kind =
9CE 11068               mmc$ssk_segment_number) THEN
9CE 11069           mmp$assign_page_frame (sva, aste_p, page_count, 0, assigned_page_count, pfti, pstatus);
A02 11070
A02 11071           IF (assigned_page_count = page_count) THEN
A0E 11072             pstatus := ps_new_page_assigned;
A0E 11073
A0E 11074             next_pfti := pfti;
A0E 11075             WHILE next_pfti <> 0 DO
A1C 11076               mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].v := TRUE;
A1C 11077               mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].m := TRUE;
A1C 11078               next_pfti := mmv$pft_p^ [next_pfti].link.bkw;
A1C 11079             WHILEND;
A58 11080
A58 11081             mmp$update_eoi (fde_p, sva.offset + bytes_to_read - osv$page_size, mmc$uer_page_assigned);
A86 11082             cst_p^.xcb_p^.shadow_reference_info.source_pva :=
A86 11083                   #ADDRESS (1, stxe_p^.shadow_info.shadow_segment_number, shadow_au_offset);
A86 11084             cst_p^.xcb_p^.shadow_reference_info.destination_pva :=
A86 11085                   #ADDRESS (1, fde_p^.last_segment_number, sva.offset);
A86 11086             cst_p^.xcb_p^.shadow_reference_info.page_count := page_count;
A86 11087             tmp$set_monitor_flag (cst_p^.taskid, mmc$mf_shadow_file_reference, status);
AF4 11088           ELSE
AF4 11089
AF4 11090 { Not all pages assigned.  Release those that were.
AF4 11091
AF4 11092             WHILE pfti <> 0 DO
AFC 11093               mmp$delete_pt_entry (pfti, TRUE);
B14 11094               next_pfti := mmv$pft_p^ [pfti].link.bkw;
B14 11095               mmp$relink_page_frame (pfti, mmc$pq_free);
B48 11096               pfti := next_pfti;
B48 11097             WHILEND;
B50 11098
```



```
B50 11099           IFEND;
B54 11100
B54 11101 { If the job cannot trap or if the file is not shadowed by segment number, issue
B54 11102 { the IO requests to read the pages.
B54 11103
B54 11104         ELSE
B54 11105           read_pages_from_disk_or_server (io_function, passive_fde_p, sva, page_count, aste_p,
B88 11106                 shadow_au_offset, io_id, TRUE, pstatus, pfti);
B88 11107
B88 11108           IF (pstatus = ps_found_on_disk) OR (pstatus = ps_found_on_server) THEN
B98 11109             next_pfti := pfti;
B98 11110             WHILE next_pfti <> 0 DO
BA0 11111               mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].m := TRUE;
BA0 11112               next_pfti := mmv$pft_p^ [next_pfti].link.bkw;
BA0 11113             WHILEND;
BD4 11114             mmp$update_eoi (fde_p, sva.offset + bytes_to_read - osv$page_size, mmc$uer_page_assigned);
C04 11115           IFEND;
C04 11116         IFEND;
C04 11117       IFEND;
C06 11118
C06 11119     ELSE
C06 11120       mtp$error_stop ('mm - unexpected DM error');
C26 11121     CASEND;
C26 11122
C26 11123   PROCEND mmp$page_pull;
  0 11124
```

```
        0 11126
        0 11127    PROCEDURE read_pages_from_disk_or_server
        0 11128      (     io_function: iot$io_function;
        0 11129            fde_p: gft$locked_file_desc_entry_p;
        0 11130            sva: ost$system_virtual_address;
        0 11131            page_count: mmt$page_frame_index;
        0 11132            aste_p: ^mmt$active_segment_table_entry;
        0 11133            file_offset: integer;
        0 11134            io_id: mmt$io_identifier;
        0 11135            all_requested_needed: boolean;
        0 11136       VAR pstatus: mmt$page_pull_status;
        0 11137       VAR pfti: mmt$page_frame_index);
        0 11138
        0 11139      VAR
        0 11140        buffer_descriptor: mmt$buffer_descriptor,
        0 11141        assigned_page_count: mmt$page_frame_index,
        0 11142        next_pfti: mmt$page_frame_index,
        0 11143        status: syt$monitor_status;
        0 11144
        0 11145
        0 11146  {   Assign the page frame for the incoming page.
        0 11147
        0 11148      mmp$assign_page_frame (sva, aste_p, page_count, 0, assigned_page_count, pfti, pstatus);
       32 11149
       32 11150      IF (assigned_page_count > 0) AND ((assigned_page_count = page_count) OR NOT all_requested_needed) THEN
       46 11151        buffer_descriptor.buffer_descriptor_type := mmc$bd_paging_io;
       46 11152        buffer_descriptor.sva := sva;
       46 11153        buffer_descriptor.page_count := assigned_page_count;
       46 11154        IF fde_p^.media = gfc$fm_mass_storage_file THEN
       60 11155          iop$pager_io (fde_p, file_offset, buffer_descriptor, assigned_page_count * osv$page_size, io_function,
       9A 11156                io_id, status);
       9A 11157          pstatus := ps_found_on_disk;
       A4 11158
       A4 11159        ELSE
       A4 11160          dfp$server_io (fde_p, ioc$read_page, file_offset, assigned_page_count * osv$page_size, io_id,
       DE 11161                buffer_descriptor, status);
       DE 11162          pstatus := ps_found_on_server;
       E4 11163        IFEND;
       E4 11164        IF status.normal THEN
       EC 11165          RETURN;
       F2 11166        ELSEIF (status.condition = dme$transient_error) OR (status.condition = ioe$requests_full) THEN
      10A 11167          pstatus := ps_io_temp_reject;
      114 11168        ELSEIF status.condition = ioe$unit_disabled THEN
      11C 11169          pstatus := ps_volume_unavailable;
      126 11170        ELSEIF status.condition = dfe$server_has_terminated THEN
      12E 11171          pstatus := ps_server_terminated;
      13A 11172        ELSE
      13A 11173          mtp$error_stop ('MM - unexpected phy io error');
      156 11174        IFEND;
      156 11175      IFEND;
      156 11176
      156 11177  { Not enough frames.  Delete the page table entries for the ones just found.
      156 11178
      156 11179      WHILE pfti <> 0 DO
      164 11180        mmp$delete_pt_entry (pfti, TRUE);
      17C 11181        next_pfti := mmv$pft_p^ [pfti].link.bkw;
```



```
      17C 11182        mmp$relink_page_frame (pfti, mmc$pq_free);
      1B0 11183        pfti := next_pfti;
      1B0 11184      WHILEND;
      1B8 11185
      1B8 11186    PROCEND read_pages_from_disk_or_server;
```

PROCESS_MEMORY_IMAGE_PF

```
        0 11189
        0 11190    PROCEDURE process_memory_image_pf
        0 11191      (     sva: ost$system_virtual_address;
        0 11192            aste_p: ^mmt$active_segment_table_entry;
        0 11193       VAR pfti: mmt$page_frame_index;
        0 11194       VAR pstatus: mmt$page_pull_status);
        0 11195
        0 11196      VAR
        0 11197        mpt_status: mmt$make_pt_entry_status,
        0 11198        pfte_p: ^mmt$page_frame_table_entry;
        0 11199
        0 11200      IF (((sva.offset - mmv$image_file.file_offset) + osv$180_memory_limits.lower) DIV osv$page_size) >
       34 11201            UPPERVALUE (pfti) THEN
       34 11202        pstatus := ps_beyond_file_limit;
       34 11203        pfti := 0;
       34 11204        RETURN; {<----}
       42 11205      IFEND;
       42 11206
       42 11207      pfti := ((sva.offset - mmv$image_file.file_offset) + osv$180_memory_limits.lower) DIV osv$page_size;
       42 11208      pfte_p := ^mmv$pft_p^ [pfti];
       42 11209      pfte_p^.age := 0;
       42 11210      pfte_p^.cyclic_age := 0;
       42 11211      pfte_p^.sva := sva;
       42 11212      pfte_p^.aste_p := aste_p;
       42 11213      pfte_p^.locked_page := mmc$lp_aging_lock;
       42 11214      pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
       42 11215
       42 11216      mmp$make_pt_entry (sva, pfti, aste_p, pfte_p, mpt_status);
       C4 11217
       C4 11218      IF mpt_status = mmc$mpt_done THEN
       CC 11219        mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := TRUE;
       CC 11220        pstatus := ps_found_in_avail;
      100 11221      ELSEIF mpt_status = mmc$mpt_page_table_full THEN
      104 11222        pstatus := ps_pt_full;
      104 11223        mmv$async_work.pt_full_aste_p := aste_p;
      104 11224        mmv$async_work.pt_full_sva := sva;
      104 11225        mmv$async_work.pt_full := TRUE;
      104 11226        mmv$time_to_call_mem_mgr := 0;
      104 11227        osv$time_to_check_asyn := 0;
      128 11228      ELSE
      128 11229        mtp$error_stop ('MM - error in processing memory image pf');
      148 11230      IFEND;
      148 11231
      148 11232    PROCEND process_memory_image_pf;
        0 11233
```


PR_PF - Primary entry point for PF processing

```
        0 11236
        0 11237 {-------------------------------------------------------------------------------------------------------------
        0 11238 {Name:
        0 11239 {  pr_pf
        0 11240 {Purpose:
        0 11241 {  This routine is called by monitor to process a page fault.
        0 11242 {Input:
        0 11243 {  none
        0 11244 {Output:
        0 11245 {  none
        0 11246 {Error Codes:
        0 11247 {  none
        0 11248 {-------------------------------------------------------------------------------------------------------------
        0 11249
        0 11250
        0 11251
        0 11252    PROCEDURE [XDCL] pr_pf
        0 11253      (     dummy: ^cell;
        0 11254            cst_p: ^ost$cpu_state_table);
        0 11255
        0 11256
        0 11257      TYPE
        0 11258        trick_ptr = record
        0 11259          case boolean of
        0 11260          = TRUE =
        0 11261            pva: ost$pva,
        0 11262          = FALSE =
        0 11263            p: ^cell,
        0 11264          casend,
        0 11265        recend;
        0 11266
        0 11267      VAR
        0 11268        aste_p: ^mmt$active_segment_table_entry,
        0 11269        check_aio_slowdown: boolean,
        0 11270        count: 1 .. 32,
        0 11271        cptime: ost$cp_time_value,
        0 11272        faulted_tu: integer,
        0 11273        fde_p: gft$locked_file_desc_entry_p,
        0 11274        file_limit: integer, {must be integer}
        0 11275        found: boolean,
        0 11276        gtid: ^0 .. 0ffffff(16),
        0 11277        i: integer,
        0 11278        ipti: integer,
        0 11279        ijle_p: ^jmt$initiated_job_list_entry,
        0 11280        io_id: mmt$io_identifier,
        0 11281        keypoint_page_fault_status: mmt$keypoint_page_fault_status,
        0 11282        last_faulted_tu: integer,
        0 11283        last_page_fault: ost$segment_offset,
        0 11284        mcount: integer,
        0 11285        monitor_fault: ost$monitor_fault,
        0 11286        nominal_page_fault: boolean,
        0 11287        null_utp: [STATIC, READ] ost$pva := [1, 0fff(16), 7fffffff(16)],
        0 11288        OFF: integer,
        0 11289        page_count: mmt$page_frame_index,
        0 11290        page_streaming_available_page: boolean,
        0 11291        pages_pulled: integer,
```

PR_PF - Primary entry point for PF processing

```
 0 11292       pages_to_be_pulled: integer,
 0 11293       pages_to_read: integer,
 0 11294       pfti: mmt$page_frame_index,
 0 11295       pfti_of_faulted_page: mmt$page_frame_index,
 0 11296       pstatus: mmt$page_pull_status,
 0 11297       pstatus_of_faulted_page: mmt$page_pull_status,
 0 11298       pva: trick_ptr,
 0 11299       rcount: integer,
 0 11300       relative_transfer_unit: integer,
 0 11301       sac_p: ^mmt$segment_access_condition,
 0 11302       seg: integer,
 0 11303       ste_p: ^mmt$segment_descriptor,
 0 11304       streaming_transfer_pages: integer,
 0 11305       streaming_transfer_unit: integer,
 0 11306       stxe_p: ^mmt$segment_descriptor_extended,
 0 11307       sva: ost$system_virtual_address,
 0 11308       sva_current: ost$system_virtual_address,
 0 11309       sva_start: ost$segment_offset,
 0 11310       transfer_unit_count: integer,
 0 11311       tu_to_stream: integer,
 0 11312       xsva: ost$system_virtual_address,
 0 11313       xpfti: mmt$page_frame_index;
 0 11314
 0 11315
 0 11316     io_id.specified := FALSE;
 4 11317
 4 11318 {Use a special routine for page faults which occur during deadstart.
 4 11319
 4 11320     IF NOT mmv$tables_initialized THEN
1A 11321       pf_proc_tables_not_initialized (cst_p^.xcb_p);
2E 11322       RETURN;
3O 11323     IFEND;
3O 11324
3O 11325 { Process page fault on keypoint segment separate from other page faults
3O 11326
3O 11327     IF (cst_p^.xcb_p^.xp.untranslatable_pointer.seg = osc$kpt_pva_segment) THEN
44 11328       osp$process_keypoint_page_fault (cst_p^.xcb_p^.xp.untranslatable_pointer.offset,
66 11329           keypoint_page_fault_status);
66 11330       CASE keypoint_page_fault_status OF
7E 11331
7E 11332       = mmc$kpfs_normal =
7E 11333         RETURN;
84 11334
84 11335       = mmc$kpfs_disable_keypoints =
84 11336         cst_p^.xcb_p^.xp.p_register.pva.offset := cst_p^.xcb_p^.xp.p_register.pva.offset + 4;
84 11337         RETURN;
9E 11338
9E 11339       = mmc$kpfs_invalid_keypoint =
A2 11340
A2 11341 { Do nothing; subsequent page fault processing will reject with an access violation.
A2 11342
A2 11343       ELSE
A2 11344       CASEND;
A2 11345     IFEND;
A2 11346
A2 11347 {Get the PVA that caused the page fault from the exchange package of the current user task and convert it to
```


PR_PF - Primary entry point for PF processing

```
A2 11348 {an SVA.
A2 11349
A2 11350     pva.pva := cst_p^.xcb_p^.xp.untranslatable_pointer;
A2 11351     #KEYPOINT (osk$mtr, osk$monitor_multiplier * pva.pva.seg, mmk$page_fault);
B6 11352     IF syv$perf_keypoints_enabled.memory_keypoints THEN
BE 11353       #KEYPOINT (osk$performance, osk$m * pva.pva.seg, ptk$page_fault_segment);
C6 11354       seg := cst_p^.xcb_p^.xp.p_register.pva.seg;
C6 11355       #KEYPOINT (osk$performance, osk$m * seg, ptk$page_fault_p_segment);
D6 11356       OFF := cst_p^.xcb_p^.xp.p_register.pva.offset MOD 100000(16);
D6 11357       #KEYPOINT (osk$performance, osk$m * OFF, ptk$page_fault_p_lower_offset);
F8 11358       OFF := cst_p^.xcb_p^.xp.p_register.pva.offset DIV 100000(16);
F8 11359       #KEYPOINT (osk$performance, osk$m * OFF, ptk$page_fault_p_upper_offset);
108 11360      gtid := #LOC (cst_p^.xcb_p^.global_task_id);
10C 11361      #KEYPOINT (osk$performance, osk$m * gtid^, ptk$page_fault_gtid);
118 11362      OFF := pva.pva.offset MOD 100000(16);
118 11363      #KEYPOINT (osk$performance, osk$m * OFF, ptk$page_fault_lower_offset);
13A 11364      OFF := pva.pva.offset DIV 100000(16);
13A 11365      #KEYPOINT (osk$performance, osk$m * OFF, ptk$page_fault_upper_offset);
14A 11366    IFEND;
14A 11367
14A 11368    mmp$convert_pva (pva.p, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
190 11369
190 11370    ijle_p := cst_p^.ijle_p;
190 11371    IF aste_p = NIL THEN
1A2 11372      faulted_tu := 0;
1A2 11373      page_count := 0;
1A2 11374      pstatus := ps_read_beyond_eoi;
1B2 11375    ELSEIF stxe_p^.access_state <> mmc$sas_allow_access THEN
1BE 11376      IF stxe_p^.access_state = mmc$sas_inhibit_access THEN
1C4 11377        mmv$refs_to_unrec_df_file_inhib := mmv$refs_to_unrec_df_file_inhib + 1;
1C4 11378        pstatus := ps_volume_unavailable;
1D8 11379      ELSEIF stxe_p^.access_state = mmc$sas_terminate_access THEN
1DE 11380        mmv$refs_to_unrec_df_file_term := mmv$refs_to_unrec_df_file_term + 1;
1DE 11381        pstatus := ps_server_terminated;
1FO 11382      IFEND;
1FO 11383      faulted_tu := 0;
1FO 11384      page_count := 0;
1FA 11385    ELSE
1FA 11386
1FA 11387      streaming_transfer_pages :=
1FA 11388      #SHIFT (1, stxe_p^.stream.transfer_size);
1FA 11389      streaming_transfer_unit := streaming_transfer_pages * osv$page_size;
1FA 11390      faulted_tu := sva.offset DIV streaming_transfer_unit;
1FA 11391      last_page_fault := stxe_p^.stream.last_page_fault;
1FA 11392      stxe_p^.stream.last_page_fault := sva.offset;
1FA 11393      last_faulted_tu := last_page_fault DIV streaming_transfer_unit;
1FA 11394      relative_transfer_unit := faulted_tu - last_faulted_tu;
1FA 11395
1FA 11396    /find_the_page/
1FA 11397      BEGIN
1FA 11398
1FA 11399        mmp$page_pull_hash_sva (sva, aste_p, page_count, pstatus, pfti);
36E 11400
36E 11401 { If the page was found, the code below will exit /find_the_page/ quickly if the segment is not in prestream
36E 11402 { or stream mode.  If the page is not found or if the segment is in a stream mode, page_count, pstatus, and
36E 11403 { pfti will be used later to determine if a call to mmp$page_pull is necessary.  Segments in a stream mode do
```

PR_PF - Primary entry point for PF processing

```
36E 11404 { not attempt the quick exit because:   A) If an available page is found, the stream code will pull another
36E 11405 { page.   B) If a locked/valid page is found, a segment in stream mode may need to read another transfer unit.
36E 11406
36E 11407
36E 11408         nominal_page_fault := TRUE;
36E 11409
36E 11410 { To detect sequential processing of a segment, the segment is logically divided into page streaming transfer
36E 11411 { units.  If a page fault is either in the same transfer unit or in the next transfer unit then it is
36E 11412 { considered to be sequential processing.   Prestream mode is entered when the number of consecutive page
36E 11413 { faults that appear to be sequential exceeds the mmv$page_streaming_prestream.  In prestream mode the
36E 11414 { pages from the current page fault to the end of the transfer unit will be pulled.  Once we get to
36E 11415 { page streaming mode the process will keep IO requests outstanding to read the pages from the current
36E 11416 { transfer unit plus read ahead one or more TU (total number of TU read = mmv$page_streaming_reads)
36E 11417 { The code below checks the prestream threshold, counts sequential accesses, determines if the current page
36E 11418 { fault is to be considered sequential, and either continues or terminates the sequential process.
36E 11419
36E 11420         IF stxe_p^.stream.sequential_accesses < mmv$page_streaming_prestream THEN
382 11421           IF (relative_transfer_unit < 0) OR (relative_transfer_unit > 1) THEN
38C 11422             stxe_p^.stream.sequential_accesses := 1; {Reset counter, count current fault as first fault
39A 11423           ELSE
39A 11424             stxe_p^.stream.sequential_accesses := stxe_p^.stream.sequential_accesses + 1;
3AO 11425           IFEND;
3AO 11426           IF page_count = 1 THEN
3A8 11427             EXIT /find_the_page/;
3AC 11428           IFEND;
3BO 11429
3BO 11430         ELSE {prestream mode or stream mode}
3BO 11431
3BO 11432 { The segment is in prestream or stream.  If it is very well behaved the relative transfer unit will be
3BO 11433 { equal to zero or one.  If zero, do not increment the sequential accesses count since a fault has already
3BO 11434 { occurred in this TU. (The algorithm to switch to page streaming mode assumes each fault after prestream
3BO 11435 { mode is entered is equal to a transfer_unit of data)  If one, increment the sequential accesses count.
3BO 11436 { For the special cases listed below, continue but don't increment the sequential count, wait for
3BO 11437 { confirming sequential faults before going to page streaming mode.  the special cases:
3BO 11438 { a) Faulted_tu = 0; A fault in the first transfer unit.  Since the last fault was in a higher TU, a fault
3BO 11439 {                  in TU=0 may indicate a rewind of the file.
3BO 11440 { b) Relative_Transfer_Unit greater than one but less than mmv$page_streaming_reads+1;  If in prestream mode
3BO 11441 {                  the task has skipped at least one transfer unit, In stream mode we are not sure if it is
3BO 11442 {                  normal or if a tranfer unit has been skipped.
3BO 11443 { c) Preset_Streaming;  A special case to immediately enter page streaming mode.
3BO 11444 {
3BO 11445 { Note that nominal_page_fault  = TRUE
3BO 11446
3BO 11447           IF relative_transfer_unit = 1 THEN
3B6 11448             nominal_page_fault := FALSE;
3B6 11449             IF stxe_p^.stream.random_faults > 0 THEN
3C6 11450               stxe_p^.stream.random_faults := 0;
3C6 11451               mmv$paging_statistics.page_streaming.random_faults :=
3DA 11452                     mmv$paging_statistics.page_streaming.random_faults + 1;
3DA 11453             IFEND;
3DA 11454             IF (stxe_p^.stream.sequential_accesses < UPPERVALUE (stxe_p^.stream.sequential_accesses)) THEN
3EA 11455               stxe_p^.stream.sequential_accesses := stxe_p^.stream.sequential_accesses + 1;
3FO 11456             IFEND;
3F4 11457
3F4 11458           ELSEIF (((relative_transfer_unit >= 0) AND (relative_transfer_unit <=
416 11459                 (mmv$page_streaming_reads + 1))) OR (faulted_tu = 0) OR (stxe_p^.stream.preset_streaming))
```


PR_PF - Primary entry point for PF processing

```
416 11460                 THEN
416 11461             nominal_page_fault := FALSE;
41E 11462
41E 11463 { This fault is considered random because it is either prior to the transfer unit of the last fault or it is
41E 11464 { in a TU that is more than mmv$page_streaming_reads past the TU of the last fault.  If in prestream mode
41E 11465 { terminate prestream.  If in page streaming mode, allow up to mmv$page_streaming_random_limit random faults.
41E 11466
41E 11467           ELSEIF NOT stxe_p^.stream.streaming THEN
428 11468
428 11469 {terminate prestream mode,  count current fault as first fault
428 11470
428 11471             mmv$paging_statistics.page_streaming.prestream_only :=
428 11472                   mmv$paging_statistics.page_streaming.prestream_only + 1;
428 11473             stxe_p^.stream.sequential_accesses := 1;
43C 11474
43C 11475           ELSE {Page fault is in a transfer unit that is considered random, terminate if appropriate
43C 11476             stxe_p^.stream.random_faults := stxe_p^.stream.random_faults + 1;
43C 11477             IF stxe_p^.stream.random_faults < mmv$page_streaming_random_limit THEN
460 11478
460 11479 { Doing nothing will suspend streaming for this fault
460 11480
460 11481               ELSE { Terminate Streaming
460 11482                 mmv$paging_statistics.page_streaming.terminated :=
460 11483                       mmv$paging_statistics.page_streaming.terminated + 1;
460 11484                 stxe_p^.stream.sequential_accesses := 1;
460 11485                 stxe_p^.stream.random_faults := 0;
460 11486                 stxe_p^.stream.streaming := FALSE;
480 11487               IFEND;
486 11488             IFEND;
48A 11489           IFEND; {prestream mode or stream mode}
48A 11490
48A 11491
48A 11492 { If this page fault is to be processed via normal page fault processing nominal_page_fault will be TRUE.
48A 11493 { Otherwise this fault will be processed as a page streaming fault in which one or more transfer units will
48A 11494 { be read.  If page streaming, the first call to page pull is for the actual page that faulted.  The status
48A 11495 { from that call must be saved so that the processing at the end can be determined by the page that faulted.
48A 11496 { All page streaming calls that are reading ahead must not cause the allocation of a new page (we will wait
48A 11497 { to allocate until the task actually faults for the page) and if an error occurs it is just to terminate
48A 11498 { this instant of read ahead without being processed as an error (again, if the task actually faults for the
48A 11499 { page that got an error, the error will be processed at that time)
48A 11500 { Note that if the earlier call to mmp$page_pull_hash_sva found a page, page_count = 1 and pstatus
48A 11501 { and pfti refer to that page.
48A 11502
48A 11503           IF nominal_page_fault THEN
48E 11504             IF page_count = 0 THEN {call mmp$page_pull if the page was not found by mmp$page_pull_hash_sva
496 11505               IF mmc$sa_read_transfer_unit IN stxe_p^.software_attribute_set THEN
4A6 11506                 pages_to_read := streaming_transfer_pages;
4A6 11507                 IF stxe_p^.stream.sequential_accesses < mmv$page_streaming_prestream THEN
4B4 11508                   stxe_p^.stream.sequential_accesses := stxe_p^.stream.sequential_accesses + 1;
4BA 11509                 IFEND;
4BE 11510               ELSEIF (ste_p^.ste.xp = osc$non_executable) OR (ste_p^.ste.wp <> osc$non_writable) THEN
4D6 11511                 pages_to_read := mmv$read_tu_read_write;
4EO 11512               ELSE
4EO 11513                 pages_to_read := mmv$read_tu_execute;
4E6 11514               IFEND;
4E6 11515               mmp$page_pull (sva, fde_p, cst_p, aste_p, stxe_p, io_id, pages_to_read, ioc$read_page, TRUE,
```

PR_PF - Primary entry point for PF processing

```
546 11516                     page_count, pstatus, pfti);
546 11517
546 11518               IF pstatus = ps_done THEN
550 11519                 mtp$error_stop ('MM - internal error-ps_done status from MMP$PAGE_PULL');
570 11520               IFEND;
570 11521             IFEND;
574 11522
574 11523           ELSE {not a nominal page fault... read one or more transfer units
574 11524             transfer_unit_count := 1;
574 11525             IF stxe_p^.stream.preset_streaming THEN {segment has been preset to stream immediately
586 11526               stxe_p^.stream.preset_streaming := FALSE;
586 11527               IF NOT stxe_p^.stream.streaming THEN
596 11528                 mmv$paging_statistics.page_streaming.initiated :=
5A0 11529                       mmv$paging_statistics.page_streaming.initiated + 1;
5A0 11530               IFEND;
5A0 11531               stxe_p^.stream.streaming := TRUE;
5A0 11532               transfer_unit_count := mmv$page_streaming_reads;
5B0 11533
5B0 11534             ELSEIF relative_transfer_unit <= 0 THEN
5B4 11535
5B4 11536 { Note, more than likely this fault is awaiting the disk completion of the second allocation unit within this
5B4 11537 { transfer unit.  The relative transfer unit would only be negative if this a fault in TU=0 of the file while
5B4 11538 { the file is in page streaming mode.
5B4 11539
5B4 11540               IF (sva.offset < last_page_fault) AND (relative_transfer_unit = 0) THEN
5C6 11541                 mmv$paging_statistics.page_streaming.page_faults_tu :=
5D0 11542                       mmv$paging_statistics.page_streaming.page_faults_tu + 1;
5D0 11543               IFEND;
5D6 11544             ELSE
5D6 11545               IF stxe_p^.stream.streaming THEN
5E0 11546                 IF (relative_transfer_unit <= mmv$page_streaming_reads) THEN
5E8 11547                   transfer_unit_count := mmv$page_streaming_reads; {normal streaming, read ahead
5EC 11548                 IFEND;
5EC 11549                 IF (relative_transfer_unit = mmv$page_streaming_reads) THEN
5F4 11550                   mmv$paging_statistics.page_streaming.task_slow :=
5FE 11551                         mmv$paging_statistics.page_streaming.task_slow + 1;
5FE 11552                 IFEND;
602 11553               ELSE {not yet streaming
602 11554                 tu_to_stream := (mmv$page_streaming_threshold DIV
602 11555                       streaming_transfer_unit) + mmv$page_streaming_prestream;
602 11556                 IF stxe_p^.stream.sequential_accesses > tu_to_stream THEN {Initiate streaming
616 11557                   transfer_unit_count := mmv$page_streaming_reads;
616 11558                   stxe_p^.stream.streaming := TRUE;
616 11559                   mmv$paging_statistics.page_streaming.initiated :=
628 11560                         mmv$paging_statistics.page_streaming.initiated + 1;
628 11561                 IFEND; {Initiate streaming
628 11562               IFEND; { streaming boolean
628 11563             IFEND;
628 11564
628 11565 { Prepare to read pages from the current page fault to the end of one or more transfer units.  The counters
628 11566 { for the pages are setup so that we can skip to the end of a transfer unit in some case (locked page).
628 11567 {   pages_to_read -  number of pages to be read from the current transfer unit
628 11568 {   pages_to_be_pulled - total number of pages in all of the TU being pulled (includes pages before fault)
628 11569 {   pages_pulled - number of pages already pulled including pages in the faulted TU prior to fault.
628 11570 {   sva_start - the sva offset of the beginning of the faulted transfer unit
628 11571
```


PR_PF - Primary entry point for PF processing

```
628 11572             sva_start := faulted_tu * streaming_transfer_unit;
628 11573             sva_current := sva;
628 11574             pages_to_read := streaming_transfer_pages - ((sva.offset - sva_start) DIV osv$page_size);
628 11575             pages_pulled := streaming_transfer_pages - pages_to_read;
628 11576             pages_to_be_pulled := streaming_transfer_pages * transfer_unit_count;
628 11577
628 11578 { The total number of pages that may actually be pulled equals the total number of pages in the TUs under
628 11579 { consideration minus the pages before the faulted page (pages_to_be_pulled - pages_pulled).  That total is
628 11580 { to be restricted to 25% of the working set.  WARNING: If at some time in the future the page pulls begin
628 11581 { at the start of the TU, the check will be more complicated to handle a large TU > 25% of MAXWS
628 11582
*WARN*  11583             IF (pages_to_be_pulled - pages_pulled) > (cst_p^.jcb_p^.max_working_set_size DIV 4) THEN
66E 11584               pages_to_be_pulled := pages_pulled + (cst_p^.jcb_p^.max_working_set_size DIV 4);
66E 11585               IF pages_to_read > (cst_p^.jcb_p^.max_working_set_size DIV 4) THEN
676 11586                 pages_to_read := (cst_p^.jcb_p^.max_working_set_size DIV 4);
67A 11587               IFEND;
67C 11588             IFEND;
67C 11589
67C 11590 { Note that page_count and pstatus are still set from the initial call to mmp$page_pull_hash_sva which
67C 11591 { was done early in pr_pf.  page_count = 1 if the page was found.  The quick exit from /find_the_page/
67C 11592 { was not taken because the segment was in a stream mode.
67C 11593
67C 11594             IF page_count = 0 THEN
684 11595               mmp$page_pull (sva, fde_p, cst_p, aste_p, stxe_p, io_id, pages_to_read, ioc$read_page, TRUE,
6E4 11596                     page_count, pstatus, pfti);
6E4 11597             IFEND;
6E4 11598
6E4 11599             pfti_of_faulted_page := pfti;
6E4 11600             pstatus_of_faulted_page := pstatus;
6E4 11601             page_streaming_available_page := FALSE;
6E4 11602
6E4 11603           /exit_and_continue_stream/
6E4 11604             BEGIN
6E4 11605
6E4 11606             /pull_pages_page_streaming/
6E4 11607               WHILE TRUE DO
6FC 11608                 pages_pulled := pages_pulled + page_count;
6FC 11609                 pages_to_read := pages_to_read - page_count; {number of pages left in the current TU
6FC 11610                 IF (pstatus <> ps_locked) AND (pstatus <> ps_valid_in_pt) THEN
714 11611                   IF stxe_p^.stream.streaming THEN
726 11612                     mmv$paging_statistics.page_streaming.pages_streaming :=
734 11613                           mmv$paging_statistics.page_streaming.pages_streaming + page_count;
734 11614                   ELSE
734 11615                     mmv$paging_statistics.page_streaming.pages_prestream :=
73E 11616                           mmv$paging_statistics.page_streaming.pages_prestream + page_count;
73E 11617                   IFEND;
740 11618                 IFEND;
740 11619
740 11620                 CASE pstatus OF
794 11621                 = ps_locked, ps_valid_in_pt = {these pages are not counted, they were counted when initiated
794 11622                   pages_pulled := pages_pulled + pages_to_read;
794 11623                   pages_to_read := 0; {skip to the next transfer unit}
79E 11624                 = ps_found_on_disk =
79E 11625                   ijle_p^.statistics.paging_statistics.page_in_count :=
79E 11626                         ijle_p^.statistics.paging_statistics.page_in_count + page_count;
79E 11627                   cst_p^.xcb_p^.paging_statistics.page_in_count :=
```

PR_PF - Primary entry point for PF processing

```
79E 11628                       cst_p^.xcb_p^.paging_statistics.page_in_count + page_count;
79E 11629                     mmv$paging_statistics.ps_pages.disk := mmv$paging_statistics.ps_pages.disk + page_count;
7CE 11630                   = ps_found_in_avail, ps_found_in_avail_modified =
7CE 11631                     ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue :=
7CE 11632                           ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue + page_count;
7CE 11633                     cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue :=
7CE 11634                           cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue + page_count;
7CE 11635                     mmv$paging_statistics.ps_pages.reclaim := mmv$paging_statistics.ps_pages.reclaim + page_count;
7CE 11636
7CE 11637 { Pages from the available queues probably indicate the file is being referenced again.  It is
7CE 11638 { likely that the next pages are also in one of available queues and therefore the potential
7CE 11639 { performance gain of reading pages ahead is reduced from saving disk accesses to saving only
7CE 11640 { page faults.  Allow up to two pages to come from the available queues and then exit.  Thus
7CE 11641 { if the pages are in the available queues, we will get two pages per page fault when streaming.
7CE 11642
7CE 11643                     IF page_streaming_available_page THEN
7FA 11644                       EXIT /exit_and_continue_stream/;
7FE 11645                     IFEND;
7FE 11646                     page_streaming_available_page := TRUE; {indicate one page has been found in available queues
80C 11647                   = ps_new_page_assigned, ps_allocate_required_on_server =
80C 11648                     ijle_p^.statistics.paging_statistics.new_pages_assigned :=
80C 11649                           ijle_p^.statistics.paging_statistics.new_pages_assigned + page_count;
80C 11650                     cst_p^.xcb_p^.paging_statistics.new_pages_assigned :=
80C 11651                           cst_p^.xcb_p^.paging_statistics.new_pages_assigned + page_count;
80C 11652                     mmv$paging_statistics.ps_pages.new := mmv$paging_statistics.ps_pages.new + page_count;
80C 11653
80C 11654 { Allocation of new pages occurs one at a time, terminate page streaming to avoid overhead
80C 11655
80C 11656                     EXIT /pull_pages_page_streaming/; { terminate page streaming
83A 11657                   = ps_found_on_server =
83A 11658                     ijle_p^.statistics.paging_statistics.pages_from_server :=
83A 11659                           ijle_p^.statistics.paging_statistics.pages_from_server + page_count;
83A 11660                     cst_p^.xcb_p^.paging_statistics.pages_from_server :=
83A 11661                           cst_p^.xcb_p^.paging_statistics.pages_from_server + page_count;
83A 11662                     mmv$paging_statistics.ps_pages.server := mmv$paging_statistics.ps_pages.server + page_count;
86A 11663                   = ps_done =
86A 11664                     mtp$error_stop ('MM - internal error-ps_done status from MMP$PAGE_PULL');
89A 11665                   ELSE
89A 11666
89A 11667 { ps_read_beyond_eoi, ps_no_extend_permission, ps_beyond_file_limit
89A 11668 { ps_server_terminated,  ps_io_temp_reject,  ps_pt_full, ps_volume_unavailable
89A 11669 { ps_no_memory, ps_low_on_memory, ps_job_work_required
89A 11670
89A 11671                     EXIT /pull_pages_page_streaming/; { terminate page streaming
89E 11672                   CASEND;
89E 11673
89E 11674                   IF pages_pulled >= pages_to_be_pulled THEN
8A2 11675                     EXIT /exit_and_continue_stream/;
8A8 11676                   IFEND;
8A8 11677                   IF pages_to_read <= 0 THEN
8AC 11678                     pages_to_read := streaming_transfer_pages;
8B0 11679                   IFEND;
8B0 11680                   sva_current.offset := sva_start + pages_pulled * osv$page_size;
*WARN*   11681              mmp$page_pull_hash_sva (sva_current, aste_p, page_count, pstatus, pfti);
9FE 11682                   IF page_count = 0 THEN
A06 11683                     mmp$page_pull (sva_current, fde_p, cst_p, aste_p, stxe_p, io_id, pages_to_read, ioc$read_page,
```


PR_PF - Primary entry point for PF processing

```
A62 11684                           FALSE, page_count, pstatus, pfti);
A62 11685                     IFEND;
A62 11686                 WHILEND /pull_pages_page_streaming/;
A68 11687
A68 11688 { Terminate page streaming if exit here, usual exit is to skip this code
A68 11689
A68 11690                 stxe_p^.stream.sequential_accesses := 0;
A68 11691                 stxe_p^.stream.random_faults := 0;
A68 11692                 stxe_p^.stream.streaming := FALSE;
A68 11693
A68 11694               END /exit_and_continue_stream/;
A88 11695
A88 11696               page_count := 0;
A88 11697               pstatus := pstatus_of_faulted_page;
A88 11698               pfti := pfti_of_faulted_page;
AB0 11699
AB0 11700             IFEND; {nominal_page_fault}
AB0 11701
AB0 11702
AB0 11703           END /find_the_page/;
AB0 11704
AB0 11705           IF syv$perf_keypoints_enabled.memory_keypoints THEN
AB8 11706             #KEYPOINT (osk$performance, osk$m * pfti, ptk$page_fault_pfti);
AC4 11707             #KEYPOINT (osk$performance, osk$m * $INTEGER (pstatus), ptk$page_fault_status);
AD0 11708             #KEYPOINT (osk$performance, osk$m * (aste_p^.ijl_ordinal.block_number *
AF2 11709                   32 + aste_p^.ijl_ordinal.block_index), ptk$page_fault_ijl);
AF2 11710           IFEND;
AFE 11711
AFE 11712
AFE 11713         IFEND; { aste_p <> NIL
AFE 11714
AFE 11715
AFE 11716 { We have gotten here in one of four ways in which pstatus may have been set:
AFE 11717 {     1. aste_p = NIL  (not likely to happen)
AFE 11718 {     2. The page was found by the proc mmp$page_pull_hash_sva and then the quick exit was taken.  At this
AFE 11719 {        point this case is exactly the same as a nominal page fault (case 3).
AFE 11720 {     3. Nominal page fault (PSTATUS and PAGE_COUNT reflect the call to pull the faulted page)
AFE 11721 {     4. Page streaming mode pulled the faulted page and saved PSTATUS.  Additional page pulls may have been
AFE 11722 {        performed after the faulted page was pulled.  PSTATUS has been restored to the value returned with
AFE 11723 {        the faulted page but PAGE_COUNT = 0 because the page counters were updated by the page streaming code.
AFE 11724
AFE 11725         cst_p^.xcb_p^.paging_statistics.page_fault_count := cst_p^.xcb_p^.paging_statistics.page_fault_count + 1;
AFE 11726         ijle_p^.statistics.paging_statistics.page_fault_count :=
AFE 11727               ijle_p^.statistics.paging_statistics.page_fault_count + 1;
AFE 11728
AFE 11729
AFE 11730         CASE pstatus OF
BBA 11731
BBA 11732         = ps_found_in_avail, ps_found_in_avail_modified =
BBA 11733           ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue :=
BBA 11734                 ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue + page_count;
BBA 11735           cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue :=
BBA 11736                 cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue + page_count;
BBA 11737           mmv$paging_statistics.pf_pages.reclaim := mmv$paging_statistics.pf_pages.reclaim + page_count;
BE4 11738
BE4 11739         = ps_new_page_assigned =
```

PR_PF - Primary entry point for PF processing

```
BE4 11740         ijle_p^.statistics.paging_statistics.new_pages_assigned :=
BE4 11741               ijle_p^.statistics.paging_statistics.new_pages_assigned + page_count;
BE4 11742         cst_p^.xcb_p^.paging_statistics.new_pages_assigned :=
BE4 11743               cst_p^.xcb_p^.paging_statistics.new_pages_assigned + page_count;
BE4 11744         mmv$paging_statistics.pf_pages.new := mmv$paging_statistics.pf_pages.new + page_count;
BE4 11745         stxe_p^.stream.sequential_accesses := 0; { zero count to prevent prestream mode on next PF
BE4 11746
BE4 11747
BE4 11748 { Since new pages are being allocated, force the transfer unit size to be at least an allocation unit
BE4 11749 { so that if free behind is performed it will go back at least an allocation size to free pages.
BE4 11750 { relative_transfer_unit is not recalculated, it will just cause free behind to be attempted early.
BE4 11751
BE4 11752         IF fde_p^.allocation_unit_size > streaming_transfer_unit THEN
C22 11753           streaming_transfer_unit := fde_p^.allocation_unit_size;
C22 11754           faulted_tu := sva.offset DIV streaming_transfer_unit;
C36 11755         IFEND;
C44 11756
C44 11757       = ps_valid_in_pt, ps_job_work_required =
C48 11758       = ps_read_beyond_eoi, ps_no_extend_permission =
C48 11759         monitor_fault.identifier := mmc$segment_fault_processor_id;
C48 11760         sac_p := #LOC (monitor_fault.contents);
C52 11761         IF pstatus = ps_read_beyond_eoi THEN
C5C 11762           sac_p^.identifier := mmc$sac_read_beyond_eoi;
C66 11763         ELSE
C66 11764           sac_p^.identifier := mmc$sac_no_append_permission;
C6E 11765         IFEND;
C6E 11766         sac_p^.segment := pva.p;
C6E 11767         tmp$send_monitor_fault (cst_p^.taskid, #LOC (monitor_fault), TRUE);
C9C 11768
C9C 11769       = ps_beyond_file_limit =
C9C 11770         monitor_fault.identifier := mmc$segment_fault_processor_id;
C9C 11771         sac_p := #LOC (monitor_fault.contents);
CA6 11772         sac_p^.identifier := mmc$sac_read_write_beyond_ms1;
CA6 11773         sac_p^.segment := pva.p;
CA6 11774         IF fde_p^.stack_for_ring > 0 THEN
CC2 11775           file_limit := fde_p^.file_limit + 500000;
CC2 11776           IF file_limit > 07fffffff(16) THEN
CD8 11777             file_limit := 7fffffff(16);
CDC 11778           IFEND;
CDC 11779           fde_p^.file_limit := file_limit;
CE4 11780         IFEND;
CE4 11781         tmp$send_monitor_fault (cst_p^.taskid, #LOC (monitor_fault), FALSE);
D0A 11782
D0A 11783       = ps_server_terminated =
D0A 11784         monitor_fault.identifier := mmc$segment_fault_processor_id;
D0A 11785         sac_p := #LOC (monitor_fault.contents);
D14 11786         sac_p^.identifier := mmc$sac_file_server_terminated;
D14 11787         sac_p^.segment := pva.p;
D14 11788         tmp$send_monitor_fault (cst_p^.taskid, #LOC (monitor_fault), FALSE);
D4A 11789
D4A 11790       = ps_found_on_disk =
D4A 11791         ijle_p^.statistics.paging_statistics.page_in_count :=
D4A 11792               ijle_p^.statistics.paging_statistics.page_in_count + page_count;
D4A 11793         cst_p^.xcb_p^.paging_statistics.page_in_count := cst_p^.xcb_p^.paging_statistics.page_in_count +
D4A 11794               page_count;
D4A 11795         mmv$paging_statistics.pf_pages.disk := mmv$paging_statistics.pf_pages.disk + page_count;
```


PR_PF - Primary entry point for PF processing

```
D4A 11796
D4A 11797 {NOTE
D4A 11798 {Job recovery uses active io count to determine the status of a page after a crash ...
D4A 11799 {DONT MESS WITH ACTIVE IO COUNT UNLESS YOU UNDERSTAND IT'S USE IN JOB RECOVERY. Thank You.
D4A 11800
D4A 11801         IF mmv$pft_p^ [pfti].active_io_count = 0 THEN
D92 11802           mtp$error_stop ('MM - page fault queue no IO');
DB2 11803         IFEND;
DB2 11804         tmp$queue_task (cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^ [pfti].task_queue);
DF0 11805         cst_p^.xcb_p^.page_wait_info.pva := pva.p;
E04 11806
E04 11807       = ps_locked =
E04 11808
E04 11809 {NOTE
E04 11810 {Job recovery uses active io count to determine the status of a page after a crash ...
E04 11811 {DONT MESS WITH ACTIVE IO COUNT UNLESS YOU UNDERSTAND IT'S USE IN JOB RECOVERY. Thank You.
E04 11812
E04 11813         IF mmv$pft_p^ [pfti].active_io_count = 0 THEN
E26 11814           mtp$error_stop ('MM - page fault queue no IO');
E46 11815         IFEND;
E46 11816         tmp$queue_task (cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^ [pfti].task_queue);
E84 11817         cst_p^.xcb_p^.page_wait_info.pva := pva.p;
E98 11818
E98 11819       = ps_io_temp_reject =
E98 11820         tmp$cause_task_switch;
EA6 11821
EA6 11822       = ps_pt_full =
EA6 11823         cst_p^.dispatch_control.asynchronous_interrupts_pending := TRUE;
EA6 11824         tmp$cause_task_switch;
EB8 11825
EB8 11826       = ps_no_memory, ps_low_on_memory =
EB8 11827         tmp$queue_task (cst_p^.taskid, tmc$ts_memory_wait, mmv$memory_wait_queue);
EE0 11828
EE0 11829 {  Process the case of the page found on the server mainframe.
EE0 11830
EE0 11831       = ps_found_on_server =
EE0 11832         ijle_p^.statistics.paging_statistics.pages_from_server :=
EE0 11833               ijle_p^.statistics.paging_statistics.pages_from_server + page_count;
EE0 11834         cst_p^.xcb_p^.paging_statistics.pages_from_server :=
EE0 11835               cst_p^.xcb_p^.paging_statistics.pages_from_server + page_count;
EE0 11836         mmv$paging_statistics.pf_pages.server := mmv$paging_statistics.pf_pages.server + page_count;
EE0 11837
EE0 11838 { Check active_io_count to insure activity exists.  No activity is fatal.
EE0 11839
EE0 11840         IF mmv$pft_p^ [pfti].active_io_count = 0 THEN
F28 11841           mtp$error_stop ('MM - page fault queue no IO, ps_found_on_server');
F48 11842         IFEND;
F48 11843
F48 11844 { Queue the task in page wait.
F48 11845
F48 11846         tmp$queue_task (cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^ [pfti].task_queue);
F86 11847         cst_p^.xcb_p^.page_wait_info.pva := pva.p;
F9A 11848
F9A 11849       = ps_allocate_required_on_server =
F9A 11850
F9A 11851         ijle_p^.statistics.paging_statistics.new_pages_assigned :=
```

PR_PF - Primary entry point for PF processing

```
F9A 11852              ijle_p^.statistics.paging_statistics.new_pages_assigned + page_count;
F9A 11853          cst_p^.xcb_p^.paging_statistics.new_pages_assigned :=
F9A 11854              cst_p^.xcb_p^.paging_statistics.new_pages_assigned + page_count;
F9A 11855          mmv$paging_statistics.pf_pages.new := mmv$paging_statistics.pf_pages.new + page_count;
F9A 11856          stxe_p^.stream.sequential_accesses := 0; [ zero  count to prevent prestream mode on next PF
F9A 11857
F9A 11858
F9A 11859 { Since new pages are being allocated, force the transfer unit size to be at least an allocation unit
F9A 11860 { so that if free behind is performed it will go back at least an allocation size to free pages.
F9A 11861 { relative_transfer_unit is not recalculated, it will just cause free behind to be attempted early.
F9A 11862
F9A 11863          IF fde_p^.allocation_unit_size > streaming_transfer_unit THEN
FD8 11864            streaming_transfer_unit := fde_p^.allocation_unit_size;
FD8 11865            faulted_tu := sva.offset DIV streaming_transfer_unit;
FEC 11866          IFEND;
FEC 11867
FEC 11868 { Check active_io_count to insure activity exists.  No activity is fatal.
FEC 11869
FEC 11870          IF mmv$pft_p^ [pfti].active_io_count = 0 THEN
100E 11871           mtp$error_stop ('MM - page fault queue no IO, ps_found_on_server');
102E 11872         IFEND;
102E 11873
102E 11874 { Queue the task in page wait.
102E 11875
102E 11876         tmp$queue_task (cst_p^.taskid, tmc$ts_page_wait, mmv$pft_p^ [pfti].task_queue);
106C 11877         cst_p^.xcb_p^.page_wait_info.pva := pva.p;
1084 11878
1084 11879       = ps_volume_unavailable =
1084 11880         cst_p^.xcb_p^.page_wait_info.pva := pva.p;
1084 11881         mmp$process_volume_unavailable (cst_p^.xcb_p, FALSE);
10B0 11882       = ps_done =
10B0 11883         mtp$error_stop ('MM - internal error-ps_done status from MMP$PAGE_PULL');
10D2 11884
10D2 11885       CASEND;
10D2 11886
10D2 11887
10D2 11888
10D2 11889 { If appropriate do free behind.  Free pages that are in the transfer units prior to the transfer unit
10D2 11890 { immediately before the current page fault transfer unit.  (i.e.  faulted_tu -2, -3, -4 ... etc.)
10D2 11891 { Stop freeing pages at the first page that is not freed because it is locked or it is not found.
10D2 11892 { Note that if a new page was assigned, a check was made to force transfer size >= allocation size.
10D2 11893 { Since this code looks at the pages in reverse order, the PFTI procs mmp$reset_store_pfti_reverse
10D2 11894 { and mmp$store_pfti_reverse are used so that  mmp$remove_pages_from_jws will free the pages in
10D2 11895 { sva ascending order  ... this is helpful if the pages are modified
10D2 11896
10D2 11897       IF (mmc$sa_free_behind IN stxe_p^.software_attribute_set) AND
10EE 11898           (fde_p^.media <> gfc$fm_transient_segment) THEN
10EE 11899         xsva := sva;
10EE 11900         IF ((relative_transfer_unit > 0)) AND (faulted_tu > 1) THEN
10FE 11901           xsva.offset := ((faulted_tu - 1) * streaming_transfer_unit);
10FE 11902           mmp$reset_store_pfti_reverse;
10FE 11903
10FE 11904         /free_behind/
10FE 11905           WHILE xsva.offset > 0 DO {since xsva is page boundary, if >0 it will be at least = 1 page.
1138 11906             xsva.offset := xsva.offset - osv$page_size;
1138 11907             #HASH_SVA (xsva, ipti, count, found);
```


PR_PF - Primary entry point for PF processing

```
114E 11908             IF NOT found THEN
1162 11909               EXIT /free_behind/; { Exit if the page was not found
116A 11910             ELSE
116A 11911               xpfti := (mmv$pt_p^ [ipti].rma * 512) DIV osv$page_size;
116A 11912               IF (mmv$pft_p^ [xpfti].queue_id >= mmc$pq_first_valid_in_pt) AND
11AA 11913                    (mmv$pft_p^ [xpfti].locked_page = mmc$lp_not_locked) THEN
11AA 11914                 mmp$store_pfti_reverse (xpfti);
11CA 11915               ELSE
11CA 11916                 EXIT /free_behind/; {Exit if the page is locked
11CE 11917               IFEND;
11CE 11918             IFEND;
11CE 11919           WHILEND /free_behind/;
11DC 11920
11DC 11921           mmp$fetch_pfti_array_size (rcount);
11DC 11922           IF rcount > 0 THEN
11F2 11923             mmp$remove_pages_from_jws (mmc$pq_avail_modified, ijle_p, mcount, rcount);
1218 11924             mmv$paging_statistics.page_streaming.pages_freed_behind :=
1226 11925                  mmv$paging_statistics.page_streaming.pages_freed_behind + rcount;
1226 11926           IFEND;
1226 11927         IFEND; {faulted tu GT 1
1226 11928       IFEND; {free behind
1226 11929
1226 11930
1226 11931
1226 11932 {Update page fault statistics.
1226 11933
1226 11934       mmv$pf_statistics [$INTEGER (pstatus)] := mmv$pf_statistics [$INTEGER (pstatus)] + 1;
1226 11935       i := mmv$pf_sva_array.next_i;
1226 11936       mmv$pf_sva_array.next_i := (i + 1) MOD num_pf_recs;
1226 11937       mmv$pf_sva_array.pf_recs [i].sva := sva;
1226 11938       mmv$pf_sva_array.pf_recs [i].pstatus_time := (#FREE_RUNNING_CLOCK (0) DIV 131072) MOD
1264 11939           100(16) + $INTEGER (pstatus) * 100(16);
1264 11940
1264 11941
1264 11942 {Scan the JWS if the job cp time exceeds the aging threshold.
1264 11943
1264 11944       IF mmv$aging_algorithm >= 4 THEN
128C 11945         cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
1294 11946       ELSE
1294 11947         cptime := ijle_p^.statistics.cp_time.time_spent_in_mtr_mode +
12A0 11948                   ijle_p^.statistics.cp_time.time_spent_in_job_mode;
12A0 11949       IFEND;
12A0 11950       IF cptime > cst_p^.jcb_p^.cptime_next_age_working_set THEN
12AC 11951         mmp$age_job_working_set (ijle_p, cst_p^.jcb_p);
12C0 11952       IFEND;
12C0 11953       check_aio_slowdown := (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count >
12C0 11954           cst_p^.jcb_p^.max_working_set_size);
12C0 11955       IF check_aio_slowdown OR (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count >
12E0 11956           mmv$max_working_set_size) THEN
12E0 11957         mmp$trim_job_working_set (ijle_p, cst_p^.jcb_p, FALSE {trim_to_swap_size=false} );
12FC 11958         IF check_aio_slowdown THEN
1300 11959           CASE pstatus OF
131E 11960           = ps_found_in_avail, ps_found_in_avail_modified, ps_new_page_assigned, ps_valid_in_pt =
131E 11961             IF ijle_p^.active_io_requests > mmv$maxws_aio_threshold THEN
132A 11962               mmv$maxws_aio_count := mmv$maxws_aio_count + 1;
132A 11963               cst_p^.xcb_p^.maxws_aio_slowdown := cst_p^.xcb_p^.maxws_aio_slowdown + 1;
```

PR_PF - Primary entry point for PF processing

```
132A 11964              ijle_p^.maxws_aio_slowdown_display := ((mmv$maxws_aio_slowdown DIV
132A 11965                   mmv$jws_queue_age_interval) + 1) MOD 256;
132A 11966              tmp$cause_task_switch;
136E 11967            IFEND;
1372 11968          ELSE
1372 11969          CASEND;
1372 11970        IFEND;
1376 11971      ELSEIF ijle_p^.active_io_requests > mmv$aio_limit THEN
1382 11972        CASE pstatus OF
13A0 11973        = ps_found_in_avail, ps_found_in_avail_modified, ps_new_page_assigned, ps_valid_in_pt =
13A0 11974          tmp$cause_task_switch;
13A8 11975          mmv$aio_limit_count := mmv$aio_limit_count + 1;
13B6 11976        ELSE
13B6 11977        CASEND;
13B6 11978      IFEND;
13B6 11979
13B6 11980      IF cst_p^.xcb_p^.paging_statistics.working_set_max_used < ijle_p^.
13C6 11981            job_page_queue_list [mmc$pq_job_working_set].count THEN
13C6 11982        cst_p^.xcb_p^.paging_statistics.working_set_max_used := ijle_p^.
13C6 11983              job_page_queue_list [mmc$pq_job_working_set].count;
13C6 11984        IF ijle_p^.statistics.paging_statistics.working_set_max_used < ijle_p^.
13D6 11985              job_page_queue_list [mmc$pq_job_working_set].count THEN
13D6 11986          ijle_p^.statistics.paging_statistics.working_set_max_used := ijle_p^.
13DA 11987                job_page_queue_list [mmc$pq_job_working_set].count;
13DA 11988        IFEND;
13DA 11989      IFEND;
13DA 11990
13DA 11991
13DA 11992  {Free queue must be replenished if the number of free+avail pages is below the threshold.
13DA 11993
13DA 11994      check_free_queues (cst_p);
1422 11995
1422 11996
1422 11997  {Reset UTP in the Exchange Package.
1422 11998
1422 11999      cst_p^.xcb_p^.xp.untranslatable_pointer := null_utp;
1422 12000
1422 12001      #KEYPOINT (osk$mtr, $INTEGER (pstatus) * osk$monitor_multiplier, mmk$page_fault + osk$m);
143A 12002
143A 12003    PROCEND pr_pf;
   0 12004
```


MMP$PROCESS_ASSIGN_PAGES_REQ

```
   0 12006
   0 12007    PROCEDURE [XDCL] mmp$process_assign_pages_req
   4 12008      (VAR rb: mmt$rb_assign_pages;
   4 12009           cst_p: ^ost$cpu_state_table);
   4 12010
   4 12011      CASE rb.sub_reqcode OF
  1A 12012      = mmc$aprc_assign =
  1A 12013        mmp$process_assign_pages (rb, cst_p);
  2E 12014
  2E 12015      = mmc$aprc_cancel_reserve =
  2E 12016        mmp$process_cancel_reserve (rb, cst_p);
  4E 12017
  4E 12018      ELSE
  4E 12019        mtp$error_stop ('MM--ASSIGN PAGES--UNKNOWN REQUEST');
  6E 12020      CASEND;
  6E 12021
  6E 12022    PROCEND mmp$process_assign_pages_req;
   0 12023
```

```
 0 12025
 0 12026   PROCEDURE mmp$process_assign_pages
 0 12027     (VAR rb: mmt$rb_assign_pages;
 0 12028          cst_p: ^ost$cpu_state_table);
 0 12029
 0 12030     CONST
 0 12031       mmc$ap_ignore_maxws_and_trim = 131072;
 0 12032
 0 12033     VAR
 0 12034       assigned_pages_count: mmt$page_frame_index,
 0 12035       aste_p: ^mmt$active_segment_table_entry,
 0 12036       count: 1 .. 32,
 0 12037       fde_p: gft$locked_file_desc_entry_p,
 0 12038       first_pfti: mmt$page_frame_index,
 0 12039       found: boolean,
 0 12040       i: integer,
 0 12041       ijle_p: ^jmt$initiated_job_list_entry,
 0 12042       ipti: integer,
 0 12043       memory_available: boolean,
 0 12044       next_pfti: mmt$page_frame_index,
 0 12045       page_status: gft$page_status,
 0 12046       pfte_p: ^mmt$page_frame_table_entry,
 0 12047       pfti: mmt$page_frame_index,
 0 12048       pstatus: mmt$page_pull_status,
 0 12049       psva: ost$system_virtual_address,
 0 12050       pte_p: ^ost$page_table_entry,
 0 12051       reject_offset: ost$segment_offset,
 0 12052       requested_page_count: mmt$page_frame_index,
 0 12053       remaining_pages_to_assign: mmt$page_frame_index,
 0 12054       ste_p: ^mmt$segment_descriptor,
 0 12055       stxe_p: ^mmt$segment_descriptor_extended,
 0 12056       sva: ost$system_virtual_address,
 0 12057       trim_pages: boolean,
 0 12058       xsva: ost$system_virtual_address,
 0 12059       valid_pages_in_memory: mmt$page_frame_index,
 0 12060       pages_not_in_memory: mmt$page_frame_index,
 0 12061       am_pages_in_memory: mmt$page_frame_index,
 0 12062       avail_pages_in_memory: mmt$page_frame_index;
 0 12063
 0 12064     rb.status.normal := TRUE;
 4 12065
 4 12066     IF NOT mmv$tables_initialized THEN
16 12067       RETURN;
18 12068     IFEND;
18 12069
18 12070     mmp$verify_pva (^rb.pva, mmc$sat_write, rb.status);
3A 12071     IF NOT rb.Status.normal THEN
42 12072       RETURN;
44 12073     IFEND;
44 12074
44 12075     mmp$convert_pva (rb.pva, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
84 12076     IF stxe_p^.access_state <> mmc$sas_allow_access THEN
90 12077       IF stxe_p^.access_state = mmc$sas_inhibit_access THEN
94 12078         mtp$set_status_abnormal ('MM', mme$volume_unavailable, rb.status);
94 12079         RETURN;
B0 12080       ELSEIF stxe_p^.access_state = mmc$sas_terminate_access THEN
```



```
B4 12081         mtp$set_status_abnormal ('DF', dfe$server_has_terminated, rb.status);
B4 12082         RETURN;
CE 12083       IFEND;
CE 12084     IFEND;
CE 12085
CE 12086     IF aste_p = NIL THEN
DC 12087       mtp$set_status_abnormal ('MM', mme$invalid_pva, rb.status);
DC 12088       RETURN;
F4 12089     IFEND;
F4 12090
F4 12091     requested_page_count := (#OFFSET (rb.pva) + rb.length - 1) DIV osv$page_size -
F4 12092           (#OFFSET (rb.pva) DIV osv$page_size) + 1;
F4 12093
F4 12094     IF (sva.offset + requested_page_count * osv$page_size) > fde_p^.file_limit THEN
12A 12095       mtp$set_status_abnormal ('MM', mme$assign_length_too_long, rb.status);
12A 12096       RETURN;
142 12097     IFEND;
142 12098
142 12099     IF (aste_p^.queue_id = mmc$pq_wired) OR (aste_p^.queue_id = mmc$pq_job_fixed) THEN
158 12100       mtp$set_status_abnormal ('MM', mme$wired_or_fixed_segs_illegal, rb.status);
158 12101       RETURN;
170 12102     IFEND;
170 12103
170 12104 { Round off the sva to a page boundary.
170 12105
170 12106     sva.offset := (sva.offset DIV osv$page_size) * osv$page_size;
170 12107     xsva := sva;
170 12108
170 12109     CASE fde_p^.media OF
19E 12110     = gfc$fm_transient_segment =
19E 12111       IF (aste_p^.pages_in_memory + requested_page_count > mmv$max_pages_no_file) THEN
1B0 12112         page_status := gfc$ps_job_mode_work_required;
1B0 12113         reject_offset := sva.offset;
1C8 12114       ELSE
1C8 12115         page_status := gfc$ps_page_doesnt_exist;
1CE 12116       IFEND;
1D2 12117     = gfc$fm_mass_storage_file =
1D2 12118       mmv$last_segment_accessed := #SEGMENT (rb.pva);
1D2 12119       dmp$fetch_multi_page_status (fde_p, sva.offset, requested_page_count * osv$page_size,
226 12120             stxe_p^.file_limits_enforced, reject_offset, page_status);
226 12121     = gfc$fm_served_file =
226 12122       dfp$fetch_multi_page_status (fde_p, sva.offset, requested_page_count * osv$page_size, page_status);
252 12123       reject_offset := sva.offset + (requested_page_count - 1) * osv$page_size;
26C 12124     CASEND;
26C 12125
26C 12126
26C 12127     CASE page_status OF
2B8 12128     = gfc$ps_page_on_disk, gfc$ps_page_on_server, gfc$ps_page_doesnt_exist =
2BC 12129
2BC 12130 { These are ok; do nothing
2BC 12131
2BC 12132     = gfc$ps_temp_reject, gfc$ps_account_limit_exceeded =
2BC 12133       mtp$set_status_abnormal ('MM', mme$temporary_reject, rb.status);
2BC 12134       tmp$cause_task_switch;
2DA 12135       RETURN;
2E0 12136
```

```
2E0 12137      = gfc$ps_volume_unavailable =
2E0 12138        mtp$set_status_abnormal ('MM', mme$volume_unavailable, rb.status);
2E0 12139        RETURN;
2FC 12140
2FC 12141      = gfc$ps_server_terminated =
2FC 12142        mtp$set_status_abnormal ('DF', dfe$server_has_terminated, rb.status);
2FC 12143        RETURN;
318 12144
318 12145      = gfc$ps_server_allocate_required, gfc$ps_job_mode_work_required =
318 12146        set_assign_active (stxe_p, reject_offset);
340 12147        tmp$set_monitor_flag (cst_p^.taskid, mmc$mf_segment_mgr_flag, rb.status);
362 12148        mtp$set_status_abnormal ('MM', mme$dm_assign_active, rb.status);
362 12149        RETURN;
37A 12150
37A 12151      ELSE
37A 12152        mtp$error_stop ('mm - unexpected DM error in assign pages');
39A 12153      CASEND;
39A 12154
39A 12155
39A 12156  { Calculate how many pages are needed; some may already be in the jws.
39A 12157
39A 12158      valid_pages_in_memory := 0;
39A 12159      pages_not_in_memory := 0;
39A 12160      am_pages_in_memory := 0;
39A 12161      avail_pages_in_memory := 0;
39A 12162
39A 12163      FOR i := 1 TO requested_page_count DO
3AC 12164        #HASH_SVA (xsva, ipti, count, found);
3AE 12165        IF found THEN
3C2 12166          pte_p := ^mmv$pt_p^ [ipti];
3C2 12167          IF pte_p^.v THEN
3D6 12168            valid_pages_in_memory := valid_pages_in_memory + 1;
3DC 12169          ELSE
3DC 12170            pfti := pte_p^.rma * 512 DIV osv$page_size;
3DC 12171            pfte_p := ^mmv$pft_p^ [pfti];
3DC 12172            IF pfte_p^.queue_id = mmc$pq_avail THEN
416 12173              avail_pages_in_memory := avail_pages_in_memory + 1;
41C 12174            ELSE
41C 12175              am_pages_in_memory := am_pages_in_memory + 1;
41E 12176            IFEND;
420 12177          IFEND;
424 12178        ELSE
424 12179          pages_not_in_memory := pages_not_in_memory + 1;
428 12180        IFEND;
428 12181        xsva.offset := xsva.offset + osv$page_size;
428 12182      FOREND;
442 12183
442 12184      ijle_p := cst_p^.ijle_p;
442 12185
442 12186  { Determine if the limit on the job's working set will allow the addition of the new pages
442 12187  { to be assigned (if pages go in the working set), and if there is memory available to assign.
442 12188
442 12189
442 12190      IF aste_p^.queue_id = mmc$pq_job_working_set THEN
456 12191        trim_pages := FALSE;
456 12192        IF rb.length <= mmc$ap_ignore_maxws_and_trim THEN
```



```
462 12193            trim_pages := TRUE;
466 12194        ELSEIF (((pages_not_in_memory + avail_pages_in_memory + am_pages_in_memory) +
488 12195              ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count) >
488 12196              cst_p^.jcb_p^.max_working_set_size) OR ((pages_not_in_memory + ijle_p^.
488 12197              job_page_queue_list [mmc$pq_job_working_set].count) > mmv$max_working_set_size) THEN
488 12198          mtp$set_status_abnormal ('MM', mme$assign_length_too_long, rb.status);
488 12199          RETURN;
49E 12200        IFEND;
49E 12201      IFEND;
49E 12202
49E 12203  { Pages that are reassignable NOW (free + available) will be decremented by new pages assigned
49E 12204  { (pages_not_in_memory) and available pages reassigned (avail_pages_in_memory).  Make sure
49E 12205  { assigning that many pages for this job will not drive memory too low.
49E 12206
49E 12207      memory_available := ((mmv$gpql [mmc$pq_free].pqle.count + mmv$gpql [mmc$pq_avail].pqle.count +
49E 12208            ijle_p^.memory_reserve_request.reserved_page_count - mmv$reserved_page_count - pages_not_in_memory -
49E 12209            avail_pages_in_memory) > mmv$aggressive_aging_level_2);
49E 12210
49E 12211      IF memory_available THEN
4C8 12212
4C8 12213  { Any pages in the requested range that are already in memory need to be marked as valid
4C8 12214  { and used; pages that are not in memory need to be assigned.  If specified, preset pages.
4C8 12215
4C8 12216        xsva := sva;
4C8 12217        WHILE requested_page_count > 0 DO
4D0 12218          #HASH_SVA (xsva, ipti, count, found);
4D2 12219          IF found THEN
4E6 12220            pte_p := ^mmv$pt_p^ [ipti];
4E6 12221            IF NOT pte_p^.v THEN
4FA 12222              pfti := (pte_p^.rma * 512) DIV osv$page_size;
4FA 12223              pfte_p := ^mmv$pft_p^ [pfti];
4FA 12224              IF aste_p^.queue_id >= mmc$pq_job_base THEN
53C 12225                pfte_p^.ijl_ordinal := cst_p^.ijl_ordinal;
548 12226              ELSE
548 12227                pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
554 12228              IFEND;
554 12229              mmp$relink_page_frame (pfti, aste_p^.queue_id);
578 12230            ELSEIF rb.preset_pages THEN
580 12231
580 12232  { Valid pages cannnot be preset; clear the valid bit, preset the page, and reset the valid bit.
580 12233
580 12234              pte_p^.v := FALSE;
584 12235            IFEND;
584 12236            IF rb.preset_pages THEN
58C 12237              mmp$preset_real_memory (xsva, fde_p^.preset_value);
5A8 12238              pte_p^.m := TRUE;
5AE 12239            IFEND;
5AE 12240            pte_p^.u := TRUE;
5AE 12241            pte_p^.v := TRUE;
5AE 12242            requested_page_count := requested_page_count - 1;
5AE 12243            xsva.offset := xsva.offset + osv$page_size;
5D4 12244          ELSE
5D4 12245
5D4 12246  { The page is not already in memory; assign a new page to memory.  MMP$ASSIGN_PAGE_FRAME
5D4 12247  { will start at the sva passed to it and assign pages until the requested count is reached or
5D4 12248  { until one of the pages in the requested range is found already in memory.  The procedure
```

```
5D4 12249 { returns the number of pages that were assigned.  "Request" the lesser of remaining_pages_to_assign
5D4 12250 { and reassignable.now from mmp$assign_page_frame.  This is done so that assign_page_frame will not
5D4 12251 { reject for no_memory.  Checks were made above to be sure that there is enough memory for the request.
5D4 12252 { The number of new pages that needs to be assigned can change if make_pt_entry is freeing some of the
5D4 12253 { pages in the total requested range to prevent a page table full condition, so pages_not_in_memory
5D4 12254 { cannot be used as the number to assign.  The total requested_page_count can include pages that are
5D4 12255 { already in the working set (they do not need to be assigned), so requested_page_count cannot be
5D4 12256 { passed to mmp$assign_page_frame.  (If there are pages already in the job working set, requested_page_
5D4 12257 { count is too large and may cause a no_memory reject.)
5D4 12258
5D4 12259           IF requested_page_count > mmv$reassignable_page_frames.now THEN
5DC 12260             remaining_pages_to_assign := mmv$reassignable_page_frames.now;
5E0 12261           ELSE
5E0 12262             remaining_pages_to_assign := requested_page_count;
5E2 12263           IFEND;
5E2 12264
5E2 12265           mmp$assign_page_frame (xsva, aste_p, remaining_pages_to_assign, 0, assigned_pages_count, first_pfti,
61A 12266                pstatus);
61A 12267
61A 12268           IF assigned_pages_count > 0 THEN
622 12269             next_pfti := first_pfti;
622 12270             psva := xsva;
622 12271             WHILE next_pfti <> 0 DO
62C 12272
62C 12273 { Preset pages if necessary; presetting can only be done one page at a time.  If the caller specified
62C 12274 { preset, then mark the page modified so that it will be writen to disk if it ages out of the working
62C 12275 { set.  Then if it is referenced the user will get the preset page.  If the preset is being done for
62C 12276 { security on the assignment of new pages to memory (to prevent a user from being able to see what had
62C 12277 { previously been on the page), then the modified bit does not need to be set.  If the page ages out
62C 12278 { of memory, it will not be written to disk.  NOTE:  Some pages may have been assigned even if a
62C 12279 { page table full condition was encountered.
62C 12280
62C 12281               IF (rb.preset_pages) OR (ste_p^.ste.r2 > 3) OR (fde_p^.stack_for_ring > 3) THEN
652 12282                 mmp$preset_real_memory (psva, fde_p^.preset_value);
66E 12283                 IF rb.preset_pages THEN
676 12284                   mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].m := TRUE;
6A2 12285                 IFEND;
6A2 12286                 psva.offset := psva.offset + osv$page_size;
6B6 12287               IFEND;
6B6 12288
6B6 12289               mmv$pt_p^ [mmv$pft_p^ [next_pfti].pti].v := TRUE;
6B6 12290               next_pfti := mmv$pft_p^ [next_pfti].link.bkw;
6B6 12291             WHILEND;
6EA 12292             requested_page_count := requested_page_count - assigned_pages_count;
6F0 12293           IFEND;
6F0 12294
6F0 12295           IF pstatus = ps_no_memory THEN
6FA 12296             mtp$error_stop ('MM - NO MEMORY IN ASSIGN PAGES');
71E 12297           ELSEIF pstatus = ps_pt_full THEN
724 12298             mtp$set_status_abnormal ('MM', mme$page_table_full, rb.status);
724 12299             RETURN;
73A 12300           IFEND;
73A 12301
73A 12302           xsva.offset := xsva.offset + (assigned_pages_count * osv$page_size);
756 12303         IFEND;
756 12304       WHILEND;
```



```
75E 12305     ELSEIF ijle_p^.statistics.tasks_not_in_long_wait = 1 THEN
766 12306
766 12307 { Memory is not available and no other tasks are ready, so cause the job to swap out.
766 12308 { Free the pages the job already has and request the total the job wants.  The requested
766 12309 { memory will be 'reserved' for the job when it swaps back in.
766 12310
766 12311       mmp$initialize_find_next_pfti (sva, rb.length, exclude_partial_pages, psc_nominal_queue, aste_p, pfti);
79A 12312
79A 12313       WHILE pfti <> 0 DO
7A2 12314         mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := FALSE;
7A2 12315         mmp$delete_pt_entry (pfti, TRUE);
7E6 12316         mmp$relink_page_frame (pfti, mmc$pq_free);
7FE 12317         mmp$find_next_pfti (pfti);
852 12318       WHILEND;
860 12319       mmp$sva_purge_all_page_map (sva);
87A 12320
87A 12321       mtp$set_status_abnormal ('MM', mme$memory_not_avail_for_assign, rb.status);
87A 12322       IF rb.waitopt = osc$wait THEN
898 12323         IF ijle_p^.entry_status <> jmc$ies_job_in_memory_non_swap THEN
8A0 12324           ijle_p^.memory_reserve_request.requested_page_count := requested_page_count;
8A0 12325           ijle_p^.memory_reserve_request.swapout_job := TRUE;
8A0 12326           jmp$set_scheduler_event (jmc$swap_job_for_memory_reserve);
8BA 12327           cst_p^.dispatch_control.call_dispatcher := TRUE;
8C2 12328         ELSE
8C2 12329           mtp$set_status_abnormal ('MM', mme$cannot_wait_for_memory, rb.status);
8D2 12330         IFEND;
8D2 12331       IFEND;
8D6 12332     ELSE
8D6 12333
8D6 12334 { Memory is not available but other tasks are ready; set status so that this task will do
8D6 12335 { a short wait before reissuing the assign_pages request.  This will let the other tasks go
8D6 12336 { idle before causing the job to swap out until memory can be found to honor the assign request.
8D6 12337
8D6 12338       mtp$set_status_abnormal ('MM', mme$wait_so_other_tasks_can_run, rb.status);
8EA 12339     IFEND;
8EA 12340
8EA 12341 { If the job had been forced to swap out and wait for memory to be reserved for it, adjust
8EA 12342 { the reserved counts now.
8EA 12343
8EA 12344     IF ijle_p^.memory_reserve_request.reserved_page_count > 0 THEN
8F2 12345       mmv$reserved_page_count := mmv$reserved_page_count - ijle_p^.memory_reserve_request.reserved_page_count;
8F2 12346       ijle_p^.memory_reserve_request.reserved_page_count := 0;
900 12347     IFEND;
900 12348
900 12349   PROCEND mmp$process_assign_pages;
  0 12350
```

MMP$PROCESS_CANCEL_RESERVE

```
o 12352
o 12353   PROCEDURE mmp$process_cancel_reserve
o 12354     (     rb: mmt$rb_assign_pages;
o 12355           cst_p: ^ost$cpu_state_table);
o 12356
o 12357     VAR
o 12358       ijle_p: ^jmt$initiated_job_list_entry;
o 12359
o 12360
o 12361     ijle_p := cst_p^.ijle_p;
4 12362
4 12363     ijle_p^.memory_reserve_request.swapout_job := FALSE;
4 12364     ijle_p^.memory_reserve_request.requested_page_count := 0;
4 12365     mmv$reserved_page_count := mmv$reserved_page_count - ijle_p^.memory_reserve_request.reserved_page_count;
4 12366     ijle_p^.memory_reserve_request.reserved_page_count := 0;
4 12367
4 12368   PROCEND mmp$process_cancel_reserve;
o 12369
```


MMP$PROCESS_MOVE_PAGES_REQUEST

```
o 12371
o 12372 { The purpose of this monitor request is to move a page frame from one PVA to another.
o 12373 { When the request completes, all pages in the range from <pva_source> to <pva_source>+
o 12374 { <length-1> will have been moved to the range of addresses specified by <pva_destination>
o 12375 { to <pva_destination>+<length-1>.
o 12376 {
o 12377 { CAUTION:  Be sure to fully understand how the 'move' is accomplished before changing this
o 12378 {           procedure.  Because mmp$delete_pt_entry USES information and mmp$make_pt_entry
o 12379 {           CHANGES information in the page frame table entry, it is necessary to 'move' the
o 12380 {           page in the following order:
o 12381 {             1.  Delete the source page table entry.
o 12382 {             2.  Change the page frame table entry to reflect destination page information.
o 12383 {             3.  Make the page table entry for the destination page.
o 12384 {             4.  If necessary, relink the page frame to the queue for the destination segment.
o 12385 {             5.  Set the valid bit on the destination page.
o 12386
o 12387   PROCEDURE [XDCL] mmp$process_move_pages_request
o 12388     (VAR rb: mmt$rb_move_pages;
o 12389          cst_p: ^ost$cpu_state_table);
o 12390
o 12391     VAR
o 12392       count: 1 .. 32,
o 12393       destination_aste_p: ^mmt$active_segment_table_entry,
o 12394       destination_fde_p: gft$locked_file_desc_entry_p,
o 12395       destination_pfti: mmt$page_frame_index,
o 12396       destination_pti: integer,
o 12397       destination_ste_p: ^mmt$segment_descriptor,
o 12398       destination_stxe_p: ^mmt$segment_descriptor_extended,
o 12399       destination_sva: ost$system_virtual_address,
o 12400       found: boolean,
o 12401       i: integer,
o 12402       job_ijle_p: ^jmt$initiated_job_list_entry,
o 12403       mpt_status: mmt$make_pt_entry_status,
o 12404       modified: boolean,
o 12405       number_of_pages_to_move: mmt$move_pages_page_count,
o 12406       page_status: gft$page_status,
o 12407       pfte_p: ^mmt$page_frame_table_entry,
o 12408       pfti: mmt$page_frame_index,
o 12409       pte_p: ^ost$page_table_entry,
o 12410       reject_offset: ost$segment_offset,
o 12411       source_aste_p: ^mmt$active_segment_table_entry,
o 12412       source_fde_p: gft$locked_file_desc_entry_p,
o 12413       source_pti: integer,
o 12414       source_ste_p: ^mmt$segment_descriptor,
o 12415       source_stxe_p: ^mmt$segment_descriptor_extended,
o 12416       source_sva: ost$system_virtual_address,
o 12417       status: syt$monitor_status,
o 12418       sva_of_last_page: ost$system_virtual_address,
o 12419       system_ijle_p: ^jmt$initiated_job_list_entry;
o 12420
o 12421
o 12422     rb.status.normal := TRUE;
4 12423
4 12424     mmp$verify_pva (^rb.pva_source, mmc$sat_read_or_write, rb.status);
30 12425    IF NOT rb.status.normal THEN
38 12426      RETURN;
```

```
3A 12427      IFEND;
3A 12428
3A 12429      mmp$convert_pva (rb.pva_source, cst_p, source_sva, source_fde_p, source_aste_p, source_ste_p,
7E 12430            source_stxe_p);
7E 12431      IF source_aste_p = NIL THEN
8C 12432        mtp$set_status_abnormal ('MM', mme$invalid_pva, rb.status);
8C 12433        RETURN;
A4 12434      IFEND;
A4 12435
A4 12436      mmp$verify_pva (^rb.pva_destination, mmc$sat_write, rb.status);
C6 12437      IF NOT rb.status.normal THEN
CE 12438        RETURN;
D0 12439      IFEND;
D0 12440
D0 12441      mmp$convert_pva (rb.pva_destination, cst_p, destination_sva, destination_fde_p, destination_aste_p,
114 12442           destination_ste_p, destination_stxe_p);
114 12443     IF destination_aste_p = NIL THEN
11E 12444       mtp$set_status_abnormal ('MM', mme$invalid_pva, rb.status);
11E 12445       RETURN;
136 12446     IFEND;
136 12447
136 12448     IF (rb.length <= 0) OR (rb.length > mmc$move_pages_max_req_length) THEN
146 12449       mtp$set_status_abnormal ('MM', mme$invalid_length_requested, rb.status);
146 12450       RETURN;
15E 12451     IFEND;
15E 12452
15E 12453     IF (rb.length MOD osv$page_size <> 0) THEN
176 12454       mtp$set_status_abnormal ('MM', mme$length_not_page_size_mult, rb.status);
176 12455       RETURN;
18E 12456     IFEND;
18E 12457
18E 12458     IF (source_sva.offset MOD osv$page_size <> 0) OR (destination_sva.offset MOD osv$page_size <> 0) THEN
1BE 12459       mtp$set_status_abnormal ('MM', mme$pva_not_on_page_boundary, rb.status);
1BE 12460       RETURN;
1D6 12461     IFEND;
1D6 12462
1D6 12463     IF (source_sva.offset + rb.length > osc$max_segment_length) OR
202 12464           (destination_sva.offset + rb.length > osc$max_segment_length) THEN
202 12465       mtp$set_status_abnormal ('MM', mme$invalid_pva_formed, rb.status);
202 12466       RETURN;
21A 12467     IFEND;
21A 12468
21A 12469     jmp$get_ijle_p (cst_p^.ijl_ordinal, job_ijle_p);
21A 12470     jmp$get_ijle_p (jmv$system_ijl_ordinal, system_ijle_p);
21A 12471
21A 12472     rb.moved_modified_page_count := 0;
21A 12473     rb.number_of_pages_moved := 0;
21A 12474
21A 12475     number_of_pages_to_move := rb.length DIV osv$page_size;
21A 12476
21A 12477     sva_of_last_page.offset := destination_sva.offset + rb.length;
21A 12478
21A 12479     IF (destination_sva.offset + number_of_pages_to_move * osv$page_size) > destination_fde_p^.file_limit THEN
2A0 12480       mtp$set_status_abnormal ('MM', mme$read_write_beyond_msl, rb.status);
2A0 12481       RETURN;
2B6 12482     IFEND;
```



```
2B6 12483
2B6 12484     IF (destination_aste_p^.queue_id = mmc$pq_wired) OR (destination_aste_p^.queue_id = mmc$pq_job_fixed) THEN
2CC 12485       mtp$set_status_abnormal ('MM', mme$wired_or_fixed_segs_illegal, rb.status);
2CC 12486       RETURN;
2E2 12487     IFEND;
2E2 12488
2E2 12489     CASE destination_fde_p^.media OF
2F6 12490     = gfc$fm_transient_segment =
2F6 12491       IF (destination_aste_p^.pages_in_memory + number_of_pages_to_move > mmv$max_pages_no_file) THEN
308 12492         page_status := gfc$ps_job_mode_work_required;
308 12493         reject_offset := destination_sva.offset;
320 12494       ELSE
320 12495         page_status := gfc$ps_page_doesnt_exist;
324 12496       IFEND;
328 12497     = gfc$fm_mass_storage_file =
328 12498       mmv$last_segment_accessed := #SEGMENT (rb.pva_destination);
328 12499       dmp$fetch_multi_page_status (destination_fde_p, destination_sva.offset,
37E 12500             number_of_pages_to_move * osv$page_size, destination_stxe_p^.file_limits_enforced,
37E 12501             reject_offset, page_status);
37E 12502     = gfc$fm_served_file =
37E 12503       dfp$fetch_multi_page_status (destination_fde_p, destination_sva.offset,
3AE 12504             number_of_pages_to_move * osv$page_size, page_status);
3AE 12505       reject_offset := destination_sva.offset + (number_of_pages_to_move - 1) * osv$page_size;
3C8 12506     CASEND;
3C8 12507
3C8 12508
3C8 12509     CASE page_status OF
414 12510     = gfc$ps_page_on_disk, gfc$ps_page_on_server, gfc$ps_page_doesnt_exist =
418 12511
418 12512 { These are ok; do nothing
418 12513
418 12514     = gfc$ps_temp_reject, gfc$ps_account_limit_exceeded =
418 12515       mtp$set_status_abnormal ('MM', mme$temporary_reject, rb.status);
418 12516       tmp$cause_task_switch;
434 12517       RETURN;
43A 12518
43A 12519     = gfc$ps_volume_unavailable =
43A 12520       mtp$set_status_abnormal ('MM', mme$volume_unavailable, rb.status);
43A 12521       RETURN;
454 12522
454 12523     = gfc$ps_server_terminated =
454 12524       mtp$set_status_abnormal ('DF', dfe$server_has_terminated, rb.status);
454 12525       RETURN;
46E 12526
46E 12527     = gfc$ps_server_allocate_required, gfc$ps_job_mode_work_required =
46E 12528       set_assign_active (destination_stxe_p, reject_offset);
496 12529       tmp$set_monitor_flag (cst_p^.taskid, mmc$mf_segment_mgr_flag, rb.status);
4B8 12530       mtp$set_status_abnormal ('MM', mme$dm_assign_active, rb.status);
4B8 12531       RETURN;
4CE 12532
4CE 12533     ELSE
4CE 12534       mtp$error_stop ('mm - unexpected DM error in assign pages');
4EE 12535     CASEND;
4EE 12536
4EE 12537
4EE 12538   /move_page/
```

```
4EE 12539
4EE 12540      FOR i := 1 TO number_of_pages_to_move DO
4F6 12541        #HASH_SVA (source_sva, source_pti, count, found);
4FC 12542        IF NOT found THEN
510 12543          mtp$set_status_abnormal ('MM', mme$source_page_not_in_memory, rb.status);
510 12544          RETURN;
526 12545        IFEND;
526 12546
526 12547        pte_p := ^mmv$pt_p^ [source_pti];
526 12548        pfti := (pte_p^.rma * 512) DIV osv$page_size;
526 12549        pfte_p := ^mmv$pft_p^ [pfti];
526 12550
526 12551        IF ((pfte_p^.active_io_count > 0) AND (pfte_p^.locked_page <> mmc$lp_page_in_lock)) THEN
56E 12552          mtp$set_status_abnormal ('MM', mme$io_active_on_move_page, rb.status);
56E 12553          RETURN;
588 12554
588 12555        ELSEIF (pte_p^.m) THEN
592 12556          IF (rb.reject_move_if_source_modified) THEN
59A 12557            mtp$set_status_abnormal ('MM', mme$modified_source_page_reject, rb.status);
59A 12558            RETURN;
5B4 12559          ELSE
5B4 12560            rb.moved_modified_page_count := rb.moved_modified_page_count + 1;
5BE 12561          IFEND;
5BE 12562        IFEND;
5BE 12563
5BE 12564  { Determine the value for modified_bit_option here; if no_change, the status of the modified bit
5BE 12565  { on the source page must be captured before the page table entry is deleted.
5BE 12566
5BE 12567        IF rb.modified_bit_option = mmc$mp_set_modified THEN
5C6 12568          modified := TRUE;
5CC 12569        ELSEIF rb.modified_bit_option = mmc$mp_clear_modified THEN
5D0 12570          modified := FALSE;
5D8 12571        ELSE
5D8 12572          modified := pte_p^.m;
5E0 12573        IFEND;
5E0 12574
5E0 12575  { Delete the source page
5E0 12576
5E0 12577        mmp$delete_pt_entry (pfti, TRUE);
5F4 12578
5F4 12579  { The destination page should not be in the page table; if it is, delete it.
5F4 12580
5F4 12581        #HASH_SVA (destination_sva, destination_pti, count, found);
5FA 12582        IF found THEN
60E 12583          destination_pfti := (mmv$pt_p^ [destination_pti].rma * 512) DIV osv$page_size;
60E 12584          mmp$delete_pt_entry (destination_pfti, TRUE);
63C 12585          mmp$relink_page_frame (destination_pfti, mmc$pq_free);
650 12586        IFEND;
650 12587
650 12588  { Change the page frame table entry to the destination page information.
650 12589
650 12590        pfte_p^.aste_p := destination_aste_p;
650 12591        pfte_p^.sva := destination_sva;
650 12592
650 12593  { Make the page table entry for the destination page.  If the page table is full, restore the source page
650 12594  { (that cannot fail) and return.  Job mode will reissue the request, starting after the last page that
```



```
650 12595  { was moved.
650 12596  { NOTE:  make_pt_entry sets the USED bit and stores the ASID and RMA in the page table entry; it
650 12597  {        also stores the PTI in the page frame table entry and increments the PAGES_IN_MEMORY count
650 12598  {        in the active segment table entry.  Make_pt_entry does NOT set the valid bit in the
650 12599  {        page table entry.
650 12600
650 12601
650 12602        mmp$make_pt_entry (destination_sva, pfti, destination_aste_p, pfte_p, mpt_status);
68C 12603        IF mpt_status = mmc$mpt_page_table_full THEN
694 12604          mmv$async_work.pt_full_aste_p := destination_aste_p;
694 12605          mmv$async_work.pt_full_sva := destination_sva;
694 12606          mmv$async_work.pt_full := TRUE;
694 12607          mmv$time_to_call_mem_mgr := 0;
694 12608          osv$time_to_check_asyn := 0;
694 12609          pfte_p^.aste_p := source_aste_p;
694 12610          pfte_p^.sva := source_sva;
694 12611          mmp$make_pt_entry (source_sva, pfti, source_aste_p, pfte_p, mpt_status);
6F0 12612          IF mpt_status <> mmc$mpt_done THEN
6F8 12613            mtp$error_stop ('MOVE_PAGES -- COULD NOT REMAKE PAGE TABLE ENTRY');
718 12614          IFEND;
718 12615          mmv$pt_p^ [pfte_p^.pti].v := TRUE;
718 12616          mtp$set_status_abnormal ('MM', mme$page_table_full, rb.status);
718 12617          RETURN;
742 12618        IFEND;
742 12619
742 12620  { Store the correct ijl ordinal in the page frame table entry.  If the queue the source
742 12621  { page was in is not the same as the queue for the destination segment, relink the page.
742 12622  { If the page is going from the shared queue to a job working set, the pft.ijl_ordinal
742 12623  { must be changed and then the page relinked.  If the page is going from a job working
742 12624  { set to the shared queue, the page must be relinked and then the pft.ijl_ordinal
742 12625  { changed.  Adjust active_io_counts if the page changes queues.
742 12626
742 12627        IF destination_aste_p^.queue_id >= mmc$pq_job_base THEN
752 12628          IF pfte_p^.queue_id >= mmc$pq_job_base THEN { MUST be the same JWS }
75A 12629            IF pfte_p^.ijl_ordinal <> destination_aste_p^.ijl_ordinal THEN
76C 12630              mtp$error_stop ('MM - MOVE PAGES - jws to jws move');
78C 12631            IFEND;
790 12632          ELSE { shared queue to job working set }
790 12633            pfte_p^.ijl_ordinal := cst_p^.ijl_ordinal;
790 12634            mmp$relink_page_frame (pfti, destination_aste_p^.queue_id);
7AC 12635            job_ijle_p^.inhibit_swap_count := job_ijle_p^.inhibit_swap_count + pfte_p^.active_io_count;
7AC 12636            job_ijle_p^.active_io_page_count := job_ijle_p^.active_io_page_count + pfte_p^.active_io_count;
7AC 12637            system_ijle_p^.active_io_page_count := system_ijle_p^.active_io_page_count -
7D6 12638                pfte_p^.active_io_count;
7D6 12639          IFEND;
7DA 12640        ELSE { destination is the shared queue }
7DA 12641          IF pfte_p^.queue_id >= mmc$pq_job_base THEN { job working set to shared queue }
7E2 12642            mmp$relink_page_frame (pfti, destination_aste_p^.queue_id);
7F6 12643            pfte_p^.ijl_ordinal := jmv$system_ijl_ordinal;
7F6 12644            job_ijle_p^.inhibit_swap_count := job_ijle_p^.inhibit_swap_count - pfte_p^.active_io_count;
7F6 12645            job_ijle_p^.active_io_page_count := job_ijle_p^.active_io_page_count - pfte_p^.active_io_count;
7F6 12646            system_ijle_p^.active_io_page_count := system_ijle_p^.active_io_page_count +
828 12647                pfte_p^.active_io_count;
828 12648          IFEND;
828 12649        IFEND;
828 12650
```

MMP$PROCESS_MOVE_PAGES_REQUEST

```
828 12651        mmv$pt_p^ [pfte_p^.pti].v := TRUE;
828 12652        mmv$pt_p^ [pfte_p^.pti].m := modified;
828 12653
828 12654        rb.number_of_pages_moved := rb.number_of_pages_moved + 1;
828 12655
828 12656        IF (rb.number_of_pages_moved < number_of_pages_to_move) THEN
854 12657          source_sva.offset := source_sva.offset + osv$page_size;
854 12658          destination_sva.offset := destination_sva.offset + osv$page_size;
878 12659        IFEND;
878 12660
878 12661      FOREND /move_page/;
87E 12662
87E 12663      IF mmv$multiple_caches OR mmv$multiple_page_maps THEN
88E 12664        mmp$purge_all_cache_map_proc;
898 12665      ELSE
898 12666        mmp$sva_purge_all_cache (destination_sva);
8AE 12667        mmp$purge_all_map_proc;
8B6 12668      IFEND;
8B6 12669
8B6 12670    PROCEND mmp$process_move_pages_request;
  0 12671
  0 12672
```


MMP$PROCESS_ASSIGN_CONTIG_MEM

```
  0 12675
  0 12676    PROCEDURE [XDCL] mmp$process_assign_contig_mem
  0 12677      (VAR rb: mmt$rb_assign_contig_memory;
  0 12678           cst_p: ^ost$cpu_state_table);
  0 12679
  0 12680      VAR
  0 12681        assign_contiguous: boolean,
  0 12682        assigned_pages: mmt$page_frame_index,
  0 12683        aste_p: ^mmt$active_segment_table_entry,
  0 12684        count: 1 .. 32,
  0 12685        fde_p: gft$locked_file_desc_entry_p,
  0 12686        first_pfti: mmt$page_frame_index,
  0 12687        found: boolean,
  0 12688        ijl_p: ^jmt$initiated_job_list_entry,
  0 12689        index: integer,
  0 12690        inhibit_io: boolean,
  0 12691        io_id: mmt$io_identifier,
  0 12692        ipti: integer,
  0 12693        mcount: integer,
  0 12694        pages_requested: 0 .. 0ffff(16),
  0 12695        pfti: integer,
  0 12696        pstatus: mmt$page_pull_status,
  0 12697        qcb_p: ^mmt$page_queue_list_entry,
  0 12698        rcount: integer,
  0 12699        save_pfti: integer,
  0 12700        starting_pfti: integer,
  0 12701        ste_p: ^mmt$segment_descriptor,
  0 12702        stxe_p: ^mmt$segment_descriptor_extended,
  0 12703        sva: ost$system_virtual_address,
  0 12704        test_sva: ost$system_virtual_address,
  0 12705        write_status: mmt$write_page_to_disk_status;
  0 12706
  0 12707      rb.status.normal := TRUE;
  4 12708      mmp$verify_pva (^rb.process_virtual_address, mmc$sat_write, rb.status);
 30 12709      IF NOT rb.status.normal THEN
 38 12710        RETURN;
 3A 12711      IFEND;
 3A 12712
 3A 12713      mmp$convert_pva (rb.process_virtual_address, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
 7E 12714      pages_requested := (#OFFSET (rb.process_virtual_address) + rb.requested_length - 1) DIV
 7E 12715           osv$page_size - (#OFFSET (rb.process_virtual_address) DIV osv$page_size) + 1;
 7E 12716      test_sva.asid := sva.asid;
 7E 12717      test_sva.offset := ((sva.offset DIV osv$page_size) * osv$page_size);
 7E 12718
 7E 12719  {Verify that none of the pages are currently assigned.
 7E 12720
 7E 12721      FOR index := 1 TO pages_requested DO
 C4 12722        #HASH_SVA (test_sva, ipti, count, found);
 C6 12723        IF found THEN
 DA 12724          mtp$set_status_abnormal ('MM', mme$pages_already_assigned, rb.status);
 DA 12725          RETURN;
 F2 12726        IFEND;
 F2 12727        test_sva.offset := test_sva.offset + osv$page_size;
 F2 12728      FOREND;
10A 12729
10A 12730
```

```
10A 12731       CASE rb.pass_count OF
120 12732       = mmc$scan_pft_for_free_or_avail, mmc$scan_pft_free_avail_notmod =
120 12733         IF rb.pass_count = mmc$scan_pft_for_free_or_avail THEN
128 12734           mmv$assign_contiguous_pass_cnt.pass_one_count := mmv$assign_contiguous_pass_cnt.pass_one_count + 1;
136 12735         ELSE
136 12736           mmv$assign_contiguous_pass_cnt.pass_two_count := mmv$assign_contiguous_pass_cnt.pass_two_count + 1;
140 12737         IFEND;
140 12738         starting_pfti := 0;
140 12739         scan_pft_for_pages (rb.pass_count, pages_requested, starting_pfti);
15C 12740
15C 12741         IF (starting_pfti = 0) THEN
164 12742           mtp$set_status_abnormal ('MM', mme$unable_to_assign_contig_mem, rb.status);
164 12743           RETURN;
17E 12744         ELSEIF rb.pass_count = mmc$scan_pft_free_avail_notmod THEN
186 12745
186 12746 { We have successfully found the requested pages. A second scan
186 12747 { of these pages is required, if a page is still usable, we go
186 12748 { ahead and remove it from the job working set.
186 12749
186 12750         /loop/
186 12751           WHILE starting_pfti <> 0 DO
18A 12752             pfti := starting_pfti;
18A 12753             assign_contiguous := TRUE;
18A 12754
18A 12755           /verify_pages_removable/
18A 12756             FOR index := 1 TO pages_requested DO
198 12757               IF (mmv$pft_p^ [pfti].queue_id = mmc$pq_avail) OR
1C4 12758                     ((mmv$pft_p^ [pfti].queue_id = mmc$pq_free) AND (mmv$pft_p^ [pfti].active_io_count = 0))
1C4 12759                     THEN
1C4 12760                 pfti := pfti + 1;
1C4 12761                 CYCLE /verify_pages_removable/;
1CE 12762               ELSE
1CE 12763                 IF ((mmv$pft_p^ [pfti].queue_id >= mmc$pq_shared_first) AND
21C 12764                       (mmv$pft_p^ [pfti].queue_id <= mmc$pq_shared_last)) OR
21C 12765                       (mmv$pft_p^ [pfti].queue_id = mmc$pq_job_working_set) THEN
21C 12766                   IF mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked THEN
23C 12767                     IF (NOT mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m) AND (mmv$pft_p^ [pfti].active_io_count =
25C 12768                           0) THEN
25C 12769                       mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, TRUE {lock ajl} , inhibit_io,
37A 12770                             ijl_p);
37A 12771                       IF NOT inhibit_io THEN
382 12772                         mmp$remove_page_from_jws (pfti, ijl_p, mcount, rcount);
3AA 12773                         pfti := pfti + 1;
3AA 12774                         jmp$unlock_ajl (ijl_p);
462 12775                         CYCLE /verify_pages_removable/;
466 12776                       IFEND;
466 12777                     IFEND;
466 12778                   IFEND;
466 12779                 IFEND;
466 12780               IFEND;
466 12781               assign_contiguous := FALSE;
466 12782               EXIT /verify_pages_removable/;
46E 12783             FOREND /verify_pages_removable/;
472 12784
472 12785             IF assign_contiguous THEN
476 12786               EXIT /loop/;
```



```
47E 12787             ELSE
47E 12788               starting_pfti := pfti;
47E 12789               scan_pft_for_pages (rb.pass_count, pages_requested, starting_pfti);
498 12790             IFEND;
498 12791           WHILEND /loop/;
4AO 12792         IFEND;
4AO 12793
4AO 12794         IF starting_pfti = 0 THEN
4A8 12795           mtp$set_status_abnormal ('MM', mme$unable_to_assign_contig_mem, rb.status);
4A8 12796           RETURN;
4BE 12797         IFEND;
4BE 12798
4BE 12799         pfti := starting_pfti;
4BE 12800         FOR index := 1 TO pages_requested DO
4CA 12801           IF mmv$pft_p^ [pfti].queue_id = mmc$pq_avail THEN
4EA 12802             mmp$delete_pt_entry (pfti, TRUE);
4FE 12803             mmp$relink_page_frame (pfti, mmc$pq_free);
512 12804           IFEND;
512 12805           pfti := pfti + 1;
512 12806         FOREND;
518 12807
518 12808         mmp$assign_page_frame (sva, aste_p, pages_requested, starting_pfti, assigned_pages, first_pfti,
558 12809               pstatus);
558 12810
558 12811 { If pages assigned was not equal to pages requested, (usually means page-table-full), free
558 12812 { any pages assigned and cause the request to be reissued.
558 12813
558 12814         IF pages_requested <> assigned_pages THEN
560 12815           pfti := first_pfti;
560 12816           IF assigned_pages > 0 THEN
568 12817             WHILE pfti <> 0 DO
56C 12818               mmp$delete_pt_entry (pfti, TRUE);
580 12819               save_pfti := mmv$pft_p^ [pfti].link.bkw;
580 12820               mmp$relink_page_frame (pfti, mmc$pq_free);
5BO 12821               pfti := save_pfti;
5BO 12822             WHILEND;
5B8 12823           IFEND;
5B8 12824           cst_p^.dispatch_control.asynchronous_interrupts_pending := TRUE;
5B8 12825           tmp$cause_task_switch;
5C8 12826           tmp$reissue_monitor_request;
5DO 12827           mmv$assign_contig_reject := mmv$assign_contig_reject + 1;
5DO 12828           RETURN;
5DC 12829         IFEND;
5DC 12830
5DC 12831 { Each of the pages assigned must be preset. After the page is preset,
5DC 12832 { the valid bit in the page table is set.
5DC 12833
5DC 12834         WHILE starting_pfti <> 0 DO
5E4 12835           mmp$preset_real_memory (mmv$pft_p^ [starting_pfti].sva, fde_p^.preset_value);
61E 12836           mmv$pt_p^ [mmv$pft_p^ [starting_pfti].pti].v := TRUE;
61E 12837           starting_pfti := mmv$pft_p^ [starting_pfti].link.bkw;
61E 12838         WHILEND;
658 12839
658 12840 { If the pages were assigned to a job-fixed segment, the pages
658 12841 { must be moved from the end of the job-fixed page queue to the
658 12842 { beginning of the job-fixed page queue. Moving the pages will
```

```
658 12843 { make swapping and job recovery of jobs with contiguous pages
658 12844 { assigned possible. A count of the job-fixed contiguous pages
658 12845 { is maintained in the IJL entry of the job.
658 12846
658 12847       IF mmv$pft_p^ [first_pfti].queue_id = mmc$pq_job_fixed THEN
67E 12848         jmp$get_ijle_p (mmv$pft_p^ [first_pfti].ijl_ordinal, ijl_p);
67E 12849         qcb_p := ^ijl_p^.job_page_queue_list [mmc$pq_job_fixed];
67E 12850         mmv$pft_p^ [qcb_p^.link.bkw].link.fwd := qcb_p^.link.fwd;
67E 12851         save_pfti := qcb_p^.link.fwd;
67E 12852         qcb_p^.link.fwd := mmv$pft_p^ [first_pfti].link.fwd;
67E 12853         mmv$pft_p^ [first_pfti].link.fwd := 0;
67E 12854         mmv$pft_p^ [qcb_p^.link.fwd].link.bkw := 0;
67E 12855         mmv$pft_p^ [save_pfti].link.bkw := qcb_p^.link.bkw;
67E 12856         qcb_p^.link.bkw := first_pfti;
67E 12857         ijl_p^.job_fixed_contiguous_pages := ijl_p^.job_fixed_contiguous_pages + pages_requested;
718 12858       IFEND;
718 12859
718 12860 { The following global variable maintains a count of all of the contiguous
718 12861 { pages currently assigned in the system. This count includes both wired and
718 12862 { job-fixed contiguous pages assigned.
718 12863
718 12864       total_contig_pages_assigned := total_contig_pages_assigned + pages_requested;
724 12865
724 12866     = mmc$scan_pft_write_mod_pages =
724 12867
724 12868 { Pass three will simply scan through the page frame table
724 12869 { and write any pages in a job working set or the shared
724 12870 { queue which are not locked and the swap status of the
724 12871 { job does not prohibit us from writing the page to disk.
724 12872
724 12873       io_id.specified := FALSE;
724 12874
724 12875 {write pages to disk
724 12876
724 12877       FOR pfti := UPPERBOUND (mmv$pft_p^) TO LOWERBOUND (mmv$pft_p^) DO
74A 12878         IF ((mmv$pft_p^ [pfti].queue_id >= mmc$pq_shared_first) AND
798 12879               (mmv$pft_p^ [pfti].queue_id <= mmc$pq_shared_last)) OR
798 12880               (mmv$pft_p^ [pfti].queue_id = mmc$pq_job_working_set) THEN
798 12881           IF mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked THEN
7B8 12882             mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, TRUE {lock ajl} , inhibit_io, ijl_p);
8D6 12883             IF NOT inhibit_io THEN
8DE 12884               gfp$mtr_get_locked_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijl_p, fde_p);
982 12885               mmp$write_page_to_disk (fde_p, pfti, ioc$write_page, io_id, mmv$multi_page_write, write_status);
9B6 12886               jmp$unlock_ajl (ijl_p);
A82 12887             IFEND;
A82 12888           IFEND;
A82 12889         IFEND;
A82 12890       FOREND;
A86 12891       mmv$assign_contiguous_pass_cnt.pass_three_count := mmv$assign_contiguous_pass_cnt.pass_three_count + 1;
A92 12892     ELSE
A92 12893       mtp$set_status_abnormal ('MM', mme$invalid_request, rb.status);
AA8 12894     CASEND;
AA8 12895
AA8 12896   PROCEND mmp$process_assign_contig_mem;
```



```
  0 12898
  0 12899   PROCEDURE scan_pft_for_pages
  0 12900     (     pass_count: mmt$assign_contig_pass_ident;
  0 12901           pages_requested: 0 .. 0ffff(16);
  0 12902       VAR starting_pfti: integer);
  0 12903
  0 12904     VAR
  0 12905       ijl_p: ^jmt$initiated_job_list_entry,
  0 12906       inhibit_io: boolean,
  0 12907       page_count,
  0 12908       pfti: integer;
  0 12909
  0 12910     IF starting_pfti = 0 THEN
  C 12911       IF mmv$image_file.active THEN
 18 12912         pfti := (osv$180_memory_limits.deadstart_upper DIV osv$page_size) - 1;
 30 12913       ELSE
 30 12914         pfti := UPPERBOUND (mmv$pft_p^) + 1;
 4A 12915       IFEND;
 4E 12916     ELSE
 4E 12917       pfti := starting_pfti;
 50 12918     IFEND;
 50 12919     page_count := 0;
 50 12920
 50 12921 { Pass one and two will check for free or available pages.
 50 12922 { In addition to searching for free or available pages, pass two
 50 12923 { will search for pages it can remove from a job working set.
 50 12924 { Pages are not removed from a job working set until we are
 50 12925 { reasonably sure we can assign the requested number of
 50 12926 { contiguous pages.
 50 12927
 50 12928   /search_loop/
 50 12929     WHILE (pfti > LOWERBOUND (mmv$pft_p^)) AND (page_count < pages_requested) DO
 6A 12930       pfti := pfti - 1;
 6A 12931
 6A 12932 {Pass one will check for free or available pages.
 6A 12933
 6A 12934       IF (mmv$pft_p^ [pfti].queue_id = mmc$pq_avail) OR ((mmv$pft_p^ [pfti].queue_id = mmc$pq_free) AND
 98 12935             (mmv$pft_p^ [pfti].active_io_count = 0)) THEN
 98 12936         page_count := page_count + 1;
 98 12937         CYCLE /search_loop/;
 A2 12938       ELSE
 A2 12939         IF pass_count = mmc$scan_pft_free_avail_notmod THEN
 A8 12940           IF ((mmv$pft_p^ [pfti].queue_id >= mmc$pq_shared_first) AND
 F6 12941                 (mmv$pft_p^ [pfti].queue_id <= mmc$pq_shared_last)) OR
 F6 12942                 (mmv$pft_p^ [pfti].queue_id = mmc$pq_job_working_set) THEN
 F6 12943             IF mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked THEN
116 12944               IF (NOT mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m) AND (mmv$pft_p^ [pfti].active_io_count = 0) THEN
136 12945                 mmp$get_inhibit_io_status (mmv$pft_p^ [pfti].ijl_ordinal, FALSE {lock ajl} , inhibit_io,
174 12946                       ijl_p);
174 12947                 IF NOT inhibit_io THEN
178 12948                   page_count := page_count + 1;
178 12949                   CYCLE /search_loop/;
17E 12950                 IFEND;
17E 12951               IFEND;
17E 12952             IFEND;
17E 12953           IFEND;
```

MMP$PROCESS_ASSIGN_CONTIG_MEM

```
182 12954          ELSE
182 12955
182 12956 { If the swap status of the job is swap resident, the jobs resources can be freed.
182 12957 { The page must be in the job working set--pages in other queues are not necessarily freed (e.g. wired).
182 12958
182 12959            jmp$get_ijle_p (mmv$pft_p^ [pfti].ijl_ordinal, ijl_p);
182 12960            IF (ijl_p^.swap_status = jmc$iss_swapped_io_complete) AND
1DA 12961                 (mmv$pft_p^ [pfti].queue_id = mmc$pq_job_working_set) THEN
1DA 12962
1DA 12963 {swap status is swap resident
1DA 12964
1DA 12965              jsp$free_swapped_jobs_memory (mmv$pft_p^ [pfti].ijl_ordinal);
1EA 12966              page_count := page_count + 1;
1EA 12967              CYCLE /search_loop/;
1FO 12968            IFEND;
1FO 12969          IFEND;
1FO 12970        IFEND;
1FO 12971        page_count := 0;
1FO 12972      WHILEND /search_loop/;
204 12973
204 12974      IF page_count < pages_requested THEN
208 12975        starting_pfti := 0;
20E 12976      ELSE
20E 12977        starting_pfti := pfti;
212 12978      IFEND;
212 12979
212 12980    PROCEND scan_pft_for_pages;
  0 12981
```


MMP$ADVISE_REQUEST_PROCESSOR

```
  0 12984
  0 12985 {-------------------------------------------------------------------------------------------------------------
  0 12986 {Name:
  0 12987 {  mmp$advise_request_processor
  0 12988 {Purpose:
  0 12989 {  This procedure processes ADVISE requests from job mode.
  0 12990 {Input:
  0 12991 {  rb - request block from job mode
  0 12992 {Output:
  0 12993 {  none
  0 12994 {Error Codes:
  0 12995 {  none
  0 12996 {notes:
  0 12997 {  - No error is generated if some (or all) of the pages are
  0 12998 {    already assigned.
  0 12999 {-------------------------------------------------------------------------------------------------------------
  0 13000
  0 13001   PROCEDURE [XDCL] mmp$advise_request_processor
  0 13002     (VAR rb: mmt$rb_advise;
  0 13003          cst_p: ^ost$cpu_state_table);
  0 13004
  0 13005     VAR
  0 13006       fde_p: gft$locked_file_desc_entry_p,
  0 13007       ijle_p: ^jmt$initiated_job_list_entry,
  0 13008       ste_p: ^mmt$segment_descriptor,
  0 13009       stxe_p: ^mmt$segment_descriptor_extended,
  0 13010       io_id: mmt$io_identifier,
  0 13011       check_aio_slowdown: boolean,
  0 13012       cptime: ost$cp_time_value,
  0 13013       page_count: integer,
  0 13014       page_in_count: mmt$page_frame_index,
  0 13015       pfti: mmt$page_frame_index,
  0 13016       aste_p: ^mmt$active_segment_table_entry,
  0 13017       pstatus: mmt$page_pull_status,
  0 13018       sva: ost$system_virtual_address;
  0 13019
  0 13020
  0 13021     rb.status.normal := TRUE;
  4 13022
  4 13023     IF NOT mmv$tables_initialized THEN
1A 13024       RETURN
1C 13025     IFEND;
1C 13026
1C 13027     io_id.specified := FALSE;
1C 13028
1C 13029
1C 13030 {Process the ADVISE OUT part of the request.
1C 13031
1C 13032   /adv_request/
1C 13033     BEGIN
1C 13034       IF ((rb.reqcode = syc$rc_advise_out) OR (rb.reqcode = syc$rc_advise_out_in)) AND
3E 13035            (rb.out_length > 0) THEN
3E 13036         mmp$verify_pva (^rb.out_pva, mmc$sat_read_or_write, rb.status);
60 13037         IF NOT rb.status.normal THEN
68 13038           EXIT /adv_request/;
6A 13039         IFEND;
```

```
6A 13040           mmp$convert_pva (rb.out_pva, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
AE 13041           IF stxe_p^.access_state <> mmc$sas_allow_access THEN
BA 13042             EXIT /adv_request/;
BC 13043           IFEND;
BC 13044           IF (aste_p^.queue_id <> mmc$pq_job_working_set) AND (aste_p^.queue_id < mmc$pq_shared_first) AND
DA 13045                (aste_p^.queue_id > mmc$pq_shared_last) THEN
DA 13046             mtp$set_status_abnormal ('MM', mme$segment_not_pageable, rb.status);
DA 13047             EXIT /adv_request/;
EC 13048           IFEND;
EC 13049
EC 13050           IF fde_p^.media = gfc$fm_transient_segment THEN {!Should this be ignored??}
F8 13051             mtp$set_status_abnormal ('MM', mme$segment_not_assigned_device, rb.status);
F8 13052             EXIT /adv_request/;
10E 13053          IFEND;
10E 13054          mmp$remove_pages_working_set (sva, rb.out_length + #OFFSET (rb.out_pva) - sva.offset, aste_p,
150 13055                page_count);
150 13056        IFEND;
150 13057
150 13058
150 13059  {Process the ADVISE IN part of the request.
150 13060
150 13061        ijle_p := cst_p^.ijle_p;
150 13062        IF ((rb.reqcode = syc$rc_advise_in) OR (rb.reqcode = syc$rc_advise_out_in)) AND (rb.in_length > 0) THEN
170 13063          mmp$verify_pva (^rb.in_pva, mmc$sat_read_or_write, rb.status);
192 13064          IF NOT rb.status.normal THEN
19A 13065            EXIT /adv_request/;
19C 13066          IFEND;
19C 13067          mmp$convert_pva (rb.in_pva, cst_p, sva, fde_p, aste_p, ste_p, stxe_p);
1E0 13068          IF stxe_p^.access_state <> mmc$sas_allow_access THEN
1EC 13069            EXIT /adv_request/;
1EE 13070          IFEND;
1EE 13071          page_count := (#OFFSET (rb.in_pva) + rb.in_length - 1) DIV osv$page_size -
1EE 13072                (#OFFSET (rb.in_pva) DIV osv$page_size) + 1;
1EE 13073
1EE 13074        /advise_in/
1EE 13075          WHILE mmv$reassignable_page_frames.now >= mmv$aggressive_aging_level_2 DO
21A 13076            mmp$page_pull_hash_sva (sva, aste_p, page_in_count, pstatus, pfti);
34E 13077            IF page_in_count = 0 THEN
356 13078              mmp$page_pull (sva, fde_p, cst_p, aste_p, stxe_p, io_id, page_count, ioc$read_page, TRUE,
3B6 13079                    page_in_count, pstatus, pfti);
3B6 13080            IFEND;
3B6 13081            page_count := page_count - page_in_count;
3B6 13082            CASE pstatus OF
44E 13083            = ps_no_memory, ps_low_on_memory =
44E 13084              tmp$cause_task_switch;
456 13085              EXIT /advise_in/;
45E 13086            = ps_io_temp_reject =
45E 13087              tmp$cause_task_switch;
466 13088              EXIT /advise_in/;
46E 13089            = ps_pt_full =
46E 13090              cst_p^.dispatch_control.asynchronous_interrupts_pending := TRUE;
46E 13091              tmp$cause_task_switch;
47A 13092              EXIT /advise_in/;
482 13093            = ps_read_beyond_eoi =
482 13094              mtp$set_status_abnormal ('MM', mme$read_beyond_eoi, rb.status);
482 13095              EXIT /advise_in/;
```



```
49E 13096            = ps_beyond_file_limit =
49E 13097              mtp$set_status_abnormal ('MM', mme$read_write_beyond_msl, rb.status);
49E 13098              EXIT /advise_in/;
4BA 13099            = ps_no_extend_permission =
4BA 13100              mtp$set_status_abnormal ('MM', mme$write_beyond_eoi_no_append, rb.status);
4BA 13101              EXIT /advise_in/;
4D6 13102            = ps_volume_unavailable, ps_server_terminated, ps_job_work_required =
4D6 13103              EXIT /advise_in/;
4DE 13104            = ps_allocate_required_on_server, ps_new_page_assigned =
4DE 13105              ijle_p^.statistics.paging_statistics.new_pages_assigned :=
4DE 13106                    ijle_p^.statistics.paging_statistics.new_pages_assigned + page_in_count;
4DE 13107              cst_p^.xcb_p^.paging_statistics.new_pages_assigned :=
4DE 13108                    cst_p^.xcb_p^.paging_statistics.new_pages_assigned + page_in_count;
4DE 13109              mmv$paging_statistics.ai_pages.new := mmv$paging_statistics.ai_pages.new + page_in_count;
4DE 13110              IF pstatus = ps_allocate_required_on_server THEN
510 13111                EXIT /advise_in/;
514 13112              IFEND;
518 13113            = ps_found_on_server =
518 13114              ijle_p^.statistics.paging_statistics.pages_from_server :=
518 13115                    ijle_p^.statistics.paging_statistics.pages_from_server + page_in_count;
518 13116              cst_p^.xcb_p^.paging_statistics.pages_from_server :=
518 13117                    cst_p^.xcb_p^.paging_statistics.pages_from_server + page_in_count;
518 13118              mmv$paging_statistics.ai_pages.server := mmv$paging_statistics.ai_pages.server + page_in_count;
542 13119
542 13120            = ps_found_in_avail, ps_found_in_avail_modified =
542 13121              ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue :=
542 13122                    ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue + page_in_count;
542 13123              cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue :=
542 13124                    cst_p^.xcb_p^.paging_statistics.pages_reclaimed_from_queue + page_in_count;
542 13125              mmv$paging_statistics.ai_pages.reclaim := mmv$paging_statistics.ai_pages.reclaim + page_in_count;
56C 13126
56C 13127            = ps_found_on_disk =
56C 13128              ijle_p^.statistics.paging_statistics.page_in_count :=
56C 13129                    ijle_p^.statistics.paging_statistics.page_in_count + page_in_count;
56C 13130              cst_p^.xcb_p^.paging_statistics.page_in_count := cst_p^.xcb_p^.paging_statistics.page_in_count +
56C 13131                    page_in_count;
56C 13132              mmv$paging_statistics.ai_pages.disk := mmv$paging_statistics.ai_pages.disk + page_in_count;
56C 13133              IF ijle_p^.active_io_requests > mmv$advise_in_aio_limit THEN
59E 13134                EXIT /advise_in/;
5A2 13135              IFEND;
5A8 13136            ELSE
5A8 13137            CASEND;
5A8 13138            IF page_count <= 0 THEN
5B0 13139              EXIT /advise_in/;
5B4 13140            IFEND;
5B4 13141            sva.offset := sva.offset + (page_in_count * osv$page_size);
5B4 13142          WHILEND /advise_in/;
5DA 13143
5DA 13144
5DA 13145  {Scan the JWS if the job cp time exceeds the aging threshold.
5DA 13146
5DA 13147           IF mmv$aging_algorithm >= 4 THEN
5E8 13148             cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
5F0 13149           ELSE
5F0 13150             cptime := ijle_p^.statistics.cp_time.time_spent_in_mtr_mode +
5FC 13151                   ijle_p^.statistics.cp_time.time_spent_in_job_mode;
```

MMP$ADVISE_REQUEST_PROCESSOR

```
5FC 13152          IFEND;
5FC 13153          IF cptime > cst_p^.jcb_p^.cptime_next_age_working_set THEN
608 13154            mmp$age_job_working_set (ijle_p, cst_p^.jcb_p);
61C 13155          IFEND;
61C 13156          check_aio_slowdown := (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count >
61C 13157                cst_p^.jcb_p^.max_working_set_size);
61C 13158          IF check_aio_slowdown OR (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count >
63C 13159                mmv$max_working_set_size) THEN
63C 13160            mmp$trim_job_working_set (ijle_p, cst_p^.jcb_p, FALSE (trim_to_swap_size:false) );
658 13161            IF check_aio_slowdown AND (ijle_p^.active_io_requests > mmv$maxws_aio_threshold) THEN
668 13162              mmv$maxws_aio_count := mmv$maxws_aio_count + 1;
668 13163              cst_p^.xcb_p^.maxws_aio_slowdown := cst_p^.xcb_p^.maxws_aio_slowdown + 1;
668 13164              ijle_p^.maxws_aio_slowdown_display := ((mmv$maxws_aio_slowdown DIV
668 13165                  mmv$jws_queue_age_interval) + 1) MOD 256;
668 13166              tmp$cause_task_switch;
6AC 13167            IFEND;
6B0 13168          ELSEIF ijle_p^.active_io_requests > mmv$aio_limit THEN
6BC 13169            tmp$cause_task_switch;
6C4 13170            mmv$aio_limit_count := mmv$aio_limit_count + 1;
6CE 13171          IFEND;
6CE 13172
6CE 13173          IF cst_p^.xcb_p^.paging_statistics.working_set_max_used < ijle_p^.
6DE 13174                job_page_queue_list [mmc$pq_job_working_set].count THEN
6DE 13175            cst_p^.xcb_p^.paging_statistics.working_set_max_used := ijle_p^.
6DE 13176                job_page_queue_list [mmc$pq_job_working_set].count;
6DE 13177            IF ijle_p^.statistics.paging_statistics.working_set_max_used < ijle_p^.
6EE 13178                job_page_queue_list [mmc$pq_job_working_set].count THEN
6EE 13179              ijle_p^.statistics.paging_statistics.working_set_max_used := ijle_p^.
6F2 13180                  job_page_queue_list [mmc$pq_job_working_set].count;
6F2 13181            IFEND;
6F2 13182          IFEND;
6F2 13183
6F2 13184 {Free queue must be replenished if the number of free+avail pages is below the threshold.
6F2 13185
6F2 13186          check_free_queues (cst_p);
73A 13187
73A 13188        IFEND; {advise-in processing}
73A 13189
73A 13190      END /adv_request/;
73A 13191
73A 13192
73A 13193
73A 13194    PROCEND mmp$advise_request_processor;
```


MMP$ASSIGN_PAGE_TO_MONITOR

```
  0 13196
  0 13197 {-----------------------------------------------------------------------
  0 13198
  0 13199 {
  0 13200 { The purpose of this request is to assign pages of real memory to the
  0 13201 { address space of monitor. If the page frames cannot be assigned because
  0 13202 { no memory is available or because of a 'page table full' condition,
  0 13203 { the request will be rejected. If the PVA is
  0 13204 { invalid or a page is already assigned, a system error halt will occur.
  0 13205 { Pages are NOT preset.
  0 13206 {
  0 13207 {       MMP$ASSIGN_PAGE_TO_MONITOR (P, PAGE_COUNT, PRESET, STATUS)
  0 13208 {
  0 13209 {   P: (INPUT) This parameter specifies the page to be assigned
  0 13210 {
  0 13211 {   PAGE_COUNT: (INPUT) This parameter specifies the number of pages to assign.
  0 13212 {
  0 13213 {   PRESET: (INPUT) This parameter is a boolean value specifying whether or
  0 13214 {                     not the pages should be preset.
  0 13215 {
  0 13216 {   STATUS: (OUTPUT) This parameter specifies request status
  0 13217 {
  0 13218
  0 13219 {-----------------------------------------------------------------------
  0 13220
  0 13221
  0 13222   PROCEDURE [XDCL] mmp$assign_page_to_monitor
  0 13223     (     p: ^cell;
  0 13224           page_count: integer;
  0 13225           preset: boolean;
  0 13226      VAR status: syt$monitor_status);
  0 13227
  0 13228     VAR
  0 13229       i: integer,
  0 13230       aste_p: ^mmt$active_segment_table_entry,
  0 13231       pstatus: mmt$page_pull_status,
  0 13232       sva: ost$system_virtual_address,
  0 13233       cell_p: ^cell,
  0 13234       count: mmt$page_frame_index,
  0 13235       pfti: mmt$page_frame_index;
  0 13236
  0 13237     status.normal := TRUE;
  4 13238     sva.offset := (#OFFSET (p) DIV osv$page_size) * osv$page_size;
  4 13239     cell_p := ^mtv$monitor_segment_table;
  4 13240     #PURGE_BUFFER (osc$pva_purge_segment_cache, cell_p);
 34 13241     sva.asid := mtv$monitor_segment_table.st [#SEGMENT (p)].ste.asid;
 34 13242
 34 13243     mmp$aste_pointer (sva.asid, aste_p);
 D4 13244     FOR i := 1 TO page_count DO
 E0 13245       mmp$assign_page_frame (sva, aste_p, 1, 0, count, pfti, pstatus);
118 13246       IF pstatus = ps_done THEN
120 13247         IF preset THEN
128 13248           mmp$preset_real_memory (sva, pmc$initialize_to_zero);
13C 13249         IFEND;
13C 13250         mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v := TRUE;
16E 13251       ELSE
```

MMP$ASSIGN_PAGE_TO_MONITOR

```
16E 13252          mmp$delete_page_from_monitor (p, i - 1, status);
18E 13253          IF (pstatus = ps_pt_full) THEN
198 13254            mtp$set_status_abnormal ('MM', mme$page_table_full, status);
1AA 13255          ELSEIF (pstatus = ps_no_memory) THEN
1BO 13256            mtp$set_status_abnormal ('MM', mme$no_free_pages, status);
1C2 13257          ELSE
1C2 13258            mtp$error_stop ('MM - unexpected reject on assign_page_to_monitor');
1E2 13259          IFEND;
1E2 13260          RETURN;
1E4 13261        IFEND;
1E4 13262        sva.offset := sva.offset + osv$page_size;
1E4 13263      FOREND;
1FC 13264
1FC 13265    PROCEND mmp$assign_page_to_monitor;
```


MMP$DELETE_PAGE_FROM_MONITOR

```
 0 13267
 0 13268 {------------------------------------------------------------------------
 0 13269
 0 13270 {
 0 13271 { This purpose of this procedure is to delete one or more pages from the address space of
 0 13272 { monitor. The deleted pages are returned to the free queue and are not updated on disk.
 0 13273 {
 0 13274 {      MMP$DELETE_PAGES_FROM_MONITOR (P, PAGE_COUNT, STATUS)
 0 13275 {
 0 13276 {   P: (INPUT) This parameter specifies the address of the first page to delete. The page must be
 0 13277 {      addressible in monitors address space.
 0 13278 {   PAGE_COUNT: (INPUT) This parameter specifies the number of pages to delete.
 0 13279 {   STATUS: (OUTPUT) This parameter is the request status.
 0 13280 {
 0 13281
 0 13282 {------------------------------------------------------------------------
 0 13283
 0 13284
 0 13285    PROCEDURE [XDCL] mmp$delete_page_from_monitor
 0 13286      (     p: ^cell;
 0 13287            page_count: integer;
 0 13288       VAR status: syt$monitor_status);
 0 13289
 0 13290      VAR
 0 13291        pva: ^cell,
 0 13292        rma: integer,
 0 13293        i: integer,
 0 13294        pfti: mmt$page_frame_index;
 0 13295
 0 13296      status.normal := TRUE;
 4 13297      pva := p;
 4 13298
 4 13299      FOR i := 1 TO page_count DO
22 13300        i#real_memory_address (pva, rma);
3A 13301        IF rma < 0 THEN
42 13302          mtp$error_stop ('MM - bad pva on delete_page_from_monitor');
62 13303        IFEND;
62 13304        pfti := rma DIV osv$page_size;
62 13305        mmp$delete_pt_entry (pfti, TRUE);
80 13306        mmp$relink_page_frame (pfti, mmc$pq_free);
94 13307        pva := #ADDRESS (1, #SEGMENT (p), #OFFSET (pva) + osv$page_size);
94 13308      FOREND;
BC 13309
BC 13310      #PURGE_BUFFER (osc$pva_purge_all_page_seg_map, pva);
CO 13311
CO 13312    PROCEND mmp$delete_page_from_monitor;
```

```
        0 13314
        0 13315   VAR
        0 13316     mmv$check_queues: [XDCL, #GATE] integer := 0;
        0 13317
        0 13318   PROCEDURE [XDCL] mmp$xcheck_queues;
        0 13319
        0 13320     PROCEDURE check_queue
        0 13321       (     qcb: mmt$page_queue_list_entry;
        0 13322             qid: integer);
        0 13323
        0 13324       VAR
        0 13325         i: integer,
        0 13326         pfti,
        0 13327         prev_pfti: integer;
        0 13328
        0 13329       pfti := qcb.link.bkw;
        8 13330       prev_pfti := 0;
        8 13331       FOR i := 1 TO qcb.count DO
       1E 13332         IF pfti = 0 THEN
       22 13333           mtp$error_stop ('MM - check queue, qcb count');
       42 13334         IFEND;
       42 13335         IF mmv$pft_p^ [pfti].link.fwd <> prev_pfti THEN
       62 13336           mtp$error_stop ('MM - check queue, bad fwd');
       82 13337         IFEND;
       82 13338         IF mmv$pft_p^ [pfti].queue_id <> qid THEN
       A2 13339           mtp$error_stop ('MM - check queue, bad qid');
       C2 13340         IFEND;
       C2 13341         prev_pfti := pfti;
       C2 13342         pfti := mmv$pft_p^ [pfti].link.bkw;
       C2 13343       FOREND;
       E6 13344       IF pfti <> 0 THEN
       EA 13345         mtp$error_stop ('MM - check queue, bad count2');
      10A 13346       IFEND;
      10A 13347     PROCEND check_queue;
        0 13348
        0 13349     VAR
        0 13350       cst_p: ^ost$cpu_state_table,
        0 13351       pit: integer,
        0 13352       last_check_time: [STATIC] integer := 0,
        0 13353       i: integer;
        0 13354
        0 13355     IF (mmv$check_queues > 0) AND mmv$tables_initialized THEN
       18 13356       pit := #READ_REGISTER (osc$pr_process_interval_timer);
       1E 13357       IF (mmv$check_queues > 1) OR (last_check_time - #FREE_RUNNING_CLOCK (0) > 1000000) THEN
       3E 13358         FOR i := 0 TO 1 DO
       48 13359           check_queue (mmv$gpq1 [i].pq1e, i);
       6C 13360         FOREND;
       6A 13361       IFEND;
       6A 13362       mtp$cst_p (cst_p);
       7A 13363       IF cst_p^.xcb_p <> NIL THEN
       9A 13364         FOR i := mmc$pq_job_fixed TO mmc$pq_job_working_set DO
       A8 13365           check_queue (cst_p^.ijle_p^.job_page_queue_list [i], i);
       D0 13366         FOREND;
       D6 13367       IFEND;
       D6 13368       #WRITE_REGISTER (osc$pr_process_interval_timer, pit);
       DC 13369       last_check_time := #FREE_RUNNING_CLOCK (0);
```



```
       EA 13370     IFEND;
       EA 13371   PROCEND mmp$xcheck_queues;
        0 13372
```

MMP$PROCESS_VOLUME_UNAVAILABLE

```
        0 13374
        0 13375   PROCEDURE [XDCL] mmp$process_volume_unavailable
        0 13376     (     xcb_p: ^ost$execution_control_block;
        0 13377           reset_p_register: boolean);
        0 13378
        0 13379     VAR
        0 13380       stxe_p: ^mmt$segment_descriptor_extended,
        0 13381       segnum: ost$segment,
        0 13382       mmv$volume_unavail_queued: [XDCL] integer := 0,
        0 13383       status: syt$monitor_status;
        0 13384
        0 13385     segnum := #SEGMENT (xcb_p^.page_wait_info.pva);
        4 13386     stxe_p := mmp$get_sdtx_entry_p (xcb_p, segnum);
        4 13387     IF (xcb_p^.xp.trap_enable <> osc$traps_enabled) OR
       60 13388
       60 13389 { Could have used p_register.ring <= 3, but that has problems too
       60 13390
       60 13391       (segnum < mmc$first_loader_predefined_seg) OR (mmc$sa_stack IN stxe_p^.software_attribute_set) THEN
       60 13392       IF tmv$ptl_p^ [xcb_p^.global_task_id.index].status < tmc$ts_first_external_queue THEN
       7A 13393         tmp$queue_task (xcb_p^.global_task_id, tmc$ts_volume_unavailable, mmv$volume_wait_queue);
       A2 13394         mmv$volume_unavail_queued := mmv$volume_unavail_queued + 1;
       A2 13395         IF (xcb_p^.system_table_lock_count > 255) OR (xcb_p^.critical_task) OR
       DC 13396               (tmv$ptl_p^ [xcb_p^.global_task_id.index].ijl_ordinal = jmv$system_ijl_ordinal) THEN
       DC 13397           mtp$step_unstep_system (syc$ic_disk_error,
      102 13398                 'ERR=VEOS9301- A critical system task has encountered an unavailable volume');
      102 13399           iop$enable_all_disk_units (status);
      116 13400         IFEND;
      116 13401       IFEND;
      11A 13402     ELSE
      11A 13403       tmp$set_monitor_flag (xcb_p^.global_task_id, mmc$mf_volume_unavailable, status);
      13C 13404     IFEND;
      13C 13405
      13C 13406     IF reset_p_register THEN
      144 13407       tmp$reissue_monitor_request;
      14C 13408     IFEND;
      14C 13409
      14C 13410   PROCEND mmp$process_volume_unavailable;
```


MMP$PROCESS_VOLUME_AVAILABLE

```
        0 13412
        0 13413   PROCEDURE [XDCL] mmp$volume_available;
        0 13414
        0 13415     VAR
        0 13416       mmv$volume_unavail_dequeued: [XDCL] integer := 0,
        0 13417       taskid: ost$global_task_id;
        0 13418
        0 13419     WHILE (mmv$volume_wait_queue.head <> 0) DO
       10 13420       tmp$dequeue_task (mmv$volume_wait_queue, taskid);
       2C 13421       mmv$volume_unavail_dequeued := mmv$volume_unavail_dequeued + 1;
       2C 13422     WHILEND;
       3E 13423
       3E 13424   PROCEND mmp$volume_available;
        0 13425
```

MMP$INCLUDE_P_REG_IN_DUMP

```
       0 13427
       0 13428   PROCEDURE [XDCL] mmp$include_p_reg_in_dump;
       0 13429
       0 13430     PROCEDURE set_page
       8 13431       (     ste_p: ^mmt$segment_descriptor;
       8 13432             offset: integer);
       8 13433
       8 13434       IF (ste_p^.ste.asid <> 0) and (offset >= 0) AND (offset <= 7fffffff(16)) THEN
      1C 13435         sva.asid := ste_p^.ste.asid;
      1C 13436         sva.offset := offset;
      1C 13437         #HASH_SVA (sva, pti, count, found);
      2A 13438         IF found THEN
      4A 13439           pfti := (mmv$pt_p^ [pti].rma * 512) DIV osv$page_size;
      4A 13440           mmv$pages_to_dump_p^ [pfti] := TRUE;
      7A 13441         IFEND;
      7A 13442       IFEND;
      7A 13443
      7A 13444     PROCEND set_page;
       0 13445
       0 13446     VAR
       0 13447       found: boolean,
       0 13448       sva: ost$system_virtual_address,
       0 13449       pti: integer,
       0 13450       count: integer,
       0 13451       cst_index: integer,
       0 13452       rma: integer,
       0 13453       ste_p: ^mmt$segment_descriptor,
       0 13454       pfti: mmt$page_frame_index,
       0 13455       xcb_p: ^ost$execution_control_block;
       0 13456
       0 13457
       0 13458     IF mmv$pages_to_dump_p <> NIL THEN
      16 13459       FOR cst_index := LOWERBOUND (mtv$cst0) TO UPPERBOUND (mtv$cst0) DO
      20 13460         IF mtv$cst0 [cst_index].xcb_p <> NIL THEN
      38 13461           xcb_p := mtv$cst0 [cst_index].xcb_p;
      38 13462           i#real_memory_address (xcb_p, rma);
      52 13463           IF rma >= 0 THEN
      5A 13464             ste_p := mmp$get_sdt_entry_p (xcb_p, xcb_p^.xp.p_register.pva.seg);
      5A 13465             set_page (ste_p, xcb_p^.xp.p_register.pva.offset);
*WARN*   13466             set_page (ste_p, xcb_p^.xp.p_register.pva.offset + osv$page_size);
      C8 13467             set_page (ste_p, xcb_p^.xp.p_register.pva.offset - osv$page_size);
      E8 13468           IFEND;
      E8 13469         IFEND;
      E8 13470       FOREND;
      EC 13471
      EC 13472     IFEND;
      EC 13473
      EC 13474   PROCEND mmp$include_p_reg_in_dump;
       0 13475
       0 13477 MODEND mmm$page_fault_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     9422   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    10850   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    11583   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    11681   Code scheduling abandoned for this block due to register jamming.
WARNING   CY 821    13466   Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY
   ****    5    warning diagnostics
```

IDENTIFIER	DEFINED ON LINE	REFERENCES							
access_rights	5536	10835	10836						
access_state	5531	11375	11376	11379	12076	12077	12080	13041	13068
active	8275	10849	12911						
active_au_offset	10791	11050/M	11051	11054	11056				
active_io_count	4171	9163	9183	9218	9420	9812	9854	9873	10006
		10044	10130	10162	10217	11801	11813	11840	11870
		12551	12635	12636	12638	12644	12645	12647	12758
		12767	12935	12944					
active_io_page_count	4542	12636/M	12636	12637/M	12637	12645/M	12645	12646/M	12646
active_io_requests	4543	11961	11971	13133	13161	13168			
adv_request	13032	13032	13038	13042	13047	13052	13065	13069	13190
advise_in	13074	13074	13085	13088	13092	13095	13098	13101	13103
		13111	13134	13139	13142				
age	4175	9993/M	10138/M	10209/M	10346/M	10357	10360	10366/M	10366
		10471/M	10735/M	11209/M					
age_exceeds_aic	2054	10393/M	10393						
age_exceeds_aif	2053	10363/M	10363						
aging_keypoints	6872	10322	10397						
aging_stack_trace	6874	10336							
ai_pages	4269	13109/M	13109	13118/M	13118	13125/M	13125	13132/M	13132
aic	10290	10311/M	10357						
aif	10291	10312/M	10360	10364/M					
aii	10289	10299/M	10302	10303/M	10357	10360	10366		
ajl_ordinal	4536	7344	7385	7557	7728	7749	8922	9706	9719
		10005	10128	10327	11053	12770	12774	12882	12884
		12886	12946						
ajl_ordinal	7714	7728/M	7728	7728/S	7728/S	7728			
ajl_ordinal	7746	7749/M	7750	7754/S	7754/S	7755			
ajl_ordinal	12676	12770/M	12770	12770/S	12770/S	12770	12882/M	12882	12882/S
		12882/S	12882						
ajl_ordinal	12899	12946/M	12946	12946/S	12946/S	12946			
ajlo	7554	7557/M	7558/S	7561/S	7561/S				
ajlo	7714	7728/P	7728/M						
ajlo	7721	7728/P							
ajlo	7743	7752/P	7755/M						
ajlo	9665	9706/M	9706/S	9706/S	9706/S	9719/M	9719/S	9719/S	9719/S
ajlo	12676	12770/P	12882/P						
ajlo	12676	12770/P	12770/M	12882/P	12882/M				
ajlo	12676	12774/M	12774/S	12774/S	12774/S	12886/M	12886/S	12886/S	12886/S
ajlo	12899	12946/P							
ajlo	12899	12946/P	12946/M						
all_requested_needed	11135	11150							
allocate_if_new	10785	10835	10886/P						
allocation_unit_size	7047	9808	10918	10918	11012	11012	11050	11050	11057
		11058	11752	11753	11863	11864			
am_pages_in_memory	12061	12160/M	12175/M	12175	12194				
amc$file_byte_limit	3055	3058	7079						
amc$max_file_id_ordinal	6514	6521							
amt$file_byte_address	3058	3042	7045						
amt$file_id_ordinal	6521	6518							
amt$file_id_sequence	6522	6519							
amt$file_identifier	6517	6481							
amt$file_limit	7079	7049							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
asid	1938	9632	9632	9648/M					
asid	2089	8598/P	8919/M	8924	8966/M	9205/M	9412	9422/P	9554/M
		9648	12716/M	12716	13241/M	13243/P	13435/M		
asid	5497	8836/M	8919	8936/M	8939/M	9554	13241	13434	13435
asid	8594	8598/M	8598/M	8598	8598	8598/S	8598/S	8598/S	8598/S
asid	8713	8717/M	8719/M	8719	8719	8720			
asid	8731	8735/M	8736/M	8736	8736	8737/S	8737/S	8738/S	8738/S
asid	8752	8754/M	8755/M	8755	8755	8756/S	8756/S	8757/S	8757/S
asid	8802	8809/P	8836						
asid	8872	8936/P	8936						
asid	8872	8938/M	8938/M	8938	8938	8938			
asid	8883	8936/P	8938/P	8939	8966				
asid	9357	9422/M	9422/M	9422	9422	9422/S	9422/S	9422/S	9422/S
asid	9364	9412/M	9419	9425/P					
asid	13222	13243/M	13243/M	13243	13243	13243/S	13243/S	13243/S	13243/S
assign_active	5541	8854	8855/M	8857/M	9697	9698	9698/M	9698/M	10877
		10877/M	10877/M	10899	10899/M	10899/M	12146	12146/M	12146/M
		12528	12528/M	12528/M					
assign_active_sfid	5354	9702	9703/M	9714	9715/M				
assign_contiguous	12681	12753/M	12781/M	12785					
assign_page_loop_count	10669	10689/M	10691	10691	10753/M	10753	10754		
assigned_page_count	10664	10677/M	10752/M	10752					
assigned_page_count	10792	11069/P	11071						
assigned_page_count	11141	11148/P	11150	11150	11153	11155/P	11160/P		
assigned_pages	12682	12808/P	12814	12816					
assigned_pages_count	12034	12265/P	12268	12292	12302				
aste_p	4176	8599	9317/M	9422	9435	9449	9794	9992	10114
		10327/P	10739/M	11212/M	12590/M	12609/M	12884/P		
aste_p	7859	7897	7898/P						
aste_p	8594	8598/M	8598	8598/M					
aste_p	8595	8598/P	8599						
aste_p	8727	8739/M	8740	8741/M					
aste_p	8803	8809/P	8818/M	8825/M	8831/M	8832/M	8834/P		
aste_p	8872	8936/P	8936/M	8936/M	8936/M	8936/M	8936/P		
aste_p	8877	8915/M	8925/M	8926	8936/P	8951/M	8953	8953	8954/M
		8955/M	8957	8961/M	8962/M	8969	8970		
aste_p	9064	9069/P							
aste_p	9253	9317							
aste_p	9357	9422/P	9422						
aste_p	9357	9422/M	9422	9422/M					
aste_p	9365	9422/P	9423	9424	9425/P				
aste_p	9752	9794/M	9795	9837/P					
aste_p	9953	9992/M	10005/P						
aste_p	10098	10114/M	10128/P						
aste_p	10661	10713/P	10717	10739	10741	10746/P			
aste_p	10780	10849	10850/P	10875	10876	10876	10930/P	10939/P	10952/P
		10964/P	11017/P	11069/P	11105/P				
aste_p	11132	11148/P							
aste_p	11192	11212	11216/P	11223					
aste_p	11252	11399	11399/P	11681	11681/P				
aste_p	11268	11368/P	11371	11399/P	11515/P	11595/P	11681/P	11683/P	11708
		11709							
aste_p	12035	12075/P	12086	12099	12099	12111	12190	12224	12229/P

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		12265/P	12311/P						
aste_p	12683	12713/P	12808/P						
aste_p	13001	13076	13076/P						
aste_p	13016	13040/P	13044	13044	13045	13054/P	13067/P	13076/P	13078/P
aste_p	13222	13243/M	13243	13243/M					
aste_p	13230	13243/P	13245/P						
asti	5466	8837/M	8926	8936/M	8940/M				
asti	7044	7838	7842/M	8811/M	8925/S	8926	8933	8933/M	8936/M
asti	7835	7838/M	7839/S	7839/S	7840/S	7841/M			
asti	8594	8598/M	8598/S						
asti	8712	8716/M	8717/S	8717/S	8718/S	8718/S			
asti	8732	8737/M	8739/S						
asti	8749	8756/M							
asti	8806	8809/P	8811	8837					
asti	8872	8933/M	8933/S	8933/S	8933/S	8933/M			
asti	8872	8936/P	8936	8936					
asti	8872	8938/M	8938/S	8938/S	8938/S	8938/S			
asti	8884	8933/P	8935	8938/P	8940	8951/S			
asti	9357	9422/M	9422/S						
asti	13222	13243/M	13243/S						
asynchronous_interrupts_pending	6387	10253/M	11823/M	11994/M	12824/M	13090/M	13186/M		
attach_count	7039	8931							
avail_pages_in_memory	12062	12161/M	12173/M	12173	12194	12209			
b	7378	7387	7387						
b	7422	7432	7433						
b	7550	7556	7556						
b	7620	7627	7628						
b	7714	7727	7727						
b	8872	8922	8922						
b	9665	9706	9706	9719	9719				
b	9942	10005	10005						
b	10088	10128	10128						
b	10776	11053	11053						
b	12676	12770	12770	12774	12774	12882	12882	12886	12886
b	12676	12884	12884						
b	12899	12946	12946						
bc	7550	7556/M	7556	7556					
bc	7621	7624/M	7625	7629					
bc	7714	7727/M	7727	7727					
bc	9665	9706/M	9706	9706	9719/M	9719	9719		
bc	12676	12770/M	12770	12770	12774/M	12774	12774	12882/M	12882
		12882	12886/M	12886	12886				
bc	12899	12946/M	12946	12946					
bits	8704	8598	8598	8598	8598	8717	8717	8718	8718
		8737	8737	8738	8738	8756	8756	8757	8757
		8938	8938	8938	8938	9422	9422	9422	9422
		13243	13243	13243	13243				
bkw	1922	9095	9101	9103	9151/M	9151	9153/M	9153	9155
		9158/S	9162/M	9194/M	9196/M	9233/M	9235/M	9298	9304
		9320	9324	9325/M	9327/M	9332/M	9336/M	9337/M	9384
		9385	9405	9413	9465/M	9465	9470	9471/M	9471
		9473/M	9474	9475/S	9482/M	9486/M	10318/M	10319/M	10319

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10324	10324	10325/S	10340	10340	10341/S	10342	10348
		10351/M	10367	10368	10371/S	10373/M	10373	10376/M	10378/M
		10461	10467	10547	10553	10590	10602	10696	10696
		10696	11026	11078	11094	11112	11181	12290	12819
		12837	12850/S	12854/M	12855/M	12855	12856/M	13329	13342
block_index	1907	7536/S	7723/S	9136	9147/S	9190/S	9268/S	9396/S	9685/S
		9837/S	10334	11709	12469/S	12470/S	12770/S	12848/S	12882/S
		12946/S	12959/S						
block_number	1906	7536/S	7723/S	9135	9147/S	9190/S	9268/S	9396/S	9685/S
		9837/S	10333	11708	12469/S	12470/S	12770/S	12848/S	12882/S
		12946/S	12959/S						
block_p	8194	7536	7723	9147	9190	9268	9396	9685	9837
		12469	12470	12770	12848	12882	12946	12959	
boffset	9753	9841/M	9844	9844	9845/M	9845	9846	9857	
buffer_descriptor	9754	9884/M	9885/M	9886/M	9893/P	9895/P			
buffer_descriptor	10794	10961/M	10962/M	10963/M	10965/P				
buffer_descriptor	11140	11151/M	11152/M	11153/M	11155/P	11161/P			
buffer_descriptor_type	2194	9884/M	10961/M	11151/M					
byte	7183	7449/M	7471/M						
bytes_to_read	10793	10918/M	10919	10920/M	10922	11056/M	11057	11058/M	11061
		11081/P	11114/P						
c	1944	9645/M							
call_dispatcher	6383	12327/M							
calls_to_age_jws	2052	10301/M	10301						
cell_p	13233	13239/M	13240						
check_aio_slowdown	11269	11953/M	11955	11958					
check_aio_slowdown	13011	13156/M	13158	13161					
check_free_queues	10241	10260	11994	13186					
check_queue	13320	13347	13359	13365					
claim_pages	9285	9285	9289	9341					
clear	7612	7563/M	7590/M	7729/M	9706/M	9719/M	12770/M	12774/M	12882/M
		12886/M	12946/M						
cmc$central_memory_element	3936	3812	4042						
cmc$central_processor_element	3937	3801							
cmc$channel_adapter_element	3938	3805	4047						
cmc$choose_any_pp	3979	3950							
cmc$choose_pp_by_barrel	3980	3952							
cmc$choose_pp_by_channel	3979	3954							
cmc$choose_specific_pp	3980	3957							
cmc$communications_element	3941	3805	4047						
cmc$controller_element	3937	3806	4048						
cmc$data_channel_element	3938	3803	4044						
cmc$external_processor_element	3941	3806							
cmc$iou_element	3939	3812							
cmc$mainframe_element	3939	3812							
cmc$max_equipment_per_channel	2411	2414							
cmc$max_esm_size	2588	3998	4003						
cmc$max_low_speed_port_number	4006	3990							
cmc$max_side_door_port_number	4007	3762	3783	3784	3991				
cmc$max_units_per_controller	3928	3931							
cmc$pem_element	3939	3813							
cmc$pp_element	3940	3809							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
cmc$storage_device_element	3940	3807
cmt$central_memory_port_number	4058	4043
cmt$channel_descriptor	3819	3804
cmt$channel_identification	3909	3902 3956
cmt$channel_ordinal	3833	3825 3910
cmt$element_name	2398	2375 3748 3749 3759 3773 3789 3790 3802 3820 3823 3892 3901 3911 3965 3986 4039 4045 4046
cmt$element_reservation	3799	3767 4072
cmt$element_state	4519	4478 4479 4509
cmt$element_type	3936	3800 4041
cmt$esm_maintenance_buffer_loc	3997	3765 3779 3993
cmt$esm_memory_size	4003	3992
cmt$hardware_address	3899	3894
cmt$model_number	4019	4014
cmt$peripheral_descriptor	3889	3808
cmt$physical_address_parts	3923	3920
cmt$physical_address_specifier	3920	3900
cmt$physical_equipment_number	2414	2385 3903
cmt$physical_unit_number	3931	3904
cmt$pp_identification	3963	3948 3958
cmt$pp_ordinal	3969	3964
cmt$pp_reservation	3944	3810
cmt$pp_reservation_choices	3979	3949
cmt$product_identification	4011	3987
cmt$product_number	4017	4012
cmt$serial_number	4022	3988
cmt$upline_connection	4025	3990 3991
condition	3333	8070/M 9005/M 9019/M 9023/M 9026/M 9028/M 9031/M 9901 9903 9905 9905 9907 9909 10969 10969 10971 10973 11166 11166 11168 11170 12078/M 12081/M 12087/M 12095/M 12100/M 12133/M 12138/M 12142/M 12148/M 12198/M 12298/M 12321/M 12329/M 12338/M 12432/M 12444/M 12449/M 12454/M 12459/M 12465/M 12480/M 12485/M 12515/M 12520/M 12524/M 12530/M 12543/M 12552/M 12557/M 12616/M 12724/M 12742/M 12795/M 12893/M 13046/M 13051/M 13094/M 13097/M 13100/M 13254/M 13256/M
condition	8065	8070
condition	8985	9005 9019 9023 9026 9028 9031
condition	12026	12078 12081 12087 12095 12100 12133 12138 12142 12148 12198 12298 12321 12329 12338
condition	12387	12432 12444 12449 12454 12459 12465 12480 12485 12515 12520 12524 12530 12543 12552 12557 12616
condition	12676	12724 12742 12795 12893
condition	13001	13046 13051 13094 13097 13100
condition	13222	13254 13256
contents	6151	11760 11771 11785
conv	6995	7003/M 7004
convert	6985	6995
count	4223	9097 9098 9098 9160/M 9160 9200/M 9200 9238/M 9238 9276 9276 9276 9276 9276 9276 9277 9277 9278 9278 9291/M 9323/M 9323 9464 9467 9483/M 9487/M 9489/M 9489 10320 10331 10463 10548 10548 10592 10594 10696 10696 10696 11953 11955

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
		11981 11983 11985 11987 12195 12197 12207 12207 13156 13158 13174 13176 13178 13180 13331
count	7609	7556/M 7556 7563 7563/M 7563 7587 7588/M 7588 7631/M 7631 7727/M 7727 7729 7729/M 7729 9706/M 9706 9706 9706/M 9706 9719/M 9719 9719 9719/M 9719 12770/M 12770 12770 12770/M 12770 12774/M 12774 12774 12774/M 12774 12882/M 12882 12882/M 12882 12886/M 12886 12886 12886/M 12886 12946/M 12946 12946 12946/M 12946
count	7866	7873
count	9263	9267/M 9270/M 9270 9272/M 9272 9276 9276 9276 9277 9287/M 9288 9291 9293/M 9293 9295 9297 9307 9307 9319/M 9319 9323 9340
count	9366	9464/M 9467/M 9469 9489 9491 9494 9496
count	9543	9623 9627 9631/M 9632 9632 9633/M 9633 9640
count	9755	9847 9866
count	10245	10247/M 10248 10251 10252
count	11252	11399 11681
count	11252	11994/M 11994 11994 11994
count	11270	11907
count	12036	12164 12218
count	12392	12541 12581
count	12684	12722
count	13001	13076
count	13001	13186/M 13186 13186 13186
count	13234	13245/P
count	13450	13437
cp_time	4836	10294 10296 10297 10452 10454 10455 11945 11947 11948 13148 13150 13151
cptime	10280	10294/M 10296/M 10299 10305
cptime	10439	10452/M 10454/M 10457
cptime	11271	11945/M 11947/M 11950
cptime	13012	13148/M 13150/M 13153
cptime_next_age_working_set	5138	10299 10305/M 10457/M 11950 13153
critical_task	5312	10826 13395
cst_index	13451	13459 13460/S 13461/S
cst_p	8052	8054/M
cst_p	8800	8815 8818
cst_p	8872	8936 8936
cst_p	8874	8892/P 8893/P 8895 8922/P 8933/P 8936/P 8955 8969
cst_p	8985	8999/M
cst_p	8991	8999/P 9009 9010 9014/P 9015/P 9018
cst_p	9054	9068/M
cst_p	9060	9068/P 9069/P
cst_p	10242	10253/M
cst_p	10779	10825 10825 10826 10834 10862 10866 10878/P 10882 10883 10898 10900 10900 10901/M 10903/P 10905 10905 10906 10907 11053/P 11066 11067 11082/M 11084/M 11086/M 11087/P
cst_p	11252	11994/M
cst_p	11254	11321/P 11327 11328/P 11336/M 11336 11350 11354 11356 11358 11360 11368/P 11370 11515/P 11583 11584 11585 11586 11595/P 11627/M 11628 11633/M 11634 11650/M 11651

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11660/M	11661	11683/P	11725/M	11725	11735/M	11736	11742/M
		11743	11767/P	11781/P	11788/P	11793/M	11793	11804/P	11805/M
		11816/P	11817/M	11823/M	11827/P	11834/M	11835	11846/P	11847/M
		11853/M	11854	11876/P	11877/M	11880/M	11881/P	11950	11951/P
		11954	11957/P	11963/M	11963	11980	11982/M	11994/P	11999/M
cst_p	12009	12013/P	12016/P						
cst_p	12028	12075/P	12147/P	12184	12196	12225	12327/M		
cst_p	12355	12361							
cst_p	12389	12429/P	12441/P	12469/P	12529/P	12633			
cst_p	12678	12713/P	12824/M						
cst_p	13001	13186/M							
cst_p	13003	13040/P	13061	13067/P	13078/P	13090/M	13107/M	13108	13116/M
		13117	13123/M	13124	13130/M	13130	13153	13154/P	13157
		13160/P	13163/M	13163	13173	13175/M	13186/P		
cst_p	13318	13362/M							
cst_p	13350	13362/P	13363	13365/P					
cyclic_age	4168	9994/M	10139/M	10210/M	10347/M	10470/M	10472	10473/M	10473
		10736/M	11210/M						
cyclic_aging_interval	5144	10450							
data	6996	7004/M	7006/S	7007/S					
data_index	6997	7005	7006/S	7007/S					
deadstart_upper	8350	9576	9590	9608	12912				
decrement_soon	9360	9441	9493						
delayed_swapin_work	4566	9271	9292	9398	9463	9480			
desc	6977	7001	7002						
destination_aste_p	12393	12441/P	12443	12484	12484	12491	12590	12602/P	12604
		12627	12629	12634/P	12642/P				
destination_fde_p	12394	12441/P	12479	12489	12499/P	12503/P			
destination_pfti	12395	12583/M	12584/P	12585/P					
destination_pti	12396	12581	12583/S						
destination_pva	5776	11084/M							
destination_ste_p	12397	12442/P							
destination_stxe_p	12398	12442/P	12500/P	12528/P					
destination_sva	12399	12441/P	12458	12464	12477	12479	12493	12499/P	12503/P
		12505	12581	12591	12602/P	12605	12658/M	12658	12666/P
dfc$active	2843	2812							
dfc$awaiting_recovery	2844	2819							
dfc$command_record_bytes	1132	1140	2757						
dfc$deactivated	2843	2828							
dfc$division_overwrite_words	1119	1147							
dfc$esm_command_record_size	1140	1148							
dfc$esm_connection	2787	2373	2784						
dfc$esm_header_record_size	1141	1148							
dfc$esm_maintenance_buf_size	1120	1151							
dfc$esm_memory_base_shift	1126	1148	1149	1149					
dfc$header_record_bytes	1131	1141							
dfc$inactive	2843	2819							
dfc$max_data_record_bytes	1135	3795							
dfc$max_esm_divisions	1129	2422							
dfc$max_esm_memory_size	1121	1150	3760	3774					
dfc$max_number_of_mainframes	1128	1113	2421						
dfc$max_number_of_queues	2853	2469	2471	2473	2858				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dfc$max_queue_entries	2854	2601	2602	2643	2644	2859			
dfc$max_req_timeout_count_value	2738	2616	2658						
dfc$max_request_buffer_entries	2418	3705							
dfc$max_retransmit_count_value	2742	2617	2659						
dfc$maximum_lifetime	2918	2915							
dfc$maximum_queue_interfaces	4076	4079							
dfc$maximum_user_buffer_area	2865	2875	2876						
dfc$maximum_user_data_area	2869	2878	2879						
dfc$min_cdcnet_errors	309	315	317	320	323	326	329	332	335
		338	341	344	364				
dfc$min_data_record_bytes	1136	1147	3795						
dfc$min_driver_test_errors	304	352	355	358	361				
dfc$min_ecc	14	20	23	26	29	32	36	40	43
		46	49	52	55	58	61	64	67
		70	73	77	80	83	87	90	93
		96	99	102	105	108	111	114	117
		120	123	126	130	134	137	140	144
		147	150	153	156	159	162	165	168
		172	175	179	182	185	188	191	194
		197	201	205	209	212	215	218	222
		226	229	232	235	238	241	245	248
		252	255	258	261	264	267	271	274
		277	281	285	290	293	297	304	306
		309	311						
dfc$min_esm_division_size	1146	1150							
dfc$min_esm_memory_size	1122	3760	3774						
dfc$min_mm_recovery_errors	311	367	371	374					
dfc$mock_connection	2788	2784							
dfc$monitor	2848	2667							
dfc$monitor_allocate	2628	2618							
dfc$monitor_io	2628	2618							
dfc$queue_assignment_strng_size	2643	2604							
dfc$recovering	2844	2833							
dfc$task_services	2848	2677							
dfc$terminated	2844	2819							
dfc$unrecovered_disk_error	3100	3128							
dfe$server_has_terminated	162	9907	10973	11170	12081/P	12142/P	12524/P		
dfp$fetch_multi_page_status	7015	12122	12503						
dfp$fetch_page_status	7209	10888							
dfp$file_server_allocation	7217	10964							
dfp$server_io	7230	9895	11160						
dft$allocated_command_buffer	2752	2751							
dft$allocated_data_rma_list	2713	2712							
dft$allocated_monitor_buffer	2776	2775							
dft$channel_definition	3788	3762	3784						
dft$channel_specification	3747	2376	2377						
dft$connection_address	2518	2513	2514						
dft$connection_descriptor	2512	2499							
dft$connection_flags	2526	2519							
dft$connection_type	2787	2372	2605						
dft$cpu_queue	2592	2486							
dft$cpu_queue_entries	2597	2594							
dft$cpu_queue_entry	2648	2597							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
dft$cpu_queue_header	2600	2593						
dft$cpu_queue_pva_entries	2473	2459						
dft$cpu_queue_pva_entry	2485	2474						
dft$data_descriptor	2573	2540	2541	2542				
dft$dma_adapter	2462	2449						
dft$driver_queue	2490	2482						
dft$driver_queue_entries	2531	2492						
dft$driver_queue_entry	2533	2531						
dft$driver_queue_header	2495	2491						
dft$driver_queue_header_flags	2502	2496						
dft$driver_queue_pva_entries	2471	2458						
dft$driver_queue_pva_entry	2481	2472						
dft$driver_queue_rma_entries	2469	2457						
dft$driver_queue_rma_entry	2476	2470						
dft$esm_base_addresses	2438	2431	3763	3777				
dft$esm_definition_table_entry	3758	3755	3768					
dft$esm_pp_information	2383	2378	2379					
dft$inquiry_message	3651	3642	3713					
dft$inquiry_tracer	3656	3652						
dft$interrupt	2507	2497						
dft$lifetime	2915	2911						
dft$mainframe_set	1113	1106	1107	4567	4568			
dft$maximum_data_bytes	3795	3764	3778					
dft$monitor_io_types	2628	2672						
dft$p_allocated_data_rma_list	2712	2624						
dft$p_command_buffer	2750	2661	2662					
dft$p_data_rma_list	2699	2664						
dft$p_queue_interface_table	2426	2369						
dft$p_send_data	2879	2679	2680					
dft$partner_status	2802	2609						
dft$pp_element_reservations	4072	2387						
dft$pp_status	2390	2384						
dft$q_interface_directory_entry	2367	2365						
dft$queue_directory	2448	2433						
dft$queue_directory_index	4079	2353						
dft$queue_entry_flags	2545	2534	2656					
dft$queue_entry_index	2859	2355						
dft$queue_entry_location	2352	2342						
dft$queue_entry_type	2848	2666						
dft$queue_index	2858	2354						
dft$queue_interface_directory	2364	2362						
dft$queue_interface_table	2428	2426						
dft$request_buffer	3700	3696						
dft$request_buffer_directory	3689	2430						
dft$request_buffer_entries	3705	3702						
dft$request_buffer_entry	3710	3706						
dft$request_buffer_entry_flags	3718	3711						
dft$response_flags	3629	3621						
dft$response_parameter	3639	3623						
dft$retransmission_digit	3662	3658						
dft$rpc_progress_record	2890	2682						
dft$send_data_size	2878	2681	2687	2689	2690	2893	2894	2899
dft$send_parameter_size	2875	2898						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dft$server_iocb_error_condition	3065	3044							
dft$server_lifetime	2911	2610							
dft$server_state	2843	2811	2846						
dft$side_door_ports	3782	3776							
dft$transaction_data	2631	2623							
dft$transaction_digit	3661	3657							
dft$transaction_state	2957	2657	2964	3653					
dfv$max_bytes_to_write	8504	9818	9819						
disk	4274	11629/M	11629	11795/M	11795	13132/M	13132		
dispatch_control	4488	10253/M	11823/M	11994/M	12327/M	12824/M	13090/M	13186/M	
display_integer_monitor	6976	7011							
display_line	6970	6972/P							
display_line	7010	7010/P							
display_monitor	6969	6973	7010						
dmc$device_manager_error_code	390	391	394	397	400	403	406	409	412
		415	418	421	424	427	430	433	436
		439	442	445	448	451	454	457	460
		463	466	469	472	475	478	481	484
		487	490	493	496	499	502	505	508
		511	514	517	520	523	526	529	532
		535	538	541	544	547	550	553	556
		559	562	565	568	571	574	577	580
		583	586	589	592	595	598	601	604
		607	610	613	616	619	622	625	628
		631	634	637	640	643	646	649	652
		655	658	661	664	667	670	673	676
		679	682	685	688	691	694	697	700
		703	706	709	712	715	718	721	724
		727	730	733	736	739	742	745	748
		751	754	757	760	763	766	769	772
		775	778	781	784	787	790	793	796
		799	802	805	808	811	814	817	820
		823	826	829	832	835	838	841	844
		847	850	853	856	859	862	865	868
		871	874	877	880	883	886	889	892
		895	898	901	904	907	910	913	916
		919	922	925	928	931	934	937	940
		943	946	949	952	958	961	964	967
		970	973	976	979	982	985	988	991
		994	997	1001	1004	1007	1012	1015	1018
		1021	1024	1027	1030	1033	1036	1039	
dme$job_mode_allocate_required	481	9909							
dme$transient_error	577	9901	10969	11166					
dme$volume_unavailable	997	9905							
dmp$fetch_multi_page_status	7243	12119	12499						
dmp$fetch_page_status	7254	10886							
dmt$chapter_number	2327	2246							
dmt$global_file_name	3132	3037							
dmt$system_file_id	2319	2245	2671	3039	4599	8276			
dpc$console_row_size	6966	6960							
dpc$top_line_message_size	6960	6955							
dpp$display_error	6954	6972	7010						
dump_a_queue	10600	10600	10607	10612					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
end_pfti	10444	10462/M	10465	10465					
entry_status	4535	12323							
eoffset	9756	9860/M	9864	9864	9865	9876/M	9876	9878	9891/P
eoi_byte_address	7045	8665	8666/M	8674	8677/M	9891	9891/M	9891	9891/M
		10016	10031/M	10151/M	10834	10919	10920	10934	10934/M
		10934	10934/M	10938	10944	10944/M	10944	10944/M	10967
		10967/M	10967	10967/M	11037	11037/M	11037	11037/M	11081
		11081/M	11081	11081/M	11114	11114/M	11114	11114/M	
eoi_modified	7068	8667/M	8676/M	9891/M	9891/M	10934/M	10934/M	10944/M	10944/M
		10967/M	10967/M	11037/M	11037/M	11081/M	11081/M	11114/M	11114/M
eoi_state	7046	8669/M	8671/M	8675	8678/M	9891/M	9891/M	9891	9891/M
		10934/M	10934/M	10934	10934/M	10944/M	10944/M	10944	10944/M
		10967/M	10967/M	10967	10967/M	11037/M	11037/M	11037	11037/M
		11081/M	11081/M	11081	11081/M	11114/M	11114/M	11114	11114/M
exclude_partial_pages	7781	12311/P							
exit_and_continue_stream	11603	11603	11644	11675	11694				
faulted_tu	11272	11372/M	11383/M	11390/M	11394	11459	11572	11754/M	11865/M
		11900	11901						
fde_p	7265	7274	7275	7276					
fde_p	7328	7350/M	7351						
fde_p	7378	7385/M	7385						
fde_p	7380	7385/P	7387/P						
fde_p	7444	7449/P							
fde_p	7832	7838	7842/M						
fde_p	8001	8007	8008						
fde_p	8661	8665	8666/M	8667/M	8669/M	8671/M	8674	8675	8676/M
		8677/M	8678/M						
fde_p	8768	8771	8777	8778	8781				
fde_p	8795	8826	8826	8826	8826				
fde_p	8795	8834	8834						
fde_p	8796	8811/M	8813	8814/M	8815/M	8826/P	8834/P		
fde_p	8872	8922/P	8922/P						
fde_p	8872	8922/M	8922						
fde_p	8872	8933	8933/M						
fde_p	8872	8936/M	8936	8936/M	8936/M	8936/P	8936/P		
fde_p	8872	8936	8936	8936	8936	8962	8962	8962	8962
fde_p	8872	8936	8936						
fde_p	8876	8922/P	8925/S	8926	8931	8931	8933/P	8936/P	8962/P
fde_p	9061	9069/P							
fde_p	9665	9686	9686	9686					
fde_p	9666	9684/P	9686/P	9694/P	9695/P	9700/P	9704/P		
fde_p	9743	9891	9891/M	9891/M	9891/M	9891	9891	9891/M	
		9891/M	9891/M						
fde_p	9744	9783	9784/P	9808	9810	9891/P	9893/P	9895/P	9899/M
		9910/P							
fde_p	9942	10005/P	10005/P						
fde_p	9942	10005/M	10005						
fde_p	9949	10005/P	10006	10015	10016	10026	10027/P	10027/P	10031/M
		10057/P							
fde_p	10088	10128/P	10128/P						
fde_p	10088	10128/M	10128						

*** 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	10095	10128/P	10130	10146	10147/P	10147/P	10151/M	10177/P	
fde_p	10271	10327/M	10327						
fde_p	10277	10327/P	10328						
fde_p	10776	10934	10934/M	10934/M	10934/M	10934/M	10934	10934	10934/M
		10934/M	10934/M	10944	10944/M	10944/M	10944/M	10944/M	10944
		10944	10944/M	10944/M	10944/M	10967	10967/M	10967/M	10967/M
		10967/M	10967	10967	10967/M	10967/M	10967/M	11037	11037/M
		11037/M	11037/M	11037/M	11037	11037	11037/M	11037/M	11037/M
		11081	11081/M	11081/M	11081/M	11081/M	11081	11081	11081/M
		11081/M	11081/M	11114	11114/M	11114/M	11114/M	11114/M	11114
		11114	11114/M	11114/M	11114/M				
fde_p	10776	11053/P	11053/P						
fde_p	10776	11053/M	11053						
fde_p	10778	10834	10848	10873	10886/P	10888/P	10907	10918	10918
		10919	10920	10930/P	10934/P	10938	10938	10941/P	10944/P
		10954/P	10967/P	11005	11006	11012	11012	11022	11022
		11023/P	11037/P	11050	11050	11057	11058	11081/P	11085
		11114/P							
fde_p	11129	11154	11155/P	11160/P					
fde_p	11273	11368/P	11515/P	11595/P	11683/P	11752	11753	11774	11775
		11779/M	11863	11864	11898				
fde_p	12037	12075/P	12094	12109	12119/P	12122/P	12237/P	12281	12282/P
fde_p	12676	12884/P	12884/P						
fde_p	12676	12884/M	12884						
fde_p	12685	12713/P	12835/P	12884/P	12885/P				
fde_p	13006	13040/P	13050	13067/P	13078/P				
file_entry_index	1879	7275/M	7340	7385	8922	9686/M	10005	10128	10327
		11053	12884						
file_hash	1881	7274/M	7339	7351	7385	7385	8922	8922	9686/M
		10005	10005	10128	10128	10327	10327	11053	11053
		12884	12884						
file_hash	7042	7274	7351	7385	8922	9686	10005	10128	10327
		11053	12884						
file_kind	7041	8771	8777	8781	8826	8826	8826	8936	8936
		8936	8962	8962	8962	11006			
file_limit	7049	10848	11775	11779/M	12094	12479			
file_limit	11274	11775/M	11776	11777/M	11779				
file_limits_enforced	5539	10867	10965/P	12120/P	12500/P				
file_limits_enforced	10796	10884/M	10887/M	10886/P					
file_offset	8277	11200	11207						
file_offset	11133	11155/P	11160/P						
find_ending_page	9863	9863	9868	9874	9877				
find_starting_page	9843	9843	9849	9855	9858				
find_the_page	11396	11396	11427	11703					
first_pfti	9265	9306/M	9325	9327	9328/S				
first_pfti	10665	10678/M	10679	10748	10749/M				
first_pfti	12038	12265/P	12269						
first_pfti	12686	12808/P	12815	12847/S	12848/S	12852/S	12853/S	12856	
flags	7035	8008	8667/M	8676/M	8778	8826	8834	8936	8936
		8962	9891/M	9891/M	10015	10934/M	10934/M	10938	10944/M
		10944/M	10967/M	10967/M	11037/M	11037/M	11081/M	11081/M	11114/M
		11114/M							
force_aggressive_aging	2043	10257/M	10257	11994/M	11994	13186/M	13186		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
force_to_global	8801	8824							
force_to_global	8872	8936							
force_to_global	8887	8931/M	8936/P	8952					
found	7867	7873	7875						
found	9367	9401	9402						
found	9544	9623	9624						
found	9757	9847	9848	9866	9867				
found	11252	11399	11399	11681	11681				
found	11275	11907	11908						
found	12039	12164	12165	12218	12219				
found	12400	12541	12542	12581	12582				
found	12687	12722	12723						
found	13001	13076	13076						
found	13447	13437	13438						
free_behind	11904	11904	11909	11916	11919				
full_scan_has_been_done	9545	9587/M	9591	9595/M					
fwd	1923	9150	9153/S	9156/M	9156	9158/M	9158	9168/M	9193
		9196/S	9198/M	9198	9199/M	9231/M	9231	9232	9235/S
		9237/M	9327/S	9328/M	9328	9331/M	9331	9333/M	9335/M
		9335	9336/S	9338/M	9473/S	9475/M	9476	9478/M	9478
		9481/M	9485/M	10350/M	10350	10369/M	10369	10371/M	10371
		10373/S	10375	10378/S	10380/M	10381/M	10462	12850/M	12850
		12851	12852/M	12852	12853/M	12854/S	13335		
gfc$fde_size	7367	7275	7340	7385	8922	9686	10005	10128	10327
		11053	12884						
gfc$fde_table_base	7365	7275	7340	7366	7385	8922	9686	10005	10128
		10327	11053	12884					
gfc$fk_catalog	7111	7123	8777	8826	8936	8962			
gfc$fk_device_file	7108	8781	8826	8936	8962	11006			
gfc$fk_job_local_file	7113	7122							
gfc$fk_job_permanent_file	7107	8771	8826	8936	8962				
gfc$fm_mass_storage_file	7126	7058	10881	11154	12117	12497			
gfc$fm_served_file	7127	7061	9810	10887	12121	12502			
gfc$fm_transient_segment	7126	9783	10874	11005	11898	12110	12490	13050	
gfc$monitor_interlocks	7399	7386	7448	8922	10005	10128	11053	12884	
gfc$ps_account_limit_exceeded	7201	10982	12132	12514					
gfc$ps_job_mode_work_required	7198	10898	10933	12112	12145	12492	12527		
gfc$ps_page_doesnt_exist	7195	10880	10908	10988	12115	12128	12495	12510	
gfc$ps_page_on_disk	7196	10916	12128	12510					
gfc$ps_page_on_server	7197	10916	12128	12510					
gfc$ps_server_allocate_required	7203	10950	12145	12527					
gfc$ps_server_terminated	7202	10985	12141	12523					
gfc$ps_temp_reject	7199	10982	12132	12514					
gfc$ps_volume_unavailable	7200	10937	12137	12519					
gfc$qs_global_shared	7155	8931							
gfc$tr_job	1892	7281	7343	7385	7840	8922	8933	9424	9435
		9449	9686	10005	10128	10327	11053	12884	
gfc$tr_system	1892	7278	7342	7385	8922	9686	10005	10128	10327
		11053	12884						
gfc$tr_system_wait_recovery	1892	8913	9029	9795					
gfp$mtr_get_fde_p	7326	7355	7385	8922	10005	10128	10327	11053	12884
gfp$mtr_get_locked_fde_p	7378	7390	8922	10005	10128	11053	12884		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gfp$mtr_get_sfid_from_fde_p	7264	7288	9686						
gfp$mtr_unlock_fde_p	7444	7452							
gft$allocation_unit_size	7085	7047							
gft$attach_count	7090	7038	7039						
gft$fde_flags	7067	7035							
gft$file_desc_entry_p	7372	7328	10277						
gft$file_descriptor_entry	7032	7024	7037	7372					
gft$file_descriptor_index	1889	1879							
gft$file_kind	7107	7041	7119	10797					
gft$file_media	7126	7057							
gft$locked_file_desc_entry_p	7024	7016	7210	7231	7244	7255	7265	7380	7444
		7488	7832	8001	8661	8768	8796	8876	9061
		9666	9744	9949	10095	10778	10803	11129	11273
		12037	12394	12412	12685	13006			
gft$open_count	7144	7040	7160						
gft$page_status	7195	7019	7212	7249	7259	10801	12045	12406	
gft$queue_status	7155	7050							
gft$segment_lock_info	7159	7043							
gft$signature_lock	7132	7033							
gft$system_file_identifier	1878	1868	2319	5355	5532	5729	6458	7218	7266
		7272	7326	7378	7833	8178	9674		
gft$table_residence	1892	1880	7336						
gft$transfer_unit_size	7096	7048							
gfv$null_sfid	8178	9702	9714						
global_task_id	5316	11360	13392/S	13393/P	13396/S	13403/P			
global_task_id	7054	8815/M	8936/M	9684/P	9700/P	9704/P	10027/P	10147/P	
global_template_file	7071	8008	8779	8826	8834	8936	8936	8962	
gtid	11276	11360/M	11361						
hash	7333	7339/M	7346/M						
hash	7378	7385/M	7385/M						
hash	8872	8922/M	8922/M						
hash	9942	10005/M	10005/M						
hash	10088	10128/M	10128/M						
hash	10271	10327/M	10327/M						
hash	10776	11053/M	11053/M						
hash	12676	12884/M	12884/M						
hcount	9372	9401							
head	4206	9217	9510	9510	10732	13419	13419		
hex_digits	6981	7006	7007						
i	11277	11935/M	11936	11937/S	11938/S				
i	12040	12163							
i	12401	12540							
i	13229	13244	13252/P						
i	13293	13299							
i	13325	13331							
i	13353	13358	13359/S	13359/P	13364	13365/S	13365/P		
i#program_error	7375	7352	7385	7563	7585	7729	8922	9706	9719
		10005	10128	10327	11053	12770	12774	12882	12884
		12886	12946						
i#real_memory_address	7477	13300	13462						
id	7610	7556	7556/M	7563	7584	7625	7629/M	7727	7727/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		7729	9706	9706/M	9706	9719	9719/M	9719	12770
		12770/M	12770	12774	12774/M	12774	12882	12882/M	12882
		12886	12886/M	12886	12946	12946/M	12946		
identifier	1262	11762/M	11764/M	11772/M	11786/M				
identifier	6143	11759/M	11770/M	11784/M					
ijl_ordinal	1861	7840	7897	8818/M	8825/M	8933	8936/M	8936/M	8955/M
		8961/M	8969	8970	9837/P	10741	11399	11681	11708
		11709	12629	13076					
ijl_ordinal	4169	7897/M	9135	9136	9147/P	9190/P	9316/M	9396/P	9685/P
		10333	10334	10741/M	11214/M	11399/M	11681/M	12225/M	12227/M
		12629	12633/M	12643/M	12769/P	12848/P	12882/P	12945/P	12959/P
		12965/P	13076/M						
ijl_ordinal	4501	8818	8933/P	8936	8955	8969	10866	12225	12469/P
		12633							
ijl_ordinal	7267	7286/M							
ijl_ordinal	7314	7282	9686						
ijl_ordinal	7532	7536/S	7536/S						
ijl_ordinal	7714	7723/S	7723/S						
ijl_ordinal	7715	7723/P	7728/P						
ijl_ordinal	8386	13396							
ijl_ordinal	9120	9147/S	9147/S	9190/S	9190/S				
ijl_ordinal	9251	9268/S	9268/S						
ijl_ordinal	9254	9268/P	9316						
ijl_ordinal	9357	9396/S	9396/S						
ijl_ordinal	9665	9685/S	9685/S						
ijl_ordinal	9665	9686/M							
ijl_ordinal	9670	9686/P							
ijl_ordinal	9743	9837/S	9837/S						
ijl_ordinal	12387	12469/S	12469/S	12470/S	12470/S				
ijl_ordinal	12676	12770/P	12770/P	12882/P	12882/P				
ijl_ordinal	12676	12770/S	12770/S	12848/S	12848/S	12882/S	12882/S		
ijl_ordinal	12899	12946/P	12946/P						
ijl_ordinal	12899	12946/S	12946/S	12959/S	12959/S				
ijl_p	9260	9268/P	9271	9272	9292	9293			
ijl_p	9368	9396/P	9397	9398	9399	9462	9463	9464	9479
		9480	9483						
ijl_p	12688	12770/P	12772/P	12774/P	12848/P	12849	12857/M	12857	12882/P
		12884/P	12886/P						
ijl_p	12905	12946/P	12959/P	12960					
ijle_p	4502	8922/P	11053/P	11370	12184	12361	13061	13365/P	
ijle_p	7327	7344							
ijle_p	7378	7385							
ijle_p	7379	7385/P							
ijle_p	7533	7536/M							
ijle_p	7551	7557	7559/P						
ijle_p	7714	7723/M							
ijle_p	7714	7728	7728/P						
ijle_p	7718	7723/P	7724	7728/P					
ijle_p	7741	7749	7751/P						
ijle_p	8872	8922/P							
ijle_p	8872	8922							
ijle_p	9120	9147/M	9190/M						
ijle_p	9126	9147/P	9148	9190/P	9191				

*** 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	9251	9268/M							
ijle_p	9357	9396/M							
ijle_p	9665	9685/M							
ijle_p	9665	9706	9706/P	9719	9719/P				
ijle_p	9671	9684/P	9693	9706/P	9712/P	9716/P	9719/P		
ijle_p	9743	9837/M							
ijle_p	9758	9837/P	9838						
ijle_p	9942	10005/P							
ijle_p	9942	10005							
ijle_p	9944	10005/P							
ijle_p	10088	10128/P							
ijle_p	10088	10128							
ijle_p	10090	10128/P							
ijle_p	10271	10327							
ijle_p	10272	10294	10296	10297	10306	10320	10327/P	10331	10337/P
		10392/P							
ijle_p	10432	10452	10454	10455	10491/P				
ijle_p	10510	10536	10547	10548	10548	10554/P			
ijle_p	10776	11053/P							
ijle_p	10776	11053							
ijle_p	11279	11370/M	11625/M	11626	11631/M	11632	11648/M	11649	11658/M
		11659	11726/M	11727	11733/M	11734	11740/M	11741	11791/M
		11792	11832/M	11833	11851/M	11852	11923/P	11945	11947
		11948	11951/P	11953	11955	11957/P	11961	11964/M	11971
		11980	11982	11984	11984	11986/M	11986		
ijle_p	12041	12184/M	12195	12196	12208	12305	12323	12324/M	12325/M
		12344	12345	12346/M					
ijle_p	12358	12361/M	12363/M	12364/M	12365	12366/M			
ijle_p	12387	12469/M	12470/M						
ijle_p	12676	12770/P	12770	12770/P	12882/P	12882	12882/P		
ijle_p	12676	12770/M	12848/M	12882/M					
ijle_p	12676	12770	12770/P	12882	12882/P				
ijle_p	12676	12774	12774/P	12886	12886/P				
ijle_p	12676	12884/P							
ijle_p	12676	12884							
ijle_p	12899	12946/P	12946	12946/P					
ijle_p	12899	12946/M	12959/M						
ijle_p	12899	12946	12946/P						
ijle_p	13007	13061/M	13105/M	13106	13114/M	13115	13121/M	13122	13128/M
		13129	13133	13148	13150	13151	13154/P	13156	13158
		13160/P	13161	13164/M	13168	13173	13175	13177	13177
		13179/M	13179						
ijlo	7714	7728/P							
ijlo	7742	7751/P							
ijlo	7834	7840							
ijlo	8872	8933							
ijlo	10804	10863/M	10866/M						
ijlo	12676	12770/P	12882/P						
ijlo	12899	12946/P							
in_length	4297	13062	13071						
in_pva	4296	13063/P	13067/P	13071	13072				
in_use	1862	7839	8598	8740	8933	9422	13243		
in_use	7313	7558	7561/M	7561	7728/M	7728	7754/M	7754	9706

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9706/M	9706	9719	9719/M	9719	12770/M	12770	12774
		12774/M	12774	12882/M	12882	12886	12886/M	12886	12946/M
		12946							
include_pages_in_dump	1869	8834/P	8936/P						
include_pages_in_dump	8003	8010/M	8012/M	8015/M					
include_pages_in_dump	8795	8834/M	8834/M	8834/M					
include_pages_in_dump	8872	8936/M	8936/M	8936/M					
increment_now	9359	9449	9490	9501					
index	2252	13392/S	13396/S						
index	9369	9399							
index	12689	12721	12756	12800					
index_p	8207	7536	7723	9147	9190	9268	9396	9685	9837
		12469	12470	12770	12848	12882	12946	12959	
inhibit_io	7717	7724/M	7725						
inhibit_io	12676	12770/M	12770	12882/M	12882				
inhibit_io	12690	12769/P	12771	12882/P	12883				
inhibit_io	12899	12946/M	12946						
inhibit_io	12906	12945/P	12947						
inhibit_swap_count	4541	12635/M	12635	12644/M	12644				
initialize_new_ast_entry	8795	8839	8936						
initiated	4245	11528/M	11529	11559/M	11560				
int	6978	7003							
int	6988	7003/M							
integ	6986	6987							
io_error	4177	9817/M	10737/M						
io_function	10784	10930/P	11105/P						
io_function	11128	11155/P							
io_id	9747	9893/P	9895/P						
io_id	9955	9963/M	10056/M	10057/P					
io_id	10100	10176/M	10177/P						
io_id	10782	10930/P	10965/P	11106/P					
io_id	11134	11156/P	11160/P						
io_id	11280	11316/M	11515/P	11595/P	11683/P				
io_id	12691	12873/M	12885/P						
io_id	13010	13027/M	13078/P						
ioc$allocate	2975	2341							
ioc$disk_min_ecc	1044	1045	1053						
ioc$max_unit_number	3670	3673							
ioc$no_error	1083	9817	10737						
ioc$read_ahead_on_server	2975	2343							
ioc$read_for_server	2974	2339							
ioc$read_from_client	2974	2340							
ioc$read_page	2969	2336	11160/P	11515/P	11595/P	11683/P	13078/P		
ioc$st_errors	1053	1054	1055	1056	1057	1058	1059	1060	1061
		1062	1063	1064	1065	1066	1067	1068	1069
		1070	1071	1072	1073	1074	1075	1076	1077
		1078							
ioc$swap_in	2970	2334							
ioc$swap_out	2970	2334							
ioc$tape_min_ecc	1046	1047							
ioc$write_for_server	2975	2339							
ioc$write_page	2969	2336	10057/P	10177/P	12885/P				
ioc$write_to_client	2975	2340							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ioe$requests_full	1062	9903	10969	11166					
ioe$unit_disabled	1076	9905	10971	11168					
iop$enable_all_disk_units	7481	13399							
iop$pager_io	7487	9893	11155						
iot$interrupt	3737	2509							
iot$io_error	1083	4177	4600						
iot$io_function	2969	2333	2670	7232	7492	9746	10784	11128	
iot$logical_unit	3673	3624							
iot$port_number	3742	3739							
iot$pp_number	3732	2450	2451						
iot$transfer_count	5817	5805							
iotype	9746	9893/P	9895/P						
ipti	7868	7873	7880/S						
ipti	9370	9401							
ipti	11252	11399	11399/S	11681	11681/S				
ipti	11278	11907	11911/S						
ipti	12042	12164	12166/S	12218	12220/S				
ipti	12692	12722							
ipti	13001	13076	13076/S						
jcb_p	4484	11583	11584	11585	11586	11950	11951/P	11954	11957/P
		12196	13153	13154/P	13157	13160/P			
jcb_p	10273	10299	10299	10300	10305/M	10305	10320		
jcb_p	10431	10449	10450/M	10450	10457/M	10457			
jcb_p	10511	10529	10530						
jmc$detached_job_wait_time_max	5163	5160							
jmc$dsw_job_recovery	1095	9271	9292	9398	9463	9480			
jmc$highest_det_job_wait_time	5173	5163	5174						
jmc$highest_prio_age_interval	5423	5414	5424						
jmc$highest_service_accumulator	4953	4954							
jmc$highest_service_factor_valu	5447	5440							
jmc$highest_working_set_size	5199	5190	5200	5202	5204	5206			
jmc$ies_job_in_memory_non_swap	4775	12323							
jmc$ies_job_swapped	4778	4787							
jmc$ies_swapin_in_progress	4777	4786							
jmc$inhibit_memory_manager_io	1198	7724	12770	12882	12946				
jmc$iss_executing	1160	9838							
jmc$iss_idle_tasks_initiated	1161	1188							
jmc$iss_swapin_io_complete	1186	1189							
jmc$iss_swapin_requested	1182	1189							
jmc$iss_swapout_complete	1181	1188							
jmc$iss_swapped_io_cannot_init	1172	1199							
jmc$iss_swapped_io_complete	1177	12960							
jmc$iss_swapped_no_io	1163	1198							
jmc$keyword_offset_maximum	4970	5191	5415						
jmc$kjl_maximum_entries	2995	2988	2989	4905					
jmc$kol_maximum_entries	3005	2990							
jmc$lock_ajl	7571	7558	7559/P	7561	7728/P	7728	7751/P	7754	9706
		9706/P	9706	9719	9719/P	9719	12770/P	12770	12774
		12774/P	12774	12882/P	12882	12886	12886/P	12886	12946/P
		12946							
jmc$max_active_jobs	2986	5396	5404	5405					
jmc$max_ajl_ord	2987	2980	2986	7763					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$max_dispatching_control	4737	4741							
jmc$max_dispatching_priority	4659	4619	4622	4623					
jmc$max_ijl_entries	1914	8241							
jmc$max_ijl_index_count	1915	8205							
jmc$maximum_job_classes	4883	4886							
jmc$maximum_job_count	3002	2995							
jmc$maximum_output_count	3012	3005							
jmc$maximum_service_classes	4986	4989							
jmc$min_dispatching_control	4736	4740							
jmc$needed_memory_available	1219	7523/S	7525/S	9180/S	9180/S	9241/S	9241/S	9506/S	9506/S
jmc$null_ajl_ordinal	7763	7728	7750	12770	12882	12946			
jmc$null_service_class	4979	4980							
jmc$priority_aging_interval_max	5414	5411							
jmc$priority_p1	4673	4620	6393						
jmc$priority_p10	4682	4621							
jmc$priority_p14	4686	4621	6393						
jmc$priority_p8	4680	4620							
jmc$required_offset	4968	5205							
jmc$reserved_ajls	2991	2986							
jmc$service_accumulator_maximum	4945	4942							
jmc$service_factor_value_max	5440	5437							
jmc$swap_job_for_memory_reserve	1213	12326/P							
jmc$system_default_offset	4969	4970	5207						
jmc$system_supplied_name_size	5021	5018							
jmc$unlimited_offset	4966	4955	5164	5175	5201	5425			
jmc$unspecified_offset	4967	5203							
jmc$working_set_size_maximum	5190	5187							
jmp$assign_ajl_with_lock	7767	7728	7751	12770	12882	12946			
jmp$check_scheduler_memory_wait	7498	7529	9180	9241	9506				
jmp$free_ajl_with_lock	7574	7559	9706	9719	12774	12886			
jmp$get_ijle_p	7532	7538	7723	9147	9190	9268	9396	9685	9837
		12469	12470	12770	12848	12882	12946	12959	
jmp$lock_ajl_with_lock	7740	7728	7758	12770	12882	12946			
jmp$recognize_thrashing	7542	10249	11994	13186					
jmp$set_scheduler_event	7545	12326							
jmp$set_scheduler_memory_event	7519	7526							
jmp$set_scheduler_memory_event	9120	9180							
jmp$set_scheduler_memory_event	9228	9241							
jmp$set_scheduler_memory_event	9357	9506							
jmp$unlock_ajl	7550	7565	9706	9719	12774	12886			
jmt$active_job_list	7320	7298							
jmt$active_job_list_entry	7312	7320							
jmt$ajl_ordinal	2980	2669	4482	4536	6884	7554	7721	7743	7746
		7772							
jmt$delayed_swapin_work	1099	1103	4566						
jmt$detached_job_wait_time	5160	5145							
jmt$dispatching_control	4707	5379							
jmt$dispatching_control_index	4740	4697	4707						
jmt$dispatching_controls	4710	4708							
jmt$dispatching_priority	4619	4474	4548	4698	4699	4700	4712	5327	5329
		8394	8395						
jmt$ijl_block_index	1911	1907	8207						
jmt$ijl_block_number	1910	1906	8195	8196					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$ijl_dispatching_control	4696	4549							
jmt$ijl_entry_status	4773	4535							
jmt$ijl_ordinal	1905*	1861	2198	2335	4169	4501	4555	4583	5036
		5037	5135	6476	6885	7267	7271	7292	7314
		7532	7647	7715	7742	7769	7834	8109	8219
		8239	8240	8386	9254	9670	9759	10804	
jmt$ijl_p	8193	8188							
jmt$ijl_page_fault_count	4802	4797	4798	4799					
jmt$ijl_page_stats	4796	4792							
jmt$ijl_service_class_stats	4790	4570							
jmt$ijl_statistics	4835	4569							
jmt$ijl_swap_count	4811	4807	4808						
jmt$ijl_swap_counts	4806	4589	4793						
jmt$ijl_swap_status	1159	4538	4539	4540					
jmt$initiated_job_list_block	8204	8210							
jmt$initiated_job_list_entry	4532	4502	5061	5134	6886	7315	7327	7379	7533
		7551	7575	7718	7741	8108	8124	8131	8162
		8207	9126	9260	9368	9671	9673	9758	9944
		10090	10272	10432	10510	11279	12041	12358	12402
		12419	12688	12905	13007				
jmt$initiated_job_list_p	8210	8194							
jmt$input_file_location	4925	4920							
jmt$jl_job_leveler_state	3022	3017							
jmt$jl_job_leveler_status	3016	2608							
jmt$job_abort_disposition	4934	4918							
jmt$job_class	4886	4594							
jmt$job_control_block	5116	4484	10273	10431	10511				
jmt$job_mode	4889	4551							
jmt$job_priority	4894	4591	4592	5388	5389	5390	5391		
jmt$job_recovery_disposition	4937	4919							
jmt$job_sched_event_selections	1224	7503							
jmt$job_scheduler_event	1222	7508							
jmt$job_scheduler_events	1202	1222	1224	7545					
jmt$job_system_id	5179	5131							
jmt$kjl_index	4905	4537	5179						
jmt$maximum_active_jobs	5396	5373							
jmt$priority_aging_interval	5411	5381							
jmt$queue_file_ijl_information	4917	4576							
jmt$scheduling_data	4582	4560							
jmt$scheduling_priority	5387	5380							
jmt$service_accumulator	4942	4584	4585	4586	5371	5372			
jmt$service_class_index	4989	4595	5364	5374					
jmt$service_class_name	5429	5366	5367						
jmt$service_factor_value	5437	5375							
jmt$service_factors	5433	5375							
jmt$swap_data	4598	4562							
jmt$swapout_reasons	4992	4590							
jmt$swapped_job_entry	5007	4607	5062	5154	9252				
jmt$system_supplied_name	5018	4533	5129	6475					
jmt$task_time_slice	4750	4730	4731						
jmt$time_slice_values	4729	4714	5340						
jmt$user_supplied_name	5183	5130							
jmt$working_set_size	5187	5141	5142						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
jmv$ajl_p	7298	7282 7558 7561/M 7561 7728/M 7728 7754/M 7754
		9686 9706 9706/M 9706 9719 9719/M 9719 12770/M
		12770 12774 12774/M 12774 12882/M 12882 12886 12886/M
		12886 12946/M 12946
jmv$ijl_p	8188	7536 7723 9147 9190 9268 9396 9685 9837
		12469 12470 12770 12848 12882 12946 12959
jmv$job_sched_events_selected	7503	7523
jmv$job_sched_events_selected	9120	9180
jmv$job_sched_events_selected	9228	9241
jmv$job_sched_events_selected	9357	9506
jmv$job_scheduler_event	7508	7525
jmv$job_scheduler_event	9120	9180
jmv$job_scheduler_event	9228	9241
jmv$job_scheduler_event	9357	9506
jmv$memory_needed_by_scheduler	7514	7524
jmv$memory_needed_by_scheduler	9120	9180
jmv$memory_needed_by_scheduler	9228	9241
jmv$memory_needed_by_scheduler	9357	9506
jmv$null_ijl_ordinal	7292	7279 9100/P 9280/P 9686 10696/P 10863
jmv$system_ijl_ordinal	8219	8825 8936 8961 8970 11214 12227 12470/P 12643
		13396
job_fixed_asid	4545	7728/P 7751/P 12770/P 12882/P 12946/P
job_fixed_contiguous_pages	4571	9272 9293 9397 9399 9462 9464 9479 9483
		12857/M 12857
job_ijle_p	12402	12469/P 12635/M 12635 12636/M 12636 12644/M 12644 12645/M
		12645
job_monitor_taskid	4550	9712/P 9716/P
job_page_queue_count	5009	9270 9287
job_page_queue_list	4561	9148 9191 10306 10320 10331 10547 10548 10548
		11953 11955 11981 11983 11985 11987 12195 12197
		12849 13156 13158 13174 13176 13178 13180 13365/P
job_page_queue_list	9255	9291/M 9324 9325/M 9327/S 9328 9331/M 9335/M
job_page_queue_list	9358	9384 9385 9464 9465/M 9467 9471 9473 9475/S
		9478 9481/M 9482/M 9483/M 9485/M 9486/M 9487/M
job_termination	9361	9397 9435 9463 9480
jsc$isqi_swapped_io_completed	5041	5043
jsc$isqi_swapped_io_not_init	5040	5043
jsp$free_swapped_jobs_memory	7646	9100 9280 10696 12965
jst$changed_asid_entry	5084	5075
jst$ijl_swap_queue_id	5040	5035 8246
jst$ijl_swap_queue_link	5034	4544
jst$ijl_swap_queue_list	8246	8230
jst$ijl_swap_queue_list_entry	8238	8246
jst$io_control_information	5048	4563
jst$swap_file_descriptor	5060	4564
jst$swapped_page_descriptor	5069	5067
jst$swapped_page_descriptors	5066	5063
jsv$free_working_set_on_swapout	8226	10468
jsv$max_pages_first_swap_task	8252	10537
jsv$maximum_pages_to_swap	8256	10539
keypoint_page_fault_status	11281	11329/P 11330

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
last_check_time	13352	13357 13369/M
last_contiguous_pfti	9371	9400/M 9465/S 9481
last_faulted_tu	11282	11393/M 11394
last_page_fault	5545	11391 11392/M
last_page_fault	11283	11391/M 11393 11540
last_pfti	10516	10552/M 10554/P
last_pfti_index	8315	7681 7694 7695 7699 7965 7966 7971 7984/M
		7993/M 8027/S 8028/M 8028 9972 9972 9972 9976
		9976 9976 9988 9988 9988 10063 10063 10063
		10317/M 10358/S 10358/M 10358 10361/S 10361/M 10361 10389
		10460/M 10479/S 10479/M 10479 10489 11902/M 11921 12317
		12317 12317
last_segment_number	7053	8814/M 8936/M 9694/P 9695/P 10328 11085
length	4323	12091 12192 12311/P
length	4342	12448 12448 12453 12463 12464 12475 12477
length	9760	9878/M 9886 9893/P 9895/P 9919
limit	10430	10472
line	6998	7001/M 7006/M 7007/M 7010/P
line_index	6999	7002/M 7006/M 7007/M 7008/M 7008 7010/P
link	4166	9150 9151 9153/S 9153/M 9153 9155 9156 9158/S
		9158/M 9158 9162/M 9168/M 9196/M 9198/M 9231/M 9235/M
		9320 9327/M 9328/M 9335 9336/S 9336/M 9338/M 9405
		9413 9465 9473/M 9475/M 10326 10343 10371/M 10373/M
		10374/M 10378/M 10380/M 10467 10553 10602 11026 11078
		11094 11112 11181 12290 12819 12837 12850/M 12852
		12853/M 12854/M 12855/M 13335 13342
link	4222	9095 9101 9103 9151/M 9156/M 9193 9194/M 9196/S
		9198 9199/M 9231 9232 9233/M 9235/S 9237/M 9298
		9304 9324 9325/M 9327/S 9328 9331/M 9331 9332/M
		9333/M 9335/M 9337/M 9384 9385 9465/M 9470 9471/M
		9471 9473/S 9474 9475/S 9476 9478/M 9478 9481/M
		9482/M 9485/M 9486/M 10319 10369/M 10376/M 10461 10462
		10547 10590 10696 10696 10696 12850/S 12850 12851
		12852/M 12854/S 12855 12856/M 13329
link	10283	10319/M 10323 10340 10340 10341/S 10342 10343/M 10350
		10368 10369 10371/S 10371 10373/S 10373
lock	7378	7387
lock	7419	7432
lock	7444	7449/M
lock	7459	7471/M
lock	7550	7556 7556 7556/M 7556/M 7556
lock	7550	7563 7563 7563/M 7563 7563/M
lock	7581	7584 7587 7588/M 7588 7590/M
lock	7617	7625 7627 7629/M 7631/M 7631
lock	7714	7727 7727 7727/M 7727/M 7727
lock	7714	7729 7729 7729/M 7729 7729/M
lock	7716	7726
lock	8872	8922
lock	9665	9706 9706 9706/M 9706/M 9706 9719 9719 9719/M
		9719/M 9719
lock	9665	9706 9706 9706/M 9706 9706/M 9719 9719 9719/M
		9719 9719/M
lock	9942	10005

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	10088	10128							
lock	10776	11053							
lock	12676	12770	12882						
lock	12676	12770	12770	12770/M	12770/M	12770	12774	12774	12774/M
		12774/M	12774	12882	12882	12882/M	12882/M	12882	12886
		12886	12886/M	12886/M	12886				
lock	12676	12770	12770	12770/M	12770	12770/M	12774	12774	12774/M
		12774	12774/M	12882	12882	12882/M	12882	12882/M	12886
		12886	12886/M	12886	12886/M				
lock	12676	12884							
lock	12899	12946							
lock	12899	12946	12946	12946/M	12946/M	12946			
lock	12899	12946	12946	12946/M	12946	12946/M			
locked	7185	7387	7432	8922	10005	10128	11053	12884	
locked	7608	7556	7627	7727	9706	9719	12770	12774	12882
		12886	12946						
locked_for_write	7161	10006	10130						
locked_page	4172	7886	7886	7887	7889	9852	9871	10109	10353
		10474	10740/M	11213/M	11399	11399	11399	11399	11681
		11681	11681	11681	11913	12551	12766	12881	12943
		13076	13076	13076	13076				
loop	12750	12750	12786	12791					
low_on_page_frames	10798	10824/M	10825	10917					
lower	8349	9575	9594	11200	11207				
lsva	9761	9839/M	9846/M	9847	9865/M	9866			
lu_link	10284	10318/M	10348	10350/M	10351/M	10367	10374	10375	10378/S
		10381/M							
lu_pfte_p	10286	10349/M	10380/M						
m	1946	9142	9173	9182	9647/M	9848	9867	10006	10043
		10130	10161	10214	11077/M	11111/M	12238/M	12284/M	12555
		12572	12652/M	12767	12944				
max_bytes_to_write	9762	9808/M	9818	9819/M	9835	9835	9836		
max_working_set_size	5141	10529	10530	11583	11584	11585	11586	11954	12196
		13157							
maximum_pages_to_swap	10518	10537/M	10539/M	10542	10543				
maxws_aio_slowdown	5343	11963/M	11963	13163/M	13163				
maxws_aio_slowdown_display	4574	11964/M	13164/M						
mcount	9956	9961/M	10059/M	10059	10067	10069	10071		
mcount	10091	10107/M	10181/M						
mcount	10287	10390/M	10392/P	10398					
mcount	10519	10554/P							
mcount	11284	11923/P							
mcount	12693	12772/P							
media	7057	9783	9810	10873	11005	11154	11898	12109	12489
		13050							
memory_available	12043	12207/M	12211						
memory_keypoints	6869	9132	9308	9414	11352	11705			
memory_reserve_request	4554	12208	12324/M	12325/M	12344	12345	12346/M	12363/M	12364/M
		12365	12366/M						
min_working_set_size	5142	10320							
minimum	4228	10594							
minimum_working_set	10434	10463							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$	1271	1277	1280	1283	1286	1289	1292	1295	1299
		1302	1305	1308	1311	1314	1317	1321	1324
		1327	1330	1333	1336	1340	1343	1346	1349
		1352	1355	1358	1361	1364	1367	1370	1373
		1376	1379	1382	1385	1388	1391	1394	1397
		1400	1404	1407	1410	1413	1416	1419	1422
		1425	1428	1431	1434	1437	1440	1443	1446
		1449	1452	1455	1458	1461	1464	1468	1472
		1475	1479	1482	1485	1488	1491	1494	1497
		1500	1503	1506	1509	1512	1515	1518	1521
		1524	1527	1530	1533	1536	1540	1544	1547
		1550	1553	1556	1559	1562	1565	1568	1571
		1574	1577	1580	1583	1586	1589	1592	1595
		1598	1602	1605	1608				
mmc$ap_ignore_maxws_and_trim	12031	12192							
mmc$aprc_assign	4331	4321	12012						
mmc$aprc_cancel_reserve	4331	4326	12015						
mmc$assign_active_escaped	5573	8857	9698	10877	10899	12146	12528		
mmc$assign_active_null	5572	5573	8854	9698	9699	10877	10899	12146	12528
mmc$bd_explicit_io	2203	2195							
mmc$bd_job_swapping_io	2202	2197							
mmc$bd_paging_io	2202	2195	9884	10961	11151				
mmc$cell_pointer	5671	5676							
mmc$debug	6898	8926	8969	9164	9173	9183			
mmc$eoi_actual	7173	8675	9891	10934	10944	10967	11037	11081	11114
mmc$eoi_rounded	7173	8671	8678	9891	9891	10934	10934	10944	10944
		10967	10967	11037	11037	11081	11081	11114	11114
mmc$eoi_uncertain	7173	8669	9891	10934	10944	10967	11037	11081	11114
mmc$first_loader_predefined_seg	1227	1229	8006	8834	8936	13391			
mmc$heap_pointer	5672	5680							
mmc$iocb_table_size	4143	4136							
mmc$iorc_await_io_completion	3325	3318							
mmc$iorc_write_pages	3325	3316							
mmc$irs_active	3247	3241							
mmc$irs_complete	3247	3242							
mmc$irs_none	3247	3241							
mmc$kpfs_disable_keypoints	4149	11335							
mmc$kpfs_invalid_keypoint	4149	11339							
mmc$kpfs_normal	4148	11332							
mmc$kw_asid	5597	5633							
mmc$kw_clear_space	5595	5620							
mmc$kw_current_segment_length	5594	5614							
mmc$kw_error_exit_procedure	5596	5624							
mmc$kw_gl_key	5596	5618							
mmc$kw_hardware_attributes	5598	5627							
mmc$kw_inheritance	5598	5635							
mmc$kw_max_segment_length	5595	5616							
mmc$kw_preset_value	5597	5622							
mmc$kw_ps_transfer_size	5599	5643							
mmc$kw_ring_numbers	5593	5608							
mmc$kw_segment_access_control	5597	5631							
mmc$kw_segment_number	5594	5612							
mmc$kw_shadow_segment	5599	5637							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmc$kw_software_attributes	5596	5629
mmc$kw_wired_segment	5599	5640
mmc$lp_aging_lock	4188	7889 11213 11399 11681 13076
mmc$lp_not_locked	4188	9852 9871 10109 10353 10474 10740 11913 12766 12881 12943
mmc$lp_page_in_lock	4189	7886 11399 11681 12551 13076
mmc$lp_server_allocate_lock	4190	7887 11399 11681 13076
mmc$lp_write_protected_lock	4189	7886 11399 11681 13076
mmc$max_rma_list_length	2208	2213 2214
mmc$mf_segment_mgr_flag	6103	9700/P 9704/P 9716/P 10878/P 10903/P 12147/P 12529/P
mmc$mf_shadow_file_reference	6104	11087/P
mmc$mf_volume_unavailable	6102	13403/P
mmc$mmu_age_interval_ceiling	6919	8506
mmc$mmu_age_interval_floor	6920	8507
mmc$mmu_aggressive_aging_one	6921	8508
mmc$mmu_aggressive_aging_two	6922	8509
mmc$mmu_min_avail_pages	6925	8532
mmc$mmu_ps_prestream	6926	8538
mmc$mmu_ps_random_limit	6930	8542
mmc$mmu_ps_reads	6929	8541
mmc$mmu_ps_threshold	6928	8539
mmc$mmu_ps_transfer_size	6927	8540
mmc$mmu_queue_age_device_file	6938	8465
mmc$mmu_queue_age_file_server	6939	8466
mmc$mmu_queue_age_other	6940	8467
mmc$mmu_queue_age_pf_execute	6936	8463
mmc$mmu_queue_age_pf_non_exec	6937	8464
mmc$mmu_queue_age_site_queues	6941	8468 8469 8470 8471 8472 8473 8474 8475 8476 8477 8478 8479 8480 8481 8482 8483 8484 8485 8486 8487 8488 8489 8490 8491 8492
mmc$mmu_queue_age_task_service	6935	8462
mmc$mmu_queue_maximum	6942	8462 8463 8464 8465 8466 8467 8468 8469 8470 8471 8472 8473 8474 8475 8476 8477 8478 8479 8480 8481 8482 8483 8484 8485 8486 8487 8488 8489 8490 8491 8492
mmc$mmu_queue_minimum	6943	8462 8463 8464 8465 8466 8467 8468 8469 8470 8471 8472 8473 8474 8475 8476 8477 8478 8479 8480 8481 8482 8483 8484 8485 8486 8487 8488 8489 8490 8491 8492
mmc$mmu_swapping_aic	6933	8576
mmc$mmu_tick_time	6934	8579
mmc$move_pages_max_req_length	1233	12448
mmc$mp_clear_modified	4351	12569
mmc$mp_set_modified	4351	12567
mmc$mpt_done	4154	10714 11218 12612
mmc$mpt_page_table_full	4154	10716 11221 12603
mmc$num_loader_predefined_segs	1228	1229
mmc$page_streaming_counters	4259	4282
mmc$pq_avail	1962	2008 7892 9097/S 9098/S 9103/S 9173 9276/S 9276/S 9277/S 9278/S 9300 9303 9304/S 9312 10050/P 10168/P 10223/P 10696/S 10696/S 10696/S 11399 11681 12172 12207/S 12757 12801 12934 13076

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmc$pq_avail_modified	1963	9142 9182 9439/P 9853 9872 10045/P 10055 10163/P 10173/P 10218/P 10392/P 10495 11923/P
mmc$pq_first_valid_in_pt	2007	7891 11399 11681 11912 13076
mmc$pq_free	1961	2020 9095/S 9098/S 9101/S 9164 9204 9276/S 9276/S 9276/S 9277/S 9278/S 9283 9416 9429 9437/P 9454 9470/S 9471/S 9473/S 9475/S 9478/S 9489/S 9489/S 10030/P 10048/P 10150/P 10166/P 10221/P 10696/S 10696/S 10696/S 10715/P 10979/P 11095/P 11182/P 12207/S 12316/P 12585/P 12758 12803/P 12820/P 12934 13306/P
mmc$pq_job_base	2009	9139 9170 12224 12627 12628 12641
mmc$pq_job_fixed	2002	2009 2021 8423 8823 8936 9271 9292 9395 9462 9479 10331/S 10876 12099 12484 12847 12849/S 13364
mmc$pq_job_io_error	2003	10215
mmc$pq_job_working_set	2004	2021 2022 8828 8936 8953 8954 9971 9996 10306/S 10320/S 10547/S 10548/S 10548/S 11953/S 11955/S 11981/S 11983/S 11985/S 11987/S 12190 12195/S 12197/S 12765 12880 12942 12961 13044 13156/S 13158/S 13174/S 13176/S 13178/S 13180/S 13364
mmc$pq_last_reassignable	2008	9140 9163 9171 9218
mmc$pq_shared_device_file	1969	8782 8826 8936 8962
mmc$pq_shared_first	2010	2056 10589 12763 12878 12940 13044
mmc$pq_shared_first_site	2012	2016
mmc$pq_shared_io_error	1998	10216/P
mmc$pq_shared_last	2017	8957 12764 12879 12941 13045
mmc$pq_shared_last_sys	2011	2056
mmc$pq_shared_num_sites	2013	2016
mmc$pq_shared_other	1971	2011 8784 8826 8936 8962
mmc$pq_shared_pf_execute	1967	8775 8826 8936 8962
mmc$pq_shared_pf_non_execute	1968	8773 8778 8826 8826 8936 8936 8962 8962
mmc$pq_shared_site_01	1973	2012
mmc$pq_shared_site_25	1997	2017
mmc$pq_shared_task_service	1966	2010 8780 8826 8936 8962
mmc$pq_swapped_io_error	2000	2020
mmc$pq_wired	1964	2007 8821 8936 10876 12099 12484
mmc$sa_fixed	5658	8822 8936
mmc$sa_free_behind	5659	11897
mmc$sa_read_transfer_unit	5659	11505
mmc$sa_stack	5659	13391
mmc$sa_wired	5658	8820 8936
mmc$sac_file_server_terminated	1256	11786
mmc$sac_no_append_permission	1254	11764
mmc$sac_read_beyond_eoi	1248	11762
mmc$sac_read_write_beyond_msl	1249	11772
mmc$sar_modify	4371	10836
mmc$sar_write_extend	4372	10835
mmc$sas_allow_access	5711	11375 12076 13041 13068
mmc$sas_inhibit_access	5712	11376 12077
mmc$sas_terminate_access	5716	11379 12080
mmc$sat_read	4376	9021
mmc$sat_read_or_write	4376	9027 9066/P 12424/P 13036/P 13063/P
mmc$sat_write	4376	9024 12070/P 12436/P 12708/P
mmc$scan_pft_for_free_or_avail	4312	12732 12733

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmc$scan_pft_free_avail_notmod	4313	12732 12744 12939
mmc$scan_pft_write_mod_pages	4313	12866
mmc$segment_fault_processor_id	6122	6148 11759 11770 11784
mmc$sequence_pointer	5671	5678
mmc$server_iocb_table_size	3029	3032
mmc$ssk_none	5761	5733 10994
mmc$ssk_segment_number	5762	5731 11068
mmc$uer_multiple_pages_assigned	4383	8668 9891 10934 10944 10967 11033 11037 11081 11114
mmc$uer_page_assigned	4382	10934/P 10944/P 10967/P 11035 11081/P 11114/P
mmc$uer_page_written	4383	8673 9891/P 9891 10934 10944 10967 11037 11081 11114
mme$assign_length_too_long	1521	12095/P 12198/P
mme$cannot_wait_for_memory	1530	12329/P
mme$dm_assign_active	1518	12148/P 12530/P
mme$invalid_length_requested	1565	12449/P
mme$invalid_pva	1292	9005/P 9019/P 9023/P 9026/P 9028/P 12087/P 12432/P 12444/P
mme$invalid_pva_formed	1488	12465/P
mme$invalid_request	1404	12893/P
mme$io_active_on_move_page	1568	12552/P
mme$length_not_page_size_mult	1550	12454/P
mme$memory_not_avail_for_assign	1515	12321/P
mme$modified_source_page_reject	1559	12557/P
mme$no_free_pages	1283	13256/P
mme$page_table_full	1280	12298/P 12616/P 13254/P
mme$pages_already_assigned	1586	12724/P
mme$pva_not_on_page_boundary	1553	12459/P
mme$read_beyond_eoi	1302	13094/P
mme$read_write_beyond_msl	1308	12480/P 13097/P
mme$ref_to_unrecovered_file	1491	9031/P
mme$segment_not_assigned_device	1385	13051/P
mme$segment_not_pageable	1410	13046/P
mme$source_page_not_in_memory	1562	12543/P
mme$temporary_reject	1452	12133/P 12515/P
mme$unable_to_assign_contig_mem	1583	12742/P 12795/P
mme$volume_unavailable	1540	12078/P 12138/P 12520/P
mme$wait_so_other_tasks_can_run	1527	12338/P
mme$wired_or_fixed_segs_illegal	1508	12100/P 12485/P
mme$write_beyond_eoi_no_append	1437	13100/P
mmk$monitor_base	1776	1616 1623 1630 1655 1676 1690 1693 1696 1699 1702 1705 1708 1711 1714 1717 1721 1724 1727 1734 1738 1741 1747
mmk$page_fault	1655	11351 12001
mmp$advise_request_processor	13001	13194
mmp$age_job_working_set	10271	10402 11951 13154
mmp$asid	8707	8722 8938
mmp$assign_asid	7652	8809 8936
mmp$assign_page_frame	10659	10760 10939 10952 11017 11069 11148 12265 12808 13245
mmp$assign_page_to_monitor	13222	13265
mmp$aste_pointer	8725	8598 8744 9422 13243
mmp$aste_pointer_from_pfti	8594	8606 9422
mmp$asti	8747	8759

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
mmp$check_queues	7659	9223 9343 9515
mmp$claim_pages_for_swapin	9251	9345
mmp$convert_pva	8872	8976 9069 11368 12075 12429 12441 12713 13040 13067
mmp$delete_page_from_monitor	13285	13252 13312
mmp$delete_pt_entry	7670	9105 9313 9403 9421 9436 10029 10047 10149 10165 10220 10696 10978 11093 11180 12315 12577 12584 12802 12818 13305
mmp$determine_shared_queue_id	8767	8787 8826 8936 8962
mmp$determine_shared_queue_id	8769	8773/M 8775/M 8778/M 8780/M 8782/M 8784/M 8826/M 8826/M 8826/M 8826/M 8826/M 8826/M 8936/M 8936/M 8936/M 8936/M 8936/M 8936/M 8962/M 8962/M 8962/M 8962/M 8962/M 8962/M
mmp$dump_shared_queue	10573	10622
mmp$fetch_pfti_array_size	7677	7683 10389 10489 11921
mmp$find_next_pfti	7686	7704 9976 10063 12317
mmp$free_asid	7707	9425
mmp$free_memory_in_job_queues	9357	9517
mmp$get_avail_page_frame	9092	9110 10696
mmp$get_inhibit_io_status	7714	7733 12769 12882 12945
mmp$get_sdt_entry_p	7803	8892 9014 9553 9694 13464
mmp$get_sdt_entry_p	7805	7807/M 8892/M 9014/M 9553/M 9694/M 13464/M
mmp$get_sdtx_entry_p	7815	8893 9015 9695 13386
mmp$get_sdtx_entry_p	7817	7819/M 8893/M 9015/M 9695/M 13386/M
mmp$get_verify_asti_in_fde	7831	7845 8933
mmp$include_p_reg_in_dump	13428	13474
mmp$initialize_find_next_pfti	7779	12311
mmp$link_page_frame_to_queue	9228	9243
mmp$make_pt_entry	7794	10713 11216 12602 12611
mmp$page_pull	10776	11123 11515 11595 11683 13078
mmp$page_pull_hash_sva	7857	7905 11399 11681 13076
mmp$preset_real_memory	7919	9653 10941 10954 11023 12237 12282 12835 13248
mmp$process_assign_contig_mem	12676	12896
mmp$process_assign_pages	12026	12013 12349
mmp$process_assign_pages_req	12007	12022
mmp$process_cancel_reserve	12353	12016 12368
mmp$process_move_pages_request	12387	12670
mmp$process_volume_unavailable	13375	11881 13410
mmp$purge_all_cache_map_proc	7940	7931 12664
mmp$purge_all_cache_proc	7942	8044 12666
mmp$purge_all_map_proc	7944	8615 8627 8639 9985 10122 10208 10395 10496 12319 12667
mmp$purge_all_page_map	8609	9985 10395 10496
mmp$relink_page_frame	9120	7898 9225 9437 9439 10030 10045 10048 10050 10053 10150 10163 10166 10168 10173 10216 10218 10221 10223 10715 10746 10979 11095 11182 11399 11681 12229 12316 12585 12634 12642 12803 12820 13076 13306
mmp$remove_page_from_job	10196	10228
mmp$remove_page_from_jws	10088	10184 10554 10603 12772
mmp$remove_pages_from_jws	9942	10073 10392 10491 11923
mmp$remove_pages_working_set	7946	13054
mmp$remove_stale_pages	10428	10501
mmp$reset_find_next_pfti	7954	7977 9972 9988

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmp$reset_store_pfti	7981	10317	10460					
mmp$reset_store_pfti_reverse	7990	11902						
mmp$send_escaped_alloc_flag	9665	9726	9784	9910				
mmp$set_include_pages_in_dump	7999	8018	8834	8936				
mmp$store_pfti	8024	10358	10361	10479				
mmp$store_pfti_reverse	8033	11914						
mmp$sva_purge_all_cache	8041	12666						
mmp$sva_purge_all_page_map	8636	12319						
mmp$sva_purge_one_page_map	8624	10122	10208					
mmp$trim_job_working_set	10509	10557	11957	13160				
mmp$update_eoi	8660	8683	9891	10934	10944	10967	11037	11081	11114
mmp$verify_pva	8985	9035	9066	12070	12424	12436	12708	13036	13063
mmp$volume_available	13413	13424							
mmp$write_page_to_disk	9743	9926	10057	10177	12885				
mmp$xcheck_queues	13318	7663	9223	9343	9515	13371			
mmp$xtask_pva_to_sva	9054	9072							
mmt$active_io_count	3324	3046	3313						
mmt$active_segment_table	1873	8517							
mmt$active_segment_table_entry	1858	1874	2071	2186	4176	5072	7654	7708	7783
		7796	7859	7948	8422	8422	8595	8727	8803
		8877	9064	9253	9365	9752	9953	10098	10661
		10780	11132	11192	11268	12035	12393	12411	12683
		13016	13230						
mmt$aging_statistics	2042	8510							
mmt$asid_list_ptf_index	2074	2069							
mmt$assign_contig_pass_ident	4312	4306	12900						
mmt$assign_contig_passes	2174	8513	8513						
mmt$assign_sub_reqcodes	4331	4320							
mmt$ast_index	5093	4606	5087	5466	7044	7653	7835	8708	8712
		8749	8806	8884					
mmt$async_work_list	2182	8518	8518						
mmt$attribute_keyword	5593	5608							
mmt$buffer_descriptor	2192	7222	7236	7490	9754	10794	11140		
mmt$buffer_descriptor_type	2202	2194							
mmt$eoi_state	7173	7046							
mmt$global_page_queue_index	2020	4235	8293						
mmt$global_page_queue_list	4235	8457	8457						
mmt$global_page_queue_list_ent	4225	4235							
mmt$hardware_attribute_set	5662	5628							
mmt$hardware_attributes	5650	5662							
mmt$image_file	8273	8267							
mmt$io_identifier	2331	2668	7221	7235	7493	9747	9955	10100	10782
		11134	11280	12691	13010				
mmt$io_request_status	3247	3240							
mmt$io_status	3239	3249	3310						
mmt$iocb_index	4136	2338	2344						
mmt$job_page_queue_index	2021	4236	5009	9269	9269	9286	9286	9377	9383
		9383							
mmt$job_page_queue_list	4236	4561	9255	9358					
mmt$keypoint_page_fault_status	4148	8094	11281						
mmt$link	1921	1859	4166	4167	4222	10284	10284	10284	
mmt$lock_segment_status	5741	5537							
mmt$locked_page	4188	4172							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmt$make_pt_entry_status	4154	7798	10673	11197	12403			
mmt$max_sdt	5476	5480						
mmt$max_sdtx	5561	5565						
mmt$memory_reserve_request	5096	4554						
mmt$modified_bit_option	4351	4343						
mmt$move_pages_page_count	4158	4345	4346	12405				
mmt$page_age	4195	4175	4199	4199				
mmt$page_frame_index	1927	1923	1923	5049	5051	5052	5053	5098	5099
		6452	6454	6456	7514	7671	7687	7691	7784
		7795	7860	7862	7955	7959	8024	8033	8316
		8594	9093	9121	9228	9258	9265	9371	9373
		9374	9376	9745	9775	9952	10089	10197	10281
		10443	10444	10445	10515	10516	10574	10578	10581
		10662	10663	10664	10665	10669	10671	10786	10788
		10792	10799	11131	11137	11141	11142	11193	11289
		11294	11295	11313	12034	12038	12044	12047	12052
		12053	12059	12060	12061	12062	12395	12408	12682
		12686	13014	13015	13234	13235	13294	13454	
mmt$page_frame_queue_id	2022	1867	4170	5050	8769	8769	8807	9122	9261
		9262	9943	10433	10583				
mmt$page_frame_table	4181	8553							
mmt$page_frame_table_entry	4165	4181	5070	7797	7869	9128	9229	9259	9375
		9667	9764	9951	10097	10286	10447	10522	10672
		11198	12046	12407					
mmt$page_pull_status	7912	7861	10666	10787	11136	11194	11296	11297	12048
		12696	13017	13231					
mmt$page_queue_list_entry	4221	4226	4236	9127	10278	10429	12697	13321	
mmt$page_selection_criteria	7790	7782							
mmt$page_streaming_statistics	4244	4270							
mmt$paging_statistics	4266	8544							
mmt$paging_statistics_source	4273	4267	4268	4269					
mmt$pf_statistics	4288	8545							
mmt$pfti_array	8312	8306							
mmt$rb_advise	4291	13002							
mmt$rb_assign_contig_memory	4302	12677							
mmt$rb_assign_pages	4317	12008	12027	12354					
mmt$rb_move_pages	4337	12388							
mmt$reassignable_page_frames	4356	8556							
mmt$rma_list_entry	2216	2211	2699	2714					
mmt$rma_list_index	2213	2211							
mmt$rma_list_length	2214	2193							
mmt$sdtx_stream_data	5544	5540							
mmt$segment_access_condition	1261	6148	11301						
mmt$segment_access_rights	4371	5536							
mmt$segment_access_state	5711	5531							
mmt$segment_access_type	4376	8987							
mmt$segment_descriptor	5463	5473	5477	7805	8002	8336	8769	8798	8878
		8994	9062	9537	9676	11303	12054	12397	12414
		12701	13008	13431	13453				
mmt$segment_descriptor_extended	5529	5558	5562	7817	7820	8574	8799	8851	8879
		8893	8995	9015	9063	9677	9695	9769	10781
		10884	11306	12055	12398	12415	12702	13009	13380
		13386							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$segment_inheritance	5579	5533	5636						
mmt$segment_pointer_kind	5671	5675							
mmt$segment_reservation_state	5751	5534							
mmt$server_iocb_entry	3036	2676	3033						
mmt$server_state	3458	3038							
mmt$shadow_info	5726	5538							
mmt$shadow_reference_info	5774	5353							
mmt$shadow_segment_kind	5761	5730							
mmt$software_attribute_set	5664	5535	5630						
mmt$software_attributes	5658	5664							
mmt$sub_reqcodes	3325	3043	3315						
mmt$update_eoi_reason	4382	8663	10806						
mmt$write_page_to_disk_status	9739	8428	9749	9954	10099	12705			
mmt$xcb_page_wait_info	5785	5339							
mmv$a_divisor	8703	8598	8719	8719	8736	8755	8938	8938	9422
		13243							
mmv$a_mult	8702	8598	8598	8719	8736	8736	8755	8755	8938
		9422	9422	13243	13243				
mmv$advise_in_aio_limit	8505	13133							
mmv$age_interval_ceiling	8506	10311							
mmv$age_interval_floor	8507	10312							
mmv$aggressive_aging_level	8508	10251	10252	11994	11994	13186	13186		
mmv$aggressive_aging_level_2	8509	10248	10609	10824	11994	12209	13075	13075	13186
mmv$aging_algorithm	8263	10293	10451	11944	13147				
mmv$aging_statistics	8510	9920/M	9921	9923/M	9923	10066/M	10067	10068/M	10068
		10170/M	10171	10179/M	10180	10257/M	10257	10301/M	10301
		10363/M	10363	10393/M	10393	11994/M	11994	13186/M	13186
mmv$aio_limit	8511	11971	13168						
mmv$aio_limit_count	8512	11975/M	11975	13170/M	13170				
mmv$assign_contig_reject	8514	12827/M	12827						
mmv$assign_contiguous_pass_cnt	8513	12734/M	12734	12736/M	12736	12891/M	12891		
mmv$assign_multiple_pages	8515	11007							
mmv$ast_p	8517	7839	7839	7840	8598	8739	8925	8933	8933
		8933	8951	9422	13243				
mmv$async_work	8518	10717/M	10718/M	10719/M	11223/M	11224/M	11225/M	12604/M	12605/M
		12606/M							
mmv$check_queues	13316	7662	9223	9343	9515	13355	13357		
mmv$gpql	8457	9095	9097	9098	9098	9101	9103	9145	9188
		9231	9232	9233/M	9235/S	9237/M	9238/M	9238	9276
		9276	9276	9276	9276	9276	9277	9277	9278
		9278	9298	9304	9323/M	9323	9331	9332/M	9333/M
		9337/M	9470	9471/M	9473/S	9475	9478/M	9489/M	9489
		10590	10592	10594	10594	10696	10696	10696	10696
		10696	10696	12207	12207	13359/P			
mmv$image_file	8267	10849	10849	11200	11207	12911			
mmv$jmtr_escaped_allocate	8522	9718/M	9718						
mmv$jws_queue_age_interval	8286	11965	13165						
mmv$last_active_shared_queue	8293	10589							
mmv$last_segment_accessed	8523	10883/M	12118/M	12498/M					
mmv$lost_escaped_allocate	8524	9722/M	9722						
mmv$max_pages_no_file	8525	10875	12111	12491					
mmv$max_working_set_size	8527	10529	10532	11956	12197	13159			
mmv$maxws_aio_count	8528	11962/M	11962	13162/M	13162				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmv$maxws_aio_slowdown	8529	11964	13164						
mmv$maxws_aio_threshold	8530	11961	13161						
mmv$memory_wait_queue	8531	9217	9219/P	9510	9510	9512/P	11827/P		
mmv$min_avail_pages	8532	9097	9276	9278	10696				
mmv$multi_page_write	8535	10057/P	10177/P	12885/P					
mmv$multiple_caches	8533	7930	8043	12663	12666				
mmv$multiple_page_maps	8534	7930	8614	8626	8638	9985	10122	10208	10395
		10496	12319	12663					
mmv$no_memory_buffering	8536	10046	10164	10219					
mmv$page_streaming_prestream	8538	11420	11507	11555					
mmv$page_streaming_random_limit	8542	11477							
mmv$page_streaming_reads	8541	11459	11532	11546	11547	11549	11557		
mmv$page_streaming_threshold	8539	11554							
mmv$pages_for_overallocation	8543	10956	10957						
mmv$pages_per_new_page_fault	8537	11008	11013	11014					
mmv$pages_to_dump_p	8300	13440/M	13458						
mmv$paging_statistics	8544	11451/M	11452	11471/M	11472	11482/M	11483	11528/M	11529
		11541/M	11542	11550/M	11551	11559/M	11560	11612/M	11613
		11615/M	11616	11629/M	11629	11635/M	11635	11652/M	11652
		11662/M	11662	11737/M	11737	11744/M	11744	11795/M	11795
		11836/M	11836	11855/M	11855	11924/M	11925	13109/M	13109
		13118/M	13118	13125/M	13125	13132/M	13132		
mmv$pf_statistics	8545	11934/M	11934						
mmv$pf_sva_array	8546	11935	11936/M	11937/M	11938/M				
mmv$pft_p	8553	7885	8598/P	8599	9130	9153/M	9158/M	9196/M	9235/M
		9311	9327/M	9328/M	9335	9336/S	9336/M	9338/M	9396/P
		9401	9405	9411	9422/P	9422	9465	9473/M	9475/M
		9782	9812	9817/M	9852	9853	9854	9871	9872
		9873	9974/S	9975/S	9991	10016	10108	10202/S	10208/P
		10209/M	10210/M	10215	10217	10325	10341	10371/M	10373/M
		10378/M	10466	10550	10602	10708	10942/S	11025/S	11026
		11076/S	11077/S	11078	11094	11111/S	11112	11181	11208
		11219/S	11399	11681	11801	11804/P	11813	11816/P	11840
		11846/P	11870	11876/P	11912	11913	12171	12223	12284/S
		12289/S	12290	12314/S	12549	12757	12758	12758	12763
		12764	12765	12766	12767/S	12767	12769/P	12801	12819
		12835/P	12836/S	12837	12847	12848/P	12850/M	12852	12853/M
		12854/M	12855/M	12877	12877	12878	12879	12880	12881
		12882/P	12884/P	12914	12929	12929	12934	12934	12935
		12940	12941	12942	12943	12944/S	12944	12945/P	12959/P
		12961	12965/P	13076	13250/S	13335	13338	13342	
mmv$pfti_array_p	8306	7681	7681	7694	7694	7695	7695	7696/M	7696
		7697	7697/S	7699	7699	7962/M	7962	7963	7963/S
		7965	7965	7966	7966	7967/M	7967	7968	7968/S
		7971	7971	7984/M	7985/M	7986/M	7993/M	7993	7994/M
		7994	7995/M	7995	8027/S	8027/M	8028/M	8028	8036/M
		8036	8037/M	8037	8038/S	8038/M	9972/M	9972	9972
		9972/S	9972	9972	9972/M	9972	9972	9972/S	9972
		9972	9972	9972	9976	9976	9976/M	9976	9976
		9976/S	9976	9976	9976	9976	9988/M	9988	9988
		9988/S	9988	9988	9988/M	9988	9988	9988/S	9988
		9988	9988	9988	10063	10063	10063/M	10063	10063
		10063/S	10063	10063	10063	10063	10317/M	10317/M	10317/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10358/S	10358/M	10358/M	10358	10361/S	10361/M	10361/M	10361
		10389	10389	10460/M	10460/M	10479/S	10479/M	10479/M	
		10479	10489	10489	11902/M	11902	11902/M	11902	11902/M
		11902	11914/M	11914	11914/M	11914	11914/S	11914/M	11921
		11921	12317	12317	12317/M	12317	12317	12317/S	12317
		12317	12317	12317					
mmv$pt_length	8554	9552	9629	9635					
mmv$pt_p	8555	7880	9142	9173	9174	9182	9434/M	9551	9632
		9632	9652/M	9654/M	9848	9851	9867	9870	9974/M
		9975/M	9995	10119	10202	10344	10345/M	10468	10469/M
		10551	10942/M	11025/M	11076/M	11077/M	11111/M	11219/M	11399
		11681	11911	12166	12220	12284/M	12289/M	12314/M	12547
		12583	12615/M	12651/M	12652/M	12767	12836/M	12944	13076
		13250/M	13439						
mmv$read_tu_execute	8557	11513							
mmv$read_tu_read_write	8558	11511							
mmv$reassignable_page_frames	8556	7524	9099	9141/M	9141	9143/M	9143	9167/M	9167
		9172/M	9172	9180	9186/M	9186	9239/M	9239	9240/M
		9240	9241	9276	9279	9295/M	9295	9442/M	9442
		9450/M	9450	9491/M	9491	9494/M	9494	9506	10054
		10054	10174	10174	10247	10247	10609	10682	10696
		10824	11007	11994	11994	12259	12260	13075	13075
		13186	13186						
mmv$refs_to_unrec_df_file_inhib	8559	11377/M	11377						
mmv$refs_to_unrec_df_file_term	8560	11380/M	11380						
mmv$reserved_page_count	8561	12208	12345/M	12345	12365/M	12365			
mmv$tables_initialized	8577	11320	12066	13023	13355				
mmv$time_to_call_mem_mgr	8330	10255/M	10720/M	11226/M	11994/M	12607/M	13186/M		
mmv$volume_unavail_dequeued	13416	13421/M	13421						
mmv$volume_unavail_queued	13382	13394/M	13394						
mmv$volume_wait_queue	8580	13393/P	13419	13419	13420/P				
mmv$write_aged_out_pages	8581	10055	10175						
mmv$write_page_statistics	8428	9786/M	9786	9797/M	9797	9814/M	9814	9918/M	9918
modified	12404	12568/M	12570/M	12572/M	12652				
modified_bit_option	4343	12567	12569						
modified_pages_removed	10435	10491/P	10494/M						
modified_pages_removed	10577	10603/P							
modified_queue_id	9943	9971	9996	10053/P	10055				
monitor_condition_register	5901	10906							
monitor_fault	11285	11759/M	11760	11767/P	11770/M	11771	11781/P	11784/M	11785
		11788/P							
monitor_lock	7034	7387/P	7449/P	8922/P	10005/P	10128/P	11053/P	12884/P	
move_page	12538	12538	12661						
moved_modified_page_count	4345	12472/M	12560/M	12560					
mpt_status	10673	10713/P	10714	10716					
mpt_status	11197	11216/P	11218	11221					
mpt_status	12403	12602/P	12603	12611/P	12612				
mtc$job_fixed_segment	4388	7282/S	7344	7385	8922	9686/S	10005	10128	10327
		11053	12884						
mtp$clear_interlock	7459	7449	7473						
mtp$cst_p	8052	8999	9068	13362					
mtp$error_stop	7415	7890	7901	8600	8896	8927	8971	9165	9175
		9184	9301	9422	9592	9625	9641	9721	9913

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10226	10698	10733	10890	10976	11120	11173	11229
		11399	11399	11519	11664	11681	11681	11802	11814
		11841	11871	11883	12019	12152	12296	12534	12613
		12630	13076	13076	13258	13302	13333	13336	13339
		13345							
mtp$set_interlock	7419	7387	7440	8922	10005	10128	11053	12884	
mtp$set_status_abnormal	8064	8071	9005	9019	9023	9026	9028	9031	12078
		12081	12087	12095	12100	12133	12138	12142	12148
		12198	12298	12321	12329	12338	12432	12444	12449
		12454	12459	12465	12480	12485	12515	12520	12524
		12530	12543	12552	12557	12616	12724	12742	12795
		12893	13046	13051	13094	13097	13100	13254	13256
mtp$step_unstep_system	8086	13397							
mtt$monitor_interlock	7180	7034	7419	7459					
mtv$cst0	8058	8054	8999	9068	13362	13459	13459	13460	13461
mtv$monitor_segment_table	8335	13239	13241						
mtv$sys_core_init_complete	8343	9020							
multiple_page_req	9748	9838							
multiple_pages_written_to_disk	2051	9920/M	9921						
nat$received_message_descriptor	5801	5794	5803						
nat$received_message_list	5793	5321							
new	4276	11652/M	11652	11744/M	11744	11855/M	11855	13109/M	13109
new_pages_assigned	4862	11648/M	11649	11650/M	11651	11740/M	11741	11742/M	11743
		11851/M	11852	11853/M	11854	13105/M	13106	13107/M	13108
next_cyclic_aging_time	5146	10450/M							
next_i	8547	11935	11936/M						
next_pfti	9373	9413/M	9456						
next_pfti	10445	10467/M	10483						
next_pfti	10578	10602/M	10604						
next_pfti	10799	11020/M	11021	11021	11025/S	11026/M	11026/S	11027	11074/M
		11075	11075	11076/S	11077/S	11078/M	11078/S	11094/M	11096
		11109/M	11110	11110	11111/S	11112/M	11112/S		
next_pfti	11142	11181/M	11183						
next_pfti	12044	12269/M	12271	12271	12284/S	12289/S	12290/M	12290/S	
next_rma	9540	9572/M	9574	9575/M	9581	9582/M	9585/M	9585	9589/M
		9589	9590	9594/M	9599	9599	9603	9613	9617
nlc$cc_connect_confirm	5833	5824							
nlc$cc_connect_request	5832	5822							
nlc$cc_expedited_data	5838	5824							
nlc$cc_max_pdu_kind	5840	5843							
nlc$channel_connection_pdu	5856	5808							
nlc$channelnet_pdu	5856	5810							
nlt$cc_pdu_kind	5843	5821							
nlt$cc_seq#_or_connect_time	5820	5809							
nlt$cc_sequence_number	5846	5825							
nlt$device_identifier	5853	5804							
nlt$pdu_type	5856	5807							
nominal_page_fault	11286	11408/M	11448/M	11461/M	11503				
normal	3332	8069/M	8998/M	9005/M	9019/M	9023/M	9026/M	9028/M	9031/M
		9067	9898	10966	11164	12064/M	12071	12078/M	12081/M
		12087/M	12095/M	12100/M	12133/M	12138/M	12142/M	12148/M	12198/M
		12298/M	12321/M	12329/M	12338/M	12422/M	12425	12432/M	12437

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		12444/M	12449/M	12454/M	12459/M	12465/M	12480/M	12485/M	12515/M
		12520/M	12524/M	12530/M	12543/M	12552/M	12557/M	12616/M	12707/M
		12709	12724/M	12742/M	12795/M	12893/M	13021/M	13037	13046/M
		13051/M	13064	13094/M	13097/M	13100/M	13237/M	13254/M	13256/M
		13296/M							
now	4357	7524	9099	9141/M	9141	9172/M	9172	9180	9240/M
		9240	9241	9276	9279	9295/M	9295	9450/M	9450
		9491/M	9491	9506	10054	10174	10247	10609	10682
		10696	10824	11007	11994	12259	12260	13075	13075
		13186							
null_pva	8500	8930							
null_sva	7928	7933	7934						
null_sva	8501	9314	9428						
null_sva	8612	8617							
null_sva	9942	9985							
null_sva	10271	10395							
null_sva	10428	10496							
null_utp	11287	11999							
num_pf_recs	8590	8548	11936						
number_of_pages_moved	4346	12473/M	12654/M	12654	12656				
number_of_pages_to_assign	10662	10682	10689						
number_of_pages_to_move	12405	12475/M	12479	12491	12500/P	12504/P	12505	12540	12656
off	11288	11356/M	11357	11358/M	11359	11362/M	11363	11364/M	11365
offset	2090	8920/M	9555/M	9556/M	9556	9649	9698/P	9835	9840
		9846/M	9860	9861/M	9865/M	9878	9893/P	9895/P	10016
		10028	10148	10329	10755/M	10755	10811/M	10811	10834
		10848	10877/P	10886/P	10888/P	10899/P	10918	10919	10920
		10930/P	10934/P	10938	10944/P	10964/P	10965/P	10967/P	10995
		11004	11004	11012	11028/M	11028	11037/P	11050	11054/M
		11081/P	11085	11114/P	11200	11207	11390	11392	11540
		11574	11680/M	11754	11865	11901/M	11905	11905	11906/M
		11906	12094	12106/M	12106	12113	12119/P	12122/P	12123
		12181/M	12181	12243/M	12243	12286/M	12286	12302/M	12302
		12458	12458	12463	12464	12477/M	12477	12479	12493
		12499/P	12503/P	12505	12657/M	12657	12658/M	12658	12717/M
		12717	12727/M	12727	13054/P	13141/M	13141	13238/M	13262/M
		13262	13436/M						
offset	2161	9004	9555	11328/P	11336/M	11336	11356	11358	11362
		11364	13465/P	13466/P	13467/P				
offset	7335	7340/M	7347/M	7350					
offset	7378	7385/M	7385/M	7385					
offset	8662	8665	8666	8674	8677				
offset	8852	8855							
offset	8872	8922/M	8922/M	8922					
offset	9665	9698							
offset	9743	9891	9891	9891	9891				
offset	9763	9840/M	9841	9857/M	9861				
offset	9942	10005/M	10005/M	10005					
offset	10088	10128/M	10128/M	10128					
offset	10271	10327/M	10327/M	10327					
offset	10776	10877	10899						
offset	10776	10934	10934	10934	10934	10944	10944	10944	10944

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10967	10967	10967	10967	11037	11037	11037	11037
		11081	11081	11081	11081	11114	11114	11114	11114
offset	10776	11053/M	11053/M	11053					
offset	12026	12146							
offset	12387	12528							
offset	12676	12884/M	12884/M	12884					
offset	13432	13434	13434	13436					
old_assign_active	9672	9697/M	9699						
original_bkw_link	9374	9385/M	9482						
osc$aging_interval_maximum	5211	5214							
osc$base_exception	384	390	1044	1046	6566				
osc$call_instruction	6036	6044							
osc$data_read	6035	6044							
osc$free_running_clock_maximum	2031	2028							
osc$invalid_ring	2107	2147							
osc$keypoint_base	6566	6568	6572	6576	6579	6583	6587	6591	6594
		6598	6602	6606	6610	6613	6616	6620	6623
		6627	6630	6635					
osc$kpt_pva_segment	6447	11327							
osc$max_channel_number	3882	3885							
osc$max_fault_contents	6161	6155							
osc$max_idle_count	5268	5276							
osc$max_integer	3471	3476	3477						
osc$max_kpt_pages	6446	6452	6454	6456	6492				
osc$max_name_size	2402	2406	2409						
osc$max_number_of_processors	5253	4469	6481	6482	7599				
osc$max_page_frames	1931	1860	1927	4223	4229	4357	4358	4359	4360
		4601	4602	5008	5010	6942	8313	8314	8315
osc$max_page_size	3232	3228							
osc$max_page_table_entries	1932	1935							
osc$max_ring	2106	2147	2148						
osc$max_segment_length	2130	2153	5541	5572	12463	12464			
osc$max_status_condition_code	3269	3265	3281						
osc$max_status_condition_number	8074	8065							
osc$max_string_size	3285	3288	3291	3296					
osc$max_tasks	2260	2257							
osc$maximum_offset	2129	2130	2150	2150	2151				
osc$maximum_processor_id	6061	6057							
osc$maximum_processor_number	5245	5240							
osc$maximum_processors	5249	5245	5253						
osc$maximum_segment	2128	2149							
osc$min_ecc	383	384							
osc$min_integer	3470	3474	3475						
osc$min_page_size	3231	3228							
osc$min_ring	2105	2148							
osc$non_executable	5510	8772	8826	8936	8962	11510			
osc$non_readable	5513	9021							
osc$non_writable	5516	8005	8772	8826	8834	8936	8936	8962	9025
		11510							
osc$pr_base_constant	4405	7556	7563	7584	7624	7727	7729	9706	9706
		9719	9719	12770	12770	12774	12774	12882	12882
		12886	12886	12946	12946				
osc$pr_process_interval_timer	4431	13356	13368						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$purge_all_cache	4446	7933							
osc$purge_all_page_seg_map	4455	7934	8617	8930	9985	10395	10496		
osc$pva_purge_all_page_seg_map	4453	13310							
osc$pva_purge_segment_cache	4448	13240							
osc$sva_purge_all_cache	4445	8046	12666						
osc$sva_purge_all_page_map	4451	8641	12319						
osc$sva_purge_one_page_map	4450	8629	10122	10208					
osc$task_time_slice_maximum	4761	4764							
osc$trap_exception	5865	10906							
osc$traps_enabled	5979	10905	11066	13387					
osc$vl_invalid_entry	5491	8895	9018	9696					
osc$wait	3453	12322							
osk$m	1853	9133	9134	9135	9309	9415	9416	10328	10329
		10331	10333	10398	10399	11353	11355	11357	11359
		11361	11363	11365	11706	11707	11708	12001	
osk$monitor_multiplier	1852	11351	12001						
osk$mtr	1817	11351	12001						
osk$performance	1818	9133	9134	9135	9309	9415	9416	10328	10329
		10331	10333	10398	10399	11353	11355	11357	11359
		11361	11363	11365	11706	11707	11708		
osk$system_class	1828	1812	1813	1814	1815	1816	1817	1818	
osp$process_keypoint_page_fault	8092	11328							
ost$aging_interval	5214	5143	5144						
ost$asid	2093	1865	2070	2089	4545	5074	5085	5086	5497
		5634	7652	7707	7768	8709	8726	8731	8748
		8752	8802	8883	9364				
ost$binary_unique_name	3140	3132	7036						
ost$byte_count	2083	7491							
ost$class_15_keypoint	6542	6499							
ost$cp_time	4823	4791	4836	5338					
ost$cp_time_value	4821	4587	4824	4825	5138	5139	5351	11271	13012
ost$cpu_element_id	5237	4500							
ost$cpu_idle_statistics	5271	4503							
ost$cpu_memory_port_mask	5239	4476							
ost$cpu_running_or_stepped	5289	5286	5286						
ost$cpu_state	5284	4485							
ost$cpu_state_reason	5295	4506							
ost$cpu_state_table	4472	4469	8052	8800	8874	8991	9060	10242	10779
		11254	12009	12028	12355	12389	12678	13003	13350
ost$cs_lock	4086	5319							
ost$cst_trace_control	6344	4504							
ost$date_time	3481	2632	6544						
ost$debug_code	6035	6023							
ost$debug_list	6031	5935							
ost$debug_list_entry	6022	6031							
ost$debug_mask	6041	5934							
ost$exchange_package	5884	5306							
ost$execute_privilege	5510	5492	5505						
ost$execution_control_block	5305	4486	5331	7804	7816	8111	8123	8161	9533
		9678	9774	13376	13455				
ost$external_interrupt_request	6332	4492							
ost$family_name	5224	5219							
ost$flags	5941	5891							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$frame_descriptor	5999	6014							
ost$free_running_clock	2028	1864	4556	4557	4558	4559	4593	4603	4604
		4605	4701	4713	5137	5146	5337	7052	7317
ost$global_task_id	2251	2244	2337	2660	4481	4550	4579	5133	5316
		5317	6073	6248	7054	7135	8102	8116	8122
		8137	8147	8154	8160	8361	8361	9125	9378
		13417							
ost$halfword	2166	5257							
ost$idle_type	5280	5275							
ost$key_lock	2136	5498	5619						
ost$key_lock_value	2142	2139	5958	5960					
ost$keypoint	6533	6543							
ost$keypoint_class	5973	5904	5975						
ost$keypoint_environment	6524	6470							
ost$keypoint_mask	5975	5907	6469	6469	6469	6469			
ost$keypoint_multipro_option	6528	6471							
ost$logical_processor_id	5240	4477							
ost$minimum_save_area	6009	5896	5984	6142					
ost$monitor_condition	5860	5867							
ost$monitor_conditions	5867	5897	5901	5989	6217	6231			
ost$monitor_fault	6138	6115	8148	11285					
ost$monitor_fault_contents	6155	6151							
ost$name	2409	2368	2398	4065	5183	5222	5224	5365	5429
		6180							
ost$non_negative_integers	3476	2615							
ost$p_register	5956	5885	6010	6209	6215				
ost$page_id	1937	1947							
ost$page_size	3228	3209	8582						
ost$page_table	1951	8555	9547						
ost$page_table_entry	1942	1951	5071	7870	9542	9765	9958	10101	10200
		10517	12050	12409					
ost$page_table_index	1935	1951	4173						
ost$paging_statistics	4859	4837	5346						
ost$parcel	2168	4498	4499						
ost$physical_channel_number	3885	3827							
ost$pre_processed_for_reconfig	6340	4507							
ost$processor_element_id	5256	5237							
ost$processor_element_number	5265	5258							
ost$processor_id	6057	5309	6051						
ost$processor_id_set	6051	5308	6483						
ost$processor_keypoint_control	6450	6485							
ost$processor_model_number	3158	3142	3494	5259					
ost$processor_serial_number	3236	3141	3495	5260					
ost$pva	2158	5929	5947	5961	6139	6232	8993	11261	11287
		11287							
ost$read_privilege	5513	5493	5506						
ost$real_memory_address	2081	2478	2579	3694	4497				
ost$register_number	5952	5926	5995	6003	6004	6005			
ost$ring	2147	2159	5495	5496	5530	5610	5611	5946	
ost$ring1_termination_reason	6069	5342							
ost$segment	2149	2160	2242	5613	5732	5924	6025	7053	7270
		7805	7817	8000	8523	8797	8885	8996	13381
ost$segment_access_control	5503	5632							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
ost$segment_descriptor	5490	5464
ost$segment_length	2153	3041 3309 4295 4297 4307 4323 4342 5615
		5617 5639 5641 5644 7018 7220 7234 7246
		7780 7947 8519 8519 10440
ost$segment_offset	2150	2090 2161 2243 3040 5545 6026 6028 7017
		7211 7219 7233 7245 7248 7256 7489 8662
		8852 11283 11309 12051 12410
ost$signature_lock	4087	2374
ost$stack_frame_save_area	5983	6017 6175
ost$state_tables	4469	8058
ost$status	3253	5626 6210
ost$status_condition	3277	3243 3312 3333
ost$status_condition_code	3281	3256 3277
ost$string	3294	3257
ost$string_size	3288	3295
ost$system_flag	6264	6260
ost$system_virtual_address	2088	2185 2196 4178 7779 7794 7858 7919 7946
		8041 8501 8501 8550 8624 8636 8875 8886
		9056 9536 9761 9770 10660 10670 10777 10795
		11130 11191 11307 11308 11312 12049 12056 12058
		12399 12416 12418 12703 12704 13018 13232 13448
ost$task_index	2257	2252 4206 4207 4511 6888 8383 8396
ost$task_time_slice	4764	4750
ost$top_of_stack_pointer	5944	5936
ost$trap_enable	5978	5893 6206
ost$user_condition	5870	5877
ost$user_conditions	5877	5895 5899 5987 6016 6178 6218
ost$user_identification	5217	5132
ost$user_name	5222	5218
ost$valid_relative_pointer	2156	5335 5336 7059 7062
ost$valid_ring	2148	5936 8117
ost$virtual_machine_identifier	5966	5887 5889 6011
ost$wait	3453	3311 4325
ost$word	2170	2515
ost$write_privilege	5516	5494 5507
ost$x_register	5953	5926 5995
osv$180_memory_limits	8348	9575 9576 9590 9594 9608 11200 11207 12912
osv$cpus_logically_on	7599	7556 7563 7583 7623 7727 7729 9706 9706
		9719 9719 12770 12770 12774 12774 12882 12882
		12886 12886 12946 12946
osv$page_size	8582	7884 8666 8674 8677 9556 9556 9585 9589
		9651 9845 9851 9860 9870 9876 9886 9891/P
		9891 9891 9891 9919 10016 10329 10755 10811
		10811 10920 10922 10934 10934 10934 10944 10944
		10944 10964/P 10967 10967 10967 10995 11013 11028
		11037 11037 11037 11051 11056 11061 11081/P 11081
		11081 11081 11114/P 11114 11114 11114 11155/P 11160/P
		11200 11207 11389 11399 11574 11680 11681 11906
		11911 12091 12092 12094 12106 12106 12119/P 12122/P
		12123 12170 12181 12222 12243 12286 12302 12453
		12458 12458 12475 12479 12500/P 12504/P 12505 12548
		12583 12657 12658 12715 12715 12717 12717 12727
		12912 13071 13072 13076 13141 13238 13238 13262

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

IDENTIFIER	DEFINED ON LINE	REFERENCES
		13304 13307 13439 13466/P 13467/P
osv$time_to_check_asyn	8356	10256/M 10721/M 11227/M 11994/M 12608/M 13186/M
out_length	4295	13035 13054/P
out_pva	4294	13036/P 13040/P 13054/P
p	8873	8890 8920
p	8986	9001
p	9055	9066/P 9069/P
p	11263	11368/P 11766 11773 11787 11805 11817 11847 11877
		11880
p	13223	13238 13241/S 13252/P
p	13286	13297 13307
p_register	5885	9009 9010 10900 10905 11067 11336/M 11336 11354
		11356 11358 13464/P 13465/P 13466/P 13467/P
page_aging_interval	5143	10299 10300 10305 10457
page_count	2193	9886/M 10963/M 11153/M
page_count	5777	11086/M
page_count	7860	7876/M 7879/M
page_count	10786	10808/M 10851/M 10922/M 10923 . 10924/M 10927/M 10930/P 10939/P
		10952/P 10963 11017/P 11019 11061/M 11069/P 11071 11086
		11105/P
page_count	11131	11148/P 11150
page_count	11252	11399/M 11399/M 11681/M 11681/M
page_count	11289	11373/M 11384/M 11399/P 11426 11504 11516/P 11594 11596/P
		11608 11609 11613 11616 11626 11628 11629 11632
		11634 11635 11649 11651 11652 11659 11661 11662
		11681/P 11682 11684/P 11696/M 11734 11736 11737 11741
		11743 11744 11792 11794 11795 11833 11835 11836
		11852 11854 11855
page_count	12907	12919/M 12929 12929 12936/M 12936 12948/M 12948 12966/M
		12966 12971/M 12974
page_count	13001	13076/M 13076/M
page_count	13013	13055/P 13071/M 13078/P 13081/M 13081 13138
page_count	13224	13244
page_count	13287	13299
page_fault_count	4864	11725/M 11725 11726/M 11727
page_faults_tu	4251	11541/M 11542
page_in_count	4860	11625/M 11626 11627/M 11628 11791/M 11792 11793/M 11793
		13128/M 13129 13130/M 13130
page_in_count	13014	13076/P 13077 13079/P 13081 13106 13108 13109 13115
		13117 13118 13122 13124 13125 13129 13131 13132
		13141
page_status	10801	10880/M 10886/P 10888/P 10898 10908/M 10915
page_status	12045	12112/M 12115/M 12120/P 12122/P 12127
page_status	12406	12492/M 12495/M 12501/P 12504/P 12509
page_streaming	4270	11451/M 11452 11471/M 11472 11482/M 11483 11528/M 11529
		11541/M 11542 11550/M 11551 11559/M 11560 11612/M 11613
		11615/M 11616 11924/M 11925
page_streaming_available_page	11290	11601/M 11643 11646/M
page_sva	10670	10688/M 10713/P 10718 10738 10755/M 10755
page_wait_info	5339	11805/M 11817/M 11847/M 11877/M 11880/M 13385
page_written_to_disk	2050	9923/M 9923
pageid	1947	9632 9632 9648/M 9649/M

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pagenum	1939	9649/M							
pages_freed_behind	4252	11924/M	11925						
pages_from_queue	10579	10592/M	10594/M	10605/M	10605	10606			
pages_from_server	4863	11658/M	11659	11660/M	11661	11832/M	11833	11834/M	11835
		13114/M	13115	13116/M	13117				
pages_in_memory	1860	8953	9423	10875	12111	12491			
pages_not_in_memory	12060	12159/M	12179/M	12179	12194	12196	12208		
pages_prestream	4248	11615/M	11616						
pages_pulled	11291	11575/M	11583	11584	11608/M	11608	11622/M	11622	11674
		11680							
pages_reclaimed_from_queue	4861	11631/M	11632	11633/M	11634	11733/M	11734	11735/M	11736
		13121/M	13122	13123/M	13124				
pages_removed	10580	10603/P							
pages_requested	12694	12714/M	12721	12739/P	12756	12789/P	12800	12808/P	12814
		12857	12864						
pages_requested	12901	12929	12929	12974					
pages_streaming	4249	11612/M	11613						
pages_to_allocate	10802	10957/M	10959/M	10964/P	11002/M	11014/M	11017/P	11032	
pages_to_be_pulled	11292	11576/M	11583	11584/M	11674				
pages_to_read	10783	10923	10924	10956	10959				
pages_to_read	11293	11506/M	11511/M	11513/M	11515/P	11574/M	11575	11585	11586/M
		11595/P	11609/M	11609	11622	11623/M	11677	11678/M	11683/P
paging_statistics	4837	11625/M	11626	11631/M	11632	11648/M	11649	11658/M	11659
		11726/M	11727	11733/M	11734	11740/M	11741	11791/M	11792
		11832/M	11833	11851/M	11852	11984	11986/M	13105/M	13106
		13114/M	13115	13121/M	13122	13128/M	13129	13177	13179/M
paging_statistics	5346	11627/M	11628	11633/M	11634	11650/M	11651	11660/M	11661
		11725/M	11725	11735/M	11736	11742/M	11743	11793/M	11793
		11834/M	11835	11853/M	11854	11980	11982/M	13107/M	13108
		13116/M	13117	13123/M	13124	13130/M	13130	13173	13175/M
pass_count	4306	12731	12733	12739/P	12744	12789/P			
pass_count	12900	12939							
pass_one_count	2175	12734/M	12734						
pass_three_count	2177	12891/M	12891						
pass_two_count	2176	12736/M	12736						
passive_fde_p	10803	11053/P	11105/P						
perf	10282	10323/M	10324	10324	10325/S	10326/M			
pf_pages	4267	11737/M	11737	11744/M	11744	11795/M	11795	11836/M	11836
		11855/M	11855						
pf_proc_tables_not_initialized	9532	9656	11321						
pf_recs	8548	11937/M	11938/M						
pfte_ijle_p	9673	9685/P	9693	9712/P					
pfte_p	7869	7885/M	7886	7886	7887	7889	7891	7892	7897/M
pfte_p	9128	9130/M	9135	9136	9139	9140	9142	9142/S	9145/S
		9147/P	9148/S	9150	9151	9153/S	9153	9155	9156
		9158/S	9158	9162/M	9163	9168/M	9173/S	9174/S	9182/S
		9183	9190/P	9198/M	9203/M	9205/M	9218		
pfte_p	9229	9231/M	9231/S	9232/S	9233/S	9235/S	9237/S	9238/S	9238/S
pfte_p	9259	9311/M	9314/M	9316/M	9317/M	9318/M	9320		
pfte_p	9375	9411/M	9412	9413	9420	9428/M	9429/M	9434/S	9435
		9449	9454/M						
pfte_p	9667	9685/P	9698/P						
pfte_p	9764	9782/M	9784/P	9794	9834	9910/P			

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pfte_p	9951	9991/M	9992	9993/M	9994/M	9995/S	10006	10028	10044
pfte_p	10097	10108/M	10109	10114	10119/S	10122/P	10130	10138/M	10139/M
		10148	10162						
pfte_p	10285	10325/M	10326	10327/P	10329	10333	10334	10341/M	10343
		10344/S	10345/S	10346/M	10347/M	10349	10353	10357	10360
		10366/M	10366	10374/M					
pfte_p	10447	10466/M	10467	10468/S	10469/S	10470/M	10471/M	10472	10473/M
		10473	10474						
pfte_p	10522	10550/M	10551/S	10553					
pfte_p	10672	10708/M	10713/P	10732	10735/M	10736/M	10737/M	10738/M	10739/M
		10740/M	10741/M						
pfte_p	11198	11208/M	11209/M	11210/M	11211/M	11212/M	11213/M	11214/M	11216/P
pfte_p	11252	11399/M	11399	11399	11399	11399	11399	11399	11399/M
		11681/M	11681	11681	11681	11681	11681	11681	11681/M
pfte_p	12046	12171/M	12172	12223/M	12225/M	12227/M			
pfte_p	12407	12549/M	12551	12551	12590/M	12591/M	12602/P	12609/M	12610/M
		12611/P	12615/S	12628	12629	12633/M	12635	12636	12638
		12641	12643/M	12644	12645	12647	12651/S	12652/S	
pfte_p	13001	13076/M	13076	13076	13076	13076	13076	13076	13076/M
pfti	7691	7693/M	7694	7694	7697/M	7700/M	7702		
pfti	7862	7874/M	7884/M	7885/S	7898/P				
pfti	7959	7963/M	7965	7965	7968/M	7972/M	7974		
pfti	8024	8027							
pfti	8033	8038							
pfti	8594	8598/S	8599/S						
pfti	9093	9095/M	9096	9101/M	9103/M	9104	9105/P		
pfti	9121	9130/S	9133	9194	9196	9199			
pfti	9228	9233	9235	9237					
pfti	9258	9298/M	9299	9304/M	9306	9307	9307	9309	9311/S
		9313/P	9320/M	9330	9335/S	9336/S	9337	9338/S	
pfti	9357	9422/S	9422/S						
pfti	9376	9384/M	9394	9396/S	9400	9401/S	9403/P	9405/M	9405/S
		9410	9410	9411/S	9415	9421/P	9422/P	9436/P	9437/P
		9439/P	9456/M						
pfti	9745	9782/S	9812/S	9817/S					
pfti	9942	9972/M	9972	9972/M	9972	9972/M	9972	9988/M	9988
		9988/M	9988	9988/M	9988				
pfti	9942	9976/M	9976	9976/M	9976	9976/M	9976	10063/M	10063
		10063/M	10063	10063/M	10063				
pfti	9952	9972/P	9973	9973	9974/S	9975/S	9976/P	9988/P	9990
		9990	9991/S	10007/M	10016/S	10018/M	10029/P	10030/P	10032/M
		10041	10045/P	10047/P	10048/P	10050/P	10053/P	10057/P	10063/P
pfti	10089	10108/S	10149/P	10150/P	10163/P	10165/P	10166/P	10168/P	10173/P
		10177/P							
pfti	10197	10202/S	10208/S	10209/S	10210/S	10215/S	10216/P	10217/S	10218/P
		10220/P	10221/P	10223/P					
pfti	10271	10358	10361						
pfti	10281	10342/M	10351	10358/P	10361/P	10376	10378	10380	10381
pfti	10428	10479							
pfti	10443	10461/M	10465	10465	10466/S	10479/P	10483/M		
pfti	10515	10547/M	10548	10549	10550/S	10552	10553/M		
pfti	10581	10590/M	10601	10601	10602/S	10603/P	10604/M		
pfti	10659	10696/M	10696	10696/M	10696/M	10696	10696/P		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pfti	10671	10680/M	10696/P	10697	10706/M	10706	10708/S	10713/P	10715/P
		10746/P	10749						
pfti	10788	10809/M	10850/P	10931/P	10939/P	10942/S	10952/P	10978/P	10979/P
		11017/P	11020	11069/P	11074	11092	11092	11093/P	11094/S
		11095/P	11096/M	11106/P	11109				
pfti	11137	11148/P	11179	11179	11180/P	11181/S	11182/P	11183/M	
pfti	11193	11201	11203/M	11207/M	11208/S	11216/P	11219/S		
pfti	11252	11399/M	11399/M	11399/S	11399/P	11681/M	11681/M	11681/S	11681/P
pfti	11252	11914							
pfti	11294	11399/P	11516/P	11596/P	11599	11681/P	11684/P	11698/M	11706
		11801/S	11804/S	11813/S	11816/S	11840/S	11846/S	11870/S	11876/S
pfti	12026	12317/M	12317	12317/M	12317	12317/M	12317		
pfti	12047	12170/M	12171/S	12222/M	12223/S	12229/P	12311/P	12313	12313
		12314/S	12315/P	12316/P	12317/P				
pfti	12408	12548/M	12549/S	12577/P	12602/P	12611/P	12634/P	12642/P	
pfti	12695	12752/M	12757/S	12758/S	12758/S	12760/M	12760	12763/S	12764/S
		12765/S	12766/S	12767/S	12767/S	12769/S	12772/P	12773/M	12773
		12788	12799/M	12801/S	12802/P	12803/P	12805/M	12805	12815/M
		12817	12817	12818/P	12819/S	12820/P	12821/M	12877	12878/S
		12879/S	12880/S	12881/S	12882/S	12884/S	12885/P		
pfti	12908	12912/M	12914/M	12917/M	12929	12929	12930/M	12930	12934/S
		12934/S	12935/S	12940/S	12941/S	12942/S	12943/S	12944/S	12944/S
		12945/S	12959/S	12961/S	12965/S	12977			
pfti	13001	13076/M	13076/M	13076/S	13076/P				
pfti	13015	13076/P	13079/P						
pfti	13235	13245/P	13250/S						
pfti	13294	13304/M	13305/P	13306/P					
pfti	13326	13329/M	13332	13335/S	13338/S	13341	13342/M	13342/S	13344
pfti	13454	13439/M	13440/S						
pfti_first	8313	7681	7962	7963/S	7986/M	7995/M	8036	8037/M	8037
		8038/S	9972	9972/S	9988	9988/S	10317/M	10389	10460/M
		10489	11902/M	11914	11914/M	11914	11914/S	11921	
pfti_index	8314	7694	7695	7696/M	7696	7697/S	7699	7962/M	7965
		7966	7967/M	7967	7968/S	7971	7985/M	7994/M	8036/M
		9972/M	9972	9972/M	9972	9972/S	9972	9972	9976
		9976/M	9976	9976/S	9976	9976	9988/M	9988	9988/M
		9988	9988/S	9988	9988	10063	10063/M	10063	10063/S
		10063	10063	10317/M	10460/M	11902/M	11914/M	12317	12317/M
		12317	12317/S	12317	12317				
pfti_of_faulted_page	11295	11599/M	11698						
pfti_size	7678	7681/M							
pfti_size	10271	10389/M							
pfti_size	10428	10489/M							
pfti_size	11252	11921/M							
pftis	8316	7697	7963	7968	7993	7994	7995	8027/M	8038/M
		9972	9972	9976	9988	9988	10063	10358/M	10361/M
		10479/M	11902	11902	11902	11914/M	12317		
pit	13351	13356/M	13368						
pmc$initialize_to_zero	5688	9653/P	13248/P						
pmc$kill_task_flag	6264	6280							
pmc$mainframe_id_size	3506	3503							
pmc$max_signal_contents	2308	2302							
pmc$max_task_id	6082	6079							

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pmc$processor_model_number_size	3514	3506	3511						
pmc$processor_serial_num_size	3564	3507	3561						
pmt$binary_mainframe_id	3493	2606	5136						
pmt$condition_identifier	1268	1262							
pmt$cpu_model_number	3218	3207	3214						
pmt$cpu_serial_number	3221	3208	3213						
pmt$initialization_value	5688	5623	7051	7920	8323				
pmt$mainframe_id	3503	2607	3791						
pmt$program_name	4065	3989							
pmt$sense_switches	5231	5147							
pmt$signal	2264	6249							
pmt$signal_contents	2302	2266							
pmt$signal_id	2269	2241	2265						
pmt$task_id	6079	5333	6074						
pqle	4226	9095	9097	9098	9098	9101	9103	9145	9188
		9231	9232	9233/M	9235/S	9237/M	9238/M	9238	9276
		9276	9276	9276	9276	9276	9277	9277	9278
		9278	9298	9304	9323/M	9323	9331	9332/M	9333/M
		9337/M	9470	9471/M	9473/S	9476	9478/M	9489/M	9489
		10590	10592	10594	10696	10696	10696	10696	10696
		10696	12207	12207	13359/P				
pqle	10429	10461	10462	10463					
pqle_p	10278	10306/M	10319	10369/M	10376/M				
pr_pf	11252	12003							
preset	13225	13247							
preset_pages	4324	12230	12236	12281	12283				
preset_streaming	5551	11459	11525	11526/M					
preset_value	7051	10941/P	10954/P	11023/P	12237/P	12282/P	12835/P		
prestream_only	4246	11471/M	11472						
prev_pfti	13327	13330/M	13335	13341/M					
process_memory_image_pf	11190	10850	11232						
process_virtual_address	4305	12708/P	12713/P	12714	12715				
ps_allocate_required_on_server	7915	10955	11647	11849	13104	13110			
ps_beyond_file_limit	7914	10853	11202	11769	13096				
ps_done	7912	7877	10687	10940	10953	11399	11518	11663	11681
		11882	13076	13246					
ps_found_in_avail	7912	7893	11220	11399	11630	11681	11732	11960	11973
		13076	13120						
ps_found_in_avail_modified	7912	7895	11399	11630	11681	11732	11960	11973	13076
		13120							
ps_found_on_disk	7913	11108	11157	11624	11790	13127			
ps_found_on_server	7915	11108	11162	11657	11831	13113			
ps_io_temp_reject	7913	10970	10983	11167	11819	13086			
ps_job_work_required	7916	10935	11757	13102					
ps_locked	7913	7888	11399	11610	11621	11681	11807	13076	
ps_low_on_memory	7913	10827	11826	13083					
ps_new_page_assigned	7914	10943	11031	11072	11647	11739	11960	11973	13104
ps_no_extend_permission	7914	10837	11758	13099					
ps_no_memory	7913	10683	11826	12295	13083	13255			
ps_pages	4268	11629/M	11629	11635/M	11635	11652/M	11652	11662/M	11662
ps_pt_full	7913	10722	11222	11822	12297	13089	13253		
ps_read_beyond_eoi	7914	10839	11374	11758	11761	13093			
ps_server_terminated	7915	10974	10986	11171	11381	11783	13102		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ps_valid_in_pt	7912	7882	10724	11399	11610	11621	11681	11757	11960
		11973	13076						
ps_volume_unavailable	7915	10947	10972	11169	11378	11879	13102		
psc_nominal_queue	7790	12311/P							
pstatus	7861	7877/M	7882/M	7888/M	7893/M	7895/M			
pstatus	10666	10683/M	10687/M	10722/M	10724/M				
pstatus	10787	10827/M	10837/M	10839/M	10850/P	10853/M	10931/P	10935/M	10939/P
		10940	10943/M	10947/M	10952/P	10953	10955/M	10970/M	10972/M
		10974/M	10983/M	10986/M	11017/P	11031/M	11069/P	11072/M	11106/P
		11108	11108						
pstatus	11136	11148/P	11157/M	11162/M	11167/M	11169/M	11171/M		
pstatus	11194	11202/M	11220/M	11222/M					
pstatus	11252	11399/M	11399/M	11399/M	11399/M	11399/M	11681/M	11681/M	11681/M
		11681/M	11681/M						
pstatus	11296	11374/M	11378/M	11381/M	11399/P	11516/P	11518	11596/P	11600
		11610	11610	11620	11681/P	11684/P	11697/M	11707	11730
		11761	11934/S	11934/S	11939	11959	11972	12001	
pstatus	12048	12266/P	12295	12297					
pstatus	12696	12809/P							
pstatus	13001	13076/M	13076/M	13076/M	13076/M	13076/M			
pstatus	13017	13076/P	13079/P	13082	13110				
pstatus	13231	13245/P	13246	13253	13255				
pstatus_of_faulted_page	11297	11600/M	11697						
pstatus_time	8549	11938/M							
psva	12049	12270/M	12282/P	12286/M	12286				
pt_full	2184	10719/M	11225/M	12606/M					
pt_full_aste_p	2186	10717/M	11223/M	12604/M					
pt_full_sva	2185	10718/M	11224/M	12605/M					
pt_length	9546	9552/M	9577	9597	9609				
pt_p	9547	9551/M	9580	9598	9601	9612			
pte	9542	9580/M	9581	9581	9581	9582	9598/M	9599	9599
		9599	9599	9601/M	9603	9603	9612/M	9613	9613
		9613	9614	9644/M	9645/M	9646/M	9647/M	9648/M	9649/M
		9650/M	9652						
pte_p	7870	7880/M	7881	7884	7899/M				
pte_p	9958	9995/M	9997/M	9998/M	10006	10008/M	10017/M	10018/M	10043
pte_p	10101	10119/M	10120/M	10121/M	10130	10131/M	10161		
pte_p	10200	10202/M	10206/M	10207/M	10214				
pte_p	10517	10551/M							
pte_p	11252	11399/M	11399	11399	11399/M	11681/M	11681	11681	11681/M
pte_p	12050	12166/M	12167	12170	12220/M	12221	12222	12234/M	12238/M
		12240/M	12241/M						
pte_p	12409	12547/M	12548	12555	12572				
pte_p	13001	13076/M	13076	13076	13076/M				
pti	4173	9142/S	9173/S	9174/S	9182/S	9434/S	9974/S	9975/S	9995/S
		10119/S	10202/S	10344/S	10345/S	10468/S	10469/S	10551/S	10942/S
		11025/S	11076/S	11077/S	11111/S	11219/S	12284/S	12289/S	12314/S
		12615/S	12651/S	12652/S	12767/S	12836/S	12944/S	13250/S	
pti	9548	9577/M	9579/M	9579	9580/S	9584	9597/M	9598/S	9599
		9599	9600/M	9600	9601/S	9609/M	9611/M	9611	9612/S
		9616	9623	9627/M	9627	9628	9629/M	9629	9632/S
		9632/S	9634/M	9634	9635	9636/M	9652/S	9654/S	
pti	9766	9847	9848/S	9851/S	9866	9867/S	9870/S		

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pti	13449	13437	13439/S						
ptk$aging_ijl_ordinal	6711	10334							
ptk$aging_job_fixed	6720	10332							
ptk$aging_modified_pages	6723	10398							
ptk$aging_page_number	6708	10329							
ptk$aging_pages_removed	6726	10399							
ptk$aging_segment	6705	10328							
ptk$page_assigned_ijl	6699	9136							
ptk$page_assigned_pfti	6666	9133	9415						
ptk$page_assigned_queue	6669	9134	9416						
ptk$page_fault_gtid	6654	11361							
ptk$page_fault_ijl	6702	11709							
ptk$page_fault_lower_offset	6657	11363							
ptk$page_fault_p_lower_offset	6651	11357							
ptk$page_fault_p_segment	6648	11355							
ptk$page_fault_p_upper_offset	6693	11359							
ptk$page_fault_pfti	6660	11706							
ptk$page_fault_segment	6645	11353							
ptk$page_fault_status	6663	11707							
ptk$page_fault_upper_offset	6696	11365							
ptk$performance_base	1758	6645	6648	6651	6654	6657	6660	6663	6666
		6669	6672	6675	6678	6681	6684	6687	6690
		6693	6696	6699	6702	6705	6708	6711	6714
		6717	6720	6723	6726	6729	6732	6735	6738
		6741	6744	6747	6750	6753	6756	6759	6762
		6765	6768	6771	6774	6777	6780	6783	6786
		6789	6792	6795	6798	6801	6804	6807	6810
		6813	6816	6819	6822	6825	6828	6831	6834
		6837	6840	6843	6846	6849	6852	6855	6858
		6861	6864						
ptk$pfti_for_swapin	6690	9309							
pull_pages_page_streaming	11606	11606	11656	11671	11686				
pva	4322	12070/P	12075/P	12091	12092	12118			
pva	5786	11805/M	11817/M	11847/M	11877/M	11880/M	13385		
pva	5961	9009	9010	10900	10905	11067	11336/M	11336	11354
		11356	11358	13464/P	13465/P	13466/P	13467/P		
pva	11261	11350/M	11351	11353	11362	11364			
pva	11298	11350/M	11351	11353	11362	11364	11368/P	11766	11773
		11787	11805	11817	11847	11877	11880		
pva	13291	13297/M	13300/P	13307/M	13307	13310			
pva_destination	4341	12436/P	12441/P	12498					
pva_p	8993	9001/M	9002	9004	9013				
pva_source	4340	12424/P	12429/P						
qcb	13321	13329	13331						
qcb_p	9127	9145/M	9148/M	9151/M	9156/M	9160/M	9160	9188/M	9191/M
		9193	9194/M	9196/S	9198	9199/M	9200/M	9200	
qcb_p	12697	12849/M	12850/S	12850	12851	12852/M	12854/S	12855	12856/M
qid	13322	13338							
queue_count	9264	9297/M	9323						
queue_id	1867	7898/P	8831/M	8936/M	8953	8954/M	8957	8962/M	10746/P
		10876	10876	11399/P	11681/P	12099	12099	12190	12224
		12229/P	12484	12484	12627	12634/P	12642/P	13044	13044

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		13045	13076/P						
queue_id	4170	7891	7892	9139	9140	9142	9145/S	9148/S	9203/M
		9231/S	9232/S	9233/S	9235/S	9237/S	9238/S	9238/S	9318/M
		9429/M	9454/M	9853	9872	10215	11399	11399	11681
		11681	11912	12172	12628	12641	12757	12758	12763
		12764	12765	12801	12847	12878	12879	12880	12934
		12934	12940	12941	12942	12961	13076	13076	13338
queue_id	8807	8821/M	8823/M	8826/M	8828/M	8831			
queue_id	8872	8936/M	8936/M	8936/M	8936/M	8936			
queue_id	9122	9134	9163	9164	9170	9171	9173	9182	9188/S
		9191/S	9203	9204	9218				
queue_id	9262	9269	9270/S	9271	9286	9287/S	9291/S	9292	9318
		9324/S	9325/S	9327/S	9328/S	9331/S	9335/S		
queue_id	9377	9383	9384/S	9385/S	9395	9462	9464/S	9465/S	9467/S
		9471/S	9473/S	9475/S	9478/S	9479	9481/S	9482/S	9483/S
		9485/S	9486/S	9487/S					
queue_id	10433	10491/P	10495						
queue_id	10583	10589	10590/S	10592/S	10594/S	10594/S			
queue_status	7050	8931							
r1	5495	8009	8834	8936	9024				
r2	5496	9021	9027	12281					
random_faults	4253	11451/M	11452						
random_faults	5548	11449	11450/M	11476/M	11476	11477	11485/M	11691/M	
rb	12008	12011	12013/P	12016/P					
rb	12027	12064/M	12070/P	12070/P	12071	12075/P	12078/P	12081/P	12087/P
		12091	12091	12092	12095/P	12100/P	12118	12133/P	12138/P
		12142/P	12147/P	12148/P	12192	12198/P	12230	12236	12281
		12283	12298/P	12311/P	12321/P	12322	12329/P	12338/P	
rb	12388	12422/M	12424/P	12424/P	12425	12429/P	12432/P	12436/P	12436/P
		12437	12441/P	12444/P	12448	12448	12449/P	12453	12454/P
		12459/P	12463	12464	12465/P	12472/M	12473/M	12475	12477
		12480/P	12485/P	12498	12515/P	12520/P	12524/P	12529/P	12530/P
		12543/P	12552/P	12556	12557/P	12560/M	12560	12567	12569
		12616/P	12654/M	12654	12656				
rb	12677	12707/M	12708/P	12708/P	12709	12713/P	12714	12714	12715
		12724/P	12731	12733	12739/P	12742/P	12744	12789/P	12795/P
		12893/P							
rb	13002	13021/M	13034	13034	13035	13036/P	13036/P	13037	13040/P
		13046/P	13051/P	13054/P	13054/P	13062	13062	13062	13063/P
		13063/P	13064	13067/P	13071	13071	13072	13094/P	13097/P
		13100/P							
rcount	9957	9962/M	10033/M	10033	10042/M	10042	10067	10070	
rcount	10092	10110/M	10132/M	10140/M					
rcount	10288	10320/M	10340	10340	10359/M	10359	10362/M	10362	10389/P
		10391	10392/P	10393	10399				
rcount	10442	10463/M	10465	10465	10480/M	10480	10489/P	10490	
rcount	10520	10554/P							
rcount	11299	11921/P	11922	11923/P	11925				
rcount	12698	12772/P							
read_pages_from_disk_or_server	11127	10930	11105	11186					
reason	8663	8668	8673						
reason	9743	9891	9891						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
reason	10776	10934	10934	10944	10944	10967	10967	11037	11037
		11081	11081	11114	11114				
reclaim	4275	11635/M	11635	11737/M	11737	13125/M	13125		
reduce_queue_below_minimum	10582	10585/M	10591	10615	10618/M				
reject_move_if_source_modified	4344	12556							
reject_offset	12051	12113/M	12120/P	12123/M	12146/P				
reject_offset	12410	12493/M	12501/P	12505/M	12528/P				
relative_transfer_unit	11300	11394/M	11421	11421	11447	11458	11458	11534	11540
		11546	11549	11900					
remaining_pages_to_assign	12053	12260/M	12262/M	12265/P					
remove_modified_page_from_ws	2049	10068/M	10068	10179/M	10180				
remove_unmodified_page_from_ws	2048	10066/M	10067	10170/M	10171				
reqcode	4292	13034	13034	13062	13062				
requested_length	4307	12714							
requested_page_count	5098	12324/M	12364/M						
requested_page_count	12052	12091/M	12094	12111	12119/P	12122/P	12123	12163	12217
		12217	12242/M	12242	12259	12262	12292/M	12292	12324
reserved_page_count	5099	12208	12344	12345	12346/M	12365	12366/M		
reset_p_register	13377	13406							
residence	1880	7278/M	7281/M	7338	7385	7840	8913	8922	8933
		9029	9424	9435	9449	9686/M	9686/M	9795	10005
		10128	10327	11053	12884				
residence	7336	7338/M	7342	7343					
residence	7378	7385/M	7385	7385					
residence	8872	8922/M	8922	8922					
residence	9942	10005/M	10005	10005					
residence	10088	10128/M	10128	10128					
residence	10271	10327/M	10327	10327					
residence	10776	11053/M	11053	11053					
residence	12676	12884/M	12884	12884					
ring	2159	9002	9009	9010	10900	10905	11067		
ring	8992	9002/M	9004	9009	9010/M	9021	9024	9027	
rma	1948	7884	9581	9581	9582	9599	9599	9603	9613
		9613	9614	9650/M	9851	9870	11399	11681	11911
		12170	12222	12548	12583	13076	13439		
rma	13292	13300/P	13301	13304					
rma	13452	13462/P	13463						
rp	5493	9021							
sac_p	11301	11760/M	11762/M	11764/M	11766/M	11771/M	11772/M	11773/M	11785/M
		11786/M	11787/M						
save_pfti	12699	12819/M	12821	12851/M	12855/S				
scan_pft_for_pages	12899	12739	12789	12980					
sdt_offset	5335	7808	8892	9014	9553	9694	13464		
sdt_p	8002	8005	8009						
sdt_p	8795	8834	8834						
sdt_p	8872	8936	8936						
sdtx_offset	5336	7820	8893	9015	9695	10883	13386		
search_loop	12928	12928	12937	12949	12967	12972			
seg	2160	9013	9553/P	11327	11351	11353	11354	13464/P	
seg	7334	7341/M	7344/M	7350					
seg	7378	7385/M	7385/M	7385					
seg	8872	8922/M	8922/M	8922					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
seg	9942	10005/M	10005/M	10005					
seg	10088	10128/M	10128/M	10128					
seg	10271	10327/M	10327/M	10327					
seg	10776	11053/M	11053/M	11053					
seg	11302	11354/M	11355						
seg	12676	12884/M	12884/M	12884					
segment	1263	11766/M	11773/M	11787/M					
segment_access	8987	9021	9024	9027					
segment_lock	7043	10006	10130						
segment_number	8000	8006							
segment_number	8795	8834							
segment_number	8872	8936							
segment_table_length	5924	8895	9018						
segnum	7270	7276/M	7277	7282/S					
segnum	7805	7808							
segnum	7817	7820							
segnum	8797	8814	8834/P						
segnum	8872	8892							
segnum	8872	8893							
segnum	8872	8936	8936/P						
segnum	8885	8890/M	8892/P	8893/P	8895	8936/P			
segnum	8985	9014							
segnum	8985	9015							
segnum	8996	9013/M	9014/P	9015/P	9018				
segnum	9532	9553							
segnum	9665	9686/M	9686	9686/S					
segnum	9665	9694							
segnum	9665	9695							
segnum	13375	13386							
segnum	13381	13385/M	13386/P	13391					
segnum	13428	13464							
sequential_accesses	5546	11420	11422/M	11424/M	11424	11454	11454	11455/M	11455
		11473/M	11484/M	11507	11508/M	11508	11556	11690/M	11745/M
		11856/M							
served_file	9767	9810/M	9811	9854	9873	9892			
server	4277	11662/M	11662	11836/M	11836	13118/M	13118		
set_assign_active	8850	8860	9698	10877	10899	12146	12528		
set_page	13430	13444	13465	13466	13467				
sfc$no_limit	5770	10864							
sfid	1868	7839	8832/M	8926	8933	8936/M	9424	9435	9449
		9795	10005/P	10128/P	10327/P	10849	10964/P	12884/P	
sfid	5532	8832	8913	8922/P	8926	8933/P	8936	9029	9696
sfid	7266	7285/M							
sfid	7326	7338	7339	7340	7351				
sfid	7378	7385/P							
sfid	7378	7385	7385	7385	7385				
sfid	7833	7839	7840						
sfid	8276	10849							
sfid	8872	8922/P							
sfid	8872	8922	8922	8922	8922				
sfid	8872	8933	8933						
sfid	9665	9686/M							
sfid	9674	9686/P	9696	9703	9715				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
sfid	9942	10005/P							
sfid	9942	10005	10005	10005	10005				
sfid	10088	10128/P							
sfid	10088	10128	10128	10128	10128				
sfid	10271	10327	10327	10327	10327				
sfid	10776	11053/P							
sfid	10776	11053	11053	11053	11053				
sfid	12676	12884/P							
sfid	12676	12884	12884	12884	12884				
sft$counter	4869	4838	4839	5148	5150	5151	5153		
sft$file_space_limit_kind	5770	5539	7223	7247	7257	10796			
shadow_au_offset	10800	11051/M	11083	11106/P					
shadow_info	5538	10994	10995	11051	11053/P	11056	11067	11083	
shadow_length_page_count	5728	10995	11056						
shadow_reference_info	5353	11082/M	11084/M	11086/M					
shadow_segment_kind	5730	10994	11067						
shadow_segment_number	5732	11083							
shadow_sfid	5729	11053/P							
shadow_start_page_number	5727	11051							
smallest_maximum_working_set	10521	10530/M	10532/M	10542	10543/M	10548	10548		
software_attribute_set	5535	8820	8822	8936	8936	11505	11897	13391	
soon	4358	9143/M	9143	9167/M	9167	9186/M	9186	9239/M	9239
		9442/M	9442	9494/M	9494	10054	10174	10247	11994
		13186							
source_aste_p	12411	12429/P	12431	12609	12611/P				
source_fde_p	12412	12429/P							
source_pti	12413	12541	12547/S						
source_pva	5775	11082/M							
source_queue_id	9261	9283/M	9298/S	9300	9303/M	9312	9323/S	9323/S	9331/S
		9332/S	9333/S	9337/S					
source_ste_p	12414	12429/P							
source_stxe_p	12415	12430/P							
source_sva	12416	12429/P	12458	12463	12541	12610	12611/P	12657/M	12657
specified	2332	9963/M	10056/M	10176/M	11316/M	12873/M	13027/M		
st	8336	13241							
stack_for_ring	7055	8007	8813	8834	8936	8936	10026	10027/P	10146
		10147/P	10907	11022	11022	11774	12281		
starting_pfti	10663	10678	10680	10695					
starting_pfti	12700	12738/M	12739/P	12741	12751	12751	12752	12788/M	12789/P
		12794	12799	12808/P	12834	12834	12835/S	12836/S	12837/M
		12837/S							
starting_pfti	12902	12910	12917	12975/M	12977/M				
static_next_rma	9538	9568	9572	9617/M	9650	9651/M	9651		
static_stop_rma	9539	9568	9573	9618/M					
statistics	4569	10294	10296	10297	10452	10454	10455	11625/M	11626
		11631/M	11632	11648/M	11649	11658/M	11659	11726/M	11727
		11733/M	11734	11740/M	11741	11791/M	11792	11832/M	11833
		11851/M	11852	11945	11947	11948	11984	11986/M	12305
		13105/M	13106	13114/M	13115	13121/M	13122	13128/M	13129
		13148	13150	13151	13177	13179/M			
status	4293	13021/M	13036/P	13037	13046/P	13051/P	13063/P	13064	13094/P
		13097/P	13100/P						
status	4304	12707/M	12708/P	12709	12724/P	12742/P	12795/P	12893/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	4319	12064/M	12070/P	12071	12078/P	12081/P	12087/P	12095/P	12100/P
		12133/P	12138/P	12142/P	12147/P	12148/P	12198/P	12298/P	12321/P
		12329/P	12338/P						
status	4339	12422/M	12424/P	12425	12432/P	12436/P	12437	12444/P	12449/P
		12454/P	12459/P	12465/P	12480/P	12485/P	12515/P	12520/P	12524/P
		12529/P	12530/P	12543/P	12552/P	12557/P	12616/P		
status	7714	7728/P							
status	7747	7752/P							
status	8066	8069/M	8070/M						
status	8387	13392							
status	8985	9005/M	9005/M	9019/M	9019/M	9023/M	9023/M	9026/M	9026/M
		9028/M	9028/M	9031/M	9031/M				
status	8988	8998/M	9005/P	9019/P	9023/P	9026/P	9028/P	9031/P	
status	9057	9066/P	9067						
status	9675	9700/P	9704/P	9716/P					
status	9768	9893/P	9895/P	9898	9901	9903	9905	9905	9907
		9909							
status	10805	10878/P	10903/P	10965/P	10966	10969	10969	10971	10973
		11087/P							
status	11143	11156/P	11161/P	11164	11166	11166	11168	11170	
status	12026	12078/M	12078/M	12081/M	12081/M	12087/M	12087/M	12095/M	12095/M
		12100/M	12100/M	12133/M	12133/M	12138/M	12138/M	12142/M	12142/M
		12148/M	12148/M	12198/M	12198/M	12298/M	12298/M	12321/M	12321/M
		12329/M	12329/M	12338/M	12338/M				
status	12387	12432/M	12432/M	12444/M	12444/M	12449/M	12449/M	12454/M	12454/M
		12459/M	12459/M	12465/M	12465/M	12480/M	12480/M	12485/M	12485/M
		12515/M	12515/M	12520/M	12520/M	12524/M	12524/M	12530/M	12530/M
		12543/M	12543/M	12552/M	12552/M	12557/M	12557/M	12616/M	12616/M
status	12676	12724/M	12724/M	12742/M	12742/M	12795/M	12795/M	12893/M	12893/M
status	12676	12770/P	12882/P						
status	12898	12946/P							
status	13001	13046/M	13046/M	13051/M	13051/M	13094/M	13094/M	13097/M	13097/M
		13100/M	13100/M						
status	13222	13254/M	13254/M	13256/M	13256/M				
status	13226	13237/M	13252/P	13254/P	13256/P				
status	13288	13296/M							
status	13383	13399/P	13403/P						
ste	5464	8005	8009	8772	8772	8826	8826	8834	8834
		8836/M	8895	8919	8936	8936	8936	8936	8936/M
		8939/M	8962	8962	9018	9021	9021	9024	9025
		9027	9554	9696	11510	11510	12281	13241	13434
		13435							
ste_p	8769	8772	8772						
ste_p	8795	8826	8826						
ste_p	8798	8826/P	8834/P	8836/M	8837/M				
ste_p	8872	8936/P	8936/P	8936/M	8936/M				
ste_p	8872	8936	8936	8962	8962				
ste_p	8878	8892/M	8895	8919	8926	8936/P	8939/M	8940/M	8962/P
ste_p	8994	9014/M	9018	9021	9021	9024	9025	9027	
ste_p	9062	9069/P							
ste_p	9537	9553/M	9554						
ste_p	9676	9694/M	9696						
ste_p	11303	11368/P	11510	11510					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ste_p	12054	12075/P	12281						
ste_p	12701	12713/P							
ste_p	13008	13040/P	13067/P						
ste_p	13431	13434	13435						
ste_p	13453	13464/M	13465/P	13466/P	13467/P				
stlc_allocation	5314	10901/M							
stop_rma	9541	9573/M	9576/M	9581	9608/M	9613	9614/M	9618	
stream	5540	11388	11391	11392/M	11420	11422/M	11424/M	11424	11449
		11450/M	11454	11454	11455/M	11455	11459	11467	11473/M
		11476/M	11476	11477	11484/M	11485/M	11486/M	11507	11508/M
		11508	11525	11526/M	11527	11531/M	11545	11556	11558/M
		11611	11690/M	11691/M	11692/M	11745/M	11856/M		
streaming	5549	11467	11486/M	11527	11531/M	11545	11558/M	11611	11692/M
streaming_transfer_pages	11304	11387/M	11389	11506	11574	11575	11576	11678	
streaming_transfer_unit	11305	11389/M	11390	11393	11555	11572	11752	11753/M	11754
		11863	11864/M	11865	11901				
strn	6990	7004							
strng	6986	6989							
stxe_p	8799	8820	8822	8832					
stxe_p	8851	8854	8855/M	8857/M					
stxe_p	8872	8936	8936	8936					
stxe_p	8879	8893/M	8913	8922/P	8926	8933/P	8936/P		
stxe_p	8995	9015/M	9029						
stxe_p	9063	9069/P							
stxe_p	9665	9698	9698/M	9698/M					
stxe_p	9677	9695/M	9696	9697	9698/P				
stxe_p	10776	10877	10877/M	10877/M	10899	10899/M	10899/M		
stxe_p	10781	10835	10836	10867	10877/P	10883	10899/P	10965/P	10994
		10994	10995	11051	11053/P	11056	11067	11083	
stxe_p	11306	11368/P	11375	11376	11379	11388	11391	11392/M	11420
		11422/M	11424/M	11424	11449	11450/M	11454	11454	11455/M
		11455	11459	11467	11473/M	11476/M	11476	11477	11484/M
		11485/M	11486/M	11505	11507	11508/M	11508	11515/P	11525
		11526/M	11527	11531/M	11545	11556	11558/M	11595/P	11611
		11683/P	11690/M	11691/M	11692/M	11745/M	11856/M	11897	
stxe_p	12026	12146	12146/M	12146/M					
stxe_p	12055	12075/P	12076	12077	12080	12120/P	12146/P		
stxe_p	12387	12528	12528/M	12528/M					
stxe_p	12702	12713/P							
stxe_p	13009	13040/P	13041	13067/P	13068	13078/P			
stxe_p	13380	13386/M	13391						
sub_reqcode	4320	12011							
sva	2196	9885/M	10962/M	11152/M					
sva	4178	8598/P	9205/M	9314/M	9401	9412	9422/P	9428/M	9698/P
		9834	10016	10028	10122/P	10148	10208/P	10329	10738/M
		11211/M	12591/M	12610/M	12835/P				
sva	7858	7873							
sva	8041	8046							
sva	8550	11937/M							
sva	8624	8629							
sva	8636	8641							
sva	8886	8919/M	8920/M	8924	8966/M	8974			
sva	9056	9069/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	9536	9554/M	9555/M	9556/M	9556	9623	9648	9649	9653/P
sva	9770	9834/M	9835	9839	9840	9860	9861/M	9878	9885
		9893/P	9895/P						
sva	10088	10122							
sva	10196	10208							
sva	10660	10688							
sva	10795	10810/M	10811/M	10811	10834	10848	10850/P	10877/P	10886/P
		10888/P	10899/P	10918	10919	10920	10930/P	10930/P	10934/P
		10938	10939/P	10941/P	10944/P	10952/P	10954/P	10962	10964/P
		10965/P	10967/P	10995	11004	11004	11012	11017/P	11023/P
		11028/M	11028	11037/P	11050	11054/M	11069/P	11081/P	11085
		11105/P	11114/P						
sva	11130	11148/P	11152						
sva	11191	11200	11207	11211	11216/P	11224			
sva	11252	11399	11681						
sva	11307	11368/P	11390	11392	11399/P	11515/P	11540	11573	11574
		11595/P	11754	11865	11899	11937			
sva	12026	12319							
sva	12056	12075/P	12094	12106/M	12106	12107	12113	12119/P	12122/P
		12123	12216	12311/P	12319/P				
sva	12387	12666							
sva	12703	12713/P	12716	12717	12808/P				
sva	13001	13076							
sva	13018	13040/P	13054/P	13054/P	13067/P	13076/P	13078/P	13141/M	13141
sva	13232	13238/M	13241/M	13243/P	13245/P	13248/P	13262/M	13262	
sva	13448	13435/M	13436/M	13437					
sva_current	11308	11573/M	11680/M	11681/P	11683/P				
sva_of_last_page	12418	12477/M							
sva_start	11309	11572/M	11574	11680					
swap_status	4538	7724	9838	12770	12882	12946	12960		
swapout_job	5097	12325/M	12363/M						
swapped_job_entry	9252	9270	9287						
syc$ic_disk_error	6370	13397/P							
syc$mf_for_keypoint_traceback	6105	10337/P							
syc$rc_advise_in	3364	13062							
syc$rc_advise_out	3365	13034							
syc$rc_advise_out_in	3366	13034	13062						
syc$ucr_condition	6166	6177							
syc$user_defined_condition	6167	6179							
system_ijle_p	12419	12470/P	12637/M	12637	12646/M	12646			
system_table_lock_count	5319	10825	10900	10907	13395				
syt$180_idle_code	6352	4493	8087						
syt$monitor_flag	6101	6086	8130	8155					
syt$monitor_flags	6086	5307	8391						
syt$monitor_request_code	3343	3306	4292	4303	4318	4338	6495		
syt$monitor_status	3331	3307	4293	4304	4319	4339	6496	7224	7237
		7482	7494	7747	7773	8066	8156	8988	9057
		9675	9768	10441	10805	11143	12417	13226	13288
		13383							
syt$perf_keypoints_enabled	6868	8583	8583						
syv$perf_keypoints_enabled	8583	9132	9308	9414	10322	10336	10397	11352	11705
syv$recovering_job_count	8585	10524							
syv$refs_to_unrecovered_seg	8586	8914/M	8914	9030/M	9030				

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
task_created_after_last_swap	4578	10536							
task_queue	4174	10732	11804/P	11816/P	11846/P	11876/P			
task_slow	4250	11550/M	11551						
taskid	4481	8815	8936	10878/P	10903/P	11087/P	11767/P	11781/P	11788/P
		11804/P	11816/P	11827/P	11846/P	11876/P	12147/P	12529/P	
taskid	9125	9219/P							
taskid	9378	9512/P							
taskid	13417	13420/P							
tasks_not_in_long_wait	4841	12305							
terminated	4247	11482/M	11483						
test_sva	12704	12716/M	12717/M	12722	12727/M	12727			
time_last_modified	7052	9899/M							
time_spent_in_job_mode	4824	10294	10297	10452	10455	11945	11948	13148	13151
time_spent_in_mtr_mode	4825	10296	10454	11947	13150				
tmc$broken_task_fault_id	6122	6144							
tmc$btc_invalid_a0	6192	6213							
tmc$btc_invalid_p	6192	6213							
tmc$btc_mcr_traps_disabled	6193	6214							
tmc$btc_mf_traps_disabled	6192	6212							
tmc$btc_mntr_fault_buffer_full	6191	6212							
tmc$btc_system_error	6194	6208							
tmc$btc_ucr_traps_disabled	6193	6214							
tmc$dummy_fault	6123	6150							
tmc$flag_available_31	6277	6281							
tmc$maximum_monitor_faults	6127	6118							
tmc$maximum_signals	6257	6254							
tmc$maximum_system_task_id	6290	6293							
tmc$mcr_fault	6122	6146							
tmc$signal_available_63	2300	2311							
tmc$stid_null_task	6296	6293							
tmc$ts_first_external_queue	3591	13392							
tmc$ts_io_wait_not_queued	3608	3590							
tmc$ts_memory_wait	3610	11827/P							
tmc$ts_page_wait	3610	3591	11804/P	11816/P	11846/P	11876/P			
tmc$ts_ready	3595	3586							
tmc$ts_timed_wait_not_queued	3602	3589							
tmc$ts_timeout_reqexp_longvlong	3600	3588							
tmc$ts_timeout_reqexp_shortshrt	3599	3587							
tmc$ts_volume_unavailable	3611	13393/P							
tmp$cause_task_switch	8098	11820	11824	11966	11974	12134	12516	12825	13084
		13087	13091	13166	13169				
tmp$clear_lock	7581	7563	7594	7729	9706	9719	12770	12774	12882
		12886	12946						
tmp$dequeue_task	8101	9219	9512	13420					
tmp$get_top_of_stack	8116	10027	10147						
tmp$get_xcb_p	8122	9712							
tmp$monitor_flag_job_tasks	8129	10337							
tmp$queue_task	8137	11804	11816	11827	11846	11876	13393		
tmp$reissue_monitor_request	8144	12826	13407						
tmp$send_monitor_fault	8147	11767	11781	11788					
tmp$set_lock	7617	7556	7635	7727	9706	9719	12770	12774	12882
		12886	12946						

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmp$set_monitor_flag	8154	9700	9704	9716	10878	10903	11087	12147	12529
		13403							
tmp$test_get_xcb_p	8160	9684							
tmt$broken_task_condition	6191	6207							
tmt$broken_task_monitor_fault	6205	6145							
tmt$dispatch_control	6382	4488							
tmt$dual_state_priority_entry	6397	4475	6394						
tmt$find_next_xcb_state	6882	8110							
tmt$fnx_search_type	6880	6883	8107						
tmt$idle_status	3613	8389							
tmt$mcr_faults	6230	6147							
tmt$monitor_fault_buffer	6112	5344							
tmt$monitor_fault_buffers	6118	6113	6114	6115					
tmt$monitor_fault_identifiers	6121	6143	6219						
tmt$primary_task_list	8400	8366							
tmt$primary_task_list_entry	8382	8400							
tmt$ptl_flags	8406	8393							
tmt$ptl_lock	7605	7581	7617	7641					
tmt$signal	6247	6242							
tmt$signal_buffer	6239	5345							
tmt$signal_buffers	6254	6240	6241	6242					
tmt$system_flags	6260	5320	8392						
tmt$system_task_id	6293	5311							
tmt$task_queue_link	4205	4174	7162	8101	8139	8390	8531	8531	8580
		8580							
tmt$task_status	3595	6385	8138	8387	8388				
tmt$wait_inhibited	8413	8408							
tmt$xcb_offset_size	8404	8385							
tmv$ptl_lock	7641	7556/P	7563/P	7727/P	7729/P	9706/P	9706/P	9719/P	9719/P
		12770/P	12770/P	12774/P	12774/P	12882/P	12882/P	12886/P	12886/P
		12946/P	12946/P						
tmv$ptl_p	8366	13392	13396						
tos	9950	10027/P	10028	10031					
tos	10096	10147/P	10148	10151					
total_contig_pages_assigned	8502	12864/M	12864						
total_pages_freed	9379	9381/M	9496/M	9496	9510	9510	9511/M	9511	
total_pages_needed	10574	10609							
total_pages_removed	10436	10491/P	10493/M						
transfer_size	5547	11388							
transfer_unit_count	11310	11524/M	11532/M	11547/M	11557/M	11576			
trap_enable	5893	10905	11066	13387					
trick_ptr	11258	11298							
trim_pages	12057	12191/M	12193/M						
trim_to_swap_size	10512	10535							
tu_end	9772	9836/M	9864	9864					
tu_start	9771	9835/M	9836	9844	9844				
tu_to_stream	11311	11554/M	11556						
two_passes	10587	10587	10610	10616	10619				
u	1945	9646/M	9975/M	9998/M	10018/M	10121/M	10207/M	10344	10345/M
		10468	10469/M	12240/M					
untranslatable_pointer	5929	9553/P	9555	11327	11328/P	11350	11999/M		
update_eoi_reason	10806	11033/M	11035/M	11037/P					

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

IDENTIFIER	DEFINED ON LINE	REFERENCES							
v	1943	7881	7899/M	9174	9434/M	9581	9599	9599	9603
		9613	9644/M	9654/M	9974/M	9997/M	10008/M	10017/M	10120/M
		10131/M	10206/M	10942/M	11025/M	11076/M	11219/M	11399	11399/M
		11681	11681/M	12167	12221	12234/M	12241/M	12289/M	12314/M
		12615/M	12651/M	12836/M	13076	13076/M	13250/M		
valid_pages_in_memory	12059	12158/M	12168/M	12168					
verify_pages_removable	12755	12755	12761	12775	12782	12783			
v1	5491	8895	9018	9696					
waitopt	4325	12322							
wire_eoi_page	7069	10015	10938						
working_set_max_used	4865	11980	11982/M	11984	11986/M	13173	13175/M	13177	13179/M
wp	5494	8005	8772	8826	8834	8936	8936	8962	9025
		11510							
write_multiple_pages	9773	9838/M	9854	9873					
write_status	9749	9785/M	9786/S	9786/S	9796/M	9797/S	9797/S	9813/M	9814/S
		9814/S	9900/M	9902/M	9904/M	9906/M	9908/M	9911/M	9918/S
		9918/S							
write_status	9954	10057/P							
write_status	10099	10177/P							
write_status	12705	12885/P							
ws_device_manager_reject	9740	9902	9911						
ws_no_file_assigned	9739	9785							
ws_ok	9739	9900							
ws_physical_io_reject	9739	9796	9813	9904					
ws_server_terminated	9740	9908							
ws_volume_unavailable	9740	9906							
x_asid	8709	8720/M							
x_asid	8872	8938/M							
xasid	8594	8598							
xasid	8726	8735							
xasid	8748	8754							
xasid	9357	9422							
xasid	13222	13243							
xasti	8708	8716							
xasti	8872	8938							
xcb_p	4486	8892/P	8893/P	8895	9009	9010	9014/P	9015/P	9018
		10825	10826	10883	10900	10900	10901/M	10905	10905
		10906	10907	11066	11067	11082/M	11084/M	11086/M	11321/P
		11327	11328/P	11336/M	11336	11350	11354	11356	11358
		11360	11627/M	11628	11633/M	11634	11650/M	11651	11660/M
		11661	11725/M	11725	11735/M	11736	11742/M	11743	11793/M
		11793	11805/M	11817/M	11834/M	11835	11847/M	11853/M	11854
		11877/M	11880/M	11881/P	11963/M	11963	11980	11982/M	11999/M
		13107/M	13108	13116/M	13117	13123/M	13124	13130/M	13130
		13163/M	13163	13173	13175/M	13363	13460	13461	
xcb_p	7804	7807	7808						
xcb_p	7816	7819	7820						
xcb_p	8872	8892	8892						
xcb_p	8872	8893	8893						
xcb_p	8985	9014	9014						

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